The Stata Journal

Volume 17 Number 2 2017

A Stata Press publication
StataCorp LLC
College Station, Texas

The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*), *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-782-8272, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$124	1-year subscription	$154
2-year subscription	$224	2-year subscription	$284
3-year subscription	$310	3-year subscription	$400
1-year student subscription	$ 89	1-year student subscription	$119
1-year institutional subscription	$375	1-year institutional subscription	$405
2-year institutional subscription	$679	2-year institutional subscription	$739
3-year institutional subscription	$935	3-year institutional subscription	$1,025
Electronic only		**Electronic only**	
1-year subscription	$ 89	1-year subscription	$ 89
2-year subscription	$162	2-year subscription	$162
3-year subscription	$229	3-year subscription	$229
1-year student subscription	$ 62	1-year student subscription	$ 62

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

Volume 17 Number 2 2017

The Stata Journal

The Stata Journal (2017)
17, Number 2, pp. 251–252

Joseph M. Hilbe (1944–2017)

Dr. Joseph M. Hilbe, a long-standing member of the Stata community, author of books published by Stata Press, and the founding editor of the *Stata Technical Bulletin* (predecessor to the *Stata Journal*), died at his home on March 12, 2017. He was 72 years old.

Hilbe, who was born in Los Angeles, CA, received his bachelors degree in philosophy from California State University, Chico in 1968 and his PhD in applied mathematics (statistics) from UCLA in 1988. He worked at the University of Hawaii, where he ultimately retired in 1990 as an emeritus professor. Not content to remain retired, however, Hilbe then joined the Department of Sociology at Arizona State University in Tempe as an adjunct professor of statistics. In 2006, he combined his love for astronomy and statistics when he was selected as a Solar System Ambassador with NASA's Jet Propulsion Laboratory.

Hilbe was a prolific author of textbooks on statistical modeling. Among his influential books include two editions of *Generalized Estimating Equations*, two editions of *Negative Binomial Regression*, and three editions of *Generalized Linear Models and Extensions*. His additional books include *Modeling Count Data*, *Logistic Regression Models*, and its companion book, *Practical Guide for Logistic Regression*. Finally, he coauthored *Quasi-Least Squares Regression* and *Methods of Statistical Model Estimation*. A fourth edition of *Generalized Linear Models and Extensions* was completed and is forthcoming this year from Stata Press.

gn0073

Joe Hilbe never seemed to stop. After writing his first textbook, he embraced the process. Suddenly, he was publishing a book a year and consistently had several projects in development. It was the same with his involvement with astrostatistics. He recognized the connection of the modeling needs of this new area of research with the modeling expertise he developed in his texts. First, his books were in demand around the world, and then Hilbe himself was in demand around the world when he became an ambassador for astrostatistics.

Among other career and lifetime achievements, Hilbe was elected as a member of the International Statistical Institute (ISI), a fellow of the American Statistical Association, a fellow of the Royal Statistical Society, and a full member of the American Astronomical Society. He founded the Astrostatistics Interest Group within the ISI and then later served as the founding president of the International Astrostatistics Association. In yet another service to the community, Hilbe served as editor in chief of the Springer Series in Astrostatistics, which began in 2011.

Joe Hilbe is survived by his wife, Cheryl, of Chandler, Arizona; his daughter, Heather; and his sons, Michael and Mitchell.

James Hardin

The Stata Journal (2017)
17, Number 2, pp. 253–278

Estimating inverse-probability weights for longitudinal data with dropout or truncation: The xtrccipw command

Eric J. Daza
Stanford Prevention Research Center
Stanford University
Stanford, CA
ericjdaza@stanford.edu

Michael G. Hudgens
Department of Biostatistics
University of North Carolina at Chapel Hill
Chapel Hill, NC

Amy H. Herring
Department of Biostatistics and Carolina Population Center
University of North Carolina at Chapel Hill
Chapel Hill, NC

Abstract. Individuals may drop out of a longitudinal study, rendering their outcomes unobserved but still well defined. However, they may also undergo truncation (for example, death), beyond which their outcomes are no longer meaningful. Kurland and Heagerty (2005, *Biostatistics* 6: 241–258) developed a method to conduct regression conditioning on nontruncation, that is, regression conditioning on continuation (RCC), for longitudinal outcomes that are monotonically missing at random (for example, because of dropout). This method first estimates the probability of dropout among continuing individuals to construct inverse-probability weights (IPWs), then fits generalized estimating equations (GEE) with these IPWs. In this article, we present the `xtrccipw` command, which can both estimate the IPWs required by RCC and then use these IPWs in a GEE estimator by calling the `glm` command from within `xtrccipw`. In the absence of truncation, the `xtrccipw` command can also be used to run a weighted GEE analysis. We demonstrate the `xtrccipw` command by analyzing an example dataset and the original Kurland and Heagerty (2005) data. We also use `xtrccipw` to illustrate some empirical properties of RCC through a simulation study.

Keywords: st0474, xtrccipw, dropout, generalized estimating equations, inverse-probability weights, longitudinal data, missing at random, truncation, weighted GEE

1 Introduction

Consider an individual's outcomes over time, which form an outcome trajectory. Events such as death can truncate the trajectory, rendering the outcome at and after truncation undefined. Death is a common truncating event in biomedical studies (Ribaudo, Thompson, and Allen-Mersh 2000; Billingham and Abrams 2002; Pauler, McCoy, and Moinpour 2003; Dufouil, Brayne, and Clayton 2004; Shardell and Miller 2008;

Basu and Manning 2010). For example, the Precipitating Events Project (PEP) is an ongoing longitudinal study of 754 community-living individuals aged 70 or older who are scheduled to be followed monthly for 2 years (Gill et al. 2001; Gill 2014). Kurland and Heagerty (2005) considered inference about the probability of activities-of-daily-living (ADL) disability conditioning on being alive, treating death as a truncating event in the PEP data. Other events, such as disease relapse and HIV infection, have also been defined as truncating events. For instance, investigators of the Breastfeeding, Antiretrovirals, and Nutrition study (van der Horst et al. 2009) wanted to draw inference about a target population of infants at high risk of HIV infection but only while they were alive and uninfected (Flax et al. 2012). In this case, HIV infection and death are truncating events. In le Cessie et al. (2009), the target population consisted of patients with advanced breast cancer who had undergone chemotherapy. The authors wanted to draw inference about patients who were alive and disease free, such that death and relapse were truncating events.

For all the aforementioned examples of truncated longitudinal data, outcomes were also missing for some individuals. Dropout events occur when an individual leaves the study permanently. For study dropout, the corresponding outcomes are unobserved, but unlike truncation, they are well defined. Three comprehensive types of such missingness were characterized by Rubin (1976) and Little and Rubin (2002). In their framework, outcomes are defined to be missing completely at random (MCAR) if missingness is independent of any outcomes. If the pattern of missingness is independent of all missing outcomes conditional only on observed outcomes, then the outcomes are missing at random (MAR). Finally, if missingness is not MAR or MCAR, the outcomes are said to be not missing at random, or missing not at random (MNAR). The method of generalized estimating equations (GEE), which is frequently used to estimate the marginal means of a longitudinal outcome, can accommodate missingness. If outcomes are MCAR, then the GEE estimator is consistent for these marginal means (Liang and Zeger 1986; Diggle et al. 2002). If outcomes are either MAR or MNAR, inverse-probability weights (IPWs) may be used to ensure consistency of the GEE estimator provided that the data missingness model is correctly specified (Robins, Rotnitzky, and Zhao 1995; Scharfstein, Rotnitzky, and Robins 1999). We refer to this approach as the weighted GEE (WEE) method.

Typical approaches to analyzing longitudinal outcomes with missing data include both WEE and maximum likelihood methods such as mixed-effects models. These approaches generally do not distinguish truncation from dropout, in essence envisaging outcomes past the point of truncation. Kurland and Heagerty (2005) described such approaches that implicitly assume the existence of outcomes after truncation as unconditional regression (UR) models, because they estimate the mean outcome averaged over individuals who have and have not been truncated. Kurland et al. (2009) consider both standard selection models and conditional submodels of pattern-mixture models to be UR models. Mean outcomes among continuing trajectories may be estimated indirectly with these two types of UR models, with additional modeling assumptions (Kurland et al. 2009). As an alternative to UR models, one can use joint modeling of longitudinal measurements and time to truncation (Henderson, Diggle, and Dobson 2000; Guo and Carlin 2004; Kurland et al. 2009).

To estimate mean outcomes directly without joint modeling, Kurland and Heagerty (2005) developed a method for regression conditioning on continuation (RCC), that is, not being truncated. The RCC method consistently estimates continuing longitudinal mean outcomes by first modeling and estimating IPWs at each time point based on the probability of dropout, but only for subjects with a continuing outcome at that time point. RCC then applies these IPWs in a WEE framework. In the absence of truncation, the usual WEE method is therefore a special case of RCC. When there is truncation, WEE is a UR approach that will generally not produce consistent estimates for RCC estimands (Kurland and Heagerty 2005). Unfortunately, there is currently no widely available Stata command for estimating the IPWs used in either RCC or WEE. The `teffects` commands `aipw` (see [TE] **teffects aipw**), `ipw` (see [TE] **teffects ipw**), and `ipwra` (see [TE] **teffects ipwra**) estimate IPWs with the goal of making causal inferences by estimating average treatment effects. The `stteffects ipwra` command (see [TE] **stteffects ipwra**) estimates IPWs that adjust for outcomes that are missing because of censoring and uses these IPWs in survival analysis of time-to-event outcomes. In this article, we introduce the `xtrccipw` command to allow Stata users to estimate the IPWs used by RCC in analyzing longitudinal outcomes subject to dropout or truncation. These IPWs can then be used as `pweight` values in the `glm` command with the `vce(cluster` *clustvar*`)` option to perform WEE estimation, which can be executed within a call to `xtrccipw` if requested. When there is no truncation, `xtrccipw` can also be used to estimate the IPWs used in a WEE analysis. When there is truncation but no dropout, the `xtrccipw` command produces IPWs that all equal 1, resulting in unweighted GEE regression.

The remainder of this article is organized as follows: In section 2, we introduce some notation and the assumptions behind the RCC method, detail the modeling of the dropout mechanism, and note some asymptotic properties of the RCC estimator. In section 3, we explain the `xtrccipw` command. In section 4, we conduct RCC on a binary outcome using an example dataset. In section 5, we perform a simulation study based on the original Kurland and Heagerty (2005) simulations and reanalyze their empirical data. In section 6, we conclude the article.

2 Background and methods

2.1 Notation and assumptions

Consider a random sample of $i = 1, \ldots, n$ individuals, each of whom is scheduled to be measured at fixed study time points $j = 1, \ldots, m$. Where it is not ambiguous, the dependence on i will be suppressed for notational ease. To illustrate the relevant concepts, we use an example wherein the outcome is individual alanine transaminase (ALT) measured in international units/liter (IU/L) measured at up to $m = 3$ study visits, and individuals may die or drop out of the study. The example data are listed in table 1, where `idvar` is the variable that denotes individual identifier, `timevar` denotes study visit date, `timeidxvar` denotes study visit number, `outcomevar` denotes ALT, `tdindepvar` denotes a time-dependent continuous-valued covariate, and `tiindepvar` denotes a time-independent binary-valued covariate (for example, a baseline variable).

Table 1. Example dataset

`idvar`	`timevar`	`timeidxvar`	`outcomevar`	`tdindepvar`	`tiindepvar`	`trtimevar`	C_j	R_j	S
1	05apr1979	1	13	432	yes	.	1	1	3
1	04may1979	2	14	65	yes	.	1	1	3
1	05jun1979	3	08	−5	yes	.	1	1	3
2	18sep1982	1	24	83	no	.	1	1	3
2	20oct1982	2	32	23	no	.	1	1	3
2	21nov1982	3	.	.	no	.	1	0	3
3	15sep1983	1	25	441	no	16nov1983	1	1	2
3	19oct1983	2	23	76	no	16nov1983	1	1	2
3	16nov1983	3	*	*	*	16nov1983	0	*	2
4	14jan1979	1	15	−23	no	24feb1979	1	1	2
4	14feb1979	2	.	.	no	24feb1979	1	0	2
4	16mar1979	3	*	*	*	24feb1979	0	*	2

We first introduce notation for the outcomes and truncation. Let Y_j denote the primary outcome of interest, for example, ALT, at time point j. Let $C_j = 1$ if the truncating event, for example, death, has not occurred by j, and let $C_j = 0$ otherwise. Thus the outcome Y_j is well defined only if $C_j = 1$. In general, we define truncation as an irreversible state transition such that $C_j = 0$ implies $C_{j'} = 0$ for all $j' > j$. Define $S = \sum_{j=1}^{m} C_j$ to be the number of time points before a trajectory is truncated, with $S = m$ indicating that the trajectory is not truncated. If truncation occurs at j, then outcomes at j and beyond (that is, $Y_j, \ldots, Y_m$) are undefined. We use "$*$" to denote all undefined values, which extends the support of the outcome Y. In table 1, where `trtimevar` denotes truncation time, individual 4 died between study visits 2 and 3.

The indicator variable for dropout is defined as follows. If truncation has not occurred by time point j, but if that individual dropped out of the study at or before j, then his or her outcome is still defined at j but is not observed. If $C_j = 1$, let $R_j = 1$ if an individual has not dropped out by j; otherwise, let $R_j = 0$. Assume that there is no dropout at $j = 1$ (that is, $R_1 = 1$) and that dropout is monotonic such that $R_j = 0$ implies $R_{j'} = 0$ for all $j' > j$. If $C_j = 0$, then we adopt the convention that $R_j = *$. In table 1, individual 2 never died during the study but dropped out by visit 3; missing values are denoted using ".". Individual 4, however, dropped out of the study between visits 1 and 2 and died between the scheduled times for visits 2 and 3.

The assumptions about the dropout mechanism are now defined. For any time-varying random variable A, let $\overline{A}_j = \left(A_1, \ldots, A_j\right)$ so that $\overline{A}_{j-1}$ represents an individual's history of A prior to j. In table 1, the full truncation vector of individual 1 is $\overline{C}_3 = (1,1,1)$, while his or her ALT history prior to study visit 3 is $\overline{Y}_2 = \left(Y_1, Y_2\right) = (13, 14)$. Let $\overline{Y}_j^{\text{obs}}$ denote the vector of observed values of $\overline{Y}_j$, that is, $(Y_k\colon R_k = 1, k \leq j)$. In table 1, $\overline{Y}_1^{\text{obs}} = (Y_1) = (25)$ and $\overline{Y}_2^{\text{obs}} = \overline{Y}_3^{\text{obs}} = (Y_1, Y_2) = (25, 23)$ for individual 3, while $\overline{Y}_1^{\text{obs}} = \overline{Y}_2^{\text{obs}} = \overline{Y}_3^{\text{obs}} = (Y_1) = (15)$ for individual 4. Let π_j denote the probability of not dropping out conditional on all outcomes and the full truncation vector, that is, $\pi_j = \Pr\left(R_j = 1 \middle| \overline{Y}_m, \overline{C}_m\right)$, and assume $\pi_1 = \Pr\left(R_1 = 1 \middle| C_1\right)$. We refer to outcomes as MAR if $\pi_j = \Pr\left(R_j = 1 \middle| \overline{Y}_{j-1}^{\text{obs}}, \overline{C}_j\right)$ for all $j > 1$. We refer to outcomes as MCAR if $\pi_j = \Pr\left(R_j = 1 \middle| \overline{C}_j\right)$ for all $j > 1$. Outcomes that are neither MAR nor MCAR are MNAR. Under MAR, $\pi_j = \prod_{k=1}^{j} \lambda_k$, where $\lambda_k = \Pr\left(R_k = 1 \middle| R_{k-1} = 1, \overline{Y}_{k-1}^{\text{obs}}, \overline{C}_k\right)$ for $k > 1$ and $\lambda_1 = \pi_1$. The `xtrccipw` command lets the user specify a model for λ_k.

2.2 The full and reduced dropout models

In the presence of dropout, the RCC method requires specification of a dropout model. The `xtrccipw` command allows the user to choose between two parametric models. In particular, let $g(\cdot)$ represent the logit or probit link function. The default dropout-mechanism model specified by `xtrccipw` is

$$g\left(\lambda_{ik}\right)=\alpha_{0k}+\mathbf{z}_{ik}'\boldsymbol{\alpha}_{1k}+I\left(k>1\right)\overline{Y}_{i(k-1)}^{\text{obs}'}\boldsymbol{\alpha}_{2k} \tag{1}$$

where α_{0k} is the intercept, $\mathbf{z}_{ik}$ represents the vector of time-dependent and time-independent covariates with conformable parameter vector $\boldsymbol{\alpha}_{1k}$, $I(a)=1$ if a is true and $I(a)=0$ otherwise, and $\boldsymbol{\alpha}_{2k}$ represents the conformable parameter vector corresponding to lagged outcome values $\overline{Y}_{i(k-1)}^{\text{obs}}$. Equation (1) is referred to as the full dropout model. Note that α_{0k}, $\boldsymbol{\alpha}_{1k}$, and $\boldsymbol{\alpha}_{2k}$ depend on time (as indexed by k); that is, the dropout model is estimated at each time point by default. If dropout is assumed or known to be completely at random, but truncation is present, the user has the option to specify an MCAR model instead, which sets $\boldsymbol{\alpha}_{2k}=\mathbf{0}$.

The user may want to estimate a reduced model with fewer lags, with possible values lag $=1,\ldots,m-1$. In this case, the dropout mechanism is instead modeled as

$$g\left(\lambda_{ik}\right)=\begin{cases}\alpha_{0k}+\mathbf{z}_{ik}'\boldsymbol{\alpha}_{1k}+L_{ik}'\boldsymbol{\alpha}_{2k} & \text{if } k\leq\text{lag}\\ \alpha_{0}+\mathbf{z}_{ik}'\boldsymbol{\alpha}_{1}+L_{ik}'\boldsymbol{\alpha}_{2} & \text{if } k>\text{lag}\end{cases} \tag{2}$$

where $L_{ik}=(0)$ at $k=1$ and $L_{ik}=\big(Y_{i\{\max(1,k-\text{lag})\}},\ldots,Y_{i(k-1)}\big)$ at $k>1$. Equation (2) is referred to as the reduced dropout model. This model is time dependent for time points $k\leq$ lag but shares the same parameters for time points $k>$ lag. This approach allows `xtrccipw` to estimate fewer parameters by assuming a common dropout model once all the requested lagged outcomes potentially become available for estimation (that is, for time points $k>$ lag). The user has the option to specify a reduced MCAR model instead, which estimates the model $g\big(\lambda_{ik}\big)=\alpha_0+\mathbf{z}_{ik}'\boldsymbol{\alpha}_1$.

Note that the full and reduced MAR models are identical when lag $=m-1$ is set, while the full and reduced MCAR models are different. The full MCAR model specifies a model at each time point, while the reduced MCAR model specifies a common model across all time points.

2.3 Inference

This section briefly describes inference about longitudinal mean outcome models for continuing individuals, conditional on covariates. Let $\mu_{ij}^{\text{RCC}}=E\big(Y_{ij}\big|C_{ij}=1\big)$ denote the mean outcome for individual i whose trajectory is still continuing at time point j. In the regression setting, we might posit a generalized linear model of the form $h\big(\mu_{ij}^{\text{RCC}}\big)=\mathbf{x}_{ij}'\boldsymbol{\beta}^{\text{RCC}}$, where $h(\cdot)$ is a link function, $\mathbf{x}_{ij}$ is an observed $p\times 1$ vector of possibly time-dependent covariates that includes a column of ones for the intercept, and $\boldsymbol{\beta}^{\text{RCC}}$ is the corresponding parameter vector. We refer to this as the outcome model. Let $\mathbf{d}_{ij}'=\partial\mu_{ij}^{\text{RCC}}/\partial\boldsymbol{\beta}^{\text{RCC}}$ denote the Jacobian of partial derivatives of μ_{ij}^{RCC} with respect to $\boldsymbol{\beta}^{\text{RCC}}$.

Following Kurland and Heagerty (2005), consider the vector-estimating equation

$$U\left(\boldsymbol{\beta}^{\text{RCC}}\right)=\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbf{d}_{ij}C_{ij}\frac{R_{ij}}{\pi_{ij}}\left(Y_{ij}-\mu_{ij}^{\text{RCC}}\right)$$

We adopt the convention that if $C_{ij} = 0$, then the summand for individual i at time point j equals 0. The IPW probability π_{ij} is generally unknown in practice but can be consistently estimated if the dropout mechanism model is correctly specified. Let $\widehat{\pi}_{ij}$ represent a consistent estimator of π_{ij}, and let $\widehat{\boldsymbol{\beta}}$ denote the solution to $U\left(\boldsymbol{\beta}^{\text{RCC}}\right) = \mathbf{0}$ under MAR when $\widehat{\pi}_{ij}$ is substituted for π_{ij}. The estimator $\widehat{\boldsymbol{\beta}}$ is consistent and asymptotically multivariate normal for $\boldsymbol{\beta}^{\text{RCC}}$ (Robins, Rotnitzky, and Zhao 1995). The `glm` command is ideal for calculating $\widehat{\boldsymbol{\beta}}$ because by default, it assumes the independence working correlation structure required by RCC, and it allows the user to specify time-varying IPWs through the `pweight` qualifier. The empirical sandwich estimator of the variance of $\widehat{\boldsymbol{\beta}}$ is readily available by specifying the `glm` command option `vce(cluster` *clustvar*`)`, where *clustvar* is the variable that identifies individuals. When computed as if the IPWs are known and fixed, the empirical sandwich estimator is expected to be conservative in general (Robins, Hernán, and Brumback 2000; Robins 2000; Wooldridge 2007). Thus 95% Wald confidence intervals constructed using the empirical sandwich estimator should in general have a coverage probability for $\boldsymbol{\beta}^{\text{RCC}}$ of at least 95%.

3 The xtrccipw command

3.1 Description

The `xtrccipw` command estimates time-specific weights equal to the inverse of the nondropout probability conditioning on continuation. This command uses either the `logit` or the `probit` command to estimate IPWs. The user may then specify that `xtrccipw` run `glm` with the `pweight` qualifier and the `vce(cluster` *clustvar*`)` option to calculate RCC estimates of the outcome-model parameters, along with variance estimates constructed using the empirical sandwich estimator. The `xtrccipw` command runs under Stata 14.

The rest of this section is organized as follows. We describe and illustrate input dataset requirements in an example. We then present the command syntax, along with definitions of all relevant variables and options. Finally, we describe the displayed outputs and stored results.

3.2 Input datasets

The `xtrccipw` command accepts datasets in Stata long format (that is, each row corresponds to one observation at one measurement time point). It then creates indicator variables for truncation and dropout based on the supplied variables for measurement time, truncation time, and outcome-model outcome.

The dataset must include the following variables: unique individual identifiers, measurement time, measurement time index, outcome, and dropout-model covariates. Each row must provide values for unique individual identifiers, measurement time, and measurement time index. For each individual, unique individual identifier values must be identical on all rows, and rows for all possible measurement times and time indices must be included to create truncation and dropout indicators, regardless of outcome value being available or unavailable on any given row (that is, because of dropout or truncation). At the current time index, values for all dropout-model covariates (except for past outcomes) must be provided if an individual had not dropped out by the previous time index (that is, if an outcome value was provided at the previous time index) and had not been truncated by the current time index. The dataset must additionally include a variable for truncation time if truncation occurred for any individual, in which case an individual's truncation time must be identical across all of that individual's rows. Truncation time must be left missing on all rows for each individual without a truncation time.

3.3 Syntax

`xtrccipw` *outcomevar* [*if*] `,` `idvar(`*varlist*`)` `timevar(`*varname*`)`
  `timeidxvar(`*varname*`)` `generate(`*newvar*`)` [`timeidxf(`#`)` `timeidxl(`#`)`
  `trtimevar(`*varname*`)` `linkfxn(`*link*`)` `tdindepvars(`*varlist*`)`
  `tiindepvars(`*varlist*`)` `mcar` `lagreduced(`#`)` `glmvars(`*indepvars*`)`
  `glmfamily(`*familyname*`)` `glmlink(`*linkname*`)`]

outcomevar is the outcome-model outcome variable used as a covariate in the dropout model. If *outcomevar* is an indicator or categorical factor variable, it must be preceded with "`i.`". The other unary operators "`c.`" and "`o.`" are not allowed.

3.4 Options

`idvar(`*varlist*`)` defines variables used to uniquely identify individuals (for example, subjects or panels). This is analogous to *panelvar* in `xtset`. If the `glmvars()` option is specified, then the call to `glm` will include the `vce(cluster` *clustvar*`)` option. `idvar()` is required.

`timevar(`*varname*`)` defines the variable representing the measurement time (for example, visit date). This is analogous to *timevar* in `xtset`. `timevar()` is required.

`timeidxvar(`*varname*`)` defines the variable representing the measurement time index (for example, visit number). All index values must be integers. `timeidxvar()` is required.

`generate(`*newvar*`)` defines the variable name for the estimated IPW. `generate()` is required.

`timeidxf(`#`)` denotes the first time-index value, which must be an integer, to be used in the outcome-model analysis. This must be specified along with `timeidxl()`. The default is the first nonmissing index value found in the current dataset after `if` is applied.

`timeidxl(`#`)` denotes the last time index value, which must be an integer, to be used in the outcome-model analysis. This must be specified along with `timeidxf()`. The default is the last nonmissing index value found in the current dataset after `if` is applied.

`trtimevar(`*varname*`)` denotes the truncation time (for example, truncation date). This must have the same scale as `timevar()`. The default is no truncation.

`linkfxn(`*link*`)` specifies the dropout-model binary link function and only accepts the values `logit` or `probit`. The default is `linkfxn(logit)`.

`tdindepvars(`*varlist*`)` defines the additional dropout-model time-dependent variables (that is, distinct from the time-dependent outcome-model outcome variable). Use spaces to separate multiple variables. Each indicator or categorical factor-variable argument in `tdindepvars()` must be preceded with "`i.`". The other unary operators "`c.`" and "`o.`" are not allowed, and neither is variable-interaction notation (that is, "`#`" or "`##`"). A variable representing the interaction between two variables must be created and included as a distinct variable. The *varlist* syntax is otherwise identical to the *indepvars* syntax for the `logit` or `probit` command. For example, suppose we have two time-dependent binary variables, that is, x and y, and the continuous variable z. If we wish to model dropout dependent on x, y, and z, the interaction between x and y, and the interaction between x and z, we would first create the interaction variables, for example, `generate xy = x * y` and `generate xz = x * z`. Then, we would correspondingly type something like `tdindepvars(i.x i.y i.xy z xz)`. The default is no additional time-dependent variables.

`tiindepvars(`*varlist*`)` defines the dropout-model time-independent variables. The same description as that for `tdindepvars()` applies. The default is no additional time-independent variables.

`mcar` defines whether to use the full MCAR model. This option cannot be specified with `lagreduced()`. The default is the full MAR model.

`lagreduced(`#`)` defines whether and how to use the reduced dropout model. The number of lags, that is, #, can range from 1 to $m - 1$, where m is the number of scheduled study time points. However, specifying $m-1$ lags is identical to specifying the full MAR model. To specify the reduced MCAR model, type `lagreduced(0)`. This option cannot be specified with `mcar`. The default is the full MAR model.

`glmvars(`*indepvars*`)` defines the outcome-model independent variables for `glm`.

`glmfamily(`*familyname*`)` specifies the distribution of *outcomevar* for `glm`. The default is `glmfamily(gaussian)`.

`glmlink(`*linkname*`)` specifies the link function for `glm`. The default is the canonical link for the specified `glmfamily()`.

An example dataset is illustrated in table 1. The variable names correspond to a unique individual identifier `idvar`, measurement time `timevar`, measurement time index `timeidxvar`, continuous outcome `outcomevar`, dropout-model time-dependent continuous covariate `tdindepvar`, dropout-model time-independent binary covariate `tiindepvar`, and truncation time `trtimevar`. The variables C_j, R_j, and S (that is, the truncation indicator, dropout indicator, and number of time points before truncation, respectively) are included only to help illustrate the example in section 2.1, but `xtrccipw` does not output them.

3.5 Displayed outputs

`xtrccipw` displays two outputs. The first is a list of all arguments for verification by the user. The second is a tabulation of the observed values of the `xtrccipw_ec` variable (where `_ec` stands for "error code"), which indicates the number of nonzero observations at each time point for which dropout regression and subsequent probability prediction are successful, or for which there are errors. The `xtrccipw_ec` variable is equal to 0 if regression and prediction are successful, 1 if regression fails because there is either no dropout or all dropout at that time point, 2 if regression fails because all eligible observations are dropped because of regression collinearities, and 3 if regression succeeds but prediction fails. In any of the failure cases, the dropout probability is estimated as the empirical mean of dropout in the risk set (that is, among observations with $R_{i(j-1)} = 1$).

3.6 Stored results

The command attaches five variables to the input dataset. The outcome variable used in estimating the dropout probability while accounting for truncation is stored as `xtrccipw_`*outcomevar*. The value of this variable can differ from that of *outcomevar* in the following way: if a truncation event and outcome are both recorded at time point j, then `xtrccipw` treats truncation as having occurred before the outcome and sets `xtrccipw_`*outcomevar* as undefined (that is, "`.`" in Stata syntax). The indicators for truncation (that is, C represented as `xtrccipwCi`) and dropout (that is, R represented as `xtrccipwRi`) are also stored, as are the estimated IPWs (that is, the *newvar* specified by `generate(`*newvar*`)`). Finally, the `xtrccipw_ec` variable is also output.

4 Example

Our example data came from the National Longitudinal Survey of Young Women (NLSYW). We took a subsample of an available Stata dataset for our analysis, generated truncation, and then analyzed a binary outcome from this analysis sample.

We started with `nlswork5.dta`, a subsample of 4,711 young women ages 14–26 in 1968 that was originally derived to illustrate how to use the `xt` commands. These data are composed of "women in years when employed, not enrolled in school and evidently having completed their education, and with wages in excess of $1/hour but less than $700/hour" (see [XT] `xt`). The longitudinal binary outcome of interest was union membership `union` (`1` if yes, `0` if no). The covariates we used were age, `age`;

ln(wage/gross national product deflator), `ln_wage`; total work experience, `ttl_exp`; birth year, `birth_yr`; and college graduate indicator, `collgrad` (`1` if yes, `0` if no). The identifier variables were NLSYW ID (`idcode`) and interview year (`year`).

For our analysis, we selected the `nlswork5.dta` subsample of women with nonmissing values for any of these outcomes or covariates from years 70 (that is, 1970) through 73, 77, 78, and 80, which gave us 357 individuals. We then generated truncation at follow-up years; no truncation was generated for baseline year 70. Truncation was generated with probability 0.2 if union membership in the previous year was missing. Otherwise, truncation was generated with higher probability if an individual was a union member in the previous year and with lower probability if she was not a member. The degree of increase or decrease in truncation probability itself increased over time. In the *Appendix*, we show the commands used to create `nlswork5-xtrccipw.dta`.

The following output characterizes the analysis dataset:

```
. use nlswork5_xtrccipw
(NLS: Young women 14-26 years of age in 1968. Example dataset for xtrccipw.)
. describe
Contains data from nlswork5_xtrccipw.dta
  obs:         2,499                          NLS: Young women 14-26 years of
                                                age in 1968. Example dataset for
                                                xtrccipw.
 vars:            10                          9 Jan 2017 07:45
 size:        42,483
────────────────────────────────────────────────────────────────────────────────
              storage   display    value
variable name   type    format     label      variable label
────────────────────────────────────────────────────────────────────────────────
idcode          int     %8.0g                 NLS ID
year            byte    %8.0g                 interview year
yearidx         byte    %9.0g                 interview year
truncyear       byte    %9.0g
union           byte    %8.0g                 1 if union
age             byte    %8.0g                 age in current year
ln_wage         float   %9.0g                 ln(wage/GNP deflator)
ttl_exp         float   %9.0g                 total work experience
birth_yr        byte    %8.0g                 birth year
collgrad        byte    %8.0g                 1 if college graduate
────────────────────────────────────────────────────────────────────────────────
Sorted by: idcode  yearidx
```

The following individuals illustrate the three possible truncation and dropout patterns. Individual 5 experienced dropout but not truncation. Individual 20 experienced neither dropout nor truncation. Individual 126 experienced both dropout and truncation.

```
. list idcode year truncyear union age ln_wage ttl_exp birth_yr collgrad if
> inlist(idcode, 5, 20, 126), sepby(idcode) abbreviate(5)

     ┌─────────────────────────────────────────────────────────────────────────┐
     │ idc~e   year   tru~r   union   age    ln_wage    ttl_exp   bir~r   col~d │
     ├─────────────────────────────────────────────────────────────────────────┤
 15. │     5     70       .       0    24   1.820858   3.076923      45       0 │
 16. │     5     71       .       0    25   1.858522   4.038462      45       0 │
 17. │     5     72       .       0    26   1.979301   5.038462      45       0 │
 18. │     5     73       .       0    27   1.990412   6.038462      45       0 │
 19. │     5     77       .       0    31   1.937521   7.576923      45       0 │
 20. │     5     78       .       .    32   2.070492   7.846154      45       0 │
 21. │     5     80       .       .    34   1.830269   9.346154      45       0 │
     ├─────────────────────────────────────────────────────────────────────────┤
 43. │    20     70       .       0    21    2.01878         .5      48       0 │
 44. │    20     71       .       0    22   2.081666        1.5      48       0 │
 45. │    20     72       .       0    23   2.117261   2.403846      48       0 │
 46. │    20     73       .       1    24   2.099896   3.442308      48       0 │
 47. │    20     77       .       0    28    2.10058   5.416667      48       0 │
 48. │    20     78       .       0    29   1.990396   6.493589      48       0 │
 49. │    20     80       .       0    31   1.958695   8.378204      48       0 │
     ├─────────────────────────────────────────────────────────────────────────┤
 64. │   126     70      77       0    21   1.657229    2.01282      48       0 │
 65. │   126     71      77       0    22   1.676201    2.99359      48       0 │
 66. │   126     72      77       0    23   1.943153    3.99359      48       0 │
 67. │   126     73      77       1    24   2.159794   4.974359      48       0 │
 68. │   126     77      77       .    28   2.087653       8.25      48       0 │
 69. │   126     78      77       .    29   2.137434       9.25      48       0 │
 70. │   126     80      77       .    31   2.026384   11.33333      48       0 │
     └─────────────────────────────────────────────────────────────────────────┘
```

We now analyze the example dataset. We regressed `union` on `age`, `ln_wage`, and `birth_yr`. We modeled dropout on `ttl_exp` and `collgrad` using a probit link. We also requested that `xtrccipw` run the RCC outcome-model regression for union membership. The IPW variable was generated as `ipw_full`.

```
* RCC and full dropout model.
. xtrccipw i.union, idvar(idcode) timevar(year) timeidxvar(yearidx)
> generate(ipw_full) trtimevar(truncyear) linkfxn(probit) tdindepvars(ttl_exp)
> tiindepvars(i.collgrad) glmvars(age ln_wage birth_yr) glmfamily(binomial)
```

The `xtrccipw` arguments were output to the Stata Results window for verification. Here `timeidxf` and `timeidxl` took on values derived from the dataset because they were not specified. The dropout-model regression result for each month can also be quickly scanned for errors using the `xtrccipw_ec` variable.

```
outcomevar = i.union
idvar = idcode
timevar = year
timeidxvar = yearidx
generate = ipw_full
timeidxf = 1
timeidxl = 7
trtimevar = truncyear
linkfxn = probit
tdindepvars = ttl_exp
tiindepvars = i.collgrad
mcar =
lagreduced =
glmvars = age ln_wage birth_yr
glmfamily = binomial
glmlink =

───────────────────────────────────────────────
interview  │             xtrccipw_ec
year       │          0            1            3
───────────────────────────────────────────────
         1 │        357
         2 │        159
         3 │        111
         4 │         79                        10
         5 │                      67
         6 │         54                         5
         7 │         42                         9
───────────────────────────────────────────────
```

At this point, the IPW `ipw_full` variable has been calculated and attached to the input dataset. The probability of being a union member was then modeled using a logit link.

```
Iteration 0:   log pseudolikelihood = -711.82082
Iteration 1:   log pseudolikelihood = -704.44499
Iteration 2:   log pseudolikelihood = -704.40354
Iteration 3:   log pseudolikelihood = -704.40354
Generalized linear models                         No. of obs      =        670
Optimization     : ML                             Residual df     =        666
                                                  Scale parameter =          1
Deviance         =  1408.807085                   (1/df) Deviance =   2.115326
Pearson          =  1731.432897                   (1/df) Pearson  =   2.599749
Variance function: V(u) = u*(1-u/1)               [Binomial]
Link function    : g(u) = ln(u/(1-u))             [Logit]
                                                  AIC             =   2.114637
Log pseudolikelihood = -704.4035425               BIC             =   -2925.04
                                  (Std. Err. adjusted for 205 clusters in idcode)
──────────────────────────────────────────────────────────────────────────────
               │               Robust
xtrccipw_union │      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
──────────────────────────────────────────────────────────────────────────────
           age │  -.1432499   .0453955    -3.16   0.002    -.2322235   -.0542764
       ln_wage │   1.230599   .3777844     3.26   0.001     .4901551    1.971043
      birth_yr │  -.0347123   .0798514    -0.43   0.664    -.1912182    .1217937
         _cons │   1.470027   4.487581     0.33   0.743     -7.32547    10.26552
──────────────────────────────────────────────────────────────────────────────
```

Note that while 893 IPW values were calculated, only 670 were used by `glm`. This is because at any given time point with a continuing outcome, `xtrccipw` estimates an IPW regardless of whether the outcome at that time point is missing. In contrast, `glm` uses only complete cases (that is, nonmissing outcomes), thereby excluding the missing outcomes from its analysis.

Excluding `trtimevar(truncyear)` from the `xtrccipw` call resulted in truncation being treated like dropout, with the following dropout-model regression error codes and UR results.

```
. use nlswork5_xtrccipw, clear
(NLS: Young women 14-26 years of age in 1968. Example dataset for xtrccipw.)
. xtrccipw i.union, idvar(idcode) timevar(year) timeidxvar(yearidx)
> generate(ipw_full) linkfxn(probit) tdindepvars(ttl_exp)
> tiindepvars(i.collgrad) glmvars(age ln_wage birth_yr) glmfamily(binomial)
  (output omitted)
------------------------------------
interview  |      xtrccipw_ec
year       |           0          3
-----------+------------------------
         1 |         357
         2 |         205
         3 |         121
         4 |         105
         5 |           6         65
         6 |          54         13
         7 |          42         11
------------------------------------

Iteration 0:   log pseudolikelihood = -997.47372
Iteration 1:   log pseudolikelihood = -985.96245
Iteration 2:   log pseudolikelihood = -985.88336
Iteration 3:   log pseudolikelihood = -985.88336
Generalized linear models                          No. of obs      =        670
Optimization     : ML                              Residual df     =        666
                                                   Scale parameter =          1
Deviance         =  1971.766711                    (1/df) Deviance =   2.960611
Pearson          =  2317.068804                    (1/df) Pearson  =   3.479082
Variance function: V(u) = u*(1-u/1)                [Binomial]
Link function    : g(u) = ln(u/(1-u))              [Logit]
                                                   AIC             =   2.954876
Log pseudolikelihood = -985.8833555                BIC             =   -2362.08
                                  (Std. Err. adjusted for 205 clusters in idcode)
---------------------------------------------------------------------------------
               |               Robust
xtrccipw_union |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
---------------+-----------------------------------------------------------------
           age |  -.1602152   .0473475    -3.38   0.001    -.2530145   -.0674159
       ln_wage |   1.377227   .4380323     3.14   0.002     .5186997    2.235755
      birth_yr |  -.0227574   .0991343    -0.23   0.818    -.2170571    .1715423
         _cons |   1.251279   5.499011     0.23   0.820    -9.526585    12.02914
---------------------------------------------------------------------------------
```

Compared with their RCC counterparts, the UR parameter estimates kept the same signs and did not change much in magnitude. Levels of statistical significance also resembled those under RCC.

The full and reduced MCAR models were also specified to illustrate how they can produce different results. The following is the output for the corresponding RCC full MCAR model:

```
. use nlswork5_xtrccipw, clear
(NLS: Young women 14-26 years of age in 1968. Example dataset for xtrccipw.)
. xtrccipw i.union, idvar(idcode) timevar(year) timeidxvar(yearidx)
> generate(ipw_mcarfull) trtimevar(truncyear) linkfxn(probit)
> tdindepvars(ttl_exp) tiindepvars(i.collgrad) mcar
> glmvars(age ln_wage birth_yr) glmfamily(binomial)
outcomevar = i.union
idvar = idcode
timevar = year
timeidxvar = yearidx
generate = ipw_mcarfull
timeidxf = 1
timeidxl = 7
trtimevar = truncyear
linkfxn = probit
tdindepvars = ttl_exp
tiindepvars = i.collgrad
mcar = mcar
lagreduced =
glmvars = age ln_wage birth_yr
glmfamily = binomial
glmlink =

interview  |            xtrccipw_ec
year       |        0            1            3
-----------+---------------------------------------
         1 |      357
         2 |      159
         3 |      111
         4 |       89
         5 |                    67
         6 |       59
         7 |       48                         3

Iteration 0:   log pseudolikelihood = -706.60703
Iteration 1:   log pseudolikelihood = -699.40805
Iteration 2:   log pseudolikelihood = -699.36553
Iteration 3:   log pseudolikelihood = -699.36553
Generalized linear models                        No. of obs      =        670
Optimization     : ML                            Residual df     =        666
                                                 Scale parameter =          1
Deviance         =  1398.731061                  (1/df) Deviance =   2.100197
Pearson          =  1689.294056                  (1/df) Pearson  =   2.536478
Variance function: V(u) = u*(1-u/1)              [Binomial]
Link function    : g(u) = ln(u/(1-u))            [Logit]
                                                 AIC             =   2.099599
Log pseudolikelihood = -699.3655307              BIC             =  -2935.116
                                (Std. Err. adjusted for 205 clusters in idcode)

                           Robust
xtrccipw_union     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]

          age  -.1464272   .0455808    -3.21   0.001    -.2357638   -.0570906
      ln_wage    1.26635   .3806582     3.33   0.001     .5202734    2.012426
     birth_yr    -.03907   .0794123    -0.49   0.623    -.1947152    .1165752
        _cons   1.720208    4.46455     0.39   0.700     -7.03015    10.47057
```

Here is the output for the corresponding RCC-reduced MCAR model for comparison:

```
. use nlswork5_xtrccipw, clear
(NLS: Young women 14-26 years of age in 1968. Example dataset for xtrccipw.)
. xtrccipw i.union, idvar(idcode) timevar(year) timeidxvar(yearidx)
> generate(ipw_mcarred) trtimevar(truncyear) linkfxn(probit)
> tdindepvars(ttl_exp) tiindepvars(i.collgrad) lagreduced(0)
> glmvars(age ln_wage birth_yr) glmfamily(binomial)
outcomevar = i.union
idvar = idcode
timevar = year
timeidxvar = yearidx
generate = ipw_mcarred
timeidxf = 1
timeidxl = 7
trtimevar = truncyear
linkfxn = probit
tdindepvars = ttl_exp
tiindepvars = i.collgrad
mcar =
lagreduced = 0
glmvars = age ln_wage birth_yr
glmfamily = binomial
glmlink =

──────────┬────────────
interview │ xtrccipw_ec
year      │           0
──────────┼────────────
        1 │         357
        2 │         159
        3 │         111
        4 │          89
        5 │          67
        6 │          59
        7 │          51
──────────┴────────────

Iteration 0:   log pseudolikelihood =  -768.5719
Iteration 1:   log pseudolikelihood = -759.70025
Iteration 2:   log pseudolikelihood =  -759.6436
Iteration 3:   log pseudolikelihood = -759.64359
```

```
Generalized linear models                         No. of obs      =        670
Optimization     : ML                             Residual df     =        666
                                                  Scale parameter =          1
Deviance         =  1519.287182                   (1/df) Deviance =   2.281212
Pearson          =  1876.698513                   (1/df) Pearson  =   2.817866

Variance function: V(u) = u*(1-u/1)               [Binomial]
Link function    : g(u) = ln(u/(1-u))             [Logit]

                                                  AIC             =   2.279533
Log pseudolikelihood = -759.6435911               BIC             =   -2814.56
                                 (Std. Err. adjusted for 205 clusters in idcode)

                              Robust
xtrccipw_union      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]

           age  -.1507094   .0461782    -3.26   0.001     -.241217   -.0602017
       ln_wage   1.261449   .4020585     3.14   0.002      .473429    2.049469
      birth_yr  -.0309284   .0848459    -0.36   0.715    -.1972234    .1353666
         _cons   1.476253   4.720117     0.31   0.754    -7.775006    10.72751
```

5 Simulation study and PEP data analysis

In this section, we report results from a simulation study and reanalysis of the PEP analysis data from Kurland and Heagerty (2005).

5.1 Simulation study

The data-generating specifications used to simulate 1,000 datasets with 1,000 individuals each were similar to those found in section 5 of Kurland and Heagerty (2005) and are summarized as follows. The outcome of interest was a binary variable representing ADL disability, denoted by $Y_{ij} = 1$ if individual i is disabled at time point $j = 1, \ldots, 5$, and $Y_{ij} = 0$ otherwise. The relevant covariates were $\text{sex}_i = 0$ for women (and $\text{sex}_i = 1$ otherwise), $\text{time}_{ij} = \text{age}_{ij} - 65$ (where $\text{age}_{ij} = 65, 70, 75, 80, 85$ years), and sex-time interaction. Let $\boldsymbol{\beta}^{\text{RCC}} = \left(\beta_0, \beta_1, \beta_2, \beta_3\right)'$ denote the corresponding vector of coefficients. The binary outcome RCC model was specified as

$$\text{logit}\left\{E\left(Y_{ij} \middle| C_{ij} = 1\right)\right\} = \beta_0 + \beta_1 \times \text{sex}_i + \beta_2 \times \text{time}_{ij} + \beta_3 \times \text{sex}_i \times \text{time}_{ij}$$

with $\boldsymbol{\beta}^{\text{RCC}} = (-2.19, 0.5, 0.1, -0.025)'$. The binary outcome was defined as $Y_{ij} = I\left(Y_{ij}^* > 0\right)$, where Y_{ij}^* was a normally distributed variable with mean μ_{ij}^U and standard deviation $\sigma_{Y^*} = 0.15$. The correlation for the vector of outcomes $\left(Y_{i1}^*, \ldots, Y_{i5}^*\right)$ was order-1 autoregressive (AR1). Nontruncation was defined as $C_{ij} = I\left(\mathcal{S}_i > \text{age}_{ij}\right)$, where $\mathcal{S}_i$ represented time of death, a normally distributed variable with mean 85 for women and 80 for men and standard deviation $\sigma_S = 5$. The correlation among Y_{ij}^* was set as 0.7, and the covariance of Y_{ij}^* and $\mathcal{S}_i$ was set as -0.4 for women and -0.3 for men. By using the identity

$$E\left(Y_{ij}\middle|C_{ij}=1\right)=\Pr\left(Y_{ij}^{*}>0\middle|\mathcal{S}_i>\text{age}_{ij}\right)=\frac{\Pr\left(Y_{ij}^{*}>0,\mathcal{S}_i>\text{age}_{ij}\right)}{\Pr\left(\mathcal{S}_i>\text{age}_{ij}\right)}$$

values for μ_{ij}^{U} were calculated via bisection with an arbitrary precision tolerance of 0.0001. All μ_{ij}^{U} values were calculated using the `pmvnorm()` function of the `mvtnorm` package in R. Dropout was generated by specifying

$$\text{logit}\left(\lambda_{ik}\right)=\phi_0+\phi_1\left(\mathcal{S}_i-\text{age}_{ij}\right)$$

where $\phi_0=-0.5$ and $\phi_1=0.15$. Truncation or dropout was not allowed at the first time point.

The following three estimators mirror those of Kurland and Heagerty (2005) and were used to estimate the mean binary outcome. (The marginalized transition model was not included because its technical specifications were beyond the scope of this article, and its inclusion was not necessary to demonstrate the simulation-based performance of RCC.)

1. IEE: GEE with independent working correlation. This is identical to the Kurland and Heagerty (2005) independence estimating equations (IEE) model (that is, model with parameters estimated using IEEE).
2. GEE-AR1: GEE with AR1 working correlation. This is similar to the Kurland and Heagerty (2005) inverse probability of censoring weighted (IPCW)-GEE model (that is, model with parameters estimated using IPCW-GEE) but without IPWs. (The original IPCW GEE model was not reproduced because to date, no Stata commands allow for GEE estimation with time-varying weights.)
3. RCC: IEE with correctly specified IPWs. This is identical to the Kurland and Heagerty (2005) IPCW-IEE model (that is, model with parameters estimated using IPCW-IEEE).

The RCC estimator was the only estimator expected to be consistent for the $\boldsymbol{\beta}^{\text{RCC}}$ coefficients. For each β_p where $p=0,\ldots,3$, the empirical relative bias was calculated by taking the average of the empirical bias over all datasets as a percentage of β_p, and the coverage probability was calculated as the percentage of all confidence intervals that contained β_p.

We generated simulated datasets in Stata 14 using the parameter values above and analyzed them as follows. RCC IPWs were estimated using the following code:

```
. xtrccipw i.Yij, idvar(idvarname) timevar(ageij) timeidxvar(timeidx)
> generate(ipw_sims) trtimevar(trunctime) linkfxn(logit) tdindepvars(Siminusageij)
> mcar
```

`i.Yij` represents Y_{ij}, `ageij` represents age_{ij}, and `Siminusageij` represents $\mathcal{S}_i-\text{age}_{ij}$. After specifying the individual-identifier and measurement-time variables using `xtset` `idvarname timeij`, where `timeij` represents time$_{ij}$, we implemented the IEE estimator via the following code:

```
. xtgee xtrccipw_Yij i.sexi timeij sexitimeij, family(binomial) vce(robust)
> corr(independent)
```

`xtrccipw_Yij` represents the outcome variable used by `xtrccipw`, `i.sexi` represents sex$_i$, and `sexitimeij` represents sex$_i$ $\times$ time$_{ij}$. The code used to implement the GEE-AR1 estimator was identical, except that the AR1 working correlation was specified using `corr(ar 1)`. The RCC estimator was implemented with the following code:

```
. glm xtrccipw_Yij i.sexi timeij sexitimeij [pweight=ipw_sims], family(binomial)
> vce(cluster idvarname)
```

The simulation results are listed in table 2. RCC produced the smallest empirical relative bias and was the only approach that exhibited coverage close to or greater than the 95% nominal level for all β_p. These results qualitatively agree with the corresponding empirical relative bias findings in table 4 of Kurland and Heagerty (2005).

Table 2. Simulation study results: Empirical relative bias (coverage probability)

	Intercept ($\beta_0 = -2.19$)	Sex ($\beta_1 = 0.50$)	Time ($\beta_2 = 0.10$)	Sex $\times$ Time ($\beta_3 = -0.025$)
IEE	2 (93.7)	3 (100.0)	−16 (73.5)	−14 (99.1)
GEE-AR1	8 (81.0)	−2 (99.6)	2 (94.9)	−33 (97.3)
RCC	0 (94.8)	1 (99.7)	0 (95.5)	1 (97.8)

5.2 PEP data analysis

We now reanalyze the Kurland and Heagerty (2005) analysis data from the PEP study. Few individuals dropped out (n = 17, 2.3%), and only 62 (8.2%) died in the first two years of the study. Kurland and Heagerty (2005) estimated the association of ADL disability with ADL-disability risk group (that is, risk levels low, medium, and high), month, month^2, and the interaction between month and risk group. Their dropout model included all of these covariates in addition to sex, ADL-disability status at the previous month to reflect the MAR assumption, and a baseline depression indicator.

To analyze the PEP data, we called the `xtrccipw` command as follows, with the relevant output displayed. The variables were study ID (`studyid`), month (`month`), month index (`monthidx`), ADL disability (`adldis` = `1` if disabled; `0` otherwise), risk group (`rgamed` = `0`, `rgahigh` = `0` for low; `rgamed` = `1`, `rgahigh` = `0` for medium; and `rgamed` = `0`, `rgahigh` = `1` for high), month^2 (`monthsq`), medium-risk interaction with month (`rgamedmonth` = `rgamed` $\times$ `month`), high-risk interaction with month (`rgahighmonth` = `rgahigh` $\times$ `month`), and ADL disability status at the previous month (`lagreduced` = `1`). The dropout mechanism was modeled using a logit link.

```
. xtrccipw i.adldis, idvar(studyid) timevar(month) timeidxvar(monthidx)
> generate(ipw_pep) trtimevar(deathmo) linkfxn(logit) tdindepvars(month monthsq
> rgamedmonth rgahighmonth) tiindepvars(i.rgamed i.rgahigh i.sex i.depresbl)
> lagreduced(1) glmvars(month monthsq rgamedmonth rgahighmonth i.rgamed i.rgahigh)
> glmfam(binomial)
outcomevar = i.adldis
idvar = studyid
timevar = month
timeidxvar = monthidx
generate = ipw_pep
timeidxf = 1
timeidxl = 24
trtimevar = deathmo
linkfxn = logit
tdindepvars = month monthsq rgamedmonth rgahighmonth
tiindepvars = i.rgamed i.rgahigh i.sex i.depresbl
mcar =
lagreduced = 1
glmvars = month monthsq rgamedmonth rgahighmonth i.rgamed i.rgahigh
glmfamily = binomial
glmlink =

──────────┬───────────
          │ xtrccipw_ec
 monthidx │    0     1
──────────┼───────────
        1 │         752
        2 │  750
        3 │  748
        4 │  743
        5 │  742
        6 │  740
        7 │  735
        8 │  731
        9 │  730
       10 │  729
       11 │  727
       12 │  721
       13 │  715
       14 │  712
       15 │  710
       16 │  706
       17 │  701
       18 │  700
       19 │  696
       20 │  690
       21 │  686
       22 │  681
       23 │  677
       24 │  674
──────────┴───────────

Iteration 0:   log pseudolikelihood = -4805.9074
Iteration 1:   log pseudolikelihood = -4456.9226
Iteration 2:   log pseudolikelihood = -4448.8392
Iteration 3:   log pseudolikelihood = -4448.7424
Iteration 4:   log pseudolikelihood = -4448.7424
```

```
Generalized linear models                         No. of obs      =     17,177
Optimization     : ML                             Residual df     =     17,170
                                                  Scale parameter =          1
Deviance         =  8897.484773                   (1/df) Deviance =   .5181995
Pearson          =  17402.11245                   (1/df) Pearson  =   1.013518

Variance function: V(u) = u*(1-u/1)               [Binomial]
Link function    : g(u) = ln(u/(1-u))             [Logit]

                                                  AIC             =   .5188033
Log pseudolikelihood = -4448.742386               BIC             =  -158532.8

                                 (Std. Err. adjusted for 752 clusters in studyid)
------------------------------------------------------------------------------
               |               Robust
xtrccipw_adldis|      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
---------------+--------------------------------------------------------------
         month |    .042531   .0136743     3.11   0.002     .0157298    .0693322
       monthsq |  -.0023904   .0007797    -3.07   0.002    -.0039185   -.0008622
  rgamedmonth  |   .0007953   .0159911     0.05   0.960    -.0305466    .0321372
 rgahighmonth  |   .0239548   .0186385     1.29   0.199     -.012576    .0604855
     1.rgamed  |   1.869464   .2275534     8.22   0.000     1.423468     2.31546
    1.rgahigh  |   2.186206   .2463283     8.88   0.000     1.703412    2.669001
         _cons |  -3.532125   .1850643   -19.09   0.000    -3.894844   -3.169405
------------------------------------------------------------------------------
```

These estimates were used to produce figure 1. The predicted trajectories match the fitted curves for the IPCW-IEE estimator in figure 3 of Kurland and Heagerty (2005). The fitted odds ratio comparing odds of disability in the high-risk group with that of the low-risk group at the last time point is 8.90, while Kurland and Heagerty (2005) estimated this odds ratio as 8.95. This minor difference likely results from 752 individuals in the data we analyzed (provided by Professor Kurland) compared with 754 individuals used by Kurland and Heagerty (2005).

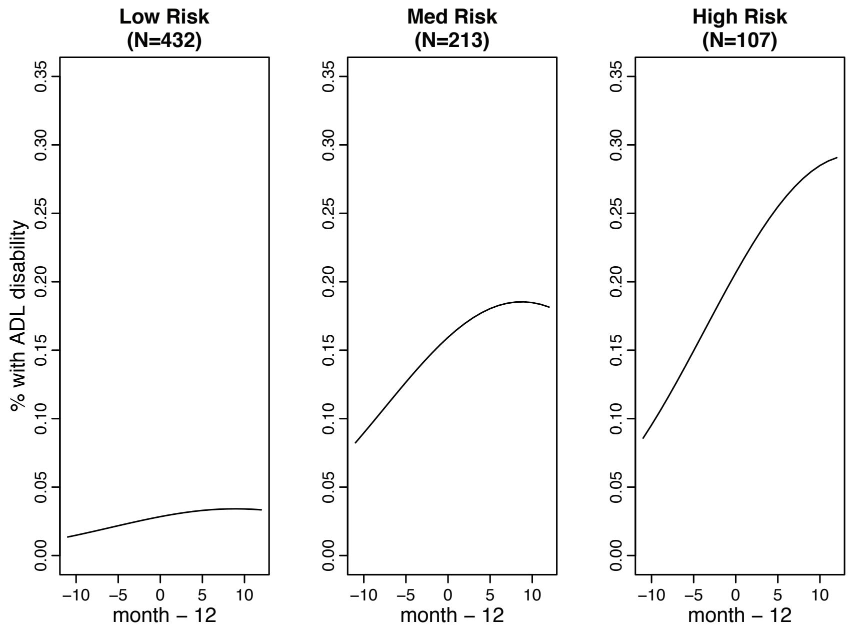


Figure 1. Predicted trajectories for PEP data by risk group

6 Discussion

In this article, we introduced the `xtrccipw` command to estimate the IPWs used to conduct WEE regression and, in particular, RCC. The assumed dropout-probability mechanism could be specified using either a logit or probit link function. We noted asymptotic properties of the subsequent `glm` mean and empirical sandwich variance estimates and demonstrated `xtrccipw` using an example with binary outcomes. Finally, we used `xtrccipw` to conduct a simulation study similar to that of Kurland and Heagerty (2005) and to reanalyze their original study findings.

The `xtrccipw` command does have some limitations. The command can estimate IPWs only if missingness is monotonic, while many studies suffer from nonmonotonic (that is, arbitrary or intermittent) missingness. To use `xtrccipw`, one may construct an "artificial" dropout indicator that treats the first instance of missingness as dropout, discarding any subsequent nonmissing outcomes (Robins, Rotnitzky, and Zhao 1995). One can also impute arbitrarily missing outcomes up to the last nonmissing outcome, as done in Kurland and Heagerty (2005); however, valid subsequent inferences would need to account for imputation.

The RCC method is appropriate when one wishes to draw inference about a target population or real-world population that is itself subject to truncation and when one is interested only in the subset of continuing outcomes in the target population. For example, the PEP study investigators were interested only in the target population of living individuals. The `xtrccipw` command gives the user readily available software to run a WEE or RCC analysis or to simply calculate the relevant IPWs for longitudinal outcomes.

7 Acknowledgments

This work was supported in part by grants SIP 13-01 U48-CCU409660-09, SIP 26-04 U48-DP000059-01, and SIP 22-09 U48-DP001944-01 from the Prevention Research Centers Special Interest Project of the U.S. Centers for Disease Control and Prevention; grant P30-AI50410 from the University of North Carolina Center for AIDS Research; grants DHHS/NIH/FIC 2-D43 Tw01039-06 and R24 Tw00798 of the American Recovery and Reinvestment Act from the National Institutes of Health Fogarty AIDS International Training and Research Program; grant OPP53107 from the Bill and Melinda Gates Foundation; grant R24 HD050924 from the Carolina Population Center; grant R01-AI029168; the U.S. National Institute of Allergy and Infectious Diseases grant R01-AI085073; and the National Institute of Environmental Health Sciences grant R01ES020619. The content of this article is solely the responsibility of the authors and does not necessarily represent the official views of Centers for Disease Control and Prevention, University of North Carolina, National Institutes of Health, Bill and Melinda Gates Foundation, Carolina Population Center, the National Institute of Allergy and Infectious Diseases, or the National Institute of Environmental Health Sciences. The authors would like to thank Dr. Brenda Kurland for access to the data from the PEP study and for her help and guidance in working with the datasets.

8 References

Basu, A., and W. G. Manning. 2010. Estimating lifetime or episode-of-illness costs under censoring. *Health Economics* 19: 1010–1028.

Billingham, L. J., and K. R. Abrams. 2002. Simultaneous analysis of quality of life and survival data. *Statistical Methods in Medical Research* 11: 25–48.

le Cessie, S., E. G. E. de Vries, C. Buijs, and W. J. Post. 2009. Analyzing longitudinal data with patients in different disease states during follow-up and death as final state. *Statistics in Medicine* 28: 3829–3843.

Diggle, P. J., P. Heagerty, K.-Y. Liang, and S. L. Zeger. 2002. *Analysis of Longitudinal Data*. 2nd ed. Oxford: Oxford University Press.

Dufouil, C., C. Brayne, and D. Clayton. 2004. Analysis of longitudinal studies with death and drop-out: A case study. *Statistics in Medicine* 23: 2215–2226.

Flax, V. L., M. E. Bentley, C. S. Chasela, D. Kayira, M. G. Hudgens, R. J. Knight, A. Soko, D. J. Jamieson, C. M. van der Horst, and L. S. Adair. 2012. Use of lipidbased nutrient supplements by HIV-infected Malawian women during lactation has no effect on infant growth from 0 to 24 weeks. *Journal of Nutrition* 142: 1350–1356.

Gill, T. M. 2014. Disentangling the disabling process: Insights from the precipitating events project. *Gerontologist* 54: 533–549.

Gill, T. M., M. M. Desai, E. A. Gahbauer, T. R. Holford, and C. S. Williams. 2001. Restricted activity among community-living older persons: Incidence, precipitants, and health care utilization. *Annals of Internal Medicine* 135: 313–321.

Guo, X., and B. P. Carlin. 2004. Separate and joint modeling of longitudinal and event time data using standard computer packages. *American Statistician* 58: 16–24.

Henderson, R., P. Diggle, and A. Dobson. 2000. Joint modelling of longitudinal measurements and event time data. *Biostatistics* 1: 465–480.

van der Horst, C., C. Chasela, Y. Ahmed, I. Hoffman, M. Hosseinipour, R. Knight, S. Fiscus, M. Hudgens, P. Kazembe, M. Bentley, L. Adair, E. Piwoz, F. Martinson, A. Duerr, A. Kourtis, A. E. Loeliger, B. Tohill, S. Ellington, and D. Jamieson. 2009. Modifications of a large HIV prevention clinical trial to fit changing realities: A case study of the Breastfeeding, Antiretroviral, and Nutrition (BAN) protocol in Lilongwe, Malawi. *Contemporary Clinical Trials* 30: 24–33.

Kurland, B. F., and P. J. Heagerty. 2005. Directly parameterized regression conditioning on being alive: Analysis of longitudinal data truncated by deaths. *Biostatistics* 6: 241–258.

Kurland, B. F., L. L. Johnson, B. L. Egleston, and P. H. Diehr. 2009. Longitudinal data with follow-up truncated by death: Match the analysis method to research aims. *Statistical Science* 24: 211–222.

Liang, K.-Y., and S. L. Zeger. 1986. Longitudinal data analysis using generalized linear models. *Biometrika* 73: 13–22.

Little, R. J. A., and D. B. Rubin. 2002. *Statistical Analysis with Missing Data*. 2nd ed. Hoboken, NJ: Wiley.

Pauler, D. K., S. McCoy, and C. Moinpour. 2003. Pattern mixture models for longitudinal quality of life studies in advanced stage disease. *Statistics in Medicine* 22: 795–809.

Ribaudo, H. J., S. G. Thompson, and T. G. Allen-Mersh. 2000. A joint analysis of quality of life and survival using a random effect selection model. *Statistics in Medicine* 19: 3237–3250.

Robins, J. M. 2000. Marginal structural models versus structural nested models as tools for causal inference. In *Statistical Models in Epidemiology, the Environment, and Clinical Trials*, ed. M. E. Halloran and D. Berry, 95–133. New York: Springer.

Robins, J. M., M. A. Hernán, and B. Brumback. 2000. Marginal structural models and causal inference in epidemiology. *Epidemiology* 11: 550–560.

Robins, J. M., A. Rotnitzky, and L. P. Zhao. 1995. Analysis of semiparametric regression models for repeated outcomes in the presence of missing data. *Journal of the American Statistical Association* 90: 106–121.

Rubin, D. B. 1976. Inference and missing data. *Biometrika* 63: 581–592.

Scharfstein, D. O., A. Rotnitzky, and J. M. Robins. 1999. Adjusting for nonignorable drop-out using semiparametric nonresponse models. *Journal of the American Statistical Association* 94: 1096–1120.

Shardell, M., and R. R. Miller. 2008. Weighted estimating equations for longitudinal studies with death and non-monotone missing time-dependent covariates and outcomes. *Statistics in Medicine* 27: 1008–1025.

Wooldridge, J. M. 2007. Inverse probability weighted estimation for general missing data problems. *Journal of Econometrics* 141: 1281–1301.

About the authors

Eric J. Daza, DrPH, is a postdoctoral research fellow in the Stanford Prevention Research Center at Stanford University School of Medicine. His research focuses on causal inference, longitudinal missing-data methods, personalized and mobile health (mhealth), quantified-self projects and *n*-of-1 trials, Asian-American health (focusing on Filipinos), gut-microbiome research, and reproducibility.

Michael G. Hudgens, PhD, is a professor in the Department of Biostatistics at the University of North Carolina at Chapel Hill. His research focuses on survival analysis, causal inference, infectious diseases, and epidemiology. He is the Director of the Center for AIDS Research Biostatistics Core.

Amy H. Herring, ScD, is a professor in both the Department of Biostatistics and the Carolina Population Center at the University of North Carolina at Chapel Hill. Her research focuses on developing semiparametric Bayesian hierarchical models for highly correlated exposures, exposures to mixtures, and multivariate outcomes; developing statistical methods for missing and mismeasured exposure data; and applications to environmental and reproductive epidemiology.

Appendix: NLSYW example creation code

```
use "http://www.stata-press.com/data/r14/nlswork5.dta"

** Only keep records for subsample women with any survey responses available
** from years 70 through 73, 77, 78, and 80. We start at year 70 because the
** binary outcome of interest (union) is completely missing for years 68 and 69.
keep idcode year
keep if (70 <= year & year <= 80 & year != 75)
generate dummy = 1
reshape wide dummy, i(idcode) j(year)
egen yearsavailable = rowtotal(dummy*)
keep if (yearsavailable == 7)
keep idcode
merge 1:m idcode using "http://www.stata-press.com/data/r14/nlswork5.dta"
keep if (_merge == 3 & 70 <= year & year <= 80 & year != 75)
keep idcode year age ln_wage ttl_exp birth_yr collgrad union
misstable summarize union

** Identify first and last years of any observations.
sort idcode year
by idcode : egen yearidx = seq()
foreach outcomevar in union {
   generate _firstyearRD1`outcomevar´ = (`outcomevar´ < .)
   generate firstyearRD1`outcomevar´ = .
   replace firstyearRD1`outcomevar´ = _firstyearRD1`outcomevar´     ///
         if (yearidx == 1)
   replace firstyearRD1`outcomevar´ =                               ///
         _firstyearRD1`outcomevar´ * firstyearRD1`outcomevar´[_n-1] ///
         if (yearidx > 1)
   drop _firstyearRD1`outcomevar´
   rename firstyearRD1`outcomevar´ RD`outcomevar´
   replace `outcomevar´ = . if (RD`outcomevar´ == 0)
}
keep idcode yearidx year union birth_yr age collgrad ttl_exp ln_wage RDunion
tempfile nlswork5_sub1
save "`nlswork5_sub1´", replace

** Generate no truncation in year 70 and generate truncation based on union
** status at previous year for all subsequent years.
use "`nlswork5_sub1´", clear
keep idcode yearidx year union
sort idcode yearidx
reshape wide union year, i(idcode) j(yearidx)
generate truncyear = .
generate Ci1 = 1
local yearidx = 1
forvalues yearidx = 2/7 {
   local yearidxminus1 = `yearidx´ - 1
   set seed 140925
   generate lambda`yearidx´ = 0.8
   replace lambda`yearidx´ = 0.8 - 0.65 * (`yearidx´ / 7) if ///
     (union`yearidxminus1´ == 1)
   replace lambda`yearidx´ = 0.8 + 0.05 * (`yearidx´ / 7) if ///
      (union`yearidxminus1´ == 0)
   generate Ci`yearidx´ = Ci`yearidxminus1´ * rbinomial(1, lambda`yearidx´)
   replace truncyear = year`yearidx´ if (Ci`yearidx´ == 0 & Ci`yearidxminus1´ == 1)
}
reshape long union year Ci, i(idcode) j(yearidx)
merge 1:1 idcode yearidx using "`nlswork5_sub1´"
drop _merge
foreach varname in union RDunion {
   replace `varname´ = . if (truncyear < . & year >= truncyear)
}
keep idcode year yearidx truncyear union age ln_wage ttl_exp birth_yr collgrad
order idcode year yearidx truncyear union age ln_wage ttl_exp birth_yr collgrad
compress
label data "NLS: Young women 14-26 years of age in 1968. Example dataset for ///
   xtrccipw."
```

The Stata Journal (2017)
17, Number 2, pp. 279–313

Heuristic criteria for selecting an optimal aspect ratio in a two-variable line plot

Demetris Christodoulou
The University of Sydney
Sydney, Australia
demetris.christodoulou@sydney.edu.au

Abstract. Line plots encode a series of slopes from adjoining coordinates and aim to reveal suggestive patterns in the sequential rates of change. The choice of aspect ratio imposed on the line plot largely determines the judged prevalence of patterns in the bivariate series and the degree of steepness in the rates of change. Choosing an appropriate aspect ratio is key to designing informative line plots. The `optaspect` command calculates the optimal aspect ratio in a two-variable line graph using a number of heuristic criteria.

Keywords: gr0069, optaspect, aspect ratio, line plot, time series, banking to 45°

1 Introduction

The aspect ratio of a data plot is defined in Stata as the height-to-width ratio of the plot region.[1] Stata's default aspect ratio is 2.6 : 3.575, which is the same as specifying the `aspectratio(0.7273)` option in any graph. This ratio imposes a shorter height than width and produces images closely in proportion to the dimensions of standard computer monitors. This is by no means a recommended aspect ratio and certainly not suitable for all graphs.

Single line plots are constructed for two main reasons: 1) as a visual table-lookup that assists in determining the ordinate value given an abscissa value and vice versa and 2) for pattern recognition to judge the rates of change by comparing the different orientations of the sequential slopes connecting the series of coordinates (Bertin 1983). The accuracy in addressing the former is invariable to the choice of aspect ratio unless the aspect ratio is so small or large that it makes information illegible. The accuracy in addressing the latter is directly determined by the choice of aspect ratio. Indeed, the optimal aspect ratio is defined as the one that maximizes the accuracy of judgment when contrasting the many rates of change.

In timeline plots, where the abscissa is represented by time, we often look for trending, cyclical and seasonal effects, plus possible structural breaks. The judged prevalence of such effects is also determined by an appropriate aspect ratio. A small aspect ratio would flatten the trend and suppress the peaks and troughs, resulting in a biased

1. Stata defines two broad regions for every image produced by its graphics engine (see `help region_options`). The plot region holds the plotted data, and its size is controlled by the `aspectratio()` option. The graph region holds all other graphical elements that make up the final image, and the image's size is controlled by the `ysize()` and `xsize()` options.

perception of less prevalent cycles or seasons. An excessively large aspect ratio would exaggerate local deviations, give the impression of much higher variability, and obfuscate the comparative degree of steepness in slopes.

For a straight line that connects two coordinates on the bivariate plane, the optimal aspect ratio to convey the rate of change is that line's slope, unless the slope is large. In this case, we best perceive its orientation (see section 2). For example, the optimal aspect ratio for two coordinates characterizing the function $y = x$ is equal to 1. By applying Stata's default aspect ratio of 0.7273, the plot would mislead by suggesting a less steep slope, whereas an aspect ratio of greater than 1 would suggest a steeper slope. Figure 1 illustrates the problem for the function $y = x$ and the aspect ratios of 0.5, 1, and 2.[2] Note that the optimal aspect ratios explored in this article are designed to address the question of how best to portray the rates of change in a series or, more precisely, how to maximize the contrast between a collection of rates of change. However, as discussed in section 2.7, an optimal aspect ratio does not necessarily translate to an optimal portrayal of significance (it might leave the impression that a trivial effect is large or vice versa), and sometimes it makes sense to override the statistical method with judgment.

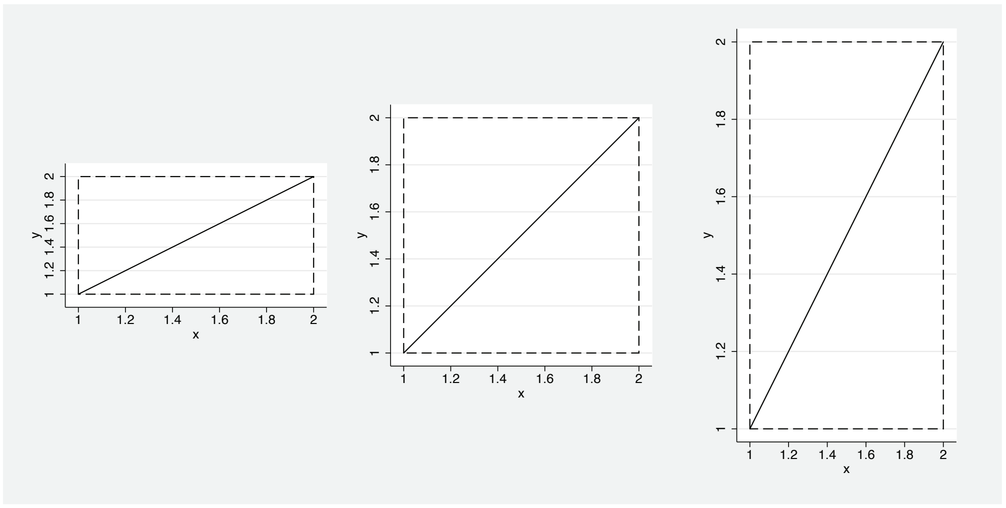


Figure 1. Aspect ratio effect on the perceived rate of change of $y = x$

Optimal aspect ratios address the discrepancy between the actual scale coordinates and the drawing of physical coordinates by producing a standardized representation of the line that maximizes the contrast between a collection of rates of change. Cox (2004) explains that the question of optimal aspect ratio was first addressed by Sir Ronald Aylmer Fisher (1925, 32) in his legendary *Statistical Methods for Research Workers*,

2. To reproduce figure 1, execute the command `twoway (function y = x, range(1 2)) (scatteri 1 1 2 1 2 2 1 2 1 1, recast(line) msymbol(i)), aspectratio(0.5) legend(off) plotregion(margin(medium))`. Do this similarly for the other aspect ratios.

who proposed that "the features of such curves are best brought out if the scales of the two axes are so chosen that the line makes approximately equal angles with the two axes; with nearly vertical, or nearly horizontal lines, changes in the slope are not so readily perceived".

Cleveland, McGill, and McGill (1988) revisit this question more formally and arrive at the same conclusion as Fisher (1925). That is, for a line plot that involves multiple line segments with varying slopes, the best way to compare the steepness of the many slopes is selecting an aspect ratio that would show the typical line slope at 45°. This principle has been dubbed by Cleveland (1993a) as banking to 45°.[3] A number of heuristic criteria have since been proposed as optimal solutions to the banking to 45° rule, starting with Cleveland, McGill, and McGill (1988) and then by Cleveland (1993b), Heer and Agrawala (2006), Talbot, Gerth, and Hanrahan (2011), and Han et al. (2016). The `optaspect` command calculates the optimal aspect ratios for a single two-variable line graph using these methods.

2 Heuristic criteria

Cleveland, McGill, and McGill (1988) explain that the accuracy in decoding rates of change depends on the resolution of the slope's orientation. However, for an $i = 1, 2, \ldots, n+1$ pair of coordinates (y_i, x_i) with ordered data on the x axis so $x_i < x_{i+1}$, there are n number of absolute slopes whose physical appearance is controlled by the aspect ratio $\gamma > 0$:

$$\gamma s_i = \gamma \left| \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right| \tag{1}$$

There is no reason to believe the sign of the orientation affects our judgment (hence, the absolute value), and what matters in perception is the magnitude of the orientation (Cleveland, McGill, and McGill 1988; Guha and Cleveland 2011). Cleveland (1993b) explains that when judging rates of changes, we do not perceive slopes but rather the physical orientation of the absolute slope, which is also determined by the aspect ratio,

$$\theta_i = \arctan(\gamma s_i)$$

where θ_i ranges from 0 to $\pi/2$ radians, or more intuitively, $\theta_i 180/\pi$ ranges from 0° to 90°. Furthermore, our ability to tell the difference between two absolute slopes, γs_i and γs_j, depends on their orientation resolution, defined as the acute angle at the intersection of two line segments:

$$g_{i,j} = \arctan(\gamma s_i) - \arctan(\gamma s_j) \tag{2}$$

The optimal accuracy in the perceived difference between the two rates of change is achieved for γ that sets $g_{i,j}$ at its maximum. If $\gamma \to 0$, then $\theta_i \to 0$ (that is, perceived

3. Cleveland (1993a) explains that the term "banking" is borrowed from the way "banked turns" aid speedy vehicles to take frictionless turns.

to be near horizontal at 0°) and if $\gamma \to \infty$, then $|\theta_i| \to \pi/2$ (that is, perceived to be near vertical at 90°), then in both cases, $g_{i,j} \to 0$, which makes it impossible to distinguish between the two orientations. Therefore, the decoding of visual information fails completely. The obvious compromise for maximizing $g_{i,j}$ is selecting a γ that sets the typical absolute slope at 45°. For the case of two slopes, $g_{i,j}$ is at maximum for the γ that makes the average of the two absolute orientations equal to 45°, described in Cleveland, McGill, and McGill (1988) as the midangle of the orientation resolution:

$$\frac{\arctan(\gamma s_i) + \arctan(\gamma s_j)}{2} = \frac{\pi}{4}$$

In other words, the optimal γ exists when the addition of the two absolute orientations add to 90°; that is, $\arctan(\gamma s_i) + \arctan(\gamma s_j) = \pi/2$.

2.1 Banking on slopes

For a collection of many line segments, Cleveland, McGill, and McGill (1988) propose that the contrast in their rates of change is maximized by selecting a γ that centers both positive and negative slopes on their midangle so the median of the absolute slopes is equal to 1. This is known as the median absolute slope (MAS) criterion. Denote the absolute median slope as $\widetilde{s} = \text{median}\{s_i\}$, where s_i is defined as absolute slopes in (1), and the ranges $R_y = y_{\max} - y_{\min}$ and $R_x = x_{\max} - x_{\min}$. The aspect ratio satisfying the MAS criterion is then

$$\gamma = \frac{1}{\widetilde{s}} \frac{R_y}{R_x} \tag{3}$$

Heer and Agrawala (2006) suggest comparing the MAS rule with the average absolute slope (AAS), which applies the same optimization criterion as (3) with the difference of substituting the median with the average absolute slope, $\bar{s} = 1/n \sum_{i=1}^{n} s_i$:

$$\gamma = \frac{1}{\bar{s}} \frac{R_y}{R_x} \tag{4}$$

As explained later, the AAS criterion performs poorly, but it is a useful comparison benchmark for understanding the performance of other, more appropriate criteria.

2.2 Banking on orientation

Cleveland (1993b) explains how the MAS criterion is also inadequate for practical applications, because the aspect ratio calculated by (3) relies on the premise that we decode slopes when perceiving rates of change. Hence, centering slopes to 1 would give an optimal display. However, behavioral experiments demonstrate that it is the orientation of slopes that affects pattern perception, not the slopes. As such, Cleveland (1993b) proposes optimizing the average discrimination of orientations, which gives the weighted average absolute orientation (WAAO) criterion,

$$\underset{\gamma\in(0,\infty)}{\operatorname{argmin}}\left\{\frac{\sum_{i=1}^{n} w_i \arctan\left(\gamma s_i \frac{R_x}{R_y}\right)}{\sum_{i=1}^{n} w_i} - \frac{\pi}{4}\right\} \tag{5}$$

where the length of each line segment, $w_i = \sqrt{(\Delta x_i/R_x)^2 + \gamma^2\,(\Delta y_i/R_y)^2}$, acts as the weight on the orientation, also a function of γ. There is no closed form solution for (5), but an iterative numerical result can be quickly reached using the Newton–Raphson method.

Heer and Agrawala (2006) note a word of caution when applying the WAAO criterion in line graphs where the x-axis scale is uniformly distributed, which is the case in most time-series lines. They explain that the above weighting scheme gives more weight to line segments with larger absolute orientations that are close to being vertical, resulting in small WAAO values. To gauge the effect of the weight on orientation minimization, they suggest comparing WAAO with its unweighted version of the average absolute orientation (AAO):

$$\underset{\gamma\in(0,\infty)}{\operatorname{argmin}}\left\{\frac{1}{n}\sum_{i=1}^{n}\arctan\left(\gamma s_i \frac{R_x}{R_y}\right) - \frac{\pi}{4}\right\}$$

Nonetheless, note that the weighting scheme of WAAO makes the criterion parameterization invariant, meaning any changes to the way the overall line is encoded in the graph should not change the aspect ratio. That is, switching from a smooth line to a jagged or stairstep connection display should have no effect on whether the x-axis values are equally spaced. This property is important, and, as discussed later, it classifies WAAO as one of the most reliable methods for calculating useful optimal aspect ratios in a wide range of applications.

2.3 Banking on orientation resolution

The MAS criterion centers slopes on 1, and the WAAO criterion centers orientations on 45°. These are optimal solutions for accurately judging the rate of change between pairs of line segments, depending on whether you perceive slopes or orientations (but you probably do the latter). Heer and Agrawala (2006) remain unconvinced on whether these are optimal solutions for a collection of many line segments. As such, instead of just banking on the typical orientation, they propose optimizing directly the collective orientation resolution of all line segments.

For all possible pairs of line segments in the data $\{i, j\}$, the global orientation resolution (GOR) criterion maximizes the orientation resolution of every pair in the data, with respect to γ, as follows,

$$\underset{\gamma\in(0,\infty)}{\operatorname{argmax}}\sum_i\sum_j \min\{{g_{i,j}}^2, (\pi - |g_{i,j}|)^2\}$$

where $g_{i,j}$ is defined in (2) and the $\min\{\cdot\}$ argument makes it clear that the criterion maximizes the orientation resolution of the smallest or acute angles. Optimizing the

GOR criterion requires the most computationally expensive exercise for `optaspect`. The calculation is of order $\mathcal{O}\left(n^2\right)$, whereas all other criteria are only of order $\mathcal{O}\left(n\right)$ or lower.

A compromise method for GOR is to maximize the orientation resolution between successive pairs of slopes, $g_{i,i-1}$, rather than between all possible pairs of slopes. This is dubbed the local orientation resolution (LOR), and it is an optimization problem of order $\mathcal{O}\left(n-1\right)$, stated as follows:

$$\underset{\gamma \in (0,\infty)}{\operatorname{argmax}} \sum_i \min\{{g_{i,i-1}}^2, \left(\pi - |g_{i,i-1}|\right)^2\} \tag{6}$$

The LOR criterion is weak in its foundation because it focuses on optimizing successive localities in the data, which makes it vulnerable to trapping in a local maximum, and it fails to adopt a global view that defines the problem of aspect ratio selection. Both GOR and LOR have been found to perform poorly in autocorrelated series and fail spectacularly in preserving symmetry, plus they are sensitive to small changes in the data (Talbot, Gerth, and Hanrahan 2011). Given the computational price of GOR and the existence of LOR only as a compromise solution with even more doubtful benefits, `optaspect` does not calculate these two criteria by default, but they can still be accessed with the `gor` and `lor` options. Moreover, because these two criteria cannot handle slopes of zero, `optaspect` requires that you also specify the `cullzero` option (see section 2.6).

A claimed improvement to LOR, proposed by Han et al. (2016), is to substitute the L2 norm in (6) with the L1 norm. The criterion then becomes parameterization invariant, satisfying a critical property for aspect ratios. The result is an objective function that maximizes the orientation resolution as a curvature-based line integral—hence, the label weighted local curvature (WLC):

$$\underset{\gamma \in (0,\infty)}{\operatorname{argmax}} \sum_i \min\{|g_{i,i-1}|, \left(\pi - |g_{i,i-1}|\right)\}$$

The WLC criterion is superior to the LOR criterion but still suffers from the same conceptual drawback of maximizing curvature by focusing on successive local optima rather than taking a global view of the problem. Han et al. (2016) admit the WLC is vulnerable to being trapped in local optima. For these reasons, like the LOR, the WLC is available via the `wlc` option and is not reported by default.

2.4 Banking on resultant vector

Optimal aspect ratios maximize the difference (the discrimination) between absolute slope orientations. Connecting the bottom end of the line segment with the smallest slope to the top end of the line segment with the largest slope gives the resultant vector (RV). The length of the resultant vector is at maximum when slopes appear with the same orientation, that is, when γ is close to 0 or ∞. In this respect, Guha and Cleveland (2011) demonstrate how an optimal aspect ratio exists when the RV line is banked with slope 1:

$$\gamma = \frac{R_y / \sum_{i=1}^{n} |\Delta y_i|}{R_x / \sum_{i=1}^{n} |\Delta x_i|} \tag{7}$$

The RV criterion has attracted praise for its parsimony in providing consistent answers with attractive aspect ratios. It also allows us to study the mathematical and geometric properties of optimal aspect ratios more closely, for example, how in the presence of a large white noise component, the aspect ratio will converge with probability to zero (for the proof, see Guha and Cleveland [2011]).

2.5 Banking on arc length

The final criterion examined is by Talbot, Gerth, and Hanrahan (2011), who suggest the objective of minimizing the arc (ARC) length of the total line under the constraint that the area of the plot region remains constant. The minimization problem, with respect to γ, is

$$\underset{\gamma\in(0,\infty)}{\text{argmin}} \sum_{i=1}^{n} \left|\left| \frac{\Delta x_i}{\sqrt{\gamma}}, \sqrt{\gamma}\Delta y_i \right|\right| \tag{8}$$

where $||.||$ indicates the Euclidean length. There is no closed form solution to this problem, but it has a unique minimum that can be quickly reached via iterative substitution. Talbot, Gerth, and Hanrahan (2011) explain how the ARC criterion is consistent with Cleveland's banking to 45° rule because the solution to (8) is a line banked to 45°.

They also show how ARC can be written as an improved version of the GOR criterion, with the main difference that the orientation resolution in ARC is aptly weighted by the lengths of the line segments. This refinement provides two key advantages to ARC over GOR. First, ARC is computationally efficient. Second, because ARC is weighted by line lengths in the same way as the WAAO criterion, it is parameterization invariant, a key property for optimal aspect ratio selection (see section 2.2). Generally, ARC is shown to provide a robust approach to choosing aspect ratios for a wide selection of line patterns by comparison with MAS, AAS, AAO, and especially with GOR and LOR. This is another key reason why `optaspect` offers only the calculation of GOR and LOR as options, whereas ARC is calculated by default.

Another key advantage of the ARC criterion is it preserves symmetry, again in the same way as the WAAO and the RV criteria. Symmetry is evident when displaying algebraic forms but is difficult to find in real data that also contain some degree of random noise. Furthermore, Talbot, Gerth, and Hanrahan (2011) demonstrate that if we replace the Euclidean distance in (8) with the Manhattan distance, the ARC criterion reduces to a closed form solution equivalent to the RV criterion of (7):

$$\underset{\gamma\in(0,\infty)}{\text{argmin}} \sum_{i=1}^{n} \left(|\Delta x_i/\sqrt{\gamma}| + |\sqrt{\gamma}\Delta y_i|\right) \quad \Longrightarrow \quad \gamma = \frac{\sum_{i=1}^{n} |\Delta y_i|}{\sum_{i=1}^{n} |\Delta x_i|} \tag{9}$$

For equally spaced x-values, that is, constant frequency time-series lines, (9) is further reduced to simply $\gamma = R_y/(\sum |\Delta y_i|)$. In this case, (9) gives an equivalent to the AAS criterion of (4), adjusted for R_y/R_x. This equivalence demonstrates the unsuitability of the AAS criterion, which is only reported by `optaspect` for completeness, because the Euclidean distance in ARC is invariant to rotational changes over the two-dimensional plane, whereas the Manhattan distance is not.

2.6 Culling zero slopes

Stagnating series with $\Delta y_i = 0$ give zero slopes (horizontal lines), and stagnating series with $\Delta x_i = 0$ yield infinite slopes (vertical lines); see (1). `optaspect` automatically deals with infinite slopes as missing values and excludes them from computation. `optaspect` reports the number of infinite slopes in the data that have been discarded as a note to the output table. Zero slopes are not discarded unless the user specifies the `cullzero` option. This option follows from Heer and Agrawala (2006), who explain that the effect on judgment on slopes of zero is invariant to the choice of aspect ratio. Therefore, such line segments should not contribute to the determination of the optimal aspect ratio.

The presence of zero slopes is particularly damaging for the LOR and GOR criteria (Talbot, Gerth, and Hanrahan 2011). Thus, in the presence of zero slopes, `optaspect` requires that the specification of `lor` or `gor` always be accompanied by the `cullzero` option. Zero slopes are also problematic for the ARC criterion (Han et al. 2016) and, depending on their frequency, may even fail to converge. `optaspect` does not require culling zero slopes for the ARC because it is reported by default together with other criteria. However, `optaspect` will issue a note that there are zero slopes and that you should consider the `cullzero` option.

2.7 General advice

The application of the aforementioned heuristic criteria as tools for providing accuracy in the judgment of rates of changes is not absent of criticism. Even Cleveland, McGill, and McGill (1988, 296) caution against the automated use of their MAS approach and advise that "application needs to be tempered with judgment".

Talbot, Gerth, and Hanrahan (2011, 2012) suggest the resulting optimal ratios proposed by Cleveland, McGill, and McGill (1988) and Cleveland (1993b) are in fact a product of research design. They find the minimum estimation error in judging slopes does not occur at 45°, and people can decode more accurately the rates of changes with lower aspect ratios than those produced by the MAS and the WAAO criteria. In fact, predating William Cleveland's work, Fisher (1974) presents evidence suggesting that people tend to misjudge the true orientation of a two-variable line by as much as 3°, and this bias is indeed most prominent for lines oriented at 45°.

As we will see in section 4, most criteria do not work well with lengthy series because they cause $\gamma \rightarrow 0$ or $\gamma \rightarrow \infty$, depending on the nature of variation. For example, in stationary series with high-frequency oscillating slopes or noisy series with large random components, as n grows, the aspect ratio $\gamma \rightarrow 0$. In such cases, it is best to split and stack multiple line plot segments under a common y-axis scale and aspect ratio.

The `optaspect` command provides a compromise solution to this problem with the `stackby()` option.

By default, `optaspect` calculates the following criteria: MAS, AAS, WAAO, AAO, ARC, and RV. For reasons explained above, the GOR, LOR, and WLC criteria are only available via the `gor`, `lor`, and `wlc` options. Overall, we know that the MAS, AAS, AAO, GOR, LOR, and WLC criteria do not work well and should be avoided but still may offer some insights on the behavior of the data.

That leaves the effective contrast between WAAO, ARC, and RV. In fact, Talbot, Gerth, and Hanrahan (2011) demonstrate equivalence between ARC and RV, and Han et al. (2016) show that WAAO, ARC, and RV are the only criteria that are parameterization invariant and capable of preserving symmetry in display. However, if I was forced to blindly choose just one criterion for universal application, I would choose the RV approach. In addition to the two key properties mentioned just above, I find that this is the most robust criterion to small changes in the data and all sorts of input.

In the end, you should always resort to common sense and consider the possibility that two diametrically opposed aspect ratios (with inverse height-to-width dimensions) may reveal equally appealing visual information. Section 4.3 demonstrates such an application.

3 The optaspect command

`optaspect` requires specification of two numerical variables intended to be displayed on a line plot and requires that the dataset be sorted in ascending order on the x variable. `sort` is offered as an option.

3.1 Syntax

`optaspect` *yvar xvar* [*if*] [*in*] [*weight*] [`, sort rank cullzero`
  `stackby(`*varname*`) y0 gor lor wlc iterate(`#`) tolerance(`#`)`]

`aweights`, `fweights`, `iweights`, and `pweights` are allowed; see [U] **11.1.6 weight**.

3.2 Options

`sort` orders the dataset on *xvar* in ascending order. `sort` is required for unordered data.

`rank` replaces *xvar* with a variable, t, that takes the values $t = 1, 2, \dots, n$.

`cullzero` ignores zero slopes in the calculation of the optimal aspect ratios.

`stackby(`*varname*`)` specifies the calculation of average optimal aspect ratios over all categories of *varname*, because the line graph will be split and stacked by these categories.

`y0` specifies that the *y* axis of the line graph will contain the baseline value of zero and that the aspect ratio should be adjusted accordingly.

`gor` calculates the computationally expensive GOR criterion.

`lor` calculates the LOR criterion.

`wlc` calculates the WLC criterion.

`iterate(`#`)` and `tolerance(`#`)` specify the number of iterations and convergence tolerance for those criteria that require iterative optimization (AAO, WAAO, LOR, GOR, ARC). The defaults are `iterate(100)` and `tolerance(1e-6)`. These options are rarely used.

3.3 Output and stored results

`optaspect` prints a two-column table with a list of methods for calculating the optimal aspect ratio, accompanied by their respective values that could then be entered in the graph option `aspectratio()`. It also prints a note to this table with relation to the culling of infinite and zero slopes. Infinite slopes are culled by default. If the `cullzero` option is not specified, but there is at least one zero slope in the series, the note will prompt the user to consider the `cullzero` option. If there are no zero or infinite slopes, no note is printed. If the `cullzero` option is specified, the note will report on the number of zero slopes that have been culled. `optaspect` stores the following in `r()`:

Scalars	
`r(mas)`	MAS aspect ratio
`r(aas)`	AAS aspect ratio
`r(waao)`	WAAO aspect ratio
`r(aao)`	AAO aspect ratio
`r(gor)`	GOR aspect ratio
`r(lor)`	LOR aspect ratio
`r(arc)`	ARC aspect ratio
`r(wlc)`	WLC aspect ratio
`r(rv)`	RV aspect ratio
`r(n)`	number of slopes in the data
`r(cull)`	number of zero slopes culled

Following execution, enter one of the above aspect ratios in the graph option `aspectratio()`, for example,

```
. twoway (line yvar xvar), aspectratio(`r(waao)')
```

3.4 Practical considerations

The `stackby(`*varname*`)` option requires that *varname* be a categorical variable, either numerical or string, and *varname* be the variable with which you plan to split and stack the long series. `optaspect` refrains from also automating the segmentation of the series, because this is an inherently personal question, given the profound impact on the aspect ratio. However, `optaspect` is fast, so you can experiment with different configurations.

You should not need to change the default of `iterate(100)` and `tolerance(1e-6)`. If an optimal aspect ratio is not found within a few iterations, it is unlikely to be found beyond 100. Setting iterations low ensures the optimization exits when struggling to find a solution—typically a problem related to GOR, LOR, and possibly ARC. Similarly, there is not much to gain by converging beyond the 1e–6 precision. However, in excessively lengthy series, the GOR criterion will take a long time to converge, so you may specify an even lower number of iterations and lower degree of tolerance to see a preliminary result prior to letting Stata work overtime. Convergence will also be affected by the presence of many consecutive zero slopes, so you should also specify the `cullzero` option.

3.5 Default settings impacting physical orientation resolution

Stata's factory settings and general default behavior in creating line graphs are not attuned to addressing the perceptual problem of accurately judging rates of change. Two key problematic features in default behavior affect the way we perceive the orientation of slopes.

First, as explained in section 1 and demonstrated in figure 1, the default aspect ratio of 2.6 : 3.575 is an ad hoc naïve solution that always warrants a more careful consideration, such as the `optaspect` command. Second, the way Stata determines the default axes labels never ceases to surprise me. It is imperative to take full control of labels in both axes. Otherwise, it is likely they may go beyond the actual range of data and extend the scale, artificially skewing the effective aspect ratio.

You also must be aware of the plot region's margins, which can affect the physical aspect ratio. The default margins are set to an equal amount of space on all sides of the plot region and therefore do not alter the balance of height to width. Adjusting the margins in all sides at once also does not affect the aspect ratio, for example, by specifying `plotregion(margin(large))`. However, adjusting the margins in only some sides would change the physical aspect ratio, for example, `plotregion(margin(top=10 bottom=10))` or `plotregion(margin(right=0 top=0))`.

4 Applications

In this section, I demonstrate `optaspect` using several datasets, also showcasing the usefulness of the various options available.

4.1 Sunspot activity

In the opening pages of Cleveland (1993b), the timeline chart of yearly sunspots from 1749 to 1924 is displayed using two aspect ratios: the naïve 1 : 1 aspect ratio and the optimal aspect ratio calculated by the WAAO criterion at 1 : 0.055. The yearly sunspot dataset is available at the Stata Press website. Obtain all optimal aspect ratios for the same time period as in Cleveland (1993b):

```
. use http://www.stata-press.com/data/r14/sunspot
(TIMESLAB: Wolfer sunspot data)

. optaspect spot time if inrange(time,1749,1924), lor gor wlc

         Aspect ratio criterion                    aspect(#)

  Median Absolute Slope                              0.0654
     Compare to Average Absolute Slope               0.0541
  Weighted Average Absolute Orientation              0.0545
     Compare to Average Absolute Orientation         0.0767
  Arc Length based                                   0.0598
  Global Orientation Resolution                      0.1183
  Local Orientation Resolution                       0.1453
  Weighted Local Curvature                           0.1128
  Resultant Vector                                   0.0541

. scalar waao = r(waao)
```

`optaspect` gives a report with the calculation of aspect ratios on the basis of all heuristic criteria presented in section 2. The WAAO result exactly reproduces the Cleveland (1993b) aspect ratio, which is stored in a `scalar` for subsequent use. As expected, WAAO is closely related to RV and ARC, which share the same properties. For this first application, we also specify the calculation of GOR, LOR, and WLC criteria, but we will refrain from doing so in most subsequent applications because, as discussed above, they have been shown to be unreliable and effectively superseded by the ARC and RV criteria. Figure 2 graphs the sunspot series using the default aspect ratio at $0.7273 : 1$ and the optimal WAAO aspect ratio, following execution of the following graph commands:

```
. local labels ylab(0(50)150, labsize(*.7) angle(0)) xlabel(1749(25)1924,
> labsize(*.7))

. twoway (line spot time) if inrange(time,1749,1924), `labels´
>         title("Default aspect ratio", size(*.75) margin(small))
>         ytitle("Number of sunspots", size(*.8)) xtitle("")
>         fysize(`=2.6*100/3.575´) fxsize(100) name(default,replace)

. twoway (line spot time) if inrange(time,1749,1924), `labels´
>         title("Optimal WAAO aspect ratio", size(*.75) margin(small))
>         ytitle("Number of" "sunspots", size(*.8)) xtitle("")
>         fysize(30) name(waao,replace) aspect(`=waao´)
```

The `fysize()` and `fxsize()` options force `graph combine` to preserve the relative graph region of the individual graphs. The first graph preserves the default graph size by 100% in the x axis and $2.6 \times 100/3.575$ percent in the y axis. The values 2.6 and 3.575 are the default $y : x$ dimensions of the image produced by Stata.[4] The `fysize(30)` option in the second line plot asks to use only 30% of the horizontal space of that graph when combined.

4. To learn more, see `help graph combine` and `help region_options`.

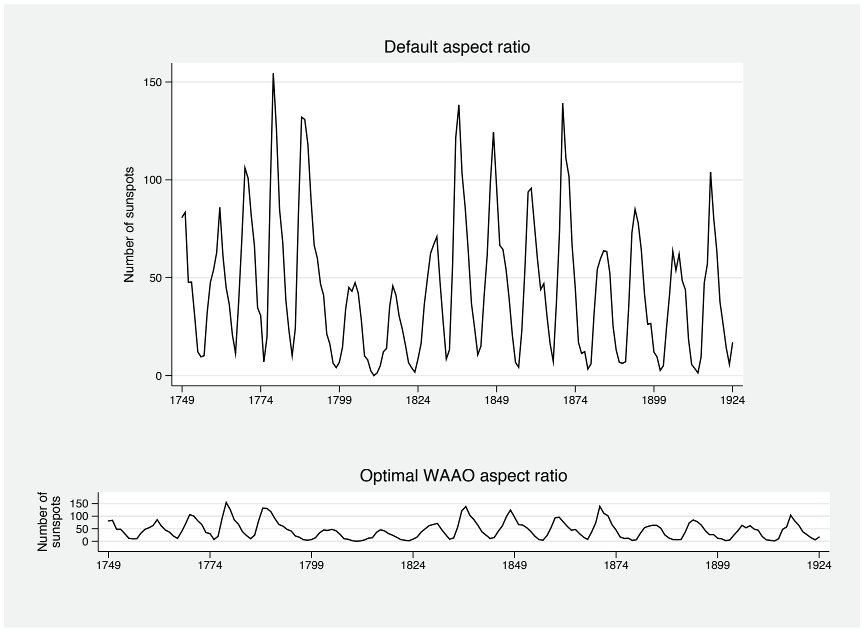


Figure 2. International sunspot number, 1749–1924

The plot with the default aspect ratio in figure 2 draws all attention to the cyclical variation in sunspot activity, spanning about 11 years in length, but makes it difficult to compare the steepness in the rates of change. The WAAO aspect ratio immediately clarifies the sunspot series and enables useful comparison; it reveals an important feature of the data that was hidden before. Sunspot activity is now seen to rise more sharply at the beginning of its cycle and decline more slowly at the end of the cycle. Also, the taller the peak, the more evident the asymmetric behavior becomes, but it is not present in flatter cycles.

The WAAO aspect ratio of 0.0545 is close to the limit of its vertical resolution. If we were to apply an even lower aspect ratio, it would be nearly impossible to decode any useful aspects of the data. The sunspot data have a strong cyclical component, with a series of steep positive slopes followed by a series of steep negative slopes. This causes the optimal aspect ratio $\gamma \rightarrow 0$ as $n \rightarrow \infty$ (Cleveland, McGill, and McGill 1988; Cleveland 1993b). For example, calculating the optimal aspect ratio for an even more lengthy sunspot series, during 1700–2015, would force γ even further toward zero:[5]

5. The data are sourced from the Solar Influences Data Analysis Center (SIDC), Royal Observatory of Belgium; see http://www.sidc.be/.

```
. use sunspot_1700_2015.dta, clear
(Source: http://www.sidc.be/)

. optaspect spot time
```

Aspect ratio criterion	aspect(#)
Median Absolute Slope	0.0370
Compare to Average Absolute Slope	0.0295
Weighted Average Absolute Orientation	0.0298
Compare to Average Absolute Orientation	0.0417
Arc Length based	0.0337
Resultant Vector	0.0295

```
Note: There are 1 zero slopes; consider option cull
```

If we were to impose the suggested WAAO aspect ratio at 0.0298, it would be impossible to discern any useful features of the data—the series would look close to a flat line. As we will see later, this effect is even stronger with seasonal series, where the alternation of positive with negative slopes is even more frequent. In this case, it is best to split and stack the data and calculate the average optimal aspect ratio. This can be done via the `stackby()` option. However, `stackby()` requires access to a categorical variable that will be used to split the series.

Figure 3 reproduces the superbly crafted split-and-stack line plot on sunspot activity offered by the SIDC at the Royal Observatory of Belgium.[6] This is a beautifully informative data plot for four reasons: 1) It heeds the advice of William Cleveland in choosing a relatively small aspect ratio to bring out the asymmetric behavior at the beginning and the end of each cycle. 2) It uses two different colors for highlighting the presence of two measurement scales (yearly and 13-month smoothed). 3) It repeats the last 10 years of each panel to the immediately next panel to emphasize the continuity in the series. 4) The bottom panel is purposefully shown as incomplete because this is a "plot in-progress" filled up with new information as it arrives. Be sure to specify the `cullzero` option in `optaspect` to ignore that one zero slope in the calculation of the aspect ratios:

```
. recode time (1700/1799=0) (1800/1899=1) (1900/1999=2) (2000/2015=3),
> generate(panels)
(316 differences between time and panels)

. generate copies = 1

. replace copies = 2 if (inrange(time,1800,1809) | inrange(time,1900,1909) |
> inrange(time,2000,2009))
(30 real changes made)

. expand copies
(30 observations created)

. by time, sort: replace copies = _n
(30 real changes made)

. replace panels = panels + copies if copies==1
(316 real changes made)
```

6. See http://www.sidc.be/silso/yearlyssnplot.

```
. optaspect spot time if copies==1, stackby(panels) cullzero

        Aspect ratio criterion                       aspect(#)

 Median Absolute Slope                                  0.1498
    Compare to Average Absolute Slope                   0.1262
 Weighted Average Absolute Orientation                  0.1272
    Compare to Average Absolute Orientation             0.1796
 Arc Length based                                       0.1439
 Resultant Vector                                       0.1262

 Note: 1 zero slopes have been culled
. scalar waao = r(waao)
. local o1 ytitle("") ylabel(0(100)200 269, grid angle(0) labsize(*.85))
. local o2 aspect(`=waao´) xtitle("") plotregion(margin(zero))
. twoway (area spot time if panels==1)
>       (area spot time if time<=1749), `o1´ `o2´ legend(off)
>        xlabel(1700(20)1800, grid labsize(*.85)) name(g1,replace)
. twoway (area spot time if panels==2), `o1´ `o2´
>        xlabel(1800(20)1900, grid labsize(*.85)) name(g2,replace)
. twoway (area spot time if panels==3), `o1´ `o2´
>        xlabel(1900(20)2000, grid labsize(*.85)) name(g3,replace)
. twoway (area spot time if panels==4), `o1´ `o2´ xscale(range(2000 2110))
>        xlabel(2000(20)2100, grid labsize(*.85)) name(g4,replace)
. graph combine g1 g2 g3 g4, col(1) imargin(zero) ysize(5) xsize(6)
>      l1title("Sunspot number", margin(zero) size(*.75))
>      b1title("Time (years)", margin(t=0 b=2) size(*.75))
>      note("Source: SIDC; http://www.sidc.be/silso/yearlyssnplot/."
>           "Yearly mean sunspot number (light gray) up to 1749 and monthly
>           13-month smoothed sunspot number (in dark gray) from 1750-2015.",
>           size(*.65)) graphregion(margin(l=10 r=10))
```

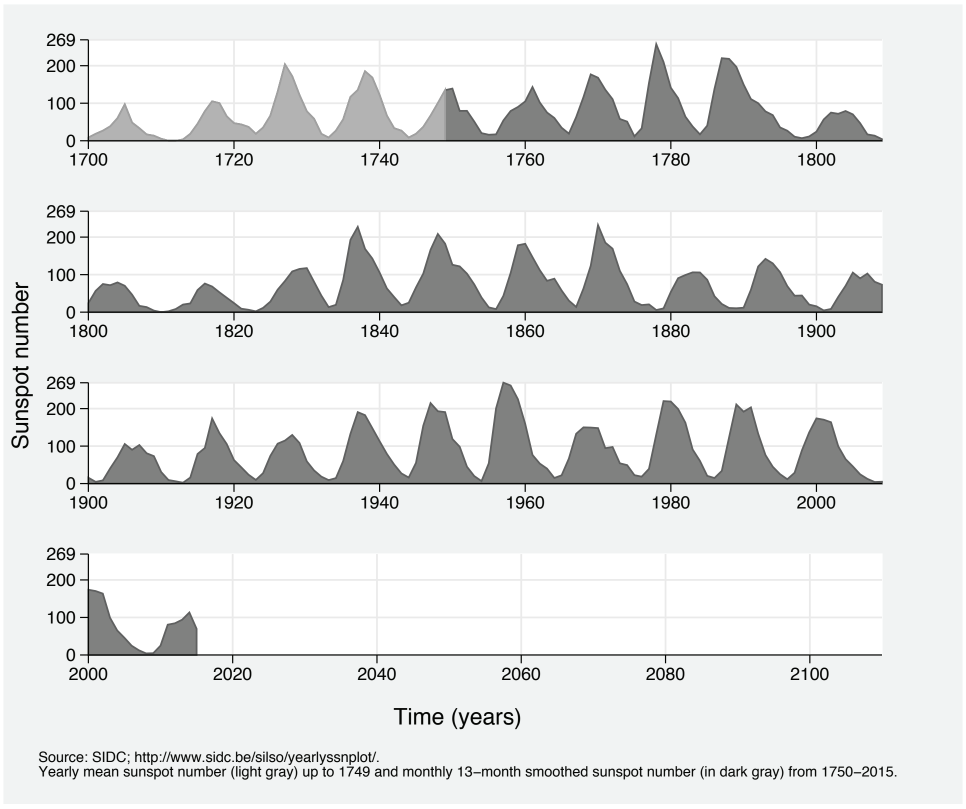


Figure 3. International sunspot number, 1700–2015

4.2 Airline passengers

The Stata Press dataset `air2.dta` holds observations on U.S. domestic airline passengers during 1949–1960. The top graph of figure 4 gives the line plot under Stata's default aspect ratio, which emphasizes how demand for air travel has a strong upward trend with a seasonal component that peaks during July–August. Another important feature of the data is the increased seasonal variation over the years, which appears to be increasing at an exponential rate. Calculate the optimal aspect ratios for this series:

```
. use http://www.stata-press.com/data/r14/air2, clear
(TIMESLAB: Airline passengers)
. optaspect air time, wlc
```

Aspect ratio criterion	aspect(#)
Median Absolute Slope	0.1907
Compare to Average Absolute Slope	0.1400
Weighted Average Absolute Orientation	0.1411
Compare to Average Absolute Orientation	0.2017
Arc Length based	0.0031
Weighted Local Curvature	0.0058
Resultant Vector	0.1401

```
 Note: There are 4 zero slopes; consider option cull
```

The aspect ratios produced by WAAO and RV are reasonable, but the ARC result is disappointingly small, possibly because of ARC's sensitivity to zero slopes (Han et al. 2016). The WLC is also forbiddingly small because it is also sensitive to how the x-axis values are measured. In these data, `time` is measured in terms of fractional years, so January 1949 takes the value 1949, but February 1949 takes the value 1949.083, where $0.083 \equiv 1/12$, March 1949 is 1949.167, and so on. There is no reason to work with such a scale, and it would make more sense to apply a continuous time scale for $t = 1, 2, \dots, n$. This is achieved by specifying `rank` and also by specifying `cullzero` to remove all zero slopes:

```
. optaspect air time, wlc cullzero rank
```

Aspect ratio criterion	aspect(#)
Median Absolute Slope	0.1725
Compare to Average Absolute Slope	0.1362
Weighted Average Absolute Orientation	0.1370
Compare to Average Absolute Orientation	0.1900
Arc Length based	0.0369
Weighted Local Curvature	0.0608
Resultant Vector	0.1362

```
 Note: 4 zero slopes have been culled
. scalar rv = r(rv)
```

The WAAO and RV criteria remain similar to before because they are parameterization invariant but still somewhat affected by the absence of zero slopes. The ARC and the WLC criteria are now more reasonable but still too low to produce a meaningful plot for these data.

The bottom graph of figure 4 gives the resulting line plot with the RV aspect ratio, where the shaded area highlights the yearly range in airline passengers (min to max). The RV aspect ratio elucidates the description of seasonal variation and reveals several important features. First, the seasonal demand for air travel rises gradually from January to June. Second, it peaks in July and remains at about the same level in August. Third, the accelerated growth from June to July is of the same steepness as the accelerated decline from August to September. Fourth, with the help of the shaded area that highlights the range of variation, we can see a dramatic increase in variation over the years, specifically the peak demand in July and August increasing much faster than the off-peak demand in January and November. Hence, there is a suggestion of an exponential growth of change. The following code produces figure 4:

```
. generate year = floor(time)
. by year, sort: egen air_max = max(air)
. by year, sort: egen air_min = min(air)
. twoway (line air time, lwidth(*1.5)), `yl´ `xl´
>         title("Default aspect ratio", size(*.75) margin(small))
>         ylabel(minmax 200(100)500, labsize(*.7) angle(0))
>         xlabel(#12, grid format(%4.0f) labsize(*.7))
>         ytitle("Airline passengers", size(*.8)) xtitle("")
>         plotregion(margin(zero))
>         fysize(`=2.6*100/3.575´) fxsize(100) name(default,replace)
. twoway (rarea air_max air_min time, connect(stair) fcolor(gs14) lcolor(gs14))
>        (line air time, lpattern(solid) lcolor(gs0) lwidth(*1.5)), `yl´ `xl´
>         ylabel(minmax 200(100)500, labsize(*.7) angle(0))
>         xlabel(#12, grid format(%4.0f) labsize(*.7))
>         title("Optimal WAAO aspect ratio", size(*.75) margin(small))
>         ytitle("Airline passengers", size(*.8)) xtitle("")
>         note("Note: the shaded area captures the yearly range (min to max).")
>         plotregion(margin(zero))
>         legend(off) fysize(30) name(waao,replace) aspect(`=waao´)
. graph combine default waao, col(1)
```

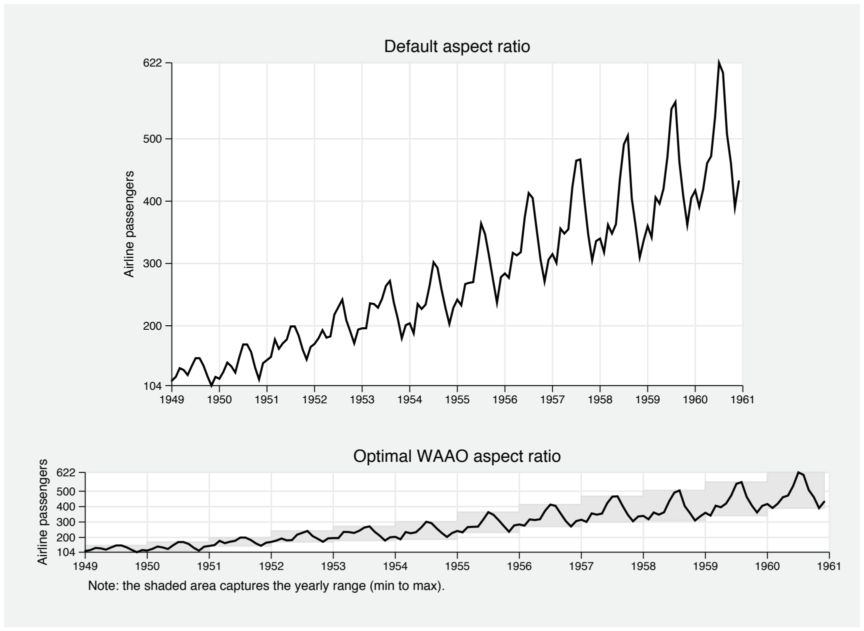


Figure 4. U.S. airline passengers (1949–1960)

The optimal aspect ratios proposed by the WAAO, ARC, and RV criteria are a weighted function of the length of each line segment, which enables them to satisfy the important parameterization invariance property. The dominant characteristic in these data is the equally steep rates of change from June to July and August to September, which generate the longest line segments in every year, particularly in the latest years. These long lines will carry a stronger influence in computation so the contrast of the steep rates of change is optimized the most. However, this also means that the contrast of the more gradual rates of change with the shorter lines bears lower weight in the calculation of the optimal aspect ratio. Thus WAAO, ARC, and RV suppress the contrast of an important feature of the data—the midseason drop in air travel demand from January to February and the even less noticeable drop from March to April.

To see a clearer contrast in the smaller rates of change and still maintain the useful contrast in the steeper slopes, apply the aspect ratio from the unweighted optimization criterion for the average orientation. Figure 5 applies the AAO aspect ratio, using the same graph code as above. This gives a clearly superior outcome:

```
. quietly optaspect air time, cull rank
. scalar aao = r(aao)
. twoway (rarea air_max air_min time, connect(stair) fcolor(gs14) lcolor(gs14))
>        (line air time, lpattern(solid) lcolor(gs0) lwidth(*2)), `yl´ `xl´
>         ylabel(minmax 200(100)500, labsize(*.9) angle(0))
>         xlabel(#12, grid format(%4.0f) labsize(*.9))
>         title("Optimal AAO aspect ratio", margin(small))
>         ytitle("Airline passengers", ) xtitle("")
>         plotregion(margin(zero))
>         note("Note: the shaded area captures the yearly range (min to max).",
>         size(*1.25)) legend(off) ysize(1) xsize(3) name(waao,replace)
>         aspect(`=aao´)
```

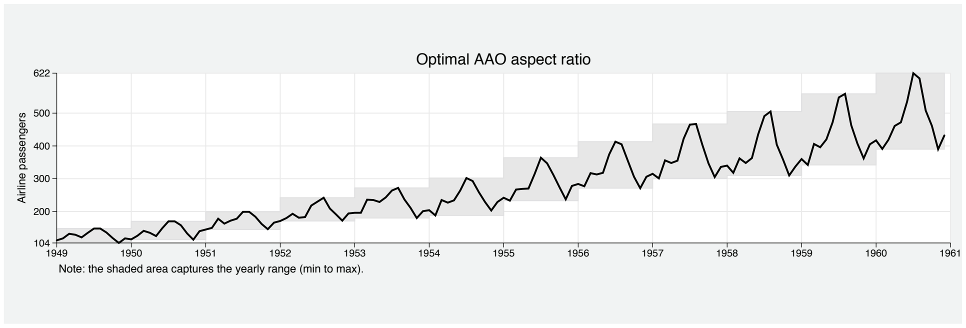


Figure 5. U.S. airline passengers (1949–1960)

4.3 Carbon dioxide concentration

The Keeling curve plots the atmospheric carbon dioxide (CO_2) concentrations using measurements obtained from the Mauna Loa Observatory in Hawaii since 1958, originally set by Charles David Keeling. These data are among the first clear evidence of increased greenhouse gas in the atmosphere. Import `co2.dta`, which holds monthly averages of CO_2 concentrations, and execute `optaspect`:[7]

```
. use co2, clear
. optaspect co2 date, cullzero
```

```
        Aspect ratio criterion                     aspect(#)

 Median Absolute Slope                               0.1219
    Compare to Average Absolute Slope                0.1232
 Weighted Average Absolute Orientation               0.1239
    Compare to Average Absolute Orientation          0.1433
 Arc Length based                                    0.8870
 Resultant Vector                                    0.1232

 Note: 2 zero slopes have been culled
```

The output suggests two diametrically opposed, but equally plausible aspect ratios. This may be because of dominant features appearing in different time frequencies. Indeed, the Keeling curve is famous for two reasons. First, it is famous for documenting the strong seasonality in CO_2 concentrations with gradual increase at the beginning of the season followed by a steep decrease. Second, it is famous for uncovering the exponential upward trend in CO_2 concentrations during the 20th century. The seasonal effect is in the higher frequency, and the trending quadratic effect is in the lower frequency. We should look at both displays, so store both the ARC and the RV aspect ratios in scalars:

```
. scalar rv = r(rv)
. scalar arc = r(arc)
```

7. The data are sourced from the Scripps Institution of Oceanography, University of California San Diego; see http://scrippsco2.ucsd.edu/. The `co2` variable holds the Scripps Institution of Oceanography's interpolated measurements in place of missing values (only a few instances).

First, plot the `co2` series using the RV aspect ratio to look at the higher-frequency seasonal variation:

```
. twoway (line co2 date, lwidth(*2)),
>         ytitle("CO{sub:2} (ppm)", size(*1.25)) xtitle("")
>         ylabel(310(20)410, grid angle(0))
>         xlabel(-22 674, format(%tmm-CY) labsize(*1.25))
>         graphregion(margin(r+5))
>         aspect(`=rv´) ysize(1) xsize(4)
```

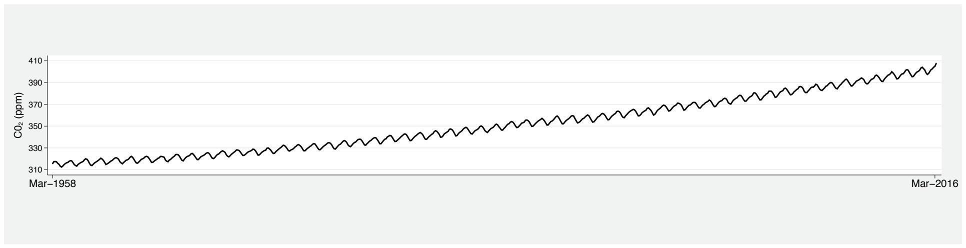


Figure 6. Monthly CO$_2$ atmospheric concentration

Figure 6 shows the recurring pattern in seasonality, beginning with a gradual increase and followed by a steep decay in CO$_2$ concentrations every year. There is no need to split and stack the series because there is no evident individual yearly variation, and the seasonality pattern appears to repeat itself consistently. This is why I chose not to label more years in the x axis and show only the beginning and end of the time period. However, figure 6 fails to clearly identify the exponential growth. To shift the focus to the trend component in the lower frequency, we need to employ a much larger aspect ratio, for example, the ARC aspect ratio of 0.8870. Simply substitute for `aspectratio('=arc')` in the graph code above.

Alternatively, a more guaranteed approach sure to capture the lower-frequency pattern is to estimate a locally weighted regression using the `lowess` command, then bank the smoothed line to 45° (Cleveland 1993a):

```
. lowess co2 date, gen(co2_sm)

. optaspect co2_sm date, sort nogor

  ┌────────────────────────────────────────────────┬──────────────┐
  │         Aspect ratio criterion                 │   aspect(#)  │
  ├────────────────────────────────────────────────┼──────────────┤
  │ Median Absolute Slope                          │     0.9959   │
  │    Compare to Average Absolute Slope           │     1.0000   │
  │ Weighted Average Absolute Orientation          │     1.0002   │
  │    Compare to Average Absolute Orientation     │     1.0321   │
  │ Arc Length based                               │     7.9391   │
  │ Resultant Vector                               │     1.0000   │
  └────────────────────────────────────────────────┴──────────────┘

. scalar waao = r(waao)
```

The WAAO and RV criteria suggest the typical orientation of the smoothed line is about 45°, and the AAO criterion explains there is a strong balance in line lengths, which means there is no single dominant rate of change, as in a structural shift. Thus, as shown in figure 7, which applies the WAAO aspect ratio, the inflection in the curvature documented by the Keeling curve follows by an upward shift in seasons. In other words, the CO_2 concentrations have uniformly increased in all months over the years. Plot the series using the WAAO aspect ratio, and superimpose a linear fit to enable the contrast with the quadratic trend in the data:

```
. twoway (lfit co2 date, lcolor(gs0) lwidth(*1.25))
>        (line co2 date, lcolor(gs8) lpattern(solid)),
>         ytitle("C0{sub:2} (ppm)") xtitle("")
>         ylabel(310(20)410, grid angle(0) labsize(*.85))
>         xlabel(-22 674, format(%tmm-CY) labsize(*.85))
>         graphregion(margin(r+5)) plotregion(margin(zero))
>         legend(ring(0) position(5) col(1) symxsize(*.4) bmargin(medium)
>                order(2 1) label(1 "Linear fit") label(2 "Observed")
>                size(*.8) region(fcolor(none) lcolor(none)))
>         aspect(`=waao´) ysize(1) xsize(1)
```

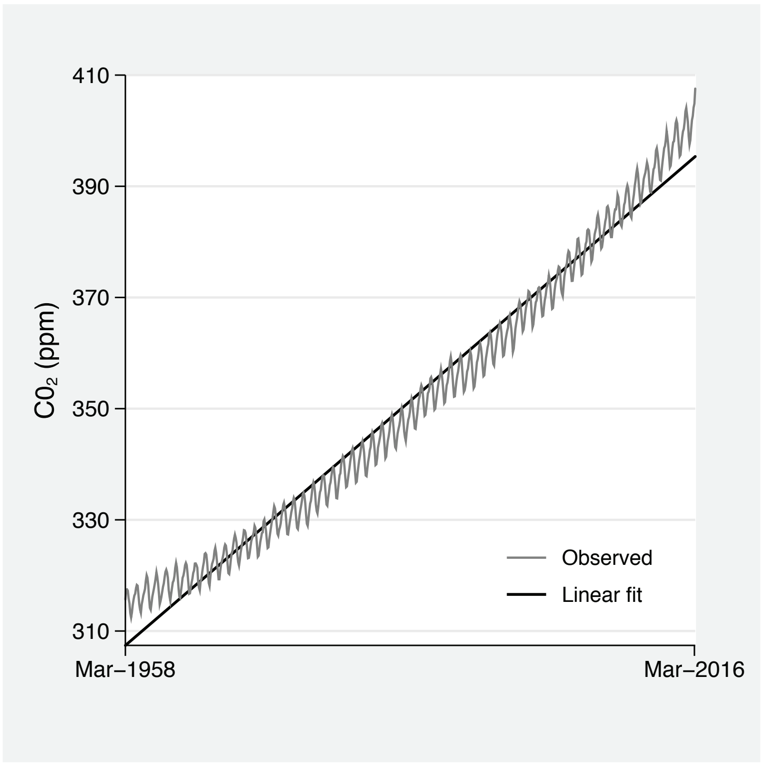


Figure 7. Monthly CO_2 atmospheric concentration

Given the regular pattern of seasonality, it suffices to show only the last year's data to get a closer view of how the series moves within a year. Focusing on the year 2015 will also showcase how CO_2 concentrations have broken the threshold of 400 parts per million for the first time in recorded history.[8] Figure 8 emphasizes the significance of this event by applying the RV aspect ratio, adding a reference line at 400 parts per million and using the `connected` line plot to assist the identification of each month:

```
. optaspect co2 date if year(dofm(date))==2015, sort

        Aspect ratio criterion                       aspect(#)

 Median Absolute Slope                                0.4800
    Compare to Average Absolute Slope                 0.4417
 Weighted Average Absolute Orientation                0.4425
    Compare to Average Absolute Orientation           0.4865
 Arc Length based                                     0.7324
 Resultant Vector                                     0.4417

. scalar rv = r(rv)
. local month_list
. local c 0
. foreach i in `c(Mons)´ {
  2.    local ++c
  3.    local month_list `month_list´ `c´ "`i´"
  4. }
. twoway (connected co2 date if year(dofm(date))==2015),
>         yline(400, lpattern(dash) lcolor(gs0))
>         ytitle("CO{sub:2} concentration (ppm)") xtitle("")
>         ylabel(minmax 399/403, format(%5.1f) grid angle(0) labsize(*.85))
>         xlabel(#12, format(%tmMon) labsize(*.85))
>         aspect(`=rv´) ysize(3) xsize(5)
```

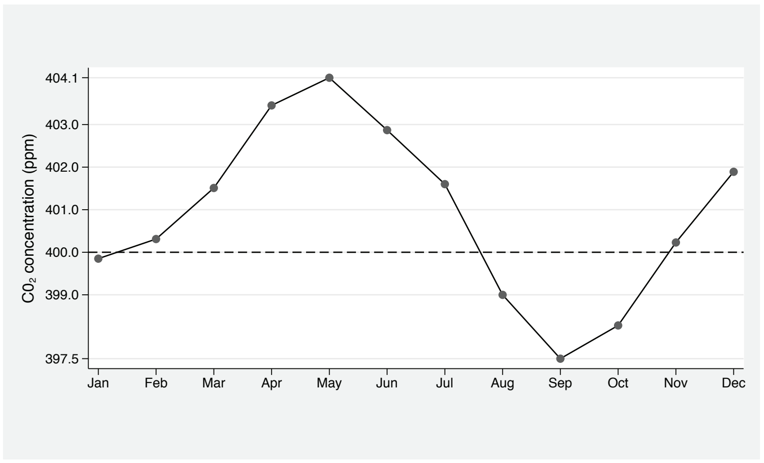


Figure 8. Monthly CO_2 atmospheric concentration during 2015

8. See http://climate.nasa.gov/400ppmquotes/ for some frightening quotes on what this means.

4.4 S&P 500 daily return

Line charts are a staple of financial time-series analysis, typically displaying the evolution in price or change in price (the return) against some time frequency. When time is measured in terms of days or some intraday frequency (for example, minutes, hours), Stata translates the frequency values as the days or minutes or hours that have lapsed since the 1st of January 1960.[9] The system dataset `sp500.dta` holds the S&P 500 daily price movement for 2001. Because U.S. capital markets operate about 255 days per year, there will be a seeming discontinuity between Friday afternoon and Monday morning plus gaps due to public holidays. The effect of interrupted calendar date and time is exacerbated in this dataset because of the 9/11 attack on the World Trade Center, which shut down Wall Street for a week.

`optaspect` ignores the calendar date and time gaps and calculates Δy_i and Δx_i without any missing values. However, because of Stata's translation of days into lapsed days, Δx_i will be overestimated during weekends and public holidays, for example, $\Delta x_i = 3$ for a regular weekend. This has a material effect on the calculation of the optimal aspect ratio. `optaspect` has a provision for dealing with this issue via the `rank` option. By specifying this option, you instruct `optaspect` to ignore calendar date and time discontinuities and replace the x-axis variable with a new variable that takes the values $t = 1, 2, \ldots, n$, thus effectively setting $\sum_{i=1}^{n} \Delta x_i = n$.

Execute `optaspect` on the abnormal market return, that is, the daily return above the yearly market average, and remember to specify the `rank` option:

```
. sysuse sp500, clear
(S&P 500)

. generate ret = change[_n]/close[_n-1]
(1 missing value generated)

. quietly summarize ret

. generate abn_ret = ret - r(mean)
(1 missing value generated)

. generate t = 0

. optaspect abn_ret t, rank

        Aspect ratio criterion                        aspect(#)

  Median Absolute Slope                                  0.0321
     Compare to Average Absolute Slope                   0.0280
  Weighted Average Absolute Orientation                  0.0282
     Compare to Average Absolute Orientation             0.0380
  Arc Length based                                      67.9707
  Resultant Vector                                       0.0280
```

The ARC aspect ratio fails to produce a meaningful result, and the WAAO and RV criteria suggest aspect ratios that are too low. This is because the abnormal return in efficient markets oscillates wildly about zero with a large random noise component.

9. That is, unless you change Stata's datum using business calendars; see `help datetime business calendars`.

Guha and Cleveland (2011) demonstrate that in the presence of such white noise, if $n \to \infty$, then $\gamma \to 0$ with probability. The only remedy in this case is to split and stack the series in smaller periods and apply an average aspect ratio across the stacked panels, which can be calculated via the `stackby()` option. This option requires access to a categorical variable that will be used to display the split-line plot. For instance, consider presenting the yearly series by calendar quarter:

```
. generate quarter = quarter(date)
. optaspect abn_ret t, rank stackby(quarter)

          Aspect ratio criterion                 aspect(#)

 Median Absolute Slope                              0.1328
    Compare to Average Absolute Slope               0.1133
 Weighted Average Absolute Orientation              0.1142
    Compare to Average Absolute Orientation         0.1520
 Arc Length based                                 274.5240
 Resultant Vector                                   0.1133

. scalar rv = r(rv)
```

The aspect ratios are now more reasonable, and we can proceed with presenting the split-and-stack time series accordingly, as shown in figure 9:

```
. levelsof quarter, local(qlevels)
1 2 3 4
. foreach i of local qlevels {
  2.   twoway (area abn_ret date if quarter==`i´), aspect(`=rv´)
>             ytitle("") ylabel(-0.05 0 0.05, grid angle(0) labsize(*.85))
>             xtitle("") xlabel(, labsize(*.85) format(%tddd/nn/CCYY))
>             fxsize(100) name(g`i´,replace)
  3. }
. graph combine g1 g2 g3 g4, col(1) graphregion(margin(l=15 r=10))
>       l1title("Abnormal market return in 2001", margin(zero) size(*.85))
```

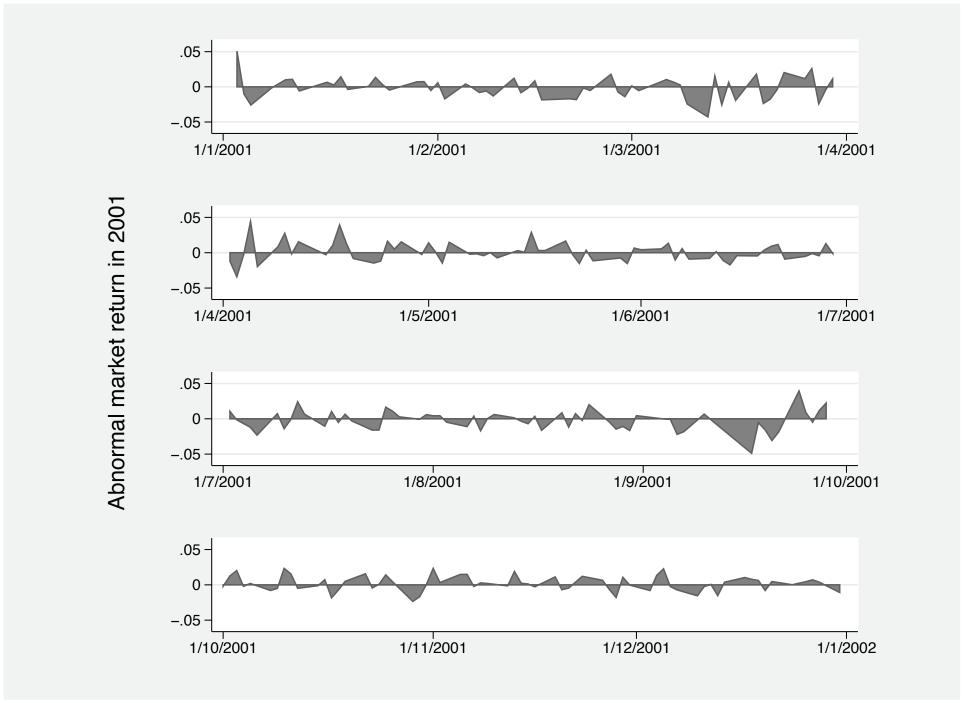


Figure 9. S&P 500 daily return during 2001

4.5 Excess rainfall in Brisbane

Brisbane has been in the news in recent years for supposedly extraordinarily high levels of rainfall, resulting in catastrophic floods around the Brisbane river basin. To investigate this claim, I calculate the excessive monthly rainfall in Brisbane and then the optimal aspect ratio for plotting excess rainfall since 1974 (the year when Brisbane city was hit by the largest flood in the 20th century). Use `rain.dta`, which holds monthly rainfall levels in the Sydney central business district and Brisbane central business district from 1915 to 2014:[10]

```
. use rain, clear
(Monthly rainfall in Brisbane)
. egen month_year = concat(month year), punct("/")
. generate date =  monthly(month_year,"MY")
. format %tmn/CY date
. drop month_year
. by month, sort: egen bri_exc = median(brisbane)
```

10. The data are sourced from the Australian Government Bureau of Meteorology; see http://www.bom.gov.au/climate/data/.

```
. replace bri_exc = brisbane - bri_exc
(1,199 real changes made, 1 to missing)
. optaspect bri_exc date if year>=1973, sort

        Aspect ratio criterion                     aspect(#)

  Median Absolute Slope                              0.0396
     Compare to Average Absolute Slope               0.0228
  Weighted Average Absolute Orientation              0.0229
     Compare to Average Absolute Orientation         0.0434
  Arc Length based                                   0.0128
  Resultant Vector                                   0.0228
```

The aspect ratios are low because of the long oscillating series about zero. The research question at hand is the frequency of excessive rainfall. Given that particularly heavy rainfall is a highly seasonal phenomenon, it would be more appropriate to calculate the aspect ratios for the maximum rainfall per year, which should appear only in one month per year and no more than two months—if the rain falls at the end of the calendar month:

```
. by year, sort: egen max_bri_exc = max(bri_exc)
. egen tag = tag(year)
. optaspect max_bri_exc year if year>=1973 & tag, sort

        Aspect ratio criterion                     aspect(#)

  Median Absolute Slope                              0.2659
     Compare to Average Absolute Slope               0.1295
  Weighted Average Absolute Orientation              0.1302
     Compare to Average Absolute Orientation         0.2279
  Arc Length based                                   0.0070
  Resultant Vector                                   0.1295

. scalar waao = r(waao)
```

Apply the WAAO aspect ratio in an `area` plot, which behaves exactly in the same way as a `line` plot with the difference that it fills the area within with color (the default is to fill in the area to the baseline of zero):

```
. twoway (area bri_exc date if year>=1974, lcolor(gs0) fcolor(gs12)),
>       xtitle("") ytitle("Excess monthly rainfall")
>       ylabel(-125 0(200)600 750, angle(0))
>       xlabel(168(12)648, format(%tmCY) angle(90))
>       plotregion(margin(b=0 t=00))
>       aspect(`=waao´) ysize(1) xsize(3.5)
```

In figure 10, the WAAO aspect ratio of 0.1302 stretches out the line plot without losing any detail. It is evident that the cycle of heavy rains experienced during 2009–2013 is no worse than the cycles of 1980–1984 and 1988–1992 followed by the massive downpour in 1996. What is indeed abnormal are the suppressed rain levels during 2000–2007 with damaging drought effects. Thus the heavy rains and resulting floods, especially of 2011, should not be surprising at all. What has been surprising is the mismanagement of information and lack of preparation.

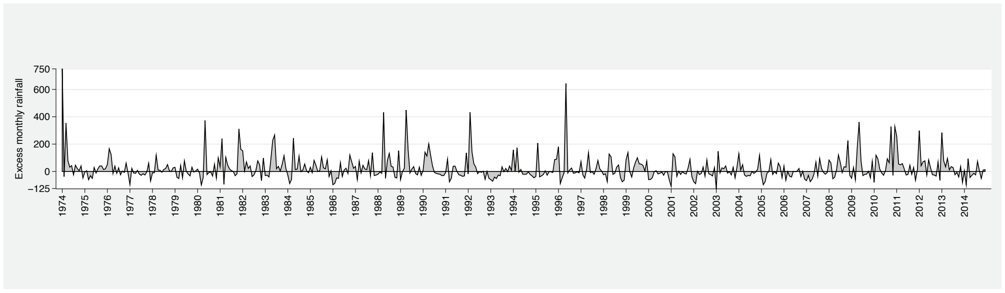


Figure 10. Excess monthly rainfall in Brisbane

5 Life expectancy

In the manual entry on `graph twoway line`, StataCorp offers some ill-fated advice on how to specify "an informative and visually pleasing graph" (StataCorp 2015, 281). Figure 11 reproduces the graph on offer, recasted here in `scheme(sj)` with some minor adjustments in line colors. The graph code extends about half a page in the manual, yet it fails to control for perhaps the most important option in a line chart—the aspect ratio. The visual is also plagued with excessive textual detail in the graph region, which reduces the effective space from the all-important plot region to a mere 30% of the overall image.[11] Specifically, we have a large grand title spanning over two lines, double x axes with labels in large font, a redundant x-axis title, a four-row legend at three o'clock, and a two-line note. When the user fails to control for the aspect ratio, the presence of all of these graph region options poorly determines the effective aspect ratio. The most important pitfall of this visual is the attempt to display multiple line plots that are measured at different scales. Thus the aspect ratio is further skewed and makes the most important line in the graph, that is, the difference in life expectancy, appear uninterestingly flat.

11. Just take a ruler and measure the physical dimensions of the plot region over the graph region.

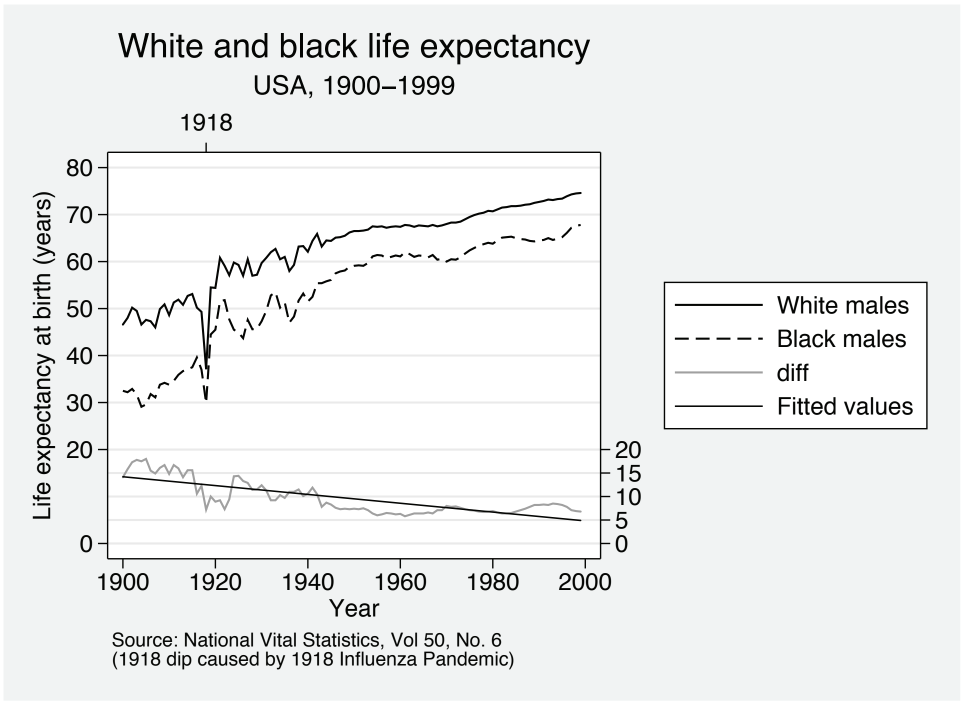


Figure 11. Poor line plot example in Stata 14 [G-2] **graph twoway line** manual

The best way to graph this information is in two separate graphs: one for the two series measured in years of life expectancy and another for the difference in life expectancy. Each graph should have its own aspect ratio, and we should limit the interference from nondata-related graph elements. Also, to make the analysis more relevant here, I update the dataset with the latest observations from the *2013 National Vital Statistics Reports* (Xu et al. 2016).[12]

12. The additional data are sourced from http://www.cdc.gov/nchs/data/nvsr/nvsr64/nvsr64_02.pdf.

```
. use http://www.stata-press.com/data/r14/uslifeexp, clear
(U.S. life expectancy, 1900-1999)
. input
          year         le    le_male  le_female       le_w   le_wmale
> le_wfem~e       le_b   le    _bmale  le_bfem~e
101. 2000 76.8 74.1 79.3 77.3 74.7 79.9 71.8 68.2 75.1
102. 2001 77.0 74.3 79.5 77.5 74.9 80.0 72.0 68.5 75.3
103. 2002 77.0 74.4 79.6 77.5 74.9 80.1 72.2 68.7 75.4
104. 2003 77.2 74.5 79.7 77.7 75.1 80.2 72.4 68.9 75.7
105. 2004 77.6 75.0 80.1 78.1 75.5 80.5 72.9 69.4 76.1
106. 2005 77.6 75.0 80.1 78.0 75.5 80.5 73.0 69.5 76.2
107. 2006 77.8 75.2 80.3 78.3 75.8 80.7 73.4 69.9 76.7
108. 2007 78.1 75.5 80.6 78.5 76.0 80.9 73.8 70.3 77.0
109. 2008 78.2 75.6 80.6 78.5 76.1 80.9 74.3 70.9 77.3
110. 2009 78.5 76.0 80.9 78.8 76.4 81.2 74.7 71.4 77.7
111. 2010 78.7 76.2 81.0 78.9 76.5 81.3 75.1 71.8 78.0
112. 2011 78.7 76.3 81.1 79.0 76.6 81.3 75.3 72.2 78.2
113. 2012 78.8 76.4 81.2 79.1 76.7 81.4 75.5 72.3 78.4
114. 2013 78.8 76.4 81.2 79.1 76.7 81.4 75.5 72.3 78.4
115. end
. optaspect le year, sort cullzero

         Aspect ratio criterion                      aspect(#)

  Median Absolute Slope                                0.8783
     Compare to Average Absolute Slope                 0.2925
  Weighted Average Absolute Orientation                0.2931
     Compare to Average Absolute Orientation           0.6870
  Arc Length based                                     0.8271
  Resultant Vector                                     0.2925

 Note: 12 zero slopes have been culled
. scalar rv = r(rv)
. twoway (line le_wmale year, lwidth(*1.5))
>        (line le_bmale year, lwidth(*1.5)),
>         ytitle("Life expectancy") xtitle("")
>         ylabel(minmax 35(5)70, labsize(*.85) angle(0))
>         xlabel(1900(10)2000 2013)
>         legend(ring(0) position(5) col(1) symxsize(*.6) bmargin(medlarge)
>                label(1 "White males") label(2 "Black males")
>                region(fcolor(none) lcolor(none)))
>         aspect(`=rv´) ysize(1) xsize(2.5) name(g1,replace)
```

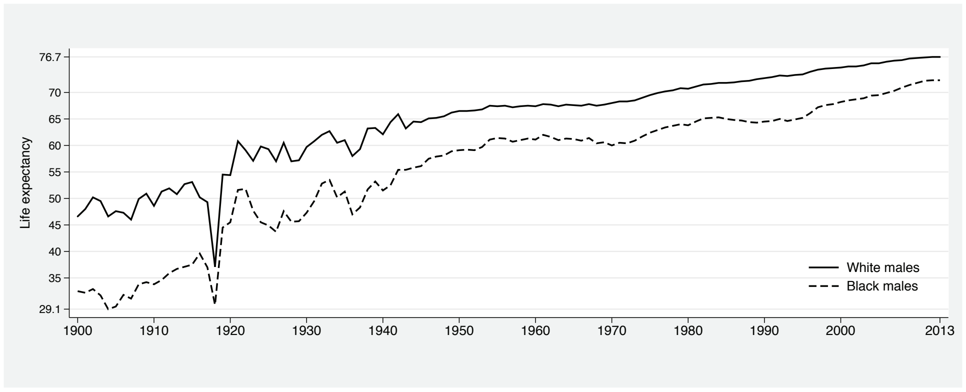


Figure 12. Life expectancy of white and black males in the United States

Figure 12 describes a steadily increasing trend in the life expectancy of males and gives the impression of an enduring constant gap between the life expectancy of white males and black males. It is well known that we are exceptionally bad at perceiving differences between two trending lines, and we form biased judgments in measuring difference as the closest distance between the two lines (for example, Cleveland [1993a]). Figure 12 is not instructive at all, even misleading, in describing the difference between life expectancies. For example, in figure 12, we are fooled to believe that the difference in life expectancy in 2013 is at the same level as it was in 1955.

To accurately judge the difference, we need a separate plot that would account for the much smaller scale differences in years and bank that series to 45°. Moreover, the benchmark of zero is a key piece of information when plotting differences in life expectancy, but including zero in the y axis elongates the plot region upward. `optaspect` has a provision to allow for the inclusion of $y = 0$ via the `y0` option and adjusts the calculation of the optimal aspect ratio accordingly:

```
. generate diff = le_wmale - le_bmale
. lowess diff year, generate(diff_sm)
. optaspect diff_sm year, y0
```

```
          Aspect ratio criterion                    aspect(#)

  Median Absolute Slope                               1.6959
     Compare to Average Absolute Slope                1.4203
  Weighted Average Absolute Orientation               1.4295
     Compare to Average Absolute Orientation          1.8345
  Arc Length based                                    9.1979
  Resultant Vector                                    1.4203
```

```
. scalar rv = r(rv)
. twoway (line diff year,  lcolor(gs0) lwidth(*1.5))
>        (line diff_sm year, lcolor(gs8) lwidth(*1.5) lpattern(solid)),
>         ytitle("Difference in life expectancy (number of years)") xtitle("")
>         ylabel(0(2)18, grid labsize(*.65) angle(0))
>         xlabel(1900(20)2000 2013, grid labsize(*.75))
>         legend(off) plotregion(margin(b=0 l=0 r=0 t=2))
>         aspect(`=rv´) ysize(6) xsize(5) name(g2,replace)
```

Figure 13 gives a clearer picture of the evolution of difference in life expectancy between white males and black males. Setting aside the 1918 shock from the Spanish influenza pandemic, the difference in life expectancy has been steadily decreasing from the beginning of the 20th century to the start of the Second World War, at which point it seems to settle to a range of about six to eight years difference until 2000 before following a sharp decline to four years' difference. The gap appears to be closing, but there is still more work that needs to be done.

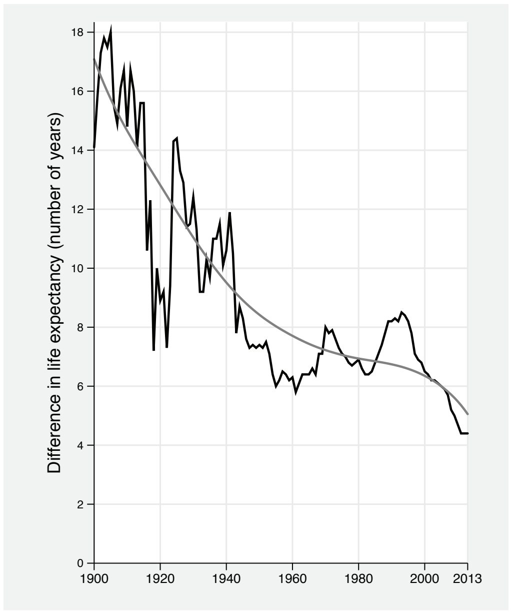


Figure 13. Difference in life expectancy between white and black males in the United States

6 Scatterplots and unordered data

`optaspect` can also be used to calculate the optimal aspect ratio for a scatterplot of two unordered variables. The only additional step in doing so is first obtaining a locally weighted regression estimate with the `lowess` command, then banking the smoothed line to 45°. For example, consider the system `auto.dta` and the scatterplot of the log of price of cars versus their log of mileage:

```
. sysuse auto, clear
(1978 Automobile Data)
. generate ln_p = ln(price)
. generate ln_m = ln(mpg)
. lowess ln_p ln_m, generate(ln_p_sm)
. optaspect ln_p_sm ln_m, sort
```

Aspect ratio criterion	aspect(#)
Median Absolute Slope	1.8003
Compare to Average Absolute Slope	1.1526
Weighted Average Absolute Orientation	1.0324
Compare to Average Absolute Orientation	1.8344
Arc Length based	1.1082
Resultant Vector	1.0207

```
. scalar rv = r(rv)
. twoway (scatter ln_p ln_m) (line ln_p_sm ln_m, sort(ln_m)), aspect(`=rv´)
  (output omitted)
```

7 Fixed aspect ratios

There are cases when the choice of aspect ratio is independent of the data. For example, diagnostic plots that contrast two same-scale distributions or cumulative distributions should always preserve an aspect ratio of 1. These include the quantile–quantile plot (`qqplot`), the quantile normal plot (`qnorm`), the standardized normal probability plot (`pnorm`), the quantile chi-squared plot (`qchi`), and the chi-squared probability plot (`pchi`). In these cases, `aspectratio(1)` effectively sets the diagonal line at 45° and acts as a reference line for enabling immediate and accurate comparison across the two distributions. However, this works only if both axes maintain the same scale range.

Similarly for when displaying functional forms with `twoway function` or constructing immediate graphs with `twoway scatteri`, `twoway pcarrowi`, or `twoway pci`, theoretical forms have well-predefined geometric shapes whose direction across the 2D plane should determine the aspect ratio. For example, a circle or the quarter of a circle should be displayed with `aspectratio(1)`, but a semicircle should be displayed with `aspectratio(0.5)` or `aspectratio(2)` depending on its orientation. When graphing the line of an algebraic formula, take two coordinates from that line and bank its absolute orientation $\arctan\{|(y_2 - y_1)/(x_2 - x_1)|\}$ to 45°. When graphing an algebraic curve, you may calculate the aspect ratio by banking its resultant vector to 1.

8 Summary

`optaspect` calculates the optimal aspect ratio for a two-variable line graph, using a number of heuristic criteria based on the principle of maximizing the contrast between the many rates of change. I demonstrated `optaspect` using a variety of datasets, with emphasis on the importance of exercising judgment when selecting aspect ratios regardless of the science underpinning the heuristic criteria.

9 Acknowledgments

I acknowledge the useful correspondence with Justin Talbot (Tableau Software) and the pointers made by Christophe Kolodziejczyk on a Statalist post. I also acknowledge the useful comments by an anonymous reviewer.

10 References

Bertin, J. 1983. *Semiology of Graphics: Diagrams, Networks, Maps*. Madison: University of Wisconsin Press.

Cleveland, W. S. 1993a. *Visualizing Data*. Summit, NJ: Hobart.

———. 1993b. A model for studying display methods of statistical graphics. *Journal of Computational and Graphical Statistics* 2: 323–343.

Cleveland, W. S., M. E. McGill, and R. McGill. 1988. The shape parameter of a two-variable graph. *Journal of the American Statistical Association* 83: 289–300.

Cox, N. J. 2004. Stata tip 12: Tuning the plot region aspect ratio. *Stata Journal* 4: 357–358.

Fisher, G. H. 1974. An experimental study of linear inclination. *Quarterly Journal of Experimental Psychology* 26: 52–62.

Fisher, R. A. 1925. *Statistical Methods for Research Workers*. Edinburgh: Oliver & Boyd.

Guha, S., and W. S. Cleveland. 2011. The disappearing second derivative of quadratics: Perceptual, mathematical, and statistical properties of judging dependence on visual displays. Technical report, Department of Statistics, Purdue University. http://fodava.gatech.edu/files/uploaded/DLS/Cleveland.pdf.

Han, F., Y. Wang, J. Zhang, O. Deussen, and B. Chen. 2016. Mathematical foundations of arc length-based aspect ratio selection. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, ed. C. Hansen, I. Viola, and X. Yuan, 9–15. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Heer, J., and M. Agrawala. 2006. Multi-scale banking to 45 degrees. *IEEE Transactions on Visualization and Computer Graphics* 12: 701–708.

StataCorp. 2015. *Stata 14 Graphics Reference Manual*. College Station, TX: Stata Press.

Talbot, J., J. Gerth, and P. Hanrahan. 2011. Arc length-based aspect ratio selection. https://research.tableau.com/sites/default/files/infovis2011-talbot-arclengthbanking.pdf.

———. 2012. An empirical model of slope ratio comparisons. http://www.justintalbot.com/wp-content/uploads/2013/02/banking2_corrected1.pdf.

Xu, J., S. L. Murphy, K. D. Kochanek, and B. A. Bastian. 2016. Deaths: Final data for 2013. *National Vital Statistics Reports* 64: 1–119.

About the author

Demetris Christodoulou is at the University of Sydney Business School and is the General Convenor of the research network Methodological and Empirical Advances in Financial Analysis.

The Stata Journal (2017)
17, Number 2, pp. 314–329

Regression clustering for panel-data models with fixed effects

Demetris Christodoulou
University of Sydney
Sydney, Australia
demetris.christodoulou@sydney.edu.au

Vasilis Sarafidis
Monash University
Melbourne, Australia
Vasilis.Sarafidis@monash.edu

Abstract. In this article, we describe the `xtregcluster` command, which implements the panel regression clustering approach developed by Sarafidis and Weber (2015, *Oxford Bulletin of Economics and Statistics* 77: 274–296). The method classifies individuals into clusters, so that within each cluster, the slope parameters are homogeneous and all intracluster heterogeneity is due to the standard two-way error-components structure. Because the clusters are heterogeneous, they do not share common parameters. The number of clusters and the optimal partition are determined by the clustering solution, which minimizes the total residual sum of squares of the model subject to a penalty function that strictly increases in the number of clusters. The method is available for linear short panel-data models and useful for exploring heterogeneity in the slope parameters when there is no a priori knowledge about parameter structures. It is also useful for empirically evaluating whether any normative classifications are justifiable from a statistical point of view.

Keywords: st0475, xtregcluster, panel data, parameter heterogeneity

1 Introduction

Standard panel-data analysis imposes the restriction that all individuals share the same slope coefficients, and any unobserved heterogeneity across individuals is attributed solely to the presence of individual-specific, time-invariant effects (that is, differential intercepts). This restriction can be difficult to justify, both theoretically and empirically (for example, Burnside [1996], Baltagi and Griffin [1997], Pesaran, Shin, and Smith [1999]).

The `xtregcluster` command implements the regression clustering approach developed by Sarafidis and Weber (2015), which groups individuals into distinct clusters. Within each cluster, the slope coefficients are homogeneous, and all intracluster heterogeneity is attributed to the two-way error-components structure. The clusters themselves are heterogeneous; that is, the slope coefficients are different across clusters.

Both the number of clusters and the optimal partition are treated as unknown and are determined from the data based on minimizing a model information criterion that is strongly consistent. That is, it estimates the true number of clusters with probability one as N grows for any T fixed. Therefore, the method is valid for panel datasets characterized by a large number of individuals and a short time-series length.

`xtregcluster` is useful for exploring slope parameter heterogeneity in the absence of a priori information regarding parameter structures. The algorithm can also confirm whether the restriction of slope parameter homogeneity is supported by the data. Moreover, it is useful for examining whether a priori classification of individual entities is optimal from a statistical point of view, such as industrial classifications (for example, North American Industry Classification System codes), risk classifications (for example, credit ratings), or arbitrarily imposed classification schemes (for example, univariate quantile classes).

2 A partially heterogeneous panel-data model

Consider the linear panel-data model

$$y_{i_\omega t} = \boldsymbol{\beta}'_\omega \mathbf{x}_{i_\omega t} + \epsilon_{i_\omega t} \tag{1}$$

where $y_{i_\omega t}$ denotes the observation of the dependent variable for the ith individual in cluster ω at time period t, $\mathbf{x}_{i_\omega t}$ is a $K \times 1$ vector of covariates, and $\boldsymbol{\beta}_\omega$ is a $K \times 1$ vector of fixed parameters that are common within clusters but vary across clusters. The default error term of the model is composite and subject to a one-way error-component structure:

$$\epsilon_{i_\omega t} = \eta_{i_\omega} + e_{i_\omega t} \tag{2}$$

The regressors are assumed to be exogenous with respect to the idiosyncratic error component, $e_{i_\omega t}$, but they can be endogenous with respect to η_{i_ω}. The vector, $\mathbf{x}_{i_\omega t}$, can include dynamic terms if they are exogenous. A two-way error-components structure can be easily implemented by specifying the factorial notation `i.`*timevar*. Hence, the error term structure becomes $\epsilon_{i_\omega t} = \eta_{i_\omega} + \tau_t + e_{i_\omega t}$.

In Stata, the estimation of the model described by (1) and (2) corresponds to the `xtreg, fe` command, except each cluster has its own regression structure such that $\omega = 1, 2, \ldots, \Omega_0$ with $i_\omega = 1, 2, \ldots, N_{\omega_0}$ and $t = 1, 2, \ldots, T$. The total number of individuals across clusters is $N = \sum_{\omega=1}^{\Omega_0} N_\omega$, and the total sample size is $N \times T$.

Note that the whole time series of a given individual entity belongs to a cluster, not a subset of observations of individuals. That is, an individual can be classified only in one cluster. The focus is the analysis of "short panels", where $N \gg T$ with $N \to \infty$ and T fixed (for unbalanced panels, the average $\overline{T}$ is fixed).

The key estimation problem is how to obtain estimates of the model's slope coefficients when the value of the true number of clusters, Ω_0, and membership of individuals to clusters are both unknown. Sarafidis and Weber (2015) develop a partitional clustering approach for estimating the true number of clusters, as well as the corresponding partition (see also Kaufman and Rousseeuw [2005] and Everitt et al. [2011]). Sarafidis and Weber (2015) show through analytical work and simulations that their proposed solution yields a strongly consistent estimate of Ω_0; that is, $\widehat{\Omega} \to \Omega_0$ with prob $\to 1$ as $N \to \infty$, for any fixed T.

3 Estimation algorithm

The optimal value of Ω—denoted as $\widehat{\Omega}$—and the corresponding partition are determined based on the following algorithm:

1. Specify some value for Ω, where $\Omega \leq \xi \leq \Omega_0$, with $\xi > 1$ and $\xi \in \mathbb{Z}^+$.
2. Randomize the panel identifiers.
3. Obtain an initial partition using one of the following three ways: i) a random classification based on the standard uniform distribution; ii) an a priori classification; and iii) a classification based on certain observed variables (characteristics) and obtained using the official Stata command `cluster kmeans`.
4. Reallocate the first individual to all remaining clusters, each time saving the value of the residual sum of squares (RSS) that arises for each cluster, RSS_ω. Assign the individual into the cluster that corresponds to the smallest value of the total RSS across all ω; that is, $\text{RSS} = \sum_{\omega=1}^{\Omega} \text{RSS}_\omega$.
5. Repeat step 4 for every other individual in the sample.
6. Repeat steps 3 and 4 until RSS cannot be reduced any further with some tolerance criterion.
7. Repeat steps 1 to 6 for different values of Ω. $\widehat{\Omega}$ is the value of Ω that minimizes the following model information criterion (MIC),

$$\text{MIC} = N \ln \left(\frac{\text{RSS}}{N\overline{T}} \right) + \Omega\theta_N \tag{3}$$

   where $\overline{T} = 1/N \sum_{i=1}^{N} T_i$ denotes the average time-series length for unbalanced panels. For panels with equal-length time series, labeled by Stata as strongly or weakly balanced panels, $\overline{T} = T$. The term $\Omega\theta_N$ is a required penalty because the minimum RSS is monotone decreasing in the number of clusters and will tend to overparameterize the model by allowing for more clusters than may actually exist. Essentially, the penalty provides a filter to ensure that the preferred clustering outcome partitions between clusters rather than within clusters. θ_N can take any value, provided $\lim_{N\to\infty} N^{-1}\theta_N = 0$ and $\lim_{N\to\infty}[\log\{\log(N)\}]^{-1}\theta_N = \infty$. The theoretical properties of the algorithm are discussed in Sarafidis and Weber (2015, sec. 4).

The rationale behind step 2 is to make sure the results are not dependent on the original ordering of the individuals. This is important because, in real data, numerical panel identifiers with a smaller value are often associated with early "entrants", whereas larger values can be associated with late "entrants" in the dataset (for example, panel identifiers may be associated with the age of a company). To the extent that this feature implies dependencies among individuals, one should randomize the order within our algorithm according to which individuals are reallocated to different clusters.

The total number of possible partitions is exponential in N. Therefore, it becomes infeasible to search over all possible partitions, even for relatively small values of N and Ω. Thus one should use different initial partitions (that is, iterate steps 3–7) before $\widehat{\Omega}$ is determined to avoid local minimums; we provide examples in section 5.

4 The xtregcluster command

As with all `xt` commands, `xtregcluster` requires that the data be `xtset`. It begins by obtaining an initial partition through a random uniform classification, a predetermined classification, or a classification based on certain observed variables. Then, it reclassifies individuals to clusters so that total RSS is minimized.

4.1 Syntax

`xtregcluster` *depvar indepvars* [*if*] [*in*] [*weight*] `,`
{`random` | `preclass(`*varname*`)` | `prevars(`*varlist* | `X` | `b)`} `omega(`*numlist*`)`
[`prevarsopt(`*kmeansopt*`)` `theta(`#`)` `seed(`#`)` `name(`*varname*`)` `iterate(`#`)`
`tolerance(`#`)` `nolog` `graph` `table`]

4.2 Options

`random` obtains the initial partition using random selection from the standard uniform distribution. `random`, `preclass()`, or `prevars()` is required.

`preclass(`*varname*`)` obtains the initial partition based on a predetermined classification, using a categorical variable. `omega()` is not allowed with `preclass(`*varname*`)`, because the preclassification determines the size of Ω. `random`, `preclass()`, or `prevars()` is required.

`prevars(`*varlist* | `X` | `b)` obtains the initial partition based on observed variables, using the official Stata command `cluster kmeans`. The observed variables can be part of the explanatory variables but not necessarily so. `prevars(X)` includes the whole set of regressors and is equivalent to specifying `prevars(`*indepvars*`)`. You may also combine `prevars(X` *varlist*`)`. `prevars(b)` uses the individual-specific estimated slopes of all *indepvars*. `random`, `preclass()`, or `prevars()` is required.

`omega(`*numlist*`)` specifies the numerical range of $\Omega > 1$. `omega()` is required with `random` and `prevars()` but is not allowed with `preclass()`. `omega()` takes integer values.

`prevarsopt(`*kmeansopt*`)` can be specified only with `prevars(`*varlist*`)`. *kmeansopt* takes all options from `cluster kmeans` with the exceptions of `name()`, `generate()`, and `k()`, which are already specified by the `omega()` option above.

`theta(`#`)` specifies the value of θ_N in the penalty function of (3) for overfitting Ω. The default is `theta(`$1/3\ln(N)+2/3\sqrt{N}$`)`, which is found to perform well by Sarafidis and Weber (2015) in their simulation study. `theta()` can take any other real argument. Other common values for θ_N are $\ln(N)$ and $\sqrt{N}$.

`seed(`#`)` sets the random-number seed for the entire program. The seed is relevant for randomizing the numerical panel identifiers. The seed is also relevant for the `random` method of obtaining the initial partition. The default is `seed(123)`.

`name(`*varname*`)` specifies the name prefix for the newly generated variables that identify the levels in the optimized partitions for each Ω. The default is `name(omega`#`)`, where # is a positive integer as specified in `omega()`.

`iterate(`#`)` specifies the maximum number of iterations for minimizing the total RSS, given `omega()`. The default is `iterate(100)`.

`tolerance(`#`)` specifies the tolerance for the convergence of total RSS, given `omega()`. The default is `tolerance(1e-6)`.

`nolog` suppresses the RSS iteration log.

`graph` gives a visual diagnostic with cluster-specific scatterplots of the observed dependent variables against the linear predictor together with superimposed linear fits for each ω corresponding to $\widehat{\Omega}$.

`table` prints a table of estimates of the slope coefficients for each ω corresponding to $\widehat{\Omega}$. That is, it reports results using cluster-specific fixed-effects (FE) regressions based on $\widehat{\Omega}$, for example, `xtreg if omega4==1, fe` for $\omega = 1$ and similarly for the remaining clusters, that is, $\omega = 2, \dots, \widehat{\Omega}$.

4.3 Output and stored results

`xtregcluster` prints the method used to obtain the initial partition and its associated total RSS, labeled as `Iteration 0 Total RSS`. This RSS varies according to the size of Ω and how the initial partition was obtained. Next, the output prints the RSS at the end of every iteration up to convergence, followed by a report on the MIC for the specified value of Ω, contrasted with the MIC corresponding to $\Omega = 1$. The total RSS reported for $\Omega = 1$ is obtained from the pooled FE regression, and it is the same regardless of the choice of the initial partition or the value of Ω. The report ends with a recommendation of whether to proceed either with the pooled FE regression, `xtreg, fe`, or with cluster-specific FE regressions.

`xtregcluster` stores the following in `e()`:

Scalars	
`e(N)`	N panels in estimation
`e(T)`	balanced T or average $\overline{T}_i$
`e(NT)`	NT or $N\overline{T}_i$
`e(rss_pool)`	pooled RSS
`e(mic_pool)`	pooled MIC
`e(omega_opt)`	optimal $\widehat{\Omega}$ given specified range
`e(theta)`	specified θ_N
`e(rss_tot`$_\Omega$`)`	total RSS$_\Omega$ for every Ω
`e(mic_tot`$_\Omega$`)`	MIC$_\Omega$ for every Ω
Macros	
`e(cmdline)`	estimator
`e(name_opt)`	optimal partition variable
Matrices	
`e(rss`$_\Omega$`)`	total RSS$_\Omega$ at every iteration for every Ω

`xtregcluster` generates indicator variables with common name prefixes, as provided in `name()`, for every Ω specified in `omega()`. For instance, the `name(om)` and `omega(2/3)` options will generate three indicator variables with names `om2` and `om3`. The stored result, `e(name_opt)`, takes the name of one of these variables corresponding to the smallest MIC$_\Omega$, with the optimal value of $\widehat{\Omega}$ stored in `e(omega_opt)`. The command also returns all total RSS$_\Omega$ and MIC$_\Omega$ for every Ω specified in `omega()`, plus the iteration logs in matrix form, for example, `e(rss_tot2)`, `e(mic_tot2)`, `e(rss2)` and `e(rss_tot3)`, `e(mic_tot3)`, `e(rss3)`.

Stored results can be used for subsequent analysis. For example, alternative penalties θ_N for the calculation of the MIC may be specified either through the `theta()` option or manually calculated using stored results. The latter approach is recommended, given the computational cost in running `xtregcluster` again. For example, one may wish to check the sensitivity of the suggested optimal partition using the less severe penalty, $\ln(N)$, or the stricter penalty, $\sqrt{N}$:

```
. display e(N) * ln(e(rss_tot)/e(NT)) + e(omega_opt) * ln(e(N))
. display e(N) * ln(e(rss_tot)/e(NT)) + e(omega_opt) * sqrt(e(N))
```

Asymptotically, that is, for large N, the choice of θ_N is immaterial, although in small samples it can possibly give substantially different results.

The `table` option contrasts estimation results of the heterogeneous slopes for each $\omega = 1, 2, \ldots, \widehat{\Omega}$. This `table` does not report standard errors, because clustering of individuals based on minimizing RSS implies that the usual formula for obtaining standard errors is no longer valid. However, the next section explains how to reproduce this table and obtain bootstrapped standard errors that are valid for inference. We refrain from making bootstrapping a default treatment, because it is computationally intensive and requires a large number of repetitions to produce reliable estimates for standard errors.

The `graph` option produces a single graph with overlaid scatterplots and linear fits of all heterogeneous slopes across the clustered individuals for every $\omega = 1, 2, \ldots, \widehat{\Omega}$. The `graph` applies a colored scheme to readily assist the visual distinction among the clusters.

Section 5 demonstrates how to reproduce this graph in monochrome for publication purposes and also how to plot each cluster in a separate graph against the pooled slope, which is useful for large $\widehat{\Omega}$.

4.4 Practical considerations

`xtregcluster` is computationally intensive for large N and large Ω. Therefore, one should reduce the dimension of the data to the required minimum before executing `xtregcluster`, both in terms of columns (variables) and rows (observations). For example, if the model contains the variables y, x_1, and x_2, and the dataset in memory contains many more variables, then you should reduce as follows:

```
. preserve
. keep id date y x1 x2
. keep if !missing(panelvar,timevar,y,x1,x2)
. xtregcluster y x1 x2, random
. restore
```

`xtregcluster` is relevant for short panel data where individuals have at least two observations, that is, $T_i > 1$. If an individual has only $T_i = 1$, it cannot be classified meaningfully into a cluster using an FE regression. If your dataset contains such cases, `xtregcluster` will issue a warning that individuals with $T_i = 1$ are excluded from estimation.

The initial partition can be obtained using estimates of the individual-specific slope coefficients using `prevars(b)`. This method is feasible only for panel datasets where all individuals have a sufficient number of observations for estimating the individual-specific slopes; that is, $T_i \geq k + 1$. If there is even a single individual with $T_i < k + 1$, `xtregcluster` will issue the following error:

```
. xtregcluster y x1 x2, prevars(b)
Some panels have Ti < k+1. Choose an alternative initial partition,
or qualify the sample to panels with enough observations.
insufficient observations
r(2001);
```

If the user insists on using `prevars(b)` for obtaining the initial partition, the entire analysis must be restricted to individuals with enough observations:

```
. bysort id: generate Ti = _N
. xtregcluster y x1 x2 x3 x4 if Ti>=5, prevars(b)
```

This approach is not recommended for models with a large number of explanatory variables, because many individual units may be dropped—resulting in a great loss of degrees of freedom.

Because the clustering algorithm used by `xtregcluster` aims to minimize the within-cluster RSS, the properties of the estimated standard errors obtained using standard formulas are no longer known. Therefore, once $\widehat{\Omega}$ and the corresponding partition have been determined, we recommend computing standard errors using the method of bootstrapping, which provides estimates of the distribution one would get if one were able to draw repeated samples of N points from the unknown true distribution (Sarafidis and Weber 2015). Following the execution of `xtregcluster`, one can produce bootstrapped standard errors as follows:

```
. local optomega = e(omega_opt)
. local optname  = e(name_opt)
. quietly forvalues i = 1/`optomega´ {
2. xtreg y x1 x2 if `optname´==`i´, vce(bootstrap, reps(1000) nodots)
3. estimates store omega`i´
4. }
. estimates table omega*, se stats(N_g Tbar N r2_w rho corr)
```

We recommend the minimum of 1,000 repetitions, preferably even more, to obtain reliably precise estimates. The statistics in `stats()` report key diagnostics for `xtreg, fe` and are described in [XT] **`xtreg`**.

5 Application

To demonstrate the application of `xtregcluster`, we estimate a translog production function for Spanish dairy farms. `help xtregcluster` provides another application using Stata's `productivity.dta` on U.S. public capital productivity.

5.1 Dairy farm production

Consider a translog functional form for modeling Spanish dairy farm production output. `dairy.csv` is obtained from William Greene's webpage on panel-data econometrics and contains observations on output (cow milk) and several inputs, such as the number of cows used, size of land, labor, and feed.[1] The panel structure is balanced with all $N = 247$ farms observed over the same period of 1993–1998:

```
. import delimited dairy.csv
(28 vars, 1,482 obs)
. xtset farm year
       panel variable:  farm (strongly balanced)
        time variable:  year, 93 to 98
                delta:  1 unit
```

Note that the `xtregcluster` command is also useful for hierarchical datasets with repeated observations over higher-level cross-sectional units, absent of a time variable. For example, if this dataset contained repeated observations for many farms within counties, then it would suffice to `xtset` using only the lower-level panel identifier:

```
. xtset farm
       panel variable:  farm (balanced)
```

1. We thank William Greene for giving us permission to use the data, which are available online at http://people.stern.nyu.edu/wgreene/Econometrics/PanelDataSets.htm.

The variable `yit` denotes the log of the demeaned farm output, while variables $\mathtt{x}_k$ and $\mathtt{x}_{k\ell}$ are the regressors used in the translog function, where x_k for $k = 1, \dots, K$ denotes the log of the kth input, that is, the demeaned number of cows, land size, labor, and feed, whereas $\mathtt{x}_{k\ell} = \mathtt{x}_k \mathtt{x}_\ell$.

We start by obtaining the initial partition based on a uniform random classification for $\Omega = 2, 3$:

```
. xtregcluster yit x1-x34, random omega(2/3)
Initial partition via randomized classification and seed 123
Omega = 2
Iteration 0:   Total RSS =          7.654268
Iteration 1:   Total RSS =          6.492979
Iteration 2:   Total RSS =          6.353171
Iteration 3:   Total RSS =          6.316846
Iteration 4:   Total RSS =          6.316779
Iteration 5:   Total RSS =          6.316779
Omega = 3
Iteration 0:   Total RSS =          7.579115
Iteration 1:   Total RSS =          5.759485
Iteration 2:   Total RSS =          5.603187
Iteration 3:   Total RSS =          5.543064
Iteration 4:   Total RSS =          5.502915
Iteration 5:   Total RSS =          5.469064
Iteration 6:   Total RSS =          5.461379
Iteration 7:   Total RSS =          5.461363
Iteration 8:   Total RSS =          5.461215
Iteration 9:   Total RSS =          5.461215
```

Omega	Total RSS	MIC
1	7.887	-1280.962
2	6.317	-1323.483
3	5.461	-1347.117

```
Proceed with xtreg if omega3==`i´,fe  where `i´=1,2,3
```

The output suggests that there exist at least three distinct clusters for these data, given the model. Notice how two new variables have been created to indicate cluster membership of all individuals for the two values of Ω, with names `omega2` and `omega3`. The name `omega` is the default prefix indicating the size of Ω as specified in `omega()`.

Because the value of Ω corresponding to the minimum value of MIC equals the maximum value specified in `omega()`, it is necessary to explore larger values, for example, `omega(2/10)`. To conserve space, we suppress the iteration logs by specifying the `nolog` option. We may also specify a new name prefix:[2]

2. Because this may take awhile, the user may actually wish to see the iteration log working in action.

```
. xtregcluster yit x1-x34, random omega(2/10) name(om) nolog
Initial partition via randomized classification and seed 123

      Omega        Total RSS          MIC

          1            7.887    -1280.962
          2            6.317    -1323.483
          3            5.461    -1347.117
          4            4.999    -1356.638
          5            4.626    -1363.494
          6            4.483    -1358.955
          7            4.001    -1374.719
          8            3.820    -1373.825
          9            3.781    -1364.030
         10            3.580    -1365.253

Proceed with xtreg if om7==`i´,fe  where `i´=1,2,3,4,5,6,7
```

The output suggests the optimal value of Ω is 7. However, because the results are based on a particular initial partition—uniform random selection with seed 123—we recommend trying alternative initial partitions. As an example, the initial partition can be determined based on the regressors, $\mathbf{X}$, using the `prevars(X)` option:

```
. xtregcluster yit x1-x34, prevars(X) omega(2/10) name(omX) nolog
Initial partition via the variation in x1 x2 x3 x4 x11 x22 x33 x44 x12
> x13 x14 x23 x24 x34 and seed 123

      Omega        Total RSS          MIC

          1            7.887    -1280.962
          2            6.155    -1329.891
          3            5.658    -1338.374
          4            4.976    -1357.779
          5            4.684    -1360.402
          6            4.374    -1364.992
          7            4.132    -1366.781
          8            3.863    -1371.059
          9            3.805    -1362.498
         10            3.574    -1365.671

Proceed with xtreg if omX8==`i´,fe  where `i´=1,2,3,4,5,6,7,8
```

The results now indicate that the optimal value of Ω is 8. Because the value of MIC_8 under `prevars(X)` is larger (-1371.059) than the value of MIC_7 under `random` initial partition (-1374.719), we set $\widehat{\Omega} = 7$. This example shows how important it is to experiment between different initial partitions. It is also wise to try different random seed numbers, using the `seed()` option.

To see how many individuals are assigned in each cluster, `tabulate` the variable that holds the optimal partition, as estimated above using the random initial partition with $\widehat{\Omega} = 7$. Remember to qualify the sample only to one observation per individual:

```
. egen tag = tag(farm)
. tabulate `e(name_opt)´ if tag
       omX8 |      Freq.     Percent        Cum.
------------+-----------------------------------
          1 |         33       13.36       13.36
          2 |         29       11.74       25.10
          3 |         18        7.29       32.39
          4 |         28       11.34       43.72
          5 |         28       11.34       55.06
          6 |         24        9.72       64.78
          7 |         35       14.17       78.95
          8 |         52       21.05      100.00
------------+-----------------------------------
      Total |        247      100.00
```

The `table` and `graph` options provide additional information about the final model. These options can be entered from the outset when specifying the numerical list in `omega()`. Then, a `table` of estimates and a `graph` of scatterplots with linear fits are displayed for $\omega = 1, 2, \ldots, \widehat{\Omega}$. If the user forgets to enter these options, one can repeat `xtregcluster` only for $\widehat{\Omega}$:

```
. drop om7
. xtregcluster yit x1-x34, random omega(7) name(om) nolog table graph
Initial partition via randomized classification and seed 123

  |    Omega    |   Total RSS   |      MIC      |
  |-------------|---------------|---------------|
  |      1      |        7.887  |    -1280.962  |
  |      7      |        4.001  |    -1374.719  |

Proceed with xtreg if om7==`i´,fe  where `i´=1,2,3,4,5,6,7
Table: Panel data fixed effects estimates by omega
```

```
    Variable     om7_1       om7_2       om7_3       om7_4       om7_5

          x1     0.681       0.652       1.452       0.835       0.666
          x2    -0.130       0.117      -0.173       0.251       0.413
          x3    -0.416      -0.057       0.071       0.025       0.745
          x4     0.359       0.219       0.197       0.096       0.397
         x11    -0.406      -0.082       2.267      -2.360      -1.960
         x22    -0.909       0.052       1.190      -0.328       0.050
         x33     4.581      -0.924      -0.680       0.386       1.531
         x44     0.038       0.014      -0.050      -1.233       0.319
         x12     1.153      -0.220      -1.059       0.083       0.549
         x13    -1.369       0.951      -0.575      -0.901      -0.476
         x14     0.197       0.181      -0.159       1.450       0.104
         x23    -0.590       0.571       0.707      -0.281      -1.388
         x24    -0.093       0.193      -0.040      -0.222      -0.102
         x34    -0.081      -0.761       0.247       0.727       0.304
       _cons    11.493      11.515      11.500      11.642      11.497

         N_g     31.00       50.00       24.00       24.00       32.00
        Tbar      6.00        6.00        6.00        6.00        6.00
           N       186         300         144         144         192
        r2_w      0.92        0.89        0.90        0.90        0.91
         rho      0.97        0.94        0.90        0.92        0.98
        corr      0.07        0.22       -0.35        0.09       -0.81


    Variable     om7_6       om7_7      Pooled

          x1     0.550       0.507       0.669
          x2    -0.103       0.228       0.035
          x3    -0.186       0.090       0.013
          x4     0.559       0.478       0.378
         x11     0.912      -0.403       0.220
         x22     0.320       0.311      -0.054
         x33     2.722      -0.257      -0.213
         x44    -0.130       0.309       0.105
         x12    -0.279       1.069       0.008
         x13    -0.487       0.377       0.023
         x14    -0.325      -0.037      -0.093
         x23    -0.327      -0.014       0.031
         x24     0.119      -0.644      -0.018
         x34     0.853      -0.193       0.021
       _cons    11.398      11.468      11.565

         N_g     43.00       43.00      247.00
        Tbar      6.00        6.00        6.00
           N       258         258        1482
        r2_w      0.94        0.94        0.84
         rho      0.94        0.94        0.70
        corr     -0.29       -0.40        0.15

     Note: For a description of model diagnostics see stored results in xtreg,fe.
```

The `graph` option produces a graph of the heterogeneous slopes across the clustered individuals, but applies a colored scheme to readily assist the visual distinction among the clusters, and also overlays all plots into one graph. Given $\widehat{\Omega} = 7$, `graph` will overlay seven scatterplots plus seven linear fits in a multicolor visual. Hence, the clusters may not be entirely discernible. To manually reproduce a similar graph, but in monochrome scheme for printing, while keeping each plot in a separate graph, execute the following routine:

```
. * Pooled fitted values
. quietly xtreg yit x1-x34, fe

. predict xb, xb

. * Fitted values by cluster
. forvalues i = 1/7 {
  2.    quietly xtreg yit x1-x34 if om7==`i´, fe
  3.    estimates store om`i´
  4.    quietly predict xb`i´ if om7==`i´, xb
  5.    twoway (lfit yit xb, lwidth(*3) lpattern(solid) lcolor(gs8))
>        (scatter yit xb`i´ if om7==`i´, msymbol(oh) mlwidth(*.3) mcolor(gs0))
>        (lfit yit xb`i´ if om7==`i´, lwidth(*2.25) lpattern(dash) lcolor(gs0)),
>        aspect(1) ysize(1) xsize(1) scheme(sj) legend(off)
>        ytitle("Log of milk production (output)")
>        title("{&omega} = `i´", ring(0) pos(11) margin(medium))
>        name(g`i´, replace)
  6. }

. graph combine g1 g2 g3 g4 g5 g6 g7, row(2)
>       ysize(2) xsize(4) imargin(small) scheme(sj)
```

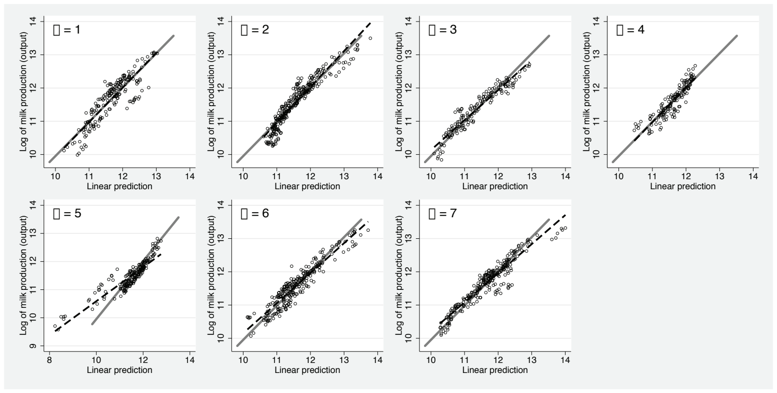


Figure 1. Heterogeneous dairy production functions

The gray solid line in figure 1 gives the pooled FE linear fit, which is the same across all plots. The dashed lines provide the linear fit for each ω. The linear fit for individuals classified in $\omega = 2$ has a slightly steeper slope, whereas the linear fit for individuals classified in $\omega = 3, 5, 6, 7$ has less steep slopes than the pooled FE model. The individuals in $\omega = 1, 4$ have virtually identical slopes as the pooled FE, but this does not necessarily mean that the estimated slope coefficients for $\omega = 1, 4$ are close to those of the pooled FE model. The fitted value, $\widehat{y}_{it}$, is a linear combination of all explanatory variables times their estimated slopes (that is, the linear predictor). Hence, it is still possible to have similar values of $\widehat{y}_{it}$ but weighted differently by the cluster-specific slope coefficients. Indeed, as shown in the output from the `table` option above, the cluster-specific coefficients for $\omega = 1, 4$ are considerably different from each other and, by comparison, from the pooled FE model.

To enhance interpretation in such cases, one may want to obtain cluster-specific plots for a given explanatory variable, rather than the linear combination of all of them. To achieve this, for every cluster, we can project the residuals obtained from a regression of the dependent variable on all other remaining independent variables onto the residuals obtained from a regression of the independent variable of interest to all other independent variables. A regression of the two predicted residuals gives the same slope coefficient as reported by the `table` option.[3] As a demonstration, we examine the slope heterogeneity in `x2` (the production input of land). From the `table` output above, the pooled FE slope coefficient is shown to be close to zero, yet there seems to be substantial variation in the slopes for $\omega = 1, 2, \ldots, 7$. We can visualize the differential slopes as follows:

```
. * First project x2 for the pooled sample
. xtreg yit x1 x3-x34, fe
  (output omitted)
. predict e1_x2_pool, e
. xtreg x2  x1 x3-x34, fe
  (output omitted)
. predict e2_x2_pool, e
. * Then project x2 for each omega
. quietly forvalues i = 1/7 {
  2.    xtreg yit x1 x3-x34 if om7==`i´, fe
  3.    predict e1_x2_`i´ if om7==`i´, e
  4.    xtreg x2 x1 x3-x34 if om7==`i´, fe
  5.    predict e2_x2_`i´ if om7==`i´, e
  6.    twoway (lfit e1_x2_pool e2_x2_pool, range(-.4 .4) lw(*2) lp(solid) lc(gs8))
>           (scatter e1_x2_`i´ e2_x2_`i´ if om7==`i´, ms(oh) mlw(*.3) mc(gs2))
>           (lfit e1_x2_`i´ e2_x2_`i´ if om7==`i´, lw(*1.5) lp(dash) lc(gs0)),
>           title("{&omega} = `i´", ring(0) pos(11) margin(medium) size(*1.25))
>           ytitle("yit projection") xtitle("x2 projection")
>           legend(off) scheme(sj) name(x2_`i´,replace)
  7. }
  (output omitted)
. graph combine x2_1 x2_2 x2_3 x2_4 x2_5 x2_6 x2_7, row(2) imargin(small)
>               ysize(2) xsize(5) scheme(sj)
```

It is clear from figure 2 that there is considerable heterogeneity in the cluster-specific coefficients of `x2`, even though the pooled FE slope is quite flat. The user may repeat the same process for every other variable of interest.

3. This result follows from the Frisch–Waugh–Lovell theorem.

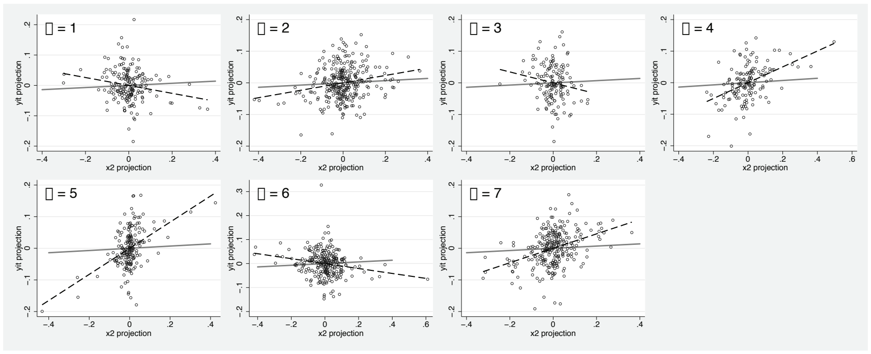


Figure 2. Heterogeneous slope coefficient for `x2` (production input of land)

Note that for `dairy.csv`, we cannot obtain the initial partition based on estimates of the individual-specific slopes using the `prevars(b)` option, because the number of degrees of freedom required to estimate an individual-specific regression exceeds the available observations; that is, $k + 1 = 16 > T = 6$. This is a balanced dataset and all individuals have $T = 6$. If one tries to estimate this model, an error message prompts the user to take a different course of action:

```
. xtregcluster y x1-x34, prevars(b) omega(2/10) name(omb)
Some panels have Ti < k+1. Choose an alternative initial partition,
or qualify the sample to panels with enough observations
insufficient observations
r(2001);
```

Lastly, one may also wish to explore a two-way error structure including time-specific FE. This can be easily implemented by specifying `i.year` as part of *indepvars* and repeating the entire analysis above.

6 Conclusions

`xtregcluster` is useful for discovering potential heterogeneous clusters in linear short panel-data models with FE, similar to an exploratory data analysis approach. It can also be used to assess the validity of the assumption of slope homogeneity or of normatively imposed preclassifications.

7 Acknowledgments

We acknowledge Karl Keesman's input during the development of the first draft of this program. We also acknowledge useful comments from participants at the 2015 Oceania Stata Users Group meeting at Canberra.

8 References

Baltagi, B. H., and J. M. Griffin. 1997. Pooled estimators vs. their heterogeneous counterparts in the context of dynamic demand for gasoline. *Journal of Econometrics* 77: 303–327.

Burnside, C. 1996. Production function regressions, returns to scale, and externalities. *Journal of Monetary Economics* 37: 177–201.

Everitt, B. S., S. Landau, M. Leese, and D. Stahl. 2011. *Cluster Analysis.* 5th ed. Chichester, UK: Wiley.

Kaufman, L., and P. J. Rousseeuw. 2005. *Finding Groups in Data: An Introduction to Cluster Analysis.* Hoboken, NJ: Wiley.

Pesaran, M. H., Y. Shin, and R. P. Smith. 1999. Pooled mean group estimation of dynamic heterogeneous panels. *Journal of the American Statistical Association* 94: 621–634.

Sarafidis, V., and N. Weber. 2015. A partially heterogeneous framework for analyzing panel data. *Oxford Bulletin of Economics and Statistics* 77: 274–296.

About the authors

Demetris Christodoulou is at the University of Sydney Business School and General Convenor of the research network Methodological and Empirical Advances in Financial Analysis.

Vasilis Sarafidis is at the Department of Econometrics and Business Statistics at Monash University, and a founding member of Methodological and Empirical Advances in Financial Analysis.

The Stata Journal (2017)
17, Number 2, pp. 330–342

Introducing the StataStan interface for fast, complex Bayesian modeling using Stan

Robert L. Grant
BayesCamp
Croydon, UK
robert@bayescamp.com

Bob Carpenter
Columbia University
New York, NY

Daniel C. Furr
University of California at Berkeley
Berkeley, CA

Andrew Gelman
Columbia University
New York, NY

Abstract. In this article, we present StataStan, an interface that allows simulation-based Bayesian inference in Stata via calls to Stan, the flexible, open-source Bayesian inference engine. Stan is written in C++, and Stata users can use the commands `stan` and `windowsmonitor` to run Stan programs from within Stata. We provide a brief overview of Bayesian algorithms, details of the commands available from Statistical Software Components, considerations for users who are new to Stan, and a simple example. Stan uses a different algorithm than `bayesmh`, BUGS, JAGS, SAS, and MLwiN. This algorithm provides considerable improvements in efficiency and speed. In a companion article, we give an extended comparison of StataStan and `bayesmh` in the context of item response theory models.

Keywords: st0476, stan, windowsmonitor, StataStan, Bayesian, bayesmh, interface, shell commands, Stan

1 Introduction

Stata users have long been able to seamlessly access other software specializing in Bayesian analysis, thanks to Stata users' abilities to write arbitrary information to ASCII text files and send commands to the operating system. This allowed for commands such as `runmlwin` (Leckie and Charlton 2013) and `wb` (Thompson 2017) to send data and code to MLwiN and WinBUGS, respectively, then collect the results and display them inside Stata for further calculation and graphing. Since 2015, when Stata 14 was released, Stata users have been able to use a native implementation of Bayesian simulation algorithms by using the `bayesmh` command. However, `bayesmh` is limited to regressionlike models where a dependent variable has a specified likelihood conditional on some function of independent variables, and can allow only certain prior distributions. It cannot, for example, fit structural equation models, Gaussian processes, or spatial correlation models.

The day after Stata 14's release, StataStan was published online (Stan Development Team 2016b). StataStan is an umbrella term for all commands and programs necessary to interface with Stan from Stata. It can be installed from Statistical Software Components by typing

```
ssc install stan
```

and Windows users should also install `windowsmonitor` by typing

```
ssc install windowsmonitor
```

Stan is an open-source, collaboratively built software project that implements an algorithm (Hamiltonian Monte Carlo) for Bayesian modeling that is faster and more stable than the algorithms (random walk Metropolis–Hastings and the Gibbs sampler) implemented in BUGS, JAGS, SAS, MLwiN, and `bayesmh`. Stan has been applied to a wide range of complex statistical models, including time series, imputation, mixture models, meta-analysis, cluster analysis, Gaussian processes, and item response theory. These extend beyond the current (Stata 14.2) capability of `bayesmh`, which is explicitly for regression. In our companion article (Grant et al. 2017), we describe the functionality of Stan and advantages of its algorithm. In this article, we give a brief overview of Hamiltonian Monte Carlo in intuitive terms, set out the syntax of the commands, and present a worked example.

2 Hamiltonian Monte Carlo

All Bayesian methods make estimates and inference by evaluating posterior distributions, combinations of likelihood based on data, and a model with prior distributions representing uncertainty about parameters of the model before the data were known. Different practitioners take the prior to mean different concepts, in the same way that "uncertainty" and "probability" are not rigorously defined concepts despite decades of hard work by statisticians and philosophers of science. Regardless of the interpretation, Bayesian methods differ from frequentist methods in that they allow probability statements to be made about any unknown value, not just those that represent eternally replicable random sampling.

Textbook examples often start with algebraically tractable posterior distributions, but in practice, this is generally either infeasible or too time consuming and prone to human error to be worthwhile. Instead, software allows the analyst to run one or more Markov chains of pseudorandom values that converge to a stationary distribution equivalent to draws from the joint posterior distribution of all parameters of interest. From a large enough number of these draws, estimation and inference can be done empirically. The older algorithms, random walk Metropolis–Hastings, and the Gibbs sampler take random steps through parameter space and accept or reject the new location based on its posterior probability.

This can work well under some circumstances but under others can require large numbers of draws before they accurately represent the posterior distribution (convergence). Problems like this commonly arise when parameters are correlated (like, for example, how the intercept and slope of a bivariate linear regression are correlated with only a small amount of data); when priors are not ideal matches for the likelihood (a subtle topic beyond the scope of this article but discussed in Bayesian textbooks [Gelman et al. 2013]); or when initial values are poor guesses. Hamiltonian Monte Carlo addresses these issues by using Hamilton's equations of motion with periodic random impulses (Neal 2011). Exploration of the posterior probability is then analogous to a particle moving in a force field (picture a beachball rolling in the hollow between sand dunes, with occasional random kicks—gravity is the force providing the Hamiltonian motion); as the joint posterior distribution guides movement to the region of highest posterior probability, the problems of sampling using random steps disappear. Even chains with poor initial values can still reveal the whole posterior distribution relatively quickly (Neal 2011). A computer requires computationally expensive numerical integration and differentiation to perform this imitation of life, but the lifting of the problems associated with random walk Metropolis–Hastings and Gibbs more than compensates for this. The no-U-turn sampler is the algorithm implemented in Stan (Hoffman and Gelman 2014), which automatically tunes the parameters of Hamiltonian Monte Carlo, achieving nearly optimal integration time in recent tests using CmdStan (Betancourt 2016).

In the companion article, we present a comparison of the efficiency of StataStan alongside `bayesmh` for an item response model (Grant et al. 2017).

3 The stan and windowsmonitor commands

3.1 Objectives and development

Building on the history of linking Stata to WinBUGS (Thompson 2017), we sought to provide one command that would dispatch a specified Stan model code along with data. Because Stata can easily issue operating system commands, we use this to run the command-line implementation of Stan (CmdStan) and display summary results inside Stata. This is the approach also taken by the Stan interfaces from MATLAB and Julia. CmdStan has to be installed before using StataStan, but this is relatively straightforward with instructions on the Stan website, http://mc-stan.org.

We believe that Stata users who are becoming familiar with Bayesian techniques will find StataStan a flexible, stable, and fast tool. Also, people who already use Stata and Stan separately will find it helpful to keep everything in one workflow, because most people find it easier to work with one piece of software than to switch among them (and find it easier to maintain quality control). This allows data processing, simple analysis, complex modeling, graphics, and report writing all in one place.

One unexpected problem we encountered was that Windows does not make its standard output on the command line available in such a way that Stata can display it in the results window until the external program has finished execution. In the case of complex Bayesian models that can take hours to run, this would be unacceptable. Therefore, we wrote a small companion program called `windowsmonitor` that displays command-line output close to real time. `windowsmonitor` may provide a useful alternative to `shell` and `winexec` in other settings too.

3.2 The stan command for Stata

`stan` specifies what data are to be sent to CmdStan, with options controlling its settings and additional requirements such as sampling diagnostics or posterior modes. Data are passed to CmdStan in a text file, and outputs are returned similarly. These files are temporarily created in the CmdStan directory, then moved to the working directory. There is an option to retain all files. Otherwise, unnecessary files are deleted afterward. Users should be mindful that any existing files in these locations with these names may be overwritten. A model has to be stored in its own file with extension `.stan`, and we discuss different ways to achieve this below.

Syntax of stan

`stan` *varlist* [*if*] [*in*] [, `datafile(`*filename*`)` `modelfile(`*filename*`)` `inline`
`thisfile(`*filename*`)` `rerun` `initsfile(`*filename*`)` `load` `diagnose`
`outputfile(`*filename*`)` `chainfile(`*filename*`)` `mode` `modesfile(`*filename*`)`
`winlogfile(`*filename*`)` `seed(`*integer*`)` `warmup(`*integer*`)` `iter(`*integer*`)`
`thin(`*integer*`)` `chains(`*integer*`)` `skipmissing` `matrices(`*string*`)` `globals(`*string*`)`
`keepfiles` `stepsize(`*integer*`)` `stepsizejitter(`*integer*`)`]

Options

`datafile(`*filename*`)` specifies the name (and path if desired) of a text file where `stan` will write the data on its way to Stan. This is done in the format used by R/S-Plus and BUGS. For example, with the sample 1978 Automobile dataset,

```
stan mpg, ...
```

would write

```
mpg=c(...)
```

The default is `datafile(statastan_data.R)`.

`modelfile(`*filename*`)` specifies the name (and path if desired) of a text file containing the Stan model. The file must have the extension `.stan`. If this file already exists, the model is read from there, or the model can be written into the file using one of the methods detailed below under *Specifying the Stan model*. The default is `modelfile(statastan_model.stan)`.

`inline` instructs Stata to read the `.stan` model from a comment block inside the do-file (see below under *Specifying the Stan model* for further discussion of `modelfile()`, `inline`, and `thisfile()`).

`thisfile(`*filename*`)` specifies the name (and path if required) of the current do-file; this is an option if `inline` has been specified (see below under *Specifying the Stan model* for further discussion of `modelfile()`, `inline`, and `thisfile()`).

`rerun` uses the existing executable file with the same name as `modelfile()` (in Windows, it will have the extension `.exe`). This should exist in the working directory. Be aware it will be copied into `cmdstandir` (see below), deleting any existing file of that name.

`initsfile(`*filename*`)` specifies the name of a text file in R/S-Plus format containing initial values. Because Stan is far less sensitive to initial values than software using older algorithms, we do not presently provide any mechanism like the `datafile()` option to write this file from inside Stata.

`load` instructs Stata to read in the resulting draws as its current dataset.

`diagnose` runs Stan's diagnostics and displays them after sampling to examine whether the algorithm has run successfully.

`outputfile(`*filename*`)` provides the name (and path if required) for the text file into which CmdStan will write its outputs. The default is `outputfile(output.csv)`.

`chainfile(`*filename*`)` provides the name for a comma-separated values format file that will contain the draws from CmdStan; this is the same as `outputfile()`, but extra information is removed so it can be read into Stata using import delimited. The default is `chainfile(statastan_chains.csv)`.

`mode` runs Stan's optimization to find posterior modes and displays the results after sampling; it will also write the output into `modesfile()` (see below).

`modesfile(`*filename*`)` provides the name of a text file to hold output from CmdStan's estimation of modes. The default is `modesfile(modes.csv)`.

`winlogfile(`*filename*`)` provides the name of a temporary file to hold Windows output (see `windowsmonitor`); `windowsmonitor` will display the output in Stata's Results window, so there is no need to change the name of the temporary file from the default, which is `winlog.txt`.

`seed(`*integer*`)` provides an integer pseudorandom-number generator seed for Stan.

`warmup(`*integer*`)` specifies the number of warmup draws, which are discarded from the output and summaries. The default is `warmup(1000)`.

`iter(`*integer*`)` specifies the number of iterations (draws) to retain after `warmup()`. The default is `iter(1000)`.

`thin(`*integer*`)` specifies how much Stan thins draws; if `thin()` is set to n, Stan will retain one out of every n draws in output files and use the thinned draws for summaries. The default is `thin(1)` (no thinning).

`chains(`*integer*`)` determines how many chains to run, in parallel if possible (regardless of the Stata flavor installed).

`skipmissing` removes missing data observations (on a cell-by-cell basis inside each column) before sending data to Stan. This is relevant if you want to send a series of vectors of different sizes by making these appear as "variables" in your Stata data. This could be useful in the context of multilevel models with smaller vectors of cluster-level data. It is not a natural way to think of Stata data, so it should be used with caution because it will apply to all the variables in *varlist*.

`matrices(`*string*`)` provides a list of matrices to send to Stan or if set to `all`, it will send all current matrices. These are written into `datafile()` as two-dimensional arrays.

`globals(`*string*`)` provides a list of global macros to send to Stan, or `all` to send all current global macros. These are written into `datafile()` as scalars. The user should not write a string value, because this will probably cause an error from CmdStan.

`keepfiles` instructs `stan` to keep all files produced along the way; otherwise, the model file, C++ file, executable file, chains file, and (if produced) modes file will be retained in the working directory.

`stepsize(`*integer*`)` sets the stepsize for Hamiltonian Monte Carlo. The default is `stepsize(1)` (see the Stan manual [Stan Development Team 2016b] for more details).

`stepsizejitter(`*integer*`)` sets the stepsize jitter for Hamiltonian Monte Carlo. The default is `stepsizejitter(0)` (see the Stan manual [Stan Development Team 2016b] for more details).

4 Specifying the Stan model

You can specify the Stan model in at least three ways. First, you can write a `.stan` file externally, for example, in a text editor, then name it with the `modelfile()` option. This has the disadvantage that updating the analysis may require synchronized changes in the do-file and model file. However, we recommend this method as the starting point for new StataStan users because it avoids any bugs in writing and reading text files and allows you to begin immediately using examples from the Stan manual and website. Second, you can include the code inside a comment block in the do-file. If you use the `inline` and `thisfile()` options, Stata will read the text contents of `thisfile()` and identify the comment block that begins (on the line following the `/*` symbol) with the word `data`, as seen below:

```
/*
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  theta ~ beta(1,1);
  y ~ bernoulli(theta);
}
*/
```

Stata will then write the contents of the block to the `.stan` file specified in `modelfile()`.

Third, you can include the model code in the do-file as a series of strings in a `foreach` loop, which writes each line to the `modelfile()`. This has the advantage that all Stata and Stan code is in one file, but does not rely on naming or finding the do-file.

At present, the `inline` approach (option two above) does not accommodate multiple blocks of code, but we intend to add this capability.

4.1 The windowsmonitor program

`windowsmonitor` is a wrapper extending the ability of `shell`. It will be called by `stan` under Windows only; it will return an error message if it is used in Mac or Linux computers. It intercepts the `stdout` stream (text displayed on the screen for command line programs) and prints it inside Stata. It does this by diverting `stdout` to a text file, checking that file every two seconds for new content, and displaying that in Stata if it finds any. This continues until it receives a message that it is finished (in the form of a final line of output, `Finished!`), which is added automatically. The user should avoid using `windowsmonitor` to carry out any task that could write one `Finished!` line for any reason, because this will terminate the display inside Stata prematurely. However, if this is unavoidable, it is relatively simple to amend the signal word `Finished!` in the source code. `windowsmonitor` creates a file called `wmbatch.bat`. If this survives execution, it can safely be deleted later.

Syntax of windowsmonitor

`windowsmonitor, command(`*string*`)` [`waitsecs(`*integer*`)` `winlogfile(`*filename*`)`]

Options

`command(`*string*`)` contains the Windows command-line code to be sent for execution. `command()` is required.

`waitsecs(`*integer*`)` specifies the number of seconds to wait for output to appear before giving up. The default is `waitsecs(20)`.

`winlogfile(`*filename*`)` specifies the file (and path if desired) to store the output in; by default, a Stata `tempfile` will be used, so there is nothing to be gained from specifying a `tempfile` macro here. The default is `winlogfile(winlog.txt)`.

5 Considerations for newcomers to Stan

Newcomers are strongly advised to work through some of the examples in the Stan manual before attempting serious applications. The Stan user must specify the type (such as integer or real number) as data or parameters. This allows Stan to make efficient calculations and helps with checking for inadvertent errors at compile time. Stan will translate the model to C++, which is itself a "typed" language. For the most part, the Stata user need not be concerned with this other than with the obvious choice when writing the Stan code. However, one potential pitfall may arise when reading in data from nonnative file formats into Stata and sending it with `stan`. Floating-point precision means that what the person reads may not match what the computer stores, and this may lead to a "type mismatch" error message from CmdStan.

The statistics reported by CmdStan and hence displayed by `stan` are the mean of draws from the posterior; the Monte Carlo standard error representing the uncertainty in the results arising from a finite number of draws; the standard deviation; the 5th, 50th, and 95th centiles of the draws; the number of effective independent samples (`N_Eff`, which accounts for autocorrelation in the chains) and number of effective independent samples obtained per second (`N_Eff/s`); and a measure of convergence (`R_hat`). The calculation of these measures is set out in Gelman et al. (2013). `N_Eff` and `R_hat` are best assessed across multiple chains, so we advise users run at least four chains as a general rule. `stan` can run parallel chains on multicore computers, even if Stata/MP is not installed, so most modern laptops can run four chains simultaneously. In the authors' experience, this runs in about half the time of serial chains.

Beyond these reported statistics, the value of loading the draws from the posterior distributions is that custom-derived values can be calculated and summarized by the user inside Stata to provide decision-theoretic outputs. To give an example from health economics, we can load a meta-analysis from Stan providing inference on the effectiveness of alternative drugs into Stata and combine it with constant costs to derive a new cost-effectiveness variable, which allows probability statements about whether the cost effectiveness exceeds a willingness-to-pay threshold. Another important benefit of working with the posterior draws is that the covariance structure among the parameters is preserved, while the tabulated summaries provide only marginal inferences.

Another consideration is that the number of available CPU cores needs to be specified when installing CmdStan itself, and the StataStan `chains()` option can parallelize only up to this number (Stan Development Team 2016a).

Stan model code allows for vectorized statements such as

```
y ~ bernoulli(theta);
```

instead of

```
for (n in 1:N) { y[n] ~ bernoulli(theta); }
```

Both can be used in Stan, but the vectorized version is generally faster in execution.

6 Example

All the models set out in the Stan manual and website can be fit directly using StataStan, including many models that are not possible in `bayesmh`. We can use StataStan for a simple example to estimate the probability of success θ in a Bernoulli process,

$$\Pr(y_i) = \theta, \quad 1 \leq i \leq 10, \quad i \in \mathbb{N}$$

when we have 10 outcomes: 8 failures and 2 successes. We will apply a flat prior distribution over $[0, 1]$, either by explicitly specifying it or by omitting it because Stan uses uniform priors as default, provided that bounds on the parameter have been specified. The corresponding `bayesmh` command is

```
bayesmh y, likelihood(dbernoulli({theta})) prior({theta},beta(1, 1))
```

The Stan code for this example is contained in the examples folder inside CmdStan.

```
data {
  int<lower=0> N;
  int<lower=0,upper=1> y[N];
}
parameters {
  real<lower=0,upper=1> theta;
}
model {
  theta ~ beta(1,1);
  y ~ bernoulli(theta);
}
```

The code is arranged in blocks of data, parameters, and model. Other block types can also be included, described fully in the Stan manual. Each object in the model, whether data or parameter, must be declared with its type and constraints before it can be used. Like BUGS and JAGS, the assignment operator $<-$ is used to calculate a value and store it in the object named on the left-hand side, while the ~ operator has two functions. In the line

```
theta ~ beta(1,1);
```

we are specifying a prior distribution (because theta is already declared as a parameter), and in the line

```
y[i] ~ bernoulli(theta);
```

we are incrementing the log probability by the likelihood contribution of one observation according to the Bernoulli probability given the current estimate of theta.

Having specified this model, we can make the data,

```
clear
set obs 10
generate y=0
replace y=1 in 2
replace y=1 in 10
```

and then call `stan`:

```
quietly count
global N=r(N)
global cmdstandir "/path_to/CmdStan"
stan y, modelfile("bernoulli.stan") cmd("$cmdstandir") globals("N")
```

StataStan first displays its own version number and then the CmdStan version installed in `cmdstandir`. The first output to be displayed concerns translating the model to C++, then compiling the resulting code. Compiling can be time consuming but does not have to be done again unless the model changes. If StataStan finds CmdStan successfully, and CmdStan is properly installed, this line will appear followed by some output that users can ignore:

```
--- Translating Stan model to C++ code ---
```

Next, a block of code will appear, starting with this line and comprising the command to the g++ compiler program (which is installed as part of CmdStan):

```
--- Linking C++ model ---
```

After compilation, we will see some settings for CmdStan, including the number of samples to retain and the number to use as warm-up:

```
method = sample (Default)
  sample
    num_samples = 1000 (Default)
    num_warmup = 1000 (Default)
```

We then see the iterations appear, followed by a total time to do the sampling:

```
Iteration: 1800 / 2000 [ 90%]  (Sampling)
Iteration: 1900 / 2000 [ 95%]  (Sampling)
Iteration: 2000 / 2000 [100%]  (Sampling)

Elapsed Time: 0.017155 seconds (Warm-up)
              0.024054 seconds (Sampling)
              0.041209 seconds (Total)

Inference for Stan model: bernoulli_model
4 chains: each with iter=(1000,1000,1000,1000);
  warmup=(0,0,0,0); thin=(1,1,1,1);
  4000 iterations saved.
```

```
Warmup took (0.017, 0.017, 0.017, 0.016) seconds,
  0.067 seconds total
Sampling took (0.024, 0.032, 0.031, 0.031) seconds,
  0.12 seconds total
```

This is followed by a summary of the parameters:

```
        Mean     MCSE   StdDev     5%    50%   95%    N_Eff  N_Eff/s    R_hat
theta   0.25  2.3e-03  1.2e-01  0.076  0.24  0.46     2784    23545  1.0e+00

Samples were drawn using hmc with nuts.
For each parameter, N_Eff is a crude measure of effective
  sample size, and R_hat is the potential scale reduction
  factor on split chains (at convergence, R_hat=1).
```

This shows us that we ran 4 chains and retained 1,000 samples from each, but because of autocorrelation, this was equivalent to 2,784 independent samples (23,545 independent samples per second). The posterior mean for θ was 0.25 (pulled upward from the maximum likelihood estimate by the flat prior and the small dataset). If `mode` is specified, we will then see the posterior mode,

```
Log-probability at maximum: -5.004020214080811

---------------------
          | Posterior
Parameter |      Mode
----------+----------
theta     |   .200004
---------------------
```

which is directly comparable (with a flat prior) with the maximum likelihood estimate, 0.2.

If we specify `diagnose`, we will see corresponding output; see the Stan manual for details on this.

```
TEST GRADIENT MODE
 Log probability=-7.10591
 param idx      value      model  finite diff          error
         0  -0.557247   -1.37022     -1.37022  -1.66588e-010
```

Finally, if we specify `load`, we will see a Stata-generated summary, including the 95% credible interval:

```
    variable |         N      mean        sd  se(mean)
-------------+-------------------------------------------------------
       theta |      1000  .2485084  .1121162  .0035454
---------------------------------------------------------------------

    variable |       min        p1        p5       p25
-------------+-------------------------------------------------------
       theta |   .019246  .0477189  .0814933     .1628
---------------------------------------------------------------------
```

```
    variable |       p50       p75       p95       p99
-------------+----------------------------------------
       theta |   .244064  .3222845  .4458295  .5513045
------------------------------------------------------

95% CI for theta: .0656607497483492 to .4934002541005615
```

This is similar to the approximate confidence interval:

```
. cii proportions 10 2, wilson

                                                         ------- Wilson -------
    Variable |        Obs  Proportion    Std. Err.       [95% Conf. Interval]
-------------+-------------------------------------------------------------------
             |         10          .2     .1264911        .0566822    .5098375
```

We see the data replaced with variables called `theta` (which contains draws for the parameter of that name), `lp__`, `accept_stat__`, `stepsize__`, `treedepth__`, `n_leapfrog__`, and `n_divergent__`, all of which are created by CmdStan to track progress of the algorithm and can be safely deleted unless needed for methodological investigations. The `theta` variable, containing the draws from the posterior, can then be used for graphics or further inference.

7 Conclusion

Stan continues to develop rapidly, with one major project being the inclusion of Riemann manifold Hamiltonian Monte Carlo, which will provide further significant improvements in speed and stability (Girolami and Calderhead 2011). StataStan can readily track this by adding new options that are passed to future versions of CmdStan.

Stan and all its interfaces have been made possible by enthusiastic contributions from developers around the world, coordinated by a core team. We encourage all interested Stata users to visit http://mc-stan.org and become involved there through reporting issues and suggesting improvements (Stan Development Team 2016b).

8 Acknowledgments

We thank the Institute of Education Sciences for partial support of this work. We are also grateful to users who tested StataStan and provided feedback and to John Thompson of the University of Leicester and Charles Opondo of the University of Oxford for suggesting ways of inline model specification.

9 References

Betancourt, M. 2016. Identifying the optimal integration time in Hamiltonian Monte Carlo. ArXiv Working Paper No. arXiv:1601.00225. https://arxiv.org/abs/1601.00225.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, eds. 2013. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: CRC Press.

Girolami, M., and B. Calderhead. 2011. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society, Series B* 73: 123–214.

Grant, R. L., D. C. Furr, B. Carpenter, and A. Gelman. 2017. Fitting Bayesian item response models in Stata and Stan. *Stata Journal* 17: 343–357.

Hoffman, M. D., and A. Gelman. 2014. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* 15: 1593–1623.

Leckie, G., and C. Charlton. 2013. runmlwin—A program to run the MLwiN multilevel modelling software from within Stata. *Journal of Statistical Software* 52(11): 1–40.

Neal, R. M. 2011. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, ed. S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, 113–162. Boca Raton, FL: Chapman & Hall/CRC.

Stan Development Team. 2016a. CmdStan Interface: User's Guide. https://github.com/stan-dev/cmdstan/releases/download/v2.14.0/cmdstan-guide-2.14.0.pdf.

———. 2016b. Stan Modeling Language: User's Guide and Reference Manual, Version 2.14.0. https://github.com/stan-dev/stan/releases/download/v2.14.0/stan-reference-2.14.0.pdf.

Thompson, J. 2017. WinBUGS from Stata. http://www2.le.ac.uk/departments/health-sciences/research/gen-epi/Progs/winbugs-from-stata.

About the authors

Robert Grant is a statistician who runs a training and consultancy startup called BayesCamp. He was senior lecturer in health and social care statistics at Kingston University and St George's, University of London, and previously worked on clinical quality and safety data for the Royal College of Physicians and evidence-based guidelines for the National Institute for Clinical Excellence. Besides Bayesian modeling, he specializes in data visualization.

Daniel Furr is a graduate student in the Graduate School of Education at the University of California, Berkeley. He is the developer of `edstan`, an R package for Bayesian item response theory modeling using Stan, and the author of several related case studies. He specializes in item response theory and predictive methods.

Bob Carpenter is a research scientist in computational statistics at Columbia University. He designed the Stan probabilistic programming language and is one of the Stan core developers. He was professor of computational linguistics (Carnegie Mellon University) and an industrial researcher and programmer in speech recognition and natural language processing (Bell Labs, SpeechWorks, LingPipe) and has written two books on programming language theory and linguistics.

Andrew Gelman is a professor of statistics and professor of political science at Columbia University and the author of several books, including *Bayesian Data Analysis*.

The Stata Journal (2017)
17, Number 2, pp. 343–357

Fitting Bayesian item response models in Stata and Stan

Robert L. Grant
BayesCamp
Croydon, UK
robert@bayescamp.com

Daniel C. Furr
University of California at Berkeley
Berkeley, CA

Bob Carpenter
Columbia University
New York, NY

Andrew Gelman
Columbia University
New York, NY

Abstract. Stata users have access to two easy-to-use implementations of Bayesian inference: Stata's native `bayesmh` command and StataStan, which calls the general Bayesian engine, Stan. We compare these implementations on two important models for education research: the Rasch model and the hierarchical Rasch model. StataStan fits a more general range of models than can be fit by `bayesmh` and uses a superior sampling algorithm, that is, Hamiltonian Monte Carlo using the no-U-turn sampler. Furthermore, StataStan can run in parallel on multiple CPU cores, regardless of the flavor of Stata. Given these advantages and given that Stan is open source and can be run directly from Stata do-files, we recommend that Stata users interested in Bayesian methods consider using StataStan.

Keywords: st0477, stan, windowsmonitor, StataStan, bayesmh, Bayesian

1 Introduction

Stata is widely used in the social sciences, economics, and biostatistics. In 2015, it became possible to routinely fit Bayesian models in Stata by using 1) Bayesian modeling commands introduced in Stata 14, which use the Metropolis–Hastings algorithm and Gibbs sampler, and 2) StataStan, an interface to the open-source Bayesian software, Stan (Grant 2015; Stan Development Team 2016). Previously, Bayesian methods were available in Stata only through user-written commands to interface with external software such as BUGS, JAGS, or MLwiN.

At the time of writing, the native Bayes implementation in Stata, `bayesmh`, allows a choice among 10 likelihood functions and 18 prior distributions. `bayesmh` is explicitly focused around regression models, although extensions to hierarchical (multilevel) models are possible with the inclusion of hyperpriors. Additionally, the user may write customized likelihood functions or customized posterior distributions.

Built for Bayesian inference, Stan is an open-source, collaboratively built software project that allows general continuous-parameter models, including all the models that can be fit in Stata's `bayesmh` and many others. Stan has been applied to a wide range of complex statistical models, including time series, imputation, mixture models, meta-analysis, cluster analysis, Gaussian processes, and item response theory. These extend beyond the current (Stata 14.2) capability of `bayesmh`. Stan can run from various data analysis environments such as Stata, R, Python, and Julia and also has a command-line interface (CmdStan). Stan uses Hamiltonian Monte Carlo (HMC) and the no-U-turn sampler (Hoffman and Gelman 2014) with the additional options of variational inference (Kucukelbir et al. 2015) and the L-BFGS optimization algorithm (Nocedal and Wright 2006). The advantages of HMC and the no-U-turn sampler in speed, stability with regard to starting values, and efficiency over Metropolis–Hastings and Gibbs have been described elsewhere (Neal 2011; Hoffman and Gelman 2014). As a result of the Hamiltonian dynamics, HMC is rotation invariant, which makes it well suited to highly correlated parameters. It is also not slowed down by nonconjugate models.

The languages used by these packages are notably different. In Stan, models are specified in a series of probability statements specifying prior distributions and likelihoods. `bayesmh` follows standard Stata syntax to give a compact specification of the most common regression and related models. Stan works by translating the user's model code into C++, then compiling and running the resulting executable file. Stan can run in parallel on multicore computers if the number of available cores was specified when installing CmdStan itself.

In this article, we compare `bayesmh` and StataStan on some item response models. These logistic regression (or Rasch) models are popular in education research and in political science, where they are called ideal-point models (Rasch 1960).

2 Models

We fit the models using data simulated as specified below. We checked that the `bayesmh` and StataStan implementations gave the same answer (modulo the inevitable Monte Carlo error of these stochastic algorithms), then compared the programs on speed and efficiency in terms of time per the number of effective independent samples.

The Rasch model can be written as

$$\Pr(y_{ip} = 1|\theta_p, \delta_i) = \text{logit}^{-1}(\theta_p - \delta_i)$$
$$\theta_p \sim N(0, \sigma^2)$$

where $y_{ip} = 1$ if person p responded to item i correctly and 0 otherwise, and $i, p \in \mathbb{N}$; $1 \leq i \leq I$; $1 \leq p \leq P$. The parameter θ_p represents the latent "ability" of person p, and δ_i is a parameter for item i. We considered a simple version of the model in which the abilities are modeled as exchangeable draws from a normal distribution with scale σ. We assigned a $N(0, 10)$ prior distribution to δ_i and took two approaches to priors for σ. First, we matched the Rasch model example in the Stata 14 manual (see [BAYES] **bayesmh**), which uses an inverse-gamma prior for σ^2, which we do not recommend (Gelman 2006). Second, we used a preferred approach of uniform priors for σ, which is the Stan default if a prior is not specified. It is easy in StataStan to add a line of code to the model to include a different prior on σ or σ^2.

A natural hierarchical extension of the Rasch model adds a hyperprior for δ_i so that

$$\Pr(y_{ip} = 1|\theta_p, \delta_i) = \text{logit}^{-1}(\theta_p - \delta_i)$$
$$\theta_p \sim N(0, \sigma^2)$$
$$\delta_i \sim N(\mu, \tau^2)$$

where μ is the model intercept. Persons and items are regarded as two sets of exchangeable draws.

3 Methods

We simulated data from the above model with 500 persons each answering 20 items. For true values of δ_i, we assigned equally spaced values from -1.5 to 1.5, and we set the true σ to 1.

We set up the Rasch and hierarchical Rasch models similarly, running four chains in series in Stan version 2.11 and Stata 14.1. We drew initial values for the chains from independent uniform distributions -1 to 1 on the location parameters $\mu^{(0)}$, $\delta^{(0)}$, and $\theta^{(0)}$ and drew uniform distributions from 0 to 2 on the scale parameters $\sigma^{(0)}$ and $\tau^{(0)}$. We assigned all δ_i's identical starting values for each chain and did the same for the θ_p's. The reason for this (admittedly unusual) choice is that this approach is much easier to use with `bayesmh`. We used the same starting values for both StataStan and `bayesmh` (and in the comparison described below, for JAGS). These item response models were not sensitive to starting values.

We ran 10 chains for 2,500 discarded warm-up iterations and 2,500 posterior draws each. For timing purposes, we ran all chains in serial, thus eliminating one of Stan's advantages—that it can automatically run multiple chains in parallel on a multicore machine regardless of the flavor of Stata. However, we made one comparison using parallel computation, which is described below. We provide the Stan programs and commands in the appendix. The options specified for `bayesmh` are nearly identical to those in the example provided in the Stata manual (see [BAYES] **bayesmh**). There is a difference in how the δ and θ parameters are sampled, which plays to the strengths of the different algorithms; Hamiltonian Monte Carlo is more efficient with distributions centered on or close to zero, regardless of correlation, while random walk Metropolis–Hastings in `bayesmh` is improved by using the random-effects option (see [BAYES] **bayesmh**). This feature, added in Stata 14.1, markedly improves effective sample size for models amenable to a random-effects parameterization. Other model forms will not benefit from it, so for comparison, we ran `bayesmh` both with and without random effects.

We monitored convergence for each parameter using the $\widehat{R}$ statistic, which is a rough estimate of the square root of the ratio of overall (across chains) posterior variance to within-chain posterior variance (Gelman et al. 2013). Values of $\widehat{R}$ near 1 indicate convergence, while greater values indicate nonconvergence. Values less than 1.1 are generally considered acceptable. The efficiency of the estimations is evaluated by the seconds per estimated effective sample size, $s/\widehat{n}_{\text{eff}}$ (Gelman et al. 2013). This reflects the fact that more highly autocorrelated chains of draws from the posterior distributions give less precise inference, equivalent to a smaller number of effectively independent samples that n_{eff} estimates. We used two versions of timings: an all-in time using the Stata command `timer` from the lines of the do-file above and below the `bayesmh` or `stan` command, as well as a simulation-only time obtained from the CmdStan output and from the value returned to `e(simtime)` by `bayesmh` (an undocumented return value). StataStan's all-in time includes compiling the model, and `bayesmh`'s all-in time includes internal model building before simulation can begin. To run multiple chains in StataStan, compilation is required only once, and if the data change but a model does not, a previously compiled executable file can be reused. The total time and simulation-only times represent opposite ends of a spectrum of performance. In real-life implementation, if there are many retained iterations compared with the warm-up iterations, and if compilation (in the case of StataStan) and model building (in the case of `bayesmh`) are not needed in every chain, total time will approach the simulation-only time.

To further investigate the efficiency of the software as models become more demanding, we carried out the same analyses on simulated data with 20 items and 100, 500, 1,000, 5,000, and 10,000 people. We compared StataStan 1.2.1 (calling CmdStan 2.11) with Stata 14.1's `bayesmh` command as above and also with the open-source software JAGS 4.0.0 (Plummer 2007) with the `rjags` package in R 3.2.3 and ran four chains in each instance. We compared $s/\widehat{n}_{\text{eff}}$ for the hyperparameters σ^2, μ, and τ^2 and for the worst parameter (lowest $\widehat{n}_{\text{eff}}$, reflecting the frequent need to run the software until all parameters are adequately estimated) in each model. We ran `bayesmh` both with and without the `exclude()` option on the θ's to examine the effect of reducing memory requirements. We also ran StataStan again with parallel chains for 20 items and 1,000 people to examine the increase in speed achieved with 4 CPU cores. All simulations were conducted on an "early 2015" MacBook Pro laptop running OS X 10.11.6 (El Capitan) with a 2.7 GHz Intel Core i5 4-core processor and 8 GB of 1867 MHz DDR3 RAM, with all networking turned off.

4 Results

For the Rasch model, we ran StataStan for 10 chains (in series) of 5,000 iterations (first half as warm-up) in 16.6 minutes; at that point, $\widehat{R}$ was less than 1.01 for all parameters. We ran `bayesmh` for 10 chains of the same length in 15.9 minutes; $\widehat{R}$ was less than 1.01 for all parameters. Convergence appears satisfactory for both. In figure 1, we compare values of time per effective independent sample for all the parameters in box plots between StataStan and `bayesmh`. Table 1 provides the same all-in timing statistics for the hyperparameters.

Table 1. Efficiency statistics for the hyperparameters in the two models

Model	Parameter	Stata 14.1 `bayesmh` $n_{\text{eff}}/\text{sec}$	StataStan $n_{\text{eff}}/\text{sec}$
Rasch	σ^2	1.44	8.99
Hierarchical Rasch	μ	3.80	1.22
Hierarchical Rasch	σ^2	1.63	5.62
Hierarchical Rasch	τ^2	3.28	2.66

Results for the hierarchical Rasch model parallel those for the Rasch model. Estimation with StataStan required 24.1 minutes for the same number of chains and iterations, and $\widehat{R}$ was less than 1.01 for all parameters. `bayesmh` ran for 16.6 minutes and yielded values of $\widehat{R}$ less than 1.01 for all parameters. Both estimations appear to have converged.

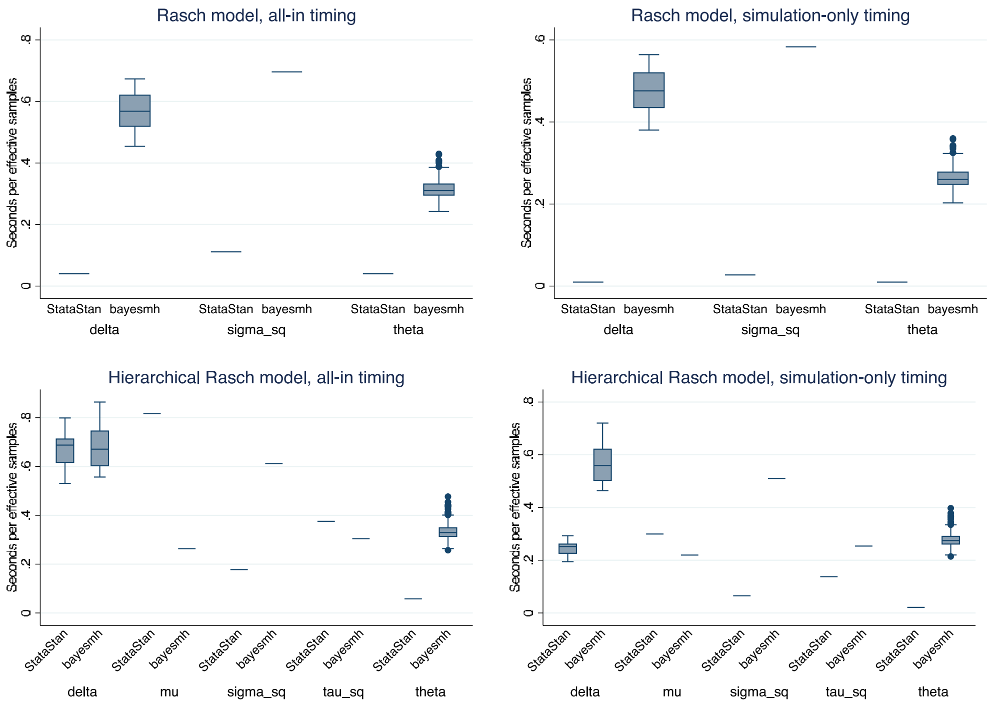


Figure 1. Box plots of seconds per effective independent sample for parameters in the Rasch model (top row of plots) and hierarchical Rasch model (bottom row), in each case fit to simulated data on 500 persons each answering 20 items. Left column shows total timing, including compilation and simulation; right column shows simulation time only. When a model is being fit multiple times, simulation-only timing is a more relevant comparison because the model needs to be compiled only once.

In terms of the total time from issuing the command to its completion, StataStan was more efficient for all parameters in the Rasch model; in the hierarchical Rasch model, it was more efficient for all θ's and σ^2, similar for the δ's, slightly less efficient for τ^2, and less efficient for μ. When we compared simulation-only time (not counting compilation, model building, or warm-up), StataStan's efficiency was improved, making all Rasch parameters even more favorable, and all hierarchical Rasch parameters except μ favor StataStan over `bayesmh`.

When we ran the models with the preferred StataStan priors and with sampling standard deviations rather than variances, results did not change much. Total computation time was somewhat faster at 11.1 minutes for the Rasch model and 22.6 minutes for the hierarchical Rasch model, but times per n_{eff} were very similar at 0.08 seconds for Rasch σ, 0.16 seconds for hierarchical Rasch σ, and 0.29 seconds for τ. However, the efficiency of μ improved to 0.49 seconds per n_{eff}.

Table 2. Efficiency statistics for hyperparameters in increasingly large models

			Total time		Simulation-only time	
			`bayesmh`	StataStan	`bayesmh`	StataStan
Model	Parameter	P	sec/n_{eff}	sec/n_{eff}	sec/n_{eff}	sec/n_{eff}
Rasch	σ^2	100	0.143	0.069	0.137	0.007
		500	0.536	0.105	0.502	0.025
		1,000	1.460	0.230	1.319	0.062
		5,000	9.333	1.649	6.404	0.576
		10,000	350.164	4.539	334.916	1.487
H. Rasch	μ	100	0.212	0.168	0.204	0.023
		500	0.211	0.760	0.197	0.287
		1,000	0.457	1.131	0.413	0.571
		5,000	2.682	22.025	1.847	11.331
		10,000	49.533	67.812	46.660	37.400
H. Rasch	σ^2	100	0.146	0.061	0.140	0.008
		500	0.595	0.177	0.558	0.067
		1,000	1.809	0.340	1.634	0.172
		5,000	11.941	4.508	8.225	2.319
		10,000	186.637	13.236	175.813	7.300
H. Rasch	τ^2	100	0.094	0.095	0.090	0.013
		500	0.350	0.385	0.328	0.145
		1,000	0.904	0.608	0.817	0.307
		5,000	5.145	8.237	3.544	4.237
		10,000	76.556	26.884	72.116	14.827

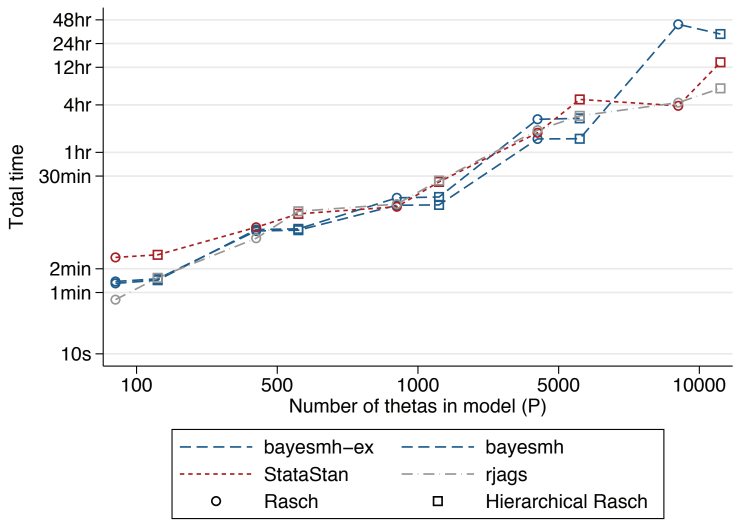


Figure 2. Total time per effective independent sample (worst efficiency across all parameters) in increasingly large Rasch and hierarchical Rasch models

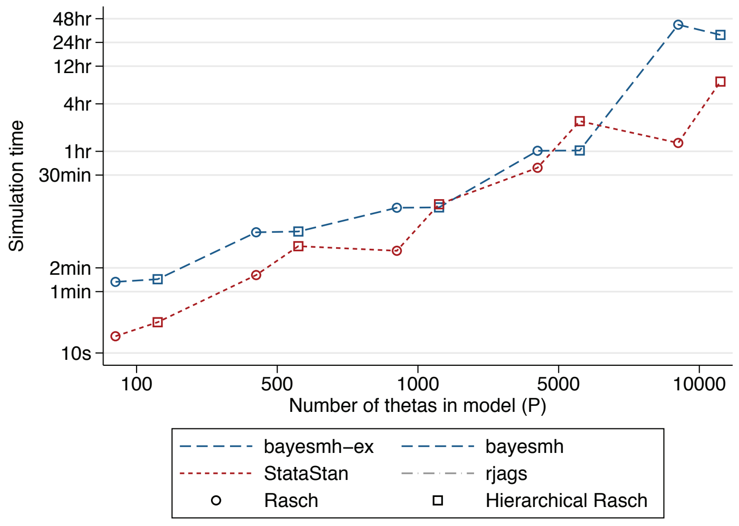


Figure 3. Simulation time per effective independent sample (worst efficiency across all parameters) in increasingly large Rasch and hierarchical Rasch models

In testing with increasingly large models, all three packages showed similar total execution times. StataStan had faster simulation-only time in all Rasch models (from 3% to 61% of the `bayesmh` times) and mixed results in hierarchical Rasch models (from 26% to 235%). The charts show the efficiency for the parameter with the lowest $n_{\rm eff}$ in each case, for total time (figure 2) and simulation-only time (figure 3). In these line charts, the form of `bayesmh` that uses the `exclude` option is denoted by `bayesmh-ex`; we found this had similar speed and efficiency to `bayesmh` without `exclude` and so did not assess it further in the most time-consuming models ($P = 10000$) or in terms of simulation-only time. JAGS does not provide simulation-only timings. In total time per $n_{\rm eff}$, no one software option dominated, though from this limited simulation, it appeared that StataStan was more efficient in the smallest models and `bayesmh` was more efficient in the largest (figure 2). In simulation-only time per $n_{\rm eff}$, StataStan was more efficient than `bayesmh`, up to 10 times more so in most models and sizes of P. However, in some cases, they were similar, with `bayesmh` being slightly better (figure 3).

StataStan consistently had the better efficiency ($s/n_{\rm eff}$) for σ^2 in both Rasch and hierarchical Rasch models but mixed results for τ^2, although four out of five models favored StataStan in simulation-only time (table 2). In the Rasch model, StataStan was 2.1 to 77.1 times faster than `bayesmh` to achieve the same $n_{\rm eff}$ for σ^2 in total time and 11.1 to 225.2 times faster in simulation-only time. In the hierarchical Rasch model, StataStan was 2.4 to 14.1 times faster for σ^2 in total time and 3.5 to 24.1 times faster in simulation-only time. StataStan was 0.6 to 2.8 times faster for τ^2 in total time and 0.8 to 6.9 times faster in simulation-only time. The μ hyperparameter in the hierarchical Rasch models was more efficiently sampled by `bayesmh` at most values of P, with StataStan being 0.1 to 1.3 times faster in total time and 0.2 to 8.9 times faster in simulation-only time (table 2). All models, with all software, could be fit with the same laptop computer without running out of memory.

The random-effects option in `bayesmh` provided a considerable improvement in both effective sample size and speed. When we ran the $I = 20, P = 100$ models without random effects, total time was 206 seconds for Rasch and 211 seconds for hierarchical Rasch, while simulation-only times were 200 and 204 seconds, respectively, which is about 2.5 times slower than the same model with random effects. The time per effective independent sample was considerably increased. In the Rasch model, it rose from 0.143 to 69 seconds for σ^2. In the hierarchical Rasch model, it rose from 0.146 to 30 seconds for σ^2, from 0.212 to 53 seconds for μ, and from 0.094 to 23 seconds for τ^2.

A further consideration is the speed-up obtained by running StataStan chains in parallel even without Stata/MP. We found that the Rasch model with $I = 20$ and $P = 1000$ had total time 383 seconds running in parallel compared with 734 seconds running in series and simulation-only time of 78 seconds compared with 198 seconds. The hierarchical Rasch model of the same size had total time 850 seconds compared with 1,520 seconds and simulation-only time of 303 seconds compared with 768 seconds. This would make parallel StataStan on a quad-core computer roughly twice as efficient as serial StataStan, while `bayesmh` will not run parallel chains without Stata/MP.

5 Discussion

We found that most of the Rasch models we compared were more efficiently sampled by StataStan than `bayesmh` and that this was more favorable to StataStan because the fixed overhead of compiling the model into an executable file was outgrown by the simulation. This suggests that longer chains of draws from the posterior distribution, precompiled models, and parallel chains will all favor StataStan, and the total time comparisons here represent a worst-case scenario for StataStan. We would expect the results we found to apply to generalized linear mixed models, given that these include the Rasch models as a special case (Rijmen et al. 2003; Zheng and Rabe-Hesketh 2007). In practice, the adaptive Markov chain Monte Carlo algorithm featured in `bayesmh` (Stata 14.1) also has a number of features that improve its performance notably over JAGS or Stata 14.0. We found that the same models without the `reffects` option took 200 to 500 times longer to achieve the same effective sample size on `bayesmh`, which users should keep in mind when considering models outside the Rasch family and without random effects.

StataStan provides a simple interface, operating by writing specified variables (as vectors), matrices, and scalars from Stata to a text file and calling the command-line implementation of Stan. The user can specify a wide variety of priors, and the algorithm is less sensitive to the prior than that used in `bayesmh`. In these Rasch models, we found it simple and more intuitive to sample the hyperparameters as standard deviations rather than variances or precisions, and we used uniform priors as the Stan default without any impact on efficiency. We give the alternative programs in the appendix. Progress is displayed inside Stata (even under Windows), and there is the option to write the Stan model inside a comment block in the Stata do-file. Results can then be read back into Stata for diagnostics, generating other values of interest, or saving in `.dta` format. StataStan can be installed from Statistical Software Components by typing

```
ssc install stan
```

Windows users should also type

```
ssc install windowsmonitor
```

In conclusion, we find StataStan to be generally faster than `bayesmh`, which is no surprise given Stan's advanced algorithms and efficient autodifferentiation code. Given that Stan is open source, offers a wider range of models than `bayesmh`, and can be run directly from Stata using StataStan, we recommend that Stata users interested in Bayesian methods consider StataStan for Bayesian modeling, especially for more complicated models.

6 Acknowledgment

We thank the Institute of Education Sciences for partial support of this work.

7 References

Gelman, A. 2006. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis* 1: 515–534.

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, eds. 2013. *Bayesian Data Analysis*. 3rd ed. Boca Raton, FL: CRC Press.

Grant, R. L. 2015. StataStan. https://github.com/stan-dev/statastan.

Hoffman, M. D., and A. Gelman. 2014. The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* 15: 1593–1623.

Kucukelbir, A., R. Ranganath, A. Gelman, and D. M. Blei. 2015. Automatic variational inference in Stan. ArXiv Working Paper No. arXiv:1506.03431. https://arxiv.org/abs/1506.03431.

Neal, R. M. 2011. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, ed. S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, 113–162. Boca Raton, FL: Chapman & Hall/CRC.

Nocedal, J., and S. J. Wright. 2006. *Numerical Optimization*. 2nd ed. Berlin: Springer.

Plummer, M. 2007. JAGS. http://mcmc-jags.sourceforge.net/.

Rasch, G. 1960. *Probabilistic Models for Some Intelligence and Attainment Tests*. Copenhagen: Danmarks Pædagogiske Institut.

Rijmen, F., F. Tuerlinckx, P. De Boeck, and P. Kuppens. 2003. A nonlinear mixed model framework for item response theory. *Psychological Methods* 8: 185–205.

Stan Development Team. 2016. Stan Modeling Language: User's Guide and Reference Manual. http://mc-stan.org/documentation/.

Zheng, X., and S. Rabe-Hesketh. 2007. Estimating parameters of dichotomous and ordinal item response models with gllamm. *Stata Journal* 7: 313–333.

About the authors

Robert Grant is a statistician who runs a training and consultancy startup called BayesCamp. He was senior lecturer in health and social care statistics at Kingston University and St George's, University of London, and previously worked on clinical quality and safety data for the Royal College of Physicians and evidence-based guidelines for the National Institute for Clinical Excellence. Besides Bayesian modeling, he specializes in data visualization.

Daniel Furr is a graduate student in the Graduate School of Education at the University of California, Berkeley. He is the developer of `edstan`, an R package for Bayesian item response theory modeling using Stan, and the author of several related case studies. He specializes in item response theory and predictive methods.

Bob Carpenter is a research scientist in computational statistics at Columbia University. He designed the Stan probabilistic programming language and is one of the Stan core developers. He was professor of computational linguistics (Carnegie Mellon University) and an industrial researcher and programmer in speech recognition and natural language processing (Bell Labs, SpeechWorks, LingPipe) and has written two books on programming language theory and linguistics.

Andrew Gelman is a professor of statistics and professor of political science at Columbia University and the author of several books, including *Bayesian Data Analysis*.

Appendix

Here is the code for the models, starting with the Rasch Stan program, which matches `bayesmh`:

```
data {
  int<lower=1> N;              // number of observations in the dataset
  int<lower=1> I;              // number of items
  int<lower=1> P;              // number of people
  int<lower=1, upper=I> ii[N]; // variable indexing the items
  int<lower=1, upper=P> pp[N]; // variable indexing the people
  int<lower=0, upper=1> y[N];  // binary outcome variable
}
parameters {
  real<lower=0> sigma_sq; // variance of the thetas (random intercepts for people)
  vector[I] delta_unit;   // normalized deltas
  vector[P] theta_unit;   // normalized thetas
}
transformed parameters {
  real<lower=0> sigma;
  sigma = sqrt(sigma_sq); // SD of the theta random intercepts
}
model {
  vector[I] delta;
  vector[P] theta;
  theta_unit ~ normal(0, 1);     // prior for normalized thetas
  delta_unit ~ normal(0, 1);     // prior for normalized deltas
  sigma_sq ~ inv_gamma(1, 1);    // prior for variance of thetas
  theta = theta_unit * sigma;    // convert normalized thetas to thetas (mean 0)
  delta = delta_unit * sqrt(10); // convert normalized deltas to deltas (mean 0)
  y ~ bernoulli_logit(theta[pp] - delta[ii]); // likelihood
}
```

This is our preferred Stan program:

```
data {
  int<lower=1> N;              // number of observations in the dataset
  int<lower=1> I;              // number of items
  int<lower=1> P;              // number of people
  int<lower=1, upper=I> ii[N]; // variable indexing the items
  int<lower=1, upper=P> pp[N]; // variable indexing the people
  int<lower=0, upper=1> y[N];  // binary outcome variable
}
parameters {
  real<lower=0> sigma;  // SD of the thetas (random intercepts for people)
  vector[I] delta_unit; // normalized deltas
  vector[P] theta_unit; // normalized thetas
}
model {
  vector[I] delta;
  vector[P] theta;
  theta_unit ~ normal(0, 1);     // prior for normalized thetas
  delta_unit ~ normal(0, 1);     // prior for normalized deltas
  theta = theta_unit * sigma;    // convert normalized thetas to thetas (mean 0)
  delta = delta_unit * sqrt(10); // convert normalized deltas to deltas (mean 0)
  y ~ bernoulli_logit(theta[pp] - delta[ii]); // likelihood
}
```

Here is the Stata call for the Rasch model:

```
bayesmh y=({theta:}-{delta:}), likelihood(logit)           ///
   redefine(delta:i.item) redefine(theta:i.person)         ///
   prior({theta:i.person}, normal(0, {sigmasq}))           ///
   prior({delta:i.item}, normal(0, 10))                    ///
   prior({sigmasq}, igamma(1, 1))                          ///
   mcmcsize(`mcmcsize´) burnin(`burnin´)                   ///
   notable saving(`draws´, replace) dots                   ///
   initial({delta:i.item} `=el(inits`jj´, `c´, 1)´         ///
      {theta:i.person} `=el(inits`jj´, `c´, 2)´            ///
      {sigmasq} `=el(inits`jj´, `c´, 3)´)                  ///
   block({sigmasq})
```

And here is the JAGS code for the Rasch model:

```
model {
  for (i in 1:I) {
    delta[i] ~ dunif(-1e6, 1e6)
  }
  inv_sigma_sq ~ dgamma(1,1)
  sigma <- pow(inv_sigma_sq, -0.5)
  for (p in 1:P) {
    theta[p] ~ dnorm(0, inv_sigma_sq)
  }
  for (n in 1:N) {
    logit(inv_logit_eta[n]) <- theta[pp[n]] - delta[ii[n]]
    y[n] ~ dbern(inv_logit_eta[n])
  }
}
```

Here is the hierarchical Rasch model in Stan, matching `bayesmh`:

```
data {
  int<lower=1> N;              // number of observations in the dataset
  int<lower=1> I;              // number of items
  int<lower=1> P;              // number of people
  int<lower=1, upper=I> ii[N]; // variable indexing the items
  int<lower=1, upper=P> pp[N]; // variable indexing the people
  int<lower=0, upper=1> y[N];  // binary outcome variable
}
parameters {
  real<lower=0> sigma_sq; // variance of the thetas (random intercepts for people)
  real<lower=0> tau_sq;   // variance of the deltas (random intercepts for items)
  real mu;                // mean of the deltas
  vector[I] delta_unit;   // normalized deltas
  vector[P] theta_unit;   // normalized thetas
}
transformed parameters {
  real<lower=0> sigma;
  real<lower=0> tau;
  sigma = sqrt(sigma_sq); // SD of the theta random intercepts
  tau = sqrt(tau_sq);     // SD of the delta random intercepts
}
model {
  vector[I] delta;
  vector[P] theta;
  theta_unit ~ normal(0, 1);       // prior for normalized thetas
  delta_unit ~ normal(0, 1);       // prior for normalized deltas
  mu ~ normal(0, sqrt(10));        // prior for the mean of the deltas
  sigma_sq ~ inv_gamma(1, 1);
  tau_sq ~ inv_gamma(1, 1);
  theta = theta_unit * sigma;      // convert normalized thetas to thetas (mean 0)
  delta = mu + (delta_unit * tau); // convert normalized deltas to deltas (mean mu)
  y ~ bernoulli_logit(theta[pp] - delta[ii]); // likelihood
}
```

This is our preferred Stan model:

```
data {
  int<lower=1> N;              // number of observations in the dataset
  int<lower=1> I;              // number of items
  int<lower=1> P;              // number of people
  int<lower=1, upper=I> ii[N]; // variable indexing the items
  int<lower=1, upper=P> pp[N]; // variable indexing the people
  int<lower=0, upper=1> y[N];  // binary outcome variable
}
parameters {
  real<lower=0> sigma;  // SD of the thetas (random intercepts for people)
  real<lower=0> tau;    // SD of the deltas (random intercepts for items)
  real mu;              // mean of the deltas
  vector[I] delta_unit; // normalized deltas
  vector[P] theta_unit; // normalized thetas
}
model {
  vector[I] delta;
  vector[P] theta;
  theta_unit ~ normal(0, 1);       // prior for normalized thetas
  delta_unit ~ normal(0, 1);       // prior for normalized deltas
  mu ~ normal(0, sqrt(10));        // prior for the mean of the deltas
  theta = theta_unit * sigma;      // convert normalized thetas to thetas (mean 0)
  delta = mu + (delta_unit * tau); // convert normalized deltas to deltas (mean mu)
  y ~ bernoulli_logit(theta[pp] - delta[ii]); // likelihood
}
```

Here is the Stata call for the hierarchical Rasch model:

```
bayesmh y=({theta:}-{delta:}),likelihood(logit)           ///
   redefine(delta:i.item) redefine(theta:i.person)        ///
   prior({theta:i.person}, normal(0, {sigmasq}))          ///
   prior({delta:i.item}, normal({mu}, {tausq}))           ///
   prior({mu}, normal(0, 10))                             ///
   prior({sigmasq} {tausq}, igamma(1, 1))                 ///
   block({sigmasq} {tausq} {mu}, split)                   ///
   initial({delta:i.item} `=el(inits`jj´, 1, 1)´          ///
      {theta:i.person} `=el(inits`jj´, 1, 2)´             ///
      {sigmasq} `=el(inits`jj´, 1, 3)´                    ///
      {tausq} `=el(inits`jj´, 1, 4)´                      ///
      {mu} `=el(inits`jj´, 1, 5)´)                        ///
   mcmcsize(`mcmcsize´) burnin(`burnin´)                  ///
   saving(`draws´, replace) dots
```

And here is the JAGS code for the hierarchical Rasch model:

```
model {
  inv_sigma_sq ~ dgamma(1,1)
  sigma <- pow(inv_sigma_sq, -0.5)
  for (p in 1:P) {
    theta[p] ~ dnorm(0, inv_sigma_sq)
  }
  inv_tau_sq ~ dgamma(1,1)
  tau <- pow(inv_tau_sq, -0.5)
  for (i in 1:I) {
    delta[i] ~ dnorm(0, inv_tau_sq)
  }
  mu ~ dunif(-1e6, 1e6)
  for (n in 1:N) {
    logit(inv_logit_eta[n]) <- mu + theta[pp[n]] - delta[ii[n]]
    y[n] ~ dbern(inv_logit_eta[n])
  }
}
```

The Stata Journal (2017)
17, Number 2, pp. 358–371

Instantaneous geometric rates via generalized linear models

Andrea Discacciati
Karolinska Institutet
Stockholm, Sweden
andrea.discacciati@ki.se

Matteo Bottai
Karolinska Institutet
Stockholm, Sweden
matteo.bottai@ki.se

Abstract. The instantaneous geometric rate represents the instantaneous probability of an event of interest per unit of time. In this article, we propose a method to model the effect of covariates on the instantaneous geometric rate with two models: the proportional instantaneous geometric rate model and the proportional instantaneous geometric odds model. We show that these models can be fit within the generalized linear model framework by using two nonstandard link functions that we implement in the user-defined link programs `log_igr` and `logit_igr`. We illustrate how to fit these models and how to interpret the results with an example from a randomized clinical trial on survival in patients with metastatic renal carcinoma.

Keywords: st0478, log_igr, logit_igr, instantaneous geometric rate, generalized linear models, glm, survival analysis

1 Introduction

The geometric rate represents the average probability of an event of interest per unit of time over a specific time interval. Recently, Bottai (Forthcoming) showed that in the case of events that occur only once, such as death or first diagnosis of a disease, the geometric rate is a better measure of occurrence than the incidence rate. In the same article, Bottai proposed a regression method to model the conditional geometric rate given covariates. That method is based on applying quantile regression to a transformation of the time variable and is implemented in the user-written `grreg` command (Bottai 2015).

As the length of the time interval over which the geometric rate is defined shrinks to zero, we obtain the instantaneous geometric rate. This measure has a very intuitive interpretation because it represents the instantaneous probability of the event per unit of time.

In this article, we propose two models for the effect of covariates on the instantaneous geometric rate: the proportional instantaneous geometric rate model and the proportional instantaneous geometric odds model. We show that these models can be fit within the generalized linear model (GLM) framework (Nelder and Wedderburn 1972) by using two nonstandard link functions that can be easily programmed into the official Stata `glm` command (Guan and Gutierrez 2002).

The remainder of this article is organized as follows: In section 2, we briefly review how the instantaneous geometric rate is defined. In section 3, we show how to model the instantaneous geometric rate via GLM and present two user-defined link programs, `log_igr` and `logit_igr`. In section 4, we use data from a randomized clinical trial to illustrate some practical examples of how these link programs can be specified as an option of the `glm` command and how to interpret and present the analysis results. In section 5, we provide a summary.

2 Geometric rate and instantaneous geometric rate

In this section, we follow the description provided by Bottai (Forthcoming). Let T be a continuous random variable with support on $(0, +\infty)$ representing the time-to-event of individuals in some population, and let $S(t)$ be the associated survival function. The geometric rate over the time interval $(0, t)$ is defined as

$$g(0,t) = 1 - S(t)^{\frac{1}{t}}$$

and represents the average probability of the event per unit of time over $(0, t)$. The geometric rate between any two time points t_1 and t_2, such that $0 < t_1 < t_2 < +\infty$, is

$$g(t_1,t_2) = 1 - \left\{\frac{S(t_2)}{S(t_1)}\right\}^{\frac{1}{t_2-t_1}}$$

The limit of the geometric rate over shrinking time intervals $(t, t + \Delta t)$ gives the instantaneous geometric rate

$$\begin{aligned}
g(t) &\equiv \lim_{\Delta t \to 0^+} g(t, t+\Delta t) \\
&= \lim_{\Delta t \to 0^+} 1 - \left\{\frac{S(t+\Delta t)}{S(t)}\right\}^{\frac{1}{\Delta t}} \\
&= \lim_{\Delta t \to 0^+} 1 - \exp\left\{\frac{\log S(t+\Delta t) - \log S(t)}{\Delta t}\right\} \\
&= 1 - \exp\left\{\frac{\partial \log S(t)}{\partial t}\right\} \\
&= 1 - \exp\left\{-\frac{f(t)}{S(t)}\right\} \\
&= 1 - \exp\{-h(t)\} \qquad (1)
\end{aligned}$$

where $f(t)$ indicates the probability density function of T and $h(t) \equiv f(t)/S(t)$, the hazard function. The instantaneous geometric rate represents the instantaneous probability of the event per unit of time.

3 Instantaneous geometric rates via GLM

In this section, we show how instantaneous geometric rates can be estimated by GLM using nonstandard link functions. See Hardin and Hilbe (2012) for an exposition of GLM specifically targeted at Stata users.

Let t_i, $i = 1, \ldots, n$, be a sample of n possibly censored observations on the time variable, d_i be the event indicator variable (0 for a censored observation, 1 for an event), $\boldsymbol{x}_i = (x_{1,i} \ldots x_{p,i})'$ be a vector of covariates, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)'$ be an unknown parameter vector.

3.1 Proportional instantaneous geometric rate model

We consider the proportional instantaneous geometric rates model

$$g_i(t|\boldsymbol{x}_i) = g_0(t) \exp(\boldsymbol{x}_i'\boldsymbol{\beta}) \tag{2}$$

By taking the logarithm of both sides of (2), we get

$$\log\{g_i(t|\boldsymbol{x}_i)\} = \log\{g_0(t)\} + \boldsymbol{x}_i'\boldsymbol{\beta}$$

and by taking the logarithm of (1), we get

$$\log[1 - \exp\{-h_i(t)\}|\boldsymbol{x}_i] = s(t;\boldsymbol{\gamma}) + \boldsymbol{x}_i'\boldsymbol{\beta} \tag{3}$$

where $s(t;\boldsymbol{\gamma})$ is a smooth parametric function of analysis time that depends on a vector of unknown parameters $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_r)'$.

To model the baseline log instantaneous geometric rate via $s(t;\boldsymbol{\gamma})$, we split each individual's follow-up into a number of intervals (or episodes) by choosing a fine grid of split points. After splitting the follow-up, let t_{ij} be the length of the jth time interval (the time at risk) relative to the ith individual, and let d_{ij} be the event indicator that takes value 1 if individual i develops the event in interval j, and 0 otherwise.

Following the same rationale behind parametric proportional hazard models (Royston and Lambert 2011, chaps. 4 and 7), (3) suggests using the following link function,

$$\eta_{ij} \equiv k(\mu_{ij}) = \log\left\{1 - \exp\left(-\frac{\mu_{ij}}{t_{ij}}\right)\right\} \tag{4}$$

where μ_{ij} is the expected value of d_{ij}, which is assumed to follow a distribution of the exponential family.

After suppressing the subscripts, the calculations to program the link function (4) are

$$
\begin{aligned}
\mu = k^{-1}(\eta) &= -t \log\left\{-\exp\left(\eta\right)+1\right\} \\
\frac{\partial \mu}{\partial \eta} &= t \exp(\eta)\left\{-\exp(\eta)+1\right\}^{-1} \\
\frac{\partial^2 \mu}{\partial \eta^2} &= t \exp(\eta)\left\{\exp(\eta)-1\right\}^{-2}
\end{aligned}
\tag{5}
$$

The following is the link program `log_igr`, contained in the `log_igr.ado` ado-file:

```
*! version 1.0.0 - 07dec2016
capture program drop log_igr
program define log_igr
        version 7
        args todo eta mu return

        if `todo´ == -1 { /* Title */
                global SGLM_lt "Log IGR"
                global SGLM_lf "log(1-exp(-u/$SGLM_p))"
                capture confirm numeric variable $SGLM_p
                        if _rc != 0 {
                                noi di as error "argument ($SGLM_p) to log_igr " /*
                                */ "link function must be a numeric variable"
                                exit 198
                        }
                exit
        }
        if `todo´ == 0 { /* eta = g(mu) */
                gen double `eta´ = log(-exp(-`mu´/$SGLM_p)+1)
                exit
        }
        if `todo´ == 1 { /* mu = g^-1(eta) */
                gen double `mu´ = -$SGLM_p*log(-exp(`eta´)+1)
                exit
        }
        if `todo´ == 2 { /* (d mu)/(d eta) */
                gen double `return´ = $SGLM_p*exp(`eta´)*(-exp(`eta´)+1)^(-1)
                exit
        }
        if `todo´ == 3 { /* (d^2 mu)/(d eta^2) */
                gen double `return´ = $SGLM_p*exp(`eta´)*(exp(`eta´)-1)^(-2)
                exit
        }
        noi di as err "Unknown call to glm link function"
        exit 198
end
```

To use this link, specify the `link(log_igr` *varname*`)` option in the `glm` command, where the existing numeric variable *varname* contains the time at risk, t_{ij}. See Guan and Gutierrez (2002) for a detailed explanation of how to program a custom link function.

3.2 Proportional instantaneous geometric odds model

We now consider the proportional instantaneous geometric odds model

$$\frac{g_i(t|\boldsymbol{x}_i)}{1-g_i(t|\boldsymbol{x}_i)} = \frac{g_0(t)}{1-g_0(t)} \exp\left(\boldsymbol{x}_i'\boldsymbol{\beta}\right) \tag{6}$$

As we did in section 3.1, we write

$$\text{logit}[1-\exp\{-h_i(t)\}|\boldsymbol{x}_i] = s(t;\boldsymbol{\gamma}) + \boldsymbol{x}_i'\boldsymbol{\beta}$$

Therefore, the second proposed nonstandard link function is

$$\eta_{ij} \equiv k(\mu_{ij}) = \text{logit}\left\{1-\exp\left(-\frac{\mu_{ij}}{t_{ij}}\right)\right\}$$

and the necessary calculations to program it are

$$\begin{aligned}\mu = k^{-1}(\eta) &= -t\log[\{\exp(\eta)+1\}^{-1}] \\ \frac{\partial\mu}{\partial\eta} &= t\exp(\eta)\left\{\exp(\eta)+1\right\}^{-1} \\ \frac{\partial^2\mu}{\partial\eta^2} &= t\exp(\eta)\left\{\exp(\eta)+1\right\}^{-2}\end{aligned}$$

The following is the content of the `logit_igr.ado` ado-file, which contains the link program `logit_igr`:

```
*! version 1.0.0 - 07dec2016
capture program drop logit_igr
program define logit_igr
        version 7
        args todo eta mu return

        if `todo´ == -1 { /* Title */
                global SGLM_lt "Logit IGR"
                global SGLM_lf "logit(1-exp(-u/$SGLM_p))"
                confirm numeric variable $SGLM_p
                       if _rc != 0 {
                             noi di as error "argument ($SGLM_p) to logit_igr " /*
                             */ "link function must be a numeric variable"
                             exit 198
                       }
                exit
        }
        if `todo´ == 0 { /* eta = g(mu) */
                gen double `eta´ = logit(1-exp(-`mu´/$SGLM_p))
                exit
        }
        if `todo´ == 1 { /* mu = g^-1(eta) */
                gen double `mu´ = -$SGLM_p*log((exp(`eta´)+1)^(-1))
                exit
        }
```

```
            if `todo´ == 2 { /* (d mu)/(d eta) */
                    gen double `return´ = $SGLM_p*exp(`eta´)*(exp(`eta´)+1)^(-1)
                    exit
            }
            if `todo´ == 3 { /* (d^2 mu)/(d eta^2) */
                    gen double `return´ = $SGLM_p*exp(`eta´)*(exp(`eta´)+1)^(-2)
                    exit
            }
            noi di as err "Unknown call to glm link function"
            exit 198
    end
```

Some notes are as follows:

1. Both models can easily accommodate time-varying covariates and time-dependent coefficients.
2. In (2), the exponentiated coefficients $\exp(\beta)$ are interpreted as instantaneous geometric rate ratios (IGRR), whereas in (6), they are interpreted as instantaneous geometric odds ratios (IGOR).
3. If the instantaneous geometric rates are proportional across different populations, the instantaneous geometric odds are not, and vice versa.
4. The inverse link function (5) is defined only for $\eta < 0$. This has two practical consequences. First, the default initial values $(\boldsymbol{\gamma}_0, \boldsymbol{\beta}_0) = (0, 0, \ldots, 0)$ used for the maximization of the log likelihood (Gould, Pitblado, and Poi 2010) are not feasible, because the log likelihood cannot be evaluated in $(\boldsymbol{\gamma}_0, \boldsymbol{\beta}_0)$. This can be solved by passing feasible initial values to `glm` or by specifying the `search` option (see [R] **maximize**). Second, the parameter space for $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ is bounded, which means the log likelihood is defined only within that parameter space. This introduces challenges in maximizing the log likelihood and may lead to failed convergence of the optimization algorithms, similarly to what happens to binomial models with a log link (Williamson, Eliasziw, and Fick 2013).

4 Example: Survival in metastatic renal carcinoma

We illustrate the use of the two proposed regression models using data from a clinical trial on 347 patients diagnosed with metastatic renal carcinoma (Medical Research Council Renal Cancer Collaborators 1999). The patients were randomly assigned to either interferon-α (IFN) or oral medroxyprogesterone (MPA). A total of 322 patients died during follow-up.

4.1 Data preparation

The numeric variable `survtime` represents the time in days to death or censoring, the binary variable `cens` indicates the death status (`0` = censored, `1` = death), and the variable `pid` contains the unique patient identifier.

First, we declare the data to be survival-time data with the `stset` command, and we rescale the analysis time from days to years with the `scale(365.24)` option.

Next, we split each patient's follow-up in intervals of length equal to one week using the `stsplit` command with the `every(‘=1/52’)` option and generate a new variable containing the time at risk within each interval (`risktime`).

Last, to model the baseline instantaneous geometric rate, we generate restricted cubic spline (RCS) transformations of analysis time, using the user-written `rcsgen` command (Lambert 2008). We use four knots, which by default are located at the minimum, maximum, and the 33rd and 66th centiles of the uncensored survival times' distribution. To do so, we add the `df(3)` and `if2(_d == 1)` options.

```
. use http://www.imm.ki.se/biostatistics/data/kidney
(Metastatic renal carcinoma trial. MRCRCC. Lancet. 1999, 353:14-7)

. stset survtime, failure(cens) id(pid) scale(365.24)

                id:  pid
     failure event:  cens != 0 & cens < .
obs. time interval:  (survtime[_n-1], survtime]
 exit on or before:  failure
    t for analysis:  time/365.24
------------------------------------------------------------------------------
        347  total observations
          0  exclusions
------------------------------------------------------------------------------
        347  observations remaining, representing
        347  subjects
        322  failures in single-failure-per-subject data
    375.687  total analysis time at risk and under observation
                                                at risk from t =         0
                                     earliest observed entry t =         0
                                          last observed exit t =  6.209616

. stsplit click, every(`=1/52´)
(19,360 observations (episodes) created)

. generate risktime = _t - _t0

. rcsgen _t, df(3) if2(_d == 1) gen(_rcs)
Variables _rcs1 to _rcs3 were created
```

4.2 Proportional instantaneous geometric rates model

We fit a proportional instantaneous geometric rates model with the `glm` command with the `log_igr` custom link program. The `risktime` variable, which contains t_{ij}, is passed as an argument to `log_igr`.

We start by including the binary treatment indicator (`trt`) and the RCS transformations of analysis time (`_rcs1`, `_rcs2`, and `_rcs3`) in the model. The outcome variable `_d` contains the event indicator d_{ij}.

```
. glm _d i.trt c._rcs?, family(poisson) link(log_igr risktime) vce(robust) nolog
> search eform
initial:       log pseudolikelihood =     -<inf>  (could not be evaluated)
feasible:      log pseudolikelihood = -4804.4455
rescale:       log pseudolikelihood = -1959.6083

Generalized linear models                         No. of obs      =     19,707
Optimization     : ML                             Residual df     =     19,702
                                                  Scale parameter =          1
Deviance         =    3239.4169                   (1/df) Deviance =   .1644207
Pearson          =  124086.9279                   (1/df) Pearson  =   6.298189

Variance function: V(u) = u                       [Poisson]
Link function    : g(u) = log(1-exp(-u/risktime)) [Log IGR]

                                                  AIC             =   .1975652
Log pseudolikelihood =  -1941.70845               BIC             =  -191588.3

------------------------------------------------------------------------------
             |               Robust
          _d |     exp(b)   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         trt |
        IFN  |   .8371623   .0568225    -2.62   0.009     .7328824    .9562799
       _rcs1 |   .9604894   .2909327    -0.13   0.894     .5304729    1.739089
       _rcs2 |   1.308916   .8565102     0.41   0.681     .3630067    4.719643
       _rcs3 |   .9010516   .2378547    -0.39   0.693     .5370987    1.511629
       _cons |   .7243848   .0642749    -3.63   0.000     .6087542    .8619789
------------------------------------------------------------------------------
```

The estimated IGRR comparing the two treatment groups (IFN versus MPA) is 0.84 (95% confidence interval: $[0.73, 0.96]$), constant throughout the entire follow-up. Under this model, the instantaneous yearly probability of death in the IFN group was estimated to be 16% lower than in the MPA group. We can predict the log instantaneous geometric death rate for the two treatment groups with the `predict` postestimation command.

```
. predict log_igr, xb
. generate igr = exp(log_igr)
```

In figure 1, we see that the instantaneous yearly risk of dying in patients on MPA decreased from about 75% to 25% over the 6 years of follow-up. Figure 1 also clearly exhibits the assumption of proportional instantaneous geometric rates in that the vertical distance between the two lines (on the log scale) is constant throughout the follow-up.

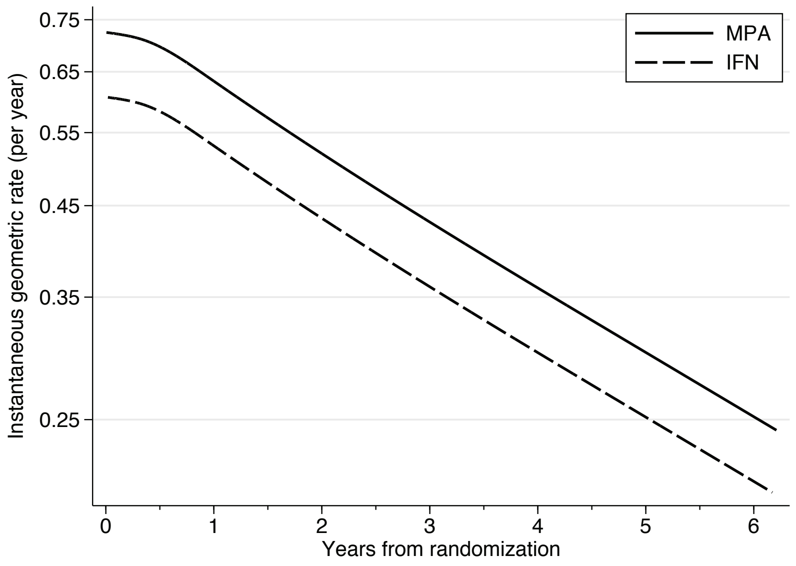


Figure 1. Predicted instantaneous geometric death rates for the two treatment groups from an instantaneous geometric proportional rates model. The vertical axis is on a log scale.

We now relax the assumption of constant IGRR. To do so, we add interactions (product terms) between `trt` and the three RCS transformations of analysis time.

```
. glm _d i.trt##c._rcs?, family(poisson) link(log_igr risktime) vce(robust)
> nolog search
initial:       log pseudolikelihood =     -<inf>  (could not be evaluated)
feasible:      log pseudolikelihood = -4804.4455
rescale:       log pseudolikelihood = -1959.6083

Generalized linear models                         No. of obs      =     19,707
Optimization     : ML                             Residual df     =     19,699
                                                  Scale parameter =          1
Deviance         =  3237.985686                   (1/df) Deviance =   .1643731
Pearson          =  122535.5358                   (1/df) Pearson  =   6.220394

Variance function: V(u) = u                       [Poisson]
Link function    : g(u) = log(1-exp(-u/risktime)) [Log IGR]

                                                  AIC             =    .197797
Log pseudolikelihood = -1940.992843                BIC             =  -191560.1

------------------------------------------------------------------------------
             |               Robust
          _d |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         trt |
         IFN |  -.3683698   .1914883    -1.92   0.054    -.7436799    .0069403
       _rcs1 |  -.3103316   .3603476    -0.86   0.389      -1.0166    .3959368
       _rcs2 |  -.2934956   .8245528    -0.36   0.722    -1.909589    1.322598
       _rcs3 |   .1167311   .3344125     0.35   0.727    -.5387053    .7721675
             |
 trt#c._rcs1 |
         IFN |   .7833079   .6631432     1.18   0.238    -.5164289    2.083045
             |
 trt#c._rcs2 |
         IFN |   1.572779   1.383879     1.14   0.256    -1.139574    4.285131
             |
 trt#c._rcs3 |
         IFN |  -.6156042   .5543412    -1.11   0.267    -1.702093    .4708846
             |
       _cons |  -.2639627   .0874417    -3.02   0.003    -.4353452   -.0925802
------------------------------------------------------------------------------

. predict log_igr, xb

. generate igr = exp(log_igr)

. predictnl log_igrr = _b[1.trt] + _b[1.trt#c._rcs1]*_rcs1 +
> _b[1.trt#c._rcs2]*_rcs2 +  _b[1.trt#c._rcs3]*_rcs3

. generate igrr = exp(log_igrr)
```

The log-time dependent IGRR is obtained with the `predictnl` postestimation command and plotted in figure 2 after exponentiation, together with the instantaneous geometric death rates for the two treatment groups.

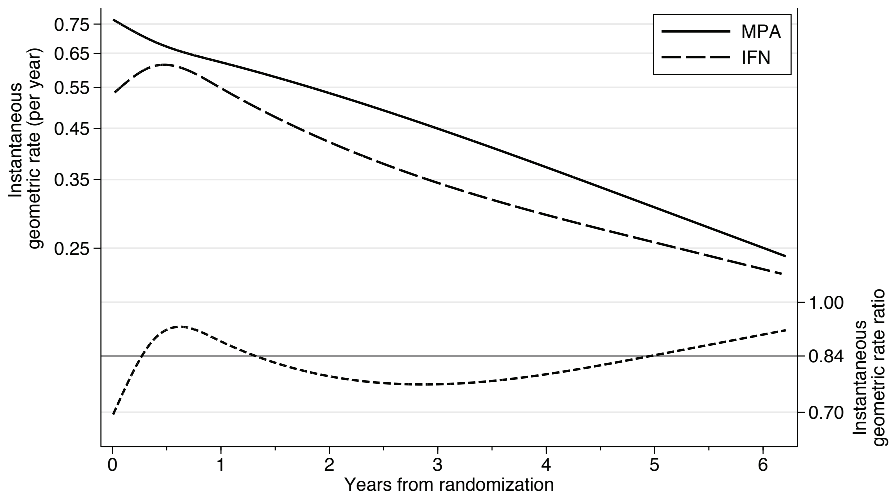


Figure 2. Predicted instantaneous geometric death rates for patients on MPA (solid black line) and IFN (long-dashed black line) and predicted time-dependent IGRR (short-dashed black line) (IFN versus MPA). The gray solid line indicates the time-fixed IGRR, equal to 0.84. The vertical axes are on a log scale.

When we inspect figure 2, it seems the assumption of constant IGRR throughout the follow-up is tenable. We can formally test this assumption by testing the coefficients of the interaction terms to be jointly equal to zero. This can be done with the `testparm` postestimation command.

```
. testparm 1.trt#c._rcs?
 ( 1)  [_d]1.trt#c._rcs1 = 0
 ( 2)  [_d]1.trt#c._rcs2 = 0
 ( 3)  [_d]1.trt#c._rcs3 = 0
            chi2(  3) =    1.43
          Prob > chi2 =    0.6983
```

From this output, we fail to reject the null hypothesis of proportionality of the instantaneous geometric rates (p-value = 0.6983).

4.3 Proportional instantaneous geometric odds model

To illustrate the proportional instantaneous geometric odds model, we now explore whether white cell count (`wcc`), a continuous prognostic factor, affects the treatment effect as measured by the IGOR. This analysis builds upon the findings reported by Royston, Sauerbrei, and Ritchie (2004), where they observed a beneficial effect of IFN—in terms of relative hazard—only among patients with a white cell count lower than about 10×10^9 L^{-1}.

We include the treatment indicator, white cell count, their interaction term, and the three RCS transformations of analysis time as covariates. We specify the option `link(logit_igr risktime)` to fit a proportional instantaneous geometric odds model.

```
. glm _d i.trt##c.wcc _rcs?, family(poisson) link(logit_igr risktime)
> vce(robust) nolog

Generalized linear models                         No. of obs      =     19,707
Optimization     : ML                             Residual df     =     19,700
                                                  Scale parameter =          1
Deviance         =  3210.596989                   (1/df) Deviance =   .1629745
Pearson          =   119282.802                   (1/df) Pearson  =   6.054965

Variance function: V(u) = u                       [Poisson]
Link function    : g(u) = logit(1-exp(-u/risktime))[Logit IGR]

                                                  AIC             =   .1963057
Log pseudolikelihood = -1927.298494               BIC             =  -191597.4

------------------------------------------------------------------------------
             |               Robust
          _d |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         trt |
        IFN  |  -1.674116   .5957372    -2.81   0.005    -2.841739   -.5064921
         wcc |   .0824596   .0453305     1.82   0.069    -.0063865    .1713058
             |
   trt#c.wcc |
        IFN  |   .1620935   .0705864     2.30   0.022     .0237467    .3004403
             |
       _rcs1 |   .7416164   .8740937     0.85   0.396    -.9715757    2.454809
       _rcs2 |   2.101511   1.771603     1.19   0.236    -1.370766    5.573789
       _rcs3 |  -.8266814   .7022011    -1.18   0.239     -2.20297    .5496075
       _cons |  -.0688033   .4756726    -0.14   0.885    -1.001105    .8634979
------------------------------------------------------------------------------
```

Based on the p-value for the interaction term, we reject the null hypothesis of constant treatment effect throughout the observed range of white cell count (p-value = 0.022).

```
. predictnl log_igor = _b[1.trt] + _b[1.trt#c.wcc]*wcc, se(log_igor_se)
. generate igor = exp(log_igor)
. generate igor_lo = exp(log_igor - 1.96*log_igor_se)
. generate igor_hi = exp(log_igor + 1.96*log_igor_se)
```

The log IGOR comparing mortality among patients on IFN and patients on MPA as a function of white cell count can be obtained with the `predictnl` postestimation command and then plotted (figure 3).

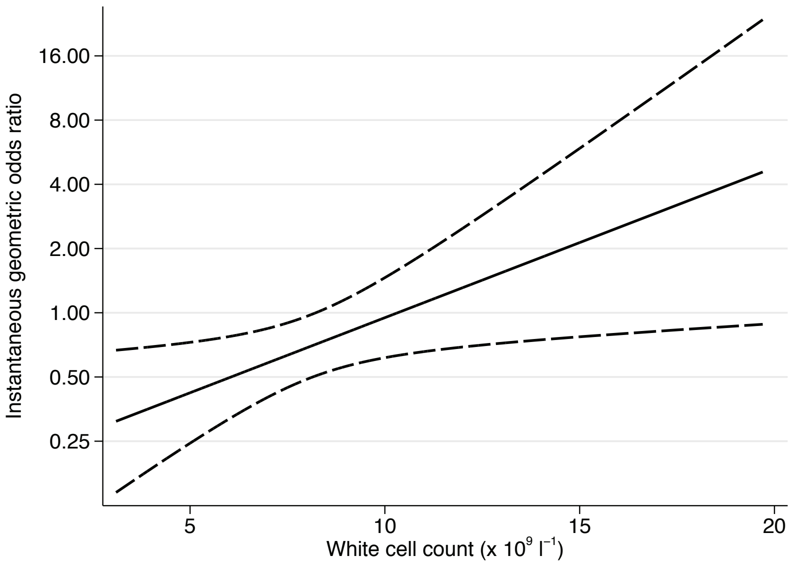


Figure 3. Predicted IGOR for IFN versus MPA (solid line) with 95% confidence interval (long-dashed lines) as a function of white cell count. The vertical axis is on a log scale.

The treatment effect seems to be largest among patients with a low white cell count. For example, the estimated IGOR for white cell counts of 4.9 and 13.7×10^9 L^{-1} (5th and 95th centiles of `wcc` distribution) were 0.40 (95% confidence interval: $[0.23, 0.72]$) and 1.72 (95% confidence interval: $[0.74, 4.04]$), respectively.

5 Summary

In this article, we proposed a method to model the effects of covariates on the instantaneous geometric rate within the GLM framework by using two nonstandard link functions. We showed how these link functions could be easily programmed into the `glm` command by creating two short, independent ado-files, `log_igr.ado` and `logit_igr.ado`.

Using data from a randomized clinical trial on survival in patients with metastatic renal carcinoma, we illustrated how to use these link programs and how to interpret results from the proportional instantaneous geometric rate model and the proportional instantaneous geometric odds model. We also demonstrated that a clear advantage of using `glm` to fit these models is that postestimation commands for `glm` are readily available.

In conclusion, the intuitive interpretation of the instantaneous geometric rate and the ease with which the proposed regression models can be fit in Stata make them a useful addition to the existing tools for the analysis of survival data.

6 References

Bottai, M. 2015. The grreg command for geometric rate regression. 2015 Nordic and Baltic Stata Users Group meeting proceedings. http://www.stata.com/meeting/nordic-and-baltic15/abstracts/materials/sweden15_bottai.pdf.

———. Forthcoming. A regression method for modelling geometric rates. *Statistical Methods in Medical Research*.

Gould, W., J. Pitblado, and B. Poi. 2010. *Maximum Likelihood Estimation with Stata*. 4th ed. College Station, TX: Stata Press.

Guan, W., and R. G. Gutierrez. 2002. Programmable GLM: Two user-defined links. *Stata Journal* 2: 378–390.

Hardin, J. W., and J. M. Hilbe. 2012. *Generalized Linear Models and Extensions*. 3rd ed. College Station, TX: Stata Press.

Lambert, P. 2008. rcsgen: Stata module to generate restricted cubic splines and their derivatives. Statistical Software Components S456986, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s456986.html.

Medical Research Council Renal Cancer Collaborators. 1999. Interferon-α and survival in metastatic renal carcinoma: Early results of a randomised controlled trial. *Lancet* 353: 14–17.

Nelder, J. A., and R. W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society, Series A* 135: 370–384.

Royston, P., and P. C. Lambert. 2011. *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*. College Station, TX: Stata Press.

Royston, P., W. Sauerbrei, and A. Ritchie. 2004. Is treatment with interferon-alpha effective in all patients with metastatic renal carcinoma? A new approach to the investigation of interactions. *British Journal of Cancer* 90: 794–799.

Williamson, T., M. Eliasziw, and G. H. Fick. 2013. Log-binomial models: Exploring failed convergence. *Emerging Themes in Epidemiology* 10: 14.

About the authors

Andrea Discacciati is a biostatistician in the Unit of Biostatistics at the Institute of Environmental Medicine at Karolinska Institutet in Stockholm, Sweden.

Matteo Bottai is a professor of biostatistics in the Unit of Biostatistics at the Institute of Environmental Medicine at Karolinska Institutet in Stockholm, Sweden.

The Stata Journal (2017)
17, Number 2, pp. 372–404

rdrobust: Software for regression-discontinuity designs

Sebastian Calonico
University of Miami
Miami, FL
scalonico@bus.miami.edu

Matias D. Cattaneo
University of Michigan
Ann Arbor, MI
cattaneo@umich.edu

Max H. Farrell
University of Chicago
Chicago, IL
max.farrell@chicagobooth.edu

Rocío Titiunik
University of Michigan
Ann Arbor, MI
titiunik@umich.edu

Abstract. We describe a major upgrade to the Stata (and R) `rdrobust` package, which provides a wide array of estimation, inference, and falsification methods for the analysis and interpretation of regression-discontinuity designs. The main new features of this upgraded version are as follows: i) covariate-adjusted bandwidth selection, point estimation, and robust bias-corrected inference, ii) cluster–robust bandwidth selection, point estimation, and robust bias-corrected inference, iii) weighted global polynomial fits and pointwise confidence bands in regression-discontinuity plots, and iv) several new bandwidth selection methods, including different bandwidths for control and treatment groups, coverage error-rate optimal bandwidths, and optimal bandwidths for fuzzy designs. In addition, the upgraded package has superior performance because of several numerical and implementation improvements. We also discuss issues of backward compatibility and provide a companion R package with the same syntax and capabilities.

Keywords: st0366_1, rdrobust, rdbwselect, rdplot, regression discontinuity

1 Introduction

The regression-discontinuity (RD) design is widely used in applied work. It is one of the most credible quasi-experimental research designs for identification, estimation, and inference of treatment effects (local to the cutoff). RD designs are also easy to present, interpret, and falsify, which are features that have contributed to their popularity among practitioners and policy makers alike. See Imbens and Lemieux (2008) and Lee and Lemieux (2010) for early reviews; Cattaneo, Titiunik, and Vazquez-Bare for a practical introduction to RD designs with a comparison between leading empirical methods; and Cattaneo and Escanciano (2017) for an edited volume with a recent overview of the literature.

In this article, we describe a major upgrade to the Stata and R software package `rdrobust` (Calonico, Cattaneo, and Titiunik 2014a, 2015b), which provides a wide array of estimation, inference, and falsification methods for the analysis and interpretation of

RD designs. These major upgrades are implemented following the technical and methodological results discussed in Calonico, Cattaneo, and Farrell (Forthcoming, 2016a) and Calonico et al. (2016b, CCFT hereafter) and its supplemental appendix. To avoid repetition, in this article we focus exclusively on the new functionalities incorporated into the package; for a description of all previously available features, refer to the previously published software articles. The main new features of the upgraded `rdrobust` package are the following (organized by underlying command or function):

1. `rdrobust`. This command now allows for covariate-adjusted point estimation and covariate-adjusted robust bias-corrected inference. In addition, this command now allows for different heteroskedasticity-robust (heteroskedasticity-consistent k class or HCk class) and cluster–robust variance estimation methods. When mean squared error (MSE)–optimal bandwidths are used, the resulting point estimator for the RD treatment effect is MSE optimal. When coverage error-rate (CER) optimal bandwidths are used, the resulting confidence intervals (CIs) for the RD treatment effect are CER–optimal.

2. `rdbwselect`. This command now offers data-driven bandwidth selection for either one common bandwidth or two distinct bandwidths on either side of the cutoff, selected to be either MSE optimal or CER optimal. In addition, MSE- and CER-optimal bandwidth choices for fuzzy RD designs are now also available. Furthermore, new regularization methods are also provided. Finally, the new implementations allow for covariate-adjusted methods, as well as for different heteroskedasticity-robust (HCk class) and cluster–robust variance estimation methods.

3. `rdplot`. This command now allows for kernel weighting and possibly different bandwidths on either side of the cutoff, which permits plotting treatment effects using RD plots. In addition, this command now allows for CIs for each bin to assess the (local) variability of the partitioning fit.

Also, all three commands now allow for optional user-defined frequency weights. Furthermore, the upgraded `rdrobust` package has new and improved numerical implementations, which now permit feasible executions with large sample sizes.

First, we tested the default implementation of the old 2014 against the new 2016 `rdrobust` command, which includes data-driven bandwidth selection via `rdbwselect` and uses nearest neighbor (NN)–based variance estimators. We used 100 replications of simulation model 1 in Calonico, Cattaneo, and Titiunik (2014b) and CCFT. The average computation time is reported in table 1 for six different sample sizes: $n = 500, 1000, 5000, 10000, 50000, 100000$. The 2016 package exhibits remarkable improvements in execution time, especially for larger sample sizes (because the old version of the software is coded in a way that does not scale well with n). For example, for $n = 50000$, the average execution time is 95.651 seconds with the 2014 version but only 1.148 seconds with the 2016 version. Thus, for this sample size, the upgraded `rdrobust` command runs 83.32 times faster than its predecessor. Importantly, the default implementation is fully backward compatible (see section 6 for details), and therefore the execution time improvements are exclusively attributable to the new way of implementing the package.

Second, the new package was tested using 30+ million observations in Stata/SE 14 with the following results: full execution, including data-driven bandwidth selection, took roughly 5 minutes if the default options were used and roughly 16 minutes if a cluster–robust option was used in addition. (We thank Quentin Brummet at the U.S. Census for carrying out these numerical tests.) In sum, we found that the new version of the package exhibits substantial speed improvements relative to its predecessor.

Table 1. Speed comparisons between 2014 and 2016 `rdrobust` versions

Sample size (n)	Old `rdrobust` (2014) (average time in seconds)	New `rdrobust` (2016) (average time in seconds)	Time improvement (old/new)
500	0.216	0.154	1.40
1,000	0.270	0.181	1.49
5,000	1.531	0.257	5.96
10,000	4.952	0.375	13.21
50,000	95.651	1.148	83.32
100,000	385.106	2.104	183.04

Notes: i) Computed using 100 replications of simulation model 1 in Calonico, Cattaneo, and Titiunik (2014b) and CCFT using an Intel Xeon CPU E5-2620 v2 @ 2.1GHz, 32Gb RAM and Stata 14.2. ii) Time is measured in seconds. iii) Time improvement is computed as the ratio of the old `rdrobust` (2014) average speed in seconds relative to the new `rdrobust` (2016) average speed in seconds.

The 2016 version of the `rdrobust` package offers a comprehensive set of tools for a systematic and objective analysis of RD designs in empirical work. For related Stata and R commands implementing manipulation testing based on discontinuity in density using local polynomial techniques, see Cattaneo, Jansson, and Ma (2017) and references therein. For Stata and R commands implementing inference procedures based on a local randomization assumption, see Cattaneo, Titiunik, and Vazquez-Bare (Forthcoming) and references therein.

In the remaining sections, we provide methodological, practical, and empirical introductions to the new functionalities of the `rdrobust` package. In section 2, we briefly review the main methodological concepts underlying the methods implemented. We then provide the full syntax and a brief explanation of the functionalities of each of the three upgraded Stata commands in sections 3, 4, and 5, explicitly highlighting what is new relative to the previous version. In section 6, we discuss issues of backward compatibility. In section 7, we present an empirical illustration of the new methods and functionalities available in the upgraded `rdrobust` package, using the same dataset previously used in Calonico, Cattaneo, and Titiunik (2014a, 2015b) to facilitate the comparison. Finally, we conclude the article in section 8. We also provide a companion R package with the same functionality and syntax.

The latest version of this software, as well as other related software for RD designs, can be found at https://sites.google.com/site/rdpackages/.

2 Overview of new methods

Here we provide a brief account of the main new features included in the upgraded version of the `rdrobust` package. All technical and methodological results are discussed in Calonico, Cattaneo, and Farrell (2016a) and CCFT and their supplemental appendices. As mentioned above, we assume that the reader is familiar with the previous versions of the `rdrobust` package, their companion software articles, and the underlying technical papers. Therefore, in this section, we focus exclusively on what is new.

In addition to several numerical and implementation upgrades, four main new features are available in the package: i) inclusion of covariates, ii) new heteroskedasticity-consistent (HC) and cluster–robust variance estimators, iii) new bandwidth selection methods, and iv) kernel weighting and CIs in RD plots. We discuss these new features in the next four subsections.

For clarity, we focus on sharp RD designs exclusively. The software does cover all other RD designs, but because all the new features are conceptually identical for any RD design, we do not spell out the details for fuzzy and kink RD designs beyond giving a few generic examples at the end of section 7. See Card et al. (2015) for identification results in kink RD designs, see the help files for specific implementation details, and see CCFT for technical and methodological results when using additional covariates or allowing for clustering.

2.1 Using additional covariates

The first main change to the software is that covariates may be included in the estimation. To make this precise, we first describe the set up. The observed data are assumed to be a random sample $(Y_i, T_i, X_i, \mathbf{Z}_i')'$, $i = 1, 2, \ldots, n$, from a large population. The score, index, or running variable is X_i, and treatment status is determined as $T_i = \mathbb{1}(X_i \geq \overline{x})$ for the known cutoff $\overline{x}$. Using the potential-outcomes framework, the observed outcome is $Y_i = Y_i(0) \times (1 - T_i) + Y_i(1) \times T_i$, where $Y_i(0)$ and $Y_i(1)$ denote the potential outcomes for each unit under control and treatment, respectively. The d-dimensional vector $\mathbf{Z}_i$ denotes a collection of "preintervention" covariates that could be continuous, discrete, or mixed.

The parameter of interest is the standard RD treatment effect at the cutoff:

$$\tau = \tau(\overline{x}) = \mathbb{E}\left\{Y_i(1) - Y_i(0) | X_i = \overline{x}\right\}$$

The goal is to estimate τ via local polynomial methods at the cutoff $\overline{x}$. Previously, the `rdrobust` package would use only the outcome Y_i and score X_i to estimate the RD treatment effect. Now, the additional covariates $\mathbf{Z}_i$ may also be included, which can increase the efficiency of the estimator. The covariate-adjusted RD estimator of τ implemented in `rdrobust` is defined as

$$\widetilde{\tau}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h) - \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$$

where $\widetilde{\boldsymbol{\beta}}_{Y+,p}(h)$ and $\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$ are defined through

$$\widetilde{\boldsymbol{\theta}}_{Y,p}(h) = \underset{\boldsymbol{\beta}_-,\boldsymbol{\beta}_+,\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\{Y_i - \mathbf{r}_{-,p}(X_i-\overline{x})'\boldsymbol{\beta}_- - \mathbf{r}_{+,p}(X_i-\overline{x})'\boldsymbol{\beta}_+ - \mathbf{Z}_i'\boldsymbol{\gamma}\right\}^2 K_h(X_i-\overline{x})$$

with $\widetilde{\boldsymbol{\theta}}_{Y,p}(h) = \{\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)', \widetilde{\boldsymbol{\beta}}_{Y+,p}(h)', \widetilde{\boldsymbol{\gamma}}_{Y,p}(h)'\}'$; $\boldsymbol{\beta}_-, \boldsymbol{\beta}_+ \in \mathbb{R}^{p+1}$ and $\boldsymbol{\gamma} \in \mathbb{R}^d$; $\mathbf{r}_{-,p}(x) = \mathbb{1}(x<0)(1,x,\ldots,x^p)'$; $\mathbf{r}_{+,p}(x) = \mathbb{1}(x \geq 0)(1,x,\ldots,x^p)'$; $\mathbf{e}_0$, the $(p+1)$ vector, with a 1 in the first position and 0s in the rest; and $K_h(u) = K(u/h)/h$ for a kernel function $K(\cdot)$ and a positive bandwidth sequence h. The kernel and bandwidth serve to localize the regression fit near the cutoff, and the most popular choices are i) the uniform kernel, giving equal weighting to observations $X_i \in [\overline{x}-h, \overline{x}+h]$, and ii) the triangular kernel that assigns linear down-weighting to the same observations. The preferred choice of polynomial order is $p=1$, which gives the standard local linear RD point estimator.

The new version of the package allows for weighted least-squares estimation and inference. To be more precise, if unit-specific weights w_i are provided by the user, then the above fitting (and all underlying estimation and inference procedures) is done with $w_i \times K_h(X_i - \overline{x})$ in place of the simple kernel weights $K_h(X_i - \overline{x})$.

As formalized in CCFT, the covariates are introduced in a joint least-squares fit to minimize the underlying assumptions required for the covariate-adjusted RD estimator $\widetilde{\tau}(h)$ to remain consistent for the standard RD treatment effect τ. This requires one to assume that some features of the marginal distributions of $\mathbf{Z}_i$ above and below the cutoff are equal. This idea matches what typically is understood as covariates being "pretreatment" in the context of randomized experiments. In the case of sharp and fuzzy RD designs, it is sufficient to assume that the potential covariates under treatment and control have equal conditional expectation at the cutoff. Indeed, this is often conceived and presented as a falsification or placebo test in RD empirical studies. This simple requirement of balanced covariates at the cutoff, or zero RD treatment effect on covariates, ensures that $\widetilde{\tau}(h) \rightarrow_{\mathbb{P}} \tau$. In the case of sharp and fuzzy kink RD designs, it is required to assume that the first derivative of the regression functions under treatment and control are equal at the cutoff. These conditions can be tested empirically using the package `rdrobust`, for example, by taking the covariates as outcome variables.

The precise form of $\widetilde{\tau}(h)$ warrants several comments. Most notably, the estimation combines units from both sides of the cutoff $\overline{x}$, whereas, typically, the two sides are estimated separately. The most salient consequence is that the coefficient on the covariates, $\widetilde{\boldsymbol{\gamma}}_{Y,p}(h)$, is common to both treatment and comparison groups. This turns out to be important for consistency of $\widetilde{\tau}(h)$, the covariate-adjusted RD estimator of τ, as mentioned above. Furthermore, this mimics standard widespread practice for experimental treatment-effects estimation with covariate adjustment, where additional covariates are typically included in an additive-separable, linear-in-parameters way. Finally, when the covariates are excluded, the standard RD treatment effect previously implemented in `rdrobust` is recovered. See section 6 for more discussion on backward compatibility.

In the upgraded version of `rdrobust`, we continue to use the same kernel function for units below and above the cutoff, but we now allow for possibly different bandwidths on either side. This feature is discussed in more detail below in the context of data-driven bandwidth selection. The presentation of $\widetilde{\tau}(h)$ assumes an equal bandwidth applied to both sides only to simplify the exposition. The supplemental appendix of CCFT contains all the details, including the possibility of using different bandwidths on either side of the cutoff.

For inference based on $\widetilde{\tau}(h)$, we continue to use robust nonparametric bias-correction methods, following the recent results in Calonico, Cattaneo, and Titiunik (2014b) and Calonico, Cattaneo, and Farrell (Forthcoming, 2016a). Although technically more cumbersome, because of the inclusion of additional covariates and joint RD estimation, the core ideas underlying robust bias-correction inference do not change much with the inclusion of $\mathbf{Z}_i$, and they have already been discussed from an implementation perspective in Calonico, Cattaneo, and Titiunik (2014a, 2015b).

Thus, we do not reproduce the details here and instead offer a quick outline of their main features for future reference and discussion: i) the misspecification bias of $\widetilde{\tau}(h)$ now depends on the curvature of $\mathbb{E}\{Y_i(t)|X_i = x\}$, as before, and also on the curvature of the conditional expectations of the (potential) additional covariates given the score included in the estimation; ii) this leading bias takes the form $h^{p+1}\mathcal{B}$, where $\mathcal{B}$ is different depending on whether additional covariates are included; iii) the bias-corrected estimator is then $\widetilde{\tau}^{\mathsf{bc}}(h,b) := \widetilde{\tau}(h) - h^{p+1}\widetilde{\mathcal{B}}(b)$, where $\widetilde{\mathcal{B}}(b)$ denotes an estimator of $\mathcal{B}$ constructed using a possibly different preliminary bandwidth sequence b; and iv) the variance of $\widetilde{\tau}^{\mathsf{bc}}(h,b)$ is denoted by $\mathcal{V}^{\mathsf{bc}}(h,b)$, which captures both the variability of the RD point estimator and the variability of bias correction.

Based on the above, and under standard regularity conditions, CCFT show that valid asymptotic inference for τ can be conducted using the usual robust bias-corrected t statistic $\sqrt{nh}\{\widetilde{\tau}^{\mathsf{bc}}(h,b)-\tau\}/\sqrt{\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)}$, where $\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)$ denotes a consistent variance estimator of $\mathcal{V}^{\mathsf{bc}}(h,b)$. Using this result, for example, we obtain a $100\times(1-\alpha)\%$ robust bias-corrected covariate-adjusted CI for the treatment effect τ,

$$\left[\widetilde{\tau}^{\mathsf{bc}}(h,b) - \frac{\Phi_{1-\alpha/2}}{\sqrt{nh}} \times \sqrt{\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)} \quad , \quad \widetilde{\tau}^{\mathsf{bc}}(h,b) + \frac{\Phi_{1-\alpha/2}}{\sqrt{nh}} \times \sqrt{\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)}\right]$$

where Φ_α denotes the α percentile of the standard normal distribution.

In the next two subsections, we discuss the choice of variance estimator, $\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)$, which now allows for different heteroskedasticity-robust and cluster–robust methods, and the choice of bandwidths, which now allows for several data-driven plug-in methods. Recall that we focus exclusively on the new features of the `rdrobust` package. See section 6 for more discussion on backward compatibility.

2.2 HCk and cluster–robust variance estimation

The variance estimator used in `rdrobust` is designed to capture the variability of the initial estimator, $\widetilde{\tau}(h)$, and the bias correction, $h^{p+1}\widetilde{\mathcal{B}}(b)$, and as such will depend on both bandwidths. The particular form of the variance, $\mathcal{V}^{\mathsf{bc}}(h,b)$, is derived with a fixed-n (preasymptotic) approach, conditional on the score observations $X_1, X_2, \ldots, X_n$. This approach, together with the fact that both sources of variability are captured, yields asymptotic refinements and increased robustness to bandwidth choice of the associated inference procedures. Importantly, in the present context, $\mathcal{V}^{\mathsf{bc}}(h,b)$ depends on the additional covariates and hence is necessarily different from prior work.

The only unknown elements of $\mathcal{V}^{\mathsf{bc}}(h,b)$ are the variances of the outcome and the covariates, and their covariance, conditional on $X_1, X_2, \ldots, X_n$ (the latter two being new here). That is, to form an estimator $\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)$, we must estimate, for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, d$,

$$\sigma_{YZ_k-,i} = \mathbb{C}\mathrm{ov}\left\{Y_i(0), Z_{ki}(0)|X_i\right\}, \qquad \sigma_{YZ_k+,i} = \mathbb{C}\mathrm{ov}\left\{Y_i(1), Z_{ki}(1)|X_i\right\}$$

The feasible variance estimators are then constructed by replacing these unknown objects with estimators thereof, according to one of the following two options:

1. Nearest-neighbor method. This method uses ideas in Müller and Stadtmuller (1987) and Abadie and Imbens (2008). We replace $\sigma_{YZ_k-,i}$ and $\sigma_{YZ_k+,i}$ by the corresponding sample covariance estimator based on the J NNs to unit i, among units belonging to the same group (that is, below or above the cutoff). Specifically, neighbors are determined using the Euclidean distance based on the score variable X_i, and J denotes a (fixed) number of neighbors chosen. Up to the change of variable being used (that is, Y_i or Z_{ki} for $k = 1, 2, \ldots, d$), the procedure is exactly the same as the one used in the previous version of the `rdrobust` package.

2. HCk plug-in residuals method. This method applies ideas from least-squares estimation and inference; see MacKinnon (2013) and Cameron and Miller (2015) for review on variance estimation in this context. In this case, we replace $\sigma_{YZ_k-,i}$ and $\sigma_{YZ_k+,i}$ by, respectively,

$$\mathbb{1}(X_i < \overline{x})\omega_{-,i}\left\{Y_i - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Y-,q}(h)\right\}\left\{Z_{ki} - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Z_k-,q}(h)\right\}$$
$$\mathbb{1}(X_i \geq \overline{x})\omega_{+,i}\left\{Y_i - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Y+,q}(h)\right\}\left\{Z_{ki} - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Z_k+,q}(h)\right\}$$

   for $k = 1, 2, \ldots, d$, which are plug-in residuals obtained from running local polynomial regressions using either the main outcome variable or the additional covariates as the dependent variable. The weights $\omega_{-,i}$ and $\omega_{+,i}$ denote a possible finite-sample adjustment for HCk variance estimators, and $q > p$ denotes the polynomial order used for bias correction.

Precise use of these estimators, and the relevant formulas, are discussed in detail in the supplemental appendix of CCFT. See also Bartalotti and Brummet (2017) for a discussion on cluster–robust inference in sharp RD designs. Cluster–robust versions of variance estimators are also implemented, following the same logic described above but also accounting for the (one-way) cluster structure of the data.

The different variance estimators are used to Studentize statistics, form CIs, and implement data-driven bandwidth selectors, allowing for both conditional heteroskedasticity (NN, HC0, HC1, HC2, HC3) and clustering (NN or plug-in residuals with the usual degrees-of-freedom adjustment). To summarize, the upgraded `rdrobust` package includes a total of 10 distinct variance estimators (NN or plug-in residuals with either HC0, HC1, HC2, or HC3, and NN-adjusted or degrees-of-freedom–adjusted clustering). As a comparison, the previous version of `rdrobust` had only two (NN or plug-in residuals using HC0 weighting). As explained above, when additional covariates are not included, the upgraded implementations may be used for standard RD inference involving only the outcome and the score variables. See section 6 for more discussion on backward compatibility.

Finally, an important practical issue regarding the NN variance estimators arises when the running variable X_i is not continuously distributed. In some applications, X_i may exhibit mass points, making the number of eligible equidistant near neighbors strictly larger than the number specified in `rdrobust` or `rdbwselect` (recall that the default is three neighbors for each observation). In the previous version of these commands, ties were broken at random to select the exact number of neighbors prespecified by the user (or set by default). In the upgraded version of these commands, the NN variance estimators use all equidistant neighbors, even if the total number exceeds the one selected. This new approach is fully replicable and also leads to a more efficient variance estimator.

2.3 Bandwidth selection

The `rdbwselect` command has also been upgraded and now offers several new data-driven bandwidth selection methods. The three main upgrades are i) homogenized and improved MSE-optimal bandwidth choices for sharp RD designs, ii) new MSE-optimal bandwidth choices for fuzzy RD designs, and iii) new CER-optimal bandwidth choices for sharp and fuzzy RD designs. As a comparison, the previous version of `rdbwselect` implemented only three main types of MSE-optimal bandwidth choices for sharp RD designs (`ik`, `cct`, and `cv`). The new version implements over 10 different choices for sharp RD designs (and also implements the corresponding choices for fuzzy RD designs). In addition, we continue to offer regularization methods for all choices implemented, following Imbens and Kalyanaraman (2012), but implemented as discussed in Calonico, Cattaneo, and Titiunik (2014a,b, 2015b) and the corresponding supplemental appendices.

In this section, we heuristically explain some of the bandwidth choices offered in the upgraded version of the `rdbwselect` command. See Cattaneo and Vazquez-Bare (2016) for a more comprehensive discussion and comparison between bandwidth selection procedures. This command now allows for different bandwidths on either side of the cutoff (previously only one common bandwidth was allowed), in addition to distinguishing between sharp and fuzzy RD designs and between MSE-optimal and CER-optimal choices. We outline only the case of sharp RD designs to avoid cumbersome notation. For technical and methodological details, see CCFT and its supplemental appendix.

Regarding the new bandwidth selectors implemented, note that a typical asymptotic MSE expansion gives

$$\text{MSE}\left\{\widehat{\theta}(h)\right\} \approx h^{2p+2}B + \frac{1}{nh}V$$

for some estimator $\widehat{\theta}(h)$, where B and V denote the squared bias and the variance of the estimator, respectively. Different estimators will have different bias and variance terms, which also depend on whether additional covariates are included. For example, in sharp RD designs, we consider four main alternative MSE expansions determined by the choice of estimator:

1. RD estimator $\widehat{\theta}(h) = \widetilde{\tau}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h) - \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$
2. Left-hand-side estimator $\widehat{\theta}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$
3. Right-hand-side estimator $\widehat{\theta}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h)$
4. Sum of the one-sided estimators $\widehat{\theta}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h) + \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$

We construct these MSE expansions for both the standard case without covariates and the covariate-adjusted case. This gives a set of alternative MSE-optimal bandwidth choices: $h_{\texttt{mse,rd}}$, $h_{\texttt{mse,l}}$, $h_{\texttt{mse,r}}$, and $h_{\texttt{mse,sum}}$, with or without additional covariates. The first three are directly applicable to RD designs, while the fourth is mostly useful for regularization purposes.

Assuming the denominator is not 0, any of these MSE-optimal bandwidth choices, with or without additional covariates, takes the following form:

$$h_{\texttt{mse},j} = \left\{\frac{V_j/n}{2(1+p)B_j}\right\}^{\frac{1}{3+2p}} \qquad j \in \{\texttt{rd}, \texttt{l}, \texttt{r}, \texttt{sum}\}$$

where the constants are specific to the option chosen. Given a choice of MSE-optimal bandwidth, preliminary estimates of the leading asymptotic constants are straightforward to construct, though they depend on whether additional covariates have been included as well as whether heteroskedasticity or clustering is assumed. This leads to the MSE-optimal plug-in bandwidth selectors:

$$\widehat{h}_{\texttt{mse},j} = \left\{\frac{\widehat{V}_j/n}{2(1+p)\widehat{B}_j}\right\}^{\frac{1}{3+2p}} \qquad j \in \{\texttt{rd}, \texttt{l}, \texttt{r}, \texttt{sum}\}$$

where the exact form of the specific preliminary estimates, and some of their asymptotic properties, are discussed in the supplemental appendix of CCFT. The upgraded `rdbwselect` command implements all these alternatives.

In addition, following the recent work in Calonico, Cattaneo, Farrell (Forthcoming, 2016a), we implement CER-optimal bandwidth choices. These alternative plug-in bandwidth selectors take the form

$$\widehat{h}_{\texttt{cer},j} = n^{-\frac{p}{(3+p)(3+2p)}} \times \widehat{h}_{\texttt{mse},j} \qquad j \in \{\texttt{rd}, \texttt{l}, \texttt{r}, \texttt{sum}\}$$

These bandwidth choices minimize the CER of the robust bias-corrected CI implemented by `rdrobust` and therefore may be preferable in practice for inference purposes.

To summarize, the major upgrade of the `rdbwselect` command offers now eight distinct MSE-optimal bandwidth choices ($\widehat{h}_{\texttt{mse,rd}}$, $\widehat{h}_{\texttt{mse,l}}$, $\widehat{h}_{\texttt{mse,r}}$, and $\widehat{h}_{\texttt{mse,sum}}$, with and without regularization) and eight distinct CER-optimal bandwidth choices ($\widehat{h}_{\texttt{cer,rd}}$, $\widehat{h}_{\texttt{cer,l}}$, $\widehat{h}_{\texttt{cer,r}}$, and $\widehat{h}_{\texttt{cer,sum}}$, with and without regularization). Furthermore, we also provide the following two additional bandwidth selectors, which have better rate properties:

- Possibly different bandwidths on either side: $\widehat{h}_{\texttt{comb,l}} = \text{median}\{\widehat{h}_{\texttt{mse,l}}, \widehat{h}_{\texttt{mse,rd}}, \widehat{h}_{\texttt{mse,sum}}\}$ and $\widehat{h}_{\texttt{comb,r}} = \text{median}\{\widehat{h}_{\texttt{mse,r}}, \widehat{h}_{\texttt{mse,rd}}, \widehat{h}_{\texttt{mse,sum}}\}$, and similarly for the CER-optimal version.
- Equal bandwidth on both sides: $\widehat{h}_{\texttt{comb}} = \min\{\widehat{h}_{\texttt{mse,rd}}, \widehat{h}_{\texttt{mse,sum}}\}$, and similarly for the CER-optimal version.

Importantly, note that the `ik`, `cct`, and `cv` bandwidth choices have been deprecated and are no longer supported as part of the `rdrobust` package. The bandwidth choice $\widehat{h}_{\texttt{mse,rd}}$ is an upgraded version of both the `ik` and the `cct` implementations of the MSE-optimal bandwidth selectors discussed in Imbens and Kalyanaraman (2012) and Calonico, Cattaneo, and Titiunik (2014b), respectively. See section 6 for more discussion on backward compatibility. Lastly, the `cv` (cross-validation) bandwidth selection method was removed because it appears to be considerably less popular than plug-in bandwidth selection methods in empirical work, and at present it is not theoretically justified nor easily portable to the new settings considered in the upgraded version of `rdrobust` (for example, inclusion of covariates or clustering).

2.4 RD plots

RD plots are commonly used in RD empirical work, and their main methodological features are discussed in great detail in Calonico, Cattaneo, and Titiunik (2015a). These plots can be easily constructed using the `rdplot` command. The upgraded version of this command now includes two main new features.

- Kernel-weighted polynomial fits with possibly different bandwidths. The `rdplot` command now allows for (global or restricted) weighted polynomial fits using any of the kernel weighting schemes available for estimation and inference in RD designs: uniform, triangular, or Epanechnikov. Furthermore, following the upgrades to `rdrobust` and `rdbwselect`, the `rdplot` command now also allows for possibly different bandwidths on either side of the cutoff when computing the global polynomial fits. These new options permit the exact graphical presentation of RD point estimation and inference by restricting the support of the running variable to the neighborhood around the cutoff determined by the bandwidth used. See section 5 for syntax details and section 7 for an empirical illustration of this new feature.

- Confidence intervals for local binned fits. The `rdplot` command now allows the user to plot and report CIs for local means within each bin. Specifically, for each bin $j = 1, 2, \ldots, J_n$, the `rdplot` command computes the following CI:

$$\mathrm{CI}_j = \left[\overline{X}_j - \mathcal{T}_{1-\alpha/2} \times \sqrt{S_j^2/N_j}\ ,\ \overline{X}_j + \mathcal{T}_{1-\alpha/2} \times \sqrt{S_j^2/N_j} \right]$$

where $\overline{X}_j$ denotes the sample mean in bin j, S_j^2 denotes the sample variance in bin j, N_j denotes the sample size in bin j, and $\mathcal{T}_\alpha$ denotes the $\alpha \in (0,1)$ quantile of the Student's t distribution with $N_j - 1$ degrees of freedom. This formula is justified by preasymptotic inference and as a nonparametric inference procedure, up to smoothing bias, following the results in Cattaneo and Farrell (2013) for partitioning regression methods.

3 The rdrobust command

This section describes the full syntax of the upgraded `rdrobust` command. Whenever possible, we retain the same syntax as in the previous version of this command (Calonico, Cattaneo, and Titiunik 2014a, 2015b).

3.1 Syntax

`rdrobust` *depvar runvar* [*if*] [*in*] [, `c(`*cutoff*`)` `p(`*pvalue*`)` `q(`*qvalue*`)`
`deriv(`*dvalue*`)` `fuzzy(`*fuzzyvar* [`sharpbw`]`)` `covs(`*covars*`)` `kernel(`*kernelfn*`)`
`weights(`*weightsvar*`)` `h(`*hvalueL hvalueR*`)` `b(`*bvalueL bvalueR*`)` `rho(`*rhovalue*`)`
`scalepar(`*scaleparvalue*`)` `bwselect(`*bwmethod*`)` `scaleregul(`*scaleregulvalue*`)`
`vce(`*vcemethod*`)` `level(`*level*`)` `all`]

where *depvar* is the dependent variable and *runvar* is the running variable (also known as the score or forcing variable).

Only new options or options that have changed in this version are discussed in section 3.2.

3.2 Options

`fuzzy(`*fuzzyvar* [`sharpbw`]`)` specifies the treatment status variable used to implement fuzzy RD estimation (or fuzzy kink RD if `deriv(1)` is also specified). The default is sharp RD design. If the `sharpbw` option is set, the fuzzy RD estimation is performed using a bandwidth selection procedure for the sharp RD model. This option is automatically selected if there is perfect compliance at either side of the threshold.

`covs(`*covars*`)` specifies additional covariates to be used for estimation and inference.

`weights(`*weightsvar*`)` specifies the variable used for optional weighting of the estimation procedure. The unit-specific weights multiply the kernel function.

`h(`*hvalueL hvalueR*`)` specifies the main bandwidth, h, to be used on the left and on the right of the cutoff, respectively. If only one value is specified, then this value is used on both sides. If not specified, the bandwidth(s) h is computed by the companion command `rdbwselect`.

`b(`*bvalueL bvalueR*`)` specifies the bias bandwidth, b, to be used on the left and on the right of the cutoff, respectively. If only one value is specified, then this value is used on both sides. If not specified, the bandwidth(s) b is computed by the companion command `rdbwselect`.

`bwselect(`*bwmethod*`)` specifies the bandwidth selection procedure to be used. By default, it computes both h and b, unless ρ is specified, in which case it computes only the h and sets $b = h/\rho$. Implementation and numerical details are given in CCFT. *bwmethod* may be one of the following:

`mserd` specifies one common MSE-optimal bandwidth selector for the RD treatment-effect estimator. `mserd` is the default.

`msetwo` specifies two different MSE-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`msesum` specifies one common MSE-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`msecomb1` specifies min(`mserd`, `msesum`).

`msecomb2` specifies median(`msetwo`, `mserd`, `msesum`) for each side of the cutoff separately.

`cerrd` specifies one common CER-optimal bandwidth selector for the RD treatment-effect estimator.

`certwo` specifies two different CER-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`cersum` specifies one common CER-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`cercomb1` specifies min(`cerrd`, `cersum`).

`cercomb2` specifies median(`certwo`, `cerrd`, `cersum`) for each side of the cutoff separately.

`vce(`*vcemethod*`)` specifies the procedure used to compute the variance–covariance matrix estimator. Implementation and numerical details are given in CCFT. *vcemethod* may be one of the following:

`nn` [*nnmatch*] specifies a heteroskedasticity-robust NN variance estimator with *nnmatch* indicating the minimum number of neighbors to be used. The default is `vce(nn 3)`.

`hc0` specifies a heteroskedasticity-robust HC0 plug-in residuals variance estimator.

`hc1` specifies a heteroskedasticity-robust HC1 plug-in residuals variance estimator.

`hc2` specifies a heteroskedasticity-robust HC2 plug-in residuals variance estimator.

`hc3` specifies a heteroskedasticity-robust HC3 plug-in residuals variance estimator.

`nncluster` *clustervar* [*nnmatch*] specifies a cluster–robust NN variance estimator with *clustervar* indicating the cluster ID variable and *nnmatch* indicating the minimum number of neighbors to be used.

`cluster` *clustervar* specifies a cluster–robust plug-in residuals variance estimator with *clustervar* indicating the cluster ID variable.

3.3 Options removed or deprecated

The following options were removed from the upgraded `rdrobust` command: `delta()`, `cvgrid_min()`, `cvgrid_max()`, `cvgrid_length()`, `cvplot`, and `matches()`.

4 The rdbwselect command

This section describes the full syntax of the upgraded `rdbwselect` command. Whenever possible, we retain the same syntax as in the previous version of this command (Calonico, Cattaneo, and Titiunik 2014a, 2015b).

4.1 Syntax

`rdbwselect` *depvar runvar* [*if*] [*in*] [`,` `c(`*cutoff*`)` `p(`*pvalue*`)` `q(`*qvalue*`)`
 `deriv(`*dvalue*`)` `fuzzy(`*fuzzyvar* [`sharpbw`]`)` `covs(`*covars*`)` `kernel(`*kernelfn*`)`
 `weights(`*weightsvar*`)` `bwselect(`*bwmethod*`)` `scaleregul(`*scaleregulvalue*`)`
 `vce(`*vcemethod*`)` `all`]

where *depvar* is the dependent variable and *runvar* is the running variable (also known as the score or forcing variable).

Only new options or options that have changed in this version are discussed in section 4.2.

4.2 Options

`fuzzy(`*fuzzyvar* [`sharpbw`]`)` specifies the treatment status variable used to implement fuzzy RD estimation (or fuzzy kink RD if `deriv(1)` is also specified). The default is sharp RD design. If the `sharpbw` option is set, the fuzzy RD estimation is performed using a bandwidth selection procedure for the sharp RD model. This option is automatically selected if there is perfect compliance at either side of the threshold.

`covs(`*covars*`)` specifies additional covariates to be used for estimation and inference.

`weights(`*weightsvar*`)` specifies the variable used for optional weighting of the estimation procedure. The unit-specific weights multiply the kernel function.

`bwselect(`*bwmethod*`)` specifies the bandwidth selection procedure to be used. Implementation and numerical details are given in CCFT. *bwmethod* may be one of the following:

`mserd` specifies one common MSE-optimal bandwidth selector for the RD treatment-effect estimator. This is the default.

`msetwo` specifies two different MSE-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`msesum` specifies one common MSE-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`msecomb1` specifies min(`mserd`, `msesum`).

`msecomb2` specifies median(`msetwo`, `mserd`, `msesum`) for each side of the cutoff separately.

`cerrd` specifies one common CER-optimal bandwidth selector for the RD treatment-effect estimator.

`certwo` specifies two different CER-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`cersum` specifies one common CER-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`cercomb1` specifies min(`cerrd`, `cersum`).

`cercomb2` specifies median(`certwo`, `cerrd`, `cersum`) for each side of the cutoff separately.

`vce(`*vcemethod*`)` specifies the procedure used to compute the variance–covariance matrix estimator. Implementation and numerical details are given in CCFT. *vcemethod* may be one of the following:

`nn` [*nnmatch*] specifies a heteroskedasticity-robust NN variance estimator with *nnmatch* indicating the minimum number of neighbors to be used. The default is `vce(nn 3)`.

`hc0` specifies a heteroskedasticity-robust HC0 plug-in residuals variance estimator.

`hc1` specifies a heteroskedasticity-robust HC1 plug-in residuals variance estimator.

`hc2` specifies a heteroskedasticity-robust HC2 plug-in residuals variance estimator.

`hc3` specifies a heteroskedasticity-robust HC3 plug-in residuals variance estimator.

`nncluster` *clustervar* [*nnmatch*] specifies a cluster–robust NN variance estimator with *clustervar* indicating the cluster ID variable and *nnmatch* indicating the minimum number of neighbors to be used.

`cluster` *clustervar* specifies a cluster–robust plug-in residuals variance estimator with *clustervar* indicating the cluster ID variable.

4.3 Options removed or deprecated

The following options were removed from the upgraded `rdbwselect` command: `delta()`, `cvgrid_min()`, `cvgrid_max()`, `cvgrid_length()`, `cvplot`, and `matches()`.

5 The rdplot command

This section describes the full syntax of the upgraded `rdplot` command. Whenever possible, we retain the same syntax as in the previous version of this command (Calonico, Cattaneo, and Titiunik 2014a, 2015b).

5.1 Syntax

`rdplot` *depvar runvar* [*if*] [*in*] [`,` `c(`*cutoff*`)` `p(`*pvalue*`)` `kernel(`*kernelfn*`)`
`weights(`*weightsvar*`)` `h(`*hvalueL hvalueR*`)` `nbins(`*nbinsvalueL nbinsvalueR*`)`
`binselect(`*binmethod*`)` `scale(`*scalevalueL scalevalueR*`)` `ci(`*cilevel*`)` `shade`
`support(`*supportvalueL supportvalueR*`)` `genvars` `graph_options(`*gphopts*`)` `hide`]

where *depvar* is the dependent variable and *runvar* is the running variable (also known as the score or forcing variable).

Only new options or options that have changed in this version are discussed in section 5.2.

5.2 Options

`kernel(`*kernelfn*`)` specifies the kernel function used to construct the global polynomial estimators. *kernelfn* may be `triangular`, `uniform`, or `epanechnikov`. The default is `kernel(uniform)` (that is, equal or no weighting to all observations on the support of the kernel).

`weights(`*weightsvar*`)` specifies the variable used for optional weighting of the estimation procedure. The unit-specific weights multiply the kernel function.

`h(`*hvalueL hvalueR*`)` specifies the main bandwidth, h, to be used on the left and on the right of the cutoff, respectively. If only one value is specified, then this value is used on both sides. If two bandwidths are specified, the first bandwidth is used for the data below the cutoff and the second bandwidth is used for the data above the cutoff. If not specified, it is chosen to span the full support of the data.

`nbins(`*nbinsvalueL nbinsvalueR*`)` specifies the number of bins used to the left of the cutoff (denoted J_-) and the number of bins used to the right of the cutoff (denoted J_+), respectively. If only one value is specified, then this value is used on both sides. If not specified, J_- and J_+ are estimated using the `binselect()` option.

`scale(`*scalevalueL scalevalueR*`)` specifies a multiplicative factor to be used with the optimal number of bins selected. Specifically, for the control and treated units, the number of bins used will be $\langle scalevalueL \times \widehat{J}_{-,n}\rangle$ and $\langle scalevalueR \times \widehat{J}_{+,n}\rangle$, respectively. If only one value is specified, then this value is used on both sides. The default is `scale(1 1)`.

`ci(`*cilevel*`)` specifies the optional graphical option to display CIs of *cilevel* coverage for each bin.

`shade` specifies the optional graphical option to replace CIs with shaded areas.

`support(`*supportvalueL supportvalueR*`)` specifies an optional extended support of the running variable to be used in the construction of the bins. The default is the sample range.

`genvars` generates the following new variables that store results:

`rdplot_id` stores a unique bin ID for each observation. Negative natural numbers are assigned to observations to the left of the cutoff, and positive natural numbers are assigned to observations to the right of the cutoff.

`rdplot_N` stores the number of observations in the corresponding bin for each observation.

`rdplot_min_bin` stores the lower end value of the bin for each observation.

`rdplot_max_bin` stores the upper end value of the bin for each observation.

`rdplot_mean_bin` stores the middle point of the corresponding bin for each observation.

`rdplot_mean_x` stores the sample mean of the running variable within the corresponding bin for each observation.

`rdplot_mean_y` stores the sample mean of the outcome variable within the corresponding bin for each observation.

`rdplot_se_y` stores the standard deviation of the mean of the outcome variable within the corresponding bin for each observation.

`rdplot_ci_l` stores the lower end value of the confidence interval for the sample mean of the outcome variable within the corresponding bin for each observation.

`rdplot_ci_r` stores the upper end value of the confidence interval for the sample mean of the outcome variable within the corresponding bin for each observation.

`rdplot_hat_y` stores the predicted value of the outcome variable given by the global polynomial estimator.

5.3 Options removed or deprecated

The following options were removed from the upgraded `rdplot` command: `numbinl()`, `numbinr()`, `scalel()`, `scaler()`, `generate()`, `lowerend()`, and `upperend()`.

6 Backward compatibility

Here we discuss backward compatibility with the previous version of `rdrobust` for Stata and the R software package. We organize the presentation in terms of the three main functions.

- `rdrobust`. For a given choice of bandwidth(s), this command is backward compatible by default when additional covariates are not included. Point estimators are identical in all cases, but variance estimators may slightly change in some cases because of internal numerical and implementation upgrades. The previous version of this command included only two variance estimators: NN and plug-in residuals without weighting (HC0). The upgraded version of this command continues to offer these two choices, but the residuals are computed in a slightly different way to improve speed and to give further compatibility with linear least-squares regression-based methods. This new way of computing residuals may generate numerical changes in the following cases:

    1. Nearest-neighbor variance estimation. This variance estimator (which is the default) will be identical to the previous version of the `rdrobust` command in the absence of ties in the running variable X_i but will change slightly when ties occur. If the running variable X_i is truly continuously distributed, there should be no ties, and hence the default option of `rdrobust` is fully backward compatible. When ties occur, the new optimized NN procedure will result in a much faster but slightly different variance estimator.
    2. Plug-in residuals variance estimation. The upgraded version of `rdrobust` includes a new variance estimator based on plug-in residuals, which is not backward compatible. This new version computes all residuals at the cutoff point and is therefore much faster. This approach also mimics exactly linear least-squares methods. In contrast, the previous version of this command computed plug-in residuals using nonparametric methods and hence evaluated the predicted values at different values near the cutoff.

- `rdbwselect`. This command is not backward compatible. Given the many new data-driven bandwidth options, and the several numerical and implementation upgrades, we were forced to redo the command completely. To allow for backward compatibility, we do provide its previous version (called `rdbwselect_2014`) as part of the upgraded `rdrobust` package. This previous (deprecated) version may be used to obtain data-driven bandwidths, which then can be inputted manually to `rdrobust` to obtain backward compatible RD estimates and inference procedures.

  As mentioned above, now two distinct approaches are available for bandwidth selection in fuzzy RD design settings:

  1. Select the bandwidth(s) focusing only on the sharp RD intention-to-treat estimator entering the numerator of the fuzzy RD treatment-effect estimator.
  2. Select the bandwidth(s) focusing only on the actual fuzzy RD treatment-effect estimator, that is, the ratio of reduced-form RD estimators.

  In the previous version of `rdbwselect`, only the first approach was available, but now both alternatives are implemented in the upgraded version. In fuzzy RD contexts, `rdbwselect` will use the second approach by default when two-sided imperfect compliance is present but will otherwise use the first approach. To force `rdbwselect` to use the first approach even when two-sided imperfect compliance is present, use the additional option `sharpbw` within the `fuzzy()` option when specifying the fuzzy variable. Because `rdrobust` selects automatic data-driven bandwidths using `rdbwselect`, the above remarks and options apply to the upgraded `rdrobust` command as well.

- `rdplot`. This command is fully backward compatible. All upgrades are included in addition to the features previously available in this command. Notice that the options for this command have been reorganized to improve consistency and homogeneity with `rdrobust` and `rdbwselect`.

7 Illustration of new methods

We illustrate our commands using the same dataset already used in Calonico, Cattaneo, and Titiunik (2014a, 2015b), where the previous version of the `rdrobust`, `rdbwselect`, and `rdplot` commands are introduced and discussed. While this facilitates the discussion and comparison, in this section, we focus almost exclusively on the new features available in the new version of the package. Whenever possible, we briefly compare and highlight any substantive changes in implementation.

`rdrobust_senate.dta` contains the outcome variable (Y_i), running variable (X_i), and four additional covariates ($\mathbf{Z}_i$) constructed by Cattaneo, Frandsen, Titiunik (2015). The illustration focuses on party advantages in U.S. Senate elections for the period 1914–2010, using a sharp RD design with the unit of analysis being the state at a given election. In this section, we focus on the running variable used to analyze the RD effect of the Democratic party winning a U.S. Senate seat on the vote share obtained in the following election for that same seat.

First, we load the database and present summary statistics:

```
. use rdrobust_senate.dta, clear
. sum vote margin class termshouse termssenate population, sep(2)
    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
        vote |      1,297    52.66627    18.12219          0        100
      margin |      1,390    7.171159    34.32488       -100        100
-------------+---------------------------------------------------------
       class |      1,390    2.023022    .8231983          1          3
  termshouse |      1,108    1.436823    2.357133          0         16
-------------+---------------------------------------------------------
 termssenate |      1,108    4.555957    3.720294          1         20
  population |      1,390     3827919     4436950      78000   3.73e+07
```

The database also includes two other variables, `state` and `year`, which record the state and year of each election. The running variable is `margin`, which ranges from −100 to 100 and records the Democratic party's margin of victory in the statewide election for a given U.S. Senate seat, defined as the vote share of the Democratic party minus the vote share of its strongest opponent. The outcome variable is `vote`, which ranges from 0 to 100 and records the Democratic vote share in the following election for the same seat (that is, six years later). The cutoff is normalized to $\overline{x} = 0$. The additional covariates are `class`, `termshouse`, `termssenate`, and `population`. The variable `class` identifies the electoral class each Senate seat belongs to (this indicates which of the possible three electoral cycles each seat is in); the variables `termshouse` and `termssenate` capture the experience of the Democratic candidate by recording the cumulative number of terms previously served in U.S. House and Senate, respectively; and the variable `population` records the population of the Senate seat's state.

7.1 RD plots with CIs

As mentioned above, the `rdplot` command is fully backward compatible. Hence, in this section, we illustrate only its new features. One of these new features is the inclusion of CIs for the binned sample mean (or partitioning) estimator. This additional option is useful in presenting and assessing the variability of the RD design.

```
. rdplot vote margin, binselect(es) ci(95)
>        graph_options(title("RD Plot: U.S. Senate Election Data")
>                      ytitle(Vote Share in Election at time t+2)
>                      xtitle(Vote Share in Election at time t)
>                      graphregion(color(white)))
RD Plot with evenly spaced number of bins using spacings estimators.
         Cutoff c = 0 | Left of c  Right of c        Number of obs  =       1297
----------------------+----------------------        Kernel         =    Uniform
        Number of obs |       595         702
   Eff. Number of obs |       595         702
  Order poly. fit (p) |         4           4
     BW poly. fit (h) |   100.000     100.000
 Number of bins scale |     1.000       1.000
Outcome: vote. Running variable: margin.
---------------------------------------------
                      | Left of c  Right of c
----------------------+----------------------
        Bins selected |         8           9
   Average bin length |    12.500      11.111
    Median bin length |    12.500      11.111
----------------------+----------------------
    IMSE-optimal bins |         8           9
  Mimicking Var. bins |        15          35
----------------------+----------------------
Rel. to IMSE-optimal: |
        Implied scale |     1.000       1.000
    WIMSE var. weight |     0.500       0.500
    WIMSE bias weight |     0.500       0.500
---------------------------------------------
```

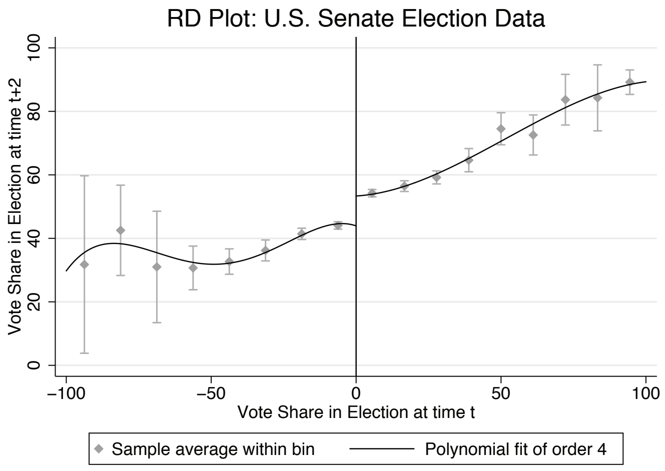


Figure 1. IMSE-optimal evenly spaced RD plot with CIs

Figure 1 provides CIs using the IMSE-optimal number of bins choice for evenly spaced bins on the support of the running variable. In theory, the CIs presented may exhibit a first-order smoothing bias, so in applications it may be useful to select a larger number of bins when including these CIs. One way of doing so is to use the mimicking-variance number of bins choice (for example, using the `binselect(esmv)` option), which we do not present here to conserve space. Alternatively, the researcher can simply "undersmooth" (that is, choose a larger number of bins) manually either by scaling up the IMSE-optimal choice with the `scale()` option or by setting the number of bins directly with the `nbins()` option.

We illustrate the other new features of the upgraded `rdplot` command in the following subsections.

7.2 Default results and backward compatibility

The upgraded `rdrobust` command works in exactly the same way as before. For example, using only the outcome and running variables, we obtain the following results with its default options. (We will refer to these default results several times in the upcoming subsections.)

```
. rdrobust vote margin
Sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c            Number of obs =       1297
-------------------+----------------------            BW type       =      mserd
     Number of obs |       595         702            Kernel        = Triangular
Eff. Number of obs |       359         322            VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.708      17.708
       BW bias (b) |    27.984      27.984
         rho (h/b) |     0.633       0.633
Outcome: vote. Running variable: margin.
--------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+------------------------------------------------------------
      Conventional |   7.416     1.4604   5.0782   0.000    4.55378      10.2783
            Robust |     -          -     4.3095   0.000    4.09441      10.9255
--------------------------------------------------------------------------------
```

The above results are not numerically identical to those reported in Calonico, Cattaneo, and Titiunik (2014a, 2015b). The differences are due to the choice of bandwidths because, as explained above, the new upgraded `rdbwselect` command is not backward compatible. Recall that, by default, the `rdrobust` command uses the companion command `rdbwselect` to select the bandwidths optimally whenever they are not specified by the user.

Nonetheless, the upgraded **rdrobust** command is backward compatible given the choice of bandwidths. To see this, we set the bandwidths manually to match those reported in Calonico, Cattaneo, and Titiunik (2014a, 2015b), which gives the following results:

```
. rdrobust vote margin, h(16.79369) b(27.43745)
Sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1297
-------------------+----------------------           BW type       =     Manual
     Number of obs |       595         702           Kernel        = Triangular
Eff. Number of obs |       343         310           VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    16.794      16.794
       BW bias (b) |    27.437      27.437
         rho (h/b) |     0.612       0.612
Outcome: vote. Running variable: margin.
--------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+------------------------------------------------------------
      Conventional |  7.4253     1.4954   4.9656   0.000    4.49446      10.3561
            Robust |     -          -     4.2675   0.000    4.06975      10.9833
--------------------------------------------------------------------------------
```

These results are equal to those reported in Calonico, Cattaneo, and Titiunik (2014a, 2015b).

7.3 Using RD plots to present treatment effects

We already showed how to incorporate local CIs in RD plots. Now we show how to use these plots to present the RD treatment effects visually, which is now possible using the new features of the **rdplot** command.

We use the default MSE-optimal RD estimate presented above, which is obtained via the command **rdrobust vote margin**. Recall that by default **rdrobust** uses a triangular kernel with a common bandwidth on both sides of the cutoff. To plot the point estimate, we can use the **rdplot** command after setting the **p()**, **kernel()**, and **h()** options appropriately.

```
. quietly rdrobust vote margin
. rdplot vote margin if -e(h_l)<= margin & margin <= e(h_r),
>       binselect(esmv) kernel(triangular) h(`e(h_l)´ `e(h_r)´) p(1)
>       graph_options(title("RD Plot: U.S. Senate Election Data")
>                     ytitle(Vote Share in Election at time t+2)
>                     xtitle(Vote Share in Election at time t)
>                     graphregion(color(white)))
RD Plot with evenly spaced mimicking variance number of bins using spacings
> estimators.
         Cutoff c = 0 | Left of c  Right of c        Number of obs  =        681
----------------------+-----------------------        Kernel         = Triangular
        Number of obs |       359         322
   Eff. Number of obs |       359         322
  Order poly. fit (p) |         1           1
     BW poly. fit (h) |    17.708      17.708
 Number of bins scale |     1.000       1.000
Outcome: vote. Running variable: margin.
----------------------------------------------
                      | Left of c  Right of c
----------------------+-----------------------
        Bins selected |        16          18
   Average bin length |     1.090       0.983
    Median bin length |     1.090       0.983
----------------------+-----------------------
    IMSE-optimal bins |         7           7
  Mimicking Var. bins |        16          18
----------------------+-----------------------
Rel. to IMSE-optimal: |
        Implied scale |     2.286       2.571
    WIMSE var. weight |     0.077       0.056
   WIMSE bias weight |     0.923       0.944
----------------------------------------------
```

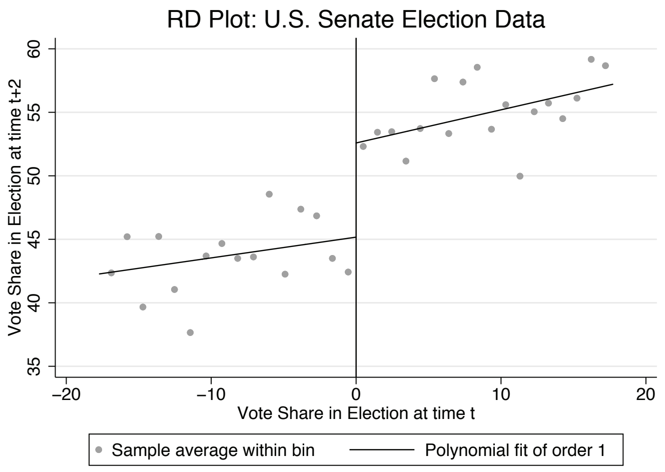


Figure 2. RD plot of treatment effect

Figure 2 is constructed using the upgraded `rdplot` command by restricting the support to the neighborhood around the cutoff defined by the choice of bandwidth h (in this example, equal on both sides). We then set the (global) fit in the RD plot to match the local polynomial point estimation conducted by `rdrobust` in that neighborhood; that is, we choose $p = 1$, $K(\cdot)$ to be the triangular kernel, and h to be the bandwidth used in the estimation, shown above. The resulting polynomial fit represents the RD point estimator exactly.

For graphical presentation purposes, we also selected in `rdplot` a mimicking-variance number of bins to exhibit the variability of the data within the window around the cutoff determined by the data-driven choice of bandwidth. In this example, the vertical distance between the two weighted linear polynomial fits is exactly 7.416, as reported in the previous section. By increasing the number of bins using the `nbins()` option, the RD plot can be used to exhibit the actual raw data instead of the average values of the outcome variable within each bin.

7.4 Robust bias-corrected inference with covariates and clustering

In this section, we illustrate the new features of the upgraded `rdrobust` command. We show how to incorporate covariates in estimation and inference and how to use cluster–robust variance estimators (with or without additional covariates).

First, we incorporate the covariates `class`, `termshouse`, and `termssenate`, keeping the neighborhood around the cutoff constant. That is, we use the same bandwidths obtained above via the default command: `rdrobust vote margin`.

```
. qui rdrobust vote margin

. local len = `e(ci_r_rb)´ - `e(ci_l_rb)´

. rdrobust vote margin, covs(class termshouse termssenate)
>          h(`e(h_l)´ `e(h_r)´) b(`e(b_l)´ `e(b_r)´)

Covariate-adjusted sharp RD estimates using local polynomial regression.

      Cutoff c = 0 | Left of c  Right of c            Number of obs =       1108
-------------------+----------------------            BW type       =     Manual
     Number of obs |       491         617            Kernel        = Triangular
Eff. Number of obs |       309         280            VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.708      17.708
       BW bias (b) |    27.984      27.984
         rho (h/b) |     0.633       0.633

Outcome: vote. Running variable: margin.
-------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+-----------------------------------------------------------
      Conventional |  6.8595     1.4165   4.8426   0.000    4.08322      9.63574
            Robust |     -          -     4.1911   0.000    3.75238       10.345
-------------------------------------------------------------------------------
Covariate-adjusted estimates. Additional covariates included: 3

. display "CI length change: "
> round(((`e(ci_r_rb)´-`e(ci_l_rb)´)/`len´-1)*100,.01) "%"
CI length change: -3.49%
```

The results above set the bandwidths manually (to be equal to the MSE-optimal bandwidths without covariates) and also compute the percentage change in the interval length of the robust bias-corrected CIs, because of the inclusion of the three additional covariates. Specifically, in this illustration, the CI length shrinks by 3.49%. The MSE-optimal point estimate (without covariates) of 7.416 changes to 6.860 when the additional covariates are included (though this change is statistically indistinguishable from 0 at conventional levels).

Second, we incorporate the same additional covariates but let rdrobust select the optimal bandwidths, which is done via rdbwselect. The data-driven bandwidths are chosen to be MSE optimal and equal on both sides of the cutoff by default.

```
. qui rdrobust vote margin
. local len = `e(ci_r_rb)´ - `e(ci_l_rb)´
. rdrobust vote margin, covs(class termshouse termssenate)
Covariate-adjusted sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1108
-------------------+----------------------           BW type       =      mserd
     Number of obs |       491         617           Kernel        = Triangular
Eff. Number of obs |       313         283           VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.987      17.987
       BW bias (b) |    28.943      28.943
         rho (h/b) |     0.621       0.621
Outcome: vote. Running variable: margin.
--------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+------------------------------------------------------------
      Conventional |  6.8514     1.4081   4.8656   0.000    4.09148      9.61125
            Robust |     -          -     4.1999   0.000    3.72856      10.2537
--------------------------------------------------------------------------------
Covariate-adjusted estimates. Additional covariates included: 3
. display "CI length change: "
> round(((`e(ci_r_rb)´-`e(ci_l_rb)´)/`len´-1)*100,.01) "%"
CI length change: -4.48%
```

In this illustration, the point estimators and the bandwidth choices change only slightly [from $\widetilde{\tau}(h) = 6.860$ and $h = 17.708$ to $\widetilde{\tau}(h) = 6.851$ and $h = 17.987$], and the interval length reduction increases because of the inclusion of the additional covariates in both point estimation and bandwidth selection (from 3.49% to 4.48%).

Third, we show that, as is well known in the literature, including covariates does not always lead to improved precision. For example, if the covariates are irrelevant, they can even increase the length of the CIs. In our illustration, the covariate population gives an example.

```
. qui rdrobust vote margin
. local len = `e(ci_r_rb)´ - `e(ci_l_rb)´
. rdrobust vote margin, covs(population)
Covariate-adjusted sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c            Number of obs =       1297
-------------------+----------------------            BW type       =      mserd
     Number of obs |       595         702            Kernel        = Triangular
Eff. Number of obs |       359         320            VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.585      17.585
       BW bias (b) |    27.857      27.857
         rho (h/b) |     0.631       0.631
Outcome: vote. Running variable: margin.
--------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+------------------------------------------------------------
      Conventional |  7.4376     1.4654   5.0754   0.000    4.56545      10.3097
            Robust |    -          -      4.3102   0.000    4.10714      10.9573
--------------------------------------------------------------------------------
Covariate-adjusted estimates. Additional covariates included: 1
. display "CI length change: "
> round(((`e(ci_r_rb)´-`e(ci_l_rb)´)/`len´-1)*100,.01) "%"
CI length change: .28%
```

As seen above, conducting covariate-adjusted RD inference with `population` as the additional covariate slightly increases the length of the resulting CIs (by roughly a quarter of a percentage point).

Fourth, as discussed above, the covariates will not affect the consistency of the RD treatment-effect estimator if they are "balanced" in the appropriate sense. For the case of sharp RD designs, "balanced" means that they should have equal conditional expectations at the cutoff. For the case of kink RD designs, "balanced" means that they should have equal first derivatives of the conditional expectations at the cutoff. This can be tested empirically.

```
. local covs "class termshouse termssenate population"
. local num: list sizeof covs
. mat balance = J(`num´,2,.)
. local row = 1
. foreach z in `covs´ {
  2.     qui rdrobust `z´ margin
  3.         mat balance[`row´,1] = round(e(tau_cl),.001)
  4.         mat balance[`row´,2] = round(e(pv_rb),.001)
  5.         local ++row
  6. }
. mat rownames balance = `covs´
. mat colnames balance = "RD Effect" "Robust p-val"
. mat lis balance
balance[4,2]
               RD Effect  Robust p-val
      class         -.021          .897
 termshouse         -.173          .561
termssenate         -.192          .901
 population    -318455.26          .634
```

Based on the empirical results above, we find that all four additional covariates have an RD treatment effect indistinguishable from 0 at conventional significance levels. In other words, we cannot reject the null hypothesis of equal conditional expectations at the cutoff. Notice that we can also use RD plots to show covariate balance at the cutoff, but we do not present these additional results to conserve space.

Fifth, the upgraded `rdrobust` command also allows for cluster–robust variance estimation, as does the underlying upgraded `rdbwselect` command used to compute data-driven bandwidth selectors. This is illustrated using the Senate data as follows, where we cluster at the `state` level using NN methods to construct the estimated residuals (recall that by default three matches per observation are used).

```
. rdrobust vote margin, vce(nncluster state)
Sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c            Number of obs =       1297
-------------------+----------------------            BW type       =      mserd
     Number of obs |       595         702            Kernel        = Triangular
 Eff. Number of obs |       359         320            VCE method    =  NNcluster
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.509      17.509
       BW bias (b) |    27.032      27.032
         rho (h/b) |     0.648       0.648
Number of clusters |        50          50
Outcome: vote. Running variable: margin.
--------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+------------------------------------------------------------
      Conventional |  7.4221     1.5225   4.8750   0.000    4.43811      10.4061
            Robust |     -          -     4.2659   0.000    4.09109      11.0456
--------------------------------------------------------------------------------
Std. Err. adjusted for clusters in state
```

In this case, the robust bias-corrected CIs change from $[4.094, 10.926]$, the (default) heteroskedasticity-robust interval reported previously, to the cluster–robust interval $[4.091, 11.046]$.

To end this section, we provide one final illustration using i) covariate-adjustment, ii) cluster–robust variance estimation, and iii) MSE-optimal bandwidth selection with (possibly) different bandwidths on either side of the cutoff.

```
. rdrobust vote margin, covs(class termshouse termssenate)
>                        bwselect(msetwo) vce(nncluster state)

Covariate-adjusted sharp RD estimates using local polynomial regression.

      Cutoff c = 0 | Left of c  Right of c            Number of obs =       1108
-------------------+----------------------            BW type       =     msetwo
     Number of obs |       491         617            Kernel        = Triangular
Eff. Number of obs |       274         310            VCE method    =  NNcluster
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    14.661      20.893
       BW bias (b) |    24.458      37.338
         rho (h/b) |     0.599       0.560
Number of clusters |        48          50

Outcome: vote. Running variable: margin.
------------------------------------------------------------------------------
            Method |   Coef.    Std. Err.    z     P>|z|    [95% Conf. Interval]
-------------------+----------------------------------------------------------
      Conventional |  6.8072     1.3696   4.9704   0.000    4.12297      9.49153
            Robust |     -          -     4.6153   0.000    4.13522      10.2398
------------------------------------------------------------------------------
Covariate-adjusted estimates. Additional covariates included: 3
Std. Err. adjusted for clusters in state
```

In this final case, we observe that the two one-sided MSE-optimal bandwidths are actually quite distinct from their common bandwidth counterpart. To be clear, these bandwidth selectors also account for the additional covariates and the clustering structure of the matrix of variances and covariances, but the numerical results show that the two bandwidths are different (for example, $\widehat{h}_{\mathtt{l}} = 14.661$ and $\widehat{h}_{\mathtt{r}} = 20.893$). The point estimate is, nonetheless, quite stable $[\widehat{\tau}(\widehat{h}_{\mathtt{l}}, \widehat{h}_{\mathtt{r}}) = 6.807]$. Finally, the robust bias-corrected covariate-adjusted cluster–robust CIs are $[4.135, 10.240]$, quite similar to the cluster–robust version reported previously.

7.5 Data-driven bandwidth selectors

As already implicitly illustrated above, the upgraded `rdbwselect` command includes several new features, such as covariate-adjusted and cluster–robust bandwidth selection. Furthermore, although not used above explicitly to conserve space, several other data-driven bandwidth selectors are now available.

In this section, we present one empirical result exhibiting all the available data-driven bandwidth selectors, in the context of our empirical illustration. We do not include additional covariates or consider cluster–robust variance estimation only for simplicity, because the main goal is to discuss the different bandwidth selectors available.

```
. rdbwselect vote margin, all
Bandwidth estimators for sharp RD local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c            Number of obs =       1297
-------------------+----------------------            Kernel        = Triangular
     Number of obs |       595         702            VCE method    =         NN
     Min of margin |  -100.000       0.036
     Max of margin |    -0.079     100.000
     Order est. (p) |         1           1
     Order bias (q) |         2           2
Outcome: vote. Running variable: margin.
-------------------+-------------------------------+-------------------------------
                   |         BW est. (h)           |         BW bias (b)
            Method | Left of c          Right of c | Left of c          Right of c
-------------------+-------------------------------+-------------------------------
             mserd |    17.708              17.708 |    27.984              27.984
            msetwo |    16.154              18.009 |    27.096              29.205
            msesum |    18.326              18.326 |    31.280              31.280
          msecomb1 |    17.708              17.708 |    27.984              27.984
          msecomb2 |    17.708              18.009 |    27.984              29.205
-------------------+-------------------------------+-------------------------------
             cerrd |    12.374              12.374 |    27.984              27.984
            certwo |    11.288              12.585 |    27.096              29.205
            cersum |    12.806              12.806 |    31.280              31.280
          cercomb1 |    12.374              12.374 |    27.984              27.984
          cercomb2 |    12.374              12.585 |    27.984              29.205
-------------------+-------------------------------+-------------------------------
```

The output shows all the different bandwidth selectors available for estimation and inference in RD designs. These options are also available in the case of covariate-adjustment or cluster–robust variance estimation. The first group considers MSE-optimal bandwidths, while the second group considers CER-optimal bandwidths, both following the methodology discussed in previous sections.

Among these choices, the most useful ones are i) `mserd` for MSE-optimal point estimation using a common bandwidth on both sides of the cutoff, ii) `msetwo` for MSE-optimal point estimation using two distinct common bandwidths on either side of the cutoff, iii) `cerrd` for robust bias-corrected CIs with faster coverage error decay rates using a common bandwidth on both sides of the cutoff, and iv) `certwo` for robust bias-corrected CIs with faster coverage error decay rates using two distinct common bandwidths on either side of the cutoff. The other options are useful for regularization and sensitivity analysis purposes.

Finally, recall that the results above include regularization, as introduced in Imbens and Kalyanaraman (2012) but implemented as discussed in Calonico, Cattaneo, and Titiunik (2014a, 2014b, 2015b) and the corresponding supplemental appendix. Including this regularization term always leads to smaller bandwidths; it can be modified or removed with the `scaleregul()` option. In particular, it can be removed by simply adding `scaleregul(0)`.

7.6 Other examples and RD designs

The companion replication file (`rdrobust_illustration.do`) includes the syntax of all the examples discussed above, as well as the following additional examples:

1. `rdrobust vote margin, kernel(uniform) vce(cluster state)`
   Robust bias-corrected inference using a uniform kernel and clustering with plug-in residuals at `state` level. This is the command that produces the closest results to those obtained using the Stata built-in `regress` command with clustering to construct the RD point estimator. (Without clustering, the most similar results to those obtained with `regress` are obtained using the `vce(hc1)` option.)
2. `rdrobust vote margin, bwselect(certwo) vce(hc3)`
   Robust bias-corrected inference using the CER-optimal bandwidth choice, allowing for a different bandwidth on each side of the cutoff, and HC3 heteroskedasticity-robust variance estimation.
3. `rdrobust vote margin, h(12 15) b(18 20)`
   Robust bias-corrected inference with user-chosen bandwidths $(h_{\mathtt{l}}, h_{\mathtt{r}}) = (12, 15)$ and $(b_{\mathtt{l}}, b_{\mathtt{r}}) = (18, 20)$.
4. `rdrobust vote margin, covs(class) bwselect(cerrd) scaleregul(0) rho(1)`
   Robust bias-corrected inference using covariate adjustment with a single covariate (`class`), CER-optimal common bandwidth selector, no regularization, and $h = b$ (that is, $\rho = 1$).
5. `rdbwselect vote margin, kernel(uniform) vce(cluster state) all`
   All data-driven bandwidth selectors using uniform kernel and clustering with plug-in residuals at `state` level.
6. `rdbwselect vote margin, covs(class) bwselect(msetwo) vce(hc2) all`
   MSE-optimal bandwidth selectors on either side of the cutoff adjusting by covariate `class` and using HC2 heteroskedasticity-robust variance estimation.

Finally, we discuss how to implement other RD designs. Let `y` be the outcome variable, `t` the treatment status variable, `x` the running variable, `z` a "preintervention" covariate, and `cid` a cluster ID variable.

1. `rdrobust y x, deriv(1) covs(z) vce(nncluster cid)`
   Sharp kink RD with additional covariates and clustering.
2. `rdrobust y x, fuzzy(t) covs(z) vce(nncluster cid)`
   Fuzzy RD with additional covariates and clustering.
3. `rdrobust y x, fuzzy(t) deriv(1) covs(z) vce(nncluster cid)`
   Fuzzy kink RD with additional covariates and clustering.

8 Conclusion

In this article, we discussed a major upgrade to the `rdrobust` package for Stata and R, which provide general-purpose software for regression-discontinuity designs. The main new features of this upgraded version are i) covariate-adjusted bandwidth selection, point estimation, and robust bias-corrected inference, ii) clustered-consistent bandwidth selection, point estimation, and robust bias-corrected inference, iii) weighted global polynomial fits and pointwise confidence bands in RD plots, and iv) several new bandwidth selection methods, including different bandwidths for control and treatment groups, CER optimal bandwidths, and optimal bandwidth for fuzzy designs. We provided a detailed account of all technical and methodological results implemented in CCFT and its supplemental appendix.

A companion R package with the same functionality and syntax is also available.

9 Acknowledgments

We thank Quentin Brummet, David Drukker, Brian Jacob, Max Kapustin, Louis-Pierre Lepage, Xinwei Ma, Zhuan Pei, Vincent Pons, and Gonzalo Vazquez-Bare for useful comments and discussions that improved this manuscript as well as our implementations. We also thank an anonymous reviewer for useful comments on an early version of this paper. Cattaneo gratefully acknowledges financial support from the National Science Foundation through grants SES-1357561 and SES-1459931. Titiunik gratefully acknowledges financial support from the National Science Foundation through grant SES-1357561.

10 References

Abadie, A., and G. W. Imbens. 2008. Estimation of the conditional variance in paired experiments. *Annales d'Économie et de Statistique* 91/92: 175–187.

Bartalotti, O., and Q. Brummet. 2017. Regression discontinuity designs with clustered data. In *Advances in Econometrics: Vol. 38—Regression Discontinuity Designs: Theory and Applications*, ed. M. D. Cattaneo and J. C. Escanciano, 383–420. Bingley, UK: Emerald.

Calonico, S., M. D. Cattaneo, and M. H. Farrell. 2016a. Coverage error optimal confidence intervals for regression discontinuity designs. Working Paper, University of Michigan. http://www-personal.umich.edu/~cattaneo/papers/Calonico-Cattaneo-Farrell_2016_wp.pdf.

———. Forthcoming. On the effect of bias estimation on coverage accuracy in nonparametric inference. *Journal of the American Statistical Association*.

Calonico, S., M. D. Cattaneo, M. H. Farrell, and R. Titiunik. 2016b. Regression discontinuity designs using covariates. Working Paper, University of Michigan. http://www-personal.umich.edu/∼cattaneo/papers/Calonico-Cattaneo-Farrell-Titiunik_2016_wp.pdf.

Calonico, S., M. D. Cattaneo, and R. Titiunik. 2014a. Robust data-driven inference in the regression-discontinuity design. *Stata Journal* 14: 909–946.

———. 2014b. Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica* 82: 2295–2326.

———. 2015a. Optimal data-driven regression discontinuity plots. *Journal of the American Statistical Association* 110: 1753–1769.

———. 2015b. rdrobust: An R package for robust nonparametric inference in regression-discontinuity designs. *R Journal* 7: 38–51.

Cameron, A. C., and D. L. Miller. 2015. A practitioner's guide to cluster–robust inference. *Journal of Human Resources* 50: 317–372.

Card, D., D. S. Lee, Z. Pei, and A. Weber. 2015. Inference on causal effects in a generalized regression kink design. *Econometrica* 83: 2453–2483.

Cattaneo, M. D., and J. C. Escanciano, eds. 2017. *Regression Discontinuity Designs: Theory and Applications (Advances in Econometrics, vol. 38)*. Bingley, UK: Emerald.

Cattaneo, M. D., and M. H. Farrell. 2013. Optimal convergence rates, Bahadur representation, and asymptotic normality of partitioning estimators. *Journal of Econometrics* 174: 127–143.

Cattaneo, M. D., B. R. Frandsen, and R. Titiunik. 2015. Randomization inference in the regression discontinuity design: An application to party advantages in the U.S. Senate. *Journal of Causal Inference* 3: 1–24.

Cattaneo, M. D., M. Jansson, and X. Ma. 2017. rddensity: Manipulation testing based on density discontinuity. https://sites.google.com/site/rdpackages/rddensity.

Cattaneo, M. D., R. Titiunik, and G. Vazquez-Bare. Forthcoming. Comparing inference approaches for RD designs: A reexamination of the effect of head start on child mortality. *Journal of Policy Analysis and Management*.

Cattaneo, M. D., and G. Vazquez-Bare. 2016. The choice of neighborhood in regression discontinuity designs. *Observational Studies* 2: 134–146.

Imbens, G. W., and K. Kalyanaraman. 2012. Optimal bandwidth choice for the regression discontinuity estimator. *Review of Economic Studies* 79: 933–959.

Imbens, G. W., and T. Lemieux. 2008. Regression discontinuity designs: A guide to practice. *Journal of Econometrics* 142: 615–635.

Lee, D. S., and T. Lemieux. 2010. Regression discontinuity designs in economics. *Journal of Economic Literature* 48: 281–355.

MacKinnon, J. G. 2013. Thirty years of heteroskedasticity-robust inference. In *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis: Essays in Honor of Halbert L. White Jr.*, ed. X. Chen and N. R. Swanson, 437–461. New York: Springer.

Müller, H.-G., and U. Stadtmuller. 1987. Estimation of heteroscedasticity in regression analysis. *Annals of Statistics* 15: 610–625.

About the authors

Sebastian Calonico is an assistant professor of economics at the University of Miami.

Matias D. Cattaneo is a professor of economics and a professor of statistics at the University of Michigan.

Max H. Farrell is an assistant professor of econometrics and statistics and is a John E. Jeuck faculty fellow at the University of Chicago Booth School of Business.

Rocío Titiunik is the James Orin Murfin associate professor of political science at the University of Michigan.

The Stata Journal (2017)
17, Number 2, pp. 405–421

A combined test for a generalized treatment effect in clinical trials with a time-to-event outcome

Patrick Royston
MRC Clinical Trials Unit
University College London
London, UK
j.royston@ucl.ac.uk

Abstract. Most randomized controlled trials with a time-to-event outcome are designed and analyzed assuming proportional hazards of the treatment effect. The sample-size calculation is based on a log-rank test or the equivalent Cox test. Nonproportional hazards are seen increasingly in trials and are recognized as a potential threat to the power of the log-rank test. To address the issue, Royston and Parmar (2016, *BMC Medical Research Methodology* 16: 16) devised a new "combined test" of the global null hypothesis of identical survival curves in each trial arm. The test, which combines the conventional Cox test with a new formulation, is based on the maximal standardized difference in restricted mean survival time (RMST) between the arms. The test statistic is based on evaluations of RMST over several preselected time points. The combined test involves the minimum p-value across the Cox and RMST-based tests, appropriately standardized to have the correct null distribution. In this article, I outline the combined test and introduce a command, `stctest`, that implements the combined test. I point the way to additional tools currently under development for power and sample-size calculation for the combined test.

Keywords: st0479, stctest, randomized controlled trial, time-to-event outcome, restricted mean survival time, treatment effect, hypothesis testing, flexible parametric model, jackknife

1 Introduction

Most randomized controlled trials with a time-to-event outcome are designed and analyzed assuming proportional hazards (PH) of the treatment effect. The sample-size calculation is based on a log-rank test or the equivalent Cox test. However, nonproportional hazards (non-PH) are increasingly recognized as an issue (for example, Trinquart et al. [2016]). Significant non-PH may be present in about a quarter of cancer trials (Trinquart et al. 2016). Nonstatistically significant, but still practically important non-PH are likely to be present in a much larger proportion of trials, particularly as trial sample size and follow-up time tend to increase, conferring higher power to detect non-PH.

Possible reasons for non-PH include treatments that really do have time-dependent effects. For example, a research treatment given only over a limited period may be effective early but wear off later. Alternatively, a treatment may have no effect for a relatively long period after randomization but "kick in" further on, which is a "late effect" of a type sometimes seen in prevention trials and screening trials. Treatments with different modes of action, such as surgery, drug treatment, and watchful waiting, may induce non-PH because of dissimilarity between the shapes of the hazard functions in the control and research arms. Additionally, the presence of a subpopulation with a differential response to the research treatment may distort the survival curve.

Concerns about use of the hazard ratio (HR) as a summary measure and as the basis of a test of the treatment effect in such trials include poor interpretability and potential loss of power of the associated Cox or log-rank test. Difference (or ratio) in restricted mean survival time (RMST) between treatment groups is gaining popularity as a summary measure and as the basis of a possible test of a treatment effect. RMST at some time point ($t^* > 0$) is the integral of the survival function at t^*, that is, the "area under the survival curve" from 0 to t^*. It is interpreted as the mean of the survival-time distribution truncated at t^*. The difference, ΔRMST, defined as RMST in a research arm minus RMST in the control arm, is the integrated difference between the survival functions—in other words, the (signed) area between the survival curves up to t^*. Conventionally, a "large" positive value of ΔRMST is regarded as a "good" trial outcome because it represents an extension of survival time because of the research regimen, at least up to t^*. Further details and an implementation of RMST and ΔRMST in the user-written `strmst` command may be found in Royston (2015); also see the `strmst2` command (Cronin, Tian, and Uno 2016).

One might surmise that, with a suitable choice of t^*, ΔRMST divided by its standard error (SE) might provide a useful test statistic for the "global" null hypothesis $H_0: S_0(t) = S_1(t)$ for any $t > 0$, where $S_j(t)$ is the survival function in the jth group ($j = 0, 1$) with $j = 0$ denoting the control group. The problem is the choice of t^*. A single value is fragile regarding power. To protect power, one would prefer to test over a range of t^* values. Recognizing such a requirement, Royston and Parmar (2016) proposed a test of H_0 based on evaluating the maximal chi-squared statistic $C_{\max} = \max\left(Z^2\right)$ over several time points, where $Z = \Delta\text{RMST}/\text{SE}(\Delta\text{RMST})$. Arguing pragmatically, Royston and Parmar (2016) determined $C_{\max}$ over 10 equally spaced values of t^* between the 30th and 100th centiles of the failure times in the dataset. Starting with $C_{\max}$, they developed an approach to testing H_0 that they called the "combined test".

My principal aim here is to present a new command, `stctest`, that implements the combined test. In section 2, I outline the methodological steps leading to the combined test and describe different "flavors" of the test. In section 3, I present the `stctest` command. In section 4, I apply the methodology to an example trial dataset. Section 5 is a discussion.

2 Methods

2.1 Estimation of RMST

Estimation of RMST at some time t^* requires determining the area under the survival curve from 0 to t^*. I consider two methods: i) `ps`, using jackknife estimation of pseudovalues (Andersen, Hansen, and Klein 2004), equivalent to integrating the Kaplan–Meier curve; and ii) `rp`, integration of smooth survival curves predicted from flexible parametric survival models (Royston and Parmar 2002; Lambert and Royston 2009; Royston and Lambert 2011), also known as Royston–Parmar (RP) models. Next, I briefly describe these methods.

Jackknife estimation from pseudovalues

Andersen, Hansen, and Klein (2004) described the use of "pseudo-observations" (I call them pseudovalues) to provide nonparametric estimates of RMST at the individual participant level. Pseudovalues are leave-one-out (jackknife) estimates of a parameter of interest, here the RMST, constructed in such a way that their sample mean estimates the RMST. They are computed from the Kaplan–Meier estimate of the survival curve for the sample. The effects of covariates on the RMST may be modeled with the pseudovalues as the response variable in generalized linear models with a suitable link function. Standard errors of parameter estimates use the robust "sandwich" estimator in Stata through the `robust` estimation option. Because pseudovalues are based on Kaplan–Meier estimates, they are distribution free.

In Stata, pseudovalues for RMST are available through the user-written `stpmean` command (Parner and Andersen 2010; Overgaard, Andersen, and Parner 2015). A treatment effect can be estimated by a command of the form `regress` *psvar trtvar*`, robust`. The response variable, *psvar*, contains the pseudovalues for some t^*, as estimated by `stpmean`. The regression coefficient for *trtvar* estimates the arithmetic difference in RMST between the treatment groups.

RP models

Conceptually, RP models fit the baseline distribution function explicitly using a suitable smoother; Royston and Parmar (2002) chose restricted cubic spline functions. Effects of covariates $\mathbf{x}$ are accommodated in generalized linear models of the form

$$g_\theta\left\{S\left(t;\mathbf{x}\right)\right\} = g_\theta\left\{S_0\left(t\right)\right\} + \mathbf{x}\boldsymbol{\beta}$$

where $S\left(t;\mathbf{x}\right)$ and $S_0\left(t\right)$ are the survival and baseline survival functions, respectively, and $g_\theta\left(\cdot\right)$ is a monotonic link function. See Royston and Lambert (2011, 118–119) for further details of this class of models.

Here I use the subclass with a complementary log-log link function, defined by $g_\theta\left\{S\left(t;\mathbf{x}\right)\right\} = \ln\left\{-\ln S\left(t;\mathbf{x}\right)\right\} = \ln\left\{H\left(t;\mathbf{x}\right)\right\}$, the log cumulative-hazard function:

$$\ln H\left(t;\mathbf{x}\right) = \ln H_0\left(t\right) + \mathbf{x}\boldsymbol{\beta}$$

The function $\ln H_0(t)$ is modeled using restricted cubic splines in $\ln t$, the complexity of which is determined by the number and position of user-selected interior knots (polynomial join points). If the covariate effects $\boldsymbol{\beta}$ are independent of time, as in the above expression, the formulation gives a parametric PH model.

RP models are easily extended to include non-PH covariate effects; see Royston and Lambert (2011, sect. 7.6). In a randomized controlled trials context, time-dependent effects are achieved by fitting different spline functions in each treatment group. Depending on the degree of freedom (d.f.) chosen for the spline functions, the approach potentially provides sufficient flexibility to represent many varieties of non-PH patterns. Here I suggest using a relatively complex spline model with five d.f. (equivalent to four internal knots) in each treatment group, providing estimates of the treatment effect that are comparably flexible with those from the method based on pseudovalues.

In general, the tool recommended for fitting RP models in Stata is **`stpm2`** (Lambert and Royston 2009). Estimation of RMST after fitting an RP model with **`stpm2`** is straightforward. One uses the **`rmst`** option of **`predict`** together with a second option, **`tmax(`**$\#$**`)`**, to define t^*. Standard errors and confidence intervals (CIs) are supported through the **`stdp`** and **`ci`** options. Further useful options are **`at()`** to predict RMST at specific values of covariates and **`zeros`** to predict at baseline (all covariates set to zero). For applications to trials, please see the user-written **`strmst`** command (Royston 2015), which conveniently packages RMST calculations from RP models.

2.2 Approximate combined test

The motivation for $C_{\max}$ (defined in the *Introduction*) as the basis of a test of the treatment effect is to try to identify the largest standardized treatment effect over a relevant time interval. Because of multiple testing at 10 time points, the null distribution of $C_{\max}$ is not central chi-squared on 1 d.f. To correct for multiplicity and arrive at a usable test statistic, Royston and Parmar (2016) took the following steps:

1. To estimate a p-value associated with $C_{\max}$, they first create M values of $C_{\max}$ in the null case. This is done by randomly permuting the treatment label in the given dataset, thereby "scrambling" any treatment-outcome association. $C_{\max}$ is calculated in each permuted sample; call the resulting values $C_1, \ldots, C_M$. Suppose that $r \geq 0$ of the C_i exceeds $C_{\max}$. The larger r is, the weaker the evidence that $C_{\max}$ is "extreme" and the larger the corresponding p-value. Their continuity-corrected estimate of the p-value is $p_{\text{perm}} = \{r + (1/2)\}/(M+1)$. (Note that r can take any of the $M+1$ values $0, 1, \ldots, M$.) The smallest p_{perm} that can result with a given M is $1/(2M+2)$. A binomial-based exact CI for r/M may be used to calculate a CI for p_{perm}.
2. Such a permutation test has a stochastic element. Let $p_{\max}$ be the p-value corresponding to $C_{\max}$ according to a chi-squared distribution on one d.f. In Stata terms, $p_{\max} =$ **`chi2tail(1,`**$C_{\max}$**`)`**. To stabilize the test, they used simulations based on several real datasets to derive $\widetilde{p}_{\text{perm}}$, an empirical approximation to p_{perm} as a function of $p_{\max}$,

$$\widetilde{p}_{\text{perm}} = 1.762\,(p_{\max})^{0.885} - 0.802\,(p_{\max})^{2.547}$$

3. Next, they developed a new test combining $\widetilde{p}_{\text{perm}}$ with the Cox test p-value, p_{Cox}. The aim was to capitalize on the strengths of each test across a range of patterns of survival curves, including PH and non-PH examples. They defined

$$p_{\min} = \min\left(p_{\text{Cox}}, \widetilde{p}_{\text{perm}}\right)$$

   Although the individual null distributions of p_{Cox} and $\widetilde{p}_{\text{perm}}$ are approximately uniform on $(0, 1)$, p_{Cox} and $\widetilde{p}_{\text{perm}}$ are correlated, and the null distribution of $p_{\min}$ is not expected to be uniform. They approximated the null distribution of $p_{\min}$ empirically using a two-parameter beta distribution. To calculate an approximate p-value, $\widetilde{p}_{CT}$, from a given $p_{\min}$, they applied the formula (in Stata terms) $\widetilde{p}_{CT}$ = `ibeta(`$a, b, p_{\min}$`)`, where `ibeta(`a, b, x`)` is the incomplete beta function with parameters a, b and argument x $(0 < x < 1)$. For the two-sided test, they estimated $a = 1$, $b = 1.5$.

Royston and Parmar (2016) did not provide an expression for $\widetilde{p}_{CT}$ for use in one-sided tests. However, in subsequent work using similarly constructed simulations, they obtained the following two-parameter beta approximation to the null distribution of $p_{\min}$ for the one-sided test: $a = 0.9642$, $b = 1.2581$.

2.3 Permutation combined test

Description

In an analysis of simulations based on data from 20 selected randomized trials, Royston and Parmar (2016) showed that $\widetilde{p}_{CT}$ maintained approximately the correct significance level in the null case of no treatment effect. However, the possibility of heterogeneity remained in other (unconsidered) trials, meaning that $\widetilde{p}_{CT}$ might be (slightly) too large or (slightly) too small in some trials. In critical cases, this might matter. Ensuring the integrity of a p-value for a treatment effect in a randomized trial is important.

With such a motivation, I extend the permutation approach used with $C_{\max}$ to create a permutation-based combined test, as follows:

1. Determine $C_{\max}$, $p_{\max}$, $\widetilde{p}_{\text{perm}}$ (but not p_{perm}), and p_{Cox} on the original dataset, as described above. Note that none of these quantities is stochastic. Let $p_{\min}^{\text{orig}} = \min\left(p_{\text{Cox}}, \widetilde{p}_{\text{perm}}\right)$.

2. Determining a permutation p-value, p_{CT}, for the combined test rests on assessing the relative position of $p_{\min}^{\text{orig}}$ in the null (permutation) distribution of $p_{\min}$. The method is similar to the determination of p_{perm} given above.

3. In each of M samples with a random permutation of the treatment label, determine $p_{\max}$, $\widetilde{p}_{\text{perm}}$, p_{Cox}, and hence $p_{\min}$, thus establishing a sample of size M from the permutation distribution of $p_{\min}$.

4. Calculate the permutation combined test as $p_{CT} = \{r + (1/2)\}/(M+1)$, where r is the number of samples in which $p_{\min}$ is smaller (that is, "more significant") than $p_{\min}^{\text{orig}}$. A CI for p_{CT} may be found via a binomial based CI for r/M and some simple algebra.

Validation of type 1 errors

If the nonstochastic combined test $\widetilde{p}_{CT}$ has the correct type 1 error probability, the expected proportion of M samples with a random permutation of the treatment label in which $\widetilde{p}_{CT} < \alpha$ should be α for any trial and choice of α. Here α is interpreted as the nominal significance level and is the appropriate critical value for the test. I tested this important characteristic through a heterogeneity chi-squared statistic, defined for a given α by $C_{H;\alpha} = \sum_{j=1}^{20} (p_{j;\alpha} - \alpha)^2 / \text{var}(p_{j;\alpha})$. For trial j, $p_{j;\alpha}$ denotes the proportion of M samples in which $\widetilde{p}_{CT} < \alpha$; and $\text{var}(p_{j;\alpha}) = p_{j;\alpha}(1 - p_{j;\alpha})/M$. If $E(p_{j;\alpha}) = \alpha$ for each j, then $C_{H;\alpha}$ is distributed approximately as central chi-squared on 20 d.f.

For each of 20 trials, I created $M = 5000$ permutation samples. I estimated $p_{\min}$ and hence $\widetilde{p}_{CT} = \texttt{ibeta(}a, b, p_{\min}\texttt{)}$ in each sample, thus generating 100,000 observations of $\widetilde{p}_{CT}$ for the two-sided combined tests.

For conventional values $\alpha \in \{0.01, 0.025, 0.05, 0.1\}$, I estimated $p_{j;\alpha}$ as the proportion of $M = 5000$ values so that $\widetilde{p}_{CT} < \alpha$ in trial j $(j = 1, \ldots, 20)$. Aside from chance variation, the $p_{j;\alpha}$ should be consistent with α. Figure 1 shows the $p_{j;\alpha}$ with 95% CIs for the two-sided combined test.

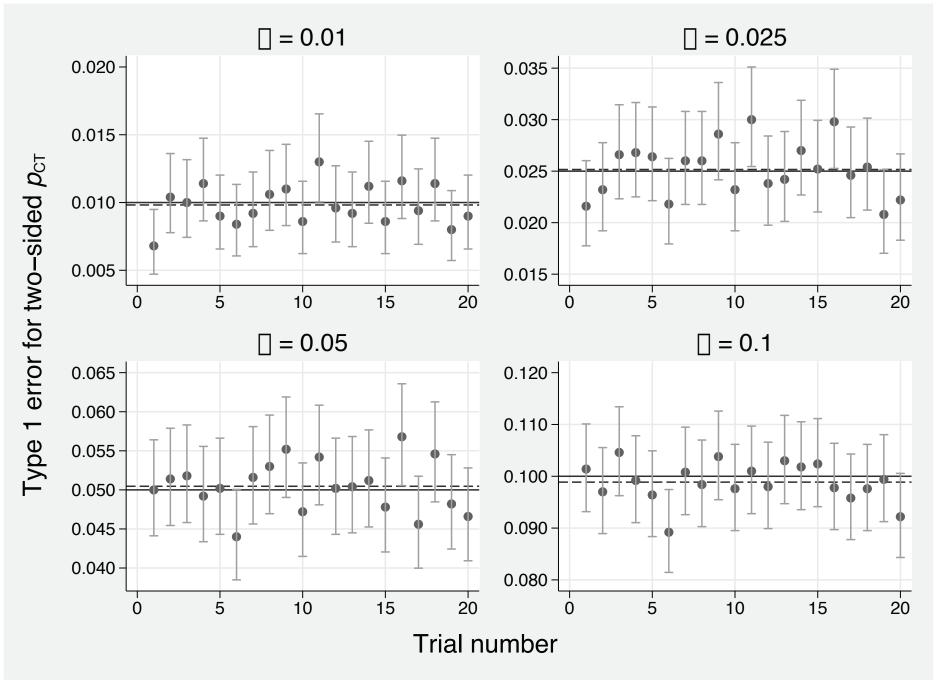


Figure 1. Empirical type 1 error probabilities $p_{1;\alpha}, \ldots, p_{20;\alpha}$ with 95% CIs for the two-sided combined test across 20 trials. Solid horizontal lines show critical values (α), and dashed lines show the corresponding p_α in the pooled dataset of 100,000 samples with a randomly permuted treatment label.

For each α, the $p_{j;\alpha}$ are scattered seemingly randomly around α and lie within about two SEs errors of α. For the pooled sample ($M = 100000$), p_α is close to α. The heterogeneity chi-squared ($C_{H;\alpha}$) is not significant at the 5% level for any of the four illustrated values of α.

Figure 2 shows an analogous plot for the one-sided combined test.

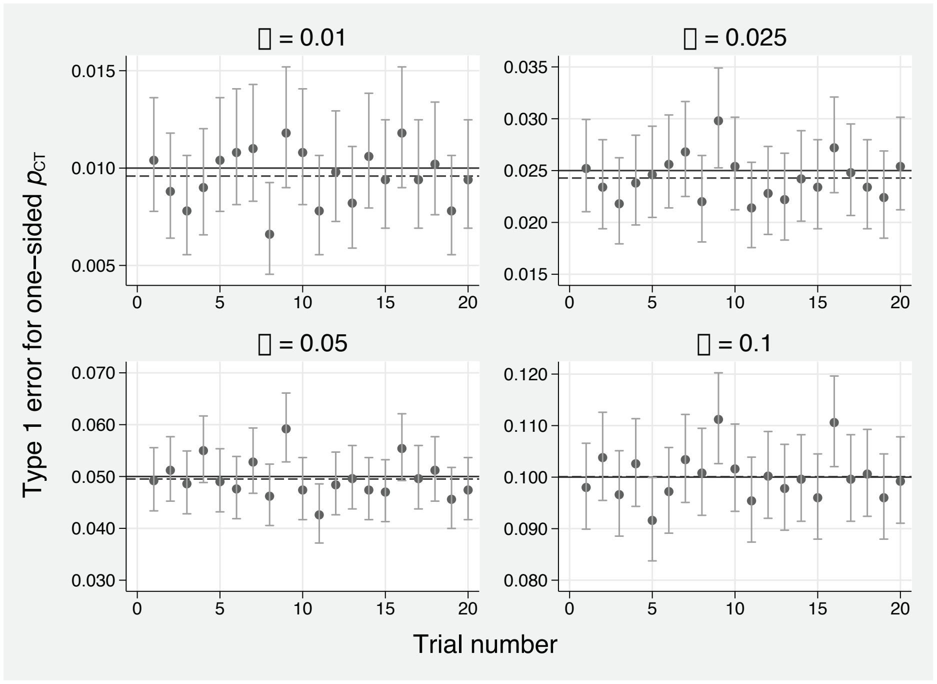


Figure 2. Empirical type 1 error probabilities $p_{1;\alpha}, \ldots, p_{20;\alpha}$ with 95% CIs for the one-sided combined test across 20 trials. Solid horizontal lines show critical values (α), and dashed lines show the corresponding p_α in the pooled dataset of 100,000 samples with a randomly permuted treatment label.

The values of $p_{j;\alpha}$ are again generally close to α. None of the four heterogeneity chi-squared is significant at the 5% level.

I conclude that the approximations that lead to $\widetilde{p}_{CT}$ for the two-sided and one-sided combined tests appear to work well. Nevertheless, the permutation combined test, p_{CT}, provides an important "safety net" for use in critical cases, for example, when $\widetilde{p}_{CT}$ is close to a critical value such as $\alpha = 0.05$.

3 The stctest command

The syntax of `stctest` is as follows:

`stctest` {`ps`|`rp`} *trt_varname* [*if*] [*in*] [`,` `adjust(`*adj_varlist*`)` `compare(`*#1 #2*`)`
`detail` `df(`*#*`)` `dftvc(`*df_list*`)` `nperm(`*#*`)` `onesided(+|-)`]

Note that before all the features of `stctest` can be used, two programs must be installed: `stpmean` and `stpm2`. `stpmean` may be installed from the *Stata Journal* website using the commands

```
    . net sj 15-3 st0202_1
    . net install st0202_1
```

Also, `stpm2` may be installed or updated from the Statistical Software Components archive using the command

```
    . ssc install stpm2, replace
```

Important: Please ensure you install (or update to) the most recent version of `stpm2`, as above.

3.1 Description

`stctest ps` and `stctest rp` carry out a combined significance test (Royston and Parmar 2016) of a generalized treatment effect comparing level *#1* of *trt_varname* with level *#2*. The data are assumed to arise from a randomized controlled trial with a time-to-event outcome and assumed to have been `stset`. Usually, *#1* denotes the control arm and *#2* a research arm. *trt_varname* may contain more than two levels (treatment arms), but `stctest` compares only selected pairs of levels as determined by the `compare(`*#1 #2*`)` option. Typically, a research treatment (a novel regimen under investigation) is compared with a control arm (standard of care or some other kind of reference therapy such as a placebo).

The combined test combines a standard log-rank or Cox test (implemented through `stcox` *trt_varname*) with a statistic derived from the maximal squared standardized between-arm difference in time-dependent RMST. Further details are given above and in the help file under *Remarks*.

`stctest ps` carries out the combined test "nonparametrically" using RMST "pseudovalues" calculated by the user-written `stpmean` command (Parner and Andersen 2010; Overgaard, Andersen, and Parner 2015). Pseudovalues are jackknife quantities derived from the Kaplan–Meier survival function and constructed so that their arithmetic mean estimates the RMST at a given time point, t^*.

`stctest rp` carries out the combined test "parametrically" using estimates of RMST derived from an RP model (Royston and Parmar 2002; Royston and Lambert 2011) fit by `stpm2` (Lambert and Royston 2009). Regression parameters are defined on the scale of the log cumulative-hazard function. To allow for the possibility of non-PH, the model includes a time-dependent treatment effect.

3.2 Options

I describe the more important options here. Lesser used options `df()` and `dftvc()` are described in the help file.

`adjust(`*adj_varlist*`)` adjusts RMST for variables in *adj_varlist*, allowed to be any mixture of binary, continuous, and factor variables. Note that `stctest ps` and `stctest rp` adjust differently for covariates. In both flavors, the Cox component of the combined test is a PH model that includes *trt_varname* and variables in *adj_varlist*. The RMST test component in `stctest ps` includes *trt_varname* and *adj_varlist* in multiple linear regression models for the RMST pseudovalues at the different time points. Thus `stctest ps` incorporates linear adjustment for covariates on the scale of RMST. In contrast, `stctest rp` includes *adj_varlist* in the hazards-scaled RP model that incorporates a time-dependent effect (non-PH) for treatment. Thus `stctest rp` adjusts linearly for covariates on the log cumulative-hazard scale, which, in the absence of time-dependent effects, is a PH model for these variables.

`compare(`*#1 #2*`)` selects the levels of the treatment variable to be compared. Usually *#1* denotes the control arm and *#2* a research arm. The default is `compare(0 1)`.

`detail` reports results of the component Cox and RMST tests in addition to p_{CT}, the p-value for the primary test (the combined test).

`nperm(`#`)` changes the mechanics of `stctest ps` and `stctest rp` so that the null distribution of the combined test statistic is derived directly from a permutation test procedure. Using the Stata `permute` command, the treatment covariate is randomly permuted # times, and the combined test is performed in each permuted dataset, providing multiple values of $\widetilde{p}_{\text{perm}}$, p_{Cox}, and p_{min}. The ensemble constitutes a sample from the permutation distribution of p_{min}. The relative position of $p_{\text{min}}^{\text{orig}}$, the test statistic from the original data, in the permutation distribution of p_{min} estimates p_{CT} for the combined permutation test.

Using `nperm(`#`)` with $\# > 0$ allows one to estimate a p-value for the combined test that does not rely on empirical approximations. However, the variance of such a p-value may be large. If a p-value with a "narrow" CI is desired, a "large" value of # will be required, for example, 5,000 or more. Computation time increases linearly with #. Computation times with `stctest rp` will be particularly long, so the approach should be used only when absolutely needed.

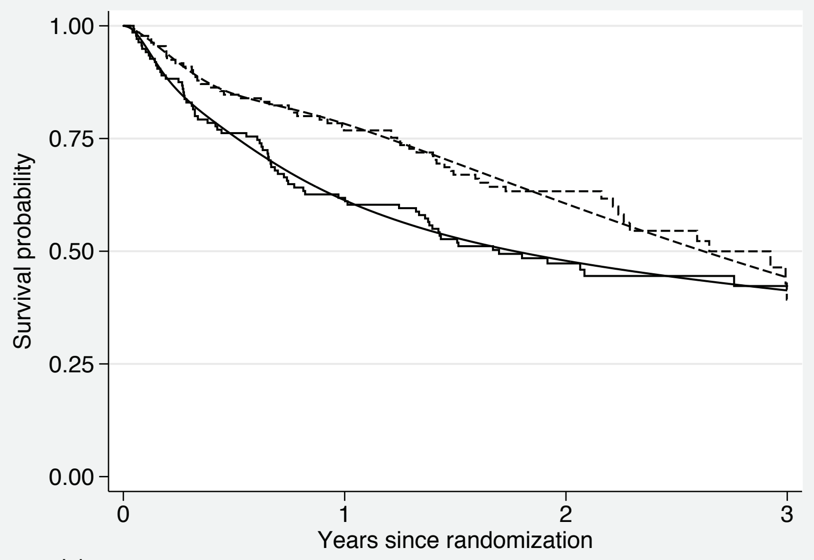
The default is `nperm(0)`, meaning that the combined test p-value, p_{CT}, is obtained nonstochastically through a beta distribution approximation (see Royston and Parmar [2016]).

`onesided(+|-)` performs one-sided tests of the treatment effect. For the Cox test, one-sided p-values are reported. With `onesided(+)`, the direction of the test is that lower HRs have smaller p-values, because HR < 1 in most trials represents a "positive" test result. `onesided(-)` may be appropriate when the event of interest is a "good" outcome, for example, time to wound healing. With `onesided(+)`, the RMST test responds to RMST being higher in the research arm than the control arm, and vice versa for `onesided(-)`. In most trials, an increase in the mean time to event is a "good" outcome. The default is `onesided()`; that is, the option is unspecified, meaning that all tests are two sided.

4 Example

An interesting example is the PATCH1 trial of treatment for cellulitis of the leg, a common bacterial infection of the skin and underlying tissue (Thomas et al. 2013). In a prophylaxis phase, 274 patients were randomly assigned to placebo or treatment with penicillin. One of the main outcomes of interest was time to first disease recurrence during a no-intervention follow-up period. Only one event occurred after three years. For presentation purposes, follow-up time was truncated at three years. There were 128 first recurrences and 146 censored observations.

Figure 3 shows estimated "survival" curves by treatment group.



Number at risk							
Placebo	138	(51)	81	(18)	36	(3)	17
Penicillin	136	(30)	97	(16)	44	(10)	11

Figure 3. PATCH1 trial: Survival curves for time to first recurrence of cellulitis by treatment group. Unbroken lines, placebo group; dashed lines, penicillin group. Jagged lines, Kaplan–Meier curves; smooth curves, estimates from an RP model. Values in parentheses denote number of events in the corresponding time interval.

The Cox test of the treatment effect "just fails to achieve significance" at conventional levels, with $p = 0.052$. However, the Kaplan–Meier curves suggest a clear difference between treatments. The median time to recurrence of cellulitis increases by almost one year, from 1.70 years on placebo to 2.65 years on penicillin (difference = 0.95, SE = 0.41, $p = 0.021$). Applying the combined test with the `ps` (pseudovalues) method produces the following result. I include the `detail` option to see the component test results:

```
. use patch1
(PATCH1 trial (public release version), PR May 2016)
. stctest ps trt, detail
       Treatment |   Obs      p(CT)*
-----------------+-------------------
        trt(0,1) |   274    0.023746
* Non-stochastic, from approximation to permutation test
p-values from tests underlying p(CT):
  p(Cox)    p(chi2)    p(perm)     p(min)
-----------------------------------------
0.051835   0.004893   0.015894   0.015894
```

The combined test is significant at the 0.05 level ($p = 0.02375$), similar to the test of medians. As discussed, the uncorrected p-value for the test of RMST differences is "too small" (0.00489). After correction, it is 0.01589, which is significant at the 0.05 level. `p(min)`, the smaller of the Cox and approximate permutation test p-values, is 0.01589. After adjustment for the null distribution of $p_{\min}$, the combined test p-value, `p(CT)`, is 0.02375.

Repeating the `stctest` command, but this time using the `nperm(5000)` option, gives the following output. To ensure reproducibility, I first set the random-number generator seed:

```
. set seed 123
. stctest ps trt, nperm(5000) detail
       Treatment |   Obs      p(CT)*  [95% Conf. Interval]
-----------------+-----------------------------------------
        trt(0,1) |   274    0.025495   0.021314   0.030242
* Stochastic, from estimated permutation null distribution of p(min)
p-values from tests underlying p(CT):
  p(Cox)    p(chi2)    p(perm)     p(min)
-----------------------------------------
0.051835   0.004893   0.015894   0.015894
```

The p-value for the permutation version of the combined test is 0.02549, similar to the nonstochastic value of 0.02375. Displaying `return list` to see the stored quantities provides the following information:

```
. return list
scalars:
               r(nperm) =  5000
           r(delta_max) =  .2652538362205551
           r(tstar_max) =  2.103439807892
                 r(pct) =  .025494901019796
              r(pct_ub) =  .030241944495036
              r(pct_lb) =  .021313527894152
                r(nsig) =  127
                r(pmin) =  .0158943253638914
               r(pperm) =  .0158943253638914
               r(pchi2) =  .0048928816887735
              r(pjoint) =  .021944650127736
                 r(pgt) =  .0495280333667285
                 r(plr) =  .0517262578749101
                  r(hr) =  .7080773912586498
                r(pcox) =  .0518346333672928
                  r(t2) =  2.997728890592487
                  r(t1) =  .3148614609571788
```

Of the M = `r(nperm)` = 5000 permuted datasets, $p_{\min}$ is less than or equal to $p_{\min}^{\text{orig}}$ = `r(pmin)` = 0.01589 in r = `r(nsig)` = 127 permuted datasets, giving p_{CT} = `r(pct)` = $\{r + (1/2)\}/(M + 1)$ = (`r(nsig)` + 0.5)/(`r(nperm)` + 1) = 128.5/5001 = 0.02549.

Note that the 95% CI for `r(pct)` = 0.02549 is `r(pct_lb)` = 0.02131, `r(pct_ub)` = 0.03024. Although the CI is fairly wide, the upper bound is well below the reference level of 0.05, confirming that at conventional significance levels, there is a real effect of treatment.

A brief description of the remaining stored quantities is given in the help file. In particular, the Grambsch–Therneau test of the PH assumption, for which the p-value is returned in `r(pgt)` as 0.04953, is just significant at the 0.05 level. Non-PH may explain why the Cox test appears to have low power, despite the HR `r(hr)` = 0.708 being well below 1.0.

Rerunning the combined test using the `rp` method gives results that are similar but not identical to the `ps` method:

```
. stctest rp trt, detail
      Treatment |   Obs      p(CT)*
----------------+-------------------
       trt(0,1) |   274    0.017265
* Non-stochastic, from approximation to permutation test
p-values from tests underlying p(CT):
  p(Cox)    p(chi2)    p(perm)     p(min)
------------------------------------------
0.051835   0.003409   0.011543   0.011543
```

Figure 4 illustrates time-dependent estimates of ΔRMST according to the `ps` and `rp` methods.

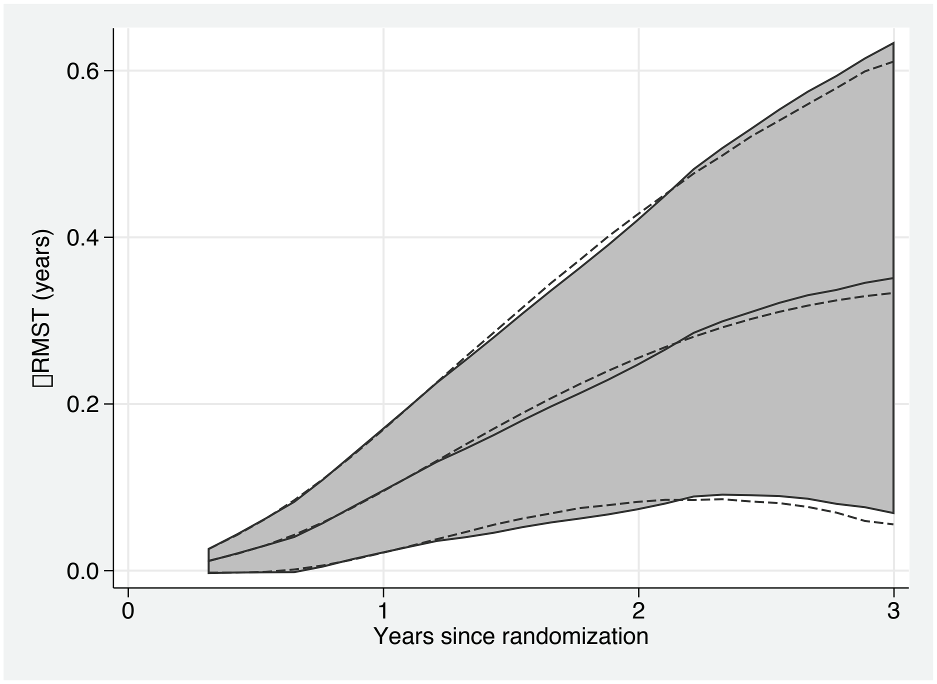


Figure 4. PATCH1 trial. Time-dependent estimates of ΔRMST, with pointwise 95% CIs, according to two methods. Solid lines and gray shaded area: `ps` method; dashed lines, `rp` method.

ΔRMST and pointwise 95% CIs were calculated at 25 equally spaced time points between the 30th and 100th centiles of the uncensored failure times (0.31 and 3.00 years, respectively). Note the considerable similarity of the two sets of estimates. At $t^* = 3.0$ years (for example), I find ΔRMST $= 0.35\ (0.07, 0.63)$ years with `ps` and $0.33\ (0.06, 0.61)$ years with `rp`. I conclude that at $t^* = 3$ years, treatment with penicillin extends the restricted mean time to recurrence of cellulitis by about four months.

5 Discussion

I have described two methods for performing the combined test that are both implemented by `stctest`, pseudovalues (`ps`), and RP models (`rp`). The two approaches give similar but not identical results. Which method would I recommend for trial design and analysis?

In essentially all relevant trials, the primary test of the null hypothesis is a Cox or log-rank test of the treatment covariate, with no other covariates in the model. Although in practice a trial may have been designed with stratification on known prognostic and structural factors, such factors are not normally accounted for in the power and sample-size calculations and arguably should not be included in the primary analysis. The issue of covariate adjustment is somewhat controversial, and this is not the place to pursue it.

Accordingly, I recommend the `ps` method when covariate adjustment is not envisioned. The reasons are as follows: First, `ps` does not require the user to choose a suitable "model" from which to estimate ΔRMST, because the pseudovalues method is based on "nonparametric" Kaplan–Meier curves. The `rp` method assumes a particular formulation of the spline model (by default, as already mentioned, `df(5)` and `dftvc(5)`) to estimate the survival curves in each treatment group and hence ΔRMST. The user can alter the spline model via the `df()` and `dftvc()` options of `stctest` if desired. However, I discourage such modifications because they are potentially data driven, which is undesirable in a trial context. I believe the default settings are sufficiently flexible to estimate ΔRMST reliably in the vast majority of trials. Second, the `ps` method is considerably faster to execute than `rp`. Such efficiency is helpful when using the `nperm()` option to check the p-value of the nonstochastic combined test.

If covariate adjustment is deemed essential, I recommend the `rp` method because adjustment for covariates with the `ps` method is done differently for the two components of the combined test, namely, the Cox and RMST models. The Cox test makes a PH assumption, whereas adjusted RMST estimation involves linear regression of the pseudovalues on the covariates and treatment. This does not seem a coherent approach. With the `rp` method, all covariates except treatment are adjusted for in a PH model, with the treatment effect permitted to have non-PH. Further elaboration of this rather complex issue is beyond the scope of this article.

A reviewer pointed out that there are rather few events (15 to be exact) during the third year and subsequent few months of follow-up. If one truncates follow-up at two years, the Cox test of the treatment effect is significant ($p = 0.0071$), with no evidence of non-PH ($p = 0.53$). The combined test gives $\widetilde{p}_{CT} = 0.0105$. This confirms that there is a real treatment effect. Most of the evidence for non-PH appears to arise from the characteristics of the event and censoring times during the third year. However, one would never present an analysis of the data truncated at two years as a primary assessment of the treatment effect, because such an analysis would certainly not have been prespecified in the trial protocol.

For practical use in trial design, Royston and Parmar (2016) suggested a simple, rough-and-ready way to power a trial under PH when the primary test of the null hypothesis is the combined test. A more precise approach to sample-size calculation requires simulation. Work on new commands implementing power and sample-size calculations is under way and will be reported in the *Stata Journal* in due course.

6 Acknowledgments

I thank Professors Kim Thomas and Hywel Williams together with the PATCH1 trial team for permission to use the trial dataset as an example, and I thank Professor Mahesh Parmar and Dr. Tim Morris for helpful comments on the manuscript. I am grateful to a reviewer whose comments prompted me to improve the presentation and the terminology.

7 References

Andersen, P. K., M. G. Hansen, and J. P. Klein. 2004. Regression analysis of restricted mean survival time based on pseudo-observations. *Lifetime Data Analysis* 10: 335–350.

Cronin, A., L. Tian, and H. Uno. 2016. strmst2 and strmst2pw: New commands to compare survival curves using the restricted mean survival time. *Stata Journal* 16: 702–716.

Lambert, P. C., and P. Royston. 2009. Further development of flexible parametric models for survival analysis. *Stata Journal* 9: 265–290.

Overgaard, M., P. K. Andersen, and E. T. Parner. 2015. Regression analysis of censored data using pseudo-observations: An update. *Stata Journal* 15: 809–821.

Parner, E. T., and P. K. Andersen. 2010. Regression analysis of censored data using pseudo-observations. *Stata Journal* 10: 408–422.

Royston, P. 2015. Estimating the treatment effect in a clinical trial using difference in restricted mean survival time. *Stata Journal* 15: 1098–1117.

Royston, P., and P. C. Lambert. 2011. *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model*. College Station, TX: Stata Press.

Royston, P., and M. K. B. Parmar. 2002. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine* 21: 2175–2197.

———. 2016. Augmenting the logrank test in the design of clinical trials in which non-proportional hazards of the treatment effect may be anticipated. *BMC Medical Research Methodology* 16: 16.

Thomas, K. S., A. M. Crook, A. J. Nunn, K. A. Foster, J. M. Mason, J. R. Chalmers, I. S. Nasr, R. J. Brindle, J. English, S. K. Meredith, N. J. Reynolds, D. de Berker, P. S. Mortimer, and H. C. Williams. 2013. Penicillin to prevent recurrent leg cellulitis. *New England Journal of Medicine* 368: 1695–1703.

Trinquart, L., J. Jacot, S. C. Conner, and R. Porcher. 2016. Comparison of treatment effects measured by the hazard ratio and by the ratio of restricted mean survival times in oncology randomized controlled trials. *Journal of Clinical Oncology* 34: 1813–1819.

About the author

Patrick Royston is a medical statistician with 40 years of experience, with a strong interest in biostatistical methods and in statistical computing and algorithms. He works largely in methodological issues in the design and analysis of clinical trials and observational studies. He is currently focusing on alternative outcome measures and tests of treatment effects in trials with a time-to-event outcome, on parametric modeling of survival data, and on novel clinical trial designs.

The Stata Journal (2017)
17, Number 2, pp. 422–441

Estimating responsiveness scores using rscore

Giovanni Cerulli
CNR-IRCrES
National Research Council of Italy
Institute for Research on Sustainable Economic Growth
Rome, Italy
giovanni.cerulli@ircres.cnr.it

Abstract. `rscore` computes unit-specific responsiveness scores using an iterated random-coefficient regression approach. The model fit by `rscore` considers a regression of a response variable y, that is, *outcome*, on a series of factors (or regressors) $\mathbf{x}$, that is, *varlist*, by assuming a different reaction (or "responsiveness") of each unit to each factor contained in $\mathbf{x}$. `rscore` allows for i) ranking units according to the obtained level of the responsiveness score; ii) detecting more influential factors in driving unit performance; and iii) studying the distribution (heterogeneity) of factors' responsiveness scores across units. Also, `rscore` offers useful graphical representation of results. We provide two illustrative applications of the model: the first is on a cross-section, and the second is on a longitudinal dataset.

Keywords: st0480, rscore, responsiveness scores, random-coefficient regression

1 Introduction

In biomedical and socioeconomic disciplines, it is commonly recognized that individual agents react heterogeneously to external stimuli. Such heterogeneity also characterizes aggregated units of analysis such as companies, regions, and entire countries.

Thus measuring unit-heterogeneous response to specific factors is relevant to providing a clearer understanding of the relationship between a stimulus (a drug administration, a policy program, etc.) and its effect on predefined target variables.

To this end, in this article, I present `rscore`, a user-written command for computing a unit-specific responsiveness score (RS) by means of an iterated random-coefficient regression (RCR) approach.

Simply put, the model fit by `rscore` considers a regression of a response variable y, that is, *outcome*, on a series of factors (or regressors) $\mathbf{x}$, that is, *varlist*, by assuming a different reaction (or "responsiveness") of each unit to each factor contained in $\mathbf{x}$.

A simple example can better illustrate the usefulness of this command for applied research. Consider the popular Stata instructional dataset `auto.dta`, and suppose we are interested in regressing the variable `price` ("price of the car") on `mpg` ("miles per gallon"). What we usually estimate is a common slope regression of this type,

$$\texttt{price}_i = \alpha + \beta \times \texttt{mpg}_i + e_i$$

where α and β are two parameters common to all units i, and e_i, an error term. However, suppose (when reasonable) each car has a different behavior in terms of the reaction of its price to its consumption. The coefficient β catches only the "average" effect of all cars, whereas knowing about the idiosyncratic behavior of each single car would be much more informative.

To this end, we may be interested in estimating a regression with a unit-specific slope, such as

$$\texttt{price}_i = \alpha + \beta_i \times \texttt{mpg}_i + e_i$$

where it is now clear that the slope refers to unit i. If, under reasonable assumptions, we could compute each β_i, we would obtain precious additional information about the relation between car price and consumption.

`rscore` aims at estimating each β_i that, in this context, we call the RS of `price` to `mpg`.[1] As will be clearer later on, it is a score, not an estimate, because we assume insufficient information in the data to perform proper inference on each β_i (without strong additional assumptions, as discussed in the next section). However, RSs may convey useful descriptive information for many phenomena in empirical research. More specifically, `rscore` may allow for i) ranking units according to the obtained level of the RS; ii) detecting more influential factors in driving unit performance; and iii) studying the distribution (heterogeneity) of factors' RSs across units. Also, `rscore` offers useful graphical representation of results.

This article is organized as follows: section 2 provides a short statistical account of the RCR, which is useful to transition into section 2.5, where I describe the model fit by `rscore`. Section 4 illustrates the syntax of `rscore`. Section 5 and section 6 provide two illustrative applications, one on a cross-section and one on a longitudinal dataset. Finally, section 7 ends the article.

2 Statistical background

In recent years, random-coefficient models have been the objective of vibrant research (Lewbel and Pendakur Forthcoming; Hoderlein, Klemelä, and Mammen 2010; Beran, Feuerverger, and Hall 1996). This literature has tried to overcome some limits of the traditional regression model by incorporating either correlated or uncorrelated random coefficients, estimated parametrically or nonparametrically. In what follows, I provide a brief description of an (exogenous) linear random-coefficient model.

To set the stage, consider a standard regression model. In such a model, parameters (that is, "regression coefficients") are singleton numbers. In a simple regression of the type

$$y_i = \alpha + \beta \times x_i + \epsilon_i \qquad i = (1, \ldots, N)$$

1. More precisely, `rscore` computes the expectation of β_i, conditional on the exogenous covariates (in this specific case, the covariate `mpg`). See section 2.5.

the regression coefficient β is a population parameter, which summarizes the effect of factor[2] x on outcome y, when it is assumed that each observation within such population shares the same β (as well as the same α).

A way to relax such an assumption is to assume each unit owns a specific regression coefficient in the population. The previous model thus becomes

$$y_i = \alpha_i + \beta_i \times x_i + \epsilon_i$$

where the generic unit i owns both a specific intercept (α_i) and a specific slope (β_i). Assume that x_i is exogenous, implying that $E(\epsilon_i|x_i) = 0$, and let's focus on β_i by letting $\alpha_i = \alpha$ for the sake of clarity. With no further information than that specified in the previous model, identifying and estimating an intercept and a slope for each individual within a sample of size N would be impossible.

However, to identify such parameters, one can impose assumptions over the probabilistic distribution of β_i by holding, for instance, that each β_i is a draw from a compounded random variable of the type

$$\beta_i = \beta + \upsilon_i$$

where υ_i is a random variable with $E(\upsilon_i) = 0$ and β is a (common) parameter. This implies that $E(\beta_i) = \beta$. Under this assumption, we obtain—by substitution—a simplified version of the previous model:

$$y_i = \alpha + \beta \times x_i + \upsilon_i x_i + \epsilon_i$$

If we assume $E(\upsilon_i|x_i) = 0$, we get

$$E(y_i|x_i) = \alpha + \beta \times x_i$$

which is a standard regression model, where the common slope β is consistently estimated by ordinary least squares (OLS) or, in the case of a heteroskedastic error, generalized least squares.

When longitudinal data are available, one can also estimate each β_i by unit-specific OLS. For this purpose, one can refer to Swamy (1970) for estimating RCRs within longitudinal data, as well as the related implementation provided by Poi (2003).

With cross-section datasets, another possible route is assuming $E(\upsilon_i|x_i) \neq 0$, which implies that β_i is a function of the covariates (or a subset of those) included in the regression model. In this case,

$$\begin{aligned} E(y_i|x_i) &= \alpha + \beta x_i + E(\upsilon_i|x_i) \times x_i + \epsilon_i \\ &= \alpha(x_i) + \beta(x_i) x_i \end{aligned} \tag{1}$$

provided again that $E(\epsilon_i|x_i) = 0$. In the previous equation, $\alpha(x_i) = \alpha + \beta x_i$ and $\beta(x_i)$ is a generic function of x that can be modeled either parametrically or nonparametrically, depending on the way one models the conditional expectation $E(\upsilon_i|x_i)$.

2. In this article, we use the term "factor" to indicate a generic covariate, although "factor" generally refers to a variable taking a discrete number of values.

Following (1), one can compute the partial effect of x on y as

$$\frac{\partial}{\partial x_i} E(y_i|x_i) = \alpha'(x_i) + \beta'(x_i)x_i + \beta(x_i) \tag{2}$$

where it is clear that each unit i owns a different partial effect depending on x_i. The mean over x of (2) is known as average partial effect (APE), which is defined as

$$\text{APE} = E_x \left\{ \frac{\partial}{\partial x_i} E(y_i|x_i) \right\} = E_x \left\{ \alpha'(x_i) + \beta'(x_i)x_i + \beta(x_i) \right\} \tag{3}$$

To estimate APE, one can use the sample equivalent of (3), that is

$$\widehat{\text{APE}} = \frac{1}{N} \sum_{i=1}^{N} \left\{ \widehat{\alpha}'(x_i) + \widehat{\beta}'(x_i)x_i + \widehat{\beta}(x_i) \right\}$$

It is clear that the heterogeneous response of each unit to a given covariate (or factor) can be interesting to evaluate per se because it brings several pieces of information about how APE takes a specific value whenever one looks at APE as the global average effect of x on y.

In this article, the partial effect of a given factor x on a unit's outcome y is called the RS of a unit's y to a unit's x, all other things being equal. Once such RSs are estimated (using a proper estimation procedure, as I will show later on), one can use them for various purposes, such as i) ranking units according to the obtained level of the RSs; ii) detecting factors that are more influential in driving unit performance; and iii) studying the distribution (heterogeneity) of factors' RSs across units.[3]

`rscore` is a command for computing these unit-specific RSs. Thus it uses an iterated RCR model whose baseline regression is similar to (1). Before presenting the syntax of `rscore`, I will illustrate the RS model's structure and assumptions.

3 The model

RSs measure the change of a given outcome y when a given factor x_j (with $j = 1, \ldots, Q$) changes, conditional on all other $(Q-1)$ factors:

$$\mathbf{x}_{-j} = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_Q)$$

Algebraically, it is the derivative of y on x_j, given $\mathbf{x}_{-j}$, when one allows for each observation to have its own RS. Importantly, we assume $\mathbf{x}_{-j}$ is a vector of all exogenous variables. RSs are obtained using an iterated RCR model, the basic econometrics of which can be found in Wooldridge (2002, 638–642; 2010, 141–145).

3. An early empirical application of the RS approach proposed in this article can be found in Cerulli (2014).

Calculating an RS follows this simple protocol:

1. Define y, the outcome (or response) variable.
2. Define a set of Q factors thought of as affecting y, and indicate the generic factor with x_j.
3. Define an RCR model linking y to the various x_j, and extract a unit-specific responsiveness effect of y to all set of factors x_j, with $j = 1, \ldots, Q$.
4. For the generic unit i and factor j, indicate such effects as b_{ij}, and collect all of them in a matrix **B**.
5. Finally, aggregate by unit (row) or by factor (column) the $E(b_{ij}|\mathbf{x}_{i,-j})$, thus getting synthetic unit and factor responsiveness measures.

Analytically, an RS is the "partial effect" of a factor x in an RCR (Wooldridge 1997; 2003; 2004). Indeed, for each $j = 1, \ldots, Q$, define an RCR model of this kind as

$$\begin{cases} y_i = a_{ij} + b_{ij} x_{ij} + e_i \\ a_{ij} = \gamma_0 + \mathbf{x}_{i,-j}\gamma + u_{ij} \\ b_{ij} = \delta_0 + \mathbf{x}_{i,-j}\delta + v_{ij} \end{cases}$$

where e_i, u_{ij}, and v_{ij} are freely correlated error terms with

$$E(e_i|\mathbf{x}_{i,-j}; x_{ij}) = E(u_{ij}|\mathbf{x}_{i,-j}; x_{ij}) = E(v_{ij}|\mathbf{x}_{i,-j}; x_{ij}) = 0$$

It is easy to see that the regression parameters, a_{ij} and b_{ij}, are both nonconstant because they depend on all the other inputs x except x_j (this is, in fact, the meaning of the vector $\mathbf{x}_{i,-j}$). Observe that δ_0 and γ_0 are, on the contrary, constant parameters. According to this model, we can define the regression line as

$$E(y_i|x_{ij}, \mathbf{x}_{i,-j}) = E(a_{ij}|\mathbf{x}_{i,-j}) + x_{ij} \times E(b_{ij}|\mathbf{x}_{i,-j})$$

Given this, we define the responsiveness effect of x_{ij} on y_i as the derivative of y_i with respect to x_{ij}; that is,

$$\frac{\partial}{\partial x_{ij}} \{E(y_i|x_{ij}, \mathbf{x}_{i,-j})\} = E(b_{ij}|\mathbf{x}_{i,-j})$$

where $E(b_{ij}|x_{ij}, \mathbf{x}_{i,-j})$ is the partial effect of x_{ij} on y_i. We can repeat the same procedure for each x_{ij} (with $j = 1, \ldots, Q$)—so it is eventually possible to define, for each unit $i = 1, \ldots, N$ and factor $j = 1, \ldots, Q$, the $N \times Q$ matrix **B** of the partial effects as follows:

$$\mathbf{B} = \begin{pmatrix} E(b_{11}|\mathbf{x}_{i,-j}) & \cdots & E(b_{1Q}|\mathbf{x}_{i,-j}) \\ \vdots & E(b_{ij}|\mathbf{x}_{i,-j}) & \vdots \\ E(b_{N1}|\mathbf{x}_{i,-j}) & \cdots & E(b_{NQ}|\mathbf{x}_{i,-j}) \end{pmatrix}$$

If all variables are standardized with zero means and unit variance, partial effects are beta coefficients and therefore independent of the unit of measurement. Thus they can be compared with each other and summed.[4]

Once the matrix $\mathbf{B}$ is known, we can define, for each unit i, the total unit responsiveness (TUR) and the mean unit responsiveness (MUR) as

$$\text{TUR}_i = \sum_{j=1}^{Q} b_{ij}$$

and

$$\text{MUR}_i = \frac{1}{Q}\sum_{j=1}^{Q} b_{ij}$$

and define, for each factor j, the total (or mean) responsiveness of y to factor j's unit change (TFR and MFR) as

$$\text{TFR}_j = \sum_{i=1}^{N} b_{ij}$$

and

$$\text{MFR}_j = \frac{1}{N}\sum_{i=1}^{N} b_{ij}$$

In a cross-section data setting, OLS provide a consistent estimation of each b_{ij} within this regression,[5]

$$\begin{aligned} &y_i = \gamma_0 + \mathbf{x}_{i,-j}\gamma + (\delta_0 + \overline{\mathbf{x}}_{-j}\delta)x_{ij} + x_{ij}(\mathbf{x}_{i,-j} - \overline{\mathbf{x}}_{-j})\delta + \eta_i \\ &\eta_i = u_{ij} + x_{ij}v_{ij} + e_i \end{aligned}$$

where $\overline{\mathbf{x}}_{-j}$ is the vector of the sample means of $\mathbf{x}_{i,-j}$. Once previous regression parameters are estimated, we can obtain an estimate of the partial effect of factor x_j on y for the generic unit i as

$$\widehat{E}(b_{ij}|\mathbf{x}_{i,-j}) = \widehat{\delta}_0 + \mathbf{x}_{i,-j}\widehat{\boldsymbol{\delta}}$$

By repeating this procedure for each unit i and factor j, we can finally obtain $\widehat{\mathbf{B}}$, that is, the estimation of matrix $\mathbf{B}$.

When a longitudinal dataset is available, the estimation of $\mathbf{B}$ can be obtained by using either random-effects or fixed-effects estimation of the following panel-data regression,

$$y_{it} = \gamma_0 + \mathbf{x}_{i,-j,t}\gamma + (\delta_0 + \overline{\mathbf{x}}_{-j,t}\delta)x_{ijt} + x_{ijt}(\mathbf{x}_{i,-j,t} - \overline{\mathbf{x}}_{-j,t})\delta + \alpha_i + \eta_{it} \qquad (4)$$

4. Because beta coefficients are measured in standard deviations, they can be compared. The meaning of a beta coefficient is straightforward; suppose that in a regression of y on x, the beta is found to be equal to 0.3. This means that one standard-deviation increase in x leads to a 0.3 standard-deviation increase in the predicted y with all the other variables in the model held constant.
5. Indeed, OLS are consistent because for each $j = 1, \ldots, Q$, we have $E(\eta_i|\mathbf{x}_{ij}) = E(u_{ij}|\mathbf{x}_{ij}) + x_{ij} \times E(v_{ij}|\mathbf{x}_{ij}) + E(e_i|\mathbf{x}_{ij}) = 0$. However, η_i is clearly heteroskedastic. Thus robust OLS provide more correct standard errors.

where the added parameter α_i represents a unit-specific effect accounting for unobserved heterogeneity. In particular, fixed-effects estimation, assuming free correlation between α_i and η_{it}, can mitigate a potential endogeneity bias due to misspecification of previous equation and measurement errors in the variables considered in the model (Wooldridge 2010, 281–315). As such, a panel dataset may allow for more reliable estimates of true RS than usual OLS.

Finally, as long as variables are standardized, (4) becomes

$$y_{it} = \gamma_0 + \mathbf{x}_{i,-j,t}\gamma + \delta_0 x_{ijt} + x_{ijt} \times \mathbf{x}_{i,-j,t}\delta + \alpha_i + \eta_{it}$$

which simplifies the formula.

4 The rscore command

4.1 Syntax

As seen above, `rscore` computes unit-specific RSs using an iterated RCR model. The model fit by `rscore` considers a regression of a response variable y, that is, *outcome*, on a series of factors $\mathbf{x}$, that is, *varlist*, by assuming a different reaction (or "responsiveness") of each unit to each factor contained in $\mathbf{x}$. The basic syntax of `rscore` is

`rscore` *outcome* [*varlist*] [*if*] [*in*] [*weight*] `, model(`*modeltype*`) rs_name(`*stub*`)`
  [`factors(`*varlist_f*`) xlist(`*varlist_c*`) graph(`#`) radar(`*numlist*`)`
  `id_string(`*varname*`) vce(`*vcetype*`) save_graph1(`*filename*`)`
  `save_graph2(`*filename*`)`]

`fweight`s, `iweight`s, and `pweight`s are allowed; see [U] **11.1.6 weight**.

4.2 Options

`model(`*modeltype*`)` specifies the model to be fit, where *modeltype* must be one of the following models: `ols` (OLS), `fe` (panel fixed effect), or `re` (panel random effect). `model()` is required.

`rs_name(`*stub*`)` specifies the beginning part of the name of the RS variables generated by `rscore`. RS variables are thus named as *stub*1, *stub*2, *stub*3, ..., *stubQ*. `rs_name()` is required.

`factors(`*varlist_f*`)` specifies that factor variables have to be included among the regressors.

`xlist(`*varlist_c*`)` specifies that control variables (which are not factors) have to be included among the regressors.

`graph(#)` provides a combined graph of the densities of the RSs. The number # defines the width of the graph's x axis. The user can set a proper # for providing a good rendering of the graph.

`radar(`*numlist*`)` provides a radar plot of the RSs for the units specified in *numlist*. To run this option, the user must specify the `id_string()` option. This option uses the user-written `radar` command provided by Mander (2007).

`id_string(`*varname*`)` requests that a string variable as identifier of each observation be specified. This is required if the user wishes to provide a radar plot of the RSs.

`vce(`*vcetype*`)` allows the user to choose *vcetype* as either `robust` or `cluster` *clustvar*.

`save_graph1(`*filename*`)` saves the graph generated by the `graph()` option in the user-specified *filename*.

`save_graph2(`*filename*`)` saves the graph generated by the `radar()` option in the user-specified *filename*.

4.3 Stored results

Finally, for each factor regression, `rscore` returns goodness-of-fit statistics—that is, R-squared—stored in scalars `e(R1)`, `e(R2)`, `e(R3)`, ..., `e(RQ)`, as well as the average R-squared, which is the overall goodness of fit of the model, stored in the scalar `e(R)`.

5 Application 1

This section presents an illustrative application of `rscore` within a cross-section data structure. It is intended to allow users to become familiar with the use of this command.

To this end, we consider the usual `auto.dta`, with a full specification of the `rscore` syntax. In this application, we are interested in identifying the main factors driving the price of a car and identifying the distribution of such an effect over observations for each declared factor. We focus on price responsiveness to five covariates (that is, `mpg`, `trunk`, `weight`, `length`, and `displacement`) by controlling for two factor variables (that is, `foreign` and `rep78`) and two controls (that is, `gear_ratio` and `headroom`). The application of `rscore` results in code like this:

```
. sysuse auto
(1978 Automobile Data)
. rscore price mpg trunk weight length displacement, rs_name(RS)
> model(ols) factors(foreign rep78) xlist(gear_ratio headroom)
> graph(100) id_string(make) radar(4 9 13 40) save_graph1(mydistr)
> save_graph2(myradar)

**********************************************************************
*** DESCRIPTIVE STATISTICS FOR SINGLE FACTOR RESPONSIVENESS SCORES ***
**********************************************************************
```

```
             Responsiveness scores for variable mpg_std
-------------------------------------------------------------
      Percentiles      Smallest
 1%    -.9230999      -.9230999
 5%    -.3402585      -.7910841
10%    -.2775861      -.7330251       Obs                  69
25%     -.090533      -.3402585       Sum of Wgt.          69

50%     .0403205                      Mean           .0112316
                        Largest       Std. Dev.      .2559057
75%     .1841674       .3397723
90%     .2829912       .3649702       Variance       .0654877
95%     .3397723       .4167671       Skewness      -1.354045
99%      .477404        .477404       Kurtosis       5.911784
            Responsiveness scores for variable trunk_std
-------------------------------------------------------------
      Percentiles      Smallest
 1%    -.5023578      -.5023578
 5%    -.2346051      -.3120776
10%    -.1627576      -.2949739       Obs                  69
25%     .0015847      -.2346051       Sum of Wgt.          69

50%     .1770424                      Mean           .2170236
                        Largest       Std. Dev.      .3706054
75%     .3745154        .718573
90%     .6115505        1.25186       Variance       .1373483
95%      .718573       1.303261       Skewness       1.583251
99%      1.77037        1.77037       Kurtosis       7.317773
            Responsiveness scores for variable weight_std
-------------------------------------------------------------
      Percentiles      Smallest
 1%     .4967597       .4967597
 5%     .5914793       .5649452
10%     .6201076       .5688947       Obs                  69
25%     .7530873       .5914793       Sum of Wgt.          69

50%     .9084411                      Mean           .9705862
                        Largest       Std. Dev.      .3131494
75%     1.128973       1.620982
90%     1.530207       1.666719       Variance       .0980625
95%     1.620982       1.668086       Skewness       .8406102
99%     1.763967       1.763967       Kurtosis       2.886559
            Responsiveness scores for variable length_std
-------------------------------------------------------------
      Percentiles      Smallest
 1%    -.8865399      -.8865399
 5%    -.8249316      -.8677544
10%    -.6593894      -.8445593       Obs                  69
25%    -.5387048      -.8249316       Sum of Wgt.          69

50%    -.4376124                      Mean          -.4303998
                        Largest       Std. Dev.      .1895008
75%    -.3094746       -.143986
90%    -.2292118      -.0476573       Variance       .0359106
95%     -.143986       .0030893       Skewness      -.0796322
99%     .0920853       .0920853       Kurtosis       3.649125
```

```
               Responsiveness scores for variable
                        displacement_std
-------------------------------------------------------------
      Percentiles      Smallest
 1%    -.8768004      -.8768004
 5%    -.5376742      -.7265469
10%    -.4575502      -.6088864       Obs                  69
25%    -.2181616      -.5376742       Sum of Wgt.          69

50%    -.0629257                      Mean           .0022286
                        Largest       Std. Dev.      .4293259
75%     .1630541        .698191
90%     .5437063       1.304059       Variance       .1843207
95%      .698191       1.332656       Skewness       1.316481
99%     1.496196       1.496196       Kurtosis       5.795581
-------------------------------------------------------------

*************************************************************
*************** RSCORE GOODNESS-OF-FIT **********************
*************************************************************
-------------------------------------------------------------
The R-squared for mpg_std is:
.63707556
-------------------------------------------------------------
The R-squared for trunk_std is:
.65798794
-------------------------------------------------------------
The R-squared for weight_std is:
.65783691
-------------------------------------------------------------
The R-squared for length_std is:
.62031188
-------------------------------------------------------------
The R-squared for displacement_std is:
.73278452
-------------------------------------------------------------
The mean R-squared is:
.66119936
-------------------------------------------------------------
```

We call `rscore` using an OLS estimation through the `model(ols)` option. This option is appropriate with a cross-section dataset such as `auto.dta`. The default output of `rscore` is a series of summary statistics tables for each factor RS considered and a table reporting the single-factor regression R-squared and the model mean (or overall) R-squared. In this specific case, we obtain five summary statistics and six R-squared. From summary statistics, we see that the variable `weight` sets out the highest average RS (with a value of 0.97). Notice that RSs are beta coefficients because `rscore` z-standardizes each variable in advance. Therefore, a mean RS for `weight` of 0.97 means that—on average—one standard-deviation change in cars' weight yields a one standard-deviation increase in cars' price. We do not have a significance test for this value because it is held only as a score. From the goodness-of-fit output, we see that the best fit is reached by variable `displacement` with an R-squared of about 0.73, which is substantial. The average R-squared of the model is 0.66, meaning that—on average—our factors' specification explains around 66% of the total variance of cars' prices. Because it is a rather high R-squared, we can accept our specification as a good one.

Because we set the `rs_name(RS)` option, the command generates five new variables called `RS1`, `RS2`, `RS3`, `RS4`, and `RS5`, which contain the RSs for the five factors. The variables' labels suggest which factor each generated variable refers to.

The `factors(foreign rep78)` and `xlist(gear_ratio headroom)` options set the factor variables and a set of additional (continuous) variables to control for. They are specified only as controls; that is, they provide `rscore` with a correct specification for `price`.

To generate graphical results, `rscore` offers two options that can also be combined: `graph(#)` and `radar(4 9 13 40)`. Note that the latter option needs to be specified jointly with the string identifier of the observations, which in this case is the `make` variable. Hence, the `radar(4 9 13 40)` option needs to be combined with the `id_string(make)` option.

If the `graph(#)` option is specified, Stata returns the graph in figure 1. This is the joint plot of the distribution of all factors' RSs. This graph allows one to visually detect two aspects:

- Factor importance: This is (positively) higher as soon as the factor's RS distribution lies on the right side of the figure.
- Factor response heterogeneity: This is higher whenever the factor's RS distribution presents long tails.

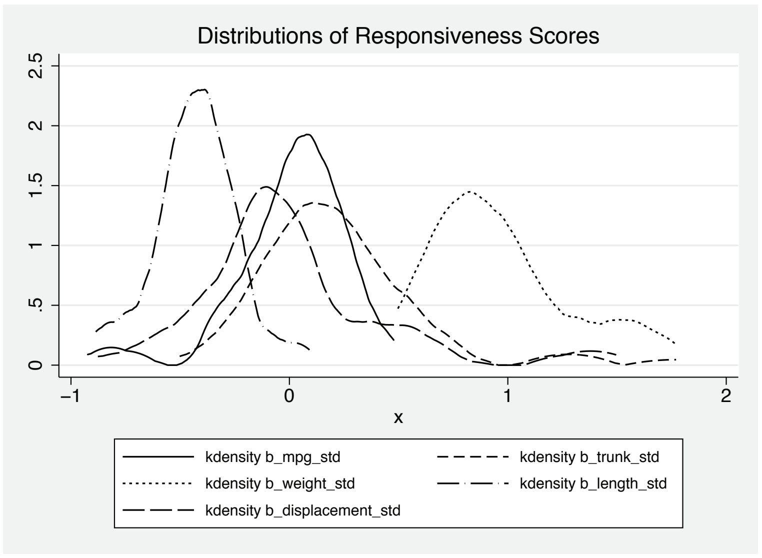


Figure 1. Joint plot of the distribution of factors' RSs

Looking at figure 1, we see that the factor whose distribution stands out in terms of a positive effect on price is a car's weight. This distribution is located at the extreme right of the graph and presents a long right tail. However, the factors with larger RS dispersions are `trunk` and `displacement`, while `length` shows a very concentrated distribution, albeit with a negative impact on price on average. It means that cars' price response to their length are weak, negative, and very similar among all car models.

If the `radar(4 9 13 40)` and `id_string(make)` options are (jointly) specified, we obtain the radar graph of figure 2.

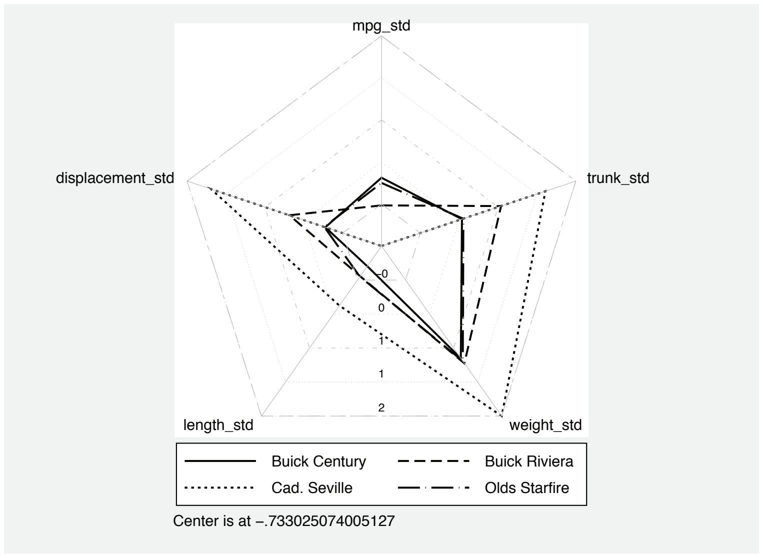


Figure 2. Units' RS radar graph

This graph is useful for comparing—factor by factor—different unit RSs. In this example, we aim to compare units 4, 9, 13, and 40 in our dataset, which correspond to specific models of cars. By looking at figure 2, we immediately see that the car model "Cadillac Seville" presents a higher RS for each factor, except for `mpg`. The car models "Buick Century" and "Old Starfire" show a similar RS on each factor, while "Buick Riviera" has a larger impact on price through `trunk` and `displacement`. Thus the radar graph is useful to have a quick snapshot of units' comparative results on single factors' RSs. This is a handy option, which emphasizes the advantage of using an RS approach over traditional regression methods.

6 Application 2

In this second application, I use `rscore` to identify the main drivers of countries' gross domestic product (GDP). I address three research questions:

- Factor importance rank: Among countries' GDP components, what are those whose growth change produces a larger or smaller response in terms of GDP growth?
- Factor response heterogeneity: Is a country's GDP growth response more or less heterogeneously or homogeneously distributed among its driving factors?
- Unit responsiveness rank: Which units have larger or smaller RSs for each given factor?

For a dataset, we consider the World Bank's "Economy & Growth" indicators database, which is made of 283 macroeconomic indicators for 250 countries collected from 1960–2014. We consider 13,695 observations, thus obtaining a huge longitudinal dataset.

As drivers, we consider the main components of GDP formation, plus the government surplus or deficit, that is,

- Cash surplus or deficit (percent of GDP)
- General government final consumption expenditure (annual percent growth)
- Household final consumption expenditure (annual percent growth)
- Gross fixed capital formation (annual percent growth)
- Exports of goods and services (annual percent growth)
- Imports of goods and services (annual percent growth)

Within this dataset, we consider the variable `ny_gdp_pcap_kd_zg` representing the "real GDP rate of growth". We plot the time pattern of this variable from 1990–2013 for the five largest European countries, namely, Great Britain (`GBR`), Germany (`DEU`), Italy (`ITA`), France (`FRA`), and Spain (`ESP`). Figure 3 shows the time pattern, and the Stata code to obtain this figure is

```
. clear all

. set maxvar 30000

. use "data_worldbank_economy&growth.dta", clear

. global time year>=1990

. twoway
> line Y year  if countrycode == "GBR" & $time, sort ||
> line Y year  if countrycode == "FRA" & $time, sort ||
> line Y year  if countrycode == "ITA" & $time, sort ||
> line Y year  if countrycode == "ESP" & $time, sort ||
> line Y year  if countrycode == "DEU" & $time, sort
> xlabel(1990(2)2015, gmax angle(horizontal))
> legend(label(1 "GBR") label(2 "FRA") label(3 "ITA")
> label(4 "ESP") label(5 "DEU")) title("GDP annual growth")
```

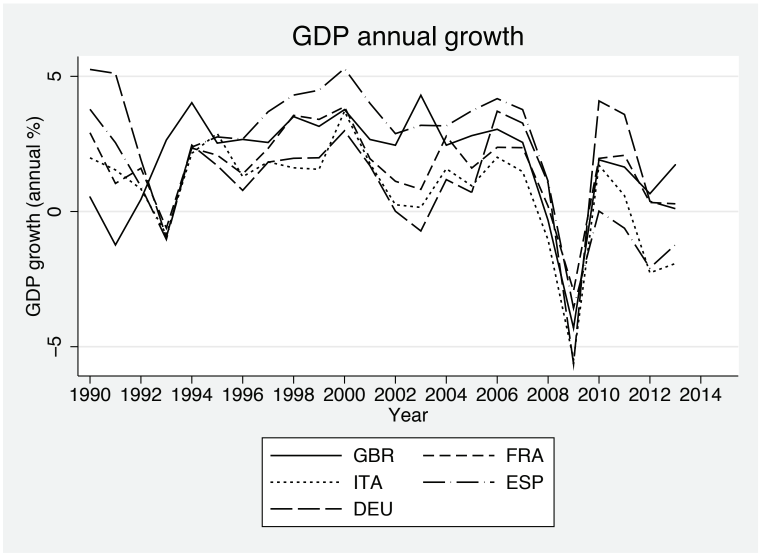


Figure 3. GDP annual growth rate: 1990–2012

We now apply `rscore` by running a fixed-effects model (because a panel-data structure is now available):

```
. * Estimate RS for the "GDP annual growth" (Y)
. global xvars B  G  C  I  E  M

. * Model with fixed effects
. capture drop w

. generate w=10

. capture drop id_country

. encode countryname, generate(id_country)

. tsset id_country year
       panel variable:  id_country (strongly balanced)
        time variable:  year, 1960 to 2014
                delta:  1 unit
```

```
. capture drop year2
. tostring year, gen(year2)
year2 generated as str4
. capture drop countryr
. generate countryr=countryname+year2
. rscore Y $xvars [pweight=w], model(fe) rs_name(_bx) graph(3)
> radar(4508 4233 5938 12978 11383 13033) id_string(countryr)
  (output omitted)
```

For the sake of brevity, we omit the RS summary tables and consider only the graphical outcomes. Indeed, as in the previous application, `rscore` generates the following combined plot of the distributions of RS for each variable considered. This graph is illustrated in figure 4.

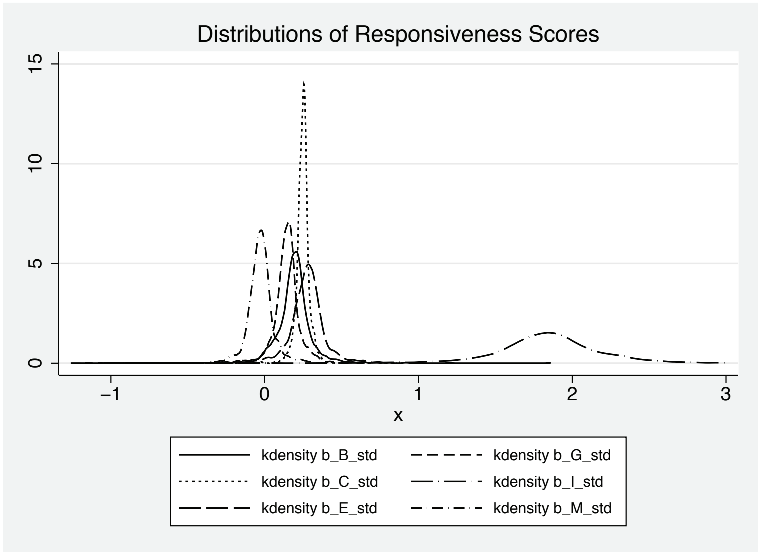


Figure 4. Distribution of RSs: 1990–2012

Quite surprisingly, figure 4 shows that the most relevant factor driving larger GDP rate of growth (by country and by time) is represented by the private fixed investment. Larger annual growth in this variable is associated with larger GDP growth, other things being equal. More precisely, one standard-deviation increase in private investment turns out to be associated—on average—with about two standard-deviations' positive increase in the GDP rate of growth.

However, private investment is also the factor showing the largest RS variance around its mean, which can be interpreted as a measure of risk whenever one wants to draw policy advice from this result. The second most relevant driver of GDP growth is export, which shows a much more concentrated RS distribution than investment. As expected, imports rank in the last position, with a negative average RS. Table 1 shows factor importance results for all the factors considered in this example.

Table 1. Factor importance results

Factor	Observations	Mean	Standard deviation	Min	Max
Deficit	1,877	0.1973794	0.3941552	−4.902884	15.50016
Public spending	1,877	0.1369482	0.2697644	−10.85175	0.6500276
Consumption	1,877	0.2479453	0.0429293	−0.1079709	0.4484287
Investment	1,877	1.847772	0.3886027	−1.261205	5.092096
Export	1,877	0.2691284	0.3522494	−14.00593	0.8352496
Import	1,877	−0.0319164	0.2415005	−9.790175	0.3935999

Figure 5 shows the time pattern of country GDP growth RS to the growth in fixed investments. As can be seen, from 2000–2010, Spain was the country with the highest GDP growth responsiveness to fixed investment, while Great Britain was the best performer during the '90s. The last year (that is, 2012) shows that Germany and Italy are particularly responsive.

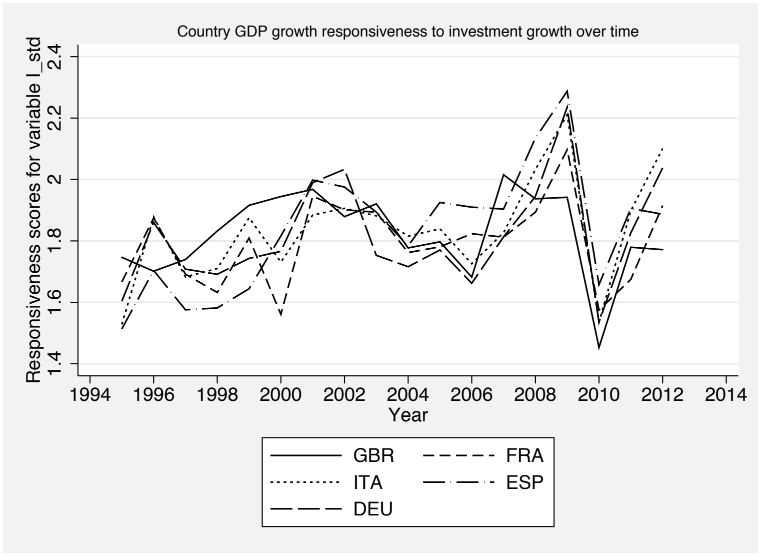


Figure 5. Time pattern of GDP growth RS to the growth in fixed investments: 1990–2012

An interesting piece of information we can obtain from matrix **B** is a ranking of observations according to their RS. For example, we could be interested in knowing the observation in our dataset with the highest investment RS. To obtain such information, we simply sort observations over variable `_bx4` and list them as follows:

```
. sort _bx4

. list countryname year _bx4 if _bx4>=3 & _bx4!=.

      +------------------------------------------+
      |          countryname   year       _bx4   |
      |------------------------------------------|
1865. |                 Mali   2005   3.036985   |
1866. |              Belarus   1992   3.045852   |
1867. |         Congo, Rep.   2008   3.075329    |
1868. |              Nigeria   2012   3.094015   |
1869. |    Macao SAR, China   2009   3.143898    |
      |------------------------------------------|
1870. |           Seychelles   2008   3.196258   |
1871. |            Argentina   2002   3.227396   |
1872. |            Indonesia   1999   3.309149   |
1873. |  Trinidad and Tobago   2007     3.3421   |
1874. |   Iran, Islamic Rep.   1994   3.403374   |
      |------------------------------------------|
1875. |             Bulgaria   1991   3.416531   |
1876. |             Bulgaria   1990   4.174945   |
1877. |              Nigeria   2004   5.092096   |
      +------------------------------------------+
```

We see that Nigeria in 2004 exhibits the highest GDP growth response to investment growth with an RS equal to 5.09; this means that one standard-deviation change in the rate of growth of fixed investment is associated with about a five standard-deviation increase in Nigerian GDP growth, which is rather high if one considers that the average RS over countries and years in this case is around two.

Finally, as a global responsiveness index, we calculate the MUR and show the first and last 10 observations according to this index:

```
. generate MUR = (_bx1 + _bx2 + _bx3 + _bx4 + _bx5 + _bx6)/6
(11,818 missing values generated)

. sort MUR

. list countryname year MUR in 1/10

     +--------------------------------------+
     |      countryname   year         MUR  |
     |--------------------------------------|
  1. |     Sierra Leone   2000   -3.208911  |
  2. |          Nigeria   2004   -.0919238  |
  3. |     Sierra Leone   2010   -.0733859  |
  4. |           Rwanda   1991   -.0355732  |
  5. | Congo, Dem. Rep.   2003   -.0349704  |
     |--------------------------------------|
  6. |          Nigeria   2007   -.0310322  |
  7. |    Venezuela, RB   1997    .0403034  |
  8. |     Sierra Leone   2011    .0478267  |
  9. |       Azerbaijan   1996    .0880654  |
 10. |       Madagascar   2003    .0898289  |
     +--------------------------------------+
. gsort - MUR
. list countryname year MUR in 1/10

     +----------------------------------------+
     |        countryname   year         MUR |
     |----------------------------------------|
  1. |           Bulgaria   1990   .8761727 |
  2. |           Bulgaria   1991   .8273186 |
  3. |        Congo, Rep.   2008    .779387 |
  4. |   Macao SAR, China   2009   .7609017 |
  5. |               Mali   2006   .7523293 |
     |----------------------------------------|
  6. |          Argentina   2002   .7505001 |
  7. |            Belarus   1992   .7342721 |
  8. | Iran, Islamic Rep.   1994   .7301948 |
  9. |   Congo, Dem. Rep.   1994   .7280833 |
 10. |         Seychelles   2003   .7250628 |
     +----------------------------------------+
```

Bulgaria in 1990 and 1991 displays the highest average global response to all the drivers considered in this application, while the worst-performing country was Sierra Leone in 2000.

Lastly, we consider the radar graph provided by `rscore` in this second application. The result for 2012 is illustrated in figure 6.

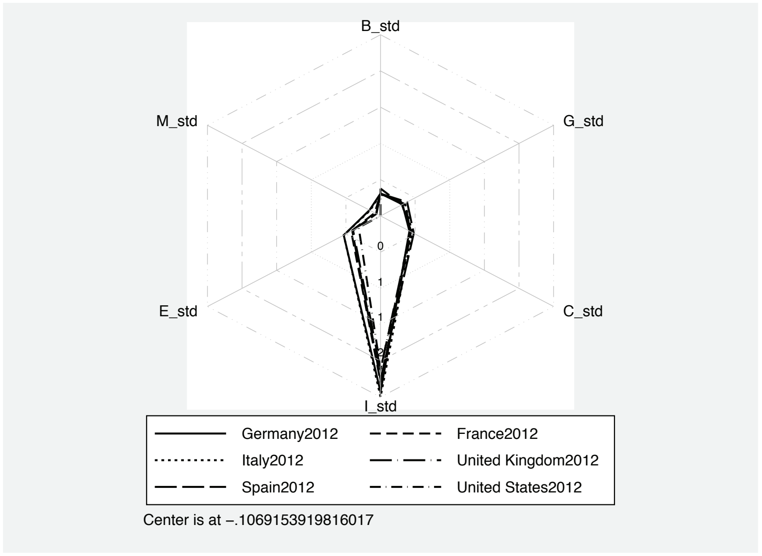


Figure 6. Radar graph for RS country comparison—year 2012

It is evident to see that no remarkable differences arise among European countries (and United States) in 2012. As expected, private fixed investment stands out as the factor with leading responsiveness.

7 Conclusion

The `rscore` command presented in this article can be a useful tool to detect both factor importance and factor heterogeneous response in a regression analysis measuring the impact of a factor x_j on an outcome y. `rscore` also allows the analyst to exploit fixed-effects estimation for a regression of y on x_j, thus mitigating potential factor endogeneity within each factor regression. This command allows one to suitably rank both factors and observations according to their RSs, providing the analyst with more detailed idiosyncratic information of the response of an outcome y on factor x_j whenever the analysis of such a relation appears to be meaningful.

8 References

Beran, R., A. Feuerverger, and P. Hall. 1996. On nonparametric estimation of intercept and slope distributions in random coefficient regression. *Annals of Statistics* 24: 2569–2592.

Cerulli, G. 2014. The impact of technological capabilities on invention: An investigation based on country responsiveness scores. *World Development* 59: 147–165.

Hoderlein, S., J. Klemelä, and E. Mammen. 2010. Analyzing the random coefficient model nonparametrically. *Econometric Theory* 26: 804–837.

Lewbel, A., and K. Pendakur. Forthcoming. Unobserved preference heterogeneity in demand using generalized random coefficients. *Journal of Political Economy*.

Mander, A. 2007. radar: Stata module to draw radar (spider) plots. Statistical Software Components S456829, Department of Economics, Boston College.
https://ideas.repec.org/c/boc/bocode/s456829.html.

Poi, B. P. 2003. From the help desk: Swamy's random-coefficients model. *Stata Journal* 3: 302–308.

Swamy, P. A. V. B. 1970. Efficient inference in a random coefficient regression model. *Econometrica* 38: 311–323.

Wooldridge, J. M. 1997. On two stage least squares estimation of the average treatment effect in a random coefficient model. *Economics Letters* 56: 129–133.

———. 2002. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT Press.

———. 2003. Further results on instrumental variables estimation of average treatment effects in the correlated random coefficient model. *Economics Letters* 79: 185–191.

———. 2004. 03.2.1. Fixed effects estimation of the population-averaged slopes in a panel data random coefficient model—Solution. *Econometric Theory* 20: 428–429.

———. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

About the author

Giovanni Cerulli is a researcher at CNR-IRCrES, National Research Council of Italy, Institute for Research on Sustainable Economic Growth. He received a degree in statistics and a PhD in economic sciences from Sapienza University of Rome, and is editor-in-chief of the *International Journal of Computational Economics and Econometrics*. His main research interest is applied microeconometrics, with a focus on counterfactual treatment-effects models for program evaluation. He is the author of the book *Econometric Evaluation of Socio-Economic Programs: Theory and Applications* (Springer, 2015). He has published articles in high-quality, refereed economics journals.

The Stata Journal (2017)
17, Number 2, pp. 442–461

Multilevel multiprocess modeling with gsem

Tamás Bartus
Corvinus University of Budapest
Budapest, Hungary
tamas.bartus@uni-corvinus.hu

Abstract. Multilevel multiprocess models are simultaneous equation systems that include multilevel hazard equations with correlated random effects. Demographers routinely use these models to adjust estimates for endogeneity and sample selection. In this article, I demonstrate how multilevel multiprocess models can be fit with the `gsem` command. I distinguish between two classes of multilevel multiprocess models: nonrecursive systems of hazard equations without observed endogenous variables and recursive systems that include a hazard equation with observed endogenous qualitative variables. I illustrate the estimation of both classes of models using sample datasets shipped with the statistical software aML. I pay special attention to identifying structural coefficients in nonrecursive simultaneous systems.

Keywords: st0481, survival analysis, multilevel multiprocess models, multilevel analysis, simultaneous equations, endogeneity, gsem

1 Introduction

Multilevel multiprocess models were developed as systems of proportional hazard models with correlated individual-level random effects. These models adjust estimates of the parameters of hazard equations for two forms of simultaneity (Lillard 1993; Lillard and Waite 1993). Suppose a researcher examines the impact of children on marital stability. Estimates of ordinary survival models of the hazard of divorce are likely to be biased; the first form of simultaneity is the endogeneity of the presence of children, because it is the outcome of a related process of timing of births. Furthermore, the conception hazard might depend on the latent dissolution hazard; if couples expect that their marriage will be short lived, they may decide to postpone the first (or higher-order) births. The second form of simultaneity arises because the latent hazard of marriage dissolution is an unobservable (endogenous) variable in the conception hazard equation.

The multilevel multiequation modeling framework has advantages. First, some of the explanatory variables in hazard models are endogenous, and estimation of the hazard model of substantive interest jointly with probit models explaining the endogenous variables eliminates the endogeneity bias (Lillard, Brien, and Waite 1995; Impicciatore and Billari 2012). Second, the multilevel multiprocess modeling framework easily deals with selection bias. Consider the estimation of the effect of education on second-birth hazards (Kravdal 2001). Finding a positive effect of higher education can be explained in terms of a selection effect. Because educated women postpone first births, unmeasured factors that also affect the timing of births will be correlated with education in the sample of mothers, even when those unmeasured factors are independent of education in the population of childless women. The selection effect is appropriately controlled if the hazard models explaining first, second, and higher-order births are jointly fit.

In this article, I show how multilevel multiprocess models can be fit with the `gsem` command, which is a natural choice for two reasons: it allows one to estimate multilevel equations with correlated latent variables, and it supports survival equations in Stata 14. My endorsement of the `gsem` command contrasts an earlier suggestion of fitting systems of survival models with the user-written `cmp` command (Roodman 2011; Bartus and Roodman 2014). The advantage of the `cmp` command is that the correlation of residuals can be modeled without including random effects. This strategy has an additional computational advantage because systems including two equations can be estimated without numerically approximating two-dimensional integrals. However, the `cmp` command forces researchers to impose lognormal duration dependence on the data, an unrealistic assumption in several applications. Additionally, the computational advantage of the `cmp` command might have been overstated because numerical integration procedures seem to be substantially faster in Stata 14 than in older versions.

I begin by identifying two classes of multilevel multiprocess models: nonrecursive systems of hazard equations without observed endogenous variables and recursive systems that include hazard equations with observed endogenous qualitative variables. Afterward, I detail how both classes of models can be fit using the `gsem` command. The examples use sample datasets shipped with the statistical software aML, which was explicitly developed for multilevel multiprocess modeling (Lillard and Panis 2003). I pay special attention to identifying structural parameters in nonrecursive systems of hazard equations, an issue often neglected in empirical applications.

2 Multilevel multiprocess hazard models

2.1 Motivation

Multilevel multiprocess modeling addresses the problem that explanatory variables are often endogenous because of selection mechanisms. Consider the classic example of estimating the impact of children on marital stability. Estimates from a separate hazard model of divorce suffer from two forms of simultaneity biases (Lillard 1993; Lillard and Waite 1993). First, the presence of children is endogenous because it is the outcome of a process of timing of births. Second, the latent birth hazard might depend on the latent dissolution hazard as well. Similar biases arise if the researcher is also interested in examining the effect of marriage on childbearing. Marriage is the outcome of the partnership formation process, which may depend on the latent propensity of becoming a parent.

The aforementioned simultaneity problems can easily be studied within the framework of simultaneous equations with qualitative variables (Heckman 1978). Let y^*_{1t} and

y^*_{2t} denote the endogenous latent hazards under study; for instance, the former might be the hazard of conception, and the latter might denote the hazard of marital dissolution. Subscript t expresses the possible time dependence of the hazards. y_{1t} and y_{2t} are observed realizations of the latent variables. The dependence of each latent variable on the other, as well as on other (possibly time-varying) explanatory variables $\mathbf{x}_{1t}$ and $\mathbf{x}_{2t}$, is described with the structural equations

$$\begin{aligned} y^*_{1t} &= \alpha_1 y_{2t} + \lambda_1 y^*_{2t} + \boldsymbol{\beta}'_1 \mathbf{x}_{1t} + \varepsilon_{1t} \\ y^*_{2t} &= \alpha_2 y_{1t} + \lambda_2 y^*_{1t} + \boldsymbol{\beta}'_2 \mathbf{x}_{2t} + \varepsilon_{2t} \end{aligned} \tag{1}$$

The two forms of simultaneity are related to the presence of latent variables and observed realizations on the right-hand side of the equations. First, the error terms are correlated with the exogenous explanatory variables because of the presence of an unobserved hazard on the right-hand side and the dependence of that hazard on the same exogenous variables. Second, the expected value of the residual is not constant across the categories of the observed realizations (Lee 1979).

Joint estimation of the system is viewed as a method for eliminating both sources of endogeneity bias. I will discuss the method separately for two classes of the model. It is well known that the parameters of the model defined by (2) are not identified without further restrictions. Using classic results on logical consistency and identification (Maddala 1983), we see that the model exists only if $\lambda_1\alpha_2 = \lambda_2\alpha_1 = 0$ and $\alpha_1\alpha_2 = 0$. This condition implies that there are two forms of estimable systems. The first form is nonrecursive systems without observed endogenous variables ($\alpha_1 = \alpha_2 = 0$):

$$\begin{aligned} y^*_{1t} &= \lambda_1 y^*_{2t} + \boldsymbol{\beta}'_1 \mathbf{x}_{1t} + \varepsilon_{1t} \\ y^*_{2t} &= \lambda_2 y^*_{1t} + \boldsymbol{\beta}'_2 \mathbf{x}_{2t} + \varepsilon_{2t} \end{aligned}$$

The second form is recursive systems with observed endogenous variables ($\lambda_1 = \lambda_2 = 0$ and $\alpha_1 = 0$):

$$\begin{aligned} y^*_{1t} &= \boldsymbol{\beta}'_1 \mathbf{x}_{1t} + \varepsilon_{1t} \\ y^*_{2t} &= \alpha_2 y_{1t} + \boldsymbol{\beta}'_2 \mathbf{x}_{2t} + \varepsilon_{2t} \end{aligned} \tag{2}$$

I will now discuss these models briefly.

2.2 Nonrecursive systems without observed endogenous variables

In these systems, endogeneity bias emerges because unobserved endogenous hazards appear on the right-hand sides of both equations. The dependence of hazards on other hazards disappears in the reduced-form system. However, the reduced-form parameters are not equal to the structural parameters of interest. In this section, I focus on identifying these parameters via excluded instruments. To emphasize the presence of excluded instruments, we rewrite the structural model as

$$
\begin{aligned}
y_{1t}^* &= \lambda_1 y_{2t}^* + \boldsymbol{\beta}_1' \mathbf{x}_t + \gamma_1 z_{1t} + \varepsilon_{1t} \\
y_{2t}^* &= \lambda_2 y_{1t}^* + \boldsymbol{\beta}_2' \mathbf{x}_t + \gamma_2 z_{2t} + \varepsilon_{2t}
\end{aligned}
$$

where $\mathbf{x}$ is a vector of exogenous variables common to both equations and the z's are excluded instruments. The system of reduced-form equations is

$$
\begin{aligned}
y_{1t}^* &= \boldsymbol{\pi}_{10}' \mathbf{x_t} + \pi_{11} z_{1t} + \pi_{12} z_{2t} + v_{1t} \\
y_{2t}^* &= \boldsymbol{\pi}_{20}' \mathbf{x_t} + \pi_{21} z_{1t} + \pi_{22} z_{2t} + v_{2t}
\end{aligned}
$$

where

$$
\begin{aligned}
\boldsymbol{\pi}_{j0} &= (1-\lambda_1\lambda_2)^{-1}(\boldsymbol{\beta}_j' + \lambda_j \boldsymbol{\beta}_k') \\
\pi_{jj} &= (1-\lambda_1\lambda_2)^{-1}\gamma_j \\
\pi_{jk} &= (1-\lambda_1\lambda_2)^{-1}\lambda_j\gamma_k \\
v_j &= \varepsilon_j + \lambda_j \varepsilon_k \qquad (3)
\end{aligned}
$$

where $j = \{1, 2\}$ indexes the equations and $k = 3 - j$. Estimation must account for the residuals in the reduced-form equations being generally correlated, even when the disturbances in the structural equations are independent of each other. If the latter error terms are normally distributed, the correlation of the residuals can easily be modeled using the multivariate normal distribution. In proportional hazard models, however, the error terms are exponentially distributed. Hence, the correlation of the underlying residuals should be modeled with the help of jointly normally distributed random intercepts (Lillard 1993). The resulting multilevel multiprocess model can be stated as follows:

$$
\begin{aligned}
y_{1t}^* &= \boldsymbol{\pi}_{10}' \mathbf{x_t} + \pi_{11} z_{1t} + \pi_{12} z_{2t} + u_1 + \eta_{1t} \\
y_{2t}^* &= \boldsymbol{\pi}_{20}' \mathbf{x_t} + \pi_{21} z_{1t} + \pi_{22} z_{2t} + u_2 + \eta_{2t} \\
\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}\right) \qquad (4)
\end{aligned}
$$

In the presence of excluded instruments, the structural coefficients can be recovered as follows. First, notice from (3) that the selection coefficient λ_j can easily be estimated as follows:

$$
\lambda_j = \pi_{jk}/\pi_{kk} \qquad (5)
$$

Second, use the estimated selection parameters to solve the system:

$$
\begin{aligned}
\boldsymbol{\pi}_{10} &= (1-\lambda_1\lambda_2)^{-1}(\boldsymbol{\beta}_1' + \lambda_1 \boldsymbol{\beta}_2') \\
\boldsymbol{\pi}_{20} &= (1-\lambda_1\lambda_2)^{-1}(\boldsymbol{\beta}_2' + \lambda_2 \boldsymbol{\beta}_1')
\end{aligned}
$$

The solution is a simple nonlinear combination of reduced-form coefficients:

$$
\boldsymbol{\beta}_j = \boldsymbol{\pi}_{j0} - \lambda_k \boldsymbol{\pi}_{k0} = \boldsymbol{\pi}_{j0} - (\pi_{jk}/\pi_{kk})\boldsymbol{\pi}_{k0} \qquad (6)
$$

Both the nonlinear combination and its standard error can be calculated using the `nlcom` command.

2.3 Recursive systems with observed endogenous variables

In recursive systems, all coefficients are structural. Endogeneity arises because the expected value of ε_1 differs in groups $y_{1t} = 1$ and $y_{1t} = 0$. The problem is the same as the problem of sample selection. The endogeneity bias is eliminated if the residuals are allowed to be correlated and the seemingly unrelated system is jointly estimated. In the case of two equations with normally distributed residuals, this boils down to the estimation of a bivariate probit model. However, the second equation is a proportional hazard model, and the joint estimation requires the inclusion of random intercepts. The multilevel multiprocess model is

$$\begin{aligned} y_{1t}^* &= \boldsymbol{\beta}_1' \mathbf{x}_{1t} + u_1 + \eta_{1t} \\ y_{2t}^* &= \alpha_2 y_{1t} + \boldsymbol{\beta}_2' \mathbf{x}_{2t} + u_2 + \eta_{2t} \\ \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}\right) \end{aligned} \tag{7}$$

To identify the correlation of the random effects, one should include in the first-stage equation at least one variable not included in the second-stage hazard equation.

In empirical applications, the latent variable y_{1t}^* is often a time-constant latent propensity to experience an event. The classic example is the propensity to form a cohabiting union before marriage that in turn will affect the (time-varying) hazard of marital dissolution (Lillard, Brien, and Waite 1995).

3 Fitting multilevel multiprocess models with gsem

The official Stata `gsem` command can fit multiprocess hazard models because it supports multiequation survival models with correlated latent variables. The description of the syntax is restricted to components of the `gsem` command specific to our purposes. We also assume a multispell data structure where each record corresponds to an episode nested within an individual.

3.1 Multilevel hazard models

In the multispell dataset, *idvar* identifies the individuals, *timevar* records the survival time, *t0var* records entry time, and *event* is a dummy variable recording the occurrence of the event under study. The inclusion of the random intercept requires specification of a latent variable at the level of individuals. This latent variable might be specified as `U[`*idvar*`]`. Instead of `U`, one can choose any word beginning with a capital letter. However, specifying `[`*idvar*`]` after the chosen word is mandatory; this syntax element tells Stata that the latent variable is random intercept, which is constant within the individuals.

To fit a multilevel hazard model assuming distribution *family*, one should type

```
gsem (timevar <- indepvars U[idvar], family(family, fail(event) lt(t0var)))
```

3.2 Piecewise-constant multilevel hazard models

Most of the empirical applications in demography use piecewise-linear exponential hazard models. Thus I will focus on simple exponential hazard models. Exponential hazard models can easily be fit as Poisson models of events, provided that the explanatory variables include the natural log of the duration of the current spell (Skrondal and Rabe-Hesketh 2004). The reason is that if survival time t follows an exponential distribution with parameter h, the expected number of failures follows a Poisson distribution with parameter ht (Holford 1980).

Define *durvar* as *timevar* minus *t0var*. *durvar* thus measures the duration of the current spell. The multilevel piecewise-constant exponential hazard model can be fit as follows:

```
gsem (event <- indepvars U[idvar], poisson exposure(durvar))
```

Duration dependence is allowed if *indepvars* includes *t0var*, other variables generated from *t0var*, or indicator variables capturing the rank order of the current spell.

3.3 Fitting nonrecursive systems without observed endogenous variables

Systems of piecewise-constant exponential models require separate latent variables for the equations. Let `U1[`*idvar*`]` and `U2[`*idvar*`]` be the equation-specific latent variables (random intercepts). Two equations can be jointly estimated as follows:

```
gsem
   (event_1 <- indepvars_1 U1[idvar], poisson exposure(durvar))
   (event_2 <- indepvars_2 U2[idvar], poisson exposure(durvar))
```

`gsem` automatically estimates the variance–covariance matrix of the random effects. The loadings of the latent variables will be constrained to 1.

event_1 and *event_2* might refer to recurrent events of the same kind. (For simplicity, the outcomes of sequential choices, like the timing for first, second, and higher-order births, are also treated as recurrent events.) The practice of multilevel modeling suggests that equations for recurrent events should share the same latent variable. Even if K different equations are used to model recurrent events of the same kind, the equations should include a single latent variable, not K different latent variables. This strategy of modeling first, second, and higher-order births is present in Lillard's (1993) seminal article. The reason for using one instead of K different latent variables is computational: integrating out one latent variable takes less time than integrating out K jointly distributed latent variables. The syntax for recurrent events is

```
gsem
   (event_1 <- indepvars_1 U[idvar], poisson exposure(durvar))
   (event_2 <- indepvars_2 U[idvar], poisson exposure(durvar))
```

`gsem` automatically constrains the loading of the latent variable to 1 in the first equation but estimates the loadings in the other equations and the variance of the latent variable.

Lillard (1993) modeled recurrent occurrences of births jointly with marital dissolution. The joint modeling of recurrent events nested within another process can be implemented as follows: Variables *event_11* and *event_12* capture the occurrences of the recurrent events. Variable *event_2* measures the termination of another process within which the occurrences of *event_11* and *event_12* are nested. The syntax, which combines the previous syntax elements, is

```
gsem
   (event_11 <- indepvars_12 U1[idvar], poisson exposure(durvar))
   (event_12 <- indepvars_12 U1[idvar], poisson exposure(durvar))
   (event_2 <- indepvars_2 U2[idvar], poisson exposure(durvar))
```

3.4 Fitting recursive systems with observed endogenous variables

For simplicity, consider one survival process and one probit equation. Again, *event* is the variable indicating failures. *xvar* is the endogenous dummy variable in the hazard equation. *indepvars* includes the exogenous variables appearing in both the hazard and the probit equations. Finally, *zvars* contains the excluded instrument (or the list of excluded instruments), which appears only in the probit equation. The syntax is

```
gsem
   (event <- xvar indepvars U[idvar], poisson exposure(durvar))
   (xvar  <- zvars indepvars V[idvar], probit)
```

gsem also allows one to estimate more complicated systems. Consider a hazard model that includes two endogenous dummy variables. (For an example, see Impicciatore and Billari [2012].) The syntax for fitting models of this kind is

```
gsem
   (event <- xvar_1 xvar_2 indepvars U[idvar], poisson exposure(durvar))
   (xvar_1  <- zvars_1 indepvars V1[idvar], probit)
   (xvar_2  <- zvars_2 indepvars V2[idvar], probit)
```

One can also fit a hazard model with an endogenous qualitative variable jointly with a multinomial selection model:

```
gsem
   (event <- xvar indepvars U[idvar], poisson exposure(durvar))
   (xvar  <- zvars indepvars V[idvar], mlogit)
```

4 Example 1. Nonrecursive simultaneous equations for hazards

4.1 Introduction: The research problem and the dataset

Our first example considers the relationship between education and second-birth rates. We hypothesize that higher education has a positive effect on second-birth hazards (even when higher education has a negative effect on first births). We use a sample dataset on married American women that was shipped with the statistical software aML (Lillard and Panis 2003). The original dataset was converted into a multispell dataset. You can obtain the data as follows:

```
. use "http://web.uni-corvinus.hu/bartus/stata/divorce2.dta"
(Data on marriages (source: divorce4.raw, shipped with aML))
```

The data have a multilevel structure: spells are nested within conception episodes, and conception episodes are nested within individuals. Our sample data include the first two conception episodes within the first marriage. Conception episodes within marriages are identified with the variable numkids, measuring the number of children at the beginning of conception episodes. The duration of a conception episode is the difference between two variables, time and mardur. mardur measures the duration of the marriage at the beginning of each spell, while time measures the date of separation (or interview date).

We begin by creating separate dummies for first and second conceptions. We use the separate command to separate the samples for the study of first and second births. Then, we define the model. The key explanatory variable is hereduc, which is a categorical variable with three categories: primary, secondary, and higher education. (Actually, these variables are computed from years of schooling.) I chose secondary education as the reference category. To keep matters simple, I used only the age at the beginning of the conception spell (that is, the mother's age when the first child was born) as a control variable. We place the independent variables and the model definition in global macros. The commands are

```
. separate birth, by(numkids)
  (output omitted)
. global xvars ib2.hereduc age
. global model poisson exposure(dur)
```

We begin our analyses with fitting the model separately. We fit a multilevel model because records within the multispell dataset are nested within individuals. The command and result are

```
. gsem (birth2 <- $xvars U[id], $model)
  (output omitted)
```

```
------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
birth2 <-    |
     hereduc |
   <12 years |  -.0389349   .0727403    -0.54   0.592    -.1815032    .1036334
   16+ years |   .4029357   .1093571     3.68   0.000     .1885998    .6172716
             |
         age |  -.0914562   .0062165   -14.71   0.000    -.1036404    -.079272
             |
       U[id] |          1  (constrained)
             |
       _cons |   -1.86012   .0506612   -36.72   0.000    -1.959414   -1.760826
     ln(dur) |          1  (exposure)
-------------+----------------------------------------------------------------
  var(U[id]) |   .6216596    .068415                      .5010442    .7713105
------------------------------------------------------------------------------
```

The third level of the `hereduc` variable (16+ years of education) has a positive and statistically significant coefficient. This suggests that second-birth rates are relatively high among educated women. In the rest of this section, we control for sample selection and endogeneity to check whether the estimate of 0.403 is robust.

4.2 Joint model for first and second births

Our first concern with the previous result is it might arise because of a selection effect. Education has a negative effect on the transition to first birth, so education will be positively correlated with unobserved causes of fertility in samples of mothers (Kravdal 2007). Therefore, the comparison of the fertility outcomes across educational categories in the sample of mothers measures not only the true effect of education but also the effect of unobserved preferences or personality traits (Kravdal 2001). This selection effect can be controlled for if the parity-specific transitions are modeled jointly by adding person-specific random intercepts to both the second- and first-birth equations. The command is

```
. gsem (birth2   <- $xvars U[id], $model)
>      (birth1   <- $xvars U[id], $model)
  (output omitted)
```

Results are not shown because the coefficient of higher education is again positive and statistically significant and, more importantly, the exact value of the estimate, 0.381, is close to the previous estimate. This finding suggests that the positive effect of higher education cannot be explained in terms of sample selection.

4.3 Joint model for second births and marital dissolutions

Our second concern is the dependence of the birth process on the latent hazard of marital dissolution; pessimistic expectations regarding the duration of the marriage are likely to affect second births. To eliminate the bias arising from simultaneity, we now turn to fitting a joint model of the timing of second births and the timing of marital dissolutions. Because the joint model includes reduced-form equations, identifying the structural parameters requires excluded instruments. We assume that age affects only second-birth rates, while the hazard of marital disruption depends exclusively on marriage duration. In other words, age and marital duration are the excluded instruments in the respective birth and dissolution equations. The reduced-form equations include all variables appearing in all structural equations. We again specify the model using a global macro:

```
. global xvars ib2.hereduc age mardur
```

We restrict the analysis to married mothers of one child. The reduced-form system is estimated as follows:

```
. gsem (birth2  <- $xvars U[id], $model)
>      (divorce <- $xvars V[id], $model)
>      if numkids==2
  (output omitted)
Generalized structural equation model           Number of obs     =      5,100
  (output omitted)

                    Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]

birth2 <-
     hereduc
  <12 years   -.0028288     .068905    -0.04   0.967    -.1378801    .1322225
  16+ years    .3128652    .1036965     3.02   0.003     .1096238    .5161066

        age   -.0396747    .0087697    -4.52   0.000     -.056863   -.0224863
     mardur   -.1071725    .0138669    -7.73   0.000    -.1343512   -.0799939

      U[id]           1  (constrained)

      _cons   -1.180374    .0971953   -12.14   0.000    -1.370873   -.9898746
    ln(dur)           1  (exposure)

divorce <-
     hereduc
  <12 years   -.1479639    .1482184    -1.00   0.318    -.4384667    .1425389
  16+ years   -.4958348    .2942543    -1.69   0.092    -1.072563    .0808931

        age   -.0970939    .0212381    -4.57   0.000    -.1387197   -.0554681
     mardur    .0857423    .0290449     2.95   0.003     .0288152    .1426693

      V[id]           1  (constrained)

      _cons   -4.391784    .3508382   -12.52   0.000    -5.079414   -3.704153
    ln(dur)           1  (exposure)

 var(U[id])    .4275274    .0636296                      .3193583    .5723342
 var(V[id])     .478624    .3789376                      .1014095    2.258969

 cov(V[id],
     U[id])   -.0836689    .1472204    -0.57   0.570    -.3722154    .2048777
```

Introducing the latent variables implies that five additional parameters should be estimated: the loadings and the variances of the latent variables and the covariance of the latent variables. We can identify three of these parameters because the variance–covariance matrix of the dependent variables includes the variances and the covariance of the outcomes. To identify these parameters, Stata constrains the loadings to unity.

To interpret the results, recall that the birth equation is not a structural equation but a reduced-form equation (see section 2.2). The structural effect of higher education must be recovered using (1). The structural effect is a nonlinear combination of four reduced-form coefficients. This nonlinear combination can easily be computed with the `nlcom` command:

```
. nlcom _b[birth2:3.hereduc] - (_b[birth2:mardur] / _b[divorce:mardur]) *
> _b[divorce:3.hereduc]
       _nl_1:  _b[birth2:3.hereduc] - (_b[birth2:mardur] / _b[divorce:mardur])
> * _b[divorce:3.hereduc]

             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
       _nl_1 |  -.3068977   .4508039    -0.68   0.496    -1.190457    .5766616
```

The structural effect of higher education in the second-birth equation, labeled `_nl_1` in the output, is small and lacks statistical significance. This suggests that the partial correlation between higher education and second-birth rates is not direct but might be mediated by the latent separation hazard. This conjecture can easily be tested. Using (2.2), we can compute the structural effect of higher education on the dissolution hazard as follows:

```
. nlcom _b[divorce:3.hereduc] - (_b[divorce:age] / _b[birth2:age]) *
> _b[birth2:3.hereduc]
       _nl_1:  _b[divorce:3.hereduc] - (_b[divorce:age] / _b[birth2:age]) *
> _b[birth2:3.hereduc]

             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
       _nl_1 |  -1.261495   .4305668    -2.93   0.003     -2.10539   -.4175992
```

Using (2.2), we see that the effect of the dissolution hazard on the second-birth hazard is

```
. nlcom _b[birth2:mardur] / _b[divorce:mardur]
       _nl_1:  _b[birth2:mardur] / _b[divorce:mardur]

             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
       _nl_1 |  -1.249938   .4477062    -2.79   0.005    -2.127426   -.3724502
```

These linear combinations support the hypothesis that the positive effect of higher education on second births is mediated by the latent hazard of marital separation: highly educated women tend to live in relatively stable marriages, and marital stability has a positive effect on second-birth rates.

4.4 Joint model for first births, second births, and marital dissolution

In the previous subsections, we first modeled first- and second-birth processes, then modeled second-birth and marital dissolution processes jointly. The respective concerns were sample selection bias and endogeneity bias. We can address these concerns at the same time and estimate the first-birth, second-birth, and marital dissolution equations jointly. Indeed, this model is very close to the classic multilevel multiprocess model presented in Lillard (1993). As described in section 3.3, we specify two correlated latent variables for the respective conception and marital dissolution processes. The syntax is

```
. gsem (birth2  <- $xvars U[id], $model)
>      (birth1  <- $xvars U[id], $model)
>      (divorce <- $xvars V[id], $model)
  (output omitted)
```

We omit the output because the ultimate interest lies in the structural coefficients. These can be recovered by computing the appropriate nonlinear combination:

```
. nlcom _b[birth2:3.hereduc] - (_b[birth2:mardur] / _b[divorce:mardur]) *
> _b[divorce:3.hereduc]

       _nl_1:  _b[birth2:3.hereduc] - (_b[birth2:mardur] / _b[divorce:mardur])
> * _b[divorce:3.hereduc]

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |   .1031687   .1907404     0.54   0.589    -.2706757    .4770131
------------------------------------------------------------------------------
```

Again, there is no evidence that higher education would have a direct effect on second-birth rates. By contrast, there is evidence that the positive effect of higher education is an indirect one, mediated by the latent dissolution hazard. The respective nonlinear combinations that estimate the direct effect of higher education on the dissolution hazard and the effect of the dissolution hazard on the second-birth hazard are as follows:

```
. nlcom _b[divorce:3.hereduc] - (_b[divorce:age] / _b[birth2:age]) *
> _b[birth2:3.hereduc]

       _nl_1:  _b[divorce:3.hereduc] - (_b[divorce:age] / _b[birth2:age]) *
> _b[birth2:3.hereduc]

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |  -.5290925   .2280571    -2.32   0.020    -.9760761   -.0821088
------------------------------------------------------------------------------

. nlcom _b[birth2:mardur] / _b[divorce:mardur]

       _nl_1:  _b[birth2:mardur] / _b[divorce:mardur]

------------------------------------------------------------------------------
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _nl_1 |  -.8249733   .2544315    -3.24   0.001     -1.32365   -.3262967
------------------------------------------------------------------------------
```

5 Example 2. Hazard models with endogenous dummy variables

5.1 Introduction: The research problem and the dataset

Our second example is about examining the impact of hospital delivery on child mortality. We hypothesize that children delivered in hospitals have a lower death hazard than similar children delivered at home. We use a modified and Stata-compatible version of the children dataset shipped with the statistical software aML to replicate one of the examples in the aML manual (Lillard and Panis 2003). You can obtain the data as follows:

```
. use "http://web.uni-corvinus.hu/bartus/stata/children1.dta", clear
(Child mortality data (source: aML))
```

The data have a multispell and multilevel structure: spells are nested within children, identified with the variable `bid`, and children are nested within mothers, identified with the variable `id`. For simplicity, the survival process is split into two spells; the first spell lasts three months (or less in case of early death). Episode splitting is motivated by the observation that child mortality is relatively large in the first three months. The survival process is described by three variables: `death` indicates deaths, `dur` records the duration of the spell (in months), and `age0` is the age of the child (in months) at the beginning of the spell.

We begin with estimating a simple multilevel child mortality hazard equation. The explanatory variables include the hospital dummy, education, and an indicator for being aged three months at the beginning of the current spell. For simplicity, we assume that the mortality hazard is constant within the spells. We use a random intercept at the level of mothers to model the interdependence of spells within mothers. We place the independent variables and the model definition in global macros. The commands are

```
. global death hospital i.edu  i.age0
. global model poisson exposure(dur)
. gsem (death <- $death  U[id], $model)
  (output omitted)
```

Estimates are not shown. The coefficient of the hospital dummy is negative (the estimate is -0.382) but statistically not significant ($p = 0.064$). The robustness of this estimate is examined in the next subsection.

5.2 Joint estimation of hazard and probit equations

Finding no statistically significant negative effect of hospital delivery might be due to a selection effect. Mothers are aware of their health status and form an expectation about the mortality of their child. Hospital delivery is chosen by mothers who fear losing their baby and believe hospitals reduce this risk. By contrast, home delivery is chosen by women with a low risk of losing their baby. In short, hospital delivery is correlated with factors affecting child mortality. To control for this endogeneity bias, one can fit the hazard model jointly with a probit model of hospital delivery on education and distance to the nearest hospital, the latter being the excluded instrument. The joint model is fit as follows:

```
. global hospital distance i.edu
. gsem (death    <- $death    U[id], $model)
>      (hospital <- $hospital V[id], probit)
  (output omitted)
Generalized structural equation model          Number of obs     =      2,002
  (output omitted)
------------------------------------------------------------------------------
              |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
death <-      |
     hospital |  -.5131628   .2411954    -2.13   0.033    -.9858971   -.0404285
              |
         educ |
 high school  |  -.2625067   .1909157    -1.37   0.169    -.6366945    .1116811
     college  |  -2.021169   .7341519    -2.75   0.006     -3.46008   -.5822573
              |
       3.age0 |  -4.920847   .1656668   -29.70   0.000    -5.245548   -4.596146
              |
        U[id] |          1  (constrained)
              |
        _cons |   -3.12697   .1432276   -21.83   0.000    -3.407691   -2.846249
      ln(dur) |          1  (exposure)
--------------+---------------------------------------------------------------
hospital <-   |
     distance |  -.0231453   .0175738    -1.32   0.188    -.0575894    .0112987
              |
         educ |
 high school  |    2.01218   .2895358     6.95   0.000       1.4447     2.57966
     college  |   3.148736   .5114086     6.16   0.000     2.146393    4.151078
              |
        V[id] |          1  (constrained)
              |
        _cons |  -2.209737   .2767038    -7.99   0.000    -2.752066   -1.667407
--------------+---------------------------------------------------------------
    var(U[id])|   .4091622   .2339894                      .1333875    1.255093
    var(V[id])|   4.149642   1.079965                      2.491617    6.910987
--------------+---------------------------------------------------------------
    cov(V[id],|
        U[id])|   .2157169   .1885667     1.14   0.253    -.1538671    .5853009
------------------------------------------------------------------------------
```

The coefficient of the hospital delivery variable is now statistically significant. The estimate of -0.513 is larger than that appearing in the separate model. This suggests that hospital delivery has the expected negative effect on mortality, but this effect was partially suppressed by the aforementioned selection effect.

The present example assumes that the hazard of death is constant within the spells. Descriptive analyses, not reported in this article, suggest this assumption is unrealistic. The hazard is monotonically decreasing in the first three months, while it is approximately constant after surviving the first three months. A more realistic model specification would be a Weibull model, which can be fit as follows:

```
. global model family(weibull, fail(death) lt(age0))
. gsem (month    <- $death    U[id], $model)
>      (hospital <- $hospital V[id], probit)
  (output omitted)
```

Note that the dependent variable in the hazard equation is the variable recording the survival time. The output is not reported because the coefficient of the hospital delivery dummy is statistically significant, and the size of the coefficient is very close to the previously estimated −0.513.

5.3 Joint estimation of hazard and multinomial logit equations

Suppose that children can be delivered in public hospitals, in private hospitals, and at home. Suppose further that hospital delivery improves life expectancy, but the negative effect of hospital delivery on child mortality differs between private and public hospitals. Women are expected to select the delivery form, which minimizes the risks but also economizes on (travel and other) costs. Again, the chosen form of delivery will be correlated with factors affecting the health of the child. To eliminate the endogeneity bias, one must fit the hazard model jointly with a multinomial model of delivery choice. The `gsem` command allows one to fit hazard models jointly with multinomial selection equations. To illustrate, we use a modified version of the child mortality dataset. The specification of the hazard and the selection equations is not changed. The only change is that we use a multinomial logit selection model instead of a probit model. The commands are

```
. use "http://web.uni-corvinus.hu/bartus/stata/children2.dta", clear
(Child mortality data (source: aML))
. global death i.hospital i.edu  i.age0
. global hospital distance i.edu
. global model poisson exposure(dur)
. gsem (death    <- $death    U[id], $model)
>      (hospital <- $hospital V[id], mlogit)
  (output omitted)
Generalized structural equation model           Number of obs     =      2,002
  (output omitted)

------------------------------------------------------------------------------
              |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
death <-      |
              |
     hospital |
           1  |  -.5085083   .2527484    -2.01   0.044    -1.003886   -.0131305
           2  |  -3.024408   .5896255    -5.13   0.000    -4.180053   -1.868763
              |
         educ |
 high school  |  -.1914267   .1871073    -1.02   0.306    -.5581502    .1752968
     college  |  -1.965382   .7309886    -2.69   0.007    -3.398093   -.5326702
              |
       3.age0 |  -4.836307   .1661777   -29.10   0.000     -5.16201   -4.510605
              |
        U[id] |          1  (constrained)
              |
        _cons |  -2.885532   .1392459   -20.72   0.000    -3.158449   -2.612615
      ln(dur) |          1  (exposure)
--------------+---------------------------------------------------------------
0.hospital    |  (base outcome)
--------------+---------------------------------------------------------------
1.hospital <- |
     distance |  -.0530196   .0337748    -1.57   0.116    -.1192171    .0131779
              |
         educ |
 high school  |   3.239981   .4779804     6.78   0.000     2.303157    4.176805
     college  |   4.849681   .7717843     6.28   0.000     3.337011     6.36235
              |
        V[id] |          1  (constrained)
              |
        _cons |  -3.793961   .4544799    -8.35   0.000    -4.684725   -2.903196
--------------+---------------------------------------------------------------
2.hospital <- |
     distance |  -.0060001   .0212732    -0.28   0.778    -.0476949    .0356946
              |
         educ |
 high school  |   1.074023   .1810285     5.93   0.000     .7192138    1.428833
     college  |   1.707243   .3263577     5.23   0.000     1.067594    2.346892
              |
        V[id] |   .2732314   .0500081     5.46   0.000     .1752174    .3712454
              |
        _cons |  -1.293195   .1406849    -9.19   0.000    -1.568932   -1.017458
--------------+---------------------------------------------------------------
    var(U[id])|   .2992881   .2126223                      .0743658    1.204497
    var(V[id])|   13.02677   2.624814                      8.776573     19.3352
--------------+---------------------------------------------------------------
    cov(V[id],|
       U[id]) |   .3196824   .3756769     0.85   0.395    -.4166308    1.055996
------------------------------------------------------------------------------
```

Again, we find that hospital delivery reduces child mortality compared with home delivery. The effect is larger in hospitals coded with 2 than in hospitals coded with 1. (It is up to the reader to interpret the 1 code as a private or as a public hospital.)

6 Concluding remarks

Demographers routinely use multilevel multiprocess models to adjust estimates for endogeneity and sample selection. In this article, I showed how multilevel multiprocess models could be fit with the `gsem` command. I provided two examples to illustrate the estimation of nonrecursive systems without observed endogenous variables and recursive systems with observed endogenous variables. The examples used sample datasets shipped with the statistical software aML, explicitly developed for multiprocess multilevel modeling (Lillard and Panis 2003). I paid special attention to identifying structural effects in nonrecursive systems.

Most of the examples in this article illustrate the estimation of systems with two equations. In some empirical applications, however, more than two equations are estimated jointly (Upchurch, Lillard, and Panis 2002; Steele et al. 2005). As the number of equations increases, the number of correlated random intercepts increases. Fitting models with a large number of random effects is slow and may have convergence problems. Referencing the classic article on multilevel multiprocess modeling (Lillard 1993), I suggested a simple rule to avoid or minimize numerical problems: the number of latent variables must be equal to the number of processes under study, but separate equations for recurrent (or sequential) occurrences of events of the same kind should share the same latent variable.

For simplicity, I used (piecewise-constant) exponential hazard models for the purpose of survival modeling. As shown in section 3.1, `gsem` supports a large class of parametric survival models. Recently, multilevel multiprocess models often rely on discrete-time (binary and multinomial) logistic regression models (Steele et al. 2005). However, the Poisson model is flexible enough to model duration dependence and represent discrete-time event-history models. In theory, systems of logit and multinomial logit models can easily be estimated with `gsem`. In conclusion, the `gsem` command is a powerful tool to fit various forms of multilevel multiprocess models. I believe the examples shown in this article will help researchers solve complicated research problems.

Acknowledgments

An earlier version of this article was presented at the 2015 German Stata Users Group Meeting in Nuremberg on June 26, 2015. The research in this article was financially supported by the János Bolyai Scholarship of the Hungarian Academy of Sciences as well as by the Hungarian Scientific Research Fund (OTKA) within the research project *Mapping Family Transitions: Causes, Consequences, Complexities, and Context* (grant number K109397).

7 References

Bartus, T., and D. Roodman. 2014. Estimation of multiprocess survival models with cmp. *Stata Journal* 14: 756–777.

Heckman, J. J. 1978. Dummy endogenous variables in a simultaneous equation system. *Econometrica* 46: 931–959.

Holford, T. R. 1980. The analysis of rates and of survivorship using log-linear models. *Biometrics* 36: 299–305.

Impicciatore, R., and F. C. Billari. 2012. Secularization, union formation practices, and marital stability: Evidence from Italy. *European Journal of Population* 28: 119–138.

Kravdal, Ø. 2001. The high fertility of college educated women in Norway: An artefact of the separate modelling of each parity transition. *Demographic Research* 5: 187–216.

———. 2007. Effects of current education on second- and third-birth rates among Norwegian women and men born in 1964: Substantive interpretations and methodological issues. *Demographic Research* 17: 211–246.

Lee, L. 1979. Identification and estimation in binary choice models with limited (censored) dependent variables. *Econometrica* 47: 977–996.

Lillard, L. A. 1993. Simultaneous equations for hazards: Marriage duration and fertility timing. *Journal of Econometrics* 56: 189–217.

Lillard, L. A., M. J. Brien, and L. J. Waite. 1995. Premarital cohabitation and subsequent marital dissolution: A matter of self-selection? *Demography* 32: 437–457.

Lillard, L. A., and C. W. A. Panis. 2003. *aML Multilevel Multiprocess Statistical Software, Version 2.0.* Los Angeles, CA: EconWare.

Lillard, L. A., and L. J. Waite. 1993. A joint model of marital childbearing and marital disruption. *Demography* 30: 653–681.

Maddala, G. S. 1983. *Limited-Dependent and Qualitative Variables in Econometrics.* Cambridge: Cambridge University Press.

Roodman, D. 2011. Fitting fully observed recursive mixed-process models with cmp. *Stata Journal* 11: 159–206.

Skrondal, A., and S. Rabe-Hesketh. 2004. *Generalized Latent Variable Modeling: Multilevel, Longitudinal, and Structural Equation Models.* Boca Raton, FL: Chapman & Hall/CRC.

Steele, F., C. Kallis, H. Goldstein, and H. Joshi. 2005. The relationship between childbearing and transitions from marriage and cohabitation in Britain. *Demography* 42: 647–673.

Upchurch, D. M., L. A. Lillard, and C. W. A. Panis. 2002. Nonmarital childbearing: Influences of education, marriage, and fertility. *Demography* 39: 311–329.

About the author

Tamás Bartus is an associate professor of sociology at the Corvinus University of Budapest. His research interests include the impact of education on partnership stability and fertility.

The Stata Journal (2017)
17, Number 2, pp. 462–489

A flexible parametric competing-risks model using a direct likelihood approach for the cause-specific cumulative incidence function

Sarwar Islam Mozumder
Department of Health Sciences
University of Leicester
Leicester, UK
si113@le.ac.uk

Mark J. Rutherford
Department of Health Sciences
University of Leicester
Leicester, UK
mark.rutherford@le.ac.uk

Paul C. Lambert
Department of Health Sciences
University of Leicester
Leicester, UK
and
Medical Epidemiology and Biostatistics
Karolinska Institutet
Stockholm, Sweden
paul.lambert@le.ac.uk

Abstract. In competing-risks analysis, the cause-specific cumulative incidence function (CIF) is usually obtained in a modeling framework by either 1) transforming on all cause-specific hazards or 2) transforming by using a direct relationship with the subdistribution hazard function. We expand on current competing-risks methodology from within the flexible parametric survival modeling framework and focus on the second approach. This approach models all cause-specific CIFs simultaneously and is more useful for answering prognostic-related questions. We propose the direct flexible parametric survival modeling approach for the cause-specific CIF. This approach models the (log cumulative) baseline hazard without requiring numerical integration, which leads to benefits in computational time. It is also easy to make out-of-sample predictions to estimate more useful measures and incorporate alternative link functions, for example, logit links. To implement these methods, we introduce a new estimation command, `stpm2cr`, and demonstrate useful predictions from the model through an illustrative melanoma dataset.

Keywords: st0482, stpm2cr, survival analysis, competing risks, flexible parametric models, subdistribution hazard, cumulative incidence function

1 Introduction

In competing-risks analysis, researchers consider the cause-specific cumulative incidence function (CIF), which is the probability of failure of an event in the presence of other competing events. From within the modeling framework, the CIF is usually obtained by either 1) estimating all the cause-specific hazard (CSH) functions or 2) transforming by using a direct relationship with the subdistribution hazard (SDH) function for the cause of interest. Many tools in Stata allow us to estimate the cause-specific CIF. We can obtain an empirical, nonparametric estimate of the cause-specific CIF using the user-written command `stcompet`, which applies the Aalen–Johansen approach (Coviello and Boggess 2004).

Alternatively, we can fit regression models on either the CSH or SDH scale depending on the research question (Sapir-Pichhadze et al. 2016; Noordzij et al. 2013; Koller et al. 2012). CSH regression models can be fit from within a semiparametric approach using a typical Cox model or from within a flexible parametric modeling framework using the user-written postestimation command `stpm2cif`. This command works with an expanded dataset in which each patient has a row for each cause and is used after fitting a cause-specific flexible parametric survival model (FPM) with `stpm2` to model all causes (Hinchliffe and Lambert 2013; Lambert and Royston 2009; Lambert et al. 2011; Royston and Parmar 2002).

The preferred method for modeling covariate effects on the cause-specific CIF is the Fine and Gray (1999) model, available through the `stcrreg` command. However, this approach allows us to model only one event using the partial likelihood. We must fit separate models for each competing event if we want to understand the overall impact of a covariate on risk.

Competing-risks models can also be fit using the user-written command `stcrprep`, which restructures the data and calculates appropriate weights (Lambert Forthcoming). Standard Stata survival analysis commands can then be used to fit models more computationally efficiently, for example, fitting the Fine and Gray model and parametric models for the cause-specific CIF (Lambert, Wilkes, and Crowther Forthcoming).

We introduce parametric methods using the full likelihood because smooth estimates can be obtained for the baseline cause-specific CIF or SDH for a particular cause that can easily extend to incorporate nonproportional SDHs. Fitting parametric models for the cause-specific CIF in this way is computationally quicker than fitting models with `stcrprep` because no numerical integration or data restructure is required. An additional advantage of these models is that we can model all cause-specific CIFs simultaneously and model covariate effects on all competing causes. Jeong and Fine (2006) investigated a direct parametric inference approach and defined a likelihood that allows us to model all the cause-specific CIFs simultaneously. We extend this approach to FPMs, in which it is easy to model time-dependent effects and obtain useful out-of-sample predictions.

Others have also proposed modeling the SDH under alternative link functions. For example, Gerds, Scheike, and Andersen (2012) propose the proportional log-odds model for the cause-specific CIF, which offers an alternative interpretation. However, the interpretation is not as simple as modeling a single event and suffers from similar issues in interpretation as the complementary log-log link function. Incorporating such alternative link functions on the cause-specific CIF is also easy to implement using the approach we outline in this article.

This article continues as follows. In section 2, we introduce the methods for direct inference on the cause-specific CIF under an FPM framework. In section 3, we outline the syntax of `stpm2cr`, which fits the models introduced in section 2. In section 4, we describe syntax for postestimation using `predict` after fitting models with `stpm2cr`. In section 5, we provide illustrative examples. Finally, we conclude by discussing the approach's limitations and potential extensions.

2 Methods

Let T be the time to event for any K competing causes $k = 1, \ldots, K$, and let D denote the type of event, where $D = 1, \ldots, K$. Here we consider the events to be death from different causes, so the cause-specific CIF, $F_k(t)$, is the probability of dying from a particular cause $D = k$ by time t while also being at risk of dying from other causes (Putter, Fiocco, and Geskus 2007):

$$F_k(t) = P(T \leq t, D = k)$$

The all-cause CIF, $F(t)$, which is the probability of dying from any of the K causes by time t, is the sum of all K cause-specific CIFs, $F_k(t)$, and can also be expressed as the complement of the overall survival function, $S(t)$:

$$F(t) = P(T \leq t) = \sum_{j=1}^{K} F_j(t) = 1 - S(t)$$

2.1 Cause-specific hazard function

The cause-specific CIF, $F_k(t)$, can be expressed as a function of either the CSH functions for all K causes or the SDH for cause k. The CSH function, $h_k^{cs}(t)$, gives the instantaneous mortality rate from a particular cause k given that the patient is still alive at time t in the presence of all other causes of death.

$$h_k^{cs}(t) = \lim_{\Delta t \to 0} \frac{P(t < T \leq t + \Delta t, D = k | T > t)}{\Delta t}$$

The cause-specific CIF can be expressed as a function of the CSHs for all K causes:

$$F_k(t) = \int_0^t \left[\exp\left\{ -\int_0^t \sum_{j=1}^{K} h_j^{cs}(u) du \right\} \right] h_k^{cs}(u) du$$

Note here that the leading term within the integral gives the overall survival function,

$$S(t) = \exp\left\{ -\int_0^t \sum_{j=1}^{K} h_j^{cs}(u) du \right\}$$

2.2 Subdistribution hazard function

Gray (1988) introduces the SDH for cause k, $h_k^{sd}(t)$, which gives a direct relationship with the cause-specific CIF. This has the mathematical formulation

$$\begin{aligned} h_k^{sd}(t) &= \lim_{\Delta t \to 0} \frac{P\left\{t < T \leq t + \Delta t, D = k | T > t \cup (T \leq t \cap D \neq k)\right\}}{\Delta t} \\ &= \frac{\frac{d}{dt}\left\{F_k(t)\right\}}{1 - F_k(t)} = -\frac{d\left[\ln\left\{1 - F_k(t)\right\}\right]}{dt} \end{aligned}$$

and is interpreted as the instantaneous rate of failure at time t from cause k among those still alive or those who have died from any of the other $K-1$ competing causes excluding cause k. The SDH rate is not a conventional epidemiological rate because of the risk set (Lau, Cole, and Gange 2009) and should not be interpreted as a standard hazard rate.

The cause-specific CIF can be expressed directly in terms of the SDH function for cause k using standard survival relationships along with the cumulative SDH for cause k, $H_k^{sd}(t)$,

$$F_k(t) = 1 - \exp\left\{-H_k^{sd}(u)\right\} \qquad \text{and} \qquad H_k^{sd}(t) = \int_0^t h_k^{sd}(u)du$$

Using the SDH functions for all K causes, we can also obtain the CSH functions, $h_k^{cs}(t)$, for all K causes (Latouche et al. 2007),

$$h_k^{cs}(t) = h_k^{sd}(t)\left[1 + \frac{\left\{\sum_{j=1}^{K} F_j(z)\right\} - F_k(t)}{1 - \sum_{j=1}^{K} F_j(t)}\right]$$

2.3 Regression modeling

The most common model for the SDH for cause k is the Fine and Gray (1999) model, which is expressed in a similar way to the cause-specific Cox proportional hazards model because it assumes proportionality of covariate effects on the SDH scale,

$$h_k^{sd}(t|\mathbf{x}) = h_{0,k}^{sd}(t)\exp\left(\mathbf{x}\boldsymbol{\beta}_k^{sd}\right) \tag{1}$$

where $\boldsymbol{\beta}_k^{sd}$ are log-SDH ratios for cause k. The SDH ratios, $\exp\left(\boldsymbol{\beta}_k^{sd}\right)$, are interpreted as the association on the effect of a covariate on risk (refer to Wolbers et al. [2014] for more details on interpretation). We focus on implementing and extending the SDH regression model in (1) from within the FPM approach.

2.4 Likelihood estimation

Jeong and Fine (2006) showed that we can simultaneously fit parametric models that directly fit covariate effects on the cause-specific CIF for all k causes, $F_k(t|\mathbf{x}_k)$ $(k = 1, \ldots, K)$, without requiring indirect specification through the CSHs. Hence, for an observable failure time t_i, with independent and noninformative right censoring, for each individual $i = 1, \ldots, N$, the likelihood for direct inference on the cause-specific CIF is

$$L = \prod_{i=1}^{N} \left[\left(\prod_{j=1}^{K} \left[h_j^{sd} \left(t_i | \mathbf{x_j} \right) \left\{ 1 - F_j\left(t_i\right) \right\} \right]^{\delta_{ij}} \right) \left\{ 1 - \sum_{j=1}^{K} F_j(t_i|\mathbf{x}_j) \right\}^{1-\sum\limits_{j=1}^{K} \delta_{ij}} \right] \tag{2}$$

where the censoring indicator, δ_{ik}, tells us whether an individual died from any cause k $(\delta_{ik} = 1)$, or not $(\delta_{ik} = 0)$. Note that the cause-specific CIF, $F_k(t)$, in (2) is not a proper cumulative distribution function and is instead referred to as a subdistribution function because $\lim_{t\to\infty} F_k(t) < 1$ (Andersen et al. 2012).

2.5 Flexible parametric regression on the cause-specific cumulative incidence function

Using the likelihood in (2), we can fit a parametric survival model simultaneously for all K cause-specific CIFs. We apply the likelihood to the FPM approach described by Royston and Parmar (2002) and extend it using restricted cubic splines, $s_k(\ln(t); \boldsymbol{\gamma}_k, \mathbf{m}_k)$, with $M - 1$ degrees of freedom, where s_k is a restricted cubic spline function for cause k on log-time and consists of a vector of M knots, $\mathbf{m}$; a vector of $M - 1$ parameters, $\boldsymbol{\gamma}$; and covariates, $\mathbf{x}_k$ (Durrleman and Simon 1989). The following model can be specified through a general link function, $g(\cdot)$, for each of the $k = 1, \ldots, K$ cause-specific CIF with covariates, $\mathbf{x}_k$,

$$\begin{aligned} g\left\{F_k(t|\mathbf{x}_{ik})\right\} &= s_k\left\{\ln(t); \boldsymbol{\gamma}_k, \mathbf{m}_k\right\} + \mathbf{x_k}\boldsymbol{\beta}_k \\ &= \gamma_{0k} + \gamma_{1k}z_{1k} + \cdots + \gamma_{(M-1)k}z_{(M-1)k} + \mathbf{x}_k\boldsymbol{\beta}_k \end{aligned} \tag{3}$$

where $z_{1k}, \ldots, z_{(M-1)k}$ are the basis functions of the restricted cubic splines and are defined as

$$\begin{aligned} z_{1k} &= \ln(t) \\ z_{jk} &= \left\{\ln(t) - m_{jk}\right\}_+^3 - \phi_{jk}\left\{\ln(t) - m_{1k}\right\}_+^3 - (1 - \phi_{jk})\left\{\ln(t) - m_{Mk}\right\}_+^3 \\ j &= 2, \ldots, M - 1 \end{aligned}$$

where

$$\phi_{jk} = \frac{m_{Mk} - m_{jk}}{m_{Mk} - m_{1k}}$$

and

$$(u)_+ = \begin{cases} u, & \text{if } u < 0 \\ 0, & \text{otherwise} \end{cases}$$

Through the general link function $g(\cdot)$ for the cause-specific CIF, $F_k(t)$, in (3), we can apply similar transformations described in Royston and Parmar (2002) for the survival function. Lambert, Wilkes, and Crowther (Forthcoming) offer more details on the various link functions available for the cause-specific CIF, but here we introduce only the complementary log-log (`cloglog`) and logit link functions (see table 1).

Table 1. Common transformations on the general link function for the cause-specific CIF

Parameters	Link function	Link name
log-subdistribution hazard ratios	$\ln\left[-\ln\{1-F_k(t\vert\mathbf{x}_k)\}\right]$	`cloglog`
log odds-ratios	$\frac{F_k(t\vert\mathbf{x}_k)}{1-F_k(t\vert\mathbf{x}_k)}$	`logit`

2.6 Time-dependent effects

To relax the proportionality assumption, we fit interactions between the associated covariates and the spline function for log-time. This allows us to introduce a new set of knots, $\mathbf{m}_{ek}$, that represent the eth time-dependent effect for cause k with associated parameters, $\boldsymbol{\alpha}_{ek}$. If there are $e = 1, \dots, E$ time-dependent effects, we can extend the model in (3) to

$$\eta_k(t) = s_k\left\{\ln(t); \boldsymbol{\gamma}_k, \mathbf{m}_{0k}\right\} + \mathbf{x}_k\boldsymbol{\beta}_k + \sum_{l=1}^{E} s_k\left\{\ln(t); \boldsymbol{\alpha}_{lk}, \mathbf{m}_{lk}\right\} x_{lk}$$

In this approach, the spline function for different time-dependent effects can be different and usually requires fewer knots for the baseline spline function. This extends the original approach proposed by Royston and Parmar (2002). As all K causes are modeled, one can also specify different time-dependent effects for each of the k cause-specific FPM regression models.

2.7 Delayed entry

`stpm2cr` can also model left-truncated data or data with delayed entry. This is when subjects are considered to be at risk some time after $t = 0$.

2.8 Cure models

Andersson et al. (2011) proposed a method to estimate the cure proportion in a relative survival FPM framework. In the competing-risks scenario, this would occur in a situation where the cause-specific CIF is constant after a certain point in time t. Hence, by adapting the approach described by Andersson et al. (2011), we can estimate the cure proportion from within a flexible parametric model for the cause-specific CIF specified in section 2.5 by forcing the log cumulative SDH to plateau after the last knot. This involves adjusting how spline variables are calculated, so the cause-specific CIF is forced to plateau (see Andersson et al. [2011] for more details). Because we use the SDH function for cause k, on which we assume the cure must be evaluated while simultaneously modeling all other causes, the final knot must be specified after the final observed time of death, which has been set at the 110th percentile of log time. Applying the methods in Andersson et al. (2011) and the above adjustment to a specific cause $k = c$, we can fit a flexible parametric cure model with a complementary log-log link for a cause-specific CIF such that

$$\begin{aligned} F_c(t|\mathbf{x}_c) &= 1-(1-\pi_c)^{\exp\left[\gamma_{2c}z_{2c}+\cdots+\gamma_{(M-1)c}z_{(M-1)c}+\sum_{i=1}^{E} s_c\{\ln(t);\boldsymbol{\alpha}_{ic},\mathbf{m}_{ic}\}\mathbf{x}_{ic}\right]} \\ 1-\pi_c &= 1-\exp\{-\exp(\gamma_{0c}+\mathbf{x}_c\boldsymbol{\beta}_c)\} \end{aligned}$$

Therefore, the parameters γ_{0c} and $\boldsymbol{\beta}_c$ are used to estimate the cure proportion for cause $k = c$. Here we also implement a constraint on the linear spline, γ_{1c}, such that it is equal to 0.

To fit a cure model, we need to observe a plateau in the "raw" data for the cause-specific CIF on which we wish to model the cure. This is usually done for a single relevant cause, particularly the event of interest.

3 Syntax

`stpm2cr` [*equation1*] [*equation2*] ... [*equationN*] [*if*] [*in*], `events(`*varname*`)` [`cause(`*numlist*`)` `censvalue(`#`)` `noorthog` `alleq` `eform` `level(`#`)` `lininit` *maximize_options*]

where *equation1*, *equation2*, ..., *equationN* are the equations for each competing event. Note that at least two equations must be specified. The syntax of each equation is

causename: [*varlist*], `scale(`*scalename*`)` [`df(`#`)` `knots(`*numlist*`)` `tvc(`*varlist*`)` `dftvc(`*df_list*`)` `knotstvc(`*knotslist*`)` `bknots(`*knotslist*`)` `bknotstvc(`*knotslist*`)` `noconstant` `cure`]

You must `stset` your data before using `stpm2cr`; see [ST] **stset**. All events must be specified in the `failure()` option of `stset`.

3.1 Main options

Model

`events(`*varname*`)` specifies the *varname* that contains the indicators for each competing event failure. `events()` is required.

`cause(`*numlist*`)` specifies the indicator values for the competing events specified in `events()`. The indicators specified in *numlist* must be listed in the same order as the equations *equation1*, *equation2*, ..., *equationN*.

`censvalue(`#`)` specifies the indicator values in `events()` for censored individuals. The default is `censvalue(0)`.

`noorthog` suppresses orthogonal transformation of spline variables.

Reporting

`alleq` reports all equations used by `ml`. The models are fit using various constraints for parameters associated with the derivatives of the spline functions. These parameters are generally not of interest and thus are not shown by default. Also, an extra equation is used when fitting delayed-entry models and is also not shown by default.

`eform` reports the exponentiated coefficients. For models on the log cumulative-subdistribution hazard scale, `scale(hazard)`, this option gives the subdistribution hazard ratios if the covariate is not time dependent. Similarly, for models on the log cumulative-subdistribution odds scale, `scale(odds)`, this option will give odds ratios for nontime-dependent effects (see the `scale()` option).

`level(`#`)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`; see [U] **20.7 Specifying the width of confidence intervals**.

Max options

`lininit` obtains initial values by fitting only the first spline basis function (that is, a linear function of log survival-time). This is useful when models fail to converge using the initial values obtained in the usual way. However, this option is seldom needed.

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(`#`)`, [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `shownrtolerance`, `tolerance(`#`)`, `ltolerance(`#`)`, `gtolerance(`#`)`, `nrtolerance(`#`)`, `nonrtolerance`, and `from(`*init_specs*`)`; see [R] **maximize**. These options are seldom used, but the `difficult` option may be useful if there are convergence problems when fitting models that use the Aranda–Ordaz family of link functions.

3.2 Equation options

`scale(`*scalename*`)` specifies the scale on which to model the cause-specific CIF. `scale()` is required.

`scale(hazard)` fits a model on the log cumulative-subdistribution hazard scale, that is, the scale of $\ln[-\ln\{1-F_k(t)\}]$. If no time-dependent effects are specified, the resulting model assumes proportionality.

`scale(odds)` fits a model on the log cumulative-odds scale, that is, the scale of $\log\{F_k(t)\}/\{1 - F_k(t)\}$. If no time-dependent effects are specified, the resulting model assumes proportionality of the odds ratios over time.

`df(#)` specifies the degrees of freedom for the restricted cubic spline function used for the baseline subdistribution hazard rate. Usually a value between 3 and 5 is sufficient, and the choice of degrees of freedom is insensitive to parameter estimates. Using `df(1)` is equivalent to fitting a Weibull model when using `scale(hazard)`. The internal knots are placed at the centiles of the distribution of the uncensored log times with boundary knots placed at the 0th and 100th centiles. An example is provided below for `df(5)`:

Degrees of freedom	Internal knots	Centile positions (log time)
5	4	20th, 40th, 60th, 80th

`knots(`*numlist*`)` specifies knot locations for the baseline distribution function, as opposed to the default knot locations set by `df()`. The locations of the knots are placed on the log-time scale. Default knot positions are determined by the `df()` option.

`tvc(`*varlist*`)` specifies the names of time-dependent variables. Time-dependent effects are fit using restricted cubic splines. The degrees of freedom are specified using the `dftvc()` option.

`dftvc(`*df_list*`)` specifies the degrees of freedom for time-dependent effects. If the same degree of freedom is used for all time-dependent effects, then the syntax is the same as `df(#)`. With one degree of freedom, a linear effect of log time is fit. If there is more than one time-dependent effect and different degrees of freedom are required for each time-dependent effect, then the following syntax can be used: `dftvc(x1:3 x2:2 1)`, where `x1` has three degrees of freedom, `x2` has two degrees of freedom, and any remaining time-dependent effects have one degree of freedom.

`knotstvc(`*knotslist*`)` defines numlist *knotslist* as the location of the interior knots for time-dependent effects. If different knots are required for different time-dependent effects, the option is specified, for example, as follows: `knotstvc(x1 1 2 3 x2 1.5 3.5)`.

`bknots(`*knotslist*`)` is a two-element list giving the boundary knots. By default, these are located at the minimum and maximum of the uncensored survival times for all cause-specific events on the log scale.

`bknotstvc(`*knotslist*`)` gives the boundary knots for any time-dependent effects. By default, these are the same as for the `bknots()` option. They are specified on the scale defined by `scale()`. For example, `bknotstvc(x1 0.01 10 x2 0.01 8)`.

`noconstant`; see [R] **estimation options**.

`cure` is specified when fitting cure models for a particular cause. It forces the cause-specific cumulative subdistribution hazard to be constant after the last knot. When the `df()` option is used together with the `cure` option, the internal knots are placed evenly according to centiles of the distribution of the uncensored log survival-times except one, which is placed at the 95th centile, and the final knot is placed outside the last uncensored cause-specific log survival-time (110th percentile by default). Alternative knot locations can be selected using the `knots()` option. Cure models can be used only when modeling on the log cumulative-subdistribution hazard scale (`scale(hazard)`).

4 Postestimation

`stpm2cr` is an estimation command and shares most features of standard Stata estimation commands; see [U] **20 Estimation and postestimation commands**. The predictions available after fitting a model using `stpm2cr` are briefly described below.

4.1 Syntax

```
predict newvar [if] [in] [, at(varname # [varname #]) cause(numlist)
   chrdenominator(varname # [varname # ...])
   shrdenominator(varname # [varname # ...])
   chrnumerator(varname # [varname # ...])
   shrnumerator(varname # [varname # ...]) ci cif
   cifdiff1(varname # [varname # ...])
   cifdiff2(varname # [varname # ...]) cifratio csh cumodds
   cumsubhazard cured subdensity subhazard timevar(varname) uncured xb
   zeros dxb level(#)]
```

Main

`at(`*varname* # [*varname* #]`)` requests that the covariates specified by *varname* be set to #. This is a useful way to obtain out-of-sample predictions. If `at()` is used together with `zeros`, then all covariates not listed in `at()` are set to zero. If `at()` is used without `zeros`, then all covariates not listed in `at()` are set to their sample values.

`cause(`*numlist*`)` specifies the causes on which to make the predictions for and that are stored in *newvar*`_c`#. If `cause()` is not specified, then predictions are made for all causes included in the model and stored in *newvar*`_c`#.

`chrdenominator(`*varname* # [*varname* # ...]`)` and `shrdenominator(`*varname* # [*varname* # ...]`)` specify the denominator of the cause-specific hazard ratio or subdistribution hazard ratio for a specific cause. By default, all covariates not specified using this option are set to zero. See the cautionary note in `chrnumerator()` and `shrnumerator()` below. If # is set to missing (`.`), then the covariate has the values defined in the dataset.

`chrnumerator(`*varname* # [*varname* # ...]`)` and `shrnumerator(`*varname* # [*varname* # ...]`)` specify the numerator of the (time-dependent) cause-specific hazard ratio or subdistribution hazard ratio for a specific cause. By default, all covariates not specified using this option are set to zero. Setting the remaining values of the covariates to zero may not always be sensible, particularly on models other than those on the cumulative subdistribution hazard scale or when more than one variable has a time-dependent effect. If # is set to missing (`.`), then the covariate has the values defined in the dataset.

`ci` calculates a confidence interval for the requested statistic and stores the confidence limits in *newvar*`_lci` and *newvar*`_uci`.

`cif` predicts the cause-specific cumulative incidence function.

`cifdiff1(`*varname* # [*varname* # ...]`)` and `cifdiff2(`*varname* # [*varname* # ...]`)` predict the difference in cause-specific cumulative incidence functions, with the first cause-specific cumulative incidence function defined by the covariate values listed for `cifdiff1()` and the second by those listed for `cifdiff2()`. By default, covariates not specified using either option are set to zero. Setting the remaining values of the covariates to zero may not always be sensible. If # is set to missing (`.`), then *varname* has the values defined in the dataset.

Example: `cifdiff1(stage 1)` (without specifying `cifdiff2()`) computes the difference in predicted cause-specific cumulative incidence functions at `stage` = 1 compared with `stage` = 0 with all other covariates set to 0.

Example: `cifdiff1(stage 2) cifdiff2(stage 1)` computes the difference in predicted cause-specific cumulative incidence functions at `stage` = 2 compared with `stage` = 1.

Example: `cifdiff1(stage 2 age 50) cifdiff2(stage 1 age 70)` computes the difference in predicted hazard functions at `stage` = 2 and `age` = 50 compared with `stage` = 1 and `age` = 70 with all other covariates set to 0.

`cifratio` predicts the relative contribution of failing from an event to the overall cumulative incidence function. For example, if the event of interest is cancer, this is the relative contribution of dying from cancer to the total mortality. `cifratio` must be used along with the `cause()` option to specify the cause-specific cumulative incidence function on the numerator of the ratio.

`csh` predicts the cause-specific hazard function.

`cumodds` predicts the cumulative odds-of-failure function.

`cumsubhazard` predicts the cumulative subdistribution hazard function.

`cured` predicts the cause-specific cure proportion after fitting a cure model.

`subdensity` predicts the subdensity function.

`subhazard` predicts the subdistribution hazard function.

`timevar(`*varname*`)` defines the variable used as time in the predictions. The default is `timevar(_t)`. `timevar()` is useful for large datasets where, for plotting purposes, predictions are needed only for 200 observations, for example. Be cautious when using this option because predictions may be made at whatever covariate values are in the first 200 rows of data. This can be avoided using the `at()` option or the `zeros` option to define the covariate patterns for which you require the predictions.

`uncured` can be used after fitting a cure model for a specific cause. It can be used with the `subhazard` and `cif` options to base predictions for the uncured group.

`xb` predicts the linear predictor, including the spline function.

`zeros` sets all covariates to zero (baseline prediction). For example, `predict cif, cause(1) cif zeros` calculates the baseline cause-specific cumulative incidence function for cause = 1.

Subsidiary

`dxb` calculates the derivatives of the linear predictors.

`level(`#`)` specifies the confidence level, as a percentage, for confidence intervals. The default is `level(95)` or as set by `set level`.

5 Examples

5.1 Northern European Cancer Registry Data (1975–1994)

In this section, we illustrate the methods outlined in this article using the Northern European cancer registry data, which were previously used to illustrate the use of `strs` for relative survival models (Dickman and Coviello 2015). We use a subset of these data that contains observations on 4,204 patients who were between 40 and 79 years old and diagnosed with melanoma between 1975 and 1994. The covariates of interest are patient age at diagnosis and stage of cancer, which is categorized into localized or regional stage cancer at diagnosis. We excluded patients with distant stage cancer because of their very high mortality rate, leaving a few patients at risk toward the end of follow-up time. Most of these deaths are due to cancer, which means the effect of competing causes of death is small and thus less interesting practically. Survival time is measured in months since diagnosis to death because of cancer or other causes. Follow-up time is restricted to 15 years from diagnosis.

5.2 Nonparametric estimates for the cause-specific cumulative incidence function

Estimated cause-specific CIFs have been predicted using the `stcompet` command, which implements the Aalen–Johansen method (Coviello and Boggess 2004). Figure 1 shows cause-specific CIFs estimated by stage at diagnosis for death from cancer and death from other causes and shows that those with a more distant stage cancer at diagnosis have an increased risk of dying from cancer and a lower risk of dying from other causes. The sum of the cancer-specific CIF and CIF for other causes gives the overall, or all-cause probability of death.

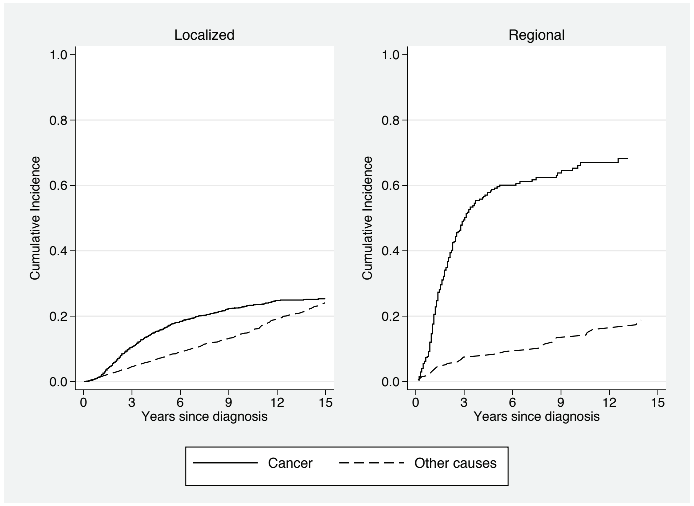


Figure 1. Predicted cause-specific cumulative incidence functions for death from cancer or death from other causes using the Aalen–Johansen method by stage at diagnosis for patients 40 to 80 years old.

5.3 Fine and Gray (1999) model

We initially fit direct regression models on the cause-specific CIF using the Fine and Gray approach, the most commonly implemented method for modeling covariate effects on the cause-specific cumulative incidence function. Fine and Gray models are fit only with stage at diagnosis as a covariate for each of the cause-specific CIFs.

We generated a new indicator variable, `status2`, to overcome a small reporting error with the `stcrreg` command when using the `exit()` option in `stset` at the time of submission. When one uses the usual censoring indicator variable in `stset` for one cause before fitting a Fine and Gray model, the number of actual competing events is under-reported because the competing events and censored events are no longer distinguished and those who die before the exit time are instead treated as censored. Although this does not directly affect the parameter estimates, the total number of overall failures reported for each cause-specific model is inconsistent. Therefore, we go on to fit Fine and Gray models using the new variable, which is generated as follows:

```
. stset surv_mm, failure(status == 1, 2) scale(12) id(id) exit(time 180)
  (output omitted)
. generate status2 = cond(_d==0,0,status)
. *Cancer
. stset surv_mm, failure(status2 == 1) scale(12) id(id) exit(time 180)
  (output omitted)
. stcrreg i.stage, compete(status2 == 2)
        failure _d:  status2 == 1
   analysis time _t:  surv_mm/12
  exit on or before:  time 180
                 id:  id

Iteration 0:   log pseudolikelihood =  -7389.917
Iteration 1:   log pseudolikelihood = -7389.4747
Iteration 2:   log pseudolikelihood = -7389.4745

Competing-risks regression                       No. of obs       =      4,204
                                                 No. of subjects  =      4,204
Failure event  : status2 == 1                    No. failed       =        937
Competing event: status2 == 2                    No. competing    =        583
                                                 No. censored     =      2,684

                                                 Wald chi2(1)     =     287.75
Log pseudolikelihood = -7389.4745                Prob > chi2      =     0.0000

                                 (Std. Err. adjusted for 4,204 clusters in id)
------------------------------------------------------------------------------
             |               Robust
          _t |        SHR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       stage |
    Regional |   4.783974   .4414379    16.96   0.000     3.992499    5.732352
------------------------------------------------------------------------------

. stset surv_mm, failure(status2 == 2) scale(12) id(id) exit(time 180)
  (output omitted)
. stcrreg i.stage, compete(status2 == 1)
        failure _d:  status2 == 2
   analysis time _t:  surv_mm/12
  exit on or before:  time 180
                 id:  id

Iteration 0:   log pseudolikelihood = -4565.6556
Iteration 1:   log pseudolikelihood = -4556.6879
Iteration 2:   log pseudolikelihood = -4556.6578
Iteration 3:   log pseudolikelihood = -4556.6578
```

```
Competing-risks regression                        No. of obs       =      4,204
                                                  No. of subjects  =      4,204
Failure event  : status2 == 2                     No. failed       =        583
Competing event: status2 == 1                     No. competing    =        937
                                                  No. censored     =      2,684

                                                  Wald chi2(1)     =       0.31
Log pseudolikelihood = -4556.6578                 Prob > chi2      =     0.5790

                                 (Std. Err. adjusted for 4,204 clusters in id)
------------------------------------------------------------------------------
             |               Robust
          _t |        SHR   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       stage |
    Regional |   .9080851   .1577827    -0.55   0.579     .6459927    1.276514
------------------------------------------------------------------------------
```

The subdistribution hazard ratio for cancer gives the association between stage at diagnosis and the cancer-specific CIF. A subdistribution hazard ratio of 4.78 indicates that those with a more severe stage at diagnosis are associated with an increased risk of dying from cancer. However, because of the awkward definition in the risk set, it is difficult to make inferences on quantitative effects. Although insignificant, the subdistribution hazard ratio from the Fine and Gray model for other causes shows that those with a more severe stage at diagnosis are associated with a decreased risk of dying from other causes. This is because patients at an earlier stage at diagnosis are healthier and therefore more likely to live longer and die from other causes before their cancer. On the other hand, patients at a later stage are unlikely to live as long and die from other causes.

After fitting each cause-specific Fine and Gray model, we can use `stcurve` to predict and store the cause-specific CIFs.

5.4 Log-cumulative subdistribution hazard models

Using the full likelihood in (2), we can fit direct flexible parametric regression models for the cause-specific CIF. Rather than fitting a model to each cause-specific CIF separately, we can instead model all cause-specific CIFs simultaneously. This is shown below with the assumption of proportionality for all causes:

```
. stset surv_mm, failure(status==1, 2) scale(12) id(id) noshow exit(time 180)
  (output omitted)
. stpm2cr [cancer: stage2, scale(hazard) df(5)]
> [other: stage2, scale(hazard) df(5)],
> events(status) cause(1 2) cens(0) eform nolog
  (output omitted)
Log likelihood = -4901.0253                    Number of obs     =      4,204

------------------------------------------------------------------------------
             |     exp(b)   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
cancer       |
      stage2 |   4.673522   .3973545    18.14   0.000     3.956153    5.520973
    _rcs_c1_1 |   2.371601   .0642335    31.88   0.000     2.248989    2.500897
    _rcs_c1_2 |    1.40679   .0445023    10.79   0.000     1.322216    1.496774
    _rcs_c1_3 |   1.061522   .0237518     2.67   0.008     1.015975    1.109111
    _rcs_c1_4 |   .9889806   .0103402    -1.06   0.289     .9689204    1.009456
    _rcs_c1_5 |   1.002836    .005948     0.48   0.633     .9912455    1.014562
       _cons |   .1390518   .0053603   -51.18   0.000     .1289329    .1499648
-------------+----------------------------------------------------------------
other        |
      stage2 |   .6867003    .115223    -2.24   0.025     .4942449    .9540964
    _rcs_c2_1 |   2.564841   .0949475    25.44   0.000     2.385338    2.757852
    _rcs_c2_2 |   1.058082   .0298144     2.00   0.045     1.001231    1.118161
    _rcs_c2_3 |   .9541731   .0196412    -2.28   0.023     .9164434    .9934562
    _rcs_c2_4 |   .9843678   .0125716    -1.23   0.217     .9600337    1.009319
    _rcs_c2_5 |   .9917352   .0082375    -1.00   0.318     .9757208    1.008012
       _cons |   .0800586   .0040859   -49.47   0.000     .0724379     .088481
------------------------------------------------------------------------------
```

An equation is specified for each cause within the square brackets along with their respective options. These are similar to those used for `stpm2` where `df(5)` implies four internal knots at default locations. The estimated subdistribution hazard ratios are displayed for each cause and their 95% confidence intervals. From the subdistribution hazard ratios for both causes, we can infer that patients with regional stage cancer at diagnosis have an increased risk of dying from cancer and a decreased risk of dying from other causes compared with those with localized stage cancer at diagnosis. The advantage of using the parametric approach is that it is easy to obtain other useful predictions to aid interpretation, because, as mentioned previously, it is difficult to interpret the subdistribution hazard ratios in terms of quantitative effects. The following code obtains the cause-specific CIFs, subdistribution hazard functions for each cause, and cause-specific hazard functions. Confidence intervals are obtained using the `ci` option.

```
. range temptime 0 15 1000
(3,204 missing values generated)
. predict cif1, cif at(stage1 1 stage2 0) timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict cif2, cif at(stage1 0 stage2 1) timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict sdh1, subhazard at(stage1 1 stage2 0) timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict sdh2, subhazard at(stage1 0 stage2 1) timevar(temptime)
Calculating predictions for the following causes: 1 2
```

```
. predict csh1, csh at(stage1 1 stage2 0) timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict csh2, csh at(stage1 0 stage2 1) timevar(temptime)
Calculating predictions for the following causes: 1 2
```

The top row in figure 2 plots the predicted subdistribution hazard function for each cause, and the bottom row illustrates the predicted cause-specific hazard function by stage at diagnosis. The subdistribution hazard gives the association on the effect of stage at diagnosis on risk, and the cause-specific hazard is the association on the effect of stage at diagnosis on the hazard rate.

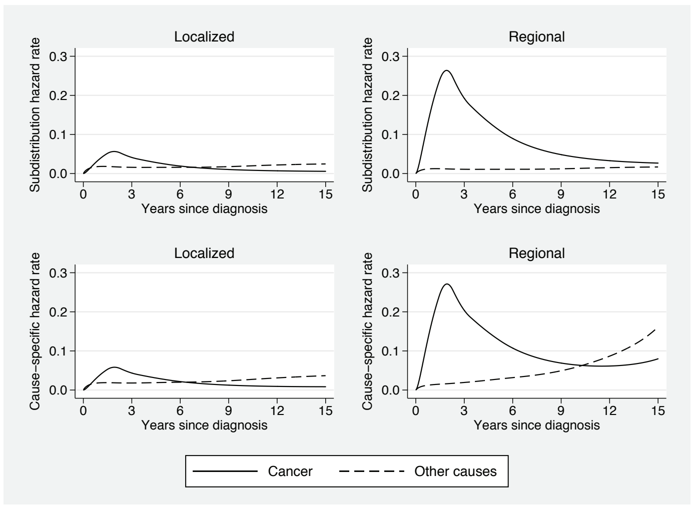


Figure 2. Subdistribution hazards predicted for each cause and cause-specific hazard predictions by stage at diagnosis for patients 40 to 80 years old from a log cumulative-proportional subdistribution hazard model for melanoma data.

Figure 3 compares the cause-specific CIFs obtained from the Fine and Gray models for each cause fit in section 5.3 with those obtained from the log cumulative-proportional subdistribution hazards model and shows sensible agreement between the two (see Mozumder, Rutherford, and Lambert [2016] for more details on the disagreement in the cause-specific CIF for death from other causes).

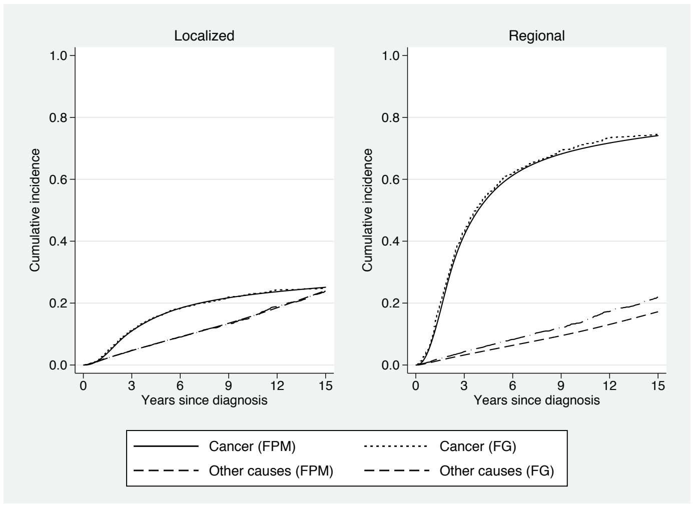


Figure 3. A comparison of cause-specific cumulative incidence functions for death from cancer or death from other causes predicted simultaneously from a log cumulative-subdistribution hazard model and from separate Fine and Gray models for each cause by stage at diagnosis for patients 40 to 80 years old.

In figure 4, the Aalen–Johansen estimates are compared with the cause-specific CIFs obtained from the log cumulative-proportional subdistribution hazard model. The estimates are reasonably similar. However, we can achieve a better fit by relaxing the assumption of proportionality through including time-dependent effects using restricted cubic splines.

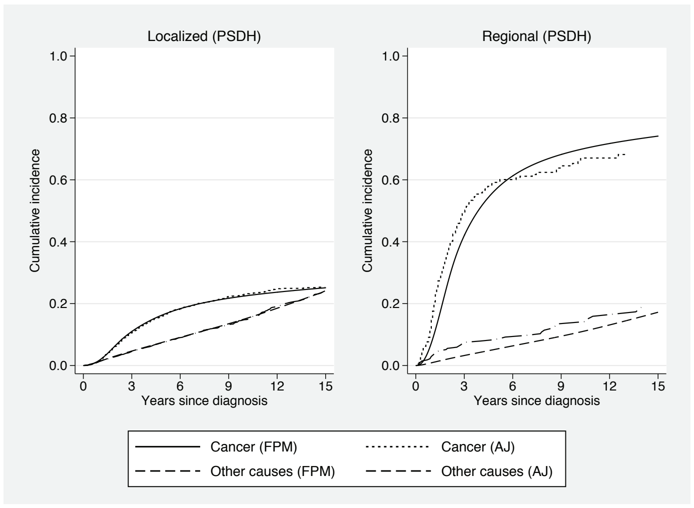


Figure 4. A comparison of cause-specific cumulative incidence functions for death from cancer or death from other causes predicted simultaneously from a log cumulative-subdistribution hazard model assuming proportionality and using the Aalen–Johansen empirical estimates for each cause by stage at diagnosis for patients 40 to 80 years old.

5.5 Time-dependent effects

Time-dependent effects can be easily incorporated by specifying the `dftvc()` and `tvc()` equation-specific options as shown in the following code:

```
. stpm2cr [cancer: stage2, scale(hazard) df(5) tvc(stage2) dftvc(3)]
> [other: stage2, scale(hazard) df(5) tvc(stage2) dftvc(3)],
> events(status) cause(1 2) cens(0) eform nolog
  (output omitted)
Log likelihood = -4877.5917                    Number of obs     =      4,204

------------------------------------------------------------------------------
              |     exp(b)   Std. Err.      z    P>|z|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
cancer        |
       stage2 |   5.225629   .4543429    19.02   0.000     4.406875    6.196499
     _rcs_c1_1 |   2.570244    .089602    27.08   0.000     2.400493       2.752
     _rcs_c1_2 |   1.440213   .0605618     8.68   0.000     1.326274     1.56394
     _rcs_c1_3 |   1.076737   .0280174     2.84   0.004     1.023201    1.133074
     _rcs_c1_4 |   .9907888   .0106845    -0.86   0.391     .9700674    1.011953
     _rcs_c1_5 |   .9997375   .0058897    -0.04   0.964     .9882603    1.011348
_rcs_stag~1_1 |   .7353858   .0413674    -5.46   0.000      .658617    .8211029
_rcs_stag~1_2 |   .9750149   .0568817    -0.43   0.664     .8696665    1.093125
_rcs_stag~1_3 |   .9458115   .0303569    -1.74   0.083     .8881458    1.007221
        _cons |   .1328929   .0053238   -50.38   0.000     .1228576     .143748
--------------+---------------------------------------------------------------
other         |
       stage2 |    1.18831   .2267027     0.90   0.366     .8175976    1.727109
     _rcs_c2_1 |   2.658485   .1059802    24.53   0.000     2.458675    2.874533
     _rcs_c2_2 |   1.062388   .0328778     1.96   0.051      .999864    1.128822
     _rcs_c2_3 |   .9584928   .0206057    -1.97   0.049     .9189454    .9997422
     _rcs_c2_4 |   .9841378   .0124521    -1.26   0.206     .9600322    1.008849
     _rcs_c2_5 |   .9926364   .0081918    -0.90   0.370     .9767099    1.008823
_rcs_stag~2_1 |     .68066   .0697333    -3.75   0.000     .5568331    .8320231
_rcs_stag~2_2 |   1.007956   .0739275     0.11   0.914     .8729933    1.163783
_rcs_stag~2_3 |   .9515855   .0501094    -0.94   0.346     .8582712    1.055045
        _cons |   .0775996   .0040571   -48.89   0.000     .0700417    .0859732
------------------------------------------------------------------------------
```

The `tvc(stage2)` and `dftvc(3)` options state that the `stage2` variable is to be time dependent using restricted cubic splines with two internal knots (that is, three degrees of freedom). Overall, 10 parameters are estimated for each cause in the model. For example, for cancer, there are five derived variables for the baseline log cumulative-subdistribution hazard (`_rcs_c1_1`-`_rcs_c1_5`) and three derived splines for the time-dependent effect `stage2` (`_rcs_stage2_c1_1`-`_rcs_stage2_c1_3`).

In a time-dependent model, parameter estimates become more complex and less useful when interpreted on their own. Instead, it is better to obtain predictions between groups for specific covariate patterns as relative or absolute differences over time using `predict`. Note that the coding is the same to generate the same predictions:

```
. range temptime 0 15 1000
(3,204 missing values generated)
. predict cif_tvc1, cif at(stage1 1 stage2 0) ci timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict cif_tvc2, cif at(stage1 0 stage2 1) ci timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict cifdiff, cifdiff1(stage1 0 stage2 1) cifdiff2(stage1 1 stage2 0) ci
> timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict shr, shrn(stage1 0 stage2 1) shrd(stage1 1 stage2 0) ci
> timevar(temptime)
Calculating predictions for the following causes: 1 2
. predict chr, chrn(stage1 0 stage2 1) chrd(stage1 1 stage2 0) ci
> timevar(temptime)
Calculating predictions for the following causes: 1 2
```

Figure 5 now shows a better fit of the model-estimated cause-specific CIFs, particularly with regional stage patients, compared with the nonparametric Aalen–Johansen estimates with very good agreement.

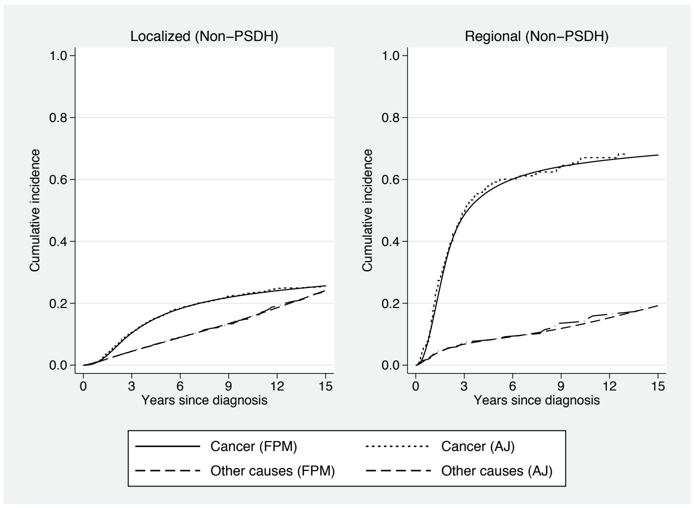


Figure 5. A comparison of cause-specific cumulative incidence functions for death from cancer or death from other causes predicted simultaneously from a log cumulative-nonproportional subdistribution hazard model and using the Aalen–Johansen empirical estimates for each cause by stage at diagnosis for patients 40 to 80 years old.

We can obtain absolute differences with 95% confidence intervals between the regional and localized stage groups over time for each cause-specific CIF. Differences are calculated using the `cifdiff1()` and `cifdiff2()` options. The obtained predictions are illustrated in figure 6, which shows us that those with a more severe stage of cancer at diagnosis are more likely to die from cancer. The difference is smaller for other causes for the first six years since diagnosis. In the later years, the cause-specific CIF for other causes is larger for localized stage patients.

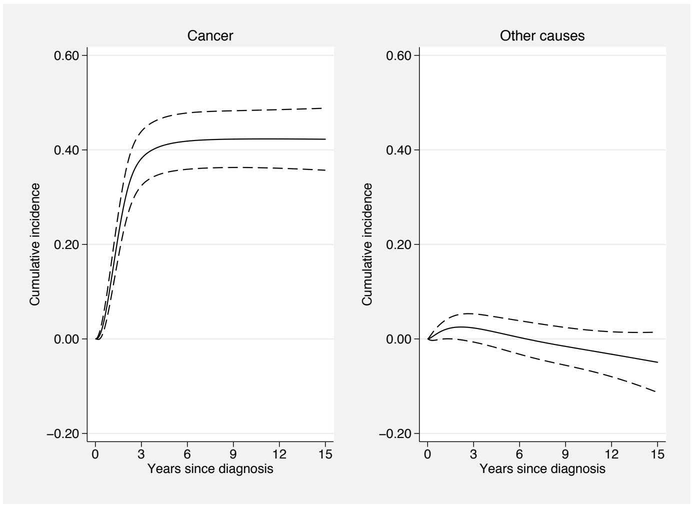


Figure 6. Predicted absolute differences (Regional − Localized) in cause-specific cumulative incidence functions with 95% confidence intervals from a log cumulative-nonproportional subdistribution hazard model.

Time-dependent subdistribution and cause-specific hazard ratios are obtained using the options `shrnumerator()` and `shrdenominator()`, and `chrnumerator()` and `chrdenominator()`, respectively. Using these options, we can obtain ratios for any two covariate patterns. Figure 7 shows the time-dependent subdistribution and cause-specific hazard ratios and compares regional stage patients with localized stage patients at diagnosis. At the start of follow-up, for both cancer-specific hazard ratios, regional stage patients have a mortality rate 17 times that of localized stage patients that decreases over follow-up time. The mortality rate of other causes on both scales for regional stage patients at the start of follow-up time is approximately 4.5 times that of localized stage patients. Beyond two years since diagnosis, the subdistribution hazard rate of other causes for regional stage patients is lower than the localized stage patients because the ratio is less than 1. This is expected because those at a later stage will die earlier from the cancer before they die from other causes. The cause-specific hazard ratios give us the association of stage at diagnosis on the rate and show a different effect on death from other causes, because patients at a later stage tend to be more sick and generally are at a higher risk of dying. This translates to a positive association between more distant stage patients and mortality rate for other causes.

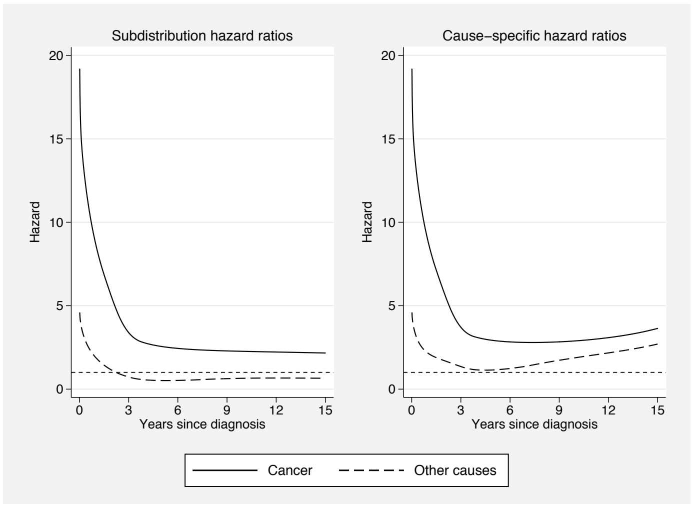


Figure 7. Predicted subdistribution and cause-specific hazard ratios for each cause from a log cumulative-nonproportional subdistribution hazard model. Ratios compare regional stage with localized stage patients at diagnosis. Dotted line is a reference line when the rate is equal to 1, that is, no difference.

5.6 Cure model

Cure models for any cause can be fit by adding the equation option `cure`. However, we highly recommend that this be done only for one cause, usually the event of interest. Predictions can be made after fitting a cure model with `predict` using the `cured` and `uncured` options. Specifying the `cured` option will calculate the cure proportion for the cause that `cured` was specified for and a variable with the suffix `_btd` that partitions those that are still alive into two groups: patients bound to die from cancer and not bound to die from cancer. The code for fitting a cure model and predictions is shown below:

```
. stpm2cr [cancer: , scale(hazard) df(5) cure]
> [other: , scale(hazard) df(5)],
> events(status) cause(1 2) cens(0) eform nolog
  (output omitted)
Log likelihood = -1742.7601                     Number of obs     =      1,692

------------------------------------------------------------------------------
             |     exp(b)   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
cancer       |
   _rcs_c1_1 |   2.168448   .0851865    19.70   0.000     2.007752    2.342007
   _rcs_c1_2 |   .9134977   .0245224    -3.37   0.001     .8666772    .9628475
   _rcs_c1_3 |   .9989706   .0182824    -0.06   0.955     .9637729    1.035454
   _rcs_c1_4 |   .9775022   .0134488    -1.65   0.098     .9514954     1.00422
   _rcs_c1_5 |          1  (omitted)
       _cons |    .348136   .0181445   -20.25   0.000     .3143294    .3855784
-------------+----------------------------------------------------------------
other        |
   _rcs_c2_1 |   2.645041   .3083898     8.34   0.000     2.104696    3.324111
   _rcs_c2_2 |   .9981758   .0919501    -0.02   0.984     .8332895    1.195689
   _rcs_c2_3 |   .9368575   .0517331    -1.18   0.238     .8407566    1.043943
   _rcs_c2_4 |   1.013603    .037129     0.37   0.712     .9433826    1.089051
   _rcs_c2_5 |   .9643029   .0211338    -1.66   0.097     .9237584    1.006627
       _cons |   .0220712   .0032665   -25.77   0.000     .0165139    .0294985
------------------------------------------------------------------------------

. range temptime 0 15 1000
(692 missing values generated)

. predict cif, cif timevar(temptime)
Calculating predictions for the following causes: 1 2

. predict cure, cured timevar(temptime)
Calculating predictions for the following causes: 1 2

. generate cif_tot = cif_c1 + cif_c2
(693 missing values generated)
```

In section 2.8, we showed that to fit cure models, we constrained the last knot to be zero to force a plateau. This is shown in the output above, where the parameter for `_rcs_c1_5` is equal to one. Analysis is restricted to localized stage patients 40 to 54 years old, where a cure is found to be reasonable. To check this, we note that the plot to the left in figure 8 compares the estimated cancer-specific CIF from the model with the Aalen–Johansen estimate and shows extremely good agreement with the cure proportion estimated at approximately 30% after 12 years since diagnosis where the cancer-specific CIF plateaus. On the right-hand side of figure 8, the cause-specific CIFs are stacked, and the dashed line is the partitioning of alive patients that are bound to or not bound to die into two groups. This estimate is provided as part of the `cured` option with the suffix `_btd`. Eloranta et al. (2014) introduce this quantity to aid better risk communication, and it is calculated as follows,

$$P_{\text{alive,can}}(t) = \pi_c - F_1(t)$$
$$P_{\text{alive,oth}}(t) = 1 - F_2(t) - \cdots - F_K(t) - \pi_c$$

where π_c is the proportion of those bound to die from cancer on which a cure is assumed. For $k = 1$, $P_{\text{alive,can}}(t)$ represents patients who will ultimately die from their cancer, and $P_{\text{alive,oth}}(t)$ represents those who will die from competing causes where $k = 2, \ldots, K$.

In our example, from the stacked probabilities in figure 8, at 6 years after diagnosis, approximately 25% have died, 6% are alive yet bound to die from cancer, and 69% are alive and not bound to die from cancer.

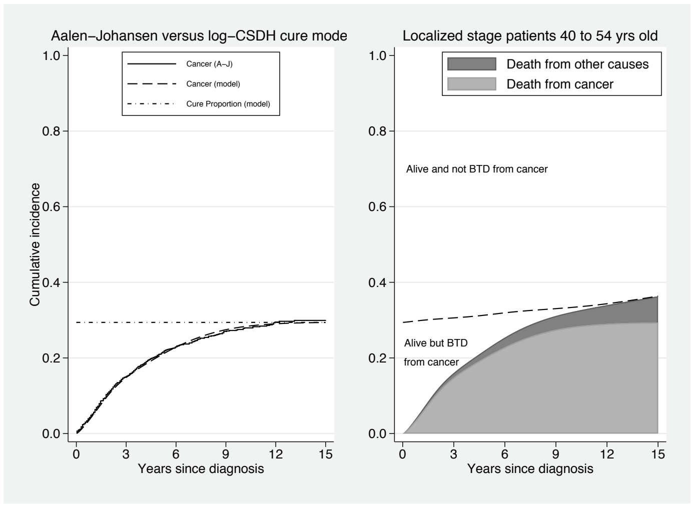


Figure 8. Left: Comparison of predicted cancer-specific CIFs obtained from the log cumulative-subdistribution hazard cure model and using the Aalen–Johansen method for localized stage patients 40 to 54 years old. Right: Stacked cause-specific CIFs obtained from a log cumulative-subdistribution hazard cure model. Dashed-line partitions living patients into those bound to die from cancer and not bound to die from cancer.

5.7 Conclusions

Competing-risks models are being widely applied in research, and fitting regression models on the subdistribution hazard scale is encouraged for researchers to make inferences on prognosis and understand the association of a covariate on risk. Analysis from within the flexible parametric modeling framework using the direct likelihood approach for the cause-specific CIF has several advantages. For example, the method saves computational time because numerical integration is not required to model the baseline log cumulative-subdistribution hazard function. All causes are modeled simultaneously, so there is no need to fit separate models for each cause. This is implemented in the new `stpm2cr` command, an adaptation of the `stpm2` command. Other useful predictions can be obtained using `predict` after fitting a model using `stpm2cr`. This complements flexible parametric regression models for competing risks on the cause-specific hazard scale and allows researchers to gain a more complete understanding on the impact of the event of interest on outcome. However, a well-known problem of direct regression models for the cause-specific CIF is that the sum of all probabilities may exceed 1 for certain covariate patterns. This is particularly problematic in the oldest age groups, where patients are at a higher risk of dying from competing events, which leads to very high overall probability of death. This is also the case in our approach, and it is sometimes avoided if models are not misspecified, for example, by adjusting for all appropriate covariates with any potential interactions and by including time-dependent effects. In some situations, models may fail to converge when specified correctly, but this will depend on the use of better initial values for the optimizer so that it is not searching in the wrong direction. Therefore, future work may involve implementing an appropriate constraint on the models to avoid issues in convergence.

6 References

Andersen, P. K., R. B. Geskus, T. de Witte, and H. Putter. 2012. Competing risks in epidemiology: Possibilities and pitfalls. *International Journal of Epidemiology* 41: 861–870.

Andersson, T. M.-L., P. W. Dickman, S. Eloranta, and P. C. Lambert. 2011. Estimating and modelling cure in population-based cancer studies within the framework of flexible parametric survival models. *BMC Medical Research Methodology* 11: 96.

Coviello, V., and M. Boggess. 2004. Cumulative incidence estimation in the presence of competing risks. *Stata Journal* 4: 103–112.

Dickman, P. W., and E. Coviello. 2015. Estimating and modeling relative survival. *Stata Journal* 15: 186–215.

Durrleman, S., and R. Simon. 1989. Flexible regression models with cubic splines. *Statistics in Medicine* 8: 551–561.

Eloranta, S., P. C. Lambert, T. M.-L. Andersson, M. Björkholm, and P. W. Dickman. 2014. The application of cure models in the presence of competing risks: A tool for improved risk communication in population-based cancer patient survival. *Epidemiology* 25: 742–748.

Fine, J. P., and R. J. Gray. 1999. A proportional hazards model for the subdistribution of a competing risk. *Journal of the American Statistical Association* 94: 496–509.

Gerds, T. A., T. H. Scheike, and P. K. Andersen. 2012. Absolute risk regression for competing risks: Interpretation, link functions, and prediction. *Statistics in Medicine* 31: 3921–3930.

Gray, R. J. 1988. A class of k-sample tests for comparing the cumulative incidence of a competing risk. *Annals of Statistics* 16: 1141–1154.

Hinchliffe, S. R., and P. C. Lambert. 2013. Flexible parametric modelling of cause-specific hazards to estimate cumulative incidence functions. *BMC Medical Research Methodology* 13: 13.

Jeong, J.-H., and J. Fine. 2006. Direct parametric inference for the cumulative incidence function. *Journal of the Royal Statistical Society, Series C* 55: 187–200.

Koller, M. T., H. Raatz, E. W. Steyerberg, and M. Wolbers. 2012. Competing risks and the clinical community: Irrelevance or ignorance? *Statistics in Medicine* 31: 1089–1097.

Lambert, P. C. Forthcoming. The estimation and modeling of cause-specific cumulative incidence functions using time-dependent weights. *Stata Journal*.

Lambert, P. C., L. Holmberg, F. Sandin, F. Bray, K. M. Linklater, A. Purushotham, D. Robinson, and H. Møller. 2011. Quantifying differences in breast cancer survival between England and Norway. *Cancer Epidemiology* 35: 526–533.

Lambert, P. C., and P. Royston. 2009. Further development of flexible parametric models for survival analysis. *Stata Journal* 9: 265–290.

Lambert, P. C., S. R. Wilkes, and M. J. Crowther. Forthcoming. Flexible parametric modelling of the cause-specific cumulative incidence function. *Statistics in Medicine*.

Latouche, A., V. Boisson, S. Chevret, and R. Porcher. 2007. Misspecified regression model for the subdistribution hazard of a competing risk. *Statistics in medicine* 26: 965–974.

Lau, B., S. R. Cole, and S. J. Gange. 2009. Competing risk regression models for epidemiologic data. *American Journal of Epidemiology* 170: 244–256.

Mozumder, S. I., M. J. Rutherford, and P. C. Lambert. 2016. Direct likelihood inference on the cause-specific cumulative incidence function: A flexible parametric regression modelling approach.

Noordzij, M., K. Leffondré, K. J. van Stralen, C. Zoccali, F. W. Dekker, and K. J. Jager. 2013. When do we need competing risks methods for survival analysis in nephrology? *Nephrology Dialysis Transplantation* 28: 2670–2677.

Putter, H., M. Fiocco, and R. B. Geskus. 2007. Tutorial in biostatistics: Competing risks and multi-state models. *Statistics in Medicine* 26: 2389–2430.

Royston, P., and M. K. B. Parmar. 2002. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in Medicine* 21: 2175–2197.

Sapir-Pichhadze, R., M. Pintilie, K. J. Tinckam, A. Laupacis, A. G. Logan, J. Beyene, and S. J. Kim. 2016. Survival analysis in the presence of competing risks: The example of waitlisted kidney transplant candidates. *American Journal of Transplantation* 16: 1958–1966.

Wolbers, M., M. T. Koller, V. S. Stel, B. Schaer, K. J. Jager, K. Leffondré, and G. Heinze. 2014. Competing risks analyses: Objectives and approaches. *European Heart Journal* 35: 2936–2941.

About the authors

Sarwar Islam Mozumder is a PhD student at the University of Leicester, UK. His PhD focuses on further development of flexible parametric modeling methods in competing risks and risk communication of cancer survival statistics.

Mark Rutherford is a lecturer of biostatistics at the University of Leicester, UK. He has a keen interest in applying survival methods in Stata, particularly for population-based cancer data.

Paul Lambert is a professor of biostatistics at the University of Leicester, UK. He also works part-time at the Department of Medical Epidemiology and Biostatistics, Karolinska Institutet, Stockholm, Sweden. His main interest is in the development and application of methods in population-based cancer research.

The Stata Journal (2017)
17, Number 2, pp. 490–502

Rate decomposition for aggregate data using Das Gupta's method

Jinjing Li
National Centre for Social and Economic Modelling
Institute for Governance and Policy Analysis
University of Canberra
Bruce, Australia
jinjing.li@canberra.edu.au

Abstract. Social, behavioral, and health scientists frequently decompose changes or differences in outcome variables into components of change and assess their relative importance. Many Stata commands facilitate this exercise using unit-level data, notably by applying the Blinder–Oaxaca approach. However, none of the comparable user-written commands decompose changes or differences in aggregate data despite their availability and the widespread use of corresponding decomposition techniques. In this article, I present the user-written command `rdecompose`, which decomposes aggregate or cross-classified data based on Das Gupta's (1993, *Standardization and Decomposition of Rates: A User's Manual, Volume 1*) approach, and demonstrate its application in multiple settings. This command extends the original method by allowing multiple factors and flexible functional specifications.

Keywords: st0483, rdecompose, decomposition, cross-classified, Das Gupta method

1 Introduction

Numeric values such as rates, means, percentages, and proportions are instrumental in measuring social, economic, health, and demographic outcomes. Researchers often study measures such as birth rates, prevalences of diseases, or income inequality and analyze differences between populations or changes over time; such factors reflect differences in relevant population characteristics that may directly or indirectly influence outcomes. Demographers, economists, and public health scientists traditionally apply standardization and decomposition techniques to distinguish real "rate" differences from the effects of compositional factors on measured outcomes. These techniques are used throughout the social sciences to determine why rates differ among populations (Guo et al. 2012; Wang et al. 2000; Yamaguchi 2011).

The literature uses decomposition techniques that fall broadly into two categories based on data requirements. The first category uses unit record data along with a multivariate regression-based technique known as the Blinder–Oaxaca approach (Blinder 1973). This approach has multiple Stata implementations, including the linear version from Jann (2008) and nonlinear extensions from Sinning, Hahn, and Bauer (2008) and Powers, Yoshioka, and Yun (2011). This approach relies on the availability of individual-level data. The second approach is designed for cross-classified data or contingency tables and often uses algebraic relationships rather than econometric estimations (Chevan and Sutherland 2009; Das Gupta 1993).

Although some Stata commands apply unit record-based Blinder−Oaxaca decomposition, no comparable user-written commands implement existing decomposition techniques for aggregate data,[1] despite wide availability of aggregate data and use of corresponding aggregate data-based decomposition methods in the literature. In this article, I introduce a new user-written command, `rdecompose`, for a variant of such methods known as Das Gupta's reformulation and demonstrate potential applications from a range of settings, including demography, public health, and economics.

Das Gupta's method for cross-classified data is based on incremental methodological developments in standardization and decomposition techniques that have occurred over several decades. Wolfbein and Jaffe (1946) were among the first to demonstrate the importance of decomposition by applying a double standardization technique, but Kitagawa (1955) developed a formal procedure for decomposing cross-classified data. Her work led to the simultaneous identification of separate but additive composition and rate components that summed to rate differences. In a series of articles, Das Gupta took a symmetric approach to the interaction component and developed functional relationships that allow deterministic allocation of the interaction components among cross-classified variables. This approach led to integration of the interaction between component effects into the additive main effects and facilitated interpretation of results. Das Gupta's approach also imposed few constraints on the nature of the variable and its distribution or on the specification of relationships for the outcome of interest. Thus the method can be applied flexibly to most cross-classified aggregate data.

2 Method

2.1 Standard rate decomposition with multiplicative factors

Demographers, health researchers, and social scientists must sometimes interpret differences between two crude rates of the same or comparable population. For instance, researchers may want to understand what drives differences in rates of death today versus 20 years ago. Differences between death rates may be decomposed into multiple confounding factors such as differences in age-specific death rates and age structures. Kitagawa's (1955) approach initially decomposed differences in job mobility rates between two cities into migrant status and time spent in the labor force. Das Gupta (1978, 1991, 1993) generalized the method, and Chevan and Sutherland (2009) made improvements so that the approach can be applied to any type and number of factors.[2] Unlike other decomposition methods that allow nonlinear specification of rates, Das Gupta's method yields stable results independent of the order in which factors are introduced and needs no special treatment for interaction terms.

1. Researchers may be able to decompose crude rate data via existing individual-record based decomposition commands such as `mvdcmp` (Powers, Yoshioka, and Yun 2011) in some cases. Doing so may involve data transformation and programming.
2. For a review of rate standardization and comparison methods, see Keiding and Clayton (2014).

Suppose rate r can be expressed by k multiplicative factors $x_1 \dots x_k$. Rate r can be a death rate, fertility rate, or any aggregate measures of interest. $x_1 \dots x_k$ are k factors (for example, age structure in the case of death rates) driving changes in the rate.

$$r(x_1 \dots x_k) = \sum_{i=1}^{k} x_i$$

If superscript a is used to denote the first population and superscript b the second population, the (unstandardized) contribution of two factors $C(x_1)$ and $C(x_2)$ to the difference of r_a and r_b in the case of two factors can be expressed mathematically as below, following Das Gupta (1991, 1993):

$$\begin{cases} C(x_1) & = \frac{1}{2}(x_2^a + x_2^b)(x_1^a - x_1^b) \\ C(x_2) & = \frac{1}{2}(x_1^a + x_1^b)(x_2^a - x_2^b) \end{cases}$$

Intuitively, the contribution of a factor lies in its conditional effect on the mean values of other factors. The relative contribution of x_1 is therefore $C(x_1)/\{C(x_1) + C(x_2)\}$, and the relative contribution of x_2 is $C(x_2)/\{C(x_1) + C(x_2)\}$. This approach is generally straightforward when there are few factors. However, calculations become cumbersome as k increases because of the need to compute all possible counterfactuals (2^k) and aggregate the result. Mathematically, the contribution of the ith factor to the rate is

$$C(x_i) = \sum_{j=1}^{k-1} \frac{R(j-1,i)}{k\binom{k-1}{j-1}}(x_i^a - x_i^b)$$

where $R(j,i)$ is the sum of all possible values of the product of $k-1$ factors (excluding x_i), from which j factors are from population a and all other factors are from population b. The number of possible values can be large when there are many factors because the number of permutations increases faster than k.

One advantage of this type of decomposition is the consideration of all possible replacements of the elementary rates of the first population with the corresponding rates of the second, thus avoiding path dependency. Das Gupta's method essentially assigns a weight to each possible path where the importance of the specific factor gradually fades when further changes to other factors are introduced in the counterfactual. This assumption differs from Shapley's decomposition approach, where all paths are treated equally (Sastre and Trannoy 2002).

The result of Das Gupta's decomposition can be presented intuitively, where observed rate differences are decomposed into different component effects with the relative contribution summing to 100%. This method can be applied to a wide range of research, such as differences in sociodemographic attributes, income inequality, and disparities in health outcome.

2.2 Generalized rate decomposition

In some cases, the crude rate cannot be represented by a series of multiplicative factors. Instead, more complex computations might be involved. In the case of death rates, for instance, researchers might need to use only summative and multiplicative operators to control for deaths attributable to different potential mechanisms and age structure. In other cases, rate functions can be more complicated. Consider, for example, $r = x_1 e^{x_2} \ln(x_3 + x_4)$. In such cases, the calculation of r can be presented in a more generic form,

$$r(x_1 \dots x_k) = f(x_1, \dots, x_k)$$

where $f(\cdot)$ is the rate function instead of a simple multiplicative equation as before. Although the principle remains the same, the calculation of the rate must be replaced by a more generic function. This necessity often increases technical complexity in practical implementation, especially when k is large. The `rdecompose` command assists researchers with such issues.

3 The rdecompose command

The `rdecompose` command implements Das Gupta's style decomposition where the aggregate rate r is calculated based on k factors and aggregated over s in the following manner:

$$r = \sum_s f(x_1 \dots x_k)$$

3.1 Syntax

The syntax of `rdecompose` is

`rdecompose` *varlist* [*if*] `,` `group(`*varname*`)` [`sum(`*varname*`)` `detail` `reverse` `function(`*string*`)` `transform(`*varname*`)` `multi` `baseline(`*#*`)`]

`rdecompose` should be immediately followed by the names of the variables (factors) that contribute to the rates. The population group indicator also must be included in the `group()` option. An `if` condition can be used in combination with `rdecompose` if required.

3.2 Options

`group(`*varname*`)` specifies the group indicator of the two populations that will be compared. *varname* can be in numeric or string format. `group()` is required.

`sum(`*varlist*`)` indicates the population rate is an aggregated value summation over each distinct value of this variable or variables (for example, age or location).

`detail` gives more detailed output when the `sum()` option is specified.

`reverse` reverses the order of the two compared groups.

`function(`*string*`)` specifies the function form of the rate. For example, the user may specify `ln(`*factor1*+*factor2*`)*exp(`*factor3*`)` to be used as a function. Most Stata-supported functions can be used here. `rdecompose` assumes multiplicative operations if a function is not specified. An error message will appear if the specified function is invalid or cannot be evaluated.

`transform(`*varname*`)` converts absolute numbers into proportions within the population.

`multi` indicates there are more than two populations in the `group()` option. Specifying this option results in multiple comparisons against the baseline population, which can be specified in the `baseline()` option.

`baseline(`#`)` specifies the value of the group variable for the baseline population.

3.3 Output and stored results

The typical output of `rdecompose` resembles what is presented in output 1. The output reports the names of variables and rates corresponding to the two compared populations, the functional form assumed in the computation, and a table listing factors and their contributions. `rdecompose` also standardizes the total contribution into 100% for the convenience of interpretation.

Output 1: An example of the rdecompose command

```
. rdecompose size rate, sum(agegroup) group(pop)
Decomposition between pop == 1 (9800.09)
                  and pop == 2 (55800.13)
Func Form = sum(agegroup)size*rate
------------------------------------------------------------------------
                Component         Absolute Difference    Proportion (%)
------------------------------------------------------------------------
                     size                       15839          34.43
                     rate                       30161          65.57
------------------------------------------------------------------------
                  Overall                       46000         100.00
------------------------------------------------------------------------
```

Besides table output, `rdecompose` returns some estimation results as scalars, macros, and matrices. The most notable include

Scalars

`e(rate1)`	contains the rate calculated for the first group
`e(rate2)`	contains the rate calculated for the second group
`e(diff)`	shows total differences between two groups

Macros

`e(basegroup_value)`	stores the value of the group variable for the baseline population

Matrices

`e(b)`	contains total contributions for each factor

4 Examples

This section demonstrates the use of `rdecompose` for a range of decomposable factors and data types. The first two examples are mostly for validation because they replicate known examples from Bongaarts (1978) and Clogg and Eliason (1988). These examples draw data from the discipline of demography and decompose changes in parity progression and total fertility rates (TFRs). Both examples are cited and discussed by Das Gupta (1993). The third example uses health expenditure data from China from 1993 to 2012 and attempts to decompose growth in health expenditures into five major factors. The fourth example shows `rdecompose`'s use in economics through an exercise of income inequality decomposition. The final example demonstrates how standard errors of decomposition results can be derived with bootstrapping.

4.1 Example 1: Explaining changes in fertility rates over time

Table 1 presents the TFR in South Korea from 1960–1970. It also shows data on proportions of married women (C_m), women not using contraception (C_c), prevalence of abortion (C_a), lactational infecundability (C_i), and total fecundity (TF) rate. The last is the level of natural fertility that the population would have attained if all women had married at an early age, practiced neither contraception nor abortion, and did not have long gestation periods during lactation. In demographic literature, these variables are collectively known as proximate determinants of fertility. TFR is expressed as a product of these five factors (that is, $\text{TFR} = \text{TF} \times C_m \times C_c \times C_a \times C_i$).

Table 1. TFRs and proximate determinants of fertility

	South Korea (1960–1970)	
Fertility measures	1960 (population 1)	1970 (population 2)
TF rate	16.158	16.573
Index of proportion married (C_m)	0.72	0.58
Index of noncontraception (C_c)	0.97	0.76
Index of induced abortion (C_a)	0.97	0.84
Index of lactational infecundability (C_i)	0.56	0.66
TFR	6.13	4.05

Source: Bongaarts (1978)

As table 1 shows, between 1960 and 1970, the TFR in South Korea declined 2.08 points from 6.13 in 1960 to 4.05 in 1970. `rdecompose` can be applied as follows to determine relative contributions of each proximate determinant factor to changes in TFR observed:

Output 2: Stata code and command output of example 1

```
. use example1-bongaarts
. rdecompose marriage noncontracept abortion lactation fecundity, group(year)
Decomposition between year == 1960 (6.13)
                  and year == 1970 (4.05)
Func Form = marriage*noncontracept*abortion*lactation*fecundity
────────────────────────────────────────────────────────────────────────
                Component         Absolute Difference    Proportion (%)
────────────────────────────────────────────────────────────────────────
                 marriage                       -1.09         52.46
            noncontracept                       -1.23         59.13
                 abortion                       -.728         35.00
                lactation                         .84        -40.38
                fecundity                        .129         -6.20
────────────────────────────────────────────────────────────────────────
                  Overall                       -2.08        100.00
────────────────────────────────────────────────────────────────────────
```

The first column in output 2 shows the factor names, and the second column reports absolute contributions of each factor to the decline in recorded fertility rates. Regarding the reduction of 2.08 children per woman between 1960 and 1970, contraception and marriage contributed to a decline of about one child each. Abortion contributed about 0.73 to the total reduction of 2.08. The last column shows relative contributions of each factor as a percent of the total difference. Here 59.1% of the decline in TFRs during the decade can be attributed to increased use of contraception. During the same time, however, the duration of lactation declined and contributed to an increase in fertility. Results derived from `rdecompose` match the outcomes reported in the original article.

4.2 Example 2: Explaining differences in desire for more children

The second example is adapted from Clogg and Eliason (1988), who examine the desire to bear more children. It illustrates decomposition using cross-classified data. Table 2 compares desire for more children in two groups of women: those with 4 or more children (represented by parity 4+) and those with 1 child (represented as parity 1). Given that age is an important determinant of fertility and that most parity 1 women are likely younger than women with 4 or more children, the question is how to isolate the effect of age composition differences in the two parity groups. `rdecompose` can be applied to isolate the effect of age composition from actual differences in rates between the two groups of women.

Table 2. Population size and percent desiring more children (rate) by age

Age groups	Parity 4+ (population 1) Size (Ni)	Rate (Ti)	Parity 1 (population 2) Size (Ni)	Rate (Ti)
20 to 24	27	37.037	363	90.083
25 to 29	152	19.079	208	76.923
30 to 34	224	15.179	96	56.25
35 to 39	239	5.021	59	20.339
40 to 44	211	6.161	48	10.417
All ages	853	11.489	774	72.093

Source: Clogg and Eliason (1988)

As shown, this example uses the `transform()` option to convert the absolute number for size into proportions within the population group. The result suggests that the rate effect contributes about 62% of the difference, whereas the size effect (size of each age group) contributes about 38% of the differences.

Output 3: Stata code and command output of example 2

```
. use example2-clogg
. rdecompose Size Rate, group(Parity) transform(Size) sum(age_group)
Decomposition between Parity == 1 (11.49)
                  and Parity == 4 (72.09)
Func Form = \sum(age_group){Size*Rate}
------------------------------------------------------------------------
                Component       Absolute Difference     Proportion (%)
------------------------------------------------------------------------
                  Size(*)                      23.1             38.07
                     Rate                      37.5             61.93
------------------------------------------------------------------------
                  Overall                      60.6            100.00
------------------------------------------------------------------------
(*) indicates transformed variables
```

Source: Zhai, Goss, and Li (2017)

4.3 Example 3: Explaining drivers of health expenditures in China

The third example is from Zhai, Goss, and Li (2017), which examines the factors driving rising health expenditures in China. Using published National Health Accounts reports and disease prevalence data from the Global Burden of Disease 2013 Study from the Institute for Health Metrics and Evaluation (2015), this example decomposes the growth of health expenditure between 1993 and 2012 in China into five factors: population increase, changes in disease prevalence rates, shifts in age structure of the population, excessive health price inflation, and changes in treatment cost per case. This example showcases a more complex decomposition with multiple factors and the `detail` option.

`rdecompose` allows users to aggregate results from multiple subgroups—disease and age groups in this example. The `detail` option allows the program to display more detailed decomposition results normally hidden by the sum process. As seen in output 4, the `detail` option shows the contribution to health expenditures by disease and age group. Given page limits, only partial results are presented.

The decomposition suggests that real expenditure per case (`exp_percase`), excess health price inflation (`ehpi`), and aging (`aging`) drove increased health expenditures in China between 1993 and 2012. Population growth (`population`) was a secondary factor. Reductions in disease prevalence rates (`prevalence_rate`) only slightly slowed the growth in expenditures. Moreover, the result suggests that more than 70% of the difference in health expenditures on neoplasms and circulatory, respiratory, endocrine, nutritional, metabolic, and digestive diseases over the period was caused by changes in expenditures per case and the excess health price inflation. Examining results in the "detail" section reveals that aging of the population contributes more to growth of expenditures on neoplasms and circulatory, endocrine, nutritional, metabolic, and digestive diseases versus other diseases.

Output 4: Stata code and command output of example 3

```
. use example3-zhai

. rdecompose prevalence_rate population ageing exp_percase ehpi, group(year)
> sum(disease age_group) detail

Decomposition between year == 1993 (124535.24)
                  and year == 2012 (1000586.64)
Func Form = \sum(disease)\sum(age_group){prevalence_rate*population*
> ageing*exp_percase*ehpi}
────────────────────────────────────────────────────────────────────────
                Component         Absolute Difference    Proportion (%)
────────────────────────────────────────────────────────────────────────

          prevalence_rate                      -18982         -2.17
               population                       59535          6.80
                   ageing                       98405         11.23
              exp_percase                      635946         72.59
                     ehpi                      101148         11.55
────────────────────────────────────────────────────────────────────────
    Value of disease and Components         Detailed Contributions
────────────────────────────────────────────────────────────────────────
           Blood   prevalence_rate                   -507         -0.06
                        population                    500          0.06
                            ageing                    426          0.05
                       exp_percase                   4183          0.48
                              ehpi                    851          0.10
     Circulatory   prevalence_rate                  -6919         -0.79
  (output omitted)
────────────────────────────────────────────────────────────────────────
                  Overall                      876051        100.00
────────────────────────────────────────────────────────────────────────
```

4.4 Example 4: Income equality decomposition

The rate decomposition method also applies to other fields such as economics. A series of studies (for example, Bargain and Callan [2010]) decomposed changes in inequalities between countries using Shorrocks–Shapley decomposition (Shorrocks 2013). Such studies often use indexes derived from counterfactual simulations to attribute the relative importance of each component. In this case, decomposing the contributing factors to the income inequality resembles a rate decomposition.

The contribution of a factor to income inequality sometimes is determined based on Shapley values, which are computed by averaging the effects of all possible permutations before and after the factor of interest is substituted. This approach avoids path dependency (Devicienti 2010; Okamoto 2011) but treats the first-order effect with the same weight as mixed effects where multiple input factors have been substituted. Alternatively, the estimation of a factor's contribution can follow Das Gupta's approach, where weights are normalized, giving greater weight to direct effects. `rdecompose` can be used for such analyses.

Because the outcome value cannot be described as a simple function, specific values can be passed via the `function()` option of the command. For instance, to assess the contribution to Gini from two factors (for example, the population structure and the tax system as in table 3), one can use `rdecompose` as demonstrated in output 5.

Table 3. Estimated Gini coefficient with two factors

Gini		Factor 1 (for example, population) 1	2
Factor 2 (for example, tax system)	1	0.31	0.39
	2	0.48	0.52

Output 5: Decompose contributions to Gini based on two factors

```
. rdecompose factor1 factor2, group(group)
> function(cond(factor1==1, cond(factor2==1,0.31,0.48),
> cond(factor2==1,0.39,0.52)))
Decomposition between group == 1 (0.31)
                  and group == 2 (0.52)
Func Form = cond(factor1==1, cond(factor2==1,0.31,0.48),
> cond(factor2==1,0.39,0.52))
----------------------------------------------------------------------
               Component        Absolute Difference    Proportion (%)
----------------------------------------------------------------------
                 factor1                        .06         28.57
                 factor2                        .15         71.43
----------------------------------------------------------------------
                 Overall                        .21        100.00
----------------------------------------------------------------------
```

4.5 Example 5: Bootstrap with rdecompose

`rdecompose` does not have the native support of standard-error estimations in its current version because sources of sampling and nonsampling errors could vary substantially for each case. Standard errors in some cases might be meaningless if population-level data are applied or possibly inaccurate because of data confidentialization processes. However, should uncertainties of input data be mathematically described, it may be possible to derive the standard errors of the estimators. One way to do this is to use the bootstrap technique (Wang et al. 2000), which can be programmed in combination with `rdecompose`.

For example, to consider the sampling errors in example 2, one may bootstrap the underlying sample, which can be presented as a unit record dataset. A short customized program can be written in Stata to extract the `rdecompose` output for each iteration of the bootstrapping process.

Output 6: A customized rdecompose program for bootstrapping

```
program mydecompose, eclass
  preserve
  collapse (count) Size= d (mean) Rate = d, by(age_group Parity)
  quietly rdecompose Size Rate, group(Parity) transform(Size) sum(age_group)
  matrix b = e(b) * 100
  ereturn post b
  restore
end
```

The dataset from example 2 needs to be transformed for the `bootstrap` command as shown in output 7, which includes both the commands and the results of this example.

Output 7: Results from bootstrapping

```
. use example2-clogg
. expand Size
(1,617 observations created)
. by age_group Parity, sort: generate d = _n<=round(Rate*Size/100)
. bootstrap, nowarn nodots reps(1000): mydecompose
Bootstrap results                               Number of obs     =      1,627
                                                Replications      =      1,000

------------------------------------------------------------------------------
             |   Observed   Bootstrap                         Normal-based
             |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     Size(*) |   23.07168   2.459112     9.38   0.000     18.25191    27.89145
        Rate |   37.53248   3.136146    11.97   0.000     31.38575    43.67922
------------------------------------------------------------------------------
```

As shown, this short program returns the standard errors of estimated rate contributions via bootstrapping. With minor changes to the program, one can estimate the standard error of the percentage values if needed.

5 Concluding remarks

This article presents a user-written command, `rdecompose`, that replicates and extends the popular rate decomposition method presented by Das Gupta (1993). This command provides researchers a user-friendly tool to decompose changes in populations, which is a task common to research in demography, health, and economics. This tool reduces the tediousness of programming large numbers of factors and relaxes the functional requirement of the original method. `rdecompose` normally assumes that the underlying relations of the rate calculation are known; however, some or all permutations can be overridden via the `function()` option. The command currently has no native support for standard-error estimation because each case may contain vastly different sources of sampling and nonsampling errors. It may be possible, however, to use bootstrap techniques to derive standard errors of the estimates.

6 Acknowledgments

The author would like to thank Yohannes Kinfu, the reviewers, and the editors for their comments and suggestions.

7 References

Bargain, O., and T. Callan. 2010. Analysing the effects of tax-benefit reforms on income distribution: A decomposition approach. *Journal of Economic Inequality* 8: 1–21.

Blinder, A. S. 1973. Wage discrimination: Reduced form and structural estimates. *Journal of Human Resources* 8: 436–455.

Bongaarts, J. 1978. A framework for analyzing the proximate determinants of fertility. *Population and Development Review* 4: 105–132.

Chevan, A., and M. Sutherland. 2009. Revisiting Das Gupta: Refinement and extension of standardization and decomposition. *Demography* 46: 429–449.

Clogg, C. C., and S. R. Eliason. 1988. A flexible procedure for adjusting rates and proportions, including statistical methods for group comparisons. *American Sociological Review* 53: 267–283.

Das Gupta, P. 1978. A general method of decomposing a difference between two rates into several components. *Demography* 15: 99–112.

———. 1991. Decomposition of the difference between two rates and its consistency when more than two populations are involved. *Mathematical Population Studies* 3: 105–125.

———. 1993. *Standardization and Decomposition of Rates: A User's Manual, Volume 1.* Washington, DC: U.S. Department of Commerce, Economics and Statistics Administration, Bureau of the Census.

Devicienti, F. 2010. Shapley-value decompositions of changes in wage distributions: A note. *Journal of Economic Inequality* 8: 35–45.

Guo, Z., Z. Wu, C. M. Schimmele, and S. Li. 2012. The effect of urbanization on China's fertility. *Population Research and Policy Review* 31: 417–434.

Institute for Health Metrics and Evaluation. 2015. GBD compare. http://vizhub.healthdata.org/gbd-compare.

Jann, B. 2008. The Blinder–Oaxaca decomposition for linear regression models. *Stata Journal* 8: 453–479.

Keiding, N., and D. Clayton. 2014. Standardization and control for confounding in observational studies: A historical perspective. *Statistical Science* 29: 529–558.

Kitagawa, E. M. 1955. Components of a difference between two rates. *Journal of the American Statistical Association* 50: 1168–1194.

Okamoto, M. 2011. Source decomposition of changes in income inequality: The integral-based approach and its approximation by the chained Shapley-value approach. *Journal of Economic Inequality* 9: 145–181.

Powers, D. A., H. Yoshioka, and M.-S. Yun. 2011. mvdcmp: Multivariate decomposition for nonlinear response models. *Stata Journal* 11: 556–576.

Sastre, M., and A. Trannoy. 2002. Shapley inequality decomposition by factor components: Some methodological issues. *Journal of Economics* 77: 51–89.

Shorrocks, A. 2013. Decomposition procedures for distributional analysis: A unified framework based on the Shapley value. *Journal of Economic Inequality* 11: 99–126.

Sinning, M., M. Hahn, and T. K. Bauer. 2008. The Blinder–Oaxaca decomposition for nonlinear regression models. *Stata Journal* 8: 480–492.

Wang, J., A. Rahman, H. A. Siegal, and J. H. Fisher. 2000. Standardization and decomposition of rates: Useful analytic techniques for behavior and health studies. *Behavior Research Methods, Instruments, and Computers* 32: 357–366.

Wolfbein, S. L., and A. J. Jaffe. 1946. Demographic factors in labor force growth. *American Sociological Review* 11: 392–396.

Yamaguchi, K. 2011. Decomposition of inequality among groups by counterfactual modeling: An analysis of the gender wage gap in Japan. *Sociological Methodology* 41: 223–255.

Zhai, T., J. Goss, and J. Li. 2017. Main drivers of health expenditure growth in China: A decomposition analysis. *BMC Health Services Research* 17: 185–193.

About the author

Jinjing Li is an associate professor at the National Centre for Social and Economic Modelling of the Institute for Governance and Policy Analysis, University of Canberra, Australia.

The Stata Journal (2017)
17, Number 2, pp. 503–510

Covariate-constrained randomization routine for achieving baseline balance in cluster-randomized trials

Eva Lorenz
Institute of Public Health
University of Heidelberg
Heidelberg, Germany
and
Institute of Medical Biostatistics
Epidemiology and Informatics
University Medical Center
Mainz, Germany
eva.lorenz@uni-mainz.de

Sabine Gabrysch
Institute of Public Health
University of Heidelberg
Heidelberg, Germany
sabine.gabrysch@uni-heidelberg.de

Abstract. In cluster-randomized trials, groups or clusters of individuals, rather than individuals themselves, are randomly allocated to intervention or control. In this article, we describe a new command, `ccrand`, that implements a covariate-constrained randomization procedure for cluster-randomized trials. It can ensure balance of one or more baseline covariates between trial arms by restriction to allocations that meet specified balance criteria. We provide a brief overview of the theoretical background, describe `ccrand` and its options, and illustrate it using an example.

Keywords: st0484, ccrand, covariate-constrained randomization, cluster-randomized trials

1 Introduction

In cluster-randomized trials, groups or clusters of individuals, rather than individuals themselves, are randomly allocated to intervention or control (Donner and Klar 2000). Provided the sample size is large, randomization will ensure comparable arms in terms of the distribution of known and, more importantly, unknown factors that may influence the outcome (Dos Santos Silva 1999). The number of clusters in cluster-randomized trials is often limited. Therefore, one cannot rely on chance alone to ensure balance of important covariates (including the baseline value of the outcome) and sample size between arms (Moulton 2004). Achieving baseline balance of important covariates not only avoids the need to adjust for them in the final analysis but also is important for a trial's credibility. Furthermore, it increases statistical power and precision (Ivers et al. 2012).

In a recent methodological review, Ivers et al. (2012) discussed several techniques to balance baseline covariates in cluster-randomized trials along with their advantages and limitations, including stratification, matching, minimization, and covariate-constrained randomization. The last is often the method of choice when baseline data are available but has rarely been used because of the need for statistical support and specialized computer software to implement it (Ivers et al. 2012). Moulton (2004) proposed a method for implementing covariate-constrained randomization, and Chaudhary and Moulton (2006) wrote a SAS command and tutorial. Carter and Hood (2008) implemented a covariate-constrained randomization tool in R.

Our aim is to make covariate-constrained randomization more accessible to non-statisticians by providing Stata code and demonstrating its use in a simple example. In this article, we give a brief overview of the covariate-constrained randomization procedure, including notation, key definitions, and concepts. Then, we describe the `ccrand` command and its implementation in Stata. Finally, we illustrate the command using an example dataset. We provide the code for the example in the supporting information (available from the journal's web page).

Stratified randomization can be useful when there is substantial variability in the outcome of interest between clusters. This means that strata are formed and clusters are randomized separately in each relatively homogeneous stratum. While stratification can also help balance on a limited number of covariates, it is recommended mainly for improving precision of estimation by reducing intracluster correlation (Hayes and Moulton 2009). Our procedure allows users to combine restricted randomization with stratification to both ensure balance and improve efficiency.

2 The covariate-constrained randomization procedure

2.1 The basic idea

The purpose of covariate-constrained randomization is to achieve balance between arms on one or more important baseline covariates that are thought to be predictive of the outcome of the trial, often including the baseline values of the primary endpoint itself (Moulton 2004). This requires that covariate data be available for all clusters prior to allocation. The main method for covariate-constrained randomization implies that all possible allocations of dividing the clusters into two arms are simulated, and differences in covariates between arms are calculated for each and checked against prespecified balance criteria. For example, for the continuous covariate education, one could specify the difference in group means between arms to be no greater than one year's difference in mean school attendance. The final set of allocations is then limited to those that meet the prespecified balance criteria, and the actual allocation is chosen randomly from this acceptable set.

2.2 Validity

If the constraints are too strict, it may be that two clusters are always or never in the same arm in all the acceptable allocations because the balance criteria can be fulfilled only this way. This challenges the validity of the design because then it would not hold true that "every pair of clusters has the same probability of being allocated to the same treatment", which "violates the assumption of independence between clusters in each arm" and could potentially change the type I error (Hayes and Moulton 2009). These situations can be identified by counting the number of times any given pair of clusters has the same treatment allocation. Examining under- or overrepresented pairs can then reveal the balance constraints responsible, which need to be relaxed (Moulton 2004). However, relaxing the balancing constraints implies a reduction of covariate balance between study arms.

3 The ccrand command

Our approach is similar to the steps described by Chaudhary and Moulton (2006) for the SAS command.

1. Separately for each stratum, we first generate all possible allocations dividing the clusters into two arms.
2. For each allocation (in each stratum), we calculate the means of relevant covariates in both arms and retain only the allocations meeting the balance criteria. If too few allocations are retained in some strata, the criteria need to be relaxed. If too many allocations are retained (which may make the subsequent computation too heavy), the criteria should be tightened.
3. By combining the acceptable stratum-level allocations, we generate all possible overall allocations.
4. For each overall allocation, we calculate the means of relevant covariates in both arms and again retain only the allocations that meet the overall balance criteria.
5. Finally, we perform a validity check by calculating how often each pair of clusters is allocated to the same arm. If n is the number of acceptable allocations, it should occur in approximately $n/2$ of these. The exact value is $m! \times \binom{n}{k}$, where m is the number of clusters in a pair of clusters (2), n is the number of clusters per stratum, and k is the number of strata.
6. We select the final allocation at random from all acceptable overall allocations.

3.1 Syntax

The syntax of `ccrand` is as follows:

`ccrand` *mainvarlist*`, ibc("`*string*`") obc("`*string*`")` [`seed(`*#*`) cluster(`*varname*`)`
`stratum(`*varname*`) validitycheck(`*string*`) selectfinal(`*string*`)`]

where *mainvarlist* contains all the covariates to be balanced. Arguments in squared brackets are optional and have default values assigned as described below.

3.2 Options

`ibc("`*string*`")` (initial balancing criteria) specifies a string of numeric values that is equal to the number of covariates separated by a blank (`" "`) and that represents the maximum allowable differences between arms for each covariate for allocations within strata. All covariates must satisfy the criteria individually for an allowable allocation. Values must be positive. `ibc()` is required.

`obc("`*string*`")` (overall balancing criteria) specifies a string of numeric values that is equal to the number of covariates separated by a blank (`" "`) and that represents the maximum allowable differences between arms for each covariate for overall allocations. All covariates must satisfy the balancing criteria individually for an allowable allocation. Values must be positive. `obc()` is required.

`seed(`*#*`)` sets the reproducible random-number seed to # for selecting one final overall allocation from all acceptable overall allocations. The seed needs to be set to reproduce the results.

`cluster(`*varname*`)` specifies the name of the covariate containing cluster IDs. The default is `cluster(cluster)`.

`stratum(`*varname*`)` specifies the name of the covariate containing stratum IDs. The default is `stratum(stratum)`.

`validitycheck(`*string*`)` specifies whether the validity check should be performed and details displayed after randomization (`no`; default is `validitycheck(yes)`).

`selectfinal(`*string*`)` specifies whether one final overall allocation should be selected at random from all acceptable overall allocations (`no`; default is `selectfinal(yes)`).

3.3 The data structure

`ccrand` requires an input Stata dataset that contains the stratum and cluster IDs. It also requires the cluster-level covariates to be balanced on.

4 Illustration using a data example

4.1 Example: A cluster-randomized trial of a complex intervention to reduce child undernutrition

The aim of the Food and Agricultural Approaches to Reducing Malnutrition (FAARM) cluster-randomized controlled trial
(http://www.clinicaltrials.gov/ct2/show/NCT02505711/) is to evaluate the impact of an integrated home gardening, nutrition, and hygiene intervention on chronic undernutrition in young children in a low-income setting. The intervention is delivered to women's groups. The trial includes 2,700 young women and their children from 96 settlements within 2 subdistricts of Habiganj District in Bangladesh. Following a baseline

survey in 2015, settlements were allocated to either intervention or control arms in a 1:1 ratio using covariate-constrained randomization. Women in the intervention arm will receive training and support over three years. The primary endpoint is linear growth (length for age) in children under three years old, which will be collected in 2019.

The dataset

The example dataset is a subset of the FAARM baseline survey containing 24 clusters in 3 strata and 5 covariates to balance on. A partial output of the data is shown in table 1 for illustration. Each of the three strata contains eight clusters, of which four clusters are to be allocated to the intervention arm. The five cluster-level baseline covariates (some converted into z scores) are the number of included women in the cluster (`women`), women's height (`z_wht`), child length for age at baseline (`z_lfa`), child age in months at baseline (`z_age`), and child diarrhea prevalence at baseline (`diarrhea`).

Table 1. Listing of the first five lines of the input Stata dataset sorted by cluster

obs	cluster	s	`women`	`z_wht`	`z_lfa`	`z_age`	`diarrhea`
1	1	3	37	−0.24	0.70	−0.22	0.10
2	2	1	14	−0.32	−0.05	−0.06	0.10
3	3	1	17	0.21	1.86	−0.26	0.11
4	4	3	51	−0.07	−0.61	0.52	0.07
5	5	2	29	−0.54	−1.17	0.01	0.16

As in the example, covariates of interest may be standardized before applying the `ccrand` command by calculating the respective z scores. The z score measures how many standard deviations above the mean (positive values) or below the mean (negative values) a data point is. z scores can be calculated using the formula $Z = (X - \mu)/\sigma$, where X is the covariate of interest, μ is the mean, and σ is the standard deviation. Constraints can then be set in terms of standard deviations instead of the original covariate units.

The command

```
ccrand women diarrhea z_wht z_lfa z_age, ibc("4 0.2 0.5 0.5 0.5")  ///
  obc("3 0.1 0.5 0.5 0.5") seed(89574) cluster(cluster) stratum(stratum)
```

The output

After processing the input dataset, the program generates all possible arm 1 and arm 2 allocations separately along with the relevant covariate data. In our example, we want to choose 4 clusters from 8 in each stratum; thus the number of combinations per stratum is 8 choose 4 = $\binom{8}{4} = 70$. The total number of arm 1 allocations calculated by the program for all 3 strata is the sum of the combinations in the strata $\binom{8}{4} + \binom{8}{4} + \binom{8}{4} =$

$70+70+70 = 210$, which yields $210\times4 = 840$ data rows. The number of combinations is multiplied by 4 because there are 4 clusters stored in a separate row in each combination. For each stratum, we compute covariate means for each allocation in each arm separately and merge the results to compute the differences in means between the two arms. These differences are compared with the values of the within-stratum initial balancing criteria (`ibc()`). We then select the allocations that satisfy these initial criteria. In this example, 62 allocations out of a total of 210 qualify: 18 in stratum 1, 22 in stratum 2, and 22 in stratum 3. Based on these acceptable allocations in each stratum, all possible overall allocations are generated by selecting one allocation from each stratum. In our example, this results in a total of $(18 \times 22 \times 22) \times 3 = 8712 \times 3 = 26136$ possible overall allocations. Again, we calculate covariate means in both arms and check the difference in means against the overall balancing criteria (`obc()`). In the example, 8,328 allocations fulfilled these criteria. A listing of these allocations with the respective stratum IDs and within-stratum allocation IDs, called `rno`, is saved in a dataset. Finally, we compute the number of times a cluster appears with another cluster in the same arm as a check for the validity of allocations. In the end, one allocation—to be used to implement the randomization for the trial—is selected at random from those allocations qualifying overall balancing criteria and displayed by the program. With the seed chosen in the example, this results in allocation 498 with the "keep it simple, stupid" (KISS) random-number generator (default until Stata 13) and allocation 283 with the Mersenne Twister generator (new default introduced in Stata 14).

Recommendations

A prerequisite of the covariate-constrained randomization method is that recruitment of clusters and collection of covariates for balancing must be completed prior to the cluster allocation.

Implementing the randomization procedure supports a large number of strata with a moderate number of clusters within strata. The initial balancing criteria should not result in more than 25 allowable allocations per stratum, because all possible combinations of clusters per strata are generated in the next step, which increases exponentially and is computationally very intense. There are no restrictions on the maximum number of covariates to balance on.

The program can also handle uneven numbers of clusters per stratum and will assign the higher number of clusters to the first study arm, that is, for a stratum size of 7, 4 clusters will be assigned to the first study arm and 3 clusters will be assigned to the second study arm.

5 Conclusions

Covariate-constrained randomization is a valuable tool for cluster-randomized trials. However, the method was used in only 2% of 300 trials published from 2000 to 2008 (Ivers et al. 2011). This is partly because of its apparent complexity and the lack of software routines until the recent programs in SAS and R were available (Chaudhary and Moulton 2006; Carter and Hood 2008). Because our program is inspired by the SAS macro written by Chaudhary and Moulton, it covers the same functionality. Additionally, the number of covariates is not limited in the Stata implementation and the performance (run time) is better. The implementation in R does not provide the validity check but can be extended with knowledge of programming in R. We hope that this routine will make the procedure more accessible to a wider audience of applied researchers.

6 Acknowledgments

The program is inspired by the `CCR_V1.0` macro in SAS, which was written by Chaudhary and Moulton (2006). We thank Larry Moulton for helpful discussions via email.

We are very grateful to Anja Schoeps, Robin C. Nesbitt, and especially Andreas Deckert and an anonymous reviewer for their valuable suggestions on how to improve this command.

7 Funding sources

Sabine Gabrysch is funded by a grant (award number 01ER1201) of the German Federal Ministry of Education and Research. The content of this publication is solely the responsibility of the authors.

8 Contributions

Eva Lorenz implemented `ccrand` and wrote the first draft of this article. Sabine Gabrysch had the idea for `ccrand`, supervised its implementation, and contributed substantially to the writing of this article. Both authors read and approved the final version of the manuscript.

9 References

Carter, B. R., and K. Hood. 2008. Balance algorithm for cluster randomized trials. *BMC Medical Research Methodology* 8: 65.

Chaudhary, M. A., and L. H. Moulton. 2006. A SAS macro for constrained randomization of group-randomized designs. *Computer Methods and Programs in Biomedicine* 83: 205–210.

Donner, A., and N. Klar. 2000. *Design and Analysis of Cluster Randomization Trials in Health Research*. London: Arnold.

Dos Santos Silva, I., ed. 1999. *Cancer Epidemiology: Principles and Methods*. Geneva: World Health Organization.

Hayes, R. J., and L. H. Moulton. 2009. *Cluster Randomised Trials*. Boca Raton, FL: Chapman & Hall/CRC.

Ivers, N. M., I. J. Halperin, J. Barnsley, J. M. Grimshaw, B. R. Shah, K. Tu, R. Upshur, and M. Zwarenstein. 2012. Allocation techniques for balance at baseline in cluster randomized trials: A methodological review. *Trials* 13: 120.

Ivers, N. M., M. Taljaard, S. Dixon, C. Bennett, A. McRae, J. Taleban, Z. Skea, J. C. Brehaut, R. F. Boruch, M. P. Eccles, J. M. Grimshaw, C. Weijer, M. Zwarenstein, and A. Donner. 2011. Impact of CONSORT extension for cluster randomised trials on quality of reporting and study methodology: Review of random sample of 300 trials, 2000-8. *British Medical Journal* 343(d5886): 1–14.

Moulton, L. H. 2004. Covariate-based constrained randomization of group-randomized trials. *Clinical Trials* 1: 297–305.

About the authors

Eva Lorenz is a research associate at the Institute of Medical Biostatistics, Epidemiology and Informatics in Mainz, Germany. Her research interests include development and application of epidemiological methodology and statistical programming.

Sabine Gabrysch is head of the Unit of Epidemiology and Biostatistics and deputy head of the Institute of Public Health at Heidelberg University, Germany. Her main research interest is maternal and child health in low-income settings. She is leading the FAARM trial on malnutrition in Bangladesh.

The Stata Journal (2017)
17, Number 2, pp. 511–514

Stata tip 127: Use capture noisily groups

Roger B. Newson
Department of Primary Care and Public Health
Imperial College London
London, UK
r.newson@imperial.ac.uk

1 The main idea

Do you know about the `capture noisily` group? It is a group of Stata commands, starting with `capture noisily {` and ending with `}`. The commands in between are executed as usual, producing the standard output (because of `noisily`). If any of these commands fail, then execution resumes with the command immediately following the group (because of `capture`). If you do not like typing `capture noisily`, then you can abbreviate it to `cap noi`, or even to `cap n`.

A simple application is where the user wishes to generate a Stata log file, which the user inspects afterward to see whether the logged commands work (and especially if they do not). In a do-file, the user may open the log file and begin the `capture noisily` block, using the commands

```
log using mylog.log, replace
capture noisily {
```

and then add a sequence of Stata commands, such as

```
sysuse auto, clear
regress mpg weight
predict mpghat
twoway scatter mpg weight || line mpghat weight, sort
```

and then end the `capture noisily` block and close the log file, using the commands

```
}
log close
```

The commands inside the block will then be executed until one of them fails (or until all of them end execution, if none fail), and their output will be stored in the file `mylog.log`. Whether or not any of the intervening commands fail, the log file `mylog.log` will be closed by the `log close` command. The user may then inspect the log file with a text editor and view the results if execution was successful or find out what went wrong otherwise. Typically, the number of Stata commands inside the block will be more than the four used here, and there may be program loops and other complicated programming constructions (see [P] **forvalues** and [P] **foreach**), increasing the probability of a failure somewhere.

2 Application in estimation command files

The `capture noisily` prefix is commonly used in do-files containing sequences of estimation commands. If the user is worried that one or more of them might fail (possibly because of insufficient observations), then the user may add a `capture noisily` prefix to each estimation command so that if one estimation command fails, then Stata will resume execution, starting with the next estimation command. If each estimation command is followed by one or more postestimation commands (such as `predict` or `margins`), then each estimation command, and its own subsequent postestimation commands, may be placed in its own `capture noisily` group. That way, if either the estimation command or the postestimation commands fail, then Stata will continue to the next estimation command.

For instance, in `auto.dta`, a user might want to fit a regression model of mileage (`mpg`) with respect to each of the car-size variables `weight`, `length`, and `displacement`, together with the factor `foreign`, indicating whether a car model is made by a non-U.S. company. After each regression model, the user might want to estimate the mean mileages expected if all cars were U.S. models and if all cars were non-U.S. models, assuming that the car-size variable was distributed as in the real-world sample. The code to do this might be as follows:

```
sysuse auto, clear
describe, full
foreach X of var weight length displacement {
  capture noisily {
    regress mpg ibn.foreign `X´, noconst vce(robust)
    margins i.foreign
  }
}
```

As it happens, this code executes without any failed commands (not shown). However, if (for any reason) the analysis with respect to `weight` had failed, either in the `regress` command or in the `margins` command, then Stata would have proceeded to the analysis with respect to `length`. If the analysis with respect to `length` had failed, then Stata would have proceeded to the analysis with respect to `displacement`. This feature of `capture noisily` blocks can be very useful if the user is executing a long list of multistep analyses, especially if these analyses involve commands with a higher failure probability than `regress`. Note that if a multistep analysis fails at an earlier command in a `capture noisily` block, then the later commands in the same `capture noisily` block are not attempted. Note also that if the multiple analyses are simply the same command executed on multiple by-groups, then the user does not need the `capture noisily` block, because the user can use `statsby` (see [D] **statsby**).

3 Application in file-generation programs

capture noisily may also be used to good effect in file-generation programs. For example, it is used internally by the dolog, dologx, and dotex packages, which can be downloaded from the Statistical Software Components (SSC) archive, and used to execute a do-file while automatically generating a log file. However, more advanced users may want to write output to an arbitrary file, using the file command documented in [P] **file**. For instance, the new file may be a TEX, an HTML, an Extensible Markup Language, or a Rich Text Format file, produced as an automatically generated report for a reproducible-research project. The user may be using a sequence of commands to generate this new file and may want to close the file after executing those commands, whether or not they all work. The generated file will then be available for the user to inspect (although it may be incomplete), and the user will not have to close it manually. The capture block may begin with the commands

```
tempname buff1
file open `buff1´ using "myoutput.txt", write text replace
capture noisily {
```

and contain any amount of intervening code, including file write statements, such as

```
file write `buff1´ "Hello, world!!!!"
```

and end with the commands

```
}
file close `buff1´
```

In this case, a new file myoutput.txt is created with a buffer, whose name is stored in the local macro buff1, and filled with output from the intervening file write statements. However, if any statement in the intervening code fails, then Stata executes the file close statement, and the new file (usually incomplete) is available for the user to inspect.

Alternatively, the capture noisily block may be preceded and followed by file opening and closing commands other than file open and file close. For instance, if the user is generating an HTML file, then the file opening and closing commands might be the htopen and htclose commands of the ht package (see Quintó et al. [2012]) or the htmlopen and htmlclose commands of the SSC package htmlutil (see Newson [2015]). Or if the user is generating a Rich Text Format file, then they might be the rtfopen and rtfclose commands of the SSC package rtfutil (see Newson [2012]). And more user-written file-generating packages are likely to be written on similar lines in the future, possibly for generating files in Extensible Markup Language-based document formats yet to be invented. Such future packages are likely to contain their own file-opening and file-closing commands, suitable for use before and after a capture noisily block.

4 References

Newson, R. 2015. htmlutil: Stata module to provide utilities for writing hypertext markup language (HTML) files. Statistical Software Components S458085, Department of Economics, Boston College.
https://ideas.repec.org/c/boc/bocode/s458085.html.

Newson, R. B. 2012. From resultssets to resultstables in Stata. *Stata Journal* 12: 191–213.

Quintó, L., S. Sanz, E. De Lazzari, and J. J. Aponte. 2012. HTML output in Stata. *Stata Journal* 12: 702–717.

The Stata Journal (2017)
17, Number 2, pp. 515–516

Software Updates

dm0078_2: newspell: Easy management of complex spell data. H. Kröger. *Stata Journal* 16: 244; 15: 155–172.

In previous versions of the program, the `newspell fillin` command did not fill in gaps correctly if the `both` or `bothpost` suboption of the `fill()` option was specified. This has been fixed. Further, the `newspell gap` command would not find gaps in certain situations (if the earliest spell ends on the same time point as it begins). This has also been fixed.

gr0064_1: graphlog: Creating log files with embedded graphics. M. R. Hansen. *Stata Journal* 15: 594–596.

The new version fixes several bugs and allows user control of text color and character encoding (useful if your Stata code contains non-English characters).

st0376_1: Estimating and modeling relative survival. P. W. Dickman and E. Coviello. *Stata Journal* 15: 186–215.

The new version corrects a bug (line 691) that resulted in incorrect estimates of the Pohar Perme (actuarial) estimator when all individuals in an interval died.

st0389_4: Conducting interrupted time-series analysis for single- and multiple-group comparisons. A. Linden. *Stata Journal* 17: 73–88; 16: 813–814; 16: 521–522; 15: 480–500.

The time variable (`_t`) was reset to start at 0 rather than 1 (as originally reflected in update `st0389_1` but lost in update `st0389_2`).

st0446_1: Hot and cold spot analysis using Stata. K. Kondo. *Stata Journal* 16: 613–631.

The following changes have been made, with further details in the help file:

1. A bug where a long variable name breaks layout has been fixed.
2. A calculation process of a large distance matrix has been improved.
3. The maximum number of iterations in the Vincenty formula has been changed from 100 to 100,000.
4. The `nomatsave` option has been added to save computer memory space in the calculation process.
5. The `largesize` option has been added to deal with large datasets. This new option computes the Getis–Ord $G_i^*(d)$ statistic faster when the dataset is quite large.

st0470_1: Spatial panel-data models using Stata. F. Belotti, G. Hughes, and A. Piano Mortari. *Stata Journal* 17: 139–180.

This update fixes the `xsmle` command to allow for the proper use of a user-specified weight variable, through either `iweight` or `aweight`.[1]

1. We thank Augusto Cerqua for pointing out this bug.