The Stata Journal (2023)
23, Number 2, pp. 386–401 DOI: 10.1177/1536867X231175265

lgrgtest: Lagrange multiplier test after constrained maximum-likelihood estimation

Harald Tauchmann
Friedrich-Alexander-Universität Erlangen-Nürnberg
Nüremberg, Germany
harald.tauchmann@fau.de

Abstract. Besides the Wald and likelihood-ratio tests, the Lagrange multiplier test (Rao, 1948, *Mathematical Proceedings of the Cambridge Philosophical Society* 44: 50–57; Aitchison and Silvey, 1958, *Annals of Mathematical Statistics* 29: 813–828; Silvey, 1959, *Annals of Mathematical Statistics* 30: 389–407) is the third canonical approach to testing hypotheses after maximum likelihood estimation. While the Stata commands `test` and `lrtest` implement the first two, Stata does not have an official command for implementing the third. The community-contributed `boottest` package (Roodman et al., 2019, *Stata Journal* 19: 4–60) focuses on methods of bootstrap inference and also implements the Lagrange multiplier test functionality. In this article, I introduce the new community-contributed postestimation command `lgrgtest`, which allows for straightforwardly using the Lagrange multiplier test after constrained maximum-likelihood estimation. `lgrgtest` is intended to be compatible with all Stata estimation commands that use maximum likelihood and allow for the options `constraints()`, `iterate()`, and `from()`. `lgrgtest` can also be used after `cnsreg`.

Keywords: st0712, lgrgtest, test, lrtest, constraint, Lagrange multiplier test, score test, constraints, maximum likelihood

1 Introduction

Testing hypotheses after model estimation is day-to-day business for any empirical researcher. For Stata users, using Stata's `test` command is probably the most obvious choice for implementing such tests. After maximum likelihood (ML) estimation, `test` performs a Wald test. Yet, besides the Wald test, the likelihood-ratio (LR) test and the Lagrange multiplier (LM) test are further canonical approaches to hypothesis testing after ML estimation (compare Engle [1984] and Greene [2018, 551–560]). The three tests take different approaches to evaluating whether imposing restrictions on a model is warranted by the observed data. The Wald test uses as its criterion the deviation of the estimated parameter values from the values assumed under the restrictions. The LR test looks at the loss of log likelihood imposing the restrictions. The LM test considers the slope of the log-likelihood function at the restricted maximum. Under the null of the restriction being valid, the three tests are asymptotically equivalent. In the rather special case of the log-likelihood function being quadratic in all parameters subject to estimation, they are even numerically identical irrespective of the sample size (Buse 1982; Engle 1984). In general, however, their properties are unknown in finite samples and may differ. In applied work, the Wald, LR, and LM tests will hence usually yield results that are more or less different from one another.

The choice between these different tests is often guided by practical considerations (compare Greene [2018, 555]). While the LR test requires estimating both the unrestricted and the restricted model, the Wald test requires estimating only the unrestricted model, and the LM test requires estimating only the null model. Imposing restrictions on the model may ease estimation for poorly behaved optimization problems, which in such cases can be regarded as an argument in favor of using the LM test. (Depending on the specific problem, however, imposing restrictions may also complicate estimation.) Moreover, tying the test to the null model has some appeal if the null model is the one on which the economic discussion focuses. Beyond practical considerations, Greene (2018, 557) mentions weaker required assumptions as an argument in favor of the Wald test, while he regards a lack of invariance to how the restrictions are formulated as an argument against it. Thus, despite their asymptotic equivalence, having all three tests available appears to be valuable for applied empirical research.

The LR test—as an alternative to the Wald test implemented by `test`—is made available to Stata users through the command `lrtest`. In contrast—though LM tests are available for specific settings, `xttest0` after `xtreg, re` and `estat scoretests` after `sem`, for instance—Stata does not have an official command for carrying out LM tests after (constrained ML) estimation. This article introduces `lgrgtest`, a community-contributed postestimation command for LM testing of constraints imposed on a model fit by ML. `lgrgtest` closely accompanies Stata's estimation option `constraints()`, which implements constrained estimation. The community-contributed `scoretest` command, which is part of the `boottest` package (Roodman et al. 2019), provides LM testing functionality. `lgrgtest` and `scoretest` use different syntax but yield identical results for comparable models. Because `boottest` is primarily concerned with implementing methods of bootstrap inference, the LM testing functionality of `scoretest` is not well known among Stata users; indeed, I was unaware of it until shortly before this article went to print. Arguably, there is value in having a standalone command available that focuses on the LM test. The remainder of this article is organized as follows. Section 2 briefly discusses the methodology of the LM test. Section 3 elaborates on the implementation in Stata. Section 4 introduces the syntax of `lgrgtest`. Section 5 provides an empirical illustration and further examples. Section 6 concludes.

2 The LM test

The LM test, also known as the score test, was introduced independently in different versions by Rao (1948) and Silvey (1959); see Bera and Bilias (2001) for details of its historical development. This strategy for hypothesis testing rests on maximizing a log-likelihood[1] function $\mathcal{L}(\boldsymbol{\theta})$ with respect to the $k \times 1$ parameter vector $\boldsymbol{\theta}$, subject to a set of q constraints $\boldsymbol{r}(\boldsymbol{\theta}) = \mathbf{0}$. These constraints are the restrictions to be tested, and the

1. The principle of the LM test can be generalized to other M estimators than ML (Wooldridge 2010, 421). Yet, following the bulk of econometric textbooks, we focus on the context of ML estimation.

null hypothesis is hence H_0: $\boldsymbol{r}(\boldsymbol{\theta}) = \mathbf{0}$. Following the depiction in Arellano (2004), the LM test statistic (LMS) reads as

$$\text{LMS} = \boldsymbol{g}\left(\widetilde{\boldsymbol{\theta}}\right)' \boldsymbol{I}\left(\widetilde{\boldsymbol{\theta}}\right)^{-1} \boldsymbol{g}\left(\widetilde{\boldsymbol{\theta}}\right) \tag{1}$$

$$= \widetilde{\boldsymbol{\lambda}}' \boldsymbol{R}\left(\widetilde{\boldsymbol{\theta}}\right)' \boldsymbol{I}\left(\widetilde{\boldsymbol{\theta}}\right)^{-1} \boldsymbol{R}\left(\widetilde{\boldsymbol{\theta}}\right) \widetilde{\boldsymbol{\lambda}} \tag{2}$$

with (1) representing the score version (Rao 1948) and (2) representing the LM version (Silvey 1959) of the test statistic. $\widetilde{\boldsymbol{\theta}}$ denotes the parameter estimates that maximize $\mathcal{L}(\boldsymbol{\theta})$ respecting the constraints. $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}})$ denotes the score vector $\partial\mathcal{L}(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$, $\boldsymbol{I}(\widetilde{\boldsymbol{\theta}})$ denotes the information matrix $-E\{\partial^2\mathcal{L}(\boldsymbol{\theta})/\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}'\}$, and $\boldsymbol{R}(\widetilde{\boldsymbol{\theta}})$ denotes the Jacobian $\partial\boldsymbol{r}(\boldsymbol{\theta})/\partial\boldsymbol{\theta}$, each evaluated at $\boldsymbol{\theta} = \widetilde{\boldsymbol{\theta}}$. The q LM values that the constrained maximization yields are collected in the vector $\widetilde{\boldsymbol{\lambda}}$. Under the null, LMS is asymptotically χ^2 distributed with q degrees of freedom.

The equivalence of the two representations of the LM statistic becomes obvious from the first-order condition $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}}) - \boldsymbol{R}(\widetilde{\boldsymbol{\theta}})\widetilde{\boldsymbol{\lambda}} = \mathbf{0}$ of the constrained maximization problem $\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) - \boldsymbol{\lambda}'\boldsymbol{r}(\boldsymbol{\theta})$ (compare Arellano [2004]). If the constraints do not bind and $\widetilde{\boldsymbol{\theta}}$ maximizes the likelihood function both with and without the restrictions imposed, $\widetilde{\boldsymbol{\lambda}}$ equals $\mathbf{0}$ and so does $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}})$ by the logic of ML estimation. Consequently, the test of the null can be implemented both as a test of $\widetilde{\boldsymbol{\lambda}} = \mathbf{0}$ and as a test of $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}}) = \mathbf{0}$. The LM test hence rejects the null if $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}})$ deviates so strongly from $\mathbf{0}$ that the observed deviation cannot be attributed to a sampling error given a certain type-one error probability. This equivalently applies to $\boldsymbol{R}(\widetilde{\boldsymbol{\theta}})\widetilde{\boldsymbol{\lambda}}$ from the multiplier representation of the LM test. The squared score vector enters the LM statistic weighted by the inverse information matrix; see (1). This weighting by $\boldsymbol{I}(\widetilde{\boldsymbol{\theta}})^{-1}$ takes into account that, for a given slope of $\mathcal{L}(\boldsymbol{\theta})$, a greater curvature implies a smaller distance from the unrestricted optimum, leading to a smaller value of LMS; see Buse (1982) for more detailed discussion of this argument, including an excellent graphical illustration.

Even with estimates $\widetilde{\boldsymbol{\theta}}$ in hand, $\boldsymbol{I}(\widetilde{\boldsymbol{\theta}})^{-1}$ is usually unknown and needs to be estimated for calculating the LM statistic. The two canonical approaches to estimation (compare Greene [2018, 559]) are 1) using the actually observed Hessian of the log-likelihood function evaluated at $\widetilde{\boldsymbol{\theta}}$, that is, $\widehat{\boldsymbol{I}}(\widetilde{\boldsymbol{\theta}})^{-1} = -\{\partial^2\mathcal{L}(\widetilde{\boldsymbol{\theta}})/\partial\widetilde{\boldsymbol{\theta}}\partial\widetilde{\boldsymbol{\theta}}'\}^{-1}$, and 2) using the outer product of the observation-level gradient vector, that is, $\widehat{\boldsymbol{I}}(\widetilde{\boldsymbol{\theta}})^{-1} = \{\sum_i \boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})\boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})'\}^{-1}$, with i indexing observations. Estimating $\boldsymbol{I}(\widetilde{\boldsymbol{\theta}})^{-1}$ for calculating the LM statistic thus parallels estimating the variance–covariance matrix of the ML estimator. This, however, applies only if either the observed Hessian (observed information matrix) or the outer product of the gradient vector is used for variance estimation, but no other approach such as the sandwich estimator or bootstrapping. The score version of the LM statistic is very conveniently computed, particularly if the outer product gradient approach is used for estimating $\boldsymbol{I}(\widetilde{\boldsymbol{\theta}})^{-1}$. In this case, only the observation-level gradient vectors $\boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})$ are required because of $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}}) = \sum_i \boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})$ and $\widehat{\boldsymbol{I}}(\widetilde{\boldsymbol{\theta}})^{-1} = \{\sum_i \boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})\boldsymbol{g}_i(\widetilde{\boldsymbol{\theta}})'\}^{-1}$.

3 Implementation in Stata

`lgrgtest` implements the above-described LM test in Stata based on its score version (1). `lgrgtest` performs a test of those restrictions that were imposed on the most recently fit model through specifying the option `constraints()`. `lgrgtest` hence directly draws on Stata's `constraint` command and the corresponding estimation option `constraints()`; see [R] **constraint**. `lgrgtest` will run only after estimation commands that implement constrained ML estimation, with a score vector stored in `e(gradient)` serving as indication for ML. The only exception to this is `cnsreg`, after which `lgrgtest` is also available as a postestimation command; see below for more details on how `lgrgtest` is implemented after `cnsreg`. Other special cases are `sem` and `gsem`. After these two commands, `lgrgtest` tests not only the restrictions that are imposed through the option `constraints()` but also all restrictions that are specified using the path notation (symbol `@`) of `sem` and `gsem`; see [SEM] **sem and gsem path notation**. Users must be aware that after `sem` and `gsem`, `lgrgtest` mechanically tests all constraints (on paths, variances, covariances, and means) that are manually specified in the estimation syntax. Depending on the specific model, this might result in a pointless attempt of testing constraints that are required for identification.

Because `constraint` allows only for defining linear constraints, `lgrgtest` is confined to implementing LM tests of linear restrictions imposed on the model parameters. The syntax for appropriately defining the constraints may be specific to the particular estimation command that precedes `lgrgtest`. All Stata commands that allow for the option `constraints()` store the constraints in `e(Cns)`, from which `lgrgtest` retrieves the information about the restrictions imposed. Yet, in creating `e(Cns)`, different commands may deal differently with constraints that are technically required, for example, exclusion restrictions for avoiding perfect collinearity. Nonetheless, `lgrgtest` will correctly identify the number of test degrees of freedom in almost any case. The option `df()` still allows for manually specifying that number.

After constrained ML estimation, Stata does not store $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}})$ in `e(gradient)` but rather the gradient vector of the constrained model evaluated at $\widetilde{\boldsymbol{\theta}}$, which is—provided that convergence is achieved—just a vector of (values close to) zeros. To obtain $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}})$, `lgrgtest` internally reexecutes the respective estimation command without the constraints imposed, chooses $\widetilde{\boldsymbol{\theta}}$ as starting values, and does not allow the optimization algorithm to iterate. This requires that the respective estimation command allows for the maximization options `from()` and `iterate()`; see [R] **Maximize**.[2] Changing the data between fitting the constrained model and using `lgrgtest` is not advisable. This could cause `lgrgtest` to fail and even lead to a misleading test result.

Reexecuting the respective estimation command with adjusted options further requires that `lgrgtest` interpret the command-line entry used for fitting the model. Hence, `lgrgtest` runs only if the full command-line entry is stored in `e(cmdline)`

2. `from()` and `iterate()` need to fully control the choice of starting values and the number of iterations. `fmm`, for instance, allows for the additional options `startvalues()` and `emopts()` that—besides `from()`—control the choice of starting values. Thus, `lgrgtest` runs only after `fmm` because this special case is explicitly considered by `lgrgtest`.

and the command name is stored in `e(cmd)` (or in `e(cmd2)`). Moreover, the estimation command generally needs to have a syntax that complies with Stata's standard syntax conventions; that is, it is structured into 1) the command name, 2) lists of variable names, 3) a comma, and 4) a list of options (compare Drukker [2015]). Multiple-equation estimation commands are compatible with `lgrgtest`, provided they use in 2) a syntax that either encloses equation-specific *varlist*s in parentheses `()` or separates them by `||`. For some selected Stata commands—`sem`, `gsem`, `nlogit`, and `fmm`, for instance—that have a more complex syntax, `lgrgtest` is still available as a postestimation command. Yet `lgrgtest` will not run after community-contributed commands that use an unconventional syntax. A list of compatible Stata estimation commands is provided in section 4.4.

`lgrgtest` uses `e(V)` from internally reexecuting the estimation command as the estimate of $\boldsymbol{I}(\widetilde{\boldsymbol{\theta}})^{-1}$. However, this is a valid approach only if *vcetype* is either `oim` or `opg`; see [R] ***vce_option***. Thus, if `e(vce)` differs from `oim` and `opg`, `lgrgtest` issues an error message and does not perform a test. Yet, by specifying the option `forcevce`, one can force `lgrgtest` to perform the test. With `forcevce`, provided that `e(vce)` is neither `oim` nor `opg`, `lgrgtest` uses the model-based variance[3] `e(V_modelbased)` instead of `e(V)`. Users of `lgrgtest` need to seriously think about whether this way of forcing the command into carrying out the test makes sense in the specific setting. If, for instance, a *vcetype* different from `oim` or `opg` is requested in the estimation syntax to account for some (possible) misspecification, the enforced LM test may suffer from severe size distortion.

Constrained linear least squares, which is implemented in Stata by the command `cnsreg`, is the ML estimator of the constrained linear model only if the errors are independent and identically normally distributed. If `lgrgtest` is used after `cnsreg`, `lgrgtest` issues a warning that independent and identically distributed normal errors are assumed and exploits that under this assumption, ML estimation of the linear model is equivalent to estimating a normal censored regression model with censoring limits $-\infty$ and $+\infty$. This can be implemented in Stata by using `tobit` with `ll()` and `ul()` left unspecified. That is, $\boldsymbol{g}(\widetilde{\boldsymbol{\theta}})$ and $\widehat{\boldsymbol{I}}(\widetilde{\boldsymbol{\theta}})^{-1}$ are obtained from evaluating the tobit log-likelihood function at the coefficient estimates that `cnsreg` yields. The estimated error variance is calculated as `e(rmse)^2*e(df_r)/e(N)` because the ML estimator does not involve a degrees-of-freedom correction. After `cnsreg`, `lgrgtest` expects `e(vce)` to be `ols`, which is translated to *vcetype* `oim` in the internal run of `tobit`. If the option `forcevce` is specified, `lgrgtest` switches to `e(V_modelbased)` for *vcetype*s different from `ols`.

4 The lgrgtest command

`lgrgtest` requires Stata 15 or higher. `lgrgtest` can be used only after estimation commands that allow for the option `constraints()`. With `cnsreg` being the only ex-

3. `e(V_modelbased)` will usually coincide with `e(V)` for *vcetype* `oim`, that is, with the observed sample Hessian of the log-likelihood function.

ception, the preceding estimation command also needs to allow for the maximization options `iterate()` and `from()` and needs to store the constraints and the score vector in the matrices `e(Cns)` and `e(gradient)`, respectively. Further technical requirements are that the *vcetype* be stored in `e(vce)`, the command name be stored in `e(cmd)` (or alternatively in `e(cmd2)`), and the full command-line entry be stored in `e(cmdline)`. `lgrgtest` generally runs only after estimation commands that have a syntax that complies with Stata's standard syntax conventions. `lgrgtest` can be used if the prefix commands `fmm` and `fp` are specified in the preceding estimation. With the option `forcevce`, this technically also applies to the prefix commands `bootstrap`, `jackknife`, and `svy`. However, because the option `forcevce` makes `lgrgtest` use `e(V_modelbased)`, applying `lgrgtest` probably makes little sense in such settings. The prefix commands `bayes`, `mfp`, `mi estimates`, `nestreg`, `rolling`, `statsby`, and `stepwise` preclude using `lgrgtest` in postestimation. Except for `quietly` and `noisily`, `lgrgtest` itself does not allow for prefix commands.

4.1 Syntax

The syntax for `lgrgtest` reads as follows:

`lgrgtest` [`,` <u>`not`</u>`est` <u>`nocnsr`</u>`eport` <u>`d`</u>`f(`#`)` <u>`noomit`</u>`ted` <u>`forcev`</u>`ce`]

The syntax for `lgrgtest` does not specify the restrictions to be tested. Rather, this is done in the preceding estimation syntax via specifying the option `constraints()`.

4.2 Options

`notest` prevents `lgrgtest` from displaying any output on the screen.

`nocnsreport` prevents `lgrgtest` from displaying the imposed constraints.

`df(`#`)` makes `lgrgtest` use a user-specified number of degrees of freedom, that is, a number of tested restrictions, in calculating the *p*-value for the test. The default is to use the rank of `e(Cns)`. If reexecution of the command without specified constraints also stores `e(Cns)`, which applies to models that involve automatically imposed constraints in addition to those specified in the command-line entry, the difference in ranks is used as the default. Specifying `df()` will rarely be required.

`noomitted` makes `lgrgtest` not consider omitted variables as exclusion restrictions to be tested. Because some estimation commands, `mlogit`, for instance, label exclusion restrictions specified by `constraints()` as omitted variables in `e(Cns)`, the default is to consider omitted variables as exclusion restrictions that are to be tested. Specifying `noomitted` will not affect the number of test degrees of freedom but only which restrictions are displayed.

`forcevce` makes `lgrgtest` perform the LM test even if *vcetype* is neither `oim` nor `opg` (not `ols` after `cnsreg`). Because `lgrgtest` estimates the inverse information matrix as `e(V)` obtained from reevaluating the model log-likelihood function at the restricted estimates, *vcetype*s other than `oim` and `opg` are most likely inappropriate. With the option `forcevce`, `lgrgtest` issues a warning and uses `e(V_modelbased)` as an estimate of the inverse information matrix. If *vcetype* is `oim` or `opg` (`ols` after `cnsreg`), specifying `forcevce` has no effect.

4.3 Stored results

`lgrgtest` stores the following in `r()`:

Scalars	
`r(p)`	p-value
`r(chi2)`	LM test statistic (χ^2)
`r(df)`	test constraints degrees of freedom
`r(rank)`	rank of `e(Cns)` adjusted for the number automatically imposed constraints (only stored if the option `df()` is specified)
Macros	
`r(modelbased)`	`modelbased` if `e(V_modelbased)` is used as estimate of the inverse information matrix

4.4 Compatible Stata estimation commands

This section lists official Stata commands that, according to a series of tests (Stata/SE 17.0, update level: 04 Oct 2022), allow for `lgrgtest` being used as a postestimation command:

[CM] `cmclogit`, `cmmixlogit`, `cmmprobit`, `cmroprobit`, `cmxtmixlogit`, `nlogit`; [DSGE] `dsge`; [ERM] `eintreg`, `xteinreg`, `eprobit`, `xteprobit`, `eregress`, `xteregress`; [FMM] `fmm`; [ME] `mecloglog`, `meglm`, `meintreg`, `melogit`, `menbreg`, `meologit`, `meoprobit`, `mepoisson`, `meprobit`, `mestreg`, `metobit`; [R] `betareg`, `binreg`, `clogit`, `cloglog`, `cnsreg`, `cpoisson`, `fracreg`, `frontier`, `glm`, `heckman`, `heckoprobit`, `heckpoisson`, `heckprobit`, `hetoprobit`, `hetprobit`, `hetregress`, `intreg`, `ivprobit`, `ivtobit`, `logistic`, `logit`, `mlexp`, `mlogit`, `mprobit`, `nbreg`, `gnbreg`, `ologit`, `oprobit`, `poisson`, `probit`, `scobit`, `slogit`, `tnbreg`, `tobit`, `tpoisson`, `truncreg`, `zinb`, `ziologit`, `zioprobit`, `zip`; [SEM] `sem`, `gsem`; [ST] `stcrreg`, `stintreg`, `streg`; [TE] `etpoisson`, `etregress`; [TS] `arch`, `arfima`, `arima`, `dfactor`, `mgarch ccc`, `mgarch dcc`, `mgarch dvech`, `mgarch vcc`, `sspace`, `ucm`; [XT] `xtcloglog`, `xtfrontier`, `xtheckman`, `xtintreg`, `xtlogit`, `xtmlogit`, `xtnbreg`, `xtologit`, `xtoprobit`, `xtpoisson`, `xtprobit`, `xttobit`

For each command, typically a single basic example (from the respective help entry) was tried. Hence, it may still not be possible to use `lgrgtest` as a postestimation command if some options are specified. A systematic test of compatibility with community-contributed Stata estimation commands was not carried out.

5 Illustrations and applications

This section illustrates the use of `lgrgtest` based on different applications. In the fashion of a replication study, subsection 5.2 links using `lgrgtest` to a serious econometric application (Egger et al. 2011b). The accompanying subsection 5.1 just sets the ground for this exercise by providing information required for understanding what the analysis in Egger et al. (2011b) is about. Subsection 5.3 is based on a simple example using one of Stata's example datasets. It is closely related to examples from the help files for `lgrgtest` and `mlogit`. The focus of subsection 5.3 is on illustrating the use of options available for `lgrgtest`.

5.1 Egger et al. (2011b) in a nutshell

The application below of `lgrgtest` partly replicates and uses data from Egger et al. (2011b). The replication data are published in Egger et al. (2011a) and are made available via the Inter-university Consortium for Political and Social Research. In their empirical analysis, Egger et al. (2011b) address the question of whether preferential trade agreements (PTA) positively affect bilateral trade flows. One focus of the econometric work is on PTA being most likely endogenous. That is, naïvely regressing exports from country i to country j on a dummy ($\texttt{pta}_{ij}$), indicating a common PTA that covers i and j, might be a heavily biased estimator for the causal effect of trade agreements on exports if common unobserved factors matter for both trade flows and the existence of such agreements.

Egger et al. (2011b) use country-level, cross-sectional (year 2005) data that consist of 15,750 country dyads. In our replication, we focus on one empirical model—among several—estimate in Egger et al. (2011b) that considers the extensive margin of bilateral trade. In that model, the dependent variable $\texttt{i}_{ij}$ is a binary dummy indicating that country i exports to country j. Because both the outcome variable $\texttt{i}_{ij}$ and the possibly endogenous regressor $\texttt{pta}_{ij}$ are binary, the endogeneity issue can be addressed through estimating a recursive bivariate probit model, provided one is willing to assume joint normality. Though the nonlinearity of the probit model alone would—in principle—give identification of the model parameters, Egger et al. (2011b) still impose exclusion restrictions on their preferred model. They argue that, conditional on further covariates, 1) sharing a past colonizer-colony relationship ($\texttt{colony}_{ij}$), 2) sharing a common past colonizer ($\texttt{comcol}_{ij}$), and 3) having a common history as one joint country ($\texttt{smctry}_{ij}$) might affect trade between two countries only through the existence of a mutual PTA and might hence be excluded from the equation explaining $\texttt{i}_{ij}$. The nonlinearity of the model provides—in addition to having more instruments than endogenous regressors—another margin for testing the validity of the exclusion restrictions because the model is technically identified even without these restrictions imposed. Consequently, one can directly test whether these three variables affect $\text{Prob}(\texttt{i}_{ij} = 1)$ beyond the effect that operates through $\texttt{pta}_{ij}$ (Egger et al. 2011b, 134). Egger et al. (2011b) base this test on estimating the unrestricted model. Yet the entire discussion in Egger et al. (2011b) focuses on the restricted model. Thus, basing this test on the preferred model specification may also be appealing. This is what we use `lgrgtest` for.

5.2 Replication and further statistical tests

After changing the current working directory to where the replication data (`data.dta`) are stored, we run the initial lines (a few lines redundant for the present purpose are dropped) of the replication code file (`bivprobtest.do`) provided in Egger et al. (2011a) under version control (Stata 10.1, the Stata version under which the code was originally run; see `README.pdf` from the replication package). These lines of code are intended to implement a test of the exclusion restrictions. The variables `_x_*` are 248 (pregenerated) exporter- and importer-country indicators.

```
. version 10.1 : {
. use data, clear
. global pcolfex "_x_59 _x_162"
. global z "dist bord lang cont durab polcomp autoc curcol"
. global instr "colony comcol smctry"
. gen const = 1
. drop $pcolfex
. quietly biprobit (pta = $z $instr const _x_*, nocons)
> (i = pta $z $instr const _x_*, nocons)
. test [i]colony [i]curcol [i]smctry
 ( 1)  [i]colony = 0
 ( 2)  [i]curcol = 0
 ( 3)  [i]smctry = 0
           chi2(  3) =    3.78
         Prob > chi2 =    0.2864
. }
```

Next, we use `lgrgtest` to implement an LM test of the same restrictions. As an initial step, this requires defining constraints using Stata's `constraint` command.

```
. constraint 1 [i]colony = 0
. constraint 2 [i]curcol = 0
. constraint 3 [i]smctry = 0
. quietly biprobit (pta = $z $instr const _x_*, nocons)
> (i = pta $z $instr const _x_*, nocons), constraints(1 2 3)
. lgrgtest
LM test of constraints(1 2 3)
 ( 1)   [i]colony = 0
 ( 2)   [i]curcol = 0
 ( 3)   [i]smctry = 0
        LM chi2(  3) =    3.79
         Prob > chi2 =    0.2856
```

`lgrgtest` yields a result very close to the one we got from `test`. Given the substantial sample size and the asymptotic equivalence of the Wald and LM tests, this does not come as a surprise. In economic terms, both tests are far from rejecting the null of `colony`$_{ij}$, `smctry`$_{ij}$, and `curcol`$_{ij}$ exerting no direct effect on `i`$_{ij}$, which warrants using them as instruments for the endogenous regressor `pta`$_{ij}$.[4]

Only to demonstrate that `lgrgtest` can be employed for testing cross-equation restriction and restrictions imposed on ancillary parameters—contentwise, the restrictions subject to the test make probably little sense—we estimate a specification in which the coefficient of `dist`$_{ij}$ is restricted to be the same in both equations and a cross-equation error correlation of 0.7 is imposed. Here we compare the LM test with the corresponding LR test, using `lrtest`. We base this exercise on a small model without exporter and importer fixed effects.

```
. scalar atanh07 = 0.5*ln((1+0.7)/(1-0.7))
. constraint 5 [i]dist - [pta]dist = 0
. constraint 6 [/]athrho = atanh07
. quietly biprobit (pta = $z $instr) (i = pta $z $instr), constraints(5 6)
. lgrgtest
LM test of constraints(5 6)
 ( 5)   - [pta]dist + [i]dist = 0
 ( 6)   [/]athrho = .8673005
        LM chi2( 2) =   16.70
         Prob > chi2 =    0.0002
. estimates store nofe_constrained
. quietly biprobit (pta = $z $instr) (i = pta $z $instr)
. estimates store nofe_unconstrained
. lrtest nofe_unconstrained nofe_constrained
Likelihood-ratio test
Assumption: nofe_constra~d nested within nofe_unconst~d
 LR chi2(2) =  17.63
Prob > chi2 = 0.0001
```

The results that `lgrgtest` and `lrtest` yield are almost the same. This is in line with the theoretical results of the LR and LM tests—in the absence of misspecification under the null—being asymptotically equivalent.

5.3 Further examples

To illustrate using `lgrgtest` after a command different from `biprobit` and to illustrate some options of `lgrgtest`, we make use of an example that is—subject to minor modifications—borrowed from the manual entry for `mlogit`; see [R] **mlogit** for more details. It draws on Tarlov et al. (1989) and Wells et al. (1989) and is, among other examples, also used in the help file for `lgrgtest`. From an economic perspective, the ex-

4. Though of no immediate relevance to illustrating the use of `lgrgtest`, note that `curcol`$_{ij}$, indicating a post-1945 colonizer-colony relationship, is not among the variables that actually serve as instruments in the specification for which results are reported in the article (see Egger et al. [2011b, 132, table 3, column 6]). In fact, the variables listed in the global macro `instr` are subject to exclusion restrictions. That is, `comcol`$_{ij}$, not `curcol`$_{ij}$, serves as an instrument. The actually imposed exclusion restrictions are clearly rejected, irrespective of whether a Wald test, implemented by `test`, or an LR test, implemented by `lgrgtest`, is used.

ample is concerned with the sociodemographic determinants of health insurance choice among 615 individuals in poor mental health. The outcome variable `insure` codes three choice options: an indemnity plan (`Indemnity`), a prepaid plan (`Prepaid`), and no health insurance at all (`Uninsure`). We impose a set of restrictions on the original multinomial logit model that effectively reduces it to a simple binary logit model with the two outcomes insured and uninsured. Subsequently, we use `lgrgtest` to test the simple model against the richer alternative.

```
. webuse sysdsn1, clear
(Health insurance data)
. constraint 53 [Prepaid]
. mlogit insure age male nonwhite i.site, constraint(53)
Iteration 0:  Log likelihood = -555.85446
Iteration 1:  Log likelihood = -550.37015
Iteration 2:  Log likelihood = -549.98844
Iteration 3:  Log likelihood =  -549.9878
Iteration 4:  Log likelihood =  -549.9878
Multinomial logistic regression                      Number of obs =    615
                                                     Wald chi2(5)  =   9.51
Log likelihood = -549.9878                           Prob > chi2   = 0.0904
 ( 1)  [Prepaid]o.age = 0
 ( 2)  [Prepaid]o.male = 0
 ( 3)  [Prepaid]o.nonwhite = 0
 ( 4)  [Prepaid]2o.site = 0
 ( 5)  [Prepaid]3o.site = 0
------------------------------------------------------------------------------
      insure | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
Indemnity    |  (base outcome)
-------------+----------------------------------------------------------------
Prepaid      |
         age |          0  (omitted)
        male |          0  (omitted)
    nonwhite |          0  (omitted)
             |
        site |
          2  |          0  (omitted)
          3  |          0  (omitted)
             |
       _cons |  -.0561551   .0838038    -0.67   0.503    -.2204075    .1080973
-------------+----------------------------------------------------------------
Uninsure     |
         age |  -.0023922   .0110472    -0.22   0.829    -.0240443    .0192598
        male |   .1777916   .3511016     0.51   0.613    -.5103549    .8659382
    nonwhite |  -.2732876   .4021485    -0.68   0.497    -1.061484    .5149089
             |
        site |
          2  |  -1.260906   .4582332    -2.75   0.006    -2.159027   -.3627856
          3  |    .059544   .3497673     0.17   0.865    -.6259873    .7450753
             |
       _cons |  -1.453019   .5715737    -2.54   0.011    -2.573282   -.3327548
------------------------------------------------------------------------------
```

```
. lgrgtest
LM test of constraints(53)
 ( 1)   [Prepaid]o.age = 0
 ( 2)   [Prepaid]o.male = 0
 ( 3)   [Prepaid]o.nonwhite = 0
 ( 4)   [Prepaid]2bno.site = 0
 ( 5)   [Prepaid]3o.site = 0
        LM chi2(  5) =   30.43
         Prob > chi2 =    0.0000
```

The LM test rejects the simple binary choice model. To illustrate the `lgrgtest` option `forcevce`, we rerun `mlogit` and request robust standard-error estimation by specifying the option `robust`. Subsequently, we call `lgrgtest` without and with `forcevce` specified.

```
. quietly mlogit insure age male nonwhite i.site, constraint(53) robust
. capture noisily lgrgtest
vcetype oim or opg expected; specify option forcevce to switch to
  e(V_modelbased)
. lgrgtest, forcevce
vcetype robust not allowed; switched to e(V_modelbased)
LM test of constraints(53)
 ( 1)   [Prepaid]o.age = 0
 ( 2)   [Prepaid]o.male = 0
 ( 3)   [Prepaid]o.nonwhite = 0
 ( 4)   [Prepaid]2bno.site = 0
 ( 5)   [Prepaid]3o.site = 0
        LM chi2(  5) =   30.43
         Prob > chi2 =    0.0000
```

Without `forcevce`, `lgrgtest` denies carrying out the test because `oim` or `opg` is expected as *vcetype* but not `robust`. With `forcevce`, `lgrgtest` informs us about `using e(V_modelbased)` for calculating the LM statistic. Indeed, the test result is exactly the same as the one we got above. This just reflects that `e(V_modelbased)` coincides with `e(V)` from `mlogit` with the default *vcetype* `oim`.

Finally, we illustrate the option `noomitted`. For this purpose, we generate the variable `agecollin`, which is perfectly collinear with the explanatory variable `age`, and add it as an additional right-hand-side variable to the model. `mlogit`, like any Stata estimation command, just drops perfectly collinear regressors from the model. Yet this exercise is to illustrate how `lgrgtest` deals with this situation.

```
. generate agecollin = 2*age
(1 missing value generated)
. quietly mlogit insure age male nonwhite i.site agecollin, constraint(53)
. lgrgtest
LM test of constraints(53)
 ( 1)   [Prepaid]o.agecollin = 0
 ( 2)   [Uninsure]o.agecollin = 0
 ( 3)   [Prepaid]o.age = 0
 ( 4)   [Prepaid]o.male = 0
 ( 5)   [Prepaid]o.nonwhite = 0
 ( 6)   [Prepaid]2bno.site = 0
 ( 7)   [Prepaid]3o.site = 0
        LM chi2(  5) =   30.43
         Prob > chi2 =    0.0000
. lgrgtest, noomitted
LM test of constraints(53)
        LM chi2(  5) =   30.43
         Prob > chi2 =    0.0000
```

The value of the LM statistic remains completely unaffected. Likewise, the *p*-value does not change and is still calculated using the correct number of test degrees of freedom. Yet, somewhat confusingly, not 5 but 7 restrictions are displayed. This is because exclusion restrictions automatically imposed to deal with perfect collinearity are also stored in `e(Cns)`. The option `noomitted` is intended to fix this issue. However, it does not succeed in this example but makes `lgrgtest` display no constraints at all. This is due to `mlogit` labeling variables as omitted in `e(Cns)`, irrespective of whether they are automatically dropped because of collinearity or intentionally excluded by specifying `constraints()`. To illustrate that different Stata commands deal differently with automatically imposed restrictions in terms of what is stored in `e(Cns)`, we estimate the constrained multinomial logit using `gsem, mlogit` instead of using `mlogit`; see [SEM] **Example 37g**.

```
. quietly gsem (2.insure <- age@0 male@0 nonwhite@0 i.site@0 agecollin@0)
> (3.insure <- age male nonwhite i.site agecollin), mlogit
. lgrgtest, noomitted
LM test of constraints()
 ( 1)   [2.insure]age = 0
 ( 2)   [2.insure]male = 0
 ( 3)   [2.insure]nonwhite = 0
 ( 4)   [2.insure]2bn.site = 0
 ( 5)   [2.insure]3.site = 0
        LM chi2(  5) =   30.43
         Prob > chi2 =    0.0000
```

The restrictions to be tested are imposed using `gsem`'s path notion instead of specifying the option `constraints()`. Yet, in terms of model estimation, `gsem` does exactly the same as `mlogit` did above. Nevertheless, the option `noomitted` makes `lgrgtest` behave slightly differently after `gsem` than after `mlogit`. After `gsem`, the actually tested restrictions are displayed. This works because `gsem`, unlike `mlogit`, does not label manually imposed exclusion restriction as omitted variables in `e(Cns)`.

6 Conclusions

In this article, we introduced the postestimation command `lgrgtest`, which provides a convenient way to employ an LM test for testing the constraints imposed on a model previously estimated by ML. It complements the Stata commands `test` and `lrtest`, which offer the Wald test and the LR test, respectively, to be used for testing these restrictions. `lgrgtest` is an alternative to the existing community-contributed command `boottest` and its accompanying wrapper `scoretest` (Roodman et al. 2019). Because the three tests may behave differently in finite samples, having them all available appears to be valuable for Stata users. The Stata implementation of the LM test is, however, subject to one major limitation, which it shares with `scoretest`. `lgrgtest` draws on Stata's `constraint` command and the accompanying option `constraints()`, which allows only for imposing linear restrictions on a model. Consequently, though the Lagrange multiplier test in theory is well suited for testing genuinely nonlinear constraints, such tests cannot be implemented in Stata using `lgrgtest`. Another limitation, which also applies to `scoretest`, is that `lgrgtest`, albeit applicable after numerous Stata estimation commands, cannot be used as a postestimation command after `ml maximize`. This is because the syntax of `ml` and what is stored in `e()` substantially deviate from the standard for regular Stata estimation commands. Hence, making `lgrgtest` available after `ml maximize` may be a major future update of the command.

7 Acknowledgments

I would like to thank Michael Oberfichtner and Irina Simankova for many valuable comments and suggestions. Research assistance from Irina Iwanow is gratefully acknowledged. When this article was in proof, David Roodman brought to my attention via Statalist that `boottest` (Roodman et al. 2019) also has Lagrange multiplier test functionality.

8 Programs and supplemental materials

To install a snapshot of the corresponding software files as they existed at the time of publication of this article, type

```
. net sj 23-2
. net install st0712      (to install program files, if available)
. net get st0712          (to install ancillary files, if available)
```

9 References

Aitchison, J., and S. D. Silvey. 1958. Maximum-likelihood estimation of parameters subject to restraints. *Annals of Mathematical Statistics* 29: 813–828. https://doi.org/10.1214/aoms/1177706538.

Arellano, M. 2004. Lagrange multiplier test. In *An Eponymous Dictionary of Economics: A Guide to Laws and Theorems Named After Economists*, ed. J. Segura and C. Rodríguez Braun, 144–146. Cheltenham, U.K.: Edward Elgar Publishing.

Bera, A. K., and Y. Bilias. 2001. Rao's score, Neyman's C(α) and Silvey's LM tests: An essay on historical developments and some new results. *Journal of Statistical Planning and Inference* 97: 9–44. https://doi.org/10.1016/S0378-3758(00)00343-8.

Buse, A. 1982. The likelihood ratio, Wald, and Lagrange multiplier tests: An expository note. *American Statistician* 36: 153–157. https://doi.org/10.2307/2683166.

Drukker, D. M. 2015. Programming estimators in Stata: Why you should. The Stata Blog: Not Elsewhere Classified. http://blog.stata.com/2015/10/020/programming-estimators-in-stata-why-you-should/.

Egger, P., M. Larch, K. E. Staub, and R. Winkelmann. 2011a. Replication data for: The trade effects of endogenous preferential trade agreements. Ann Arbor, MI: Inter-university Consortium for Political and Social Research. https://doi.org/10.3886/E114762V1.

———. 2011b. The trade effects of endogenous preferential trade agreements. *American Economic Journal: Economic Policy* 3: 113–143. https://doi.org/10.1257/pol.3.3.113.

Engle, R. F. 1984. Wald, likelihood ratio, and Lagrange multiplier tests in econometrics. In Vol. 2 of *Handbook of Econometrics*, ed. Z. Griliches and M. D. Intriligator, 775–826. Amsterdam: Elsevier. https://doi.org/10.1016/S1573-4412(84)02005-5.

Greene, W. H. 2018. *Econometric Analysis*. 8th ed. New York: Pearson.

Rao, C. R. 1948. Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. *Mathematical Proceedings of the Cambridge Philosophical Society* 44: 50–57. https://doi.org/10.1017/S0305004100023987.

Roodman, D., M. Ø. Nielsen, J. G. MacKinnon, and M. D. Webb. 2019. Fast and wild: Bootstrap inference in Stata using boottest. *Stata Journal* 19: 4–60. https://doi.org/10.1177/1536867X19830877.

Silvey, S. D. 1959. The Lagrangian multiplier test. *Annals of Mathematical Statistics* 30: 389–407. https://doi.org/10.1214/aoms/1177706259.

Tarlov, A. R., J. E. Ware, Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study: An application of methods for monitoring the results of medical care. Research Note No. N-3038-RWJ/HJK/PMT, RAND Corporation, Santa Monica, CA. https://www.rand.org/content/dam/rand/pubs/notes/2009/N3038.pdf.

Wells, K. B., R. D. Hays, M. A. Burnam, W. Rogers, S. Greenfield, and J. E. Ware, Jr. 1989. Detection of depressive disorder for patients receiving prepaid or fee-for-service care. Results from the Medical Outcomes Study. *Journal of the American Medical Association* 262: 3298–3302. https://doi.org/10.1001/jama.1989.03430230083030.

Wooldridge, J. M. 2010. *Econometric Analysis of Cross Section and Panel Data*. 2nd ed. Cambridge, MA: MIT Press.

About the author

Harald Tauchmann is a professor of health economics at the Friedrich-Alexander-Universität Erlangen-Nürnberg, a research fellow at RWI—Leibniz Institut für Wirtschaftsforschung in Essen, Germany, and research fellow at CINCH—Health Economics Research Center in Essen, Germany. His research interests include health economics, applied econometrics, and statistical methods.