The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–34

Weighted-average least squares: Improvements and extensions

Giuseppe De Luca
University of Palermo
Palermo, Italy
giuseppe.deluca@unipa.it

Jan R. Magnus
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
jan@janmagnus.nl

Abstract. This paper presents version 3.0 of the `wals` command, which implements the weighted-average least squares estimator of Magnus et al. (2010, *Journal of Econometrics* 154, 139–153). Version 3.0 improves earlier versions of `wals` in several respects: a new syntax supporting factor variables, time-series operators, and weights; an enlarged set of prior distributions; extended quadrature methods for computing the posterior mean; new plug-in estimates of the sampling moments; simulation-based confidence intervals; and other options to control accuracy, computational speed, and output of `wals`. We also offer three new post-estimation commands: the `predict` command associated with `wals`; the `lcwals` command for estimating linear combinations of the parameters; and the `margwals` command for estimating smooth, possibly nonlinear, functions of the parameters at given values of regressors. Finally, we compare our new commands with two suites of Stata commands for tackling issues of model uncertainty.

Keywords: st0001, `wals`, `predict`, `lcwals`, `margwals`, weighted-average least squares, linear model, Bayesian shrinkage, inference, post-estimation

1 Introduction

In recent years, Stata has made a substantial effort in implementing specialized model selection and model averaging approaches to tackle issues of model uncertainty. Notably, Stata 16 introduced the suite of LASSO (least absolute shrinkage and selection operator) commands, while Stata 18 introduced the suite of BMA (Bayesian model averaging) commands. These developments, along with other user-contributed commands, have significantly enriched the toolkit of statistical and econometric procedures that account for model uncertainty, promoting their empirical application in various disciplines.

This paper contributes to this ongoing trend by focusing on the weighted-average least squares (WALS) approach of Magnus et al. (2010). WALS is a frequentist model averaging method designed to address uncertainty in the choice of regressors of a linear model. After a preliminary semi-orthogonal transformation of the regressors, the approach builds on a Bayesian analysis of the normal location model. Since its introduction in 2010, the WALS approach has been studied, extended, and applied in many papers (see, e.g., Magnus et al. 2011; De Luca and Magnus 2011; Dardanoni et al. 2011; Seya and Tsutsumi 2012; Kumar and Magnus 2013; Magnus and Wang 2014; Magnus and De Luca 2016; Magnus et al. 2016; De Luca et al. 2018, 2022, 2023, 2025; Duval et al. 2021; Magkonis et al. 2021; and Picchio and Ubaldi 2023). WALS is attractive,

because it performs well in finite samples, offers a transparent notion of prior ignorance, and is not restricted to sequences of nested models. Equally important, it is numerically stable and fast to compute due to the preliminary semi-orthogonal transformation of the regressors.

Versions 1.0 and 2.0 of the `wals` command were developed by De Luca and Magnus (2011) and Magnus and De Luca (2016), but these routines are now obsolete and do not reflect the latest advancements in the WALS theory, such as issues related to the semi-orthogonal transformation, the choice of the prior distribution, the frequentist properties of the WALS estimator in repeated samples, its finite-sample and asymptotic distributions, and its confidence and prediction intervals. This justifies the new version 3.0 of `wals`, which improves and extends earlier versions of this command in various respects:

1. a new syntax supporting factor variables, time-series operators, and frequency, analytic, and importance weights;
2. an enlarged set of priors for the Bayesian shrinkage step of the WALS estimator;
3. extended quadrature methods for computing the posterior means;
4. new plug-in estimates of the sampling moments (bias, standard error, and root mean squared error) of the WALS estimator;
5. simulation-based confidence intervals that account for the finite-sample bias and the possibly nonnormal distribution of the WALS estimator;
6. additional options for controlling the accuracy, computation speed, and output of the `wals` command.

In addition to these extensions, we offer three new post-estimation commands: the `predict` command associated with `wals`; the `lcwals` command, designed for WALS estimation of linear combinations of the parameters; and the `margwals` command, designed for WALS estimation of smooth, possibly nonlinear, real-valued functions of the parameters at given values of the regressors.

The remainder of the paper is organized as follows. Section 2 provides an overview of the WALS approach to linear regression models with i.i.d. errors. Section 3 details the syntax and various options of `wals` version 3.0. Section 4 outlines the stored results, while Section 5 describes the new post-estimation commands. Section 6 presents comparisons of WALS with LASSO and BMA. Finally, Section 7 offers concluding remarks.

2 The WALS approach for linear models

Our framework is the homoskedastic linear model

$$y = X_1\beta_1 + X_2\beta_2 + \epsilon, \tag{1}$$

where $X = (X_1 : X_2)$ is a nonstochastic $n \times k$ matrix of full column-rank $k = k_1 + k_2 < n$, ϵ is an $n \times 1$ vector of independent and identically distributed disturbances which we assume to be distributed as $\mathrm{N}(0, \sigma_\epsilon^2 I_n)$, where $0 < \sigma_\epsilon^2 < \infty$ and I_n denotes the identity matrix of order n. The normality assumption plays a role in finite samples, but it can be relaxed in the asymptotic theory.

The reason to distinguish between two sets of regressors is that X_1 contains the k_1 *focus* regressors which we want in the model on theoretical or other grounds, while X_2 contains the k_2 *auxiliary* regressors of which we are less certain. Similarly, β_1 and β_2 are called the focus and auxiliary parameters. The focus regressors can be those of which we wish to study the causal effects on y or those whose presence in the model is well established from previous studies, while the auxiliary regressors could be controls that are added to avoid omitted variable bias or transformations and interactions of a set of original regressors, such as indicator variables, polynomials, and splines. Model (1) also includes as a special case the usual BMA setup where the constant term is a focus regressor and all other regressors are auxiliary regressors. More generally, we assume that $k_1 \geq 0$ and $k_2 \geq 1$. The case $k_2 = 0$ is excluded because it implies that there is no uncertainty, so that model (1) can be estimated by ordinary least squares (LS, `regress`).

In WALS, we first apply the following one-to-one transformations of X_2 and β_2:

$$Z_2 = X_2 \Delta_2 \Psi^{-1/2}, \qquad \gamma_2 = \Psi^{1/2} \Delta_2^{-1} \beta_2, \tag{2}$$

where Δ_2 is a diagonal $k_2 \times k_2$ matrix such that the diagonal elements of the positive definite matrix $\Psi = \Delta_2 X_2' M_1 X_2 \Delta_2 / n$ are all equal to one, $\Psi^{1/2}$ is the unique square root of Ψ, and $M_1 = I_n - X_1 (X_1' X_1)^{-1} X_1'$. Next, we rescale X_1 and β_1 using:

$$Z_1 = X_1 \Delta_1, \qquad \gamma_1 = \Delta_1^{-1} \beta_1, \tag{3}$$

where Δ_1 is a diagonal $k_1 \times k_1$ matrix such that the diagonal elements of $Z_1' Z_1 / n$ are all equal to one. Since $Z_1 \gamma_1 = X_1 \beta_1$ and $Z_2 \gamma_2 = X_2 \beta_2$, we can rewrite model (1) as

$$y = Z_1 \gamma_1 + Z_2 \gamma_2 + \epsilon, \tag{4}$$

where $Z_2' M_1 Z_2 / n = I_{k_2}$. The transformations in (2) ensure that the k_2 components of the LS estimator of γ_2 in model (4) are independent, while the rescaling in (3) serves only to increase the numerical accuracy of the inversion and eigenvalue routines.

The WALS estimators of γ_1 and γ_2 are obtained by averaging the LS estimators $\hat{\gamma}_{1(j)}$ and $\hat{\gamma}_{2(j)}$ resulting from the $J = 2^{k_2}$ models that contain all focus regressors and a subset of the auxiliary regressors in (4):

$$\tilde{\gamma}_1 = \sum_{j=1}^{J} w_j \hat{\gamma}_{1(j)}, \qquad \tilde{\gamma}_2 = \sum_{j=1}^{J} w_j \hat{\gamma}_{2(j)}, \tag{5}$$

where the w_j are nonnegative model weights that depend only on $M_1 y$ and add up to one. Given $\tilde{\gamma}_1$ and $\tilde{\gamma}_2$, the WALS estimators of β_1 and β_2 in model (1) are:

$$\tilde{\beta}_1 = \Delta_1 \tilde{\gamma}_1, \qquad \tilde{\beta}_2 = \Delta_2 \Psi^{-1/2} \tilde{\gamma}_2. \tag{6}$$

To gain additional insights on these estimators we rewrite $\tilde{\gamma}_1$ and $\tilde{\gamma}_2$ as:

$$\tilde{\gamma}_1 = \hat{\gamma}_{1r} - Q\tilde{\gamma}_2, \qquad \tilde{\gamma}_2 = \Lambda\hat{\gamma}_{2u}, \tag{7}$$

where $\hat{\gamma}_{1r} = (Z_1'Z_1)^{-1}Z_1'y$ is the LS estimator of γ_1 in the fully restricted model (with $\gamma_2 = 0$), $\hat{\gamma}_{2u} = Z_2'M_1y/n$ is the LS estimator of γ_2 in the unrestricted model, $Q = (Z_1'Z_1)^{-1}Z_1'Z_2$, $\Lambda = \sum_{j=1}^{J} w_j W_j$, $W_j = I_{k_2} - S_jS_j'$ is a diagonal matrix whose diagonal elements are equal to zero or one, and S_j is a $k_2 \times r_j$ selection matrix of rank $0 \leq r_j \leq k_2$ representing the r_j exclusion restrictions implied by model j, that is, $S_j' = (I_{r_j} : 0)$ or a column-permutation thereof.

The *Equivalence Theorem* of Magnus and Durbin (1999) shows that the mean squared error (MSE) matrix of $\tilde{\gamma}_1$ depends on the MSE matrix of $\tilde{\gamma}_2$, so attention can be restricted to the estimation of γ_2. The k_2 diagonal elements λ_h of Λ, called the *WALS weights*, are partial sums the model weights w_j. Specifically, we have

$$\lambda_h = \sum_{j \in \mathcal{M}_h} w_j, \tag{8}$$

where $\mathcal{M}_h$ is the subspace of 2^{k_2-1} models that contain the hth auxiliary regressor in (4). It follows that $0 \leq \lambda_h \leq 1$ and hence the k_2 components of $\tilde{\gamma}_2$ in (7) are shrinkage estimators of the components of γ_2. Like the posterior inclusion probabilities in the context of BMA, each λ_h measures the importance of including the hth auxiliary regressor in the transformed model (4). There are three useful benchmarks for the λ_h. The WALS estimator coincides with the LS estimator of the fully restricted model if all λ_h are equal to zero, it coincides with the LS estimator of the unrestricted model if all λ_h are equal to one, and it becomes a simple average (i.e., equal w_j) of the LS estimators from the 2^{k_2} possible models if all λ_h are equal to 1/2.

Since $Z_2'M_1Z_2/n = I_{k_2}$, we also know that the k_2 components $(\hat{\gamma}_{2u})_h$ of $\hat{\gamma}_{2u}$ are distributed independently as

$$(\hat{\gamma}_{2u})_h \sim \mathrm{N}\left(\gamma_{2h}, \sigma_\epsilon^2/n\right), \tag{9}$$

where γ_{2h} denotes the hth component of γ_2.[1] Thus, if we further restrict each λ_h to depend only on $(\hat{\gamma}_{2u})_h$, then the components of $\tilde{\gamma}_2$ are also independent and our k_2-dimensional estimation problem for γ_2 reduces to k_2 identical one-dimensional problems of the following type: given a single observation $x \sim \mathrm{N}\left(\theta, \sigma_\epsilon^2/n\right)$, find a 'good' shrinkage estimator of the location parameter θ. This stylized setting is known as the *normal location problem* (NLP). Since risk properties of shrinkage estimators are little affected by the estimation of the variance parameter (see, e.g., Danilov 2005), we also assume that σ_ϵ^2 is known. An equivalent representation of the NLP is $x^* \sim \mathrm{N}(\theta^*, 1)$, where, with σ_ϵ^2 known, $x^* = \sqrt{n}x/\sigma_\epsilon$ is the t-ratio for testing the hypothesis $\theta = 0$ and $\theta^* = \sqrt{n}\theta/\sigma_\epsilon$ is the associated 'population t-ratio'.

An obvious estimator of θ in the NLP is x itself, which is unbiased and consistent, and is sometimes called the *usual* estimator. Another estimator, sometimes called the

1. Relaxing normality of ϵ, (9) holds asymptotically under the conditions of the central limit theorem.

silly estimator, is 0 (zero). Since $\text{MSE}(x) = \sigma_\epsilon^2/n$ and $\text{MSE}(0) = \theta^2$ we prefer the silly estimator if and only if $|\theta| < \sigma_\epsilon/\sqrt{n}$ or, equivalently, $|\theta^*| < 1$. No estimator dominates x uniformly in the MSE sense. However, for θ sufficiently close to zero, shrinkage estimators of the form $m(x) = \lambda(x)x$ with $0 \leq \lambda(x) \leq 1$ may achieve substantial MSE improvements by trading some bias with a reduction of the variance. The closeness of θ to zero should be evaluated relative to the standard deviation $\sigma_\epsilon/\sqrt{n}$ of x. For the silly estimator, where $\lambda(x) = 0$, we obtain a lower MSE for $|\theta^*| < 1$. More generally, the region of the parameter space where $\text{MSE}(m(x)) < \text{MSE}(x)$ depends on the population t-ratio θ^* and the shrinkage function $\lambda(x)$.

Based on theoretical considerations concerning admissibility, bounded risk, robustness, and optimality in terms of minimax regret, we adopt a Bayesian shrinkage approach that places a proper prior π on the population t-ratio θ^*. Under quadratic loss, the Bayes estimator of θ^* is the posterior mean $m^* = \mathbb{E}(\theta^*|x^*)$ and the implied estimator of θ is $m = \sigma_\epsilon m^*/\sqrt{n}$. To ensure that m enjoys attractive sampling properties as a estimator of θ, we require that the prior π on θ^* satisfies the conditions:

(C1) π is symmetric around zero;
(C2) π is positive and nonincreasing on $(0, \infty)$;
(C3) π is differentiable, except possibly at 0; and
(C4) $\psi(\theta) = -\pi'(\theta)/\pi(\theta) \to \psi_0$ as $\theta \to \infty$, where $\psi_0 \geq 0$ is some finite constant.

As discussed in De Luca et al. (2025), (C1)–(C3) are mild regularity conditions, while (C4) ensures that the prior π has bounded influence on m^*. If $\psi_0 = 0$, (C4) also implies the stronger property of *Bayesian robustness*: $x^* - m^* \to 0$ as $x^* \to \infty$, so that prior information is essentially ignored when the data information is sufficiently strong.

Let $\Gamma(u)$ and $B(u, v)$ denote, respectively, the gamma and beta functions. Examples of priors satisfying (C1)–(C4) are the class of reflected gamma-type priors

$$\pi(\theta) = \frac{bc^{(1-a)/b}}{2\Gamma((1-a)/b)}|\theta|^{-a}e^{-c|\theta|^b} \qquad (0 \leq a < 1,\, 0 < b \leq 1,\, c > 0), \tag{10}$$

the class of the reflected beta-type priors

$$\pi(\theta) = \frac{c^{1/b}b}{2\,B(1/a - 1/b, 1/b)}(1 + c|\theta|^b)^{-1/a} \qquad (0 < a < b,\, c > 0), \tag{11}$$

the horseshoe prior (Carvalho et al. 2010)

$$\pi(\theta; c) = \frac{c\sqrt{2}}{\pi^{3/2}} \int_0^\infty \frac{e^{-t}}{\theta^2 + 2c^2 t}\, dt \qquad (c > 0), \tag{12}$$

and the log prior, also known as horseshoe-like prior (Bhadra et al. 2017),

$$\pi(\theta) = \frac{1}{2\pi c} \log(1 + c^2/\theta^2) \qquad (c > 0). \tag{13}$$

As special cases of (10), we consider the Laplace ($a = 0, b = 1$), Weibull ($a + b = 1$), and Subbotin ($a = 0$) priors. The normal prior ($a = 0, b = 2$) is excluded because it violates

(C4). As special cases of (11), we consider instead the Pareto ($b = 1$) and the Student prior with d degrees of freedom ($a = 2/(d+1), b = 2, c = 1/d$). Our prior densities are also required to satisfy a notion of ignorance which we call *neutrality*, namely not knowing whether the silly estimator of θ has a lower MSE than the usual estimator, that is, whether $|\theta^*| < 1$:

(C5) π satisfies $\int_0^1 \pi(\theta)\, d\theta = \int_1^\infty \pi(\theta)\, d\theta = 1/4$.

Together with (C1), condition (C5) fixes one free prior parameter to ensure that the events $|\theta^*| < 1$ and $|\theta^*| > 1$ are equally likely a priori. The neutral Laplace and Weibull priors are obtained for $b = \log(2)$, the neutral Pareto prior for $c = 2^{a/(1-a)} - 1$, and the neutral Student prior for $d = 1$ (the Cauchy distribution). Neutrality conditions for the Subbotin, horseshoe and log priors are solved numerically. For the Subbotin, Weibull, and Pareto priors, we also fix the other free prior parameter by the minimax regret criterion for m^*, where regret is defined as the difference between the MSE of m^* and its lower bound (Magnus 2002). Table 1 shows the values of a, b, and c resulting from the neutrality condition and the minimax regret criterion.

Table 1: Prior parameters, neutrality, and minimax regret

Class	Prior	a	b	c	$\theta^*_{\max}$	$r(\theta^*_{\max})$
Reflected gamma-type	Laplace	**0.0**	**1.0**	0.6931	4.9312	0.5127
	Subbotin	**0.0**	0.7994	0.9378	3.6920	0.4697
	Weibull	**0.1126**	0.8874	0.6931	3.5625	0.4546
Reflected beta-type	Pareto	0.0862	**1.0**	0.0676	4.0313	0.4959
	Cauchy	**1.0**	**2.0**	**1.0**	3.5354	0.6332
Horseshoe-type	log	—	—	2.4458	3.1859	0.5975
	horseshoe	—	—	1.7329	3.0879	0.6043

Note: $\theta^*_{\max}$ and $r(\theta^*_{\max})$ denote, respectively, the argmax and the maximum regret. Pre-assigned parameter values are written in boldface, while parameter values resulting from the neutrality condition and the minimax regret criterion are written in normal face.

The Bayesian approach is used to construct our shrinkage estimators m^* and m of θ^* and θ, but the properties of such estimators are then assessed in a classical frequentist setup. De Luca et al. (2022) studied the bias and variance of m^* in repeated samples and proposed the plug-in maximum likelihood (ML) and double-shrinkage (DS) estimators of its sampling moments. More recently, De Luca et al. (2025) derived the finite-sample distribution of $m^* - \theta^*$. In the asymptotic theory, they also proved the uniform $\sqrt{n}$-consistency of m and obtained its asymptotic distribution. This estimator represents our estimator of the hth component γ_{2h} of γ_2 in model (4), and its sampling properties carry over, more or less straightforwardly, to the WALS estimators $\tilde{\gamma} = (\tilde{\gamma}_1', \tilde{\gamma}_2')'$ of $\gamma = (\gamma_1', \gamma_2')'$ in (5) and the WALS estimators $\tilde{\beta} = (\tilde{\beta}_1', \tilde{\beta}_2')$ of $\beta = (\beta_1', \beta_2')'$ in (6). Specifically, under (C1)–(C4), the finite-sample distribution of the WALS estimator is generally nonnormal and the choice of prior may have sizeable effects on the estimation bias. In large samples, these estimators are uniformly $\sqrt{n}$-consistent and their asymptotic distribution is normal only under certain conditions on β_2. The issue of inference has been addressed by De Luca et al. (2023), who proposed a simulation method for ob-

taining re-centered and asymmetric WALS confidence and prediction intervals based on the bias-corrected posterior mean. Although uniformly consistent estimators of the sampling distribution of shrinkage-type estimators may not exist (see Lemmas 3.1 and 3.5 in Leeb and Pötscher 2006), the Monte Carlo (MC) simulations of De Luca et al. (2023) suggest that coverage errors of WALS intervals are small.

3 The `wals` command

`wals` provides the WALS estimates of a linear regression model with i.i.d. errors. The earliest version of Stata that supports this command is version 14.0. It requires installing the Mata library `lscrhalton.mlib` (scrambled Halton sequences, st0244) from the web archive of the Stata Journal and the Mata library `lwalsgaussint.mlib` described in Section 3.3.[2] The new syntax of `wals` is:

`wals` *depvar* [(*focvars*)] *auxvars* [*if*] [*in*] [*weight*], [*wals_options*]

where *depvar* is the dependent variable, (*focvars*) is the list of focus regressors, *auxvars* is the list of auxiliary regressors, and *wals_options* are the options listed in Table 2. Although not shown in the syntax diagram, *auxvars* can be omitted only if one specifies the `auxconstant` option. Below, we describe the syntax and options of `wals` under the following headings: model setup, choice of the prior, computing the posterior mean, estimating sampling moments, confidence intervals, and reporting.

3.1 Options: model setup

The *varlist* of focus regressors in (*focvars*) must be enclosed within curved parentheses and, by default, it includes the constant term. The *varlist* of auxiliary regressors in *auxvars* must be specified without parentheses. The new version 3.0 of `wals` also supports factor variables, time-series operators, and three of the four types of weights available in Stata. Specifically, (*focvars*) and *auxvars* may contain factor variables and time-series operators, while *depvar* may only contain time-series operators. Weights can be specified as [*wtype*=*wvar*], where *wtype* denotes the type of weights (`fweights`, `aweights`, or `iweights`) and *wvar* is the weighting variable.[3] `pweights` are not allowed due to the lack of robust standard errors. Default features of the model setup can be changed using the following options:

`auxconstant` specifies that the constant term should be treated as an auxiliary regressor rather than as a focus regressor.

2. The Mata library `lwalsgaussint.mlib` is included in the `wals` package. To install the Mata library `lscrhalton.mlib`, you can type `net sj 12-1` and `net install st0244`.
3. Let n be the sample size and let D be an $n \times n$ diagonal matrix whose diagonal elements are the observations of *wvar*. In a weighted WALS regression, all objects like $X_p'y$, $Z_p'y$, $X_p'X_q$, and $Z_p'Z_q$ are replaced by $X_p'Dy$, $Z_p'Dy$, $X_p'DX_q$, and $Z_p'DZ_q$ $(p, q = 1, 2)$, respectively. `iweights` are treated like `fweights` (i.e., they are not normalized), but they are not restricted to be integers. `aweights` are instead normalized by the mean of the weights over the estimation sample.

Table 2: Basic options of the `wals` command

wals_options	Description
Model setup	
`auxconstant`	treat the constant term as an auxiliary regressor, rather than as a focus regressor (default)
`noconstant`	remove the constant term from the model, default includes a constant term
`sigma(#)`	fix the standard deviation of *depvar*, default uses the LS estimate from the unrestricted model
Choice of the prior	
`prior(`*priorname*`)`	specify the prior distribution, default is `pareto`
Computing the posterior mean	
`quadmethod(`*qmetname*`)`	define the integration method, default is `gauss`
`quadnpts(#)`	set the number of quadrature points for `gauss` integration method, default is `500`
`quadext(`*matname*`)`	define an external Mata matrix containing the quadrature points for the `gauss` integration method†
`quadatol(#)`	set the absolute tolerance for `adaptive` integration method default is `1e-9`
`quadrtol(#)`	set the relative tolerance for `adaptive` integration method default is `1e-7`
Estimating sampling moments (bias and variance)	
`plugin(`*pmetname*`)`	specify plug-in estimators of moments, default is `ds`
`pathtab(`*path)*	define directory for tabulated moments of the posterior mean, default is the working directory
`fast`	compute point estimates only†
Confidence intervals	
`level(#)`	set the confidence level, default is 95
`reps(#)`	set the number of MC replications for confidence intervals, default is `1000`
`rseed(#)`	set the random-number seed, default is the current random-number seed
`saving(`*filename*`[, replace])`	save MC replications of bias-corrected estimator to *filename.dta*, default is a temporary file
Reporting	
`noheader`	suppress display of output header†
`nofocus`	suppress display of focus coefficients†
`noauxiliary`	suppress display of auxiliary coefficients†
`notable`	suppress display of coefficient table†
`cformat(%fmt)`	set the format of statistics in the coefficient table, default is %8.0g

Note: The symbol † means that, by default, this option is not active.

noconstant removes the constant term from the model. Note that only one of the noconstant and auxconstant options can be specified.

sigma(#) fixes the standard deviation σ_ϵ of *depvar* to #, where # is a positive real number. By default, σ_ϵ^2 is estimated by its unbiased LS estimator from the unrestricted model. This option is typically used in programs that extend the WALS estimation procedure to more general models.

Three additional remarks on the model setup are in order. First, as with most Stata estimation commands, the estimation sample of wals is determined by the nonmissing observations in *depvar*, (*focvars*), *auxars*, and *wvar* that satisfy any restriction imposed by the [*if*] and [*in*] qualifiers. By default, the sample size n is equal to the number of observations in the estimation sample. However, in weighted WALS regressions based on fweights and iweights, n is defined as the sum of the weights over the estimation sample (truncated to an integer for iweights).

Second, wals performs a preliminary check for perfect collinearity in the joint set of focus and auxiliary regressors (including, if any, the constant term). When the regressors form a collinear set, wals drops the right-most collinear variables and displays a warning message for each dropped variable. Collinear variables in *auxvars* are always dropped before those in (*focvars*), irrespective of the order in which these variables are declared.

Third, wals displays an error message and aborts when the number of (noncollinear) focus and auxiliary regressors is greater than the sample size. This restriction excludes high-dimensional models with $k > n$, but it does not place additional constraints on the maximum number of focus and auxiliary regressors.

3.2 Options: choice of the prior

The choice of the prior for the Bayesian shrinkage step of the WALS estimation procedure is controlled by the following option:

prior(*priorname*) specifies the prior distribution for the population t-ratios in the NLP, where *priorname* can be one the following:

laplace subbotin weibull pareto cauchy horseshoe log

Names of prior distributions are *not* case sensitive and can be abbreviated as indicated by underlining. The default prior is pareto.

When the model includes multiple auxiliary regressors, the same prior is automatically assigned to all components of the $k_2 \times 1$ vector of population t-ratios. Moreover, prior parameters are fixed to the values reported in Table 1 to achieve desirable theoretical properties such as neutrality, robustness, and minimax regret optimality. Recall that no prior distribution uniformly dominates the others in terms of MSE of the underlying posterior mean. Furthermore, as with many Bayesian procedures, the choice of prior may have relevant effects on the statistical properties of our estimator. Performing a sensitivity analysis of the WALS estimates to the choice of the prior distribution is

therefore recommended.

Unlike all other priors, the Laplace prior is not robust. This lack of robustness does not affect the uniform $\sqrt{n}$-consistency of our estimator, but it introduces a persistent bias in its asymptotic distribution that does not vanish with increasing values of the population t-ratios. The Subbotin and Weibull priors perform well in terms of minimax regret, but they have low degrees of robustness. On the other hand, the Cauchy, horseshoe, and log priors have a high degree of robustness and good MSE performance in a neighborhood of the origin, but their minimax regret properties are less good. The Pareto prior has been selected as our default prior, because it offers a good compromise between these two competing criteria: its minimax regret is slightly larger than that of the Weibull prior and its degree of robustness is the same as that of the horseshoe prior. Of course, this does not mean that the Pareto prior is the 'best' prior in all situations.

3.3 Options: computing the posterior mean

Despite its lack of robustness, the Laplace prior can be computationally convenient due to the availability of a closed-form expression for its posterior mean. The posterior means resulting from the other priors are instead approximated numerically by quadrature integration methods. Accuracy and speed of these numerical methods are controlled by the following options:

`quadmethod(`*qmetname*`)` specifies the quadrature integration method used for computing the posterior mean under the `subbotin`, `weibull`, `pareto`, `cauchy`, `horseshoe`, and `log` priors. The options for *qmetname* are `gauss` (the default) and `adaptive`, where `gauss` uses the Gauss–Legendre quadrature method for the `horseshoe` prior and the Gauss–Laguerre quadrature method for the other five priors, while `adaptive` uses the adaptive quadrature method for all priors.

`quadnpts(#)` sets the number of quadrature points for the `gauss` method to #, where # is a positive integer. The default is `quadnpts(500)`.

`quadext(`*matname*`)` specifies the name of a two-column Mata matrix that contains the quadrature points (first column) and weights (second column) needed for the `gauss` method. In this case, the number of quadrature points is determined implicitly by the number of rows of *matname*. By default, this matrix is generated internally at any call of `wals` based on the `gauss` method. However, in repeated calls to `wals` (e.g., MC simulations), this option may substantially reduce the computational burden by generating the quadrature points and weights externally only once.

`quadatol(#)` sets the absolute tolerance for the convergence criterion of the `adaptive` method to #, where # is a positive real number. The default is `quadatol(1e-9)`.

`quadrtol(#)` sets the relative tolerance for the convergence criterion of the `adaptive` method to #, where # is a positive real number. The default is `quadrtol(1e-7)`.

If specified together with `prior(laplace)`, these five options are irrelevant. Similarly, `quadnpts` and `quadgext` are irrelevant if specified together with the `adaptive` method,

and `quadatol` and `quadrtol` are irrelevant if specified together with `gauss` method. In these cases, `wals` displays a warning message without stopping its execution.

In our experience, numerical differences between the `gauss` and `adaptive` methods are often negligible. The default method is `gauss` because of its relative advantages in terms of computational speed, especially when the number of auxiliary regressors or the number of MC replications for the simulation-based confidence intervals are large.

The `lwalsgaussint.mlib` library contains two functions for generating the external Mata matrix of quadrature points and weights: `gausslegendre(#)` is designed for the horseshoe prior, while `gausslaguerre(#)` is designed for the other five priors, where # denotes the desired number of quadrature points.[4] `wals` performs some checks on the quadrature points and weights parsed externally through the `quadgext` option, and aborts with an error whenever any of them is not passed.

3.4 Options: estimating sampling moments

`wals` supports two plug-in methods for estimating the sampling bias and variance of the WALS estimator: *double-shrinkage* (DS) and *maximum likelihood* (ML). Both methods are straightforward extensions of those used for estimating the bias and variance of m^* as a frequentist estimator of θ^*. Simulation results from De Luca et al. (2022) show that the plug-in DS estimators have relatively better risk performance for small values of θ^*, while the plug-in ML estimators have relatively better risk performance for large values of θ^*. Since $\theta^* = 1$ can be viewed as the key value of interest and the DS estimator performs better than ML at $\theta^* = 1$, we use the DS method as our default.

To speed up this step of the WALS estimation procedure, we provide a Stata dataset for each prior containing the tabulations of the first two moments of m^* for values of θ^* ranging in the interval $[0, 30]$ with stepsize 0.01. Given an estimate $\tilde{\theta}^*$ of θ^* based on either x^* or m^*, `wals` first computes the plug-in estimates of the bias and variance of m^* using linear interpolations of the tabulations for $\tilde{\theta}^* \leq 30$ and their asymptotic approximations for $\tilde{\theta}^* > 30$. Next it computes the plug-in estimates of the moments of $\tilde{\beta} = (\tilde{\beta}_1', \tilde{\beta}_2')$ as described in De Luca et al. (2022, Section 5). We offer three options for the estimation of the sampling moments:

`plugin(`*pmetname*`)` specifies the plug-in method for estimating the sampling moments of the WALS estimator. The available choices for *pmetname* are `ds` for the plug-in DS estimators (the default) and `ml` for the plug-in ML estimators.

`pathtab(`*path*`)` specifies the directory path where `wals` should look for datasets containing the tabulations of the sampling bias and variance. There are seven datasets, named according to the available priors:

```
wals_pm_tsm_laplace.dta
```

4. The functions `gausslegendre(#)` and `gausslaguerre(#)` were extracted from the `integrate` command of Adrian Mander and then stored in the `lwalsgaussint.mlib` library to increase the computational efficiency of `wals`.

`wals_pm_tsm_subbotin.dta` `wals_pm_tsm_weibull.dta`
`wals_pm_tsm_pareto.dta` `wals_pm_tsm_cauchy.dta`
`wals_pm_tsm_horseshoe.dta` `wals_pm_tsm_log.dta`.

By default, `wals` assumes these datasets are located in the current working directory or the ado-path. If they are located in a different directory, then this option can be used to specify the appropriate directory path.

`fast` restricts the calculations of `wals` to point estimates only, excluding sampling moments and confidence intervals. Specifying `fast`, all options related to the sampling moments and confidence intervals become ineffective.

3.5 Options: confidence intervals

`wals` also computes simulation-based confidence intervals for β_1 and β_2 using the five-step algorithm described in De Luca et al. (2023, Appendix B). Since the plug-in estimator of the bias of m^* is a key element of this algorithm, these intervals are labeled as `MC-DS` or `MC-ML` depending on the selected plug-in method. Other features of the WALS confidence intervals are controlled by the following options:

`level(#)` specifies the confidence level, where # is a percentage between 10 and 99.99. The default is `level(95)` or the confidence level resulting from `c(level)`. This option can be specified during estimation or on replay.

`reps(#)` sets the number of MC replications for the simulation-based confidence intervals to #, where # is an integer. The default is `reps(1000)`.

`rseed(#)` sets the random-number seed, which is crucial for the reproducibility of the simulation-based confidence intervals.

`saving(`*filename*`[, replace])` saves the MC replications of the bias-corrected WALS estimator to *filename*`.dta`. This option can be specified during estimation or on replay. In both cases, `replace` specifies to overwrite *filename.dta* if it exists. When `saving` is not specified, `wals` saves simulation results in a temporary file named `STWALS_000001.tmp` for later access by its post-estimation commands. This temporary file is overridden at every run of `wals`. The saved dataset contains one observation for each MC replication and a set of variables named `wals_bc_`# for the not-omitted regressors, where # is a progressive index for the regressor number. The order of the `wals_bc_`# variables corresponds to the order in which the regressors are declared in (*focvars*) and *auxvars*. The variable labels of `wals_bc_`# contain the names of the corresponding regressors.

3.6 Options: reporting

The output of `wals` consists of a header and a coefficient table. The header provides information on the sample size, the numbers of (not-omitted) focus and auxiliary regressors, the prior distribution, the estimate of σ_ϵ obtained from the unrestricted model,

and the number of MC replications used for the confidence intervals. The coefficient table displays the WALS point estimates, the estimated bias, standard error (SE), and root mean squared error (RMSE), and the lower and upper bounds of the confidence intervals. Estimates of the focus parameters are displayed before those of the auxiliary parameters. However, if the model includes a constant term, this is always displayed at the end of the coefficient table regardless of its treatment as a focus or an auxiliary regressor. There are five additional options to control the default output of `wals`:

`noheader` suppresses the display of the header at the top of the output.

`nofocus` suppresses the display of the estimation results for the focus parameters from the coefficient table.

`noauxiliary` suppresses the display of the estimation results for the auxiliary parameters from the coefficient table.

`notable` suppresses the display of the coefficient table from the output.

`cformat(%fmt)` specifies the display format of coefficients, estimated moments, and confidence limits in the coefficient table. The default format is %8.0g and the maximum format width is 8.

These five options can be specified during estimation or on replay. Specifying `fast` automatically enables the `noheader` and `notable` options.

4 Stored results

`wals` stores the following results in `e()`:

Scalars

`e(N)`	number of observations
`e(k0)`	number of columns of `e(b)` (including base levels of factor variables)
`e(k1)`	number of (not-omitted) focus regressors
`e(k2)`	number of (not-omitted) auxiliary regressors
`e(rank)`	rank of `e(V)` (also `e(k1)+e(k2)`)
`e(df_r)`	residual degrees of freedom
`e(sigma)`	estimate of σ_ϵ
`e(quadnpts)`	number of quadrature points for `gauss` method
`e(quadatol)`	absolute tolerance for `adaptive` method
`e(quadrtol)`	relative tolerance for `adaptive` method
`e(reps)`	number of MC replications for confidence intervals
`e(level)`	confidence level

Macros

`e(cmdline)`	command as typed
`e(cmd)`	`wals`
`e(title)`	title appearing in header
`e(depvar)`	name of dependent variable
`e(allvars)`	names of all regressors
`e(omitvars)`	names of omitted regressors
`e(focvars)`	names of (not-omitted) focus regressors
`e(auxvars)`	names of (not-omitted) auxiliary regressors
`e(constype)`	type of constant term: `noconstant`, `focus`, `auxiliary`
`e(wtype)`	weight type (if specified)
`e(wexp)`	weight expression (if specified)

`e(prior)`	prior distribution for the population t-ratios
`e(quadmethod)`	quadrature method: `gauss`, `adaptive`, `analytic`
`e(plugin)`	estimate of sampling moments: `ds`, `ml`
`e(bcsimdata)`	file of MC replications of the bias-corrected WALS estimator
`e(properties)`	`b V`
`e(predict)`	program to implement `predict`
`e(marginsok)`	predictions allowed by `margins` and `margwals`

Matrices

`e(b)`	estimated parameter vector
`e(V)`	estimated variance matrix
`e(bias)`	estimated bias vector
`e(rmse)`	estimated RMSE vector
`e(MSE)`	estimated MSE matrix
`e(ci)`	matrix of confidence intervals
`e(t_ratios)`	$k_2 \times 1$ vector of t-ratios in the normal location model
`e(wals_pm)`	$k_2 \times 1$ vector of posterior means in the normal location model
`e(wals_wgt)`	$k_2 \times 1$ vector of WALS weights (diagonal elements of Λ)
`e(priorpar)`	vector of prior parameters

Functions

`e(sample)`	marks estimation sample

Using `fast`, results for sampling moments and confidence intervals are not available.

5 Post-estimation commands

After WALS estimation, we need to pay attention to post-estimation commands. Such commands typically work under the implicit assumption of a correctly specified model, and therefore ignore the presence of a finite-sample estimation bias. Moreover, the distribution of the WALS estimator is not necessarily normal. To address these issues, we offer three specialized post-estimation commands designed for WALS inference.

5.1 The `predict` command

The syntax of the `predict` command associated with `wals` can be either

`predict` [*type*] *newvar* [*if*] [*in*] , [*predict_options*$_1$]

or

`predict` [*type*] *newvar*$_l$ *newvar*$_u$ [*if*] [*in*] , [*predict_options*$_2$]

where *predict_options*$_1$ and *predict_options*$_2$ are the options listed in Table 3. The first syntax creates in *newvar* one of the statistics available in *predict_options*$_1$. If no option is specified, the default `xb` is assumed. The second syntax creates in *newvar*$_l$ and *newvar*$_u$ the lower and upper bounds of the confidence intervals for $\mathbb{E}(y_i|x_i)$ if one specifies the `cinterval` option, and the lower and upper bounds of the prediction intervals for $y_i|x_i$ if one specifies the `pinterval` option. The confidence level of both intervals is controlled by the `level` option.

Table 3: Options of the `predict` command for `wals`

Options	Description
predict_options$_1$	
`xb`	linear prediction (default)
`biasp`	bias of the linear prediction
`stdp`	SE of the linear prediction
`rmsep`	RMSE of the linear prediction
`bcp`	bias-corrected linear prediction
`stdbcp`	SE of the bias-corrected linear prediction
`residuals`	residuals from the linear prediction
`bcresiduals`	residuals from the bias-corrected linear prediction
predict_options$_2$	
`cinterval`	confidence intervals for $\mathbb{E}(y_i\|x_i)$
`pinterval`	prediction intervals for $y_i\|x_i$
`level(#)`	sets the confidence level
`rseed(#)`	sets the random-number seed for prediction intervals

We now define the various statistics. Let $\tilde{\beta}$ be the $k \times 1$ vector of WALS estimates, and let b and V be the estimated bias vector and variance matrix of $\tilde{\beta}$. The bias-corrected estimator is $\tilde{\beta}^* = \tilde{\beta} - b$ and its variance matrix V^* can be estimated using the sample variances and covariances of the $R \times k$ MC replications $\tilde{B}^*$ of $\tilde{\beta}^*$ stored in `e(bcsimdata)`. If y_i and x_i denote the ith observations of y and X, then the linear prediction is defined as $\tilde{y}_i = x_i'\tilde{\beta}$, its bias as $x_i'b$, its SE as $\sqrt{x_i'Vx_i}$, and its RMSE as $\sqrt{(x_i'b)^2 + x_i'Vx_i}$. The bias-corrected linear prediction is $\tilde{y}_i^* = x_i'\tilde{\beta}^*$ and its SE is $\sqrt{x_i'V^*x_i}$. The residuals and bias-corrected residuals are $y_i - \tilde{y}_i$ and $y_i - \tilde{y}_i^*$, respectively. The confidence intervals for $\mathbb{E}(y_i|x_i)$ are obtained from the empirical percentiles of $\tilde{B}^*x_i$, while the prediction intervals for $y_i|x_i$ are obtained from the empirical percentiles of $\tilde{B}^*x_i + s\epsilon_i^*$, where ϵ_i^* is an $R \times 1$ vector of $\mathrm{N}(0,1)$ random draws and s is the estimate of σ_ϵ. The `rseed(#)` option sets the random-number seed to ensure reproducibility of the prediction intervals.

5.2 The `lcwals` command

`lcwals` extends the official `lincom` command by providing the WALS estimate of a linear combination of the focus and auxiliary parameters. Its syntax is

`lcwals` *exp*`,` [`level(#) eform loneside roneside cformat(%fmt)`]

where *exp* defines a linear combination $c_0 + c_1\beta_1 + \ldots + c_k\beta_k$ of the parameters $\beta = (\beta_1, \ldots, \beta_k)'$ from the latest WALS regression, `level(#)` sets the level of the confidence interval for this linear combination, `eform` reports the results in exponential form, `loneside` and `roneside` compute, respectively, the left and right one-sided confidence

intervals instead of the (default) two-sided confidence intervals, and `cformat(%fmt)` specifies the display format for the statistics in the coefficient table.

To specify $c_0 + c_1\beta_1 + \ldots + c_k\beta_k$, one can refer to the parameters $\beta_1, \ldots, \beta_k$ using the same shorthands as in the `lincom` command. However, arithmetical operators are restricted to "+", "-", "*", and "/", parentheses are not allowed, and the coefficients $c_0, c_1, \ldots, c_k$ must be specified as numerical values without using other mathematical functions. Rewriting our linear combination as $c_0 + c_*'\beta$, with $c_* = (c_1, \ldots, c_k)'$, the WALS point estimate is defined as $c_0 + c_*'\tilde{\beta}$, the estimated bias as $c_*'b$, and the SE as $\sqrt{c_*'Vc_*}$. The lower and upper bounds of the simulation-based confidence interval are computed through the empirical percentiles of $c_0\imath_R + \tilde{B}^*c_*$, where $\imath_R$ is the $R \times 1$ vector of ones. The `lcwals` command stores in `r()` the following results:

Scalars

`r(estimate)`	point estimate of the linear combination
`r(bias)`	estimated bias
`r(se)`	estimated SE
`r(rmse)`	estimated RMSE
`r(lb)`	lower bound of confidence interval; "." if `loneside` is specified
`r(ub)`	upper bound of confidence interval; "." if `roneside` is specified
`r(level)`	confidence level

Matrices

`r(lc_coef)`	coefficient vector $c = (c_0, c_1, \ldots c_k)$

5.3 The `margwals` command

`margwals` exploits certain functionalities of the `margins` command to calculate the WALS estimate of a smooth, possibly nonlinear, real-valued function $g(\beta, x)$ of the parameters β at some value x of the regressors. The extension to factor variables represents a significant advancement for using `margins` as a post-estimation command for `wals`. However, validity of inference also requires accounting for the finite-sample bias and the potential nonnormal distribution of the WALS estimator $\tilde{g} = g(\tilde{\beta}, x)$ of $g(\beta, x)$. An approximated SE of $\tilde{g}$ could be achieved by the (first-order) delta method, but accurate bias approximations often require higher-order expansions. While the `margwals` command does not fully resolve this challenging issue, it provides a simulation-based confidence interval for $g(\beta, x)$ using the empirical percentiles of $g(\tilde{\beta}^*_{(r)}, x)$ $(r = 1, \ldots, R)$, where $\tilde{\beta}^*_{(r)}$ is the rth MC replication of the bias-corrected WALS estimates $\tilde{\beta}^*$ of β. Its syntax is:

`margwals` [*marginlist*] [*if*] [*in*] [*weight*] , [*margwals_options*]

where *marginlist* is a list of factor variables or interactions that appear in the latest WALS regression, and *margwals_options* are the options listed in Table 4. A detailed description of the available options can be found in [R] **margins**. `margwals` stores in `r()` the following results:

Table 4: Options of the `margwals` command

margwals_options	Description
Response options	
`predict(`*pred_opt*`)`	estimate margins for `predict,`*pred_opt*
`dydx(`*varlist*`)`	estimate marginal effects $d(y)/d(x)$
`eyex(`*varlist*`)`	estimate elasticities $d(\ln y)/d(\ln x)$
`dyex(`*varlist*`)`	estimate semi-elasticities $d(y)/d(\ln x)$
`eydx(`*varlist*`)`	estimate semi-elasticities $d(\ln y)/d(x)$
`continuous`	treat factor-level indicators as continuous
Other options from the `margins` command	
`grand`	add the overall margin
`at(`*atspec*`)`	estimate margins at specified values of regressors
`atmeans`	estimate margins at the means of regressors
`asbalanced`	treat factor variables as balanced
`over(`*varlist*`)`	estimate margins at unique values of *varlist*
`within(`*varlist*`)`	estimate margins at unique values of the nesting factors in *varlist*
`level(`#`)`	set confidence level, default level is 95
advanced_options	advanced options of `margins`

Note: *advanced_options* include `noweights`, `noesample`, `emptycells(`*empspec*`)`, `estimtolerance(`#`)`, `noestimcheck`, `force`, `chainrule`, and `nochainrule`. The `nose` and `nopvalue` options are enforced in the internal call to `margins` to avoid misleading interpretations of the results.

Scalars

`r(N)`	number of observations
`r(k_margins)`	number of terms in *marginlist*
`r(k_predict)`	number of `predict()` options
`r(k_at)`	number of `at()` options
`r(level)`	confidence level

Macros

`r(cmd)`	`margwals`
`r(cmdline)`	command as typed
`r(est_cmd)`	`e(cmd)` from original estimation results
`r(est_cmdline)`	`e(cmdline)` from original estimation results
`r(title)`	title in output
`r(expression)`	response expression
`r(predict`#`_opts)`	the #th `predict()` option
`r(predict`#`_label)`	label from the #th `predict()` option
`r(xvars)`	*varlist* from `dydx()`, `dyex()`, `eydx()`, `eyex()`
`r(derivatives)`	"", "dy/dx", "dy/ex", "ey/dx", "ey/ex"
`r(atstats`#`)`	the #th `at()` specification
`r(over)`	*varlist* from `over()`
`r(within)`	*varlist* from `within()`
`r(emptycells)`	*empspec* from `emptycells()`

Matrices

`r(b)`	point estimates
`r(ci)`	confidence intervals
`r(_N)`	sample size corresponding to each margin estimate

`r(at)`	matrix of values from the `at()` options
`r(error)`	margin estimability codes

6 Comparisons with other methods

This section presents two examples: one on the comparison between WALS and LASSO methods for inference, and another on the out-of-sample predictive performance of WALS and BMA. In both cases, we consider models with $k_2 \leq n$ because WALS cannot be applied to high-dimensional models with $k_2 > n$. The three approaches differ in many other respects and each of them has its own advantages and disadvantages. Our objective is to emphasize similarities and differences in their application within Stata using simple real-data examples where we do not know the true data-generation process (DGP). For extensive MC simulations, see De Luca et al. (2018, 2022, 2023).

6.1 Comparisons with LASSO methods for inference

Our first example compares WALS with the cross-fit partialing-out LASSO linear regression method of Chernozhukov et al. (2018), using a dataset that includes children's performance on a test of reaction time, levels of nitrogen dioxide (NO2) pollution in the classroom, and other controls related to children's characteristics and environmental factors. This is the primary dataset used in [LASSO] **Inference examples** to illustrate the suite of LASSO methods for inference available in Stata. A description of the cross-fit partialing-out algorithm, which is also known as double machine learning and represents the recommended LASSO method for inference in Stata, can be found in [LASSO] **Lasso inference intro** and [LASSO] **xporegress**.

```
. use "https://www.stata-press.com/data/r18/breathe", clear
(Nitrogen dioxide and attention)
. describe
  (output omitted)
```

Variable name	Storage type	Display format	Value label	Variable label
react	double	%10.0g		* Reaction time (ms)
correct	int	%10.0g		* Number of correct responses
omissions	byte	%10.0g		* Failure to respond to stimulus
no2_class	float	%9.0g		Classroom NO2 levels (ug/m3)
no2_home	float	%9.0g		Home NO2 levels (ug/m3)
age	float	%9.0g		Age (years)
age0	double	%4.1f		Age started school
sex	byte	%9.0g	sex	Sex
grade	byte	%9.0g	grade	Grade in school
overweight	byte	%32.0g	overwt	* Overweight by WHO/CDC definition
lbweight	byte	%18.0g	lowbw	* Low birthweight
breastfeed	byte	%19.0f	bfeed	Duration of breastfeeding
msmoke	byte	%10.0f	smoke	* Mother smoked during pregnancy
meducation	byte	%17.0g	edu	Mother´s education level
feducation	byte	%17.0g	edu	Father´s education level
siblings_old	byte	%1.0f		Number of older siblings in house
siblings_young	byte	%1.0f		Number of younger siblings in

```
                                                house
sev_home        float   %9.0g                 Home socio-economic vulnerability
                                                index
green_home      double  %10.0g                Home greenness (NDVI), 300m buffer
noise_school    float   %9.0g                 School noise levels (dB)
sev_school      float   %9.0g                 School socio-economic
                                                vulnerability index
precip          double  %10.0g                Daily total precipitation
                                              * indicated variables have notes
```

```
. run "https://www.stata-press.com/data/r18/no2"
. display "$cc"
no2_home age age0 sev_home green_home noise_school sev_school precip siblings_ol
> d siblings_young
. display "$fc"
sex grade overweight lbweight breastfeed msmoke meducation feducation
. rename no2_class NO2
```

As described in Section 1.4 of [LASSO] **Inference examples**, the do-file `no2` from the Stata Press website exploits the `vl` tools to create the macros `$cc` and `$fc` of continuous and categorical controls that are used in the analysis. Of course, these tools are also useful for `wals`. After running this do-file, we have renamed the variable of interest as `NO2` to shorten the width of the estimation tables. Below, we reproduce the cross-fit partialing-out estimate of the effect of `NO2` on the children's reaction time (`react`) presented in Section 2.2 of the manual

```
. xporegress react NO2, controls($cc i.($fc)) rseed(12345)
Cross-fit fold 1 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for NO2 using plugin
  (output omitted)
Cross-fit fold 10 of 10 ...
Estimating lasso for react using plugin
Estimating lasso for NO2 using plugin
Cross-fit partialing-out            Number of obs                =      1,036
linear model                        Number of controls           =         32
                                    Number of selected controls  =         10
                                    Number of folds in cross-fit =         10
                                    Number of resamples          =          1
                                    Wald chi2(1)                 =      22.87
                                    Prob > chi2                  =     0.0000
```

react	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
NO2	2.316063	.4843097	4.78	0.000	1.366834	3.265293

```
(Note omitted)
```

The dataset contains `1,089` observations, but `53` cases have been excluded due to missing values in `react` and the control variables. By default, `xporegress` partitions the sample in 10 folds and performs two LASSO regressions for each fold to select `10` out of the `32` controls that predict either `NO2` or `react`. The post-LASSO estimates computed on the observations excluded from a specific fold are used to fill in the moment

condition for the relationship of interest over the observations in that fold. Finally, `xporegress` estimates the effect of `NO2` on `react` by solving this moment condition on all observations. The estimated effect is `2.3` milliseconds for each microgram per cubic meter increase in NO2, with a 95% confidence interval of `[1.4,3.3]`. Other LASSO methods for inference select different controls, but they yield similar estimates of the effect of interest (see Sections 2.3–2.5 of [LASSO] **Inference examples**). These methods require some type of sparsity restriction that places an upper bound on the number of controls with a nonzero coefficient. Furthermore, they do not produce estimates for the coefficients of the control variables.

In WALS, we can estimate a model that includes the constant term and `NO2` as focus regressors and the other controls as auxiliary regressors:

```
. wals react (NO2) $cc i.($fc), rseed(12345)
WALS estimates                                  Number of obs =    1036
Prior : pareto                                  Residual df   =    1010
                                                k1            =       2
                                                k2            =      24
                                                sigma         = 127.694
                                                MC reps       =    1000
```

react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus						
NO2	2.20101	-.12058	.461191	.476693	1.30847	3.30532
auxiliary						
no2_home	-.10725	.009491	.198128	.198355	-.657218	.403275
age	-28.7581	-.483926	11.6497	11.6598	-49.3404	-5.61847
age0	4.0096	-.944844	2.23474	2.42627	-.221951	11.0171
sev_home	14.9389	-5.81467	16.3552	17.3581	-23.6047	59.7817
green_home	58.9717	-15.2963	52.0017	54.2047	-55.7543	204.279
noise_school	-.335386	.093282	.707619	.71374	-2.28504	1.5061
sev_school	-6.03702	2.7736	15.9972	16.2359	-53.9989	35.7345
precip	3.42157	-.640466	2.89776	2.9677	-3.40603	12.3593
siblings_old	.830905	-.174333	4.45695	4.46036	-10.2613	13.0242
siblings_young	10.7838	-2.84237	4.84093	5.6137	.254502	25.5255
1.sex	46.1431	-4.28666	8.27412	9.31861	33.4499	65.3221
2.grade	-52.855	8.19543	15.4836	17.5188	-91.737	-32.5193
3.grade	-76.2981	9.49753	26.2408	27.9067	-134.773	-35.5985
1.overweight	1.84903	-.466519	6.35557	6.37267	-12.319	20.5821
1.lbweight	-11.7096	3.67913	10.2183	10.8605	-42.7771	12.0117
2.breastfeed	-2.67128	1.55011	7.97715	8.12637	-24.1532	17.324
3.breastfeed	1.18779	.434658	8.66136	8.67226	-21.7588	24.6437
1.msmoke	-11.445	4.61364	14.7494	15.4542	-53.6803	23.6845
2.meducation	105.766	-28.4469	47.7028	55.5409	7.22016	249.971
3.meducation	83.1082	-22.4668	43.988	49.3933	-8.18059	215.184
4.meducation	77.192	-21.4828	43.4221	48.4458	-17.9758	204.325
2.feducation	53.3833	-13.348	37.2088	39.5306	-22.7146	160.08
3.feducation	45.4975	-10.573	36.2068	37.719	-34.851	155.661
4.feducation	24.4833	-4.65777	35.8386	36.14	-60.26	121.674
focus						
_cons	825.118	37.6298	111.359	117.545	558.689	1017.75

There are several differences with respect to `xporegress`. First, we see from the header that the number of auxiliary regressors in `wals` is smaller than the number of controls in `xporegress` (i.e., 24 instead of 32). This is due to the different treatment of the constant term and the categorical controls: WALS excludes the base category of the categorical regressors to avoid collinearity with the constant term (like `regress`), whereas LASSO methods rely on a preliminary standardization of the variables which eliminates the constant term, and they do not exclude the base category of the categorical controls.

Second, `wals` provides estimates of the full vector of focus and auxiliary parameters. This reflects the fact that WALS is not a model selection method and it does not require sparsity restrictions on the unknown DGP. Model averaging methods like WALS and BMA are known to work well when the DGP is 'dense', i.e. most of its parameters are nonzero but small in magnitude (see, e.g, Giannone et al. 2021).

Third, `wals` shrinks linear combinations of the auxiliary parameters to zero by placing a symmetric prior with zero mean on their population t-ratios. Although the WALS estimator is $\sqrt{n}$-consistent, this Bayesian shrinkage step introduces a nonzero finite-sample estimation bias to improve the MSE of the estimated coefficients. In addition to the point estimates, `wals` provides the plug-in DS estimates of the sampling moments (bias, SE and RMSE) and the 95% simulation-based confidence intervals. Under the default Pareto prior, the estimated effect of `NO2` on `react` is slightly smaller than the effect obtained in `xporegress`: `2.2` with a 95% confidence interval of `[1.3,3.3]`. The plug-in DS estimates of the bias and SE for this coefficient are `-0.121` and `0.461`, respectively. The resulting RMSE of `0.477` for the WALS estimator is slightly smaller than the (robust) SE of `0.484` for the (unbiased) cross-fit partialing-out estimator.

For a sensitivity analysis on the choice of prior, we now present the WALS estimates based on the Weibull and the horseshoe priors:

```
. wals react (NO2) $cc i.($fc), rseed(12345) nohead noaux pri(wei)
```

	react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus	NO2	2.19025	-.118825	.460456	.475541	1.30368	3.29048
focus	_cons	820.292	33.9173	110.599	115.683	556.543	1015.16

```
. wals react (NO2) $cc i.($fc), rseed(12345) nohead noaux pri(hor)
```

	react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus	NO2	2.16063	-.104053	.451081	.462926	1.27567	3.21278
focus	_cons	827.68	41.029	110.28	117.665	556.605	1012.24

Here, we added the `nohead` and `noaux` options to suppress the display of the heading and the auxiliary parameters. The estimated effect of `NO2` on `react` is slightly smaller than the effect obtained with the Pareto prior, but differences are small. The plug-in DS estimates of the sampling moments also suggest that the horseshoe prior may lead to further efficiency gains in terms of both bias and SE.

Next we assess the sensitivity of the results to the choice of the integration method for computing the posterior mean in the NLP:

```
. wals react (NO2) $cc i.($fc), rseed(12345) nohead noaux pri(hor) quadm(adaptive)
```

	react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus	NO2	2.16052	-.104009	.451077	.462913	1.27562	3.21251
focus	_cons	827.692	41.0262	110.279	117.663	556.642	1012.27

```
. wals react (NO2) $cc i.($fc), rseed(12345) nohead noaux pri(hor) quadnpts(1000)
```

	react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus	NO2	2.16057	-.104031	.451079	.46292	1.27564	3.21265
focus	_cons	827.686	41.0276	110.279	117.664	556.624	1012.25

The `adaptive` method is expected to be more reliable than the `gauss` method, but it could be subject to convergence issues and is relatively more time consuming. Similar considerations apply to the `gauss` method when we increase the number of quadrature points and weights. Differences between the integration methods are typically negligible. If not, then one should use more computationally demanding methods.

Our plug-in DS estimators of the sampling moments suggest that the WALS estimator is slightly more efficient (in the MSE sense) than the cross-fit partialing-out estimator. However, the plug-in ML estimators give considerably larger estimates of the bias that offset the efficiency gains:

```
. wals react (NO2) $cc i.($fc), rseed(12345) nohead noaux pri(hor) plugin(ml)
```

	react	Coef.	ML Bias	ML Std.Err.	ML RMSE	MC-ML [95% Conf.	Int.]
focus	NO2	2.16063	-.189547	.462759	.500074	1.32491	3.36419
focus	_cons	827.68	43.7087	113.827	121.931	547.074	1012.81

Hence, uncertainty about the estimation of the bias of the WALS estimator makes it difficult to draw sharp conclusions on the relative efficiency of the two approaches. We conclude that, in this example, WALS and cross-fit partialing-out lead to similar results.

Let us consider an additional example, where we investigate whether the marginal effect `NO2` on `react` is nonconstant and possibly heterogeneous across grade levels. Accordingly, we include into our model the square of `NO2` and a set of interactions between `NO2` and grade levels. As discussed in Section 2.10 of [LASSO] **Inference examples**, LASSO methods for inference assume that *controls are just controls* and they do not interact with the variable of interest. Thus, we are forced to treat all these variables as variables of interest:

```
. quietly vl create fc7 = fc - (grade)
. rename grade G
. global ne "c.NO2#i.G c.NO2#c.NO2 c.NO2#c.NO2#i.G"
. local l_opts "nolog nohead nofvlab rseed(12345)"
. xporegress react c.NO2 i.G ${ne}, controls($cc i.($fc7)) `l_opts´
```

react	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
NO2	-.4165934	3.059533	-0.14	0.892	-6.413167	5.57998
G						
2	-83.45485	58.31447	-1.43	0.152	-197.7491	30.83941
3	-46.81376	61.13867	-0.77	0.444	-166.6434	73.01583
G#c.NO2						
2	2.080714	3.720957	0.56	0.576	-5.212227	9.373655
3	-2.282959	3.755203	-0.61	0.543	-9.643022	5.077105
c.NO2#c.NO2	.0469711	.0453383	1.04	0.300	-.0418903	.1358324
G#c.NO2#c.NO2						
2	-.0393269	.057894	-0.68	0.497	-.1527971	.0741433
3	.0308117	.0638852	0.48	0.630	-.0944009	.1560244

```
(Note omitted)
```

In WALS, we can always treat these variables as focus regressors. However, given that the partition between focus and auxiliary regressors is not subject to special restrictions, it could also be reasonable to treat the second-order effects of squared and interaction terms (the variables included in the macro `$ne`) as auxiliary parameters. Let us compare the WALS estimates of these two alternative models using the default Pareto prior:

```
. wals react (c.NO2 i.G $ne) $cc i.($fc7), rseed(12345) nohead noaux
```

react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus						
NO2	-1.58316	-.068473	3.55122	3.55188	-8.98812	5.47316
2.G	-114.485	-5.44805	71.4446	71.652	-252.421	19.477
3.G	-20.4347	-6.787	89.7055	89.9619	-187.134	163.98

```
      2.G#c.NO2 |   3.95805   -.118132   4.66364   4.66513   -4.85983    13.0808
      3.G#c.NO2 |  -4.95054   -.377818   6.08899    6.1007   -16.9815    6.99094
    c.NO2#c.NO2 |   .065649    -.00128   .053818   .053833    -.04134    .177489
2.G#c.NO2#c.NO2 |  -.072708    .003039   .072144   .072208   -.216087    .064694
3.G#c.NO2#c.NO2 |   .072196    .006669   .101045   .101265    -.13212    .266935
----------------+------------------------------------------------------------------
focus           |
          _cons |    799.88   -7.34962   108.245   108.494    536.759    1061.53
------------------------------------------------------------------------------
```

```
. wals react (c.NO2 i.G) $ne $cc i.($fc7), rseed(12345) nohead
```

react	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
focus						
NO2	-.197243	.670188	2.5953	2.68043	-7.83271	5.69755
2.G	-95.8808	3.98724	49.0821	49.2438	-234.869	28.242
3.G	-55.4199	-27.0453	64.4529	69.8972	-213.023	131.153
auxiliary						
2.G#c.NO2	2.48722	-.851438	3.15801	3.27078	-5.29842	11.7815
3.G#c.NO2	-2.72679	.93855	4.22261	4.32566	-13.9747	8.10946
c.NO2#c.NO2	.041585	-.014154	.038817	.041317	-.042114	.153952
2.G#c.NO2#c.NO2	-.047322	.015696	.049046	.051497	-.192749	.067174
3.G#c.NO2#c.NO2	.038745	-.013245	.069421	.070674	-.140339	.218753
(*output omitted*)						
focus						
_cons	780.863	-16.5739	101.136	102.485	523.665	1053.44

We see that treating these second-order effects as auxiliary parameters leads to a substantial reduction in the SE that more than compensates for the increase (in absolute value) of the bias. Thus, other thing being equal, the second model leads to a sizeable reduction of the estimated RMSE. Similar results also hold for the plug-in ML estimates of the sampling moments. More generally, this example clarifies that the partition between *variables of interest* and *control variables* in LASSO methods for inference is different and probably more restrictive than the partition between *focus* and *auxiliary* regressors in WALS.

Another issue is that the interpretation of the estimates is now complicated by the presence of interaction and nonlinear terms. LASSO methods for inference do not allow using `margins` because, by construction, their results are restricted to the coefficients of the variables of interest. In WALS, however, we can use `margwals` to compute relevant summaries like the average marginal effects of NO2 pollution and grade levels on children's reaction time:

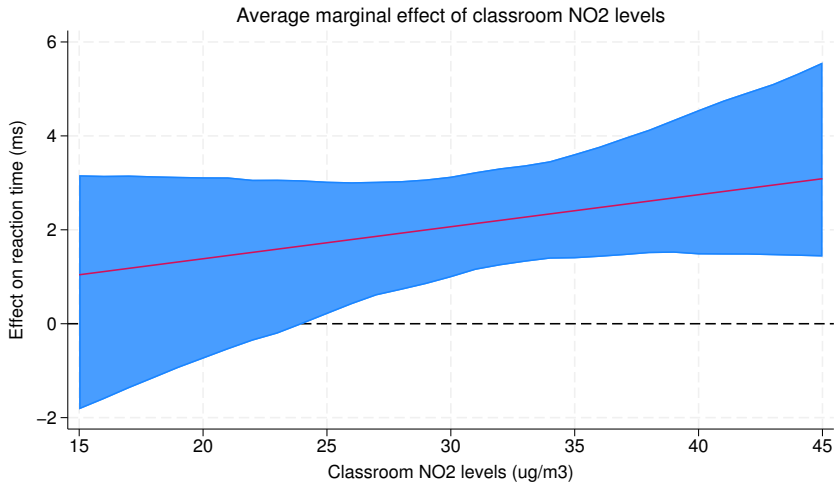
```
. margwals, dydx(NO2 i.G)
Average marginal effects                  Number of obs = 1036
                                          MC reps       = 1000
```

	Coef.	MC-DS [95% Conf.	Interval]
NO2	2.053827	.9922122	3.102509

```
                2.G |   -68.55383   -91.68103   -34.95337
                3.G |   -98.63643   -136.7043   -34.17751
```

We conclude our analysis by showing how to exploit the results stored by `margwals` to plot the estimated profile of the average marginal effects of `NO2` on `react` and their 95% confidence intervals:

```
. quietly margwals, dydx(NO2) at(NO2=(15(1)45))
. matrix AME=r(b)´
. matrix CI_AME=r(ci)´
. matrix AT=r(at)
. matrix X=AT[1...,"NO2"]
. svmat double AME
. svmat double X
. svmat double CI_AME
. #delimit;
. twoway  (rarea CI_AME1 CI_AME2 X, sort)
>         (line AME X, sort)
>         , legend(off) xlab(15(5)45) yline(0)
>         xtit("Classroom NO2 levels (ug/m3)")
>         ytit("Effect on reaction time (ms)")
>         t1title("Average marginal effect of classroom NO2 levels");
. #delimit cr
. gr export AME.eps, replace
file AME.eps saved as EPS format
```



WALS estimates suggest that the average marginal effect of NO2 pollution in the classroom on children's reaction time is increasing and statistically significant at the 5% level only for values of `NO2` greater than 24 microgram per cubic meter.

6.2 Comparisons with BMA

In the second example, we compare the predictive performance of BMA and WALS using the `uscrime` dataset from Vandaele (1978). Raftery et al. (1997) provide an extensive BMA analysis of this dataset in an attempt to determine robust determinants of criminal activity in the USA. The same dataset is also used in the Stata manual for `bmaregress`. We will argue that `wals` is a good competitor to `bmaregress`.

The `uscrime` dataset contains 16 variables and 47 observations, one for each of 47 states in the USA. The dependent variable, `ln_offenses`, measures the rate of criminal activity per head of population on the log scale. The other variables represent different socio-economic and punishment-related factors, also measured on the log scale. Like Raftery et al. (1997) and the Stata manual for `bmaregress`, we treat the constant term as a focus regressor and the other 15 regressors as auxiliary regressors, so the model space contains $2^{15} = 32,768$ different models. Although this setup is widely used in BMA, we note that this is not necessarily the best specification for WALS (and BMA). For example, the theory of Ehrlich (1973) could be used to find a better partition of the focus and auxiliary regressors. Here, we maintain this model setup only for comparability with Raftery et al. (1997) and the Stata manual for `bmaregress`. The default method of `bmaregress` for this study is

```
. quietly use "https://www.stata-press.com/data/r18/uscrime", clear
. local Y  "ln_offenses"
. local X2 "ln_malepop-ln_prisont"
. set seed 12345
. bmaregress `Y´ `X2´
Burn-in ...
Simulation ...
Computing model probabilities ...
Bayesian model averaging                        No. of obs         =     47
Linear regression                               No. of predictors  =     15
MC3 sampling                                                Groups =     15
                                                            Always =      0
                                                No. of models      =    733
                                                    For CPMP >= .9 =    208
Priors:                                         Mean model size    =  5.103
  Models: Beta-binomial(1, 1)                   Burn-in            =  2,500
   Cons.: Noninformative                        MCMC sample size   = 10,000
   Coef.: Zellner´s g                           Acceptance rate    = 0.2539
       g: Benchmark, g = 225                    Shrinkage, g/(1+g) = 0.9956
  sigma2: Noninformative                        Mean sigma2        =  0.045
Sampling correlation = 0.9437
```

ln_offenses	Mean	Std. dev.	Group	PIP
ln_ineq	1.441125	.3916633	13	.98735
ln_meaneduc	1.426249	.8779296	3	.79453
ln_police60	.7346608	.5844326	4	.64867
ln_malepop	.7964977	.7904622	1	.5711
ln_prisonp	-.1206976	.127545	14	.54922
ln_police59	.3999186	.5664395	5	.37174
ln_nonwhite	.0321224	.0525094	9	.32331

ln_unemp39	.0944666	.1725628	11	.28272
ln_pop	-.0099394	.0291575	8	.1406
ln_wealth	.0741559	.2562308	12	.11024
ln_prisont	-.0230383	.0929454	15	.097757
southern	.0116989	.0545887	2	.075721
ln_mtofpop	.0988766	.5714795	7	.058721
ln_unemp24	-.0021081	.0776143	10	.054354
ln_labor	.0150732	.1498661	6	.037395
Always				
_cons	-18.32688	8.037452	0	1

Note: Coefficient posterior means and std. dev. estimated from 733 models.
Note: Default priors are used for models and parameter g.

These are the BMA estimates based on a beta-binomial$(1, 1)$ prior for the model space, noninformative priors for the constant and the error variance, and a Zellner's g prior for the other regression coefficients with a fixed benchmark prior for $g = \max(n, k_2^2) = 225$. The default method to explore the model space is the MCMC model composition (MC3) sampling of Madigan and York (1995), with 10,000 replications after a burn-in of 2,500 iterations. The algorithm visited 733 models. The table shows the posterior mean and standard deviation of the coefficients, with the predictors ordered on the basis of their posterior inclusion probabilities (PIP). Example 15 in the Stata manual for `bmaregress` shows that in this application the highest posterior model probability is equal to `0.041`. This indicates that there is significant model uncertainty as posterior model probabilities are distributed across numerous models.

The default WALS estimates of the same model are:

```
. wals `Y´ `X2´
```

WALS estimates — Number of obs = 47
Prior : pareto — Residual df = 31
k1 = 1
k2 = 15
sigma = .180872
MC reps = 1000

ln_offen~s	Coef.	DS Bias	DS Std.Err.	DS RMSE	MC-DS [95% Conf.	Int.]
auxiliary						
ln_malepop	1.21199	-.298884	.409252	.506773	.538651	2.5059
southern	.020114	-.017163	.10095	.102399	-.227349	.287408
ln_meaneduc	1.81686	-.336815	.560557	.653963	.915122	3.26357
ln_police60	.841855	.016609	.797096	.797269	-.7186	2.34385
ln_police59	-.091406	-.051631	.806305	.807956	-1.58549	1.53633
ln_labor	.342163	-.167219	.54492	.57	-.85448	1.93187
ln_mtofpop	-2.53626	.240503	1.35013	1.37139	-6.34867	.713122
ln_pop	-.082048	.004593	.035126	.035425	-.174386	.006029
ln_nonwhite	.105312	-.010048	.045172	.046276	.026426	.216141
ln_unemp24	-.027291	.057485	.231182	.238222	-.693311	.534475
ln_unemp39	.289326	-.117884	.19291	.226078	-.064942	.871101
ln_wealth	.467571	-.191033	.383059	.428052	-.17372	1.50621
ln_ineq	1.34662	-.254776	.340875	.425566	.832646	2.36167
ln_prisonp	-.263329	.040662	.096171	.104414	-.484458	-.100594
ln_prisont	-.237222	.036532	.131007	.136005	-.575099	.057146

focus						
	_cons	-6.8083	5.31593	7.07821	8.85213	-29.3383	6.20396

We find sizeable differences in the magnitude of the estimated coefficients and their SEs. Here, we focus on the out-of-sample predictive performance of the two approaches by comparing five variants of the BMA and WALS estimators and their prediction intervals.

The default methods of `bmaregress` and `wals` are labeled BMA_1 and WALS_1, respectively. The other four BMA estimators are obtained by varying the prior for the model space, the prior for the parameters of each model, and the method to explore the model space. Specifically, BMA_2 uses the same priors as BMA_1, but it considers a complete enumeration of the model space. This is the default of `bmaregress` when $k_2 \leq 12$. BMA_3 sets $g = \sqrt{n} = 6.86$, leading to a stronger shrinkage of the coefficients toward zero relative to BMA_1. BMA_4 uses the `gprior(hyperg 3)` option, an example of a random prior for g, while BMA_5 uses a uniform prior for the model space.[5] After each BMA regression, we use `bmacoefsample` to simulate the regression coefficients from their posterior distributions and then `bmapredict` to compute the posterior means and the credible intervals for the simulated outcome.

For WALS, we vary the prior distribution for the vector of population t-ratios in the NLP. In addition to the Pareto prior (WALS_1), we use the Laplace (WALS_2), Weibull (WALS_3), Cauchy (WALS_4), and horseshoe (WALS_5) priors. The Subbotin and log priors are excluded because of their similarities with the Weibull and horseshoe priors, respectively. To speed-up computing time, we generate the Mata matrix of quadrature points and weights externally by the `gausslaguerre(500)` and `gausslegendre(500)` functions and then specify the `quadext` option. After each WALS regression, we also use `predict` to compute the linear predictions and the prediction intervals.

We measure the predictive performance of the ten estimators and the predictive coverage of their intervals by leave-one-out cross validation. Compared to the two-sample splitting strategy used by Raftery et al. (1997) and the Stata manual for `bmaregress`, this method has the advantage of not being sensitive to random sorting of the observations in the estimation and test samples. Out-of-sample prediction errors are evaluated by both the squared error loss and the absolute error loss. Our Stata script is:

```
. set seed 12345
. quietly sum `Y´
. local Obs =r(N)
. generate obs=_n
. matrix MSPE=J(`Obs´, 10, .)
. matrix MAPE=J(`Obs´, 10, .)
. matrix PI_C=J(`Obs´, 10, .)
. matrix coln MSPE = bma1 bma2 bma3 bma4 bma5 wals1 wals2 wals3 wals4 wals5
```

5. Examples 15 and 16 of the Stata manual for `bmaregress` are based on BMA_5. In this example, the `ric` prior for g coincides with benchmark prior, while the `uip` and `ebl` priors give fixed values for g that are in between $g = \sqrt{n} = 6.86$ and $g = \max(n, k_2^2) = 225$.

```
. matrix coln MAPE = bma1 bma2 bma3 bma4 bma5 wals1 wals2 wals3 wals4 wals5
. matrix coln PI_C = bma1 bma2 bma3 bma4 bma5 wals1 wals2 wals3 wals4 wals5
.
. * BMA options
. local b1_opts   "sampling         saving(bma1_1, replace)"
. local b2_opts   "enumeration      saving(bma2_1, replace)"
. local b3_opts   "gprior(sqrtn)    saving(bma3_1, replace)"
. local b4_opts   "gprior(hyperg 3) saving(bma4_1, replace)"
. local b5_opts   "mprior(uniform)  saving(bma5_1, replace)"
. local bc1_opts  "saving(bma1_2, replace) mcmcsize(10000) nodots"
. local bc2_opts  "saving(bma2_2, replace) mcmcsize(10000) nodots"
. local bc3_opts  "saving(bma3_2, replace) mcmcsize(10000) nodots"
. local bc4_opts  "saving(bma4_2, replace) mcmcsize(10000) nodots"
. local bc5_opts  "saving(bma5_2, replace) mcmcsize(10000) nodots"
.
. * WALS options
. mata: GLag=gausslaguerre(500)
. mata: GLeg=gausslegendre(500)
. local w1_opts   "prior(par) quadext(GLag)"
. local w2_opts   "prior(lap) quadext(GLag)"
. local w3_opts   "prior(wei) quadext(GLag)"
. local w4_opts   "prior(cau) quadext(GLag)"
. local w5_opts   "prior(hor) quadext(GLeg)"
.
. * Loop over observations
. forvalue n=1(1)`Obs´ {
  2.
.    * BMA estimates
.    local j=1
  3. forvalues m=1(1)5 {
  4.    * Estimate and prediction
.       quietly bmaregress `Y´ `X2´               if obs!=`n´, `b`m´_opts´
  5.    quietly bmacoefsample, `bc`m´_opts´
  6.    quietly bmapredict double bma`m´_xb if obs==`n´, mean
  7.
.       * Squared prediction error
.       quietly generate double bma`m´_serr = (`Y´-bma`m´_xb)^2
  8.    quietly summarize bma`m´_serr             if obs==`n´, meanonly
  9.    quietly matrix MSPE[`n´,`j´]=r(mean)
 10.
.       * Absolute prediction error
.       quietly generate double bma`m´_aerr = abs(`Y´-bma`m´_xb)
 11.    quietly summarize bma`m´_aerr             if obs==`n´, meanonly
 12.    quietly matrix MAPE[`n´,`j´]=r(mean)
 13.
.       * Coverage of 90% credible interval
.       quietly bmapredict bma`m´_low bma`m´_upp if obs==`n´, cri clevel(90)
 14.    quietly generate bma`m´_cov = `Y´<bma`m´_upp & `Y´>bma`m´_low if obs==`n´
 15.    quietly summarize bma`m´_cov, meanonly
 16.    quietly matrix PI_C[`n´,`j´]=r(mean)
 17.
.       drop bma`m´_*
 18.    local j=`j´+1
 19. }
```

```
 20.
.    * WALS estimates
.    forvalues m=1(1)5 {
 21.    * Estimate and prediction
.       quietly wals `Y´ `X2´                     if obs!=`n´, `w`m´_opts´
 22.    quietly predict double wals`m´_xb         if obs==`n´
 23.
.       * Squared prediction error
.       quietly generate double wals`m´_serr = (`Y´-wals`m´_xb)^2
 24.    quietly summarize wals`m´_serr, meanonly
 25.    quietly matrix MSPE[`n´,`j´]=r(mean)
 26.
.       * Absolute prediction error
.       quietly generate double wals`m´_aerr = abs(`Y´-wals`m´_xb)
 27.    quietly summarize wals`m´_aerr            if obs==`n´, meanonly
 28.    quietly matrix MAPE[`n´,`j´]=r(mean)
 29.
.       * Coverage of 90% prediction interval
.       quietly predict wals`m´_low wals`m´_upp if obs==`n´, pint level(90)
 30.    quietly generate wals`m´_cov=`Y´<wals`m´_upp & `Y´>wals`m´_low if obs==`n´
 31.    quietly summarize wals`m´_cov, meanonly
 32.    quietly matrix PI_C[`n´,`j´]=r(mean)
 33.
.       drop wals`m´_*
 34.    local j=`j´+1
 35. }
 36.}
. forvalues m=1(1)5 {
  2.    cap erase bma`m´_1.dta
  3.    cap erase bma`m´_2.dta
  4.}
. cap erase `e(bcsimdata)´
```

Results on the mean squared prediction errors are store in the matrix `MSPE`, those on the mean absolute prediction errors in the matrix `MAPE`, and those on the coverage of the prediction intervals in the matrix `PI_C`. After the loop over observations, we also erase the datasets of simulations created by `bmaregress`, `bmacoefsample`, and `wals`.

Now we summarize the results of our little experiment, starting from MSPE:

```
. clear
. quietly set obs `Obs´
. quietly svmat double MSPE, names(col)
. tabstat bma?, stat(mean) format(%6.4f)
   Stats |      bma1      bma2      bma3      bma4      bma5
---------+-------------------------------------------------
    Mean |    0.0782    0.0762    0.0573    0.0603    0.0698
----------------------------------------------------------

. tabstat wals?, stat(mean) format(%6.4f)
   Stats |     wals1     wals2     wals3     wals4     wals5
---------+-------------------------------------------------
    Mean |    0.0571    0.0569    0.0571    0.0579    0.0582
----------------------------------------------------------
```

Apparently, WALS performs well relative to BMA. The lowest MSPE is achieved

by the WALS estimator with Laplace prior, but we see that our approach has good performance irrespective of the prior. The relative advantages of the Laplace, Weibull, and Pareto priors may be due to their better properties in terms of minimax regret, which lead in this case to a smaller maximum of the squared prediction error. Of the five BMA estimators under consideration the best is BMA_3, but we also observe an interesting reduction of the MSPE in the comparison of BMA_1 and BMA_2. Hence, other things being equal, a complete enumeration of the model space may improve the prediction performance of model averaging estimators. This issue has received some attention in the frequentist and Bayesian model averaging literatures (see, e.g., Steel 2020). WALS is different because it explores the entire model space after the semi-orthogonal transformation of the regressors in equation (2).

Let us consider next the MAPE:

```
. clear
. quietly set obs `Obs´
. quietly svmat double MAPE, names(col)
. tabstat bma?, stat(mean) format(%6.4f)
```

Stats	bma1	bma2	bma3	bma4	bma5
Mean	0.2078	0.2066	0.1878	0.1909	0.2007

```
. tabstat wals?, stat(mean) format(%6.4f)
```

Stats	wals1	wals2	wals3	wals4	wals5
Mean	0.1928	0.1926	0.1928	0.1941	0.1946

WALS has good performance also with respect to this alternative criterion, but now BMA_3 and BMA_4 are even better than the five WALS estimators considered here. This is not surprising because the WALS estimator is primarily concerned with the MSE in the estimation of the focus parameters.

We conclude our analysis with the coverage of the 90% prediction intervals:

```
. clear
. quietly set obs `Obs´
. quietly svmat double PI_C, names(col)
. tabstat bma?, stat(mean) format(%6.4f)
```

Stats	bma1	bma2	bma3	bma4	bma5
Mean	0.8085	0.8298	0.9149	0.8723	0.8298

```
. tabstat wals?, stat(mean) format(%6.4f)
```

Stats	wals1	wals2	wals3	wals4	wals5
Mean	0.8511	0.8723	0.8723	0.8723	0.8723

The estimated coverage of the WALS prediction intervals varies between 85% and 87%, which is close to the theoretical value of 90%. This result suggests that the choice of the

prior distribution has little impact on the coverage of the WALS prediction intervals, likely due to the fact that these intervals are based on the bias-corrected posterior mean. The credible intervals from BMA_1, BMA_2 and BMA_5 have somewhat larger coverage errors. Based on two-sample splitting, Example 16 in the Stata manual for `bmaregress` reports a predictive coverage of 87% for BMA_5. Unfortunately, this approach is sensitive to the choice of the random-number seed and our leave-one-out cross validation results reveal that the credible interval from BMA_5 can be subject to a larger coverage error. Similarly, for the same study, Raftery et al. (1997) report a predictive coverage of 80% for Occam's window algorithm and 67% or below for various other model selection procedures.

7 Conclusions

In this paper, we discussed version 3.0 of the `wals` command for WALS estimation of linear regression models with i.i.d. errors. Version 3.0 improves earlier versions of `wals` in several important respects, both theoretical and practical. Moreover, we developed new post-estimation commands that are likely to play an important role in applications of the WALS approach. Last, we compared the WALS approach with two recent suites of Stata commands for dealing with model uncertainty: LASSO methods for inference and BMA.

We are presently working on the implementation of various extensions of the WALS theory, where we introduce Stata commands for WALS estimation of linear models with nonspherical errors, fixed-effect and random-effects panel data models, and generalized linear models. The `wals` command described in the current paper serves as a baseline routine for these extensions. Moreover, `lcwals` and `margwals` can be used as post-estimation commands for the whole suite of WALS commands. We hope to present these new Stata commands in a sequel to the current paper.

8 Acknowledgments

Giuseppe De Luca acknowledges financial support by the European Union - Next Generation EU, in the framework of the GRINS - Growing Resilient, Inclusive and Sustainable Project (GRINS PE00000018 – CUP E63C22002140007). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, and the European Union cannot be held responsible for them. We are grateful to Federico Belotti, Domenico Depalo, and Franco Peracchi for helpful discussions, and to the editor and an anonymous referee for their constructive comments.

9 Programs and supplemental material

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

```
. net sj XX
. net install stXXX (to install program files, if available)
. net get stXXX (to install ancillary files, if available)
```

10 References

Bhadra, A., J. Datta, N. G. Polson, and B. Willard. 2017. Horseshoe regularization for feature subset selection. ArXiv preprint arXiv:1702.07400.

Carvalho, C. M., N. G. Polson, and J. G. Scott. 2010. The horseshoe estimator for sparse signals. *Biometrika* 97: 465–480.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. B. Hansen, W. K. Newey, and J. M. Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *Econometrics Journal* 21: C1–C68.

Danilov, D. 2005. Estimation of the mean of a univariate normal distribution when the variance is not known. *Econometrics Journal* 8: 277–291.

Dardanoni, V., S. Modica, and F. Peracchi. 2011. Regression with imputed covariates: A generalized missing indicator approach. *Journal of Econometrics* 162: 362–368.

De Luca, G., and J. R. Magnus. 2011. Bayesian model averaging and weighted-average least squares: Equivariance, stability, and numerical issues. *Stata Journal* 11: 518–544.

De Luca, G., J. R. Magnus, and F. Peracchi. 2018. Weighted-average least squares estimation of generalized linear models. *Journal of Econometrics* 204: 1–17.

———. 2022. Sampling properties of the Bayesian posterior mean with an application to WALS estimation. *Journal of Econometrics* 230: 299–317.

———. 2023. Weighted-average least squares (WALS): Confidence and prediction intervals. *Computational Economics* 61: 1637–1664.

———. 2025. Bayesian estimation of the normal location model: A non-standard approach. *Oxford Bulletin of Economics and Statistics* Forthcoming.

Duval, R., D. Furceri, and J. Miethe. 2021. Robust political economy correlates of major product and labor market reforms in advanced economies: Evidence from BAMLE for logit models. *Journal of Applied Econometrics* 36: 98–124.

Ehrlich, I. 1973. Participation in illegitimate activities: A theoretical and empirical investigation. *Journal of Political Economy* 81: 521–565.

Giannone, D., M. Lenza, and G. E. Primiceri. 2021. Economic predictions with big data: The illusion of sparsity. *Econometrica* 89: 2409–2437.

Kumar, K., and J. R. Magnus. 2013. A characterization of Bayesian robustness for a normal location parameter. *Sankhya (Series B)* 75: 216–237.

Leeb, H., and B. M. Pötscher. 2006. Performance limits for estimators of the risk or distribution of shrinkage-type estimators, and some general lower risk-bound results. *Econometric Theory* 22: 69–97.

Madigan, D., and J. York. 1995. Bayesian graphical models for discrete data. *Journal of Statistical Review* 63: 215–232.

Magkonis, D., K.-M. Zekente, and V. Logothetis. 2021. Does the left spend more? An econometric survey of partisan politics. *Oxford Bulletin of Economics and Statistics* 83: 1077–1099.

Magnus, J. R. 2002. Estimation of the mean of a univariate normal distribution with known variance. *Econometrics Journal* 5: 225–236.

Magnus, J. R., and G. De Luca. 2016. Weighted-average least squares: A review. *Journal of Economic Surveys* 30: 117–148.

Magnus, J. R., and J. Durbin. 1999. Estimation of regression coefficients of interest when other regression coefficients are of no interest. *Econometrica* 67: 639–643.

Magnus, J. R., O. Powell, and P. Prüfer. 2010. A comparison of two model averaging techniques with an application to growth empirics. *Journal of Econometrics* 154: 139–153.

Magnus, J. R., A. T. K. Wan, and X. Zhang. 2011. Weighted average least squares estimation with nonspherical disturbances and an application to the Hong Kong housing market. *Computational Statistics & Data Analysis* 55: 1331–1341.

Magnus, J. R., and W. Wang. 2014. Concept-based Bayesian model averaging and growth empirics. *Oxford Bulletin of Economics and Statistics* 76: 874–897.

Magnus, J. R., W. Wang, and X. Zhang. 2016. Weighted-average least squares prediction. *Econometric Reviews* 35: 1040–1074.

Picchio, M., and M. Ubaldi. 2023. Unemployment and health: A meta-analysis. *Journal of Economic Survey* In press, DOI: 10.1111/joes.12588.

Raftery, A. E., D. Madigan, and J. A. Hoeting. 1997. Bayesian model averaging for linear regression models. *Journal of the American Statistical Association* 92: 179–191.

Seya, H., and M. Tsutsumi. 2012. Application of model averaging techniques to spatial hedonic land price models. In *Econometrics: New Developments*, ed. S. A. Mendez and A. M. Vega, 63–88. New York: NOVA Science Publishers.

Steel, M. F. J. 2020. Model averaging and its use in economics. *Journal of Economic Literature* 58: 644–719.

Vandaele, M. 1978. Participation in illegitimate activities: Ehrlich revisited. In *Deterrence and Incapacitation: Estimating the Effects of Criminal Sanctions on Crime Rates*, ed. A. Blumstein, J. Cohen, and D. Nagin, 270–335. Washington, DC: National Academy of Sciences.

About the authors

Giuseppe De Luca is full professor of econometrics at the Department of Economics, Business and Statistics (SEAS) of the University of Palermo.

Jan R. Magnus is emeritus professor of econometrics at Tilburg University, and extraordinary professor at the Vrije Universiteit Amsterdam, both in The Netherlands.