Revealing Volatility Dynamics in ARCH-GARCH Models

Allyson L. Benton
Department of Government
University of Essex
Colchester, UK
allyson.benton@essex.ac.uk

Soren Jordan
Department of Political Science
Texas A&M University
College Station, Texas, USA
sorenjordanpols@gmail.com

Andrew Q. Philips
Department of Political Science
University of Colorado Boulder
Boulder, Colorado, USA
andrew.philips@colorado.edu

Forthcoming in *The Stata Journal*

Software availability

The Stata commands `gbmake` (GARCH Bootstrap Make) and `gbplot` (GARCH Bootstrap Plot) introduced in this article are freely available at:

- Documentation and installation:
  `https://andyphilips.github.io/gbmake-gbplot/`
- GitHub repository:
  `https://github.com/andyphilips/gbmake-gbplot`

Revealing Volatility Dynamics in ARCH-GARCH Models

Allyson L. Benton
Department of Government
University of Essex
Colchester, UK
allyson.benton@essex.ac.uk

Soren Jordan
Department of Political Science
Texas A&M University
College Station, Texas, USA
sorenjordanpols@gmail.com

Andrew Q. Philips
Department of Political Science
University of Colorado Boulder
Boulder, Colorado, USA
andrew.philips@colorado.edu

short author list: A.L. Benton, S. Jordan, A.Q. Philips
short title: Volatility Dynamics in ARCH-GARCH Models
Abstract. ARCH-GARCH models are notoriously difficult to interpret. Benton et al. (2026) show that interpreting the regressors in the heteroskedastic variance equation is particularly challenging and offer an approach to gaining inferences about these regressors beyond their sign and statistical significance. This manuscript implements their proposed strategy for dynamic inference through two commands: `gbmake` and `gbplot`. `gbmake` generates simulated parameter values from an estimated ARCH-GARCH model, using either a parametric- or residual-based bootstrap. `gbplot` produces dynamic simulations using these bootstrapped values or through the delta method for approximating uncertainty. An applied example illustrates how these tools can help users better understand the effects of ARCH-GARCH model regressors on the conditional error variance.
Keywords: ARCH; GARCH; variance; volatility; dynamic simulations

1 Introduction

Time series data often exhibit time-varying variance: stock return volatility clusters in periods of calm and turbulence, government approval ratings become more erratic during crises, and budget allocations shift between stable and volatile periods. Understanding these dynamics requires not just modeling the conditional mean of the time series but also its volatility, captured by the conditional error variance. Autoregressive Conditional Heteroskedasticity (ARCH) (Engle 1982) and Generalized ARCH (GARCH) (Bollerslev 1986) models simultaneously estimate both, allowing researchers to examine how regressors affect both the *level* of a series and its *volatility*. Applications

of the ARCH-GARCH approach span public opinion (e.g., Hellwig 2007), political economy (e.g., Benton and Philips 2020), budgetary politics (e.g., Flink 2017), financial economics (e.g., Kumar et al. 2022), commodity pricing (e.g., Sadorsky 1999), computational finance (e.g., Naimy et al. 2021), risk management (e.g., Léber and Egyed 2025), and environmental sciences (e.g., Yao et al. 2022), among other fields.

Stata's `arch` command provides a flexible framework for estimating ARCH-GARCH models, including extensions such as T-GARCH and E-GARCH. As with any estimation command, `arch` reports coefficient estimates, standard errors, and significance tests for all parameters in both the conditional mean and conditional error variance equations. After estimation, users can generate point predictions of both the conditional mean (via `predict, xb` or `predict, y` for the differenced or levels series) and the conditional error variance (via `predict, variance`) or recover contemporaneous marginal effects on either quantity using `margins` with the corresponding statistic from `predict`. Each of these tools, however, describes the conditional mean or error variance, or a regressor's effect on it, in a single time period. Because ARCH-GARCH models are dynamic, any shift in a regressor also propagates forward through both the mean and variance in subsequent periods. Neither the tabular ARCH-GARCH output nor existing post-estimation tools reveal how these effects evolve, how long they persist, or whether they remain statistically significant over time.

Testing theories of volatility requires drawing these substantive inferences about regressors' effects. For the conditional mean, community-contributed packages trace the dynamic path of a regressor's impact (e.g., Williams and Whitten 2012; Philips et al. 2016; Jordan and Philips 2018). For the conditional error variance, however, no similar package exists. Instead, scholars have been limited to interpretation of the estimated parameters—including the sign and significance of regressor coefficients and the magnitude and significance of the GARCH term—and to the post-estimation point predictions noted above. Neither is able to show how a temporary shift in a specific regressor propagates through the conditional error variance over time. These limitations are compounded by the structure of the conditional error variance equation. Because

Stata's `arch` command uses an exponential link function to ensure that the conditional error variance remains positive, the effect of any regressor on expected volatility is non-linear and depends on the values of all other regressors and the GARCH persistence parameter. Substantive interpretation of ARCH-GARCH conditional error variance equation results thus requires more than existing tabular output and post-estimation point predictions provide. More generally, quantities of interest such as expected values provide more interpretable substantive answers than tables of coefficients, and are easier to communicate (King et al. 2000; Kropko and Harden 2020).

We introduce two new Stata commands for simulating and visualizing the expected conditional error variance estimated via ARCH-GARCH models: GARCH Bootstrap Make (`gbmake`) and GARCH Bootstrap Plot (`gbplot`). `gbmake` performs two bootstrapping procedures, introduced by Benton and Philips (2020) and developed by Benton et al. (2026), to simulate parameter values from an estimated ARCH-GARCH model and quantify uncertainty about predictions from the conditional error variance equation. `gbplot` uses these bootstrapped values to generate dynamic simulations of the conditional error variance with associated measures of uncertainty and produces visualizations of the results. `gbplot` also implements the delta method for quantifying uncertainty directly, providing an analytic alternative to the bootstrap options. We offer both bootstrap and analytic approaches to accommodate differences in computational resources (bootstrapping is much more intensive) and the need to balance efficiency against the distributional assumptions required for analytic estimates (the delta method requires more assumptions). `gbplot` is highly flexible: users can examine any impulse-response path by setting regressors manually through Stata's matrix capabilities. Below, after briefly introducing ARCH-GARCH models, we show how `gbmake` and `gbplot` reveal substantive dynamics that tabular results obscure.

2 Modeling conditional heteroskedasticity using ARCH-GARCH

ARCH-GARCH models (Engle 1982; Bollerslev 1986) simultaneously estimate two equations: a conditional mean equation describing the evolution of the mean of y_t and a conditional variance equation describing the evolution of the error variance of y_t.

The conditional mean equation can be expressed using a standard time series form, such as (but not exclusively) a first-order autoregressive AR(1) process:

$$y_t = \beta_0 + \phi y_{t-1} + \mathbf{x}_t\boldsymbol{\beta} + \varepsilon_t, \tag{1}$$

where y_t is a stationary dependent variable, $\mathbf{x}_t$ is a vector of exogenous regressors with coefficients $\boldsymbol{\beta}$, and ε_t is an error term with mean zero and time-varying variance σ_t^2.

The main innovation of the ARCH-GARCH framework is to recognize and model this conditional, time-varying error variance σ_t^2 as:

$$\sigma_t^2 = \omega_0 + \omega_1\varepsilon_{t-1}^2 + \alpha_1\sigma_{t-1}^2, \tag{2}$$

where ω_1 is the ARCH term capturing how unanticipated shocks in the prior period affect subsequent error variance and α_1 is the GARCH term capturing persistence in volatility.[1] When α_1 is close to zero, shocks to volatility dissipate quickly; when α_1 is close to one, shocks persist over time. Equation 2 specifies a first-order $(t-1)$ lag for both the ARCH and GARCH terms, but higher order processes are possible ($\omega_q\varepsilon_{t-q}^2$ and $\alpha_p\sigma_{t-p}^2$ on the ARCH and GARCH terms, respectively). The most common specification is $p = q = 1$, or the GARCH(1,1).

Equation 2 models time-varying variance as a function of its own history, but regressors can also enter the equation, allowing researchers to model volatility with the-

[1] The standard restrictions $\omega_0 > 0$, $0 \leq \omega_1 < 1$, $0 \leq \alpha_1 < 1$, and $(\omega_1 + \alpha_1) < 1$ ensure that σ_t^2 remains positive and stationary (Bollerslev 1986). If bootstrap draws violate these restrictions, `gbmake` and `gbplot` warn the user.

oretically motivated predictors. The GARCH(1,1) with regressors is typically written in econometrics textbooks as:

$$\sigma_t^2 = \omega_0 + \omega_1 \varepsilon_{t-1}^2 + \alpha_1 \sigma_{t-1}^2 + \mathbf{z}_t \boldsymbol{\gamma}, \tag{3}$$

where $\mathbf{z}_t$ is a vector of regressors with coefficients $\boldsymbol{\gamma}$. In practice $\mathbf{z}_t$ often shares regressors with $\mathbf{x}_t$ (from the conditional mean equation 1), but this overlap is not required (and our commands below impose no such restriction).

Because the conditional error variance must remain strictly positive, regressors cannot enter the equation linearly as in Equation 3; linear entry could produce negative fitted values of σ_t^2 whenever elements of $\boldsymbol{\gamma}$ are negative. Stata's `arch` command addresses this by incorporating regressors and the constant through an exponential link function, yielding the *applied* GARCH(1,1) specification:

$$\sigma_t^2 = \omega_1 \varepsilon_{t-1}^2 + \alpha_1 \sigma_{t-1}^2 + \exp(\omega_0 + \mathbf{z}_t \boldsymbol{\gamma}). \tag{4}$$

Exponentiation ensures that the contributions of regressors and the constant to σ_t^2 remain positive, while allowing the coefficients $\boldsymbol{\gamma}$ to take negative or positive values. This is the specification estimated when users include regressors in the conditional error variance equation via the `het` option of the `arch` command.

Yet, like any link function, exponentiation complicates interpretation of the tabular results for the conditional error variance equation. In Equation 4, the effect of any single regressor z on the conditional error variance σ_t^2 depends on the values at which z and the other regressors in $\mathbf{z}_t$ are held, the magnitude and direction of the shock to z, and the GARCH persistence parameter α_1, which governs how long the effect persists. Even seemingly marginal changes in the values of other regressors can therefore produce large differences in the estimated effect of a regressor of interest, as Benton et al. (2026, p. 22) demonstrate (see their Figure 1). Without rigorous interpretive tools, practitioners are left with little more than the sign and significance of the parameters

for their variables of interest, which limits the interpretation of results.

3 Simulating conditional heteroskedasticity under ARCH-GARCH

A standard approach for addressing this limitation in other contexts (such as in non-linear maximum-likelihood models, which also use link functions) has been to simulate quantities of interest along with measures of uncertainty (King et al. 2000; Arel-Bundock et al. 2024). In the ARCH-GARCH context, we can use the same approach to draw reliable substantive inferences: simulating the expected conditional error variance under counterfactual regressor scenarios along with measures of uncertainty. Benton et al. (2026) thus propose interpreting ARCH-GARCH results by constructing counterfactual predictions of the conditional error variance. We implement three of their approaches: the `parametric` bootstrap, the `residual` bootstrap, and the `delta` method approximation.[2]

The `parametric` bootstrap draws simulated parameter values from a multivariate normal distribution centered on the original point estimates, with the estimated variance–covariance matrix supplying its covariance. Let $\hat{\boldsymbol{\theta}}$ include all estimated parameters in the conditional variance equation in Equation 3, including the constant $\hat{\omega}_0$, the ARCH $\hat{\omega}_1$ and GARCH $\hat{\alpha}_1$ terms, and the regressor coefficients $\hat{\boldsymbol{\gamma}}$. The parametric bootstrap draws $\tilde{\boldsymbol{\theta}}_b \sim MVN(\hat{\boldsymbol{\theta}}, \hat{\mathbf{V}}(\hat{\boldsymbol{\theta}}))$. For each draw, the expected conditional error variance is then constructed forward in time: regressors are fixed at baseline values (typically means for continuous regressors and modes for categorical ones), a burn-in period is used to get a stable starting value, and at a chosen simulation period a counterfactual shock is imposed on the regressor of interest (such as a one-unit or one standard deviation increase or decrease). In subsequent periods, the shocked regressor

[2]The typeface reflects that these are arguments to `gbmake` and `gbplot`, as shown below. We do not implement Benton et al.'s (2026) maximum entropy bootstrap (Vinod and Lopez-de Lacalle 2009), since the package they use is not available in Stata. Their simulation results indicate that each approach performs equivalently, so this is not a meaningful limitation of our commands.

either returns to the baseline (an impulse) or is held at its new value (a permanent shift), and the conditional error variance evolves accordingly through the GARCH term. Uncertainty is captured through the estimated covariance of the coefficients. This approach is computationally fast, since it requires estimating the ARCH-GARCH model only once.

The `residual` bootstrap resamples the rescaled residuals from the original model to construct new realizations of the dependent variable, then re-estimates the ARCH-GARCH model on each replicate. The residual term is rewritten as $\varepsilon_t = \sigma_t \nu_t$, where $\nu_t \sim N(0,1)$ is the white-noise component and σ_t the time-varying scale. The standardized residuals $\hat{\nu}_t = \hat{\varepsilon}_t / \hat{\sigma}_t$ are then recentered to mean zero, $\tilde{\nu}_t = \hat{\nu}_t - \frac{1}{T-1}\sum_{t=2}^{T} \hat{\nu}_t$, where T is the total number of time periods. These form an empirical distribution $\hat{\mathcal{F}}$ from which B bootstrap error series $\tilde{\nu}_t$ are drawn and used to recursively construct synthetic series $\tilde{y}_{b,t}$, on each of which the ARCH-GARCH model is re-estimated to obtain bootstrapped parameters $\tilde{\boldsymbol{\theta}}_b$. Each $\tilde{\boldsymbol{\theta}}_b$ is then used to construct the expected conditional error variance forward in time under the same counterfactual scenario, with the shocked regressor either returning to baseline (impulse) or persisting (permanent shift). Uncertainty is captured through repeated bootstrapped realizations of the series. Since B ARCH-GARCH models are estimated (to obtain a distribution of parameters), the residual-based bootstrap is computationally intensive. However, it avoids distributional assumptions about the parameter estimates and thus produces better-calibrated confidence intervals than the `parametric` bootstrap or `delta` method approximation (Benton et al. 2026).

The `delta` method is not a bootstrap. Let $g(\hat{\boldsymbol{\theta}})$ denote the predicted conditional error variance as a nonlinear function of the estimated parameters $\hat{\boldsymbol{\theta}}$. The delta method approximates $\widehat{\mathrm{Var}}[g(\hat{\boldsymbol{\theta}})]$ using the estimated variance–covariance matrix of $\hat{\boldsymbol{\theta}}$ and the partial derivatives of $g(\hat{\boldsymbol{\theta}})$, with uncertainty intervals then constructed from $\sqrt{\widehat{\mathrm{Var}}[g(\hat{\boldsymbol{\theta}})]}$ and the appropriate t-quantile. The expected conditional error variance is constructed forward in time under the same counterfactual scenario as the bootstraps, with the delta method supplying the standard errors around each prediction analytically rather

than through repeated draws. The main drawback of this approach is that the resulting confidence intervals can fall below zero, which is nonsensical given that the conditional variance must be strictly positive (Benton et al. 2026). However, like the parametric bootstrap, this requires estimating the ARCH-GARCH model only once and is computationally fast.

Benton et al. (2026) provide code in their replication materials for implementing these three approaches on chosen examples, but they do not provide general-purpose commands that practitioners can apply in their research. Our `gbmake` and `gbplot` commands fill this gap, making it easy for any practitioner with an ARCH-GARCH model to obtain robust substantive inferences of the conditional error variance. `gbmake` and `gbplot` offer three benefits over interpreting tabular output alone. First, they trace the entire temporal path of a shock to one or more regressors as it propagates through the conditional error variance. Users can assess the initial magnitude of the effect, how quickly it dissipates, and whether it remains statistically significant in subsequent periods. Second, they quantify uncertainty around this path, either from percentiles of bootstrapped simulations or analytically via the delta method. Third, they allow users to compare scenarios directly: temporary, permanent, or user-designed shocks; positive versus negative shocks; and shocks under different baseline values of other regressors. These comparisons reveal asymmetries and conditional effects otherwise invisible in tabular output. We describe the syntax and options for each in the following sections.

4 Simulating and visualizing conditional heteroskedasticity using `gbmake` and `gbplot`

4.1 The `gbmake` (GARCH Bootstrap Make) command

4.1.1 Syntax for `gbmake`

`gbmake` [, `btype(`*type*`)` `reps(#)` `saving(`*filename*`)` `replace` `coldstart` `fcast_arch`]

4.1.2 Options for `gbmake`

`btype(`*type*`)` can be either `parametric` for the parametric-based bootstrap (which is the default if not specified) or `residual` for the residual-based bootstrap.[3]

`reps(#)` allows users to customize the number of B bootstraps. By default, this is set at $B = 1000$ for the parametric-based bootstrap and $B = 100$ for the residual-based bootstrap. Higher values should result in greater precision, but at the cost of increased computational time (most notably for the `residual` bootstrap).

`saving(`*filename*`)` specifies the "filename" to save the bootstrapped parameter values for use in `gbplot`. By default this dataset is named "gbmake_results.dta".

`replace` replaces the "filename" above, if provided. Otherwise, the program will not overwrite an existing dataset associated with "filename" in the working directory.

`coldstart` applies only to `btype(residual)`. By default, `gbmake` uses the `arch` estimates as starting values when re-estimating the model on each bootstrap replicate. `coldstart` uses the original `arch` starting values instead for each replicate.

`fcast_arch` applies only to `btype(residual)`. This option saves the recentered standardized residuals that make up $\hat{\mathcal{F}}$ as the variable `gb_innov` in the saved

[3]The delta method approach is integrated in the second command: `gbplot`; users wishing to utilize the `delta` method should run `gbplot` immediately after `arch`.

output dataset for future use with `gbplot`'s `fcast_arch_resid` forecasting option.

4.1.3 Description for `gbmake`

`gbmake` requires that the user first estimate a GARCH(1,0) or GARCH(1,1) using Stata's `arch` command, with a list of $\mathbf{z}_t$ regressors in the heteroskedastic variance equation (via `het`). Contemporaneous ARCH-in-mean terms are allowed, but AR and MA terms, along with "boutique" ARCH models such as AAARCH, PARCH, APARCH, etc. are not allowed. Regressors in `het()` cannot use time-series or factor-variable operators (e.g., `L.x`, `D.x`, or `i.x`).

If `btype(parametric)` is specified, `gbmake` performs the parametric-based bootstrap described above, drawing B vectors $\tilde{\boldsymbol{\theta}}_b$ from a multivariate normal distribution:

$$\tilde{\boldsymbol{\theta}}_b \sim MVN(\hat{\boldsymbol{\theta}}, \hat{\mathbf{V}}(\hat{\boldsymbol{\theta}})), \tag{5}$$

where $\hat{\boldsymbol{\theta}}$ are the original parameter estimates from the conditional error variance equation and $\hat{\mathbf{V}}(\hat{\boldsymbol{\theta}})$ is their estimated variance–covariance matrix. The $\tilde{\boldsymbol{\theta}}_b$ parametric bootstraps are stored as a $B \times k$ dataset containing bootstraps for all k parameters appearing in Equation 4, for use by `gbplot`. By default this dataset is saved as "gbmake_results.dta," but the "filename" can be changed using the `saving()` option.

If `btype(residual)` is specified, `gbmake` performs the residual-based bootstrap described above. The program computes and recenters the rescaled residuals to form the empirical distribution $\hat{\mathcal{F}}$. If `fcast_arch` is specified, these recentered residuals are also saved in the output dataset later forecasting with `gbplot`. It then repeats the following process B times:

1. Draw a bootstrap residual series $\tilde{\mathrm{v}}_t$ of length $T-1$ from $\hat{\mathcal{F}}$, where T is the total number of time periods.

2. Recursively construct synthetic time series for both the variance and the dependent variable. To do this:

(a) The recursion is initialized at $t = 1$, since values for both the variance and the rescaled residuals (as well as the dependent variable if a lag appears in the conditional mean equation) are unknown at $t = 0$ but are needed to create $\tilde{\sigma}_1^2$. `gbmake` initializes the bootstrap by drawing $\tilde{\sigma}_0^2$ and $\tilde{y}_0$ jointly from a common time point in the original series, and drawing $\tilde{\nu}_0$ independently from $\hat{\mathcal{F}}$.[4] The bootstrap values at $t = 1$ are then:

$$\tilde{\sigma}_1^2 = \hat{\omega}_1(\tilde{\nu}_0^2\tilde{\sigma}_0^2) + \hat{\alpha}_1\tilde{\sigma}_0^2 + \exp(\hat{\omega}_0 + \mathbf{z}_1\hat{\boldsymbol{\gamma}}) \tag{6}$$

$$\tilde{y}_1 = \mathbf{x}_1\hat{\boldsymbol{\beta}} + \tilde{\nu}_1\tilde{\sigma}_1 \tag{7}$$

where $\hat{\omega}_1$, $\hat{\alpha}_1$, $\hat{\omega}_0$, and $\hat{\boldsymbol{\gamma}}$ are the original parameter estimates called from the estimated `arch` model, and $\mathbf{z}_1$ are the observed regressor values at $t = 1$. Equation 7 uses $\sqrt{\tilde{\sigma}_1^2}$ from Equation 6. If a lagged dependent variable appears in the conditional mean equation, bootstrap values of $\tilde{\sigma}_0^2$ and $\tilde{y}_0$ are drawn jointly so that both come from the same time point.

(b) For $t = 2, \ldots, T$, the recursion uses bootstrap residuals at each time point, with lagged values now known from the prior period:

$$\tilde{\sigma}_t^2 = \hat{\omega}_1(\tilde{\nu}_{t-1}^2\tilde{\sigma}_{t-1}^2) + \hat{\alpha}_1\tilde{\sigma}_{t-1}^2 + \exp(\hat{\omega}_0 + \mathbf{z}_t\hat{\boldsymbol{\gamma}}) \tag{8}$$

$$\tilde{y}_t = \mathbf{x}_t\hat{\boldsymbol{\beta}} + \tilde{\nu}_t\tilde{\sigma}_t \tag{9}$$

where regressors $\mathbf{x}_t$ and $\mathbf{z}_t$ are held fixed at their observed values across all bootstraps.

3. Re-estimate the ARCH-GARCH model on the synthetic series $\tilde{y}_{b,t}$, keeping $\mathbf{x}_t$ and $\mathbf{z}_t$ fixed. Each realization of $\tilde{y}_{b,t}$ produces a new set of variance-equation param-

[4] E.g., if the joint draw selects $t = 45$, then $\tilde{\sigma}_0^2 = \sigma_{45}^2$ and $\tilde{y}_0 = y_{45}$. Bootstrap draws of $\tilde{\nu}_t$ are independent because $\tilde{\nu}_t$ is white noise by construction.

eter estimates, and the set of estimates across B replicates forms an empirical distribution of the bootstrap parameters.

4. Save the estimated variance-equation parameters as row b of the $B \times k$ output dataset.

The `btype(residual)` option re-estimates the ARCH-GARCH model on each replicate, but some of these estimations may fail to converge, as is common when estimating ARCH-GARCH models. The program will continue to draw new residual-based bootstrap samples until it obtains the user-specified number of converged B `reps()`. The total number of estimation attempts is therefore $B + r$, where r is the number of failed attempts, up to a maximum of $B \times 3$ attempts. `gbmake` flags failures with a red "X" in the console (similar to Stata's `bootstrap` command). If repeated convergence issues arise, users may specify `coldstart` and/or adjust the convergence options in the original `arch` call (e.g., starting values), as these settings are inherited by `gbmake`. The resulting dataset produced by `btype(parametric)` or `btype(residual)` is saved as "gbmake_results.dta" by default, although the "filename" can be changed via the `saving()` option, as noted. These data are then used to quantify and visualize uncertainty with `gbplot`.

4.2 The `gbplot` (GARCH Bootstrap Plot) command

4.2.1 Syntax for `gbplot`

`gbplot [if] [in] , [delta using(`*filename*`) shockvariable(`*varname*`) shockvalue(#) customvalues(`*numlist*`) burnin(#) shocktime(#) permanent range(#) usematrix(`*matname*`) psv psv_usematrix(#) level(#) saving(`*filename*`) replace noplot areaplot fcast_epu fcast_arch fcast_arch_resid]`

4.2.2 Options for `gbplot`

`gbplot` can simulate a shock in two ways, which we refer to as *Method 1* and *Method 2*. *Method 1* applies a single shock to one regressor, specified with `shockvariable()` and `shockvalue()`, while holding all other regressors at baseline values. *Method 2* lets the user set the value of every regressor in every period through a matrix supplied with `usematrix()`. Some options apply to only one of the two methods, as noted below. Section 4.2.3 describes both methods in detail.

`delta` applies the delta method (instead of one of the bootstrapping approaches) to approximate the standard errors around the predicted conditional error variance. This is one of the approaches proposed by Benton et al. (2026), and can be used with either Method 1 or Method 2.

`using(`*filename*`)` specifies the dataset created by `gbmake` to be used when applying either the `parametric` or `residual` bootstrap-based approach.

`shockvariable(`*varname*`)` is the regressor in **z** (the `het` equation of the original `arch` model) whose impulse-response the user wishes to examine. This is a required option if using Method 1 and is ignored if using Method 2.

`shockvalue(#)` is the size of the shock applied to `shockvariable()` and may be positive or negative in value. This is a required option if using Method 1 and is ignored if using Method 2.

`customvalues(`*numlist*`)` specifies the baseline values at which the regressors in **z** are held during the simulation, one value per regressor, replacing the default of holding each regressor at its estimation-sample mean. This is optional under Method 1 and ignored under Method 2.

`burnin(#)` specifies the number of burn-in periods, which precede the periods shown in the plot (and are therefore omitted from the plot). A burn-in allows the conditional error variance to settle to a stable value before the simulation begins and a shock is applied. Under Method 1, `burnin()` sets how many burn-in

periods are simulated and defaults to 20 if no burn-in is specified. Under Method 2, the burn-in periods must be included directly in the user-supplied matrix as leading rows in which all regressors in **z** are held constant (at values chosen by the user), with `burnin()` then used to indicate to `gbplot` how many leading rows are to be treated as burn-in periods (and also to be omitted from the plot). Users should ensure that their pre-shock simulated values are stable before implementing a shock.

`shocktime(#)` is the simulation period at which the shock/impulse will occur. This option applies only to Method 1, and by default is set to $m = 5$ (post burn-in) simulation periods if not specified by the user. It cannot exceed `range()`.

`permanent` makes the shock to `shockvariable()` persist for all periods after `shocktime()`, rather than for a single period (the default). This option applies only to Method 1 and is ignored under Method 2.

`range(#)` is an option to change the total range of (post burn-in) periods plotted. This option applies only to Method 1, and by default is set to $M = 15$ simulation periods if not specified by the user.

`usematrix(`*matname*`)` specifies the user-created $m \times k$ matrix used when deploying Method 2, where the m rows are the simulation time periods (plus burn-in) and the k columns are the regressors in **z**. Element z_{mk} gives the value of regressor z_k at simulation time period m. When constructing this matrix, users should ensure that it includes enough pre-shock periods for the simulated variance to burn-in to a stable value, as well as enough post-shock periods for it to equilibrate. These lengths can be gauged first by using Method 1 with the options `burnin()` and `range()`. `usematrix()` can be used with `burnin()`, which specifies the number of pre-shock simulation periods (rows) m in the matrix constituting the burn-in.

`psv` expresses plotted predictions as a percentage relative to the "pre-shock value" of the variance, rather than the default expected value of the variance.

ARCH-GARCH models explain changes in variance which, unlike the level of a

variable, has no natural metric. We can standardize these changes by expressing the post-shock values of the expected variance relative to its pre-shock expected value. For Method 1, the pre-shock value is the expected variance in the period immediately prior to the shock. `psv` can also be used with Method 2, where it defaults to the first row of `usematrix()` (where $m = 1$) as the reference; `psv_usematrix()`, described next, provides another, more appropriate approach.

`psv_usematrix(#)` is the Method 2 counterpart to `psv`. It is used to specify which row of the user-created $m \times k$ matrix from `usematrix()` to use as the pre-shock reference value when expressing plotted predictions as a percentage relative to the "pre-shock value" of the variance. For users building a burn-in period into the matrix, as recommended for Method 2, the first row $m = 1$ is unlikely to represent a stable pre-shock value of the conditional error variance. Instead, a more appropriate approach is to identify a row after the burn-in but immediately before the shock as the pre-shock value, which this option allows. (If `usematrix()` and `psv` are specified without `psv_usematrix()`, the program uses the first row $m = 1$ of the simulation matrix as the reference value.)

`level(#)` allows the user to set significance levels other than the default 95% (for delta method) or 75%, 90%, and 95% (for the bootstraps) when creating confidence intervals. It must be expressed as a proportion (e.g., 0.90).

`saving(`*filename*`)` saves the simulated predictions as an external dataset. This option is useful for users who wish to create custom plots, including combining multiple plots together, or who wish to use the predictions in other plotting programs.

`replace` overwrites the existing dataset when using `saving()`, if present.

`noplot` suppresses the creation of the plot, although simulated values are still created, which the user can then save (using `saving()`) for future plots. We imagine that `noplot` and `saving()` might be used by practitioners who wish to use the data (such as from the delta method) to make their own custom plots.

`areaplot` creates an area plot rather than the default spikeplot.

`fcast_epu` forecasts with "expected path uncertainty," where each individual bootstrap b follows its own previous conditional variance to calculate the conditional variance at simulation time m.

`fcast_arch` forecasts with a normalized ARCH effect for each period: the "expected path uncertainty" plus an additional innovation $\sim N(0,1)$, squared and multiplied by the lagged conditional error variance, and entering as the ARCH effect.

`fcast_arch_resid` is the same as `fcast_arch`, but instead of drawing innovations from a standard normal distribution, it draws them from $\hat{\mathcal{F}}$, the recentered standardized residuals saved by `gbmake`'s `fcast_arch` option.

4.2.3 Description for `gbplot`

As noted above, `gbplot` provides two methods for creating simulated impulse-responses, *Method 1* and *Method 2* (these terms are used for exposition only and do not appear in the program's options). *Method 1* builds on the classic idea of an impulse response function in time series: when considering a counterfactual "shock" to a variable, the most common inference concerns a one-unit increase, $\hat{\beta}$, and the two most common counterfactuals are whether that "shock" is temporary (an impulse persisting for one period only) or permanent (a permanent shift persisting for all subsequent periods). *Method 2* allows for more complex shock histories, both within a single variable or across multiple variables. Method 1 is simpler but more limited. Method 2 is more complex but can examine the impulse of multiple regressors at once (e.g., for use when interactions are included). Method 2 requires using Stata's `matrix` functionality.

Before describing these methods, we explain `gbplot`'s `delta` method approximation for quantifying uncertainty. As an alternative to a bootstrap approach, users can specify the `gbplot` option `delta` to approximate confidence intervals via the `delta` method directly from the original `arch` estimates. To do this, the user first estimates the model with `arch` and then immediately runs `gbplot` with the `delta` option, with no interven-

ing estimation. The forward-simulation procedure mirrors the bootstrap case under Method 1 and Method 2, but the prediction at each period m uses the point estimates for the parameters in the conditional error variance equation $\hat{\boldsymbol{\theta}}$, rather than B bootstrap draws, with the standard error around the prediction supplied analytically using the delta method. This approach is computationally fast, but its confidence intervals can extend below zero, which is nonsensical for a variance (Benton et al. 2026).[5] Users wishing to ensure strictly positive confidence intervals should rely on one of the bootstrap approaches instead.

To use Method 1, the user specifies the saved parametric-based or residual-based bootstrap dataset (created by `gbmake`) via `using()`. For each bootstrap row b, `gbplot` substitutes the bootstrap parameters $\tilde{\boldsymbol{\theta}}_b$ from that row and user-chosen values of $\mathbf{z}$ into Equation 4 to produce $\mathrm{E}[\tilde{\sigma}^2_{b,1}]$, the expected conditional error variance for bootstrap b at simulation time $m = 1$. Recall that the parameters in $\tilde{\boldsymbol{\theta}}_b$ include the constant $\tilde{\omega}_{0,b}$, the ARCH and GARCH coefficients $\tilde{\omega}_{1,b}$ and $\tilde{\alpha}_{1,b}$, and the regressor coefficients $\tilde{\boldsymbol{\gamma}}_b$. By default, Method 1 sets all regressors $\mathbf{z}$ to their estimation-sample-specific means $\bar{\mathbf{z}}$ throughout the simulation (though this command respects `[if/in]` statements), with `customvalues()` allowing users to specify other values instead of means. Since the conditional error variance is unknown before this first period, `gbplot` initializes the GARCH term at zero and sets the ARCH term (if present) to zero throughout. In the subsequent period, the expected value of the previous period's conditional variance, taken across B bootstraps, enters as the GARCH value $\mathrm{E}[\tilde{\sigma}^2_{b,1}]$, which is then used to compute $\tilde{\sigma}^2_{b,2}$. Next, $\mathrm{E}[\tilde{\sigma}^2_{b,2}]$ is used to compute $\tilde{\sigma}^2_{b,3}$, and so on. Thus the expected conditional variance is simulated as (see Equation 4):

$$\tilde{\sigma}^2_{b,1} = \tilde{\omega}_{1,b} \cdot 0 + \tilde{\alpha}_{1,b} \cdot 0 + \exp(\tilde{\omega}_{0,b} + \bar{\mathbf{z}}\tilde{\boldsymbol{\gamma}}_b) \tag{10}$$

[5] In addition, we find that for some applications where the expected conditional variance is near-zero, or estimates are highly unstable in the underlying ARCH-GARCH model, we exceed the maximum number of iterations when our program tries to approximate confidence intervals using the delta method through `nlcom`, issuing an error.

$$\tilde{\sigma}^2_{b,2} = \tilde{\omega}_{1,b} \cdot 0 + \tilde{\alpha}_{1,b} \mathrm{E}[\tilde{\sigma}^2_{b,1}] + \exp(\tilde{\omega}_{0,b} + \bar{\mathbf{z}}\tilde{\boldsymbol{\gamma}}_b) \tag{11}$$

$$\tilde{\sigma}^2_{b,3} = \tilde{\omega}_{1,b} \cdot 0 + \tilde{\alpha}_{1,b} \mathrm{E}[\tilde{\sigma}^2_{b,2}] + \exp(\tilde{\omega}_{0,b} + \bar{\mathbf{z}}\tilde{\boldsymbol{\gamma}}_b) \tag{12}$$

and so on, where $\tilde{\omega}_{0,b}$, $\tilde{\omega}_{1,b}$, $\tilde{\alpha}_{1,b}$, and $\tilde{\boldsymbol{\gamma}}_b$ are the elements of $\tilde{\boldsymbol{\theta}}_b$. To reduce the influence of the initial first period "guess" that $\tilde{\sigma}^2_{b,0} = 0$ for simulation period $m = 1$, a burn-in of 20 periods is conducted before simulating the entire range, M, which is $M = 15$ by default (so the user observes M periods after the burn-in has completed). The burn-in can be changed using `burnin()` and M can be increased with `range()`.

The process above produces a perfectly "typical," but static, expected conditional error variance over the simulation periods. To gain dynamic inferences, users specify a "shock." Method 1 requires specifying `shockvariable()`, the user-chosen z regressor about which to observe a hypothetical "shock" of size `shockvalue()` at period `shocktime()` out of a total range of `range()` periods. This shock can be temporary (i.e., for one period) or permanent (i.e., for all periods after `shocktime()`); this is controlled with `permanent`. With Method 1, only the user-chosen `shockvariable()` experiences a change at `shocktime()`, while all other regressors are kept at their means (or fixed at other user-chosen values using `customvalues()`). After all periods M have been simulated, we obtain the median value of $\tilde{\sigma}^2_{b,m}$ in each period, along with confidence intervals calculated using percentiles of the B bootstraps. By default, `gbplot` returns the 2.5, 5, 12.5, 87.5, 95, and 97.5 percentiles, giving 75%, 90%, and 95% confidence intervals around the median, but this can be changed using `level()`.

To summarize, Method 1 generates the substantive response to a shock (of size `shockvalue()`) to a `shockvariable()` that is either temporary or `permanent`. All other variables are held at their means (or at `customvalues()`). However, users might also be interested in understanding effects which are more complex: for example, the effect of a gradual increase in a `shockvariable()`, the effect of a shock that persists for five periods, or the effect of two variables shocked at once. Because these options are not

possible to investigate under Method 1, we provide another method, Method 2, for estimating more complex shocks.

Method 2 generalizes Method 1 by allowing users to set all $\mathbf{z}$ to any values throughout the period of simulation, including multiple, complex shocks. To use Method 2, users must create a matrix of size $m \times k$, where each row m corresponds to a simulation period (including both the burn-in periods and the M post-burn-in periods described in Method 1) and each column k corresponds to a regressor z_k in the conditional error variance equation. Each cell holds the user-chosen value of z_k at simulation period m. The matrix is constructed using Stata's `matrix` command; we provide an illustration in our empirical example in the next section and an additional illustration in the Online Supplementary Material (OSM). This matrix must contain:

1. a column for each regressor in the variance equation, mapped one to one (where the first column represents the first regressor in the variance equation, etc.);

2. enough rows to cover any desired burn-in periods plus the M post-burn-in periods described in Method 1; and

3. desired values for every regressor in every row of the matrix (including both the burn-in rows and the M post-burn-in periods), even if those values do not change over the simulation period.

This matrix is passed to `gbplot` with the option `usematrix()`. Because Method 2 obtains all relevant simulation values from this matrix, the options `shockvariable()`, `shockvalue()`, `shocktime()`, `range()`, and `customvalues()` are ignored. If `burnin()` is supplied, `gbplot` assumes that the first `burnin()` rows of the matrix are pre-shock values used to achieve simulation stability before applying a shock; these rows will not be shown in the plot. We encourage users to ensure that their simulations are stable before applying a shock. Although more complex, the advantage of Method 2 is the ability to program *any* shock scenario, with more than one regressor allowed to change at any period during the simulation. This is useful for models with dummy variables,

interaction terms, and/or complicated impulses.

`gbplot` also supports three forms of forecasting that incorporate uncertainty besides parametric uncertainty. The `fcast_epu` option incorporates "expected path uncertainty" (EPU) inherent in each bootstrap b by using b's own $m-1$ conditional variance at each time point. For instance, in Equation 12, instead of taking expectations we use $\tilde{\sigma}^2_{b,2}$ when generating $\tilde{\sigma}^2_{b,3}$ to incorporate EPU:

$$\tilde{\sigma}^2_{b,3} = \tilde{\omega}_{1,b} \cdot 0 + \tilde{\alpha}_{1,b}\tilde{\sigma}^2_{b,2} + \exp(\tilde{\omega}_{0,b} + \bar{\mathbf{z}}\tilde{\boldsymbol{\gamma}}_b) \tag{13}$$

Option `fcast_arch` incorporates EPU *plus* generates an innovation $\nu_m \sim N(0,1)$, which, when squared and multiplied by the lagged conditional error variance, becomes $\tilde{\varepsilon}^2_{m-1}$, the ARCH(1) effect entering in each period; again using Equation 12 as an illustration, we now simulate:

$$\tilde{\sigma}^2_{b,3} = \tilde{\omega}_{1,b} \cdot \tilde{\varepsilon}^2_{b,2} + \tilde{\alpha}_{1,b}\tilde{\sigma}^2_{b,2} + \exp(\tilde{\omega}_{0,b} + \bar{\mathbf{z}}\tilde{\boldsymbol{\gamma}}_b) \tag{14}$$

Finally, option `fcast_arch_resid` is the same as `fcast_arch`, but instead of using $N(0,1)$ innovations uses saved $\tilde{\nu}_m$ innovations from the residual bootstrap and saved in the dataset using the `fcast_arch` option when first running `gbmake`. Thus, `fcast_arch_resid` only works for the residual bootstrap, while `fcast_epu` and `fcast_arch` can be used for the parametric bootstrap, too. For all three approaches, `gbplot` plots the median, along with percentile confidence intervals.

5 Example

We illustrate the power of `gbmake` and `gbplot` by reproducing inferences from a paper that lies at the intersection of financial and energy economics. Hammoudeh and Yuan (2008) examine whether and how oil-price shocks and short-term interest rate adjustments shape the volatility of three commodities: gold, silver, and copper.

Table 1: Oil Price Shocks, Interest Rates, and Gold Price Volatility, from Hammoudeh and Yuan (2008)

	Coefficient	Standard Error
Mean Equation		
Logged Gold Return$_{t-1}$	-0.0160	(0.0122)
Absolute Value of Oil Price Change$_t$	-0.0001	(0.0002)
Log Change in Interest Rate$_{t-1}$	-0.0015*	(0.0009)
Iraq War (2003)	0.0008***	(0.0003)
COVID Pandemic	0.0003	(0.0005)
Iran War (2026)	-0.0047	(0.0049)
Constant	-0.0000	(0.0001)
Variance Equation		
ARCH(1)	0.0365***	(0.0103)
GARCH(1)	0.9601***	(0.0127)
Absolute Value of Oil Price Change$_t$	0.4329**	(0.1994)
Log Change in Interest Rate$_{t-1}$	-1.9598***	(0.3450)
Iraq War (2003)	0.2562	(0.6528)
COVID Pandemic	-0.7974	(2.6708)
Iran War (2026)	1.0046	(1.9591)
Constant	-15.0139***	(0.7871)
N	8590	
Log Lik.	27789.16	

Note: Dependent variable is logged gold returns. ARCH-GARCH model with robust standard errors in parentheses, reproducing Hammoudeh and Yuan (2008), Table 3, Model 1 on an extended sample through March 2026. $T = 8590$. Two-tailed tests. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$. Significance levels and digits reflect Hammoudeh and Yuan's original presentation, standardized to four significant digits.

Table 1 reproduces Hammoudeh and Yuan's gold volatility analysis using a related but not identical specification: a GARCH(1,1) model and an extended sample from January 1990 through March 2026. The absolute change in oil futures prices measures oil-price shocks, while the lagged log change in the three-month US Treasury bill rate measures interest rate shocks. Dummies for the 2003–2011 Iraq War and subsequent intervention, the 2020–2023 COVID-19 pandemic, and the 2026 Iran war capture their effects. The negative and significant interest-rate coefficient in our variance equation is consistent with the calming effect reported by Hammoudeh and Yuan, whereas our oil-shock coefficient is positive and significant, in contrast to their finding. Although Hammoudeh and Yuan report volatility half-life, they do not examine the magnitude of individual variance-regressor effects, how those effects propagate over time, or how the effect of one variance regressor depends on another.

We first estimate the ARCH-GARCH model in Table 1 using Stata's `arch` command and then immediately run `gbmake` to produce the bootstrap. For simplicity, we use the default `parametric` bootstrap:

```
arch dlngold ldlngold absdoil ldlninterest iraq covid iran, ///
    robust nolog arch(1) garch(1) het(absdoil ldlninterest iraq covid iran)

gbmake, reps(1000) saving("resources-param-bootstrap") replace
```

Users preferring the residual bootstrap would instead specify `btype(residual)`. Users preferring the delta method would skip `gbmake` and add the `delta` option to `gbplot` immediately after estimating the `arch` model. The OSM compares all three approaches using a different research example.

After generating the parametric bootstrap using `gbmake`, we then deploy `gbplot`. Because the `arch` model needs to be bootstrapped only once, every `gbplot` call below reuses the same file, `resources-param-bootstrap.dta`. Following Method 1, we first set baseline values for each regressor in the variance equation via `customvalues`: the absolute value of the change in oil prices at its mean, and the lagged log change in interest rates, the Iraq war, the COVID pandemic, and the Iran war each at 0. The program sets all variables to their estimation-sample mean values by default; `customvalues` can be used to set baselines to preferred or more sensible values.

We then apply a shock to one regressor via `shockvariable()` and `shockvalue()`. The program makes all shocks temporary by default; the `permanent` option can be used to make the shock persist for all subsequent periods. (We illustrate a permanent shock in the OSM.) For any shock, a standard starting point could be a mean value combined with a one-standard-deviation increase, which is conventional in the counterfactual simulation literature (e.g., Jordan and Philips 2018). But users should ensure that the chosen shock value reflects changes observed (or at least plausible) in the empirical data. Here, we choose a $+2$-unit shock to the absolute value of the change in oil prices.

Because the estimated GARCH coefficient is close to 1, any simulation will need a lengthy burn-in to reach a stable pre-shock value and a long range to observe the

shock dissipate. Figure 1a illustrates the cost of getting this wrong; with `burnin(10) range(10)`, the series is still burning in, and the shock accelerates a trend already underway.

```
gbplot, shockvar(absdoil) shockval(2) ///
    customvalues(.8629127 0 0 0 0) using("resources-param-bootstrap.dta") ///
    psv burnin(10) range(10) saving("fig1a") replace noplot
```

Raising the burn-in to 150 periods (`burnin(150)`), as shown in Figure 1b, alleviates most of the issue, but a 10-period (10-day) plotting range remains too short for the shock to dissipate. Combining `burnin(150) range(100)` produces a stable plot where variance is steady pre-shock and returns to that level afterwards, as shown in Figure 1c. This plot traces the entire temporal path of the shock as it decays at a rate governed by the persistence parameters of the ARCH-GARCH terms. Users investigating a temporary shock should expect this pattern and treat any deviation as a sign to revisit the burn-in and/or range. We also recommend that users preferring Method 2 still first explore the burn-in and range using Method 1 before building any shock matrix required for Method 2.

Once the burn-in and range are sufficient, we can generate simulations across multiple scenarios. We recommend that users vary shock magnitudes (both positive and negative), baseline values of the regressor(s) of interest, and configurations of control variables. Following Method 1, we can investigate whether interest-rate shocks have asymmetric effects on gold price volatility, and whether those effects depend on macroeconomic conditions. For example, in Figure 2 we compare the effect of a $+1$-unit versus a -1-unit shock to the lagged log change in the three-month Treasury bill rate, setting the absolute value of the change in oil prices to its mean and varying whether the shock occurs during the COVID-19 pandemic (right column) or outside it (left column).

```
gbplot, shockvar(ldlninterest) shockval(1) ///
    customvalues(.8629127 0 0 0 0) using("resources-param-bootstrap.dta") ///
    psv burnin(150) range(100) saving("fig2a") replace noplot
gbplot, shockvar(ldlninterest) shockval(1) ///
```

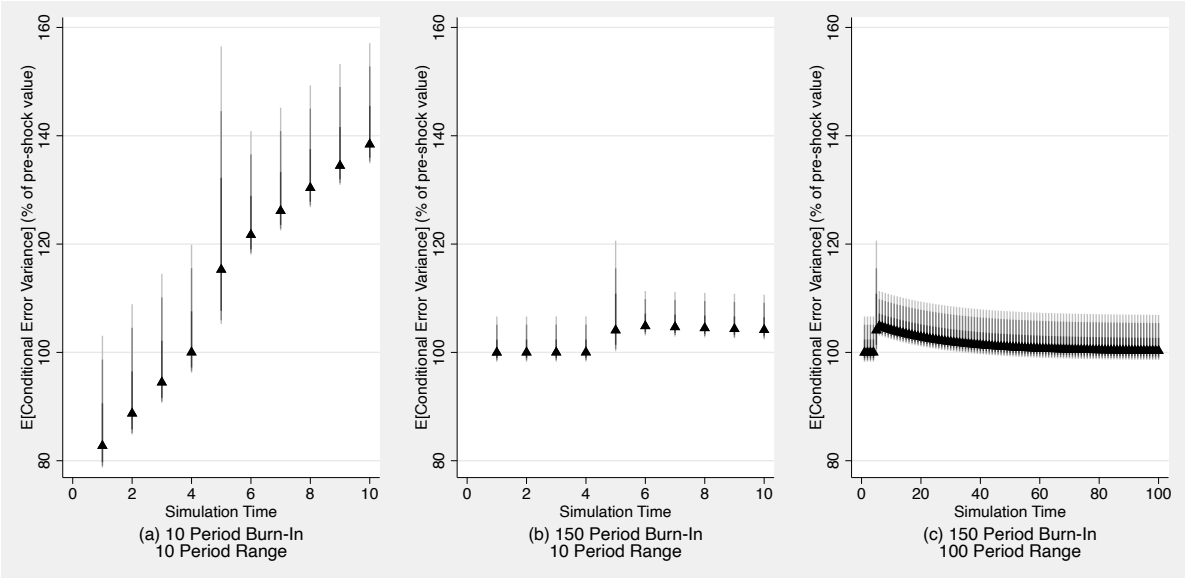


Figure 1: The +2-unit shock to the absolute value of the change in oil prices under different burn-ins and ranges

NOTE: 10 period burn-in, 10 period range (plot a), 150 period burn-in, 10 period range (b) and 150 period burn-in and 100 period range (c). For each +2-unit shock, absolute value of change in oil prices is set to its mean and all other variables are set to 0 (no change in interest rates, no COVID, and no conflict). Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

```
    customvalues(.8629127 0 0 1 0) using("resources-param-bootstrap.dta") ///
    psv burnin(150) range(100) saving("fig2b") replace noplot

gbplot, shockvar(ldlninterest) shockval(-1) ///
    customvalues(.8629127 0 0 0 0) using("resources-param-bootstrap.dta") ///
    psv burnin(150) range(100) saving("fig2c") replace noplot
gbplot, shockvar(ldlninterest) shockval(-1) ///
    customvalues(.8629127 0 0 1 0) using("resources-param-bootstrap.dta") ///
    psv burnin(150) range(100) saving("fig2d") replace noplot
```

The `saving()` option saves the plotting data (here, `fig2a`, `fig2b`, `fig2c`, `fig2d`) for customization using another Stata graphing tool.

Figure 2's panels report the response as a percentage of the baseline pre-shock variance via the `psv` option. Because variance units have no natural scale, this metric eases interpretation; without `psv`, `gbplot` reports the expected variance in its natural units (illustrated in the OSM). Figure 2 reveals key asymmetries: a positive rate shock produces a small reduction in gold volatility (Figure 2a and Figure 2b), while a negative rate shock produces a substantial increase (Figure 2c and Figure 2d). Outside the

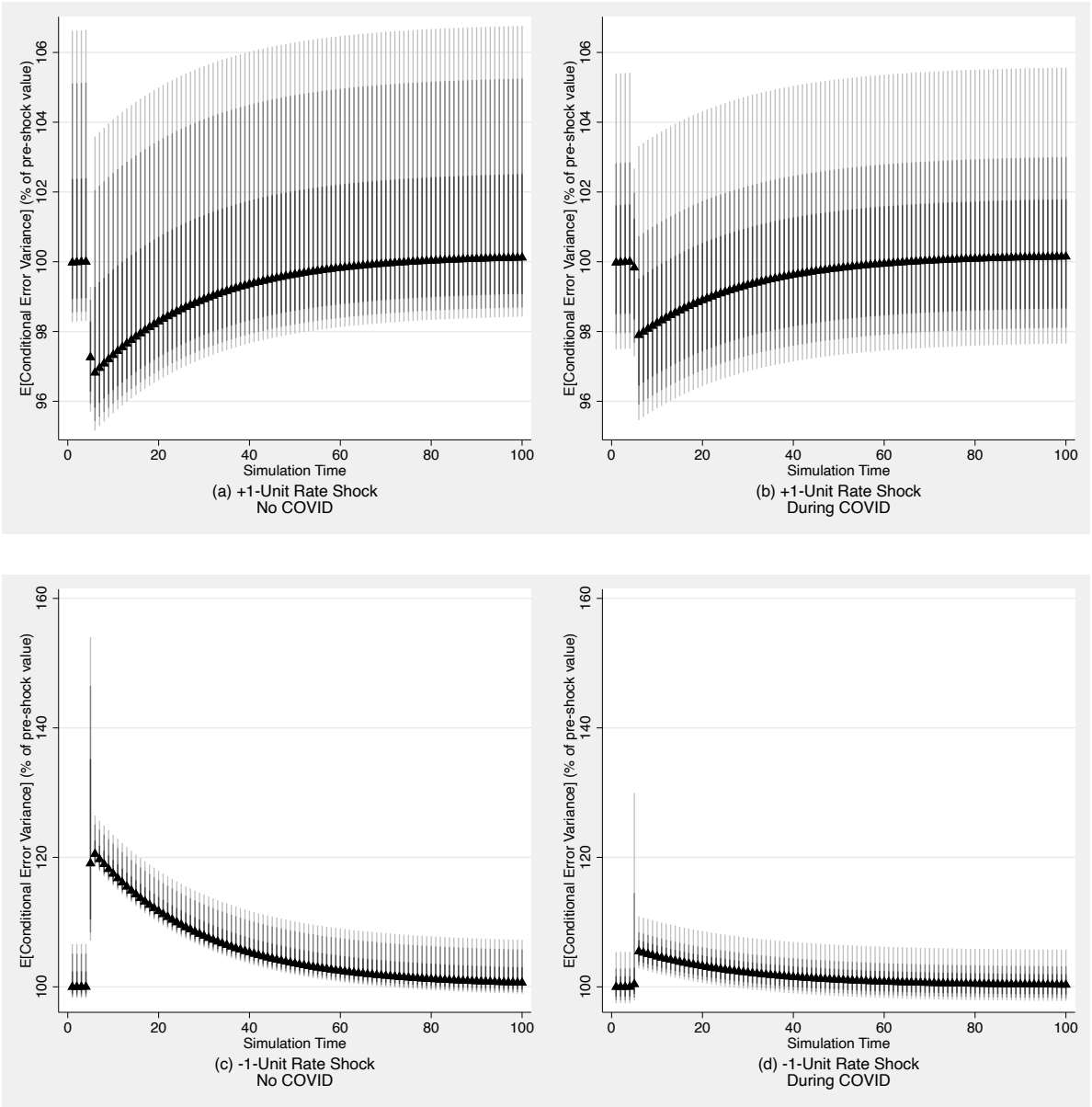


Figure 2: Positive versus negative shocks to the log change in interest rates have asymmetric effects on gold price volatility, especially during different global events

NOTE: For each shock, absolute value of the change in oil prices is set to its mean and all other variables are set to 0 (no change in interest rates, no COVID, and no conflict), except that COVID is set to 1 in panels (b) and (d). Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

pandemic, the negative-shock effect is pronounced, but during COVID it is markedly dampened. This pattern echoes the traditional advice that maximum likelihood models are inherently interactive: baseline values of other variance regressors can meaningfully shift the effect of any single parameter, so `customvalues()` should be chosen with care (Berry et al. 2010).

Method 1 shocks a single regressor. To investigate more complex shocks, we can deploy Method 2's `usematrix()` option to set the value of every regressor in every period. For example, we might want to examine the volatility consequences of a sustained but bounded conflict, such as a 50-day skirmish in either Iraq or Iran. Since the GARCH coefficient is close to 1, we know from Figure 1 that we need a substantial burn-in and range. We build an $m \times k$ simulation matrix of $M = 300$ periods across our $k = 5$ regressors (from `arch`'s `het(absdoil ldlninterest iraq covid iran)`), with a 150-period burn-in, the conflict shock active during rows 151–200 (the 50-day skirmish), and 100 post-shock periods for variance to recover.

Our two simulation matrices — `fiftydayiraq` and `fiftydayiran` — would thus be:

```
* Iraq skirmish: column 3 on for rows 151-200
matrix fiftydayiraq = J(300, 5, 0)
forvalues i = 1/300 {
    matrix fiftydayiraq[`i',1] = .8629127
}
forvalues i = 151/200 {
    matrix fiftydayiraq[`i',3] = 1
}

* Iran skirmish: column 5 on for rows 151-200
matrix fiftydayiran = J(300, 5, 0)
forvalues i = 1/300 {
    matrix fiftydayiran[`i',1] = .8629127
}
forvalues i = 151/200 {
    matrix fiftydayiran[`i',5] = 1
}
```

In each, the first column corresponds to `absdoil` and the next four columns correspond to `ldlninterest`, `iraq`, `covid`, and `iran`. `J(300, 5, 0)` initializes all entries to zero, after which we set the first column to the sample mean of `absdoil` (0.8629127) in every

period. To encode the Iraq skirmish, we set rows 151–200 of the third column (`iraq`) to 1, leaving 150 burn-in rows of pure baseline before the shock and 100 post-shock rows for variance to recover. The Iran matrix is identical except that column 5 (`iran`) is the one switched on. We pass the matrix to `gbplot` with `usematrix(fiftydayiraq)` (or `fiftydayiran`). We also use `burnin(140)` to ensure that only the final ten pre-shock rows are displayed, and use `psv_usematrix(150)` to set the `psv` percentage baseline at row 150, the last row before the shock.

Using these matrices, we then deploy `gbplot`:

```
gbplot, using("resources-param-bootstrap.dta") usematrix(fiftydayiraq) ///
    burnin(140) psv_usematrix(150) saving("fig3a") replace noplot

gbplot, using("resources-param-bootstrap.dta") usematrix(fiftydayiran) ///
    burnin(140) psv_usematrix(150) saving("fig3b") replace noplot
```

Figure 3 presents the results. In each case, gold price variance builds continually as the conflict persists, though the rate of increase declines over time. As soon as the conflict ends, variance recedes, taking almost 100 post-shock periods (days) to return to its pre-shock value. An Iran skirmish also produces much greater volatility than an Iraq skirmish, likely reflecting underlying differences between the pre- and post-2008 periods: the Iraq dummy is identified partly on pre-2008 data, while the Iran dummy sits entirely in the post-2008 volatility regime.

We emphasize this is just a slice of the options available to users who want to interpret their ARCH-GARCH models. To further explore interpretation, we provide a second illustration using an example from political economy; for space considerations, this can be found in the OSM.

6 Conclusions

We are not the first to examine the benefits of simulation for improving inferences from (conventional) time series models. However, we join Benton and Philips (2020)

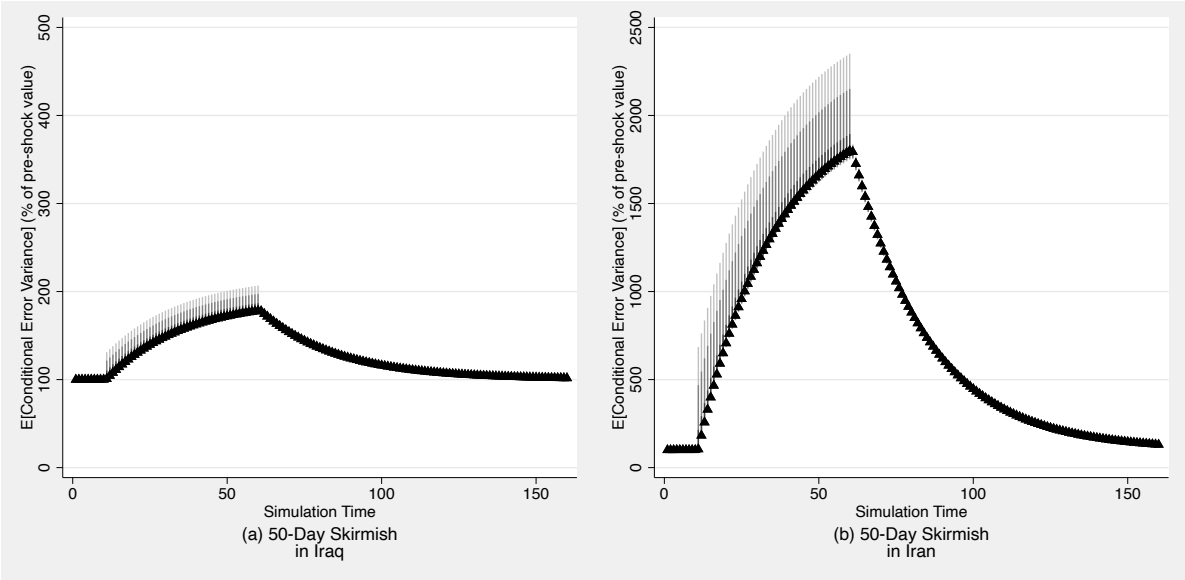


Figure 3: `usematrix()` can be used to investigate shocks of user-chosen designs

NOTE: For each shock, absolute value of the change in oil prices is set to its mean and all other variables are set to 0 (no change in interest rates, no COVID, and no conflict) outside the 50-day conflict period. Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

and Benton et al. (2026) in being among the first to apply this technique to theories about variance. Our commands currently support the standard ARCH-GARCH specification estimated by `arch`. Future work might extend this strategy to additional types of ARCH-GARCH models (e.g., Nelson 1991; Zakoian 1994) and to more complex lag structures.

7 Acknowledgments

We thank the editor and an anonymous referee for comments that improved both the article and the commands.

8 Programs and supplemental materials

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

. net sj 26-2

. net install (to install program files, if available)

. net get (to install ancillary files, if available)

(Reference to OSM here.)

References

Arel-Bundock, V., N. Greifer, and A. Heiss. 2024. How to Interpret Statistical Models Using marginaleffects for R and Python. *Journal of Statistical Software* 111: 1–32. URL https://doi.org/10.18637/jss.v111.i09.

Benton, A. L., S. Jordan, and A. Q. Philips. 2026. Taking Variance Seriously: Visualizing the Statistical and Substantive Significance of ARCH-GARCH Models. *The Journal of Politics* 88: 18–35. URL https://doi.org/10.1086/734275.

Benton, A. L., and A. Q. Philips. 2020. Does the @realDonaldTrump Really Matter to Financial Markets? *American Journal of Political Science* 64: 169–190. URL https://doi.org/10.1111/ajps.12491.

Berry, W. D., J. H. R. DeMeritt, and J. Esarey. 2010. Testing for Interaction in Binary Logit and Probit Models: Is a Product Term Essential? *American Journal of Political Science* 54: 248–266. URL https://doi.org/10.1111/j.1540-5907.2009.00429.x.

Bollerslev, T. 1986. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* 31: 307–327. URL https://doi.org/10.1016/0304-4076(86)90063-1.

Engle, R. F. 1982. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica* 50: 987–1007. URL https://doi.org/10.2307/1912773.

Flink, C. M. 2017. Rethinking punctuated equilibrium theory: A public administration

approach to budgetary changes. *Policy Studies Journal* 45(1): 101–120. URL https://doi.org/10.1111/psj.12114.

Hammoudeh, S., and Y. Yuan. 2008. Metal volatility in presence of oil and interest rate shocks. *Energy Economics* 30: 606–620. URL https://doi.org/10.1016/j.eneco.2007.09.004.

Hellwig, T. 2007. Economic openness, policy uncertainty, and the dynamics of government support. *Electoral Studies* 26: 772–786. URL https://doi.org/10.1016/j.electstud.2007.06.005.

Jordan, S., and A. Q. Philips. 2018. Cointegration testing and dynamic simulations of autoregressive distributed lag models. *The Stata Journal* 18: 902–923. URL https://doi.org/10.1177/1536867X1801800409.

King, G., M. Tomz, and J. Wittenberg. 2000. Making the Most of Statistical Analyses: Improving Interpretation and Presentation. *American Journal of Political Science* 44: 347–361. URL https://doi.org/10.2307/2669316.

Kropko, J., and J. J. Harden. 2020. Beyond the Hazard Ratio: Generating Expected Durations from the Cox Proportional Hazards Model. *British Journal of Political Science* 50: 303–320. URL https://doi.org/10.1017/S000712341700045X.

Kumar, S., G. Singh, and A. Kumar. 2022. Volatility spillover among prices of crude oil, natural gas, exchange rate, gold, and stock market: Fresh evidence from exponential generalized autoregressive conditional heteroscedastic model analysis. *Journal of Public Affairs* 22(4): 1–12. URL https://doi.org/10.1002/pa.2594.

Léber, D., and B. Egyed. 2025. The Sentiment Augmented GARCH-LSTM Hybrid Model for Value-at-Risk Forecasting. *Computational Economics* 67: 313–353. URL https://doi.org/10.1007/s10614-025-11042-8.

Naimy, V., O. Haddad, G. Fernández-Avilés, and R. El Khoury. 2021. The predictive capacity of GARCH-type models in measuring the volatility of crypto and world cur-

rencies. *PLOS ONE* 16: e0245904. URL https://doi.org/10.1371/journal.pone.0245904.

Nelson, D. B. 1991. Conditional heteroskedasticity in asset returns: A new approach. *Econometrica* 59: 347–370. URL https://doi.org/10.2307/2938260.

Philips, A. Q., A. Rutherford, and G. D. Whitten. 2016. Dynamic pie: A strategy for modeling trade-offs in compositional variables over time. *American Journal of Political Science* 60: 268–283. URL https://doi.org/10.1111/ajps.12204.

Sadorsky, P. 1999. Oil price shocks and stock market activity. *Energy Economics* 21(5): 449–469. URL https://doi.org/10.1016/S0140-9883(99)00020-1.

Vinod, H. D., and J. Lopez-de Lacalle. 2009. Maximum Entropy Bootstrap for Time Series: The meboot R Package. *Journal of Statistical Software* 29: 1–19. URL https://doi.org/10.18637/jss.v029.i05.

Williams, L. K., and G. D. Whitten. 2012. But wait, there's more! Maximizing substantive inferences from TSCS models. *The Journal of Politics* 74: 685–693. URL https://doi.org/10.1017/S0022381612000473.

Yao, K.-C., H.-W. Hsueh, M.-H. Huang, and T.-C. Wu. 2022. The Role of GARCH Effect on the Prediction of Air Pollution. *Sustainability* 14(8): 4459. URL https://doi.org/10.3390/su14084459.

Zakoian, J.-M. 1994. Threshold heteroskedastic models. *Journal of Economic Dynamics and Control* 18: 931–955. URL https://doi.org/10.1016/0165-1889(94)90039-6.

About the authors

Allyson L. Benton is a Reader in the Department of Government at the University of Essex.

Soren Jordan is an Associate Professor in the Department of Political Science at Texas A&M University.

Andrew Q. Philips is the Henry W. Ehrmann Professor in Law and Jurisprudence in the Department of Political Science at the University of Colorado Boulder.

Online Supplementary Material: Revealing Volatility Dynamics in ARCH-GARCH Models

Allyson L. Benton
Department of Government
University of Essex
Colchester, UK
allyson.benton@essex.ac.uk

Soren Jordan
Department of Political Science
Texas A&M University
College Station, Texas, USA
sorenjordanpols@gmail.com

Andrew Q. Philips
Department of Political Science
University of Colorado Boulder
Boulder, Colorado, USA
andrew.philips@colorado.edu

Example 2

We further illustrate the power of `gbmake` and `gbplot` by reproducing inferences from a paper at the nexus of public opinion and political economy. Hellwig (2007) argues that the greater a country's exposure to the global economy, the greater will be the volatility in public approval ratings of its government.

Table OSM.1 reproduces Hellwig's ARCH-GARCH Model 1 from his Table 3 examining UK trade openness and government approval. The positive and significant trade openness coefficient in the variance equation, along with the positive and significant ARCH and GARCH terms, leads Hellwig to conclude that rising trade openness raises volatility in UK government approval. He also notes that the positive and significant coefficients for the Harold Wilson/James Callaghan and Margaret Thatcher governments indicate that approval was more volatile during those premierships than under Tony Blair or John Major (which is the omitted category). Hellwig's interpretive approach is typical of the broader literature which relies on tabular coefficients.

Table OSM.1: Trade exposure and UK government support, from Hellwig (2007)

	Coefficient	Standard Error
Mean Equation		
Unemployment$_t$	0.351	(0.278)
Inflation$_t$	0.118	(0.134)
Prospective Evaluations$_t$	0.056***	(0.017)
Political Events	2.167***	(0.371)
Trade$_t$	0.058	(0.040)
Wilson/Callaghan	1.648	(1.334)
Thatcher	0.526	(0.670)
Blair	-0.752	(0.640)
Constant	-1.794	(1.567)
Variance Equation		
ARCH(1)	0.263**	(0.119)
GARCH(1)	0.333**	(0.143)
Trade$_t$	0.058*	(0.034)
Wilson/Callaghan	2.661***	(0.992)
Thatcher	1.634**	(0.697)
Blair	-0.219	(0.690)
Constant	-1.765	(1.388)
N	303	
Log Lik.	-698.854	

NOTE: The dependent variable is governing party support in the UK. ARCH-GARCH model with robust standard errors in parentheses, replicating Hellwig (2007), Table 3, Model 1. $T = 303$. Control variables included in the mean equation but not shown. Two-tailed tests. * $p < 0.10$, ** $p < 0.05$, *** $p < 0.01$. These results are almost identical to Hellwig's original results, with slight differences due to operating systems. Significance levels and digits reflect Hellwig's original presentation.

We first estimate the ARCH-GARCH model in Table OSM.1 using Stata's `arch` command. With the ARCH-GARCH model in memory, we then deploy `gbmake` to generate the bootstrap. For simplicity, we use the `parametric` bootstrap.

```
arch fdsupp84 fdunem33 fdinfl21 fdpexp17 polevent falkland el79 ///
    el83 el87 el92 el97 trade wilscall thatcher blair, ///
    robust nolog arch(1) garch(1) het(trade wilscall thatcher blair)

gbmake, btype(parametric) saving("hellwig-param-bootstrap") ///
    reps(1000) replace
```

Using Method 1, we can investigate whether trade shocks have asymmetric effects on volatility in government approval. We compare the effect of a positive versus a negative one-period, 13-point shock (equivalent to one standard deviation) to trade openness, setting all other regressors to their means and the prime minister to Margaret Thatcher (by setting all prime minister variables except `thatcher` to 0). We construct this comparison by applying `gbplot` to the parametric bootstrap results (noting that the mean of `trade` is ∼28.6 and the third variable in the `het` equation is `thatcher`). The changing arguments are the `shockval` (+13 versus -13) and `psv`:

```
gbplot, using("hellwig-param-bootstrap.dta") shockvar(trade) ///
    shockval(13) customvalues(28.64679870166277 0 1 0) ///
    saving("p1a") replace noplot
gbplot, using("hellwig-param-bootstrap.dta") shockvar(trade) ///
    shockval(-13) customvalues(28.64679870166277 0 1 0) ///
    saving("p1b") replace noplot
gbplot, using("hellwig-param-bootstrap.dta") shockvar(trade) ///
    shockval(13) psv customvalues(28.64679870166277 0 1 0) ///
    saving("p1c") replace noplot
gbplot, using("hellwig-param-bootstrap.dta") shockvar(trade) ///
```

```
    shockval(-13) psv customvalues(28.64679870166277 0 1 0) ///
    saving("p1d") replace noplot
```

The `saving` option saves the plotting data (`p1a`, `p1b`, `p1c`, and `p1d`) for customization when plotting.

Figure OSM.1 shows that a positive 13-point shock to trade from its mean raises the variance of government approval by 5.2 natural variance units (under Thatcher), from a baseline of 7.3 to 12.5, while a negative 13-point shock lowers it by 2.4 units, to 4.9. These variance units have no natural scale, so we can gain more substantive insight by conceptualizing them relative to the pre-shock baseline. Using the `psv` option, we see the positive shock raises volatility to 170 percent of its pre-shock level, and the negative shock lowers it to 67 percent. We expect most users will take advantage of `psv` for ease of interpretation.

We can also visualize differences in the volatility in government approval under each prime minister, shown in Figure OSM.2. Reusing `hellwig-param-bootstrap.dta` (since the `arch` model needs to be bootstrapped only once), we run `gbplot` as follows:

```
gbplot, using("hellwig-param-bootstrap.dta") shockvar(wilscall) ///
    shockval(1) psv permanent customvalues(28.64679870166277 0 0 0) ///
    saving("p2a") replace noplot
gbplot, using("hellwig-param-bootstrap.dta") shockvar(thatcher) ///
    shockval(1) psv permanent customvalues(28.64679870166277 0 0 0) ///
    saving("p2b") replace noplot
gbplot, using("hellwig-param-bootstrap.dta") shockvar(blair) ///
    shockval(1) psv permanent customvalues(28.64679870166277 0 0 0) ///
    saving("p2c") replace noplot
```

Since prime ministers typically hold office for more than a single quarter, we use `permanent` to examine the effect of a "permanent" shock to government, which results from prime minister turnover. We use `customvalues` to hold trade at its mean and

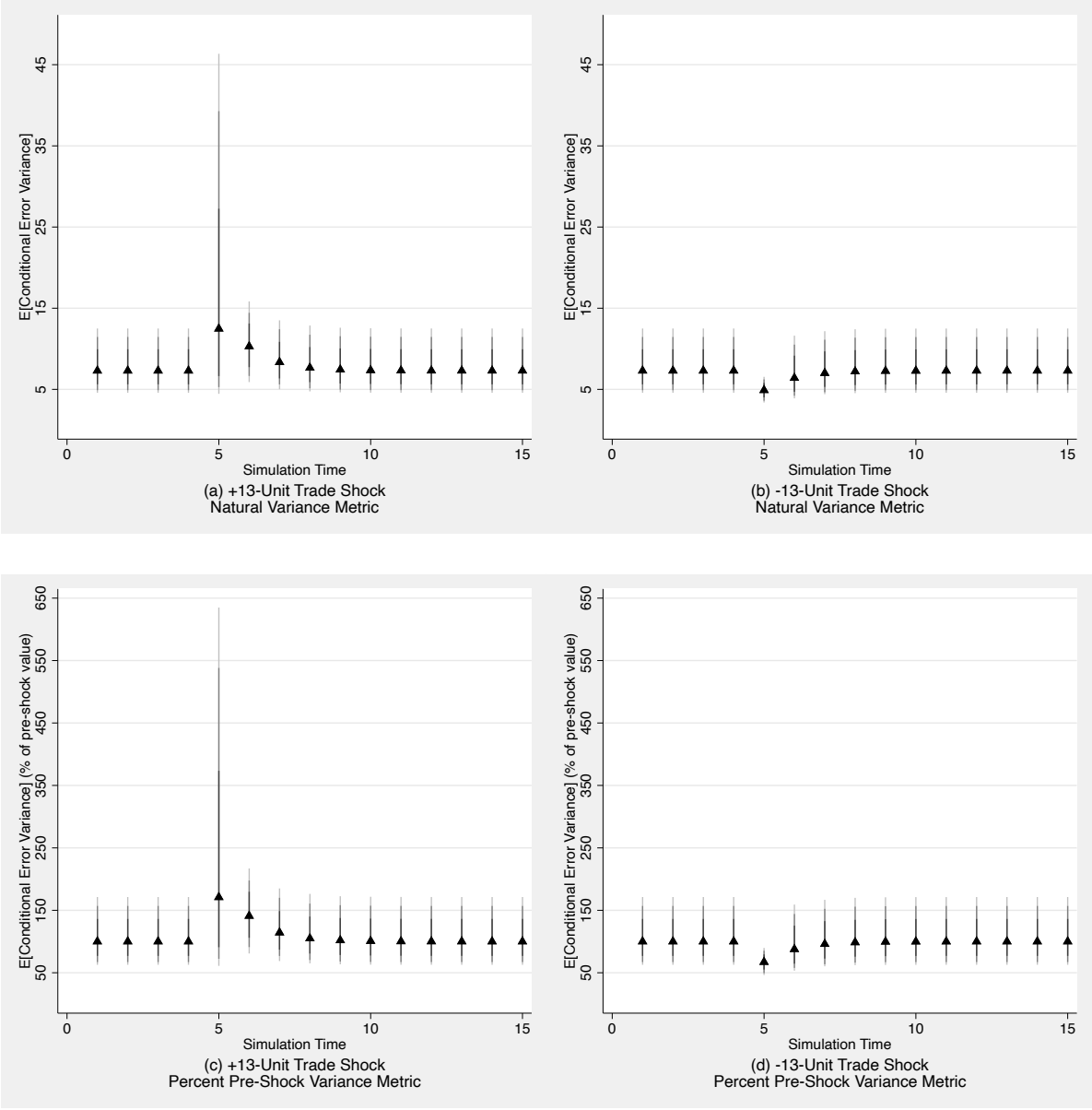


Figure OSM.1: Positive versus negative shocks to trade have asymmetric effects on the expected variance of government support

NOTE: For each shock, trade is set to its mean and prime minister is set to Thatcher. Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

each prime minister dummy variable at 0. When using `customvalues`, users should choose sensible baselines. When `customvalues` is not used, the program sets all variables to their mean values. Figure OSM.2 shows that the Wilson/Callaghan governments faced considerably greater volatility in their approval ratings compared to the (baseline, omitted) John Major premiership, while volatility was comparatively lower under Margaret Thatcher and minimal under Tony Blair.

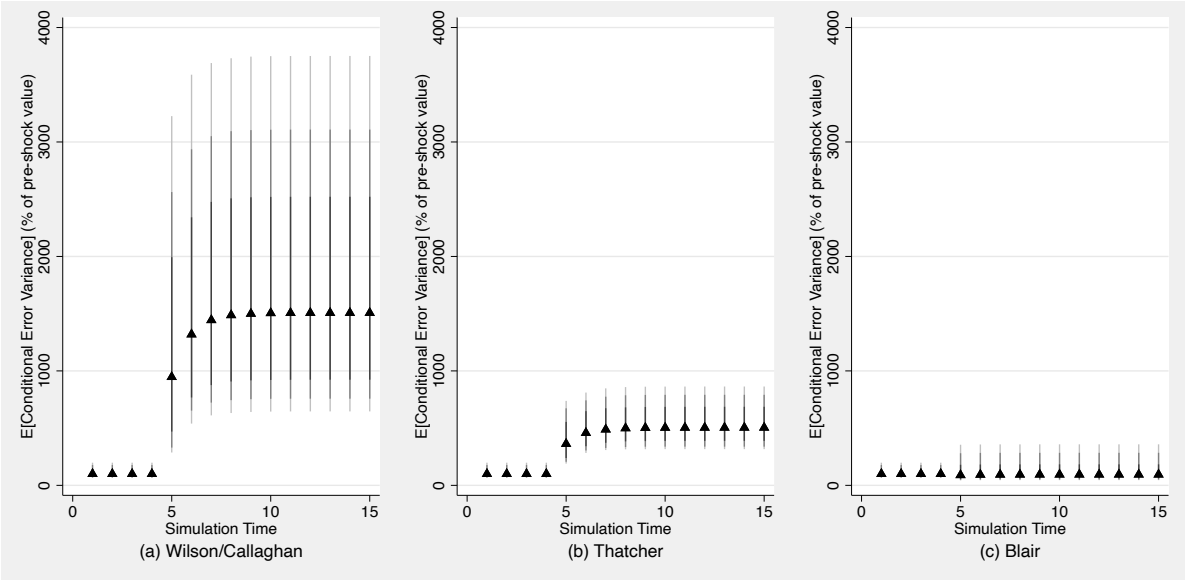


Figure OSM.2: The election of different prime ministers leads to different effects on the expected variance of government support.

NOTE: For each permanent shock, trade is set to its mean and the baseline is the John Major premiership. Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

Using Method 2, we can also investigate more complex trade shocks with the `usematrix` option. Users choosing this approach should first build an $m \times k$ simulation matrix, where the m rows are user-chosen periods and the k columns are the regressors in the `arch` variance equation. To illustrate, we build a matrix of $M = 12$ simulation time periods across our $k = 4$ regressors (from our `arch` model's `het(trade wilscall thatcher blair)`). We begin the shock at $m = 3$ (after a two-period burn-in), including a linear decay in $m = 4$ and $m = 5$, and observing its effect for the remaining seven periods (for a total of $M = 12$).

```
matrix tradematnoburn = J(12, 4, 0)
```

```
* Fill all rows with baseline values (cols 2 and 4 already 0 from J)
forvalues i = 1/12 {
    matrix tradematnoburn[`i',1] = 28.64679870166277
    matrix tradematnoburn[`i',3] = 1
}

* Modify the shock rows
matrix tradematnoburn[3,1] = 31.64679870166277
matrix tradematnoburn[4,1] = 30.64679870166277
matrix tradematnoburn[5,1] = 29.64679870166277
```

The first column in `tradematnoburn` represents the chosen values for `trade` and the next three columns represent dummy variables for the three prime ministers. We encode the mean of `trade` in the first column (28.6) and indicate the `thatcher` government by placing 1 in the third column and 0s in columns 2 and 4. To encode the shock, we replace row $m = 3$ with the $+3$ value, row $m = 4$ with the $+2$ value, and row $m = 5$ with the $+1$ value.

We next run `gbplot` using the matrix `tradematnoburn`:

```
gbplot, using("hellwig-param-bootstrap.dta") usematrix(tradematnoburn) ///
    saving("p3a") replace noplot
```

We use the saved data (`p3a`) to produce Figure OSM.3a, revealing that our user-chosen two-period burn-in is insufficient for equilibrating the series before simulating the impact of the shock. We recommend that users interested in using `usematrix` first investigate how long the series takes to equilibrate using Method 1 (as in Figure OSM.1) before using `usematrix`. Using Method 1, we find that a 20 period burn-in is needed to equilibrate the series. We thus adjust `tradematnoburn` to `tradematburn`:

```
matrix tradematburn = J(30, 4, 0)
```

```
* Fill all rows with baseline values (cols 2 and 4 are already 0 from J)
forvalues i = 1/30 {
    matrix tradematburn[`i',1] = 28.64679870166277
    matrix tradematburn[`i',3] = 1
}
* Modify the shock rows
matrix tradematburn[21,1] = 31.64679870166277
matrix tradematburn[22,1] = 30.64679870166277
matrix tradematburn[23,1] = 29.64679870166277
```

and then run the following in `gbplot`:

```
gbplot, using("hellwig-param-bootstrap.dta") usematrix(tradematburn) ///
    psv_usematrix(20) saving("p3b") replace noplot
```

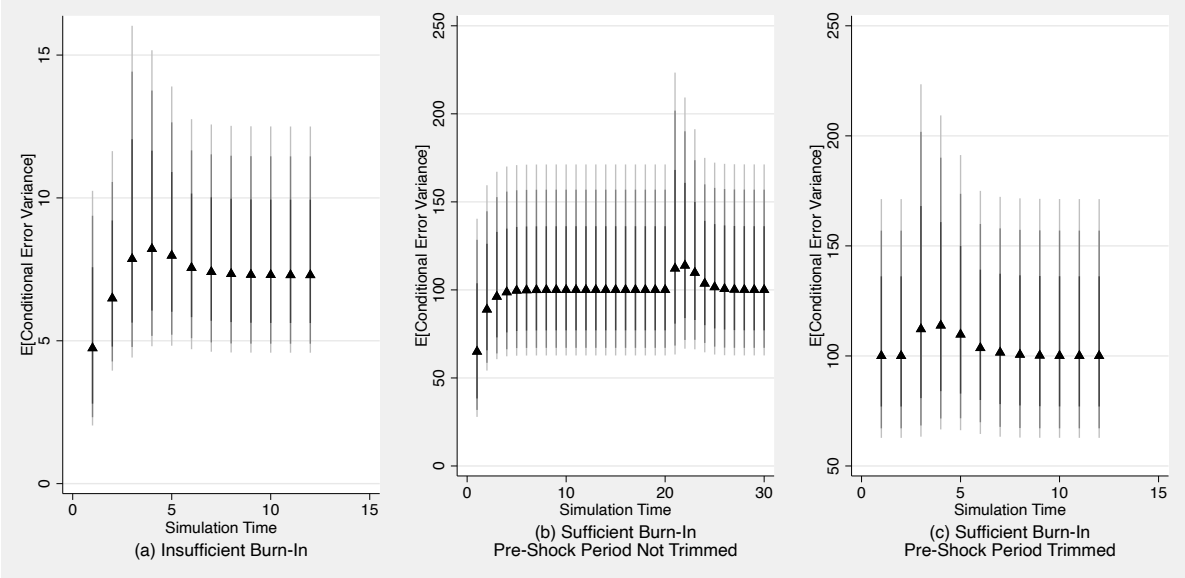


Figure OSM.3: A gradually receding shock (+3 to +2 to +1) to trade over three periods is easy to calculate

NOTE: Pre-shock trade is set to its mean, prime minister set to Thatcher. Panel (a) shows the danger of an insufficient (two-period) burn-in, with the baseline not yet settled before the shock at $t = 1$; (b) shows the full series with a 20-period burn-in; (c) trims to the period of interest while retaining the pre-shock value metric via `psv_usematrix`. Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

The simulation using `tradematburn` is shown in Figure OSM.3b, which is now sta-

ble pre-shock. We can eliminate uninformative pre-shock periods from this figure by using the `burnin` option, indicating `burnin(18) psv_usematrix(19)`, which tells `gbplot` to consider the first 18 periods of the simulation as the burn-in and sets the 19th period as the pre-shock value of the variance.

```
gbplot, using("hellwig-param-bootstrap.dta") usematrix(tradematburn) ///
    burnin(18) psv_usematrix(19) saving("p3c") replace noplot
```

This publication-ready plot is shown in Figure OSM.3c. Together, the `burnin` and `psv_usematrix` options let users conduct a sufficient burn-in while graphing only a few stable periods. They also set and visualize the pre-shock variance and show the shock and its recursive effect over time.

We can also compare the utility and consistency of the three approaches to uncertainty, using a +3-unit shock to trade under Thatcher. To do this, we run the following code:

```
* (a) parametric bootstrap
* [reuses hellwig-param-bootstrap.dta created earlier]
gbplot, using("hellwig-param-bootstrap.dta") shockvar(trade) ///
    shockval(3) psv customvalues(28.64679870166277 0 1 0) ///
    saving("p4a") replace noplot

* (b) residual bootstrap: re-estimate, build residual bootstrap, then plot
arch fdsupp84 fdunem33 fdinfl21 fdpexp17 polevent falkland el79 ///
    el83 el87 el92 el97 trade wilscall thatcher blair, ///
    robust nolog arch(1) garch(1) het(trade wilscall thatcher blair)
gbmake, btype(residual) saving("hellwig-resid-bootstrap") ///
    reps(100) replace
gbplot, using("hellwig-resid-bootstrap.dta") shockvar(trade) ///
    shockval(3) psv customvalues(28.64679870166277 0 1 0) ///
```

```
    saving("p4b") replace noplot

* (c) delta method: re-estimate so e(V) is in memory, then plot (no using())
arch fdsupp84 fdunem33 fdinfl21 fdpexp17 polevent falkland el79 ///
    el83 el87 el92 el97 trade wilscall thatcher blair, ///
    robust nolog arch(1) garch(1) het(trade wilscall thatcher blair)
gbplot, shockvar(trade) customvalues(28.64679870166277 0 1 0) ///
    shockval(3) delta psv saving("p4c") replace noplot
```

We use the plotting data (`p4a`, `p4b`, and `p4c`) to make Figure OSM.4.

As shown in Figure OSM.4, the response paths are nearly identical across the three approaches, showing a spike in volatility when the shock occurs. However, the confidence intervals are smallest for the residual bootstrap. The delta method's upper and lower 95% confidence bounds become wider during the shock period.[1] Although this was not the case in this example, as noted in the main text the analytic confidence intervals produced by the delta method can occasionally fall to "impossible" values below zero.

We conclude the Hellwig example by observing an example of the forecasting functionality. We first re-estimate the ARCH model and create residual bootstraps, adding the option `fcast_arch` to save the standardized innovations for use with `fcast_arch_resid`. We then run three different types of `gbplot`, using the same scenario as in OSM.4. Option `fcast_epu` incorporates "expected path uncertainty" by using each bootstrap's own $m-1$ conditional variance instead of taking expectations across all bootstraps when simulating. This is plotted in Figure OSM.5a. Figure OSM.5b shows option `fcast_arch`, which incorporates expected path uncertainty along with an innovation ν_m that is $\sim N(0,1)$, and when squared and multiplied by the lagged conditional error variance enters in as an ARCH(1) effect. Last, option `fcast_arch_resid`, shown in

[1]The delta method only produces one set of confidence intervals, set by `level`, as `nlcom` allows only a single call to a level of significance. Users wishing to emulate all three confidence intervals of the bootstrapping methods should re-run the delta method multiple times under different significance levels, save the dataset each time through `saving`, and combine.

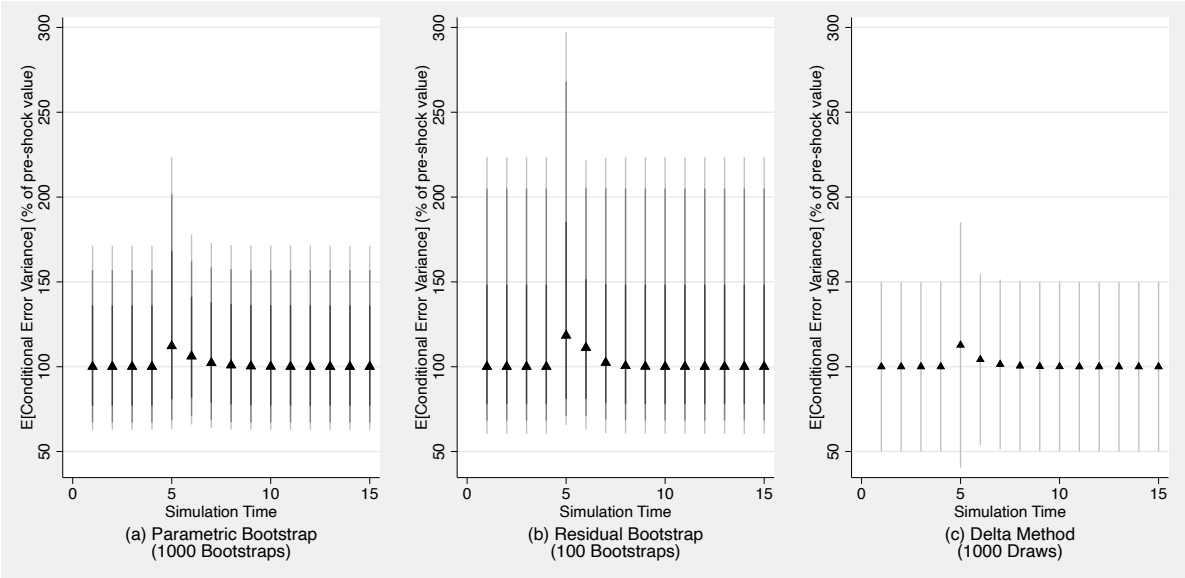


Figure OSM.4: The +3-unit shock to trade compared across different methods for estimating uncertainty

NOTE: For each shock, trade is set to its mean and prime minister is set to Thatcher. Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

Figure OSM.5c, uses saved innovations from the residual bootstrap to form ν_m instead of assuming a standard normal. As is clear from Figure OSM.5, while the median expected conditional error variance is largely the same as before, uncertainty is larger, especially for the two forecasting approaches that incorporate uncertainty through an ARCH(1) effect.

```
* re-estimate residual bootstrap with fcast_arch for use later
arch fdsupp84 fdunem33 fdinfl21 fdpexp17 polevent falkland el79 el83 el87 ///
    el92 el97 trade wilscall thatcher blair, ///
    robust nolog arch(1) garch(1) het(trade wilscall thatcher blair)

gbmake, btype(residual) saving("hellwig-resid-bootstrap") reps(250) ///
    fcast_arch replace

* (a) expected path uncertainty
gbplot, using("hellwig-resid-bootstrap.dta") shockvar(trade) shockval(3) psv ///
```

```
    customvalues(28.64679870166277 0 1 0) saving("p5a") replace noplot ///
    fcast_epu

* (b) EPU + N(0,1) innovation in ARCH
gbplot, using("hellwig-resid-bootstrap.dta") shockvar(trade) shockval(3) psv ///
    customvalues(28.64679870166277 0 1 0) saving("p5b") replace noplot ///
    fcast_arch

* (b) EPU + bootstrap innovation in ARCH
gbplot, using("hellwig-resid-bootstrap.dta") shockvar(trade) shockval(3) psv ///
    customvalues(28.64679870166277 0 1 0) saving("p5c") replace noplot ///
    fcast_arch_resid
```

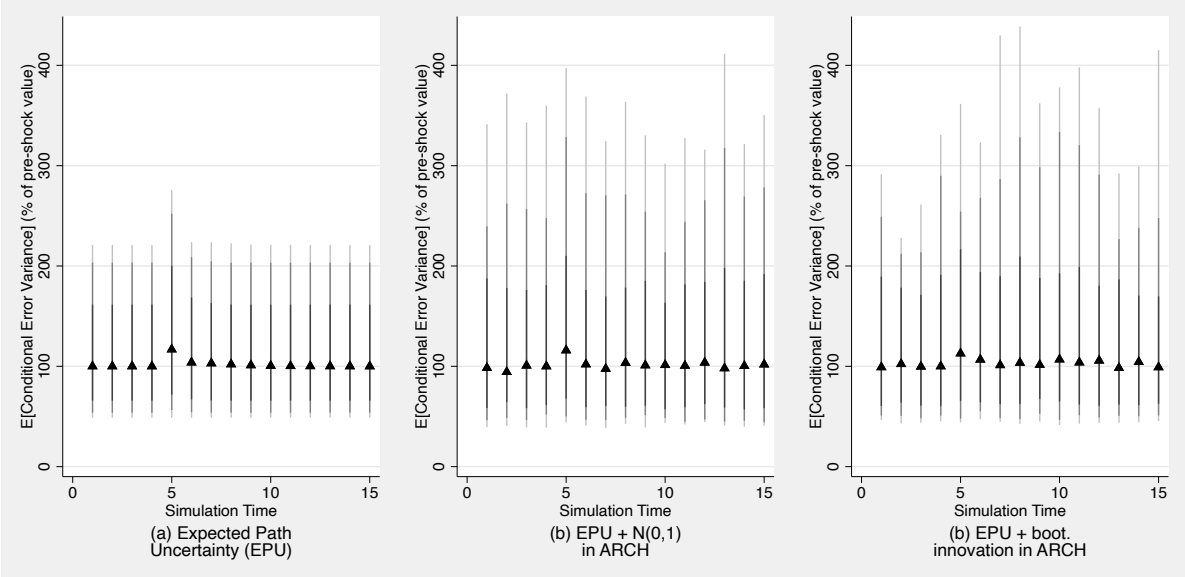


Figure OSM.5: The +3-unit shock to trade compared across three different forecasting methods

NOTE: For each shock, trade is set to its mean and prime minister is set to Thatcher. Triangle shows median expected conditional error variance, with dark to light vertical lines showing 75, 90, and 95% confidence intervals, respectively.

References

Hellwig, T. 2007. Economic openness, policy uncertainty, and the dynamics of government support. *Electoral Studies* 26: 772–786. URL https://doi.org/10.1016/j.electstud.2007.06.005.