Title

bayesgraph — Graphical summaries and convergence diagnostics

Description

`bayesgraph` provides graphical summaries and convergence diagnostics for simulated posterior distributions (MCMC samples) of model parameters and functions of model parameters obtained after Bayesian estimation. Graphical summaries include trace plots, autocorrelation plots, and various distributional plots.

Quick start

Trace plot, histogram, autocorrelation plot, and density plot for parameter `{p}`

```
bayesgraph diagnostics {p}
```

Add plots for parameter `{y:x1}`

```
bayesgraph diagnostics {p} {y:x1}
```

Same as above, but for all model parameters

```
bayesgraph diagnostics _all
```

Same as above, but for a function of model parameters `{y:x1}` and `{p}`

```
bayesgraph diagnostics ({y:x1}/{p})
```

Specify a blue trace plot line for all plots

```
bayesgraph diagnostics {p} {y:x1} {y:x2}, traceopts(lcolor(blue))
```

Specify a blue trace plot line only for the second trace plot

```
bayesgraph diagnostics {p} {y:x1} {y:x2}, trace2opts(lcolor(blue))
```

Trace plots for all parameters in a single graph

```
bayesgraph trace _all, byparm
```

Cumulative sum plot for parameter `{p}`

```
bayesgraph cusum {p}
```

Scatterplot matrix for parameters `{p}` and `{y:x1}`

```
bayesgraph matrix {p} {y:x1}
```

Autocorrelation plots for elements 1,1 and 2,1 of matrix parameter `{S}`

```
bayesgraph ac {S_1_1} {S_2_1}
```

Diagnostic plots for all parameters in the model and pause at least 3 seconds before displaying the next graph

```
bayesgraph diagnostics _all, sleep(3)
```

Same as above, but pause until the user presses any key

```
    bayesgraph diagnostics _all, wait
```

Same as above, but close the current Graph window when the next graph is displayed

```
    bayesgraph diagnostics _all, close
```

Histogram of the first 10 observations of the first simulated outcome plotted on one graph

```
    bayespredict {_ysim}, saving(predres)
    bayesgraph histogram {_ysim[1/10]} using predres, byparm
```

Density plot of the mean across observations of the simulated outcome labeled as `mymean`

```
    bayesgraph kdensity (mymean: @mean({_ysim})) using predres
```

Menu

Statistics > Bayesian analysis > Graphical summaries

Syntax

Syntax is presented under the following headings:

Graphical summaries for model parameters
Graphical summaries for predictions

Graphical summaries for model parameters

Graphical summaries and convergence diagnostics for a single parameter

`bayesgraph` *graph scalar_param* [, *singleopts*]

Graphical summaries and convergence diagnostics for multiple parameters

`bayesgraph` *graph spec* [*spec* ...] [, *multiopts*]

`bayesgraph matrix` *spec spec* [*spec* ...] [, *singleopts*]

Graphical summaries and convergence diagnostics for all parameters

`bayesgraph` *graph* `_all` [, *multiopts* `showreffects`[`(`*reref*`)`]]

scalar_param is a scalar model parameter specified as `{param}` or `{eqname:param}` or an expression *exprspec* of scalar model parameters. Matrix model parameters are not allowed, but you may refer to their individual elements.

exprspec is an optionally labeled expression of model parameters specified in parentheses:

`(`[*exprlabel*`:`]*expr*`)`

exprlabel is a valid Stata name, and *expr* is a scalar expression that may not contain matrix model parameters. See *Specifying functions of model parameters* in [BAYES] **Bayesian postestimation** for examples.

spec is either *scalar_param* or *exprspec*.

Graphical summaries for predictions

Graphical summaries for an individual prediction

`bayesgraph` *graph predspecsc* `using` *predfile* [, *singleopts*]

Graphical summaries for multiple predictions

`bayesgraph` *graph predspec* [*predspec* ...] `using` *predfile* [, *multiopts*]

`bayesgraph matrix` *predspec predspec* [*predspec* ...] `using` *predfile* [, *singleopts*]

predfile is the name of the dataset created by `bayespredict` that contains prediction results.

predspecsc may contain individual observations of simulated outcomes, `{_ysim#[#]}`; individual expected outcome values, `{_mu#[#]}`; individual simulated residuals, `{_resid#[#]}`; and other scalar predictions, `{`*label*`}`.

predspec is one of *yspec*, (*yexprspec*), or (*funcspec*). See *Different ways of specifying predictions and their functions* in [BAYES] **Bayesian postestimation**.

yspec is {*ysimspec* | *residspec* | *muspec* | *label*}.

ysimspec is `{_ysim#}` or `{_ysim#[`*numlist*`]}`, where `{_ysim#}` refers to all observations of the #th simulated outcome and `{_ysim#[`*numlist*`]}` refers to the selected observations, *numlist*, of the #th simulated outcome. `{_ysim}` is a synonym for `{_ysim1}`.

residspec is `{_resid#}` or `{_resid#[`*numlist*`]}`, where `{_resid#}` refers to all residuals of the #th simulated outcome and `{_resid#[`*numlist*`]}` refers to the selected residuals, *numlist*, of the #th simulated outcome. `{_resid}` is a synonym for `{_resid1}`.

muspec is `{_mu#}` or `{_mu#[`*numlist*`]}`, where `{_mu#}` refers to all expected values of the #th outcome and `{_mu#[`*numlist*`]}` refers to the selected expected values, *numlist*, of the #th outcome. `{_mu}` is a synonym for `{_mu1}`.

label is the name of the function simulated using `bayespredict`.

With large datasets, specifications `{_ysim#}`, `{_resid#}`, and `{_mu#}` may use a lot of time and memory and should be avoided. See *Generating and saving simulated outcomes* in [BAYES] **bayespredict**.

yexprspec is [*exprlabel*:]*yexpr*, where *exprlabel* is a valid Stata name and *yexpr* is a scalar expression that may contain individual observations of simulated outcomes, `{_ysim#[#]}`; individual expected outcome values, `{_mu#[#]}`; individual simulated residuals, `{_resid#[#]}`; and other scalar predictions, `{`*label*`}`.

funcspec is [*label*:]`@`*func*`(`*arg1*[, *arg2*]`)`, where *label* is a valid Stata name; *func* is an official or user-defined Mata function that operates on column vectors and returns a real scalar; and *arg1* and *arg2* are one of `{_ysim`[*#*]`}`, `{_resid`[*#*]`}`, or `{_mu`[*#*]`}`. *arg2* is primarily for use with user-defined Mata functions; see *Defining test statistics using Mata functions* in [BAYES] **bayespredict**.

graph	Description
`diagnostics`	multiple diagnostics in compact form
`trace`	trace plots
`ac`	autocorrelation plots
`histogram`	histograms
`kdensity`	density plots
`cusum`	cumulative sum plots
`matrix`	scatterplot matrix

`bayesgraph matrix` requires at least two parameters. `diagnostics`, `trace`, `ac`, and `cusum` are not relevant for predictions.

singleopts	Description
Chains	
chainopts	options controlling multiple chains
Options	
`skip(`*#*`)`	skip every # observations from the MCMC sample; default is `skip(0)`
`name(`*name*`, ...)`	specify name of graph
`saving(`*filename*`, ...)`	save graph in file
graphopts	graph-specific options

multiopts	Description
Chains	
chainopts	options controlling multiple chains
Options	
`byparm`[`(`*grbyparmopts*`)`]	specify the display of plots on one graph; default is separate graph for each plot; not allowed with graphs `diagnostics` and `matrix` or with options `combine()` and `bychain()`
`combine`[`(`*grcombineopts*`)`]	specify the display of plots on one graph; recommended when the number of parameters is large; not allowed with graphs `diagnostics` and `matrix` or with options `byparm()` and `bychain()`
`sleep(`*#*`)`	pause for # seconds between multiple graphs; default is `sleep(0)`
`wait`	pause until the —`more`— condition is cleared
[`no`]`close`	(do not) close Graph windows when the next graph is displayed with multiple graphs; default is `noclose`
`skip(`*#*`)`	skip every # observations from the MCMC sample; default is `skip(0)`
`name(`*namespec*`, ...)`	specify names of graphs
`saving(`*filespec*`, ...)`	save graphs in files
`graphopts(`*graphopts*`)`	control the look of all graphs; not allowed with `byparm()`
`graph#opts(`*graphopts*`)`	control the look of #th graph; not allowed with `byparm()`
graphopts	equivalent to `graphopts(`*graphopts*`)`; only one may be specified

chainopts	Description
`chains(_all` | *numlist*`)`	specify which chains to plot; default is to plot the first 10 chains
`sepchains`	draw a separate graph for each chain; default is to overlay chains
`chainslegend`	show legend keys corresponding to chain numbers; not allowed with graphs `diagnostics` and `matrix` or with options `combine()` and `byparm()`
`bychain`[`(`*grbychainopts*`)`]	plot each chain as a subgraph on one graph; default is all chains overlayed on one graph; not allowed with graphs `diagnostics` and `matrix` or with options `combine()` and `byparm()`
`chainopts(`*graphopts*`)`	control the look of all chains
`chain#opts(`*graphopts*`)`	control the look of #th chain

Options *chainopts* are relevant only when option `nchains()` is used with `bayesmh` or the `bayes` prefix.

graphopts	Description
diagnosticsopts	options for `bayesgraph diagnostics`
tslineopts	options for `bayesgraph trace` and `bayesgraph cusum`
acopts	options for `bayesgraph ac`
histopts	options for `bayesgraph histogram`
kdensityopts	options for `bayesgraph kdensity`
grmatrixopts	options for `bayesgraph matrix`

diagnosticsopts	Description
`traceopts(`*tslineopts*`)`	affect rendition of all trace plots
`trace#opts(`*tslineopts*`)`	affect rendition of #th trace plot
`acopts(`*acopts*`)`	affect rendition of all autocorrelation plots
`ac#opts(`*acopts*`)`	affect rendition of #th autocorrelation plot
`histopts(`*histopts*`)`	affect rendition of all histogram plots
`hist#opts(`*histopts*`)`	affect rendition of #th histogram plot
`kdensopts(`*kdensityopts*`)`	affect rendition of all density plots
`kdens#opts(`*kdensityopts*`)`	affect rendition of #th density plot
grcombineopts	any option documented in [G-2] **graph combine**

acopts	Description
`ci`	plot autocorrelations with confidence intervals; not allowed with `byparm()`
acopts	any options other than `generate()` documented for the `ac` command in [TS] **corrgram**

kdensityopts	Description
kdensopts	options for the overall kernel density plot
`show(`*showspec*`)`	show first-half density (`first`), second-half density (`second`), `both`, or `none`; default varies
`kdensfirst(`*kdens1opts*`)`	affect rendition of the first-half density plot
`kdenssecond(`*kdens2opts*`)`	affect rendition of the second-half density plot

Options

Chains

`chains(_all` | *numlist*`)` specifies which chains from the MCMC sample to plot. The default is to plot the first 10 chains. You can use `chains(_all)` to plot all chains.

`sepchains` specifies that a separate graph be drawn for each chain. This option is implied for `bayesgraph matrix` and may not be combined with `bychain()`.

`chainslegend` specifies that the graph be plotted with a legend showing keys corresponding to chain numbers. This option is not allowed with graphs `diagnostics` and `matrix` or with options `combine()` and `byparm()`.

`bychain`[`(`*grbychainopts*`)`] specifies that each chain be plotted as a subgraph on one graph. By default, all chains are displayed overlayed on one graph. This option is not allowed with graphs `diagnostics` and `matrix` or with options `combine()`, `byparm()`, and `sepchains`.

grbychainopts is any of the suboptions of `by()` documented in [G-3] ***by_option***.

`chainopts(`*graphopts*`)` and `chain#opts(`*graphopts*`)` control the look of chains. `chainopts()` controls the look of all chains but may be overridden for specific chains by using the `chain#opts()` option.

Chain-specific options are ignored if option `nchains()` is not specified with `bayesmh` or the `bayes` prefix.

Options

`byparm`[`(`*grbyparmopts*`)`] specifies the display of all plots of parameters as subgraphs on one graph. By default, a separate graph is produced for each plot when multiple parameters are specified. This option is not allowed with `bayesgraph diagnostics` or `bayesgraph matrix` and may not be combined with options `combine()` and `bychain()`. When many parameters or expressions are specified, this option may fail because of memory constraints. In that case, you may use option `combine()` instead.

grbyparmopts is any of the suboptions of `by()` documented in [G-3] ***by_option***.

`byparm()` allows y scales to differ for all graph types and forces x scales to be the same only for `bayesgraph trace` and `bayesgraph cusum`. Use `noyrescale` within `byparm()` to specify a common y axis, and use `xrescale` or `noxrescale` to change the default behavior for the x axis.

`byparm()` with `bayesgraph trace` and `bayesgraph cusum` defaults to displaying multiple plots in one column to accommodate the x axis with many iterations. Use `norowcoldefault` within `byparm()` to switch back to the default behavior of options `rows()` and `cols()` of the [G-3] ***by_option***.

`combine`[`(`*grcombineopts*`)`] specifies the display of all plots of parameters as subgraphs on one graph and is an alternative to `byparm()` with a large number of parameters. By default, a separate graph is produced for each plot when multiple parameters are specified. This option is not allowed with `bayesgraph diagnostics` or `bayesgraph matrix` and may not be combined with option `byparm()`. It can be used in cases where a large number of parameters or expressions are specified and the `byparm()` option would cause an error because of memory constraints.

grcombineopts is any of the options documented in [G-2] **graph combine**.

`sleep(`*#*`)` specifies pausing for # seconds before producing the next graph. This option is allowed only when multiple parameters are specified. This option may not be combined with `wait`, `combine()`, or `byparm()`.

`wait` causes `bayesgraph` to display —`more`— and pause until any key is pressed before producing the next graph. This option is allowed when multiple parameters are specified. This option may not be combined with `sleep()`, `combine()`, or `byparm()`. `wait` temporarily ignores the global setting that is specified using `set more off`.

[`no`]`close` specifies that, for multiple graphs, the Graph window be closed when the next graph is displayed. The default is `noclose` or to not close any Graph windows.

`skip(`*#*`)` specifies that every # observations from the MCMC sample not be used for computation. The default is `skip(0)` or to use all observations in the MCMC sample. Option `skip()` can be used to subsample or thin the chain. `skip(`*#*`)` is equivalent to a thinning interval of #+1. For example, if you specify `skip(1)`, corresponding to the thinning interval of 2, the command will skip every other observation in the sample and will use only observations 1, 3, 5, and so on in the computation. If you specify `skip(2)`, corresponding to the thinning interval of 3, the command will skip every 2 observations in the sample and will use only observations 1, 4, 7, and so on in the computation. `skip()` does not thin the chain in the sense of physically removing observations from the sample, as is done by, for example, `bayesmh`'s `thinning()` option. It only discards selected observations from the computation and leaves the original sample unmodified.

`name(`*namespec*[`, replace`]`)` specifies the name of the graph or multiple graphs. See [G-3] ***name_option*** for a single graph. If multiple graphs are produced, then the argument of `name()` is either a list of names or a *stub*, in which case graphs are named *stub*1, *stub*2, and so on. With multiple graphs, if `name()` is not specified and neither `sleep()` nor `wait` is specified, `name(Graph__#, replace)` is assumed, and thus the produced graphs may be replaced by subsequent `bayesgraph` commands.

The `replace` suboption causes existing graphs with the specified name or names to be replaced.

`saving(`*filespec*[`, replace`]`)` specifies the filename or filenames to use to save the graph or multiple graphs to disk. See [G-3] ***saving_option*** for a single graph. If multiple graphs are produced, then the argument of `saving()` is either a list of filenames or a *stub*, in which case graphs are saved with filenames *stub*1, *stub*2, and so on.

The `replace` suboption specifies that the file (or files) may be replaced if it already exists.

`showreffects` and `showreffects(`*reref*`)` are for use after multilevel models, and they specify that the results for all or a list *reref* of random-effects parameters be provided in addition to other model parameters. By default, all random-effects parameters are excluded from the results to conserve computation time.

`graphopts(`*graphopts*`)` and `graph#opts(`*graphopts*`)` affect the rendition of graphs. `graphopts()` affects the rendition of all graphs but may be overridden for specific graphs by using the `graph#opts()` option. The options specified within `graph#opts()` are specific for each type of graph.

The two specifications

```
        bayesgraph ..., graphopts(graphopts)
```

and

```
        bayesgraph ..., graphopts
```

are equivalent, but you may specify one or the other.

These options are not allowed with `byparm()` and when only one parameter is specified.

graphopts specifies options specific to each graph type.

diagnosticsopts specifies options for use with `bayesgraph diagnostics`. See the corresponding table in the syntax diagram for a list of options.

tslineopts specifies options for use with `bayesgraph trace` and `bayesgraph cusum`. See the options of [TS] **tsline** except `by()`.

acopts specifies options for use with `bayesgraph ac`.

`ci` requests that the graph of autocorrelations with confidence intervals be plotted. By default, confidence intervals are not plotted. This option is not allowed with `byparm()`.

acopts specifies any options except `generate()` of the `ac` command in [TS] **corrgram**.

histopts specifies options for use with `bayesgraph histogram`. See options of [R] **histogram** except `by()`.

kdensityopts specifies options for use with `bayesgraph kdensity`.

kdensopts specifies options for the overall kernel density plot. See the options documented in [R] **kdensity** except `generate()` and `at()`.

`show(`*showspec*`)` specifies which kernel density curves to plot. *showspec* is one of `first`, `second`, `both`, or `none`. If `show(first)` is specified, only the first-half density curve, obtained from the first half of an MCMC sample, is plotted. If `show(second)` is specified, only the second-half density curve, obtained from the second half of an MCMC sample, is plotted. `show(both)`, the default with graph `diagnostics`, overlays both the first-half density curve and the second-half density curve with the overall kernel density curve. `show(none)`, the default with graph `kdensity`, shows only the overall kernel density curve.

`kdensfirst(`*kdens1opts*`)` specifies options of [G-2] **graph twoway kdensity** except `by()` to affect rendition of the first-half kernel density plot.

`kdenssecond(`*kdens2opts*`)` specifies options of [G-2] **graph twoway kdensity** except `by()` to affect rendition of the second-half kernel density plot.

grmatrixopts specifies options for use with `bayesgraph matrix`. See the options of [G-2] **graph matrix** except `by()`.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Using bayesgraph

bayesgraph requires specifying at least one parameter with all graph types except matrix, which requires at least two parameters. To request graphs for all parameters, use _all.

When multiple graphs are produced, they are automatically stored in memory with names Graph__*#* and will all appear on the screen. After you are done reviewing the graphs, you can type

```
. graph close Graph__*
```

to close these graphs or type

```
. graph drop Graph__*
```

to close the graphs and drop them from memory.

If you would like to see only one graph at a time, you can specify option close to close the Graph window when the next graph is displayed. You can also use option sleep() or option wait to pause between the subsequent graphs. The sleep(*#*) option causes each graph to pause for # seconds. The wait option causes bayesgraph to wait until a key is pressed before producing the next graph.

You can combine separate graphs into one by specifying one of byparm() or combine(). These options are not allowed with diagnostics or matrix graphs. The byparm() option produces more compact graphs, but it may not be feasible with many parameters or expressions and large sizes of MCMC samples.

With multiple graphs, you can control the look of each individual graph with graph*#*opts(). Options common to all graphs may be specified in graphopts() or passed directly to the command as with single graphs.

With multiple chains, bayesgraph plots only the first 10 chains by default. If you have more than 10 chains, although only four chains are commonly used in practice, you can use the chains(_all) option to plot all the chains. You can also use the chains() option to handpick the chains you want to be plotted. For example, chains(1/3 5) will plot chains 1, 2, 3, and 5. If desired, you can see which plot corresponds to which chain by using the chainslegend option.

By default, the chains will be plotted overlaid on one graph. You can specify the sepchains option to plot each chain on a separate graph, in which case the graphs will be automatically stored in memory with names Graph__*#* and will all appear on the screen. Or, you can use the bychain option to plot each chain separately but one graph.

To control the look of an individual chain, you can use the chain*#*opts() options. For example, to change the line color to red for chain 2, you would specify the chain2opts(lcolor(red)) option. To control the look of all chains, you can use the chainopts() option.

You can use bayesgraph to plot predicted quantities when you supply the prediction dataset generated by bayespredict in the using specification. Also see *Different ways of specifying predictions and their functions* in [BAYES] **Bayesian postestimation**.

Examples

We demonstrate the bayesgraph command using an example of Bayesian normal linear regression applied to auto.dta. We model the mpg variable using a normal distribution with unknown mean and variance. Our Bayesian model thus has two parameters, {mpg:_cons} and {var}, for which we need to specify prior distributions. We consider fairly noninformative prior distributions for these parameters: $N(0, 1000)$ for the constant and inverse gamma with shape and scale of 0.1 for the variance. Because the specified prior distributions are independent and semiconjugate relative to the normal data distribution, we can use Gibbs sampling for both parameters instead of the default MH sampling. To illustrate, we will use Gibbs sampling for the variance and MH sampling (default) for the mean.

We use bayesmh to fit our model.

```
. use https://www.stata-press.com/data/r18/auto
(1978 automobile data)

. bayesmh mpg, likelihood(normal({var}))
> prior({mpg:_cons}, normal(0,1000))
> prior({var}, igamma(0.1,0.1)) block({var}, gibbs) rseed(14)
Burn-in ...
Simulation ...

Model summary
------------------------------------------------------------------------------
Likelihood:
  mpg ~ normal({mpg:_cons},{var})

Priors:
  {mpg:_cons} ~ normal(0,1000)
        {var} ~ igamma(0.1,0.1)
------------------------------------------------------------------------------

Bayesian normal regression                       MCMC iterations  =     12,500
Metropolis-Hastings and Gibbs sampling           Burn-in          =      2,500
                                                 MCMC sample size =     10,000
                                                 Number of obs    =         74
                                                 Acceptance rate  =      .7133
                                                 Efficiency:  min =      .2331
                                                              avg =      .6166
Log marginal-likelihood =  -242.1155                          max =          1

------------------------------------------------------------------------------
             |                                                Equal-tailed
             |      Mean   Std. dev.     MCSE     Median  [95% cred. interval]
-------------+----------------------------------------------------------------
mpg          |
       _cons |  21.29231   .6648867   .013771   21.29419   19.94367   22.56746
-------------+----------------------------------------------------------------
         var |   34.2805   5.844213   .058442    33.6464   24.65882    47.5822
------------------------------------------------------------------------------
```

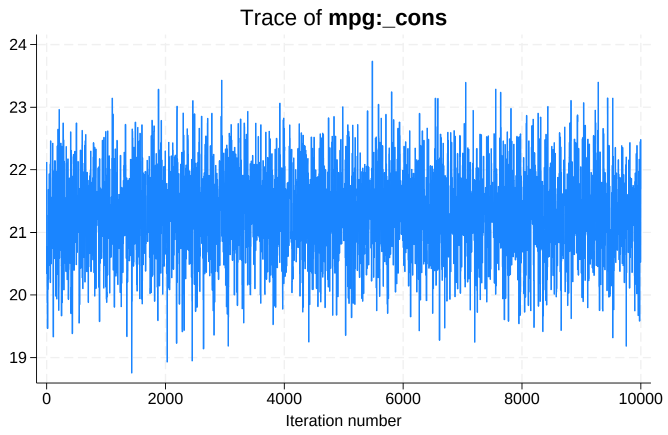
The MCMC simulation has a fairly high efficiency for the MH algorithm of 23% for the mean and an efficiency of 1 for the variance because of the Gibbs sampling. The output suggests no convergence problems. However, it is important to verify this and to also inspect various other graphical summaries of the parameters. This example demonstrates graphical summaries for a well-mixing MCMC chain that has converged and that generates samples from the posterior distribution of the model. For examples of poor-mixing MCMC chains, see *Convergence diagnostics in MCMC* in [BAYES] **Intro**.

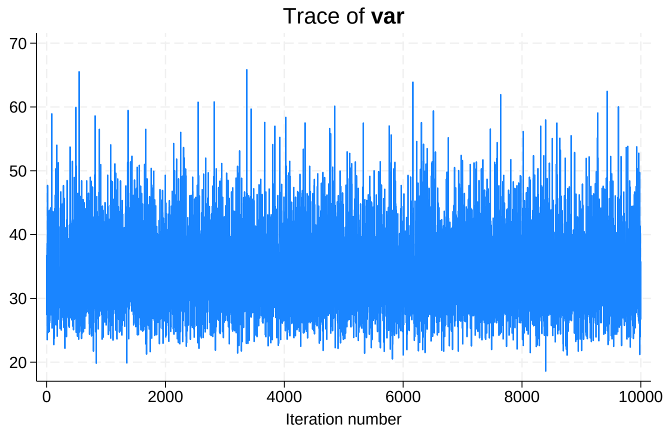
Trace plots

We start with trace plots, which plot the values of the simulated parameters against the iteration number and connect consecutive values with a line. For a well-mixing parameter, the range of the parameter is traversed rapidly by the MCMC chain, which makes the drawn lines look almost vertical and dense. Sparseness and trends in the trace plot of a parameter suggest convergence problems.

Let's use `bayesgraph trace` to obtain trace plots for `{mpg:_cons}` and `{var}`. We specify `_all` to request both plots at once.

```
. bayesgraph trace _all
```

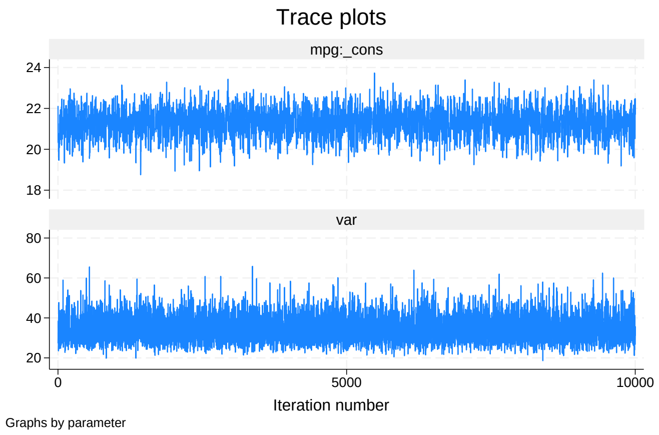




The mean parameter mixes very well and the variance parameter mixes perfectly.

Alternatively, we can use the byparm() option to plot results on one graph.

```
. bayesgraph trace _all, byparm
```



bayesgraph trace (as well as bayesgraph cusum) with option byparm() displays multiple plots in one column to accommodate an x axis with many iterations. You can specify byparm(norowcoldefault) to switch to the default behavior of options rows() and cols() documented in [G-3] ***by_option***.

Also see *Convergence diagnostics using multiple chains* in [BAYES] **bayesmh** for an example of trace plots with multiple chains.

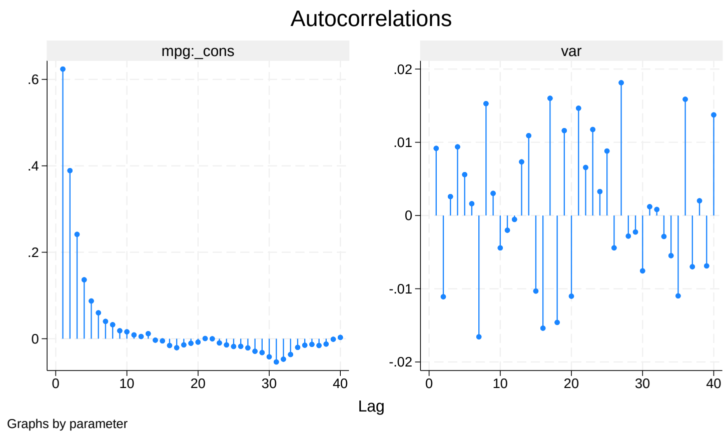
Autocorrelation plots

The second graphical summary we demonstrate is an autocorrelation plot. This plot shows the degree of autocorrelation in an MCMC sample for a range of lags, starting from lag 0. At lag 0, the plotted value corresponds to the sample variance of MCMC.

Autocorrelation is usually present in any MCMC sample. Typically, autocorrelation starts from some positive value for lag 0 and decreases toward 0 as the lag index increases. For a well-mixing MCMC chain, autocorrelation dies off fairly rapidly.

For example, autocorrelation for {mpg:_cons} becomes negligible after about lag 8 and is basically nonexistent for {var}.

```
. bayesgraph ac _all, byparm
```



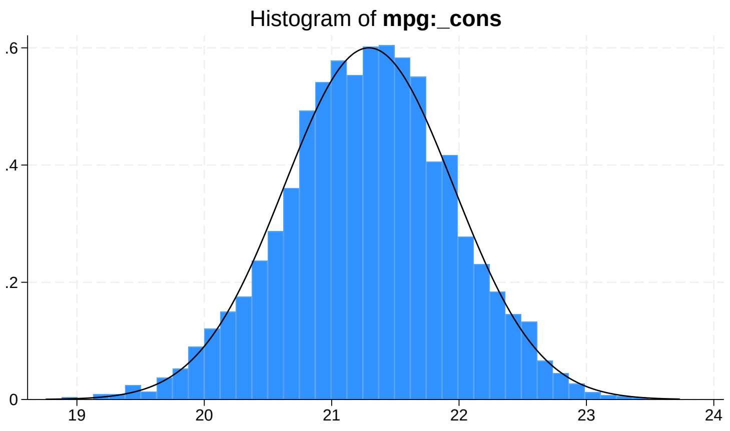
Autocorrelation lags are approximated by correlation times of parameters as reported by the bayesstats ess command; see [BAYES] **bayesstats ess** for details. Autocorrelation lags are also used to determine the batch size for the batch-means estimator of the MCMC standard errors; see [BAYES] **bayesstats summary**.

Histogram plots

Graphical posterior summaries such as histograms and kernel density estimates provide useful additions to the various numerical statistics (see [BAYES] **bayesstats summary**) for summarizing MCMC output. It is always a good practice to inspect the histogram and kernel density estimates of the marginal posterior distributions of parameters to ensure that these empirical distributions behave as expected. These plots can be used to compare the empirical posterior and the specified prior distributions to visualize the impact of the data.

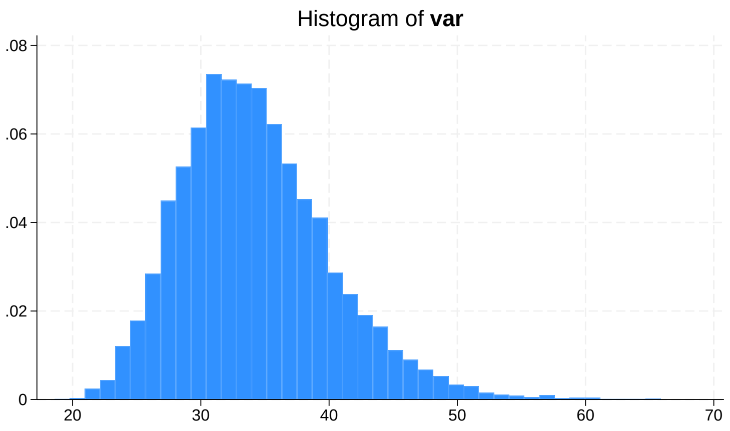
A histogram depicts the general shape of the marginal posterior distribution of a model parameter. Let's look at histograms of our parameters.

```
. bayesgraph histogram {mpg:_cons}, normal
```



The distribution of {mpg:_cons} is in good agreement with the normal distribution. This is not surprising, because the specified conjugate normal prior implies that the marginal posterior for {mpg:_cons} is a normal distribution. The unimodal histogram is also another confirmation that we have obtained a good simulation of the marginal posterior distribution of {mpg:_cons}.

```
. bayesgraph histogram {var}
```



The histogram for `{var}` is also unimodal but is slightly skewed to the right. This is also in agreement with the specified prior because the marginal posterior for the variance is inverse gamma for the specified model.

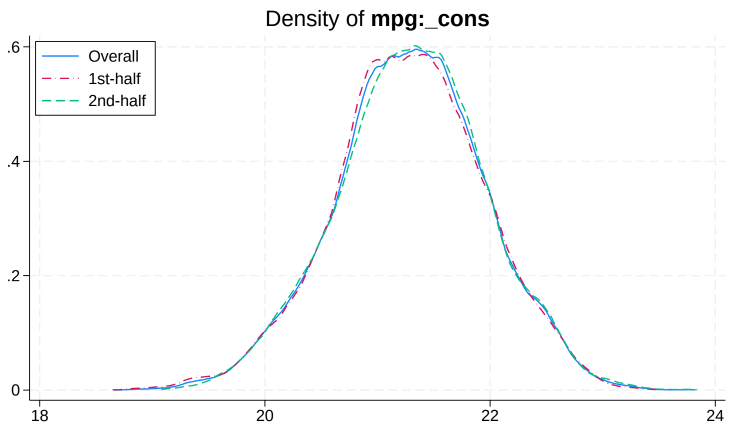
For examples of histograms for prediction quantities, see example 4 and example 7 in [BAYES] **bayespredict** and example 1 and example 3 in [BAYES] **bayesstats ppvalues**.

Kernel density plots

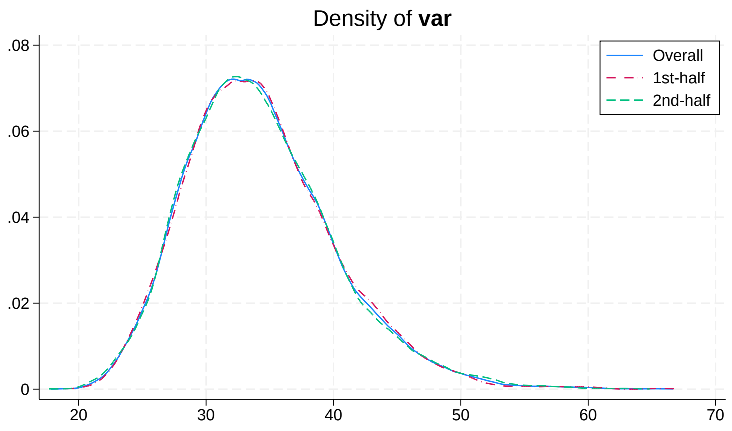
Kernel density plots provide alternative visualizations of the simulated marginal posterior distributions. They may be viewed as smoothed histograms. By default, the `bayesgraph kdensity` command shows an overall density of the entire MCMC sample. To explore convergence, the command provides the `show(both)` option, which additionally plots two density curves: the first-half density obtained using the first half of the MCMC sample and the second-half density obtained using the second half of the MCMC sample. If the chain has converged and mixes well, we expect the three density curves to be close to each other. Large discrepancies between the first-half curve and the second-half curve suggest convergence problems.

Let's look at the three kernel density curves for our two parameters.

```
. bayesgraph kdensity {mpg:_cons}, show(both)
```



```
. bayesgraph kdensity {var}, show(both)
```



Kernel density plots for {mpg:_cons} and {var} are similar in shape to the histograms' plots from the previous section. All three density curves are close to each other for both parameters.

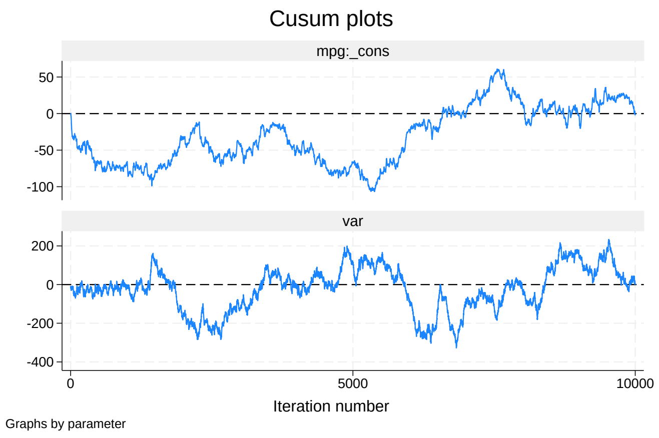
Also see *Convergence diagnostics using multiple chains* in [BAYES] **bayesmh** for an example of kernel density plots with multiple chains.

Cumulative sum plots

Cumulative sum (cusum) plots are useful graphical summaries for detecting persistent trends in MCMC chains. All cusum plots start and end at 0 and may or may not cross the x axis. There is great variability in the looks of cusum plots, which make them difficult to interpret sometimes. Typically, if the cusum line never crosses the x axis, this may indicate a problem. See, for example, *Convergence diagnostics of* MCMC in [BAYES] **Intro** for a cusum plot demonstrating convergence problems.

By inspecting a cusum plot, we may detect an early drift in the simulated sample because of an insufficient burn-in period. In cases of pronounced persistent trends, the cusum curve may stay either in the positive or in the negative y plane. For a well-mixing parameter, the cusum curve typically crosses the x axis several times. This is the case for the cusum plots of {mpg:_cons} and {var}.

```
. bayesgraph cusum _all, byparm
```



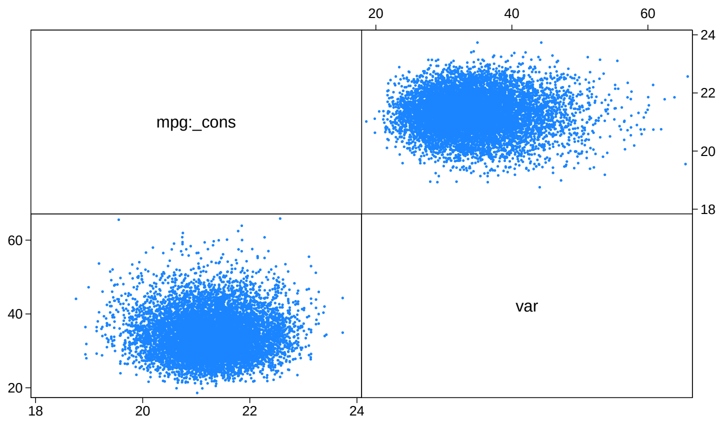
Bivariate scatterplots

The `bayesgraph matrix` command draws bivariate scatterplots of model parameters based on MCMC samples. A bivariate scatterplot represents a joint sample posterior distribution for pairs of parameters. It may reveal correlation between parameters and characterize a general shape of a multivariate posterior distribution. For example, bivariate scatterplots are useful for detecting multimodal posterior distributions.

Typically, scatterplots depict clouds of points. Sparseness and irregularities in the scatterplots can be strong indications of nonconvergence of an MCMC. For a well-mixing chain, the scatterplots have an ellipsoidal form with an increasing concentration around the posterior mode.

This scatterplot of {mpg:_cons} and {var} is an example of a well-behaved scatterplot.

```
. bayesgraph matrix {mpg:_cons} {var}
```

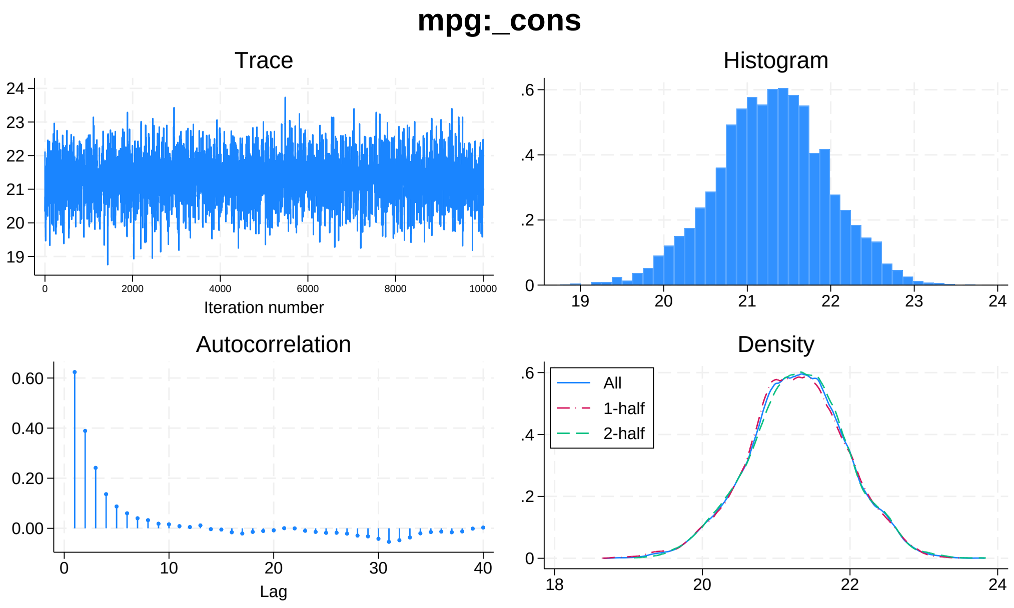


Diagnostic plots

Finally, we demonstrate the `bayesgraph diagnostics` command, which combines the trace, histogram, autocorrelation, and kernel density plots compactly on one graph. We already discussed the individual plots in the previous sections. Diagnostic plots are convenient for inspecting the overall behavior of a particular model parameter. We recommend that diagnostic plots for all parameters be inspected routinely as a part of the convergence-checking process.

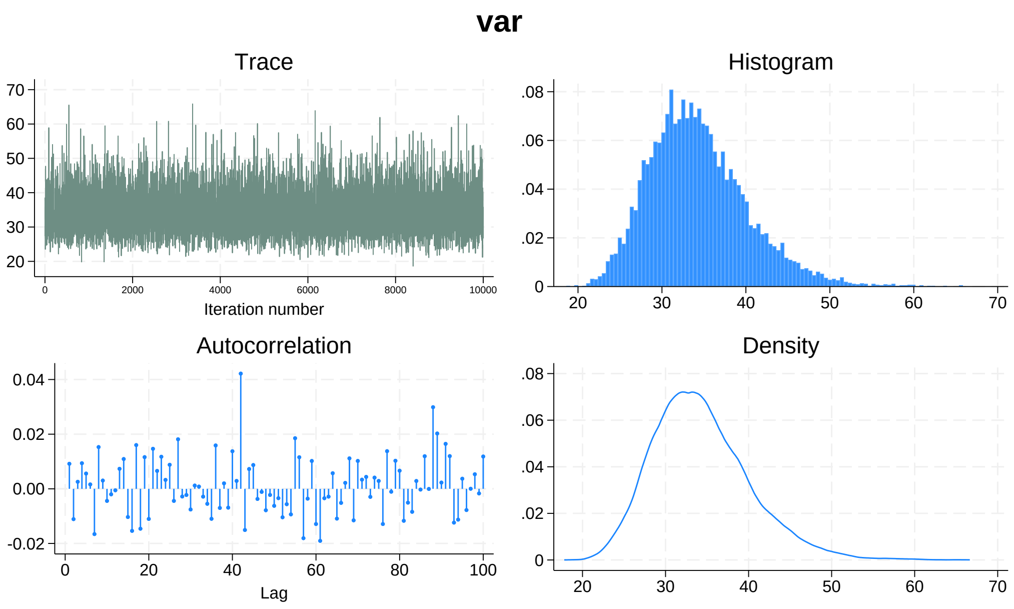
Let's obtain the diagnostic plot for {mpg:_cons}.

```
. bayesgraph diagnostics {mpg:_cons}
```



In the diagnostics plot for {var}, let's also demonstrate the use of several options of the depicted plots.

```
. bayesgraph diagnostics {var}, traceopts(lwidth(0.2) lcolor(teal))
> acopts(lag(100)) histopts(bins(100)) kdensopts(show(none))
```



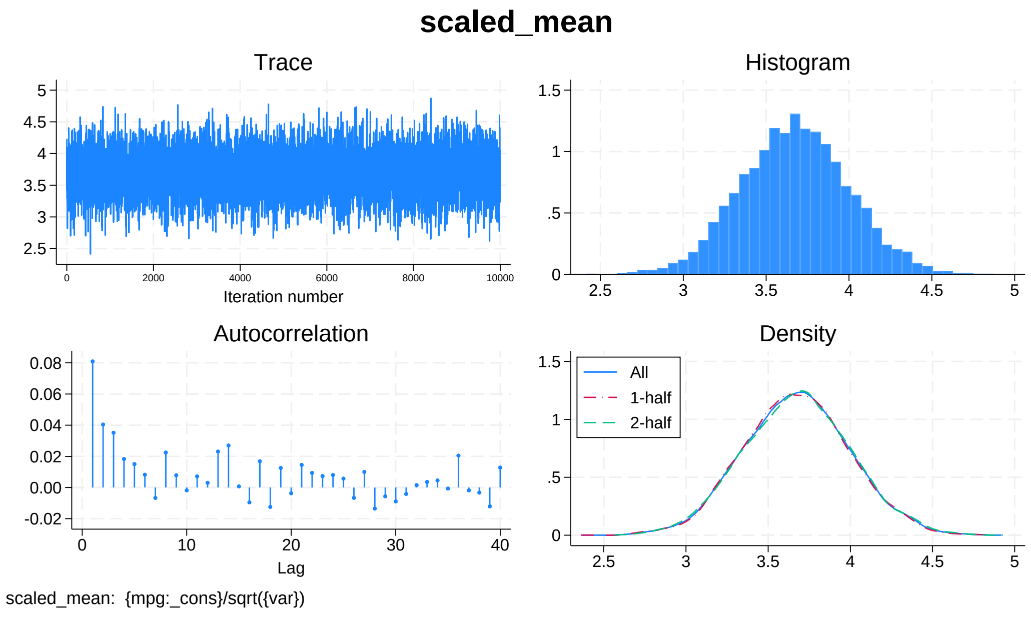
In the above, we changed the width and color of the trace line, the maximum lag for calculating the autocorrelation, the number of bins for the histogram, and requested that the two subsample kernel densities not be shown on the kernel density plot.

Also see *Convergence diagnostics using multiple chains* in [BAYES] **bayesmh** for an example of diagnostics plots with multiple chains.

Functions of model parameters

All `bayesgraph` subcommands can provide graphical summaries of functions of model parameters. Below we apply `bayesgraph diagnostics` to the expression `{mpg:_cons}/sqrt({var})`, which we label as `scaled_mean`.

```
. bayesgraph diagnostics (scaled_mean: {mpg:_cons}/sqrt({var}))
```



If you detect convergence problems in a function of parameters, you must inspect every parameter used in the expression individually. In fact, we recommend that you inspect all model parameters before you proceed with any postestimation analysis.

Methods and formulas

Let θ be a scalar model parameter and $\{\theta_t\}_{t=1}^{T}$ be an MCMC sample of size T drawn from the marginal posterior distribution of θ.

The trace plot of θ plots θ_t against t with connecting lines for $t = 1, \dots, T$.

The autocorrelation plot of θ shows the autocorrelation in the $\{\theta_t\}_{t=1}^{T}$ sample for lags from 0 to the `lag(#)` option of the `ac` command.

The histogram and kernel density plots of θ are drawn using the `histogram` and `kdensity` commands.

Yu and Mykland (1998) proposed a graphical procedure for assessing the convergence of individual parameters based on cumulative sums, also known as a cusum plot. The cusum plot for θ plots S_t against t for $t = 1, \ldots, T$ and connects the successive points. S_t is the cumulative sum at time t:

$$S_t = \sum_{k=1}^{t} (\theta_k - \widehat{\theta}), \quad \widehat{\theta} = \frac{1}{T} \sum_{k=1}^{T} \theta_k$$

and $S_0 = 0$.

The scatterplot of two model parameters θ^1 and θ^2 plots points (θ_t^1, θ_t^2) for $t = 1, \ldots, T$.

With multiple chains, the plots are produced separately for each chain.

References

Huber, C. 2016. Introduction to Bayesian statistics, part 2: MCMC and the Metropolis–Hastings algorithm. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/11/15/introduction-to-bayesian-statistics-part-2-mcmc-and-the-metropolis-hastings-algorithm/.

Yu, B., and P. Mykland. 1998. Looking at Markov samplers through cusum path plots: A simple diagnostic idea. *Statistics and Computing* 8: 275–286. https://doi.org/10.1023/A:1008917713940.

Also see

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

[BAYES] **bayesmh** — Bayesian models using Metropolis–Hastings algorithm

[BAYES] **Bayesian postestimation** — Postestimation tools for bayesmh and the bayes prefix

[BAYES] **bayesstats ess** — Effective sample sizes and related statistics

[BAYES] **bayesstats summary** — Bayesian summary statistics

[G-2] **graph matrix** — Matrix graphs

[G-2] **graph twoway kdensity** — Kernel density plots

[R] **histogram** — Histograms for continuous and categorical variables

[R] **kdensity** — Univariate kernel density estimation

[TS] **corrgram** — Tabulate and graph autocorrelations

[TS] **tsline** — Time-series line plots