Title

stata.com

bayesirf create — Obtain Bayesian IRFs, dynamic-multiplier functions, and FEVDs

Description

`bayesirf create` computes posterior summaries of impulse–response functions (IRFs), dynamic-multiplier functions, and forecast-error variance decompositions (FEVDs). Posterior means, medians, and credible intervals of all of these functions are referred to collectively as Bayesian IRF results and are saved in an IRF file under a specified filename. Once you have created a set of Bayesian IRF results, you can use the other `bayesirf` commands to analyze them.

Quick start

Create IRF `myirf` with 8 forecast periods in the active IRF file

    bayesirf create myirf

As above, but save the entire Markov chain Monte Carlo (MCMC) sample of results in `myirfmcmc.dta` (required when option `clevel()` or `hpd` is specified with other `bayesirf` subcommands)

    bayesirf create myirf, mcmcsaving(myirfmcmc)

Compute IRF for 12 periods and use `myirfs.irf` file for saving results

    bayesirf create myirf, set(myirfs) step(12)

As above, but compute 80% highest posterior density (HPD) credible intervals instead of 95% equal-tailed credible intervals

    bayesirf create myirf, set(myirfs) step(12) clevel(80) hpd

Note: `bayesirf` commands can be used after `bayes: var`, `bayes: dsge`, or `bayes: dsgenl`; see [BAYES] **bayes: var**, [BAYES] **bayes: dsge**, or [BAYES] **bayes: dsgenl**.

Menu

Statistics > Multivariate time series > Bayesian models > IRF and FEVD analysis

Syntax

`bayesirf create` *irfname* [, *options*]

irfname is any valid name that does not exceed 15 characters.

options	Description
Main	
`set(`*filename*[`, replace`]`)`	make *filename* active
`replace`	replace *irfname* if it already exists
`step(#)`	set forecast horizon to #; default is `step(8)`
`order(`*varlist*`)`	specify Cholesky ordering of endogenous variables; available only after `bayes: var`
`estimates(`*estname*`)`	use previously stored results *estname*; default is to use active results
Bayesian	
`clevel(#)`	set credible interval level; default is `clevel(95)`
`equaltailed`	save equal-tailed credible intervals; the default
`hpd`	save HPD credible intervals instead of the default equal-tailed credible intervals
`mcmcsaving(`*filename*[`, replace`]`)`	save simulation results to *filename*`.dta`
`mcmcsaving`	save simulation results to *irfname*`_mcmc.dta`

`bayesirf create` can be used only after `bayes: var`, `bayes: dsge`, and `bayes: dsgenl`.

You must `tsset` your data before using `bayes: var` or `bayes: dsge` and, hence, before using `bayesirf create`; see [TS] **tsset**.

Options

Main

`set(`*filename*[`, replace`]`)`, `replace`, `step(#)`, `order(`*varlist*`)`, and `estimates(`*estname*`)`; see [TS] **irf create**. Option `order()` is available only after estimation using `bayes: var`.

Bayesian

`clevel(#)` specifies the credible level, as a percentage, for equal-tailed and HPD credible intervals. The default is `clevel(95)` or as set by [BAYES] **set clevel**.

`hpd` displays the HPD credible intervals instead of the default equal-tailed credible intervals.

`mcmcsaving(`*filename*[`, replace`]`)` saves simulation results in *filename*`.dta`. The `replace` option specifies to overwrite *filename*`.dta` if it exists. If the `mcmcsaving()` option is not specified, simulation results are not saved.

The saved dataset has the following structure. Variable `_chain` records chain identifiers. Variable `_index` records iteration numbers. `bayesirf create` saves only states (sets of values) that are different from one iteration to another and the frequency of each state in variable `_frequency`. As such, `_index` may not necessarily contain consecutive integers. Remember to use `_frequency` as a frequency weight if you need to obtain any summaries of this dataset. MCMC values for each computed function *func* for each combination of an impulse $\#_1$ and response $\#_2$ variables and for each time period t are saved in a separate variable in the dataset. These variables are named as *func*_#$_1$_#$_2$_t.

`mcmcsaving` saves the simulation results in *irfname*`_mcmc.dta`.

Remarks and examples

stata.com

Please read [TS] **irf** first. An introductory example using IRFs is presented there.

`bayesirf create` estimates several types of IRFs, dynamic-multiplier functions, and FEVDs. Which estimates are saved depends on the estimation method previously used to fit the model.

Saves	Estimation command `var`	`dsge/` `dsgenl`
simple IRFs	x	x
orthogonalized IRFs	x	
dynamic multipliers	x	
cumulative IRFs	x	
cumulative orthogonalized IRFs	x	
cumulative dynamic multipliers	x	
Cholesky FEVDs	x	

`bayesirf` computes results based on the MCMC sample from the corresponding posterior distributions of IRF and other functions, which we will call the IRF MCMC sample. `bayesirf create` computes posterior means, medians, standard deviations, and, by default, 95% equal-tailed credible intervals for all functions and saves them in *irfname*`.dta`. When you later display or graph credible intervals by using, for instance, `bayesirf table` or `bayesirf graph`, the default credible intervals will be reported. If, for instance, you want to change the default level by using `clevel()` or compute HPD credible intervals by using `hpd` with those commands, you must first save the IRF MCMC sample by using `mcmcsaving()` with `bayesirf create`. For example,

```
. bayesirf create myirf, mcmcsaving(myirfmcmc)
```

You can also specify the `clevel()` or `hpd` option directly with `bayesirf create` to save the desired credible intervals in the current IRF file to be used by all `bayesirf` subcommands by default.

Remarks and examples are presented under the following headings:

IRFs after Bayesian vector autoregression (VAR) models

▷ Example 1: Bayesian VAR(2) model with default prior

We revisit example 1 from the documentation of the `irf create` command. It uses the `lutkepohl2` dataset of West Germany microeconomic quarterly data for the years between 1960 and 1978. The example studies the relationships between investment, `dln_inv`, income, `dln_inc`, and consumption, `dln_consump`.

```
. use https://www.stata-press.com/data/r17/lutkepohl2
. tsset
```

Using the `bayes: var` command, we fit a Bayesian VAR model with two lags on the dependent variables `dln_inv`, `dln_inc`, and `dln_consump`.

```
. bayes, rseed(17) saving(bvarex1) nomodelsummary:
> var dln_inv dln_inc dln_consump if qtr>=tq(1961q2) & qtr<=tq(1978q4)
Burn-in ...
Simulation ...

Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
```

```
Sample: 1961q2 thru 1978q4                      Number of obs    =         71
                                                Acceptance rate  =          1
                                                Efficiency:  min =      .9556
                                                             avg =      .9962
Log marginal-likelihood =   467.75286                        max =          1
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
dln_inv						
dln_inv						
L1.	.4749526	.1046821	.001071	.4762824	.2706787	.6790291
L2.	.0062935	.063174	.000632	.0058376	-.1181113	.129959
dln_inc						
L1.	.1150521	.4145854	.004146	.1155755	-.7122031	.9358321
L2.	.0096558	.2461088	.002464	.0129206	-.4780951	.490937
dln_consump						
L1.	-.0693822	.4910385	.004828	-.0712677	-1.016477	.9050535
L2.	.0182113	.2919327	.002919	.0169657	-.5563898	.6010627
_cons	.0067839	.0153897	.000154	.0067986	-.0233363	.0367596
dln_inc						
dln_inv						
L1.	.0152113	.0248328	.000248	.0154024	-.0341219	.0635173
L2.	.000957	.0149204	.000147	.0010833	-.0285813	.0306545
dln_inc						
L1.	.600281	.0981275	.000981	.5997577	.4077653	.7928394
L2.	.011757	.0577031	.000577	.0123101	-.1009659	.1245041
dln_consump						
L1.	-.0331359	.1151265	.001151	-.0318916	-.2594495	.1939938
L2.	-.0266197	.0694851	.000695	-.0263958	-.1637059	.1123704
_cons	.0084678	.0036265	.000037	.0084371	.0013034	.0155666
dln_consump						
dln_inv						
L1.	-.0183312	.0220482	.00022	-.0182937	-.062597	.0243933
L2.	.0092806	.0135179	.000135	.0094044	-.0171007	.036166
dln_inc						
L1.	-.0365965	.0875614	.000876	-.0368425	-.2086565	.1364804
L2.	.0345945	.0520216	.000514	.0339648	-.0668323	.136918
dln_consump						
L1.	.5444814	.1030406	.001027	.5432019	.3416401	.7489821
L2.	.0555939	.0617942	.000618	.055126	-.063175	.1763757
_cons	.0078414	.0032597	.000033	.0078245	.001402	.0141132
Sigma_1_1	.003945	.0006693	6.4e-06	.0038783	.0028446	.0054382
Sigma_2_1	-.0000314	.0001118	1.1e-06	-.0000291	-.0002548	.0001897
Sigma_3_1	.000138	.0001007	1.0e-06	.0001355	-.0000512	.0003478
Sigma_2_2	.0002195	.0000373	3.7e-07	.0002158	.0001579	.0003039
Sigma_3_2	.0000502	.0000238	2.4e-07	.000049	6.46e-06	.0001007
Sigma_3_3	.0001743	.0000294	2.9e-07	.0001714	.0001261	.0002408

file **bvarex1.dta** saved.

There are 21 regression coefficients in the model. By default, `bayes: var` applies a conjugate Minnesota prior on regression coefficients, the effect of which may be difficult to observe directly from the output table. The IRF functions provide a more accessible interpretation of estimation results by assessing the effect of an instant change in one variable on the rest as this effect develops in time. It would be interesting to see a comparison between Bayesian and frequentist results.

Before continuing, let's check the stability condition of the model. The interpretation of IRFs assumes that this condition is satisfied.

```
. bayesvarstable
Eigenvalue stability condition                 Companion matrix size =     6
                                               MCMC sample size      = 10000
```

Eigenvalue modulus	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
1	.7295294	.0952871	.000953	.7272906	.547312	.9209245
2	.6039037	.1045099	.001045	.6094994	.3810883	.7904044
3	.428933	.1272649	.001273	.4239249	.2113325	.6645651
4	.2126552	.0780213	.00078	.1997342	.0900884	.3846134
5	.1378018	.0565196	.000565	.1349177	.0385605	.2577174
6	.0759403	.05052	.000505	.0700686	.0035577	.1847619

```
Pr(eigenvalues lie inside the unit circle) = 0.9966
```

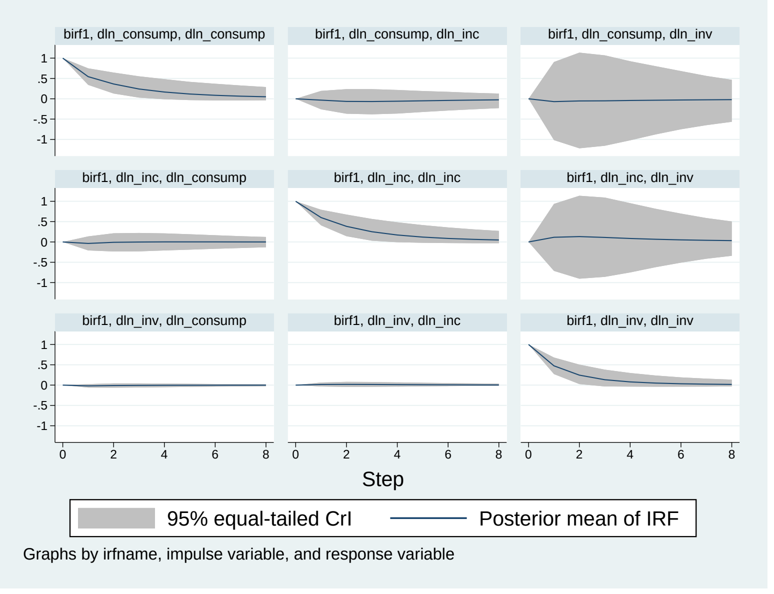
The unit circle inclusion probability for eigenvalues is essentially 1, so the stability condition is satisfied.

We continue with computing IRFs for 8 steps ahead and save the results as `birf1` in `birfex1.irf`.

```
. bayesirf create birf1, step(8) set(birfex1)
(file birfex1.irf created)
(file birfex1.irf now active)
(file birfex1.irf updated)
```

A quick way to inspect IRF estimates is by using `bayesirf graph`.

```
. bayesirf graph irf
```



There are nine IRF graphs, one for each combination of the three impulses and three responses. ◁

▷ Example 2: Bayesian VAR(2) model with weakly informative prior

To see the effect of priors on regression coefficients, we fit a second model in which we relax the Minnesota prior by changing the `selftight()` parameter from the default of 0.1 to 1. The effect of this change is that now the Bayesian estimates will be closer to the frequentist ones, as would be obtained from the corresponding [TS] **var** command.

```
. bayes, minnconjprior(selftight(1)) rseed(17) saving(bvarex2) nomodelsummary:
> var dln_inv dln_inc dln_consump if qtr>=tq(1961q2) & qtr<=tq(1978q4)
Burn-in ...
Simulation ...

Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
```

Sample: 1961q2 thru 1978q4	Number of obs = 71
	Acceptance rate = 1
	Efficiency: min = .9551
	avg = .9982
Log marginal-likelihood = 516.18125	max = 1

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
dln_inv						
dln_inv						
L1.	-.291233	.1205245	.001233	-.2896978	-.5273294	-.0564433
L2.	-.147377	.1174619	.001175	-.1479881	-.37888	.0835443
dln_inc						
L1.	.2349793	.5412359	.005412	.2376725	-.8448062	1.296301
L2.	.0318927	.5068351	.005074	.0364385	-.9534818	1.014282
dln_consump						
L1.	.7590264	.6437021	.006356	.7512454	-.4969188	2.034697
L2.	.7816876	.6184552	.006185	.7857257	-.4503459	2.015964
_cons	-.0115762	.0166601	.000167	-.0115223	-.0447488	.0209634
dln_inc						
dln_inv						
L1.	.0437786	.031111	.000311	.0439332	-.017398	.1045939
L2.	.0455046	.0301702	.000296	.0456176	-.0144909	.1057367
dln_inc						
L1.	-.1070955	.1398919	.001399	-.1073961	-.3828335	.1651545
L2.	.0235544	.1295408	.001295	.0245432	-.2289609	.2773168
dln_consump						
L1.	.2556043	.1658887	.001659	.2566669	-.0714113	.5763302
L2.	-.0311667	.1611506	.001612	-.0307495	-.3473144	.2870275
_cons	.0158357	.004275	.000043	.0158581	.0074012	.024185
dln_consump						
dln_inv						
L1.	-.0043581	.0251223	.000251	-.0044075	-.0539712	.0445555
L2.	.0340665	.024665	.000247	.0340276	-.0140267	.082563
dln_inc						
L1.	.1833481	.1134026	.001134	.1830458	-.0411146	.4053818
L2.	.3091415	.1060541	.001049	.3090028	.1014922	.5166988
dln_consump						
L1.	-.2203787	.1344117	.001314	-.2190475	-.479903	.0415251
L2.	.0221078	.1295494	.001295	.0226184	-.228624	.2798039
_cons	.0128598	.0034698	.000035	.0128702	.0060369	.0195489
Sigma_1_1	.0020092	.0003405	3.3e-06	.0019742	.0014548	.0027654
Sigma_2_1	.0000578	.0000625	6.2e-07	.0000563	-.0000618	.0001857
Sigma_3_1	.0001097	.0000518	5.2e-07	.0001073	.0000149	.0002205
Sigma_2_2	.0001322	.0000223	2.2e-07	.0001301	.0000954	.0001828
Sigma_3_2	.0000562	.0000143	1.4e-07	.000055	.0000316	.0000877
Sigma_3_3	.000087	.0000147	1.5e-07	.0000855	.0000629	.0001202

file **bvarex2.dta** saved.

We compute IRFs for the second model and save them as birf2 in the same dataset birfex1.

```
. bayesirf create birf2, step(8) set(birfex1)
(file birfex1.irf now active)
(file birfex1.irf updated)
```

Using the bayesirf ctable command, we show the posterior means of FEVDs of the impulse dln_inc on the response dln_consump along with estimates of posterior standard deviations.

```
. bayesirf ctable (birf1 dln_inc dln_consump fevd)
> (birf2 dln_inc dln_consump fevd), nocri stddev
```

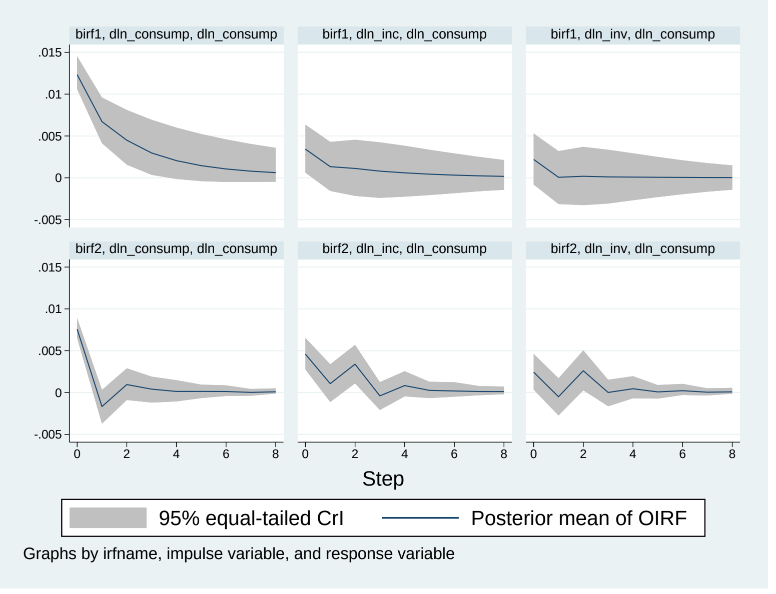
Step	(1) fevd	(1) Std. dev.	(2) fevd	(2) Std. dev.
0	0	0	0	0
1	.078122	.054559	.249063	.08115
2	.077138	.053865	.254958	.077739
3	.083944	.058845	.313267	.084101
4	.090341	.064417	.31425	.083694
5	.095177	.068994	.318057	.085284
6	.098524	.072337	.318697	.085481
7	.100779	.074699	.319035	.085732
8	.102291	.076363	.31923	.085885

```
Posterior means reported.
(1) irfname = birf1, impulse = dln_inc, and response = dln_consump.
(2) irfname = birf2, impulse = dln_inc, and response = dln_consump.
```

We notice that the FEVD estimates for the second model are much closer to those in the original example 1. In contrast, for the first model, the contribution of dln_inc to the variance of dln_consump is substantially lower, starting from 8% for step 1 and increasing only to 10% for step 8. The difference between the two models can be explained by the effect of using different priors for regression coefficients. The default conjugate Minnesota prior with the selftight() parameter of 0.1 shrinks the cross-variables lag coefficients to zero, thus reducing the corresponding FEVDs. For example, the posterior mean estimates of {dln_consump:L1.dln_inc} and {dln_consump:L2.dln_inc} are about 0.18 and 0.31 in the second model but only −0.04 and 0.03 in the first model.

Finally, let's examine the orthogonalized IRF (OIRF) response on `dln_consump` using the `bayesirf graph` command.

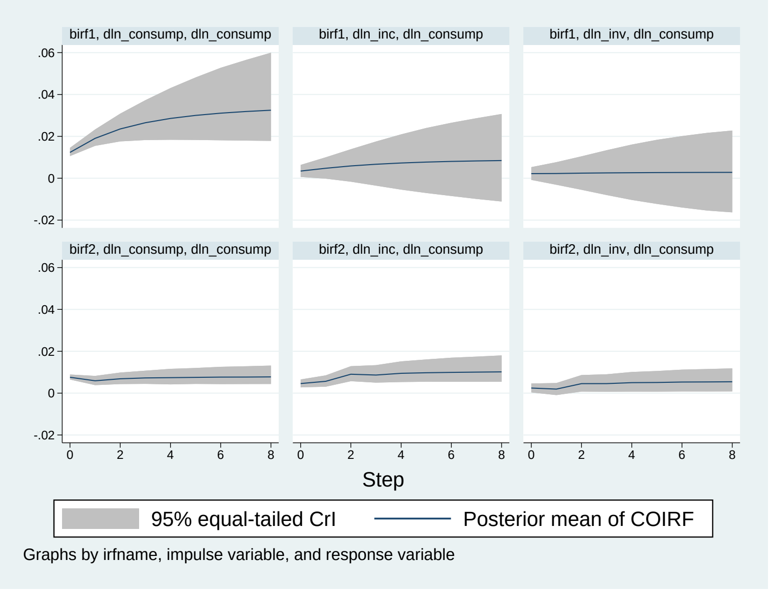
```
. bayesirf graph oirf, response(dln_consump)
```



The IRF graphs confirm the differences between the two models caused by the effect of the Minnesota prior on regression coefficients. For the first model, which has stronger priors, the impulse responses on `dln_consump` are smoother and have larger uncertainty, as evident by their credible bands. For the second model, the prior effect is minimal, and the graphs have ups and downs that may be due to some seasonal trends. There are no general rules for choosing the right amount of prior strength. The choice should be based on subject matter and prior experience. We also observe that all OIRFs converge to 0 relatively fast, as we expect from a stable VAR model.

The cumulative OIRFs show equilibrium convergence clearly:

```
. bayesirf graph coirf, response(dln_consump)
```



◁

Technical aspects of IRF files

`bayesirf create` computes posterior statistics of a series of IRFs and saves them in an IRF file. IRF files are just Stata datasets that have names ending in `.irf` instead of `.dta`. The dataset in the file has a nested panel structure.

Variable `irfname` contains the *irfname* specified by the user. Variable `impulse` records the name of the endogenous variable whose innovations are the impulse. Variable `response` records the name of the endogenous variable that is responding to the innovations. In a model with K endogenous variables, there are K^2 combinations of `impulse` and `response`. Variable `step` records the periods for which these estimates were computed.

Below is a catalog of the statistics that `bayesirf create` estimates after the `bayes: var` command and the variable names under which they are saved in the IRF file.

Posterior statistic	Name
Posterior mean of IRFs	`irf`
Posterior mean of OIRFs	`oirf`
Posterior mean of cumulative IRFs	`cirf`
Posterior mean of cumulative OIRFs	`coirf`
Posterior mean of dynamic-multiplier functions	`dm`
Posterior mean of cumulative dynamic-multiplier functions	`cdm`
Posterior mean of Cholesky forecast-error decomposition	`fevd`
Posterior standard deviation of the IRFs	`stdirf`
Posterior standard deviation of the OIRFs	`stdoirf`
Posterior standard deviation of the cumulative IRFs	`stdcirf`
Posterior standard deviation of the cumulative OIRFs	`stdcoirf`
Posterior standard deviation of dynamic-multiplier functions	`stddm`
Posterior standard deviation of cumulative dynamic-multiplier functions	`stdcdm`
Posterior standard deviation of the Cholesky forecast-error decomposition	`stdfevd`
Posterior median of the IRFs	`medirf`
Posterior median of the OIRFs	`medoirf`
Posterior median of the cumulative IRFs	`medcirf`
Posterior median of the cumulative OIRFs	`medcoirf`
Posterior median of dynamic-multiplier functions	`meddm`
Posterior median of cumulative dynamic-multiplier functions	`medcdm`
Posterior median of the Cholesky forecast-error decomposition	`medfevd`
Lower CrI of the IRFs	`irfl`
Lower CrI of the OIRFs	`oirfl`
Lower CrI of the cumulative IRFs	`cirfl`
Lower CrI of the cumulative OIRFs	`coirfl`
Lower CrI of dynamic-multiplier functions	`dml`
Lower CrI of cumulative dynamic-multiplier functions	`cdml`
Lower CrI of the Cholesky forecast-error decomposition	`fevdl`
Upper CrI of the IRFs	`irfu`
Upper CrI of the OIRFs	`oirfu`
Upper CrI of the cumulative IRFs	`cirfu`
Upper CrI of the cumulative OIRFs	`coirfu`
Upper CrI of dynamic-multiplier functions	`dmu`
Upper CrI of cumulative dynamic-multiplier functions	`cdmu`
Upper CrI of the Cholesky forecast-error decomposition	`fevdu`

In addition to the variables, information is stored in _dta characteristics. See *Technical aspects of IRF files* for the list of main characteristics. Below we list the characteristics that are specific to the bayes prefix models. For each *irfname* in _dta[irfnames], these are the additional characteristics:

Name	Contents
_dta[*irfname*_bayes]	it is bayes if *irfname* is created by bayesirf create
_dta[*irfname*_level]	level of the saved credible intervals
_dta[*irfname*_hpd]	it is hpd if HPD instead of equal-tailed CrIs are saved
_dta[*irfname*_mcmcfile]	MCMC file of simulated IRFs
_dta[*irfname*_mcmcsize]	MCMC sample size

Methods and formulas

Bayesian estimates of IRFs and other functions are obtained from their respective posterior distributions.

Let $\mathbf{\Phi_i} = (\phi_{jk,i})$ denote the impulse–response matrix after i periods; see *Methods and formulas* in [TS] **irf create** for its definition. Bayesian computation of IRFs involves estimation of the posterior distribution of each coefficient $\phi_{jk,i}$. Specifically, we recycle the MCMC sample created by the bayes: prefix command that contains draws from the posterior distribution of the model parameters such as regression coefficients and error covariance. For each draw, the IRF coefficients are computed according to the formulas in [TS] **irf create** and saved as MCMC samples, one for each coefficient. Finally, the resulting MCMC samples of IRF coefficients are summarized, and standard statistics such as posterior means, medians, and credible intervals are saved in the .irf file produced by bayesirf create.

Other functions are computed similarly; see *Methods and formulas* in [TS] **irf create** for their definitions.

Also see

[BAYES] **bayesirf** — Bayesian IRFs, dynamic-multiplier functions, and FEVDs

[TS] **irf** — Create and analyze IRFs, dynamic-multiplier functions, and FEVDs

[BAYES] **bayes: dsge** — Bayesian linear dynamic stochastic general equilibrium models

[BAYES] **bayes: dsgenl** — Bayesian nonlinear dynamic stochastic general equilibrium models

[BAYES] **bayes: var** — Bayesian vector autoregressive models