Introduction to Bayesian Analysis in Stata

Isabel Cañette
Principal Mathematician and Statistician
StataCorp LLC

Webinar
Nov 3, 2022

Introduction

The main concept.
In classical statistical analysis, we assume fixed unknown parameters, a dataset generated with a distribution based on them, and we use the data to construct an estimate of those underlying parameters.

In Bayesian statistic, parameters are considered random, according to a distribution, and our aim is to use previous knowledge of this distribution to estimate an updated version of it conditional on the observed data.

Stata commands

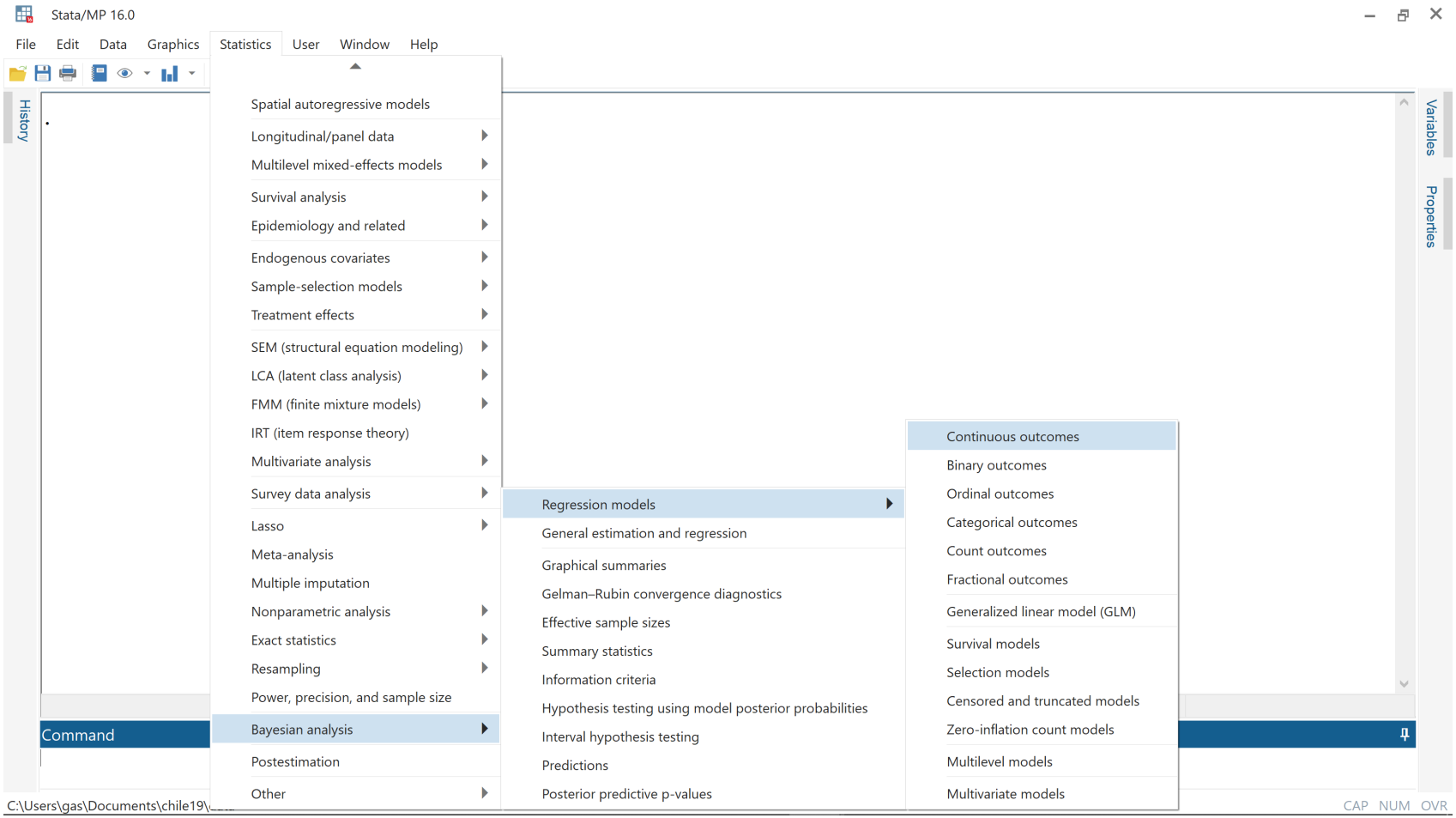
Stata commands for Bayesian estimation

- `bayes:` prefix provides a simple way to fit bayesian regression models. For example:

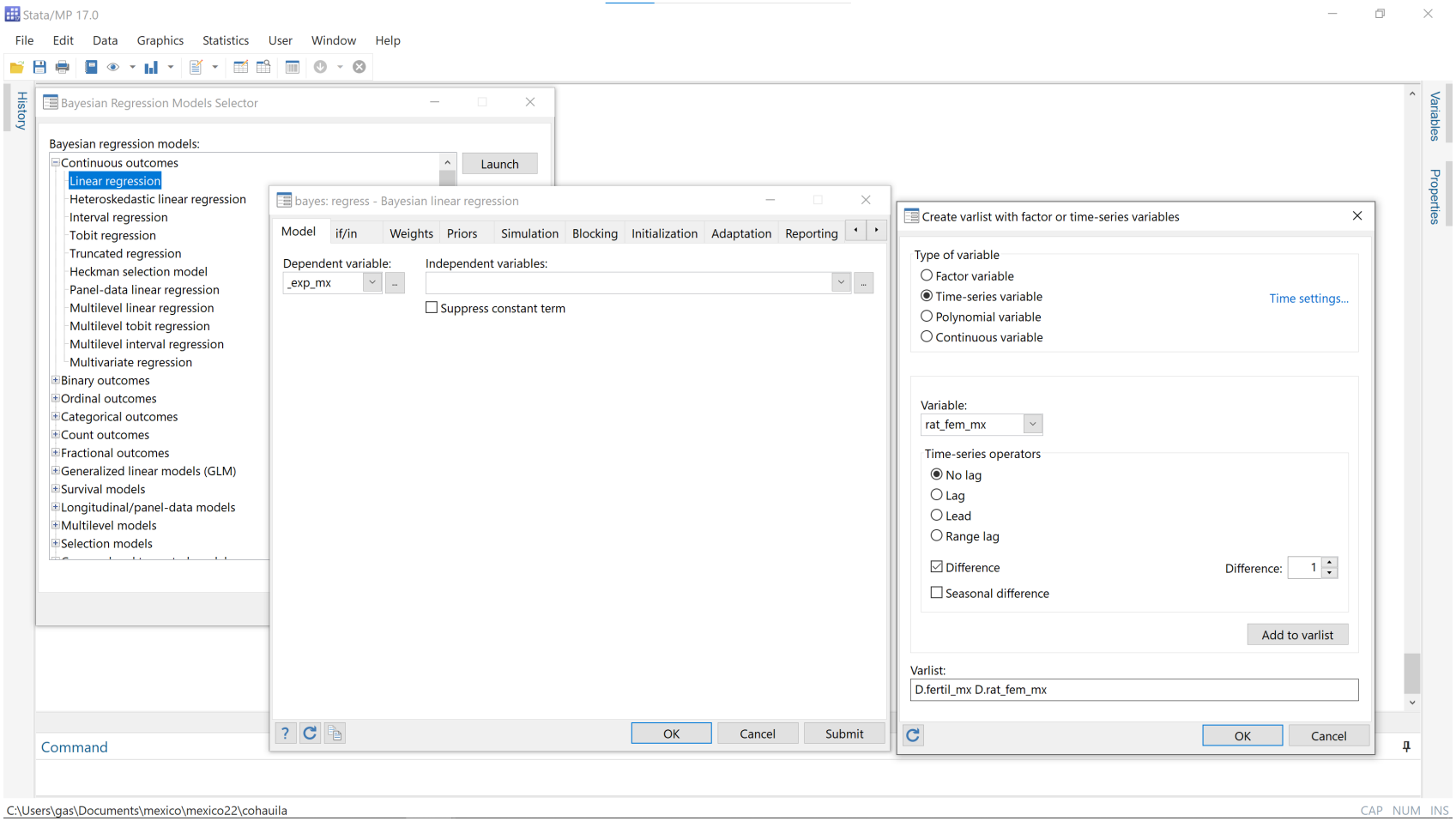
  ```
  . bayes: regress y x1 x2
  ```

  It supports a wide range of commands including regressions for continous, binary, ordinal, categorical, count or fractional outcomes, survival analysis, sample selection, panel data, multilevel, time series, and dynamic stochastic general equilibrium models. Type `help bayes estimation` to see the complete list.
- `bayesmh` allows us to fit customized Bayesian regressions by choosing among a set of available prior and likelihood functions, or with evaluators provided by the user. It can be used for linear and/or non-linear, one-level or multilevel, and one or multiple-equations models.

Stata Graphical User Interface



Stata Graphical User Interface



Stata's Bayesian suite consists of the following commands

Command	*Description*
Estimation	
`bayes:`	Bayesian regression models using the `bayes` prefix
`bayesmh`	General Bayesian models using MH
`bayesmh` *evaluators*	User-defined Bayesian models using MH
Postestimation	
`bayesgraph`	Graphical convergence diagnostics
`bayesstats ess`	Effective sample sizes and more
`bayesstats grubin`	Gelman–Rubin convergence diagnostics
`bayesstats summary`	Summary statistics
`bayesstats ic`	Information criteria and Bayes factors
`bayestest model`	Model posterior probabilities
`bayestest interval`	Interval hypothesis testing
`bayespredict`	Bayesian predictions (available only after `bayesmh`)
`bayesstats ppvalues`	Bayesian predictive *p*-values (available only after `bayesmh`)

Estimation

Bayes' Theorem: $$p(\theta|y) = \frac{f(y|\theta)\pi(\theta)}{m(y)}$$

- Assume that we know $\pi(\theta)$ ("prior")
- We have already assumed that we know $f(Y|\theta) = L(y;\theta)$
- We observe the data, Y

Bayes' theorem tell us that we can obtain the "posterior" distribution of the parameter, $p(\theta|y)$

$$p(\theta|y) \propto L(y;\theta) \times \pi(\theta)$$

In theory, we don't need the constant because densities integrate to 1. In practice, we won't need the constant to simulate a sample for $p(\theta|y)$.

Example 1: Weight of sugar packets

Example: weight of sugar packets.

Let's assume we have a random sample $y_1, \ldots y_{70} \sim N(\mu, \sigma^2)$ and we are interested in estimating the mean, μ. This can be estimated as the constant of a regression without covariates.

```
. use sugar, clear
(Weights of Domino sugar packets, Triola, Elementary Statistics.)
. regress weight , noheader
```

weight	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
_cons	3.586043	.0088481	405.29	0.000	3.568391	3.603694

The Bayesian version would be:

```
. bayes, rseed(3876): regress weight
```

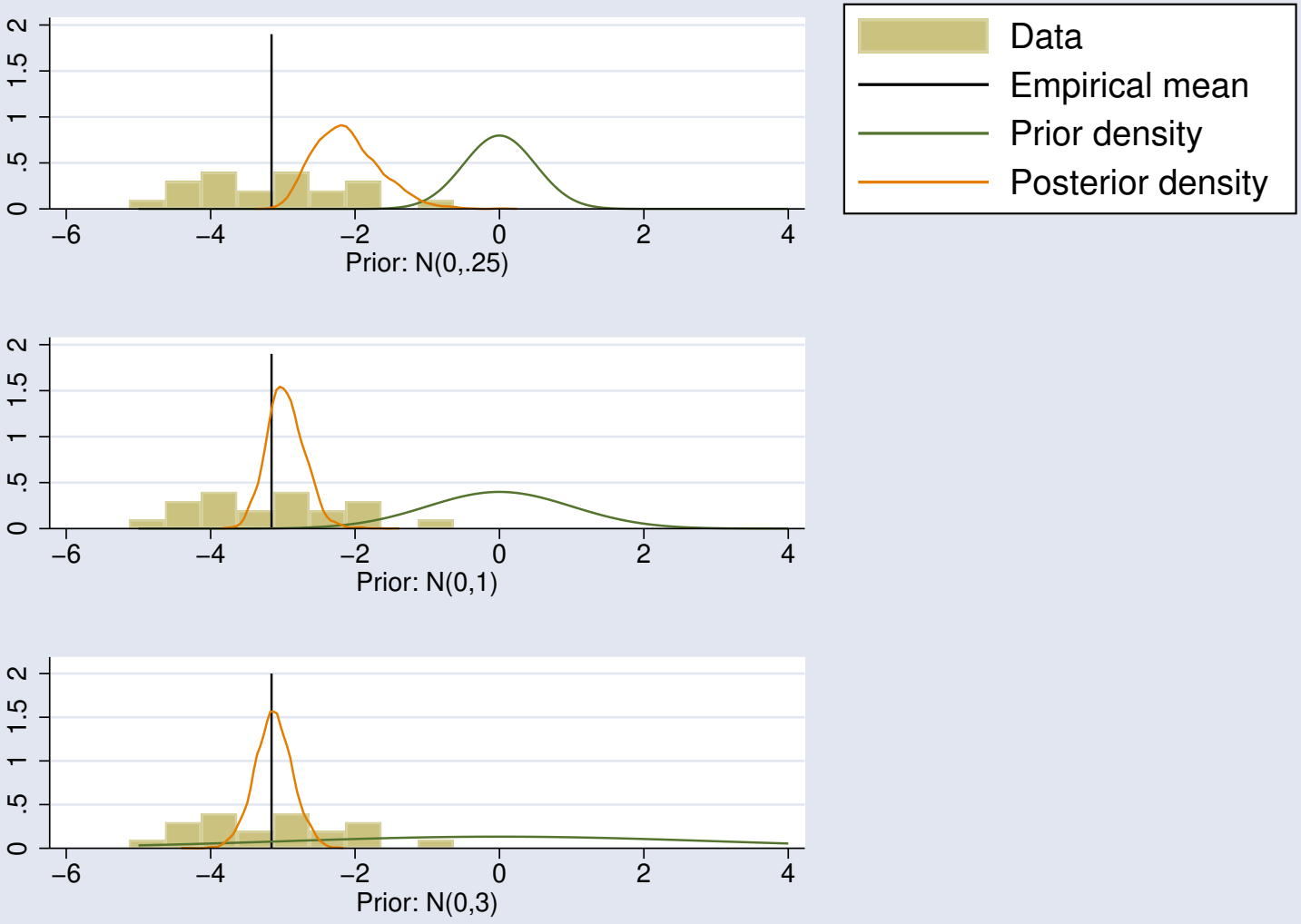
Bayesian estimation is performed via simulations; we set a seed for reproducibility.

Model summary:

```
. bayes, rseed(3876): regress weight ,vsquish notable

Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  weight ~ regress({weight:_cons},{sigma2})
Priors:
  {weight:_cons} ~ normal(0,10000)
        {sigma2} ~ igamma(.01,.01)
------------------------------------------------------------------------------

Bayesian linear regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =         70
                                                Acceptance rate  =      .4382
                                                Efficiency:  min =      .1988
                                                             avg =      .2231
Log marginal-likelihood =  66.950733            max =      .2475
```

Why are we using this prior by default?

The less “informative” the prior, the more we rely on the data.



The table

```
. bayes, rseed(3876) : regress weight ,vsquish noheader

Burn-in ...
Simulation ...
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
weight						
_cons	3.586146	.009181	.000185	3.586458	3.567973	3.604502
sigma2	.0059266	.0010415	.000023	.0057973	.0042246	.0083358

- Mean, median and std. dev. are estimates of the mean, the median and the standard deviation of the posterior distribution.
- A 95% credibility interval is interpreted as an interval such us the probability of the parameter being there is 0.95.

Implementation: Monte Carlo Markov Chains

How is this density estimated? Because there is, in most cases, not a closed form for the posterior distribution, this is estimated via simulation (i.e., generating a large random sample of this distribution rather than having a functional form).
We use MCMC, i.e. create an ergodic Markov Chain whose limit (stationary) distribution is theoretically proven to be the posterior we are looking for.

Stata implements two methods: Gibbs sampling and Metropolis-Hastings algorithm.

Metropolis-Hasting algorithm.
We choose a "proposal" distribution $q(.)$ (unrelated with our prior or our posterior, we actually use a Gaussian distribution) and start with θ_0 in the domain of the posterior p. Then, for each iteration t:

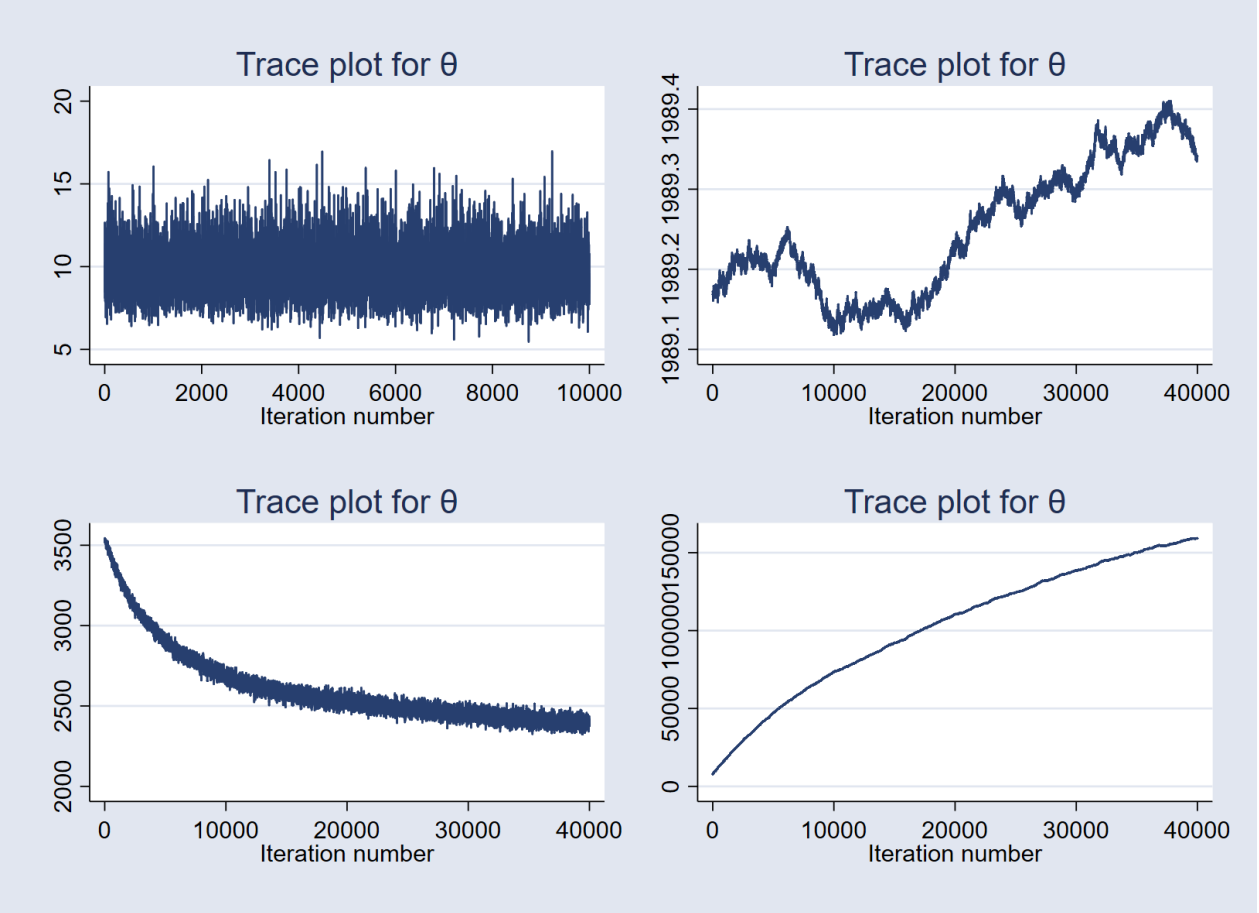
- Generate a proposal state $\theta_* \sim q(.|\theta)$
- Compute the acceptance probability

$$r(\theta_*|\theta_{t-1}) = \frac{p(\theta_*|y)}{p(\theta_{t-1}|y)}$$

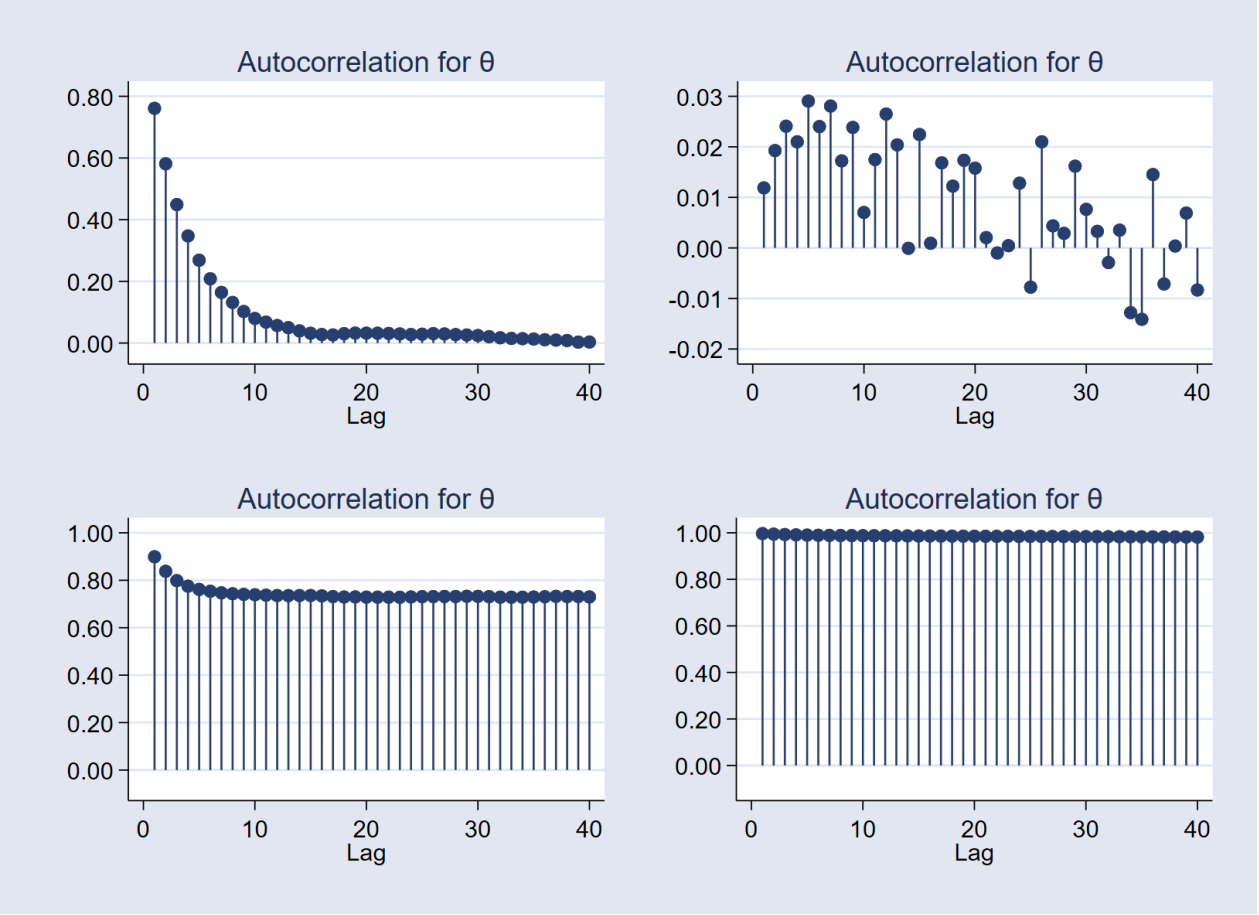
- We accept θ_* with probability $r(\theta_*|\theta_{t-1})$ (or with probability 1 if $r(\theta_*|\theta_{t-1}) > 1$).
- Accepting means $\theta_t = \theta_*$; otherwise $\theta_t = \theta_{t-1}$

The concept of convergence

Converge is achieved if the simulated values reach stationarity. Trace plots show the trajectory of those values.



Autocorrelation plots: we expect the correlation to be negligible after a few lags. High autocorrelations imply low efficiency, so reaching stationarity will take more iterations than for more efficient problems.



Example 2: Bike rentals vs weather

```
. use bikes, clear
(Bike sharing dataset, Hadi Fanaee-T)
. describe
Contains data from bikes.dta
 Observations:           731                  Bike sharing dataset, H. Fanaee-T
    Variables:             4                  4 Sep 2022 16:08
```

Variable name	Storage type	Display format	Value label	Variable label
precip	byte	%15.0g	preclab	Precipitation
ntemp	float	%9.0g		Normalized Temperature (Celsius)
count100	float	%9.0g		Hundreds of bikes rented
temp	float	%9.0g		Temperature (Celsius)

We fit a Bayesian linear model to the rental counts ($\times 0.01$) vs temperature and indicators of levels of precipitations (we set a seed for reproducibility).

```
. bayes, rseed(1357): regress count100 temp i.precip
```

```
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
  count100 ~ regress(xb_count100,{sigma2})
Priors:
  {count100:temp i.precip _cons} ~ normal(0,10000)                        (1)
                       {sigma2} ~ igamma(.01,.01)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_count100.
Bayesian linear regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =        731
                                                Acceptance rate  =      .3475
                                                Efficiency:  min =       .051
                                                             avg =     .09622
Log marginal-likelihood = -3008.9227                         max =      .2236
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
count100						
temp	1.347828	.0626141	.002614	1.346385	1.233055	1.469331
precip						
Mist	-5.802201	1.160169	.047753	-5.785858	-8.105655	-3.573989
Light rain/snow	-25.8168	3.281888	.109326	-25.84465	-32.31738	-19.29535
_cons	27.13917	1.198321	.053062	27.12183	24.85218	29.45991
sigma2	206.305	10.80463	.228511	206.0586	185.9393	228.7717

Note: Default priors are used for model parameters.

The header is:

```
. bayes, rseed(1357) nomodelsummary:regress count100 temp i.precip, vsquish

Burn-in ...
Simulation ...
Bayesian linear regression                      MCMC iterations  =     12,500
Random-walk Metropolis-Hastings sampling        Burn-in          =      2,500
                                                MCMC sample size =     10,000
                                                Number of obs    =        731
                                                Acceptance rate  =      .3758
                                                Efficiency:  min =     .02825
                                                             avg =     .07818
Log marginal-likelihood = -3013.5765                         max =      .2101
------------------------------------------------------------------------------
```

- Marginal log-likelihood $m(y) = p(Y = y|(\theta \sim \pi))$ $= \int (p(y|\theta, \pi)p(\theta|\pi)\, d\theta$. (i.e., integrate $p(y|\theta)$ over the distribution π of θ), evaluated at the observed data y.
- MCMC iterations - total number of iterations
- Burn-in - discarded iteration to eliminate influence of the initial values
- MCMC sample size - iterations used for estimation
- Acceptance rate - fraction of proposal values accepted. We expected it to be neither too small nor too large - optimal value for multivariate posteriors and proposal: 0.234; for univariate posteriors:0.45
- Efficiency - Indicator of the mixing quality of the chain

The table:

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
count100						
temp	.0135365	.0006187	.000034	.0135331	.0122963	.01472
precip						
Mist	-5.747835	1.143186	.042002	-5.774701	-7.882743	-3.509883
Light rain/snow	-25.65604	3.173647	.149708	-25.59269	-31.96759	-19.44203
_cons	27.00479	1.169644	.069591	27.02894	24.73269	29.2549
sigma2	206.1646	10.93242	.238485	205.9812	185.4657	228.1383

Note: Default priors are used for model parameters.

- Mean ($\hat{\theta}$), median and standard deviation ($\hat{s}$) are the mean, the median and the standard deviation of the posterior sample. Estimate respectively the mean ($E(\theta_t)$), the median and the standard deviation $\sqrt{Var(\theta_t)}$ of the posterior distribution.
- A 95% credibility interval - an interval such us the probability of the parameter being there is 0.95.
- MCSE - Monte Carlo standard error - an indicator of the precision of the sample posterior mean ("Mean" in the table). MCSE($\hat{\theta}$)=$\hat{s}/\sqrt{\text{ESS}}$

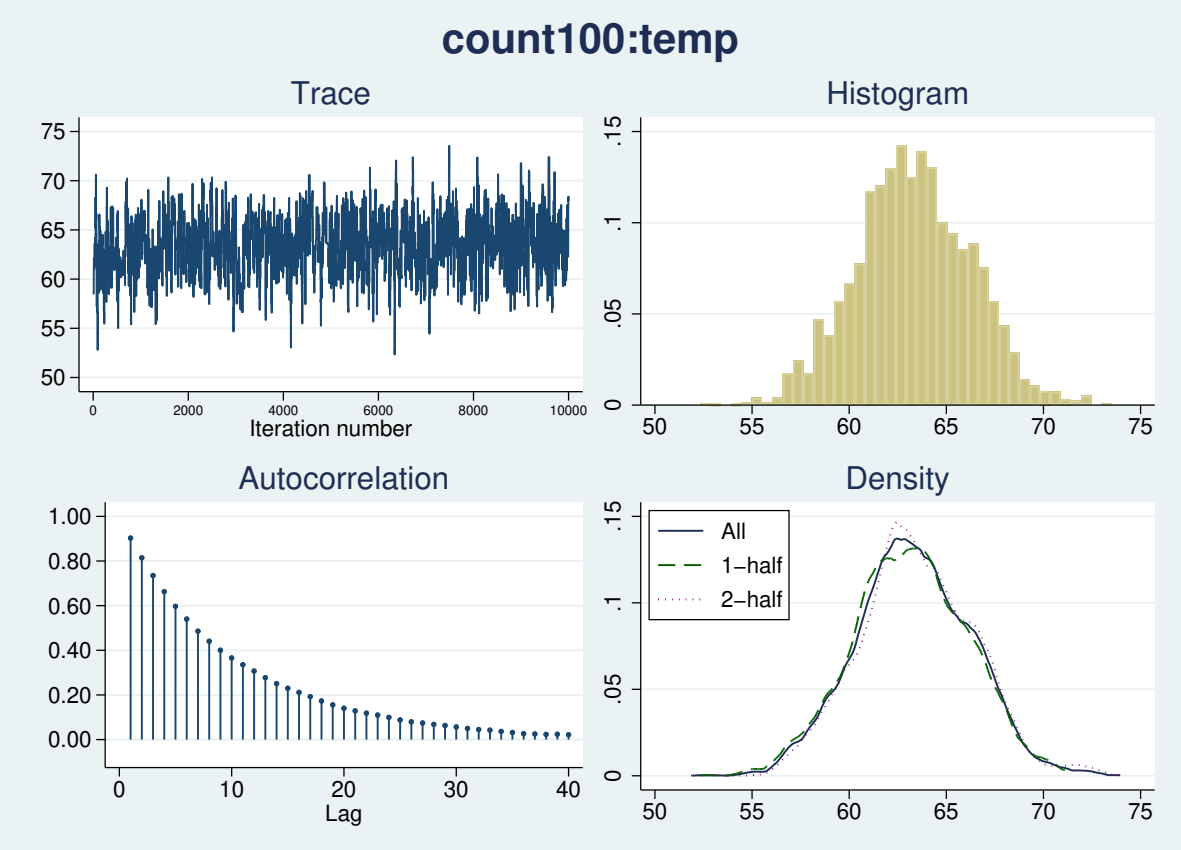
Convergence and Diagnostics

Convergence is attained when the chain achieves stationarity (and therefore the sample is drawn from the posterior distribution).

- Inspecting mixing and time trends within the chains of individual parameters
  - `bayesgraph diagnostics, trace, ac, histogram, kdensity`
  - `bayesgraph csum`
  - `bayesstats ess`
- Inspecting multiple chains for each parameter
  - `bayesgraph diagnostics, trace, ac, histogram, kdensity`
  - `bayesstats grubin`

Diagnostics need to be performed for each parameter. In this example, we will do it only for the coefficient for variable `temp`.

Visual assessment of convergence: `bayesgraph diagnostics`.



- The trace doesn't show convergence problems
- Correlation becomes negligible after 20 lags
- Density estimates with first and second half look similar

Efficiency summaries: bayesstats ess

```
. bayesstats ess
Efficiency summaries                MCMC sample size =     10,000
                                    Efficiency:  min =     .02825
                                                 avg =     .07818
                                                 max =      .2101
```

	ESS	Corr. time	Efficiency
count100			
temp	334.69	29.88	0.0335
precip			
Mist	740.79	13.50	0.0741
Light rain/snow	449.39	22.25	0.0449
_cons	282.49	35.40	0.0282
sigma2	2101.42	4.76	0.2101

- ESS -Effective sample size - Number of i.i.d observations that would contain the same information as in our MCMC sample.
- Corr time - T/ESS - Number of iterations where autocorrelation becomes negligible (T=MCMC sample size).
- Efficiency - ESS/T - Indicator of the the mixing quality of the MCMC procedure. The higher the better.
  - Efficiencies over 10% are considered good for MH.
  - Efficiencies under 1% would be a source of concern.

See Methods and Formulas section in manual entry for [BAYES] `bayesstats ess` for details.

Multiple chains

Simulating multiple chains allows us to further assess convergence; in addition, it improves precision (lowers MCMC errors).

- Reported results are based on all the chains
- Literature recommends 4 chains
- It allows us to compare chains with different starting values

We fit our regression model with three chains.

```
. bayes, rseed(1357) nchains(3): regress count100 temp i.precip
```

The output header:

```
Chain 1
  Burn-in ...
  Simulation ...
  (output omitted)
Model summary
------------------------------------------------------------------------------
  (output omitted)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_count100.
Bayesian linear regression                      Number of chains    =          3
Random-walk Metropolis-Hastings sampling        Per MCMC chain:
                                                     Iterations     =     12,500
                                                     Burn-in        =      2,500
                                                     Sample size    =     10,000
                                                Number of obs       =        731
                                                Avg acceptance rate =      .3413
                                                Avg efficiency: min =      .04364
                                                                avg =      .09016
                                                                max =       .2174
Avg log marginal-likelihood = -3008.9373        Max Gelman-Rubin Rc =      1.004
```

- Summary statistics (acceptance rate, efficiency) are averaged over all the chains (use `(chainsdetail)` to see them separately for each chain)
- With multiple chains, the maximum Gelman-Rubin convergence statistic (R_c) is reported.
  - It is based on the between-chains variance *B* and the within-chains variance *W* (See Methods and Formulas for `bayesstats grubin`)
  - It is required that *Rc* be less than 1.1 to declare convergence.
  - In any case, it important to also visually assess convergence.

The output table:

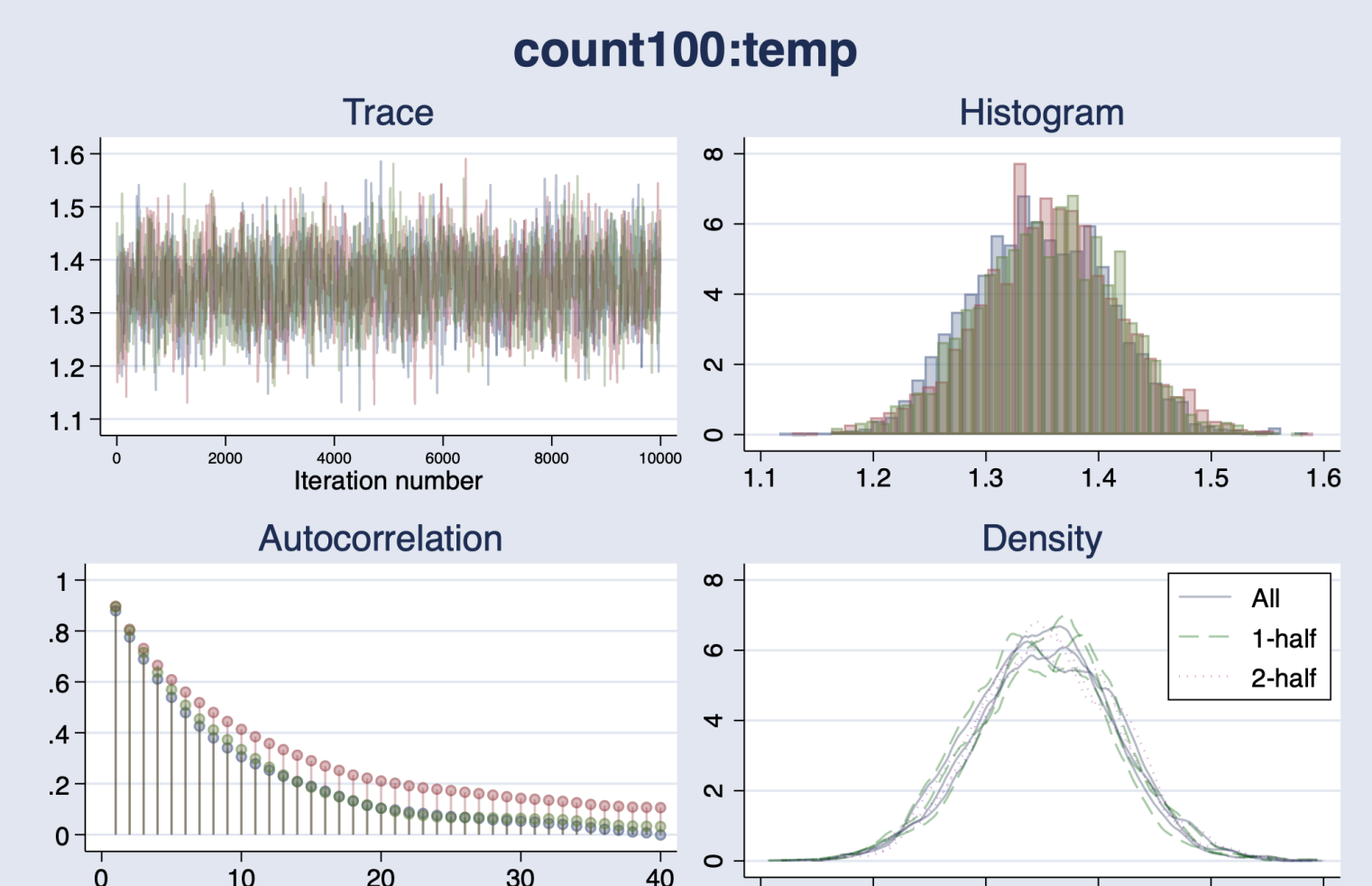
```
Bayesian linear regression                      Number of chains    =          3
Random-walk Metropolis-Hastings sampling        Per MCMC chain:
                                                     Iterations     =     12,500
                                                     Burn-in        =      2,500
                                                     Sample size    =     10,000
                                                Number of obs       =        731
                                                Avg acceptance rate =      .3413
                                                Avg efficiency: min =      .04364
                                                                avg =      .09016
                                                                max =       .2174
Avg log marginal-likelihood = -3008.9373        Max Gelman-Rubin Rc =      1.004
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
count100						
temp	1.353114	.0630514	.001667	1.353777	1.230509	1.475987
precip						
Mist	-5.779365	1.152578	.025497	-5.753522	-8.116835	-3.535928
Light rai..	-25.82078	3.229853	.068592	-25.84465	-32.03808	-19.4659
_cons	27.04732	1.231426	.034035	27.06031	24.71367	29.45991
sigma2	206.1531	10.92119	.13522	205.834	185.7931	228.9021

```
Note: Default priors are used for model parameters.
Note: Default initial values are used for multiple chains.
```

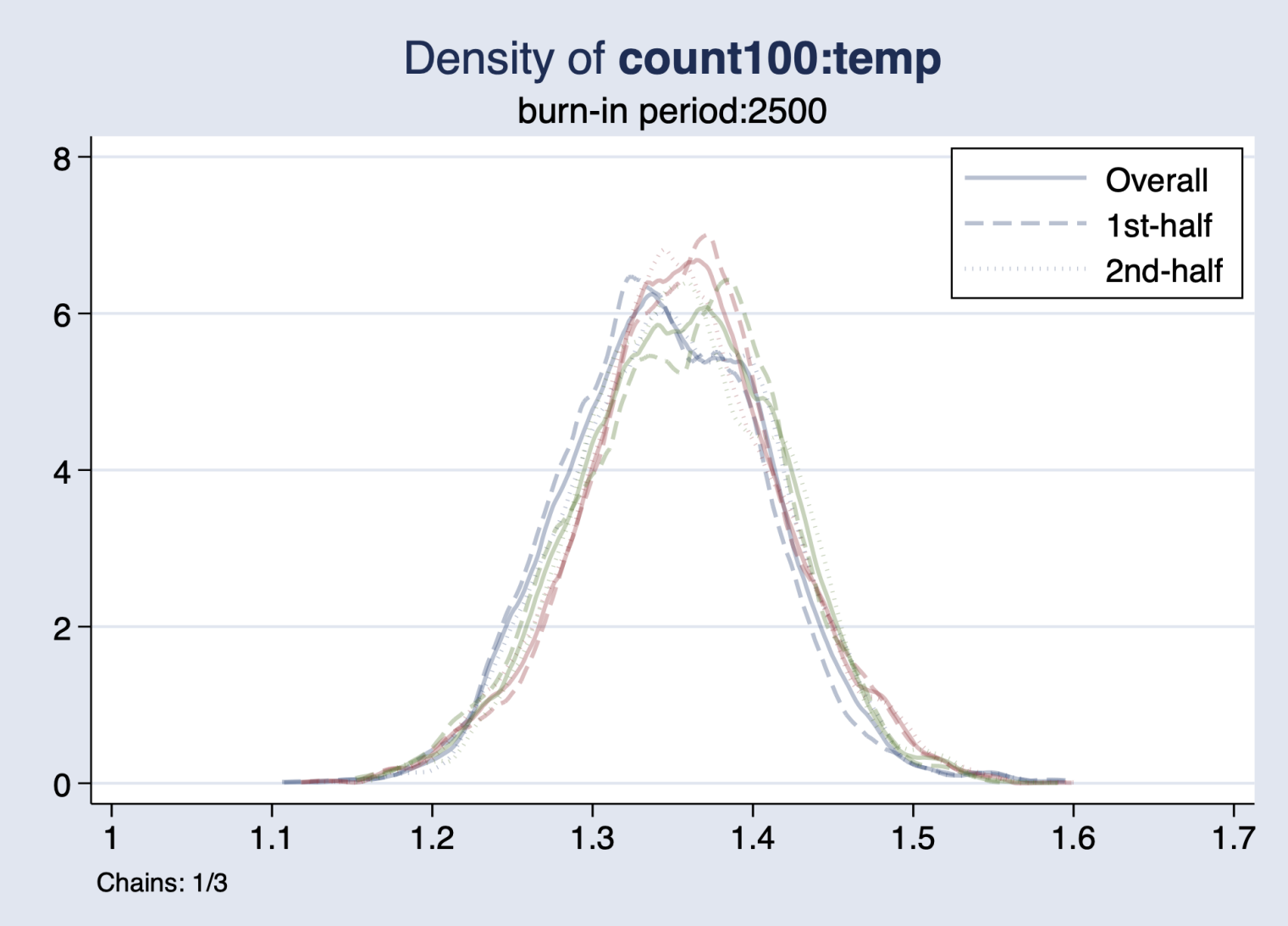
Graphic diagnostics with multiple chains

bayesgraph diagnostics allows us to compare the behavior of the different chains.



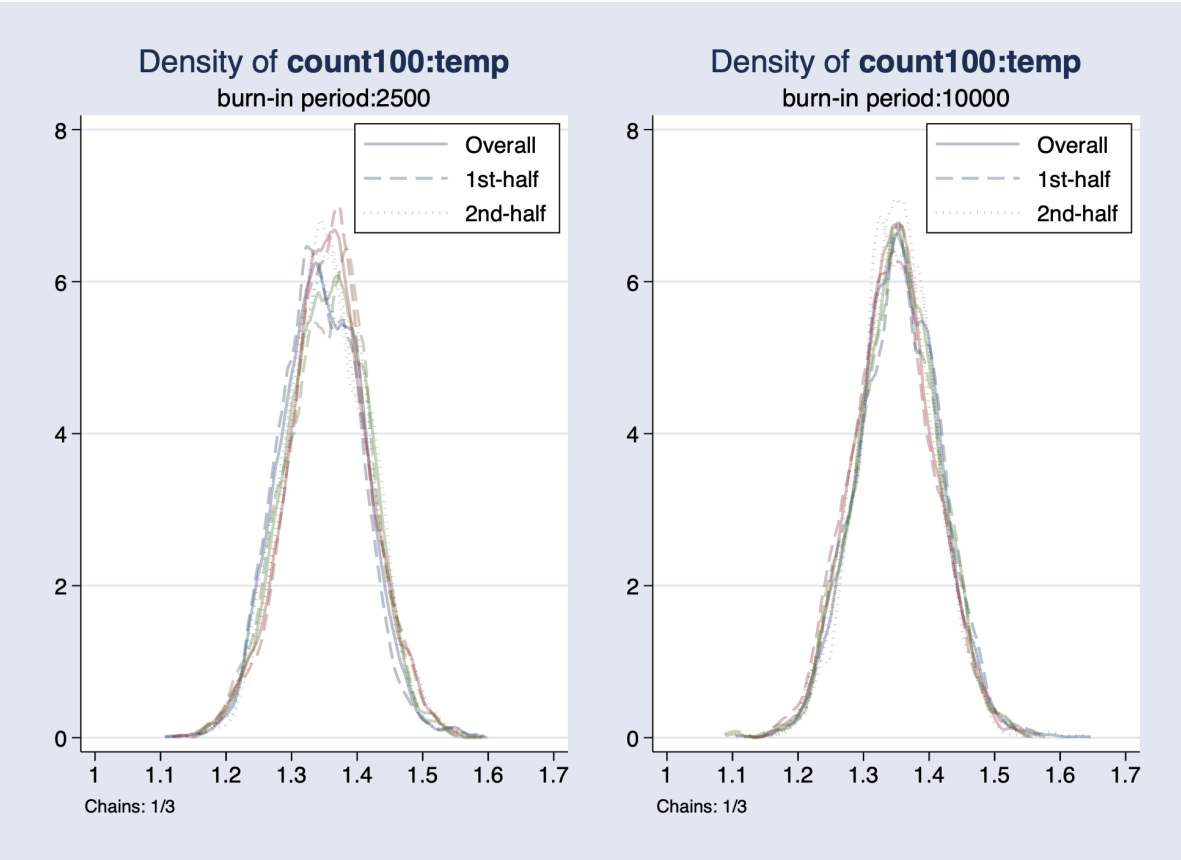
It looks like there is room for improvement in the density plot.

```
bayesgraph kdensity count100:temp, show(both)
```



We increase the burn-in period from 25000 to 10000:

```
bayes, rseed(1357) nchains(3) burnin(10000): regress count100 temp
i.precip
```



Increasing the burn-in period improved our graphs.

Performing tests on the parameters

`bayestest interval` allows us to test the null hypothesis that a parameter is in a certain interval

```
. bayestest interval {count100:temp}, lower(1.3) upper(.)
Interval tests      MCMC sample size =     10,000
       prob1 : {count100:temp} > 1.3
```

	Mean	Std. dev.	MCSE
prob1	.7757	0.41714	.0156717

The (posterior mean estimate of the) probability of the coefficient for `temp` being larger than 3 is 0.78 (It is estimated as the proportion of simulated values for this coefficient in our chain that are larger than 3)

Comparing models

We can use `bayesstats ic` or `bayestest model` to compare different models. The only requirement is that they are fitted on the same data.

```
. bayes, rseed(1357) saving(reg, replace): regress count temp i.precip
  (output omitted)
. est store reg
. bayes, rseed(1357) saving(poi, replace): poisson count temp i.precip
  (output omitted)
. bayesstats ic reg poi
Bayesian information criteria
```

	DIC	log(ML)	log(BF)
reg	5973.529	-3008.923	.
poi	7871.914	-3970.533	-961.6104

```
Note: Marginal likelihood (ML) is computed
      using Laplace-Metropolis approximation.
```

The smaller the DIC and larger the log-likelihood, the better.

`bayestest model` allows us to compare posterior probabilities for different models.

```
. bayestest model reg poi

Bayesian model tests
```

	log(ML)	P(M)	P(M|y)
reg	-3.01e+03	0.5000	1.0000
poi	-3.97e+03	0.5000	0.0000

```
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

In this case, P(M) is 0.5 for the two cases, because we are assuming that both models are equally probable a priori. This can be modified with `prior()` option. Under this assumption, the Gaussian model (`bayes:regress`) and the Poisson model have respectively probabilities 1 and 0 given the data.

Bayesian predictions

After fitting the model with `bayesmh`, we can simulate replications of the dependent variable.

- This allows us to assess the model, by comparing the simulated distribution of the dependent variable with the observed.
- Computing predictions for observations with missing values on the dependent variable allows us to forecast the response, given our estimated model and the covariate patterns.
- The mechanism is the same in the two cases, so we will start by showing the behavior of `bayespredict` in general, and then we will apply it to the two situations.

`bayespredict` is also supported by panel data models like `bayes:xtreg`.

Bayesian predictions: `bayespredict`

We started with a prior distribution, $\pi(\theta)$, and updated that prior with the information in our dataset, y, obtaining the posterior distribution, $p(\theta)$.

Now we can consider that the data we have already observed are fixed, and the actual distribution of θ is our posterior p.

Under this assumption, we can predict the distribution of future outcomes, y^{new}.

Assuming that $\theta \sim p$, and we can use this (posterior) distribution and the likelihood ($f(y|\theta)$) to compute the predictive posterior distribution for a new value y^{new} of Y:

$$p(y^{new}) = \int f(y|\theta)p(\theta)\, d\theta.$$

We can see it as:

$$p(y^{new}|y^{obs}) = \int f(y|\theta)p(\theta|y^{obs})\, d\theta.$$

To obtain predictions, first we fit our model with `bayesmh` and save the chains in a new file.

```
. bayesmh count100 temp i.precip, likelihood(normal({sigma2}))  ///
 prior({count100:}, normal(0, 10000)) ///
 prior({sigma2}, igamma(.01, .01)) rseed(2476) ///
 saving(bikespost, replace) vsquish
```

We saved the simulated values for the posterior distribution of the parameters in a new file (`bikepost`). This file will be needed to perform predictions.

(*output omitted*)

Bayesian normal regression
Random-walk Metropolis-Hastings sampling

MCMC iterations	= 12,500
Burn-in	= 2,500
MCMC sample size	= 10,000
Number of obs	= 731
Acceptance rate	= .1968
Efficiency: min	= .02171
avg	= .03907
max	= .05898

Log marginal-likelihood = -3009.1647

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
count100						
temp	1.357601	.0627935	.002586	1.358048	1.237237	1.480288
precip						
Mist	-5.637735	1.121772	.058148	-5.604706	-7.831796	-3.528415
Light rain/snow	-25.61719	3.111537	.156311	-25.54275	-31.87835	-19.8676
_cons	26.90482	1.178267	.060584	26.87755	24.60462	29.20782
sigma2	204.306	10.741	.72906	203.9492	184.8968	226.2071

file **bikespost.dta** saved.

. est store bmh_bikes

`bayespredict` creates simulated outcomes for each observation.

```
. bayespredict {_ysim}, rseed(1357) saving(ypred0, replace)
Computing predictions ...
file ypred0.dta saved.
file ypred0.ster saved.
```

The dataset created by `bayespredict` contains one variable per observation in the original dataset, and one observation per replication in the estimation chain. (also, the posterior means ($\_\mu_1, \_\mu_2, \ldots$) based on the chains created in the estimation and the data).

```
. use ypred0, clear
. list _ysim1_1-_ysim1_5  _fr in 1/10, noobs
```

_ysim1_1	_ysim1_2	_ysim1_3	_ysim1_4	_ysim1_5	_frequ~y
34.949	35.325	20.281	21.341	7.3716	1
36.916	67.079	34.021	24.457	21.03	1
34.022	48.495	11.672	20.878	50.914	1
56.217	13.915	31.911	54.263	11.735	1
13.649	41.505	39.603	56.341	29.562	1
52.75	34.509	25.022	39.288	25.493	1
16.08	25.435	63.214	8.9858	56.277	1
47.346	47.817	49.882	31.385	42.355	1
49.107	52.311	20.986	33.556	32.482	1
16.692	16.116	34.526	36.091	51.351	1

Using replications to visually assess the model fit

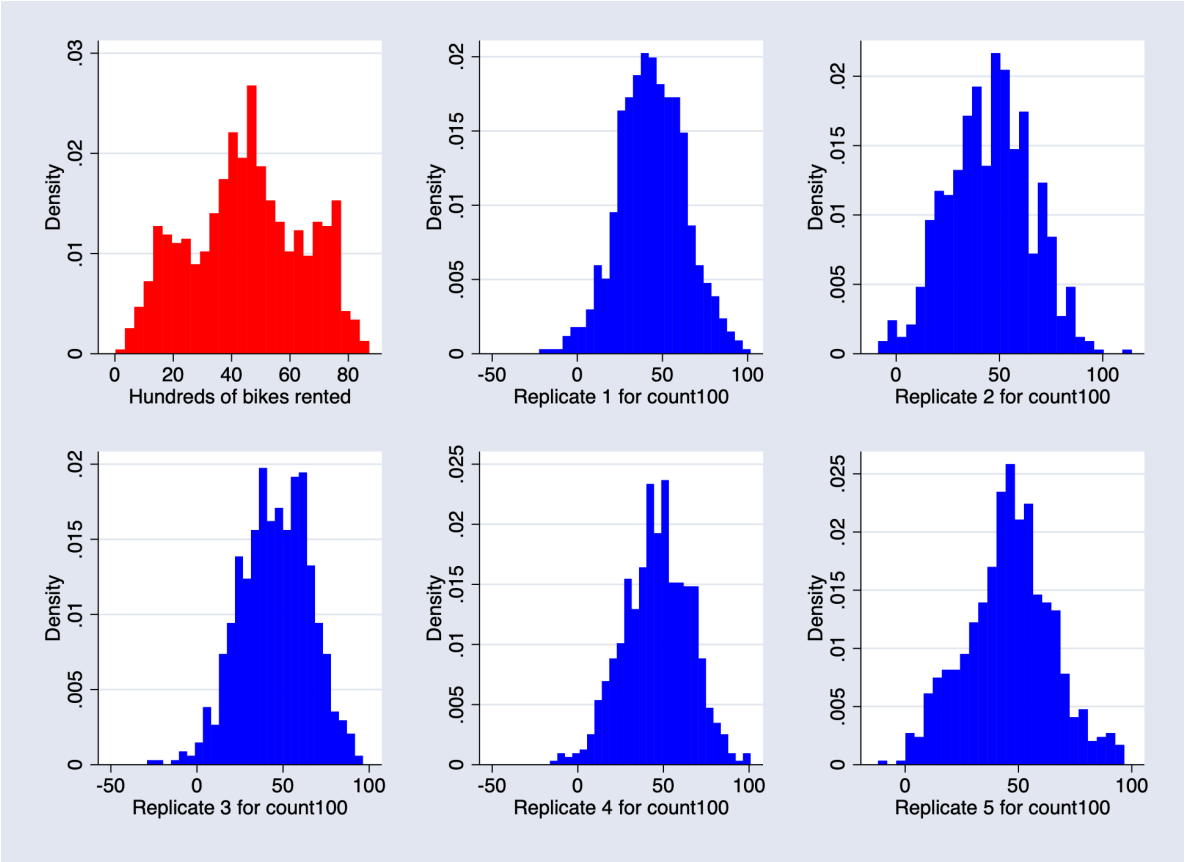
We can use `bayesrep` to check how well our assumed model captures the distribution of the dependent variable. We create 5 replicates of the dependent variable `count100`, and compare the distribution of those replications with the distribution of `count100`.

```
. use bikes, clear
(Bike sharing dataset, Hadi Fanaee-T)

. estimates restore bmh_bikes
(results bmh_bikes are active now)

. bayesreps c100rep*, nreps(5) rseed(9451)

Computing predictions ...
```

Instead of creating a new file, `bayesrep` adds the replications to our dataset.

We draw histograms to compare distribution of the observed dependent variable with the replications.

```
. quietly histogram count100, name(hist0, replace) nodraw color(red)
. local histlist hist0
. forvalues i = 1/5 {
  2.  quietly histogram c100rep`i´, name(hist`i´, replace) nodraw color(blue)
  3.  local histlist `histlist´ hist`i´
  4.  }
. graph combine `histlist´
```



The distribution of `count100` looks different from those of the replications. We might consider a different specification, either changing the covariate patter or the assumed distribution for the residuals.

-bayespredict- allows us to compute a statistic (such as the mean) for the actual sample and the replicated samples, and compare them.

```
. bayespredict (mean:@mean({_ysim})) (min:@min({_ysim})) ///
>  (max:@max({_ysim})), saving(count100_stats, replace) rseed(1359)
Computing predictions ...
file count100_stats.dta saved.
file count100_stats.ster saved.
.
. bayesstats ppvalues {mean} {min} {max} using count100_stats
Posterior predictive summary   MCMC sample size =    10,000
```

T	Mean	Std. dev.	E(T_obs)	P(T>=T_obs)
mean	45.00876	.740575	45.04349	.4797
min	-16.38561	7.217064	.22	.0005
max	102.0574	6.034314	87.14	.9993

Note: P(T>=T_obs) close to 0 or 1 indicates lack of fit.

Out of sample predictions: `bayespredict`

Now, let's assume that the weather forecast for tomorrow is no precipitacions (`precip=1`) and a temperature of 20 ºCelsius (`temp=20`); given this weather, how do we predict the number of bikes to be rented?

```
. use bikes, clear
(Bike sharing dataset, Hadi Fanaee-T)
. estimates restore bmh_bikes
(results bmh_bikes are active now)
. local N1 = _N + 1
. set obs `N1´
Number of observations (_N) was 731, now 732.
. replace precip = 1 in `N1´
(1 real change made)
. replace temp = 20 in `N1´
(1 real change made)
. global N1 = `N1´
. * _ysim represents the outcome
. bayespredict {_ysim} if _n == `N1´, rseed(1357) saving(ypred, replace)
Computing predictions ...
file ypred.dta saved.
file ypred.ster saved.
```

We can use `bayesstats summary` to display posterior summaries for the prediction.

```
. bayesstats summary _ysim_$N1 using ypred
Posterior summary statistics                     MCMC sample size =    10,000
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
_ysim1_732	54.20513	14.37852	.143785	54.23792	26.1864	82.50352

There is 95% probability of renting between 2618 and 8250 bikes.

Final remarks:

- Bayesian analysis can be used to answer questions about unknown parameters in terms of probability statements, using prior information on such probability.
- Stata provides a suite of commands for Bayesian estimation, diagnostics, visualization and prediction. Today we have just described a few of them. Please see the [Bayes] manual for a complete reference.