Package: Bayesplot

SongZhai

Description of 'bayesplot'

- Plotting functions for posterior analysis, model checking, and MCMC diagnostics.

Packages I will use for this tutorial

```
library("bayesplot")
library("ggplot2")
# install.packages("retanarm")
library("rstanarm")
library("mcmcse")
```

0. Outline

1. Simulate the data

- stan_glm

2. Visualize statistic information of the chain

- MCMC-intervals: mean, median, quantiles, 95% confidence interval
- MCMC-recover: compare MCMC estimate with "true"
- MCMC-distribution: histogram, density estimation

3. MCMC-diagnostics

- mcmc_trace: trace
- mcmc_acf: autocorrelation
- mcmc_rhat: $\hat{R}$ statistic
- mcmc_neff: effective sample size

1. Simulate the data

---

stan_glm: Solve Bayesian generalized linear models via MCMC

```
data(mtcars)
mtcars <- mtcars[,1:4]
head(mtcars)
```

```
##                   mpg cyl disp  hp
## Mazda RX4         21.0   6  160 110
## Mazda RX4 Wag     21.0   6  160 110
## Datsun 710        22.8   4  108  93
## Hornet 4 Drive    21.4   6  258 110
```

```
## Hornet Sportabout 18.7   8  360 175
## Valiant           18.1   6  225 105
```

```
fit <- stan_glm(mpg ~ .,data = mtcars,seed = 1)
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.127 seconds (Warm-up)
##                0.149 seconds (Sampling)
##                0.276 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.129 seconds (Warm-up)
##                0.097 seconds (Sampling)
##                0.226 seconds (Total)
##
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.126 seconds (Warm-up)
##                0.097 seconds (Sampling)
##                0.223 seconds (Total)
##
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.122 seconds (Warm-up)
##                0.095 seconds (Sampling)
##                0.217 seconds (Total)
```

```r
# Posterior MCMC draws with 4 chains
posterior <- as.array(fit)

# Take 1st chain as example
```

```r
chain <- posterior[,1,]

# Gibbs Sampler will give us a 1000*5 matrix
# 1000 = # of iterations
# 5 = # of parameters
head(chain)
```

```
##           parameters
## iterations (Intercept)        cyl         disp           hp    sigma
##       [1,]    37.54211 -1.5953525 -0.003410527 -0.047285674 3.133655
##       [2,]    36.62432 -1.4035704  0.005143233 -0.058735576 3.940134
##       [3,]    40.48507 -2.8354804  0.009227668 -0.032918983 3.562376
##       [4,]    32.20719 -0.3396091 -0.008391830 -0.056087074 3.518481
##       [5,]    32.28238 -0.4483917 -0.017487522 -0.037144416 3.301781
##       [6,]    35.09456 -2.3206253 -0.006683100  0.003948312 2.945567
```

```r
#
summary(fit)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      mpg ~ .
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 32
##  predictors:   4
##
## Estimates:
##                 mean   sd    2.5%   25%   50%   75%   97.5%
## (Intercept)    34.2    2.7  28.9   32.4  34.2  36.0  39.5
## cyl            -1.2    0.8  -2.9   -1.8  -1.2  -0.7   0.5
## disp            0.0    0.0   0.0    0.0   0.0   0.0   0.0
## hp              0.0    0.0   0.0    0.0   0.0   0.0   0.0
## sigma           3.2    0.4   2.5    2.8   3.1   3.4   4.1
## mean_PPD       20.1    0.8  18.5   19.6  20.1  20.6  21.7
## log-posterior -90.8    1.7 -95.1  -91.7 -90.5 -89.6 -88.6
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  2616
## cyl           0.0  1.0  2105
## disp          0.0  1.0  2479
## hp            0.0  1.0  2855
## sigma         0.0  1.0  3209
## mean_PPD      0.0  1.0  3720
## log-posterior 0.0  1.0  1627
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
# Parameters we are interested in
mypara <- chain[,c("cyl","sigma")]
```

```r
#
head(mypara)
```

```
##           parameters
## iterations        cyl    sigma
##       [1,] -1.5953525 3.133655
##       [2,] -1.4035704 3.940134
##       [3,] -2.8354804 3.562376
##       [4,] -0.3396091 3.518481
##       [5,] -0.4483917 3.301781
##       [6,] -2.3206253 2.945567
```

As a conclusion

- fit: Bayesian generalized linear regression by MCMC along with Stan method
- posterior: 3-D matrix, [# of iterations, # of chains, # of parameteres]
- chain: 2-D matrix / 1st chain of posterior, [# of iterations, # of parameters]
- mypara: 2-D matrix / subset of chain, [# of iterations, # of parameters we are interested in]

2. Visualize statistic information of the chain

MCMC-intervals: Plot interval estimates from MCMC draws.

Description

- mcmc_intervals: Plot central (quantile-based) posterior interval estimates from MCMC draws.
- mcmc_areas: To show the uncertainty intervals as shaded areas under the estimated posterior density curves.

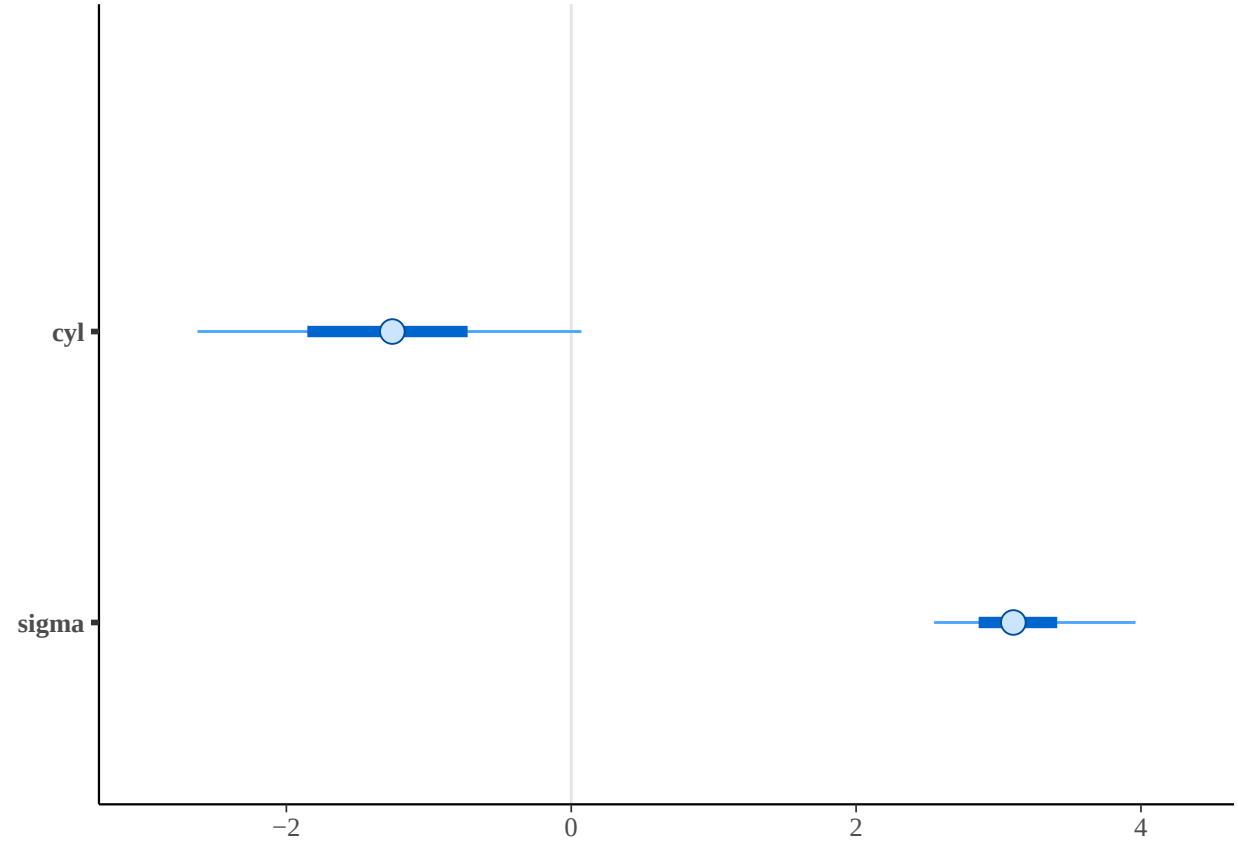
Usage

- mcmc_intervals(x,pars = character(),prob = 0.5,prob_outer = 0.9,point_est = c("median", "mean", "none"))

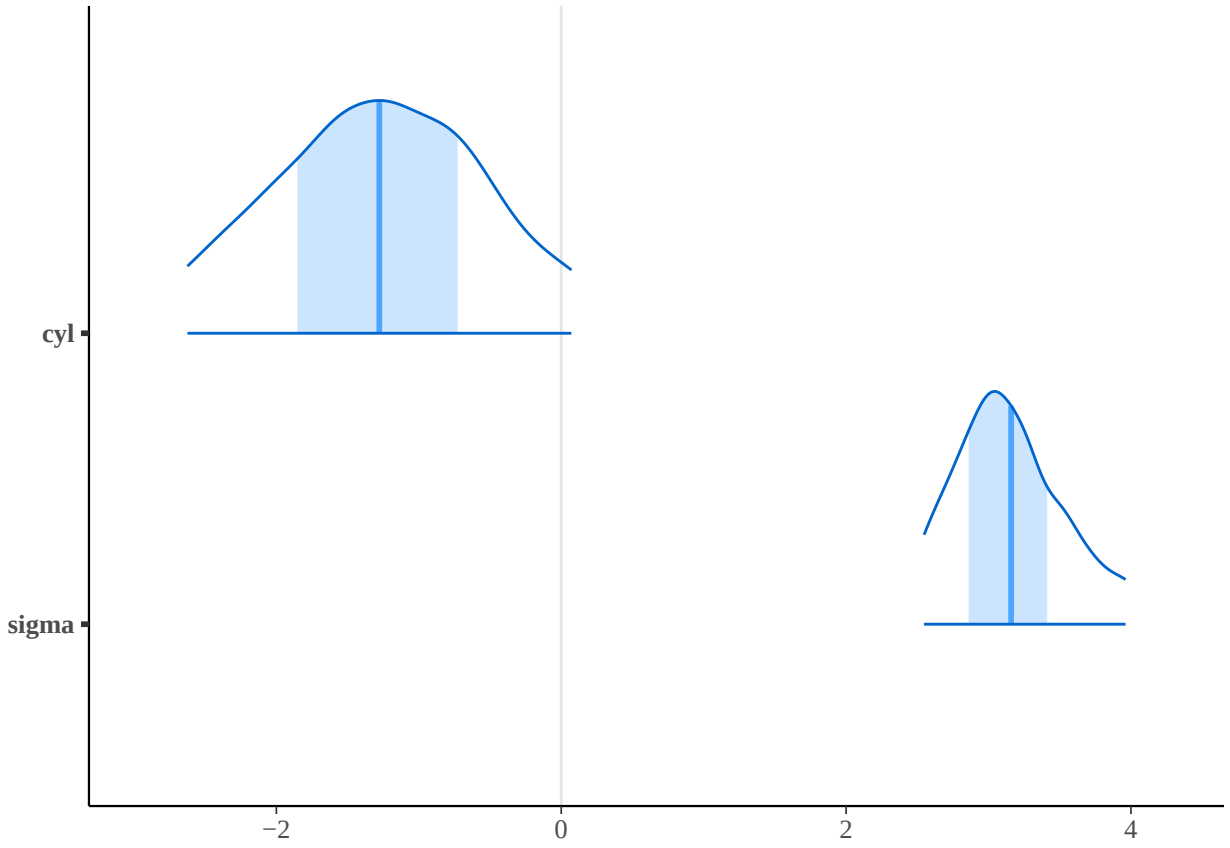
Arguments

- x: A 2-D matrix / data frame of MCMC draws.
- par: Parameters you are interested in.
- prob: The probability mass to include in the inner interval (for mcmc_intervals) or in the shaded region (for mcmc_areas). The default is 0.5 (50% interval).
- prob_outer: The probability mass to include in the outer interval. The default is 0.9 for mcmc_intervals (90% interval) and 1 for mcmc_areas.
- point_est: The point estimate to show. Either "median" (the default), "mean", or "none".

```r
color_scheme_set("brightblue")
#
mcmc_intervals(mypara, pars = c("cyl","sigma"))
```




```r
#
mcmc_areas(
  mypara,
  pars = c("cyl","sigma"),
  prob = 0.5,
  prob_outer = 0.9,
  point_est = "mean"
)
```



MCMC-recover: Compare MCMC estimates to "true" parameter values

---

Description

- Plots comparing MCMC estimates to "true" parameter values. Before fitting a model to real data it is useful to simulate data according to the model using known (fixed) parameter values and to check that these "true" parameter values are (approximately) recovered by fitting the model to the simulated data.

Usage

- mcmc_recover_intervals(x,true,prob = 0.5,prob_outer = 0.9,point_est = c("median", "mean", "none"),size = 4, alpha = 1)
- mcmc_recover_scatter(x,true,point_est = c("median", "mean"), size = 3,alpha = 1)

Arguments

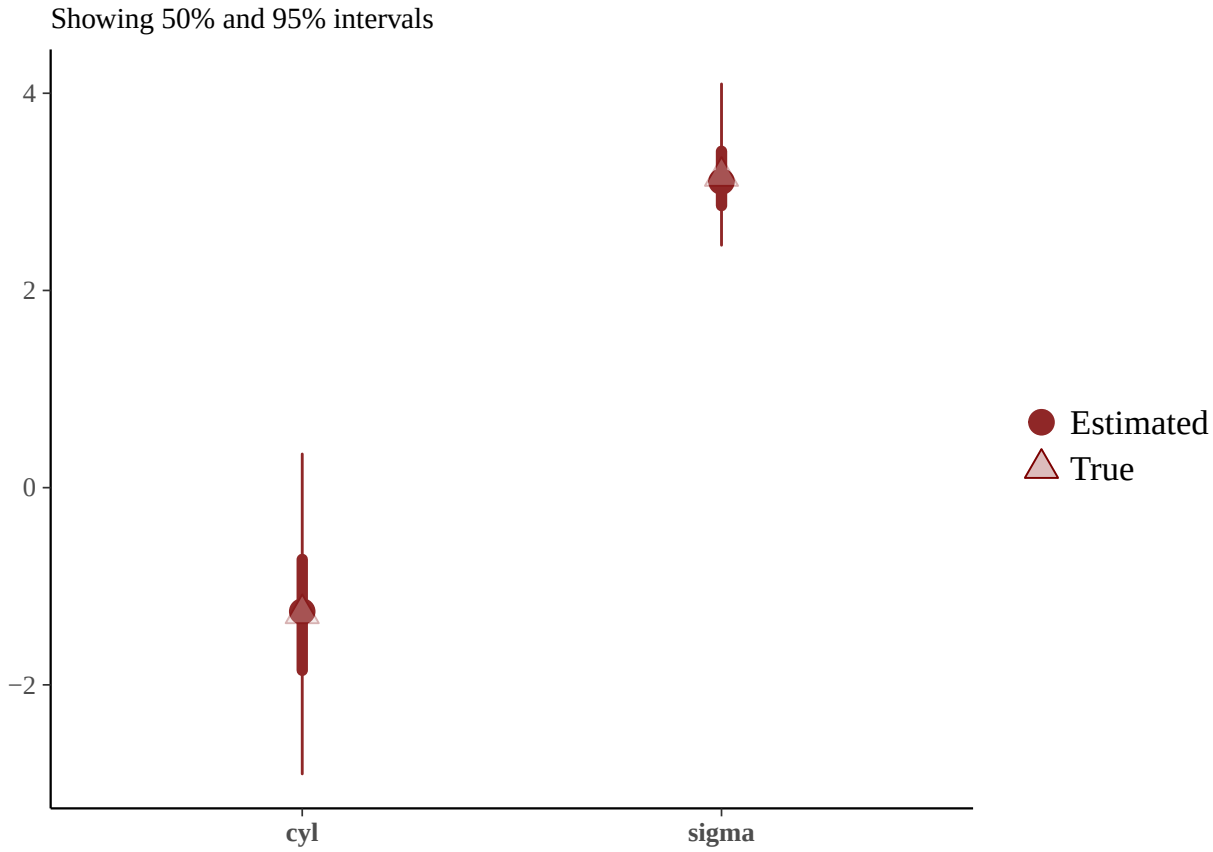
- x: A 2-D matrix / data frame of MCMC draws.
- true: A numeric vector of "true" values of the parameters in x. There should be one value in true for each parameter included in x and the order of the parameters in true should be the same as the order of the parameters in x.

- prob: The probability mass to include in the inner interval for mcmc_recover_intervals. The default is 0.5 (50% interval).
- prob_outer: The probability mass to include in the outer interval. The default is 0.9 for mcmc_recover_intervals (90% interval).
- point_est: The point estimate to show. Either "median" (the default), "mean", or "none".
- size, alpha: Passed to 'geom_point' to control the appearance of plotted points.

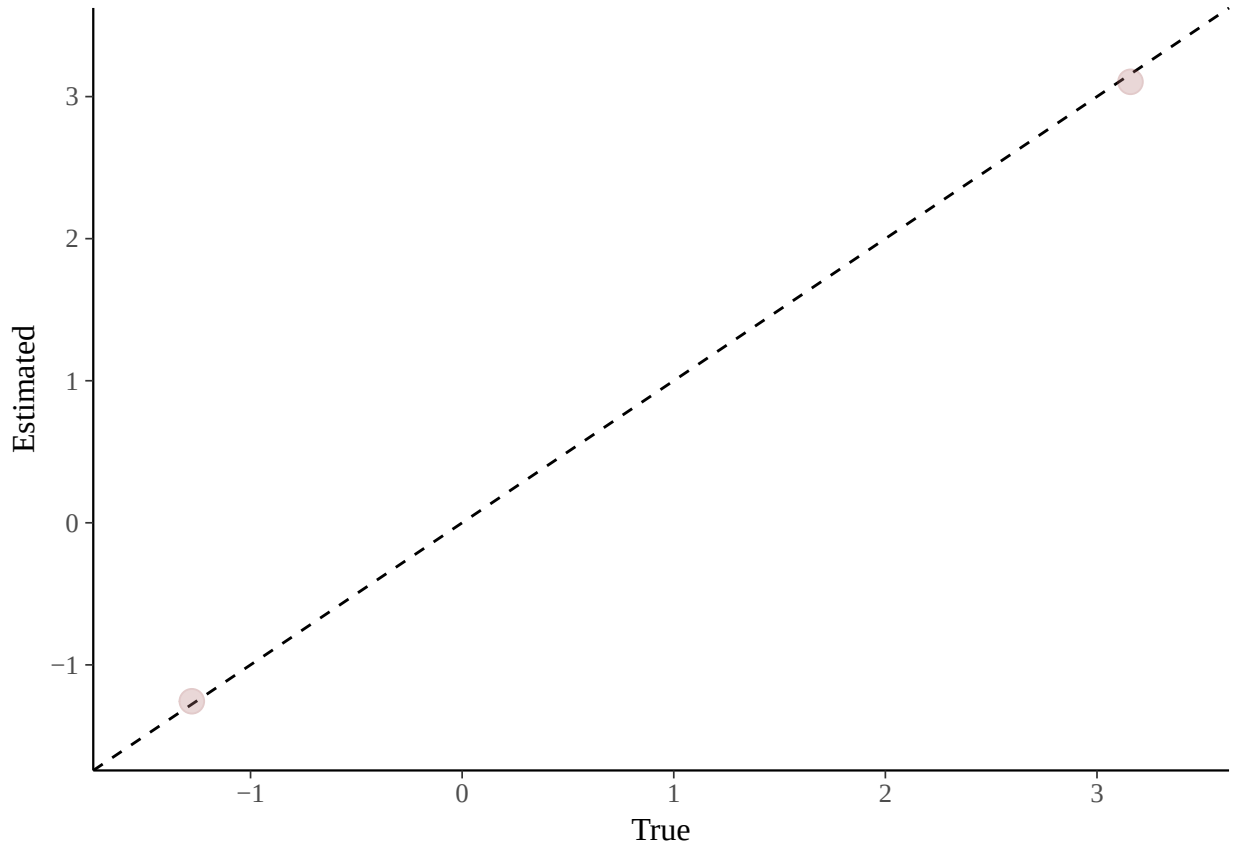
```
true <- apply(mypara,2,mean)
true
```

```
##       cyl     sigma
## -1.277434  3.158333
```

```
#
color_scheme_set("red")
mcmc_recover_intervals(mypara,true,
                       prob_outer = 0.95,
                       size = 4,alpha = 0.3)
```




```
mcmc_recover_scatter(mypara,true,
                       size = 4,alpha = 0.3)
```



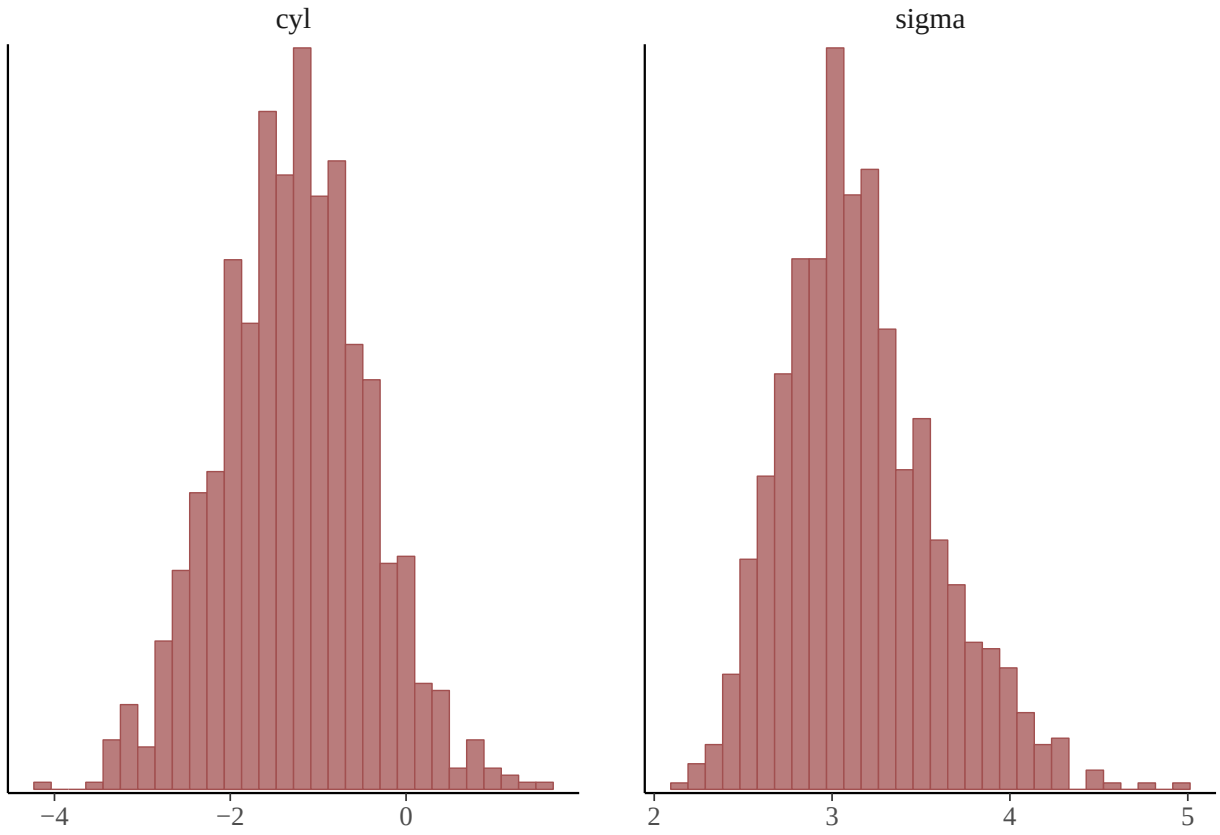
MCMC-distributions: Histograms and kernel density plots of MCMC draws

---

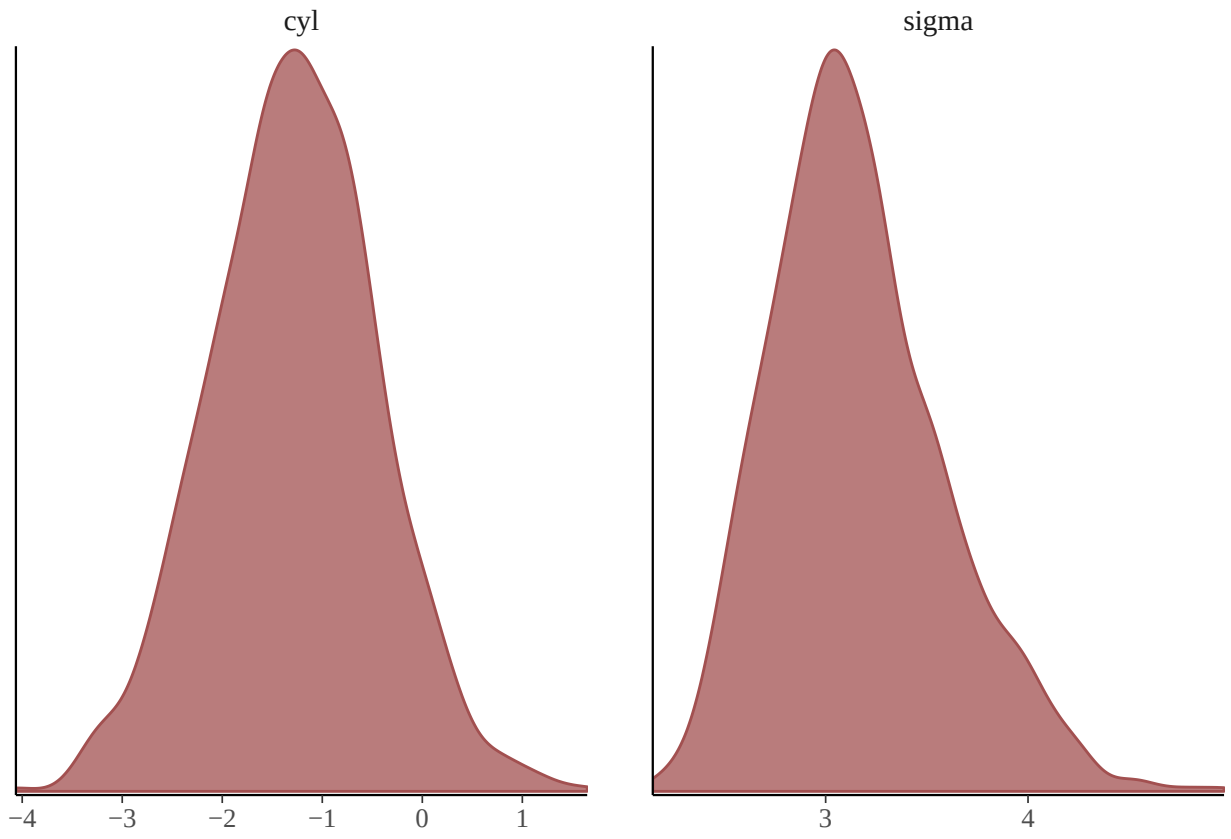
Description

- Various types of histograms and kernel density plots of MCMC draws.

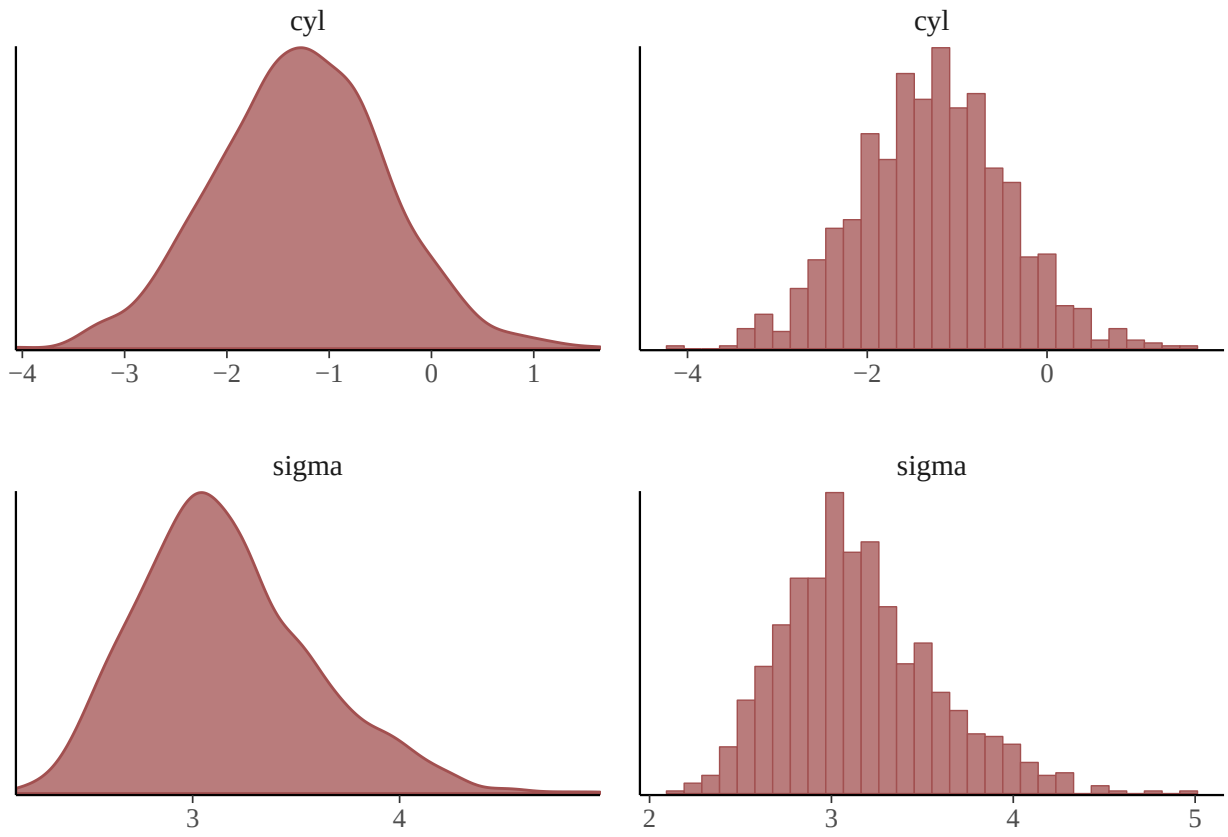
```
mcmc_hist(mypara)
```




```
mcmc_dens(mypara,pars = c("cyl","sigma"))
```




```
mcmc_combo(mypara,c("dens","hist"))
```



3. Whether or not my chain is good enough to use

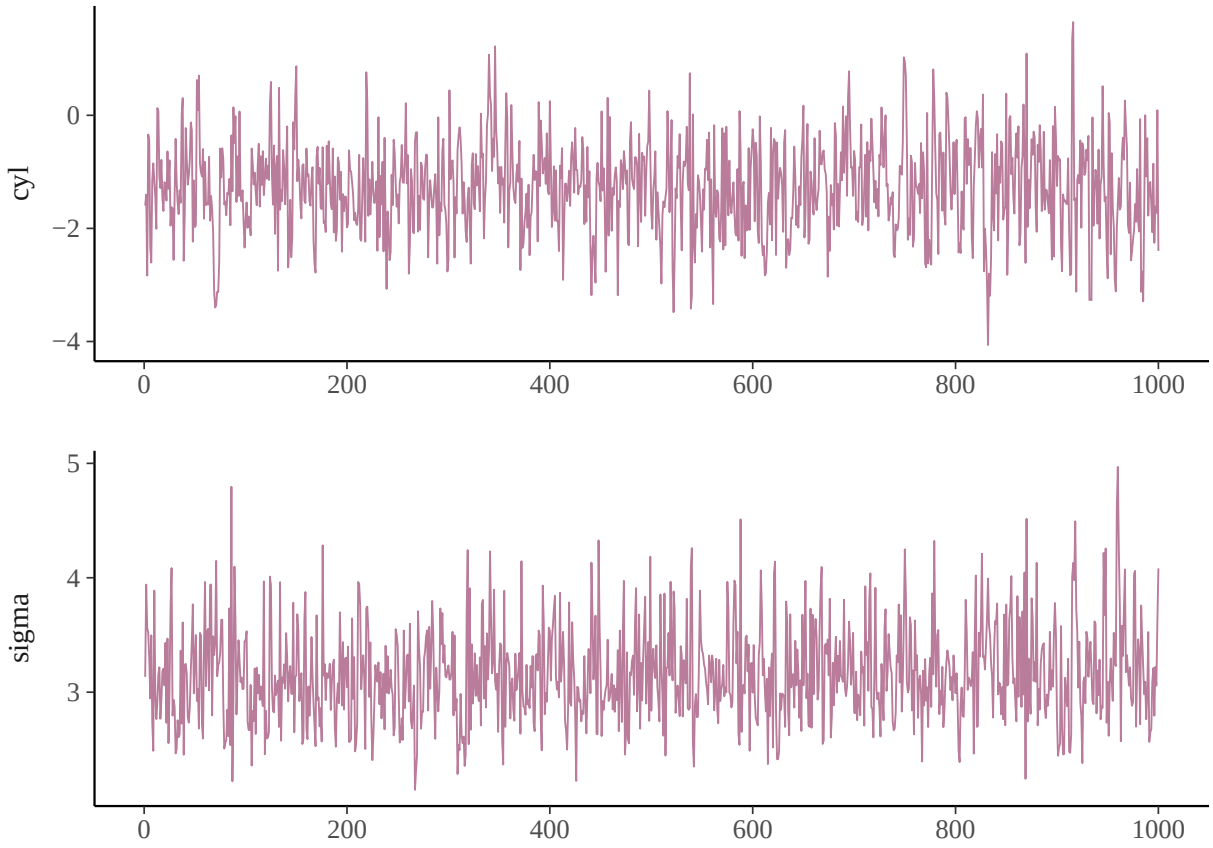
MCMC-diagnostics: General MCMC diagnostics

---

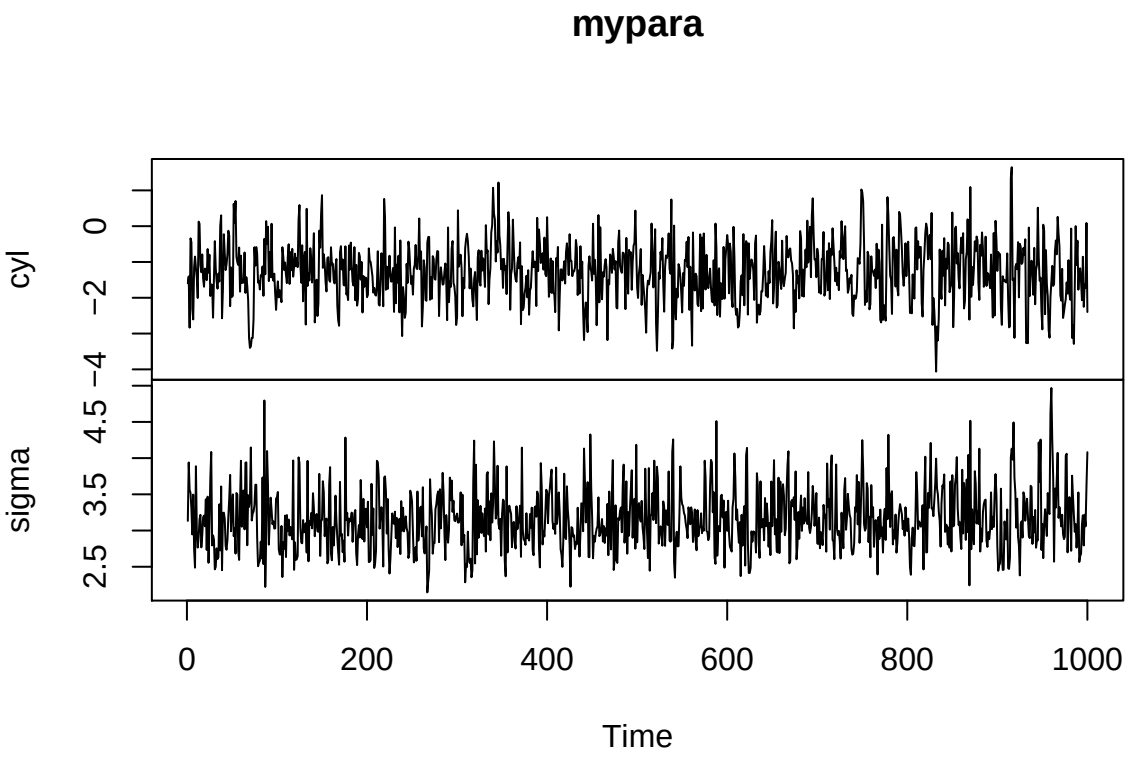
mcmc_trace: Trace plot (time series plot) of MCMC draws

```
color_scheme_set("pink")

mcmc_trace(mypara,pars = c("cyl","sigma"),
           facet_args = list(ncol=1,strip.position="left"))
```



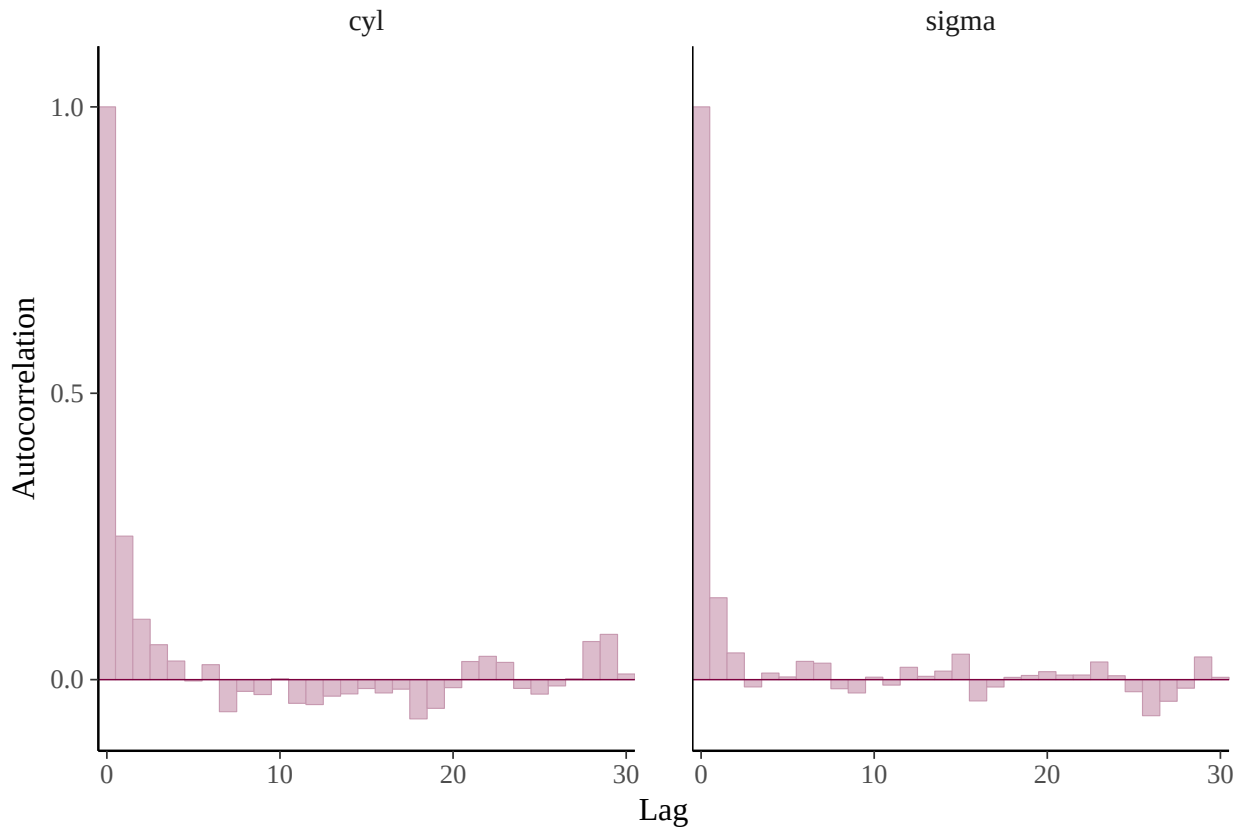

```
# compare results with function 'plot.ts' from package 'mcmcse'
plot.ts(mypara)
```



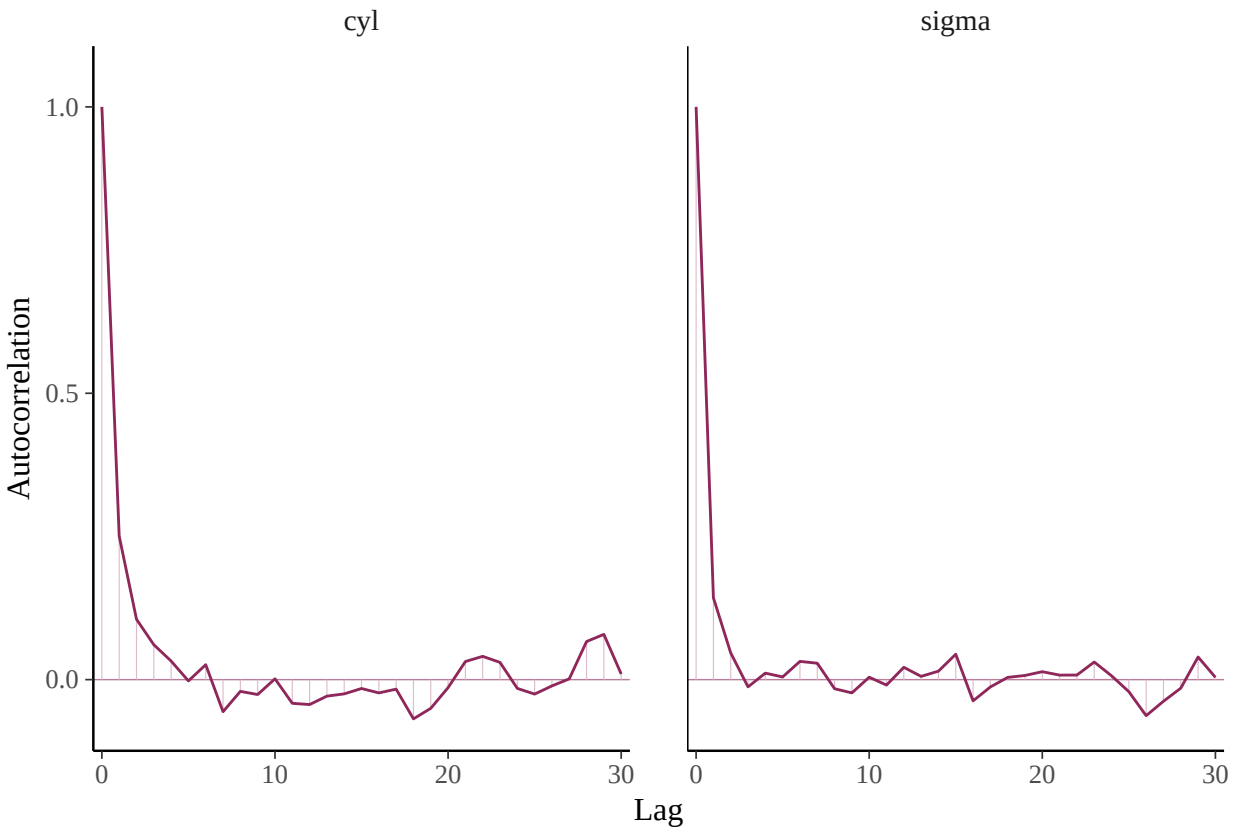
---

mcmc_acf: Plots of autocorrelation of MCMC draws

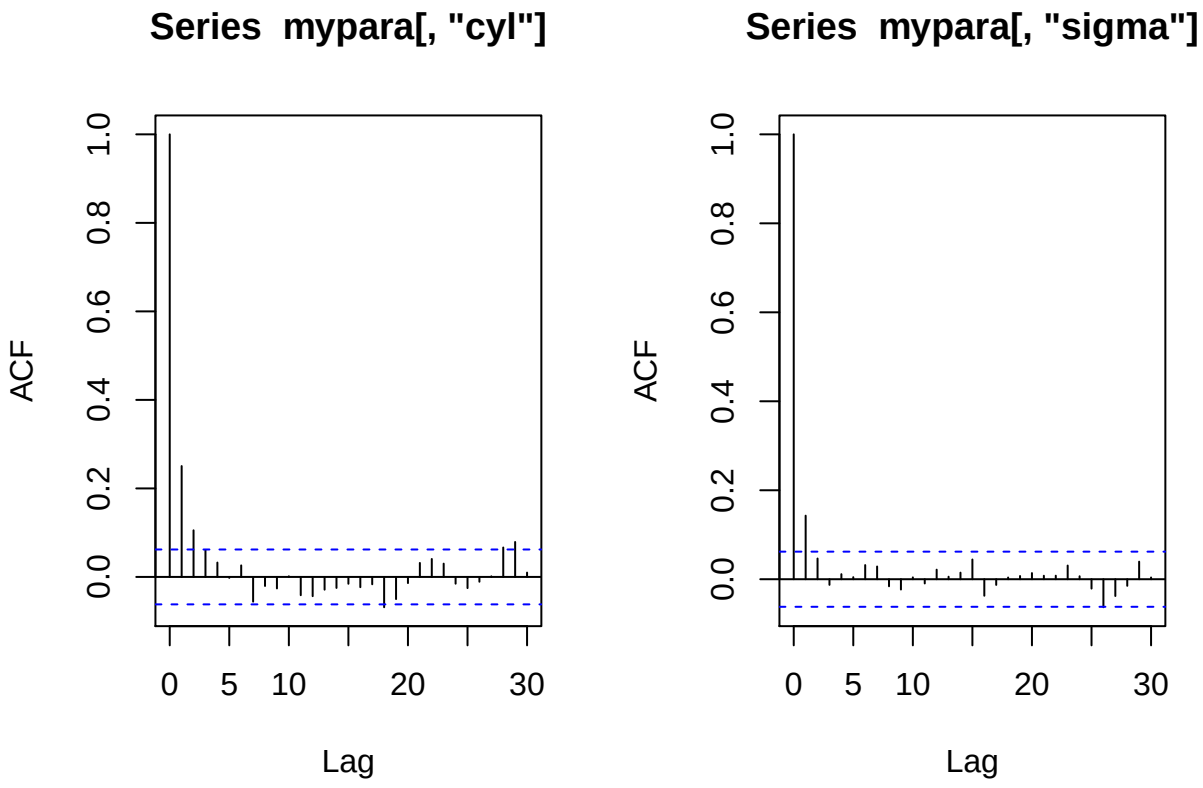
```
# bar plot
mcmc_acf_bar(mypara,lags = 30)
```




```
# line plot
mcmc_acf(mypara,pars = c("cyl","sigma"),
         # facet_args = list(ncol=1,strip.position="left"),
         lags = 30)
```




```
# compare results with function 'acf' from package 'mcmcse'
par(mfrow=c(1,2))
acf(mypara[,"cyl"])
acf(mypara[,"sigma"])
```




```
par(mfrow=c(1,1))
```

---

mcmc_rhat: Plots of Rhat statistics of MCMC draws

mcmc_neff: Plots of ratios of effective sample size to total sample size ofMCMC draws

---

Rhat: potential scale reduction statistic

One way to monitor whether a chain has converged to the equilibrium distribution is to compare its behavior to other randomly initialized chains. This is the motivation for the Gelman and Rubin (1992) potential scale reduction statistic, $\hat{R}$. The $\hat{R}$ statistic measures the ratio of the average variance of samples within each chain to the variance of the pooled samples across chains; if all chains are at equilibrium, these will be the same and $\hat{R}$ will be one. If the chains have not converged to a common distribution, the $\hat{R}$ statistic will be greater than one.

```
# Go back to 'fit'
# fit <- stan_glm(mpg ~ .,data = mtcars,seed = 1)
#
# rhat
rhat <- rhat(fit)[c(2,5)]
rhat
```

```
##      cyl    sigma
## 1.001245 1.001387
```

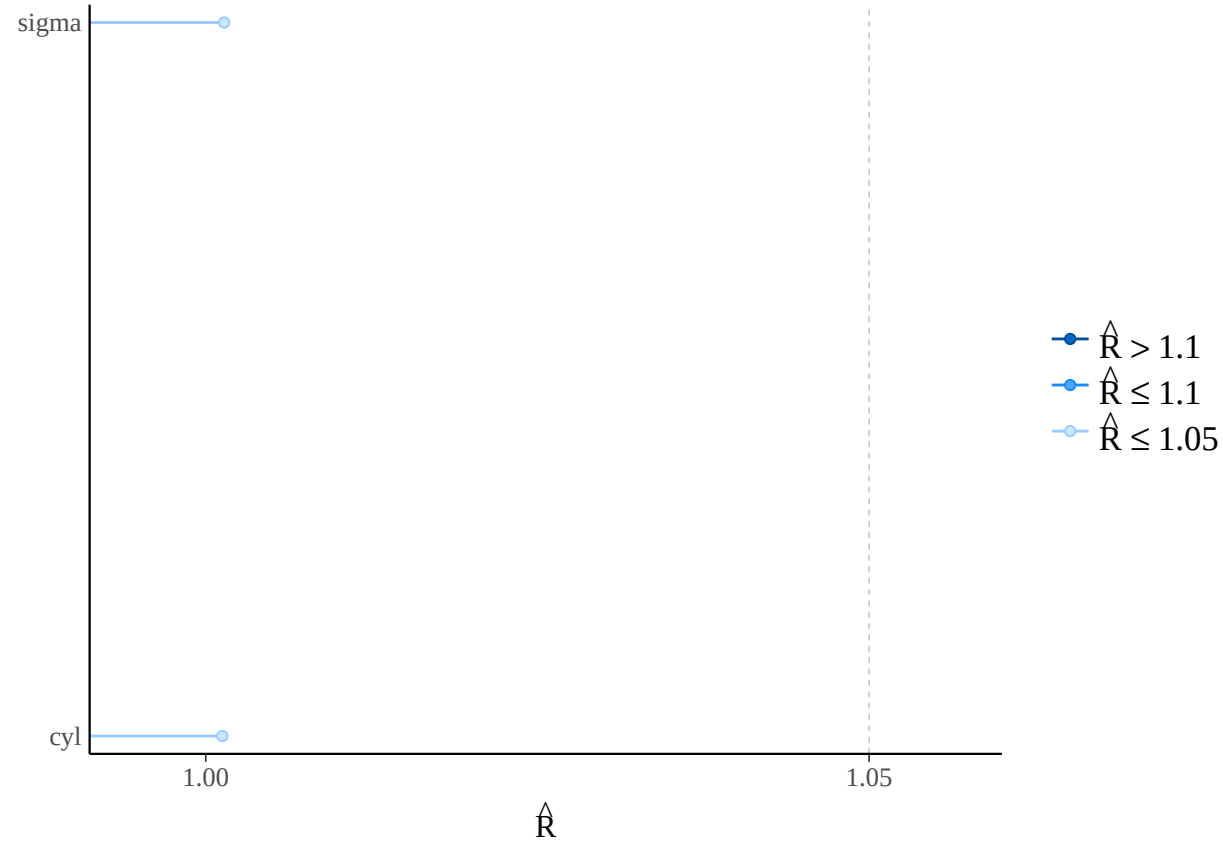
```
# ratio of effective sample size to the total sample size by 'neff_ratio'
ratio <- neff_ratio(fit)[c(2,5)]
ratio
```

```
##     cyl   sigma
## 0.52625 0.80225
```

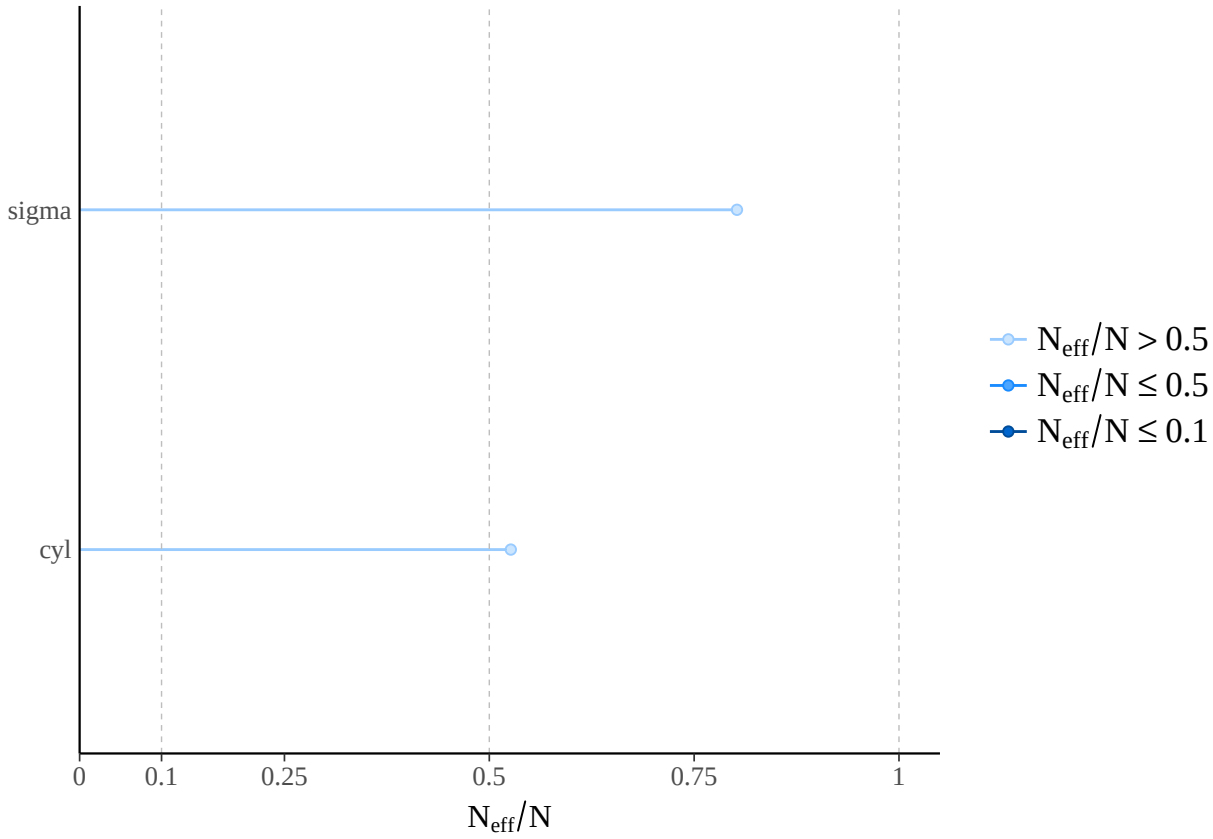
```
# # ratio of effective sample size to the total sample size by 'ess'
ess(mypara)/1000
```

```
##       cyl     sigma
## 0.7061574 0.9124397
```

```
color_scheme_set("brightblue")
mcmc_rhat(rhat) + yaxis_text(hjust = 1)
```




```
mcmc_neff(ratio) + yaxis_text(hjust = 1)
```


sigma
cyl
0
0.1
0.25
0.5
0.75
1
N_{eff}/N
$N_{eff}/N > 0.5$
$N_{eff}/N \leq 0.5$
$N_{eff}/N \leq 0.1$