ptb: an R package for prevalence estimation and diagnostic test evaluation in a Bayesian framework

Oswaldo Santos Baquero

contact: baquero@usp.br

The ptb R package is a work in progress to implement functions, under a Bayesian framework, for prevalence estimation, diagnostic test evaluation and prior elicitation. The ptb runs with JAGS (http://sourceforge.net/projects/mcmc-jags/) as backend. The functions implemented so far are generalizations of code presented by professor Ian Garner in a course he offered in São Paulo, Brazil. As ptb functions return R objects, other nice packages can be used for model diagnostics.

Installation

After installing JAGS (http://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/), use the following commands:

```
library(devtools)
install_github("leb-fmvz-usp/ptb")
```

Load packages

```
library(ptb); library(coda); library(ggmcmc)
```

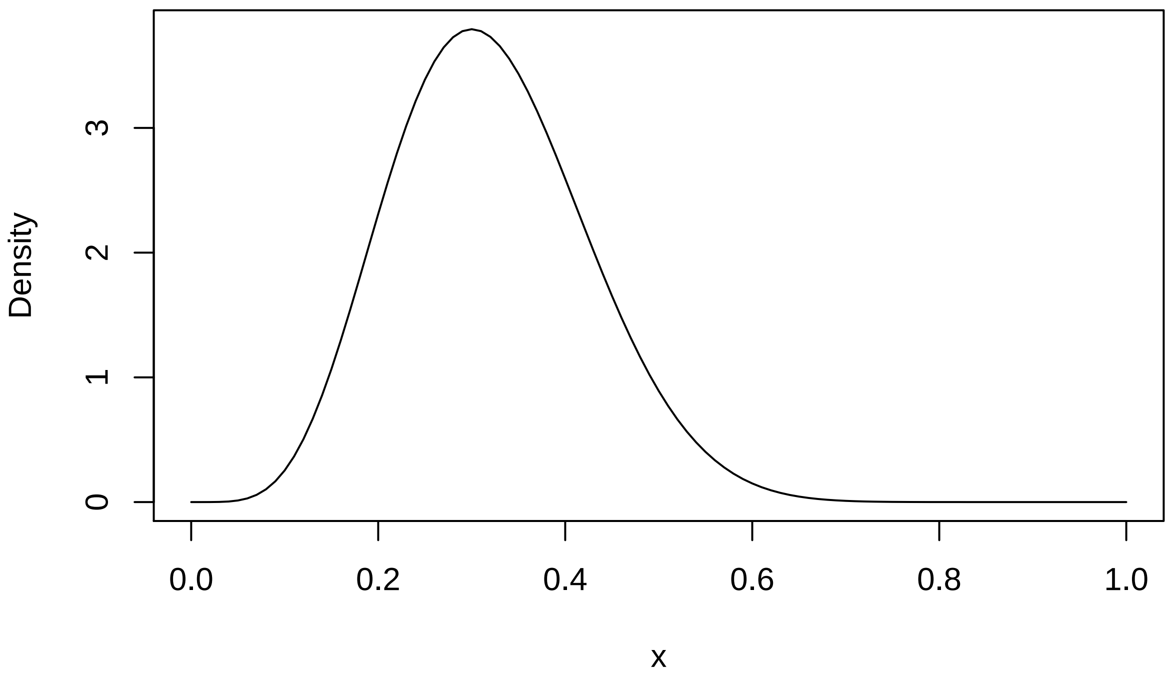
Fit a Beta distribution from elicited information

The function `ElicitBeta` can be used to calculate the parameters a and b of a Beta distribution. The function takes as imputs the elicited mode for the variable of interest and the maximum or minimum elicited value for that variable. A confidence about this maximum or minimum must be provided too. The function `PlotElicitedPrior` plots the resulting ditribution.

```
(se_prior <- ElicitBeta(mode = 0.3, maximum = 0.5, confidence = 0.95))
```

```
## a (shape1) b (shape2)
##       6.28      13.32
## attr(,"class")
## [1] "ElicitBeta"
```

```
PlotElicitedPrior(se_prior)
```



The argument `summary` provides summary statistics and the argument `quantiles` allows the specification of specific quantiles.

```
ElicitBeta(mode = 0.3, maximum = 0.5, confidence = 0.95, summary = TRUE)
```

```
## $`a (shape1)`
## [1] 6.28
##
## $`b (shape2)`
## [1] 13.32
##
## $Mean
## [1] 0.3204082
##
## $Variance
## [1] 0.01057023
##
## $Quantiles
##     0.025      0.25       0.5      0.75     0.975
## 0.1390975 0.2459714 0.3141901 0.3882998 0.5363675
##
## attr(,"class")
## [1] "ElicitBeta"
```

```
ElicitBeta(mode = 0.3, maximum = 0.5, confidence = 0.95,
           summary = TRUE, quantiles = c(0.1, 0.9))
```

```
## $`a (shape1)`
## [1] 6.28
##
## $`b (shape2)`
## [1] 13.32
##
## $Mean
## [1] 0.3204082
##
## $Variance
## [1] 0.01057023
##
## $Quantiles
##       0.1       0.9
## 0.1913223 0.4579182
##
## attr(,"class")
## [1] "ElicitBeta"
```

Models

To compile and run the models, we always need to define the data, the priors and the parameters to be monitored.
By default, three parallel chains are run, the "burn in" is equal to 1e3, starting values are automatically generated, the number of effective iterations is equal to 1e4 and chains are not "thinned". For details, see the help pages.

One test and one population binomial model

```r
# help(OneTestOnePopBM)

# Data
dataset <- list(pop_size = 91, positives = 1)

# Initial conditions for chains (optional)
inits <- list(list(true_prev = 0.05, se = 0.8, sp = 0.9),
              list(true_prev = 0.02, se = 0.3, sp = 0.7),
              list(true_prev = 0.09, se = 0.1, sp = 0.5))

# Priors
priors <- c(true_prev_a = 1, true_prev_b = 1,
            se_a = 6.28, se_b = 13.32,
            sp_a = 212.12, sp_b = 3.13)

# Prevalence estimate
prev_est <- OneTestOnePopBM(dataset = dataset, inits = inits,
                            priors = priors,
                            pars = c('true_prev', 'ppv', 'npv'))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 3
##    Total graph size: 19
##
## Initializing model
```

```
summary(prev_est)
```

```
##
## Iterations = 2001:12000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## npv       0.93954 0.08377 0.0004837       0.001949
## ppv       0.53795 0.25349 0.0014635       0.002673
## true_prev 0.07525 0.09008 0.0005201       0.001992
##
## 2. Quantiles for each variable:
##
##               2.5%     25%     50%     75%  97.5%
## npv       0.725463 0.92642 0.96464 0.98517 0.9986
## ppv       0.045050 0.34287 0.56945 0.74794 0.9281
## true_prev 0.002114 0.02157 0.04924 0.09572 0.3109
```

One test and one population binomial mixture model

```
# help(OneTestOnePopBMM)

# Data
dataset <- list(pop_size = 91, positives = 1)

# Priors
priors <- list(true_prev_wph_a = 1.8, true_prev_wph_b = 26.74,
               se_a = 6.28, se_b = 13.32,
               sp_a = 212.12, sp_b = 3.13,
               prev_h = 0.1)


# Prevalence estimates
prev_est <- OneTestOnePopBMM(dataset = dataset, priors = priors, n_iter = 3e3,
                             pars = c('true_prev', 'true_prev_wph', 'prev_h'))
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 4
##    Total graph size: 24
##
## Initializing model
```

```
summary(prev_est)
```

```
##
## Iterations = 2001:5000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 3000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                    Mean      SD  Naive SE Time-series SE
## prev_h        0.063111 0.24318 0.0025633      0.0026016
## true_prev     0.002635 0.01274 0.0001343      0.0001357
## true_prev_wph 0.061357 0.04450 0.0004691      0.0007996
##
## 2. Quantiles for each variable:
##
##                    2.5%     25%     50%    75%   97.5%
## prev_h        0.000000 0.00000 0.00000 0.0000 1.00000
## true_prev     0.000000 0.00000 0.00000 0.0000 0.04183
## true_prev_wph 0.006603 0.02816 0.05119 0.0844 0.17159
```

Difference between estimates

```r
# help(DiffBetweenEstimates)

# Priors
priors <- c(true_prev_a = 1, true_prev_b = 1,
            se_a = 6.28, se_b = 13.32, sp_a = 212.12, sp_b = 3.13)

# First estimate
dataset1 <- list(pop_size = 100, positives = 5)
prev_est1 <- OneTestOnePopBM(dataset = as.list(dataset1), n_iter = 3e3,
                                  priors = priors, pars = "true_prev",
                                  burn_in = 5e2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 3
##    Total graph size: 19
##
## Initializing model
```

```r
# Second estimate
dataset2 <- list(pop_size = 91, positives = 1)
prev_est2 <- OneTestOnePopBM(dataset = as.list(dataset2), n_iter = 3e3,
                                   priors = priors, pars = "true_prev",
                                   burn_in = 5e2)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1
##    Unobserved stochastic nodes: 3
##    Total graph size: 19
##
## Initializing model
```

```
# Estimated difference.
diffs <- DiffBetweenEstimates(list(prev_est1, prev_est2))
summary(diffs)
```

```
##
## Iterations = 1501:4500
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 3000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                                Mean     SD Naive SE Time-series SE
## true_prev1 - true_prev2     0.1172 0.1619 0.001707       0.004218
## Pr(true_prev1 > true_prev2) 0.8257 0.3794 0.003999       0.007743
##
## 2. Quantiles for each variable:
##
##                                2.5%     25%     50%    75%  97.5%
## true_prev1 - true_prev2     -0.1653 0.02718 0.09916 0.1886 0.5118
## Pr(true_prev1 > true_prev2)  0.0000 1.00000 1.00000 1.0000 1.0000
```

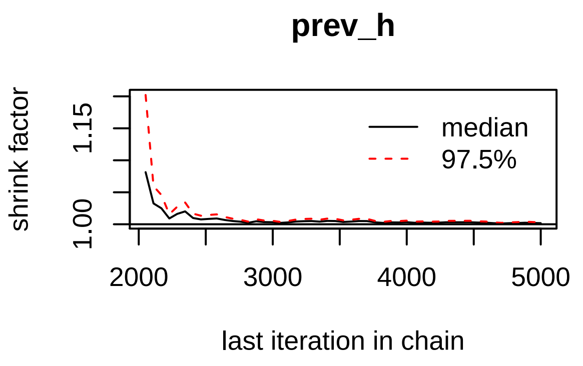
Diagnostics

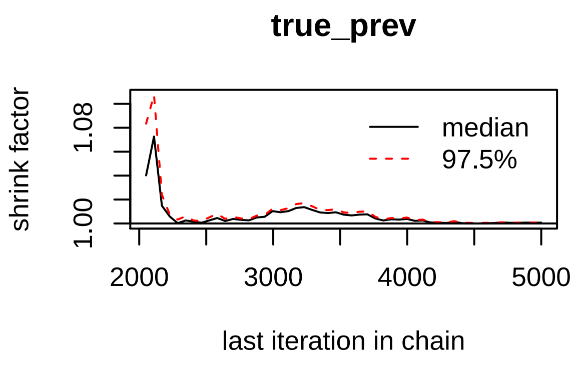
The model outputs are ready for diagnostics.

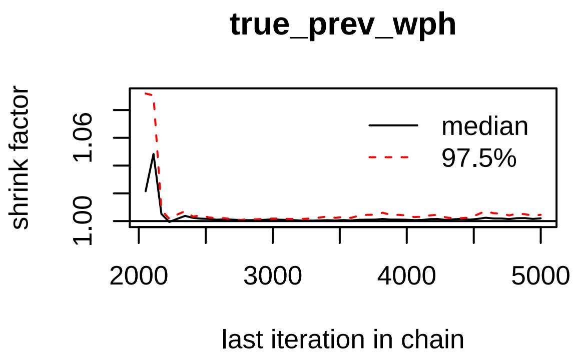
```
gelman.diag(prev_est)
```

```
## Potential scale reduction factors:
##
##              Point est. Upper C.I.
## prev_h                1          1
## true_prev             1          1
## true_prev_wph         1          1
##
## Multivariate psrf
##
## 1
```

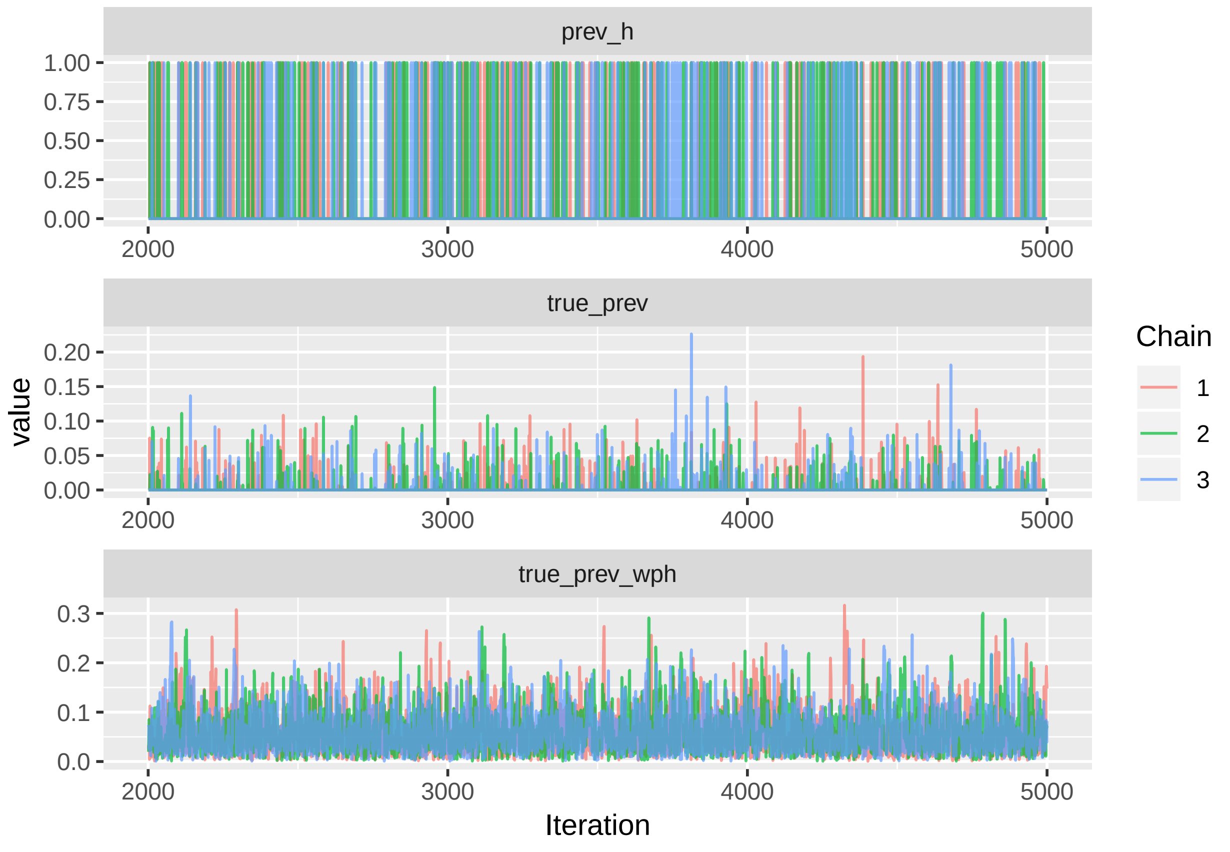
```
gelman.plot(prev_est)
```



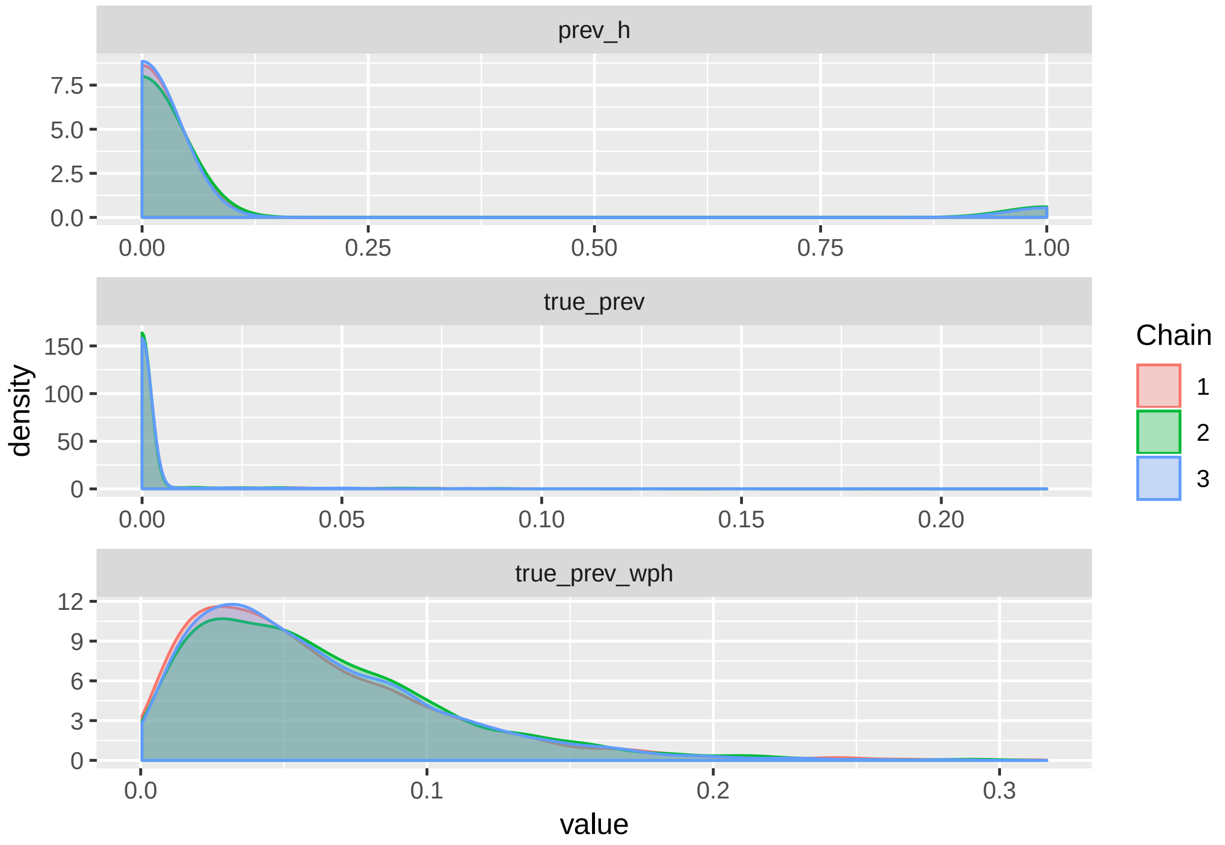




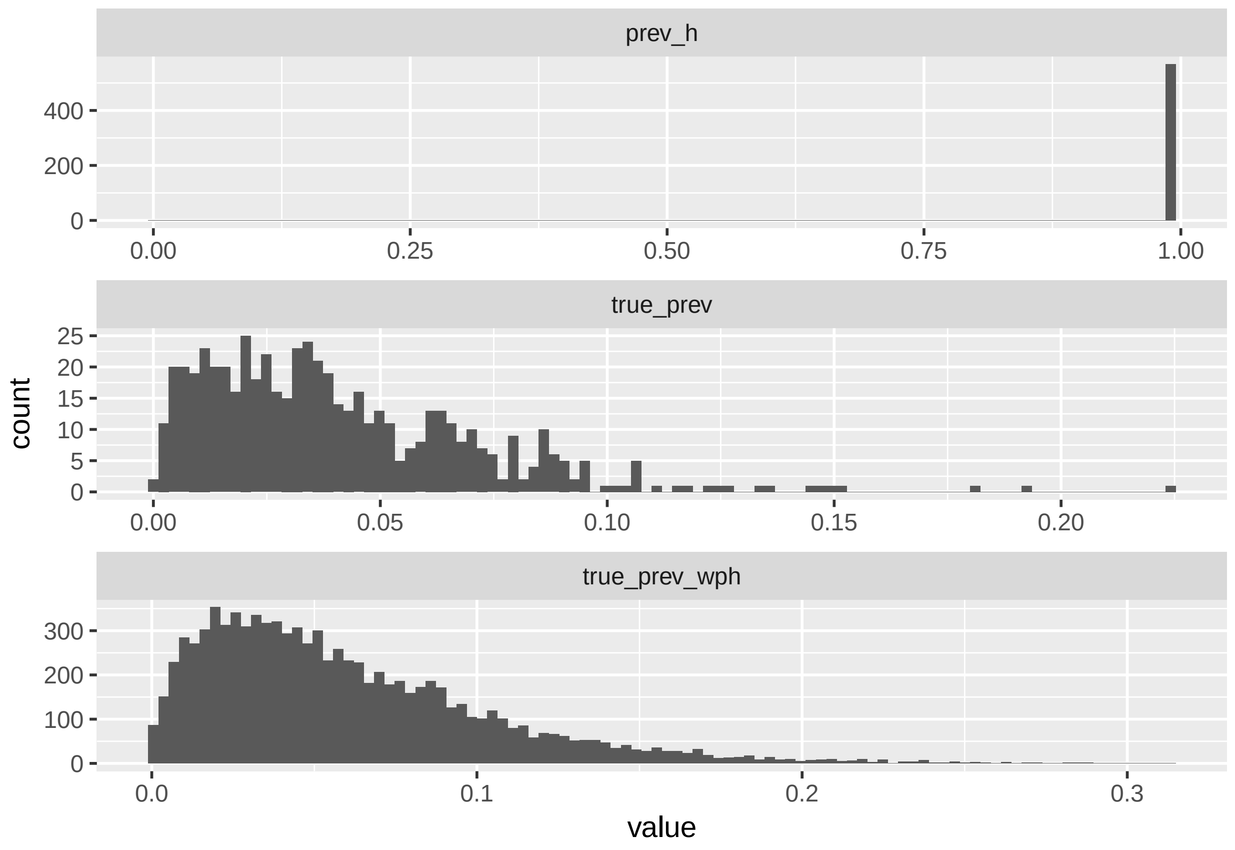
```
gg_res <- ggs(prev_est)
ggs_traceplot(gg_res)
```



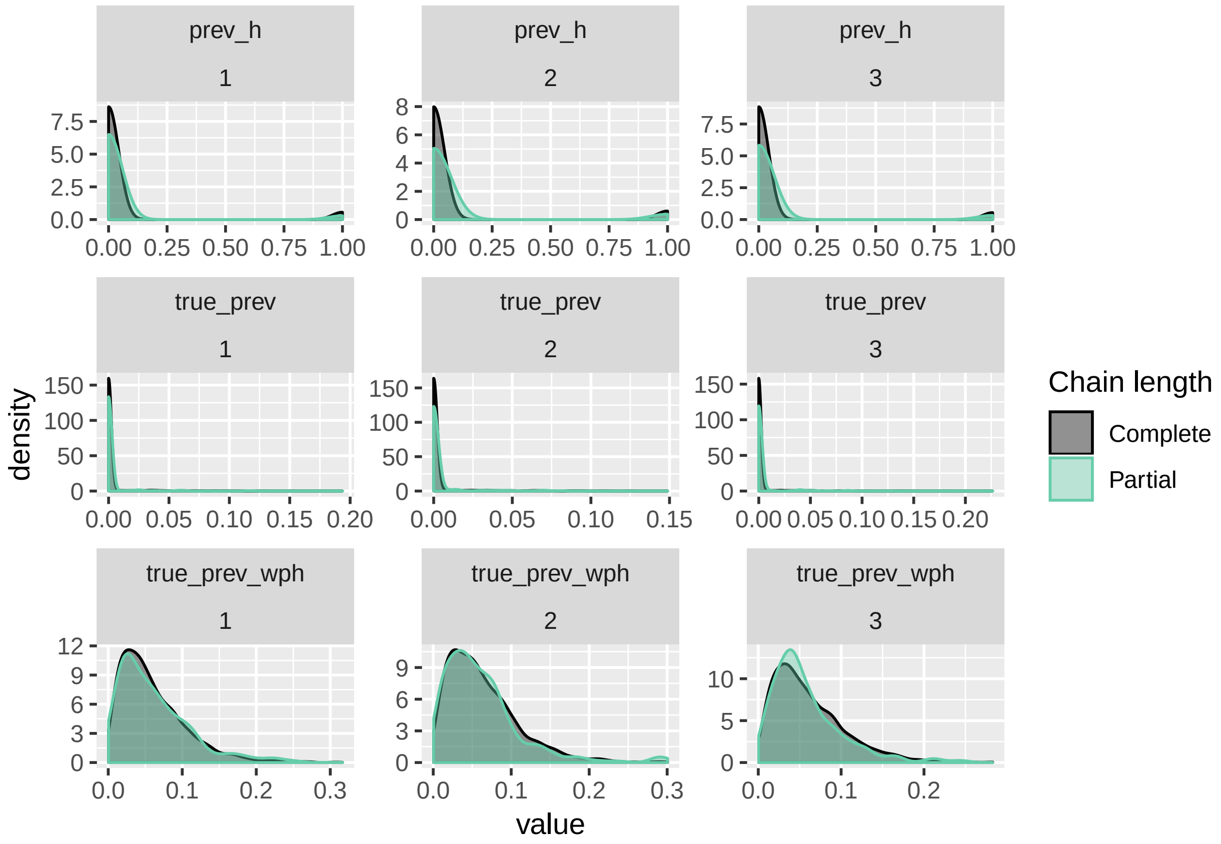
```r
ggs_density(gg_res)
```



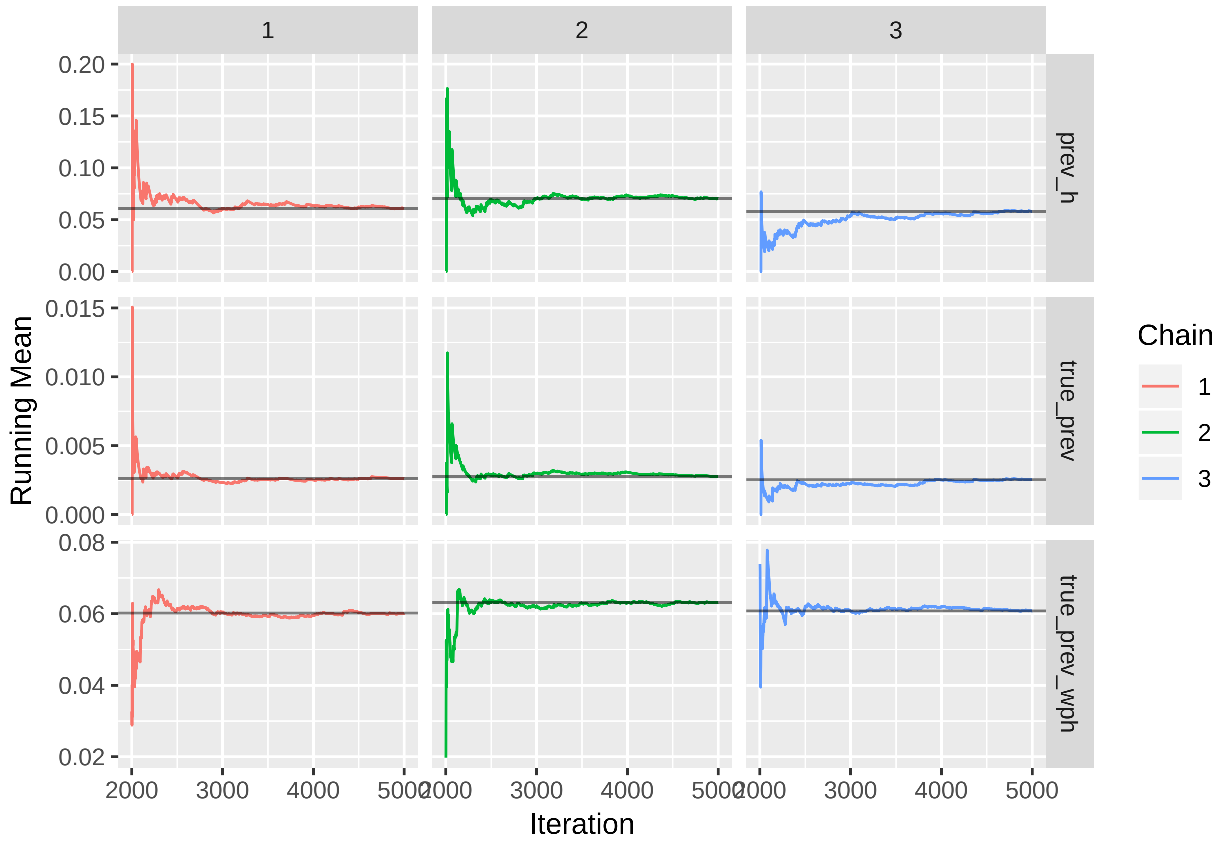
```r
ggs_histogram(gg_res, bins = 100)
```



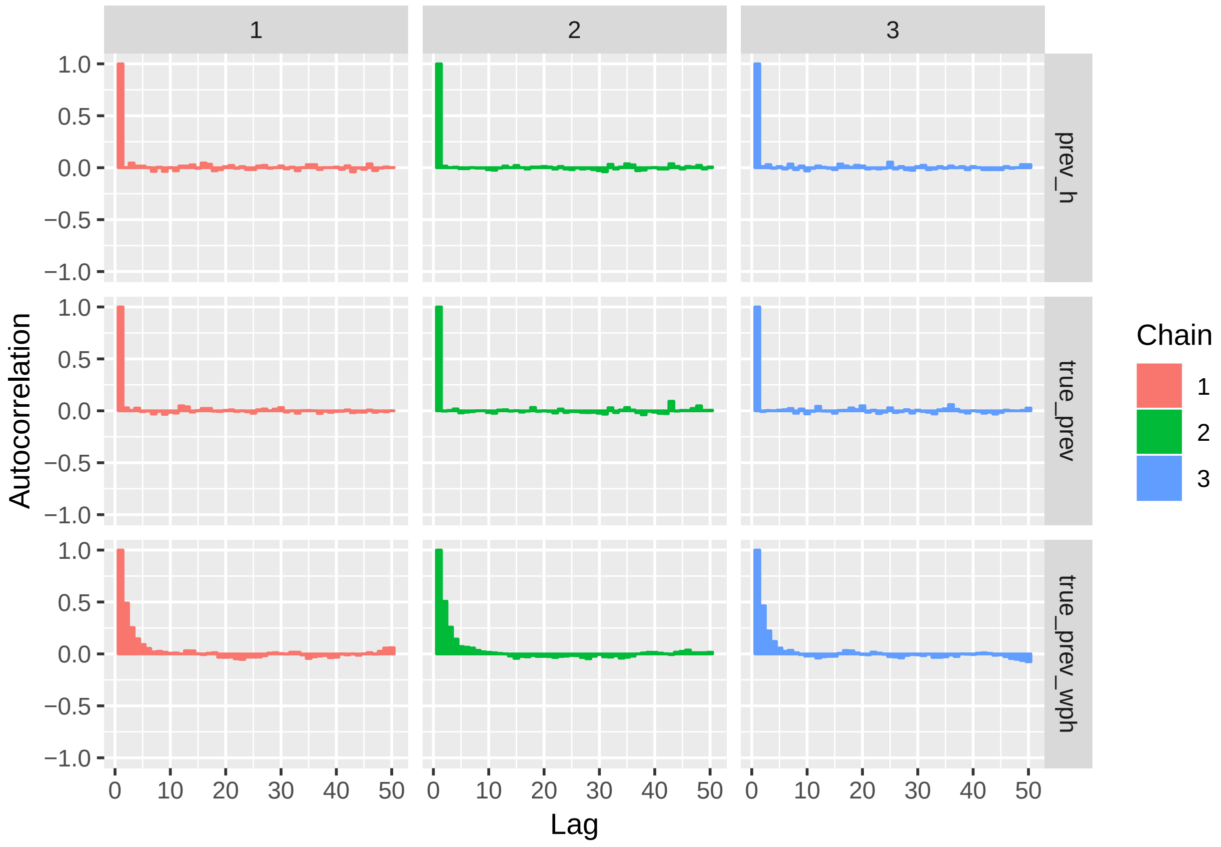
```r
ggs_compare_partial(gg_res)
```



```r
ggs_running(gg_res)
```



```
ggs_autocorrelation(gg_res)
```



1
2
3
prev_h
true_prev
true_prev_wph
Autocorrelation
1.0
0.5
0.0
−0.5
−1.0
Lag
0
10
20
30
40
50
Chain
1
2
3