Package 'abcsmcrf'

August 17, 2025

Type Package

Version 1.0.0

Date 2024-07-10

Title Approximate Bayesian Computation sequential Monte Carlo via random forests

Description Parameter inference using random forests in the framework of Approximate Bayesian Computation sequential Monte Carlo.

License GPL (>= 2)

Imports abcrf,
crayon,
data.table,
dplyr,
drf,
EasyABC,
ggplot2,
Hmisc,
MASS,
matrixStats,
parallel,
pbapply,
RColorBrewer

Suggests coala,
deSolve,
invgamma,
knitr,
LaplacesDemon,
Pareto,
patchwork,
reshape2,
rmarkdown,
sensitivity,
tidyverse,
tseries,
weights

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

R topics documented:

`plot_compare_joint` *Plot and compare joint posterior distribution(s) from ABC-SMC-(D)RF result*

Description

`plot_compare_joint` plots the joint posterior distribution(s) for the provided ABC-SMC-(D)RF result. It can also compare the joint posterior distributions for the provided ABC-SMC-(D)RF result with several ABC-SMC-(D)RF plots results.

Usage

```
plot_compare_joint(
  plots = NULL,
  abc_results,
  parameters_truth = NULL,
  parameters_labels = NULL,
  lims = NULL,
  nBins = 4
)
```

Arguments

`plots`	An existed ABC-SMC-(D)RF joint plots result. If provided, `plot_compare_joint` will plot the new ABC-SMC-(D)RF result and compare it with provided plots results. If plots = NULL, `plot_compare_joint` will make a new plot for the ABC-SMC-(D)RF result.
`abc_results`	An ABC-SMC-(D)RF result. Will be plotted out by the function.
`parameters_truth`	A dataframe containing true values of parameters from the ground-truth distributions. If provided, the function will plot the true values or distributions of parameters.
`parameters_labels`	A dataframe containing labels in the plots for corresponding parameters. If provided, parameter labels will be exhibited on the plots' axes.
`lims`	A dataframe containing the maximum and minimum bounds for parameters. If provided, x-axis and y-axis will be scaled by them.
`nBins`	Number of contour bins shown in the plot. Default is 5.

Value

A list of ggplot2 objects containing the joint plots results of posterior distributions. The user can use the function to compare ABC-SMC-(D)RF joint plots with the joint posterior distribution(s) of other ABC-SMC-(D)RF result(s).

`plot_compare_marginal` *Plot and compare marginal posterior distribution(s) from ABC-SMC-(D)RF result*

Description

`plot_compare_marginal` plots the marginal posterior distribution(s) for the provided ABC-SMC-(D)RF result. It can also compare the marginal posterior distributions for the provided ABC-SMC-(D)RF result with several ABC-SMC-(D)RF plots results.

Usage

```
plot_compare_marginal(
  plots = NULL,
  abc_results,
  parameters_truth = NULL,
  parameters_labels = NULL,
  statistics_labels = NULL,
  xlimit = NULL,
  plot_statistics = FALSE,
  plot_truth_hist = TRUE,
  plot_hist = FALSE,
  plot_hist_point = FALSE,
  breaks = NULL,
  alpha_truth = 0.8,
  alpha = 0.3,
  plot_prior = FALSE
)
```

Arguments

`plots`	An existed ABC-SMC-(D)RF marginal plots result. If provided, `plot_compare_marginal` will plot the new ABC-SMC-(D)RF result and compare it with provided plots results. If plots = NULL, `plot_compare_marginal` will make a new plot for the ABC-SMC-(D)RF result.
`abc_results`	An ABC-SMC-(D)RF result. Will be plotted out by the function.
`parameters_truth`	A dataframe containing true values of parameters from the ground-truth distributions. If provided, the function will plot the true values or distributions of parameters.
`parameters_labels`	A dataframe containing labels in the plots for corresponding parameters. If provided, parameter labels will be exhibited on the plots' axes.

`statistics_labels`
: A dataframe containing labels in the plots for corresponding statistics. If provided, statistics labels will exhibit on the plots' axes.

`xlimit`
: A dataframe containing the maximum and minimum bounds for parameters. If provided, the x-axis will be scaled by them.

`plot_statistics`
: A logic variable (plot_statistics = FALSE by default). If plot_statistics = TRUE, the marginal distributions in each iteration for corresponding statistics will also be output.

`plot_hist`
: A logic variable (plot_hist = FALSE by default). If plot_hist = TRUE, marginal distributions will be plotted in histograms.

`plot_hist_point`
: A logic variable (plot_hist_point = FALSE by default). If plot_hist_point = TRUE, marginal distributions will be plotted in histograms with points in the middle.

`alpha`
: The numeric number to modify transparency. Default is 0.3.

`plot_prior`
: A logic variable (plot_prior = FALSE by default) If plot_prior = TRUE, the prior distribution will be plotted out.

Value

A list of `ggplot2` objects containing the marginal plots results of posterior distributions. The user can use the function to compare ABC-SMC-(D)RF marginal plots with the marginal posterior distribution(s) of other ABC-SMC-(D)RF result(s).

See Also

`smcrf`

`plot_compare_qqplot` *Plot and compare marginal quantile-quantile plots from ABC-SMC-(D)RF result*

Description

`plot_compare_qqplot` plots the marginal quantile-quantile plots for inferred parameters and parameters from ground-truth distributions.

Usage

```
plot_compare_qqplot(
  plots = NULL,
  abc_results,
  parameters_truth,
  parameters_labels = NULL,
  lims = NULL
)
```

Arguments

plots	An existed ABC-SMC-(D)RF quantile-quantile plots result. If provided, plot_compare_qqplot will plot the quantile-qantile plot for inferred parameters in the new ABC-SMC-(D)RF result and compare it with provided quantile-quantile plots result. If plots = NULL, plot_compare_qqplot will make a new quantile-quantile plot for inferred parameters in the ABC-SMC-(D)RF result and true parameters.
abc_results	An ABC-SMC-(D)RF result. The function will plot the quantile-quantile plot between the inferred parameters from the ABC-SMC-(D)RF result and true parameters.
parameters_truth	A dataframe containing true values of parameters from the ground-truth distributions.
parameters_labels	A dataframe containing labels in the plots for corresponding parameters. If provided, parameter labels will be exhibited on the plots' axes.
lims	A dataframe containing the maximum and minimum bounds for parameters. If provided, x-axis and y-axis will be scaled by them.

Value

A list of ggplot2 objects containing the quantile-quantile plots results. The user can use the function to compare ABC-SMC-(D)RF quantile-quantile plots with the quantile-quantile plots of other ABC-SMC-(D)RF result(s).

Examples

```
#    parameters_truth is optional here!
parameters_truth <- data.frame(
    theta = rnorm(10000, mean = 2, 1)
)
#    Dataframe containing the parameter labels
parameters_labels <- data.frame(
    parameter = c("theta"),
    label = c(deparse(expression(theta)))
)
#    Dataframe containing the x-axis bounds
xlimit <- data.frame(
    parameter = c("theta"),
    min = c(1),
    max = c(20)
)
```

---

plot_smcrf_joint *Plot joint distribution(s) of each iteration from ABC-SMC-(D)RF result*

Description

plot_smcrf_joint plots the joint distribution(s) of two parameters for each iteration from an Approximate Bayesian Computation sequential Monte Carlo via random forest result.

Usage

```
plot_smcrf_joint(
  smcrf_results,
  parameters_truth = NULL,
  parameters_labels = NULL,
  lims = NULL,
  nBins = 5
)
```

Arguments

smcrf_results	An ABC-SMC-(D)RF result containing the inference distributions of parameters from each iteration.
parameters_truth	A dataframe containing true values of parameters from the ground-truth distributions. If provided, the function will plot the true values or distributions of parameters.
parameters_labels	A dataframe containing labels in the plots for corresponding parameters. If provided, parameter labels will be exhibited on the plots' axes.
lims	A dataframe containing the maximum and minimum bounds for parameters. If provided, x-axis and y-axis will be scaled by them.
nBins	Number of contour bins shown in the plot. Default is 5.

See Also

smcrf

Examples

```
lims <- data.frame(
    parameter = c("theta", "mu"),
    min = c(0, 0),
    max = c(10, 10)
)
parameters_truth <- data.frame(
         theta = rnorm(10000, mean = 2, sd = 1),
         mu = rnorm(10000, mean = 1, sd = 2)
)
```

---

plot_smcrf_marginal *Plot distribution(s) in each iteration from ABC-SMC-(D)RF result*

Description

plot_smcrf_marginal plot the marginal distribution(s) for each iteration from an Approximate Bayesian Computation sequential Monte Carlo via random forest result.

Usage

```
plot_smcrf_marginal(
  smcrf_results,
  parameters_truth = NULL,
  parameters_labels = NULL,
  statistics_labels = NULL,
  plot_statistics = FALSE,
  xlimit = NULL,
  alpha = 0.3,
  plot_hist = FALSE
)
```

Arguments

smcrf_results	An ABC-SMC-(D)RF result containing the inference distributions of parameters from each iteration.
parameters_truth	A dataframe containing true values of parameters from the ground-truth distributions. If provided, the function will plot the true values or distributions of parameters.
parameters_labels	A dataframe containing labels in the plots for corresponding parameters. If provided, parameter labels will be exhibited on the plots' axes.
statistics_labels	A dataframe containing labels in the plots for corresponding statistics. If provided, statistics labels will exhibit on the plots' axes.
plot_statistics	A logic variable (plot_statistics = FALSE by default). If plot_statistics = TRUE, the marginal distributions in each iteration for corresponding statistics will also be output.
xlimit	A dataframe containing the maximum and minimum bounds for parameters. If provided, the x-axis will be scaled by them.
alpha	The numeric number to modify transparency. Default is 0.3.
plot_hist	A logic variable (plot_hist = FALSE by default). If plot_hist = TRUE, marginal distributions will be plotted in histograms.

See Also

smcrf

Examples

```
#    Dataframe containing the true parameters
parameters_truth <- data.frame(
    theta = 2 # this will plot a vertical line
)
if you prefer to have a histogram of true posterior
parameters_truth <- data.frame(
    theta = rnorm(10000, mean = 2, 1)
)
#    Dataframe containing the parameter labels
parameters_labels <- data.frame(
```

```
    parameter = c("theta"),
    label = c(deparse(expression(theta)))
)
#    Dataframe containing the x-axis bounds
xlimit <- data.frame(
    parameter = c("theta"),
    min = c(1),
    max = c(20)
)
```

---

smcrf *Approximate Bayesian Computation sequential Monte Carlo via random forests*

Description

smcrf uses random forests to find the posterior distribution(s) for one or more parameters in a model. It implements the sequential Monte Carlo framework, where each iteration uses either ABC-RF (functions regAbcrf and predict in R package abcrf) or ABC-DRF (functions drf and predict in R package drf) to update the posterior distribution(s).

Usage

```
smcrf(
  method = "smcrf-single-param",
  statistics_target = NULL,
  statistics_selection = NULL,
  smcrf_results = NULL,
  model,
  rprior,
  dprior,
  perturbation = "Gaussian",
  perturbation_parameters = NULL,
  nParticles,
  model_redo_if_NA = FALSE,
  parallel = FALSE,
  save_model = TRUE,
  save_rds = FALSE,
  filename_rds = "ABCSMCDRF.rds",
  ...
)
```

Arguments

method	Random forest method to implement in each iteration ("smcrf-single-param" by default). method = "smcrf-single-param": implements ABC-RF for each parameter and results in their marginal posterior distributions. method = "smcrf-multi-param": implements ABC-DRF for all parameters and results in the joint posterior distribution.
statistics_target	A dataframe containing statistics from data. Column names are the statistics IDs. smcrf only supports one row of statistics. If there are multiple observations, we recommend applying smcrf to each row individually.

statistics_selection
: A dataframe indicating selection of statistics for fitting individual parameters (only works for method "smcrf-single-param"; NULL by default). Each column's name matches one statistic ID, and each row's name matches one parameter ID. The value is 1 if the statistic is used for the parameter, 0 otherwise.

smcrf_results
: An existing ABC-SMC-RF result. If provided, smcrf will continue ABC-SMC-RF from the last iteration of the previous run.

model
: Model for the statistics. The function must take two inputs: a dataframe parameters and logic variable parallel. The model must output a reference table, where each row contains parameters for each simulation and corresponding statistics. The column names of the reference table must match the parameter and statistics IDs.

rprior
: Function to generate particles from the prior distribution. The function must take one input: Nparameters, the number of particles to generate. The output is a dataframe where column names match parameter IDs, and each row contains one parameter set.

dprior
: Function to compute the prior density. The function must take two inputs: parameters and parameter_id. The dataframe parameters contains parameter sets in each row, with column names as parameter IDs. The parameter_id is either "all" or one of the parameter IDs. The output is a vector of prior probabilities corresponding to rows in parameters, either for the parameter indicated by parameter_id or jointly for all parameters (if parameter_id = "all").

perturbation
: Perturbation method for the parameters. smcrf supports perturbation = "Gaussian" (default) or "Uniform".

perturbation_parameters
: A dataframe containing the parameters for the perturbation. Each row corresponds to one iteration, and each column corresponds to one parameter (the column names must match parameter_ids). The values are the normal distribution variances (for perturbation = "Gaussian") or ranges (for perturbation = "Uniform").

nParticles
: A vector of particle counts. Each entry indicates the number of simulations (e.g. particles) in the corresponding iteration.

model_redo_if_NA
: A logic variable (FALSE by default). If model_redo_if_NA = TRUE, the particles where model returns NA will be simulated again.

parallel
: A logic variable (FALSE by default). If parallel = TRUE, the ABC-RF functions will be computed in parallel.

save_model
: A logic variable (FALSE by default). If save_model = TRUE, the random forest will be saved in smcrf's output.

save_rds
: A logic variable (FALSE by default). If save_rds = TRUE, the ABC-SMC-RF results will be saved in an rds file.

filename_rds
: A string ("ABCSMCDRF.rds" by default). If save_rds = TRUE, the output from ABC-SMC-(D)RF will be saved in a file with this name.

...
: Additional arguments to be passed to abcrf or drf.

Value

An object smcrf_results containing the results of the inference. If the posterior distributions have not converged to a satisfactory level, the user may continue with smcrf(smcrf_results = smcrf_results, ...), in which case ABC-SMC-(D)RF will continue iterating from the last run in smcrf_results.

Examples

```
library(abcsmcrf)
#------------------------------------------------------------------------
#--------------------------ABC-SMC-RF for a model with one parameter
#------------------------------------------------------------------------
#    Data to be fitted consists of two statistics s1 and s2
statistics_target <- data.frame(s1 = 0, s2 = 2)
#    We define a parametrized model for the statistics
model <- function(parameters) {
    statistics <- data.frame(
        s1 = parameters$theta - 1 + runif(nrow(parameters), -0.1, 0.1),
        s2 = parameters$theta + 1 + runif(nrow(parameters), -0.1, 0.1)
    )
    cbind(parameters, statistics)
}
#    and the perturbation parameters for a uniform perturbation
perturbation_parameters <- data.frame(
     theta = rep(0.1, 2) # vector length is equal to number of ABC-SMC-(D)RF iterations
)
#    We then define rprior and dprior
rprior <- function(Nparameters){
     theta <- runif(Nparameters, -10, 10)
     return(data.frame(theta = theta))
}
dprior <- function(parameters, parameter_id = "theta"){
     return(rep(1/20, nrow(parameters)))
}
#    Finally, we run ABC-SMC-RF with 2 iterations, each with 1000 particles
smcrf_results <- smcrf(
    method = "smcrf-single-param",
    statistics_target = statistics_target,
    model = model,
    rprior = rprior,
    dprior = dprior,
    perturbation = "Uniform",
    perturbation_parameters = perturbation_parameters,
    nParticles = c(1000, 1000),
)
#    Now we examine the posterior distribution of theta
posterior_iteration <- paste0("Iteration_", (smcrf_results$nIterations + 1))
posterior_theta <- smcrf_results[[posterior_iteration]]$parameters$theta
#    We look at the posterior mean of theta
theta_mean <- mean(posterior_theta)
print(theta_mean)
#    Notice that the mean is close to 1, the true value of theta.
#    We can also look at the posterior variance of theta
theta_var <- var(posterior_theta)
print(theta_var)
#   We can also continue the ABC-SMC-RF run if the posterior convergence is not satisfactory
smcrf_results <- smcrf(
    method = "smcrf-single-param",
    smcrf_results = smcrf_results,
    model = model,
    rprior = rprior,
    dprior = dprior,
    perturbation = "Uniform",
```

```
    perturbation_parameters = perturbation_parameters,
    nParticles = c(1000, 1000),
)
#    We can look again at the posterior mean and variance of theta
posterior_iteration <- paste0("Iteration_", (smcrf_results$nIterations + 1))
posterior_theta <- smcrf_results[[posterior_iteration]]$parameters$theta
theta_mean <- mean(posterior_theta)
print(theta_mean)
theta_var <- var(posterior_theta)
print(theta_var)
#    and notice whether there is any improvement in the posterior distribution
#    We can continue the runs of ABC-SMC-(D)RF similarly for the examples below
#------------------------------------------------------------------------
#---------------------ABC-SMC-RF for a model with multiple parameters
#------------------------------------------------------------------------
#    Data to be fitted consists of two statistics s1 and s2
statistics_target <- data.frame(s1 = 4, s2 = 4)
#    We then define a parametrized model for the statistics
model <- function(parameters) {
    statistics <- data.frame(
        s1 = parameters$mu + parameters$theta + runif(nrow(parameters), -0.1, 0.1),
        s2 = parameters$mu * parameters$theta + runif(nrow(parameters), -0.1, 0.1)
    )
    cbind(parameters, statistics)
}
#    and the perturbation parameters
perturbation_parameters <- data.frame(
     theta = rep(0.1, 3),
     mu = rep(0.1, 3)
)
#    We define the rprior and dprior functions
rprior <- function(Nparameters){
     theta <- runif(Nparameters, -10, 10)
     mu <- runif(Nparameters, -10, 10)
     return(data.frame(theta = theta, mu = mu))
}
dprior <- function(parameters, parameter_id = "all"){
     probs <- rep(1, nrow(parameters))
     if (parameter_id %in% c("all", "theta")){
         probs <- probs * dunif(parameters[["theta"]], -10, 10)
     }
     if (parameter_id %in% c("all", "mu")){
         probs <- probs * dunif(parameters[["mu"]], -10, 10)
     }
     return(probs)
}
#    Finally, we run ABC-SMC-RF with 3 iterations, each with 1000 particles
smcrf_results <- smcrf(
    method = "smcrf-single-param",
    statistics_target = statistics_target,
    model = model,
    rprior = rprior,
    dprior = dprior,
    perturbation = "Uniform",
    perturbation_parameters = perturbation_parameters,
    nParticles = c(1000, 1000, 1000),
)
```

```
#    Now we examine the posterior distribution of each parameter
posterior_iteration <- paste0("Iteration_", (smcrf_results$nIterations + 1))
posterior_params <- smcrf_results[[posterior_iteration]]$parameters
posterior_means <- colMeans(posterior_params)
posterior_vars <- var(posterior_params)
print(posterior_means)
print(posterior_vars)
#------------------------------------------------------------------------
#-------------------------------ABC-SMC-DRF for a multivariate model
#------------------------------------------------------------------------
#    Data to be fitted consists of two statistics s1 and s2
statistics_target <- data.frame(s1 = 9, s2 = 18)
#    We then define a parametrized model for the statistics
model <- function(parameters) {
    statistics <- data.frame(
        s1 = parameters$mu + parameters$theta + runif(nrow(parameters), -0.1, 0.1),
        s2 = parameters$mu * parameters$theta + runif(nrow(parameters), -0.1, 0.1)
    )
    cbind(parameters, statistics)
}
#    and the perturbation parameters
perturbation_parameters <- data.frame(
     theta = rep(0.1, 3),
     mu = rep(0.1, 3)
)
#    We define the rprior and dprior functions
rprior <- function(Nparameters){
     theta <- runif(Nparameters, -10, 10)
     mu <- runif(Nparameters, -10, 10)
     return(data.frame(theta = theta, mu = mu))
}
dprior <- function(parameters, parameter_id = "all"){
     probs <- rep(1, nrow(parameters))
     if (parameter_id %in% c("all", "theta")){
         probs <- probs * dunif(parameters[["theta"]], -10, 10)
     }
     if (parameter_id %in% c("all", "mu")){
         probs <- probs * dunif(parameters[["mu"]], -10, 10)
     }
     return(probs)
}
#    Finally, we run ABC-SMC-DRF with 3 iterations, each with 1000 particles
smcrf_results <- smcrf(
    method = "smcrf-multi-param",
    statistics_target = statistics_target,
    model = model,
    rprior = rprior,
    dprior = dprior,
    perturbation = "Uniform",
    perturbation_parameters = perturbation_parameters,
    nParticles = c(1000, 1000, 1000),
)
#    Now we examine the posterior distribution of each parameter
posterior_iteration <- paste0("Iteration_", (smcrf_results$nIterations + 1))
posterior_params <- smcrf_results[[posterior_iteration]]$parameters
posterior_means <- colMeans(posterior_params)
posterior_vars <- var(posterior_params)
```

```
print(posterior_means)
print(posterior_vars)
```

Index