Package ‘flimo’

April 2, 2022

Type Package
Title Fixed Landscape Inference MethOd
Version 0.1.1
Author Sylvain Moinard
Maintainer Sylvain Moinard <sylvain.moinard@univ-grenoble-alpes.fr>
Description Likelihood-free inference method for stochastic models.
Uses a deterministic optimizer on simple simulations of the model
that are performed with a prior drawn randomness by applying the inverse transform method.
Is designed to work on its own and also by using the Julia package Jflimo.
See the git page of the project: <https://metabarcoding.org/flimo>.
Imports ggplot2, gridExtra, JuliaConnectoR
License CeCILL
Encoding UTF-8
RoxygenNote 7.1.2
SystemRequirements Julia >= 1.0
VignetteBuilder knitr
Suggests knitr, rmarkdown
NeedsCompilation no
Repository CRAN
Date/Publication 2022-04-01 23:40:06 UTC

R topics documented:

check_simulator ......................................................... 2
flimobjective ........................................................ 3
flimoptim ............................................................... 4
flimoptim_Julia ......................................................... 6
flimoptim_R ............................................................. 8
julia_load ............................................................... 10
julia_setup ............................................................. 10
plot.flimo_result ...................................................... 11
Description

Run simulations to catch random variations. Warning: does not check it formally. Warning: does not check if quantiles are used several times.

Usage

```r
check_simulator(
  simulatorQ,
  ndraw,
  Theta_lower = 0,
  Theta_upper = 1,
  ntheta = 5,
  nruns = 3
)
```

Arguments

- `simulatorQ` Function of type `simulatorQ(Theta, quantiles)` where Theta is the parameter set for the simulations and quantiles are drawn in U(0,1).
- `ndraw` Integer. Number of random variables to draw for one simulation of the model.
- `Theta_lower` 1D numeric array. Lower bounds of Theta parameters.
- `Theta_upper` 1D numeric array. Upper bounds of Theta parameters.
- `ntheta` Integer. Number of Theta parameters to test.
- `nruns` Integer. For each Theta, number of simulations to run.

Value

Boolean. True if no random effect was detected, False else.

Examples

```r
simulatorQ <- function(Theta, quantiles){
  qpois(quantiles, lambda = Theta)
}
check_simulator(simulatorQ, 5, Theta_lower = 50, Theta_upper = 150)
```
Description

Computes the summary statistics between simulations w.r.t. Theta and data. This function is to be minimized by flimoptim.

Usage

flimobjective(Theta, quantiles, data, sumstats, simulatorQ)

Arguments

- **Theta**: 1D array. parameters for the simulations.
- **quantiles**: 2D array containing values following U(0,1). Row number = number of simulations. Column number = number of random variables to draw in one simulation.
- **data**: 1D array containing the observations.
- **sumstats**: Function computing the distance between simulations and data of form sumstats(simulations, data) where simulations : 2D array and data : 1D array. ncol(simulations) = length(data) mandatory.
- **simulatorQ**: Function of type simulatorQ(Theta, quantiles) where Theta is the parameter set for the simulations and quantiles are drawn in U(0,1). See README for details.

Value

Numeric value. Distance between summary statistics of data and simulations w.r.t. Theta.

Examples

```r
quantiles <- matrix(runif(50), nrow = 10)
data <- rep(100, 5)

sumstats <- function(simulations, data){
  mean_simu <- mean(rowMeans(simulations))
  mean_data <- mean(data)
  (mean_simu-mean_data)^2
}
simulatorQ <- function(Theta, quantiles){
gpois(quantiles, lambda = Theta)
}

flimobjective(100, quantiles, data, sumstats, simulatorQ)
```
flimoptim  flimoptim

Description
Computes several parameter inferences with R optimizer or Julia optimizer in a full Julia mode. In R mode (default): L-BFGS-B optimization. In Julia mode: either IPNewton with or without Automatic Differentiation or Brent optimization.

Usage
flimoptim(
data,
ndraw,
sumstats,
simulatorQ,
obj = NULL,
nsim = 10,
ninfer = 1,
lower = 0,
upper = 1,
Theta0 = (lower + upper)/2,
randomTheta0 = FALSE,
mode = c("R", "Julia"),
AD = TRUE,
method = "",
maxit = 1000,
time_lim = NaN,
factr = 1e+07,
p gtol = 0,
xtol = 0,
ftol = 0,
gtol = 1e-08,
reltol = sqrt(.Machine$double.eps),
abstol = .Machine$double.eps,
show_trace = FALSE,
store_trace = FALSE,
store_quantiles = FALSE,
par_names = NULL,
load_julia = FALSE
)

Arguments

data 1D array containing the observations.
ndraw Integer. Number of random variables to draw for one simulation of the model.
sumstats

Summary statistics to measure distance between simulations and data. In R mode: R function of type `sumstats(simulations, data)` where `simulations` is a 2D array and `data` is a 1D array. `ncol(simulations) = length(data)` is mandatory. In Julia mode: a string containing the script of the Julia function `sumstats(simulations, data)`. The name "sumstats" is mandatory.

simulatorQ

Simulator of the stochastic process with fixed quantiles (see README) or a string (in mode "Julia") containing the script of the Julia function `simulatorQ(Theta, quantiles)`. In Julia mode, the name "simulatorQ" is mandatory. Theta is the parameter set for the simulations and quantiles are drawn in U(0,1).

obj

Objective function to minimize. Default: is directly computed from `sumstats` and `simulatorQ`. Either an R function of type `objective(Theta, quantiles)` (in mode "R") or a string (in mode "Julia") containing the script of the Julia function `julia_obj(Theta, quantiles)`. Warning: could be tricky if mode = "Julia" to call data. In Julia mode, the name "julia_obj" is mandatory.

nsim

Integer. Number of simulations to run for each step of the optimization algorithm. Computation time grows linearly with this number. Default to 10.

ninfer

Integer. Number of independent inferences to run. Default to 1.

lower

1D array. Lower bounds for parameters. Same length as `upper`.

upper

1D array. Upper bounds for parameters. Same length as `lower`.

Theta0

1D array. Initial values of the parameters. Default: `mean(lower, upper)`.

randomTheta0

Boolean. If True, `Theta0` is randomly drawn between lower and upper bounds.

mode

String. "R" (default) or "Julia". See README.

AD

Boolean. Only in Julia mode, uses Automatic Differentiation with IPNewton method. Default to true.

method

String. In Julia mode, allows to choose the optimization method: "Brent", "IPNewton". Default: IPNewton.

maxit

Integer. Max number of iterations during optimization. Default to 1000.

time_lim

Float. Time limit in second for each inference. Default to no limit. Not available for R mode and Brent method in Julia mode.

factr


pgtol

Float. In R-mode: control parameter for L-BFGS-B method in `stats::optim`. Default to 0.

xtol

Float. In Julia mode with IPNewton method: xtol option in `Optim.Options`. Default to 0.

ftol


gtol


reltol

Float. In Julia mode with Brent method: reltol of `Optim.optimize`. Default is `sqrt(Machine$double_eps)`, about 1e-8.

abstol

Float. In Julia mode with Brent method: abstol of `Optim.optimize`. Default is `Machine$double_eps`, about 1e-16.
show_trace  Boolean. If true, shows standard trace. Default to false.
store_trace  Boolean. If true, stores standard trace as an array of strings. Default to false. Not available for R mode.
store_quantiles  Boolean. If true, stores every quantiles used for inference, to reproduce the results. Default to false.
par_names  vector of names for parameters. Default is "par1", ..., "parn".
load_julia  Boolean. If true, run julia_load. It can take few seconds. Default to False.

Value
Object of class flimo_result (list) (converted from Julia object in Julia mode) containing every information about convergence results.

Examples
```
data <- rep(100, 5)
sumstats <- function(simulations, data){
    mean_simu <- mean(rowMeans(simulations))
    mean_data <- mean(data)
    (mean_simu-mean_data)^2
}
simulatorQ <- function(Theta, quantiles){
go pois(quantiles, lambda = Theta)
}
flimoptim(data, 5, sumstats, simulatorQ,
nsim = 10,
lower = 50,
upper = 150,
method = "Brent")
```

Description
Computes several parameter inferences with Julia optimizer and either IPNewton with or without Automatic Differentiation or Brent method.
Usage

```r
flimoptim_Julia(
  data,
  ndraw,
  sumstats,
  simulatorQ,
  julia_obj = NULL,
  nsim = 10,
  ninfer = 1,
  lower = 0,
  upper = 1,
  Theta0 = (lower + upper)/2,
  randomTheta0 = FALSE,
  AD = TRUE,
  method = "",
  maxit = 1000,
  time_lim = NULL,
  xtol = 0,
  ftol = 0,
  gtol = 1e-08,
  reltol = sqrt(.Machine$double.eps),
  abstol = .Machine$double.eps,
  show_trace = FALSE,
  store_trace = FALSE,
  store_quantiles = FALSE,
  par_names = NULL,
  load_julia = FALSE
)
```

Arguments

data : 1D array containing the observations.

ndraw : Integer. Number of random variables to draw for one simulation of the model.

sumstats : String containing the script of the Julia function sumstats(simulations, data). The name "sumstats" is mandatory.

simulatorQ : Simulator of the stochastic process with fixed quantiles (see README). simulatorQ(Theta, quantiles). The name "simulatorQ" is mandatory. Theta is the parameter set for the simulations and quantiles are drawn in U(0,1).

julia_obj : Objective function to minimize. Default : is directly computed from sumstats and simulatorQ. String containing the script of the Julia function julia_obj(Theta, quantiles). The name "julia_obj" is mandatory.

nsim : Integer. Number of simulations to run for each step of the optimization algorithm. Computation time grows linearly with this number. Default to 10.

ninfer : Integer. Number of independent inferences to run. Default to 1.
lower
1D array. Lower bounds for parameters. Same length as upper.
upper
1D array. Upper bounds for parameters. Same length as lower.
Theta0
1D array. Initial values of the parameters. Default: mean(lower, upper).
randomTheta0
Boolean. If True, Theta0 is randomly drawn between lower and upper bounds.
AD
Boolean. Only in Julia mod, uses Automatic Differentiation with IPNewton method. Default to true.
method
String. Allows to choose the optimization method: "Brent", "IPNewton". Default: IPNewton.
maxit
Integer. Max number of iterations during optimization. Default to 1000.
time_lim
Float. Time limit in second for each inference. Default to no limit. Not available for Brent method.
xtol
Float. With IPNewton method: xtol option in Optim.Options. Default to 0.
ftol
Float. With IPNewton method: ftol option in Optim.Options. Default to 0.
gtol
reltol
abstol
show_trace
Boolean. If true, shows standard trace. Default to false.
store_trace
Boolean. If true, stores standard trace as an array of strings. Default to false. Not available for R mod.
store_quantiles
Boolean. If true, stores every quantiles used for inference, to reproduce the results. Default to false.
par_names
vector of names for parameters. Default is "par1", ..., "parn".
load_julia
Boolean. If true, run julia_load. It can take few seconds. Default to False.

Value
Object of class flimo_result (list) converted from Julia object containing every information about convergence results.

Description
Computes several parameter inferences with R optimizer (method L-BFGS-B).
flimoptim_R

Usage
flimoptim_R(
  data,
  ndraw,
  sumstats,
  simulatorQ,
  obj = NULL,
  nsim = 10,
  ninfer = 1,
  lower = 0,
  upper = 1,
  Theta0 = (lower + upper)/2,
  randomTheta0 = FALSE,
  maxit = 1000,
  factr = 1e+07,
  pgtol = 0,
  show_trace = FALSE,
  store_quantiles = FALSE,
  par_names = NULL
)

Arguments
data  1D array containing the observations.
ndraw  Integer. Number of random variables to draw for one simulation of the model.
sumstats  Summary statistics to measure distance between simulations and data. R function of type sumstats(simulations, data) where simulations : 2D array and data : 1D array. ncol(simulations) = length(data) mandatory.
simulatorQ  Simulator of the stochastic process with fixed quantiles (see README). Theta is the parameter set for the simulations and quantiles are drawn in U(0,1).
obj  Objective function to minimize. Default : is directly computed from sumstats and simulatorQ. R function of type objective(Theta, quantiles)
nsim  Integer. Number of simulations to run for each step of the optimization algorithm. Computation time grows linearly with this number. Default to 10.
ninfer  Integer. Number of independent inferences to run. Default to 1.
lower  1D array. Lower bounds for parameters. Same length as upper.
upper  1D array. Upper bounds for parameters. Same length as lower.
Theta0  1D array. Initial values of the parameters. Default : mean(lower, upper).
randomTheta0  Boolean. If True, Theta0 is randomly drawn between lower and upper bounds.
maxit  Integer. Max number of iterations during optimization. Default to 1000.
factr  Float. Control parameter for L-BFGS-B method in stats::optim. Default to 1e7.
pgtol  Float. Control parameter for L-BFGS-B method in stats::optim. Default to 0.
show_trace  Boolean. If true, shows standard trace. Default to false.
store_quantiles

Boolean. If true, stores every quantiles used for inference, to reproduce the results.

par_names

vector of names for parameters. Default is "par1", ..., "parn".

Value

Object of class flimo_result (list) containing every information about convergence results.

julia_load

Description

Load needed Julia packages. Run to use Jflimo.

Usage

julia_load()

Value

Boolean. True if load is done correctly

julia_setup

Description

Checks installation of Julia and install the needed packages. May take little time to run. Only run the first time you use Jflimo.

Usage

julia_setup()

Value

Boolean. True if correct setup, False else.
Description

Shows the plots for most important inference results. Default only shows normalized boxplots for each inferred parameter.

Usage

```r
## S3 method for class 'flimo_result'
plot(
  x,
  y, ...
  hist = FALSE,
  bins = 1 + as.integer(nrow(x$minimizer)^{(1/3)}),
  par_minimum = FALSE,
  pairwise_par = FALSE,
  boxplot = TRUE,
  par_names = NULL
)
```

Arguments

- **x**: Object of class `flimo_result`.
- **y**: unused generic argument.
- **...**: optional args for generic method
- **hist**: Boolean. If True, plots the histogram of each inferred parameter. Default to false.
- **bins**: Integer. Number of bins if hist is True.
- **par_minimum**: Boolean. If True, plots each inferred parameter by reached minimum. Default to false.
- **pairwise_par**: Boolean. If True, plots each pairs of inferred parameters. Default to false.
- **boxplot**: Boolean. If True, plots the boxplots of each inferred parameter scaled by their mean. Default to true.
- **par_names**: Vector of names for parameters. Default is "par1", ..., "parn".

Value

Nothing. Prints the asked ggplot objects.
Description

Plot of objective = f(theta_index).

Usage

plot_objective(
  ndraw,
  nsim,
  data,
  sumstats,
  simulatorQ,
  quantiles = NULL,
  obj = NULL,
  index = NULL,
  other_param = NULL,
  lower = 0,
  upper = 1,
  dim2 = TRUE,
  visualize_min = TRUE,
  plot_legend = TRUE,
  add_to_plot = NULL
)

Arguments

ndraw  Integer. Number of random variables to draw for one simulation of the model.
nsim  Integer. Number of simulations to run for each step of the optimization algorithm. Computation time grows linearly with this number. Default to 10.
data  1D array containing the observations.
sumstats  Function computing the distance between simulations and data of form sumstats(simulations, data) where simulations : 2D array and data : 1D array. ncol(simulations) = length(data) mandatory.
simulatorQ  Function of type simulatorQ(Theta, quantiles) where Theta is the parameter set for the simulations and quantiles are drawn in U(0,1).
quantiles  2D array containing values following U(0,1). Row number = number of simulations. Column number = number of random variables to draw in one simulation.
obj  objective function of type objective(Theta). Default : directly computed with "sumstats" and "simulatorQ".
index  Integer. Index of the moving parameter.
other_param  Other parameters of the model. If NULL: assume 1D-model. If numeric: 2D-model, one curve. If 1D-array and dim2 is True (default): 2D-model, one curve by value in other_param. If 1D-array and dim2 is False or 2D-array: (n>2)D-model, one curve by row in other_param. If your model has n>2 dimensions, you should define other_param as a matrix even if you have only one parameter set to test (with as.matrix(t(vect_param)) where vect_param is a 1D-array).

lower   Numeric. Lower value of the plot.
upper   Numeric. Upper value of the plot.
dim2   Boolean. True if model is 2-dimensional.
visualize_min  Boolean. If True, show explicitly the minimum point.
plot_legend  Boolean. If True (default), plots the legend.
add_to_plot  ggplot object. If not NULL, will add all curves/points on previous plot instead of creating a new one. Does not change title/labels/limits defined in previous plot.

Value

ggplot object representing the objective function to be minimized.

Examples

data <- rep(100, 5)

sumstats <- function(simulations, data){
  mean_simu <- mean(rowMeans(simulations))
  mean_data <- mean(data)
  (mean_simu-mean_data)^2
}

simulatorQ <- function(Theta, quantiles){
  qpois(quantiles, lambda = Theta)
}

plot_objective(5, 10, data, sumstats, simulatorQ, lower = 0, upper = 200)

Description

Prints most important information about inference results.

Usage

## S3 method for class 'flimo_result'
print(x, ...)
Arguments

x  Object of class flimo_result from any mode/method algorithm of the flimo package.

...  optional args for generic method

Value

String containing most important information about argument of class flimo_result.

Description

Most important information about inference results.

Usage

```r
## S3 method for class 'flimo_result'
summary(object, ...)
```

Arguments

object  Object of class flimo_result from any mode/method algorithm of the flimo package.

...  optional args for generic method summary

Value

List containing most important information about argument of class flimo_result.
Index

check_simulator, 2
flimobjective, 3
flimoptim, 4
flimoptim_Julia, 6
flimoptim_R, 8
julia_load, 10
julia_setup, 10
plot.flimo_result, 11
plot_objective, 12
print.flimo_result, 13
summary.flimo_result, 14