Package 'calibrar'

October 12, 2022

Version 0.2.0

Title Automated Parameter Estimation for Complex (Ecological) Models

Description Automated parameter estimation for complex (ecological) models in R.
This package allows the parameter estimation or calibration of complex models,
including stochastic ones. It is a generic tool that can be used for fitting
any type of models, especially those with non-differentiable objective functions.
It supports multiple phases and constrained optimization.
It implements maximum likelihood estimation methods and automated construction
of the objective function from simulated model outputs.
See <http://roliveros-ramos.github.io/calibrar> for more details.

Depends R (>= 2.15)

Imports cmaes, optimx, foreach, parallel, stats, utils

Suggests deSolve

License GPL-2

URL http://roliveros-ramos.github.io/calibrar

BugReports https://github.com/roliveros-ramos/calibrar/issues

ByteCompile TRUE

NeedsCompilation no

Author Ricardo Oliveros-Ramos [aut, cre]

Maintainer Ricardo Oliveros-Ramos <ricardo.oliveros@gmail.com>

Repository CRAN

Date/Publication 2016-02-17 13:43:47

R topics documented:

`calibrar-package` *Automated Calibration for Complex (Ecological) Models*

Description

Automated Calibration for Complex (Ecological) Models

Author(s)

Ricardo Oliveros-Ramos Maintainer: Ricardo Oliveros-Ramos <ricardo.oliveros@gmail.com>

References

calibrar: an R package for the calibration of ecological models (Oliveros-Ramos and Shin 2014)

Examples

```
## Not run:
require(calibrar)
set.seed(880820)
path = NULL # NULL to use the current directory
# create the demonstration files
demo = calibrarDemo(model="PoissonMixedModel", L=5, T=100)
# get calibration information
calibrationInfo = getCalibrationInfo(path=demo$path)
# get observed data
observed = getObservedData(info=calibrationInfo, path=demo$path)
# read forcings for the model
forcing = read.csv(file.path(demo$path, "master", "environment.csv"), row.names=1)
# Defining 'runModel' function
runModel = function(par, forcing) {
output = calibrar:::.PoissonMixedModel(par=par, forcing=forcing)
# adding gamma parameters for penalties
output = c(output, list(gammas=par$gamma))
return(output)
}
# real parameters
cat("Real parameters used to simulate data\n")
print(demo$par)
# objective functions
obj  = createObjectiveFunction(runModel=runModel, info=calibrationInfo,
                               observed=observed, forcing=forcing)
cat("Starting calibration...\n")
control = list(weights=calibrationInfo$weights, maxit=3.6e5) # control parameters
cat("Running optimization algorithms\n", "\t", date(), "\n")
```

```
cat("Running optim AHR-ES\n")
ahr = calibrate(par=demo$guess, fn=obj, lower=demo$lower, upper=demo$upper, control=control)
summary(ahr)

## End(Not run)
```

`calibrarDemo` *Demos for the calibrar package*

Description

Creates demo files able to be processed for a full calibration using the calibrar package

Usage

```
calibrarDemo(path = NULL, model = NULL, ...)
```

Arguments

`path`	Path to create the demo files
`model`	Model to be used in the demo files, see details.
`...`	Additional parameters to be used in the construction of the demo files.

Value

A list with the following elements:

`path`	Path were the files were saved
`par`	Real value of the parameters used in the demo
`constants`	Constants used in the demo

Author(s)

Ricardo Oliveros–Ramos

References

Oliveros-Ramos and Shin (2014)

Examples

```
## Not run:
require(calibrar)
set.seed(880820)
path = NULL # NULL to use the current directory
# create the demonstration files
demo = calibrarDemo(model="PoissonMixedModel", L=5, T=100)
# get calibration information
```

```
calibrationInfo = getCalibrationInfo(path=demo$path)
# get observed data
observed = getObservedData(info=calibrationInfo, path=demo$path)
# read forcings for the model
forcing = read.csv(file.path(demo$path, "master", "environment.csv"), row.names=1)
# Defining 'runModel' function
runModel = function(par, forcing) {
output = calibrar:::.PoissonMixedModel(par=par, forcing=forcing)
# adding gamma parameters for penalties
output = c(output, list(gammas=par$gamma))
return(output)
}
# real parameters
cat("Real parameters used to simulate data\n")
print(demo$par)
# objective functions
obj  = createObjectiveFunction(runModel=runModel, info=calibrationInfo,
                               observed=observed, forcing=forcing)
cat("Starting calibration...\n")
control = list(weights=calibrationInfo$weights, maxit=3.6e5) # control parameters
cat("Running optimization algorithms\n", "\t", date(), "\n")
cat("Running optim AHR-ES\n")
ahr = calibrate(par=demo$guess, fn=obj, lower=demo$lower, upper=demo$upper, control=control)
summary(ahr)

## End(Not run)
```

`calibrate` *Sequential parameter estimation for the calibration of models*

Description

This function performs the optimization of a function, possibly in sequential phases of increasing complexity, and it is designed for the calibration of a model, by minimizing the error function `fn` associated to it.

Usage

```
calibrate(par, fn, gr = NULL, ..., method = "default", lower = NULL,
  upper = NULL, control = list(), hessian = FALSE, phases = NULL,
  replicates = 1)
```

Arguments

`par`	A numeric vector. The length of the par argument defines the number of parameters to be estimated (i.e. the dimension of the problem).
`fn`	The function to be minimized.
`gr`	the gradient of `fn`. Ignored, added for portability with other optimization functions.

method	The optimization method to be used. The 'default' method is the AHR-ES (Oliveros & Shin, 2016). All the methods from stats::optim, optimx::optimx and cmaes::cma_es are available.
lower	Lower threshold value(s) for parameters. One value or a vector of the same length as par. If one value is provided, it is used for all parameters. NA means -Inf. By default -Inf is used (unconstrained).
upper	Upper threshold value(s) for parameters. One value or a vector of the same length as par. If one value is provided, it is used for all parameters. NA means Inf. By default Inf is used (unconstrained).
control	Parameter for the control of the algorithm itself, see details.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned? Currently not implemented.
phases	An optional vector of the same length as par, indicating the phase at which each parameter becomes active. If omitted, default value is 1 for all parameters, performing a single optimization.
replicates	The number of replicates for the evaluation of fn. The default value is 1. A value greater than 1 is only useful for stochastic functions.
...	Additional parameters to be passed to fn.

Details

In the control list, aggFn is a function to aggregate fn to a scalar value if the returned value is a vector. Some optimization algorithm can exploite the additional information provided by a vectorial output from fn.

Author(s)

Ricardo Oliveros-Ramos

Examples

```
calibrate(par=rep(NA, 5), fn=SphereN)
## Not run:
calibrate(par=rep(NA, 5), fn=SphereN, replicates=3)
calibrate(par=rep(0.5, 5), fn=SphereN, replicates=3, lower=-5, upper=5)
calibrate(par=rep(0.5, 5), fn=SphereN, replicates=3, lower=-5, upper=5, phases=c(1,1,1,2,3))
calibrate(par=rep(0.5, 5), fn=SphereN, replicates=c(1,1,4), lower=-5, upper=5, phases=c(1,1,1,2,3))

## End(Not run)
```

createObjectiveFunction *Create an objective function to be used with optimization routines*

Description

Create a new function, to be used as the objective function in the calibration, given a function to run the model within R, observed data and information about the comparison with data.

Usage

```
createObjectiveFunction(runModel, info, observed, aggFn = .weighted.sum,
  aggregate = FALSE, ...)
```

Arguments

runModel	Function to run the model and produce a list of outputs.
info	A data.frame with the information about the calibration, normally created with the getCalibrationInfo function. See details.
observed	A list of the observed variables created with the function getObservedData
aggFn	A function to aggregate fn to a scalar value if the returned value is a vector. Some optimization algorithm can explote the additional information provided by a vectorial output from fn
aggregate	boolean, if TRUE, a scalar value is returned using the aggFn.
...	More arguments passed to the runModel function.

Value

A function, integrating the simulation of the model and the comparison with observed data.

Author(s)

Ricardo Oliveros-Ramos

See Also

getObservedData, getCalibrationInfo.

getCalibrationInfo *Get information to run a calibration using the* calibrar *package.*

Description

A wrapper for `read.csv` checking column names and data types for the table with the calibration information.

Usage

```
getCalibrationInfo(path, file = "calibrationInfo.csv",
  stringsAsFactors = FALSE, ...)
```

Arguments

`path`	The path to look for the file.
`file`	The file with the calibration information, see details.
`stringsAsFactors`	To be passed to `read.csv`.
`...`	Additional arguments to `read.csv` function.

Value

A data.frame with the information for the calibration of a model, to be used with the `createObjectiveFunction` and `getObservedData`.

Author(s)

Ricardo Oliveros-Ramos

See Also

`createObjectiveFunction`, `getObservedData`.

getObservedData *Get observed data for the calibration of a model*

Description

Create a list with the observed data with the information provided by its main argument.

Usage

```
getObservedData(info, path, data.folder = "data", ...)
```

Arguments

info	A data.frame with the information about the calibration, normally created with the getCalibrationInfo function. See details.
path	Path to the directory to look up for the data.
data.folder	folder in the path containing the data.
...	Additional arguments to read.csv function to read the data files.

Value

A list with the observed data needed for a calibration, to be used in combination with the createObjectiveFunction.

Author(s)

Ricardo Oliveros-Ramos

See Also

createObjectiveFunction, getCalibrationInfo.

---

optimES *Optimization using Evolutionary Strategies*

Description

This function performs the optimization of a function using evolutionary strategies, by default the AHR-ES (Oliveros & Shin, 2015).

Usage

```
optimES(par, fn, gr = NULL, ..., lower = -Inf, upper = Inf,
  active = NULL, control = list(), hessian = FALSE, method = "default")
```

Arguments

par	A numeric vector. The length of the par argument defines the number of parameters to be estimated (i.e. the dimension of the problem).
fn	The function to be minimized.
gr	the gradient of fn. Ignored, added for portability with other optimization functions.
lower	Lower threshold value(s) for parameters. One value or a vector of the same length as par. If one value is provided, it is used for all parameters. NA means -Inf. By default -Inf is used (unconstrained).
upper	Upper threshold value(s) for parameters. One value or a vector of the same length as par. If one value is provided, it is used for all parameters. NA means Inf. By default Inf is used (unconstrained).

active	A boolean vector of the same length of par. If TRUE, the parameter is optimized, if FALSE the parameter is fixed to the value specified in par.
control	Parameter for the control of the algorithm itself, see details.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned? Currently not implemented.
method	The optimization method to be used. Currently, the only implemented is the 'default' method, corresponding to the AHR-ES (Oliveros & Shin, 2015).
...	Additional parameters to be passed to fn.

Author(s)

Ricardo Oliveros-Ramos

Examples

```
optimES(par=rep(1, 5), fn=SphereN)
```

SphereN *Sphere function with random noise*

Description

This function calculates the Euclidian distance from a point to the origin after a random displacement of it position.

Usage

```
SphereN(x, sd = 0.1, aggregate = TRUE)
```

Arguments

x	The coordinates of the point
sd	The standard deviation of the noise to be added to the position of x, a normal distribution with mean zero is used.
aggregate	If aggregate is TRUE the distance is returned, otherwise the size of the projection of the distance among each axis.

Value

The distance from the point x to the origin after a random displacement.

Author(s)

Ricardo Oliveros–Ramos

Examples

```
SphereN(rep(0, 10))
```

Index