

# Package ‘calibrar’

May 8, 2026

**Version** 0.9.0

**Title** Automated Parameter Estimation for Complex Models

**Description** General optimisation and specific tools for the parameter estimation (i.e. calibration) of complex models, including stochastic ones. It implements generic functions that can be used for fitting any type of models, especially those with non-differentiable objective functions, with the same syntax as `base::optim`.

It supports multiple phases estimation (sequential parameter masking), constrained optimization (bounding box restrictions) and automatic parallel computation of numerical gradients.

Some common maximum likelihood estimation methods and automated construction of the objective function from simulated model outputs is provided.

See <https://roliveros-ramos.github.io/calibrar/> for more details.

**Depends** R (>= 3.5.0)

**Imports** BB, cmaes, DEoptim, dfoptim, GenSA, graphics, minqa, optimx, foreach, lbfgsb3c, parallel, pso, rgenoud, soma, stats, stringr, utils

**Suggests** deSolve, ibm, knitr, rmarkdown, testthat (>= 3.0.0)

**License** GPL-2

**Encoding** UTF-8

**VignetteBuilder** knitr

**URL** <https://roliveros-ramos.github.io/calibrar/>

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

**ByteCompile** TRUE

**RoxygenNote** 7.2.3

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Ricardo Oliveros-Ramos [aut, cre]

**Maintainer** Ricardo Oliveros-Ramos <[ricardo.oliveros@gmail.com](mailto:ricardo.oliveros@gmail.com)>

**Repository** CRAN

**Date/Publication** 2024-02-14 21:30:15 UTC

## Contents

calibrar-package . . . . .	2
.get_command_argument . . . . .	3
.read_configuration . . . . .	4
ahres . . . . .	5
calibrar_demo . . . . .	7
calibrate . . . . .	9
calibration_data . . . . .	11
calibration_objFn . . . . .	12
calibration_setup . . . . .	13
gaussian_kernel . . . . .	13
gradient . . . . .	14
objFn . . . . .	15
optim2 . . . . .	15
optimh . . . . .	17
sphereN . . . . .	19
spline_par . . . . .	19
<b>Index</b>	<b>21</b>

---

calibrar-package

*Automated Calibration for Complex Models*

---

### Description

Automated Calibration for Complex Models

### Details

calibrar package: Automated Calibration for Complex Models

This package allows the parameter estimation (i.e. calibration) of complex models, including stochastic ones. It implements generic functions that can be used for fitting any type of models, especially those with non-differentiable objective functions, with the same syntax as `base::optim`. It supports multiple phases estimation (sequential parameter masking), constrained optimization (bounding box restrictions) and automatic parallel computation of numerical gradients. Some common maximum likelihood estimation methods and automated construction of the objective function from simulated model outputs is provided. See <https://roliveros-ramos.github.io/calibrar/> for more details.

### Author(s)

Ricardo Oliveros-Ramos Maintainer: Ricardo Oliveros-Ramos <[ricardo.oliveros@gmail.com](mailto: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 = calibrar_demo(model="PoissonMixedModel", L=5, T=100)
# get calibration information
calibrationInfo = calibration_setup(file=demo$path)
# get observed data
observed = calibration_data(setup=calibrationInfo, path=demo$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$par)
# objective functions
obj = calibration_objFn(model=runModel, setup=calibrationInfo,
                       observed=observed, forcing=forcing)
cat("Starting calibration...\n")
control = list(weights=calibrationInfo$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)
```

---

`.get_command_argument` *Get an specific argument from the command line*

---

## Description

Get an specific argument from the command line

## Usage

```
.get_command_argument(  
  x,  
  argument,  
  prefix = "--",  
  default = FALSE,
```

```

    verbose = FALSE
  )

```

### Arguments

x	The command line arguments, from <code>x = commandArgs()</code>
argument	The name of the argument.
prefix	The prefix to any argument of interest, the default is "-"
default	Default value to return if argument is missing, default to FALSE.
verbose	Boolean, if TRUE, shows a warning when the parameter is not found.

### Value

The value of the argument, assumed to be followed after '=' or, TRUE if nothing but the argument was found. If the argument is not found, FALSE is returned.

### Examples

```

.get_command_argument(commandArgs(), "interactive")
.get_command_argument(commandArgs(), "RStudio")
.get_command_argument(commandArgs(), "RStudio", prefix="")
.get_command_argument(commandArgs(), "vanilla")
.get_command_argument("--control.file=baz.txt", "control.file")

```

---

```

.read_configuration Read a configuration file.

```

---

### Description

File is expected to have lines of the form 'key SEP value' where key is the name of the parameter, SEP a separator (can be '=' ';' ':') and value the value of the parameter itself. The SEP for each line is determined and parameters values are returned as a list.

### Usage

```

.read_configuration(
  file,
  recursive = TRUE,
  keep.names = TRUE,
  conf.key = NULL,
  ...
)

```

**Arguments**

file	File to read the configuration
recursive	Should 'conf.key' keys be read as additional configuration files? Default is TRUE.
keep.names	Should names be kept as they are? By default, are converted to lower case.
conf.key	String indicating the leading key to find an additional configuration file.
...	Additional arguments, not currently in use.

---

ahres	<i>Adaptative Hierarchical Recombination Evolutionary Strategy (AHR-ES) for derivative-free and black-box optimization</i>
-------	--

---

**Description**

This function performs the optimization of a function using the Adaptative Hierarchical Recombination Evolutionary Strategy (AHR-ES, Oliveros & Shin, 2015).

**Usage**

```
ahres(
  par,
  fn,
  gr = NULL,
  ...,
  lower = -Inf,
  upper = +Inf,
  active = NULL,
  control = list(),
  hessian = FALSE,
  parallel = FALSE
)
```

**Arguments**

par	A numeric vector or list. 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	A function computing the gradient of fn. If NULL, a numerical approximation of the gradient is used. It can be also a character specifying the method for the computation of the numerical gradient: 'central', 'forward' (the default), 'backward' or 'richardson'.
...	Additional parameters to be passed to fn.
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	Boolean vector of the same length as par, indicating if the parameter is used in the optimization (TRUE) or hold at a fixed value (FALSE).
control	Parameter for the control of the algorithm itself, see details.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned? Currently not implemented.
parallel	Logical. Use parallel computation numerical of gradient?

### Value

A list with components:

**par** The best set of parameters found.

**value** The value of fn corresponding to par.

**counts** A two-element integer vector giving the number of calls to fn and gr respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to fn to compute a finite-difference approximation to the gradient.

**convergence** An integer code. 0 indicates successful completion.

**message** A character string giving any additional information returned by the optimizer, or NULL.

**hessian** Only if argument hessian is true. A symmetric matrix giving an estimate of the Hessian at the solution found. Note that this is the Hessian of the unconstrained problem even if the box constraints are active.

### Author(s)

Ricardo Oliveros-Ramos

### See Also

Other optimisers: [calibrate\(\)](#), [optim2\(\)](#), [optimh\(\)](#)

### Examples

```
## Not run: ahres(par=rep(1, 5), fn=sphereN)
```

**Description**

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

**Usage**

```
calibrar_demo(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.

**Details**

Current implemented models are:

**PoissonMixedModel** Poisson Autoregressive Mixed model for the dynamics of a population in different sites:

$$\log(\mu_{i,t+1}) = \log(\mu_{i,t}) + \alpha + \beta X_{i,t} + \gamma_t$$

where  $\mu_{i,t}$  is the size of the population in site  $i$  at year  $t$ ,  $X_{i,t}$  is the value of an environmental variable in site  $i$  at year  $t$ . The parameters to estimate were  $\alpha$ ,  $\beta$ , and  $\gamma_t$ , the random effects for each year,  $\gamma_t \sim N(0, \sigma^2)$ , and the initial population at each site  $\mu_{i,0}$ . We assumed that the observations  $N_{i,t}$  follow a Poisson distribution with mean  $\mu_{i,t}$ .

**PredatorPrey** Lotka Volterra Predator-Prey model. The model is defined by a system of ordinary differential equations for the abundance of prey  $N$  and predator  $P$ :

$$\frac{dN}{dt} = rN(1 - N/K) - \alpha NP$$

$$\frac{dP}{dt} = -lP + \gamma \alpha NP$$

The parameters to estimate are the prey's growth rate  $r$ , the predator's mortality rate  $l$ , the carrying capacity of the prey  $K$  and  $\alpha$  and  $\gamma$  for the predation interaction. Uses deSolve package for numerical solution of the ODE system.

**SIR** Susceptible-Infected-Recovered epidemiological model. The model is defined by a system of ordinary differential equations for the number of susceptible  $S$ , infected  $I$  and recovered  $R$  individuals:

$$\frac{dS}{dt} = -\beta SI/N$$

$$\frac{dI}{dt} = \beta SI/N - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

The parameters to estimate are the average number of contacts per person per time  $\beta$  and the instant probability of an infectious individual recovering  $\gamma$ . Uses deSolve package for numerical solution of the ODE system.

**IBMLotkaVolterra** Stochastic Individual Based Model for Lotka-Volterra model. Uses ibm package for the simulation.

## Value

A list with the following elements:

path	Path were the files were saved
par	Real value of the parameters used in the demo
setup	Path to the calibration setup file
guess	Values to be provided as initial guess to the calibrate function
lower	Values to be provided as lower bounds to the calibrate function
upper	Values to be provided as upper bounds to the calibrate function
phase	Values to be provided as phases to the calibrate function
constants	Constants used in the demo, any other variable not listed here.
value	NA, set for compatibility with summary methods.
time	NA, set for compatibility with summary methods.
counts	NA, set for compatibility with summary methods.

## Author(s)

Ricardo Oliveros–Ramos

## References

Oliveros-Ramos and Shin (2014)

## Examples

```
## Not run:

summary(ahr)
set.seed(880820)
path = NULL # NULL to use the current directory
# create the demonstration files
demo = calibrar_demo(path=path, model="PredatorPrey", T=100)
# get calibration information
calibration_settings = calibration_setup(file = demo$setup)
# get observed data
observed = calibration_data(setup = calibration_settings, path=demo$path)
# Defining 'run_model' function
run_model = calibrar:::PredatorPreyModel
```

```

# real parameters
cat("Real parameters used to simulate data\n")
print(unlist(demo$par)) # parameters are in a list
# objective functions
obj = calibration_objFn(model=run_model, setup=calibration_settings, observed=observed, T=demo$T)
obj2 = calibration_objFn(model=run_model, setup=calibration_settings, observed=observed,
T=demo$T, aggregate=TRUE)
cat("Starting calibration...\n")
cat("Running optimization algorithms\n", "\t")
cat("Running optim AHR-ES\n")
ahr = calibrate(par=demo$guess, fn=obj, lower=demo$lower, upper=demo$upper, phases=demo$phase)
summary(ahr)

## End(Not run)

```

---

calibrate

*Sequential parameter estimation for the calibration of complex 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,
  ...,
  method,
  lower,
  upper,
  phases,
  control,
  hessian,
  replicates,
  parallel
)

## Default S3 method:
calibrate(
  par,
  fn,
  gr = NULL,
  ...,
  method = NULL,

```

```

lower = NULL,
upper = NULL,
phases = NULL,
control = list(),
hessian = FALSE,
replicates = 1,
parallel = FALSE
)

```

### Arguments

par	A numeric vector or list. 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	A function computing the gradient of fn. If NULL, a numerical approximation of the gradient is used. It can be also a character specifying the method for the computation of the numerical gradient: 'central', 'forward' (the default), 'backward' or 'richardson'.
...	Additional parameters to be passed to fn.
method	The optimization method to be used. The default method is the AHR-ES (Adaptive Hierarchical Recombination Evolutionary Strategy, Oliveros-Ramos & Shin, 2016). See details for the methods 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).
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.
control	Parameter for the control of the algorithm itself, see details.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned? Currently not implemented.
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.
parallel	Logical. Use parallel computation numerical of gradient?

### 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 exploit the additional information provided by a vectorial output from `fn`.

### Author(s)

Ricardo Oliveros-Ramos

**See Also**

Other optimisers: [ahres\(\)](#), [optim2\(\)](#), [optimh\(\)](#)

**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)
```

---

calibration_data	<i>Get observed data for the calibration of a model</i>
------------------	---

---

**Description**

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

**Usage**

```
calibration_data(setup, path = ".", file = NULL, verbose = TRUE, ...)
```

**Arguments**

setup	A data.frame with the information about the calibration, normally created with the <a href="#">calibration_setup</a> function. See details.
path	Path to the directory to look up for the data. Paths in setup are considered relatives to this path.
file	Optional file to save the created object (as an 'rds' file.)
verbose	If TRUE, detailed messages of the process are printed.
...	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 [calibration\\_objFn](#).

**Author(s)**

Ricardo Oliveros-Ramos

**See Also**

[calibration\\_objFn](#), [calibration\\_setup](#).

---

calibration_objFn	<i>Create an objective function to be used with optimization routines</i>
-------------------	---

---

### 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

```
calibration_objFn(model, setup, observed, aggFn = NULL, aggregate = FALSE, ...)
```

### Arguments

model	Function to run the model and produce a list of outputs.
setup	A data.frame with the information about the calibration, normally created with the <a href="#">calibration_setup</a> function. See details.
observed	A list of the observed variables created with the function <a href="#">calibration_data</a>
aggFn	A function to aggregate fn to a scalar value if the returned value is a vector. Some optimization algorithm can explore 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 model function.

### Value

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

### Author(s)

Ricardo Oliveros-Ramos

### See Also

[calibration\\_data](#), [calibration\\_setup](#).

---

calibration\_setup      *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**

```
calibration_setup(file, control = list(), ...)
```

**Arguments**

file	The file with the calibration information, see details.
control	Control arguments for generating the setup. See details.
...	Additional arguments to <code>read.csv</code> function.

**Value**

A `data.frame` with the information for the calibration of a model, to be used with the [calibration\\_objFn](#) and [calibration\\_data](#).

**Author(s)**

Ricardo Oliveros-Ramos

**See Also**

[calibration\\_objFn](#), [calibration\\_data](#).

---

gaussian\_kernel      *Calculate a discretization of the 2D Gaussian Kernel*

---

**Description**

Calculate a discretization of the 2D Gaussian Kernel

**Usage**

```
gaussian_kernel(par, lower, upper, n = 10, checkSymmetry = TRUE, ...)
```

**Arguments**

par	A list, including the mean and covariance matrix.
lower	A vector, indicating the lower bound for the calculation.
upper	A vector, indicating the upper bound for the calculation.
n	The number of cells for each dimension, can be one or two numbers.
checkSymmetry	TRUE by default, checks if the covariance matrix is symmetric.
...	Additional arguments, currently not used.

**Value**

A list, with 'x', 'y' and 'z' components.

---

gradient	<i>Numerical computation of the gradient, with parallel capabilities</i>
----------	--

---

**Description**

This function calculates the gradient of a function, numerically, including the possibility of doing it in parallel.

**Usage**

```
gradient(fn, x, method, control, parallel, ...)
```

**Arguments**

fn	The function to calculate the gradient.
x	The value to compute the gradient at.
method	The method used. Currently implemented: central, backward, forward and Richardson. See details.
control	A list of control arguments.
parallel	Boolean, should numerical derivatives be calculated in parallel?
...	Additional arguments to be passed to fn.

**Value**

The gradient of fn at x.

**Examples**

```
gradient(fn=function(x) sum(x^3), x=0)
```

---

objFn	<i>Calculated error measure between observed and simulated data</i>
-------	---

---

**Description**

Calculated error measure between observed and simulated data

**Usage**

```
objFn(obs, sim, FUN, ...)
fitness(obs, sim, FUN, ...)
```

**Arguments**

obs	observed data as expected by FUN.
sim	simulated data matching 'obs'
FUN	the error function. Current accepted values area: 'norm2', 'lnorm2', 'lnorm3', 'multinomial', 'pois', 'penalty0', 'penalty1', 'penalty2' and 'normp'.
...	Additional arguments to FUN

**Value**

the value of FUN(obs, sim, ...)

---

optim2	<i>General-purpose optimization with parallel numerical gradient computation</i>
--------	--

---

**Description**

General-purpose optimization with parallel numerical gradient computation

**Usage**

```
optim2(
  par,
  fn,
  gr = NULL,
  ...,
  method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "Brent", "nlm", "nlminb",
    "Rcgmin", "Rvmin", "hjn", "spg", "LBFGSB3", "AHR-ES"),
  lower = -Inf,
  upper = +Inf,
  active = NULL,
```

```

control = list(),
hessian = FALSE,
parallel = FALSE
)

```

### Arguments

<code>par</code>	A numeric vector or list. The length of the <code>par</code> argument defines the number of parameters to be estimated (i.e. the dimension of the problem).
<code>fn</code>	The function to be minimized.
<code>gr</code>	A function computing the gradient of <code>fn</code> . If <code>NULL</code> , a numerical approximation of the gradient is used. It can be also a character specifying the method for the computation of the numerical gradient: <code>'central'</code> , <code>'forward'</code> (the default), <code>'backward'</code> or <code>'richardson'</code> .
<code>...</code>	Additional parameters to be passed to <code>fn</code> .
<code>method</code>	The optimization method to be used. The default method is the AHR-ES (Adaptive Hierarchical Recombination Evolutionary Strategy, Oliveros-Ramos & Shin, 2016). See details for the methods available.
<code>lower</code>	Lower threshold value(s) for parameters. One value or a vector of the same length as <code>par</code> . If one value is provided, it is used for all parameters. <code>NA</code> means <code>-Inf</code> . By default <code>-Inf</code> is used (unconstrained).
<code>upper</code>	Upper threshold value(s) for parameters. One value or a vector of the same length as <code>par</code> . If one value is provided, it is used for all parameters. <code>NA</code> means <code>Inf</code> . By default <code>Inf</code> is used (unconstrained).
<code>active</code>	Boolean vector of the same length as <code>par</code> , indicating if the parameter is used in the optimization ( <code>TRUE</code> ) or hold at a fixed value ( <code>FALSE</code> ).
<code>control</code>	Parameter for the control of the algorithm itself, see details.
<code>hessian</code>	Logical. Should a numerically differentiated Hessian matrix be returned? Currently not implemented.
<code>parallel</code>	Logical. Use parallel computation numerical of gradient?

### Value

A list with components:

**par** The best set of parameters found.

**value** The value of `fn` corresponding to `par`.

**counts** A two-element integer vector giving the number of calls to `fn` and `gr` respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to `fn` to compute a finite-difference approximation to the gradient.

**convergence** An integer code. 0 indicates successful completion.

**message** A character string giving any additional information returned by the optimizer, or `NULL`.

**hessian** Only if argument `hessian` is true. A symmetric matrix giving an estimate of the Hessian at the solution found. Note that this is the Hessian of the unconstrained problem even if the box constraints are active.

**Author(s)**

Ricardo Oliveros-Ramos

**See Also**Other optimisers: [ahres\(\)](#), [calibrate\(\)](#), [optimh\(\)](#)**Examples**

```
optim2(par=rep(NA, 5), fn=sphereN)
```

---

 optimh

*General-purpose optimization using heuristic algorithms*


---

**Description**

General-purpose optimization using heuristic algorithms

**Usage**

```
optimh(
  par,
  fn,
  gr = NULL,
  ...,
  method = c("AHR-ES", "Nelder-Mead", "SANN", "hjn", "bobyqa", "CMA-ES", "genSA", "DE",
    "soma", "genoud", "PSO", "hybridPSO", "mads", "hjk", "hjkb", "nmk", "nmkb"),
  lower = -Inf,
  upper = +Inf,
  active = NULL,
  control = list(),
  hessian = FALSE,
  parallel = FALSE
)
```

**Arguments**

par	A numeric vector or list. 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	Function to compute the gradient of fn. Ignored by most methods, added for consistency with other optimization functions.
...	Additional parameters to be passed to fn.
method	The optimization method to be used. The default method is the AHR-ES (Adaptive Hierarchical Recombination Evolutionary Strategy, Oliveros-Ramos & Shin, 2016). See details for the methods 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).
active	Boolean vector of the same length as par, indicating if the parameter is used in the optimization (TRUE) or hold at a fixed value (FALSE).
control	Parameter for the control of the algorithm itself, see details.
hessian	Logical. Should a numerically differentiated Hessian matrix be returned? Currently not implemented.
parallel	Logical. Use parallel computation numerical of gradient?

### Value

A list with components:

**par** The best set of parameters found.

**value** The value of fn corresponding to par.

**counts** A two-element integer vector giving the number of calls to fn and gr respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to fn to compute a finite-difference approximation to the gradient.

**convergence** An integer code. 0 indicates successful completion.

**message** A character string giving any additional information returned by the optimizer, or NULL.

**hessian** Only if argument hessian is true. A symmetric matrix giving an estimate of the Hessian at the solution found. Note that this is the Hessian of the unconstrained problem even if the box constraints are active.

### Author(s)

Ricardo Oliveros-Ramos

### See Also

Other optimisers: [ahres\(\)](#), [calibrate\(\)](#), [optim2\(\)](#)

### Examples

```
optim2(par=rep(NA, 5), fn=sphereN)
```

---

sphereN	<i>Sphere function with random noise</i>
---------	--

---

**Description**

This function calculates the Euclidian distance from a point to the origin after a random displacement of its 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))
```

---

spline_par	<i>Predict time-varying parameters using splines.</i>
------------	---

---

**Description**

Predict time-varying parameters using splines.

**Usage**

```
spline_par(par, n, knots = NULL, periodic = FALSE, period = NULL)
```

**Arguments**

<code>par</code>	Values at knots
<code>n</code>	Number of points. Time (independent variable) is assumed to be between 0 and n with <code>length(par)</code> equidistant points (including 0 and n).
<code>knots</code>	Position of knots. Default, is <code>length(x)</code> equidistant points between 0 and 1. Always are re-scaled to 0 to 1.
<code>periodic</code>	boolean, is the spline periodic?
<code>period</code>	If periodic is TRUE, it specify the time period.

**Value**

A list with the interpolates values as 'x' and 'time'.

# Index

- \* **calibration**
  - calibrar-package, 2
  - calibrar\_demo, 7
- \* **demo**
  - calibrar\_demo, 7
- \* **optimisers**
  - ahres, 5
  - calibrate, 9
  - optim2, 15
  - optimh, 17
- \* **random**
  - sphereN, 19
- \* **stochastic**
  - sphereN, 19
- .get\_command\_argument, 3
- .read\_configuration, 4
  
- ahres, 5, 11, 17, 18
  
- calibrar (calibrar-package), 2
- calibrar-package, 2
- calibrar\_demo, 7
- calibrate, 6, 9, 17, 18
- calibration\_data, 11, 12, 13
- calibration\_objFn, 11, 12, 13
- calibration\_setup, 11, 12, 13
  
- fitness (objFn), 15
  
- gaussian\_kernel, 13
- gradient, 14
  
- objFn, 15
- optim2, 6, 11, 15, 18
- optimh, 6, 11, 17, 17
  
- sphereN, 19
- spline\_par, 19