Package 'FAMoS'

May 26, 2019

Type Package

Title A Flexible Algorithm for Model Selection

Version 0.2.0

Author Michael Gabel, Tobias Hohl

Maintainer Michael Gabel <m_gabel@gmx.de>

Description Given a set of parameters describing model dynamics and a corresponding cost function, FAMoS performs a dynamic forward-backward model selection on a specified selection criterion. It also applies a non-local swap search method. Works on any cost function.

License MIT + file LICENSE

Depends R (>= 3.6)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports future, R.utils

Suggests future.batchtools, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-05-26 09:40:03 UTC

R topics documented:

`aicc.weights` *Plot evidence ratios*

Description

Calculates the evidence ratios for each parameter based on Akaike weights and plots them.

Usage

```
aicc.weights(input = getwd(), mrun = NULL, reorder = TRUE,
  save.output = NULL)
```

Arguments

`input`	Either a string containing the directory which holds the "FAMoS-Results" folder or a matrix containing the tested models along with the respective information criteria. Default to `getwd()`.
`mrun`	A string giving the number of the corresponding FAMoS run, e.g "004". If NULL (default), all FAMoS runs in the "FAMoS-Results/TestedModels/" folder will be used for evaluation.
`reorder`	If TRUE, results will be ordered by evidence ratios (descending). If FALSE, the order of parameters will be the same as the order specified in `init.par` in `famos`. Default to TRUE.
`save.output`	A string containing the location and name under which the figure should be saved (format is pdf). Default to NULL.

Details

The plot shows the relative support or evidence ratio for each parameter. Parameters included in the best model are printed bold and the corresponding lines are coloured in red.

Value

A plot showing the evidence ratios for all model parameters. Additionally, the evidence ratios are returned.

Examples

```
#plot evidence ratios
aicc.weights(input = famos.run)
aicc.weights(input = famos.run, reorder = FALSE)
```

---

base.optim *Fit Parameters of a Model*

Description

Given a specific model, base.optim fits the corresponding parameters and saves the output.

Usage

```
base.optim(binary, parms, fit.fn, homedir, use.optim = TRUE,
  optim.runs = 1, default.val = NULL, random.borders = 1,
  con.tol = 0.01, control.optim = list(maxit = 1000),
  parscale.pars = FALSE, scaling = NULL, verbose = FALSE, ...)
```

Arguments

binary	A vector containing zeroes and ones. Zero indicates that the corresponding parameter is not fitted.
parms	A named vector containing the initial values for optim. Must be in the same order as binary.
fit.fn	A cost function. Has to take the complete parameter vector as an input argument (needs to be named parms) and must return the corresponding negative log-likelihood (-2LL, see Burnham and Anderson 2002). The binary vector containing the information which parameters are fitted, can also be used by taking binary as an additional function input argument.
homedir	A string giving the directory in which the result folders generated by famos are found.
use.optim	Logical. If true, the cost function fit.fn will be fitted via optim. If FALSE, the cost function will only be evaluated.
optim.runs	The number of times that each model will be optimised. Default to 1. Numbers larger than 1 use random initial conditions (see random.borders).
default.val	A named list containing the values that the non-fitted parameters should take. If NULL, all non-fitted parameters will be set to zero. Default values can be either given by a numeric value or by the name of the corresponding parameter the value should be inherited from (NOTE: In this case the corresponding parameter entry has to contain a numeric value). Default to NULL.
random.borders	The ranges from which the random initial parameter conditions for all optim.runs > 1 are sampled. Can be either given as a vector containing the relative deviations for all parameters or as a matrix containing in its first column the lower and in its second column the upper border values. Parameters are uniformly sampled

based on runif. Default to 1 (100% deviation of all parameters). Alternatively, functions such as rnorm, rchisq, etc. can be used if the additional arguments are passed along as well.

con.tol	The relative convergence tolerance. famos will rerun optim until the relative improvement between the current and the last fit is less than con.tol. Default is set to 0.01, meaning the fitting will terminate if the improvement is less than 1% of the previous value.
control.optim	Control parameters passed along to optim. For more details, see optim.
parscale.pars	Logical. If TRUE (default), the parscale option will be used when fitting with optim. This is helpful, if the parameter values are on different scales.
scaling	Numeric vector determining how newly added model parameters are scaled. Only needed if parscale.pars is TRUE.
verbose	Logical. If TRUE, FAMoS will output all details about the current fitting procedure.
...	Additional parameters.

Details

The fitting routine of base.optim is based on the function optim. The number of fitting runs can be specified by the optim.runs parameter in the famos function. Here, the first fitting run takes the parameters supplied in parms as a starting condition, while all following fitting runs sample new initial sets according to a uniform distribution based on the intervals [parms - abs(parms), parms + abs(parms)]. Additionally, each fitting run is based on a while-loop that compares the outcome of the previous and the current fit. Each fitting run is terminated when the specified convergence tolerance con.tol is reached.

Value

Saves the results obtained from fitting the corresponding model parameters in the respective files, from which they can be accessed by the main function famos.

Examples

```
#setting data
true.p2 <- 3
true.p5 <- 2
sim.data <- cbind.data.frame(range = 1:10,
                             y = true.p2^2 * (1:10)^2 - exp(true.p5 * (1:10)))

#define initial parameter values and corresponding test function
inits <- c(p1 = 3, p2 = 4, p3 = -2, p4 = 2, p5 = 0)

cost_function <- function(parms, binary, data){
  if(max(abs(parms)) > 5){
    return(NA)
  }
  with(as.list(c(parms)), {
   res <- p1*4 + p2^2*data$range^2 + p3*sin(data$range) + p4*data$range - exp(p5*data$range)
    diff <- sum((res - data$y)^2)
```

```
    #calculate AICC
    nr.par <- length(which(binary == 1))
    nr.data <- nrow(data)
    AICC <- diff + 2*nr.par + 2*nr.par*(nr.par + 1)/(nr.data - nr.par -1)

    return(AICC)
  })
}

#create directories if needed
make.directories(tempdir())

#optimise the model parameters
base.optim(binary = c(0,1,1,0,1),
           parms = inits,
           fit.fn = cost_function,
           homedir = tempdir(),
           data = sim.data)

#delete tempdir
unlink(paste0(tempdir(),"/FAMoS-Results"), recursive = TRUE)
```

`combine.and.fit` *Combine and Fit Parameters*

Description

Combines fitted and non-fitted parameters and calls the fitting function. Serves as a wrapping function for the user-specified fitting function `fit.fn` (see `famos`).

Usage

```
combine.and.fit(par, par.names, fit.fn, binary = NULL,
  default.val = NULL, ...)
```

Arguments

`par`	A named vector containing all parameters that are supposed to be fitted.
`par.names`	The names of all parameters
`fit.fn`	The cost function (see `famos` for more details).
`binary`	A vector containing zeroes and ones. Zero indicates that the corresponding parameter is not fitted.
`default.val`	A named list containing the values that the non-fitted parameters should take. If NULL, all non-fitted parameters will be set to zero. Default values can be either given by a numeric value or by the name of the corresponding parameter the value should be inherited from (NOTE: In this case the corresponding parameter entry has to contain a numeric value). Default to NULL.
`...`	Other arguments.

Value

Returns the negative log-likelihood as calculated by the specified cost function

Examples

```
#set parameters and cost function
fit.par <- c(p1 = 2, p2 = 4)
name.par <- c("p1", "p2", "p3")
defaults <- list(p1 = 0, p2 = 2, p3 = 4)
cost.function <- function(parms){
    parms[1] + parms[2] + parms[3]
}

#call combine.and.fit
combine.and.fit(par = fit.par, par.names = name.par, fit.fn = cost.function)
combine.and.fit(par = fit.par, par.names = name.par, fit.fn = cost.function, default.val = defaults)
```

---

combine.par *Combine Fitted and Non-fitted Parameters*

Description

Combines fitted and non-fitted parameters into a single vector, taking into account the specified default values.

Usage

```
combine.par(fit.par, all.names, default.val = NULL)
```

Arguments

fit.par	A named vector containing all parameters that are supposed to be fitted.
all.names	A vector containing the names of all parameters (fitted and non-fitted).
default.val	A named list containing the values that the non-fitted parameters should take. If NULL, all non-fitted parameters will be set to zero. Default values can be either given by a numeric value or by the name of the corresponding parameter the value should be inherited from (NOTE: If a string is supplied, the corresponding parameter entry has to contain a numeric value). Default to NULL.

Value

A named vector containing the elements of fit.par and the non-fitted parameters, in the order given by all.names. The non-fitted parameters are determined by the remaining names in all.names and their values are set according to default.val.

Examples

```
#set parameters, names and default values
fits <- c(p1 = 3, p4 = -2)
par.names <- c("p1", "p2", "p3", "p4", "p5")
defaults <- list(p1 = 4, p2 = 10, p3 = "p1", p4 = 0, p5 = "p4")

#combine the parameters in different ways
combine.par(fit.par = fits, all.names = par.names)
combine.par(fit.par = fits, all.names = par.names, default.val = defaults)
```

`famos` *Automated Model Selection*

Description

Given a vector containing all parameters of interest and a cost function, the `FAMoS` looks for the most appropriate subset model to describe the given data.

Usage

```
famos(init.par, fit.fn, homedir = getwd(), do.not.fit = NULL,
  method = "forward", init.model.type = "random", refit = FALSE,
  use.optim = TRUE, optim.runs = 5, default.val = NULL,
  swap.parameters = NULL, critical.parameters = NULL,
  random.borders = 1, control.optim = list(maxit = 1000),
  parscale.pars = FALSE, con.tol = 0.1, save.performance = TRUE,
  future.off = TRUE, log.interval = 600, verbose = FALSE, ...)
```

Arguments

`init.par`	A named vector containing the initial parameter values.
`fit.fn`	A cost function. Has to take the complete parameter vector as an input (needs to be names `parms`) and must return a selection criterion value (e.g. AICc or BIC). See Details for more information.
`homedir`	The directory to which the results should be saved to.
`do.not.fit`	The names of the parameters that are not supposed to be fitted. Default is NULL.
`method`	The starting method of the FAMoS. Options are "forward" (forward search), "backward" (backward elimination) and "swap" (only if `critical.parameters` or `swap.parameters` are supplied). Methods are adaptively changed over each iteration of the FAMoS. Default to "forward".
`init.model.type`	The starting model. Options are "global" (starts with the complete model), "random" (creates a randomly sampled starting model) or "most.distant" (uses the model most dissimilar from all other previously tested models). Alternatively, a specific model can be used by giving the corresponding names of the parameters one wants to start with. Default to "random".

refit — If TRUE, previously tested models will be tested again. Default to FALSE.

use.optim — Logical. If true, the cost function fit.fn will be fitted via optim. If FALSE, the cost function will only be evaluated.

optim.runs — The number of times that each model will be optimised. Default to 1. Numbers larger than 1 use random initial conditions (see random.borders).

default.val — A named list containing the values that the non-fitted parameters should take. If NULL, all non-fitted parameters will be set to zero. Default values can be either given by a numeric value or by the name of the corresponding parameter the value should be inherited from (NOTE: In this case the corresponding parameter entry has to contain a numeric value). Default to NULL.

swap.parameters — A list specifying which parameters are interchangeable. Each swap set is given as a vector containing the names of the respective parameters. Default to NULL.

critical.parameters — A list specifying sets of critical parameters. Critical sets are parameters sets, of which at least one parameter per set has to be present in each tested model. Default to NULL.

random.borders — The ranges from which the random initial parameter conditions for all optim.runs larger than one are sampled. Can be either given as a vector containing the relative deviations for all parameters or as a matrix containing in its first column the lower and in its second column the upper border values. Parameters are uniformly sampled based on runif. Default to 1 (100% deviation of all parameters). Alternatively, functions such as rnorm, rchisq, etc. can be used if the additional arguments are passed along as well.

control.optim — Control parameters passed along to optim. For more details, see optim.

parscale.pars — Logical. If TRUE, the parscale option will be used when fitting with optim. This can help to speed up the fitting procedure, if the parameter values are on different scales. Default to FALSE.

con.tol — The absolute convergence tolerance of each fitting run (see Details). Default is set to 0.1.

save.performance — Logical. If TRUE, the performance of FAMoS will be evaluated in each iteration via famos.performance, which will save the corresponding plots into the folder "FAMoS-Results/Figures/" (starting from iteration 3) and simultaneously show it on screen. Default to TRUE.

future.off — Logical. If TRUE (default), FAMoS runs without the use of futures.

log.interval — The interval (in seconds) at which FAMoS informs about the current status, i.e. which models are still running and how much time has passed. Default to 600 (= 10 minutes).

verbose — Logical. If TRUE, FAMoS will output all details about the current fitting procedure.

... — Other arguments that will be passed along to future, optim or the user-specified cost function fit.fn.

Details

In each iteration, the FAMoS finds all neighbouring models based on the current model and method, and subsequently tests them. If one of the tested models performs better than the current model, the model, but not the method, will be updated. Otherwise, the method, but not the model, will be adaptively changed, depending on the previously used methods.

The cost function `fit.fn` can take the following inputs:

parms A named vector containing all parameter values. This input is mandatory. If `use.optim = TRUE`, FAMos will automatically subset the complete parameter set into fitted and non-fitted parameters.

binary Optional input. The binary vector contains the information which parameters are currently fitted. Fitted parameters are set to 1, non-fitted to 0. This input can be to split the complete parameter set into fitted and non-fitted parameters if a customised optimisation function is used (see `use.optim`).

... Other parameters that can should be passed to `fit.fn`

If `use.optim = TRUE`, the cost function needs to return a single numeric value, which corresponds to the selection criterion value. However, if `use.optim = FALSE`, the cost function needs to return a list containing in its first entry the selection criterion value and in its second entry the named vector of the fitted parameter values (non-fitted parameters are internally assessed).

Value

A list containing the following elements:

SCV The value of the selection criterion of the best model.

par The values of the fitted parameter vector corresponding to the best model.

binary The binary information of the best model.

vector Vector indicating which parameters were fitted in the best model.

total.models.tested The total number of different models that were analysed. May include repeats.

mrun The number of the current FAMoS run.

initial.model The first model evaluated by the FAMoS run.

Examples

```
#setting data
true.p2 <- 3
true.p5 <- 2
sim.data <- cbind.data.frame(range = 1:10,
                             y = true.p2^2 * (1:10)^2 - exp(true.p5 * (1:10)))

#define initial parameter values and corresponding test function
inits <- c(p1 = 3, p2 = 4, p3 = -2, p4 = 2, p5 = 0)

cost_function <- function(parms, binary, data){
  if(max(abs(parms)) > 5){
    return(NA)
```

```
  }
  with(as.list(c(parms)), {
   res <- p1*4 + p2^2*data$range^2 + p3*sin(data$range) + p4*data$range - exp(p5*data$range)
    diff <- sum((res - data$y)^2)

    #calculate AICC
    nr.par <- length(which(binary == 1))
    nr.data <- nrow(data)
    AICC <- diff + 2*nr.par + 2*nr.par*(nr.par + 1)/(nr.data - nr.par -1)

    return(AICC)
  })
}


#set swap set
swaps <- list(c("p1", "p5"))

#perform model selection
famos(init.par = inits,
      fit.fn = cost_function,
      homedir = tempdir(),
      method = "swap",
      swap.parameters = swaps,
      init.model.type = c("p1", "p3"),
      optim.runs = 1,
      data = sim.data)

#delete tempdir
unlink(paste0(tempdir(),"/FAMoS-Results"), recursive = TRUE)
```

`famos.performance` *Plot FAMoS Performance*

Description

For each FAMoS run `famos.performance` plots the corresponding best model and selection criterion value.

Usage

```
famos.performance(input, path = getwd(), save.output = NULL, ...)
```

Arguments

`input`	Either a string giving the three-digit number of the corresponding FAMoS run, e.g "004", or a matrix containing the tested models along with the respective information criteria.
`path`	If `input` is the string of an FAMoS run, the directory containing the "FAMoS-Results" folder needs to be supplied as well. Default to `getwd`.

save.output	A string containing the location and name under which the figure should be saved (format is .pdf). Default to NULL.
...	Other graphical parameters.

Details

The upper plot shows the improvement of the selection criterion over each FAMoS iteration. The best value is shown on the right axis. The lower plot depicts the corresponding best model of each iteration. Here, green colour shows added, red colour removed and blue colour swapped parameters. The parameters of the final model are printed bold.

Value

A plot showing the value of the selection criterion and best model of each FAMoS iteration.

Examples

```
#plot the performance of an FAMoS run
famos.performance(input = famos.run)
```

---

famos.run *FAMoS performance test run*

Description

The data shows the performance of the FAMoS over the course of 8 iterations.

Usage

```
famos.run
```

Format

A matrix with 7 rows and 17 columns

Source

Created by famos.

`get.most.distant` *Get Most Distant Model*

Description

This function can be used to find a model that is most distinct from all previously tested models.

Usage

```
get.most.distant(input = getwd(), mrun = NULL, max.number = 100)
```

Arguments

`input`	Either a string containing the directory which holds the "FAMoS-Results" folder or a matrix containing the tested models along with the respective information criteria. Default to `getwd()`.
`mrun`	A string giving the number of the corresponding FAMoS run, e.g "004". If NULL (default), all FAMoS runs in the "FAMoS-Results/TestedModels/" folder will be used for evaluation.
`max.number`	The maximum number of times that the `get.most.distant` function tries to find the most distant model (see details). Default to 100.

Details

Taking the order from the 'TestedModels' files found in 'FAMoS-Results/TestedModels/', this function successively tries to obtain a previously untested model that is most distant from all previously tested ones (here, distance means the number of difference in fitted parameters). To this end, a model is taken from 'TestedModels' and its corresponding complement model is calculated (i.e. the model containing all parameters that the original didn't fit). From thereon, the distance of neighbouring models is calculated and the path of increasing distances is followed. This process is repeated until all models in 'TestedModels' have been assessed or the `max.number` is reached.

Value

A list containing in its first entry the maximal distance found, the second entry the parameter names and in its third entry the corresponding binary vector. Note that the model may not fulfill previously specified critical conditions.

Examples

```
get.most.distant(input = famos.run)
```

make.directories *Create Results Directories*

Description

Creates the directories, in which the results are going to be saved.

Usage

```
make.directories(homedir)
```

Arguments

homedir	The directory in which the result folders will be created.

Details

All files will be stored in the main folder "FAMoS-Results". It contains the following subdirectories:

BestModel Contains the information criteria and the corresponding parameter estimates of the best fitting model.

Figures Contains figures showing the performance of the FAMoS.

Fits Contains the fitted parameter values of each of the tested models.

TestedModels Contains the binary information of all of the tested models.

Value

Creates directories.

model.appr *Test for Model Appropriateness*

Description

Tests if a model is violating specified critical conditions.

Usage

```
model.appr(current.parms, critical.parms, do.not.fit = NULL)
```

Arguments

current.parms	A vector containing the indices of the current model.
critical.parms	A list containing vectors which specify the critical parameter sets. Needs to be given by index and not by name (for conversion see set.crit.parms).
do.not.fit	A vector containing the indices of the parameters that are not to be fitted.

Value

TRUE, if the model is appropriate, FALSE, if it violates the specified critical conditions.

Examples

```
#define the models to be checked
model1 <- c(1,2,5)
model2 <- c(1,4,5)
#define the critical conditions
crits <- list(c(2,3))

#test the models
model.appr(current.parms = model1, critical.parms = crits)
model.appr(current.parms = model2, critical.parms = crits)
```

---

parscale.famos *Control Function for Optim*

Description

This function is designed to use the built-in option `parscale` in `optim` with absolute scaling values.

Usage

```
parscale.famos(par, scale, fix = 1, correction = NULL)
```

Arguments

`par`	A vector containing the values of the original parameters.
`scale`	A vector containing the corresponding absolute scaling values that will be used during the first steps in `optim`.
`fix`	Integer containing the index of the parameter that will be scaled by 10% of its original value, meaning the corresponding entry in scale will be overwritten (`parscale` in `optim` needs one value like this). Default to 1.
`correction`	Used for scaling newly added parameter values by their original scale as specified in `init.par` in `famos`. Default to NULL.

Value

A vector that can be used to scale `parscale` in `optim` accordingly.

Examples

```
test.func<-function(x){
print(x)
3*x[1]+4*x[2]
}

pars <- c(1, 1000, 10)
scaling <- c(0.1, 3000, 10)

p.scale <- parscale.famos(par = pars, scale = scaling)

optim(par = pars, fn = test.func, control = list(maxit = 10, parscale = p.scale, trace = TRUE))
```

---

random.init.model *Generate Random Starting Model*

Description

Generates a random starting model (if specified by the user in famos), taking into account the critical conditions and the parameters which should not be fitted.

Usage

```
random.init.model(number.par, crit.parms = NULL, no.fit = NULL)
```

Arguments

number.par	The number of total parameters available.
crit.parms	A list containing vectors which specify the critical parameter sets. Needs to be given by index and not by name (for conversion see set.crit.parms).
no.fit	A vector containing the indices of the parameters which are not to be fitted.

Value

A vector containing the parameter indices of the random model.

Examples

```
#set critical conditions
crits <- list(c(1,2,3), c(4,5))
#generate random model
random.init.model(number.par = 20)
random.init.model(number.par = 20, crit.parms = crits)
```

`retrieve.results` *Retrieve Model Results*

Description

Returns the results of a specified model that was tested by `famos`.

Usage

```
retrieve.results(model, homedir = getwd(), all.names = NULL)
```

Arguments

`model`	Either the binary number of the model as a string (e.g. "011001"), a named vector containing the names the names of the fitted parameters or a vector containing ones and zeroes to indicate which model parameter were fitted. If the names of the fitted parameters are supplied, `all.names` must be specified as well.
`homedir`	A string giving the directory in which the result folders are found. This is the same directory in which `famos` was run.
`all.names`	A vector containing the names of all parameters.

Value

A vector containing the calculated selection criteria and estimated parameters of the specified model.

Examples

```
#setting data
x.values <- 1:7
y.values <-  3^2 * x.values^2 - exp(2 * x.values)

#define initial conditions and corresponding test function
inits <- c(p1 = 3, p2 = 4, p3 = -2, p4 = 2, p5 = 0)

cost_function <- function(parms, x.vals, y.vals){
 if(max(abs(parms)) > 5){
   return(NA)
 }
 with(as.list(c(parms)), {
   res <- p1*4 + p2^2*x.vals^2 + p3*sin(x.vals) + p4*x.vals - exp(p5*x.vals)
   diff <- sum((res - y.vals)^2)
 })
}
```

```
#perform model selection
res <- famos(init.par = inits,
             fit.fn = cost_function,
             nr.of.data = length(y.values),
             homedir = tempdir(),
             init.model.type = c("p2", "p3"),
             optim.runs = 1,
             x.vals = x.values,
             y.vals = y.values)

#get results
retrieve.results(model = "01110", homedir = tempdir())
retrieve.results(model = c("p2", "p3", "p4"), homedir = tempdir(),
                 all.names = c("p1","p2", "p3", "p4", "p5"))
retrieve.results(model = c(0,1,1,1,0), homedir = tempdir())

#delete tempdir
unlink(paste0(tempdir(),"/FAMoS-Results"), recursive = TRUE)
```

`return.results` *Return Final Results*

Description

Returns the results of one FAMoS run. Includes the best parameter sets and corresponding selection criterion.

Usage

```
return.results(homedir, mrun)
```

Arguments

`homedir`	A string giving the directory in which the result folders are found. This is the same directory in which `famos` was run.
`mrun`	The number of the FAMoS run that is to be evaluated. Must be a three digit string in the form of '001'. Alternatively, supplying 'best' will return the best result that is found over several FAMoS runs.

Value

A list containing the following elements:

SCV The value of the selection criterion of the best model.

par The values of the fitted parameter vector corresponding to the best model.

binary The binary information of the best model.

vector Vector indicating which parameters were fitted in the best model.

total.models.tested The total number of different models that were analysed. May include repeats.

mrun The number of the current FAMoS run.

initial.mode The first model evaluated by the FAMoS run.

Examples

```
#setting data
x.values <- 1:7
y.values <-  3^2 * x.values^2 - exp(2 * x.values)

#define initial conditions and corresponding test function
inits <- c(p1 = 3, p2 = 4, p3 = -2, p4 = 2, p5 = 0)

cost_function <- function(parms, x.vals, y.vals){
 if(max(abs(parms)) > 5){
   return(NA)
 }
 with(as.list(c(parms)), {
   res <- p1*4 + p2^2*x.vals^2 + p3*sin(x.vals) + p4*x.vals - exp(p5*x.vals)
   diff <- sum((res - y.vals)^2)
 })
}


#perform model selection
res <- famos(init.par = inits,
           fit.fn = cost_function,
           nr.of.data = length(y.values),
           homedir = tempdir(),
           init.model.type = c("p2", "p3"),
           optim.runs = 1,
           x.vals = x.values,
           y.vals = y.values)

#get results
return.results(homedir = tempdir(), mrun = res$mrun)

#delete tempdir
unlink(paste0(tempdir(),"/FAMoS-Results"), recursive = TRUE)
```

`sc.order` *Plot Model selection Criteria*

Description

Plots the selection criteria of the tested models in ascending order.

Usage

```
sc.order(input = getwd(), mrun = NULL, number = NULL,
  colour.par = NULL, save.output = NULL, ...)
```

Arguments

input	Either a string containing the directory which holds the "FAMoS-Results" folder or a matrix containing the tested models along with the respective selection criteria. Default to getwd().
mrun	A string giving the number of the corresponding FAMoS run, e.g "004". If NULL (default), all FAMoS runs in the folder will be used for evaluation.
number	Specifies the number of models that will be plotted. If NULL (default), all tested models will be used for plotting.
colour.par	The name of a model parameter. All models containing this parameter will be coloured red. Default to NULL.
save.output	A string containing the location and name under which the figure should be saved (format is .pdf). Default to NULL.
...	Additional parameters that will be passed on to barplot.

Value

Barplot showing the ordered selection criteria of the tested models. Also returns a data frame containing each unique tested model with its best selection criteria.

Examples

```
#plot the selection criteria
sc.order(input = famos.run)
sc.order(input = famos.run, colour.par = "p1")
```

---

set.crit.parms *Convert Critical and Swap Sets*

Description

Converts the user-specified list containing the names of the critical or swap parameters into a format that is more convenient for calculations.

Usage

```
set.crit.parms(critical.parameters, all.names)
```

Arguments

critical.parameters	A list containing vectors which specify either critical parameter sets or swap parameter sets (see Details).
all.names	A vector containing the names of all parameters.

Details

Critical sets are parameters sets, of which at least one per set has to be present in each tested model. If a model violates on of the critical conditions, it will not be fitted (see also `model.appr`). On the other hand, swap parameter sets define which parameters are interchangeable. For more information see `famos`.

Value

A list containing the indices of the respective critical or swap sets.

Examples

```
#set critical set and names
crits <- list(c("p1", "p2"), c("p5"))
par.names <- c("p1", "p2", "p3", "p4", "p5")
#convert the critical conditions
set.crit.parms(critical.parameters = crits, all.names = par.names)
```

Index