Package ‘ino’

June 5, 2023

Title Initialization of Numerical Optimization

Version 1.0.1

Description Analysis of the initialization for numerical optimization of real-valued functions, including likelihood functions of statistical models. See <https://loelschlaeger.de/ino/> for more details.

License GPL (>= 3)

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.2.3

Imports cli, crayon, doSNOW, dplyr, forcats, foreach, ggplot2, glue, mvtnorm, reshape2, rlang, scales

Suggests covr, devtools, DT, fHMM, knitr, markdown, purrr, pracma, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3


BugReports https://github.com/loelschlaeger/ino/issues

VignetteBuilder knitr

Depends R (>= 4.0.0), optimizeR (>= 0.3.1)

NeedsCompilation no

Author Lennart Oelschläger [aut, cre]
          (<https://orcid.org/0000-0001-5421-9313>),
          Marius Ötting [aut] (<https://orcid.org/0000-0002-9373-0365>),
          Dietmar Bauer [ctb]

Maintainer Lennart Oelschläger <oelschlaeger.lennart@gmail.com>

Repository CRAN

Date/Publication 2023-06-05 18:30:06 UTC
**Description**

Ackley function

**Usage**

```r
defackley(x)
```

**Arguments**

`x`  
A numeric vector of length 2.

**Details**

The function has multiple local minima and one global minimum in the origin.

**Value**

The function value at `x`, a single numeric.
References

https://en.wikipedia.org/wiki/Ackley_function

Examples

f_ackley(c(0, 0))

f_beale

Beale function

Description

Beale function

Usage

f_beale(x)

Arguments

x  A numeric vector of length 2.

Details

The function has multiple local minima and one global minimum in (3, 0.5).

Value

The function value at x, a single numeric.

References

https://en.wikipedia.org/wiki/Test_functions_for_optimization

Examples

f_beale(c(3, 0.5))
### f_easom

**Easom function**

**Description**

Easom function

**Usage**

\[ f_{\text{easom}}(x) \]

**Arguments**

- \( x \) A numeric vector of length 2.

**Details**

The function has multiple local minima and one global minimum in \((\pi, \pi)\).

**Value**

The function value at \( x \), a single numeric.

**References**

https://en.wikipedia.org/wiki/Test_functions_for_optimization

**Examples**

\[ f_{\text{easom}}(c(\pi, \pi)) \]

---

### f_ll_hmm

**Log-likelihood function of a Gaussian-hidden Markov model**

**Description**

Log-likelihood function of a Gaussian-hidden Markov model

**Usage**

\[ f_{\text{ll_hmm}}(\theta, \text{data}, N, \text{neg} = \text{FALSE}) \]
Arguments

theta  A numeric vector of model parameters.

• The first $N*(N-1)$ elements are the logarithms of the non-diagonal elements of the transition probability matrix.
• The next $N$ elements are the mean values of the state-dependent normal distributions.
• The last $N$ elements are the logarithms of the standard deviations of the state-dependent normal distributions.

data A numeric vector, the time series data.

N An integer, the number of states.

neg Set to TRUE to return the negative log-likelihood value.

Value

A numeric, the log-likelihood value at theta given data.

References


Examples

theta <- c(-1, -1, -2, 2, 0.5, 0.5)
data <- sim_hmm(Tp = 1000, N = 2, theta = theta)
f_ll_hmm(theta = theta, data = data, N = 2)

nlm(f_ll_hmm, p = theta, data = data, N = 2, neg = TRUE)$estimate
Arguments

theta  A numeric, the vector of model coefficients. The order of theta is c(b, o, l), see the details.
data  A data.frame, the output of sim_mnp.
neg  Set to TRUE to return the negative log-likelihood value.
normal_cdf  A function that evaluates the n-variate normal CDF. It must take the arguments
  • lower, the vector of lower limits of length n,
  • upper, the vector of upper limits of length n,
  • mean, the mean vector of length n,
  • sigma, the n times n covariance matrix,
and return a single numeric.
  By default, normal_cdf = mvtnorm::pmvnorm.
threshold  A small numeric between 0 and 1. Choice probabilities below this value are set to this
  value to avoid numerical problems.

Details

The order of theta is supposed to be c(b, o, l), where
  • b is the vector of mean effects without the first entry,
  • o the lower-triangular elements of the lower-triangular Cholesky root of the effect covariance
    matrix Omega (if any),
  • and l the lower-triangular elements of the lower-triangular Cholesky root of the differenced
    (with respect to the last alternative) error term covariance matrix Sigma.

Value

A numeric, the log-likelihood value at theta given data.

References

https://en.wikipedia.org/wiki/Multinomial_probit

See Also

sim_mnp() for simulating a data set from a probit model.

Examples

data <- sim_mnp(N = 500, J = 3, P = 2, b = c(1, 3), Sigma = diag(3))
theta <- attr(data, "true")
f_ll_mnp(theta = theta, data = data)

nlm(f_ll_mnp, p = theta, data = data, neg = TRUE)$estimate
Description

Matyas function

Usage

f_matyas(x)

Arguments

x A numeric vector of length 2.

Details

The function has multiple local minima and one global minimum in the origin.

Value

The function value at x, a single numeric.

References

https://en.wikipedia.org/wiki/Test_functions_for_optimization

Examples

f_matyas(c(0, 0))

Description

Example application to HMM likelihood

Usage

data("hmm_ino")

Format

A Nop object.
**Description**

This function checks whether the input is proper count, i.e., a single, positive integer.

**Usage**

`is_count(x, allow_zero = FALSE, error = TRUE)`

**Arguments**

- `x`: Any object.
- `allow_zero`: Either `TRUE` to allow a zero value, or `FALSE` (default) if not.
- `error`: In the case that `x` is not a proper count, either `TRUE` (default) to throw an error or `FALSE` to return invisibly `FALSE`.

**Value**

If `error = TRUE`, either invisibly `TRUE` or an error is thrown. If `error = FALSE`, invisibly `TRUE` or `FALSE`.

---

**Description**

This function checks whether the input is proper index vector, i.e., a vector of positive integer values.

**Usage**

`is_index_vector(x, error = TRUE)`

**Arguments**

- `x`: Any object.
- `error`: In the case that `x` is not a proper index vector, either `TRUE` (default) to throw an error or `FALSE` to return invisibly `FALSE`.

**Value**

If `error = TRUE`, either invisibly `TRUE` or an error is thrown. If `error = FALSE`, invisibly `TRUE` or `FALSE`. 
is_name

Check for proper name

Description

This function checks whether the input is a proper name, i.e., a single (non-trivial) character.

Usage

is_name(x, error = TRUE)

Arguments

x Any object.
error In the case that x is not a proper name, either TRUE (default) to throw an error or FALSE to return invisibly FALSE.

Value

If error = TRUE, either invisibly TRUE or an error is thrown. If error = FALSE, invisibly TRUE or FALSE.

is_name_vector

Check for proper name vector

Description

This function checks whether the input is a proper name vector, i.e., a vector of (non-trivial) character values.

Usage

is_name_vector(x, error = TRUE)

Arguments

x Any object.
error In the case that x is not a proper name vector, either TRUE (default) to throw an error or FALSE to return invisibly FALSE.

Value

If error = TRUE, either invisibly TRUE or an error is thrown. If error = FALSE, invisibly TRUE or FALSE.
is_proportion

Description

This function checks whether the input is proper proportion, i.e., a single numeric.

Usage

is_proportion(x, error = TRUE)

Arguments

x  Any object.
error  In the case that x is not a proper proportion, either TRUE (default) to throw an error or FALSE to return invisibly FALSE.

Value

If error = TRUE, either invisibly TRUE or an error is thrown. If error = FALSE, invisibly TRUE or FALSE.

is_number

Check for proper number

Description

This function checks whether the input is proper number, i.e., a single numeric.

Usage

is_number(x, error = TRUE)

Arguments

x  Any object.
error  In the case that x is not a proper number, either TRUE (default) to throw an error or FALSE to return invisibly FALSE.

Value

If error = TRUE, either invisibly TRUE or an error is thrown. If error = FALSE, invisibly TRUE or FALSE.
is_time_limit

Check for proper time limit

Description
This function checks whether the input is proper time limit, i.e., a single, positive numeric.

Usage
is_time_limit(x, error = TRUE)

Arguments
- x: Any object.
- error: In the case that x is not a proper time limit, either TRUE (default) to throw an error or FALSE to return invisibly FALSE.

Value
If error = TRUE, either invisibly TRUE or an error is thrown. If error = FALSE, invisibly TRUE or FALSE.

is_TRUE_FALSE

Check for proper boolean

Description
This function checks whether the input is proper boolean, i.e., either TRUE or FALSE.

Usage
is_TRUE_FALSE(x, error = TRUE)

Arguments
- x: Any object.
- error: In the case that x is not a proper boolean, either TRUE (default) to throw an error or FALSE to return invisibly FALSE.

Value
If error = TRUE, either invisibly TRUE or an error is thrown. If error = FALSE, invisibly TRUE or FALSE.
Example application to mixture likelihood

Description

See the introduction vignette for details: https://loelschlaeger.de/ino/articles/ino.html

Usage

```r
data("mixture_ino")
```

Format

A `Nop` object.

Value

A `Nop` object, which is an R6 class that specifies the numerical optimization problem, stores optimization results, and provides methods for analyzing the results, see the details.

Getting Started

**Step 1: Create a new `Nop` object:**

Call `object <- Nop$new(f, npar, ...) where`
- `f` is the function to be optimized over its first argument,
- `npar` is the length of the first argument of `f`,
- and `...` are additional arguments for `f`.

**Step 2: Specify numerical optimizer:**

Call `object$set_optimizer(<optimizer object>)`, where `<optimizer object>` is an object of class optimizer, which can be created with the `define_optimizer` function from the {optimR} package. Two optimizer objects are already available:
- `optimizer_nlm`
- `optimizer_optim`

**Step 3: Test the configuration:**

Call `object$test()` to validate your configuration. An overview of the configuration yields the `$print()` method.
Optimization

Call object$evaluate() to evaluate the target function at some point. Call object$optimize() for optimization. See also the methods object$standardize(), object$reduce(), and object$continue() for initialization strategies.

Analysis of the results

The following are methods for the analysis of optimization results, with different filter options for optimization runs, optimizers, and elements:

- $results() returns all saved optimization results,
- $summary() summarizes the results,
- $optima() returns a frequency table of identified optima,
- $plot() visualizes the optimization time or value,
- $best_parameter() returns the parameter vector at which the optimum value is obtained,
- $best_value() returns the found optimum value of \( f \),
- $closest_parameter() returns parameter closest to a specified value.

Active bindings

- \( f \) The function to be optimized.
- f_name The name of the function to be optimized.
- f_target The name of the target argument, i.e., the argument over which \( f \) is optimized.
- npar The length of the target argument, i.e., the argument over which \( f \) is optimized.
- arguments A list of specified additional arguments for \( f \).
- true_value The true numeric optimum value of \( f \) (if available).
- true_parameter The true optimum numeric parameter vector of length npar (if available), i.e., the point where \( f \) obtains its optimum.
- minimized A logical, set to TRUE (default) to show best minimum in $best_value(), $best_parameter(), and $optima().
- optimizer A list of specified optimizers.
- new_label A character, a new optimization label that has not been used yet.

Methods

Public methods:

- Nop$new()
- Nop$print()
- Nop$set_argument()
- Nop$get_argument()
- Nop$remove_argument()
- Nop$set_optimizer()
- Nop$remove_optimizer()
• Nop$evaluate()
• Nop$optimize()
• Nop$test()
• Nop$standardize()
• Nop$reduce()
• Nop$reset_argument()
• Nop$continue()
• Nop$results()
• Nop$number_runs()
• Nop$elements_available()
• Nop$clear()
• Nop$summary()
• Nop$optima()
• Nop$plot()
• Nop$deviation()
• Nop$best_value()
• Nop$best_parameter()
• Nop$closest_parameter()
• Nop$clone()

**Method new():** Creates a new Nop object.

*Usage:*
Nop$new(f, npar, ...)

*Arguments:*

f The function to be optimized. It is optimized over its first argument, which should be a numeric vector of length npar.
npar An integer, the length of the first argument of f (the argument over which f is optimized).
... Optionally additional and named arguments for f.

*Returns:* A new Nop object.

**Method print():** Prints details of the numerical optimization problem.

*Usage:*
Nop$print(digits = getOption("ino_digits", default = 2), ...)

*Arguments:*
digits An integer, the number of shown decimal places. The default is 2.
... Currently not used.

*Returns:* Invisibly the Nop object.

**Method set_argument():** Sets additional arguments for f.

*Usage:*
Nop$set_argument(...)


Arguments:
... Optionally additional named arguments for f.

Returns: Invisibly the Nop object.

**Method** get_argument(): Gets an argument value for f.

Usage:
Nop$get_argument(argument_name)

Arguments:
argument_name A character, the argument to extract.

Returns: The argument.

**Method** remove_argument(): Removes an additional argument for f.

Usage:
Nop$remove_argument(argument_name)

Arguments:
argument_name A character, the argument to remove.

Returns: Invisibly the Nop object.

**Method** set_optimizer(): Sets a numerical optimizer.

Usage:
Nop$set_optimizer(optimizer, label = NULL)

Arguments:
optimizer An object of class optimizer, which can be created with the define_optimizer function from the {optimizeR} package.
label A character, a unique label for the optimizer. By default label = NULL, in which case the default label saved inside optimizer is used.

Returns: Invisibly the Nop object.

**Method** remove_optimizer(): Removes numerical optimizer.

Usage:
Nop$remove_optimizer(
  which_optimizer,
  verbose = getOption("ino_verbose", default = TRUE)
)

Arguments:
which_optimizer Select specified numerical optimizers. Either:
  • "all", all specified optimizers,
  • "removed", all removed optimizers,
  • "active", all active optimizers (i.e., not removed),
  • a character (vector) of specified optimizer labels,
  • a numeric (vector) of optimizer ids (see $print() output).
verbose A logical, which indicates whether progress/details should be printed. Set to TRUE (FALSE) to print (hide) such messages. The default is TRUE.
Returns: Invisibly the Nop object.

Method evaluate(): Evaluates the function.

Usage:
Nop$evaluate(at = rnorm(self$npar), time_limit = NULL, hide_warnings = FALSE)

Arguments:
at A numeric vector of length npar, the point where the function is evaluated. Per default, at
  = rnorm(self$npar), i.e., random values drawn from a standard normal distribution.

time_limit An integer, the time limit in seconds for computations. No time limit if time_limit
  = NULL (the default). This currently only works reliably under Windows.

hide_warnings A logical. Set to TRUE (FALSE) to hide (show) warning messages.

Returns: Either:
  • a numeric value, the function value at at,
  • "time limit reached" if the time limit was reached,
  • the error message if the evaluation failed.

Method optimize(): Optimizes the function.

Usage:
Nop$optimize(
  initial = "random",
  runs = 1,
  which_optimizer = "all",
  seed = NULL,
  return_results = FALSE,
  save_results = TRUE,
  label = self$new_label,
  ncores = getOption("ino_ncores", default = 1),
  verbose = getOption("ino_verbose", default = TRUE),
  simplify = TRUE,
  time_limit = NULL,
  hide_warnings = TRUE,
  fail_bad_initial = TRUE
)

Arguments:
initial Specify the initial point where the optimizer should start. Either:
  • the character "random" (the default) for random initial values drawn from a standard
    normal distribution,
  • a numeric vector of length npar, the starting point for optimization,
  • a list of such vectors (in this case, runs is set to the length of the list),
  • or a function without any arguments that returns a numeric vector of length npar. In
    all these cases, the same initial values are used for each optimizer. For more flexibility,
    a function for initial can have two arguments, where the first argument specifies the
    optimization run, and the second argument specifies the optimizer.

runs An integer, the number of optimization runs. By default, runs = 1.

which_optimizer Select specified numerical optimizers. Either:
• "all", all specified optimizers,
• "removed", all removed optimizers,
• "active", all active optimizers (i.e., not removed),
• a character (vector) of specified optimizer labels,
• a numeric (vector) of optimizer ids (see $print() output).

seed  An integer, passed to set.seed for reproducibility. Can be NULL for no seed, which is the default.

return_results A logical, which indicates whether the optimization results should be returned as a list. By default, return_results = FALSE.

save_results A logical, which indicates whether the optimization results should be saved inside the Nop object. By default, save_results = TRUE.

label Only relevant if save_results = TRUE. In this case, an optional character to specify a custom label of the optimization. By default, label = self$new_label creates a new label. Labels can be useful to distinguish optimization runs later.

ncores An integer, the number of CPU cores for parallel computation. The default is 1. You can use parallel::detectCores() to detect the number of available CPU cores.

verbose A logical, which indicates whether progress/details should be printed. Set to TRUE (FALSE) to print (hide) such messages. The default is TRUE.

simplify Only relevant if return_results = TRUE and runs = 1 and/or only one optimizer is specified. In this case, if simplify = TRUE, the nested list output of optimization results is flattened if possible.

time_limit An integer, the time limit in seconds for computations. No time limit if time_limit = NULL (the default). This currently only works reliably under Windows.

hide_warnings A logical. Set to TRUE (FALSE) to hide (show) warning messages.

fail_bad_initial Either TRUE to immediately fail if initial contains any misspecifications (default), or FALSE to include the failed runs in the results.

Returns:  The return value depends on the value of return_results:
• if return_results = FALSE, invisibly the Nop object,
• if return_results = TRUE, a nested list of optimization results. Each element corresponds to one optimization run and is a list of results for each optimizer. The results for each optimizer is a list, the output of apply_optimizer. If simplify = TRUE, the output is flattened if possible.

Method test():  Validates the configuration of a Nop object.

Usage:
Nop$test(
at = rnorm(self$npar),
which_optimizer = "active",
time_limit = 10,
verbose = getOption("ino_verbose", default = TRUE),
digits = getOption("ino_digits", default = 2)
)

Arguments:
at A numeric of length npar, the point at which the function f and the specified optimizer are tested. Per default, at = rnorm(self$npar), i.e., random values drawn from a standard normal distribution.
which_optimizer  Select specified numerical optimizers. Either:
• "all", all specified optimizers,
• "removed", all removed optimizers,
• "active", all active optimizers (i.e., not removed),
• a character (vector) of specified optimizer labels,
• a numeric (vector) of optimizer ids (see $print() output).
time_limit  An integer, the time limit in seconds for computations. No time limit if time_limit
= NULL (the default). This currently only works reliably under Windows.
verbose  A logical, which indicates whether progress/details should be printed. Set to TRUE
(FALSE) to print (hide) such messages. The default is TRUE.
digits  An integer, the number of shown decimal places. The default is 2.

Returns:  Invisibly TRUE if the tests are successful.

Method  standardize():  Standardizes the optimization problem.

Usage:
Nop$standardize(
  argument_name,
  by_column = TRUE,
  center = TRUE,
  scale = TRUE,
  ignore = integer(),
  verbose = getOption("ino_verbose", default = TRUE)
)

Arguments:
argument_name  A character, the name of the argument of f to be standardized. The argument
must a numeric vector, matrix, or data.frame.
by_column  Only relevant if the argument argument_name is a matrix or a data.frame. In that
case, either TRUE to standardize column-wise (default) or FALSE to standardize row-wise.
Currently, only by_column = TRUE is implemented.
center  Passed to scale. Default is TRUE.
scale  Passed to scale. Default is TRUE.
ignore  A integer (vector) of column indices (or row indices if by_column = FALSE) to not
standardize.
verbose  A logical, which indicates whether progress/details should be printed. Set to TRUE
(FALSE) to print (hide) such messages. The default is TRUE.

Returns:  Invisibly the Nop object.

Method  reduce():  Reduces the optimization problem.

Usage:
Nop$reduce(
  argument_name,
  by_row = TRUE,
  how = "random",
  proportion = 0.5,
  centers = 2,
ignore = integer(),
seed = NULL,
verbose = getOption("ino_verbose", default = TRUE)
)

Arguments:

argument_name  A character, the name of the argument of f to be reduced.
by_row Only relevant if the argument argument_name is a matrix or a data.frame. In that
case, either TRUE to reduce row-wise (default) or FALSE to reduce column-wise. Currently,
only by_row = TRUE is implemented.
how  A character, specifying how to reduce. Can be one of:
  • "random" (default), reduce at random
  • "first", reduce to the first elements
  • "last", reduce to the last elements
  • "similar", reduce to similar elements
  • "dissimilar", reduce to dissimilar elements Note that "similar" and "dissimilar"
    are based on k-means clustering via kmeans. To apply these options, the argument
    argument_name must be numeric.
proportion  A numeric between 0 and 1, specifying the reduction proportion. By default,
  proportion = 0.5.
centers  Only relevant, if how = "(dis)similar". In that case, passed to kmeans. By default,
  centers = 2.
ignore  Only relevant, if how = "(dis)similar". In that case a integer (vector) of row indices
  (or column indices if by_row = FALSE) to ignore for clustering.
seed An integer, passed to set.seed for reproducibility. Can be NULL for no seed, which is
  the default.
verbose A logical, which indicates whether progress/details should be printed. Set to TRUE
  (FALSE) to print (hide) such messages. The default is TRUE.

Returns: Invvisibly the Nop object.

Method reset_argument(): Resets an additional argument for f after transformation with
$standardize() or $reduce().

Usage:

Nop$reset_argument(
  argument_name,
  verbose = getOption("ino_verbose", default = TRUE)
)

Arguments:

argument_name  A character, the name of the argument to reset.
verbose A logical, which indicates whether progress/details should be printed. Set to TRUE
  (FALSE) to print (hide) such messages. The default is TRUE.

Returns: Invvisibly the Nop object.

Method continue(): Continues optimization runs, e.g., with a transformed parameter.

Usage:
Nop$continue(
  which_run = "last",
  which_optimizer = "active",
  seed = NULL,
  return_results = FALSE,
  save_results = TRUE,
  ncores = getOption("ino_ncores", default = 1),
  verbose = getOption("ino_verbose", default = TRUE),
  simplify = TRUE,
  time_limit = NULL,
  hide_warnings = TRUE
)

Arguments:

which_run  Select saved results of optimization runs. Either:
  • "all", all results,
  • "last", the results from the last optimization,
  • "failed", the results from failed optimization runs,
  • a character (vector) of labels specified in $optimize(),
  • a numeric (vector) of run ids.

which_optimizer  Select specified numerical optimizers. Either:
  • "all", all specified optimizers,
  • "removed", all removed optimizers,
  • "active", all active optimizers (i.e., not removed),
  • a character (vector) of specified optimizer labels,
  • a numeric (vector) of optimizer ids (see $print() output).

seed  An integer, passed to set.seed for reproducibility. Can be NULL for no seed, which is the default.

return_results  A logical, which indicates whether the optimization results should be returned as a list. By default, return_results = FALSE.

save_results  A logical, which indicates whether the optimization results should be saved inside the Nop object. By default, save_results = TRUE.

ncores  An integer, the number of CPU cores for parallel computation. The default is 1. You can use parallel::detectCores() to detect the number of available CPU cores.

verbose  A logical, which indicates whether progress/details should be printed. Set to TRUE (FALSE) to print (hide) such messages. The default is TRUE.

simplify  Only relevant if return_results = TRUE and runs = 1 and/or only one optimizer is specified. In this case, if simplify = TRUE, the nested list output of optimization results is flattened if possible.

time_limit  An integer, the time limit in seconds for computations. No time limit if time_limit = NULL (the default). This currently only works reliably under Windows.

hide_warnings  A logical. Set to TRUE (FALSE) to hide (show) warning messages.

Returns:  The same as the return value of $optimize().

Method results():  Returns saved optimization results.

Usage:
Nop$results(
  which_run = "all",
  which_optimizer = "all",
  which_element = "all",
  only_comparable = FALSE,
  simplify = TRUE
)

**Arguments:**

- **which_run** Select saved results of optimization runs. Either:
  - "all", all results,
  - "last", the results from the last optimization,
  - "failed", the results from failed optimization runs,
  - a character (vector) of labels specified in $optimize(),
  - a numeric (vector) of run ids.

- **which_optimizer** Select specified numerical optimizers. Either:
  - "all", all specified optimizers,
  - "removed", all removed optimizers,
  - "active", all active optimizers (i.e., not removed),
  - a character (vector) of specified optimizer labels,
  - a numeric (vector) of optimizer ids (see $print() output).

- **which_element** Select elements of saved optimization results. See $elements_available() for the names of all available elements. Either:
  - "all", all elements,
  - "basic", the elements
    - "value", the numeric value of the found optimum,
    - "parameter", the parameter vector at which the optimum value is obtained,
  - "default", the elements that are saved for all optimization runs by default, i.e.
    - "run", the run id,
    - "optimizer", the label for the optimizer,
    - "value" and "parameter" (see above),
    - "seconds", the optimization time in seconds,
    - "label", the label for the optimization run,
  - a character (vector), names of specific elements in the optimization output.

- **only_comparable** Either TRUE to show only comparable results (i.e., results obtained for the original optimization problem), or FALSE to include all optimization results.

- **simplify** Only relevant if return_results = TRUE and runs = 1 and/or only one optimizer is specified. In this case, if simplify = TRUE, the nested list output of optimization results is flattened if possible.

**Returns:** A nested list of optimization results. Each element corresponds to one optimization run and is a list of results for each optimizer. If simplify = TRUE, the output is flattened if possible.

**Method** `number_runs()`: Returns the number of saved optimization runs.

**Usage:**
Nop$number_runs(
  which_run = "all",
  which_optimizer = "all",
  only_comparable = FALSE
)

Arguments:

which_run  Select saved results of optimization runs. Either:
  • "all", all results,
  • "last", the results from the last optimization,
  • "failed", the results from failed optimization runs,
  • a character (vector) of labels specified in $optimize(),
  • a numeric (vector) of run ids.

which_optimizer  Select specified numerical optimizers. Either:
  • "all", all specified optimizers,
  • "removed", all removed optimizers,
  • "active", all active optimizers (i.e., not removed),
  • a character (vector) of specified optimizer labels,
  • a numeric (vector) of optimizer ids (see $print() output).

only_comparable  Either TRUE to show only comparable results (i.e., results obtained for the original optimization problem), or FALSE to include all optimization results.

Returns:  An integer.

Method elements_available(): Returns names of available elements per optimizer.

Usage:
Nop$elements_available(which_optimizer = "all")

Arguments:

which_optimizer  Select specified numerical optimizers. Either:
  • "all", all specified optimizers,
  • "removed", all removed optimizers,
  • "active", all active optimizers (i.e., not removed),
  • a character (vector) of specified optimizer labels,
  • a numeric (vector) of optimizer ids (see $print() output).

Returns:  A list.

Method clear(): Deletes optimization results.

Usage:
Nop$clear(which_run, which_optimizer = "all", which_element = "all")

Arguments:

which_run  Select saved results of optimization runs. Either:
  • "all", all results,
  • "last", the results from the last optimization,
  • "failed", the results from failed optimization runs,
• a character (vector) of labels specified in $optimize(),
• a numeric (vector) of run ids.

which_optimizer Select specified numerical optimizers. Either:
• "all", all specified optimizers,
• "removed", all removed optimizers,
• "active", all active optimizers (i.e., not removed),
• a character (vector) of specified optimizer labels,
• a numeric (vector) of optimizer ids (see $print() output).

which_element Select elements of saved optimization results. See $elements_available() for the names of all available elements. Either:
• "all", all elements,
• "basic", the elements
  – "value", the numeric value of the found optimum,
  – "parameter", the parameter vector at which the optimum value is obtained,
• "default", the elements that are saved for all optimization runs by default, i.e.
  – "run", the run id,
  – "optimizer", the label for the optimizer,
  – "value" and "parameter" (see above),
• "default", the elements that are saved for all optimization runs by default, i.e.
  – "run", the run id,
  – "optimizer", the label for the optimizer,
  – "seconds", the optimization time in seconds,
  – "label", the label for the optimization run,
• a character (vector), names of specific elements in the optimization output.

Returns: Invisibly the Nop object.

Method summary(): Provides an overview of the optimization runs.

Usage:
Nop$summary(
  which_element = "basic",
  which_run = "all",
  which_optimizer = "all",
  digits = getOption("ino_digits", default = 2),
  only_comparable = FALSE,
  ...
)

Arguments:
which_element Select elements of saved optimization results. See $elements_available() for the names of all available elements. Either:
• "all", all elements,
• "basic", the elements
  – "value", the numeric value of the found optimum,
  – "parameter", the parameter vector at which the optimum value is obtained,
• "default", the elements that are saved for all optimization runs by default, i.e.
  – "run", the run id,
  – "optimizer", the label for the optimizer,
- "value" and "parameter" (see above),
- "seconds", the optimization time in seconds,
- "label", the label for the optimization run,
- a character (vector), names of specific elements in the optimization output.

which_run Select saved results of optimization runs. Either:
  • "all", all results,
  • "last", the results from the last optimization,
  • "failed", the results from failed optimization runs,
  • a character (vector) of labels specified in $optimize(),
  • a numeric (vector) of run ids.

which_optimizer Select specified numerical optimizers. Either:
  • "all", all specified optimizers,
  • "removed", all removed optimizers,
  • "active", all active optimizers (i.e., not removed),
  • a character (vector) of specified optimizer labels,
  • a numeric (vector) of optimizer ids (see $print() output).

digits An integer, the number of shown decimal places. The default is 2.

only_comparable Either TRUE to show only comparable results (i.e., results obtained for the
  original optimization problem), or FALSE to include all optimization results.

... Optionally named expressions of elements. See $elements_available() for the names
  of all available elements. In addition, "true_value", "true_parameter", "best_value",
  and "best_parameter" can be accessed. As an example, you could add
distance = "sqrt(sum((parameter - true_parameter) ^ 2))" for the euclidean distance between the estimated and true parameter vector.

Returns: A data.frame with optimization results.

Method optima(): Provides an overview of the identified optimum values.

Usage:
Nop$optima(
digits = getOption("ino_digits", default = 2),
  sort_by = "frequency",
  which_run = "all",
  which_optimizer = "all",
  only_comparable = TRUE
)

Arguments:
digits An integer, the number of shown decimal places. The default is 2.
sort_by Either:
  • "frequency" (default) to sort rows by frequency,
  • "value" to sort by value.
which_run Select saved results of optimization runs. Either:
  • "all", all results,
  • "last", the results from the last optimization,
• "failed", the results from failed optimization runs,
• a character (vector) of labels specified in $optimize() ,
• a numeric (vector) of run ids.

which_optimizer  Select specified numerical optimizers. Either:
• "all", all specified optimizers,
• "removed", all removed optimizers,
• "active", all active optimizers (i.e., not removed),
• a character (vector) of specified optimizer labels,
• a numeric (vector) of optimizer ids (see $print() output).

only_comparable  Either TRUE to show only comparable results (i.e., results obtained for the
original optimization problem), or FALSE to include all optimization results.

Returns:  A data.frame.

Method  plot(): Visualizes the optimization time or value.

Usage:
Nop$plot(
    which_element = "seconds",
    by = NULL,
    relative = FALSE,
    which_run = "all",
    which_optimizer = "all",
    only_comparable = FALSE,
    title = paste("Optimization of", x$f_name),
    xlim = c(NA, NA)
)

Arguments:
which_element  Either:
• "seconds" to plot the optimization times (default)
• "value" to plot the optimization values
by  Either:
• "label" to group by optimization label
• "optimizer" to group by optimizer
• NULL to not group (default)
relative  Only if which_element = "seconds". In this case, set to TRUE to plot relative time
differences with respect to the overall median.
which_run  Select saved results of optimization runs. Either:
• "all", all results,
• "last", the results from the last optimization,
• "failed", the results from failed optimization runs,
• a character (vector) of labels specified in $optimize(),
• a numeric (vector) of run ids.
which_optimizer  Select specified numerical optimizers. Either:
• "all", all specified optimizers,
• "removed", all removed optimizers,
• "active", all active optimizers (i.e., not removed),
• a character (vector) of specified optimizer labels,
• a numeric (vector) of optimizer ids (see $print()$ output).

only_comparable Either TRUE to show only comparable results (i.e., results obtained for the
original optimization problem), or FALSE to include all optimization results.
title A character, the plot title.
xlim Passed on to coord_cartesian.

Returns: A ggplot object.

Method deviation(): Visualizes deviation of parameters.

Usage:
Nop$deviation(
  reference = self$true_parameter,
  which_element = "initial",
  which_run = "all",
  which_optimizer = "all",
  only_comparable = FALSE,
  title = "Parameter deviation",
  ylim = c(NA, NA),
  parameter_labels = paste0("theta", 1:self$npar)
)

Arguments:
reference A numeric of length self$npar, the reference parameters.
which_element Either:
• "initial" to compute deviations to the initial values (default)
• "parameter" to compute deviations to the estimated parameters
which_run Select saved results of optimization runs. Either:
• "all", all results,
• "last", the results from the last optimization,
• "failed", the results from failed optimization runs,
• a character (vector) of labels specified in $optimize(),
• a numeric (vector) of run ids.
which_optimizer Select specified numerical optimizers. Either:
• "all", all specified optimizers,
• "removed", all removed optimizers,
• "active", all active optimizers (i.e., not removed),
• a character (vector) of specified optimizer labels,
• a numeric (vector) of optimizer ids (see $print()$ output).
only_comparable Either TRUE to show only comparable results (i.e., results obtained for the
original optimization problem), or FALSE to include all optimization results.
title A character, the plot title.
ylim Passed on to coord_cartesian.
**Method** best_value(): Returns the best found numeric value of \( f \).

**Usage:**

\[
\text{Nop\$best\_value(}
\begin{array}{c}
\text{which\_run = "all",} \\
\text{which\_optimizer = "all",} \\
\text{only\_comparable = TRUE,} \\
\text{digits = getOption("ino\_digits", default = 2)}
\end{array}
\)

**Arguments:**

- **which_run** Select saved results of optimization runs. Either:
  - "all", all results,
  - "last", the results from the last optimization,
  - "failed", the results from failed optimization runs,
  - a character (vector) of labels specified in optimize(),
  - a numeric (vector) of run ids.

- **which_optimizer** Select specified numerical optimizers. Either:
  - "all", all specified optimizers,
  - "removed", all removed optimizers,
  - "active", all active optimizers (i.e., not removed),
  - a character (vector) of specified optimizer labels,
  - a numeric (vector) of optimizer ids (see print() output).

- **only_comparable** Either TRUE to show only comparable results (i.e., results obtained for the original optimization problem), or FALSE to include all optimization results.

- **digits** An integer, the number of shown decimal places. The default is 2.

**Returns:** A numeric, the best found numeric value of \( f \). The output has two attributes:

- run, the run id that led to the best value,
- optimizer, the optimizer that led to the best value. In the case that multiple optimization runs led to the best value, the first one of them is returned.

**Method** best_parameter(): Returns the best found numeric parameter vector.

**Usage:**

\[
\text{Nop\$best\_parameter(}
\begin{array}{c}
\text{which\_run = "all",} \\
\text{which\_optimizer = "all",} \\
\text{only\_comparable = TRUE,} \\
\text{digits = getOption("ino\_digits", default = 2)}
\end{array}
\)

**Arguments:**

- **which_run** Select saved results of optimization runs. Either:
Nop

- "all", all results,
- "last", the results from the last optimization,
- "failed", the results from failed optimization runs,
- a character (vector) of labels specified in $optimize(),
- a numeric (vector) of run ids.

which_optimizer Select specified numerical optimizers. Either:
- "all", all specified optimizers,
- "removed", all removed optimizers,
- "active", all active optimizers (i.e., not removed),
- a character (vector) of specified optimizer labels,
- a numeric (vector) of optimizer ids (see $print() output).

only_comparable Either TRUE to show only comparable results (i.e., results obtained for the original optimization problem), or FALSE to include all optimization results.

digits An integer, the number of shown decimal places. The default is 2.

Returns: A numeric of length self$npar. The output has two attributes:
- run, the run id that led to the best parameter vector,
- optimizer, the optimizer that led to the best parameter vector. Note that this parameter vector is not necessarily unique.

Method closest_parameter(): Extracts the parameter vector that led to a function value closest (in absolute value) to value. Note that this parameter vector is not necessarily unique.

Usage:
Nop$closest_parameter(
  value,
  which_run = "all",
  which_optimizer = "all",
  only_comparable = TRUE,
  digits =getOption("ino_digits", default = 2)
)

Arguments:

value A single numeric.

which_run Select saved results of optimization runs. Either:
- "all", all results,
- "last", the results from the last optimization,
- "failed", the results from failed optimization runs,
- a character (vector) of labels specified in $optimize(),
- a numeric (vector) of run ids.

which_optimizer Select specified numerical optimizers. Either:
- "all", all specified optimizers,
- "removed", all removed optimizers,
- "active", all active optimizers (i.e., not removed),
- a character (vector) of specified optimizer labels,
- a numeric (vector) of optimizer ids (see $print() output).
only_comparable Either TRUE to show only comparable results (i.e., results obtained for the original optimization problem), or FALSE to include all optimization results.

digits An integer, the number of shown decimal places. The default is 2.

Returns: A numeric vector of length self$npar. The output has two attributes:
- run, the run id that led to this parameter vector,
- optimizer, the optimizer that led to this parameter vector.

Method clone(): The objects of this class are cloneable with this method.

Usage:
Nop$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Examples
Nop$new(f = f_ackley, npar = 2)@
s_set_optimizer(optimizer_nlm())$
  optimize(runs = 100, verbose = FALSE)$
  optima()

---

probit_ino Example application of to probit likelihood

Description
See the vignette about the probit likelihood for details: https://loelschlaeger.de/ino/articles/example_probit.html

Usage
data(probit_ino)

Format
A Nop object.
Simulate a time series from a Gaussian-hidden Markov model

**Description**

Simulate a time series from a Gaussian-hidden Markov model

**Usage**

`sim_hmm(Tp, N, theta)`

**Arguments**

- **Tp** An integer, the number of observations.
- **N** An integer, the number of states.
- **theta** A numeric vector of model parameters.
  - The first \(N(N-1)\) elements are the logarithms of the non-diagonal elements of the transition probability matrix.
  - The next \(N\) elements are the mean values of the state-dependent normal distributions.
  - The last \(N\) elements are the logarithms of the standard deviations of the state-dependent normal distributions.

**Value**

A numeric vector, the simulated time series.

**See Also**

`f_ll_hmm()` for computing the log-likelihood of a Gaussian-hidden Markov model

**Examples**

```r
 TPM <- matrix(c(0.8, 0.1, 0.2, 0.9), nrow = 2)
 mu <- c(-2, 2)
 sigma <- c(0.5, 1)
 theta <- c(log(TPM[row(TPM) != col(TPM)]), mu, log(sigma))
 data <- sim_hmm(Tp = 100, N = 2, theta = theta)
 plot(data, type = "l")
```
Simulate data from a (normally mixed) multinomial probit model

Description

Simulate data from a (normally mixed) multinomial probit model

Usage

```r
sim_mnp(
  N,
  Tp = 1,
  J,
  P,
  b = stats::rnorm(P),
  Omega = NULL,
  Sigma = diag(J),
  X = function(n, t) matrix(stats::rnorm(J * P), nrow = J, ncol = P),
  seed = NULL
)
```

Arguments

- `N`: An integer, the number of observations.
- `Tp`: An integer, the number of choice occasions. By default, `Tp = 1`.
- `J`: An integer, the number of alternatives. Must be greater or equal 2.
- `P`: An integer, the number of choice covariates.
- `b`: A numeric vector, the mean effects vector of length `P`. The first element must be 1 (for normalization).
- `Omega`: A matrix, the covariance matrix of the normal mixing distribution of dimension `P` times `P`. Set to `NULL` (the default) for no mixing distribution.
- `Sigma`: A matrix, the error term covariance matrix of dimension `J` times `J`.
- `X`: A function that samples the covariates of decider `n` at choice occasion `t`. It must
  - have two arguments `n` and `t`.
  - return a numeric matrix of dimension `J` times `P`.
- `seed`: Optionally set a seed for the choice data simulation.

Value

A data.frame. The first column (`n`) is the identifier for the decider, the next column (`t`) the identifier for the choice occasion. Next comes the column `y` with the indices of the chosen alternatives. The last columns contain the column-wise entries of the covariate matrices.
The true model coefficients are added to the output via the attribute "true". They are already normalized and can be directly compared with the maximum likelihood estimate.

Additional attributes are "J" (the number of alternatives), "P" (the number of choice covariates), and "mix" (a boolean which is TRUE if Omega is not NULL).

See Also

f_ll_mnp() for computing the log-likelihood of a (normally mixed) multinomial probit model.

Examples

sim_mnp(N = 3, J = 3, P = 2, b = c(1, -1), Omega = diag(2), Sigma = diag(3))
Index

* demo
  hmm_ino, 7
  mixture_ino, 12
  probit_ino, 29

* function
  f_ackley, 2
  f_beale, 3
  f_easom, 4
  f_ll_hmm, 4
  f_ll_mnp, 5
  f_matyas, 7
  sim_hmm, 30
  sim_mnp, 31

* utils
  is_count, 8
  is_index_vector, 8
  is_name, 9
  is_name_vector, 9
  is_number, 10
  is_proportion, 10
  is_time_limit, 11
  is_TRUE_FALSE, 11
  kmeans, 19
  mixture_ino, 12
  Nop, 7, 12, 12, 29
  optimizer_nlm, 12
  optimizer_optim, 12
  probit_ino, 29
  scale, 18
  set.seed, 17, 19, 20
  sim_hmm, 30
  sim_mnp, 6, 31
  sim_mnp(), 6

apply_optimizer, 17
coord_cartesian, 26
define_optimizer, 12, 15
f_ackley, 2
f_beale, 3
f_easom, 4
f_ll_hmm, 4
f_ll_hmm(), 30
f_ll_mnp, 5
f_ll_mnp(), 32
f_matyas, 7
ggplot, 26, 27
hmm_ino, 7