Package 'libcmaesr'

August 7, 2026

Title R Interface to 'libcmaes'

Version 0.1.0

Copyright See the file COPYRIGHTS for details on the bundled 'libcmaes' copyrights

Description A lightweight interface to the 'libcmaes' C++ library for the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). CMA-ES is a state-of-the-art evolutionary algorithm for the optimization of difficult non-linear, non-convex black-box functions, as described in Hansen and Ostermeier (2001) <doi:10.1162/106365601750190398>. Supports the active, separable, and VD (diagonal plus rank-one covariance) variants of the algorithm as well as the IPOP (increasing population size) and BIPOP (bi-population) restart strategies. A patched copy of 'libcmaes' (LGPL >= 3) is bundled; see the COPYRIGHTS file for details.

License LGPL (>= 3)

URL https://libcmaesr.mlr-org.com, https://github.com/mlr-org/libcmaesr

BugReports https://github.com/mlr-org/libcmaesr/issues

Depends R (>= 4.2.0)

Imports checkmate, mlr3misc, stats

LinkingTo RcppEigen

Suggests testthat (>= 3.0.0), callr

Config/testthat/edition 3

Config/testthat/parallel false

Encoding UTF-8

Language en-US

NeedsCompilation yes

Config/roxygen2/version 8.0.0

Author Marc Becker [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-8115-0400>), Bernd Bischl [aut] (ORCID: <https://orcid.org/0000-0001-6002-6980>),

Martin Binder [aut],
Lars Kotthoff [aut],
Emmanuel Benazera [ctb, cph] (author of the bundled 'libcmaes' library),
Inria [cph] (copyright holder of parts of the bundled 'libcmaes' library)

Maintainer Marc Becker <marcbecker@posteo.de>

Repository CRAN

Date/Publication 2026-08-07 10:30:02 UTC

Contents

`libcmaesr-package` *libcmaesr: R Interface to 'libcmaes'*

Description

A lightweight interface to the 'libcmaes' C++ library for the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). CMA-ES is a state-of-the-art evolutionary algorithm for the optimization of difficult non-linear, non-convex black-box functions, as described in Hansen and Ostermeier (2001) doi:10.1162/106365601750190398. Supports the active, separable, and VD (diagonal plus rank-one covariance) variants of the algorithm as well as the IPOP (increasing population size) and BIPOP (bi-population) restart strategies. A patched copy of 'libcmaes' (LGPL >= 3) is bundled; see the COPYRIGHTS file for details.

Author(s)

Maintainer: Marc Becker <marcbecker@posteo.de> (ORCID) [copyright holder]

Authors:

- Marc Becker <marcbecker@posteo.de> (ORCID) [copyright holder]
- Bernd Bischl <bernd_bischl@gmx.net> (ORCID)
- Martin Binder <mlr.developer@mb706.com>
- Lars Kotthoff <lk223@st-andrews.ac.uk>

Other contributors:

- Emmanuel Benazera (author of the bundled 'libcmaes' library) [contributor, copyright holder]
- Inria (copyright holder of parts of the bundled 'libcmaes' library) [copyright holder]

See Also

Useful links:

- https://libcmaesr.mlr-org.com
- https://github.com/mlr-org/libcmaesr
- Report bugs at https://github.com/mlr-org/libcmaesr/issues

`cmaes` *Covariance Matrix Adaptation Evolution Strategy*

Description

Implements the CMA-ES variants provided by libcmaes, see here: https://github.com/CMA-ES/libcmaes/ via a very light-weight C wrapper.

The control structure allows access to most control params of the ES, but CMAES is supposed to handle most of them internally. Quoting Niko Hansen from here: https://cma-es.github.io/:

"The CMA-ES does not require a tedious parameter tuning for its application. In fact, the choice of strategy internal parameters is not left to the user (arguably with the exception of population size λ). Finding good (default) strategy parameters is considered as part of the algorithm design, and not part of its application — the aim is to have a well-performing algorithm as is. The default population size λ is comparatively small to allow for fast convergence. Restarts with increasing population size (Auger & Hansen 2005) improve the global search performance. For the application of the CMA-ES, an initial solution, an initial standard deviation (step-size, variables should be defined such that the same standard deviations can be reasonably applied to all variables, see also here) and, possibly, the termination criteria (e.g. a function tolerance) need to be set by the user. The most common applications are model calibration (e.g. curve fitting) and shape optimisation."

Whether you believe in this completely for any problem is up to you, but the general idea is to run it in its defaults, and only change them if you know what you are doing.

1. libcmaes could handle unbounded search spaces, but this is currently not supported, you need to set lower and upper bounds.
2. Noisy functions / noisy handling CMAES is currently not supported.
3. Surrogate variants are currently not supported.
4. Geno-Pheno transformation is automatically applied, in the sense that we use the linear scaling to handle the bounds.
5. Setting gradients is currently not supported.
6. OpenMP is currently not supported. libcmaes uses OpenMP for the population evaluation mainly, but also for some Eigen stuff. Calling into the R Api via threading is not allowed, which would happen in the former.
7. The number of function evaluations is not reported. libcmaes returns the solution object of the best restart run, which only counts the evaluations of that run and therefore undercounts the total whenever restarts happen (`"ipop"`, `"bipop"` and their separable variants), see https://github.com/CMA-ES/libcmaes/issues/258. Count the calls to your objective function yourself if you need this number.

In general, more details can be found here: https://github.com/CMA-ES/libcmaes/wiki/.

Usage

```
cmaes(objective, x0, lower, upper, control = cmaes_control(), batch = FALSE)
```

Arguments

objective	(function(x)) Objective function, to minimize. If batch is FALSE, x is a numeric vector of length n and the function must return a scalar numeric. If batch is TRUE, x is a numeric matrix with one row per candidate and n columns. The number of rows can vary between iterations, e.g., restart strategies like IPOP and BIPOP increase the population size over time, so do not assume it is always lambda. The function must return a numeric vector with one element per row of x. The latter usually reduces overhead and allows you to orchestrate parallelization yourself if you need it because the objective function is more expensive.
x0	(numeric(n)) Initial point. NB: This point is IGNORED if you also set x0_lower and x0_upper in the control object, but it must still be a vector of length n!
lower	(numeric(n)) Lower bounds of search space.
upper	(numeric(n)) Upper bounds of search space.
control	(cmaes_control) A control object created by cmaes_control(). Default is a control object with all parameters set to their default values.
batch	(logical(1)) Whether the objective function evaluates a batch of points at once. Default is FALSE.

Value

(named list). List with elements:

- 'x': (numeric(n))
  The best point found, length corresponds to x0, lower and upper.
- 'y': (numeric(1))
  The objective value of the best point.
- 'edm': (numeric(1))
  Expected distance to the minimum.
- 'time': (numeric(1))
  The time taken to find the solution in seconds.
- 'status_code': (integer(1))
  The status code, indicating success, failure, or the reason for stopping. See here: https://github.com/CMA-ES/libcmaes/wiki/Optimizing-a-function
- 'status_msg': (character(1))
  A human-readable status message from libcmaes.

Examples

```
# minimize a simple quadratic function
objective = function(x) sum(x^2)
control = cmaes_control(seed = 1, max_fevals = 500)
res = cmaes(objective, x0 = c(0.5, 0.5), lower = c(-5, -5), upper = c(5, 5), control = control)
res$x
res$y
```

---

cmaes_algos *CMAES Algorithm Names*

Description

A vector of strings containing the names of the CMAES variants. See https://cma-es.github.io/libcmaes/doc/html/classlibcmaes_1_1CMAParameters.html for details.

Usage

```
cmaes_algos
```

Value

A `character` vector of algorithm names.

Examples

```
cmaes_algos
```

---

cmaes_control *CMA-ES Control Object*

Description

Create a control object for the CMA-ES algorithm. For more information on the parameters, see here: https://cma-es.github.io/libcmaes/doc/html/classlibcmaes_1_1CMAParameters.html.

Usage

```
cmaes_control(
  maximize = FALSE,
  algo = "acmaes",
  max_fevals = 100,
  max_iter = NA_integer_,
  ftarget = NA_real_,
  f_tolerance = NA_real_,
```

```
  x_tolerance = NA_real_,
  lambda = NA_integer_,
  sigma = NA_real_,
  max_restarts = NA_integer_,
  elitism = NA_integer_,
  tpa = NA_integer_,
  tpa_dsigma = NA_real_,
  seed = NA_integer_,
  quiet = TRUE,
  x0_lower = NULL,
  x0_upper = NULL
)
```

Arguments

`maximize`	(`logical(1)`) Whether to maximize the objective function. Default is `FALSE`.
`algo`	(`character(1)`) The CMAES variant to use. Possible values are: cmaes_algos. Default is "acmaes", as recommended by https://github.com/CMA-ES/libcmaes/wiki/Practical-hints. For multimodal problems, you likely want to use "ipop" or "bipop".
`max_fevals`	(`integer(1)`) The maximum number of function evaluations. NA to disable. Default is 100.
`max_iter`	(`integer(1)`) The maximum number of ES iterations. NA to disable (default).
`ftarget`	(`numeric(1)`) Stop when this target function value is reached. NA to disable (default).
`f_tolerance`	(`numeric(1)`) Sets function tolerance as stopping criterion; monitors the (absolute) difference in function value over iterations and stops optimization when below tolerance. NA to disable (default).
`x_tolerance`	(`numeric(1)`) Sets parameter (absolute) tolerance as stopping criterion. This checks entries of the covariance matrix, only touch when you know what you are doing. NA to disable (default).
`lambda`	(`integer(1)`) Number of generated descendants per iteration. Must be at least 2; NA for default handling by libcmaes.
`sigma`	(`numeric(1)`) Initial sigma for covariance. NA for default handling by libcmaes.
`max_restarts`	(`integer(1)`) The maximum number of restarts, for IPOP and BIPOP. NA for default handling by libcmaes.
`elitism`	(`integer(1)`) Sets elitism:

0 no elitism

1 elitism: reinjects the best-ever seen solution

2 initial elitism: reinject x0 as long as it is not improved upon

3 initial elitism on restart: restart if best encountered solution is not the the final solution and reinjects the best solution until the population has better fitness, in its majority

NA for default handling by libcmaes.

`tpa` (`integer(1)`)
Activates / deactivates two-point adaptation step-size mechanism. 0: no, 1: auto, 2: yes. NA for default handling by libcmaes.

`tpa_dsigma` (`numeric(1)`)
Sets tpa dsigma value, use with care. NA for default handling by libcmaes.

`seed` (`integer(1)`)
The seed for the random number generator. If `NA` (default), the seed is generated randomly by R and thereby coupled to the RNG-state of R. Otherwise, the RNG of the libcmaes is different to the one in R and is hence not subject to R's seeding. Special value `0` is used for handling by libcmaes, where system time is used in libcmaes to seed.

`quiet` (`logical(1)`)
Whether to suppress libcmaes output. Internal logging of libcmaes is rerouted to Rprintf, so things like capture.output() will work. Useful for debugging. Default is `TRUE`.

`x0_lower` (`numeric`)
Optional lower bounds for randomizing the initial mean `x0`, also after restarts. Use `NULL` to disable. If this is non-`NULL`, `x0_upper` must also be set and have the same length as `x0_lower`.

`x0_upper` (`numeric`)
Optional upper bounds for randomizing the initial mean `x0`, also after restarts. Use `NULL` to disable.

Value

A cmaes_control S3 object, which is a list with the passed arguments.

Examples

```
control = cmaes_control(algo = "bipop", max_fevals = 1000, seed = 42)
print(control)
```

Index