Package 'Uno'

November 12, 2026

Type Package

Title R Interface to the 'Uno' Nonlinear Optimization Solver

Version 2.9.2-2

Date 2026-11-04

Description Bindings to 'Uno' (Unifying Nonlinear Optimization), a C++ solver for smooth nonlinearly constrained optimization. 'Uno' unifies Lagrange-Newton methods, including sequential quadratic programming and interior-point methods, by decomposing them into interacting building blocks (constraint-relaxation, inequality-handling, Hessian, and globalization strategies) that can be freely combined, either through options or through presets that reproduce established solvers such as 'filterSQP' and 'IPOPT'. The framework is described in Vanaret and Leyffer (2026) <doi:10.1007/s12532-026-00310-9>.

License MIT + file LICENSE

Copyright file inst/COPYRIGHTS

URL https://bnaras.github.io/Uno/, https://github.com/bnaras/Uno

BugReports https://github.com/bnaras/Uno/issues

SystemRequirements CMake (>= 3.16), C++17, GNU make

Imports highs (>= 1.15.1-2), rmumps (>= 5.2.1-43)

LinkingTo cpp11, highs, rmumps

Suggests cpp11, tinytest, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Encoding UTF-8

Config/roxygen2/version 8.1.1

Author Balasubramanian Narasimhan [aut, cre, cph] (R package: R/, src/uno_binding.cpp, src/dmumps_shim.c, build scripts),
Charlie Vanaret [aut, cph] (Bundled Uno C++ solver (inst/uno, MIT); designed and implemented Uno),
Alexis Montoison [ctb, cph] (Uno C API (inst/uno/interfaces/C) and

Fortran interface, MA86 interface),
Joris Gillis [ctb, cph] (Uno dynamic loader
(inst/uno/uno/tools/DynamicLoader.*)),
David Kiessling [ctb, cph] (Uno funnel globalization method
(inst/uno/uno/ingredients/glob/switching/funnel_methods)),
Manuel Schaich [ctb, cph] (Uno MA27 interface
(inst/uno/uno/ingredients/subproblem_solvers/MA27); compiled only
when HSL is found)

Maintainer Balasubramanian Narasimhan <naras@stanford.edu>

Repository CRAN

Date/Publication 2026-11-12 15:20:18 UTC

Contents

uno_mumps_version 2
uno_options 3
uno_solve 3
uno_status_labels 5
uno_version 6

Index **7**

`uno_mumps_version` *MUMPS version reached through rmumps*

Description

The version of the MUMPS linear solver that Uno's interior-point preset uses, as provided by the installed **rmumps** package, or 'NA' for a build made without MUMPS ('R_UNO_NO_MUMPS=1').

Usage

```
uno_mumps_version()
```

Value

A character scalar such as '"5.9.1"', or 'NA_character_'.

Examples

```
uno_mumps_version()
```

uno_options *Uno solver options*

Description

Every option the linked Uno accepts in the ‘options‘ argument of [uno_solve()], with its declared type (‘"integer"‘, ‘"double"‘, ‘"bool"‘ or ‘"string"‘), as read from Uno’s C API option iterator.

Usage

```
uno_options()
```

Value

A data frame with the columns ‘name‘ and ‘type‘, one row per option, sorted by name.

Examples

```
opts <- uno_options()
head(opts)
opts[opts$name == "preset", ]
```

uno_solve *Solve a nonlinear program with Uno*

Description

Solve a nonlinear program with Uno

Usage

```
uno_solve(
  n,
  lb,
  ub,
  sense,
  obj,
  grad,
  m,
  cl,
  cu,
  cons,
  jac_rows,
  jac_cols,
  jac,
  hess_rows,
```

```
  hess_cols,
  hess,
  x0,
  preset,
  base_indexing,
  verbose,
  options = list(),
  lagrangian_sign = c("negative", "positive"),
  dual0 = NULL,
  iter_callback = NULL,
  log_callback = NULL,
  name = NULL
)
```

Arguments

n	number of variables.
lb, ub	variable lower/upper bounds (length ‘n‘; use ‘-Inf‘/‘Inf‘).
sense	‘"minimize"‘ or ‘"maximize"‘.
obj, grad	objective ‘function(x)‘ and its gradient ‘function(x)‘.
m	number of constraints (0 for unconstrained).
cl, cu	constraint lower/upper bounds (length ‘m‘).
cons	constraint ‘function(x)‘ returning a length-‘m‘ vector.
jac_rows, jac_cols	COO row/column indices of the Jacobian nonzeros.
jac	Jacobian ‘function(x)‘ returning the nonzero values.
hess_rows, hess_cols	COO indices of the lower-triangular Hessian.
hess	Lagrangian Hessian ‘function(x, sigma, lambda)‘ returning the lower-triangular nonzero values, or ‘NULL‘ (Uno then uses an L-BFGS approximation, which the HiGHS subproblem solver cannot use).
x0	initial primal iterate (length ‘n‘).
preset	Uno preset, e.g. ‘"filtersqp"‘ (SQP) or ‘"ipopt"‘ (interior point, using MUMPS as the linear solver).
base_indexing	0 for C-style or 1 for Fortran-style COO indices.
verbose	if ‘FALSE‘, suppress Uno’s solution printout.
options	a named list of Uno solver options applied AFTER the preset (so they override it), e.g. ‘list(max_iterations = 200L, primal_tolerance = 1e-8, linear_solver = "MUMPS")‘; [uno_options()] lists them. Each value is coerced to the option’s declared Uno type; an unknown option name or an unacceptable value raises an error.
lagrangian_sign	the Lagrangian multiplier sign convention the ‘hess‘ callback uses: ‘"positive"‘ for $L = \sigma f + y^\top c$ (the standard convention used by IPOPT and the sparsediff

	oracle) or '"negative"' for $L = \sigma f - y^\top c$. Defaults to '"negative"', matching Uno's own C-API default. **Must match the convention your 'hess' returns**, otherwise the Lagrangian Hessian's constraint terms get the wrong sign (invisible when all constraints are linear, since their Hessian is zero).
`dual0`	optional warm-start dual iterate, or 'NULL' (Uno's default).
`iter_callback`	optional 'function(info)' called at each acceptable iterate; return 'TRUE' to terminate the solve early. 'info' is a named list with 'primals', 'lower_bound_dual', 'upper_bound_dual', 'constraint_dual', 'objective_multiplier', and the 'primal_feasibility'/'stationarity'/'complementarity' residuals. Errors in the callback are caught and treated as "do not terminate". 'NULL' disables it.
`log_callback`	optional 'function(text)' that receives Uno's output stream in chunks (a sink for the solver log); 'NULL' sends Uno's output to the R console. Independent of 'verbose' (which controls how much Uno prints).
`name`	optional model name (a string) shown in Uno's log; 'NULL' for Uno's default.

Value

a named list. The 'optimization_status' and 'solution_status' are **named integers** of the form 'c(SUCCESS = 0L)': the value is Uno's enum code and the name is its canonical label, so you can key a status map by 'names(status)' and still read the code (e.g. 'status[[1L]]'). The labels are those of Uno's C API (see [uno_status_labels()]). The list also holds the objective, primal and dual solutions ('constraint_dual', 'lower_bound_dual', 'upper_bound_dual'), KKT residuals, and per-callback evaluation counters.

Interrupting a solve

Uno installs its own 'SIGINT' handler for the duration of a solve: a Ctrl-C stops the solve cleanly at the next iteration with 'optimization_status' 'USER_INTERRUPTION', and R's handler is restored afterwards. Uno resets 'SIGINT' to the default action after the first Ctrl-C, so a second Ctrl-C before the solve returns terminates the R process. An interrupt that arrives inside a MUMPS factorization is caught by **rmumps** (version 5.2.1-44 or later) and reported the same way.

`uno_status_labels` *Uno status labels*

Description

The optimization and solution statuses of the linked Uno, as named integer vectors: the names are the labels [uno_solve()] returns in 'optimization_status' and 'solution_status', the values are Uno's enum codes. Generated from Uno's C API header at build time, so they follow the bundled Uno exactly.

Usage

```
uno_status_labels()
```

Value

A list with two named integer vectors, ‘optimization‘ and ‘solution‘.

Examples

```
uno_status_labels()
```

`uno_version` *Linked Uno version*

Description

Returns the version string of the Uno C++ library this package is built against.

Value

A character scalar, e.g. `"2.9.2"`.

Examples

```
uno_version()
```

Index

uno_mumps_version, 2
uno_options, 3
uno_solve, 3
uno_status_labels, 5
uno_version, 6