Package 'ompr'

November 18, 2017

Type Package

Title Model and Solve Mixed Integer Linear Programs

Version 0.7.0

Description Model mixed integer linear programs in an algebraic way directly in R. The model is solver-independent and thus offers the possibility to solve a model with different solvers. It currently only supports linear constraints and objective functions. See the 'ompr' website <https://dirkschumacher.github.io/ompr> for more information, documentation and examples.

License GPL-3

LazyData TRUE

RoxygenNote 6.0.1

URL https://github.com/dirkschumacher/ompr

BugReports https://github.com/dirkschumacher/ompr/issues

Depends R (>= 3.3.0)

Imports dplyr, lazyeval, progress, Rcpp (>= 0.12.12), Matrix

Suggests magrittr, testthat

ByteCompile Yes

LinkingTo Rcpp

NeedsCompilation yes

Author Dirk Schumacher [aut, cre]

Maintainer Dirk Schumacher <mail@dirk-schumacher.net>

Repository CRAN

Date/Publication 2017-11-17 23:09:28 UTC

R topics documented:

`add_constraint` *Add a constraint*

Description

Add one or more constraints to the model using quantifiers.

Usage

```
add_constraint(.model, .constraint_expr, ..., .show_progress_bar = TRUE)

add_constraint_(.model, .constraint_expr, ..., .dots,
  .show_progress_bar = TRUE)
```

Arguments

`.model`	the model
`.constraint_expr`	the constraint. Must be a linear (in)equality with operator "<=", "==" or ">=".
`...`	quantifiers for the indexed variables. For all combinations of bound variables a new constraint is created. In addition you can add filter expressions
`.show_progress_bar`	displays a progressbar when adding multiple constraints
`.dots`	Used to work around non-standard evaluation.

Value

a Model with new constraints added

Examples

```
library(magrittr)
MIPModel() %>%
 add_variable(x[i], i = 1:5) %>%
 add_constraint(x[i] >= 1, i = 1:5) # creates 5 constraints
```

`add_variable` *Add a variable to the model*

Description

A variable can either be a name or an indexed name. See examples.

Usage

```
add_variable(.model, .variable, ..., type = "continuous", lb = -Inf,
  ub = Inf)

add_variable_(.model, .variable, ..., type = "continuous", lb = -Inf,
  ub = Inf, .dots)
```

Arguments

`.model`	the model
`.variable`	the variable name/definition
`...`	quantifiers for the indexed variable. Including filters
`type`	must be either continuous, integer or binary
`lb`	the lower bound of the variable
`ub`	the upper bound of the variable
`.dots`	Used to work around non-standard evaluation.

Examples

```
library(magrittr)
MIPModel() %>%
 add_variable(x) %>% # creates 1 variable named x
 add_variable(y[i], i = 1:10, i %% 2 == 0,
              type = "binary") # creates 4 variables
```

extract_constraints *Extract the constraint matrix, the right hand side and the sense from a model*

Description

Extract the constraint matrix, the right hand side and the sense from a model

Usage

```
extract_constraints(model)
```

Arguments

model	the model

Value

a list with three named elements. 'matrix' the (sparse) constraint matrix from the Matrix package. 'rhs' is the right hand side vector in the order of the matrix. 'sense' is a vector of the constraint senses

Examples

```
library(magrittr)
model <- MIPModel() %>%
  add_variable(x[i], i = 1:3) %>%
  add_variable(y[i], i = 1:3) %>%
  add_constraint(x[i] + y[i] <= 1, i = 1:3)
extract_constraints(model)
```

get_solution *Get variable values from a solution*

Description

Get variable values from a solution

Usage

```
get_solution(solution, expr)
get_solution_(solution, expr)

get_solution_(solution, expr)
```

Arguments

solution	the solution object
expr	a variable expression. You can partially bind indexes.

Value

a data.frame. One row for each variable instance and a column for each index. Unless it is a single variable, then it returns a single number.

Examples

```
## Not run:
library(magrittr)
result <- MIPModel() %>%
     add_variable(x[i], i = 1:5) %>%
     add_variable(y[i, j], i = 1:5, j = 1:5) %>%
     add_constraint(x[i] >= 1, i = 1:5) %>%
     set_bounds(x[i], lb = 3, i = 1:3) %>%
     set_objective(0) %>%
     solve_model(with_ROI("glpk"))
solution <- get_solution(result, x[i])
solution2 <- get_solution(result, y[i, 1])
solution3 <- get_solution(result, y[i, j])

## End(Not run)
```

MIPModel *Create a new MIP Model*

Description

Create a new MIP Model

Usage

```
MIPModel()
```

new_solution *Create a new solution*

Description

This function/class should only be used if you develop your own solver.

Usage

```
new_solution(model, objective_value, status, solution)
```

Arguments

model	the optimization model that was solved
objective_value	a numeric objective value
status	the status of the solution
solution	a named numeric vector containing the solution values

nvars *Number of variables of a model*

Description

Number of variables of a model

Usage

```
nvars(model)
```

Arguments

model	the model

Value

a list with three named elements. 'binary' => number of binary variables, 'integer' => number of integer variables, 'continuous' => number of continuous variables.

Examples

```
library(magrittr)
model <- MIPModel() %>%
  add_variable(x[i], i = 1:10, type = "binary") %>%
  add_variable(y[i], i = 1:5, type = "continuous") %>%
  add_variable(z[i], i = 1:2, type = "integer")
nvars(model)
```

objective_function *Extract the objective function from a model*

Description

Extract the objective function from a model

Usage

```
objective_function(model)
```

Arguments

model	the model

Value

a list with two named elements, 'solution' and 'constant'. 'solution' is a sparse vector from the Matrix package. 'constant' is a constant that needs to be added to get the final obj. value.

Examples

```
library(magrittr)
model <- MIPModel() %>%
  add_variable(x[i], i = 1:5) %>%
  set_objective(sum_expr(i * x[i], i = 1:5) + 10)
objective_function(model)
```

objective_value *Extract the numerical objective value from a solution*

Description

Extract the numerical objective value from a solution

Usage

```
objective_value(solution)
```

Arguments

solution	a solution

Value

numeric single item vector

ompr *A package to Model (Mixed) Integer Programs*

Description

A package to model (mixed) integer programs. It provides an algebraic way to model mixed integer linear optimization problems directly in R. The model is solver-independent and thus offers the possibility to solve a model with different solvers. See the ompr website <https://dirkschumacher.github.io/ompr/> for more information, documentation and examples.

set_bounds *Set the bounds of a variable*

Description

Change the lower and upper bounds of a named variable, indexed variable or a group of variables.

Usage

```
set_bounds(.model, .variable, ..., lb = NULL, ub = NULL)

set_bounds_(.model, .variable, ..., lb = NULL, ub = NULL, .dots)
```

Arguments

.model	the model
.variable	the variable name/definition
...	quantifiers for the indexed variable
lb	the lower bound of the variable
ub	the upper bound of the variable
.dots	Used to work around non-standard evaluation.

Examples

```
library(magrittr)
MIPModel() %>%
 add_variable(x[i], i = 1:5) %>%
 add_constraint(x[i] >= 1, i = 1:5) %>% # creates 5 constraints
 set_bounds(x[i], lb = 3, i = 1:3)
```

set_objective *Set the model objective*

Description

Set the model objective

Usage

```
set_objective(model, expression, sense = c("max", "min"))

set_objective_(model, expression, sense = c("max", "min"))
```

Arguments

model	the model
expression	the linear objective as a sum of variables and constants
sense	the model sense. Must be either "max" or "min".

Value

a Model with a new objective function definition

Examples

```
library(magrittr)
MIPModel() %>%
 add_variable(x, lb = 2) %>%
 add_variable(y, lb = 40) %>%
 set_objective(x + y, sense = "min")
```

solver_status *Get the solver status from a solution*

Description

Get the solver status from a solution

Usage

```
solver_status(solution)
```

Arguments

solution	a solution

Value

character vector being either "infeasible", "optimal", "unbounded", "userlimit" or "error

---

`solve_model` *Solve a model*

Description

Solve a model

Usage

```
solve_model(model, solver)
```

Arguments

`model`	the model
`solver`	a function mapping a model to a solution

Value

solver(model)

`sum_expr` *Construct a sum expression*

Description

This functions helps to create dynamic sum expression based on external variables. Should only be used within other 'ompr' functions.

Usage

```
sum_expr(expr, ...)
```

Arguments

`expr`	an expression that can be expanded to a sum
`...`	bind variables in expr using dots. See examples.

Value

the expanded sum as an AST

See Also

add_constraint

set_objective

Examples

```
# create a sum from x_1 to x_10
sum_expr(x[i], i = 1:10)
# create a sum from x_2 to x_10 with even indexes
sum_expr(x[i], i = 1:10, i %% 2 == 0)
```

---

variable_bounds *Variable lower and upper bounds of a model*

Description

Variable lower and upper bounds of a model

Usage

```
variable_bounds(model)
```

Arguments

model	the model

Value

a list with two components 'lower' and 'upper' each having a numeric vector of bounds. One for each variable.

Examples

```
library(magrittr)
model <- MIPModel() %>%
  add_variable(x, type = "binary") %>%
  add_variable(y, type = "continuous", lb = 2) %>%
  add_variable(z, type = "integer", ub = 3)
variable_bounds(model)
```

variable_keys *Get all unique names of the model variables*

Description

Get all unique names of the model variables

Usage

```
variable_keys(model)
```

Arguments

model	the model

Value

a character vector ordered in the same way as the constraint matrix columns and objective vector

Examples

```
library(magrittr)
model <- MIPModel() %>%
  add_variable(x[i], i = 1:3)
variable_keys(model)
```

variable_types *Variable types of a model*

Description

One component for each variable in the correct order

Usage

```
variable_types(model)
```

Arguments

model	the model

Value

a factor with levels binary, continuous, integer

Examples

```
library(magrittr)
model <- MIPModel() %>%
  add_variable(x, type = "binary") %>%
  add_variable(y, type = "continuous") %>%
  add_variable(z, type = "integer")
variable_types(model)
```

Index