The **ROI** Package in Action

Portfolio Optimization and Beyond

Stefan Theußl

Rmetrics Workshop, June 26–28, 2014

Optimization Problem

Following Boyd and Vandenberghe (2004), a mathematical optimization problem, or just *optimization problem*, has the form

$$\min f(x)$$

$$s.t.$$

$$g_i(x) \leq b_i; i = 1, \ldots, m.$$

where

- $x = (x_1, \ldots, x_n)$ is the *optimization variable* of the problem,
- the function $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*,
- the functions $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$, are the (inequality) constraint functions, and the constants $b_1, \ldots, b_m$ are the limits, or bounds, for the constraints.

Solution

A vector x^* is called *optimal*, or a *solution* of the above problem, if it has

- the smallest *objective value* among all vectors that satisfy the constraints
- for any z with $g_1(z) \leq b_1, \ldots, g_m(z) \leq b_m$, we have $f(z) \geq f(x^*)$.

We generally consider families or *classes of optimization problems*, characterized by particular forms of the objective and constraint functions.

Motivation

Mean-Variance Portfolio Optimization (Markowitz, 1952)

- Minimum Risk

$$\begin{aligned} \min_{w} \quad & w^\top \hat{\Sigma}\, w \\ s.t. \quad & \\ & Aw^\top \leq b \end{aligned}$$

- Maximum Return

$$\begin{aligned} \max_{w} \quad & w^\top \hat{\mu} \\ s.t. \quad & \\ & Aw \leq b \\ & w^\top \hat{\Sigma} w \leq \sigma \end{aligned}$$

Problem Classes

Given N objective variables, $x_i, i = 1, \ldots, N$, to be optimized we can differentiate between

- Linear Programming (LP, $\min_x c^\top x$ s.t. $Ax = b$, $x \geq 0$)
- Quadratic Programming (QP, $\min_x x^\top Qx$ s.t. $Ax = b$, $x \geq 0$)
- Nonlinear Programming (NLP, $\min_x f(x)$ s.t. $x \in S$)

If variables have to be of *type* integer, formally $x_j \in \mathbb{N}$ for $j = 1, \ldots, p$, $1 \leq p \leq N$:

- Mixed Integer Linear Programming (MILP),
- Mixed Integer Quadratic Programming (MIQP),
- NonLinear Mixed INteger Programming (NLMINP)

Solvers in R

Subset of available solvers categorized by the capability to solve a given problem class:

	LP	QP	NLP
LC	Rglpk*, lpSolve*, Rsymphony*	quadprog, ipop	optim(), nlminb()
QC		Rcplex*	
NLC			donlp2, solnp

* … integer capability

For a full list of solvers see the CRAN task view *Optimization* (http://CRAN.R-Project.org/view=Optimization).

Solving Optimization Problems (1)

- **lpSolve**:

```
> args(lp)
function (direction = "min", objective.in, const.mat, const.dir,
   const.rhs, transpose.constraints = TRUE, int.vec, presolve = 0,
   compute.sens = 0, binary.vec, all.int = FALSE, all.bin = FALSE,
   scale = 196, dense.const, num.bin.solns = 1, use.rw = FALSE)
NULL
```

- **quadprog**:

```
> args(solve.QP)
function (Dmat, dvec, Amat, bvec, meq = 0, factorized = FALSE)
NULL
```

- **Rglpk**:

```
> args(Rglpk_solve_LP)
function (obj, mat, dir, rhs, types = NULL, max = FALSE, bounds = NULL,
   verbose = FALSE)
NULL
```

Solving Optimization Problems (2)

- **Rcplex**:

```
> args(Rcplex)
function (cvec, Amat, bvec, Qmat = NULL, lb = 0, ub = Inf, control = list(),
    objsense = c("min", "max"), sense = "L", vtype = NULL, n = 1)
NULL
```

- optim() from **stats**:

```
> args(optim)
function (par, fn, gr = NULL, ..., method = c("Nelder-Mead",
    "BFGS", "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf,
    control = list(), hessian = FALSE)
NULL
```

- nlminb() from **stats**:

```
> args(nlminb)
function (start, objective, gradient = NULL, hessian = NULL,
    ..., scale = 1, control = list(), lower = -Inf, upper = Inf)
NULL
```

ROI Modeling (1)

ROI optimization problems are R objects (S3) specified by

- a function $f(x)$ to be optimized: *objective*
  - linear: coefficients c expressed as a 'numeric' (a vector)
  - quadratic: a 'matrix' Q of coefficients representing the quadratic form as well as a linear part L
  - nonlinear: an adequate (R) 'function'
- *constraints* $g(x)$ describing the feasible set S
  - linear: coefficients expressed as a 'numeric' (a vector), or several constraints as a (sparse) 'matrix'
  - quadratic: a quadratic part Q and a linear part L
  - nonlinear: a well-defined (R) 'function'
  - equality ("==") or inequality ("<=", ">=", ">", etc.) constraints
  - bounds (the right hand side).

ROI Modeling (2)

Additional properties:

- variable *bounds* (or so-called box constraints)
- variable *types* (continuous, integer, binary, etc.)
- direction of optimization (search for minimum, *maximum*)

The problem constructor in **ROI** is named OP() and takes the following arguments:

```
function( objective, constraints = NULL, types = NULL,
      bounds = NULL, maximum = FALSE )
```

ROI signatures

Pre-defined optimization problem signatures:

- ROI_make_<class>_signatures()
- where <class> is one of LP, QP, MILP, MIQP, MIQCP.

```
> ROI:::ROI_make_LP_signatures()
  objective constraints bounds maximum    C     I     B
1         L           L   TRUE    TRUE TRUE FALSE FALSE
2         L           L  FALSE    TRUE TRUE FALSE FALSE
3         L           L   TRUE   FALSE TRUE FALSE FALSE
4         L           L  FALSE   FALSE TRUE FALSE FALSE
```

```
> dim( ROI:::ROI_make_MIQCP_signatures() )
[1] 112   7
```

```
> #ROI:::OP_signature()
```

Examples: ROI and Constraints

```
> require( "ROI" )
> (constr1 <- L_constraint(c(1, 2), "<", 4))
An object containing 1 linear constraints.

> (constr2 <- L_constraint(matrix(c(1:4), ncol = 2), c("<", "<"), c(4, 5)))
An object containing 2 linear constraints.

> rbind(constr1, constr2)
An object containing 3 linear constraints.

> (constr3 <- Q_constraint(matrix(rep(2, 4), ncol = 2), c(1, 2), "<", 5))
An object containing 1 constraints.
Some constraints are of type quadratic.

> foo <- function(x) {
+     sum(x^3) - seq_along(x) %*% x
+ }
> F_constraint(foo, "<", 5)
An object containing 1 constraints.
Some constraints are of type nonlinear.
```

Examples: Optimization Instances

```
> lp <- OP(objective = c(2, 4, 3), L_constraint(L = matrix(c(3,
+     2, 1, 4, 1, 3, 2, 2, 2), nrow = 3), dir = c("<=", "<=", "<="),
+     rhs = c(60, 40, 80)), maximum = TRUE)
> lp
ROI Optimization Problem:

Maximize a linear objective function with
- 3 objective variables,

subject to
- 3 constraints of type linear.
```

```
> qp <- OP(Q_objective(Q = diag(1, 3), L = c(0, -5, 0)), L_constraint(L = matrix(c(-4,
+     -3, 0, 2, 1, 0, 0, -2, 1), ncol = 3, byrow = TRUE), dir = rep(">=",
+     3), rhs = c(-8, 2, 0)))
> qp
ROI Optimization Problem:

Minimize a quadratic objective function with
- 3 objective variables,

subject to
```

ROI Solver Interface

A given problem is solved via

- ROI_solve(problem, solver, control, ...)

where

problem
: represents an object containing the description of the corresponding optimization problem

solver
: specifies the solver to be used ("glpk", "quadprog", "symphony", etc.)

control
: is a list containing additional control arguments to the corresponding solver

...
: is a wildcard for additional control arguments

See https://R-Forge.R-project.org/projects/roi/.

ROI Plug-ins (1)

- **ROI** is very easy to extend via “plug-ins” (**ROI.plugin.**$<$**solver**$>$ packages)
- Link between “API packages” and **ROI**
- Capabilities registered in data base
- Solution canonicalization
- Status code canonicalization

ROI Plug-ins (2)

A plug-in is an R package with the following requirements

- A DESCRIPTION file which *imports* **ROI** and typically the API package (e.g., **Rglpk** for using the GLPK solver)
- A NAMESPACE file containing the following entries:

```
import("ROI")
import( _API_package_ )
importFrom("methods", "getPackageName")
importFrom("methods", "getFunction")
```

ROI Plug-ins (3)

- An onLoad() function registering the solver method for a given signature:

```
> .onLoad <- function( libname, pkgname ) {
+     ## Solver plugin name (based on package name)
+     if( ! pkgname %in% ROI_registered_solvers() ){
+         ## Register solver methods here.
+         ## One can assign several signatures a single solver method
+         solver <- ROI:::get_solver_name( pkgname )
+         ROI:::ROI_register_solver_method( signatures = ROI:::ROI_make_MILP_signatures(),
+                                           solver = solver,
+                                           method =
+                                           getFunction( "solve_OP", where = getNamespace(pkgname)) )
+         ## Finally, for status code canonicalization add status codes to data base
+         .add_status_codes()
+     }
+ }
```

- A Function (usually solve_OP()) which extracts from the optimization object necessary information and calls the solver

ROI Plug-ins (4)

The version which is published on CRAN can handle LP up to MILP and MIQCP problems using the following supported solvers:

- **lpSolve** (soon)
- **ipop** (R-Forge, not compatible with recent **ROI**)
- **quadprog**
- **Rcplex** (R-Forge, on CRAN soon)
- **Rglpk** (default)
- **Rsymphony**

Additional requirements to run **ROI**:

- **slam** for storing coefficients (constraints, objective) as sparse matrices
- **registry** providing a pure R data base system

Examples: Solving LPs

```
> (sol <- ROI_solve(lp, solver = "glpk"))
Optimal solution found.
The objective value is: 7.666667e+01
```

```
> unclass( sol )
$solution
[1]  0.000000  6.666667 16.666667

$objval
[1] 76.66667

$status
$status$code
[1] 0

$status$msg
  solver glpk
    code 5
  symbol GLP_OPT
 message Solution is optimal.
roi_code 0
```

Examples: Solving QCPs

```
> ROI_solve(qcp, solver = "cplex")
$solution
[1] 0.1291236 0.5499528 0.8251539

$objval
         [,1]
[1,] 2.002347

$status
$status$code
[1] 0

$status$msg
  solver cplex
    code 1
  symbol CPX_STAT_OPTIMAL
 message (Simplex or barrier): optimal solution.
roi_code 0


attr(,"class")
```

Examples: Computations on Objects

```
> obj <- objective(qcp)
> obj
function (x)
crossprod(L, x) + 0.5 * .xtQx(Q, x)
<environment: 0x29f34c8>
attr(,"class")
[1] "function"    "Q_objective" "objective"

> constr <- constraints(qcp)
> length(constr)
[1] 3

> x <- ROI_solve(qcp, solver = "cplex")$solution
> obj(x)
         [,1]
[1,] 2.002347
```

Portfolio Optimization

See accompanying R code.

Thank you for your attention

Stefan Theußl
email: Stefan.Theussl@R-Project.org