WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

ROI — the R Optimization Infrastructure Package

Stefan Theußl Kurt Hornik David Meyer

R/Finance 2012, May 12

Motivation (1)

Mean-Variance Portfolio Optimization (Markowitz, 1952)

- Minimum Risk

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

- Maximum Return

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

Least absolute deviations (LAD) or L_1 regression problem

$$\min \sum_{i}^{n} |y_i - \hat{y}_i|$$

can be expressed as (see Brooks and Dula, 2009)

$$\min_{\beta_0, \beta, \mathbf{e}^+, \mathbf{e}^-} \sum_{i=1}^{n} e_i^+ + e_i^-$$

s.t.

$$\begin{aligned} \beta_0 + \beta^\top \mathbf{x}_i + e_i^+ - e_i^- &= 0 \qquad i = 1, \dots, n \\ \beta_j &= -1 \\ e_i^+, e_i^- &\geq 0 \qquad i = 1, \dots, n \end{aligned}$$

given a point set $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \dots, n$ and the j^{th} column representing the dependent variable.

Problem Classes

Several different *problem classes* (in Mathematical Programming, MP) have been identified. Given N objective variables, $x_i, i = 1, \dots, N$, 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. $x \in S$)

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

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)

A general framework for optimization should be capable of handling several different problem classes in a transparent and uniform way. We define optimization problems as R objects (S3). These objects contain:

- 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 arbitrary (R) '`function`'
- one or several **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: an arbitrary (R) '`function`'
  - equality (`"=="`) or inequality (`"<="`, `">="`, `">"`, etc.) constraints

ROI Modeling (2)

Additionally we have:

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

Thus, a problem constructor (say for a MIQP) usually takes the following arguments:

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

In ROI this constructor is named `OP()`.

Examples: ROI and Constraints

```
> library("ROI")
ROI: R Optimization Infrastructure
Installed solver plugins: cplex, lpsolve, glpk, quadprog, symphony, nlminb.
Default solver: glpk.
> (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
A linear programming problem with 3 constraints of type linear.

> qp <- OP(Q_objective(Q = diag(1, 3), L = c(0, -5, 0)), L_constraint(L = matrix(c(
+     -3, 0, 2, 1, 0, 0, -2, 1), ncol = 3, byrow = TRUE), dir = rep(">=",
+     3), rhs = c(-8, 2, 0)))
> qp
A quadratic programming problem with 3 constraints of type linear.

> qcp <- OP(Q_objective(Q = matrix(c(-33, 6, 0, 6, -22, 11.5,
+     0, 11.5, -11), byrow = TRUE, ncol = 3), L = c(1, 2, 3)),
+     Q_constraint(Q = list(NULL, NULL, diag(1, nrow = 3)), L = matrix(c(-1,
+         1, 1, 1, -3, 1, 0, 0, 0), byrow = TRUE, ncol = 3), dir = rep("<=",
+         3), rhs = c(20, 30, 1)), maximum = TRUE)
> qcp
A quadratic programming problem with 3 constraints of type quadratic.
```

ROI Solver Interface

The R Optimization Infrastructure (ROI) package promotes the development and use of interoperable (open source) optimization problem solvers for R.

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

The main function takes 3 arguments:

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

... replacement for additional control arguments

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

ROI Plugins (1)

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

ROI Plugins (2)

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)
- quadprog
- Rcplex (R-Forge)
- 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

```
> ROI_solve(lp, solver = "glpk")
$solution
[1]  0.000000  6.666667 16.666667

$objval
[1] 76.66667

$status
$status$code
[1] 0

$status$msg
  solver glpk
    code 0
  symbol GLP_OPT
 message (DEPRECATED) Solution is optimal. Compatibility status code
         will be removed in Rglpk soon.
roi_code 0


attr(,"class")
[1] "MIP_solution"
```

Examples: Solving LPs

```
> 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")
[1] "MIP_solution"
```

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 (1)

Example[1]:

```
> library("fPortfolio")
> data(LPP2005.RET)
> lppData <- 100 * LPP2005.RET[, 1:6]
> r <- mean(lppData)
> r
[1] 0.04307677

> foo <- Q_objective(Q = cov(lppData), L = rep(0, ncol(lppData)))
> full_invest <- L_constraint(rep(1, ncol(lppData)), "==", 1)
> target_return <- L_constraint(apply(lppData, 2, mean), "==",
+     r)
> op <- OP(objective = foo, constraints = rbind(full_invest, target_return))
> op
A quadratic programming problem with 2 constraints of type linear.
```

[1] Portfolio Optimization with R/Rmetrics by Würtz et al (2009)

Solve the portfolio optimization problem via `ROI_solve()`

```
> sol <- ROI_solve(op, solver = "cplex")
> w <- sol$solution
> round(w, 4)

[1] 0.0000 0.0086 0.2543 0.3358 0.0000 0.4013

> sqrt(t(w) %*% cov(lppData) %*% w)

          [,1]
[1,] 0.2450869

> sol <- ROI_solve(op, solver = "quadprog")
> w <- sol$solution
> round(w, 4)

[1] 0.0000 0.0086 0.2543 0.3358 0.0000 0.4013

> sqrt(t(w) %*% cov(lppData) %*% w)

          [,1]
[1,] 0.2450869
```

Portfolio Optimization (3)

Solve the max-return portfolio optimization problem:

```
> sigma <- sqrt(t(w) %*% cov(lppData) %*% w)
> zero_mat <- simple_triplet_zero_matrix(ncol(lppData))
> foo <- Q_objective(Q = zero_mat, L = colMeans(lppData))
> maxret_constr <- Q_constraint(Q = list(cov(lppData), NULL), L = rbind(rep(0,
+     ncol(lppData)), rep(1, ncol(lppData))), c("<=", "<="), c(sigma^2,
+     1))
> op <- OP(objective = foo, constraints = maxret_constr, maximum = TRUE)
> op

A quadratic programming problem with 2 constraints of type quadratic.

> sol <- ROI_solve(op, solver = "cplex")
> w <- sol$solution
> round(w, 4)

[1] 0.0000 0.0086 0.2543 0.3358 0.0000 0.4013

> w %*% colMeans(lppData)

           [,1]
[1,] 0.04307677
```

L1 Regression (1)

```
> library("quantreg")
> data(stackloss)
> create_L1_problem <- function(x, j) {
+     len <- 1 + ncol(x) + 2 * nrow(x)
+     beta <- rep(0, len)
+     beta[j + 1] <- 1
+     OP(L_objective(c(rep(0, ncol(x) + 1), rep(1, 2 * nrow(x)))),
+         rbind(L_constraint(cbind(1, as.matrix(x), diag(nrow(x)),
+             -diag(nrow(x))), rep("==", nrow(x)), rep(0, nrow(x))),
+             L_constraint(beta, "==", -1)), bounds = V_bound(li = seq_len(ncol(x) +
+             1), ui = seq_len(ncol(x) + 1), lb = rep(-Inf, ncol(x) +
+             1), ub = rep(Inf, ncol(x) + 1), nobj = len))
+ }
```

L1 Regression (2)

```
> ROI_solve(create_L1_problem(stackloss, 4), solver = "glpk")$solution
 [1] -39.68985507   0.83188406   0.57391304  -0.06086957  -1.00000000
 [6]   5.06086957   0.00000000   5.42898551   7.63478261   0.00000000
[11]   0.00000000   0.00000000   0.00000000   0.00000000   0.00000000
[16]   0.52753623   0.04057971   0.00000000   0.00000000   1.18260870
[21]   0.00000000   0.00000000   0.00000000   0.48695652   1.61739130
[26]   0.00000000   0.00000000   0.00000000   0.00000000   0.00000000
[31]   1.21739130   1.79130435   1.00000000   0.00000000   1.46376812
[36]   0.02028986   0.00000000   0.00000000   2.89855072   1.80289855
[41]   0.00000000   0.00000000   0.42608696   0.00000000   0.00000000
[46]   0.00000000   9.48115942
> rq(stack.loss ~ stack.x, 0.5)
Call:
rq(formula = stack.loss ~ stack.x, tau = 0.5)

Coefficients:
      (Intercept)   stack.xAir.Flow stack.xWater.Temp stack.xAcid.Conc.
     -39.68985507        0.83188406        0.57391304       -0.06086957

Degrees of freedom: 21 total; 17 residual
```

Outlook and Future Work

- Optimization terminology (What is a solution?)
- Status codes (What is a reasonable set of status codes?)
- File reader for standard formats like MPS.
- Parallel computing and optimizers (e.g., SYMPHONY's or CPLEX' parallel solver)
- NLP solvers (`optim()`, `nlminb()`, Rsolnp, etc.)
- Interface to NLMINP solver Bonmin?
- AMPL?
- Applications (e.g., fPortfolio, relations, etc.)
- Compare the performance of MILP solvers in R via standard test sets like MIPLIB2003 (`http://miplib.zib.de/miplib2003.php`)

Thank you for your attention

Stefan Theußl
Department of Finance, Accounting and Statistics
Institute for Statistics and Mathematics
email: `Stefan.Theussl@wu.ac.at`
URL: `http://statmath.wu.ac.at/~theussl`

WU Wirtschaftsuniversität Wien
Augasse 2–6, A-1090 Wien