**NEOS Optimization Solver in R code**

NEOS can solve optimization problems with hundreds or thousands of control variables and constraints with so many time-tested algorithms. For more information, you can visit [NEOS (https://neos-server.org/neos/)](https://neos-server.org/neos/).   
  
To use NEOS service in R, ROI package and its plug in packages are needed. As we use **ROI** and **ROI.plugin.neos** package, install these packages in R studio.

**Point To Note**

ROI provides too many functionalities to be covered in this post. It, also, has many plug-in packages as you can see the plug-in list from the above figure, Therefore this post deal with small part of it.

As our focus centers mainly on portfolio optimization or asset allocation. we consider a quadratic optimization problem.

**Quadratic Optimization**

The mean-variance portfolio optimization problem has the following form.   
  
\[\begin{align} & min\_{\boldsymbol{w}} ⁡\frac{1}{2} \boldsymbol{w}^{‘}\Sigma \boldsymbol{w} \\ s.t. & \\ & \boldsymbol{w}^{‘} \boldsymbol{1} = 1 \\ & \boldsymbol{w}^{‘} \boldsymbol{\mu} = E(R) \\ & 0 \le \boldsymbol{w} \le 1 \end{align}\] \(\boldsymbol{w}\) : weight vector   
\(\Sigma\) : covariance matrix   
\(\mu\) : mean return vector   
\(E(R)\) : target expected return

**ROI and ROI.plugin.neos**

ROI package has the following format.   
  
**1) Objective Function**

|  |  |  |
| --- | --- | --- |
| 1  2  3 | # objective function to minimize      obj <– Q\_objective(Q = 2\*cov)    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

Objective function is set by using the function **Q\_objective()** with arguments of **Q** or **L** or **both**. **Q** means the **quadratic coefficient matrix** and **L** means the **linear coefficient vector**.   
  
**2) Constraints**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | # 2 linear constraints      const <– L\_constraint(          L   = rbind(rep(1, nvar), mu),          dir = c(“==”,“<="),          rhs = c(1, rset[10]))    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

Since portfolio optimization use the linear constraints, the function **L\_constraint()** which has arguments of **L**, **dir**, and **rhs** is used. **L** means a **left hand side coefficient matrix**. **dir** means a vector of **equalities or inequalities or both**. **rhs** means a vector of **right hand side values**. If quadratic constraints are used, function **Q\_constraint()** is used.   
  
**3) Upper and Lower Bounds**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | # lower and upper bounds      lb\_ub <– V\_bound(lb = rep(0, nvar),                       ub = rep(1, nvar))    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

Bounds or ranges for control variables are set by using the function **V\_bound()** which has two arguments, **lb** and **ub**. This usage is straightforward. **lb** and **ub** mean vectors of lower and upper bounds for control variables respectively.   
  
**4) Optimization Problem**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | # create optimization problem      op <– OP( objective   = obj,                constraints = const,                bounds      = lb\_ub)    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

Final optimization problem is set by using the function **OP()**. This function take the above three arguments as input arguments.   
  
**5) Solving Problems using NEOS Solver**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | res\_scip  <– ROI\_solve(op, solver = “neos”, method = “scip”,                             email=“your email address”)      res\_mosek <– ROI\_solve(op, solver = “neos”, method = “mosek”,                             email=“your email address”)      res\_cplex <– ROI\_solve(op, solver = “neos”, method = “cplex “,                             email=“your email address”) | [cs](http://colorscripter.com/info#e) |

Using function **ROI\_solve()** with **solver = “neos”**, we can use NEOS solver service. Of course, arguments such as solver and method have several options. In particular, method can take several optimizers. In this case, we use three alternative optimizers: **scip**, **mosek**, **cplex**. These are time-tested optimizers for the large-scale optimization problems. Here, **solver = “neos”** requires the package **ROI.plugin.neos**.   
  
From recently, **user email address** is required so that you sign up and log in the NEOS website and **your email should be verified**. Every time you run **ROI\_solve()** with **solver = “neos”**, You receive email which contains the full results from NEOS

**R code using NEOS solver through ROI**

The following R code is implemented based on the above format. Of course, traditional optimization using function **solveQPXT()** is the same as the previous post regarding Michaud portfolio optimization.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117 | #=================================================================#  #—————————————————————–#  # Portfolio Optimization using ROI NEOS  #=================================================================#    graphics.off()  # clear all graphs  rm(list = ls()) # remove all files from your workspace    library(quadprogXT)      # solveQPXT  library(ROI)             # ROI\_solve  library(ROI.plugin.neos) # NEOS    #—————————————————–  # Michuad dataset  #—————————————————–        setwd(“D:/SHLEE/blog/rneos/michuad”)        # dataset in Michaud and Michaud (2007) appendix      df.data <– read.csv(‘mu\_sd\_corr\_michaud.csv’)      head(df.data)        # mean, std, correlation      mu   <– as.vector(df.data[,1])      sd   <– as.vector(df.data[,2])      corr <– as.matrix(df.data[,–c(1,2)])        # number of asset      nvar <– length(mu)  [var.name](http://var.name) <– colnames(df.data[,–c(1,2)])        # convert correlation to covariance matrix      cov <– diag(sd)%\*%corr%\*%diag(sd)    #—————————————————–  # Portfolio optimization using solveQPXT  #—————————————————–    [n.er](http://n.er) <– 100  # number of EF points      rset <– seq(min(mu),max(mu),length=[n.er](http://n.er)+2)      rset <– rset[2:[n.er](http://n.er)+1]        # given returns and unknown std and weight      port1.ret <– rset      port1.std <– rset\*0      port1.wgt <– matrix(0,[n.er](http://n.er),nvar)        # ith portfolio problem setting      i = 10;      Dmat <– 2\*cov      dvec <– rep(0,nvar) #c(0,0)      Amat <– t(rbind(t(rep(1,nvar)),t(mu),diag(nvar)))      bvec <– c(1,rset[i],rep(0,nvar))        # mean-variance optimization      m<–solveQPXT(Dmat,dvec,Amat,bvec,meq=2,factorized=FALSE)        # output      port1.std[i]  <– sqrt(m$value)      port1.wgt[i,] <– t(m$solution)    #—————————————————–  # Portfolio optimization using ROI.neos.plugin  #—————————————————–        #——————–      # min w’\*cov\*w      # s.t.      #    sum(w) = 1      #    w\*r   <= mu      #    0 <= w <= 1      #——————–        # objective function to minimize      obj <– Q\_objective(Q = 2\*cov)        # 2 linear constraints      const <– L\_constraint(          L   = rbind(rep(1, nvar), mu),          dir = c(“==”,“<="),          rhs = c(1, rset[10]))        # lower and upper bounds      lb\_ub <– V\_bound(lb = rep(0, nvar),                       ub = rep(1, nvar))        # create optimization problem      op <– OP( objective   = obj,                constraints = const,                bounds      = lb\_ub)        res\_scip  <– ROI\_solve(op, solver = “neos”, method = “scip”,                             email=“your email address”)      res\_mosek <– ROI\_solve(op, solver = “neos”, method = “mosek”,                             email=“your email address”)      res\_cplex <– ROI\_solve(op, solver = “neos”, method = “cplex “,                             email=“your email address”)  #—————————————————–  # Comparisons  #—————————————————–        # function value      cat(paste(“\nobj val : solveQPXT   =”, m$value, “\n”,                ”         NEOS(scip)  =”, res\_scip$objval, “\n”,                ”         NEOS(mosek) =”, res\_mosek$objval, “\n”,                ”         NEOS(cplex) =”, res\_cplex$objval, “\n”))        # weight comparisons : solveQPXT versus NEOS(mosek)      output\_mosek <– cbind(res\_mosek$solution,  port1.wgt[i,],                            res\_mosek$solution – port1.wgt[i,])      colnames(output\_mosek) <– c(“NEOS(mosek)”, “solveQPXT”, “diff”)      print(output\_mosek)    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

Seeing the results below, we can find that although MOSEK optimizer attains the most minimized function value, the differences between optimizers are negligible. Therefore the optimal weight differences between solveQPXT and MOSEK is also so small and insignificant.

Too small weight differences can also be found the following bar plot.

**Final Thoughts**

From this post, we can learn how to use NEOS optimizer using ROI and ROI.plugin.neos packages.

It is worth noting that NEOS is not for this small problem but for so big problem which has so many variables and constraints. Therefore, when you have to solve this big problem, it is suitable to use NEOS solver using R and its packages.   
  
Before anything else, this NEOS solver can be used to the multi-stage stochastic linear programming because it can solve the problems written by GAMS or AMPL format using MOSEK or CPLEX. \(\blacksquare\)