Interfacing NEOS from R
The R package rneos

Bernhard Pfaff
bernhard_pfaff@fra.invesco.com

Invesco Asset Management Deutschland GmbH, Frankfurt am Main

5th R/Rmetrics Meielisalp Workshop
June 26–30, 2011
Overview

- Network-Enabled Optimisation System (NEOS)
  - Overview
  - Available Solvers
  - Interfaces
- The R package rneos
  - Overview
  - Implementation of the API
  - Example
- Outlook
http://www.neos-server.org

Server framework for solving optimisation problems.

Why using NEOS?
- Optimisation software does not need to be installed locally.
- Computational burdensome problems are transferred to remote machines.

Help: via Email, FAQ, User Guide (Wiki).
NEOS

Optimisation problems (in alphabetical order)

- Bound Constrained Optimisation
- Combinatorial Optimisation and Integer Programming
- Complementarity Problems
- Global Optimisation
- Linear Network Programming
- Linear Programming
- Mixed Integer Linear Programming
- Mixed Integer Nonlinearly Constrained Optimisation
- Nonlinearly Constrained Optimisation
- Non-differentiable Optimisation
- Semidefinite Programming
- Semi-infinite Optimisation
- Stochastic Linear Programming
- Second Order Conic Programming
- Unconstrained Optimisation
NEOS Interfaces

- Through Internet: Upload of model and data files
- Through Email: Upload of model and data files
- AMPL/GAMS *via* Kestrel
- NEOS API (XML-RPC): Available clients
  - Python
  - Perl
  - PHP
  - C and C++
  - Java
  - Ruby
  - and now, in R, too

In all four cases, the input is dependent on the chosen solver; but problems casted in AMPL and/or GAMS are most commonly encountered.
Implementation of XML-RPC client-side API
Employs S4 classes and methods (NAMESPACE)
Dependencies: XMLRPC, RCurl, XML
Availability:
R-Forge: http://r-forge.r-project.org/projects/rneos/
CRAN:
http://cran.r-project.org/web/packages/rneos/index.html
R package rneos

Package Structure

- **Classes:** `NeosComm`, `NeosAns`, `NeosXml`, `NeosJob`
- **Functions:**
  - API: `NemailHelp()`, `NgetFinalResults()`, `NgetFinalResultsNonBlocking()`, `NgetIntermediateResults()`, `NgetIntermediateResultsNonBlocking()`, `NgetJobInfo()`, `NgetJobStatus()`, `NgetSolverTemplate()`, `Nhelp()`, `NkillJob()`, `NlistAllSolvers()`, `NlistCategories()`, `NlistSolversInCategory()`, `Nping()`, `NprintQueue()`, `NsubmitJob()`, `Nversion()`, `Nwelcome()`
  - Utility: `CreateNeosComm()`, `CreateXmlString()`
- **Methods:** `show`, `update`

Nota bene: API functions are prefixed with 'N', hence `Nfoo()` designates the API function `foo`. 
$TITLE Stochastic Two-stage program
* TwoStageStochastic.gms: Stochastic Two-stage program.
* Consiglio, Nielsen and Zenios.
* PRACTICAL FINANCIAL OPTIMIZATION: A Library of GAMS Models, Section 6.3.1
SET Assets Available assets
   /Stock, Put_1, Call_1, Put_2, Call_2/;
SET Assets_1(Assets) Assets available up to the end of the first stage
   /Stock, Put_1, Call_1/;
SET Assets_2(Assets) Assets available up to the end of the second stage
   /Stock, Put_2, Call_2/;
SET Scenarios Set of scenarios
   /SS_1 * SS_3/;
ALIAS (Assets, i);
ALIAS (Assets_1, j);
ALIAS (Assets_2, k);
ALIAS (Scenarios, l);
PARAMETER pr(l) Scenario probability
   /SS_1 = 0.25,
      SS_2 = 0.50,
      SS_3 = 0.25/;
PARAMETER P_1(j) Asset prices at the beginning of the first stage
   /Stock = 43,
      Put_1 = 0.81,
      Call_1 = 4.76/;
R package rneos

Work flow: Two-Stage in GMS (cont’d.)

TABLE P_2(1,i) Asset prices (values) at the beginning of the second stage

<table>
<thead>
<tr>
<th>Stock</th>
<th>Put_1</th>
<th>Call_1</th>
<th>Put_2</th>
<th>Call_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS_1</td>
<td>44</td>
<td>1</td>
<td>0</td>
<td>0.92</td>
</tr>
<tr>
<td>SS_2</td>
<td>36</td>
<td>0</td>
<td>4</td>
<td>1.40</td>
</tr>
<tr>
<td>SS_3</td>
<td>47</td>
<td>2</td>
<td>0</td>
<td>3.02</td>
</tr>
</tbody>
</table>

TABLE V(l,k) Asset prices (values) at the end of the second stage

<table>
<thead>
<tr>
<th>Stock</th>
<th>Put_2</th>
<th>Call_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS_1</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>SS_2</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>SS_3</td>
<td>55</td>
<td>4</td>
</tr>
</tbody>
</table>

POSITIVE VARIABLES

x(j) First-stage holdings
y(l,k) Second-stage holdings;

VARIABLE

z Objective function value;

EQUATIONS

BudgetCon Equation defining the budget contraint
ObjDef Objective function definition
MinReturnCon(l) Equation defining the minimum return contraint
RebalanceCon(l) Equation defining the rebalance contraint;

ObjDef .. z =E= SUM((k,l), pr(l) * V(l,k) * y(l,k));
BudgetCon .. SUM(j, P_1(j) * x(j)) =L= 10000;
MinReturnCon(l) .. SUM(k, V(l,k) * y(l,k)) =G= 11500;
RebalanceCon(l) .. SUM(j, P_2(l,j) * x(j)) =G= SUM(k, P_2(l,k) * y(l,k));

MODEL StochasticTwoStage /ALL/;
SOLVE StochasticTwoStage MAXIMIZING z USING LP;
DISPLAY x.l,z.l;
```r
library(rneos)
## NEOS: ping
Nping()
## NEOS: listCategories
NlistCategories()
## NEOS: listSolversInCategory
NlistSolversInCategory(category = "lp")
## NEOS: getSolverTemplate
template <- NgetSolverTemplate(category = "lp", solvername = "MOSEK", inputMethod = "GAMS")
template
modc <- paste(paste(readLines("TwoStageStochastic.gms"), collapse = "\n"), "\n")
cat(modc)
argslist <- list(model = modc, options = "", wantlog = "", comments = "")
xmls <- CreateXmlString(neosxml = template, cdatalist = argslist)
## NEOS: printQueue
NprintQueue()
## NEOS: submitJob
(test <- NsubmitJob(xmlstring = xmls, user = "rneos", interface = "", id = 0))
## NEOS: getJobStatus
NgetJobStatus(obj = test)
## NEOS: getFinalResults
NgetFinalResults(obj = test)
```
Intended package enhancements:

- Offer methods for updating model specifications
- Offer methods for updating data/parameters of optimisation problems.
- Implement API for solver maintenance.
References

The neos server.
*IEEE Journal on Computational Science and Engineering* 5, 68–75.

The neos server 4.0 administrative guide.
Technical memorandum anl/mcs-tm-250, Mathematics and Computer Science Division, Argonne National Laboratory.


*RCurl: General network (HTTP/FTP/...) client interface for R.*
R package version 1.4-4.1.

*XML: Tools for parsing and generating XML within R and S-Plus.*
R package version 3.2-0.1.

*XMLRPC: Remote Procedure Call (RPC) via XML in R.*
R package version 0.2-0.