Evaluating Time Series Models

L. Torgo

ltorgo@fc.up.pt
Faculdade de Ciências / LIAAD-INESC TEC, LA
Universidade do Porto

Jan, 2017

Jožef Stefan Institute

Goals of an Evaluation Method

- The golden rule:

  The data used for evaluating (or comparing) any models cannot be seen during model development.

- The goal of any evaluation procedure:
  - Obtain a reliable estimate of some evaluation measure.
    High probability of achieving the same score on other samples of the same population.
- Evaluation Measures
  - Predictive accuracy.
  - Model size.
  - Computational complexity.

Jožef Stefan Institute

Obtaining Reliable Estimates

- The usual techniques for model evaluation revolve around resampling.
  - Simulating the reality.
    - Obtain an evaluation estimate for unseen data.
- Examples of Resampling-based Methods
  - Holdout.
  - Cross-validation.
  - Bootstrap.

Time Series Data Are Special!

Any form of resampling changes the natural order of the data!

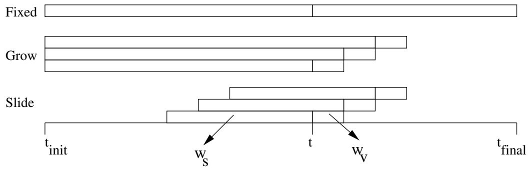
Jožef Stefan Institute

Correct Evalution of Time Series Models

- General Guidelines
  - Do not “forget” the time tags of the observations.
  - Do not evaluate a model on past data.
- A possible method
  - Divide the existing data in two time windows
    - Past data (observations till a time *t*).
    - “Future” data (observations after *t*).
  - Use one of these three learn-test alternatives
    - Fixed learning window.
    - Growing window.
    - Sliding window.

Jožef Stefan Institute

Learn-Test Strategies



Fixed Window

A single model is obtained with the available “training” data, and applied to all test period.

Growing Window

Every w_v test cases a new model is obtained using all data available till then.

Sliding Window

Every w_v test cases a new model is obtained using the previous w_s observations of the time series.

Dealing with model selection

- Most modelling techniques involve some form of parameters that usually need to be tunned.
- The following describes an evaluation methodology considering this issue:

	y_1 • • • y_s	• • • y_t	• • • y_n
Stage 1	Data used for obtaining the model alternatives	Model tunning and selection period	
Stage 2	Data used for obtaining the selected model alternative / variant		Final Evaluation Period

Jožef Stefan Institute

Some Metrics for Evaluating Predictive Performance

Absolute Measures

- Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2$$

- Mean Absolute Deviation (MAD)

$$MAD = \frac{1}{n}\sum_{i=1}^{n}|\hat{x}_i - x_i|$$

Relative Measures

- Theil Coefficient

$$U = \frac{\sqrt{\sum_{i=1}^{n}(\hat{x}_i - x_i)^2}}{\sqrt{\sum_{i=1}^{n}(x_i - x_{i-1})^2}}$$

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{(\hat{x}_i - x_i)}{x_i}\right|$$

The Metrics in R

```
set.seed(1234)
library(xts)
someSeries <- xts(rnorm(1000),seq.Date(from=Sys.Date(), length.out=1000,
                                       by ="1 day"))
somePreds <- xts(rnorm(1000),seq.Date(from=Sys.Date(), length.out=1000,
                                      by ="1 day"))
(mse <- mean((someSeries-somePreds)^2))

## [1] 1.846431

(mad <- mean(abs(someSeries-somePreds)))

## [1] 1.070021

(U <- sqrt(sum(((someSeries-somePreds)^2)[-1])) /
     sqrt(sum(((someSeries-lag(someSeries,1))^2)[-1])))

## [1] 0.9771318

(mape <- mean(abs((someSeries-somePreds)/someSeries)))

## [1] 4.551463
```

The Goal of an Experimental Comparison

- Given a set of observations of a time series X.
- Given a set of alternative modelling approaches M.
- Obtain estimates of the predictive performance of each m_i for this time series.

More specifically,
given a forecasting period size, w_{test},
and a predictive performance statistic, Err,
we want to obtain a reliable estimate of the value of Err
for each m_i.

Jožef Stefan Institute

Using Monte Carlo Simulations for Obtaining Reliable Estimates of *Err*

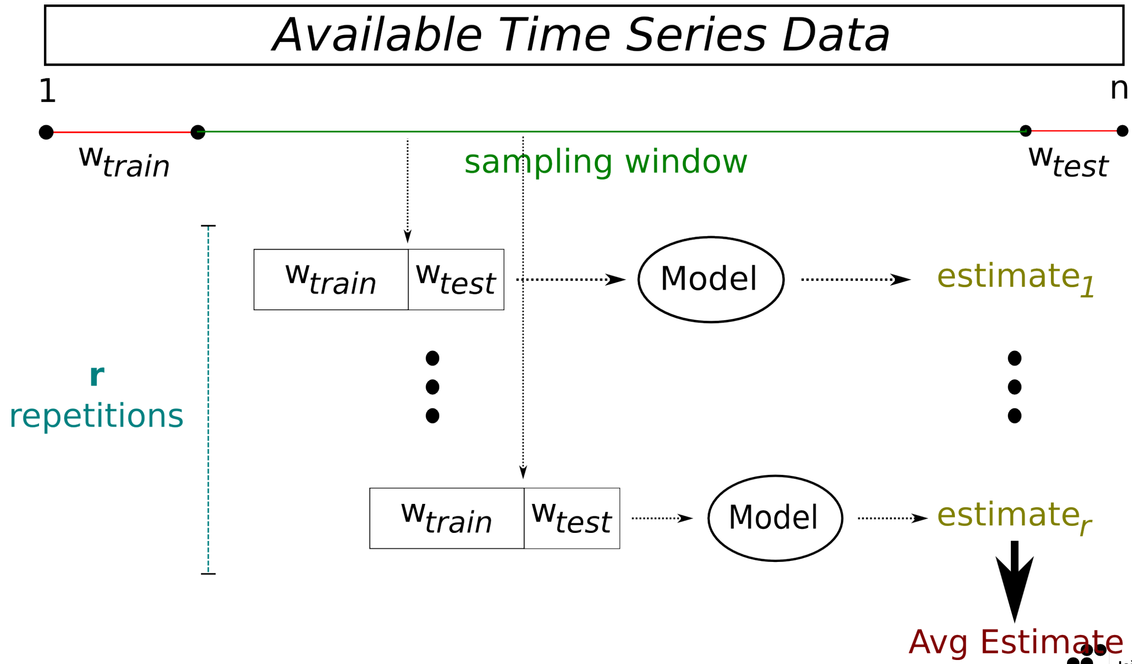
- A possible approach would be to use our proposed method of Model Selection.
- This would give us one estimate of Err.
- More reliability is achievable if more repetitions of the process are carried out.

Monte Carlo Estimates for Time Series Forecasting

Given: a time series, a training window size, w_{train}, a testing window size, w_{test}, and a number of repetitions, r,
- randomly generate r points in the interval $]w_{train}..(n - w_{test})[$,
- for each point proceed according to our Model Selection strategy.

Jožef Stefan Institute

Using Monte Carlo Simulations for Obtaining Reliable Estimates of *Err* - 2



The Infra-Structure of package **performanceEstimation**

- The package **performanceEstimation** provides a set of functions that can be used to carry out comparative experiments of different models on different predictive tasks
- This infra-structure can be applied to any model/task/evaluation metric
- Installation:
  - Official release (from CRAN repositories):

```r
install.packages("performanceEstimation")
```

  - Development release (from Github):

```r
library(devtools)  # You need to install this package before!
install_github("ltorgo/performanceEstimation",ref="develop")
```

The Infra-Structure of package **performanceEstimation**

- The main function of the package is `performanceEstimation()`
  - It has 3 arguments:
    1. The predictive tasks to use in the comparison
    2. The models to be compared
    3. The estimation task to be carried out
- The function implements a wide range of experimental methodologies including all we have discussed

Jožef Stefan Institute

A Simple Example

- Suppose we want to estimate the mean squared error of regression trees in a certain regression task using cross validation

```
library(performanceEstimation)
library(DMwR)
data(Boston,package='MASS')
res <- performanceEstimation(
    PredTask(medv ~ .,Boston),
    Workflow("standardWF",learner="rpartXse"),
    EstimationTask(metrics="mse",method=CV(nReps=1,nFolds=10)))
```

Jožef Stefan Institute

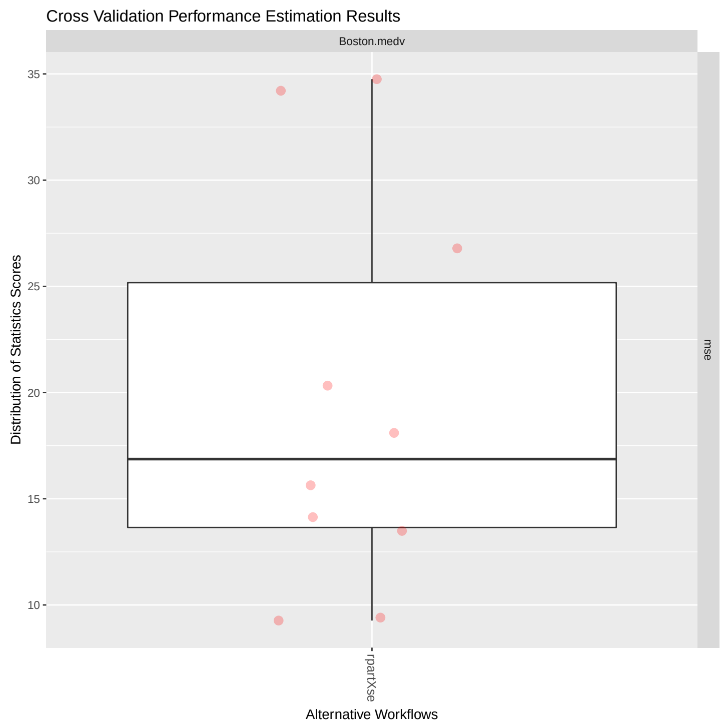
A Simple Example (2)

```
summary(res)

##
## == Summary of a  Cross Validation Performance Estimation Experiment ==
##
## Task for estimating  mse  using
##  1 x 10 - Fold Cross Validation
##    Run with seed =  1234
##
## * Predictive Tasks ::  Boston.medv
## * Workflows  ::  rpartXse
##
## -> Task:  Boston.medv
##   *Workflow: rpartXse
##                mse
## avg      19.610531
## std       9.375305
## med      16.867969
## iqr      11.523275
## min       9.266761
## max      34.752888
## invalid  0.000000
```

Jožef Stefan Institute

A Simple Example (3)

```
plot(res)
```



Jožef Stefan Institute

Predictive Tasks

- Objects of class **PredTask** describing a predictive task
  - Classification
  - Regression
  - Time series forecasting
- Created with the constructor with the same name

```
data(iris)
PredTask(Species ~ ., iris)

## Prediction Task Object:
##   Task Name          :: iris.Species
##   Task Type          :: classification
##   Target Feature     :: Species
##   Formula            :: Species ~ .
##   Task Data Source   :: iris

PredTask(Species  ~ ., iris,"IrisDS",copy=TRUE)

## Prediction Task Object:
##   Task Name          :: IrisDS
##   Task Type          :: classification
##   Target Feature     :: Species
##   Formula            :: Species ~ .
##   Task Data Source   :: internal  150x5 data frame.
```


Workflows

- Objects of class **Workflow** describing an approach to a predictive task
  - Standard Workflows
    - Function `standardWF` for classification and regression
    - Function `timeseriesWF` for time series forecasting
  - User-defined Workflows

Standard Workflows for Classification and Regression Tasks

```
library(e1071)
Workflow("standardWF",learner="svm",learner.pars=list(cost=10,gamma=0.1))

## Workflow Object:
##  Workflow ID       ::  svm
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner  -> svm
##   learner.pars  -> cost=10 gamma=0.1
```

"standardWF" can be omitted ...

```
Workflow(learner="svm",learner.pars=list(cost=5))

## Workflow Object:
##  Workflow ID       ::  svm
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner  -> svm
##   learner.pars  -> cost=5
```


Standard Workflows for Classification and Regression Tasks (cont.)

- Main parameters of the constructor:
  - Learning stage
    - `learner` - which function is used to obtain the model for the training data
    - `learner.pars` - list with the parameter settings to pass to the learner
  - Prediction stage
    - `predictor` - function used to obtain the predictions (defaults to `predict()`)
    - `predictor.pars` - list with the parameter settings to pass to the predictor

Jožef Stefan Institute

Standard Workflows for Classification and Regression Tasks (cont.)

- Main parameters of the constructor (cont.):
  - Data pre-processing
    - `pre` - vector with function names to be applied to the training and test sets before learning
    - `pre.pars` - list with the parameter settings to pass to the functions
  - Predictions post-processing
    - `post` - vector with function names to be applied to the predictions
    - `post.pars` - list with the parameter settings to pass to the functions

Jožef Stefan Institute

Standard Workflows for Classification and Regression Tasks (cont.)

```
data(algae,package="DMwR")
res <- performanceEstimation(
    PredTask(a1 ~ .,algae[,1:12],"A1"),
    Workflow(learner="lm",pre="centralImp",post="onlyPos"),
    EstimationTask("mse",method=CV())       # defaults to 1x10-fold CV
                          )

##
##
## ##### PERFORMANCE ESTIMATION USING  CROSS VALIDATION  #####
##
## ** PREDICTIVE TASK :: A1
##
## ++ MODEL/WORKFLOW :: lm
## Task for estimating  mse  using
##  1 x 10 - Fold Cross Validation
##   Run with seed =  1234
## Iteration :**********
```

Evaluating Variants of Workflows

Function `workflowVariants()`

```
library(e1071)
data(Boston,package="MASS")
res2 <- performanceEstimation(
    PredTask(medv ~ .,Boston),
    workflowVariants(learner="svm",
                     learner.pars=list(cost=1:5,gamma=c(0.1,0.01))),
    EstimationTask(metrics="mse",method=CV()))
```

Jožef Stefan Institute

Evaluating Variants of Workflows (cont.)

```
summary(res2)

##
## == Summary of a  Cross Validation Performance Estimation Experiment ==
##
## Task for estimating  mse  using
##  1 x 10 - Fold Cross Validation
##   Run with seed =  1234
##
## * Predictive Tasks ::  Boston.medv
## * Workflows  ::  svm.v1, svm.v2, svm.v3, svm.v4, svm.v5, svm.v6, svm.v7, svm.v8, svm.v9, svm.v10
##
## -> Task:  Boston.medv
##   *Workflow: svm.v1
##                mse
## avg     14.80685
## std     10.15295
## med     12.27015
## iqr     11.87737
## min      5.35198
## max     38.39681
## invalid  0.00000
##
##   *Workflow: svm.v2
##                 mse
## avg     11.995178
## std      7.908371
## med      8.359433
## iqr     11.626306
## min      4.842848
## invalid  0.000000
##
```

Exploring the Results

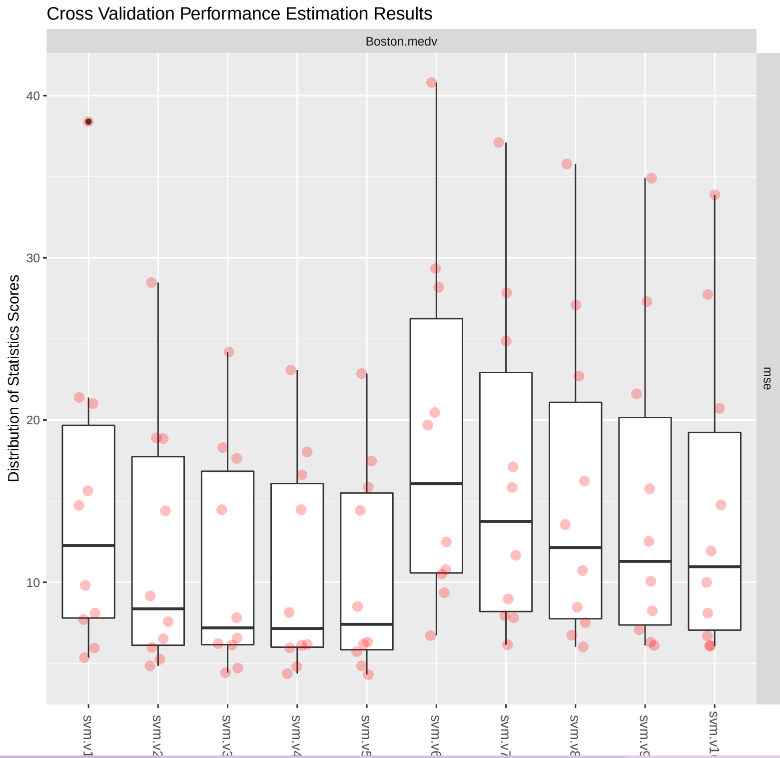
```
getWorkflow("svm.v1",res2)

## Workflow Object:
##  Workflow ID       ::  svm.v1
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner.pars  -> cost=1 gamma=0.1
##   learner  -> svm

topPerformers(res2)

## $Boston.medv
##     Workflow Estimate
## mse   svm.v5    10.65
```

Jožef Stefan Institute

Package Performance Estimation | Workflows and Workflow Variants

Visualizing the Results

```
plot(res2)
```



Jožef Stefan Institute

Estimation Tasks

- Objects of class **EstimationTask** describing the estimation task
  - Main parameters of the constructor
    - **metrics** - vector with names of performance metrics
    - **method** - object of class **EstimationMethod** describing the method used to obtain the estimates

```
EstimationTask(metrics=c("F","rec","prec"),method=Bootstrap(nReps=100))

## Task for estimating  F,rec,prec  using
## 100  repetitions of  e0  Bootstrap experiment
##   Run with seed =  1234
```

Jožef Stefan Institute

Performance Metrics

- Many classification and regression metrics are available
  - Check the help page of functions `classificationMetrics` and `regressionMetrics`
- User can provide a function that implements any other metric she/he wishes to use
  - Parameters **evaluator** and **evaluator.pars** of the `EstimationTask` constructor

Jožef Stefan Institute

Comparing Different Algorithms on the Same Task

```
library(randomForest)
library(e1071)
res3 <- performanceEstimation(
    PredTask(medv ~ ., Boston),
    workflowVariants("standardWF",
             learner=c("rpartXse","svm","randomForest")),
    EstimationTask(metrics="mse",method=CV(nReps=2,nFolds=5)))
```

Jožef Stefan Institute

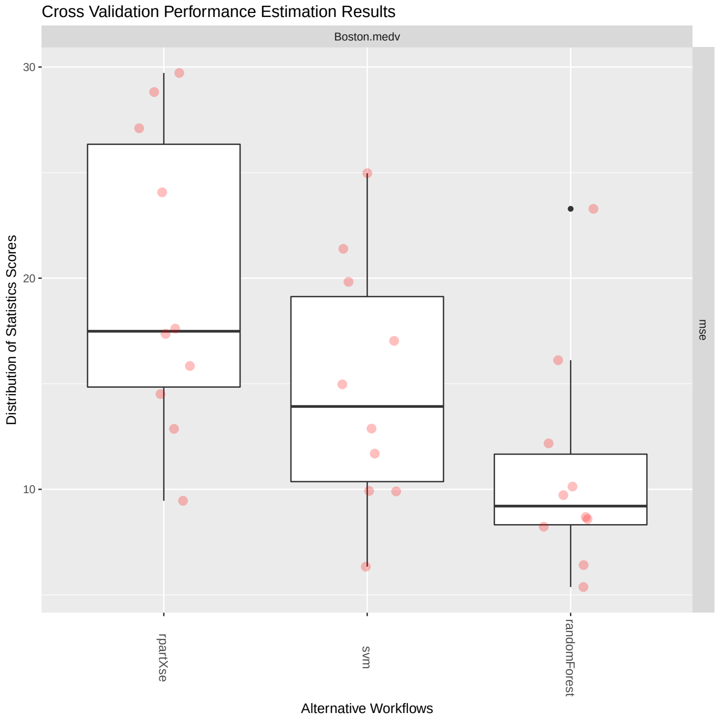
Some auxiliary functions

```
rankWorkflows(res3,3)

## $Boston.medv
## $Boston.medv$mse
##        Workflow Estimate
## 1  randomForest 10.87221
## 2           svm 14.89183
## 3      rpartXse 19.73468
```

Jožef Stefan Institute

The Results

```
plot(res3)
```

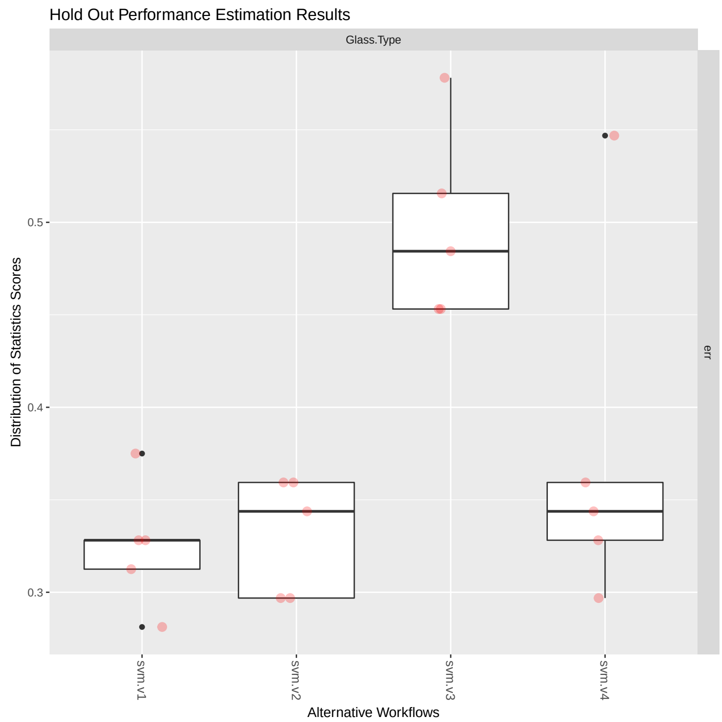


An example using Holdout and a classification task

```
data(Glass,package='mlbench')
res4 <- performanceEstimation(
    PredTask(Type ~ ., Glass),
    workflowVariants(learner="svm",   # You may omit "standardWF" !
                     learner.pars=list(cost=c(1,10),
                                       gamma=c(0.1,0.01))),
    EstimationTask(metrics="err",method=Holdout(nReps=5,hldSz=0.3)))
```

The Results

```
plot(res4)
```



Jožef Stefan Institute

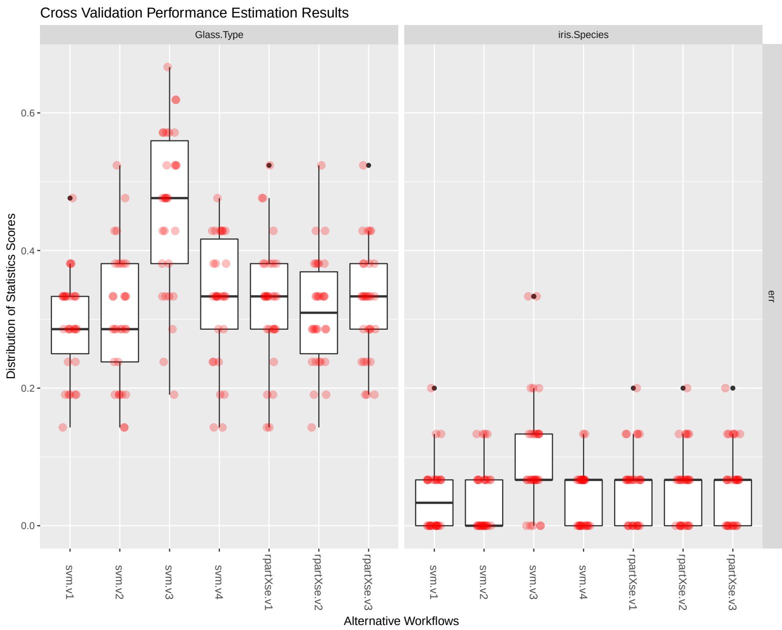
An example involving more than one task

```
data(Glass,package='mlbench')
data(iris)
res5 <- performanceEstimation(
    c(PredTask(Type ~ ., Glass),PredTask(Species ~.,iris)),
    c(workflowVariants(learner="svm",
                       learner.pars=list(cost=c(1,10),
                                         gamma=c(0.1,0.01))),
      workflowVariants(learner="rpartXse",
                       learner.pars=list(se=c(0,0.5,1)),
                       predictor.pars=list(type="class"))),
    EstimationTask(metrics="err",method=CV(nReps=3)))
```

Jožef Stefan Institute

The Results

```r
plot(res5)
```



Jožef Stefan Institute

The Results (2)

```r
topPerformers(res5)

## $Glass.Type
##      Workflow Estimate
## err    svm.v1    0.294
##
## $iris.Species
##      Workflow Estimate
## err    svm.v2     0.04

topPerformer(res5,"err","Glass.Type")

## Workflow Object:
##  Workflow ID       ::  svm.v1
##  Workflow Function ::  standardWF
##       Parameter values:
##   learner.pars  -> cost=1 gamma=0.1
##   learner  -> svm
```

Jožef Stefan Institute

An example involving time series

First getting the data and building an illustrative data set

```
library(quantmod)
library(lubridate)
getSymbols("GOOGL",from=Sys.Date() - years(5))

## [1] "GOOGL"

gg <- Delt(Cl(GOOGL))
library(DMwR2)
library(TTR)
dat <- createEmbedDS(gg, emb=7)
dat <- data.frame(cbind(lag(gg,-1),
                        dat,
                        MA10=SMA(gg,10),
                        RSI=RSI(gg),
                        BB=BBands(gg)$pctB))
colnames(dat)[1] <- "FutureT"
dat <- na.omit(dat)
```

Jožef Stefan Institute

An example involving time series - 2

Now comparing models

```
library(e1071)
library(randomForest)
tsExp <- performanceEstimation(
    PredTask(FutureT ~ .,dat,'GG'),
    c(Workflow('timeseriesWF',wfID="slideSVM",
               type="slide",relearn.step=90,
               learner='svm',learner.pars=list(cost=10,gamma=0.01)),
      Workflow('timeseriesWF',wfID="slideRF",
               type="slide",relearn.step=90,
               learner='randomForest',learner.pars=list(ntrees=500))
      ),
    EstimationTask(metrics="theil",
                   method=MonteCarlo(nReps=10,szTrain=0.5,szTest=0.25)))
```

Jožef Stefan Institute

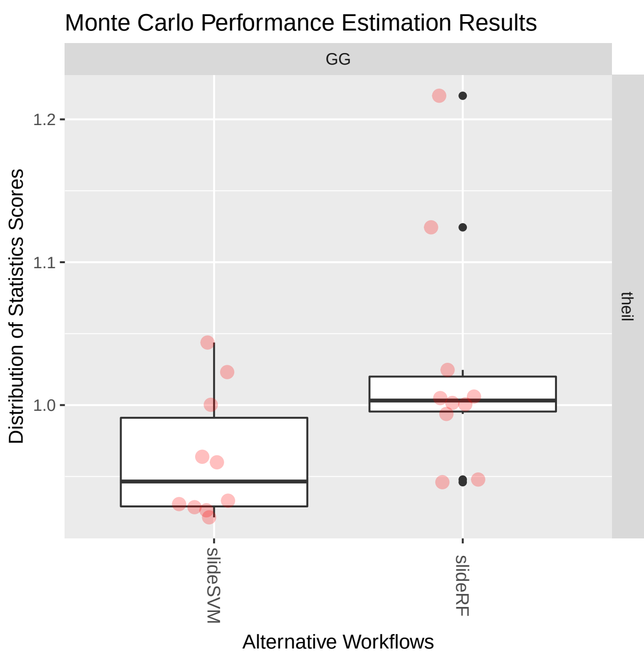
Checking the results

```
summary( tsExp )
```

```
##
## == Summary of a  Monte Carlo Performance Estimation Experiment ==
##
## Task for estimating  theil  using
## 10  repetitions Monte Carlo Simulation using:
##   seed =  1234
##   train size =  0.5 x NROW(DataSet)
##   test size =  0.25 x NROW(DataSet)
##
## * Predictive Tasks ::  GG
## * Workflows  ::  slideSVM, slideRF
##
## -> Task:  GG
##   *Workflow: slideSVM
##               theil
## avg      0.96309293
## std      0.04438200
## med      0.94650894
## iqr      0.06204175
## min      0.92142661
## max      1.04377023
## invalid 0.00000000
##
##   *Workflow: slideRF
##               theil
## avg      1.02660932
## std      0.08274638
## med      1.00318257
## min      0.94601480
## max      1.21653003
## invalid 0.00000000
```


Checking the results - 2

```
plot( tsExp )
```



Hands on Performance Estimation

the Algae data set

Load in the data set `algae` from package **DMwR** and answer the following questions:

1. Estimate the MSE of a regression tree for forecasting alga *a1* using 10-fold Cross validation.
2. Repeat the previous exercise this time trying some variants of random forests. Check what are the characteristics of the best performing variant.
3. Compare the results in terms of mean absolute error of the default variants of a regression tree, a linear regression model and a random forest, in the task of predicting alga a3. Use 2 repetitions of a 5-fold Cross Validation experiment.

Jožef Stefan Institute