Evaluation Methods for Forecasting Time Series in R

The following is a script file (RCode/RtimeseriesEvaluation.R) containing all R code of all sections in this slide set.

Reliable Estimates

```
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))
(mad <- mean(abs(someSeries-somePreds)))
(U <- sqrt(sum(((someSeries-somePreds)^2)[-1])) /
     sqrt(sum(((someSeries-lag(someSeries,1))^2)[-1])))
(mape <- mean(abs((someSeries-somePreds)/someSeries)))
```

Experimental Comparisons

Package Performance Estimation

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

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

```
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)))
```

```
summary(res)
```

```
plot(res)
```

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

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

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

```
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
                             )
```

```
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()))
```

```
summary(res2)
```

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

```
plot(res2)
```

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

```
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)))
```

```
rankWorkflows(res3,3)
```

```
plot(res3)
```

```
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)))
```

```
plot(res4)
```

```
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)))
```

```
plot(res5)
```

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

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

```
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)))
```

```
summary( tsExp )
```

```
plot( tsExp )
```