

Metaheurystyki i Technologie Wieloagentowe - Lab3

Bartłomiej Twardowski, B.Twardowski@ii.pw.edu.pl

1/2/2015

Organizational information

e-learning part

```
e_learning_part <- function(student) {  
  if(student.missing_classes > 1) {  
    TRUE  
  } else {  
    FALSE // ;-)  
  }  
}
```

Lab3 - Exercise

Data preparation

Simple analysis

Predict feature with metaheuristic

Data preparation

Go to Polish Stock Exchange website:

http:

[//www.gpwinfostrefa.pl/GPWIS2/pl/quotes/archive/1](http://www.gpwinfostrefa.pl/GPWIS2/pl/quotes/archive/1)

Download historical data

Choose one year period (e.g. from *2013/01/01* to *2014/01/01*) and download data for selected company (e.g. *IVMX*).

Data will be downloaded in Excel file named like:

PLMATRX00017.xls

TODO: Open Excel and look at the downloaded data.

Load data to R

To load data from Excel to R we can use package `xlsx`.

```
install.packages('xlsx')
```

Reading data into R:

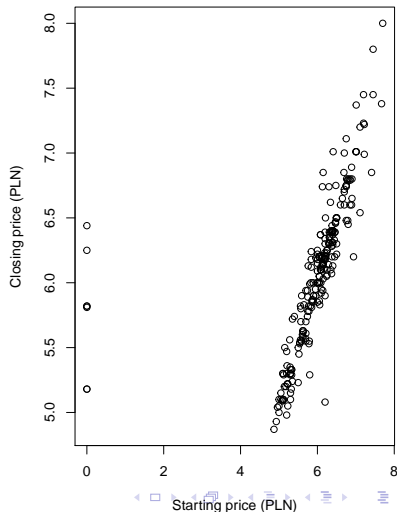
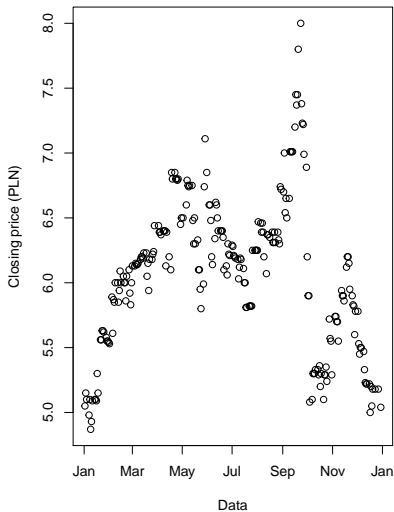
```
require('xlsx')
stock_data <- read.xlsx('../data/PLMATRX00017.xls', 1)
# Date type conversion
stock_data$Data=as.Date(stock_data$Data,format="%Y-%m-%d")
```

Look what's inside dataframe:

```
summary(stock_data)
head(stock_data)
```

Analysis - Plot the data

```
par(mfrow=c(1,2))  
plot(stock_data$Kurs.zamknienia~stock_data$Data, xlab="Data", ylab="Closing price (PLN)")  
plot(stock_data$Kurs.zamknienia~stock_data$Kurs.otwarcia, xlab="Starting price (PLN)", ylab="Closing price (PLN)")
```

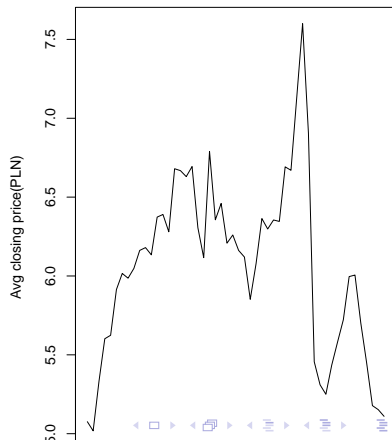
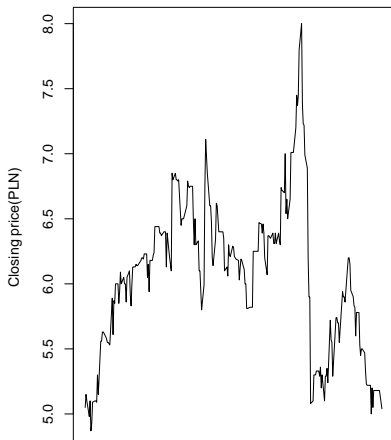


Aggregate data by week

```
stock.week.avg <- as.numeric(  
  aggregate(  
    stock_data$Kurs.zamknienia,  
    by=list(floor((stock_data$Data-stock_data$Data[1])/7)  
    FUN=mean)$x  
  )
```

Plot the result

```
par(mfrow=c(1,2))  
with(stock_data, {  
  plot(Kurs.zamknienia~Data,type='l',xlab="Date",ylab="Closing price(PLN)")  
})  
plot(stock.week.avg,type='l',xlab="Week",ylab="Avg closing price(PLN)")
```



Predicting average week price

Predicting week price base on waighted avarage from last weeks.
Parameters of prediction are *weights* for avarage.

```
predict_price <- function(weights=c(w1,w2,w3,w4), weekNum,  
  prediction <- 0  
  weight_sum <-0  
  for(i in 1:length(weights)) {  
    if(weekNum>i) {  
      prediction <- prediction + weights[i]*weekAvg[weekNum-i]  
      weight_sum <- weight_sum + weights[i]  
    }  
  }  
  return (prediction/weight_sum)  
}
```

Predicting average week price - example

Simple average from last 4 weeks:

```
test_week_num <- 10  
predict_price(c(1,1,1,1),test_week_num, stock.week.avg)
```

```
## [1] 5.991
```

```
stock.week.avg[test_week_num]
```

```
## [1] 6.162
```

Parameters evaluation

Cost function will be Mean Root Square function calculated on all dataset for given parameters (weights for our prediction).

```
predict_eval <- function(weights=c(w1,w2,w3,w4), weekAvg =  
  predictions <-as.numeric(lapply(4:length(weekAvg),  
                                FUN=function(w) predict_price(weights,w  
                                )  
                                )  
                                )  
  
  ## data normalization!  
  max_val=max(abs(c(weekAvg[-(1:3)],predictions)))  
  return(sqrt(mean((weekAvg[-(1:3)]/max_val-predictions/max_val  
})
```

Finding best weights

Optimization task -> CRAN Task View: Optimization and Mathematical Programming

To find weights for our prediction which best fits the model we use one of metaheuristic algorithm: **Particle Swarm Optimization** from package **pso**.

```
install.packages('pso')  
?pso::psoptim
```



Particle Swarm Optimization

- ▶ optimizes a problem by iteratively trying to improve a candidate solution
- ▶ population -> swarm, candidate solutions -> called particles
- ▶ movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position
- ▶ example visualization: <http://vimeo.com/17407010>
- ▶ nice lecture about stochastic optimization
<https://www.youtube.com/watch?v=C3jyhZhyNE4>

Running optimizer

```
library(pso)
#random initial weights
init_weights=runif(4,0,1)
pso_result<-psoptim(init_weights, predict_eval, gr=NULL, st
                    lower=rep(0,length(init_weights)),
                    upper=rep(1,length(init_weights)),
                    control=list(maxit=100,type='SPS02011',
                                reltol=1e-8,vectorize=TRUE))

pso_result$par #best weights
```

```
## [1] 0.99109029 0.00000000 0.00000000 0.01830776
```

```
pso_result$value #minimal cost(eval) function
```

```
## [1] 0.0424164
```

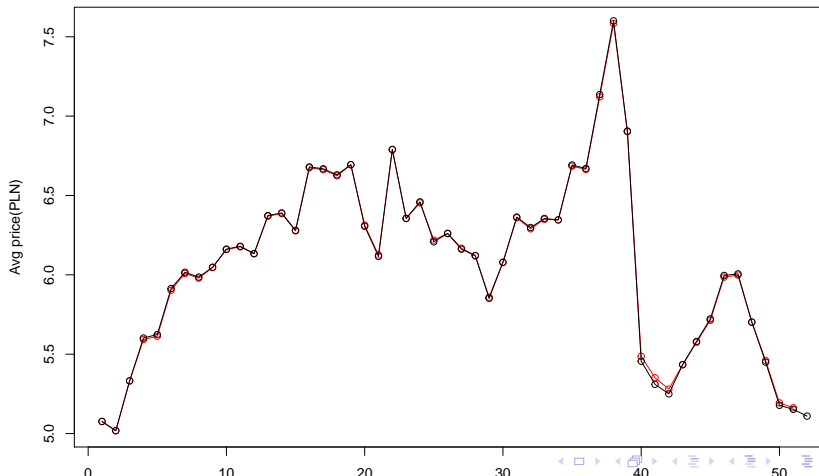
PSO Result visualization

Helping function

```
all_week_predictions<-function(par,weekAvg=stock.week.avg)
  return (as.numeric(lapply(
    2:length(weekAvg),
    FUN=function(n) predict_price(par,n,weekAvg)
  )))
}
```

PSO Result visualization - Plot

```
pso_predictions <- all_week_predictions(pso_result$par)
plot(pso_predictions,type='o',col='red',
     ylim=c(min(pso_predictions),max(pso_predictions)),xlab=
points(stock.week.avg,type='o')
```



PSO Result visualization - With dummy (1,1,1,1) prediction

```
dummy_predictions <- all_week_predictions(c(1,1,1,1))  
points(dummy_predictions,type='o',col='green')
```

