Visualizations for Algorithmic trading is rising in demanded by the economic sector. In R there are a lot of great packages for getting data, visualizations and model strategies for algoritmic trading. In this article you learn how to perform visualizations and modelling for algorithmic trading in R

**Introduction to Algorithmic Trading**

Algorithmic trading is a very popular machine learning method within the economic and financial sector. Typically it involves a lot of programming and visualization. The programming is nesecary in order to get the financial data for the Algorithmic Trading analysis.

**Read packages into R library**

First things first! We need to read these great packages into the R library:

# Load R package

library(PortfolioEffectHFT)

library(rvest)

library(pbapply)

library(TTR)

library(dygraphs)

library(lubridate)

library(tidyquant)

library(timetk)

library(pacman)

library(quantmod)

library(parallelMap)

library(BiocParallel)

library(parallel)

library(plotly)

pacman::p\_load(dygraphs,DT)

In the above packages you need to install BiocParallel with this code:

## Install BiocParallel

source("https://bioconductor.org/biocLite.R")

biocLite("BiocParallel")

**Read data into R for Algorithmic Trading**

Next it is time to get the data. This involves a lot of programming in R:

# Data Collection

website <- read\_html("https://www.marketwatch.com/tools/industry/stocklist.asp?bcind\_ind=9535&bcind\_period=3mo")

table <- html\_table(html\_nodes(website, "table")[[4]], fill = TRUE)

stocks.symbols<-table$X2

stocks.names<-table$X3

table1<-table[-1,-1]

colnames(table1)<-table[1,-1]

DT::datatable(table1)

stock.list<-"https://www.marketwatch.com/tools/industry/stocklist.asp?bcind\_ind=9535&bcind\_period=3mo"

stocks<-read\_html(stock.list)

stocks.names<-html\_nodes(stocks,".lk01")

stocks.names<-html\_text(stocks.names)

table1[table1==""] <- NA

table1<-table1[complete.cases(table1$Symbol),]

DT::datatable(table1)

# Date creation

start.date<-Sys.Date()

end.date<-Sys.Date()-years(3)

start.date<-gsub('-','', start.date)

end.date<-gsub('-','', end.date)

start.date

# The symbols vector holds our tickers.

symbols <- c("SPY","EFA", "IJS", "EEM","AGG")

# The prices object will hold our raw price data

prices %

map(~Ad(get(.))) %>% #Extract (transformed) data from a suitable OHLC object. getSymbols('IBM',src='yahoo') Ad(IBM)

reduce(merge) %>% #reduce() combines from the left, reduce\_right() combines from the right

`colnames<-`(symbols)

head(prices)

tickers <- c("AAPL", "MSFT","GOOGL","IBM","FB")

getSymbols(tickers)

closePrices <- do.call(merge, lapply(tickers, function(x) Cl(get(x))))

# Mapping

parallelStartSocket(2)

parallelStartMulticore(cpus=6)# start in socket mode and create 2 processes on localhost

f = function(x) Cl(get(x)) # define our job

y = parallelMap(f, tickers) # like R's Map but in parallel

mapdata%head()

# Bioconductor

f = function(x) Ad(get(x))

options(MulticoreParam=quote(MulticoreParam(workers=4)))

param <- SnowParam(workers = 2, type = "SOCK")

vec=c(tickers[1],tickers[2],tickers[3],tickers[4])

multicoreParam <- MulticoreParam(workers = 7)

bio=bplapply(tickers, f, BPPARAM = multicoreParam)

biodata%head()

**Make visualizations for Algorithmic Trading**

Now we can make the first graph visualizations:

# Visualization graph

AdjustedPrices<-biodata

dateWindow %

dyRebase(value = 100) %>%

dyRangeSelector(dateWindow = dateWindow)

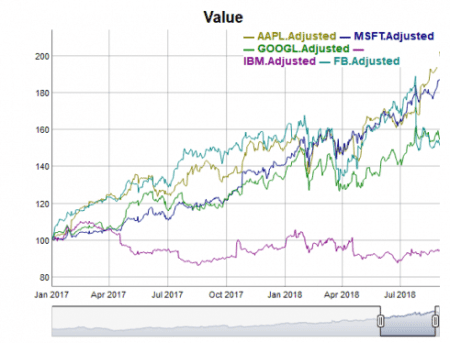
end<-Sys.Date()

start<-Sys.Date()-years(3)

prices %head()

pacman::p\_load(dygraph)

This gives us the following visualization graph with dygraph:

[](https://i0.wp.com/datascienceplus.com/wp-content/uploads/2018/10/V1-5.png?ssl=1)

Now loet us calculate the number of cores in the data and apply this in the visualization graph:

# Calculate the number of cores

no\_cores <- detectCores() - 1

# Initiate cluster

cl <- makeCluster(no\_cores)

f<-function(x) Ad(get(x))

AdjustedPrices %head()

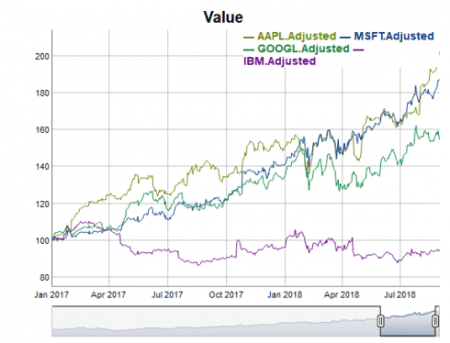
# vISUALIZATION inkl number of cores

dateWindow %

dyRebase(value = 100) %>%

dyRangeSelector(dateWindow = dateWindow)

This gives us the following visualization:

[](https://i1.wp.com/datascienceplus.com/wp-content/uploads/2018/10/v2.png?ssl=1)

Before making the third visualisation we need to do some more programming:

# QUANTITATIVE MODELLING

getSymbols("AAPL",src='yahoo')

# basic example of ohlc charts

df <- data.frame(Date=index(AAPL),coredata(AAPL))

df <- tail(df, 30)

# cutom colors

i <- list(line = list(color = '#FFD700'))

d <- list(line = list(color = '#0000ff'))

p %

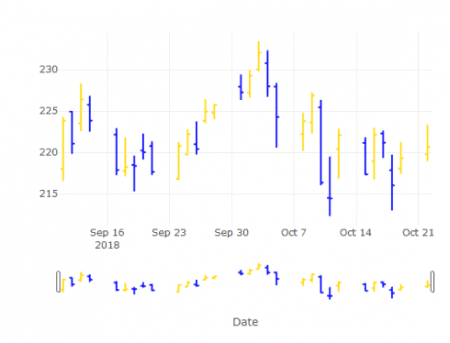
plot\_ly(x = ~Date, type="ohlc",

open = ~AAPL.Open, close = ~AAPL.Close,

high = ~AAPL.High, low = ~AAPL.Low,

increasing = i, decreasing = d)

p

The above programming gives us the following visualization:  
[](https://i1.wp.com/datascienceplus.com/wp-content/uploads/2018/10/v3.png?ssl=1)

Next let us make a programmed visualization of adjusted stock prices over time:

# Programmed visualization of adjusted stock prices over time

biodatadf%rename(Date=index)

data%filter(Date>"2014-01-11")

p %

add\_trace(y = ~MSFT.Adjusted, name = 'MSFT', mode = 'lines')%>%

add\_trace(y = ~IBM.Adjusted, name = 'IBM', mode = 'lines')%>%

add\_trace(y = ~GOOGL.Adjusted, name = 'GOOGL', mode = 'lines')%>%

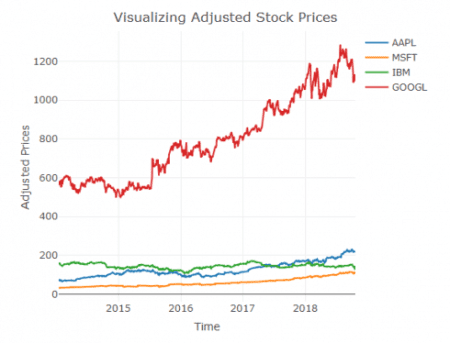
layout(title = "Visualizing Adjusted Stock Prices",

xaxis = list(title = "Time"),

yaxis = list (title = "Adjusted Prices"))

p

The above programming gives us the following visualization:

[](https://i0.wp.com/datascienceplus.com/wp-content/uploads/2018/10/V4.png?ssl=1)

**Simple Trading Strategy with Trend Following**

Next we will program a visualization of interactive Apple Moving Avereges over time:

## Simple Trading Strategy: Trend Following

tail(SMA(AdjustedPrices$AAPL.Adjusted, 200))

tail(SMA(AdjustedPrices$AAPL.Adjusted, 50))

data.frame(sma200=SMA(AdjustedPrices$AAPL.Adjusted, 200),sma50=SMA(AdjustedPrices$AAPL.Adjusted, 50))%>%head()

sdata%select(-Date)

#sdata<-tk\_xts(data,date\_var=Date)

df\_50=as.data.frame.matrix(apply(sdata, 2, SMA,50))

colnames(df\_50)=paste0(colnames(df\_50),"\_sma50")

df\_200=as.data.frame.matrix(apply(sdata, 2, SMA,200))

colnames(df\_200)=paste0(colnames(df\_200),"\_sma200")

df\_all%drop\_na()

# sma 50

f50<- function(x) SMA(x,50)

# sma 50

f200<- function(x) SMA(x,200)

df\_all%head()

dim(df\_all)

# Moving avereges

mov.avgs<-function(df){

ifelse((nrow(df)<(2\*260)),

x<-data.frame(df, 'NA', 'NA'),

x<-data.frame( SMA(df, 200), SMA(df, 50)))

colnames(x)<-c( 'sma\_200','sma\_50')

x%head()

var=names(df\_all)[str\_detect(names(df\_all), "AAPL")]

df\_all[,var]%>%head()

dateWindow=c("2015-01-01","2018-09-01")

dygraph(df\_all[,var],main = 'Apple Moving Averages') %>%

dySeries('AAPL.Adjusted\_sma50', label = 'sma 50') %>%

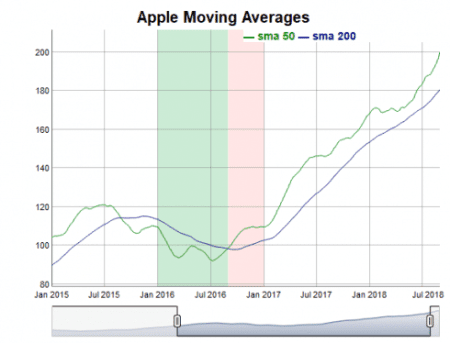
dySeries('AAPL.Adjusted\_sma200', label = 'sma 200') %>%

dyRangeSelector(height = 30) %>%

dyShading(from = '2016-01-01', to = '2016-9-01', color = '#CCEBD6') %>%

dyShading(from = '2016-9-01', to = '2017-01-01', color = '#FFE6E6')%>%

dyRangeSelector(dateWindow = dateWindow)

The above programming gives us the following visualization:  
[](https://i2.wp.com/datascienceplus.com/wp-content/uploads/2018/10/V5-1.png?ssl=1)