Review 2 Big Five

19BPS1103 Shivaditya Shivganesh

19BPS1104 J Gauthum

Setup

Load the Libraries

```
library(keras)
library(tensorflow)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(quantmod)
```

```
## Loading required package: xts
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Attaching package: 'xts'
## The following objects are masked from 'package:dplyr':
##
##     first, last
## Loading required package: TTR
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(TTR)
```

Load the Data

```
data <- read.csv('big_five_stocks.csv')
```

Dataset Description

Name of the Dataset Big Five

Description of the fields

```
str(data)
```

```
## 'data.frame':    41660 obs. of  7 variables:
##  $ X     : Factor w/ 12257 levels "1971-02-05","1971-02-08",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ name  : Factor w/ 6 levels "^IXIC","AAPL",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ open  : num  100 101 101 101 101 ...
##  $ close : num  100 101 101 101 101 ...
##  $ high  : num  100 101 101 101 101 ...
##  $ low   : num  100 101 101 101 101 ...
##  $ volume: num  0 0 0 0 0 0 0 0 0 0 ...
```

Build a time series prediction model using keras and tensorflow ## Transform data to stationary

```
get_ticker_data <- function (data,nam){
  return(filter(data, name == nam))
}

diffedMSFT = diff(get_ticker_data(data,'MSFT')$close,differences = 1)
diffedAAPL = diff(get_ticker_data(data,'AAPL')$close,differences = 1)
diffedAMZN = diff(get_ticker_data(data,'AMZN')$close,differences = 1)
diffedGOOGL = diff(get_ticker_data(data,'GOOGL')$close,differences = 1)
diffedIXIC = diff(get_ticker_data(data,'^IXIC')$close,differences = 1)
```

Lagged Dataset

```
lag_transform <- function(x, k=1){
  lagged = c(rep(NA, k), x[1:length(x)-k])
  DF = as.data.frame(cbind(lagged,x))
  colnames(DF) <- c(paste0('x-',k),'x')
  DF[is.na(DF)] <- 0
  return (DF)
}

SuperMSFT = lag_transform(diffedMSFT,1)
SuperAAPL = lag_transform(diffedAAPL,1)
SuperAMZN = lag_transform(diffedAMZN,1)
SuperGOOGL = lag_transform(diffedGOOGL,1)
SuperIXIC = lag_transform(diffedIXIC,1)
```

Amazon (NASDAQ: AMZN)

Split the data into training and teasting sets

```
N = nrow(SuperAMZN)
n = round(N*0.7,digits = 0)
train = SuperAMZN[1:n, ]
test = SuperAMZN[(n+1):N, ]
```

Normalize the data

```
scale_data = function(train, test, feature_range = c(0,1)){
  x = train
  fr_min = feature_range[1]
  fr_max = feature_range[2]
  std_train = ((x - min(x)) / (max(x) - min(x)))
  std_test = ((test - min(x)) / (max(x) - min(x) ))

  scaled_train = std_train * (fr_max - fr_min) + fr_min
  scaled_test = std_test * (fr_max - fr_min) + fr_min

  return (list(scaled_train = as.vector(scaled_train),scaled_test = as.vector(scaled_test),scaler = c(m

}

Scaled = scale_data(train,test,c(-1,1))
y_train = Scaled$scaled_train[,2]
x_train = Scaled$scaled_train[,1]

y_test = Scaled$scaled_test[, 2]
x_test = Scaled$scaled_test[, 1]
```

Invert from Normalized Scale to Orginal Scale

```
inver_scaling = function(scaled, scaler, feature_range = c(0,1)){
  min = scaler[1]
  max = scaler[2]
  t = length(scaled)
  mins = feature_range[1]
  maxs = feature_range[2]
  inverted_dfs = numeric(t)

  for (i in 1:t){
    X = (scaled[i] - mins)/(maxs - mins)
    rawValues = X*(max-min) + min
    inverted_dfs[i] <-rawValues
  }
  return(inverted_dfs)
  }
```

Modeling

```
dim(x_train) <- c(length(x_train),1 ,1)
```

```
#dim(y_train) <- c(length(y_train),1 ,1)

x_shape2 = dim(x_train)[2]
x_shape3 = dim(x_train)[3]
batch_size = 1
units = 1

model <- keras_model_sequential()
model %>%
  layer_lstm(units, batch_input_shape = c(batch_size, x_shape2, x_shape3), stateful = TRUE) %>%
  layer_dense(units = 1)

model %>% compile(
  loss = 'mean_squared_error',
  optimizer = optimizer_adam(learning_rate = 0.02, decay = 1e-6),
  metrics = c("accuracy")
)
summary(model)
```

Define the model

```
## Model: "sequential"
## _________________________________________________________________________________
## Layer (type)                         Output Shape                     Param #
## =================================================================================
## lstm (LSTM)                          (1, 1)                           12
## _________________________________________________________________________________
## dense (Dense)                        (1, 1)                           2
## =================================================================================
## Total params: 14
## Trainable params: 14
## Non-trainable params: 0
## _________________________________________________________________________________
```

```
Epochs = 50
for(i in 1:Epochs){
  model %>% fit(x_train, y_train, epochs = 1, batch_size = batch_size, verbose = 1, shuffle = FALSE)
  model %>% reset_states()
}
```

Fit the model

```
L = length(x_test)
scaler = Scaled$scaler
predictions = numeric(L)

for (i in 1:L){
  X  = x_test[i]
  dim(X) = c(1,1,1)
  yhat = model %>% predict(X, batch_size = batch_size)

  yhat = inver_scaling(yhat, scaler, c(-1,1))
```

```r
  yhat = yhat + get_ticker_data(data,'AMZN')$close[(n+i)]
  predictions[i] <- yhat
}
```

Make Predictions

```r
y_train_p = Scaled$scaled_train[,1]
scaler = Scaled$scaler
L = length(y_train_p)
y_train_p_uns = numeric(L)
for (i in 1:L){
  y_train_p_uns[i] <- train[,1] +  get_ticker_data(data,'AMZN')$close[(n+i)]
}
```

Line plots

```
## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length
```

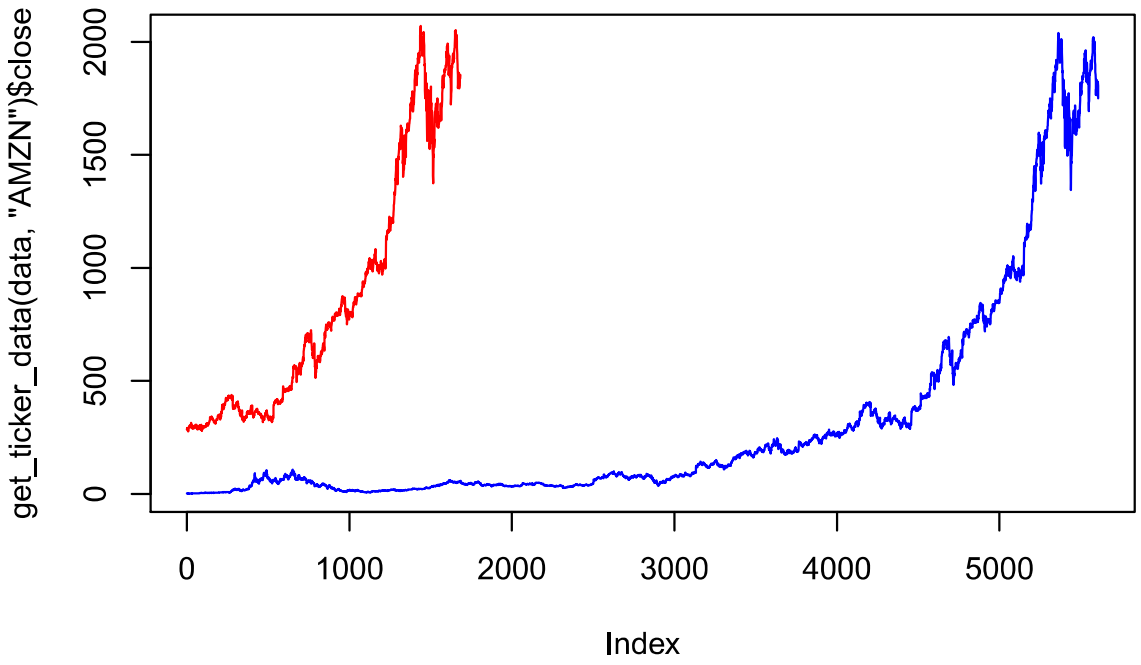
```
## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length

## Warning in y_train_p_uns[i] <- train[, 1] + get_ticker_data(data, "AMZN")
## $close[(n + : number of items to replace is not a multiple of replacement length
```

```
plot(get_ticker_data(data,'AMZN')$close,type="line",col="blue")
```

```
## Warning in plot.xy(xy, type, ...): plot type 'line' will be truncated to first
## character
```

```
lines(predictions,type="line",col="red")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): plot type 'line' will be
## truncated to first character
```



Charting the Data (Exploring the Data in a Financial Prespective)

#Apple (NASDAQ: AAPL)

Load the data

```
head(get_ticker_data(data,'AAPL'))
```

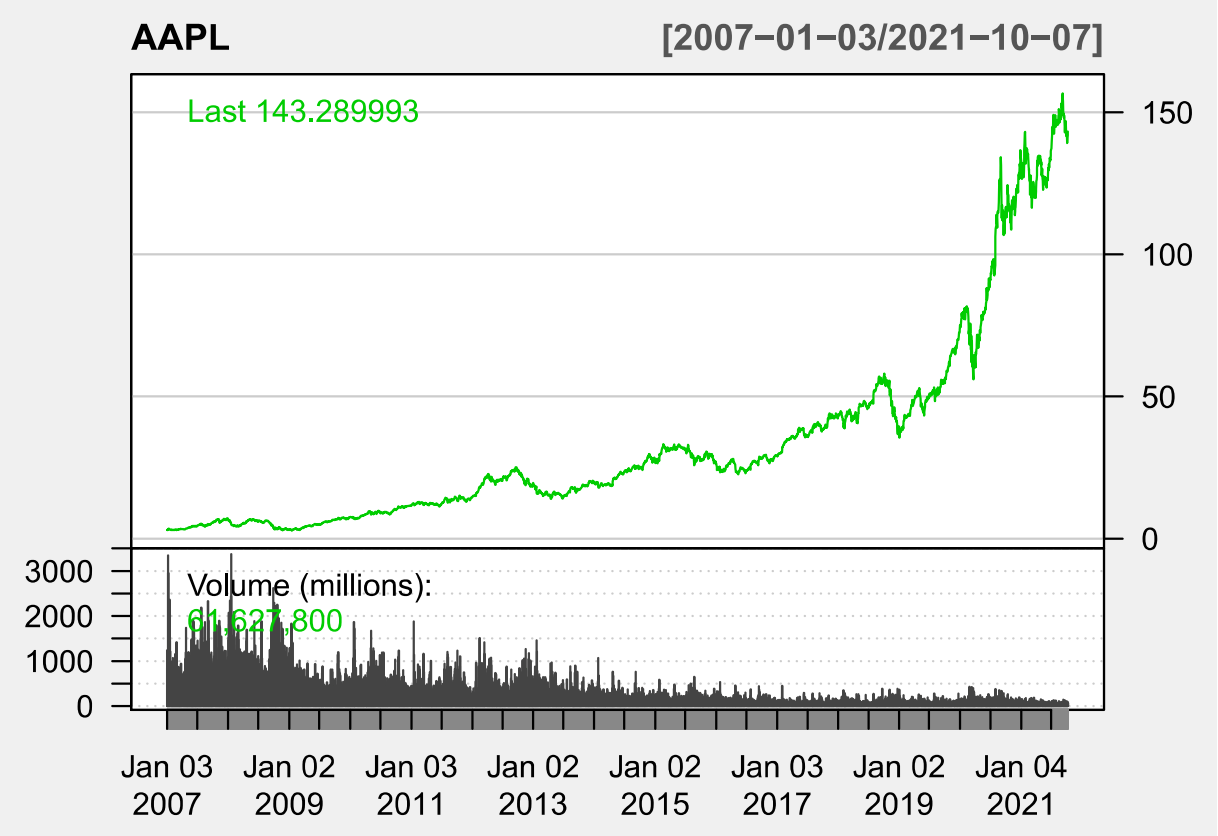
```
##            X name open close high  low  volume
## 1 1980-12-12 AAPL 0.51  0.51 0.52 0.51 2093900
## 2 1980-12-15 AAPL 0.49  0.49 0.49 0.49  785200
## 3 1980-12-16 AAPL 0.45  0.45 0.45 0.45  472000
## 4 1980-12-17 AAPL 0.46  0.46 0.46 0.46  385900
## 5 1980-12-18 AAPL 0.48  0.48 0.48 0.48  327900
## 6 1980-12-19 AAPL 0.50  0.50 0.51 0.50  217100
```

```
getSymbols("AAPL") #Gets Live data from Yahoo Finance
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "AAPL"
```

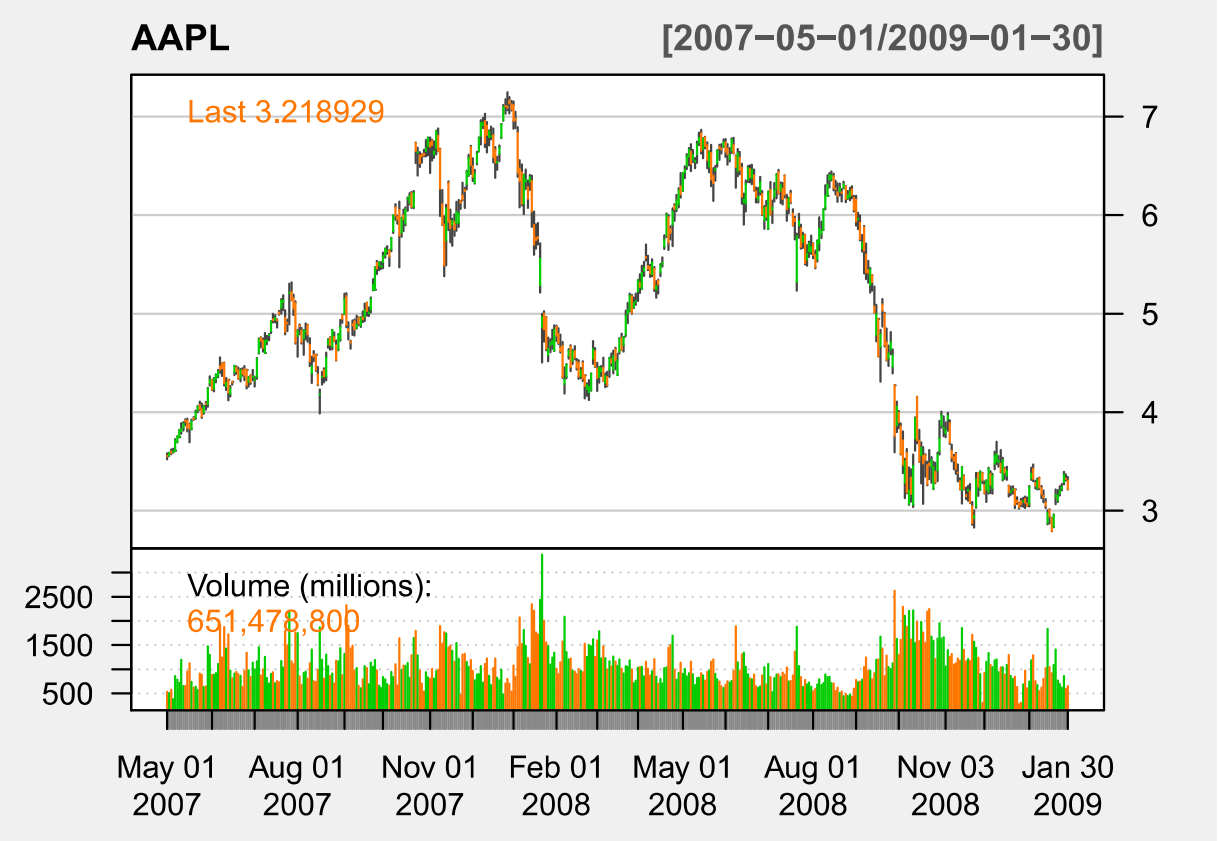
```
chartSeries(AAPL, type="line",subset=2007,theme=chartTheme(('white')))
```



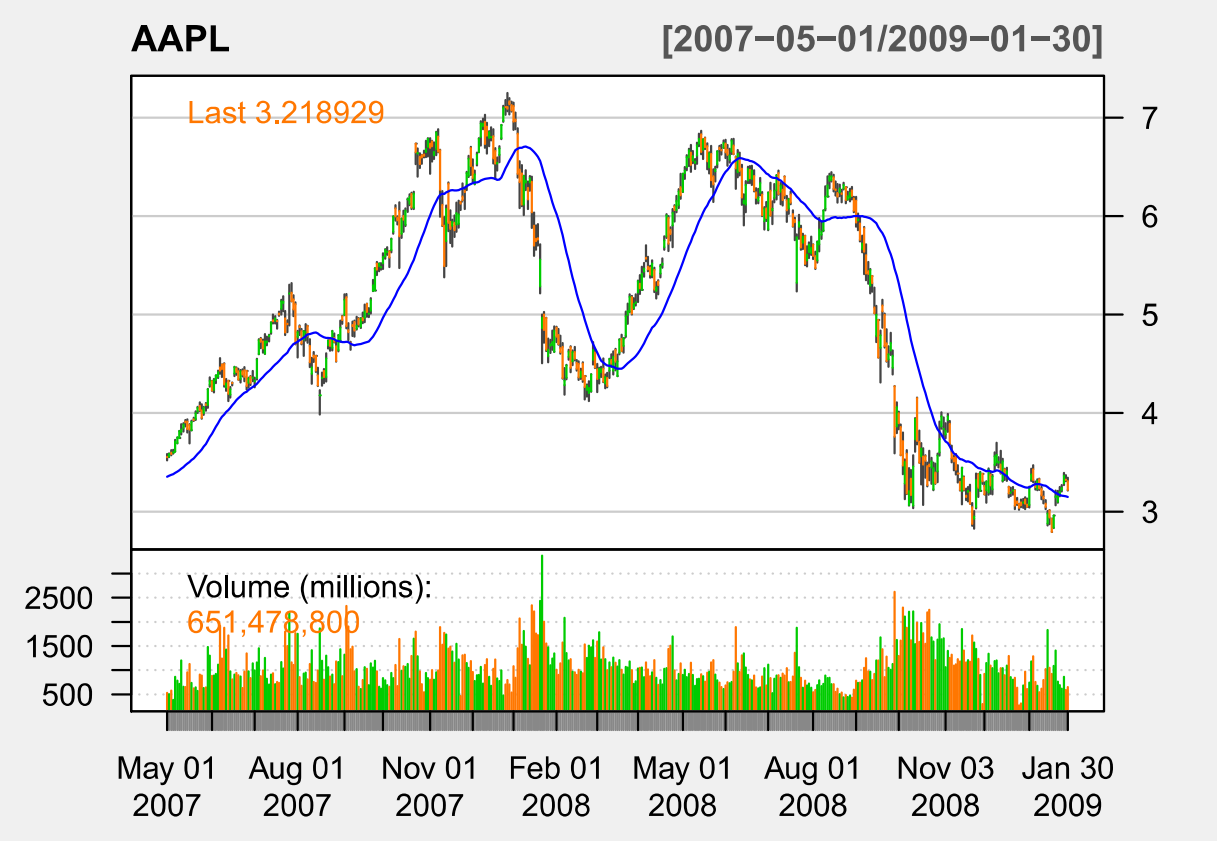
Technical Analysis

Simple Moving Average

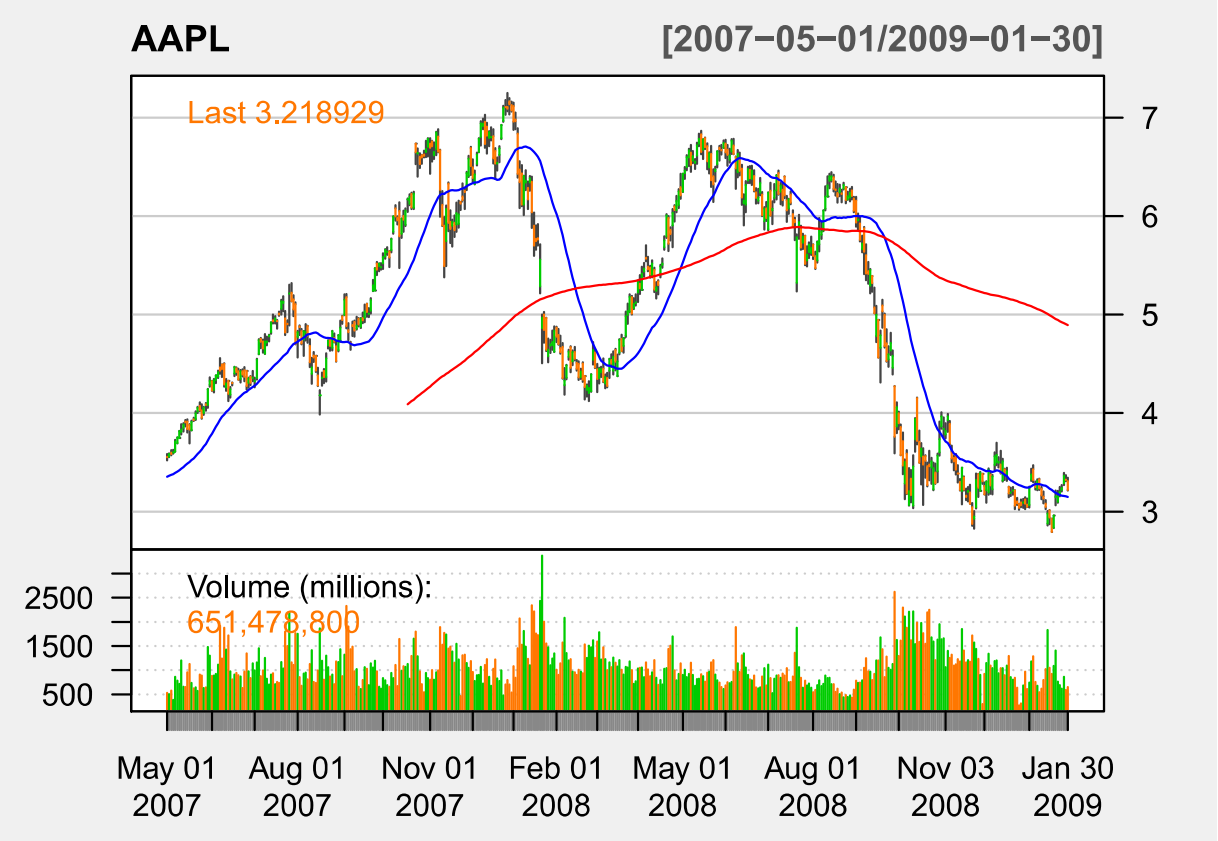
```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```




```
addSMA(n = 30,on =1, col="blue") #30 day period
```

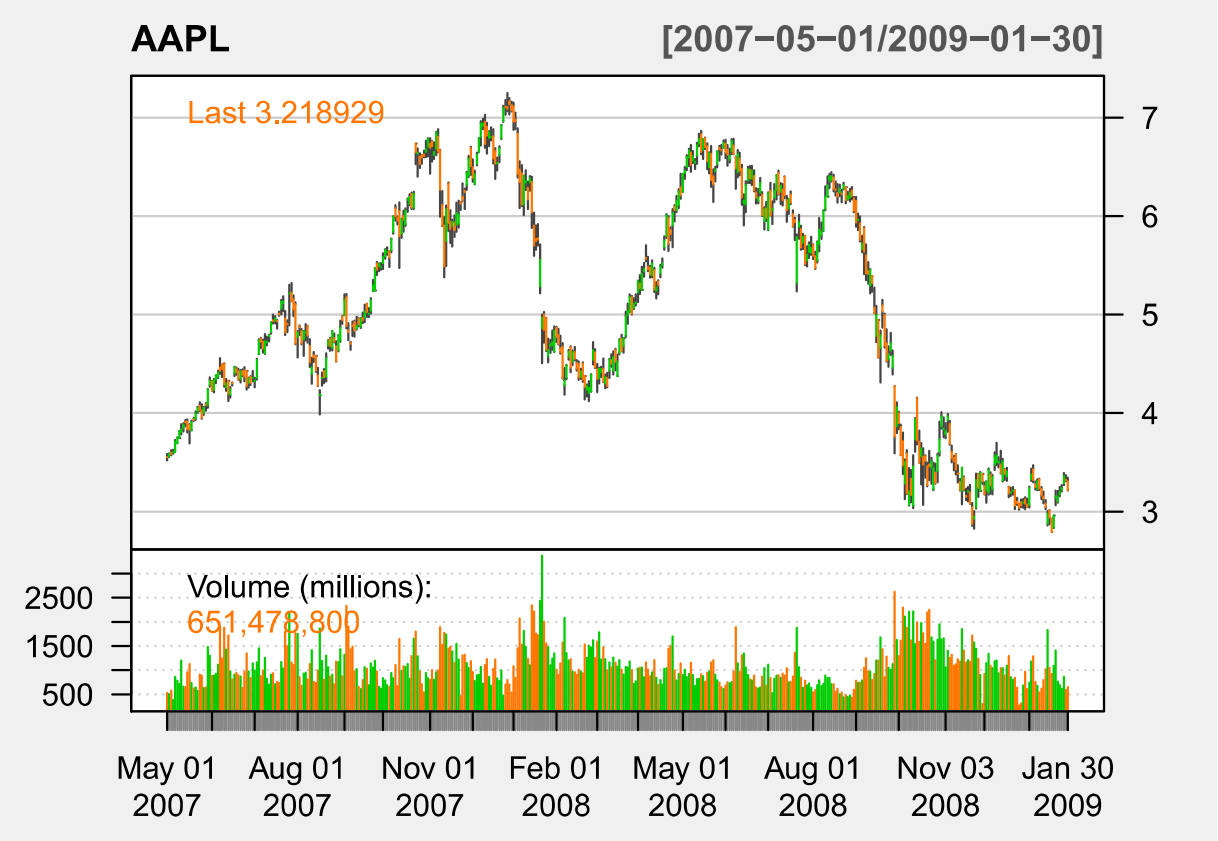



```
addSMA(n=200,on=1,col="red") #200 day period
```

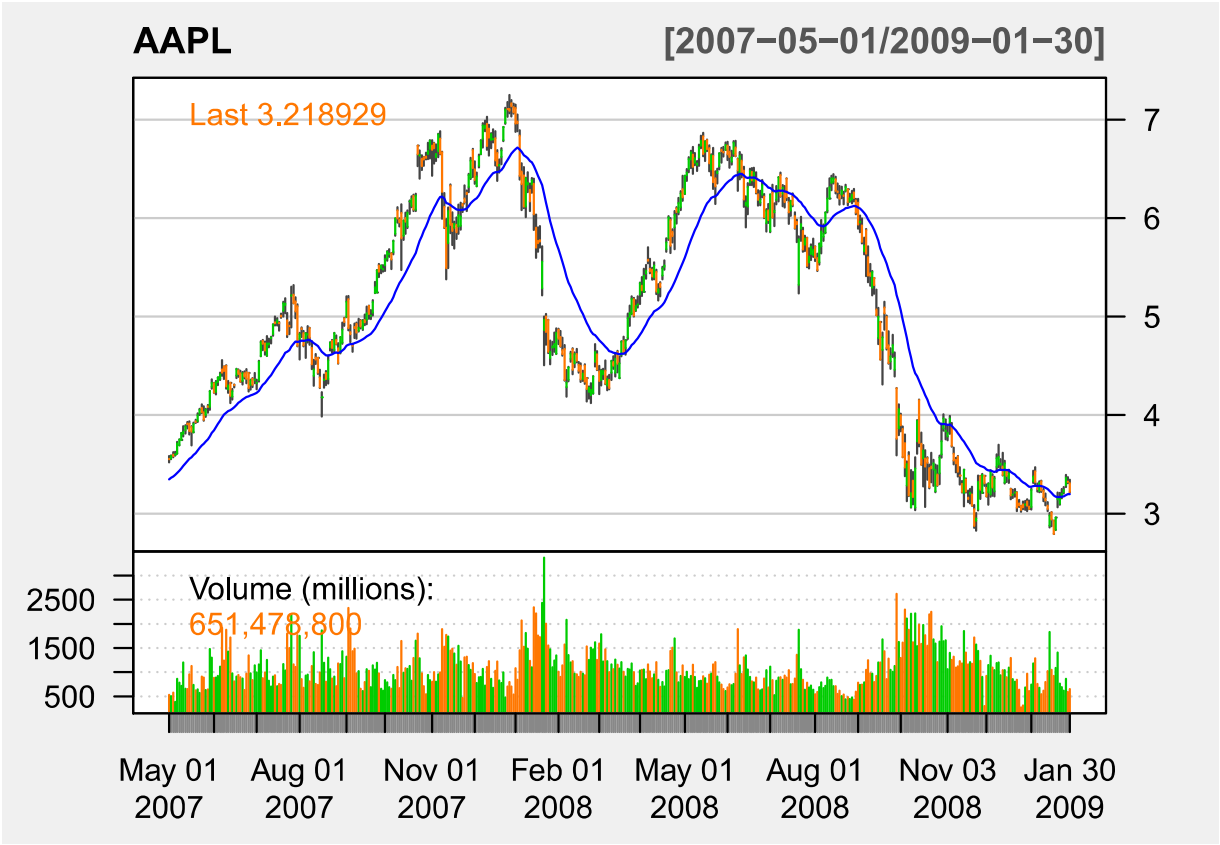


Exponenetial Moving Averages

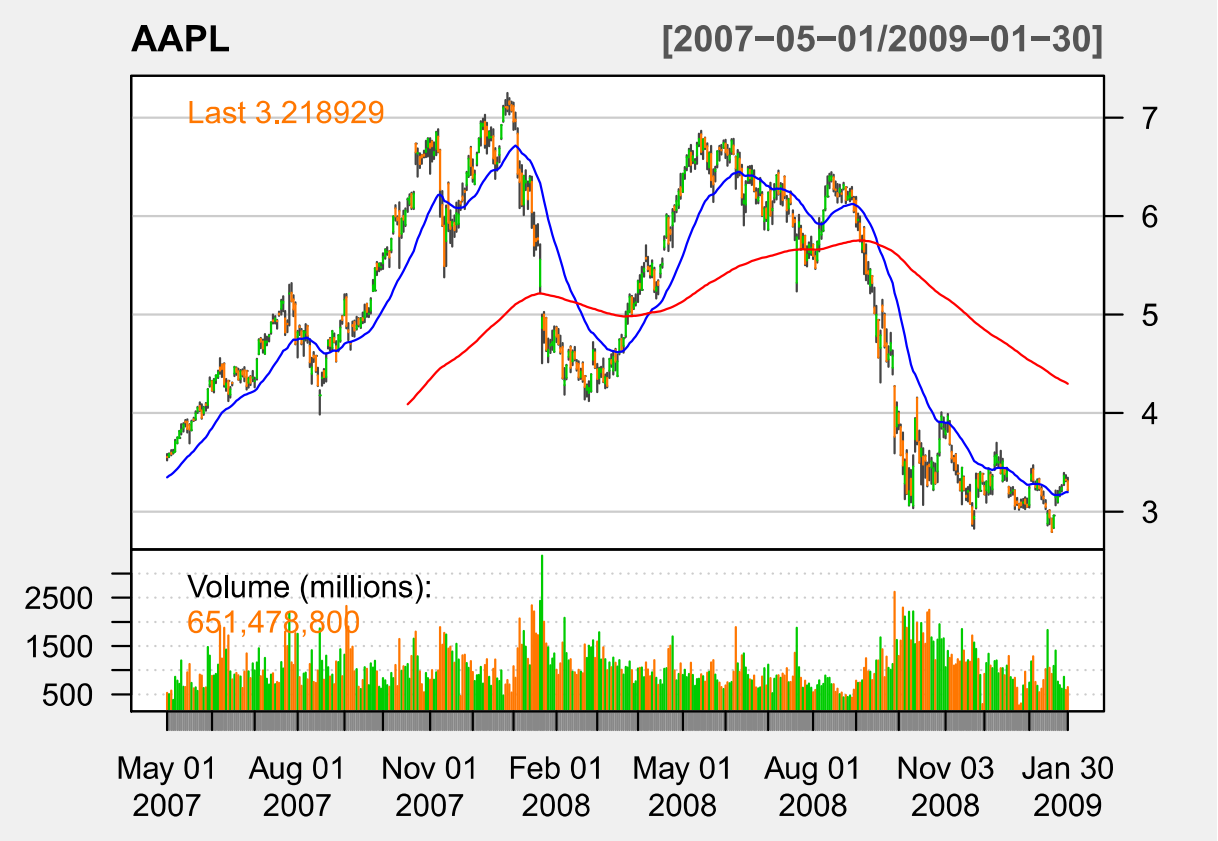
```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```




```
addEMA(n=30,on=1,col="blue")
```

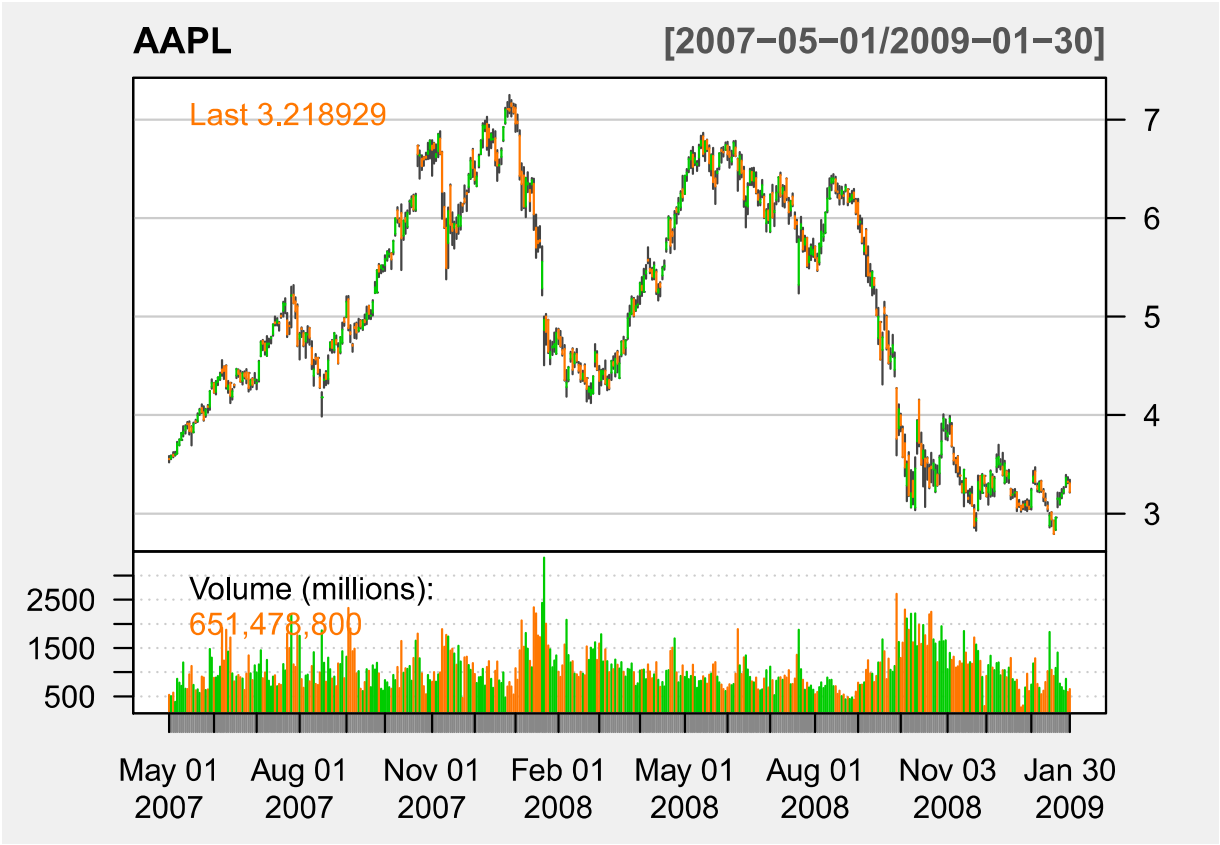



```
addEMA(n=200,on = 1,col="red")
```

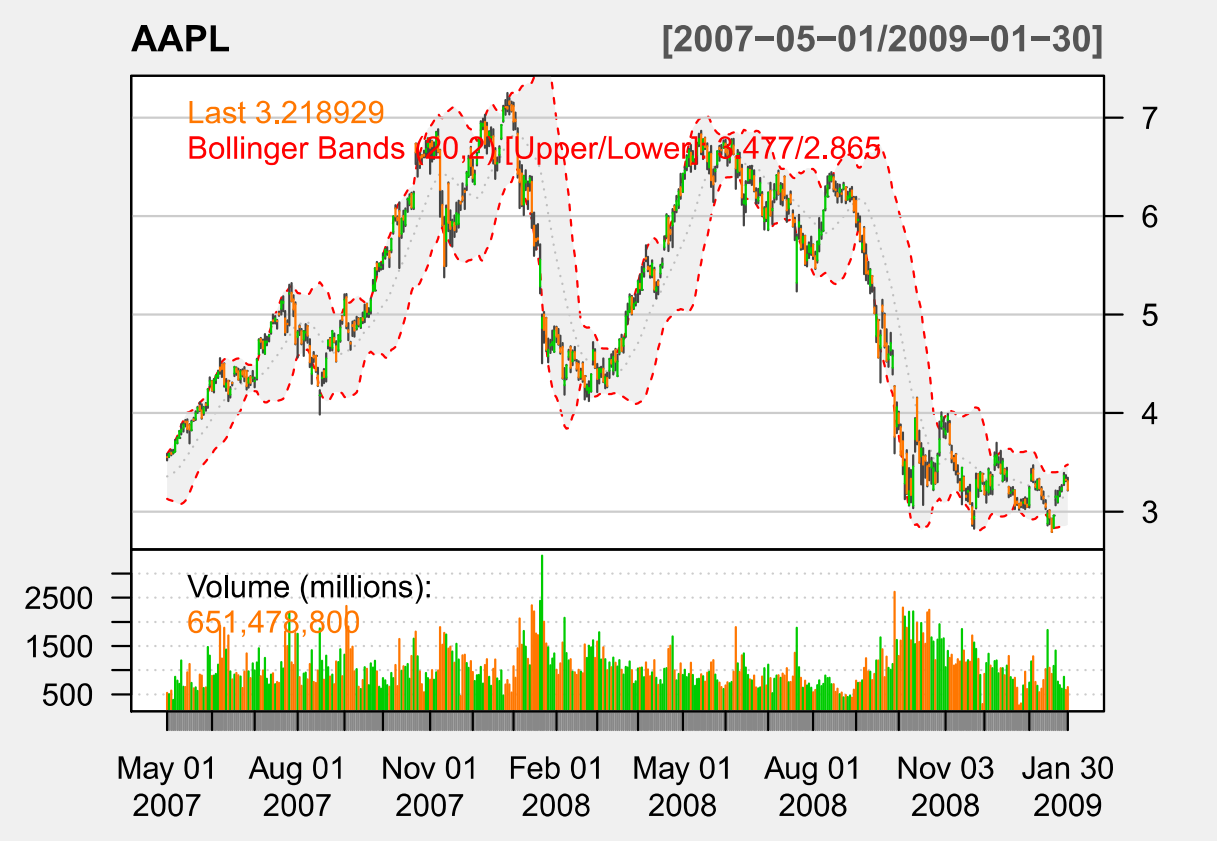


Bollinger Bands

```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```

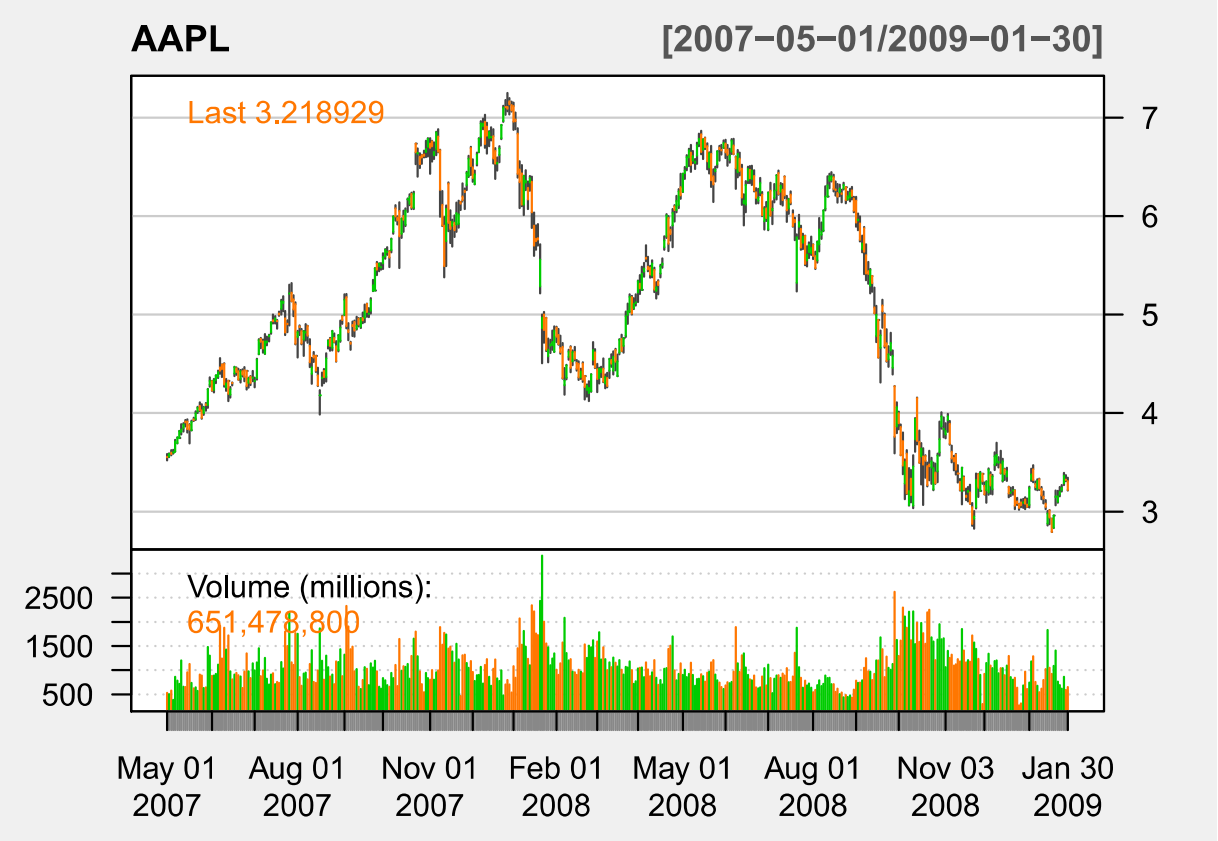



```
addBBands(n=20,sd=2) #Period 20 with Std dev of 2
```

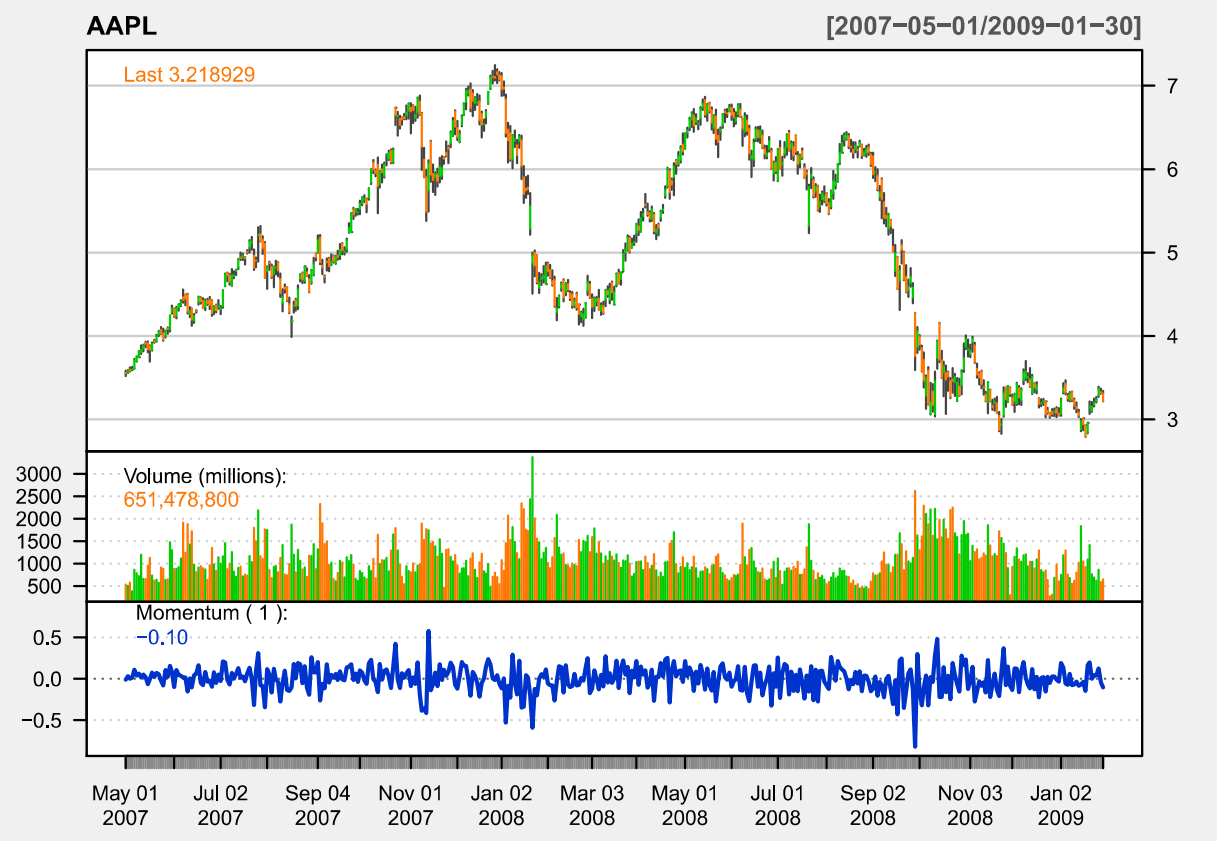


Momentum

```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```

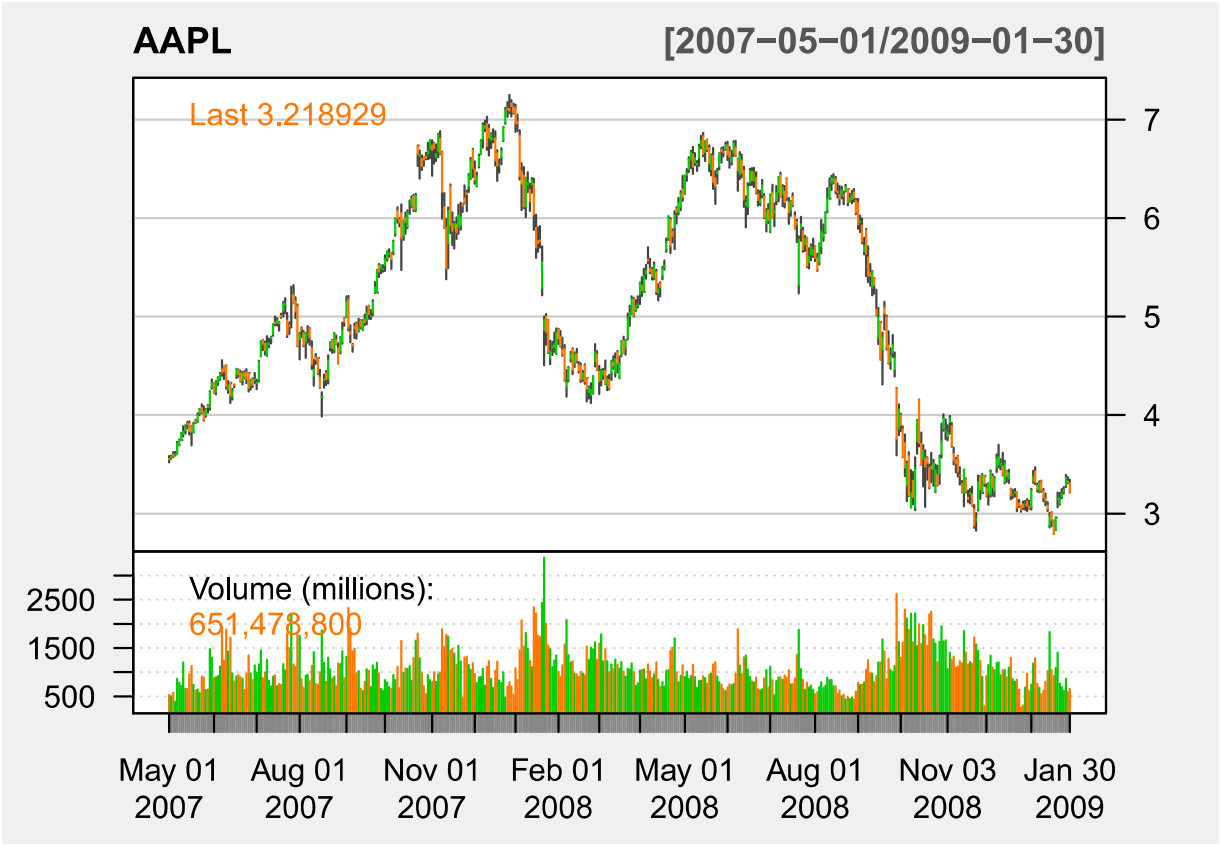



```
addMomentum(n=1)
```

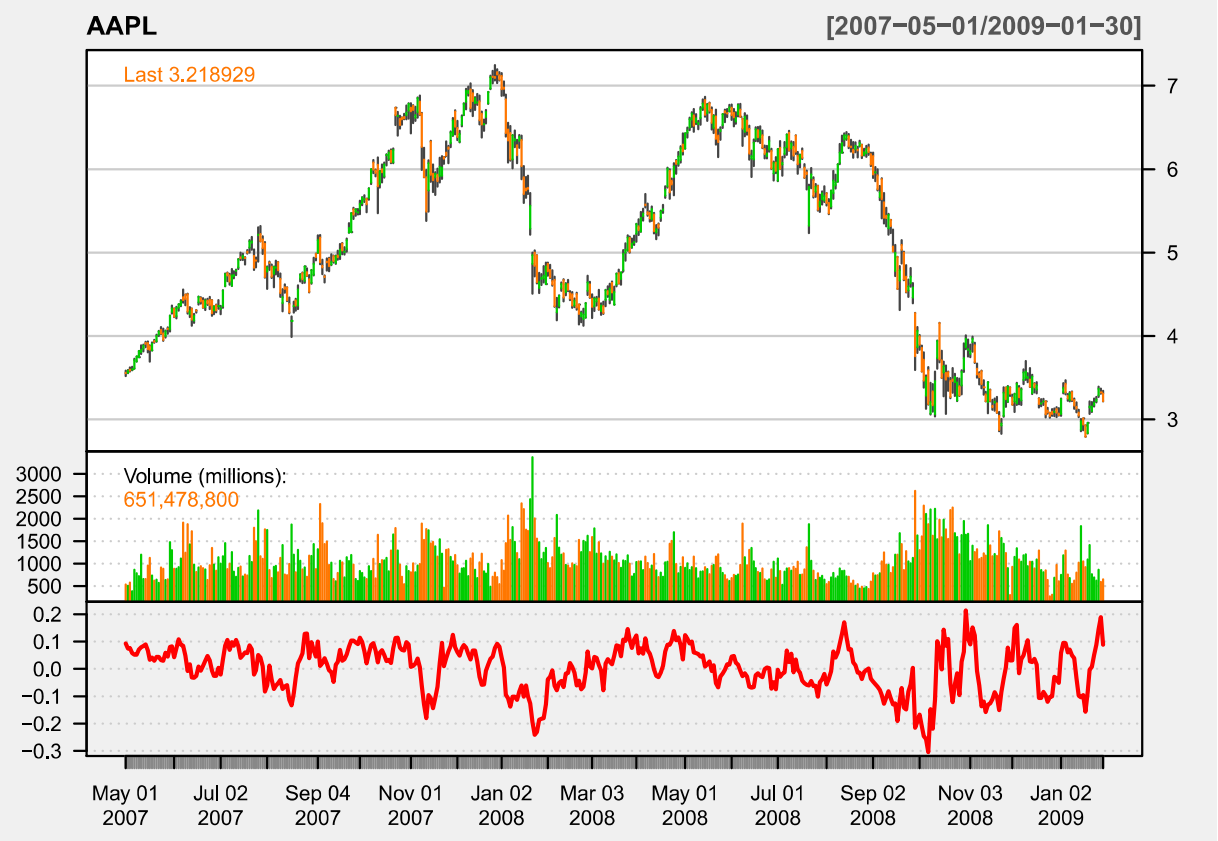


ROC

```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```

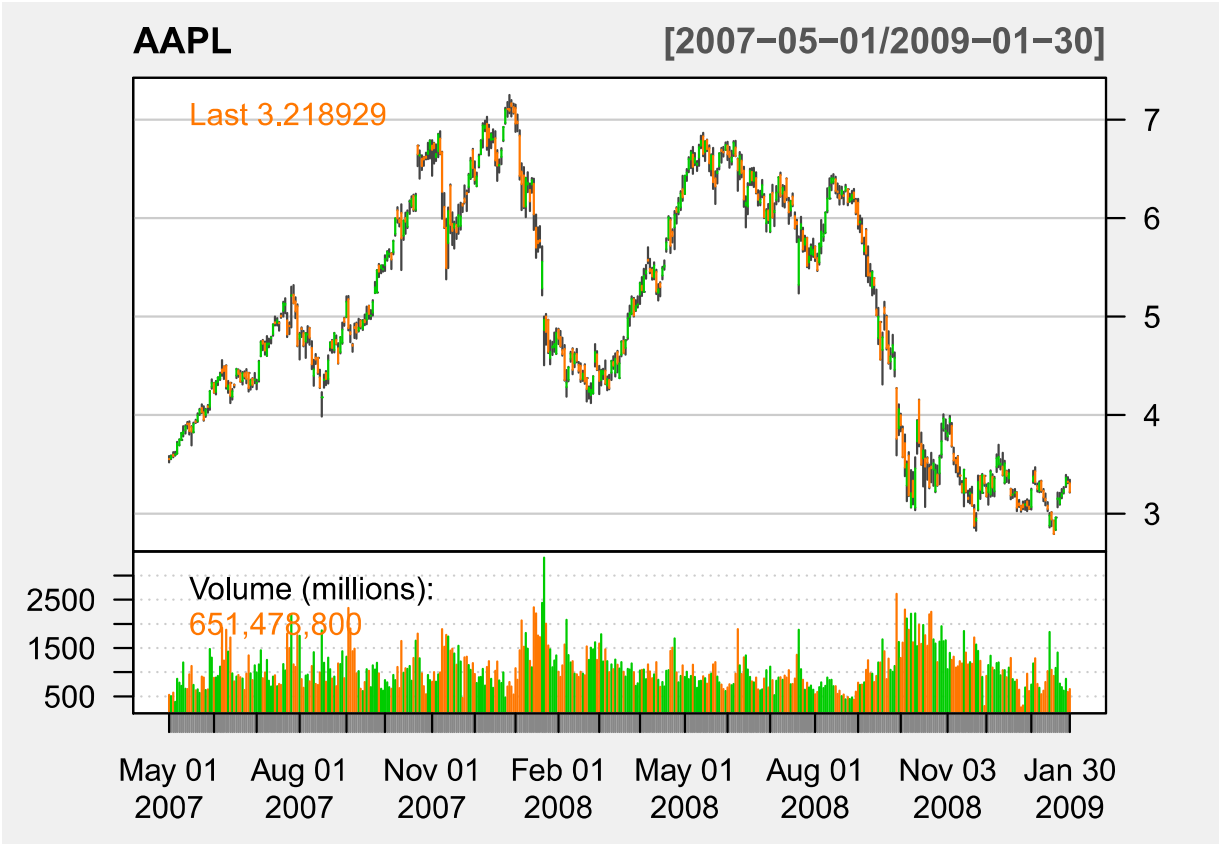

AAPL
[2007-05-01/2009-01-30]
Last 3.218929
7
6
5
4
3
Volume (millions):
651,478,800
2500
1500
500
May 01
2007
Aug 01
2007
Nov 01
2007
Feb 01
2008
May 01
2008
Aug 01
2008
Nov 03
2008
Jan 30
2009

```
addROC(n=7)
```

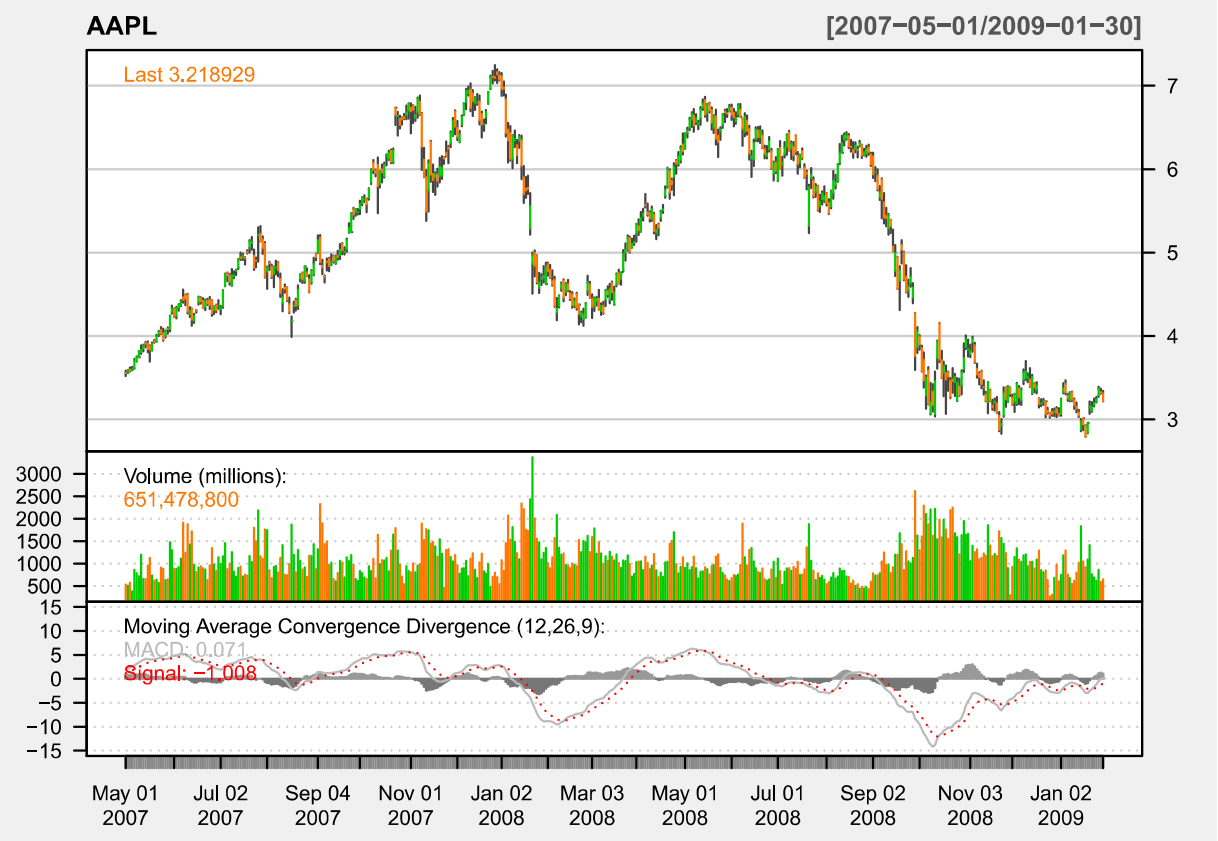


MACD (Moving Average Convergence Divergence)

```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```

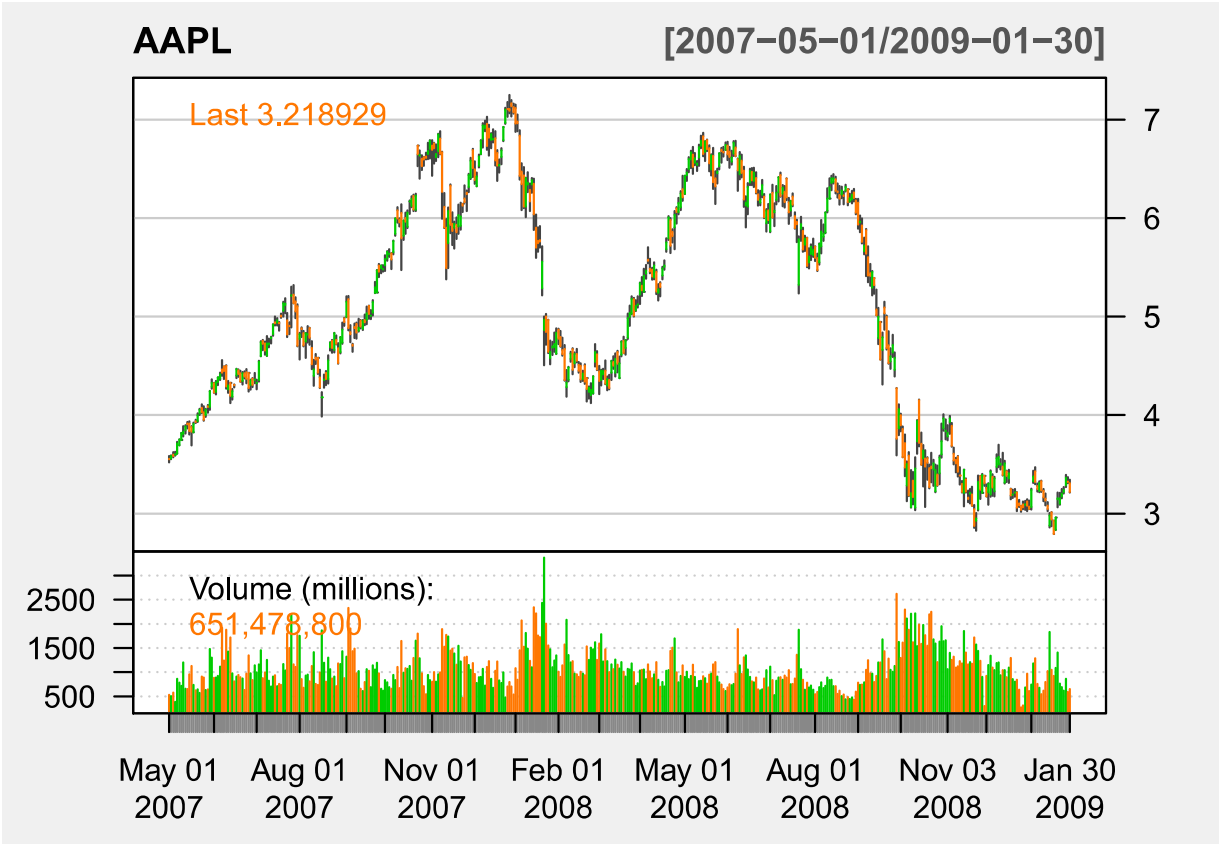



```
addMACD(fast=12,slow=26,signal=9,type="EMA")
```

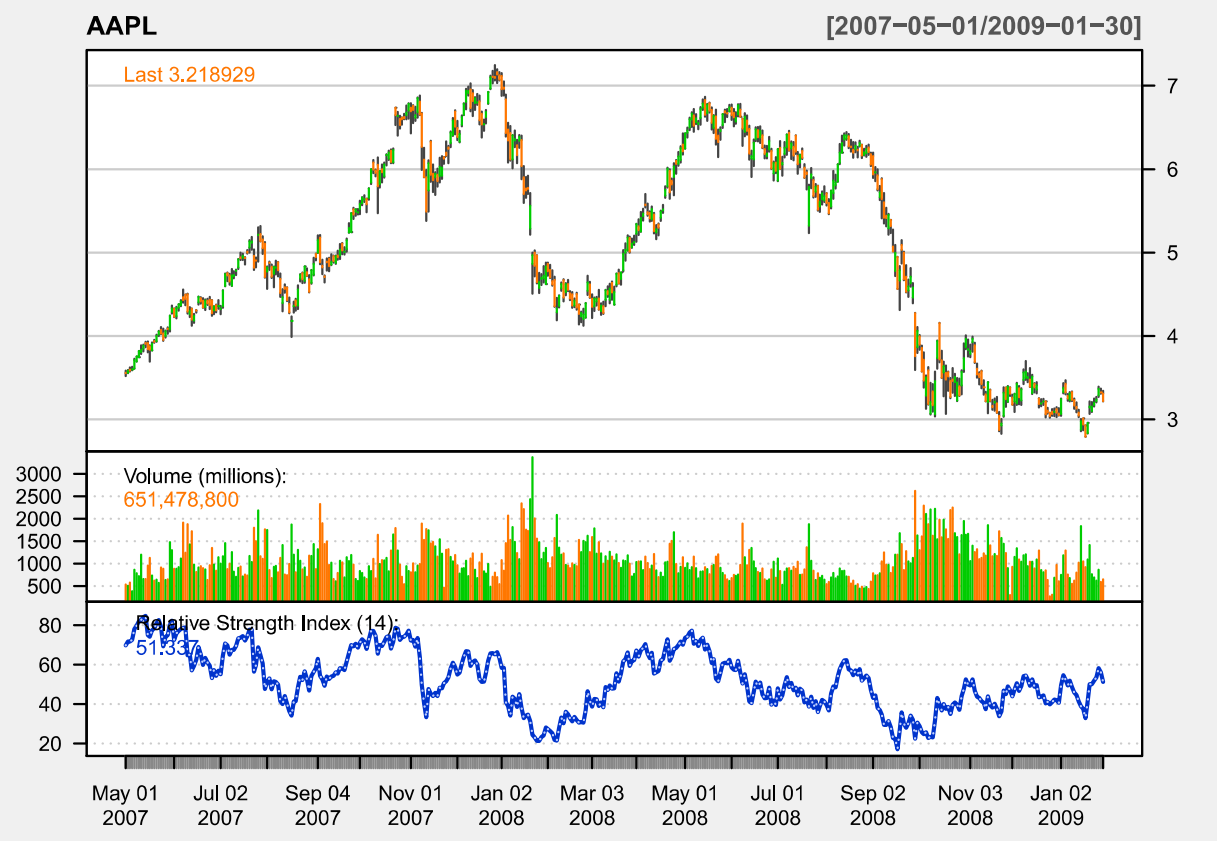


RSI (Realtive Strength Index)

```
chartSeries(AAPL, subset='2007-05::2009-01',theme = chartTheme('white'))
```




```
addRSI(n=14,maType="EMA")
```



Conclusion

Constructed a Simple LSTM based Time-Series forcasting model. Explained the AAPL Stock Closing prices in Financial Terms

---