Portfolio Performance Plotter

Nils Yannikos

23 Januar 2019

Introduction

I have tried out many stock apps on my smartphone, and quite a few of them have an option to track your portfolio, which allows you to calculate the absolute and percentage gain of each single stock and the whole stock portfolio. But what if you would like to see your portfolio performance over time? Did your trades benefit you or made you lose even more money than if you had just left the stocks alone after you bought them? What was the highest percentage gain that you had at any time since you bought the stocks? What will be your percentage gain if you keep buying and selling stocks over a certain duration? Unfortunately, I haven't seen a single app that can do the same thing and plot it over time (which doesn't mean that no such app exists), although it shouldn't be too difficult. Basically, you could record the gain and gain % of your app every day, put it into a .txt file and plot it, but this would be too tedious and you would have missing data if you forgot to record your values for one day. Thus, I wrote this script to play around with some basic R features and find my own solution to this problem. This is part of my own learning experience, while proceeding, I hope to find some more sophisticated and efficient procedures.

This script was written to plot the performance of a stock portfolio, accounting for all transactions (buy or sell) that are done over time. For example, several shares can be bought on different days, some sold a week later, some bought again etc. In this case, the decisions of the trader will affect the portfolio performance over time in the same way as a fund manager will affect the success of his portfolio through his decisions. Stock transactions of the portfolio can be provided as a tab delimited .txt file (described below) and the script will then automatically read how many different stocks there are, which stocks there are, and how many shares of which stock were bought or sold on what date etc. It will then download the historical prices and calculate the gain or loss of the portfolio as a whole. Let's get started and see what the output of the script looks like.

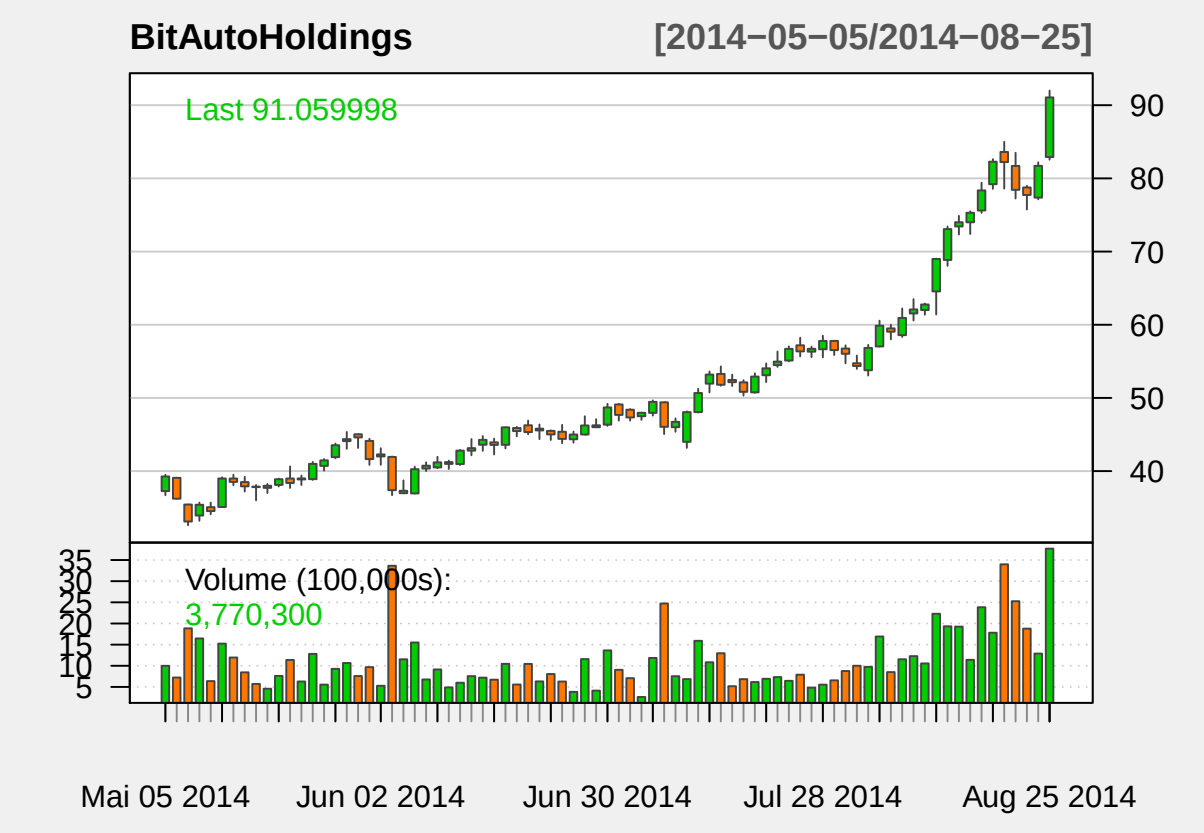
The script begins

Here we need to load the packages for gathering stock market data, for plotting and for the pipe.

```
library(quantmod) # download stock data from the internet
library(magrittr) # the pipe
library(tidyverse) # used for data wrangling, written by Hadley Wickham
library(zoo) # used for dealing with xts objects
```

Let's say you found the stock BitAuto Holdings in 2014 and witnessed it's crazy run. Let's take a look at what happened between the 5th of May and the 26th of August, 2014:

```
BitAutoHoldings<-(getSymbols("BITA",
                  from = "2014-05-05",
                  to = "2014-08-26",
                  src="yahoo",
                  auto.assign=FALSE))
chartSeries(BitAutoHoldings,theme = chartTheme("white"))
```



Let's imagine you decided to join the market and bought 289 BITA at $ 34.54 on May 5th, 2014. It kept going up, so you added another 159 BITA at $ 62.77 on August 4th, 2014. On August 26th, you got greedy and bought 216 more shares of BITA at $ 96.14, as the stock was clearly heading towards the sky. At the same time, because you didn't want to put all your eggs in one basket, you added 32 AMZN at $ 312.55 on May 5th, 2014.

The cost price of all BITA shares is $ 40,728.73, that of all AMZN shares is $ 10,001.60:

```
289*34.54 + 159*62.77 + 216*96.14 # BitAuto Holdings
```

```
## [1] 40728.73
```

```
32 * 312.55 # Amazon
```

```
## [1] 10001.6
```

Now we load the file "transactions.txt", which contains all the trades that we just talked about, such as stock symbol, transaction type (buy or sell), transaction date, share volume, cost price, and commission fee. We store the transactions in the object "transactions". Also, we define the start and end date of the stock data that we want to visualize:

```
transactions <- read.table(file.choose(),
                           dec=".",
                           header = T)
# Define start and end date for stock data
startdate <- "2014-05-05"
enddate <- "2019-01-24"
#enddate <- Sys.Date() # We can also use today as the end date
```

Let's take a look at the data file. We can see that the first two rows contain sell transactions with no volume or price. This is (for now) necessary for the script to work. Also, we can see how the text file needs to be set-up. Remember that the stock symbols need to be exactly identical to the stock symbols found on the US stock exchanges. Typos in the stock symbol column will prevent the "getSymbols" function from collecting stock market data.

```
transactions
```

```
##         date vol  price stocksymbol stockname buyorsell comm
## 1 2014-05-05   0   0.00        BITA   BitAuto      sell    0
## 2 2014-05-05   0   0.00        AMZN    Amazon      sell    0
## 3 2014-05-05  32 312.55        AMZN    Amazon       buy    0
## 4 2014-05-05 289  34.54        BITA   BitAuto       buy    0
## 5 2014-08-04 159  62.77        BITA   BitAuto       buy    0
## 6 2014-08-26 216  96.14        BITA   BitAuto       buy    0
```

Please note that in the current version of the script, for every stock type that was bought, at least one corresponding row with sales information is required in the input file. If no stock has been sold yet, then this row can be set to 0 and the same date as the date of purchase can be selected. For example, if AMZN was bought 10 times, one single row with "buyorsell" set to "sell" is enough, the price and volume can be set to 0. This necessity will be removed in a later version of the script. Furthermore, the commission fee has no effect on the output yet, but this will be also addressed in a future version of the script.

We see that the 4th column contains the stock symbols, not to be confused with the stock names. The stock symbol is needed to pull the data from Yahoo Finance. Depending on how many different stocks were traded, there are different levels of stock symbols in this column.

The next command will read how many different levels of stock symbols (fourth column) there are, and this number will be used for the following loop, which will conduct all the following steps for each individual stock symbol. This principle will be used several times in this script. We see that there are two different levels of stocksymbols (i.e., AMZN and BITA):

```
aN<-length(levels(transactions[,4]))
aN
```

```
## [1] 2
```

The loop will now read through the transactions object, check how many different stocks there are and begin downloading one stock at a time. Then another loop within the main loop will subset the "transactions" object to "buy" and create a vector for each buy transaction. The vector will have the same length as the data frame that was downloaded using getSymbols(). If the date of the newly created vector is before the date of transaction (=buy) from the "transactions" subset, then the value will be set to 0, otherwise the value will be set to the cost price of the stock, as set in the "transactions" subset. The same will be done with the stock volume that was purchased. If the date of the vector is before the transaction date, then the volume will be set to 0, otherwise to the volume set in the "transactions" subset. The inner loop will cycle through the whole "transactions" subset until every buy transaction has resulted in a price and volume vector. The total cost of all buy transactions so far will then can be calculated buy multiplying the tmp_cost vector by the tmp_vol vector and storing it in the tot_cost vector.

This tot_cost vector will grow with every iteration of the loop, until all the buy transactions have been addressed. The temporary loop vectors (tmp_cost and tmp_vol) will be merged with the respective stock data of the main loop. The newly created vectors in the merged data frame will be renamed by either "buyprice" or "buy_volume", followed by a number of the respective loop iteration (= number of the buy transaction). After the first loop ends, one "buyprice" and one "buy_volume" vector has been created for each corresponding buy transaction from the transactions subset. Now, the vectors "tot_cost" and "tot_buyvol", which have been growing with every iteration, are merged with the previously pulled stock data.

The second loop within the main loop does exactly the same as the previous sub-loop, but this time the transactions will be subset for "sell" transactions only.

Let's now run the big loop and get our data prepared.

```
# Loop 1 begin (main loop)

for(a in 1:aN){
loop_stocksym <- levels(transactions[,4])[a] # select stock symbol
tmpstock_data <- getSymbols(loop_stocksym, #pull stock data
```

```
                            from = startdate,
                            to = enddate,
                            src="yahoo",
                            auto.assign=FALSE) %>% .[,4]
stocknam <- filter(transactions,
                   #select stock name from transactions (5th col of transactions)
                   stocksymbol == loop_stocksym) %>%
                   select(stockname) %>%
                   .[1,] %>%
                   as.character()
sub_transactions<-transactions[transactions[,6]=="buy" & transactions[,5]==stocknam,]
# subset transactions to stockname and buy
tot_cost <- rep(0,nrow(tmpstock_data)); tot_cost
tot_buyvol <- rep(0,nrow(tmpstock_data)); tot_buyvol
# create empty vectors to be filled during the loop
# vectors have the same length as the data that was pulled from Yahoo Finance


# Loop 1.1 begin (1st loop within main loop)

for(i in 1:nrow(sub_transactions)){ # run the loop for every buy transaction (nrow)
tmp_cost <- ifelse(index(tmpstock_data) < as.Date(sub_transactions[i,1]),
                    0,
                    sub_transactions[i,3])
# if date is before purchase, then price = 0. If date is after purchase,
# then price = transaction cost price (3rd col of sub_transactions)

tmp_vol <- ifelse(index(tmpstock_data) < as.Date(sub_transactions[i,1]),
                  0,
                  sub_transactions[i,2])
# if date is before purchase, then volume = 0. If date is after purchase,
# then volume = transaction volume (2nd col of sub_transactions)

tot_cost <- tot_cost + tmp_cost * tmp_vol; tot_cost
# add price * vol to the empty vector that was defined before this loop.
# Value increases with every iteration of this loop (i.e., with every buy of same stock)

tot_buyvol <- tot_buyvol + tmp_vol; tot_buyvol
# add vol to the empty vector that was defined before this loop.
# Value increases with every iteration of this loop (i.e., with every buy of same stock)

tmpstock_data<-merge(tmpstock_data, tmp_cost, tmp_vol)
# merge those dataframes to the pulled stock data

colnames(tmpstock_data)[colnames(tmpstock_data)=="tmp_cost"] <- paste("buyprice",
    i, sep = "")
# rename those colnames of tmp_cost by "buyprice" and the corresponding number

colnames(tmpstock_data)[colnames(tmpstock_data)=="tmp_vol"] <- paste("buy_volume",
    i, sep = "")
# same as above, rename by "buy_volume" and the corresponding number
} # Loop 1.1 end

tmpstock_data <- merge(tmpstock_data, tot_cost, tot_buyvol)
# merge the created data frames with the stock data from the main loop

# Now we do exactly the same, but with sell data in the next loop.
tot_sellval <- rep(0,nrow(tmpstock_data)); tot_sellval
```

```
tot_sellvol <- rep(0,nrow(tmpstock_data)); tot_sellvol

sub_transactions<-transactions[transactions[,6]=="sell" & transactions[,5]==stocknam,]
# subset transactions to "buyorsell" (6th col) = sell"

# Loop 1.2 begin (2nd loop in main loop)

for(k in 1:nrow(sub_transactions)){
tmp_cost <- ifelse(index(tmpstock_data) < as.Date(sub_transactions[k,1]),
                   0, sub_transactions[k,3])

tmp_vol <- ifelse(index(tmpstock_data) < as.Date(sub_transactions[k,1]),
                  0, sub_transactions[k,2])

tot_sellval <- tot_sellval + tmp_cost * tmp_vol; tot_sellval
tot_sellvol <- tot_sellvol + tmp_vol; tot_sellvol
tmpstock_data<-merge(tmpstock_data, tmp_cost, tmp_vol)

colnames(tmpstock_data)[colnames(tmpstock_data)=="tmp_cost"] <- paste("sellprice",
    k, sep = "")

colnames(tmpstock_data)[colnames(tmpstock_data)=="tmp_vol"] <- paste("sell_volume",
    k, sep = "")

} # Loop 1.2 end

tmpstock_data <- merge(tmpstock_data, tot_sellval, tot_sellvol)
# merge the created data frames with the stock data from the main loop

tmpstock_data$vol_owned <- tmpstock_data$tot_buyvol - tmpstock_data$tot_sellvol
# volume of stock shares currently owned = bought shares - sold shares
tmpstock_data$val_owned <- tmpstock_data$vol_owned * tmpstock_data[,1]
# value owned = volume * price
tmpstock_data$sum_eq <- tmpstock_data$val_owned + tmpstock_data$tot_sellval
# the sum of our equity is owned stock + value of sales
tmpstock_data$gain <- tmpstock_data$val_owned + tmpstock_data$tot_sellval -
  tmpstock_data$tot_cost
# our gain is value owned shares + value sold shares - total cost of shares

tmpstock_data$gain_perc <- (tmpstock_data$val_owned + tmpstock_data$tot_sellval -
  tmpstock_data$tot_cost) / tmpstock_data$tot_cost * 100
# our gain % is (value owned shares + value sold shares -
# total cost of shares) / total cost of shares

assign(paste(stocknam, sep = ""), tmpstock_data)
# save the data as an object with the name of the stock (stocknam)
} # Loop 1 end
```

So now we have been manipulating a lot of data. Let's take a look at the object that was created under the name "BitAuto". We select the first and the 400th row of the data: (Note: This line of code needs to be removed if the script is to run with other stock data, because the object "BitAuto" might not be created by the script and might thus cause an error).

```
BitAuto[c(1,400),]
```

```
##            BITA.Close buyprice1 buy_volume1 buyprice2 buy_volume2
## 2014-05-05      39.27     34.54         289      0.00           0
## 2015-12-02      25.86     34.54         289     62.77         159
##            buyprice3 buy_volume3 tot_cost tot_buyvol sellprice1
```

```
## 2014-05-05      0.00           0  9982.06        289          0
## 2015-12-02     96.14         216 40728.73        664          0
##            sell_volume1 tot_sellval tot_sellvol vol_owned val_owned
## 2014-05-05            0           0           0       289  11349.03
## 2015-12-02            0           0           0       664  17171.04
##              sum_eq      gain gain_perc
## 2014-05-05 11349.03   1366.97  13.69427
## 2015-12-02 17171.04 -23557.69 -57.84047
```

We can see how the first sub loop created many vecotrs called "buyprice" followed by a number of the loop's iteration. We see that on the first date of our snapshot (2014-05-05), "buyprice2" and "buy_volume2" are still set to 0, whereas on the second date (2015-12-02) they contain values. This is because the respective transaction took place after 2014-05-05.
We can also see that on the first day after buying BITA (2014-05-05), the gain was $ 1366.97, that's 13.7% up. But only 1.5 years later (2015-12-02), we lost $ 23557.69 of our initial investment, so what's left is 17171.04 dollars, which comes to a loss of 57.8%.

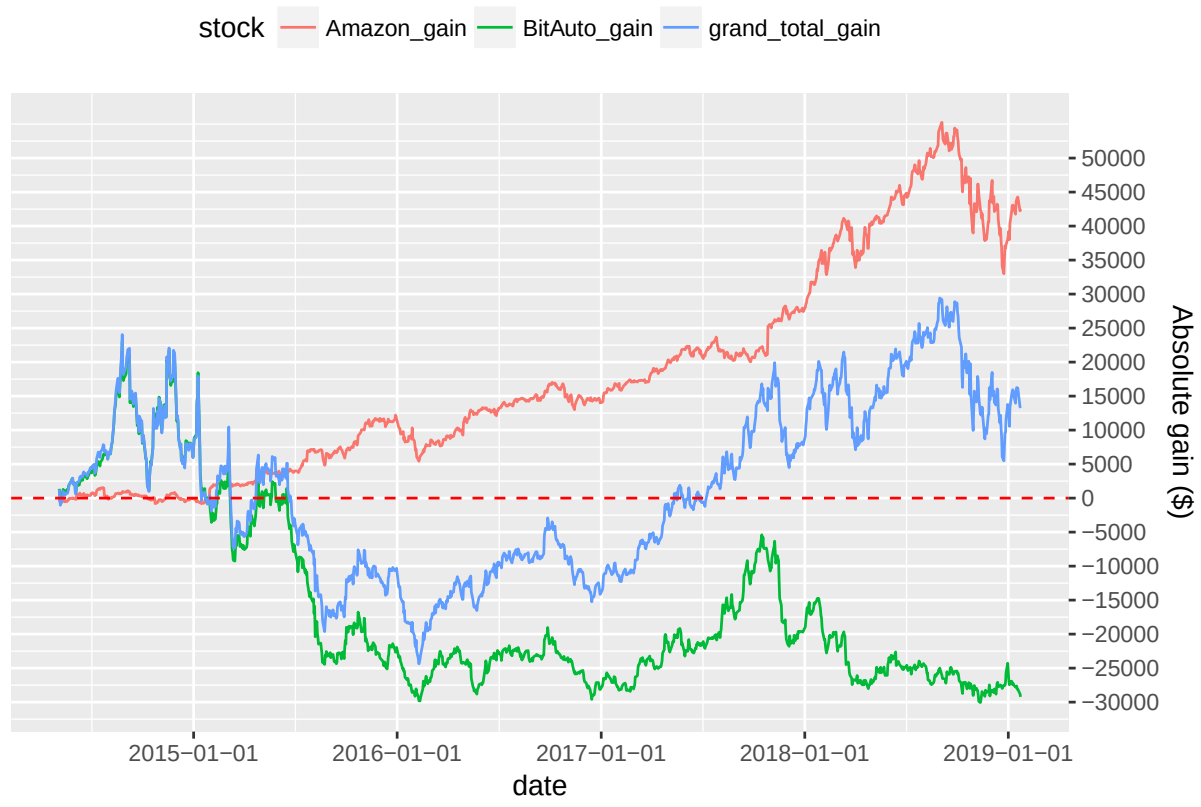
Now we have an object for every stock type that we traded, no matter, how often we traded. Now we want to put all gains and total costs together in one data frame, which we call "merg_df" (merged data frame). Within every iteration of this next loop, we calculate the gain % for the currently selected stock and add it to the data frame. Then we rename the three columns to the stockname + either "gain", "tot_cost", or "gain_perc". Note: in order to create an empty data frame with the length of our previously created vectors, we simply take the "gain" column (and automatically the index from the xts object) and delete it again after the loop has merged the calculated vectors to it. I know this is ugly, but it works until I find a better solution.

```
merg_df <- get(levels(transactions[,5])[1])$gain
names(merg_df)[1] <- "to_be_deleted" # this way we get a vector of the correct length

bN <- length(levels(transactions[,5]))
for(c in 1:bN){
  tmp_merg_df <- merge(get(levels(transactions[,5])[c])$gain,
                       get(levels(transactions[,5])[c])$tot_cost)
  # merge gain and tot_cost into temp dataframe
  tmp_merg_df$gain_perc <- tmp_merg_df$gain / tmp_merg_df$tot_cost * 100
  # calculate percent gain in temp dataframe
  names(tmp_merg_df)[1] <- paste(as.character(levels(transactions[,5])[c]),
                                 "gain", sep = "_")
  names(tmp_merg_df)[2] <- paste(as.character(levels(transactions[,5])[c]),
                                 "tot_cost", sep = "_")
  names(tmp_merg_df)[3] <- paste(as.character(levels(transactions[,5])[c]),
                                 "gain_perc", sep = "_")
  merg_df <- merge(tmp_merg_df,merg_df)
}

merg_df<-merg_df[,1:ncol(merg_df)-1]
# now delete the last column (I know this is ugly, but for now it has to suffice)

ncol_merg_df <- ncol(merg_df) # how many cols are in merg_df?

merg_df$grand_total_gain <- rowSums(merg_df[,seq(1,ncol_merg_df,3)],na.rm = TRUE)
merg_df$grand_tot_cost <- rowSums(merg_df[,seq(2,ncol_merg_df,3)])
merg_df$grand_total_gain_perc <- rowSums(merg_df[,seq(1,ncol_merg_df,3)],na.rm = TRUE)/
  rowSums(merg_df[,seq(2,ncol_merg_df,3)],na.rm = TRUE) * 100
# calculate grand total gain = gain of all stocks.
# same principle for grand total cost and gain percentage.

merg_df_noxts <- data.frame(date=index(merg_df), coredata(merg_df))
ncol_merg_df_noxts <- ncol(merg_df_noxts)
```

```
# we need to turn xts object into non-xts object, so ggplot can read it

gain_df <-  select(merg_df_noxts,
                   names(merg_df_noxts[,c(1,seq(2,ncol_merg_df_noxts,3))]))

gain_perc_df <-  select(merg_df_noxts,
                        names(merg_df_noxts[,c(1,seq(4,ncol_merg_df_noxts,3))]))
```

Now it's finally time to look at the data. First we need to gather the different stock gains from our "merg_df_noxts" dataframe, so we can plot them.

```
g1 <- gather(gain_df, key = "stock", value = "gain",-date, na.rm = FALSE)
#I have to exclude the date column, else it will be gathered too.


ggplot() +
  geom_line(data = g1, mapping = aes(x = date, y = gain, color = stock), size = 0.5) +
  theme(legend.position="top") +
  ylab("Absolute gain ($)") +
  scale_y_continuous(breaks = seq(-30000,50000,5000), position = "right") +
  geom_hline(yintercept=0, linetype="dashed", color = "red") +
  scale_x_date(date_breaks = "1 years")
```




```
merg_df_noxts[merg_df_noxts[,1]=="2019-01-23",]
```

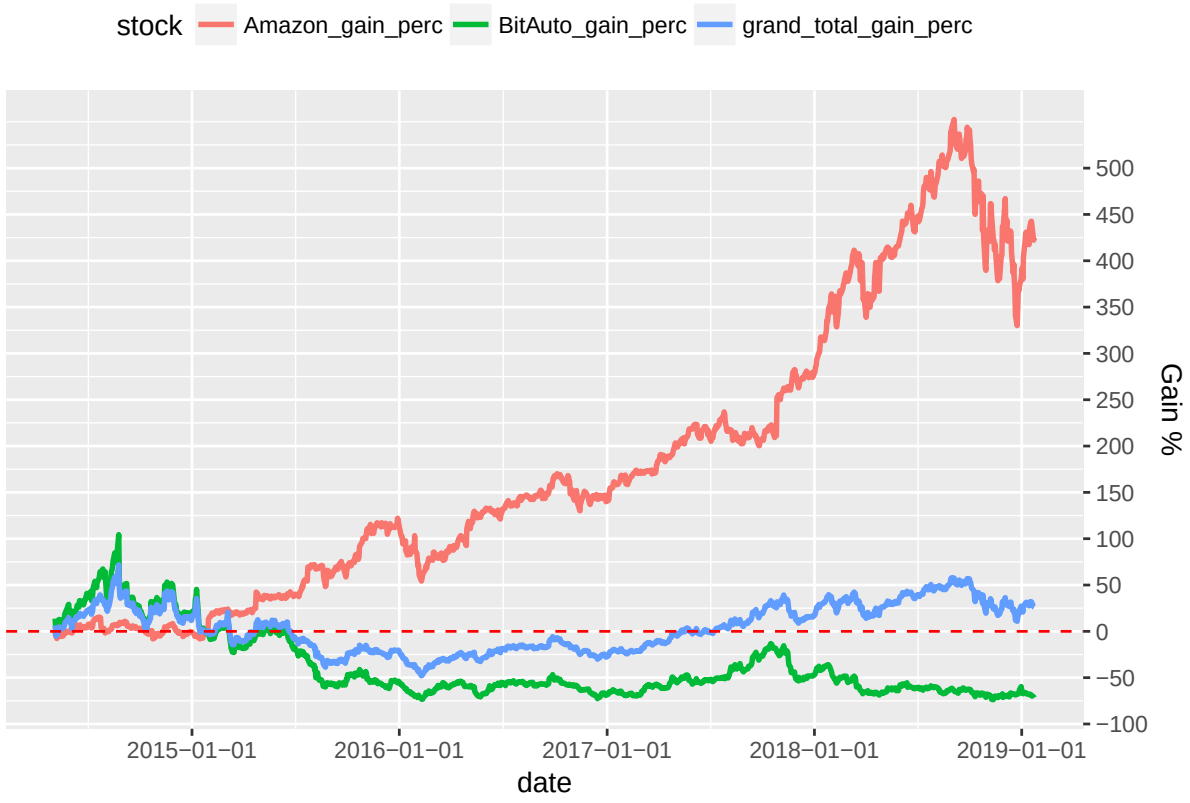
```
##            date BitAuto_gain BitAuto_tot_cost BitAuto_gain_perc
## 1189 2019-01-23    -29221.61         40728.73         -71.74692
##      Amazon_gain Amazon_tot_cost Amazon_gain_perc grand_total_gain
## 1189    42479.04         10001.6         424.7225         13257.43
##      grand_tot_cost grand_total_gain_perc
## 1189       50730.33              26.13314
```

```
# this is the dataset before we shortened it for plotting
```

We can see from the table and the graph that we lost $ 29,221.61 on BitAuto, but we made $ 42,479.04 on Amazon. This comes to a total gain of $ 13,257.43 (= $ 42,479.04 - $ 29,221.61). But considering that we invested approx. 4 times as much money in BitAuto, this is a huge loss. On the other hand, only $ 10,001.60 of Amazon yielded $ 42,479.04, which is a gain of 424.7%. But the absolute gain should always be seen in relation to the invested amount.

Let's take a look at the percentage gain of all stocks and the whole portfolio over time:

```
g2 <- gather(gain_perc_df, key = "stock", value = "gain_percent",-date)
#I have to exclude the date column, else it will be gathered too.


ggplot() +
  geom_line(data = g2, mapping = aes(x = date, y = gain_percent, color = stock), size = 1) +
  theme(legend.position="top") +
  ylab("Gain %") +
  scale_y_continuous(breaks = seq(-100,500,50),position = "right") +
  geom_hline(yintercept=0, linetype="dashed", color = "red") +
  scale_x_date(date_breaks = "1 years")
```




```
gain_perc_df[gain_perc_df[,1]=="2019-01-23",]
```

```
##            date BitAuto_gain_perc Amazon_gain_perc grand_total_gain_perc
## 1189 2019-01-23         -71.74692         424.7225              26.13314
```

We see that in the first year (2014), our portfolio gain % is mostly following BitAuto, because AMZN is not going anywhere. But in 2015, as BitAuto keeps losing money and Amazon is aiming at becoming the most valuable company in the world, we see that the losses of BitAuto have a larger impact on our portfolio gain % than Amazon, because we invested approx. 4 x as much in BitAuto than in Amazon. It is really interesting to see that because of the 71.7% loss of BitAuto, the portfolio only gained 26.1%, despite the huge gain of Amazon of 424.7%. We could now use the script to find out what would have

happened if we had invested only $ 10,000 in Amazon in the year of 2000. But before you do this, please make sure you have a box of Kleenex at hand unless you actually did invest in Amazon in 2000.