Statistical Methods in Finance Final Project

May 09, 2022

This document is for calculation purposes only and does not represent the final analysis.

Data Processing

```
require(tidyverse)
require(tidyquant)
require(xts)

require(MASS) # for fitdistr() and kde2d() functions
require(copula) # for copula functions
require(fGarch) # for standardized t density

require(ggplot2)
```

```
FILENAME = "./data/portfolio_historical_data.csv"
DOWNLOAD.DATA = !file.exists(FILENAME)
```

```
if(DOWNLOAD.DATA) {
    # Asset symbols that will be used for this analysis
    asset.symbols = c(
        "AMD", "MSFT", "SBUX", "AAPL",
        "ITUB", "FB", "NVDA", "F",
        "BAC", "T", "XOM", "VALE"
    )

    # Download the assets' hisotrical data and load the
    # variables to the environment
    historical.data = getSymbols(
        asset.symbols,
        src = "yahoo",
        from = "2017-04-05",
        to = "2022-05-05",
        periodicity = "monthly"
    )

    # This is a helper function that fetches an asset's
    # data from the environment variabels by using its
    # symbol
    adjusted.price.by.symbol = function(symbol) {
        Ad(get(symbol))
    }
```

```
    # Extract the adjusted price for each asset by
    # using its symbol
    adj.returns = asset.symbols %>%
        map(adjusted.price.by.symbol) %>%
        reduce(cbind)
    names(adj.returns) = map_chr(asset.symbols, ~ paste0(.x, ".adj"))

    # Calculate the net return for each asset
    net.returns = (diff(adj.returns) / stats::lag(adj.returns) * 100)
    names(net.returns) = map_chr(asset.symbols, ~ paste0(.x, ".net"))

    # Combine the ajusted prices and net returns for each
    # asset and save it into a tibble
    historical.data = cbind(adj.returns, net.returns)[-1, ] %>%
        as_tibble(rownames = "Date")

    # Save the net returns into a csv file
    write_csv(historical.data, FILENAME)

} else {
    historical.data = read_csv(FILENAME)

}
```

```
## Rows: 60 Columns: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (24): AMD.adj, MSFT.adj, SBUX.adj, AAPL.adj, ITUB.adj, FB.adj, NVDA.adj...
## date  (1): Date
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
adj.returns = historical.data %>%
                dplyr::select(Date, ends_with("adj"))

net.returns = historical.data %>%
                dplyr::select(Date, ends_with("net"))

risk.free = 0.03
```

Summary

Descriptive Statistics

```
returns.summary = summary(net.returns[, -1])

# Means
m = apply(net.returns[, -1], 2, mean)
m
```

```
##   AMD.net  MSFT.net  SBUX.net  AAPL.net  ITUB.net    FB.net  NVDA.net     F.net
## 4.8228678 2.5980821 0.7651816 2.8415787 0.7000566 0.9811454 3.6604329 1.3043109
##   BAC.net     T.net   XOM.net  VALE.net
## 1.3969590 0.2202332 1.0429256 2.1540932
```

```r
# Standard Deviations
cov.mat = cov(net.returns[, -1])
std.dev = diag(cov.mat) %>% sqrt()
std.dev
```

```
##   AMD.net  MSFT.net  SBUX.net  AAPL.net  ITUB.net    FB.net  NVDA.net     F.net
## 15.900872  5.691618  7.396289  8.629926 12.175097  9.442116 12.942200 10.313335
##   BAC.net     T.net   XOM.net  VALE.net
##  8.448155  5.586696  9.120811 11.076548
```

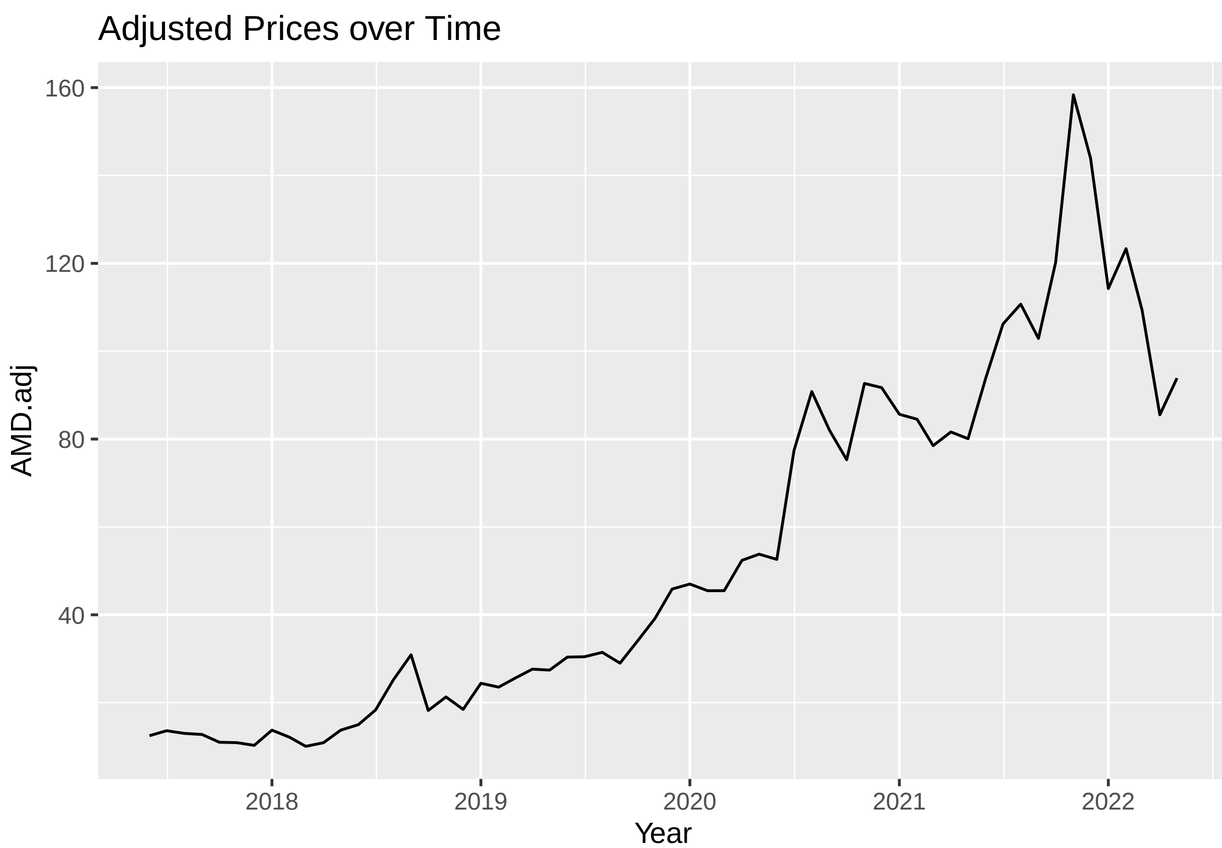
```r
# Skewness Coefficients
skewness.coeff = skewness(net.returns[, -1])
skewness.coeff
```

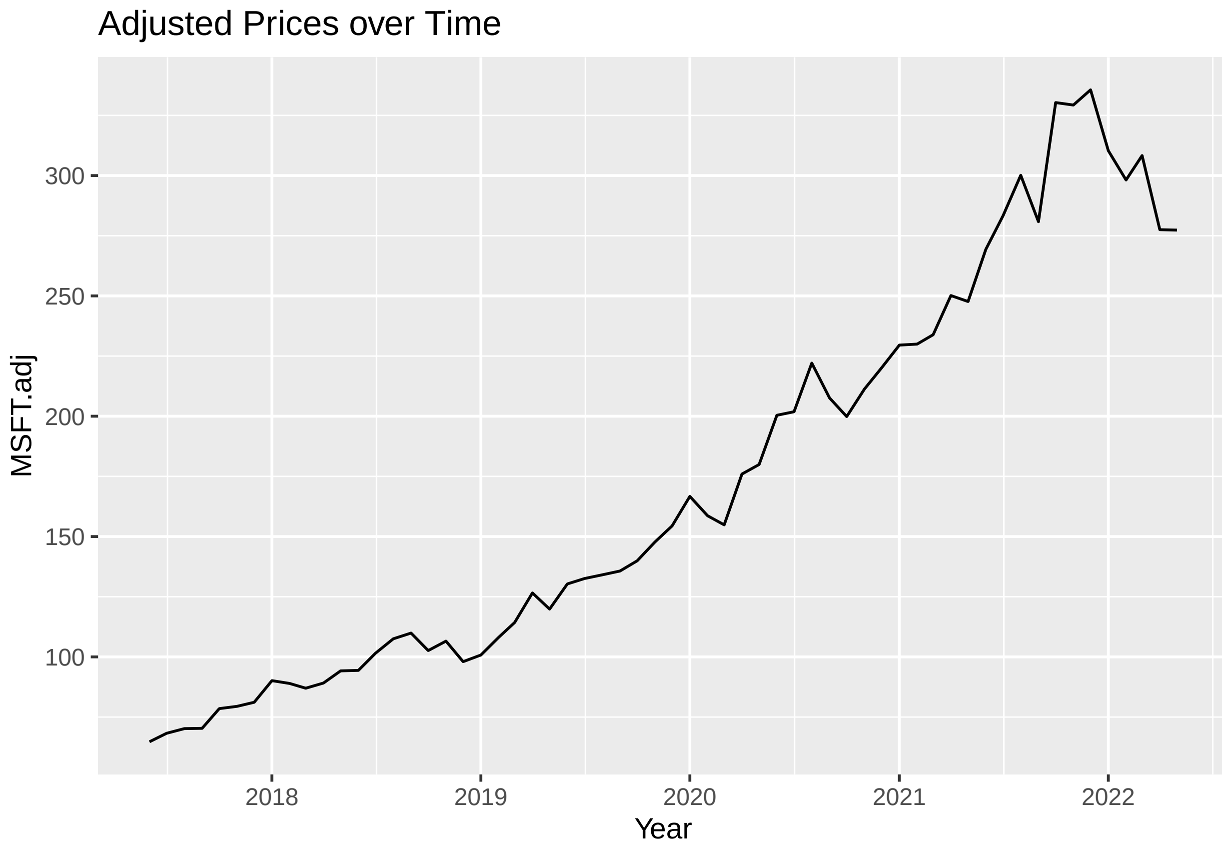
```
##      AMD.net     MSFT.net     SBUX.net     AAPL.net     ITUB.net       FB.net
##  0.135322794  0.042620769 -0.312409205 -0.160860757  0.057764285 -0.175890780
##     NVDA.net        F.net      BAC.net        T.net      XOM.net     VALE.net
## -0.469815917 -0.240321849 -0.444071970 -0.586152408 -0.009022973  0.257591674
```

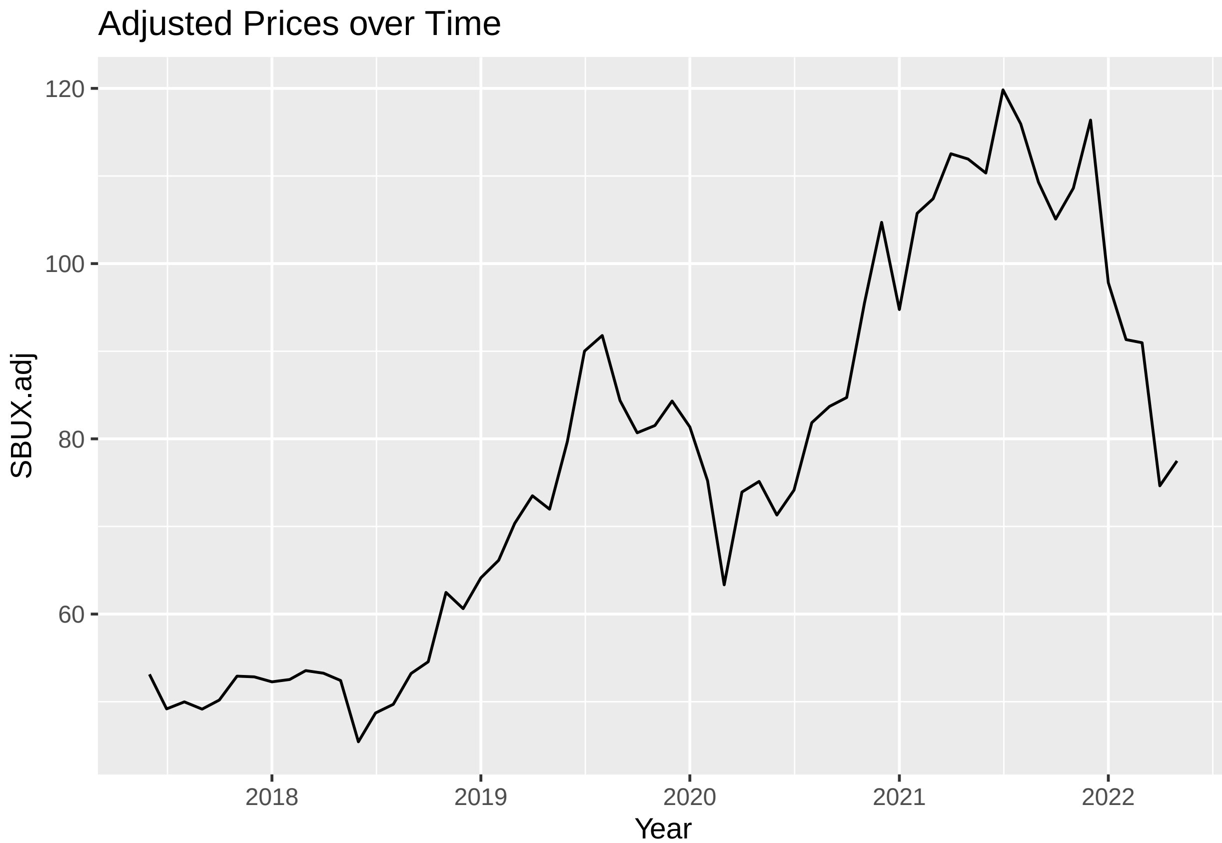
```r
# Kurtosis Coefficients
kurtosis.coeff = kurtosis(net.returns[, -1]) + 3
kurtosis.coeff
```

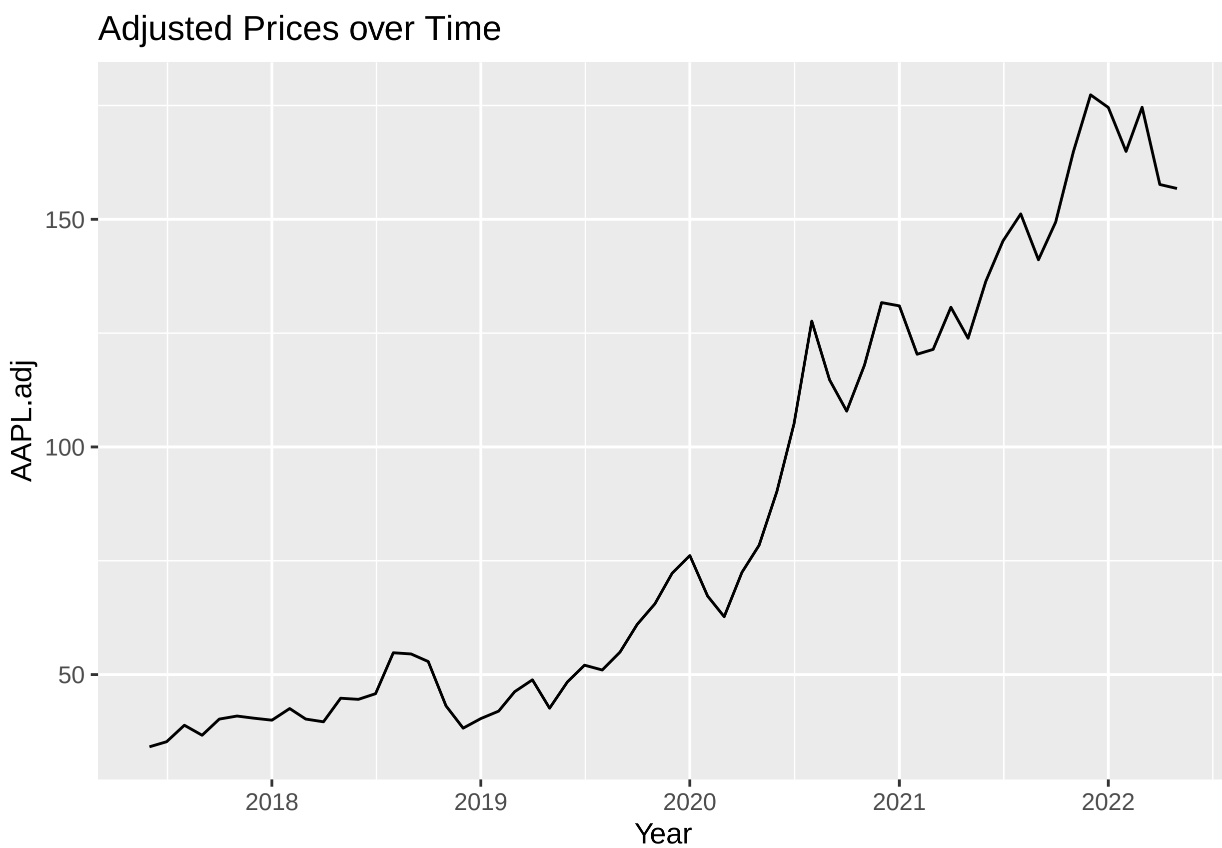
```
##  AMD.net MSFT.net SBUX.net AAPL.net ITUB.net   FB.net NVDA.net    F.net
## 3.384457 2.830736 3.056109 2.416976 3.300231 4.885452 3.247782 3.728845
##  BAC.net    T.net  XOM.net VALE.net
## 3.406410 3.496752 3.715535 3.729696
```

```r
# price.over.time.plot = adj.returns %>%
#                          ggplot(aes(x = Date))
for (asset in colnames(adj.returns)[-1]) {
    price.over.time.plot = adj.returns %>%
                           ggplot(aes(x = Date)) +
                               geom_line(aes(y = .data[[asset]])) +
                               labs(
                                   title = "Adjusted Prices over Time",
                                   x = "Year"
                               )
    price.over.time.plot
    print(price.over.time.plot)
}
```

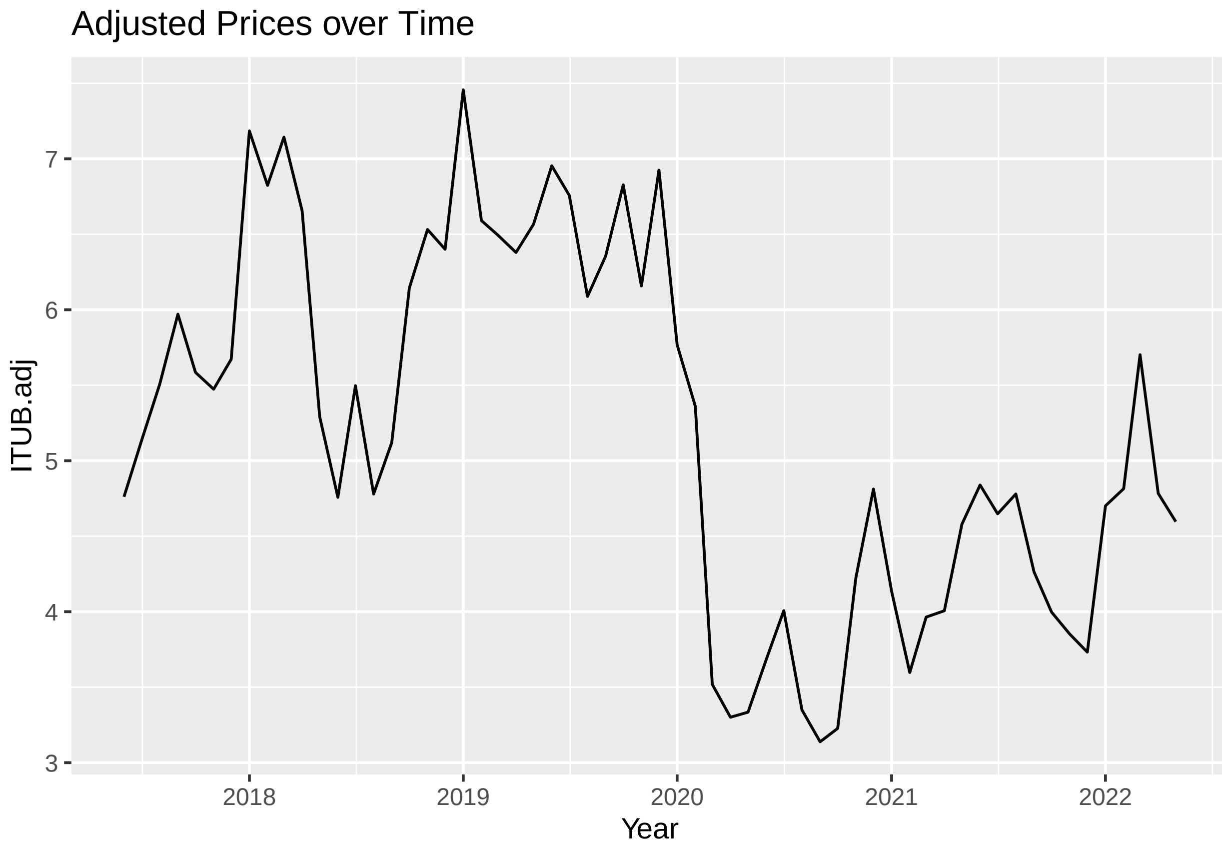

Adjusted Prices over Time
160
120
80
40
AMD.adj
2018
2019
2020
2021
2022
Year

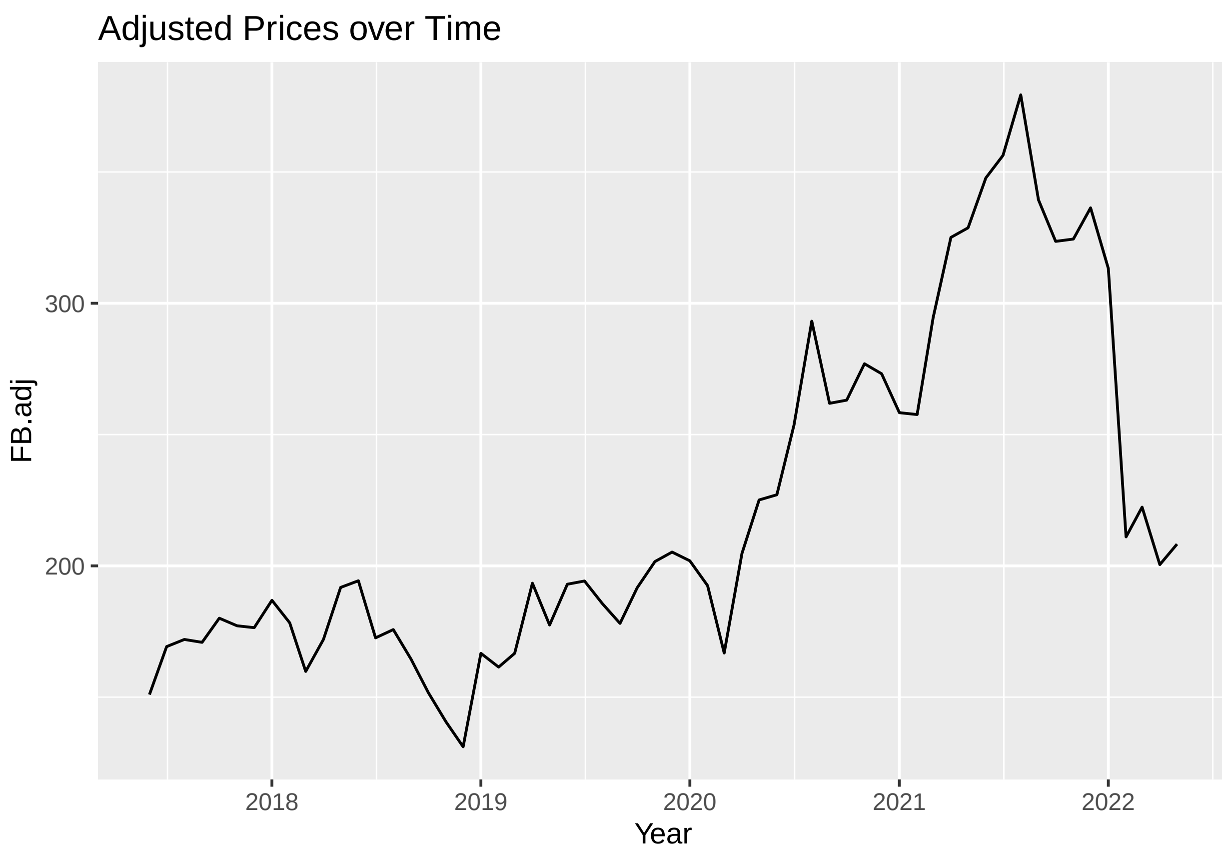

Adjusted Prices over Time
MSFT.adj
300
250
200
150
100
2018
2019
2020
2021
2022
Year


Adjusted Prices over Time
120
100
80
60
SBUX.adj
2018
2019
2020
2021
2022
Year

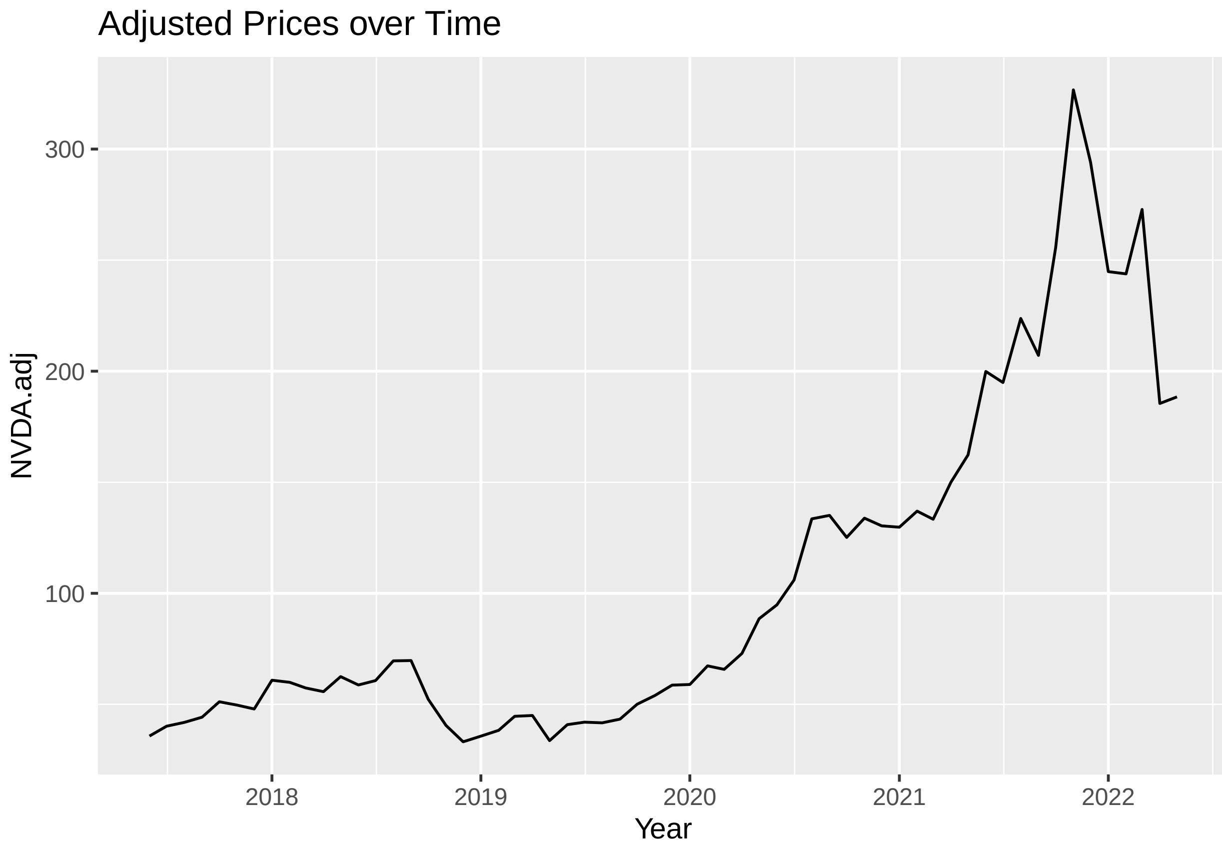


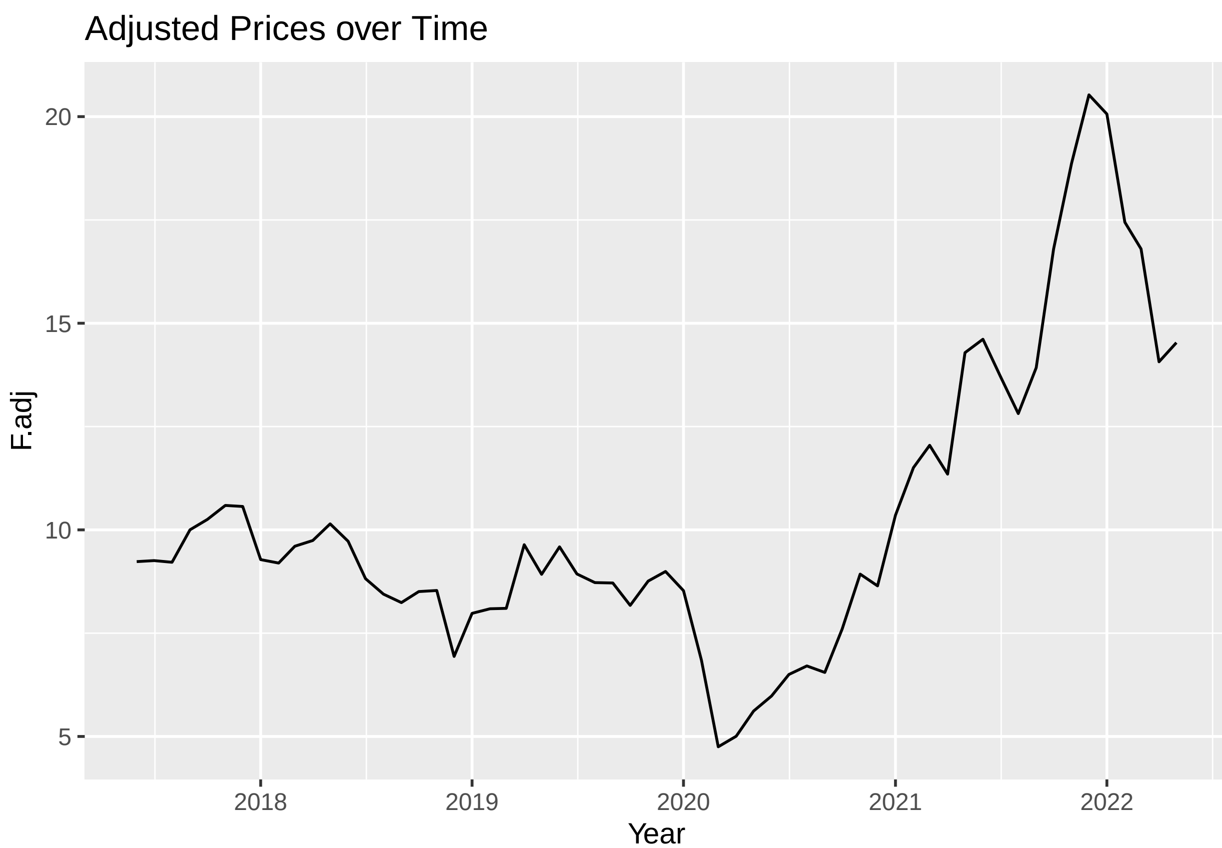
Adjusted Prices over Time
AAPL.adj
150
100
50
2018
2019
2020
2021
2022
Year

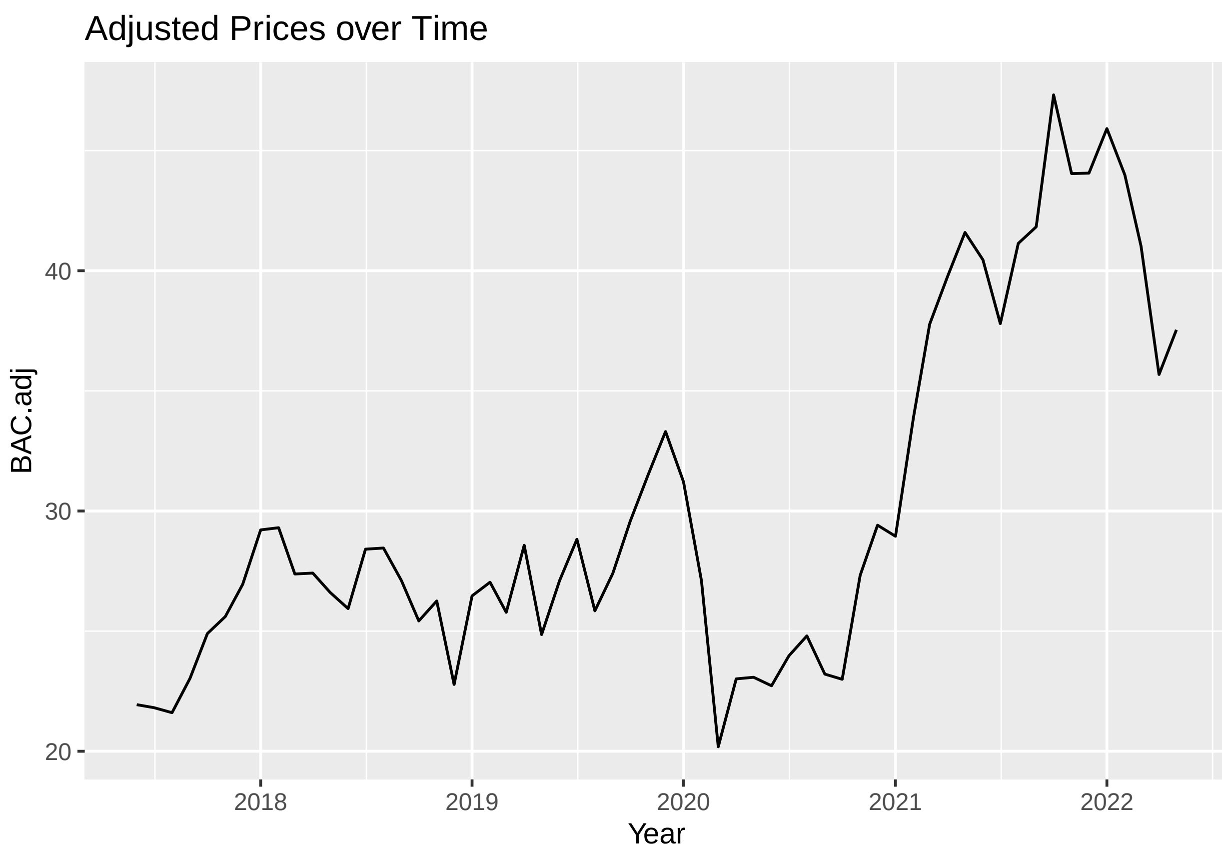

Adjusted Prices over Time
ITUB.adj
7
6
5
4
3
2018
2019
2020
2021
2022
Year

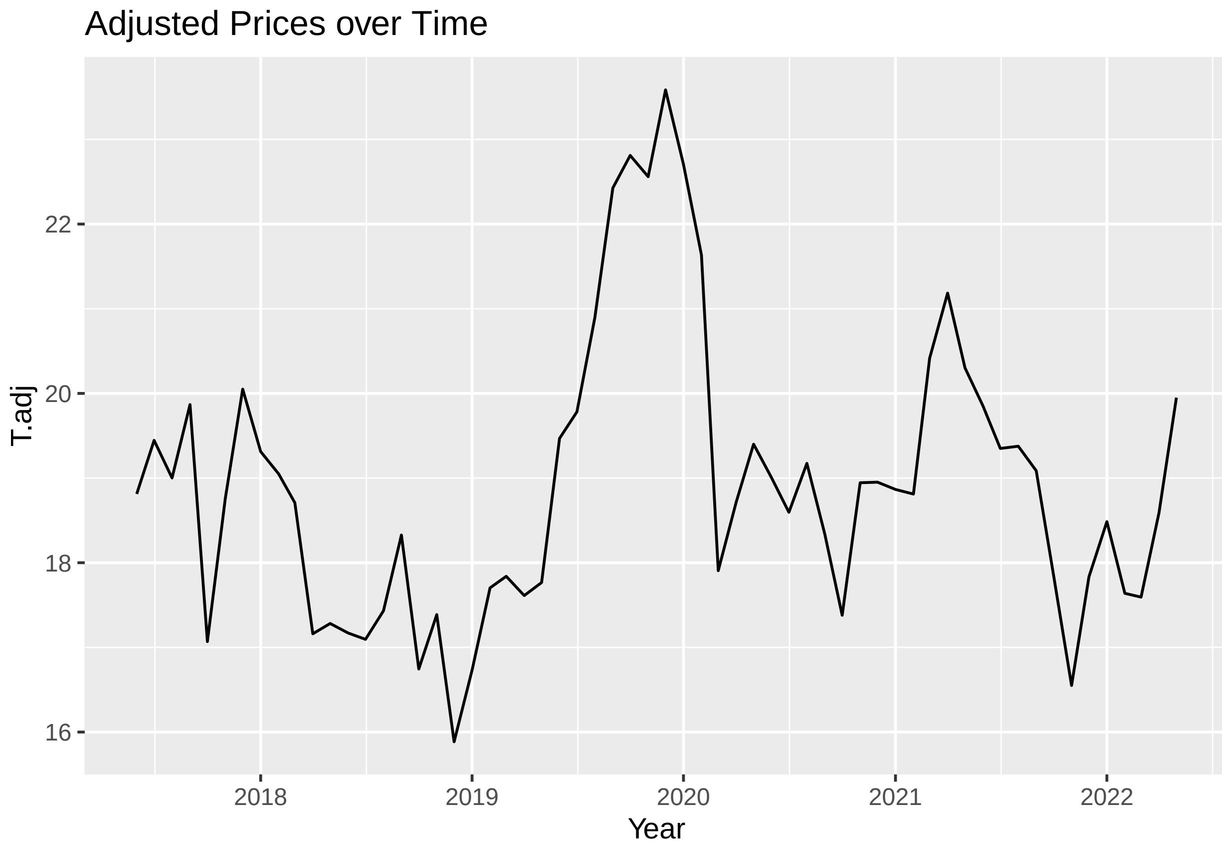


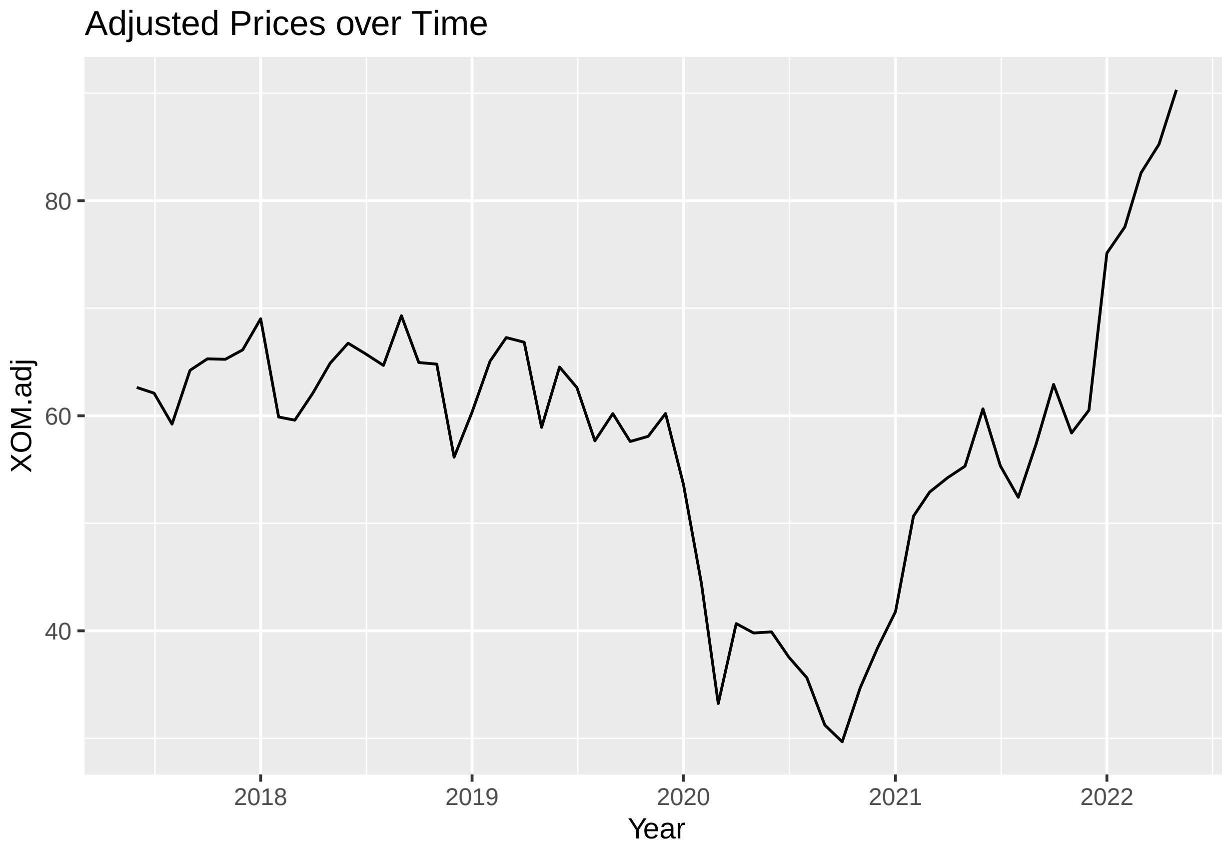
Adjusted Prices over Time
300
200
FB.adj
2018
2019
2020
2021
2022
Year

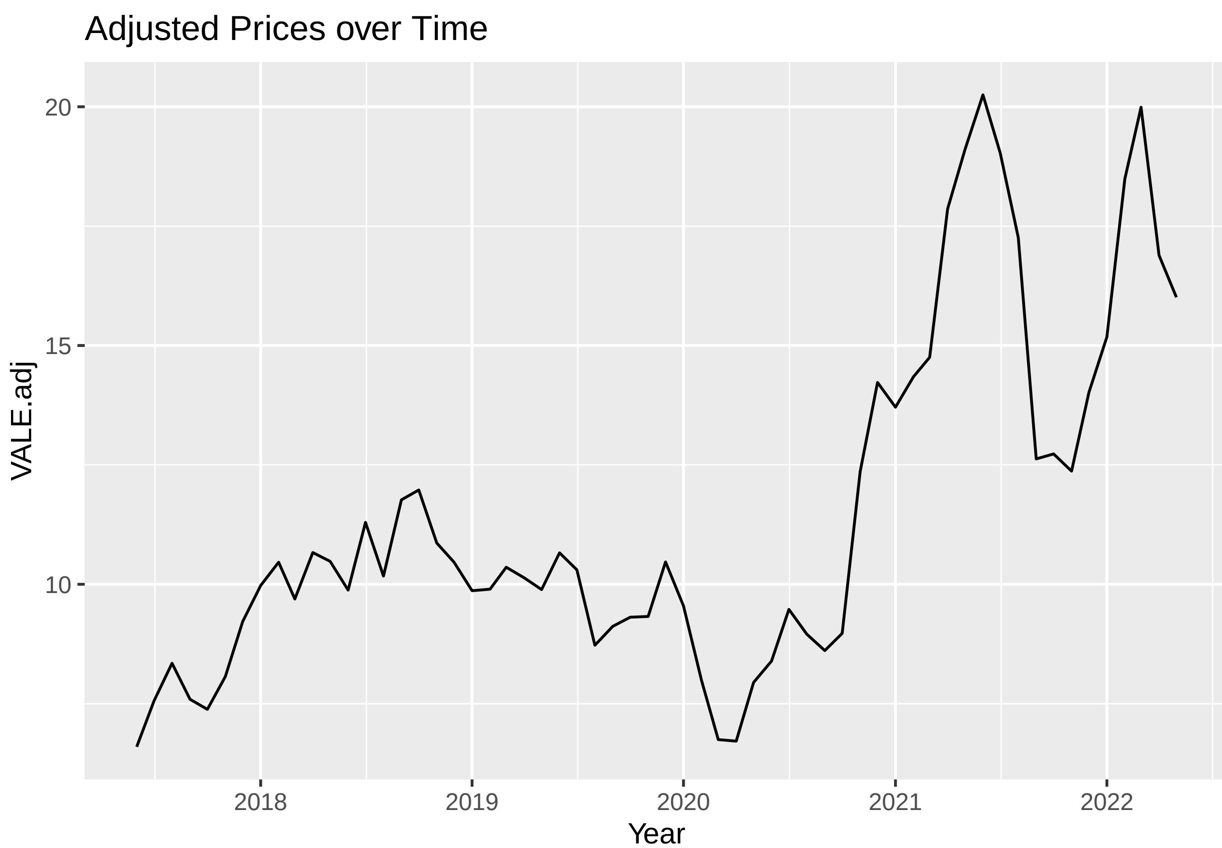

Adjusted Prices over Time
NVDA.adj
300
200
100
2018
2019
2020
2021
2022
Year


Adjusted Prices over Time
20
15
10
5
F.adj
2018
2019
2020
2021
2022
Year


Adjusted Prices over Time
BAC.adj
20
30
40
2018
2019
2020
2021
2022
Year


Adjusted Prices over Time
22
20
18
16
T.adj
2018
2019
2020
2021
2022
Year


Adjusted Prices over Time
XOM.adj
80
60
40
2018
2019
2020
2021
2022
Year




```r
# price.over.time.plot = price.over.time.plot +
#                          labs(
#                              title = "Adjusted Prices over Time",
#                              x = "Year",
#                              y = "Adjusted Price"
#                          )
# price.over.time.plot
```

Portfolio Theory

```r
inv.cov.mat = solve(cov.mat)
ones = matrix(1, dim(inv.cov.mat))

# Compute the MVP
w.mvp = (inv.cov.mat %*% ones) / as.numeric(t(ones) %*% inv.cov.mat %*% ones)
w.mvp
```

```
##                    [,1]
## AMD.net  -0.032573052
## MSFT.net  0.612858449
## SBUX.net  0.079631611
## AAPL.net -0.044901860
## ITUB.net  0.085210183
```

```
## FB.net   -0.041928315
## NVDA.net -0.006761453
## F.net     0.066097492
## BAC.net  -0.174039961
## T.net     0.484933697
## XOM.net  -0.017578298
## VALE.net -0.010948492
```

```
# Mean return
m.mvp = t(w.mvp) %*% m %>% as.numeric()
m.mvp
```

```
## [1] 1.270232
```

```
# Standard deviation
var.mvp = t(w.mvp) %*% cov.mat %*% w.mvp %>% as.numeric()
sd.mvp = sqrt(var.mvp)
sd.mvp
```

```
## [1] 3.870967
```

```
# VaR
S0 = 1e5
alpha.mvp = 0.05

# TODO: Change this to the quantile funciton of the
# appropriate (we don't know if the mvp is normally
# distributed)
VaR.mvp = -S0 * (m.mvp + sd.mvp * qnorm(alpha.mvp))
var.mvp
```

```
## [1] 14.98439
```

```
# Expected Shortfall


# Comment on the weights
```

Asset Allocation

```
target.expected.return.yearly = 0.06
target.expected.return.monthly = target.expected.return.yearly / 12
```

Principal Component analysis

Risk Management

Copulas

Conclusion