data

Zhiying Hu

2022-12-10

```
packs.inst <- c("readxl","foreign","dplyr","tidyr","ggplot2","stargazer","haven",
                "dummies","Hmisc","lmtest","sandwich", "doBy", "multiwayvcov",
                "miceadds", "car", "purrr", "knitr", "zoo", "readstata13",
                "tidyverse", "psych","wesanderson", "lubridate","reporttools",
                "data.table", "devtools","rmarkdown","estimatr", "ivpack",
                "Jmisc", "lfe", "plm", "tinytex", "xts", "psych",
                "PerformanceAnalytics","roll", "rollRegres", "glmnet", "hdm",
                "broom", "RCurl", "learnr", "maps", "fGarch", "remotes",
                "RPostgreSQL", "DBI", "RPostgreSQL", "remotes", "RPostgres",
                "Rmisc", "ggthemes", "splitstackshape", "gginference",
                "MASS","fBasics","xtable")
packs.load <- c("fGarch")
# lapply(packs.inst, install.packages, character.only = FALSE)
lapply(packs.inst, require, character.only = TRUE)
```

Part 1: deal with the data

import data

```
# real inflation rate from monthly year-on-year CPI
CPI <- read.csv("~/Desktop/DTFF_final/data/CPI.csv")
# expected inflation rate from 1-year bond yield to maturity
bond <- read.csv("~/Desktop/DTFF_final/data/bond.csv")
# commodity futures' price data
commodity <- read.csv("~/Desktop/DTFF_final/data/commodity.csv")[-1,]
# stock and gold price
stock_gold <- read.csv("~/Desktop/DTFF_final/data/stock_gold.csv")[-1,]
# real estate price
realestate <- read.csv("~/Desktop/DTFF_final/data/real_estate.csv")
```

time-series data

```
dateymd <- as.Date(ymd(commodity[,1]))
CPI_ts <- xts(x = CPI[,-1], order.by = dateymd)
bond_ts <- xts(x = bond[,-1], order.by = dateymd)
commodity_ts <- xts(x = commodity[,-1], order.by = dateymd)
```

```
stock_gold_ts <- xts(x = stock_gold[,-1], order.by = dateymd)
realestate_ts <- xts(x = realestate[,-1], order.by = dateymd[18:156])
```

merger the returns

```
# calculate the monthly year-on-year logarithm yield
colnames(CPI_ts) <- "monthly year-on-year CPI"
CPI_log_ts <- log(1+CPI_ts[,1])
colnames(CPI_log_ts) <- "real_inflation"
# 1-year bond t-1 the yield of maturity is used as the
# expected inflation rate at time t
bond_lag_ts <- stats::lag(bond_ts,1)
colnames(bond_lag_ts) <- "expected_inflation"
# monthly year-on-year logarithm yield of commodities
commodity_log_ts <- log(commodity_ts) - log(stats::lag(commodity_ts,12))
# monthly year-on-year logarithm yield of stock and gold
stock_gold_log_ts <- log(stock_gold_ts) - log(stats::lag(stock_gold_ts,12))
# monthly year-on-year logarithm yield of real estate
realestate_log_ts <- log(realestate_ts) - log(stats::lag(realestate_ts,12))

CPI_log_ts <- CPI_log_ts[13:156,]
bond_lag_ts <- bond_lag_ts[13:156,]
commodity_log_ts <- commodity_log_ts[13:156,]
stock_gold_log_ts <- stock_gold_log_ts[13:156,]
unexpected_inflation <- CPI_log_ts - bond_lag_ts
colnames(unexpected_inflation) <- "unexpected_inflation"
realestate_log_ts <- realestate_log_ts[13:139,]
DATA3 <- merge.xts(CPI_log_ts, bond_lag_ts, unexpected_inflation,
                   commodity_log_ts, stock_gold_log_ts, realestate_log_ts)
save(DATA3,file = "DATA.RData")
```

Part 2: Examination of the Inflation Hedging Ability

regression of commodity futures

```
## commodity futures
library(stargazer)
cm_lm1 =list()
for (i in 4:7) {
  cm_lm1[[i-3]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# cm_reg1 <- stargazer(cm_lm1, dep.var.labels = "commodity futures", align = TRUE, df = FALSE)
stargazer(cm_lm1, dep.var.labels = "commodity futures", align = TRUE, df = FALSE,
          type = "text")
```

```
##
## ==========================================================
##                              Dependent variable:
##                        ------------------------------------
```

```
##                                 commodity futures
##                           (1)        (2)      (3)     (4)
## -----------------------------------------------------------
## expected_inflation   -5.983*** -8.515*** 4.295*   -1.201
##                        (2.063)    (2.253)  (2.480) (2.207)
##
## unexpected_inflation   0.280     1.704*    0.928  -2.303**
##                        (0.876)    (0.957)  (1.053) (0.937)
##
## Constant             0.205***  0.251***  -0.078   0.012
##                        (0.057)    (0.062)  (0.068) (0.061)
##
## -----------------------------------------------------------
## Observations              144       144      144     144
## R2                       0.071     0.157    0.021   0.042
## Adjusted R2              0.058     0.145    0.007   0.029
## Residual Std. Error      0.138     0.151    0.166   0.148
## F Statistic            5.366***  13.119***  1.530  3.103**
## ===========================================================
## Note:                           *p<0.1; **p<0.05; ***p<0.01
```

```r
cm_lm2 =list()
for (i in 8:11) {
  cm_lm2[[i-7]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# cm_reg2 <- stargazer(cm_lm2, dep.var.labels = "commodity futures", align = TRUE, df = FALSE)
stargazer(cm_lm2, dep.var.labels = "commodity futures", align = TRUE, df = FALSE,
          type = "text")
```

```
##
## ============================================================
##                                 Dependent variable:
##                      ---------------------------------------
##                                  commodity futures
##                          (1)      (2)        (3)       (4)
## ------------------------------------------------------------
## expected_inflation   -2.744  -9.813*** -10.418*** -5.093**
##                      (2.186)  (2.878)    (2.420)   (2.575)
##
## unexpected_inflation  0.616   2.774**    2.056**   -2.176**
##                      (0.928)  (1.222)    (1.028)   (1.042)
##
## Constant              0.094  0.302***   0.311***  0.152**
##                      (0.060)  (0.079)    (0.067)   (0.072)
##
## ------------------------------------------------------------
## Observations           144      144        144       140
## R2                    0.021    0.157      0.193     0.043
## Adjusted R2           0.007    0.145      0.182     0.029
## Residual Std. Error   0.146    0.193      0.162     0.164
## F Statistic           1.540  13.171***  16.912***   3.049*
## ============================================================
## Note:                            *p<0.1; **p<0.05; ***p<0.01
```

```r
cm_lm3 =list()
for (i in 12:15) {
  cm_lm3[[i-11]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# cm_reg3 <- stargazer(cm_lm3, dep.var.labels = "commodity futures", align = TRUE, df = FALSE)
stargazer(cm_lm3, dep.var.labels = "commodity futures", align = TRUE, df = FALSE,
          type = "text")
```

```
##
## ==========================================================
##                                 Dependent variable:
##                         ----------------------------------
##                                 commodity futures
##                           (1)      (2)      (3)      (4)
## ----------------------------------------------------------
## expected_inflation      -7.728** -4.968**  -4.505   -7.502***
##                         (3.181)  (2.188)  (2.936)  (1.733)
##
## unexpected_inflation    -1.291   -1.685*  -5.235*** 5.103***
##                         (1.351)  (0.929)  (1.247)  (0.736)
##
## Constant                0.251*** 0.157***  0.145*   0.269***
##                         (0.087)  (0.060)  (0.081)  (0.048)
##
## ----------------------------------------------------------
## Observations              144      144      144      144
## R2                       0.040    0.042    0.111    0.425
## Adjusted R2              0.027    0.029    0.098    0.417
## Residual Std. Error      0.213    0.146    0.196    0.116
## F Statistic              2.952*  3.113**  8.806*** 52.096***
## ==========================================================
## Note:                            *p<0.1; **p<0.05; ***p<0.01
```

```r
cm_lm4 =list()
for (i in 16:19) {
  cm_lm4[[i-15]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# cm_reg4 <- stargazer(cm_lm4, dep.var.labels = "commodity futures", align = TRUE, df = FALSE)
stargazer(cm_lm4, dep.var.labels = "commodity futures", align = TRUE, df = FALSE,
          type = "text")
```

```
##
## =======================================================
##                                Dependent variable:
##                         -------------------------------
##                                commodity futures
##                             (1)       (2)     (3)     (4)
## -------------------------------------------------------
## expected_inflation      -14.338*** -4.748  -2.444   0.377
##                          (3.966)   (4.215) (3.704) (3.346)
##
## unexpected_inflation     3.739**   -1.961  -1.508  -0.351
##                          (1.685)   (1.790) (1.535) (1.387)
##
```

```
## Constant              0.392***   0.095   0.080   0.015
##                       (0.109)   (0.116) (0.102) (0.093)
##
## ---------------------------------------------------------
## Observations             144       144     142     140
## R2                      0.166     0.013   0.008   0.001
## Adjusted R2             0.154    -0.001  -0.007  -0.014
## Residual Std. Error     0.265     0.282   0.241   0.218
## F Statistic           14.007***   0.897   0.533   0.057
## =========================================================
## Note:                         *p<0.1; **p<0.05; ***p<0.01
```

regression of spot gold

```
## spot gold
sg <- lm(DATA3$Gold_price ~ DATA3$expected_inflation +
           DATA3$unexpected_inflation)
# sg_reg <- stargazer(sg, dep.var.labels = "Spot Gold", align = TRUE, df = FALSE)
stargazer(sg, dep.var.labels = "Spot Gold", align = TRUE, df = FALSE,
          type = "text")
```

```
##
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                               Spot Gold
## ------------------------------------------------
## expected_inflation            -7.454***
##                                (1.714)
##
## unexpected_inflation           5.128***
##                                (0.728)
##
## Constant                       0.267***
##                                (0.047)
##
## ------------------------------------------------
## Observations                     144
## R2                              0.431
## Adjusted R2                     0.423
## Residual Std. Error             0.115
## F Statistic                   53.374***
## ================================================
## Note:                  *p<0.1; **p<0.05; ***p<0.01
```

regression of industry stocks

```
## industry stocks
stock_lm1 =list()
for (i in 21:25) {
```

```r
  stock_lm1[[i-20]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# stock_reg1 <- stargazer(stock_lm1, dep.var.labels = "industry stocks", align = TRUE, df = FALSE)
stargazer(stock_lm1, dep.var.labels = "industry stocks", align = TRUE,
          df = FALSE, type = "text")
```

```
##
## ===================================================================
##                                   Dependent variable:
##                         -------------------------------------------
##                                     industry stocks
##                           (1)      (2)       (3)       (4)      (5)
## -------------------------------------------------------------------
## expected_inflation     -2.188  -9.063**   -7.416    -3.864    -4.955
##                        (3.515)  (4.328)   (4.488)   (3.589)  (3.551)
##
## unexpected_inflation -2.023  -6.328*** -6.728*** -5.326***  -1.565
##                        (1.493)  (1.838)   (1.907)   (1.525)  (1.509)
##
## Constant               -0.017   0.250**   0.206*    0.169*   0.300***
##                        (0.097)  (0.119)   (0.123)   (0.099)  (0.098)
##
## -------------------------------------------------------------------
## Observations             144      144       144       144      144
## R2                      0.013    0.082     0.082     0.080    0.016
## Adjusted R2            -0.001    0.069     0.069     0.067    0.002
## Residual Std. Error     0.235    0.290     0.300     0.240    0.238
## F Statistic             0.925   6.296***  6.287***  6.134***  1.126
## ===================================================================
## Note:                                   *p<0.1; **p<0.05; ***p<0.01
```

```r
stock_lm2 =list()
for (i in 26:30) {
  stock_lm2[[i-25]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# stock_reg2 <- stargazer(stock_lm2, dep.var.labels = "industry stocks", align = TRUE, df = FALSE)
stargazer(stock_lm2, dep.var.labels = "industry stocks", align = TRUE,
          df = FALSE, type = "text")
```

```
##
## ===================================================================
##                                   Dependent variable:
##                         -------------------------------------------
##                                     industry stocks
##                            (1)         (2)       (3)     (4)      (5)
## -------------------------------------------------------------------
## expected_inflation     -11.363***   -2.650    -6.993*  4.802    -4.069
##                          (3.302)    (3.117)   (4.168) (4.398)  (3.401)
##
## unexpected_inflation -4.228***   -5.384*** -2.201   -1.412   -7.689***
##                          (1.403)    (1.324)   (1.770) (1.868)  (1.445)
##
## Constant                 0.416***    0.090    0.253** -0.135    0.085
##                          (0.091)    (0.086)   (0.115) (0.121)  (0.094)
```

```
##
## ------------------------------------------------------------------
## Observations           144        144       144     144      144
## R2                    0.098      0.108     0.023   0.019    0.171
## Adjusted R2           0.085      0.095     0.009   0.005    0.159
## Residual Std. Error   0.221      0.208     0.279   0.294    0.228
## F Statistic          7.641***   8.541***   1.625   1.390  14.542***
## ==================================================================
## Note:                                   *p<0.1; **p<0.05; ***p<0.01
```

regression of real estate

```r
## real estate
estate_lm = list()
for (i in 31:33) {
  estate_lm[[i-30]]  <- lm(DATA3[,i] ~ DATA3$expected_inflation + DATA3$unexpected_inflation)}
# estate_reg <- stargazer(estate_lm, dep.var.labels = "real estate", align = TRUE, df = FALSE)
stargazer(estate_lm, dep.var.labels = "real estate", align = TRUE,
          df = FALSE, type = "text")
```

```
##
## ==================================================
##                           Dependent variable:
##                       ----------------------------
##                               real estate
##                         (1)       (2)       (3)
## --------------------------------------------------
## expected_inflation     0.720    2.110**  2.589***
##                       (1.294)   (0.850)   (0.835)
##
## unexpected_inflation  -0.172     0.449    0.704*
##                       (0.568)   (0.373)   (0.367)
##
## Constant               0.046    -0.010    -0.030
##                       (0.036)   (0.024)   (0.023)
##
## --------------------------------------------------
## Observations            127       127       127
## R2                     0.004     0.049     0.079
## Adjusted R2           -0.012     0.034     0.064
## Residual Std. Error    0.077     0.051     0.050
## F Statistic            0.277    3.189**  5.338***
## ==================================================
## Note:                  *p<0.1; **p<0.05; ***p<0.01
```

result of inflation hedging effect

```r
assetnames <- colnames(DATA3[,-(1:3)])
Expected   = c(assetnames[c(1,2,3,6,7,8,9,10,12,13,17,19,23,25,29,30)],NA)
Unexpected = assetnames[c(2,4,6,7,8,10,11,12,13,17,19,20,21,23,24,27,30)]
```

```r
HedgingAbility <- cbind(Expected, Unexpected)
as.data.frame(HedgingAbility)
```

```
##                          Expected                       Unexpected
## 1                  soybeans.No..1                   soybeans.No..2
## 2                  soybeans.No..2                            LLDPE
## 3                     yellow.corn                         palm.oil
## 4                        palm.oil                      soybean.oil
## 5                     soybean.oil                              PVC
## 6                             PVC                         aluminum
## 7                  cathode.copper                             zinc
## 8                        aluminum                             gold
## 9                            gold                   natural.rubber
## 10                 natural.rubber                       Gold_price
## 11                     Gold_price                   Material_price
## 12                 Material_price                   Industry_price
## 13   Medicine_health_care_price    Optional_consumption_price
## 14 Information_technology_price    Medicine_health_care_price
## 15                    second.tier Finance_and_real_estate_price
## 16                     third.tier                    Utility_price
## 17                           <NA>                       third.tier
```

```r
xtable(as.data.frame(HedgingAbility))
```

```
## % latex table generated in R 4.1.2 by xtable 1.8-4 package
## % Mon Dec 19 11:35:45 2022
## \begin{table}[ht]
## \centering
## \begin{tabular}{rll}
##   \hline
##  & Expected & Unexpected \\
##   \hline
## 1 & soybeans.No..1 & soybeans.No..2 \\
##   2 & soybeans.No..2 & LLDPE \\
##   3 & yellow.corn & palm.oil \\
##   4 & palm.oil & soybean.oil \\
##   5 & soybean.oil & PVC \\
##   6 & PVC & aluminum \\
##   7 & cathode.copper & zinc \\
##   8 & aluminum & gold \\
##   9 & gold & natural.rubber \\
##   10 & natural.rubber & Gold\_price \\
##   11 & Gold\_price & Material\_price \\
##   12 & Material\_price & Industry\_price \\
##   13 & Medicine\_health\_care\_price & Optional\_consumption\_price \\
##   14 & Information\_technology\_price & Medicine\_health\_care\_price \\
##   15 & second.tier & Finance\_and\_real\_estate\_price \\
##   16 & third.tier & Utility\_price \\
##   17 &  & third.tier \\
##    \hline
## \end{tabular}
## \end{table}
```

Part 3: Inflation Hedgig Portfolio by Mean Variance Method

exclude assets which have no inflation hedging effect

```
asset_returns <- DATA3[,-c(1:3)]
asset_returns <- asset_returns[,-c(5,14,15,16,18,22,26,28)]
save(asset_returns, file="asset_returns.RData")
dim(asset_returns)
```

```
## [1] 144  22
```

the set of attainable portfolios

```
# Calculate the mean vector and covariance matrix
mean_ret <- colMeans(asset_returns, na.rm = T)
cvar_ret <- cov(na.omit(asset_returns))

# Calculate individual weights
for (i in colnames(asset_returns)[-(dim(asset_returns)[2])]) {
  weight = runif(10000, min=-1.5, max=1.5)
  names = paste0(i,"_weight")
  if (i == "soybeans.No..1"){
    weight_final = weight
    names_final = names}
  else {
    weight_final = cbind(weight_final, weight)
    names_final = cbind(names_final, names)}
  }

# Get the dataframe and matrix on the weights
weight_df <- as.data.frame(weight_final)
colnames(weight_df) <- names_final

weight_df$sum <- rowSums(weight_df)
weight_df$third.tier <- 1 - weight_df$sum
weight_df$sum <- NULL

matrix_weights <- as.matrix(weight_df)

# report the weights of the matrix
str(data.frame(matrix_weights))
```

```
## 'data.frame':    10000 obs. of  22 variables:
##  $ soybeans.No..1_weight             : num  -0.87 0.376 0.694 1.14 -0.264 ...
##  $ soybeans.No..2_weight             : num  1.0855 0.0248 0.8569 -0.1449 -1.1355 ...
##  $ yellow.corn_weight                : num  -1.05 -0.281 -0.229 0.759 -0.685 ...
##  $ LLDPE_weight                      : num  -0.861 1.491 -0.764 -0.608 0.642 ...
##  $ palm.oil_weight                   : num  0.2531 -0.0476 -1.1641 0.9255 1.2068 ...
##  $ soybean.oil_weight                : num  -0.3239 1.4303 -1.1986 -0.0766 -0.916 ...
##  $ PVC_weight                        : num  0.685 0.428 0.603 0.044 1.135 ...
```

```
##  $ cathode.copper_weight              : num  -1.214 -0.243 -0.629 0.521 0.789 ...
##  $ aluminum_weight                    : num  -1.3646 -0.5749 0.7313 -0.0288 -1.1799 ...
##  $ zinc_weight                        : num  -0.285 -0.922 1.432 -0.456 0.193 ...
##  $ gold_weight                        : num  -1.37 0.737 0.227 -1.451 -0.283 ...
##  $ natural.rubber_weight              : num  -1.32334 -0.21498 -0.87405 0.00257 -1.03066 ...
##  $ Gold_price_weight                  : num  0.477 -0.745 0.18 -1.336 0.266 ...
##  $ Material_price_weight              : num  0.674 0.625 -0.948 0.37 -0.706 ...
##  $ Industry_price_weight              : num  -0.76 -1.22 1.39 1.03 1.19 ...
##  $ Optional_consumption_price_weight  : num  -0.737 -0.0134 1.1319 0.1763 -0.6044 ...
##  $ Medicine_health_care_price_weight  : num  0.8245 -0.3463 0.5812 -0.0853 -0.663 ...
##  $ Finance_and_real_estate_price_weight: num  -0.547 1.276 0.776 -0.695 0.129 ...
##  $ Information_technology_price_weight : num  0.957 0.763 1.468 1.413 1.192 ...
##  $ Utility_price_weight               : num  0.966 -0.873 1.237 1.146 -0.577 ...
##  $ second.tier_weight                 : num  1.038 -1.226 0.185 -1.267 -0.711 ...
##  $ third.tier                         : num  4.746 0.554 -4.685 -0.382 3.011 ...
```

```
head(data.frame(matrix_weights))
```

```
##   soybeans.No..1_weight soybeans.No..2_weight yellow.corn_weight LLDPE_weight
## 1            -0.8697047            1.08548322         -1.0497068   -0.8609648
## 2             0.3759407            0.02475116         -0.2812018    1.4912421
## 3             0.6936359            0.85686681         -0.2286985   -0.7640489
## 4             1.1403399           -0.14493818          0.7590308   -0.6083235
## 5            -0.2640885           -1.13554971         -0.6852093    0.6417833
## 6            -0.5831766           -0.97397548          0.5415216   -0.2937817
##   palm.oil_weight soybean.oil_weight  PVC_weight cathode.copper_weight
## 1      0.25310190        -0.32388840  0.68516101            -1.2135206
## 2     -0.04757492         1.43029518  0.42792848            -0.2427104
## 3     -1.16414544        -1.19860180  0.60256111            -0.6294600
## 4      0.92546779        -0.07661164  0.04395994             0.5210182
## 5      1.20676953        -0.91595253  1.13454171             0.7892131
## 6      1.07274809        -1.22016066 -1.46965804             0.3209216
##   aluminum_weight zinc_weight gold_weight natural.rubber_weight
## 1     -1.36459750  -0.2850450  -1.3703162          -1.323335533
## 2     -0.57489271  -0.9216528   0.7372608          -0.214977851
## 3      0.73127362   1.4319539   0.2266326          -0.874045140
## 4     -0.02879568  -0.4556886  -1.4513554           0.002572911
## 5     -1.17989209   0.1925896  -0.2828577          -1.030664707
## 6     -1.48339413  -1.0799827   0.3287778          -0.021488145
##   Gold_price_weight Material_price_weight Industry_price_weight
## 1         0.4774139             0.6736064            -0.7602397
## 2        -0.7448301             0.6249821            -1.2201865
## 3         0.1801266            -0.9483180             1.3885380
## 4        -1.3360170             0.3698051             1.0335330
## 5         0.2659361            -0.7055816             1.1929115
## 6        -0.7198242             0.7463962            -0.1778390
##   Optional_consumption_price_weight Medicine_health_care_price_weight
## 1                       -0.73696109                        0.82445603
## 2                       -0.01338238                       -0.34625060
## 3                        1.13186456                        0.58115829
## 4                        0.17627947                       -0.08534822
## 5                       -0.60442845                       -0.66303665
## 6                        1.35497046                        1.00981924
##   Finance_and_real_estate_price_weight Information_technology_price_weight
```

```
## 1                          -0.5470076                         0.9567959
## 2                           1.2761527                         0.7630503
## 3                           0.7764694                         1.4682895
## 4                          -0.6948591                         1.4126797
## 5                           0.1293245                         1.1920905
## 6                           1.1144662                         1.3980168
##   Utility_price_weight second.tier_weight third.tier
## 1            0.9657767          1.0379293  4.7455637
## 2           -0.8728914         -1.2255513  0.5544992
## 3            1.2374428          0.1853518 -4.6848473
## 4            1.1460413         -1.2669733 -0.3818175
## 5           -0.5773016         -0.7113386  3.0107418
## 6            1.4683490         -1.0771927  0.7444863
```

```r
# Calculate the feasible expected returns and standard deviations
feasible_pf_mu = matrix_weights%*%mean_ret
feasible_pf_sd = apply(matrix_weights, 1,
                       function(x) sqrt(t(x) %*% cvar_ret %*% x))

# Construct the feasible dataframe
# consisting of 10000 differently weighted risk and return combinations
feasible_pf <- as.data.frame(cbind(feasible_pf_mu, feasible_pf_sd))
colnames(feasible_pf) <- c("Portfolio_Return", "Portfolio_Risk")

# report the feasible dataframe
str(feasible_pf)
```

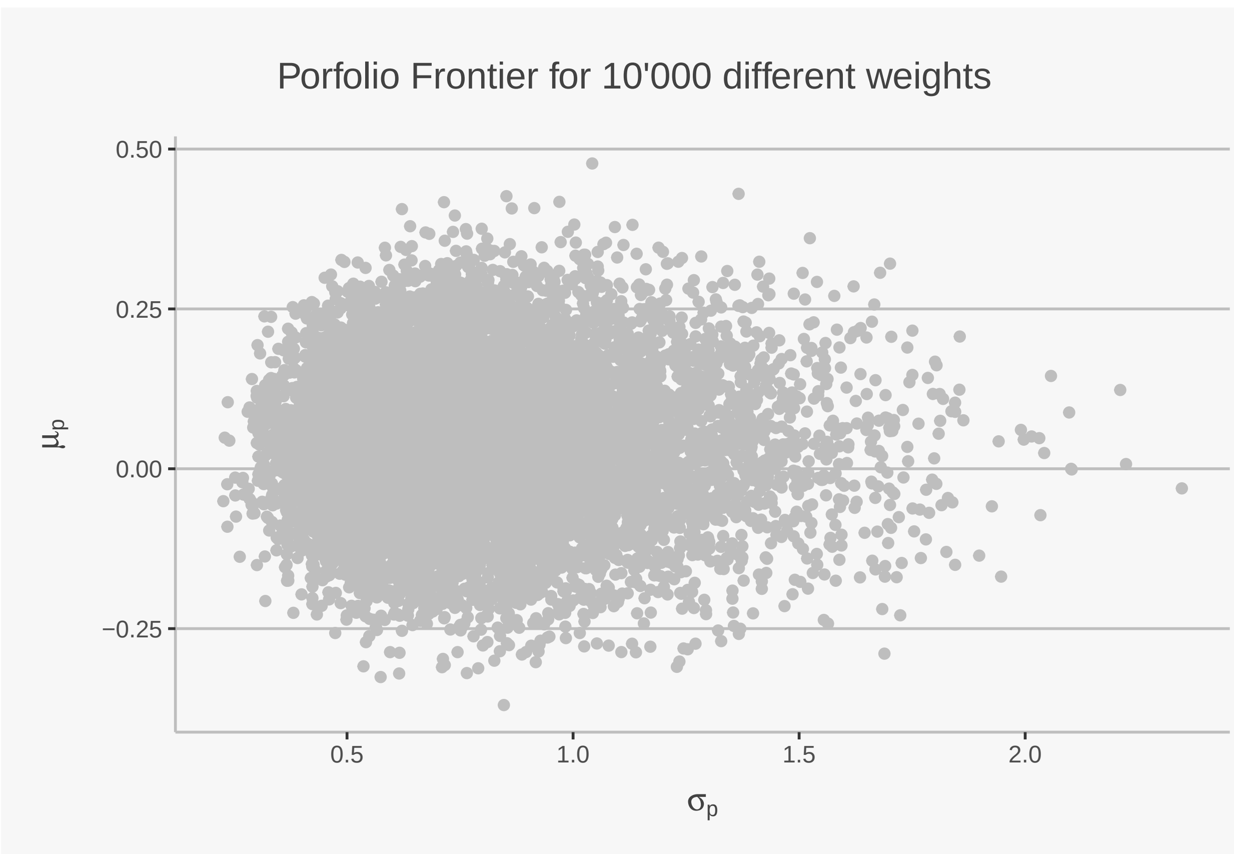
```
## 'data.frame':    10000 obs. of  2 variables:
##  $ Portfolio_Return: num  0.15516 -0.03699 0.20303 0.00912 0.05502 ...
##  $ Portfolio_Risk  : num  0.725 0.378 1.511 1.218 0.418 ...
```

```r
head(feasible_pf)
```

```
##   Portfolio_Return Portfolio_Risk
## 1      0.155161098      0.7245180
## 2     -0.036992912      0.3776066
## 3      0.203027942      1.5105364
## 4      0.009122775      1.2177005
## 5      0.055018535      0.4182612
## 6      0.229974323      1.6610747
```

```r
# Now, let's visualise the relationship
feasible_pf %>%
  ggplot(aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(color = "grey") +
  geom_point(data = subset(feasible_pf, Portfolio_Risk <= 0.12 &
                             Portfolio_Return >= 0), color = "darkorchid3",
             shape = 1, aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.12 &
                             Portfolio_Risk <= 0.14 & Portfolio_Return >= 0.07),
             color = "darkorchid3", shape = 1,aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.14 &
```

```
                              Portfolio_Risk <= 0.16 & Portfolio_Return >= 0.09),
           color = "darkorchid3", shape = 1,aes(x= Portfolio_Risk, y = Portfolio_Return)) +
geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.16 &
                              Portfolio_Risk <= 0.18 & Portfolio_Return >= 0.1),
           color = "darkorchid3", shape = 1,aes(x= Portfolio_Risk, y = Portfolio_Return)) +
geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.18 & Portfolio_Risk
                            <= 0.20 & Portfolio_Return >= 0.11),
           color = "darkorchid3",shape = 1, aes(x= Portfolio_Risk, y = Portfolio_Return)) +
geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.2 &
                              Portfolio_Risk <= 0.22 & Portfolio_Return >= 0.11),
           color = "darkorchid3", shape = 1, aes(x= Portfolio_Risk, y = Portfolio_Return)) +
ylab(expression(mu[p])) + xlab(expression(sigma[p])) +
ggtitle("Porfolio Frontier for 10'000 different weights") +
labs(color='Factor Portfolios') +
theme(plot.title= element_text(size=14, color="grey26",
hjust=0.3,lineheight=2.4, margin=margin(15,0,15,0)),
panel.background = element_rect(fill="#f7f7f7"),
panel.grid.major.y = element_line(size = 0.5,linetype ="solid",color="grey"),
panel.grid.minor = element_blank(),
panel.grid.major.x = element_blank(),
plot.background = element_rect(fill="#f7f7f7", color = "#f7f7f7"),
axis.title.y = element_text(color="grey26", size=12, margin=margin(0,10,0,10)),
axis.title.x = element_text(color="grey26", size=12, margin=margin(10,0,10,0)),
axis.line = element_line(color = "grey"))
```



the minmum - variance portfolio

```r
# Define the matrix A. It consists of:

## the covariance matrix multiplied by two
## a column right to the covariance matrix, consisting of 1's
## a row right below the covariance matrix and the additional column,
## consisting of 1's and one zero (the zero is in the right-bottom
## of the resulting matrix)
mat_A <- rbind(cbind(2*cvar_ret, rep(1, dim(cvar_ret)[1])),
               c(rep(1, dim(cvar_ret)[1]), 0))
# Define the vector b as vector of zeros with dimension of the covariance
# matrix and one 1 at the bottom
vec_b <- c(rep(0, dim(cvar_ret)[1]), 1)
# Calculate the inverse and perform matrix multiplication to get the vector z
z <- solve(mat_A)%*%vec_b
# Derive the first N elements of the vector to retrieve the actual values
x_MV <- z[1:dim(cvar_ret)[1]]
# Check that the sum adds up to 1
sum(x_MV)
```

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

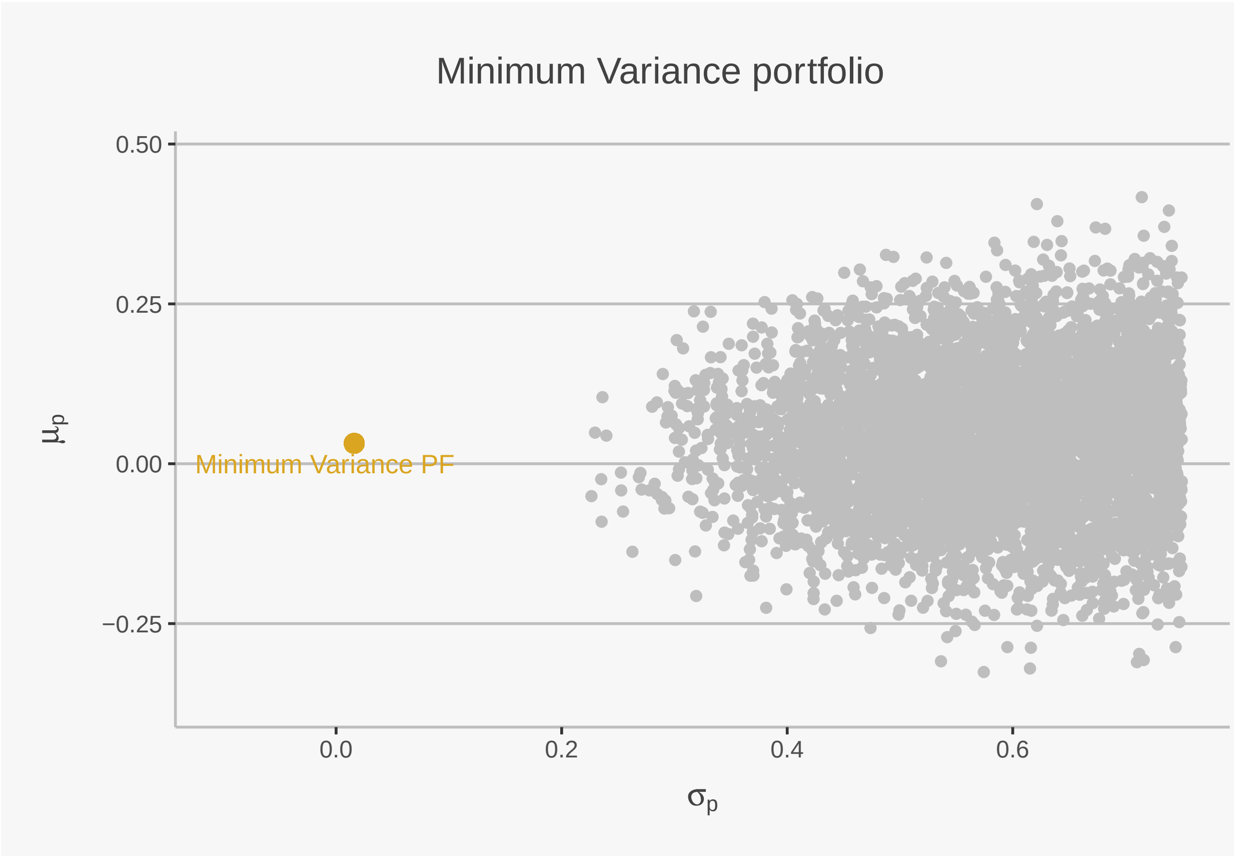
```r
# Calculate the expected return:
mu_MV <- x_MV %*% mean_ret
sd_MV <- sqrt(t(x_MV) %*% cvar_ret %*% x_MV)
# Create the appropriate dataframe
MV_PF <- as.data.frame(cbind(mu_MV, sd_MV, t(x_MV)))
colnames(MV_PF) <- c("Mu_MV", "Sd_MV",names_final, "third.tier_weight")
as.data.frame(t(MV_PF))
```

```
##                                               V1
## Mu_MV                                0.0319395807
## Sd_MV                                0.0160374182
## soybeans.No..1_weight                0.0765986471
## soybeans.No..2_weight                0.0326756746
## yellow.corn_weight                  -0.0344377668
## LLDPE_weight                         0.1648036033
## palm.oil_weight                     -0.0838180552
## soybean.oil_weight                   0.2365841107
## PVC_weight                          -0.1307315134
## cathode.copper_weight               -0.0890288755
## aluminum_weight                      0.1249637803
## zinc_weight                         -0.0126080941
## gold_weight                          0.1016810587
## natural.rubber_weight               -0.1133200768
## Gold_price_weight                   -0.0522855354
## Material_price_weight               -0.0965429281
## Industry_price_weight                0.0400729290
## Optional_consumption_price_weight   -0.0210382016
## Medicine_health_care_price_weight    0.0004652801
## Finance_and_real_estate_price_weight 0.0791534282
## Information_technology_price_weight  0.0237215941
## Utility_price_weight                 0.1031607913
## second.tier_weight                   0.3177368548
```

```
## third.tier_weight                     0.3321932949
```

```
# Now, let's visualize the relationship
feasible_pf %>%
  ggplot(aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(color = "grey") +
  # This is just to color in the "optimal PFs"
  geom_point(data = subset(feasible_pf, Portfolio_Risk <= 0.12 &
                              Portfolio_Return >= 0.02), color = "darkorchid3",
             shape = 1, aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.12 &
                              Portfolio_Risk <= 0.14 & Portfolio_Return >= 0.07),
             color = "darkorchid3", shape = 1,aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.14 &
                              Portfolio_Risk <= 0.16 & Portfolio_Return >= 0.09),
             color = "darkorchid3", shape = 1,aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.16 &
                              Portfolio_Risk <= 0.18 & Portfolio_Return >= 0.1),
             color = "darkorchid3", shape = 1,aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.18 &
                              Portfolio_Risk <= 0.20 & Portfolio_Return >= 0.11),
             color = "darkorchid3",shape = 1, aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  geom_point(data = subset(feasible_pf, Portfolio_Risk > 0.2 &
                              Portfolio_Risk <= 0.22 & Portfolio_Return >= 0.11),
             color = "darkorchid3", shape = 1, aes(x= Portfolio_Risk, y = Portfolio_Return)) +
  # Calculate and plot the Minimum Variance PF
  geom_point(color = "goldenrod",  aes(x= MV_PF$Sd_MV, y = MV_PF$Mu_MV),
             size = 3) +
  annotate('text',x = -0.01 ,y = 0, label = "Minimum Variance PF",
           size = 3.5, color = "goldenrod") +
  ylab(expression(mu[p])) + xlab(expression(sigma[p])) +
  ggtitle("Minimum Variance portfolio") +
  labs(color='Factor Portfolios') +
  xlim(-0.10, 0.75) +
  theme(plot.title= element_text(size=14, color="grey26",
  hjust=0.43,lineheight=2.4, margin=margin(15,0,15,0)),
  panel.background = element_rect(fill="#f7f7f7"),
  panel.grid.major.y = element_line(size = 0.5, linetype = "solid",color="grey"),
  panel.grid.minor = element_blank(),
  panel.grid.major.x = element_blank(),
  plot.background = element_rect(fill="#f7f7f7", color = "#f7f7f7"),
  axis.title.y = element_text(color="grey26", size=12, margin=margin(0,10,0,10)),
  axis.title.x = element_text(color="grey26", size=12, margin=margin(10,0,10,0)),
  axis.line = element_line(color = "grey"))
```

```
## Warning: Removed 4846 rows containing missing values (geom_point).
```



portfolio efficient frontier

```
# First we calculate the first Efficient Portfolio
# Define the EW return
mu_spec_x <- mu_MV
mu_spec_y <- 0.1

# We first define again the Matrix A
mat_A_EF <- rbind(cbind(2*cvar_ret, mean_ret,  rep(1,dim(cvar_ret)[1])),
                  cbind(t(mean_ret), 0, 0),
                  cbind(t(rep(1,dim(cvar_ret)[1])), 0, 0))

# Then, we define the vector b
vec_b_EF_x <- c(rep(0, dim(cvar_ret)[1]), mu_spec_x, 1)
vec_b_EF_y <- c(rep(0, dim(cvar_ret)[1]), mu_spec_y, 1)

# Now, we can solve for the respective weights
z_EF_x <- solve(mat_A_EF)%*%vec_b_EF_x
z_EF_y <- solve(mat_A_EF)%*%vec_b_EF_y

# Then, we take the first N elements to get the respective weights
x_EF <- z_EF_x[1:dim(cvar_ret)[1]]
y_EF <- z_EF_y[1:dim(cvar_ret)[1]]

# Now, let's calculate the risk
sd_EF_x <- sqrt(t(x_EF) %*% cvar_ret %*% x_EF)
sd_EF_y <- sqrt(t(y_EF) %*% cvar_ret %*% y_EF)
```

```r
sd_EF_xy <- t(x_EF) %*% cvar_ret %*% y_EF
```

```r
# Lastly, compute the weights and results
a = seq(from=0, to=1, by=0.05)
# Create the expected return as well as the variance and standard
# deviation of each portfolios
mu_z = a * mu_spec_x + (1-a) * mu_spec_y
```

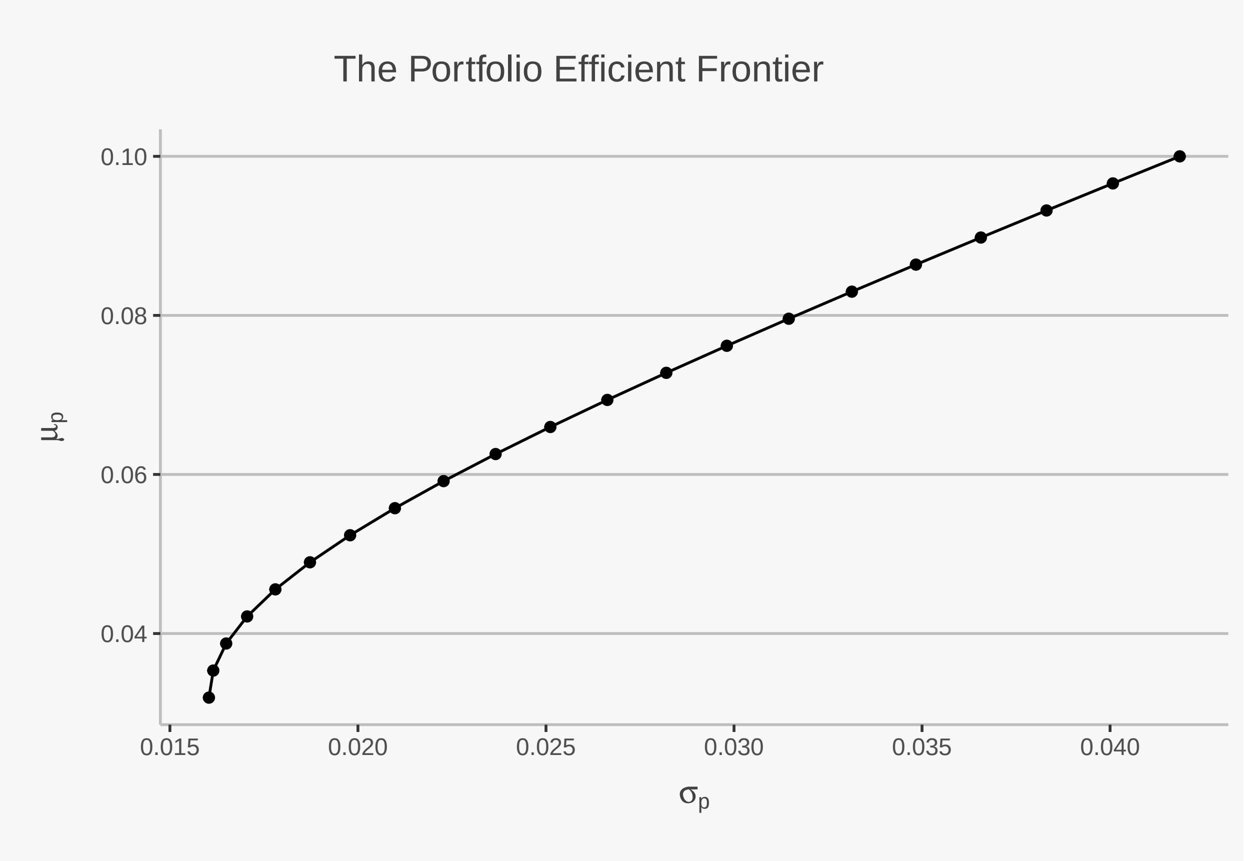
```
## Warning in a * mu_spec_x: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```r
sd_z = sqrt(a^2*sd_EF_x^2 + (1-a)^2*sd_EF_y^2 + 2*a*(1-a)*sd_EF_xy)
```

```
## Warning in a^2 * sd_EF_x^2: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in (1 - a)^2 * sd_EF_y^2: Recycling array of length 1 in vector-array arithmetic is deprecat
##   Use c() or as.vector() instead.

## Warning in 2 * a * (1 - a) * sd_EF_xy: Recycling array of length 1 in vector-array arithmetic is dep
##   Use c() or as.vector() instead.
```

```r
# Create a dataframe consisting of different weights
z = matrix(0, length(a), dim(asset_returns)[2])

for (i in 1:length(a)){
  z[i, ] = a[i]*x_EF + (1-a[i])*y_EF
  }

# Create a dataframe consisting of only the efficient linear transformation
# portfolios
z_df <- as.data.frame(cbind(mu_z, sd_z))
colnames(z_df) <- c("Efficient_PF_Return", "Efficient_PF_Risk")
```

```r
# Now, let's visualise the relationship
z_df %>%
  ggplot(aes(x= Efficient_PF_Risk, y = Efficient_PF_Return), color = "goldenrod") +
  geom_point() +
  geom_path() +
  ylab(expression(mu[p])) + xlab(expression(sigma[p])) +
  ggtitle("The Portfolio Efficient Frontier") +
  labs(color='Factor Portfolios') +
  theme(plot.title= element_text(size=14, color="grey26",
  hjust=0.3,lineheight=2.4, margin=margin(15,0,15,0)),
  panel.background = element_rect(fill="#f7f7f7"),
  panel.grid.major.y = element_line(size = 0.5, linetype="solid", color="grey"),
  panel.grid.minor = element_blank(),
  panel.grid.major.x = element_blank(),
  plot.background = element_rect(fill="#f7f7f7", color = "#f7f7f7"),
  axis.title.y = element_text(color="grey26", size=12, margin=margin(0,10,0,10)),
  axis.title.x = element_text(color="grey26", size=12, margin=margin(10,0,10,0)),
  axis.line = element_line(color = "grey"))
```




```r
# Calculate the TP
## HINT: Choose a similar risk free rate as in the exercise session.
### import the risk-free data set
rf <- DATA3[,2]
### calculate the TP
mu_f = mean(rf)
## First the numerator and denominator
numerator_T_N <- inv(cvar_ret) %*% (mean_ret - mu_f*rep(1, length(mean_ret)))
denominator_T_N <- t(rep(1, length(mean_ret))) %*%
  inv(cvar_ret) %*%(mean_ret - mu_f*rep(1, length(mean_ret)))
## calculate the weights
weights_T_N <- numerator_T_N[,1] / denominator_T_N
```

```
## Warning in numerator_T_N[, 1]/denominator_T_N: Recycling array of length 1 in vector-array arithmeti
##   Use c() or as.vector() instead.
```

```r
weights_T_N
```

```
##              soybeans.No..1              soybeans.No..2
##                 0.092002203                -0.563613294
##                 yellow.corn                       LLDPE
##                -0.018099874                -0.468700215
##                    palm.oil                 soybean.oil
##                 0.005661339                 0.445684816
##                         PVC              cathode.copper
```

```
##                   -0.376695617                   0.438619433
##                       aluminum                          zinc
##                    0.947920112                  -0.335028039
##                           gold                natural.rubber
##                    0.877784704                  -0.594614747
##                     Gold_price                Material_price
##                   -0.564797082                  -0.447550157
##                 Industry_price    Optional_consumption_price
##                    0.666089223                   0.464717306
##    Medicine_health_care_price Finance_and_real_estate_price
##                    0.675056516                   0.203704284
##  Information_technology_price                 Utility_price
##                   -0.813196211                  -0.625157650
##                    second.tier                    third.tier
##                    4.440973314                  -3.450760364
```

```
return_T_N <- weights_T_N %*% mean_ret
sd_T_N <- sqrt(t(weights_T_N) %*% cvar_ret %*% weights_T_N)
```

```
# Create the tangent portfolio
### define the risk of the risk-free asset and the covariance
### of it with risky assets
sigma_f = 0
sigma_Af = 0

## Define the sequence
x_tan = seq(from=-0.8, to=2.2, by=0.1)
x_f = 1 - x_tan

## Calculate the metrics
### Calculate the risk and return metrics
mu_T_N <- weights_T_N %*% mean_ret
sd_T_N <- sqrt(t(weights_T_N)%*%cvar_ret%*%weights_T_N)

### Create another dataframe
mu_sd_T_N_df <- as.data.frame(cbind(mu_T_N, sd_T_N))
colnames(mu_sd_T_N_df) <- c("mu_T_N", "sd_T_N")

mu_tanf = x_tan*mu_T_N + x_f*mu_f
```

```
## Warning in x_tan * mu_T_N: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
var_tanf = x_tan^2*sd_T_N^2 + x_f^2*sigma_f^2 + 2*x_tan*x_f*sigma_Af
```

```
## Warning in x_tan^2 * sd_T_N^2: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
sd_tanf = sqrt(var_tanf)

# Only get the "positive returns" from the r_f on
mu_tanf_real <- mu_tanf[1:9]
```

```r
sd_tanf_real <- sd_tanf[1:9]
# Create a df
cml_N <- as.data.frame(cbind(mu_tanf_real, sd_tanf_real))
colnames(cml_N) <- c("MU_CML", "SD_CML")
mu_sd_T_N_df
```

```
##      mu_T_N    sd_T_N
## 1 0.2083802 0.1014944
```

```r
port <- as.data.frame(cbind(Minimum_Variance = c(t(x_MV), mu_MV, sd_MV),
                            Tangency = c(weights_T_N, return_T_N, sd_T_N)))
rownames(port) <-c(colnames(asset_returns), "Return", "sd")
xtable(port, caption = "Weights of two portfolios", digits = 5)
```

```
## % latex table generated in R 4.1.2 by xtable 1.8-4 package
## % Mon Dec 19 11:35:49 2022
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrr}
##   \hline
##  & Minimum\_Variance & Tangency \\
##   \hline
## soybeans.No..1 & 0.07660 & 0.09200 \\
##   soybeans.No..2 & 0.03268 & -0.56361 \\
##   yellow.corn & -0.03444 & -0.01810 \\
##   LLDPE & 0.16480 & -0.46870 \\
##   palm.oil & -0.08382 & 0.00566 \\
##   soybean.oil & 0.23658 & 0.44568 \\
##   PVC & -0.13073 & -0.37670 \\
##   cathode.copper & -0.08903 & 0.43862 \\
##   aluminum & 0.12496 & 0.94792 \\
##   zinc & -0.01261 & -0.33503 \\
##   gold & 0.10168 & 0.87778 \\
##   natural.rubber & -0.11332 & -0.59461 \\
##   Gold\_price & -0.05229 & -0.56480 \\
##   Material\_price & -0.09654 & -0.44755 \\
##   Industry\_price & 0.04007 & 0.66609 \\
##   Optional\_consumption\_price & -0.02104 & 0.46472 \\
##   Medicine\_health\_care\_price & 0.00047 & 0.67506 \\
##   Finance\_and\_real\_estate\_price & 0.07915 & 0.20370 \\
##   Information\_technology\_price & 0.02372 & -0.81320 \\
##   Utility\_price & 0.10316 & -0.62516 \\
##   second.tier & 0.31774 & 4.44097 \\
##   third.tier & 0.33219 & -3.45076 \\
##   Return & 0.03194 & 0.20838 \\
##   sd & 0.01604 & 0.10149 \\
##    \hline
## \end{tabular}
## \caption{Weights of two portfolios}
## \end{table}
```