Project 7

Yaman Yucel

2023-05-17

PART A. Assume the multigroup model holds with short sales allowed. Find the composition of the optimal portfolio and its expected return and standard deviation and place it on the plot you constructed in previous projects with all the other portfolios and stocks. Note: Please see the numerical example of handout #37 for more details.

```
#Read all data
a_all <- read.csv("stockData.csv", sep=",", header=TRUE)

# Use 5 year data to train
a <- a_all[1:60,]

#Convert adjusted close prices into returns:
r <- (a[-1,3:ncol(a)]-a[-nrow(a),3:ncol(a)])/a[-nrow(a),3:ncol(a)] # return of stocks + market
#r_m <- (a[-1,3]-a[-nrow(a),3])/a[-nrow(a),3] # return of market
rrr <- r[,-1]

n_industries = 5
n_stock_in_industry = 6

cor_30 = cor(rrr)
var_stocks = var(rrr)
std_i_j = sqrt(diag(var_stocks))
cor_group = matrix(rep(0,n_industries*n_industries),n_industries,n_industries)
for (i in 1:n_industries){
  for (j in 1:n_industries){
    rho = 0

    if( i == j){
      for(k in 1:n_stock_in_industry)
      {
        for(l in 1:n_stock_in_industry)
        {
          rho = rho + cor_30[(i - 1)*n_stock_in_industry + k,( j - 1)*n_stock_in_industry + l]
         #print(c((i - 1)*n_stock_in_industry + k,( j - 1)*n_stock_in_industry + l))
        }
      }
      #print("NEXT")
      rho = (rho - n_stock_in_industry)/(n_stock_in_industry*(n_stock_in_industry-1))
      cor_group[i,j] = rho

    }
    else{
      for(k in 1:n_stock_in_industry)
```

```
      {
        for(l in 1:n_stock_in_industry)
        {
          rho = rho + cor_30[(i - 1)*n_stock_in_industry + k,( j - 1)*n_stock_in_industry + l]
          #print(c((i - 1)*n_stock_in_industry + k,( j - 1)*n_stock_in_industry + l))
        }
      }
      # print("NEXT")
      rho = (rho/(n_stock_in_industry*n_stock_in_industry))
      cor_group[i,j] = rho
    }
  }
}
#Construct covariance matrix using the rho's
cov_matrix_group = matrix(rep(0,30*30),30,30)
for (i in 1:n_industries){
  for (j in 1:n_industries){
    for(k in 1:n_stock_in_industry){
      for(l in 1:n_stock_in_industry){
        if(i== j && k == l){
          cov_matrix_group[(i - 1)*n_stock_in_industry + k,( j - 1)*n_stock_in_industry + l] = std_i_j[
          }
        else{
          cov_matrix_group[(i - 1)*n_stock_in_industry + k,( j - 1)*n_stock_in_industry + l] = cor_grou
        }
      }
    }
  }
}
# All stocks

#Compute mean vector:
means <- colMeans(r)
#Compute variance covariance matrix
covmat <- cov(r)
#Compute correlation matrix:
cormat <- cor(r)
#Compute the vector of variances:
variances <- diag(covmat)
#Compute the vector of standard deviations:
stdev <- diag(covmat)^.5
#Plot the 31 assets on the space expected return against standard deviation
plot(stdev, means,
     main="Expected Return against Standard Deviation",
     xlab="Standard Deviation",
     ylab="Expected Return",
     xlim = c(0, 0.3),
     ylim = c(-0.25, 0.25),
     col = "black",
     pch=19)

# Equal Allocation Portfolio
```

```r
new_means <- colMeans(rrr)
new_covmat <- cov(rrr)
new_cormat <- cor(rrr)
new_variances <- diag(new_covmat)
new_stdev <- diag(new_covmat)^.5

number_of_stocks = 30
ones_vector <- rep(1, number_of_stocks)
equal_weight_vector <- ones_vector/number_of_stocks # COMPOSITION HERE

equal_varp <- t(equal_weight_vector) %*% new_covmat %*% equal_weight_vector
equal_sdp <- sqrt(equal_varp)
equal_Rp <- t(equal_weight_vector) %*% new_means
points(equal_sdp, equal_Rp, pch = 19, lwd=1, col="red")

# Minimum Risk Portfolio

ones_vector <- rep(1, number_of_stocks)
inverse_new_covmat <- solve(new_covmat)
min_risk_weight_vector <- inverse_new_covmat  %*% ones_vector / as.numeric(t(ones_vector)  %*% inverse_n
#COMPOSITION MIN RISK
min_risk_varp <- t(min_risk_weight_vector) %*% new_covmat %*% min_risk_weight_vector
min_risk_sdp <- sqrt(min_risk_varp)
min_risk_Rp <- t(min_risk_weight_vector) %*% new_means

points(min_risk_sdp, min_risk_Rp, pch=19, lwd=1, col="green")

# Efficient Frontier

n_stocks = 30
#Compute mean vector:
means <- colMeans(rrr)
#Compute variance covariance matrix
covmat <- cov(rrr)
#Compute correlation matrix:
cormat <- cor(rrr)
#Compute the vector of variances:
variances <- diag(covmat)
#Compute the vector of standard deviations:
stdev <- diag(covmat)^.5
#Compute inverse of variance covariance matrix
inv_covmat <- solve(covmat)
#ones vector
ones = rep(1,n_stocks)

#Compute A,B,C,D
A = as.numeric(t(means) %*% inv_covmat %*% ones)
B = as.numeric(t(means) %*% inv_covmat %*% means)
C = as.numeric(t(ones)  %*% inv_covmat %*% ones)
D = B*C - A^2

#Compute lambda1 and lambda2
E = 0.025
```

```r
l1 = (function(x) (C*x - A) / D)
l2 = (function(x) (B - A*x) / D)

lambda_1 = l1(E)
lambda_2 = l2(E)

#Composition of the efficient portfolio given the return E
investor_weight = inv_covmat %*% (l1(E) * means + l2(E) * ones)

#Span values of E:
E <- seq(-0.2,0.2,.001)

#Compute variance of efficient frontier portfolios (parabola)
var_ef_p <- (C * E^2 - 2*A*E + B) / D

#Parabola: part d
points(sqrt(var_ef_p), E, type="l", ylab = 'E',col = "blue") # Parabola

r_gpsc = (a[-1,3]-a[-nrow(a),3])/a[-nrow(a),3]
gspc_mean = mean(r_gpsc)
gspc_var = var(r_gpsc)
points(sqrt(gspc_var),gspc_mean, pch = 19, lwd=1, col = "brown")

#Tangent historical Rf = 0.002

R_f = 0.002
R = means - R_f

Z = inv_covmat %*% R
x_G_historic = Z / sum(Z)

print("Composition of tangent")
```

```
## [1] "Composition of tangent"
```

```r
print(x_G_historic)
```

```
##              [,1]
## AAPL  -0.61302599
## MSFT   0.22306449
## NVDA   0.26372275
## TSM    0.50835452
## ASML   0.42626347
## AVGO   0.01720819
## GOOGL -0.06385573
## META   0.30565850
## DIS    0.29701980
## TMUS   0.45727592
## VZ    -0.00943165
## CMCSA -0.76697726
## AMZN   0.17507402
## TSLA   0.04306091
## HD    -0.30503140
## BABA  -0.26813873
```

```
## MCD    0.84672527
## TM    -0.95420760
## WMT   -0.40381782
## PG     0.95450570
## KO    -0.90641933
## PEP   -0.08484008
## COST   0.43941270
## FMX   -0.11368787
## BHP   -0.05363246
## LIN    0.98487505
## RIO   -0.06422106
## VALE   0.04730955
## APD   -0.15130260
## SCCO  -0.23094127
```

```
varg_historic <- t(x_G_historic) %*% covmat %*% x_G_historic
Rg_historic <- t(x_G_historic) %*% means
sigmag_historic <- sqrt(varg_historic)
points(sigmag_historic,Rg_historic, pch=19,lwd=1,col="darkgreen")
abline(a = R_f, b = (Rg_historic - R_f)/sigmag_historic , lwd = 1, col = "firebrick")

#Single Index Model

r_m <- (a[-1,3]-a[-nrow(a),3])/a[-nrow(a),3]
n_stocks = 30
mean_Rm = mean(r_m)
var_Rm <- var(r_m)
stdev_Rm <- var_Rm^.5

mean_Ri = colMeans(rrr)

betas = rep(0,n_stocks)
alphas = rep(0,n_stocks)
var_es = rep(0,n_stocks)
var_betas = rep(0,n_stocks)

for (i in 1:n_stocks){
  fit <- lm(rrr[,i] ~ r_m)
  betas[i] = fit$coefficients[2]
  alphas[i] = fit$coefficients[1]
  var_es[i] = sum(fit$residuals^2)/ (nrow(rrr) - 2)
  var_betas[i] = vcov(fit)[2,2]
}

#find beta_i
print("Betas are:")
```

```
## [1] "Betas are:"
```

```
print(betas)
```

```
##  [1] 1.2929549 1.1593250 2.0248862 1.0718453 1.2582033 0.9442843 1.0393931
##  [8] 1.0530930 1.0025766 0.4342799 0.4776197 1.0354303 1.6022329 0.6063894
## [15] 1.0001717 2.2070856 0.4415925 0.7638836 0.3725548 0.3437258 0.4144253
## [22] 0.5458005 0.8674676 0.5747603 0.9214674 0.7875896 0.9919519 1.1254699
```

```
## [29] 0.8974680 0.9648233
```

```r
print("Number of negative betas")
```

```
## [1] "Number of negative betas"
```

```r
print(sum(which(betas < 0)))
```

```
## [1] 0
```

```r
#find alpha_i
print("Alphas are:")
```

```
## [1] "Alphas are:"
```

```r
print(alphas)
```

```
##  [1]  8.655393e-03  1.667838e-02  3.383866e-02  1.181871e-02  9.787943e-03
##  [6]  1.609522e-02  8.128193e-03  1.012012e-02  1.934431e-03  1.388844e-02
## [11]  5.867074e-03  3.144260e-03  1.750136e-02  1.420789e-02  6.910948e-03
## [16]  8.588360e-04  1.230233e-02 -3.223398e-03  5.838137e-03  7.126095e-03
## [21]  4.799198e-03  4.797339e-03  8.091456e-03  3.196362e-05  3.180270e-03
## [26]  5.679208e-03  5.579035e-03  1.433911e-02  4.663745e-03  4.477371e-03
```

```r
#Compute covariance matrix using single index model
covariance_matrix_sim = matrix(0,n_stocks,n_stocks)
for (i in 1:n_stocks)
{
  for(j in 1:n_stocks){
    if(i == j)
    {
      covariance_matrix_sim[i,j] = betas[i] * betas[i] * var_Rm + var_es[i]
    }else{
      covariance_matrix_sim[i,j] = betas[i] * betas[j] * var_Rm
    }
  }
}

inv_covmat_single_index = solve(covariance_matrix_sim)
ones = rep(1,n_stocks)

A = as.numeric(t(means) %*% inv_covmat_single_index %*% ones)
B = as.numeric(t(means) %*% inv_covmat_single_index %*% means)
C = as.numeric(t(ones)  %*% inv_covmat_single_index %*% ones)
D = B*C - A^2

E <- seq(-0.2,0.2,.001)

sigmas <- sqrt(seq(1/C,0.03,.0001))
upper_part <- (A + sqrt(D*(C*sigmas^2 - 1)))*(1/C)
lower_part <- (A - sqrt(D*(C*sigmas^2 - 1)))*(1/C)

lines(sigmas, upper_part, lwd=1,type = "l",col = "orange",xlim = c(0, 0.2), ylim= c(-0.2,0.2))
lines(sigmas,lower_part, lwd=1,type = "l",col = "orange")

R_f = 0.002
R = means - R_f
```

```r
Z = inv_covmat_single_index %*% R
x_G_sim = Z / sum(Z)

print("Composition of tangent")
```

```
## [1] "Composition of tangent"
```

```r
print(x_G_sim)
```

```
##               [,1]
##  [1,]  0.010937796
##  [2,]  0.319321021
##  [3,]  0.131841756
##  [4,]  0.101903830
##  [5,]  0.053551902
##  [6,]  0.113994968
##  [7,]  0.037017266
##  [8,]  0.055757221
##  [9,] -0.120504236
## [10,]  0.240965202
## [11,]  0.049880298
## [12,] -0.107303614
## [13,]  0.141226189
## [14,]  0.041272294
## [15,]  0.012135308
## [16,] -0.123990706
## [17,]  0.352251678
## [18,] -0.447051001
## [19,]  0.053015460
## [20,]  0.161050368
## [21,]  0.052121839
## [22,]  0.013297532
## [23,]  0.077646359
## [24,] -0.102002979
## [25,] -0.029028214
## [26,]  0.001020932
## [27,] -0.010733009
## [28,]  0.020750892
## [29,] -0.077968692
## [30,] -0.022377660
```

```r
varg_sim <- t(x_G_sim) %*% covariance_matrix_sim %*% x_G_sim
Rg_sim <- t(x_G_sim) %*% means
sigmag_sim <- sqrt(varg_sim)
points(sigmag_sim,Rg_sim, pch=19,lwd=1,col="blue")
abline(a = R_f, b = (Rg_sim - R_f)/sigmag_sim , lwd = 1, col = "red")

#Short sales allowed SIM

a_all <- read.csv("stockData.csv", sep=",", header=TRUE)

# Use 5 year data to train
a <- a_all[1:60,]
```

```r
#Convert adjusted close prices into returns:
r <- (a[-1,3:ncol(a)]-a[-nrow(a),3:ncol(a)])/a[-nrow(a),3:ncol(a)] # return of stocks + market
#r_m <- (a[-1,3]-a[-nrow(a),3])/a[-nrow(a),3] # return of market

n_stocks = 30

covmat <- var(r)
beta <- covmat[1,-1]/ covmat[1,1]

rrr <- r[,-c(1,which(beta<0)+1)]


beta <- rep(0,ncol(rrr))
alpha <- rep(0,ncol(rrr))
mse <- rep(0,ncol(rrr))
Ribar <- rep(0,ncol(rrr))
Ratio <- rep(0,ncol(rrr))
stock <- rep(0,ncol(rrr))

rf <- 0.002


for(i in 1:ncol(rrr)){
  q <- lm(data=rrr, formula=rrr[,i] ~ r[,1])
  beta[i] <- q$coefficients[2]
  alpha[i] <- q$coefficients[1]
  mse[i] <- summary(q)$sigma^2
  Ribar[i] <- q$coefficients[1]+q$coefficients[2]*mean(r[,1])
  Ratio[i] <- (Ribar[i]-rf)/beta[i]
  stock[i] <- i
}

xx <- (cbind(stock,alpha, beta, Ribar, mse, Ratio))

A <- xx[order(-xx[,6]),]

col1 <- rep(0,nrow(A))
col2 <- rep(0,nrow(A))
col3 <- rep(0,nrow(A))
col4 <- rep(0,nrow(A))
col5 <- rep(0,nrow(A))

col1 <- (A[,4]-rf)*A[,3]/A[,5]
col3 <- A[,3]^2/A[,5]
for(i in(1:nrow(A))) {
  col2[i] <- sum(col1[1:i])
  col4[i] <- sum(col3[1:i])
}

#Compute the Ci (col5):
for(i in (1:nrow(A))) {
  col5[i] <- var(r[,1])*col2[i]/(1+var(r[,1])*col4[i])
}
```

```
#SHORT SALES ALLOWED:
#Compute the zi:
z_short <- (A[,3]/A[,5])*(A[,6]-col5[nrow(A)])
#Compute the xi:
x_short <- z_short/sum(z_short)

#The final table when short sales allowed:
Weights_with_short <- cbind(A, col1, col2, col3, col4, col5, z_short, x_short)
print(Weights_with_short)
```

```
##       stock         alpha      beta       Ribar         mse       Ratio
##  [1,]    10  1.388844e-02 0.4342799 0.017706285 0.002683414 0.036166273
##  [2,]    17  1.230233e-02 0.4415925 0.016184457 0.001534396 0.032121146
##  [3,]    14  1.420789e-02 0.6063894 0.019538775 0.014905473 0.028923287
##  [4,]     3  3.383866e-02 2.0248862 0.051639839 0.011146357 0.024514878
##  [5,]     6  1.609522e-02 0.9442843 0.024396608 0.005579442 0.023718078
##  [6,]    20  7.126095e-03 0.3437258 0.010147857 0.001435659 0.023704525
##  [7,]     2  1.667838e-02 1.1593250 0.026870239 0.001906601 0.021452345
##  [8,]    28  1.433911e-02 1.1254699 0.024233340 0.022451618 0.019754717
##  [9,]    19  5.838137e-03 0.3725548 0.009113340 0.002604566 0.019093408
## [10,]    13  1.750136e-02 1.6022329 0.031586905 0.003738721 0.018466046
## [11,]     4  1.181871e-02 1.0718453 0.021241516 0.003111891 0.017951765
## [12,]    11  5.867074e-03 0.4776197 0.010065923 0.002165591 0.016887753
## [13,]     8  1.012012e-02 1.0530930 0.019378066 0.003794271 0.016501929
## [14,]    23  8.091456e-03 0.8674676 0.015717535 0.001740458 0.015813312
## [15,]    21  4.799198e-03 0.4144253 0.008442493 0.001099256 0.015545609
## [16,]     5  9.787943e-03 1.2582033 0.020849058 0.002379235 0.014980932
## [17,]     7  8.128193e-03 1.0393931 0.017265705 0.002303044 0.014687132
## [18,]     1  8.655393e-03 1.2929549 0.020022016 0.003900169 0.013938627
## [19,]    22  4.797339e-03 0.5458005 0.009595579 0.001294483 0.013916404
## [20,]    15  6.910948e-03 1.0001717 0.015703656 0.001438093 0.013701304
## [21,]    26  5.679208e-03 0.7875896 0.012603065 0.001402875 0.013462678
## [22,]    27  5.579035e-03 0.9919519 0.014299481 0.006269398 0.012399272
## [23,]    29  4.663745e-03 0.8974680 0.012553564 0.001263365 0.011759265
## [24,]    30  4.477371e-03 0.9648233 0.012959325 0.005862893 0.011358893
## [25,]    25  3.180270e-03 0.9214674 0.011281073 0.006992219 0.010072058
## [26,]    12  3.144260e-03 1.0354303 0.012246933 0.002236589 0.009896304
## [27,]     9  1.934431e-03 1.0025766 0.010748281 0.002566266 0.008725798
## [28,]    16  8.588360e-04 2.2070856 0.020261764 0.006017150 0.008274153
## [29,]    24  3.196362e-05 0.5747603 0.005084795 0.002977672 0.005367099
## [30,]    18 -3.223398e-03 0.7638836 0.003492054 0.001285056 0.001953248
##             col1      col2      col3       col4        col5     z_short
##  [1,]  2.5418822  2.541882  70.28322   70.28322 0.002786044  3.67880990
##  [2,]  4.0822262  6.624108 127.08843  197.37166 0.006372699  5.37781783
##  [3,]  0.7135183  7.337627  24.66934  222.04099 0.006895485  0.63010311
##  [4,]  9.0177464 16.355373 367.84789  589.88889 0.011421603  2.01282489
##  [5,]  3.7904802 20.145853 159.81397  749.70286 0.012656161  1.74035841
##  [6,]  1.9507615 22.096615  82.29490  831.99776 0.013199281  2.45875206
##  [7,] 15.1225588 37.219174 704.93733 1536.93508 0.015644786  4.87506629
##  [8,]  1.1145280 38.333701  56.41832 1593.35340 0.015739995  0.31680336
##  [9,]  1.0174857 39.351187  53.28989 1646.64329 0.015811800  0.80938574
## [10,] 12.6795005 52.030688 686.63863 2333.28193 0.016385753  2.15609681
## [11,]  6.6274593 58.658147 369.18149 2702.46341 0.016548861  1.55576331
```

```
## [12,]  1.7789341  60.437081 105.33870 2807.80212 0.016558642  0.76152130
## [13,]  4.8232503  65.260331 292.28404 3100.08616 0.016554437  0.85124414
## [14,]  6.8370035  72.097335 432.35747 3532.44362 0.016481188  1.18542509
## [15,]  2.4288541  74.526189 156.24053 3688.68416 0.016448925  0.79574286
## [16,]  9.9678867  84.494076 665.37159 4354.05574 0.016260946  0.81757560
## [17,]  6.8896059  91.383682 469.09130 4823.14704 0.016130632  0.56514170
## [18,]  5.9745241  97.358206 428.63074 5251.77778 0.015976450  0.16698706
## [19,]  3.2025689 100.560775 230.12906 5481.90684 0.015901485  0.20301310
## [20,]  9.5306816 110.091456 695.60400 6177.51084 0.015683459  0.18526946
## [21,]  5.9526805 116.044137 442.16170 6619.67254 0.015551862  0.01558655
## [22,]  1.9460392 117.990176 156.94785 6776.62040 0.015486918 -0.16386059
## [23,]  7.4970299 125.487206 637.54238 7414.16278 0.015199070 -1.19034613
## [24,]  1.8035144 127.290720 158.77554 7572.93832 0.015126613 -0.34163919
## [25,]  1.2231032 128.513823 121.43529 7694.37361 0.015054710 -0.44317304
## [26,]  4.7438232 133.257647 479.35301 8173.72662 0.014780447 -1.63820168
## [27,]  3.4177369 136.675384 391.68186 8565.40848 0.014528361 -1.83973526
## [28,]  6.6984001 143.373784 809.55719 9374.96567 0.014032803 -1.89296310
## [29,]  0.5954376 143.969221 110.94217 9485.90785 0.013939717 -1.55727701
## [30,]  0.8869307 144.856152 454.07995 9939.98779 0.013434915 -6.82511681
##           x_short
##  [1,]  0.240965202
##  [2,]  0.352251678
##  [3,]  0.041272294
##  [4,]  0.131841756
##  [5,]  0.113994968
##  [6,]  0.161050368
##  [7,]  0.319321021
##  [8,]  0.020750892
##  [9,]  0.053015460
## [10,]  0.141226189
## [11,]  0.101903830
## [12,]  0.049880298
## [13,]  0.055757221
## [14,]  0.077646359
## [15,]  0.052121839
## [16,]  0.053551902
## [17,]  0.037017266
## [18,]  0.010937796
## [19,]  0.013297532
## [20,]  0.012135308
## [21,]  0.001020932
## [22,] -0.010733009
## [23,] -0.077968692
## [24,] -0.022377660
## [25,] -0.029028214
## [26,] -0.107303614
## [27,] -0.120504236
## [28,] -0.123990706
## [29,] -0.102002979
## [30,] -0.447051001
```

```
#SHORT SALES NOT ALLOWED:
#First create a matrix up to the maximum of col5:
table1 <- cbind(A, col1, col2, col3, col4, col5)
```

```r
table2 <- table1[1:which(col5==max(col5)), ]

#Compute the zi:
z_no_short <- (table2[,3]/table2[,5])*(table2[,6]-max(col5))

#Compute the xi:
x_no_short <- z_no_short/sum(z_no_short)

#The final table when short sales are not allowed:
Weights_no_short <- cbind(table2, z_no_short, x_no_short)
print(Weights_no_short)
```

```
##       stock       alpha      beta      Ribar         mse      Ratio       col1
##  [1,]    10 0.013888444 0.4342799 0.01770628 0.002683414 0.03616627  2.5418822
##  [2,]    17 0.012302330 0.4415925 0.01618446 0.001534396 0.03212115  4.0822262
##  [3,]    14 0.014207885 0.6063894 0.01953878 0.014905473 0.02892329  0.7135183
##  [4,]     3 0.033838662 2.0248862 0.05163984 0.011146357 0.02451488  9.0177464
##  [5,]     6 0.016095218 0.9442843 0.02439661 0.005579442 0.02371808  3.7904802
##  [6,]    20 0.007126095 0.3437258 0.01014786 0.001435659 0.02370453  1.9507615
##  [7,]     2 0.016678383 1.1593250 0.02687024 0.001906601 0.02145235 15.1225588
##  [8,]    28 0.014339111 1.1254699 0.02423334 0.022451618 0.01975472  1.1145280
##  [9,]    19 0.005838137 0.3725548 0.00911334 0.002604566 0.01909341  1.0174857
## [10,]    13 0.017501358 1.6022329 0.03158691 0.003738721 0.01846605 12.6795005
## [11,]     4 0.011818710 1.0718453 0.02124152 0.003111891 0.01795177  6.6274593
## [12,]    11 0.005867074 0.4776197 0.01006592 0.002165591 0.01688775  1.7789341
##            col2      col3       col4        col5 z_no_short  x_no_short
##  [1,]  2.541882  70.28322   70.28322 0.002786044 3.17327045 0.182462891
##  [2,]  6.624108 127.08843  197.37166 0.006372699 4.47882246 0.257532066
##  [3,]  7.337627  24.66934  222.04099 0.006895485 0.50302260 0.028923774
##  [4,] 16.355373 367.84789  589.88889 0.011421603 1.44535770 0.083107995
##  [5,] 20.145853 159.81397  749.70286 0.012656161 1.21168792 0.069671994
##  [6,] 22.096615  82.29490  831.99776 0.013199281 1.71086871 0.098374864
##  [7,] 37.219174 704.93733 1536.93508 0.015644786 2.97565759 0.171100162
##  [8,] 38.333701  56.41832 1593.35340 0.015739995 0.16021505 0.009212357
##  [9,] 39.351187  53.28989 1646.64329 0.015811800 0.36257066 0.020847795
## [10,] 52.030688 686.63863 2333.28193 0.016385753 0.81742011 0.047001615
## [11,] 58.658147 369.18149 2702.46341 0.016548861 0.47984103 0.027590835
## [12,] 60.437081 105.33870 2807.80212 0.016558642 0.07258531 0.004173652
```

```r
#find the return of the portfolio with short sales allowed

R_p_short <- Weights_with_short[,13] %*% Weights_with_short[,4]

covariance_matrix_ss = matrix(0,n_stocks,n_stocks)
var_Rm = var(r[,1])
for (i in 1:n_stocks)
{
  for(j in 1:n_stocks){
    if(i == j)
    {
      covariance_matrix_ss[i,j] = Weights_with_short[i,3] * Weights_with_short[i,3] * var_Rm + Weights_w
    }else{
      covariance_matrix_ss[i,j] = Weights_with_short[i,3] * Weights_with_short[j,3] * var_Rm
    }
```

```
  }
}

#find the risk of the portfolio with short sales allowed

var_p_short <- Weights_with_short[,13] %*% covariance_matrix_ss %*% Weights_with_short[,13]

#find the return of the portfolio with no short sales allowed
n_long = nrow(Weights_no_short)
R_p_no_short <- Weights_no_short[1:n_long,13] %*% Weights_no_short[1:n_long,4]
#find the risk of the portfolio with no short sales allowed
var_p_no_short <- Weights_no_short[1:n_long,13] %*% covariance_matrix_ss[1:n_long,1:n_long]  %*% Weights

points(sqrt(var_p_short),R_p_short, pch=19,lwd=1,col="dimgray")
points(sqrt(var_p_no_short),R_p_no_short, pch=19,lwd=1,col="gold")


#Const Corr Model

#Read all data
a_all <- read.csv("stockData.csv", sep=",", header=TRUE)

# Use 5 year data to train
a <- a_all[1:60,]

#Convert adjusted close prices into returns:
r <- (a[-1,3:ncol(a)]-a[-nrow(a),3:ncol(a)])/a[-nrow(a),3:ncol(a)] # return of stocks + market
#r_m <- (a[-1,3]-a[-nrow(a),3])/a[-nrow(a),3] # return of market
rrr <- r[,-1]

n_stocks= ncol(rrr)

#Compute the average correlation:
rho <- (sum(cor(rrr[1:n_stocks]))-n_stocks)/(n_stocks*(n_stocks-1))

#Initialize the vectors:
col1 <- rep(0,n_stocks)
col2 <- rep(0,n_stocks)
col3 <- rep(0,n_stocks)

#Initialize the var-covar matrix:
y <- rep(0,n_stocks*n_stocks)
mat <- matrix(y, ncol=n_stocks, nrow=n_stocks)

#Compute necessary quantities:
R_f = 0.002
Rbar <- colMeans(rrr[1:n_stocks])
Rbar_f <- Rbar-R_f
sigma <- ( diag(var(rrr[1:n_stocks])) )^0.5
Ratio <- Rbar_f/sigma

#Initial table:
xx <- (cbind(seq(1,30),Rbar, Rbar_f, sigma, Ratio))
```

```r
#Order the table based on the excess return to sigma ratio:
aaa <- xx[order(-Ratio),]

#Create the last 3 columns of the table:
for(i in(1:n_stocks)) {

  col1[i] <- rho/(1-rho+i*rho)

  col2[i] <- sum(aaa[,5][1:i])
}

#Compute the Ci:
for(i in (1:n_stocks)) {

  col3[i] <- col1[i]*col2[i]

}

#Create the entire table until now:
xxx <- cbind(aaa, col1, col2, col3)


#SHORT SALES ALLOWED:
#Compute the Zi:
z <- (1/((1-rho)*xxx[,4]))*(xxx[,5]-xxx[,8][nrow(xxx)])

#Compute the xi:
x <- z/sum(z)

#The final table:
aaaa <- cbind(xxx, z, x)
print(aaaa)
```

```
##                Rbar      Rbar_f      sigma      Ratio       col1      col2
## MSFT   2 0.026870239 0.024870239 0.05890517 0.42220812 0.25943203 0.4222081
## NVDA   3 0.051639839 0.049639839 0.12579059 0.39462282 0.20599129 0.8168309
## AMZN  13 0.031586905 0.029586905 0.08199295 0.36084693 0.17080662 1.1776779
## MCD   17 0.016184457 0.014184457 0.04170749 0.34009376 0.14588798 1.5177716
## TMUS  10 0.017706285 0.015706285 0.05348941 0.29363355 0.12731435 1.8114052
## ASML   5 0.020849058 0.018849058 0.06494747 0.29022006 0.11293598 2.1016252
## TSM    4 0.021241516 0.019241516 0.06650219 0.28933657 0.10147572 2.3909618
## AVGO   6 0.024396608 0.022396608 0.08088349 0.27689963 0.09212706 2.6678614
## COST  23 0.015717535 0.013717535 0.05103010 0.26881262 0.08435562 2.9366740
## HD    15 0.015703656 0.013703656 0.05100238 0.26868660 0.07779332 3.2053606
## GOOGL  7 0.017265705 0.015265705 0.05955063 0.25634833 0.07217833 3.4617090
## META   8 0.019378066 0.017378066 0.07103405 0.24464416 0.06731933 3.7063531
## AAPL   1 0.020022016 0.018022016 0.07627691 0.23627091 0.06307328 3.9426240
## LIN   26 0.012603065 0.010603065 0.04599253 0.23053885 0.05933107 4.1731629
## APD   29 0.012553564 0.010553564 0.04688371 0.22510085 0.05600806 4.3982637
## PG    20 0.010147857 0.008147857 0.03938542 0.20687499 0.05303753 4.6051387
## PEP   22 0.009595579 0.007595579 0.04032281 0.18836929 0.05036623 4.7935080
## KO    21 0.008442493 0.006442493 0.03583658 0.17977422 0.04795111 4.9732822
## CMCSA 12 0.012246933 0.010246933 0.05891692 0.17392172 0.04575701 5.1472040
```

```
## BABA  16 0.020261764 0.018261764 0.10815806 0.16884331 0.04375492 5.3160473
## VZ    11 0.010065923 0.008065923 0.04898117 0.16467397 0.04192068 5.4807212
## VALE  28 0.024233340 0.022233340 0.15352118 0.14482263 0.04023404 5.6255439
## RIO   27 0.014299481 0.012299481 0.08561427 0.14366158 0.03867788 5.7692054
## DIS    9 0.010748281 0.008748281 0.06095641 0.14351700 0.03723760 5.9127224
## TSLA  14 0.019538775 0.017538775 0.12282162 0.14279877 0.03590075 6.0555212
## WMT   19 0.009113340 0.007113340 0.05219661 0.13627973 0.03465655 6.1918009
## SCCO  30 0.012959325 0.010959325 0.08286898 0.13224882 0.03349570 6.3240498
## BHP   25 0.011281073 0.009281073 0.08876938 0.10455263 0.03241011 6.4286024
## FMX   24 0.005084795 0.003084795 0.05760759 0.05354842 0.03139267 6.4821508
## TM    18 0.003492054 0.001492054 0.04422498 0.03373781 0.03043716 6.5158886
##            col3          z           x
## MSFT  0.1095343  5.1321900  0.42710370
## NVDA  0.1682601  2.1071820  0.17536085
## AMZN  0.2011552  2.6765175  0.22274127
## MCD   0.2214246  4.5898773  0.38197213
## TMUS  0.2306179  2.4060153  0.20022993
## ASML  0.2373491  1.9105757  0.15899917
## TSM   0.2426246  1.8479702  0.15378910
## AVGO  0.2457822  1.3117668  0.10916596
## COST  0.2477250  1.8651791  0.15522123
## HD    0.2493556  1.8628562  0.15502792
## GOOGL 0.2498604  1.3156796  0.10949158
## META  0.2495092  0.8804966  0.07327541
## AAPL  0.2486742  0.6717461  0.05590308
## LIN   0.2475982  0.9457761  0.07870801
## APD   0.2463382  0.7711764  0.06417773
## PG    0.2442452  0.2931278  0.02439426
## PEP   0.2414309 -0.3333983 -0.02774559
## KO    0.2384744 -0.6989957 -0.05817081
## CMCSA 0.2355207 -0.5593016 -0.04654539
## BABA  0.2326032 -0.3680704 -0.03063102
## VZ    0.2297556 -0.9276974 -0.07720349
## VALE  0.2263384 -0.4705883 -0.03916262
## RIO   0.2231406 -0.8621582 -0.07174928
## DIS   0.2201756 -1.2141181 -0.10103958
## TSLA  0.2173977 -0.6104635 -0.05080311
## WMT   0.2145865 -1.6051017 -0.13357745
## SCCO  0.2118285 -1.0766859 -0.08960240
## BHP   0.2083517 -1.4264202 -0.11870748
## FMX   0.2034920 -3.3935492 -0.28241305
## TM    0.1983252 -5.0253219 -0.41821007
```

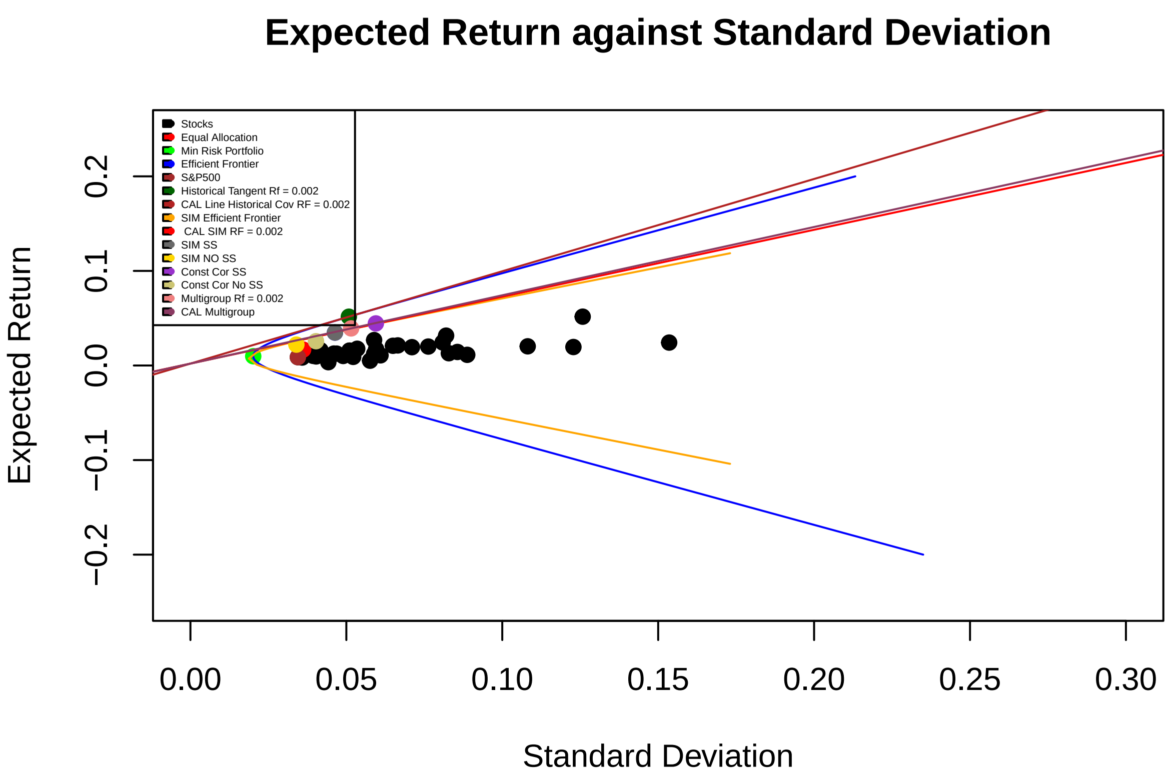
```r
#SHORT SALES NOT ALLOWED:
#Find composition of optimum portfolio when short sales are not allowed:
aaaaa <- aaaa[1:which(aaaa[,8]==max(aaaa[,8])), ]
z_no <- (1/((1-rho)*aaaaa[,4]))*(aaaaa[,5]-aaaaa[,8][nrow(aaaaa)])
x_no <- z_no/sum(z_no)

#Final table:
a_no <- cbind(aaaaa, z_no, x_no)
print(a_no)
```

```
##              Rbar     Rbar_f      sigma     Ratio       col1      col2
## MSFT   2 0.02687024 0.02487024 0.05890517 0.4222081 0.25943203 0.4222081
```

```
## NVDA   3 0.05163984 0.04963984 0.12579059 0.3946228 0.20599129 0.8168309
## AMZN  13 0.03158691 0.02958691 0.08199295 0.3608469 0.17080662 1.1776779
## MCD   17 0.01618446 0.01418446 0.04170749 0.3400938 0.14588798 1.5177716
## TMUS  10 0.01770628 0.01570628 0.05348941 0.2936335 0.12731435 1.8114052
## ASML   5 0.02084906 0.01884906 0.06494747 0.2902201 0.11293598 2.1016252
## TSM    4 0.02124152 0.01924152 0.06650219 0.2893366 0.10147572 2.3909618
## AVGO   6 0.02439661 0.02239661 0.08088349 0.2768996 0.09212706 2.6678614
## COST  23 0.01571754 0.01371754 0.05103010 0.2688126 0.08435562 2.9366740
## HD    15 0.01570366 0.01370366 0.05100238 0.2686866 0.07779332 3.2053606
## GOOGL  7 0.01726570 0.01526570 0.05955063 0.2563483 0.07217833 3.4617090
##            col3        z         x      z_no       x_no
## MSFT  0.1095343 5.132190 0.4271037 3.9508206 0.27063873
## NVDA  0.1682601 2.107182 0.1753609 1.5539708 0.10644996
## AMZN  0.2011552 2.676518 0.2227413 1.8278012 0.12520786
## MCD   0.2214246 4.589877 0.3819721 2.9213818 0.20012021
## TMUS  0.2306179 2.406015 0.2002299 1.1050333 0.07569688
## ASML  0.2373491 1.910576 0.1589992 0.8391135 0.05748087
## TSM   0.2426246 1.847970 0.1537891 0.8015571 0.05490819
## AVGO  0.2457822 1.311767 0.1091660 0.4514088 0.03092236
## COST  0.2477250 1.865179 0.1552212 0.5014984 0.03435360
## HD    0.2493556 1.862856 0.1550279 0.4984344 0.03414371
## GOOGL 0.2498604 1.315680 0.1094916 0.1471150 0.01007766
```

```
#Var-covar matrix based on the constant correlation model:
for(i in 1:30){

  for(j in 1:30){

    if(i==j){
      mat[i,j]=aaaa[i,4]^2
    } else
    {
      mat[i,j]=rho*aaaa[i,4]*aaaa[j,4]
    }
  }
}


#Calculate the expected return and sd of the point of tangency
#when short sales allowed
sd_p_opt <- (t(x) %*% mat %*% x)^.5
R_p_opt <- t(x) %*% aaaa[,2]


#Calculate the expected return and sd of the point of tangency
#when short sales are not allowed
sd_p_opt_no <- (t(x_no) %*% mat[1:which(aaaa[,8]==max(aaaa[,8])),1:which(aaaa[,8]==max(aaaa[,8]))] %*% 
R_p_opt_no <- t(x_no) %*% aaaaa[,2]

#Trace out efficient portfolio
inv_covmat_const_corr= solve(mat)
ones = rep(1,n_stocks)

points(sd_p_opt,R_p_opt, pch=19,lwd=1,col="darkorchid")
```

```
points(sd_p_opt_no,R_p_opt_no, pch=19,lwd=1,col="khaki3")

#MULTIGROUP MODEL

R_f = 0.002
R = means - R_f

Z = solve(cov_matrix_group) %*% R
x_G_group = Z / sum(Z)


varg_group <- t(x_G_group) %*% cov_matrix_group %*% x_G_group
Rg_group <- t(x_G_group) %*% means
sigmag_group <- sqrt(varg_group)
points(sigmag_group,Rg_group, pch=19,lwd=1,col="lightcoral")
abline(a = R_f, b = (Rg_group - R_f)/sigmag_group , lwd = 1, col = "hotpink4")

legend("topleft",
       legend=c("Stocks", "Equal Allocation","Min Risk Portfolio","Efficient Frontier","S&P500", "Histor
       col=c("black","red","green","blue","brown","darkgreen","firebrick","orange","red","dimgray","gold
       pch = 19,
       fill =c("black","red","green","blue","brown","darkgreen","firebrick","orange","red","dimgray","go
       cex=0.35)
```



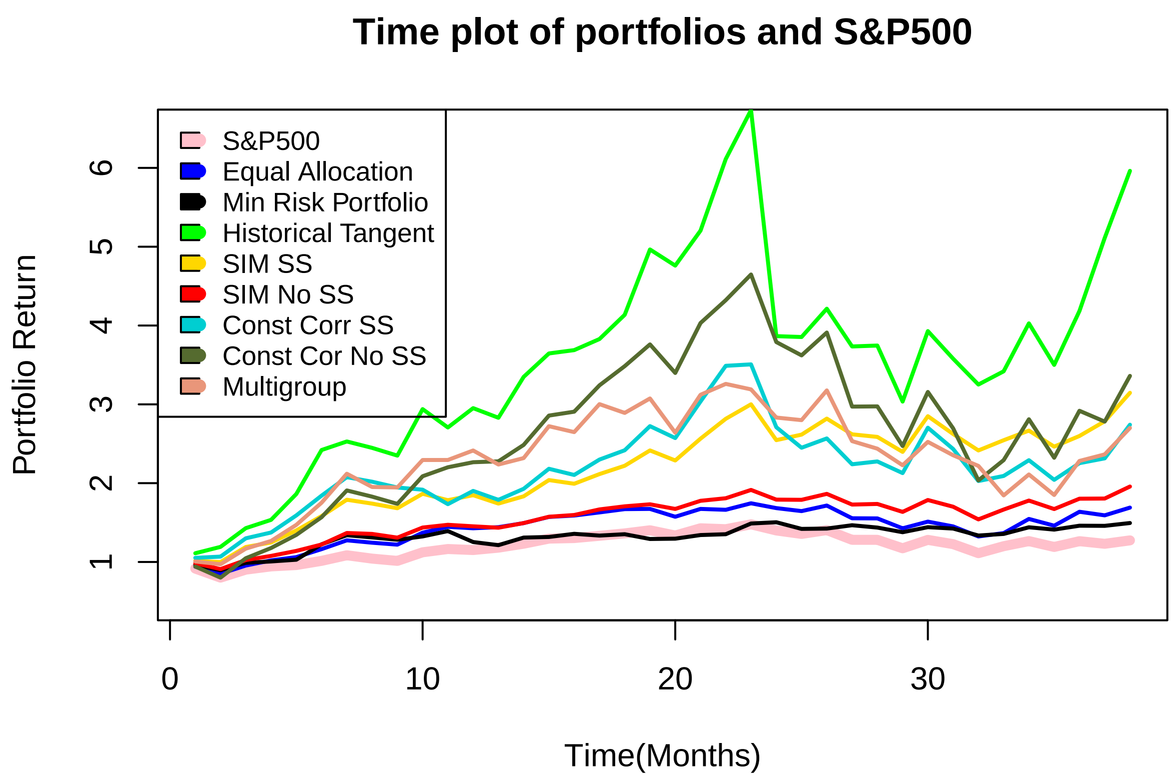
PART B. Evaluate your portfolios that you constructed in the previous projects. In your analysis you should include the following

1. Time plots of the performance of all portfolios compared to the S&P 500 (see the graph constructed using handout #17) under “Labs”.

```r
# Equal allocation portfolio
#equal_weight_vector, means, new_covmat, equal_Rp, equal_sdp

# Min risk portfolio
#min_risk_weight_vector, means, new_covmat, min_risk_Rp, min_risk_sdp

# Tangent historical Rf = 0.002
#x_G_historic, means, new_covmat,Rg_historic, sigmag_historic

# Short sales allowed SIM
# Weights_with_short[,13],Weights_with_short[,4], covariance_matrix_ss, sqrt(var_p_short),R_p_short
Weights_with_short_inv_sorted = Weights_with_short[order(Weights_with_short[,1]),]

# No short sales allowed SIM
# Weights_with_short[1:n_long,13],Weights_with_short[1:n_long,4], covariance_matrix_ss[1:n_long,1:n_lon
Weights_with_no_short_inv_sorted = rep(0,30)
for (i in 1:n_long){
  Weights_with_no_short_inv_sorted[Weights_no_short[i,1]] = Weights_no_short[i,13]
}

# Short sales allowed Constant Corr
#aaaa[,10],aaaa[,2],mat,sd_p_opt,R_p_opt
const_corr_ss_allowed_inv_sorted = aaaa[order(aaaa[,1]),]


# No short sales allowed Constant Corr
n_long_2 = nrow(aaaaa)
#aaaaa[,10],aaaaa[,2],mat[1:n_long_2,1:n_long_2],sd_p_opt_no,R_p_opt_no
const_corr_ss_not_allowed_inv_sorted = rep(0,30)
for (i in 1:n_long_2){
  const_corr_ss_not_allowed_inv_sorted[aaaaa[i,1]] = aaaaa[i,10]
}

# Multi Group
# x_G_group, means, cov_matrix_group, Rg_group, sigmag_group

# TIME PLOTS OF PERFORMANCE OF ALL PORTFOLIOS COMPARED TO MARKET
#Historical period
a_all <- read.csv("stockData.csv", sep=",", header=TRUE)

#Testing period:
a1 <- a_all[1:60,]
a2 <- a_all[61:99,]

#Convert adjusted close prices into returns:
r1 <- (a1[-1,3:ncol(a1)]-a1[-nrow(a1),3:ncol(a1)])/a1[-nrow(a1),3:ncol(a1)]

r2 <- (a2[-1,3:ncol(a2)]-a2[-nrow(a2),3:ncol(a2)])/a2[-nrow(a2),3:ncol(a2)]

#Time plot of equal allocation portfolio
```

```
#Monthly return in period 2015-01-01 to 2018-05-01:
r22 <- as.matrix(r2)

#Market (S&P500) performance in period 2015-01-01 to 2018-05-01:
plot(cumprod(1+(r22[,1])), ylim=c(0.5,6.5), type="l",col="pink", lwd=5,main = "Time plot of portfolios a

#Assume equal allocation:
x <- rep(1/30, 30)

#Compute montly returns in period 2015-01-01 to 2018-05-01:
r22 <- as.matrix(r2)

EquRet <-  r22[,-1] %*% x

lines(cumprod(1+EquRet), col="blue", lwd=2) #equal allocation


#Assume min risk allocation:
x <- min_risk_weight_vector

MinRet <-  r22[,-1] %*% x

lines(cumprod(1+MinRet), col="black", lwd=2) #min portfolio

#Assume tangent rf = 0.002 historical allocation:
x <- x_G_historic

tangentRet <-  r22[,-1] %*% x

lines(cumprod(1+tangentRet), col="green", lwd=2) #tangent portfolio

#Assume sim ss allowed Rf = 0.002  allocation:
x <- Weights_with_short_inv_sorted[,13]

simSSRet <-  r22[,-1] %*% x

lines(cumprod(1+simSSRet), col="gold", lwd=2) #tangent portfolio

#Assume sim ss not allowed Rf = 0.002  allocation:
x <- Weights_with_no_short_inv_sorted

simNoSSRet <-  r22[,-1] %*% x

lines(cumprod(1+simNoSSRet), col="red", lwd=2) #tangent portfolio

#Assume const corr ss allowed Rf = 0.002  allocation:
x <- const_corr_ss_allowed_inv_sorted[,10]

constcorrSSRet <-  r22[,-1] %*% x

lines(cumprod(1+constcorrSSRet), col="darkturquoise", lwd=2) #tangent portfolio

#Assume const corr ss not allowed Rf = 0.002  allocation:
```

```
x <- const_corr_ss_not_allowed_inv_sorted

constcorrNoSSRet <-  r22[,-1] %*% x

lines(cumprod(1+constcorrNoSSRet), col="darkolivegreen", lwd=2) #tangent portfolio

#Assume multi group model with Rf = 0.002
x <- x_G_group

multiGroupRet <-  r22[,-1] %*% x

lines(cumprod(1+multiGroupRet), col="darksalmon", lwd=2) #tangent portfolio

legend("topleft",
       legend=c("S&P500", "Equal Allocation","Min Risk Portfolio","Historical Tangent","SIM SS", "SIM No
       col=c("pink","blue","black","green","gold","red","darkturquoise","darkolivegreen","darksalmon"),
       pch = 19,
       fill =c("pink","blue","black","green","gold","red","darkturquoise","darkolivegreen","darksalmon")
       cex=0.8)
```



2. Average growth of each portfolio (use geometric mean).

```
# AVERAGE GROWTH OF EACH PORTFOLIO

#Instead compute geometric average:
comp <- cumprod(1+ EquRet)
geoMean1 <- comp[length(comp)]^(1/length(comp)) - 1
```

```r
#Instead compute geometric average:
comp <- cumprod(1+ MinRet)
geoMean2 <- comp[length(comp)]^(1/length(comp)) - 1
#Instead compute geometric average:
comp <- cumprod(1+ tangentRet)
geoMean3 <- comp[length(comp)]^(1/length(comp)) - 1
#Instead compute geometric average:
comp <- cumprod(1+ simSSRet)
geoMean4 <- comp[length(comp)]^(1/length(comp)) - 1
#Instead compute geometric average:
comp <- cumprod(1+ simNoSSRet)
geoMean5 <- comp[length(comp)]^(1/length(comp)) - 1
#Instead compute geometric average:
comp <- cumprod(1+ constcorrSSRet)
geoMean6 <- comp[length(comp)]^(1/length(comp)) - 1
#Instead compute geometric average:
comp <- cumprod(1+ constcorrNoSSRet)
geoMean7 <- comp[length(comp)]^(1/length(comp)) - 1
#Instead compute geometric average:
comp <- cumprod(1+ multiGroupRet)
geoMean8 <- comp[length(comp)]^(1/length(comp)) - 1

print("Equal Portfolio Average Geometric Growth")
```

```
## [1] "Equal Portfolio Average Geometric Growth"
```

```r
print(geoMean1)
```

```
## [1] 0.01390225
```

```r
print("Min Risk Portfolio Average Geometric Growth")
```

```
## [1] "Min Risk Portfolio Average Geometric Growth"
```

```r
print(geoMean2)
```

```
## [1] 0.01062536
```

```r
print("Historical Tangent Average Geometric Growth")
```

```
## [1] "Historical Tangent Average Geometric Growth"
```

```r
print(geoMean3)
```

```
## [1] 0.04810756
```

```r
print("SIM SS Average Geometric Growth")
```

```
## [1] "SIM SS Average Geometric Growth"
```

```r
print(geoMean4)
```

```
## [1] 0.03062585
```

```r
print("SIM No SS Average Geometric Growth")
```

```
## [1] "SIM No SS Average Geometric Growth"
```

```r
print(geoMean5)
```

```
## [1] 0.01782876
```

```r
print("Const Corr SS Average Geometric Growth")
```

```
## [1] "Const Corr SS Average Geometric Growth"
```

```r
print(geoMean6)
```

```
## [1] 0.02689265
```

```r
print("Const Cor No SS Average Geometric Growth")
```

```
## [1] "Const Cor No SS Average Geometric Growth"
```

```r
print(geoMean7)
```

```
## [1] 0.03241373
```

```r
print("Multigroup Average Geometric Growth")
```

```
## [1] "Multigroup Average Geometric Growth"
```

```r
print(geoMean8)
```

```
## [1] 0.02648725
```

3. Calculate the Sharpe ratio, differential excess return, Treynor measure, and Jensen differential performance index.

```r
# 3.Calculate the Sharpe ratio, differential excess return, Treynor measure, and Jensen differential pe
R_f = 0.002
#Sharpe Ratio
print("Equal Portfolio Sharpe Ratio")
```

```
## [1] "Equal Portfolio Sharpe Ratio"
```

```r
x <- rep(1/30, 30)
Sharpe1 = (mean(EquRet) - R_f)/sqrt(var(EquRet))
print(Sharpe1)
```

```
##           [,1]
## [1,] 0.2173056
```

```r
print("Min Risk Portfolio Sharpe Ratio")
```

```
## [1] "Min Risk Portfolio Sharpe Ratio"
```

```r
Sharpe2 = (mean(MinRet) - R_f)/sqrt(var(MinRet))
print(Sharpe2)
```

```
##           [,1]
## [1,] 0.1819622
```

```r
print("Historical Tangent Sharpe Ratio")
```

```
## [1] "Historical Tangent Sharpe Ratio"
```

```r
Sharpe3 = (mean(tangentRet) - R_f)/sqrt(var(tangentRet))
print(Sharpe3)
```

```
##           [,1]
## [1,] 0.3904008
```

```r
print("SIM SS Sharpe Ratio")
```

```
## [1] "SIM SS Sharpe Ratio"
```

```r
Sharpe4 = (mean(simSSRet) - R_f)/sqrt(var(simSSRet))
print(Sharpe4)
```

```
##           [,1]
## [1,] 0.3865003
```

```r
print("SIM No SS Sharpe Ratio")
```

```
## [1] "SIM No SS Sharpe Ratio"
```

```r
Sharpe5 = (mean(simNoSSRet) - R_f)/sqrt(var(simNoSSRet))
print(Sharpe5)
```

```
##           [,1]
## [1,] 0.3018398
```

```r
print("Const Corr SS Sharpe Ratio")
```

```
## [1] "Const Corr SS Sharpe Ratio"
```

```r
Sharpe6 = (mean(constcorrSSRet) - R_f)/sqrt(var(constcorrSSRet))
print(Sharpe6)
```

```
##           [,1]
## [1,] 0.2742825
```

```r
print("Const Cor No SS Sharpe Ratio")
```

```
## [1] "Const Cor No SS Sharpe Ratio"
```

```r
Sharpe7 = (mean(constcorrNoSSRet) - R_f)/sqrt(var(constcorrNoSSRet))
print(Sharpe7)
```

```
##           [,1]
## [1,] 0.2769915
```

```r
print("Multigroup Sharpe Ratio")
```

```
## [1] "Multigroup Sharpe Ratio"
```

```r
Sharpe8 = (mean(multiGroupRet) - R_f)/sqrt(var(multiGroupRet))
print(Sharpe8)
```

```
##           [,1]
## [1,] 0.2632039
```

```r
#Differential Excess Return
marketRet = r22[,1]
sigma_market = sqrt(var(marketRet))
mean_market = mean(marketRet)
print("Equal Portfolio Differential Excess Return")
```

```
## [1] "Equal Portfolio Differential Excess Return"
```

```r
x <- rep(1/30, 30)
R_A_apos_Equ = R_f + (((mean_market- R_f)*sqrt(var(EquRet)))/sigma_market)
```

```r
DER1 = mean(EquRet) - R_A_apos_Equ
print(DER1)
```

```
##             [,1]
## [1,] 0.007333067
```

```r
print("Min Risk Portfolio Differential Excess Return")
```

```
## [1] "Min Risk Portfolio Differential Excess Return"
```

```r
R_A_apos_Min = R_f + (((mean_market- R_f)*sqrt(var(MinRet)))/sigma_market)
DER2 = mean(MinRet) - R_A_apos_Min
print(DER2)
```

```
##             [,1]
## [1,] 0.004406324
```

```r
print("Historical Tangent Differential Excess Return")
```

```
## [1] "Historical Tangent Differential Excess Return"
```

```r
R_A_apos_hist_tangent = R_f + (((mean_market- R_f)*sqrt(var(tangentRet)))/sigma_market)
DER3 = mean(tangentRet) - R_A_apos_hist_tangent
print(DER3)
```

```
##            [,1]
## [1,] 0.04230426
```

```r
print("SIM SS Differential Excess Return")
```

```
## [1] "SIM SS Differential Excess Return"
```

```r
R_A_apos_sim_ss = R_f + (((mean_market- R_f)*sqrt(var(simSSRet)))/sigma_market)
DER4 = mean(simSSRet) - R_A_apos_sim_ss
print(DER4)
```

```
##            [,1]
## [1,] 0.02343302
```

```r
print("SIM No SS Differential Excess Return")
```

```
## [1] "SIM No SS Differential Excess Return"
```

```r
R_A_apos_sim_no_ss = R_f + (((mean_market- R_f)*sqrt(var(simNoSSRet)))/sigma_market)
DER5 = mean(simNoSSRet) - R_A_apos_sim_no_ss
print(DER5)
```

```
##            [,1]
## [1,] 0.01152722
```

```r
print("Const Corr SS Differential Excess Return")
```

```
## [1] "Const Corr SS Differential Excess Return"
```

```r
R_A_apos_rho_ss = R_f + (((mean_market- R_f)*sqrt(var(constcorrSSRet)))/sigma_market)
DER6 = mean(constcorrSSRet) - R_A_apos_rho_ss
print(DER6)
```

```
##            [,1]
## [1,] 0.01948765
```

```r
print("Const Cor No SS Differential Excess Return")
```

```
## [1] "Const Cor No SS Differential Excess Return"
```

```r
R_A_apos_rho_no_ss = R_f + (((mean_market- R_f)*sqrt(var(constcorrNoSSRet)))/sigma_market)
DER7 = mean(constcorrNoSSRet) - R_A_apos_rho_no_ss
print(DER7)
```

```
##             [,1]
## [1,] 0.02593833
```

```r
print("Multigroup Differential Excess Return")
```

```
## [1] "Multigroup Differential Excess Return"
```

```r
R_A_apos_group = R_f + (((mean_market- R_f)*sqrt(var(multiGroupRet)))/sigma_market)
DER8 = mean(multiGroupRet) - R_A_apos_group
print(DER8)
```

```
##             [,1]
## [1,] 0.01906038
```

```r
#Treynor Measure
marketRet = r22[,1]

print("Equal Portfolio Treynor Measure")
```

```
## [1] "Equal Portfolio Treynor Measure"
```

```r
x <- rep(1/30, 30)
beta_p_Equ = cov(marketRet,EquRet)/var(marketRet)
T1 = (mean(EquRet) - R_f)/beta_p_Equ
print(T1)
```

```
##            [,1]
## [1,] 0.0140465
```

```r
print("Min Risk Portfolio Treynor Measure")
```

```
## [1] "Min Risk Portfolio Treynor Measure"
```

```r
beta_p_Min = cov(marketRet,MinRet)/var(marketRet)
T2 = (mean(MinRet) - R_f)/beta_p_Min
print(T2)
```

```
##             [,1]
## [1,] 0.01753705
```

```r
print("Historical Tangent Treynor Measure")
```

```
## [1] "Historical Tangent Treynor Measure"
```

```r
beta_p_hist_tangent = cov(marketRet,tangentRet)/var(marketRet)
T3 = (mean(tangentRet) - R_f)/beta_p_hist_tangent
print(T3)
```

```
##             [,1]
## [1,] 0.03945302
```

```r
print("SIM SS Differential Treynor Measure")
```

```
## [1] "SIM SS Differential Treynor Measure"
```

```
beta_p_sim_ss = cov(marketRet,simSSRet)/var(marketRet)
T4 = (mean(simSSRet) - R_f)/beta_p_sim_ss
print(T4)
```

```
##            [,1]
## [1,] 0.03092866
```

```
print("SIM No SS Differential Treynor Measure")
```

```
## [1] "SIM No SS Differential Treynor Measure"
```

```
beta_p_sim_no_ss = cov(marketRet,simNoSSRet)/var(marketRet)
T5 = (mean(simNoSSRet) - R_f)/beta_p_sim_no_ss
print(T5)
```

```
##            [,1]
## [1,] 0.01951102
```

```
print("Const Corr SS Differential Treynor Measure")
```

```
## [1] "Const Corr SS Differential Treynor Measure"
```

```
beta_p_rho_ss = cov(marketRet,constcorrSSRet)/var(marketRet)
T6 = (mean(constcorrSSRet) - R_f)/beta_p_rho_ss
print(T6)
```

```
##            [,1]
## [1,] 0.02566679
```

```
print("Const Cor No SS Differential Treynor Measure")
```

```
## [1] "Const Cor No SS Differential Treynor Measure"
```

```
beta_p_rho_no_ss = cov(marketRet,constcorrNoSSRet)/var(marketRet)
T7 = (mean(constcorrNoSSRet) - R_f)/beta_p_rho_no_ss
print(T7)
```

```
##            [,1]
## [1,] 0.01809161
```

```
print("Multigroup Treynor Measure")
```

```
## [1] "Multigroup Treynor Measure"
```

```
beta_p_group = cov(marketRet,multiGroupRet)/var(marketRet)
T8 = (mean(multiGroupRet) - R_f)/beta_p_group
print(T8)
```

```
##            [,1]
## [1,] 0.02338816
```

```
#Jensen Differential Performance index
marketRet = r22[,1]

print("Equal Portfolio Jensen Differential Performance index")
```

```
## [1] "Equal Portfolio Jensen Differential Performance index"
```

```
x <- rep(1/30, 30)
beta_p_Equ = cov(marketRet,EquRet)/var(marketRet)
```

```r
R_A_apos_Equ2 = R_f + (mean_market - R_f)*beta_p_Equ
JDPI1 = mean(EquRet) - R_A_apos_Equ2
print(JDPI1)
```

```
##             [,1]
## [1,] 0.007755595
```

```r
print("Min Risk Portfolio Jensen Differential Performance index")
```

```
## [1] "Min Risk Portfolio Jensen Differential Performance index"
```

```r
beta_p_Min = cov(marketRet,MinRet)/var(marketRet)
R_A_apos_Min2 = R_f + (mean_market - R_f)*beta_p_Min
JDPI2 = mean(MinRet) - R_A_apos_Min2
print(JDPI2)
```

```
##             [,1]
## [1,] 0.006514807
```

```r
print("Historical Tangent Jensen Differential Performance index")
```

```
## [1] "Historical Tangent Jensen Differential Performance index"
```

```r
beta_p_hist_tangent = cov(marketRet,tangentRet)/var(marketRet)
R_A_apos_hist_tangent2 = R_f + (mean_market - R_f)*beta_p_hist_tangent
JDPI3 = mean(tangentRet) - R_A_apos_hist_tangent2
print(JDPI3)
```

```
##            [,1]
## [1,] 0.04833879
```

```r
print("SIM SS Jensen Differential Performance index")
```

```
## [1] "SIM SS Jensen Differential Performance index"
```

```r
beta_p_sim_ss = cov(marketRet,simSSRet)/var(marketRet)
R_A_apos_sim_ss2 = R_f + (mean_market - R_f)*beta_p_sim_ss
JDPI4 = mean(simSSRet) - R_A_apos_sim_ss2
print(JDPI4)
```

```
##            [,1]
## [1,] 0.02549549
```

```r
print("SIM No SS Jensen Differential Performance index")
```

```
## [1] "SIM No SS Jensen Differential Performance index"
```

```r
beta_p_sim_no_ss = cov(marketRet,simNoSSRet)/var(marketRet)
R_A_apos_sim_no_ss2 = R_f + (mean_market - R_f)*beta_p_sim_no_ss
JDPI5 = mean(simNoSSRet) - R_A_apos_sim_no_ss2
print(JDPI5)
```

```
##            [,1]
## [1,] 0.01191014
```

```r
print("Const Corr SS Jensen Differential Performance index")
```

```
## [1] "Const Corr SS Jensen Differential Performance index"
```

```r
beta_p_rho_ss = cov(marketRet,constcorrSSRet)/var(marketRet)
R_A_apos_rho_ss2 = R_f + (mean_market - R_f)*beta_p_rho_ss
```

```r
JDPI6 = mean(constcorrSSRet) - R_A_apos_rho_ss2
print(JDPI6)
```

```
##             [,1]
## [1,] 0.02359007
```

```r
print("Const Cor No SS Jensen Differential Performance index")
```

```
## [1] "Const Cor No SS Jensen Differential Performance index"
```

```r
beta_p_rho_no_ss = cov(marketRet,constcorrNoSSRet)/var(marketRet)
R_A_apos_rho_no_ss2 = R_f + (mean_market - R_f)*beta_p_rho_no_ss
JDPI7 = mean(constcorrNoSSRet) - R_A_apos_rho_no_ss2
print(JDPI7)
```

```
##             [,1]
## [1,] 0.02706913
```

```r
print("Multigroup Jensen Differential Performance index")
```

```
## [1] "Multigroup Jensen Differential Performance index"
```

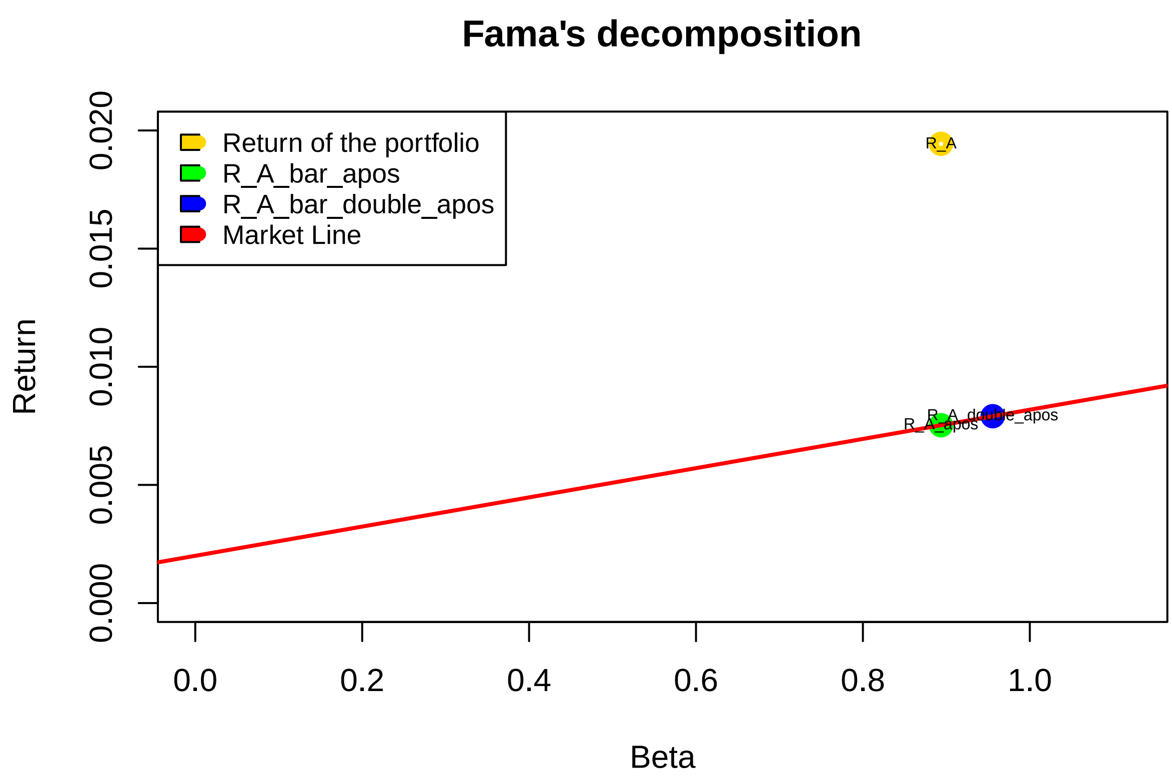
```r
beta_p_group = cov(marketRet,multiGroupRet)/var(marketRet)
R_A_apos_group2 = R_f + (mean_market - R_f)*beta_p_group
JDPI8 = mean(multiGroupRet) - R_A_apos_group2
print(JDPI8)
```

```
##             [,1]
## [1,] 0.02293353
```

4. Decompose the overall performance using Fama's decomposition (net selectivity and diversification) for the single index model when short sales are not allowed. Please show this decomposition on the plot expected return against beta.

```r
R_A_bar_apos = R_f + (mean_market - R_f)*beta_p_sim_no_ss

beta_A_double_apos = sqrt(var(simNoSSRet)/var(marketRet))
R_A_bar_double_apos = R_f + (mean_market - R_f)*beta_A_double_apos
# plot function is used to plot
# the data type with "n" is used
# to remove the plotted data

plot(1, type = "n", xlab = "Beta",
     ylab = "Return", xlim = c(0, 1.12),
     ylim = c(0, 0.02),main = "Fama's decomposition")

points(beta_p_sim_no_ss,mean(simNoSSRet),lwd = 5, col = "gold")
points(beta_p_sim_no_ss,R_A_bar_apos,lwd = 5,col = "green")
points(beta_A_double_apos,R_A_bar_double_apos,lwd = 5,col = "blue")
abline(a = R_f, b = (mean_market - R_f) , lwd = 2, col = "red")
text(beta_p_sim_no_ss, mean(simNoSSRet), labels="R_A",cex = 0.5)
text(beta_p_sim_no_ss, R_A_bar_apos, labels="R_A_apos",cex = 0.5)
text(beta_A_double_apos, R_A_bar_double_apos, labels="R_A_double_apos",cex = 0.5)
legend("topleft",
       legend=c("Return of the portfolio","R_A_bar_apos","R_A_bar_double_apos","Market Line"),
       col=c("gold","green","blue","red"),
       pch = 19,
```

```
       fill =c("gold","green","blue","red"),
       cex=0.8)
```




```
print("Return from selectivitiy")
```

```
## [1] "Return from selectivitiy"
```

```
print(mean(simNoSSRet) - R_A_bar_apos )
```

```
##             [,1]
## [1,] 0.01191014
```

```
print("Return from net selectivitiy")
```

```
## [1] "Return from net selectivitiy"
```

```
print(mean(simNoSSRet) - R_A_bar_double_apos )
```

```
##             [,1]
## [1,] 0.01152722
```

```
print("Return from diversification")
```

```
## [1] "Return from diversification"
```

```
print(R_A_bar_double_apos - R_A_bar_apos )
```

```
##               [,1]
## [1,] 0.0003829151
```