Workshop 10

- The Markowitz portfolio
    - Closed-form solution
    - Comparing to a convex optimisation problem solver
    - Markowitz frontier
    - Real-world example
    - With short-selling constraints
- Over-prediction of returns
    - Realistic parameters
    - A run-through of the simulation steps
    - Full simulation
- Filtered covariance

The Markowitz portfolio

Closed-form solution

Define the (population) returns of the stocks.

```
mu <- c(4.27, 0.15, 2.85)
names(mu) <- c("S1", "S2", "S3")
```

Define the (population) covariance matrix.

```
Sigma <- matrix(c(1, 0.18, 0.11,
                  0.18, 1.1, 0.26,
                  0.11, 0.26, 1.99), nrow=3, ncol=3)

rownames(Sigma) <- c("S1", "S2", "S3")
colnames(Sigma) <- c("S1", "S2", "S3")
```

Show the correlations.

```
cov2cor(Sigma)
```

```
##            S1        S2         S3
## S1 1.00000000 0.1716233 0.07797693
## S2 0.17162327 1.0000000 0.17573184
## S3 0.07797693 0.1757318 1.00000000
```

Define the Markowitz function that solves the optimal portfolio problem given (μ, Σ).

```
markowitz <-function(mu, Sigma){
  InvSigma <- try(solve(Sigma), TRUE)
  if("try-error" %in% class(InvSigma)) stop

  One = rep.int(1,length(mu))

  a = as.double(t(mu)%*%InvSigma%*%mu)
  b = as.double(t(mu)%*%InvSigma%*%One)
  c = as.double(t(One)%*% InvSigma %*% One)
  d = a*c - b^2

  Phi = (a/d)*(InvSigma %*% One) - (b/d)*(InvSigma %*% mu)
  Theta = (c/d)*(InvSigma %*% mu) - (b/d)*(InvSigma %*% One)

  R = b/c
  Pi = Phi + Theta*R

  sigma <- Vectorize(function(r)  sqrt((c/d)*((r-(b/c))^2) + (1/c)))

  return(list(Pi=Pi, R=R, sigma=sigma))
}
```

We obtain the result using our function.

```
mp <- markowitz(mu, Sigma)
```

We can extract the optimal returns.

```
mp$R
```

```
## [1] 2.498643
```

We can extract the optimal portfolio weights.

```
mp$Pi
```

```
##         [,1]
## S1 0.4428060
## S2 0.3630153
## S3 0.1941788
```

Comparing to a convex optimisation problem solver

Our function is similar to solving the quadratic programming problems of the form

$$\min \frac{1}{2} x' \Sigma x - \mathbb{1}' x$$

with the constraints $\mathbb{1}' x = 1$.

The function `solve.QP` is available with the `quadprog` package. Install it using `install.packages("quadprog")` and make it available in your R session.

```
library("quadprog")
```

The function `solve.QP` solves problems of the form

$$\min \frac{1}{2} x' D x - d' x$$

with the constraints $A' x \geq b_0$. Here we set $D = \Sigma$, $d = \mathbb{1}$, $A = \mathbb{1}$, and $b_0 = 1$.

```
d <- matrix(0, ncol(Sigma), 1)
A <- matrix(1, ncol(Sigma), 1)
Pi <- solve.QP(Sigma, d, A, bvec=1, meq=1)$solution
```

We get the same optimal portfolio as our own solver (above).

```
Pi
```

```
## [1] 0.4428060 0.3630153 0.1941788
```

The portfolio return is given by

```
R <- t(Pi) %*% mu
R
```

```
##          [,1]
## [1,] 2.498643
```

Markowitz frontier

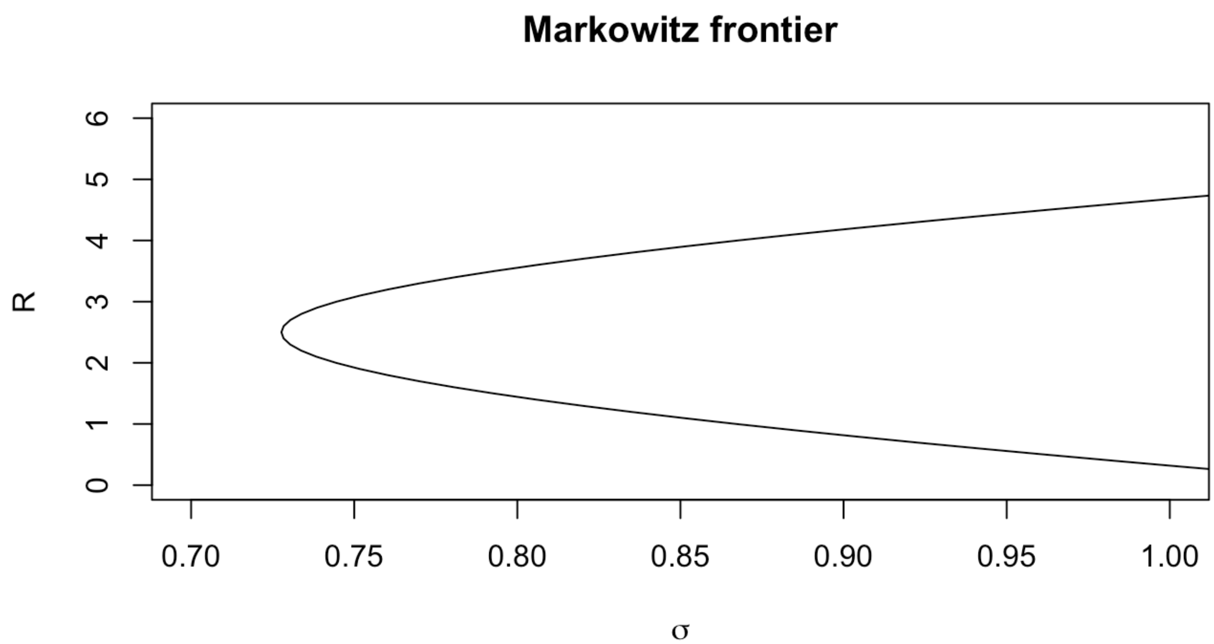
The closed-form solver above also provided a `sigma` function that maps from return R to level of risk

σ_0. We can use this to draw the Markowitz frontier.

```
mp <- markowitz(mu, Sigma)
returns <- seq(0, 6, 0.1)
sigmas <- mp$sigma(returns)
```

Notice how we flip the axes and draw the function the other way around so it is one-to-one.

```
plot(sigmas, returns, xlab=expression(sigma), ylab=expression(R), type = "l", m
ain ="Markowitz frontier", xlim=c(0.7,1))
```



Real-world example

We can do a real-world example but we need to install a few packages.

```
install.packages("rvest")
install.packages("quantmod")
install.packages("corpcor")
install.packages("tawny")
```

First we get the names of stocks within the ASX200 by scraping the table found on the Wikipedia page https://en.wikipedia.org/wiki/S%26P/ASX_200 (https://en.wikipedia.org/wiki/S%26P/ASX_200). We use the `rvest` package to use fancy syntax.

```
library("rvest")
```

```
## Loading required package: xml2
```

```
url <- "https://en.wikipedia.org/wiki/S%26P/ASX_200"
asx200 <- url %>%
  read_html() %>%
  html_nodes(xpath='//*[@id="mw-content-text"]/table[1]') %>%
  html_table()
```

We now get a list of the tickers.

```
tickers <- as.character(lapply(asx200[[1]][,1], function(t) paste(t, ".AX", sep
="")))
```

We are going to restrict the data downloaded to only the first 10 stocks. Do not evaluate this line if you want everything.

```
tickers <- tickers[1:10]
```

Alternatively, evaulate this next line only if you want BHP and RIO.

```
tickers <- as.character(c("BHP.AX", "RIO.AX"))
```

Now we download the returns using the `quantmod` package.

```
library("quantmod")
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

We get prices from the start of 2016 for all tickers.

```
getSymbols(tickers, from="2016-01-01")
```

```
##     As of 0.4-0, 'getSymbols' uses env=parent.frame() and
##  auto.assign=TRUE by default.
##
##  This  behavior  will be  phased out in 0.5-0  when the call  will
##  default to use auto.assign=FALSE. getOption("getSymbols.env") and
##  getOptions("getSymbols.auto.assign") are now checked for alternate defaults
##
##  This message is shown once per session and may be disabled by setting
##  options("getSymbols.warning4.0"=FALSE). See ?getSymbols for more details.
```

```
## [1] "BHP.AX" "RIO.AX"
```

We download daily returns.

```
DailyReturns <- do.call(merge, lapply(tickers, function(x){ periodReturn(get(x)
, period='daily') }))
names(DailyReturns) <- tickers
```

We now solve for the optimal portfolio.

```
require("quadprog")
```

We will use the sample covariance matrix.

```
S <- cov(DailyReturns)
```

We check correlations.

```
cov2cor(S)
```

```
##           BHP.AX    RIO.AX
## BHP.AX 1.0000000 0.8980749
## RIO.AX 0.8980749 1.0000000
```

We will calculate the “plug-in optimal portfolio”.

```
mu.hat <- colMeans(DailyReturns)
mp <- markowitz(mu.hat, S)
```

We get the optimal returns (in percent p.a.)

```
mp$R * 100 * 250 # % p.a.
```

```
## [1] 21.97238
```

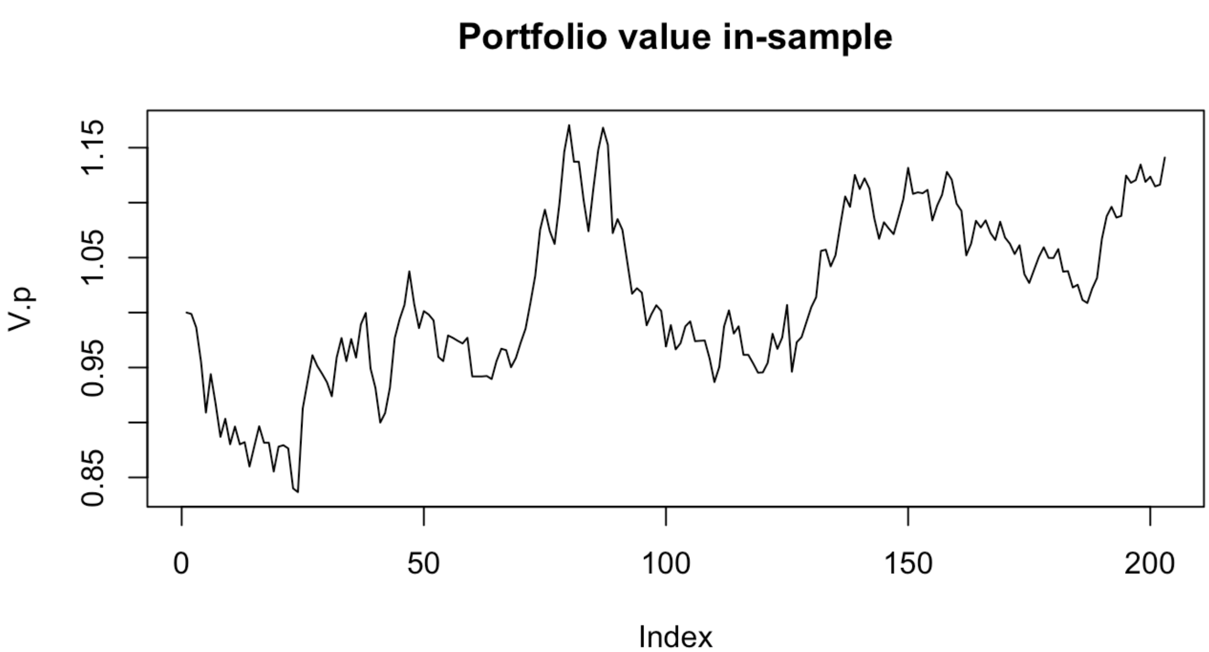
We get the optimal portfolio.

```
mp$Pi
```

```
##              [,1]
## BHP.AX -0.317713
## RIO.AX  1.317713
```

We plot the value of the portfolio assuming that we have normalised it so the value starts at $1 on the first day.

```
V.p <- cumprod(1+DailyReturns %*% mp$Pi)
plot(V.p, type='l', main="Portfolio value in-sample")
```



With short-selling constraints

If you want to get fancy, we can also solve a similar optimization with a short selling constraint. In other words, we create a “long-only” portfolio. Mathematically, this means that we restrict non-negative weights for the portfolio.

```
N <- ncol(S)
aMat <- cbind(t(array(1, dim = c(1,N))), diag(N))
b0 <- as.matrix(c(1, rep.int(0,N)))
zeros <- array(0, dim = c(N,1))
res <- solve.QP(S, zeros, aMat, bvec=b0, meq = 1)
```

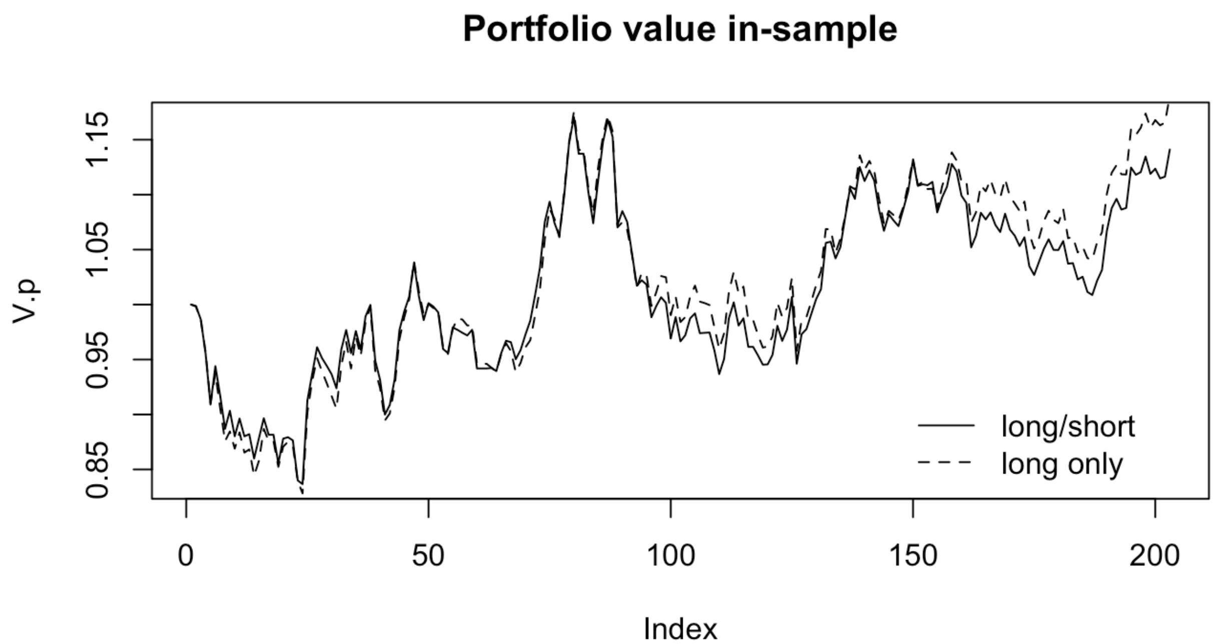
We get the optimal returns (in percent p.a.)

```
Pi.p.longonly <- res$solution
R <- Pi.p.longonly %*% mu.hat
R * 100 * 250 # % p.a.
```

```
##          [,1]
## [1,] 27.31819
```

We compare the long/short optimal portfolio to the long-only portfolio in-sample.

```
Vl.p <- cumprod(1+DailyReturns %*% Pi.p.longonly)
plot(V.p, type='l', main="Portfolio value in-sample")
lines(Vl.p, type='l', lty=2)
legend("bottomright", c("long/short", "long only"), lty=c(1,2), text.width=40,
bty = "n")
```



Over-prediction of returns

We illustrate the over-prediction phenomena by performing a Monte-Carlo simulation.

Realistic parameters

We choose some “realistic” parameters for our stock price simulation. We use the back-of-the-envelope approximation that there are 250 “business days” in a year.

```
avg.yearly.return <- 0.03 / 250
avg.yearly.retsd <- 0.02 / 250
```

A run-through of the simulation steps

First, we run through the steps that we are going to take in our simulation to make sure that everything works well.

First, we set the number of stocks.

```
p <- 10
```

Generate stock names.

```
tickers <- as.character(lapply(seq(1,p), function(x) paste("S", x, sep="")))
```

Generate some realistic (population) daily returns and print the corresponding percentage return per annum.

```
mu <- rnorm(p, mean=avg.yearly.return, sd=avg.yearly.retsd)
names(mu) <- tickers
mu * 100 * 250 # % p.a.
```

```
##        S1        S2        S3        S4        S5        S6        S7
## 4.3689123 4.1697871 4.8810086 2.3732367 4.9838637 0.9353538 0.7783619
##        S8        S9       S10
## 3.0807182 6.6733091 3.6376965
```

We generate a reasonable (population) covariance matrix for daily returns.

```
pcor <- function(rho, p) {
  Tn <- matrix(0, p, p)
  for (i in 1:p) {
    for (j in 1:p) {
      Tn[i,j] <- rho^abs(i-j)
    }
  }
  return(Tn)
}

Sigma <- rWishart(1, df=500, pcor(0.7, p))[,,1]
Sigma <- avg.yearly.retsd*Sigma/max(Sigma)
rownames(Sigma) <- tickers
colnames(Sigma) <- tickers

print(Sigma * 100 * 250, digits=2) # % p.a.
```

```
##       S1   S2   S3   S4   S5   S6   S7   S8   S9  S10
## S1  1.82 1.27 0.94 0.67 0.48 0.32 0.25 0.15 0.10 0.12
## S2  1.27 1.65 1.22 0.87 0.66 0.46 0.36 0.28 0.17 0.12
## S3  0.94 1.22 1.89 1.43 1.09 0.77 0.56 0.45 0.34 0.28
## S4  0.67 0.87 1.43 1.97 1.45 0.97 0.67 0.51 0.39 0.27
## S5  0.48 0.66 1.09 1.45 1.99 1.42 1.02 0.76 0.54 0.34
## S6  0.32 0.46 0.77 0.97 1.42 2.00 1.42 1.04 0.76 0.53
## S7  0.25 0.36 0.56 0.67 1.02 1.42 1.91 1.38 0.98 0.69
## S8  0.15 0.28 0.45 0.51 0.76 1.04 1.38 2.00 1.35 0.90
## S9  0.10 0.17 0.34 0.39 0.54 0.76 0.98 1.35 1.83 1.33
## S10 0.12 0.12 0.28 0.27 0.34 0.53 0.69 0.90 1.33 1.91
```

We look at the correlations.

```
print(cov2cor(Sigma), digits=2)
```

```
##         S1    S2   S3   S4   S5   S6   S7    S8    S9   S10
## S1  1.000 0.732 0.51 0.35 0.25 0.17 0.13 0.079 0.056 0.067
## S2  0.732 1.000 0.69 0.48 0.37 0.25 0.20 0.153 0.097 0.065
## S3  0.507 0.691 1.00 0.74 0.56 0.40 0.30 0.231 0.182 0.149
## S4  0.354 0.484 0.74 1.00 0.74 0.49 0.35 0.259 0.204 0.139
## S5  0.252 0.367 0.56 0.74 1.00 0.71 0.53 0.380 0.283 0.176
## S6  0.168 0.250 0.40 0.49 0.71 1.00 0.73 0.519 0.398 0.272
## S7  0.133 0.204 0.30 0.35 0.53 0.73 1.00 0.706 0.527 0.360
## S8  0.079 0.153 0.23 0.26 0.38 0.52 0.71 1.000 0.706 0.460
## S9  0.056 0.097 0.18 0.20 0.28 0.40 0.53 0.706 1.000 0.712
## S10 0.067 0.065 0.15 0.14 0.18 0.27 0.36 0.460 0.712 1.000
```

We calculate the optimal theoretical portfolio based on the (population) μ and Σ.

```
mp <- markowitz(mu, Sigma)
```

We get the optimal portfolio.

```
Pi <- mp$Pi
Pi
```

```
##            [,1]
## S1   0.20985771
## S2   0.17982689
## S3  -0.04112536
## S4   0.10952573
## S5   0.04820117
## S6   0.05685283
## S7   0.06616085
## S8   0.06745777
## S9   0.08046403
## S10  0.22277838
```

And the optimal returns.

```
R <- mp$R
R * 100 * 250 # % p.a.
```

```
## [1] 3.62597
```

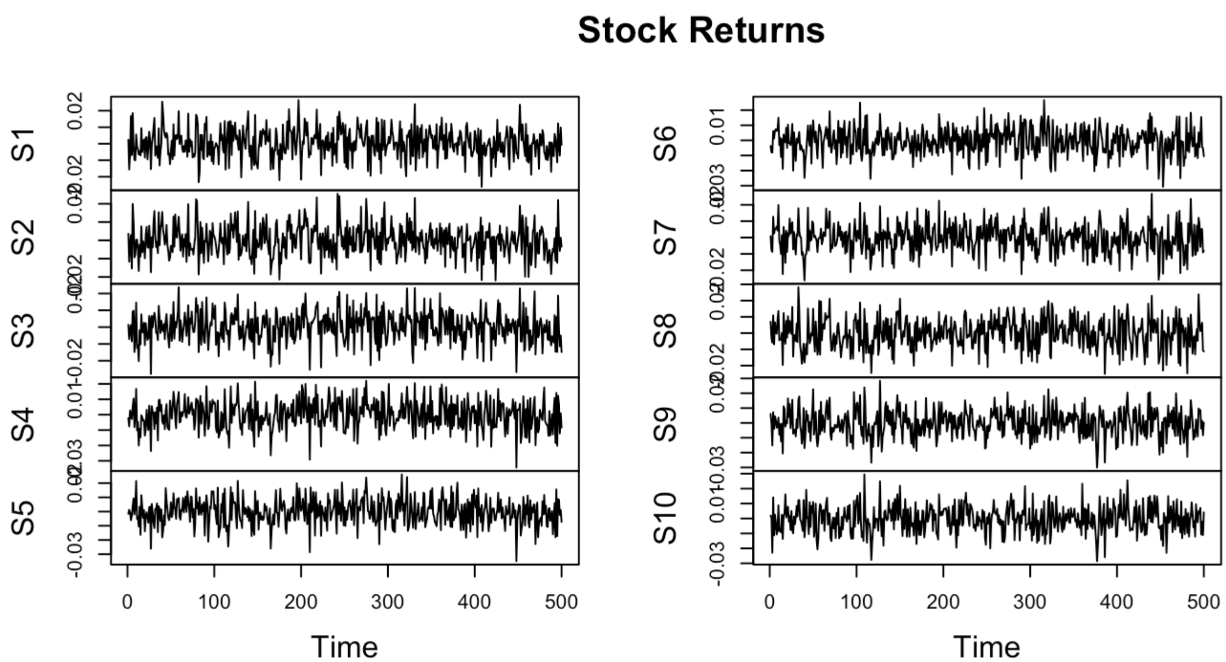
We now simulate large number of returns for the stocks. We use the `mvtnorm` package so we can take into account Σ.

```
library(mvtnorm)
```

```
n <- 500
Returns <- rmvnorm(n, mean=mu, sigma=Sigma)
colnames(Returns) <- tickers
```

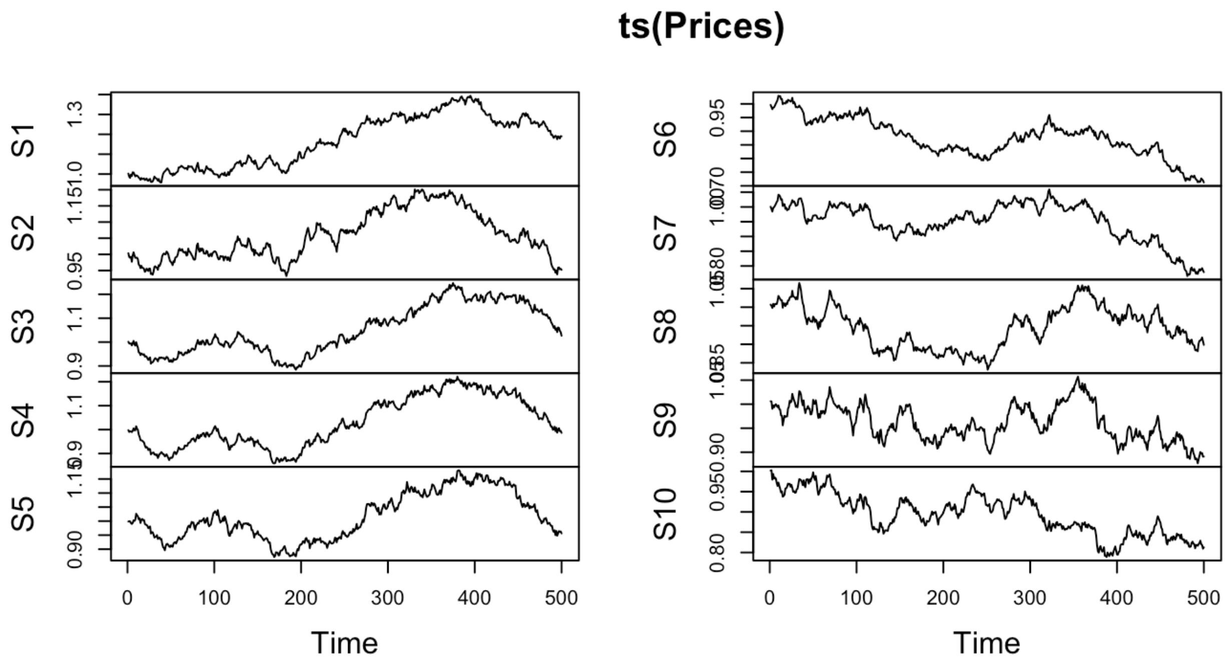
We plot the simulated stock returns.

```
plot(ts(Returns), type='l', main="Stock Returns")
```



We plot the simulated stock prices, assuming that all stocks start at $1 price at the initial time. This gives us “realistic” prices over $n = 500$ days.

```
Prices <- apply(Returns, 2, function(x) cumprod(1+x))
plot(ts(Prices))
```



We now calculate the sample mean.

```
sample.mu <- colMeans(Returns)
names(sample.mu) <- tickers
sample.mu * 100 * 250 # % p.a.
```

```
##          S1          S2          S3          S4          S5          S6
##   9.6643380  -1.5677629   2.3225558   0.4088148  -1.1515996 -15.9229335
##          S7          S8          S9         S10
## -11.5094699  -4.2503366  -4.7335209  -9.3620221
```

And sample covariance matrix.

```
S <- cov(Returns)
rownames(S) <- tickers
colnames(S) <- tickers
```

We print the correlations.

```
print(cov2cor(S), digits=2)
```

```
##        S1   S2   S3   S4   S5    S6    S7    S8    S9   S10
## S1  1.000 0.76 0.52 0.34 0.18 0.056 0.033 0.035 0.027 0.022
## S2  0.756 1.00 0.70 0.49 0.32 0.157 0.135 0.167 0.121 0.090
## S3  0.519 0.70 1.00 0.78 0.58 0.405 0.347 0.331 0.298 0.272
## S4  0.336 0.49 0.78 1.00 0.76 0.511 0.364 0.309 0.247 0.202
## S5  0.180 0.32 0.58 0.76 1.00 0.749 0.564 0.485 0.395 0.248
## S6  0.056 0.16 0.41 0.51 0.75 1.000 0.761 0.599 0.483 0.301
## S7  0.033 0.14 0.35 0.36 0.56 0.761 1.000 0.732 0.616 0.400
## S8  0.035 0.17 0.33 0.31 0.48 0.599 0.732 1.000 0.736 0.485
## S9  0.027 0.12 0.30 0.25 0.40 0.483 0.616 0.736 1.000 0.723
## S10 0.022 0.09 0.27 0.20 0.25 0.301 0.400 0.485 0.723 1.000
```

We get the ratio of p to n.

```
y <- p/n
```

We calculate the “plug-in portfolio” based on the sample mean $\bar{\mathbb{x}}$ and sample covariance $\mathbb{S}$.

```
mp <- markowitz(sample.mu, S)
```

We get the “plug-in portfolio”.

```
Pi.p <- mp$Pi
Pi.p
```

```
##             [,1]
## S1   0.277764006
## S2   0.210897275
## S3  -0.204923566
## S4   0.149823201
## S5  -0.006271902
## S6   0.136174078
## S7   0.132720741
## S8   0.030888607
## S9  -0.002228579
## S10  0.275156139
```

And the “plug-in returns”.

```
R.p <- mp$R
R.p * 100 * 250 # % p.a.
```

```
## [1] -4.446297
```

Full simulation

We are going to perform this simulation over the following values of p.

```
ps <- c(10, 50, 100, 120, 160, 200, 220, 240)
```

For each value of p, we are going to perform M simulations and take the average of the plug-in returns.

```
M <- 50
```

The number of days for returns.

```
n <- 250
```

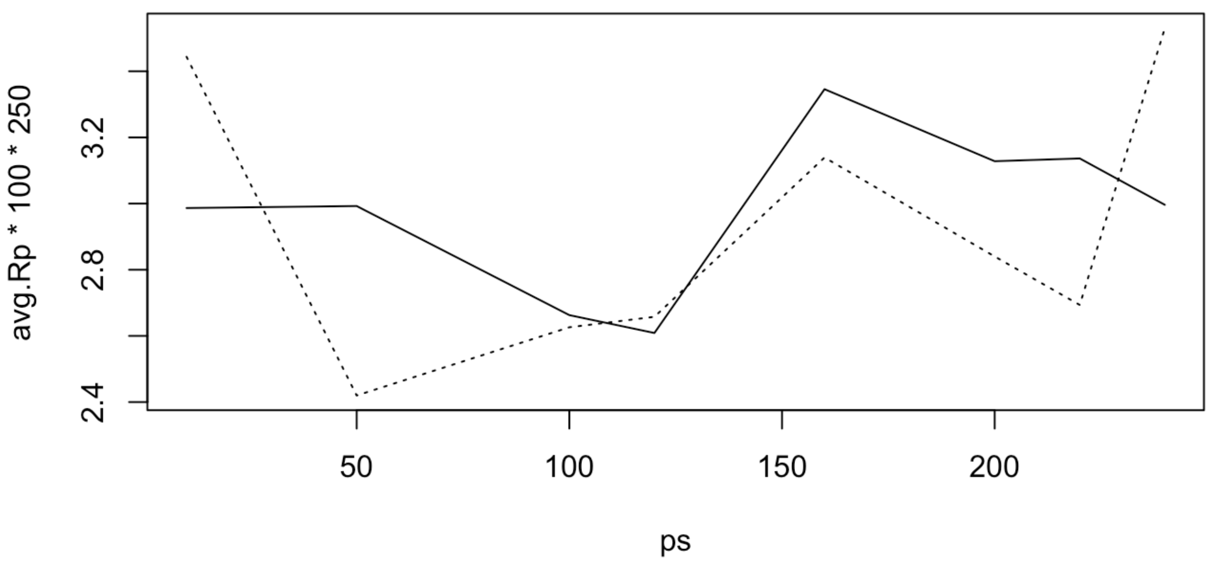
We now do the full simulation based on the steps above.

```
R <- rep(0, length(ps))
avg.Rp <- rep(0, length(ps))

for (k in 1:length(ps)) {
  p <- ps[k]
  cat('p:', p, '\n')

  # population values for mu and Sigma
  mu <- rnorm(p, mean=avg.yearly.return, sd=avg.yearly.retsd)
  Sigma <- rWishart(1, df=500, diag(0.01, p))[,,1]
  Sigma <- avg.yearly.retsd*Sigma/max(Sigma)

  # theoretical return
  mp <- markowitz(mu, Sigma)
  R[k] <- mp$R

  Rp <- rep(0, M)
  for (m in 1:M) {
    # simulated returns
    Returns <- rmvnorm(n, mean=mu, sigma=Sigma)
    sample.mu <- colMeans(Returns)
    S <- cov(Returns)
    mp <- markowitz(sample.mu, S)
    Rp[m] <- mp$R
  }

  avg.Rp[k] <- mean(Rp)
}
```

```
## p: 10
## p: 50
## p: 100
## p: 120
## p: 160
## p: 200
## p: 220
## p: 240
```

Plot in terms of % p.a.

```
plot(ps, avg.Rp * 100 * 250, type='l', lty=3)
lines(ps, R * 100 * 250)
```



Filtered covariance

There are some packages that can perform more robust estimation of the covariance matrix. Some examples are `corpcor` with their `cov.shrink` function and `tawny` which uses RMT.

```
require("corpcor")
```

```
## Loading required package: corpcor
```

```
require("tawny")
```

```
## Loading required package: tawny
```

```
##
## Attaching package: 'tawny'
```

```
## The following object is masked from 'package:corpcor':
##
##     cov.shrink
```

Have a go at using them…