FIN511 Week 1

First code chunk used to load the necessary libraries in R if you don't have these libraries, please uncomment the code and install them or install them in R-Studio.

```
# load libraries

# install readxl and tidyverse (or just ggplot2 and dyplr)
# install.packages("readxl","ggplot2","dplyr")

library(ggplot2)
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(reshape2)
```

The following code chunk shows a very "dumb" way to load excel spreadsheets in R directly, skipping lines if necessary, like in this case and use the function "head" to see if the spreadsheet is loaded properly

```
# Load all the datasets and separate sheets, clean up the data and leave a single row for column names.
# You will still need the original excel sheet to read the instructions at the top or additional inform
# modify the files paths to match your needs
# I didn't set working directories or anything else to keep the customization to a minimum

df1_mom_returns <- read_xlsx("/Users/ataru074/Desktop/Education/MBA/03 2020 Fall/FIN511 Investments/213

df1_portfolio_strategies <- read_xlsx("/Users/ataru074/Desktop/Education/MBA/03 2020 Fall/FIN511 Investm
```

```
## New names:
## * `` -> ...3
## * `` -> ...7
## * `` -> ...11
```

```
rt_vol_risky_and_rf <- read_xlsx("/Users/ataru074/Desktop/Education/MBA/03 2020 Fall/FIN511 Investments/
```

```
## New names:
## * `` -> ...2
## * `` -> ...9
## * `` -> ...12
```

```
rt_vol_risky_assets <- read_xlsx("/Users/ataru074/Desktop/Education/MBA/03 2020 Fall/FIN511 Investments/
```

```
## New names:
## * `` -> ...2
## * `` -> ...10
## * `` -> ...13
## * `` -> ...15
```

```r
head(df1_mom_returns)
```

```
## # A tibble: 6 x 12
##   Year    Low    `2`   `3`    `4`    `5`    `6`    `7`    `8`    `9`  High
##   <chr> <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>
## 1 1927   18.5   9.07  17.2  26.3   32.7   25.1   35.4   27.2   39.4   67.4
## 2 1928   14.7  23.2   21.4  26.1   25.5   34.3   41.9   42.8   48.7   89.3
## 3 1929  -57.6 -47.8  -34.2 -21.3   -2.32 -13.3    5.95   1.32   0.25 -29.9
## 4 1930  -54.6 -48.4  -50.4 -49.6  -31.7  -31.7  -32.4  -19.7  -22.9  -20.9
## 5 1931  -64.5 -58.3  -60.6 -52.9  -53.9  -49.7  -49.4  -42.5  -34.8  -24.2
## 6 1932   25.7  -2.46 -12.2  -7.47   9.31  -1.33 -17.1  -14.4  -12.4  -25.8
## # ... with 1 more variable: `High-Low` <dbl>
```

Lets replicate the "two risky assets" spreadsheet

Step 1, let's get all the information together and organize it in a dataframe (a structure similar to a spreadsheet)

```r
# Risk Free rate
RF <- 3
# The large Weight is a sequence that goes from -1 to 2 in 0.01 steps
Large_Weight <- seq(-1,2,0.01)
# The Small Weight is simply 1 - Large Weight
Small_Weight <- 1 - Large_Weight

# Large return
LR <- 8
# Small return
SR <- 15

# Correlation (using CR because cor is an R funcion therefore reserved word)
CR <- 0.4

# Large Standard Deviation
LSD <- 25
# Small Standard Deviation
SSD <- 50

# lets create the dataframe. Note, R automatically recycle variables to fit the lenght of the dataframe
# therefore is important that the first variable is the correct lenght.

df <- data.frame("RF" = RF ,"Large_Weight" = Large_Weight, "Small_Weight" = Small_Weight, "LR" = LR, "S
                 "CR" = CR, "LSD" = LSD, "SSD" = SSD)

# We could have also avoided all the variable declarations and just do:
# df <- data.frame("Large_Weight" = seq(-1,2,0.01), "Small_Weight" = 1 - seq(-1,2,0.01), "LR" = 8)

# Now let's add the calculated columns (note that column can be added just using df "$" name_of_the_col
# and we use the dollar sign to select the column by name as well
# Important Don't use spaces in the name of columns or variables or anything in general, it's just bad
```

```r
# practice at best and in most cases the software will give you an error
df$Portfolio_SD <- sqrt(df$Large_Weight^2 * df$LSD^2 + df$Small_Weight^2 * df$SSD^2 +
                        2*df$CR*df$Large_Weight*df$LSD*df$Small_Weight*df$SSD)

df$Portfolio_Return <- df$Large_Weight*df$LR + df$Small_Weight*df$SR

df$Sharpe_Ratio <- (df$Portfolio_Return - df$RF)/df$Portfolio_SD

# let's verify that our dataframe is created correctly
head(df)
```

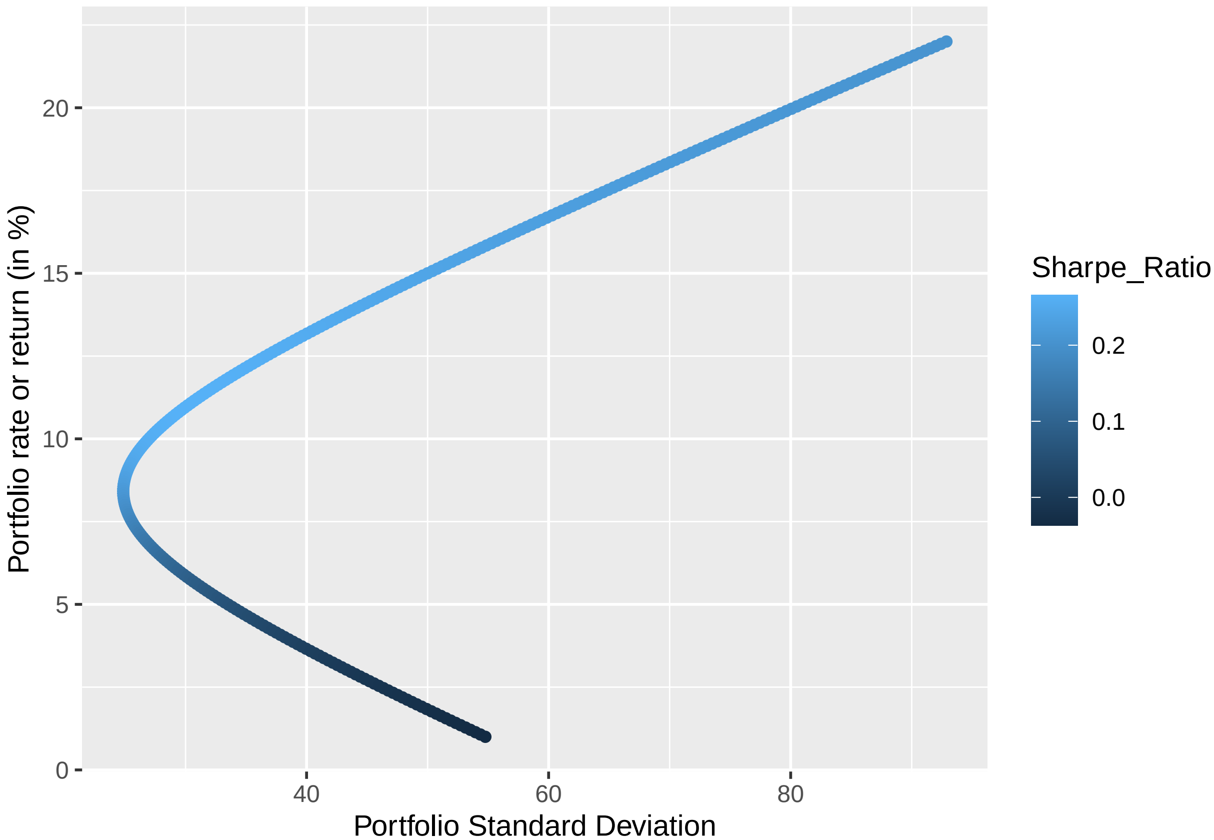
```
##   RF Large_Weight Small_Weight LR SR  CR LSD SSD Portfolio_SD Portfolio_Return
## 1  3        -1.00         2.00  8 15 0.4  25  50     92.87088            22.00
## 2  3        -0.99         1.99  8 15 0.4  25  50     92.42680            21.93
## 3  3        -0.98         1.98  8 15 0.4  25  50     91.98288            21.86
## 4  3        -0.97         1.97  8 15 0.4  25  50     91.53913            21.79
## 5  3        -0.96         1.96  8 15 0.4  25  50     91.09555            21.72
## 6  3        -0.95         1.95  8 15 0.4  25  50     90.65215            21.65
##   Sharpe_Ratio
## 1    0.2045851
## 2    0.2048107
## 3    0.2050382
## 4    0.2052674
## 5    0.2054985
## 6    0.2057315
```

```r
tail(df)
```

```
##     RF Large_Weight Small_Weight LR SR  CR LSD SSD Portfolio_SD
## 296  3         1.95        -0.95  8 15 0.4  25  50     52.72867
## 297  3         1.96        -0.96  8 15 0.4  25  50     53.13568
## 298  3         1.97        -0.97  8 15 0.4  25  50     53.54356
## 299  3         1.98        -0.98  8 15 0.4  25  50     53.95229
## 300  3         1.99        -0.99  8 15 0.4  25  50     54.36187
## 301  3         2.00        -1.00  8 15 0.4  25  50     54.77226
##     Portfolio_Return Sharpe_Ratio
## 296             1.35  -0.03129227
## 297             1.28  -0.03236997
## 298             1.21  -0.03343073
## 299             1.14  -0.03447490
## 300             1.07  -0.03550283
## 301             1.00  -0.03651484
```

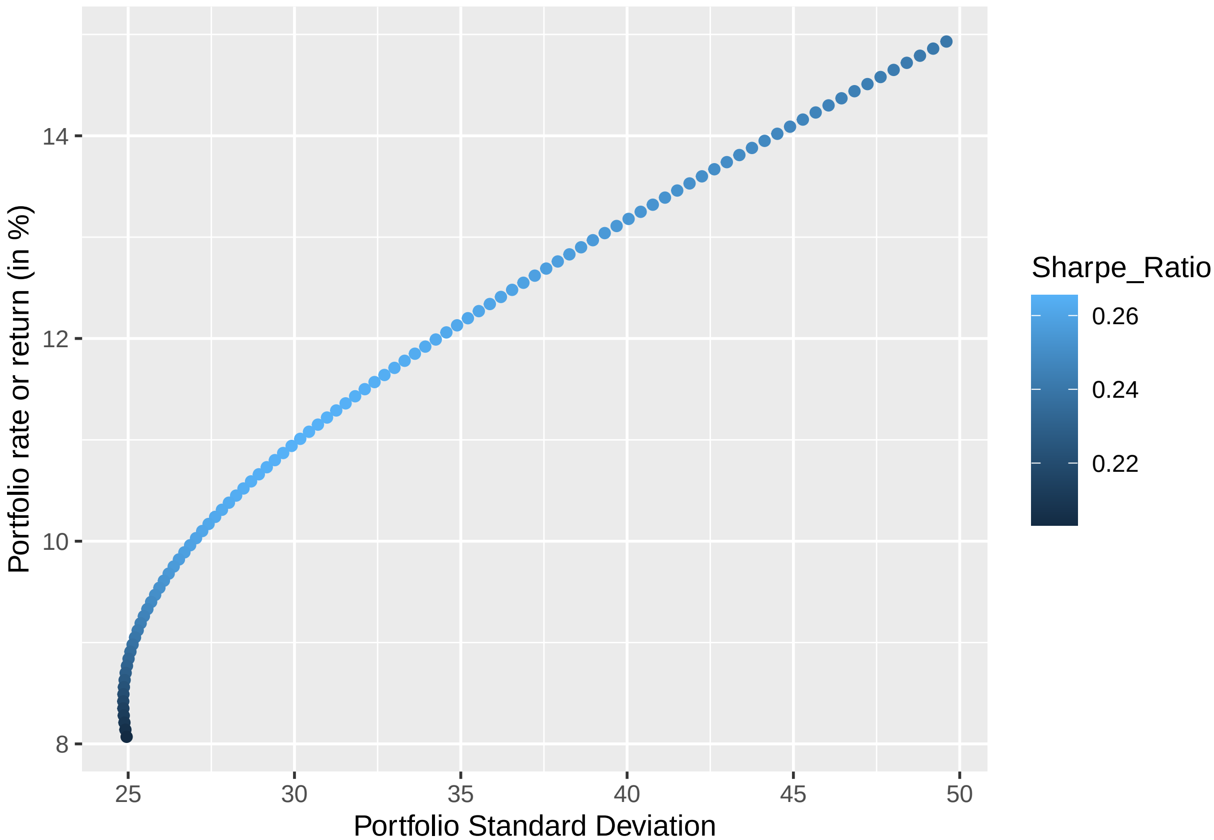
Now that the data has been created, let's create the charts in ggplot2 protip: download the ggplot2 cheat sheet https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

```r
# Let's use GGPLOT to create our charts, it gives us more customization abilities
# for example we can encode the sharpe ratio as color along the curve

ggplot(df, aes(x=Portfolio_SD, y=Portfolio_Return)) +
  geom_point(aes(color=Sharpe_Ratio)) +
  xlab("Portfolio Standard Deviation") +
  ylab("Portfolio rate or return (in %)")
```



Let's create the second chart without the option of shorting

```
# Careful, I'm using the filter function from the package dplyr, which is different from the standard
# one, therefore, load dyplr as library.

ggplot(filter(df, Large_Weight < 1, Small_Weight < 1), aes(x=Portfolio_SD, y=Portfolio_Return)) +
  geom_point(aes(color=Sharpe_Ratio)) +
  xlab("Portfolio Standard Deviation") +
  ylab("Portfolio rate or return (in %)")
```



How to generate the answers automatically

First let's create a second dataframe with the modified parameters and call it df2

```
# Risk Free rate
RF <- 3
# The large Weight is a sequence that goes from -1 to 2 in 0.01 steps
Large_Weight <- seq(-1,2,0.01)
# The Small Weight is simply 1 - Large Weight
Small_Weight <- 1 - Large_Weight

# Large return
LR <- 8
# Small return
SR <- 15

# Correlation (using CR because cor is an R funcion therefore reserved word)
CR <- -0.8

# Large Standard Deviation
LSD <- 25
# Small Standard Deviation
SSD <- 50

# lets create the dataframe. Note, R automatically recycle variables to fit the lenght of the dataframe
# therefore is important that the first variable is the correct lenght.

df2 <- data.frame("RF" = RF ,"Large_Weight" = Large_Weight, "Small_Weight" = Small_Weight, "LR" = LR, "S
```

```r
                 "CR" = CR, "LSD" = LSD, "SSD" = SSD)

# We could have also avoided all the variable declarations and just do:
# df <- data.frame("Large_Weight" = seq(-1,2,0.01), "Small_Weight" = 1 - seq(-1,2,0.01), "LR" = 8)

# Now let's add the calculated columns (note that column can be added just using df "$" name_of_the_col
# and we use the dollar sign to select the column by name as well
# Important Don't use spaces in the name of columns or variables or anything in general, it's just bad
# practice at best and in most cases the software will give you an error
df2$Portfolio_SD <- sqrt(df2$Large_Weight^2 * df2$LSD^2 + df2$Small_Weight^2 * df2$SSD^2 +
                         2*df2$CR*df2$Large_Weight*df2$LSD*df2$Small_Weight*df2$SSD)

df2$Portfolio_Return <- df2$Large_Weight*df2$LR + df2$Small_Weight*df2$SR

df2$Sharpe_Ratio <- (df2$Portfolio_Return - df2$RF)/df2$Portfolio_SD

# let's verify that our dataframe is created correctly
head(df2)
```

```
##   RF Large_Weight Small_Weight LR SR   CR LSD SSD Portfolio_SD Portfolio_Return
## 1  3        -1.00         2.00  8 15 -0.8  25  50     120.9339            22.00
## 2  3        -0.99         1.99  8 15 -0.8  25  50     120.2207            21.93
## 3  3        -0.98         1.98  8 15 -0.8  25  50     119.5075            21.86
## 4  3        -0.97         1.97  8 15 -0.8  25  50     118.7944            21.79
## 5  3        -0.96         1.96  8 15 -0.8  25  50     118.0813            21.72
## 6  3        -0.95         1.95  8 15 -0.8  25  50     117.3683            21.65
##   Sharpe_Ratio
## 1    0.1571107
## 2    0.1574604
## 3    0.1578143
## 4    0.1581724
## 5    0.1585348
## 6    0.1589015
```

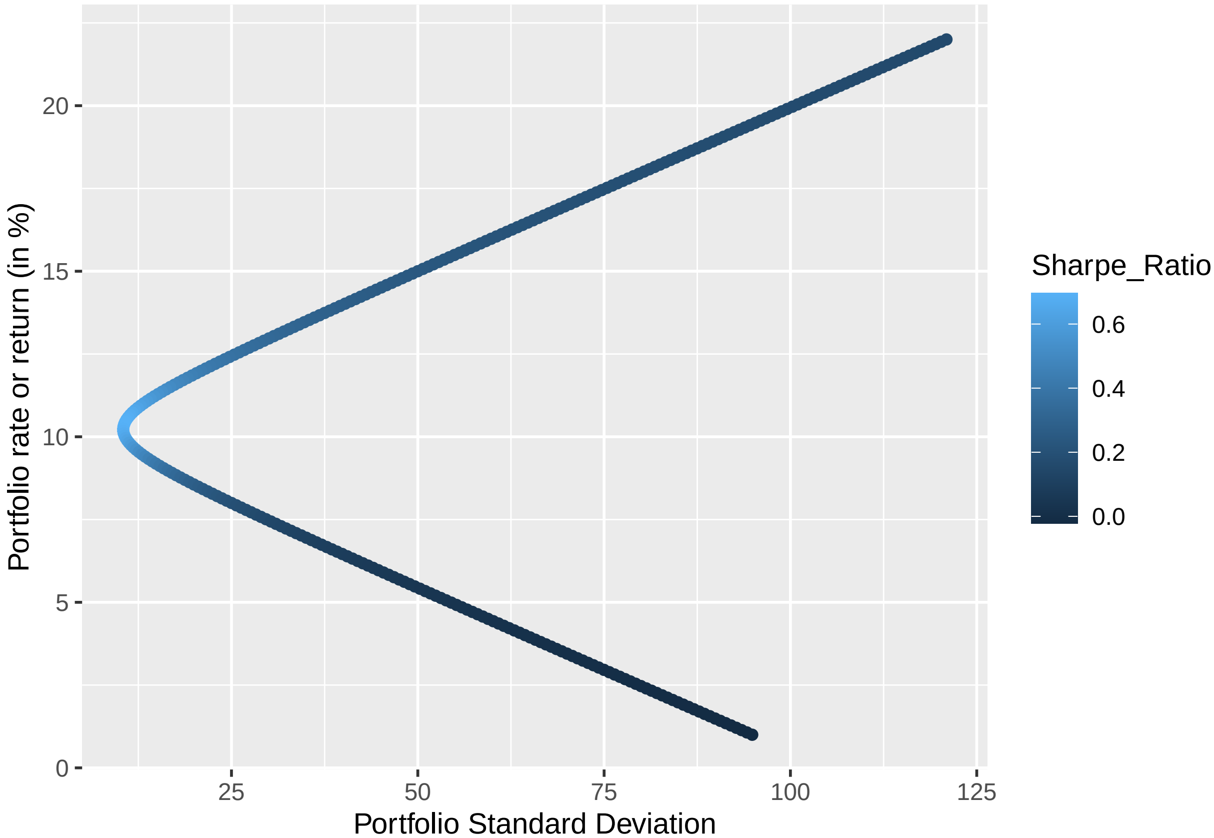
```r
tail(df2)
```

```
##     RF Large_Weight Small_Weight LR SR   CR LSD SSD Portfolio_SD
## 296  3         1.95        -0.95  8 15 -0.8  25  50     91.31162
## 297  3         1.96        -0.96  8 15 -0.8  25  50     92.02282
## 298  3         1.97        -0.97  8 15 -0.8  25  50     92.73410
## 299  3         1.98        -0.98  8 15 -0.8  25  50     93.44544
## 300  3         1.99        -0.99  8 15 -0.8  25  50     94.15685
## 301  3         2.00        -1.00  8 15 -0.8  25  50     94.86833
##     Portfolio_Return Sharpe_Ratio
## 296             1.35  -0.01806999
## 297             1.28  -0.01869102
## 298             1.21  -0.01930250
## 299             1.14  -0.01990466
## 300             1.07  -0.02049771
## 301             1.00  -0.02108185
```

Let see the modified chart

```r
# Let's use GGPLOT to create our charts, it gives us more customization abilities
# for example we can encode the sharpe ratio as color along the curve
```

```r
ggplot(df2, aes(x=Portfolio_SD, y=Portfolio_Return)) +
  geom_point(aes(color=Sharpe_Ratio)) +
  xlab("Portfolio Standard Deviation") +
  ylab("Portfolio rate or return (in %)")
```



And now overlay them. In R this is not very automated, please follow the code This example is simplified by the fact that the portfolio rate of return does not depend on the correlation coefficent but only by the weights, so we can use it "as is" for our Y axis

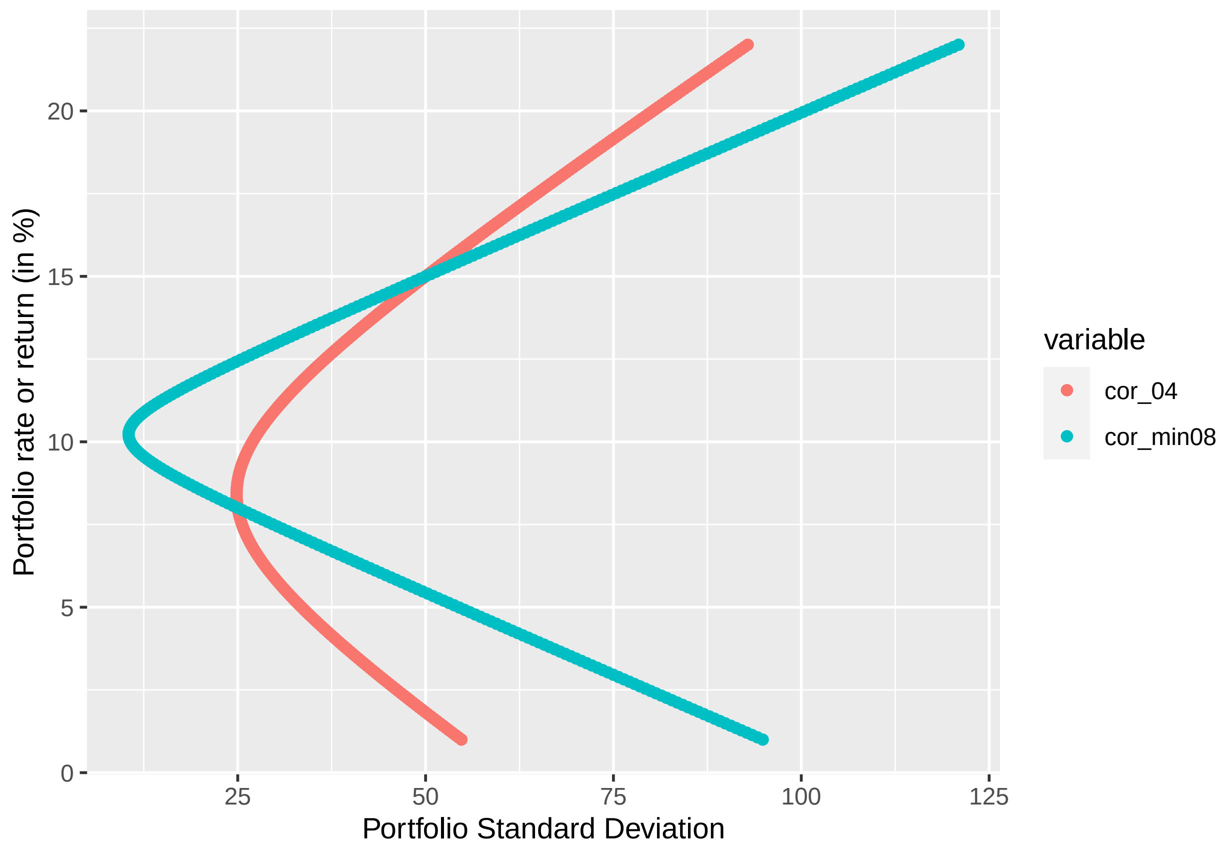
```r
# first we need to create a dataframe with the information we need.
# the portfolio rate of return is our Y and cor_04 and cor_min08 are the standard deviations

# create a simple dataframe with only the variables of interest (this step is not necessary but it is e
overlay_df <- data.frame(Y=df$Portfolio_Return, cor_04=df$Portfolio_SD, cor_min08=df2$Portfolio_SD)

# melt the dataframe to stack the variables in a single column
overlay_df.m <- melt(overlay_df, id.vars = "Y", measure.vars = c("cor_04", "cor_min08"))

# double check the stacking is correct
head(overlay_df.m)
```

```
##       Y variable    value
## 1 22.00   cor_04 92.87088
## 2 21.93   cor_04 92.42680
## 3 21.86   cor_04 91.98288
## 4 21.79   cor_04 91.53913
## 5 21.72   cor_04 91.09555
## 6 21.65   cor_04 90.65215
```

```r
tail(overlay_df.m)
```

```
##        Y  variable    value
## 597 1.35 cor_min08 91.31162
## 598 1.28 cor_min08 92.02282
## 599 1.21 cor_min08 92.73410
## 600 1.14 cor_min08 93.44544
## 601 1.07 cor_min08 94.15685
## 602 1.00 cor_min08 94.86833
```

```r
# plot reversing the axis (we put Y where X is expected and so on)
ggplot(overlay_df.m, aes(value, Y, colour = variable)) +
  geom_point() +
  xlab("Portfolio Standard Deviation") +
  ylab("Portfolio rate or return (in %)")
```



Question 1: Find the composition of the portfolio at minimum variance

```r
# the function filter from dyplr is very useful in this case.
filter(df2, Portfolio_SD == min(Portfolio_SD))
```

```
##   RF Large_Weight Small_Weight LR SR   CR LSD SSD Portfolio_SD Portfolio_Return
## 1  3         0.68         0.32  8 15 -0.8  25  50     10.47855            10.24
##   Sharpe_Ratio
## 1    0.6909353
```

Question 2:

Question 3 shows something that is very common… what if there isn't a portfolio return exactly equal to 25.

Note: -check the numbers, I'm giving the template, not the solution can be approached in 2 ways. First, a brute force approach using the filter function from dyplr Second, we can calculate exactly the

Question 3:

```
# Test if there is a return exactly equal to 25.
filter(df2, Portfolio_Return == 25)
```

```
##  [1] RF               Large_Weight     Small_Weight     LR
##  [5] SR               CR               LSD              SSD
##  [9] Portfolio_SD     Portfolio_Return Sharpe_Ratio
## <0 rows> (or 0-length row.names)
```

```
# then we just filter the portfolio around the value and we select the correct row
filter(df2, Portfolio_Return > 24.5 & Portfolio_Return < 25.5)
```

```
##  [1] RF               Large_Weight     Small_Weight     LR
##  [5] SR               CR               LSD              SSD
##  [9] Portfolio_SD     Portfolio_Return Sharpe_Ratio
## <0 rows> (or 0-length row.names)
```

```
# or we just use a neat function "which.min" which returns the row where the absolute value
# of the difference between the Portfolio_Return and 20 is at its minimum

row.number <- which.min(abs(df2$Portfolio_Return - 25))

# and we verify that is correct.
df2[row.number,]
```

```
##   RF Large_Weight Small_Weight LR SR   CR LSD SSD Portfolio_SD Portfolio_Return
## 1  3           -1            2  8 15 -0.8  25  50     120.9339               22
##   Sharpe_Ratio
## 1    0.1571107
```