CVaR-Optimization

Jesse Keränen

1/30/2022

Prologue

In PortolioOptimization and Markowitz files I optimized portfolio formed from big tech companies. Then I constructed basic mean-variance portfolios. There is also other ways to measure risk of the portfolio than just variance. One example is value at risk or conditional value at risk. I have made separate files from these both also. In this file I am going to combine what I have learned in these previous files. I am going to optimize a portfolio containing the exact same stocks as before but this time I will minimize CVaR of the portfolio instead of variance of the portfolio. I will use CVaR instead of VaR, because it can be argued that it is better risk measure and because it is much easier to implement to optimization formula. I will first try to show how CVaR optimization problem can be formulized to a linear programming problem. Then I try to implement this problem to solver from CVXR library.

Conditional Value at Risk

Value at risk is the maximum loss we will face with given confidence level. Conditional value at risk, or expected shortfall, on the other hand gives average loss beyond value at risk. According to Rockafellar and Uryasev (2000) we can use following auxiliary function to minimize CVaR for given set of stocks:

$$F_\alpha(x,\zeta) = \zeta + \alpha^{-1} \int_{l(x,w)<\zeta} [l(x,w) - \zeta] p(w) dw$$

Where α is our confidence level, ζ is VaR at given α and $l(w,x)$ is so called loss function that gives us distribution of the portfolio returns with given weights. Because we use finite amount of simulations to estimate loss function we can write our objective function in form of:

$$F_\alpha(x,\zeta) = \alpha^{-1} \sum_{j=1}^{J} \pi_j [l(x,w) - \zeta] - \zeta$$

Here π is the probability of single simulation. We can also replace $[l(x,w) - \zeta$ with variable z and add contraints to z:

$$F_\alpha(x,\zeta) = \alpha^{-1} \sum_{j=1}^{J} \pi_j z_j - \zeta$$
$$s.t.\ z_j \geq \zeta - f(x,w)$$
$$z_j \geq 0$$

Due to our formulation z will have value of 0 when given simulation results higher return than VaR and when given simulation value is less than VaR z will get value of difference between VaR and that simulation. We don't have to define our VaR or ζ before, but our optimization will tell us that. Why is that? VaR has negative sign in our objective function and because of that solver has insentive to set it as big as possible.

We also have sum of z's in our objective function with positive sign. If it would be possible solver would set all z's to 0, but since we have constraint that z_i must be greater or equal to $\zeta - f(x, w)$ this would only be possible if ζ would be relatively small. So the solver has to find optimal trade off between ζ and sum of z's. Optimal value for ζ will be equal to VaR at given α. Later in this file I try to show why our ζ should converge to VaR.

There are multiple different ways to calculate VaR and CVaR. I think that this problem will be easier to solve if estimate distribution of stock returns with finite number of iterations. That's why we are going to use Monte Carlo-method that relies on simulating large amount of stock returns and expecting them to represent distribution of that stocks returns. We will use geometric Brownian motion with Cholesky's decomposition to create set of correlated stock return simulations.

```
library(ggrepel)
library(ggplot2)
library(Quandl)
library(tidyverse)
library(data.table)
library(moments)
library(CVXR)
library(scales)

set.seed(13)

Quandl.api_key("bx1qdehfWXg6SNKnicQC")

names <- c("AAPL", "MSFT", "TSLA", "GOOGL", "AMZN")
weights <-c(0.3, 0.1, 0.2, 0.2, 0.2)

# For monthly data use collapse = "monthly"
prices <- as.data.table(Quandl(c("WIKI/AAPL.11","WIKI/MSFT.11",
        "WIKI/TSLA.11", "WIKI/GOOGL.11", "WIKI/AMZN.11"), start_date
        = "2010-06-01", end_date = "2015-03-01", collapse = "monthly"))

colnames(prices) <- c("Date", "AAPL", "MSFT", "TSLA", "GOOGL", "AMZN")

# Tidying data table
prices <- melt(prices, id.vars = "Date", measure.vars = names, variable.name
               = "Company", value.name = "Adj_Close")

# Calculate simple price changes
prices[, "Returns" := Adj_Close/shift(Adj_Close) -1, by = Company]
prices <- drop_na(prices)
prices[, Weight := 0.2]

mean_std <- prices[, .(means = mean(Returns), stds = sd(Returns)), by= Company]

corr_matrix <- cor(as.data.table(split(prices[,4] ,prices$Company)))
cov_matrix <- cov(as.data.table(split(prices[,4] ,prices$Company)))
time <- 1

datalist <- list()

# Simulate stock prices
for (i in 1:15000){
```

```
  random_num <- replicate(n=5, qnorm(runif(1)))
  Cholesky_Decomposition <- chol(corr_matrix)
  corr_rand <- random_num %*% Cholesky_Decomposition
  simul_prices <- (100*exp((mean_std$means-0.5*(mean_std$stds^2))*
                  time+mean_std$stds*sqrt(time)*corr_rand))
  datalist[[i]] <- simul_prices
}
big_data = as.data.table(do.call(rbind, datalist))
simul_ret <- big_data/100 - 1


# Define variables and constants
alpha <- 0.05
pi <- 1/nrow(simul_ret)
x <- Variable(length(names))
z <- Variable(nrow(simul_ret))
zeta <- Variable(1)

# Objective function
objective <- Minimize((1/(alpha))*pi*sum(z) - zeta)

# Constraints
constraints <- list(
  x >= 0,
  sum(x) == 1,
  z >= 0,
  z >= zeta - as.matrix(simul_ret) %*% x
)
problem <- Problem(objective, constraints)

# Results
ans<-solve(problem)
print(ans$value)
```

```
## [1] 0.06833569
```

```
print(round(ans$getValue(x), 2))
```

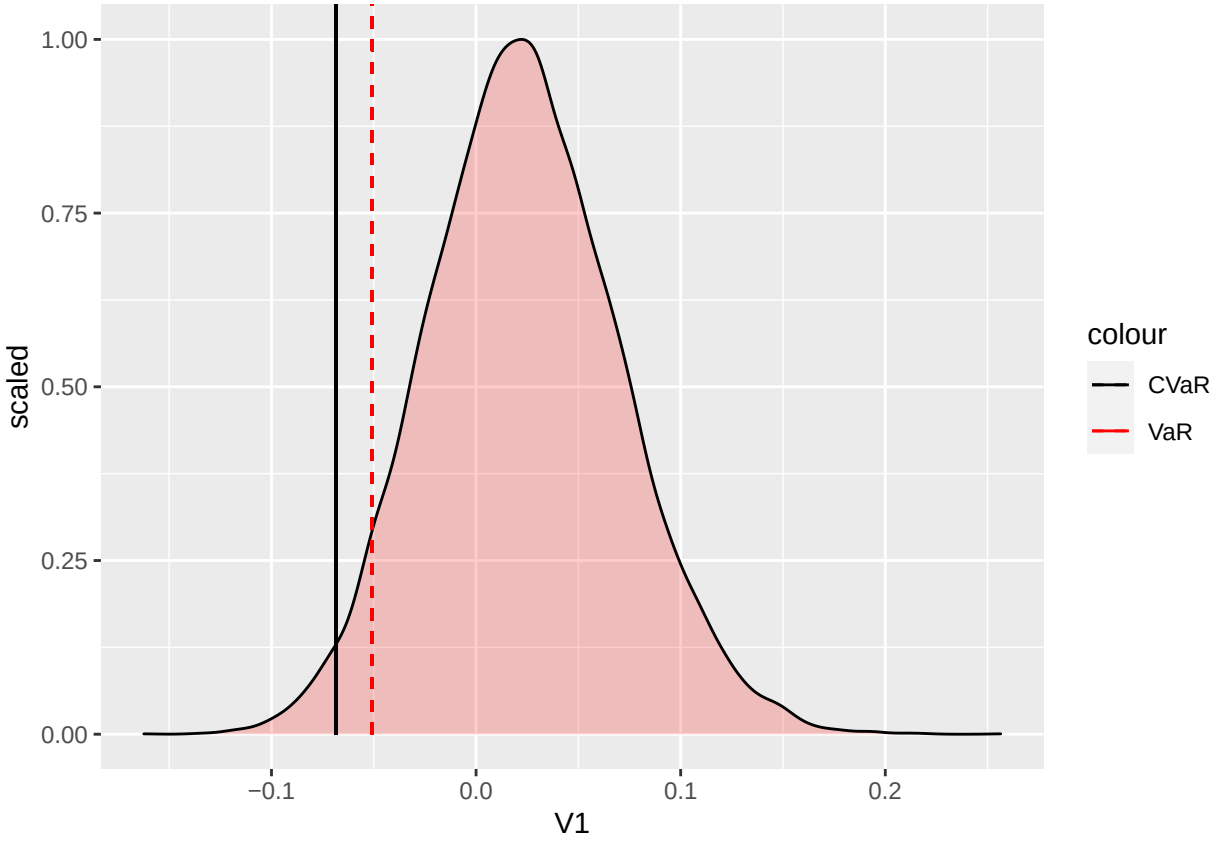
```
##      [,1]
## [1,] 0.16
## [2,] 0.41
## [3,] 0.07
## [4,] 0.13
## [5,] 0.23
```

```
print(round(ans$getValue(zeta), 8))
```

```
## [1] -0.0507577
```

```
p <- as.data.table(as.matrix(simul_ret) %*% ans$getValue(x))
```

```
ggplot(p, aes(V1)) + geom_density(aes(y = ..scaled..,), fill = "red", alpha =
                                  0.2) +
  geom_segment(aes(x=ans$getValue(zeta), xend=ans$getValue(zeta), y=0, yend =
              Inf, color = "VaR"), linetype = "dashed") +
  geom_segment(aes(x=-ans$value, xend=-ans$value, y=0, yend = Inf, color =
              "CVaR"), linetype = "solid", legend = "CVaR") +
  scale_color_manual(values = c("CVaR" = "black", "VaR" = "red"))
```



For me it was first a bit unclear why we didn't have to set value for VaR, but instead we just created variable ζ. This is how I have understood why ζ converges to VaR. Let's say that ζ is in optimal value VaR and we increase it by b. Let's also assume that increase b is so small that no additional z will become different from zero. We can write our objective function as:

$$\alpha^{-1}\sum(\zeta - z_j + b)\pi - (\zeta + b)$$
$$\frac{\sum(\zeta - z_j)\pi}{\alpha} + \frac{\sum b\pi}{\alpha} - \zeta - b$$

Now we can see that direction where our objective function value changes depends ratio between second and last term. We assumed that no additional z's become different from zero. That means that terms we sum doesn't change. If we denote total number of iterations by X then π is one divided by X and number of z's different from zero is X times α:

$$\frac{\sum(\zeta - z_j)\pi}{\alpha} + \frac{b\frac{1}{X}\alpha X}{\alpha} - \zeta - b$$
$$\frac{\sum(\zeta - z_j)\pi}{\alpha} + b - \zeta - b$$
$$\frac{\sum(\zeta - z_j)\pi}{\alpha} - \zeta$$

So our objective function value doesn't change. If we have extremely many simulation (close to continuous distribution) making so little increase that id wouldn't increase number of z's different from zero comes unlikely. What I mean that if we increase our ζ remarkably there most likely are iterations that are between our old ζ and new $\zeta + b$. These values result in z's that are different from zero. If number of z's different from zero increases, i.e. number of terms we sum increases, total effect to our objective function will be positive. This is not improvement since we are minimizing our objective function. In that case both our first term increases as well as our second term. I want to underline that this is just my attempt to understand this and I can't guarantee that my prove would hold mathematically.

We previously estimated that we could form minimum variance portfolio from above stocks by following weights:

$$AAPL\ 0.13865260$$

$$MSFT\ 0.43687927$$

$$TSLA\ 0.04831945$$

$$GOOGL\ 0.16201185$$

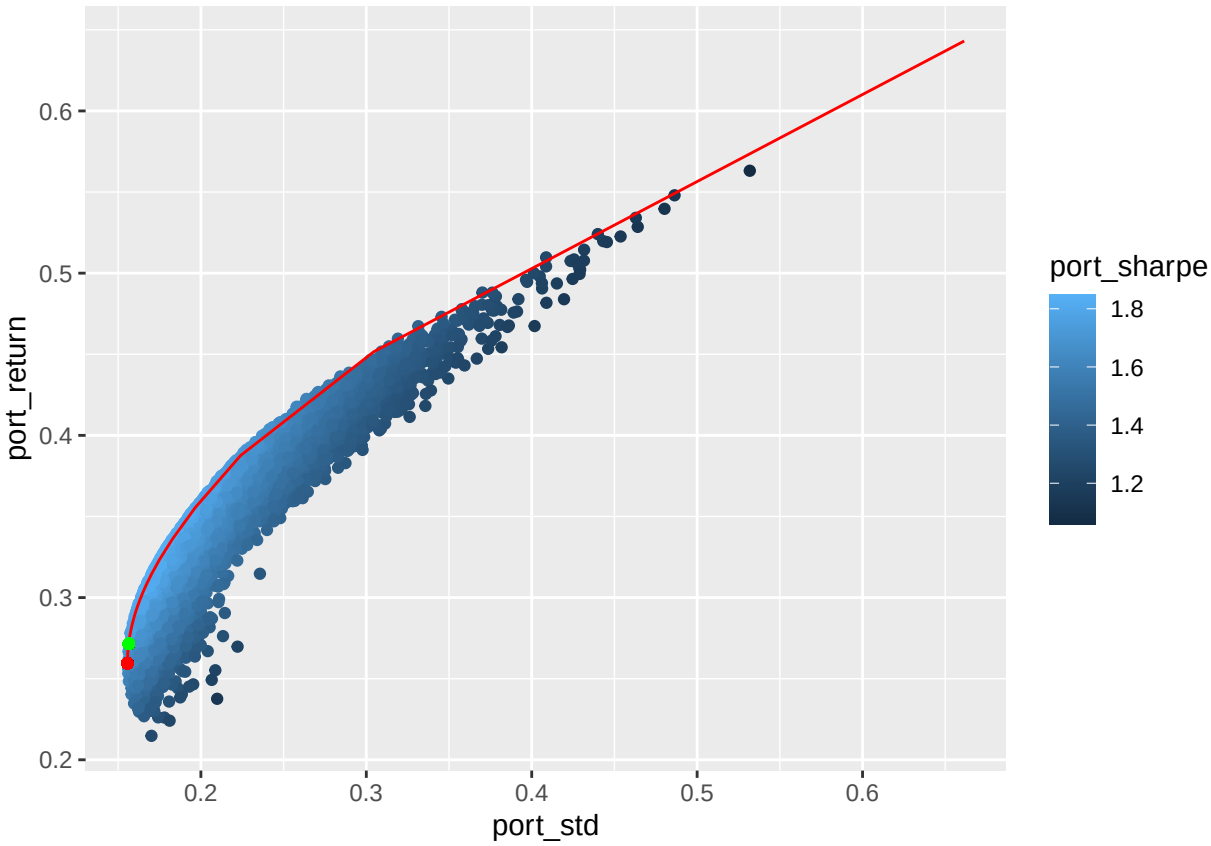
$$AMZN\ 0.21413684$$

These weights are pretty similar to what we got for minimum CVaR portfolio, but not identical. I would take that as a hint that our optimization succeeded. One important note is that if we don't use $set.seed()$ command every time that we run the code we get different results. This is normal since we have included random factors in our stock return simulation. We can take our CVaR optimal weights and calculate portfolio return and standard deviation similar as in mean variance optimization and plug it to our efficient frontier plot. Note that we used simulated returns in the optimization, but now I use historical returns to estimate portfolio return and standard deviation so that our results are line with other points in out efficient frontier set.

```
means <- prices[, mean(Returns), by = Company]
portf_return <- as.numeric(means$V1 %*% ans$getValue(x) * 12)
portf_sd <- as.numeric(sqrt(t(ans$getValue(x)) %*% cov_matrix %*% ans$getValue(x)) * sqrt(12))

# Plot from portfolio optimization file
plot <- readRDS("extdata/efficient_frontier.rds")

ggplot(data = plot$data) +
geom_point(data = plot$data, aes(port_std, port_return, color = port_sharpe)) +
geom_point(aes(plot$layers[[2]]$data$sd, plot$layers[[2]]$data$return),
           color = "red") +
geom_line(data = plot$layers[[4]]$data, aes(plot$layers[[4]]$data$V2,
          plot$layers[[4]]$data$V1), color = "red") +
geom_point(aes(x = portf_sd, y = portf_return), color = "green")
```



Epilogue

Minimum CVaR portfolio seems to be more riskier than minimum variance portfolio. We have to remember that we set our confidence level to 0.05. Changing confidence level changes portfolios position on efficient frontier. We have to take that into consideration when comparing portfolio from different optimizations. On the other hand in 0.05 confidence level our minimum CVaR portfolio was already pretty close to minimum variance portfolio.