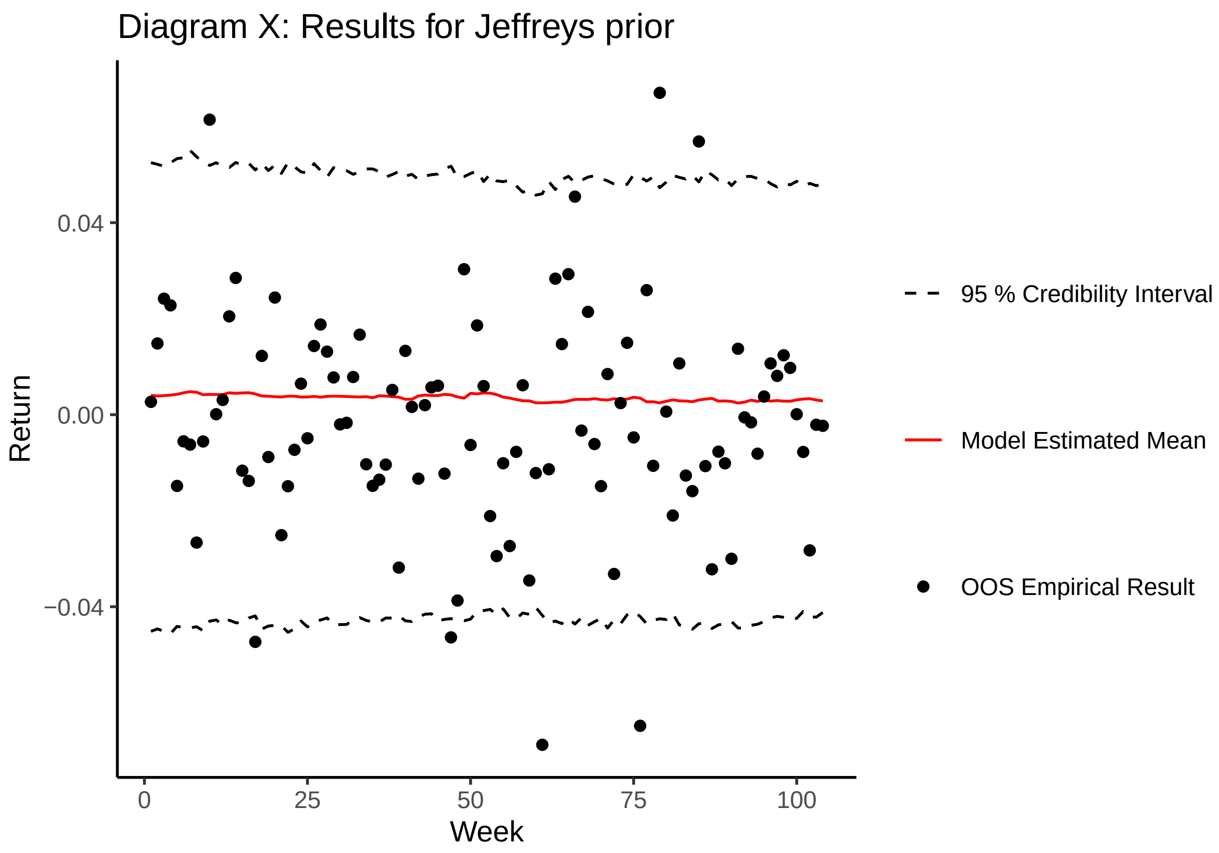
Skiss optimering

Sam Linderoth

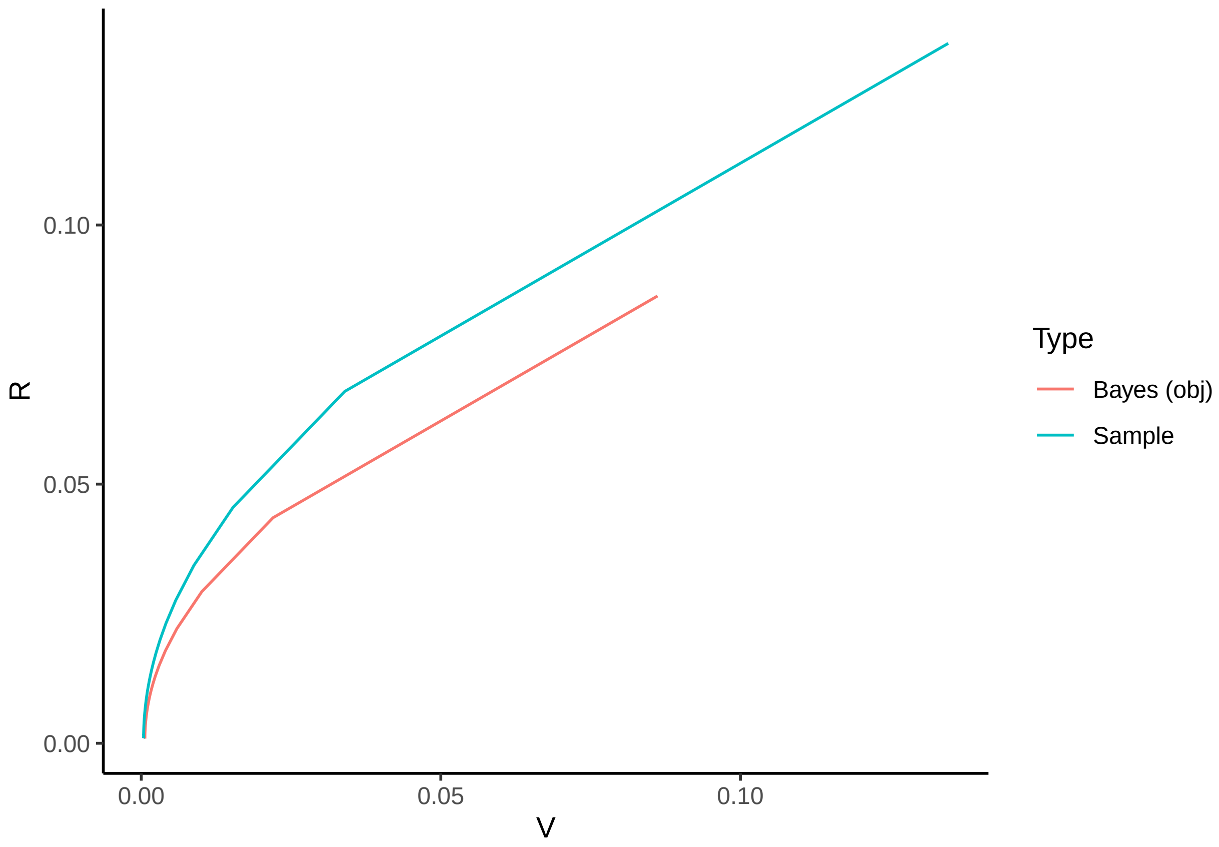
2023-05-30

```r
library(tidyverse)
library(quantmod)

# Function to optimize portfolio
opti <- function(x_bar, rets, gamma, type, d = nrow(rets), r = nrow(rets)){

  n <- nrow(rets)
  k <- ncol(rets)


  if(type == "sample"){
    c_kn <- 1/(n-1)
  }

  if (type == "jeff"){
   # Coefficient
    c_kn <- 1/(n-k-1) + (2*n - k - 1)/(n*(n-k-1)*(n-k-2))
  }

  if (type == "conj"){
   # Coefficient
    c_kn <- 1/(n+d-2*k-1) + (2*n+r+d-2*k-1)/((n+r)*(n+d-2*k-1)*(n+d-2*k-2))
  }

  # Covariance matrix
  S <- (n+1)*cov(rets)

  # Invert
  S_inv <- solve(S)

  # Vector of ones
  ones <- matrix(1:1, nrow = k, ncol = 1)

  # Normalization denominator
  den <- as.vector(t(ones) %*% S_inv %*% ones)

  # Q matrix
  Q <-  S_inv - (S_inv %*% ones %*% t(ones) %*% S_inv)/
                den

  # Quantites for MV
  w_mv <- (S_inv %*% ones)/den + gamma^(-1)*c_kn^(-1)*Q %*% x_bar
  r_mv <- (t(ones) %*% S_inv %*% x_bar)/den +
        (t(x_bar) %*% Q %*% x_bar)/(gamma*c_kn)

  v_mv <- c_kn/den +
          (t(x_bar) %*% Q %*% x_bar)/(gamma^2 * c_kn)

  return(list(ret = r_mv, var = v_mv, weights = w_mv, covmat = S, typeof = type))

}


# Simulation
sim_jeff <- function(weights, x_bar, n_samp, cov, n, k){

  samp_store <- c()

  for (i in 1:n_samp){

    t_1 <- rt(n = 1, df = n-k)
    t_2 <- rt(n = 1, df = n-k+1)

    samp <- t(weights) %*% x_bar +
            (sqrt(t(weights) %*% cov %*% weights))*(t_1/(sqrt(n*(n-k))) +
                                          sqrt(1 + t_1^2/(n-k))*(t_2/sqrt(n-k+1)))

    samp_store <- c(samp_store, samp)

  }

  return(samp_store)

}

sim_con <- function(weights, x_bar, n_samp, cov, n, k, d = n, r = n){

  samp_store <- c()

  for (i in 1:n_samp){

    t_1 <- rt(n = 1, df = n+d-2*k)
    t_2 <- rt(n = 1, df = n+d-2*k+1)

    samp <- t(weights) %*% x_bar +
            (sqrt(t(weights) %*% cov %*% weights))*(t_1/(sqrt((n+r)*(n+d-2*k))) +
                                          sqrt(1 + t_1^2/(n+d-2*k))*(t_2/sqrt(n+d-2*k+1)))

    samp_store <- c(samp_store, samp)

  }

  return(samp_store)

}
```



```
## [1] "var"

## [1] 0.000636665

## [1] 0.0005619879

## [1] 0.0005054087

## [1] 0.0005035083

## [1] "mean"

## [1] 0.004797881

## [1] 0.003507001

## [1] 0.00242399

## [1] -0.001603114
```

```
df_bay <- data.frame()
df_samp <- data.frame()

# Efficient frontier

for (i in 1:500){

  #i <- i/10

  d_0 <- 5
  r_0 <- 5

  bay_eff <- opti(meanvector, return_mat, i, type = "conj", d = d_0, r = r_0)
  sample_eff <- opti(meanvector, return_mat, i, type = "sample")

  df_bay <- rbind(df_bay, c(bay_eff[1], bay_eff[2]))
  df_samp <- rbind(df_samp, c(sample_eff[1], sample_eff[2]))

}

df_bay <- cbind(df_bay, rep("Bayes (obj)",nrow(df_bay)))
colnames(df_bay) <- c("ret", "var", "type")

df_samp <- cbind(df_samp, rep("Sample",nrow(df_samp)))
colnames(df_samp) <- c("ret", "var", "type")

df <- rbind(df_bay, df_samp)

colnames(df) <- c("ret", "var", "type")

ggplot(df,
       aes(x = var,
           y = ret,
           color = as.factor(type))) +
  geom_line() +
  theme_classic() +
  labs(x = "V",
       y = "R") +
  guides(color = guide_legend(title="Type"))
```


0.10
0.05
0.00
R
0.00
0.05
0.10
V
Type
Bayes (obj)
Sample