# Multivariate Normal distribution

Suppose that we want to generate data from a Multivariate Normal distribution with a specific covariance structure (nonzero correlations).

The PDF of a Multivariate Normal distribution is:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \mid \boldsymbol{\Sigma} \mid^{1/2}} \exp\left( -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

with $\boldsymbol{\mu} = (\mu_1, ..., \mu_d)^T$ and $\boldsymbol{\Sigma}$ is a $d$ x $d$ symmetric positive definite covariance matrix whose general element $cov(X_i, X_j) = \rho_{ij} s_i s_j$

# Cholesky factorization algorithm

The Cholesky algorithm to simulate random variates from a $N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the following:

1. Generate an $n$ x $d$ matrix $\mathbf{Z}$ containing $nd$ random variates $N(0, 1)$

2. Decompose the covariance matrix using Cholesky factorization $\boldsymbol{\Sigma} = \mathbf{Q^T Q}$ , where $\mathbf{Q}$ is an upper triangular matrix.

3. Apply the transformation $\mathbf{X} = \mathbf{ZQ} + \mathbf{1_n}\boldsymbol{\mu}^T$, where $\mathbf{1_n}$ is a column vector of ones of size $n$.

4. Return the $n$ x $d$ matrix $\mathbf{X}$ for which each row is a random vector from the $N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

# Smulate a portfolio of four assets

**Example** *All numbers are rounded* Suppose that we want to generate a random sample of size $n = 10,000$ from a $N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with the following mean vector and covariance matrix, corresponding to means and covariances of the log returns of four assets (see conde on slide 6):

$$\boldsymbol{\mu} = \begin{bmatrix} 0.0036 & 0.0012 & -0.0001 & 0.0004 \end{bmatrix}^T \quad \text{and}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.0022 & 0.0006 & 0.0005 & 0.0005 \\ 0.0006 & 0.0005 & 0.0004 & 0.0004 \\ 0.0005 & 0.0004 & 0.0008 & 0.0004 \\ 0.0005 & 0.0004 & 0.0004 & 0.0005 \end{bmatrix}$$

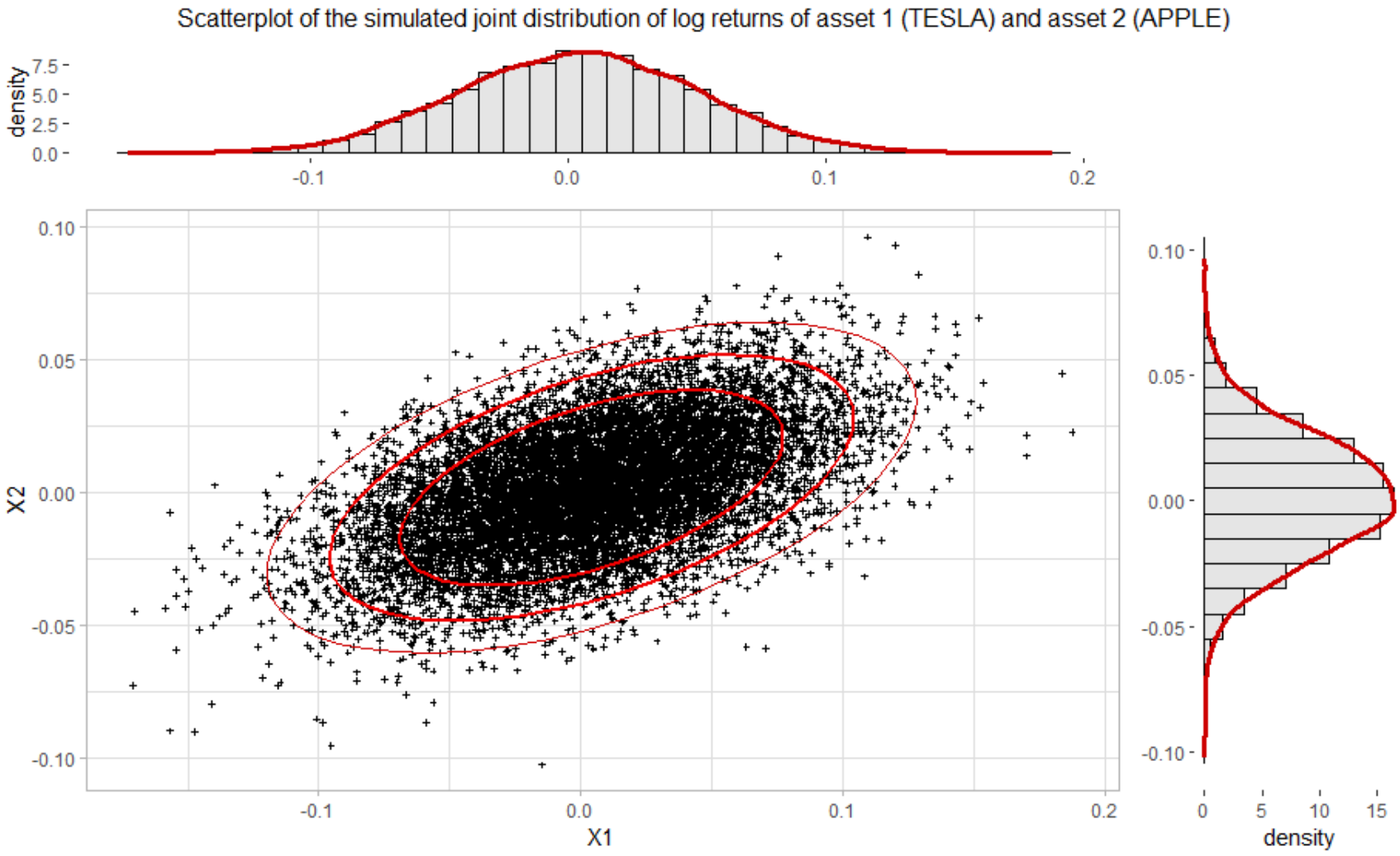
We then defined the standard deviations and correlation coefficients:

$$\rho_{12} = \rho_{21} = 0.52 \qquad \rho_{13} = \rho_{31} = 0.39 \qquad s_1 = 0.04673$$
$$\rho_{14} = \rho_{41} = 0.46 \qquad \rho_{23} = \rho_{32} = 0.64 \qquad s_2 = 0.02341$$
$$\rho_{24} = \rho_{42} = 0.64 \qquad s_3 = 0.02907$$
$$\rho_{34} = \rho_{43} = 0.62 \qquad s_4 = 0.02322$$

# Cholesky factorization in R

```
# parameters
n <- 10000
assets <- matrix(cbind(TSLA$TSLA.Adjusted, AAPL$AAPL.Adjusted, META$META.Adjusted,
                       AMZN$AMZN.Adjusted), byrow = FALSE, ncol = 4)
colnames(assets) <- c("TESLA", "APPLE", "META", "AMZN")
Assets <- cor(assets) # correlations
rho_12 <- Assets[1,2]; rho_13 <- Assets[1,3]; rho_14 <- Assets[1,4]
rho_23 <- Assets[2,3]; rho_24 <- Assets[2,4]; rho_34 <- Assets[3,4]

sds = apply(assets,2, sd) # standard deviations
s1 <- sds[[1]]; s2 <- sds[[2]]; s3 <- sds[[3]]; s4 <- sds[[4]]
mu <- as.numeric(colMeans(assets)) # means

# covariance matrix, eg. cov_12 = s1*s2*rho_12
Sigma <- round(matrix(c(s1^2, s1*s2*rho_12, s1*s3*rho_13, s1*s4*rho_14,
                        s1*s2*rho_12, s2^2, s2*s3*rho_23, s2*s4*rho_24,
                        s1*s3*rho_13, s2*s3*rho_23, s3^2, s2*s4*rho_24,
                        s1*s4*rho_14, s2*s4*rho_24,s3*s4*rho_34,s4^2), nrow = 4))

# function to sample from a Multivariate Normal distribution
mv.cholesky <- function(n, mu, Sigma) {
  d <- length(mu)
  Q <- chol(Sigma)
  Z <- matrix(rnorm(n*d), nrow = n, ncol = d)
  X <- Z %*% Q +  rep(1,n) %*% t(mu)
  X <- data.frame(X)
  return(X)
}
# call the function and create a sample of desired size
set.seed(1986)
random.sample <- mv.cholesky(n = n, mu = mu, Sigma = Sigma)
```

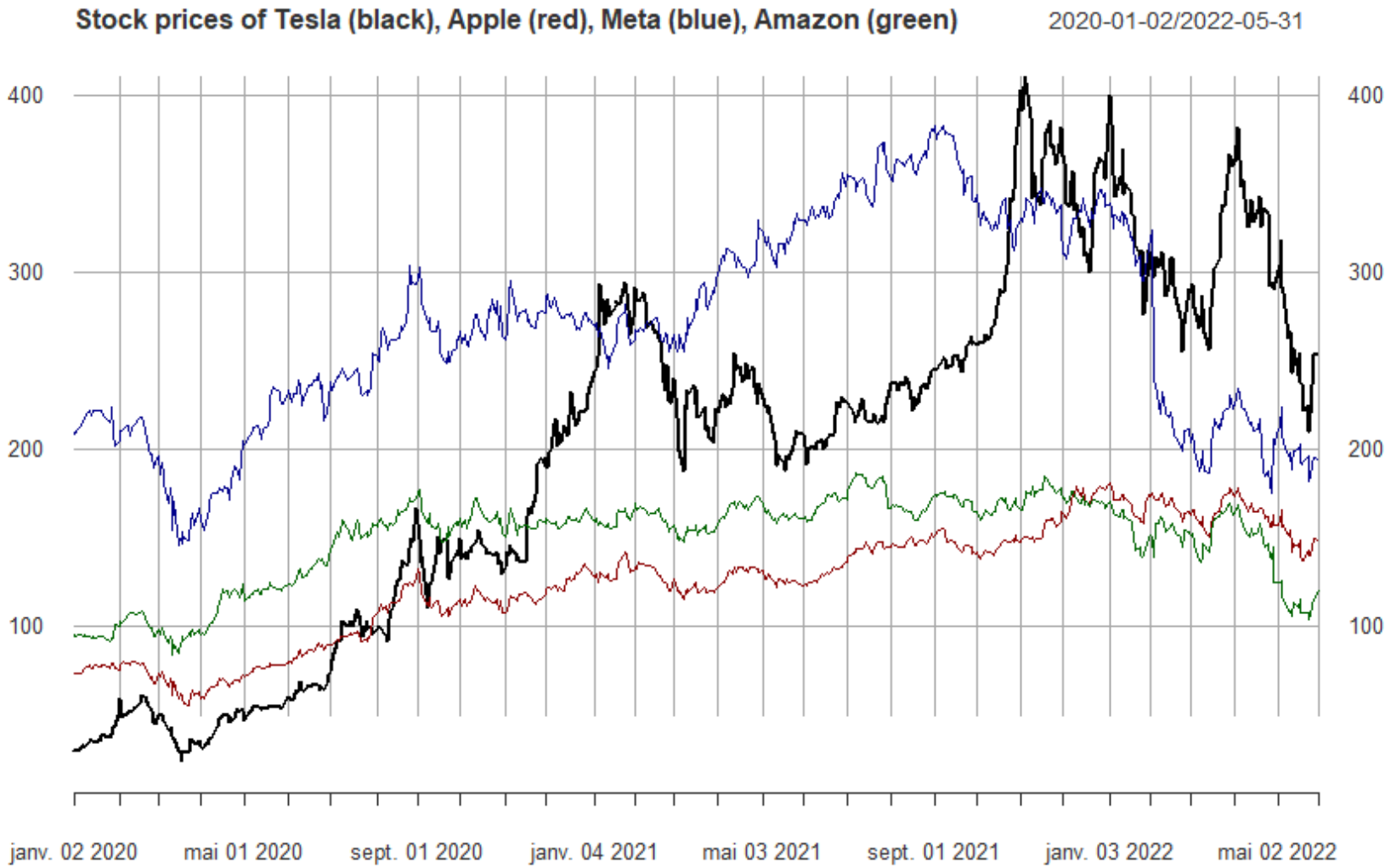
# Scatterplot of a joint distribution



# R code for retrieving and plotting stock prices of different assets

```
library(quantmod)
# retrieve stock prices from Yahoo finance
# Tesla, Inc.
TSLA <- getSymbols("TSLA", src = "yahoo", from = "2020-01-01", to = "2022-06-01",
                   auto.assign = FALSE)
# Apple Inc.
AAPL <- getSymbols("AAPL", src = "yahoo", from = "2020-01-01", to = "2022-06-01",
                   auto.assign = FALSE)
# Meta Platforms, Inc.
META <- getSymbols("META", src = "yahoo", from = "2020-01-01", to = "2022-06-01",
                    auto.assign = FALSE)
# Amazon.com, Inc.
AMZN <- getSymbols("AMZN", src = "yahoo", from = "2020-01-01", to = "2022-06-01",
                    auto.assign = FALSE)
# time series plot of the different stocks
plot(TSLA$TSLA.Adjusted, main = 'Stock prices ...')
lines(AAPL$AAPL.Adjusted, col = 'darkred')
lines(META$META.Adjusted, col = 'darkblue')
lines(AMZN$AMZN.Adjusted, col = 'darkgreen')
}
# daily log returns
TSLAreturns = Return.calculate(TSLA[,6],method="log")
TSLAreturns = TSLAreturns[(-1)]
AAPLreturns = Return.calculate(AAPL[,6],method="log")
AAPLreturns = AAPLreturns[(-1)]
METAreturns = Return.calculate(META[,6],method="log")
METAreturns = METAreturns[(-1)]
AMZNreturns = Return.calculate(AMZN[,6],method="log")
AMZNreturns = AMZNreturns[(-1)]
```

# Time series plot of different stock prices



# Markowitz mean-variance criterion: rationale

1. Main assumption: investors are risk-averse and therefore we have to account for the risk-return tradeoff in our portfolio selection. For a given return, an investor will chose the portfolio associated with the lowest risk.

2. The log return is computed as $r_t = log\left(\frac{P_t}{P_{t-1}}\right)$.

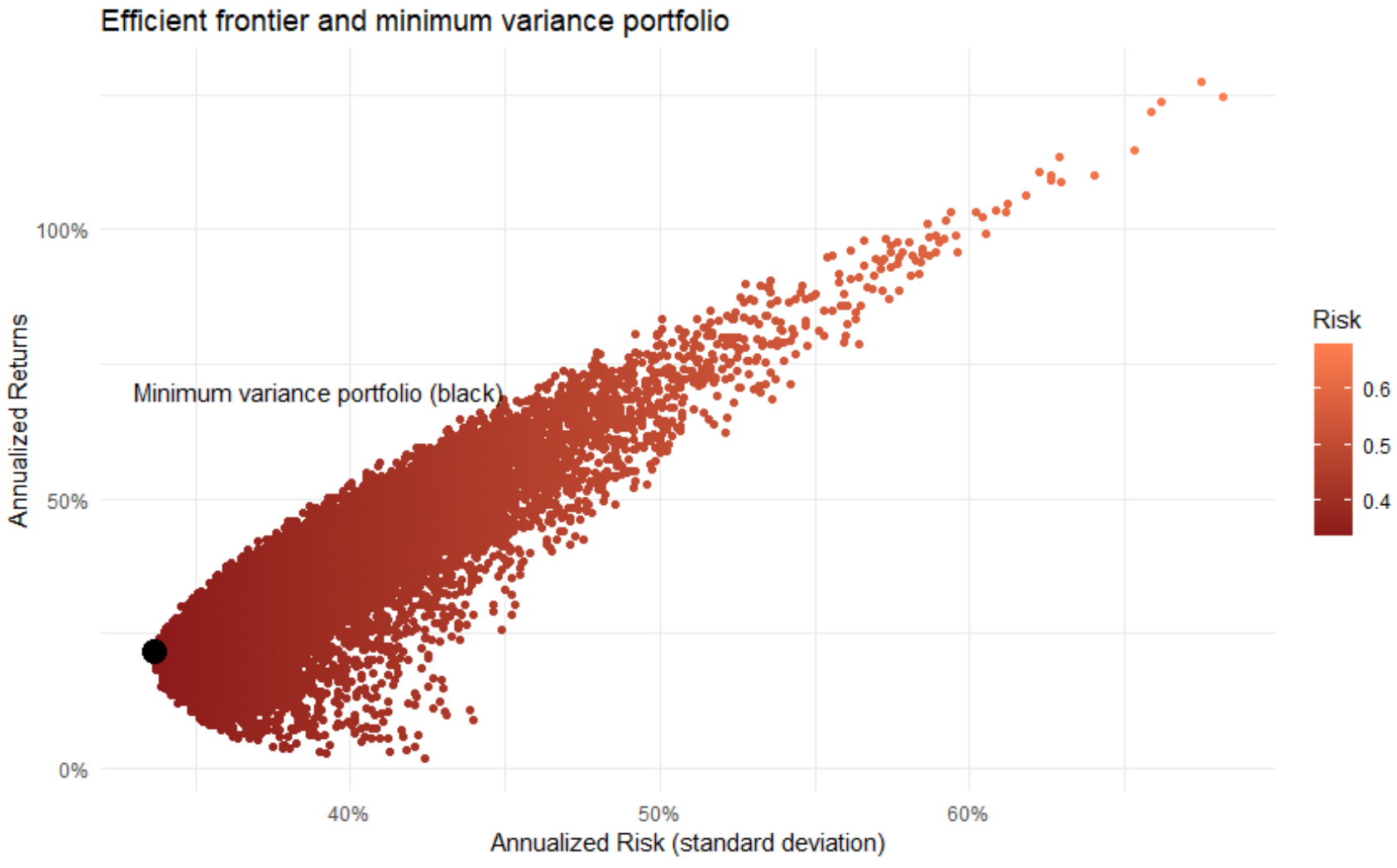
3. The expected return for a portfolio consisting in $n$ assets is given by $\hat{r}_p = \sum_{i=1}^{n} w_i r_i$, where $w_i$ is the weight of the asset $r_i$ in the portfolio.

4. The most efficient portfolio in the sense of Markowitz is the one which lies on the efficient frontier and minimizes the risk, as measured by the anualized standard deviation.

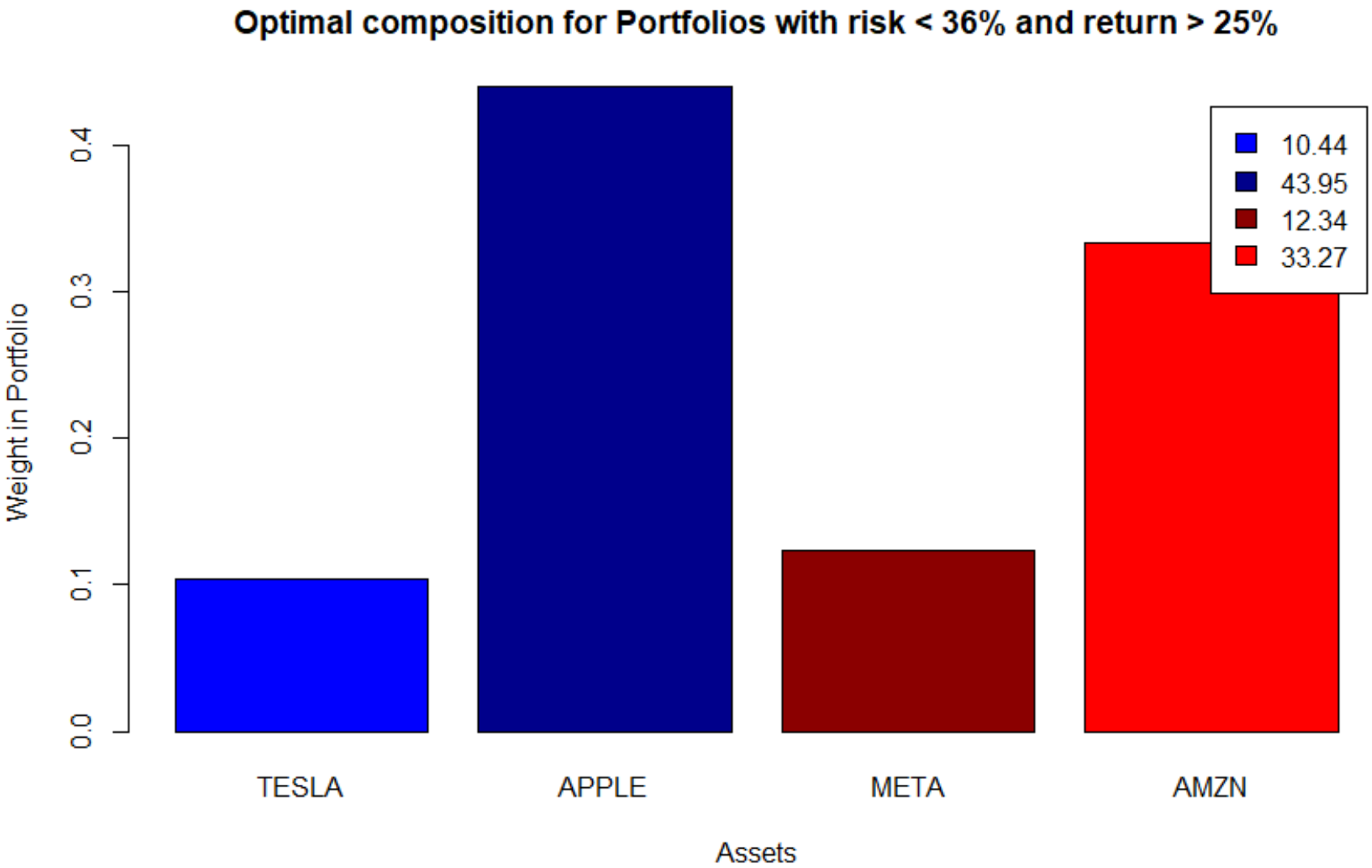
# Simulating portfolios for Markowitz

```r
# mean returns
mean.returns <- as.numeric(colMeans(returns))
# anualized risk (stadard deviation) of returns
cov.returns.anualized <- cov(returns) * 252
# simulations
nsim <- 10000
# storage objects
Aweights <- matrix(rep(0, nsim*4), nrow = nsim, ncol = 4)
Returns.Port <- numeric(nsim)
Risk.Port <- numeric(nsim)

set.seed(1986)
for(i in 1:nsim) {
  weights <- runif(4)
  sweights <- sum(weights)
  Weights[i, ] <- weights/sweights
  # Portfolio return
  returns.Port <- sum((weights/sweights) * mean.returns)
  Returns.Port[i] <- ((returns.Port + 1)^252) - 1
  # Rortfolio risk
  Risk.Port[i] <- sqrt(t((weights/sweights)) %*% (cov.returns.anualized
                          %*% (weights/sweights)))
}
Portfolios <- matrix(cbind(Weights,Returns.Port,Risk.Port),
                     byrow = FALSE, ncol = 6)
colnames(Portfolios) <- c("TESLA", "APPLE", "META", "AMZN", "Return", "Risk")
head(round(Portfolios,4))
#       TESLA  APPLE   META   AMZN Return   Risk
# [1,] 0.2950 0.1283 0.3739 0.2028 0.3650 0.4144
# [2,] 0.2771 0.3041 0.0774 0.3414 0.4472 0.3938
...
```

# Efficient frontier and minimum variance portfolio



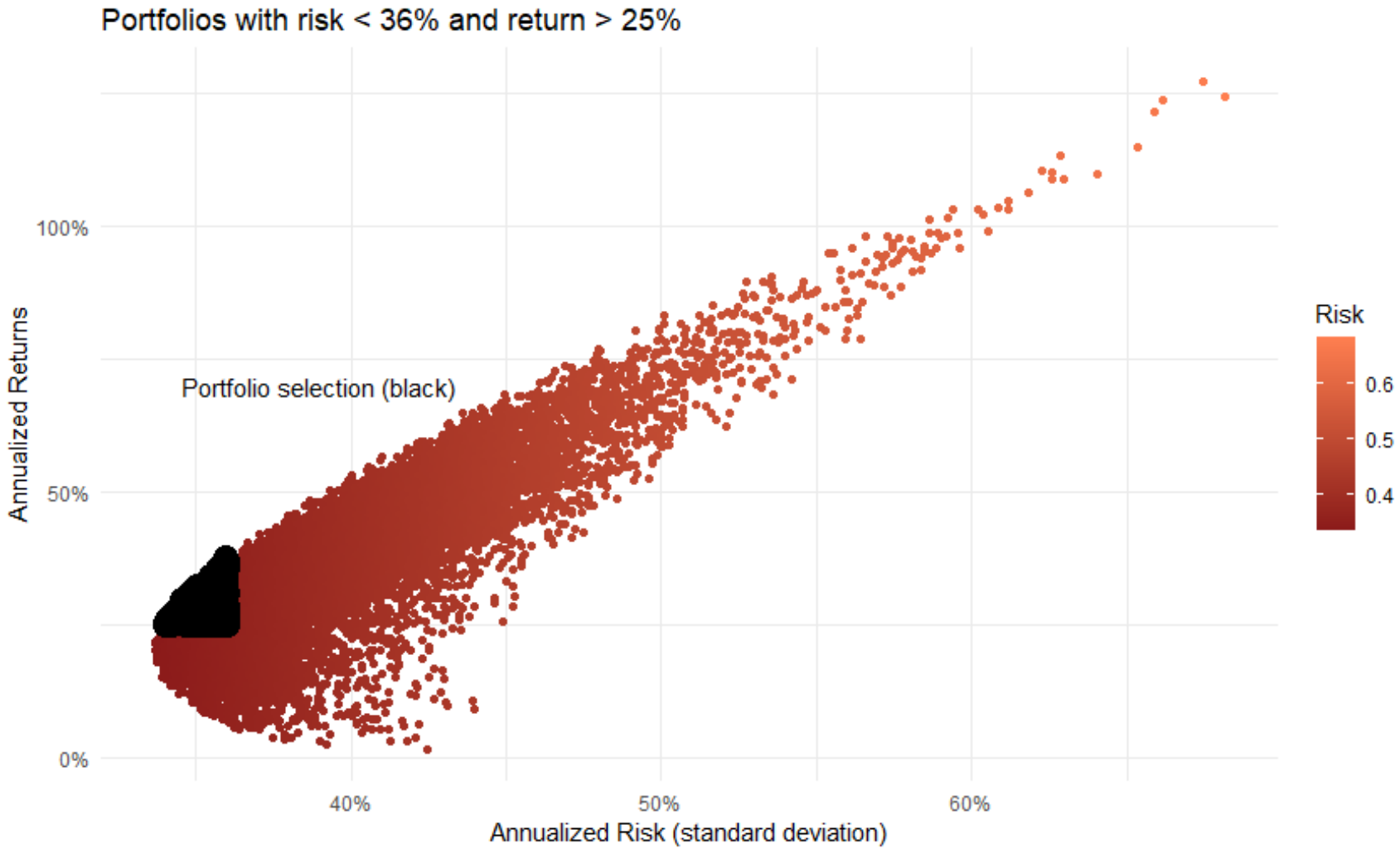
# Optimal portfolio composition



# R code for plot of portfolio selection

```r
# portfolios with risk < 36% and return > 25%
library(tidyverse)

Portfolios <- data.frame(Portfolios)

Portfolios %>%
  ggplot(aes(x = Risk, y = Return, color = Risk)) +
  geom_point() +
  scale_y_continuous(labels = scales::percent) +
  scale_x_continuous(labels = scales::percent) +
  scale_color_gradient(low = "firebrick4", high = "coral") +
  labs(x = 'Annualized Risk (standard deviation)',
       y = 'Annualized Returns',
       title = "Portfolios with risk < 30% and return -> 20%") +
  geom_point(aes(x = Risk,
                 y = Return), data = Portfolios[which(Portfolios$Risk < 0.36 &
                                                      Portfolios$Return > 0.25), ],
             color = 'black', size = 5) +
  annotate('text', x = 0.39, y = 0.7, label = "Portfolio selection (black)") +
  theme_minimal()

PortfolioSelection <- Portfolios[which(Portfolios$Risk < 0.36 &
                                       Portfolios$Return > 0.25), ]

barplot(colMeans(PortfolioSelection[, 1:4]),
        main = 'Optimal composition for Portfolios with risk < 36% and return -> 25%',
        xlab = 'Assets', ylab = 'Weight in Portfolio',
        col = c("blue", "darkblue", "darkred", "red"),
        legend = round(colMeans(PortfolioSelection[, 1:4]), 4)*100, beside = TRUE)
```

# Visualizing optimal portfolio selection



# References

Rizzo, M.L. (2019). Statistical Computing with R, Second Edition (2nd ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9780429192760

Elton, Gruber, Brown and Goetzmann, (2014). Modern Portfolio Theory and Investment Analysis, John Wiley, 9th ed.

Portfolio Optimization in R, DD. (2018), https://www.codingfinance.com/post/2018-05-31-portfolio-opt-in-r/

A Gentle Introduction to Finance using R: Efficient Frontier and CAPM – Part 1, D. Zimmermann. (2016), https://www.r-bloggers.com/2016/05/a-gentle-introduction-to-finance-using-r-efficient-frontier-and-capm-part-1/