Master-Thesis

Axel Roth

2022-08-23

Contents

Preface

Blah blah blah

Chapter 1

Abstract

Things about this thesis. why and what question should be answered. and what are the answers. (zusammenfassung)

Chapter 2

Software information and usage

wie ich das buch schreibe, R markodwn bookdown und so und welche versionen ich nutze

2.1 R-Version and Packages

2.2 reproducibility

github und code im bookdown

2.3 R-functions

zb plotly_save

Chapter 3

Open Data Sources

To increase the reproducibility, all data is free and can be loaded with the function `getSymbols` from the quantmod R-Package. There can be chosen between different data-sources like yahoo-finance (default), alpha-vantage, google and more.

3.1 R-functions

The following functions are created to increase the simplicity of data-gathering with the quantmod R-package that can be found in the directory named `R/` in the attached github repository.

3.1.1 `get_yf`

This function is the main wrapper for gathering data with `getSymbols` from yahoo-finance and transforms prices to returns with the `pri_to_ret` function explained in 4.5.1. The output is a list that contains prices and returns as a (xts){https://cran.r-project.org/web/packages/xts/xts.pdf} object. The arguments that can be passed to `get_yf` are:

- `tickers`: Vector of symbols like APPL, IBM, GOOG
- `from ="2020-01-01"`: R-Date
- `to = "2021-01-01"`: R-Date
- `prices_type = "close"`: Type of prices to gather (e.g "open", "high", "low", "close", "adjusted")
- `return_type = "adjusted"`: Type of returns to gather (e.g "open", "high", "low", "close", "adjusted")
- `print = F`: Should the function print the return of `getSymbols`

3.1.2 `buffer`

To make data reusable and decrease compiling time, this thesis saves all data gathered with `get_yf`. It takes an R expression, evaluates it and saves it in the `buffer_data/` directory with the given name. If this name already exists it will load the R-object from the RData-file without evaluating the expression. Forcing the evaluation and overwriting the existing RData-file can be done with `force=T`.

Chapter 4

Mathematical Fundations

This chapter provides an overview of the mathematical calculations and conventions used in this Thesis. It's important to note that most of the time mathematical formulas are written in matrix notation. In the majority of cases, this will result in a direct translation into R-code. All necessary assumptions needed for the modeled return structure are provided in this chapter to enable each reader to make sense of the stated formulas. It is crucial to note that reality is too complicated and can only be partially modeled. Simplistic, basic models are employed that don't hold up in real-world situations, but these models or variations on them are frequently used in finance and have proven to be helpful. The complexity of solving advanced and basic models do not differ for the PSO, because the dimension of the objective function is based on the number of selectable elements, see chapter 6.

4.1 Basic Operators

A compendium that compares commonly used mathematical symbols to R-code and its meanings can be found in the table below:

Latex/Displayed	R-Code	Meaning
×	%*%	Matrix or vector multiplication and cross-product
A^T	t(A)	transpose of Matrix or vector A
·	*	Scalar or elementwise vector multiplication

4.2 Return calculation

Any portfolio optimization strategy based on historical data must start with returns. These returns are calculated using adjusted closing prices, which show the percentage change over time. Adjusted closing prices are reflecting dividends and are cleaned of by stock splits and rights offerings. These Returns are essential for comparing assets and for analyzing dependencies.

4.2.1 daily returns

The default timeframe for all raw data in this thesis is one workday and only simple returns are used. The simple returns can be calculated as follows if we know the adjusted closing price P of one asset on workdays t_i and t_{i+1}:

$$R_{i+1} = \frac{P_{t_{i+1}}}{P_{t_i}} - 1$$

4.2.2 annualized returns

4.3 Markowitz Modern Portfolio Theory (MPT)

In 1952, Harry Markowitz published his first ground-breaking work, which had a significant influence on modern finance, primarily by outlining the effects of diversification and efficient portfolio. The definition of an efficient portfolio is one that has either the maximum expected return for a given risk target or the minimum risk for the given expected return target. A simple quote to define diversification could be: “A portfolio has the same return but less variance than the sum of its parts”. This is true if the assets are not perfectly correlated because

bad and good movers can make up for each other, reducing the likelihood of extreme events. You can find more specific information at (Maringer, 2005).

4.3.1 Assumptions of Markowitz Portfolio Theory

The following are the Markowitz assumptions that can be combined, according to (Maringer, 2005):

- Perfect market without taxes or transaction costs.
- Short sales are disallowed.
- Assets are infinitely divisible.
- Expected Returns, Variances and Covariances contain all information.
- Investors are risk-adverse, they will only accept greater risk if they are compensated with a higher expected return.

The assumption that the returns are normally distributed is not required, but it will be assumed in this case to make the problem simpler. (Maringer, 2005) has further details regarding the requirements for utilizing other distributions. It is obvious that these assumptions are unrealistic in real-life.

4.4 Portfolio Math

Proofs for the fundamental calculations required for portfolio optimization as shown in (Zivot, 2021) will be provided in this section. The returns are presented differently than in most sources, because its the most common data-format used in practice. Suppose there are N assets that are described by a return vector R of random variables and a portfolio weight vector w, respectively:

$$R = \begin{bmatrix} R_1 & R_2 & \cdots & R_N \end{bmatrix}, \quad w = \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_N \end{bmatrix}$$

Each return is additionally simplified in this thesis so that it is normally distributed with $R_i = \mathcal{N}(\mu_i, \sigma_i^2)$. As a result, linear combinations of normally distributed random variables are jointly normal distributed and have a mean, variance, and covariance that can be used to fully describe them.

4.4.1 expected returns

The following formula can be used to get the expected returns of a vector with normally distributed random variables $R \in \mathbb{R}^N$:

$$\begin{aligned} E[R] &= \begin{bmatrix} E[R_1] & E[R_2] & \cdots & E[R_N] \end{bmatrix} \\ &= \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_N \end{bmatrix} = \mu \end{aligned}$$

and μ_i can be estimated in R with the base-function `mean()` and historical data.

4.4.2 expected portfolio return

The following equation can be used to get the linear combination of expected returns mu and a weighting vector w (for example, portfolio weights):

$$\begin{aligned} \mu \times w &= \begin{bmatrix} E[\mu_1] & E[\mu_2] & \cdots & E[\mu_N] \end{bmatrix} \times \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_N \end{bmatrix} \\ &= E[\mu_1] \cdot w_1 + E[\mu_2] \cdot w_2 + \cdots + E[\mu_N] \cdot w_N = \mu_P \end{aligned}$$

4.4.3 portfolio returns

Let $R \in \mathbb{R}^{T \times N}$ denote a realized return Matrix of N assets and T days in the past. The portfolio return on each day can be calculated with the formula of the expected portfolio return. This is possible on all days T by:

$$R \times w = \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,N} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ R_{T,1} & R_{T,2} & \cdots & R_{T,N} \end{bmatrix} \times \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} R_1^P \\ R_2^P \\ \vdots \\ R_N^P \end{bmatrix} = R_P$$

4.4.4 Covariance

The general formula of the covariance matrix $\sum$ of a random vector R with N normally distributed elements and $\sigma_{i,j}$ as correlation of two unique assets is described as:

$$\begin{aligned} Cov(R) &= E[(R - \mu)^T \times (R - \mu)] \\ &= \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} & \cdots & \sigma_{1,N} \\ \sigma_{2,1} & \sigma_2^2 & \cdots & \sigma_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{N,1} & \sigma_{N,2} & \cdots & \sigma_N^2 \end{bmatrix} \\ &= \sum \end{aligned}$$

and can be estimated in R with the base-function `cov()` and historical data.

4.4.5 Portfolio Variance

Let R be a random vector with N normally distributed elements and w a weighting vector. Suppose the covariance matrix $\sum$ of R is known, then the variance of the linear combination of R can be calculated as:

$$\begin{aligned} Var(R \times w) &= E[(R \times w - \mu \times w)^2] \\ &= E[((R - \mu) \times w)^2] \end{aligned}$$

Since $(R - \mu) \times w$ is a scalar, it can be transformed from $((R - \mu) \times w)^2$ to $((R - \mu) \times w)^T \times ((R - \mu) \times w)$ and results in:

$$\begin{aligned} Var(R \times w) &= E[((R - \mu) \times w)^T \times ((R - \mu) \times w)] \\ &= E[(w^T \times (R - \mu)^T) \times ((R - \mu) \times w)] \\ &= w^T \times E[(R - \mu)^T \times (R - \mu)] \times w \\ &= w^T \times \sum \times w \end{aligned}$$

The same hold for a estimation of $\sum$.

4.5 Created R-Functions

4.5.1 pri_to_ret

Chapter 5

Activ vs Passiv Investing

The fundation of Asset Management

passiv vs activ studie https://www.scirp.org/journal/paperinformation.aspx?paperid=92983

gut gut file:///C:/Users/Axel/Desktop/Master-Thesis-All/Ziel%20was%20beantwortet%20werden%20soll/Quellen%20nur%20wichtige/Rasmussen2003_Book_QuantitativePortfolioOptimisat.pdf

Chapter 6

Challenges of Passiv Investing

This Chapter will analyse two common challenges of Passiv-Investing and create simple examples to test the PSO. The first one is the mean-variance portfolio (MVP) from the modern portfolio theory of Markowitz which is simply said an optimal allocation of assets regarding risk and return. The second challenge is the index-tracking-problem which tries to construct a portfolio which has a minimal tracking error to a given benchmark.

6.1 Mean-variance portfolio (MVP)

Markowitz has shown that diversifying the risk on multiple assets will reduce the overall risk of the portfolio. This result was the beginning of the widely used modern portfolio theorie which uses mathematical models to archive portfolios with minimal variance for a given return target. All these optimal portfolios for a given return target are called efficient and create the efficient frontier.

6.1.1 MVP

Let there be N assets and its returns on T different days which creates a return matrix $R \in \mathbb{R}^{T \times N}$. Each element $R_{t,i}$ contains the return of the i-th asset on day t. The covariance matrix of the returns is $\sum \in \mathbb{R}^{N \times N}$ and the expected returns are $\mu \in \mathbb{R}^N$. The MVP with risk aversion parameter $\lambda \in [0, 1]$ like shown in (Maringer, 2005) can be formalized as follows:

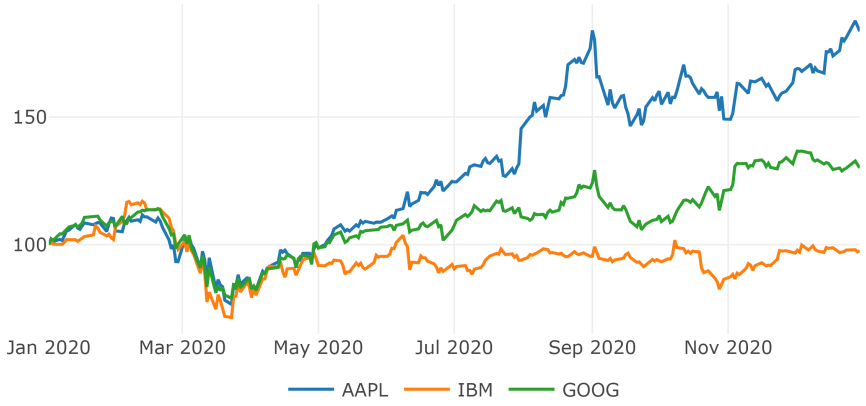
$$\underset{w}{minimize} \quad \lambda \, w^T \sum w - (1 - \lambda) \, \mu^T w \tag{6.1}$$

The risk aversion parameter λ defines the trade-off between risk and return. With $\lambda = 1$, the minimization problem only contains the the variance term and so on results in a minimum variance portfolio and $\lambda = 0$ transforms the problem to a minimization of the negative expected returns so on results in a maximum return portfolio. All possible $\lambda \in [0, 1]$ represent the efficient frontier.

6.1.2 MVP example

All possible MVPs combined create the efficient frontier, that is analyzed in this section without going into the details of its calculation. This example uses three assets (equitys: IBM, Google, Apple) and calculates the solution of 6.1 for each λ. First of all are the daily returns of these three assets are loaded from the year 2020.

The cumulated daily returns are:



The expected daily returns and the covariance matrix for the 3 assets can be calculated with:

```
p0("estimation of expected daily returns")
```

```
## [1] "estimation of expected daily returns"
```

```
mu <- sapply((1+returns), prod)^(1/nrow(returns))-1
mu
```

```
##          AAPL           IBM          GOOG
##  0.0024166167 -0.0000952478  0.0010448549
```

```
p0("estimation of positiv definite covariance matrix")
```

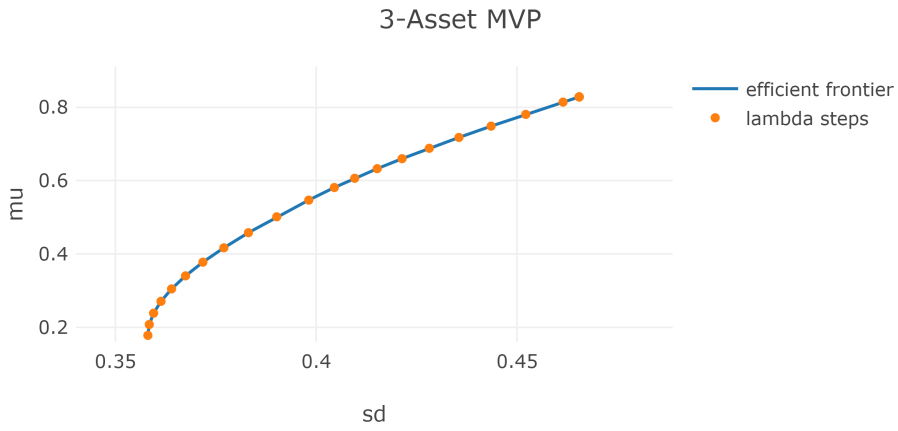
```
## [1] "estimation of positiv definite covariance matrix"
```

```
cov <- as.matrix(nearPD(cov(returns))$mat)
cov
```

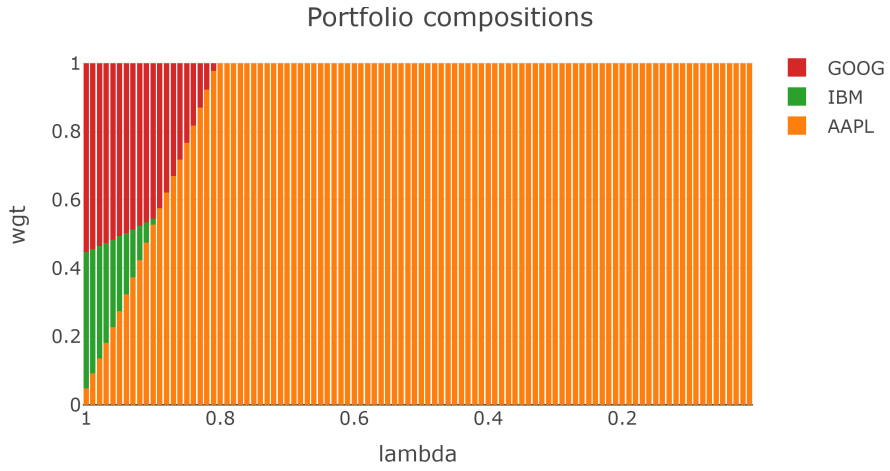
```
##              AAPL          IBM         GOOG
## AAPL 0.0008665678 0.0004378721 0.0005361398
## IBM  0.0004378721 0.0006646758 0.0004100227
## GOOG 0.0005361398 0.0004100227 0.0005849202
```

These are all the necessary data to solve the MVP (6.1) with $\lambda \in \{0.01, 0.02, ..., 0.99, 1\}$. All 100 portfolios are calculated by solving a quadratic minimization problem with long only constraint.

The resulting daily returns and standard deviation are transformed to annual returns and standard deviation and are plotted to create the efficient frontier:



The portfolio compositions for each λ are:



6.2 Index-tracking portfolio (ITP)

Indices are asset baskets that are used to track the performance of a specific asset group. The well-known Standard and Poor's 500 index (short: S&P 500), for example, tracks the top 500 stocks in the United States. All indices are not investible and only serve to visualize the performance of these asset groups without incurring transaction costs. Asset managers use such indices as benchmarks to compare the performance of their funds. Each fund has its own benchmark, which contains roughly the same assets that the manager could purchase. If the fund underperforms its benchmark, it may be an indication that the fund manager made a poor decision. That is why all fund managers strive to outperform their benchmarks through carefully chosen investments. The past has proven that this is rearly achived with activ managemnt after costs (Desmond Pace and Grima, 2016). This is the reason why passiv managed funds with the goal to track there benchmarks are becoming more frequent. This is why passively managed funds with the purpose of tracking their benchmarks are becoming more common. This can be accomplished through either full or sparse replication. In most circumstances, using a full replication that achieves the exact performance is not achievable, because not all assets in an index are investable. And, if so, it would be unwise because benchmarks with numerous indexes can contain over ten thousand separate assets, resulting in a massive amount of transaction costs. A sparse replication of the performance is the most prevalent approach. To do so, the portfolio manager must define his benchmark, which should overlap with his fund's investing universe. Following that, he will reduce this universe using investor principles such as liquidity and availability. Now he can begin to optimize a portfolio, taking into account the investor constraints, in order to match the benchmark performance. Typically, this is accomplished by lowering the variance between the ITP's daily returns and the benchmark:

$$minimize \;\; Var(r_p - r_{bm})$$

To obtain the portfolio weights w, its necessary to substitute r_p as shown below:

$$r_p = R * w$$

The Variance is then solved up until a quadratic problem dependent on the portfolio weights w is represented:

$$Var(r_p - r_{bm}) = Var(R*w - r_{bm}) = Var(R*w) + Var(r_{bm}) - 2 \cdot Cov(R*w, r_{bm})$$

Now the three terms can be solved, beginning with the easiest.

$$Var(r_{bm}) = \sigma_{bm}^2 = constant$$

The variance of the portfolio can be solved by looking at the Portfolio math Using Matrix Algebra section in (Zivot, 2021):

$$Var(R * w) = w^T * Cov(R) * w$$

And the last term can be solved in general as (https://bookdown.org/compfinezbook/introcompfinr/Multivariate-Probability-Distributions-Using-Matrix-Algebra.html 3.6.5):

$$Cov(A*a, b) = Cov(b, A*a) = E[(b-\mu_b)(A*a-\mu_A*a)] = E[(b-\mu_b)(A-\mu_A)*a] = E[(b-\mu_b)(A-\mu_A)]*a = Cov(A$$

```
A = matrix(c(1,4,2,4,6,3,8,4,4,10), ncol=2)
a = c(0.2, 0.8)
b = c(4,4,5,5,7)

cov(A %*% a, b)
```

```
##      [,1]
## [1,] 2.15
```

```
t(a) %*% cov(A, b)
```

```
##      [,1]
## [1,] 2.15
```

```
t(cov(A, b)) %*% a # das hier wird gebraucht
```

```
##      [,1]
## [1,] 2.15
```

This results in the final formula of the ITP:

$$\begin{aligned} Var(r_p - r_{bm}) &= Var(R \times w - r_{bm}) \\ &= Var(R \times w) - 2 \cdot Cov(R \times w, r_{bm}) + Var(r_{bm}) \\ &= w^T \times Cov(R) \times w - 2 \cdot Cov(r_{bm}, R)^T \times w + \sigma_{bm}^2 \end{aligned} \tag{6.2}$$

The minimization problem of the ITP in the general stricture which all optimizers need is:

$$min(\frac{1}{2} \cdot b^T \times D \times b - d^T \times b)$$

Minimization problems can ignore constant terms and global stretching coefficients and still find the same minimum. This results in the general substitution of the ITP as follows:

$$D = Cov(R)$$

and

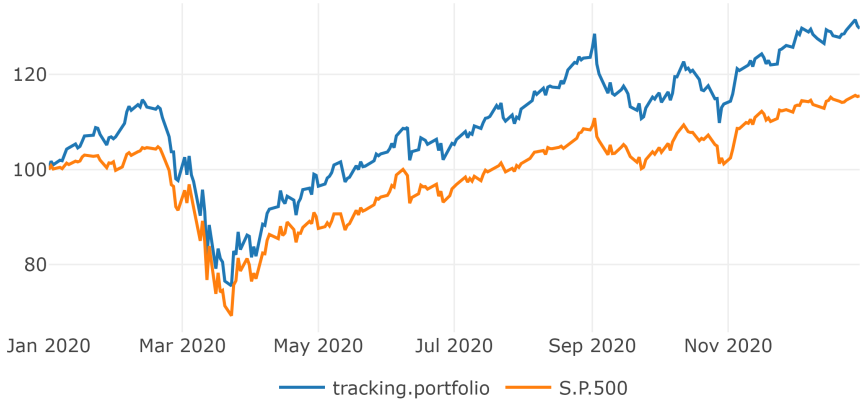
$$d = Cov(r_{bm}, R)$$

Its possible to add some basic constraints like in the MVP, to sum up the weights to 1 and being long only.

6.2.1 Example ITP

This section will show the results of tracking the S&P 500 with a tracking portfolio that can only invest in IBM, Apple and Google without going into details:

```
##      AAPL       IBM      GOOG
## 0.2681058 0.4040352 0.3278591
```


120
100
80
Jan 2020
Mar 2020
May 2020
Jul 2020
Sep 2020
Nov 2020
tracking.portfolio
S.P.500

Chapter 7

Analytic Solver for Quadratic Programming Problems

The benefits and drawbacks of analytic solvers for quadratic programming problems will be discussed in this chapter after chapter 6's discussion of some common problems and their solutions. It would go beyond the scope of this thesis to explain the mathematical underlying principles of how a solver addresses quadratic problems, only the applications and analysis are discussed here. The foremost reason of addressing quadratic programming solvers is to use it as a benchmark for the PSO.

7.1 Quadratic Programming (QP)

A quadratic program is a minimization problem of some function that returns a scalar and consists of an quadratic term and an linear term dependent on the variable of interest. Additionally can the problem be simply constrained by several linear inequalities that restrict the solution. The general formulation used, is to find x that minimizes the following problem:

$$min\ \frac{1}{2} \cdot x^T \times D \times x - d^T \times x$$

and holds under the linear constraints:

$$A^T \times x >= b_0$$

Some other sources notate the problem with different signs or coefficients that are all exchangeable with the above stated problem. Additionally has the

problem above the same notation that is used in the R-Package `quadprog` which will reduce substitution efforts. All modern programming languages do have many solvers for quadratic problem. They differ mostly in computational time on specific problems and the requirements. Some commercial QP-solvers do additionally accept more complex constraints, like absolute (e.g. $|A^T \times x| >= a_0$) or mixed-integer (e.g. $x \in \mathbb{N}$). Specially the mixed-integer constraint problems will result in a enormously increase of memory.

7.2 QP Solver from quadprog

The most common free QP-solver used in R is quadprog which consists of a single function named `solve.QP`. Its implementation routine is the dual method of Goldfarb and Idnani that was published in (Goldfarb and Idnani, 1982) and (Goldfarb and Idnani, 1983). It uses the above stated QP with the requirement that D needs to be a symmetric positive definite matrix. It means that $x^T Dx > 0 \, \forall \, x \in R^N$ and $D \in R^{N \times N}$ which is equivalent to, all eigenvalues are bigger than null. In most cases this is not achieved by using the estimation of the covariance matrix $\sum$, but its possible to find the nearest positive definite matrix of $\sum$ with the function `nearPD` from the Matrix R-Package. The error that occurs is printed and often do not exceed a percentage change of elements above 10^{-15} %, which is negligible for the context of this thesis. The `solve.QP` functions for a N dimensional vector of interest, has the following arguments that can be found in the above stated formulation of a QP:

- Dmat: Symmetric positive definite matrix $D \in R^{N \times N}$ of the quadratic term.
- dvec: Vector $d \in R^N$ of the linear term
- Amat: Constraint matrix A
- bvec: Constraint vector b_0
- meq = 1: means that the first row of A is treated as equality constraint (used to achieve fully invested portfolios)

The return of `solve.QP` is a list and contains among other things the following attributes of interest: + solution: Vector containing the solution x of the quadratic programming problem. (e.g. portfolio weights) + value: Scalar, the value of the quadratic function at the solution

7.3 Example: Solving MVP with `solve.QP`

This section provides insights into the effects of diversification and the use of `solve.QP`, by creating ten diffrent efficient frontiers from a pool of ten assets. Each efficient frontier i consists of $N_i = i$ assets and is created by adding the

next smallest variance first. After loading returns for ten of the biggest equitys in the US market, the variance is calculated to arrange all columns ascending, like shown in the code bellow:

```
returns_raw <- buffer(
  get_yf(
    tickers = c("IBM", "GOOG", "AAPL", "MSFT", "AMZN", "NVDA", "JPM", "META", "V", "WMT"),
    from = "2016-01-01",
    to = "2021-12-31"
  )$returns,
  "AS_10_assets"
)

# re-arrange: low var first
vars <- sapply(returns_raw, var)
returns_raw <- returns_raw[, order(vars, decreasing = F)]
```

The next step is to create a function `mvp` that has the arguments `return` and `lambda`. It calculates the expected returns `mu` and the estimated positive definite covariance `cov`. Afterwords it solves a MVP with the constraints $\sum w = 1$ and $w >= 0$ and returns key features *mu*, *var* and *composition* of the portfolio.

```
mvp <- function(returns, lambda){
  tc <- tryCatch({
    mu <- sapply((1+returns), prod)^(1/nrow(returns))-1

    cov <- as.matrix(nearPD(cov(returns))$mat)

    mat <- list(
      Dmat = lambda * cov,
      dvec = (1-lambda) * mu,
      Amat = t(rbind(
        rep(1, ncol(returns)), # sum up to 1
        diag(1, nrow=ncol(returns), ncol=ncol(returns)) # long only
      )),
      bvec = c(
        1, # sum up to 1
        rep(0, ncol(returns)) # long only
      ),
      meq = 1
    )

    qp <- solve.QP(Dmat = mat$Dmat, dvec = mat$dvec, Amat = mat$Amat, bvec = mat$bvec, meq =

    res <- list(
```

```
      "mu" = mu %*% qp$solution,
      "var" = t(qp$solution) %*% cov %*% qp$solution,
      "composition" = setNames(qp$solution, colnames(returns))
    )
    TRUE
  }, error = function(e){FALSE})


  if(tc){
    return(res)
  }else{
    return(list(
      "mu" = NA,
      "var" = NA,
      "composition" = NA
    ))
  }
}
```

Each $lambda \in 0.01, 0.02, \cdots, 1$ and each combination of ascending number of assets produces one portfolio that can be generated with two for loops.

```
df <- data.frame("index"=1, "var"=as.numeric(var(returns_raw[, 1])), "return" = prod(1+retu
for(i in 2:ncol(returns_raw)){
  returns <- returns_raw[, 1:i]
  for(lambda in seq(0.01, 1, 0.01)){
    res <- mvp(returns, lambda)

    df <- rbind(df, data.frame("index"=i, "var"=res$var, "return" = res$mu))
  }
}
```

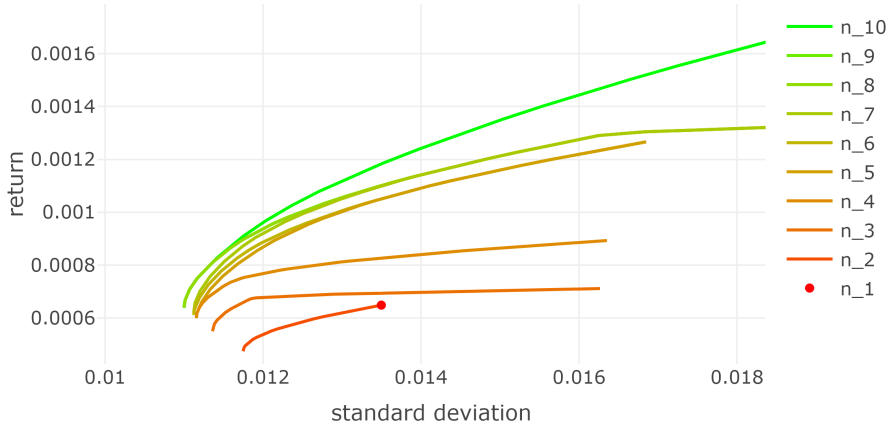
The result gets filtered and names are added that represent the number of assets. Now can the plot be genarated:

```
df <- df %>%
  filter(!is.na(return)) %>%
  distinct() %>%
  mutate(name = paste0("n_", index)) %>%
  arrange(name) %>%
  mutate(name = factor(name, levels=paste0("n_", ncol(returns_raw):1)))


max_show_sd <- df %>%
  group_by(index) %>%
```

```
  summarise(max_x = max(var)) %>%
  pull(max_x) %>%
  mean() %>%
  sqrt()

plot_ly(
    data = df[df$index!=1,],
    x=~sqrt(var),
    y=~return,
    name=~name,
    mode="lines",
    type = 'scatter',
    color = ~name,
    colors = c("green", "red")
  ) %>%
  add_trace(
    data=df[df$index==1,],
    x=~sqrt(var),
    y=~return,
    showlegend=T,
    marker=list(color="red"),
    mode="markers",
    name="n_1") %>%
  layout(
    xaxis=list(range=c(sqrt(min(df$var))*0.9, max_show_sd), title="standard deviation"),
    yaxis=list(range=c(min(df$return)*0.9, (max(df$return)+mean(df$return))*0.5), title="ret
  html_save()
```



It can be seen, that each asset added results in a minimum variance portfolio with smaller standard deviation. Nonetheless that it starter with the asset that has the smallest standard deviation of 0.0001822. This is the effect of

diversification mentioned by Markowitz.

7.4 Example: Solving ITP with `solve.QP`

```
from <- "2016-01-01"
to <- "2021-12-31"

spx_composition <- buffer(
  get_spx_composition(),
  "AS_spx_composition"
)


pool_returns_raw <- buffer(
  get_yf(
    tickers = spx_composition %>% filter(Date<=to) %>% filter(Date==max(Date)) %>% pull(Tick
    from = from,
    to = to
  )$returns,
  "AS_sp500_assets"
)
pool_returns_raw <- pool_returns_raw[, colSums(is.na(pool_returns_raw))==0]


bm_returns <- buffer(
  get_yf(tickers = "%5EGSPC", from = from, to = to)$returns,
  "AS_sp500"
) %>% setNames(., "S&P 500")


itp <- function(pool_returns, bm_returns){
  mat <- list(
    Dmat = cov(pool_returns),
    dvec = cov(pool_returns, bm_returns),
    Amat = t(rbind(
      rep(1, ncol(pool_returns)), # sum up to 1
      diag(1, nrow=ncol(pool_returns), ncol=ncol(pool_returns)) # long only
    )),
    bvec = c(
      1, # sum up to 1
      rep(0, ncol(pool_returns)) # long only
    ),
    meq = 1
```

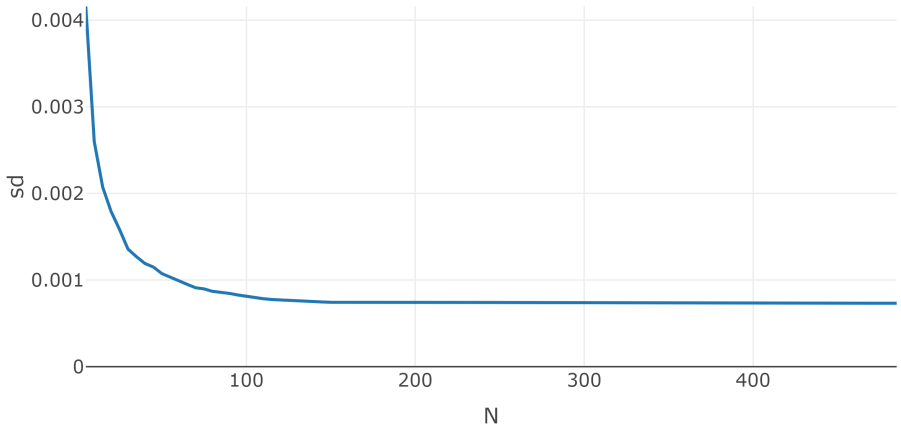
```
  )

  qp <- solve.QP(Dmat = mat$Dmat, dvec = mat$dvec, Amat = mat$Amat, bvec = mat$bvec, meq = m

  res <- list(
    "var" = as.numeric(var(pool_returns %*% qp$solution - bm_returns)),
    "solution" = setNames(qp$solution, colnames(pool_returns))
  )
}


res <- NULL
for(i in rev(seq(5, ncol(pool_returns_raw), 5))){
  if(i==ncol(pool_returns_raw)){
    temp <- itp(pool_returns_raw, bm_returns)
  }else{
    temp <- itp(pool_returns_raw[, names(sort(temp$solution, decreasing = T)[1:i])], bm_retu
  }
  res <- rbind(res, data.frame("N"=i, "var"=temp$var, "sd"=sqrt(temp$var), row.names = NULL)
}


plot_ly(data=res, x=~N, y=~sd, mode="lines", type = 'scatter') %>%
  layout(yaxis=list(range=c(0, mean(max(res$sd),mean(res$sd)) ))) %>%
  html_save()
```



Chapter 8

Simple_Particle_Swarm_Optimization

first pso examples and explainations

Bibliography

Desmond Pace, J. H. and Grima, S. (2016). Active versus passive investing: An empirical study on the us and european mutual funds and etfs.

Goldfarb, D. and Idnani, A. (1982). Dual and primal-dual methods for solving strictly convex quadratic programs.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs.

Maringer, D. (2005). *Portfolio Management with Heuristic Optimization.*

Zivot, E. (2021). *Introduction to Computational Finance and Financial Econometrics with R.*