Financial Technologies - Take Home Exam

Alexander Steeb - 17-614-611

10.11.2017

Contents

Part I: Online lenders

Question 1

Differences between an online loan marketplace and a traditional bank

Traditional Bank	Online loan marketplace
Capital intensive	Not as capital intensive
Strong regulatory restrictions and regulations	Not yet as regulated as traditional banking
Relatively lower interest rates	Relatively higher interest rates
Possibly a higher maximum credit due to higher balanc sheet capabilities and more sophisticated screening	Possibly a lower maximum credit
Could be slower then an online platform	Could be faster then a traditional bank
The bank takes all the credit risk on it's own balance sheet	The platform itself takes no or only limited credit risk on it's own balance sheet

Three risks for the users (investors and borrowers) of online loan marketplaces

Overview

Borrrowers	Investors
Paying to high interest rates	Fraud
Hidden fees and terms	Poor Selection
Using p2p loans to continue poor debt habits	Moral Hazard

Explenations

Borrrowers

Hidden fees and terms - Due to the nature of these platforms and the evolving market there are still many offers that include hidden fees or other hidden terms of use. Users have to be really cautious.

Using p2p loans to continue poor debt habits - Unlike banks p2p lending platforms will grant loans even to people with already high debt levels and a high risk of default. The ease of getting a loan can be a dangerous and risky incentive for people to continue lending money when they should start repaying their debt.

Paying to high-interest rates - Because of the nature of such p2p platforms interest rates are typically much higher than with traditional banks. Individuals without proper financial literacy can be tempted to take on credit with high-interest rates just because of the simplicity of the process. This can be highly risky for the borrower.

Investors

Fraud - Investors are subject to the risk of fraud, theft and scam. Individuals can take on a loan with a fake identity and then hide with the money. This risk could be higher for online platforms since they don't see the borrowers in person.

Poor Selection - As already mentioned p2p platforms probably attract people that would not get a loan from a bank because of their high-risk profile. This observation indicates an adverse selection problem similar to the famous lemon market.

Moral hazard - Investors could face a moral hazard problem because of limited collaterals in comparison with a traditional bank. Borrowers that don't pay back their loan cannot be as quickly held accountable as regular banking clients. Those the incentive for them to stop repaying their loans.

Question 2

A data driven model to predict default

Data preparation:

```r
rm(list=ls())
dd = read.csv('credit_scoring_EXAM.csv', header=TRUE)
```

Q 2.i.

What criterion does PCA minimize? What criterion does PCA maximize?

Principal component analysis converts a dataset with possibly correlated variables into a dataset with completely uncorrelated variables. Those new variables are called the principal components of the given dataset. PCA uses an orthogonal transformation to calculate those principal components. The number of principal components is less or equal to the number of original variables. The first principal component is always the one that accounts for the highest possible variance in the dataset. Thus it explains as much of the variability in the dataset as possible. The next principal component tries to achieve the same goal under the constraint that it is orthogonal to the preceding components.

Thus, PCA…:
… **maximize** the **variance of projection** along each component.
… **minimize** the **reconstruction error** (ie. the squared distance between the original data and its "estimate")

Q 2.ii.

How does the matrix W relate to Σ?

W is the eigenvector of the covariance matrix of the standardized feature matrix.

Q 2.iii.

Find Z and the covariance matrix S of Z

Generate the correlation-matrix first to get a sense of the data

```r
features <- c('income', 'assets', 'debt', 'loan_size', 'age', 'height')  # Names of the 6 features

cormatrix <- round(cor(dd[,features]),2)  # Correlationmatrix
print(round(cormatrix,3))
```

```
##           income assets debt loan_size  age height
## income      1.00   0.71 0.66      0.92 0.05  -0.01
## assets      0.71   1.00 0.92      0.65 0.49  -0.02
## debt        0.66   0.92 1.00      0.60 0.45  -0.02
## loan_size   0.92   0.65 0.60      1.00 0.05  -0.01
## age         0.05   0.49 0.45      0.05 1.00  -0.01
## height     -0.01  -0.02 -0.02     -0.01 -0.01   1.00
```

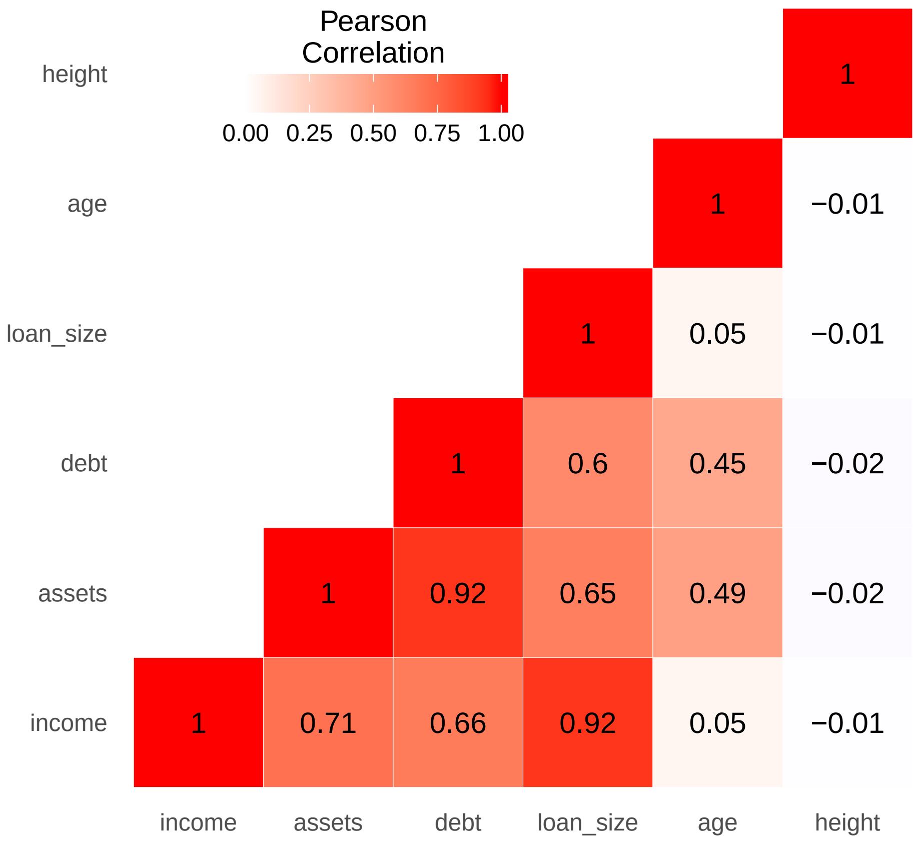
We can already see that some of the variables in the dataset are highly correlated. On the one hand this correlation could be a problem in the further analysis due to possible multicollinearity. On the other hand, it could also be a chance to apply PCA in a highly efficient manner. Since many of these variables are correlated, we can probably reduce the dimension of the dataset with PCA thereby reducing the complexity of our preditive model.

To better inspect the correlation within our dataset we will visualize the correaltion matrix in a next step.

```r
library(reshape2)
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```r
get_lower_tri<-function(cormat){    # Function to get onyl the lower half of the correlation matrix
    cormat[upper.tri(cormat)] <- NA
    return(cormat)
}

lower_tri <- get_lower_tri(cormatrix) # Get lower half

melted_cormat <- melt(lower_tri, na.rm = TRUE)  # Reformat the data

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) + # Generate plot
  geom_tile(color = "white") +
  geom_text(aes(Var1, Var2, label = value), color = "black", size = 4) +
    scale_fill_gradient2(low = "blue", high = "red", limit = c(0,1), space = "Lab",
    name="Pearson\nCorrelation") +
  theme_minimal() +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.5, 0.8),
    legend.direction = "horizontal") +
  guides(fill = guide_colorbar(barwidth = 7, barheight = 1,
    title.position = "top", title.hjust = 0.5)) +
  coord_fixed()
```



Conclusion: As already mentioned many of the variables are highly correlated. Two variables - height and age - are nearly uncorrelated to the remaining variables. The highest correlation coefficients are between assets and debt (0,92) and between income and loan_size(0,92). All in all, it seems like PCA could reduce the dimension of this dataset significantly without losing much information.

Thus we now calculate the principal components.

```
X = dd[,features]    # select relevant data and standardize it
S = cov(X)

W = eigen(S)$vectors       # compute matrix of eigenvectors
Z = as.matrix(X) %*% W     # compute projections

round(cov(Z),3)      # display the pc's
```

```
##        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
## [1,] 2.178 0.000 0.000 0.000 0.000 0.000
## [2,] 0.000 1.037 0.000 0.000 0.000 0.000
## [3,] 0.000 0.000 1.006 0.000 0.000 0.000
## [4,] 0.000 0.000 0.000 0.187 0.000 0.000
## [5,] 0.000 0.000 0.000 0.000 0.026 0.000
## [6,] 0.000 0.000 0.000 0.000 0.000 0.026
```

We can also calculate these principal components with an inbuild r function. We assume that the variables are aready standardized.

```
Z_R = prcomp(X, scale=FALSE)$x # Check

round(cov(Z_R),3)      # display the pc's
```

```
##       PC1   PC2   PC3   PC4   PC5   PC6
## PC1 2.178 0.000 0.000 0.000 0.000 0.000
## PC2 0.000 1.037 0.000 0.000 0.000 0.000
## PC3 0.000 0.000 1.006 0.000 0.000 0.000
## PC4 0.000 0.000 0.000 0.187 0.000 0.000
## PC5 0.000 0.000 0.000 0.000 0.026 0.000
## PC6 0.000 0.000 0.000 0.000 0.000 0.026
```

```
dd['PCA1'] <- Z_R[,1]    # Add the 3 relevant components to the main data frame
dd['PCA2'] <- Z_R[,2]
dd['PCA3'] <- Z_R[,3]
```

Now that we have calculated the principal components we want to check how much of the total variance each component explains. We can evaluate this question by dividing the variation of each component through the combined variation of all components.

```
PCA_var <- c()      # creat vector for the results

for (i in 1:6) {  # calculate the percentages
  PCA_var[i] <- cov(Z_R)[i,i]/sum(cov(Z_R))
}

PCA_var <- t(100 * PCA_var)

print(round(PCA_var,2))
```

```
##       [,1]  [,2]  [,3] [,4] [,5] [,6]
## [1,] 48.83 23.25 22.55  4.2 0.59 0.58
```

```
cat('\n', "The first 3 principal components explain ", round(sum(PCA_var[1:3]),1), "% of the total vari
```

```
##
##  The first 3 principal components explain  94.6 % of the total variation
```

Interpretation:

The 6 avalibel variables can be broken down to only 3 principal componants, while still retaining more than 94% of the total variability in the dataset. We will therefore proceed with only these 3 principal components.

Q 2.iv

Fit a logistic regression model and calculate the in- and out-of-sample errror-rates

First we want to split the dataset into a test and a training set

```
set.seed(2) # Random seed for reproducibility

n <- nrow(dd) # Number of rows

train <- sample(1:n, size=round(0.75 * n))
n_train <- length(train)
n_test <- n - n_train
dd_train <- dd[train, ]
```

Next we want to creat a stauts variable that states good or bad depending on default or not

```
dd['status'] = 'good'    # gen new variable
dd[dd['default'] == '1', 'status'] = 'bad'   # assign bad if failure
```

```
dd$status <- as.factor(dd$status)   # make it a factor variable
```

Now we can performe the logistic regression. We will first perform the regression with the initial 6 variables and then with the 3 most promising principal components.

Logistic regression with the initial 6 variables

```
all <- glm(formula = default ~ income + assets + debt + loan_size + age + height, family = binomial,
    data = dd[train, ])
summary(all)
```

```
##
## Call:
## glm(formula = default ~ income + assets + debt + loan_size +
##     age + height, family = binomial, data = dd[train, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6550  -0.8392  -0.6815   1.2554   2.3201
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.99819    0.02712 -36.801  < 2e-16 ***
## income      -0.21646    0.07526  -2.876  0.00403 **
## assets      -0.57084    0.07953  -7.178 7.08e-13 ***
## debt         0.09045    0.15549   0.582  0.56076
## loan_size    0.49479    0.15375   3.218  0.00129 **
## age          0.02292    0.03460   0.662  0.50779
## height      -0.01841    0.02586  -0.712  0.47665
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8935.5  on 7499  degrees of freedom
## Residual deviance: 8520.3  on 7493  degrees of freedom
## AIC: 8534.3
##
## Number of Fisher Scoring iterations: 4
```

```
anova(all)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: default
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev
## NULL                       7499     8935.5
## income     1  217.471      7498     8718.1
## assets     1  186.105      7497     8532.0
## debt       1    0.392      7496     8531.6
## loan_size  1   10.304      7495     8521.3
## age        1    0.422      7494     8520.8
## height     1    0.507      7493     8520.3
```

Evaluation of in and out of sample performance

```
# In sample performance
# =====================

all.fitted = rep(NA, length(all$fitted.values))   # create vector for predictions

all.fitted[all$fitted.values < 0.5] = 0        # depending on the predicted probability assign
all.fitted[all$fitted.values >= 0.5] = 1       # a 1 (default) or a 0 (no default)



# Calculate the error rate (How many observations are wrongly predicted)
all.fitted.error = 100*sum(diag(table(dd$status[train], all.fitted))) / n_train


table(dd$status[train], all.fitted)         # confusion matrix (how good is the   prediction?)
```

```
##       all.fitted
##           0    1
##   bad  1982  140
##   good 5255  123
```

```
cat('\n', "N = ", sum(table(dd$status[train], all.fitted)), '\n')
```

```
##
##  N =  7500
```

```
cat('\n',"Error rate = ", round(all.fitted.error,2), "% \n")
```

```
##
##  Error rate =  28.07 %
```

```
# Out of sample performance
# =========================

all.predictions = predict(all, newdata=dd[-train,]) # calculating the predictions
all.predictions = 1 / (1 + exp(-all.predictions))

all.predictions[all.predictions < 0.5] = 0     # assign value
all.predictions[all.predictions >= 0.5] = 1


# Error rate
all.predictions.error = 100*sum(diag(table(dd$status[-train], all.predictions))) / n_test

table(dd$status[-train], all.predictions)    # confusion matrix
```

```
##       all.predictions
##           0    1
##   bad   661   48
##   good 1761   30
```

```
cat('\n', "N = ", sum(table(dd$status[-train], all.predictions)),'\n')
```

```
##
##  N =  2500
```

```
cat('\n', "Error rate = ", round(all.predictions.error,2), "% ")
```

```
##
##  Error rate =  27.64 %
```

Logistic regression with the 3 most important principal components

```
PCA <- glm(formula = default ~ PCA1 + PCA2 + PCA3, family = binomial, data = dd[train, ])
summary(PCA)
```

```
##
## Call:
## glm(formula = default ~ PCA1 + PCA2 + PCA3, family = binomial,
##     data = dd[train, ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5696  -0.8388  -0.6903   1.2715   2.2258
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.984462   0.026876 -36.629   <2e-16 ***
## PCA1        -0.344356   0.018676 -18.439   <2e-16 ***
## PCA2        -0.022154   0.025754  -0.860    0.390
## PCA3         0.007149   0.026324   0.272    0.786
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8935.5  on 7499  degrees of freedom
## Residual deviance: 8568.7  on 7496  degrees of freedom
## AIC: 8576.7
##
## Number of Fisher Scoring iterations: 4
```

```
anova(PCA)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: default
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev
## NULL                  7499     8935.5
## PCA1  1   366.05      7498     8569.5
## PCA2  1     0.74      7497     8568.7
## PCA3  1     0.07      7496     8568.7
```

Evaluation of in and out of sample performance

```
# In sample performance
# =====================

PCA.fitted = rep(NA, length(PCA$fitted.values))

PCA.fitted[PCA$fitted.values < 0.5] = 0
PCA.fitted[PCA$fitted.values >= 0.5] = 1

PCA.fitted.error = 100*sum(diag(table(dd$status[train], PCA.fitted))) / n_train

table(dd$status[train], PCA.fitted)
```

```
##       PCA.fitted
##           0    1
##   bad  2018  104
##   good 5273  105
```

```
cat('\n', "N = ", sum(table(dd$status[train], PCA.fitted)), '\n')
```

```
##
##  N =  7500
```

```
cat('\n',"Error rate = ", round(PCA.fitted.error,2), "% \n")
```

```
##
##  Error rate =  28.31 %
```

```
# Out of sample performance
# =========================

PCA.predictions = predict(PCA, newdata=dd[-train,])
PCA.predictions = 1 / (1 + exp(-PCA.predictions))

PCA.predictions[PCA.predictions < 0.5] = 0
PCA.predictions[PCA.predictions >= 0.5] = 1

# confusion matrix and error rate
PCA.predictions.error = 100*sum(diag(table(dd$status[-train], PCA.predictions))) / n_test

table(dd$status[-train], PCA.predictions)
```

```
##       PCA.predictions
##           0    1
##   bad   667   42
##   good 1758   33
```

```
cat('\n', "N = ", sum(table(dd$status[train], PCA.fitted)),'\n')
```

```
##
##  N =  7500
```

```
cat('\n', "Error rate = ", round(PCA.predictions.error,2), "% ")
```

```
##
##  Error rate =  28 %
```

Q 2.v.

Lasso logistic regression

In addition to the standart logistic regression we can also perform a lasso logistic regression. Lasso stands for *least absolute shrinkage and selection operator* and describes a form a regularization and variable selection. In r we can perform a lasso logistic regression easily with the package glmnet.

We use the glmnet package to find the lambda that minimizes the classification error in a lasso logistic regression and the corresponding error.

```
require(glmnet)
```

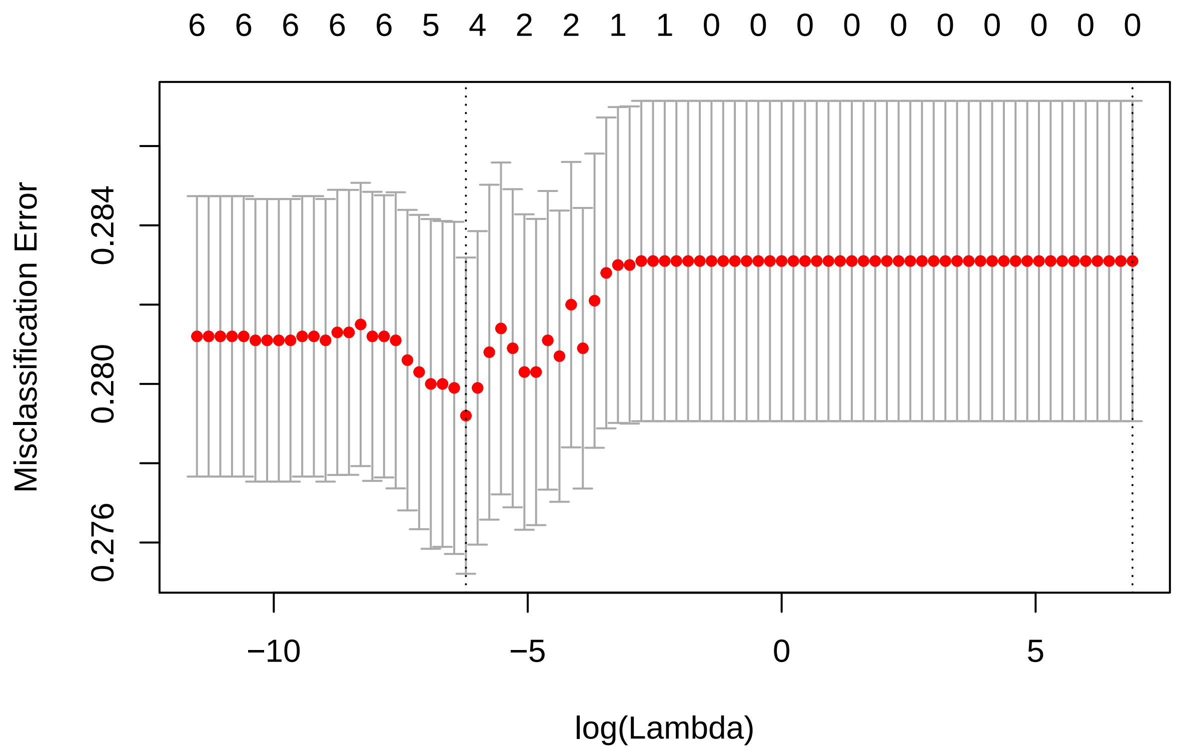
```
## Loading required package: glmnet

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-13
```

```
lambdas = 10^(seq(-5,3,by=0.1))  # list of penalty parameters lambda
x = dd[,features]                # Include all original variables

# Performing the lasso regression
# nfolds = 10 and the prefix "cv." perform a 10 fold cross validation
# alpha = 1 indicates that lasso regression or also known as L1 regularization is used

cvfit = cv.glmnet(as.matrix(x), dd$default, family = "binomial", nfolds = 10 , type.measure = "class", 

cvm <- cvfit$cvm  # get list of classificaton erroe
cat("The minimal classification error is: ", min(cvm))
```

```
## The minimal classification error is:  0.2792
```

We can also plot the differnet lambdas against the classification error to visualize the effect of different lambdas.

```
plot(cvfit)
```



Question 3

Under what circumstances should one choose a higher or lower threshold?

To answer this question, we first need to look at the two types of classification errors.

Type 1 error: The incorrect rejection of a true null hypothesis (also known as a **"false positive"** finding)

Type 2 error: The incorrect retaining a false null hypothesis (also known as a **"false negative"** finding)

If we increase the threshold, we decrease the rate of type I errors. If we decrease the threshold, we increase the rate of type II errors. This is particularly important in situations where you want to avoid either one of these error types more than the other. For example, in cancer diagnostics, it is vitally important that you decrease the type II error rate. You don't want to tell someone who has cancer that he does not have cancer. In other scenarios, it could be more important to lower the type I error rate rather than the type II error rate. For example, in a business scenario where a model predicts wheater to contact a potential customer or not, it might be best to set a higher decision threshold. Every contact is probably costly, and the organization has only limited resources. Thos one might want to contact only those potential customers who have a high chance of becoming customers. Then one can adjust the decision threshold to adapt the model to the given situation.

Part II: Robo advisors

Question 1

How is the efficient markets hypothesis related to the business model of robo advisors?

The efficient market hypothesis states that it is “impossible” to beat the market. It argues that the efficiency on the stock market leads to prices always incorporating all existing relevant information. Thus stock prices always reflect the true value of a stock. Consequently, actively managed funds should not be able to beat the market in the long run. If they do so, it is purely because of luck and not because of the skill of the manager.

Robo advisors argue that because of this theory active stock picking is useless or even harmful. Investors should much rather invest in the general market and focus on reducing fees and taxes. Depending the risk profile of an investor robo advisors calculate the optimal portfolio. These portfolios combine different asset classes and markets to generate a specific return with a minimum variance or the maximum return with a given maximum variance. Robo advisors usually invest in ETF’s or indices and never in single stocks.

Question 2

Why do different robo adivisors give different advice to the same person?

- Firslt, the different robo advisors could have different brokers and partners and thus face **different cost and fee structures for different assets**. A particuar asset could therefore be relatively more attractive with one advisor than with another one.
- Second, the different advisors could search for the optimal portfolio in **different asset universes**. For example, one advisor could also consider investments in gold while another one doesn’t.
- Last, the personal risk profile of the investor can never be truly evaluated. Every attempt to assess it is only an approximation. Thus, the different advisors will likely determine **different risk profiles for the same individual**. Because of the importance of the risk profile in determining the optimal portfolio weights, the various advisors will probably suggest different portfolios.

Question 3

Bob’s optimal portfolio

Loading the data

```
rm(list=ls())           # Clearing the workspace
dd = read.csv('prices_EXAM.csv')
colnames(dd) <- c("Asset 01",  "Asset 02",  "Asset 03",  "Asset 04",  "Asset 05",  "Asset 06",  "Asset 0
```

Q 3.i.

The covariance matrix Σ and the mean return vector μ

```
returns <- diff(as.matrix(dd)) / as.matrix(dd)[seq(1, nrow(dd)-1), ] # calculate returns
mu <- colMeans(returns) # Mean return vector
S <- cov(returns)       # covariance matrix
```

Q 3.ii.

Formular for the optimal portfolio weights with a risk free asset

$$w^* = \lambda \Sigma^{-1}(\mu - r_f)$$

with:

$$\lambda = \frac{\mu^* - r_f}{(\mu - r_f)^T \Sigma^{-1}(\mu - r_f)}$$

and:

Σ = Covariance matrix

μ = Vector of expected returns

μ^* = Target return

w = Vector of portfolio weights

w^* = Vector of optimal portfolio weights

Calculating the optimal portfolio weights
Assumptions

```
# Given:
target = 0.2/300                # Target return as given in the text
risk_free_rate = 0.001/300     # Risk free rate as given in the text
```

First we calculate λ with the above given formula

```
A <- mu - risk_free_rate      # Risk premiums over risk free rate
At <- t(A)               # Inverse of A

Si <- solve(S)           # Inverse of the covariance matrix

lambda <- ( target - risk_free_rate ) /  ( At %*% Si %*% A )  # Calucating lambda based on the formula
lambda <- lambda[[1]]    # Turning from a matrix into a scalar
```

Then we can calculate the optimal portfolio weights w^*

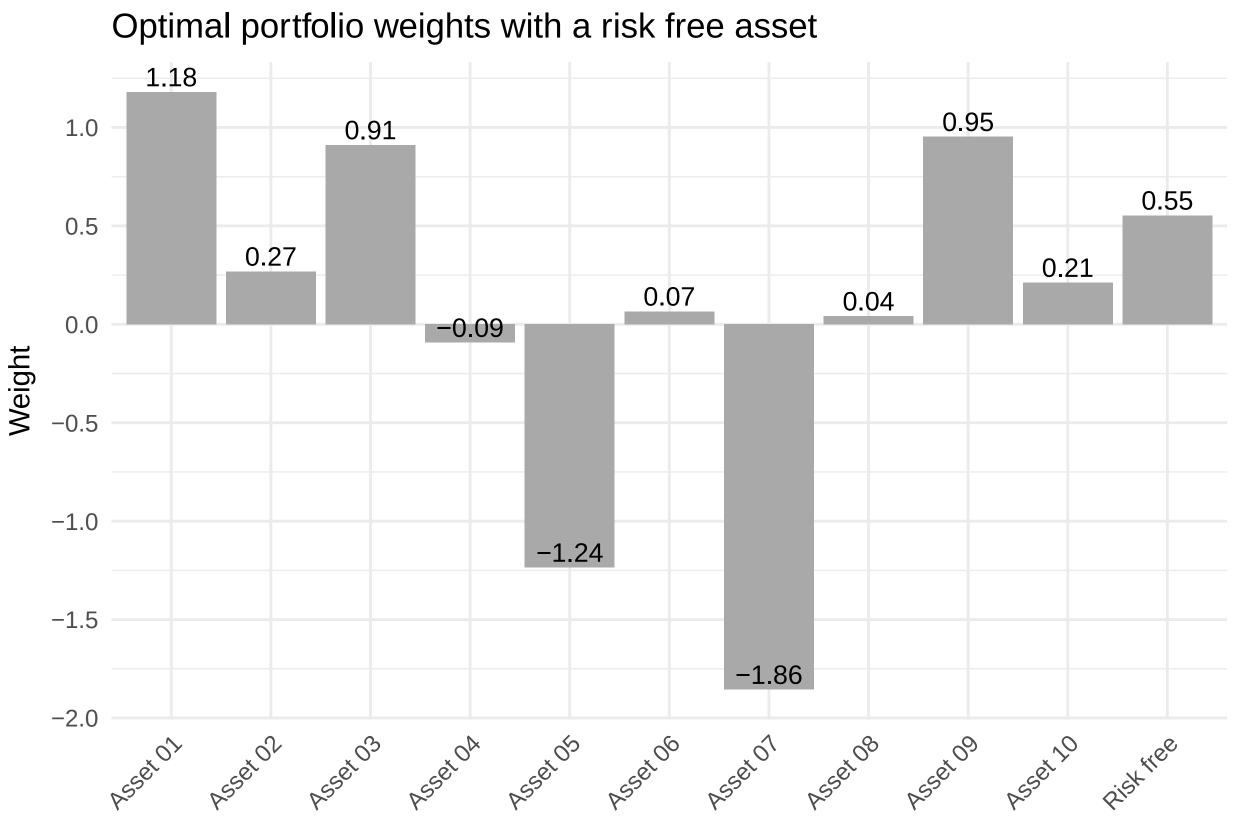
```
w_with_rfa <- lambda * Si %*% A # calculate lambda
w_with_rfa <- t(w_with_rfa) # transpose w so we get a vector with the optimal weights

risk_free_weight <- 1- sum(w_with_rfa)

w_with_rfa[ncol(w_with_rfa)+1] <- risk_free_weight
cat("Bob should invest", round(risk_free_weight,3),"% of his portfolio into the risk free asset.")
```

```
## Bob should invest 0.553 % of his portfolio into the risk free asset.
```

Displaying the portfolio

```
asset_names <- names(mu)
asset_names[11] <- c('Risk free')
```

```
df <- data.frame(Weight=w_with_rfa,
                Assets=asset_names)

library(ggplot2)
ggplot(data=df,aes(x=Assets,y=Weight )) +
  geom_bar(stat="identity", fill="darkgrey")+
  geom_text(aes(label=round(w_with_rfa,2)), vjust=-0.3, size=3.5)+
  xlab("") +
  scale_fill_manual(values=c("grey60","grey80"))+
  ggtitle('Optimal portfolio weights with a risk free asset') +
  theme_minimal() +
  scale_y_continuous(breaks=seq(-2,1.25,1/2))+
  theme_minimal() +
  theme(axis.text.x=element_text(angle=45, hjust = 1))
```



Q 3.iii.

Formula for optimal portfolio weights without a risk free asset

$$w^* = \frac{1}{D}(B\Sigma^{-1}u - A\Sigma^{-1}\mu + \mu^*(C\Sigma^{-1}\mu - A\Sigma^{-1}u))$$

with

$A = u^T\Sigma^{-1}\mu$

$B = \mu^T\Sigma^{-1}\mu$

$C = u^T\Sigma^{-1}u$

$D = BC - A^2$

Setting up the necessary functions

```
weights = function(S, mu, target) {
    u = rep(1, length(mu))
    Si = solve(S)
    A = u %*% Si %*% mu
    B = mu %*% Si %*% mu
    C = u %*% Si %*% u
    A = A[[1]]
    B = B[[1]]
    C = C[[1]]
    D = B * C - A^2

    w = 1 / D * (B * Si %*% u - A * Si %*% mu + target * (C * Si %*% mu - A * Si %*% u))

    return(w)
}

estimate_mu_S = function(returns) {

    mu = colMeans(returns)
    S = cov(returns)

    out = list(mu=mu, S=S)

    return(out)
}
```

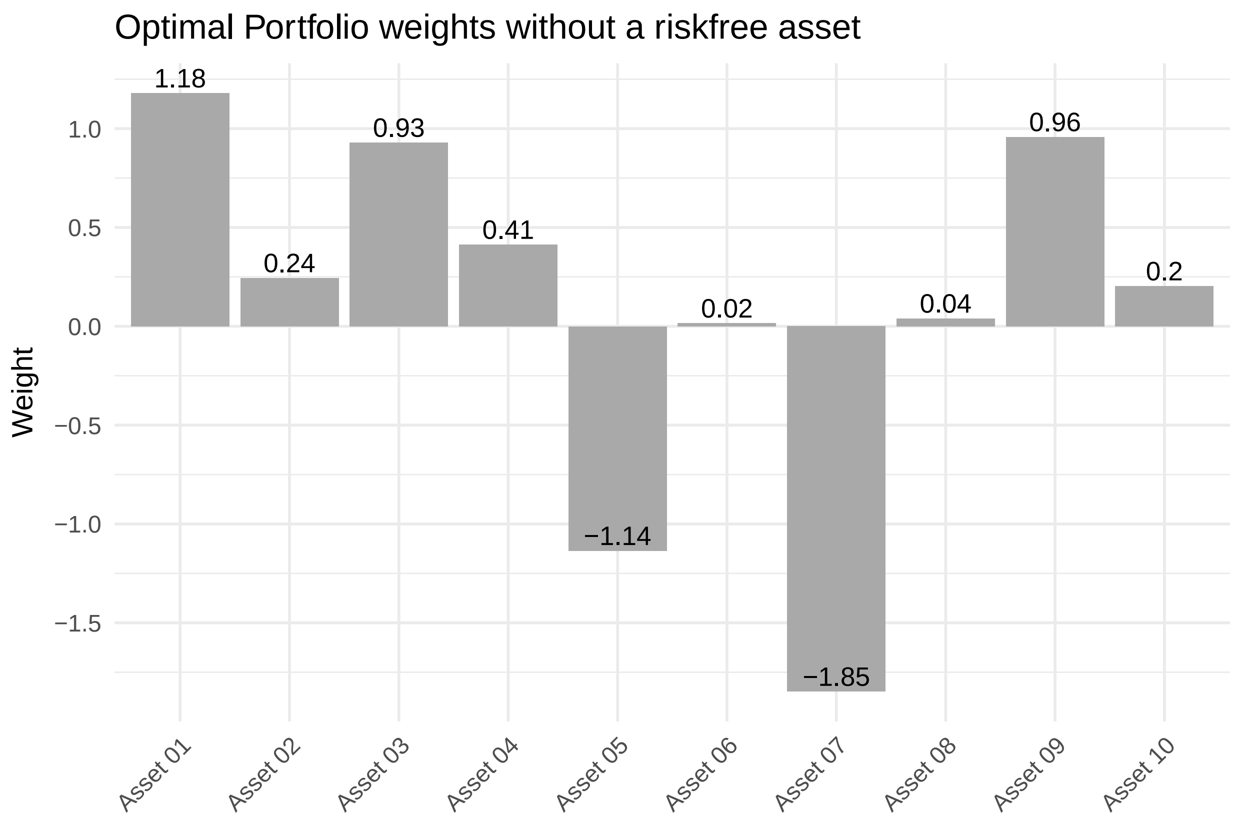
Calculating the optimal weights

```
target_new = 0.2/300 #0.1 / 300

out = estimate_mu_S(returns)
w_no_rfa = weights(out$S, out$mu, target_new)
print(t(w_no_rfa))
```

```
##      Asset 01  Asset 02  Asset 03  Asset 04  Asset 05 Asset 06  Asset 07
## [1,] 1.179529 0.2445241 0.9293527 0.4133565 -1.135576 0.015189 -1.847846
##        Asset 08  Asset 09  Asset 10
## [1,] 0.03996226 0.9581357 0.2033728
```

Displaying the portfolio

```
asset_names <- colnames(dd)

df <- data.frame(Weight=w_no_rfa,
                 Assets=asset_names)


library(ggplot2)
ggplot(data=df,aes(x=Assets,y=Weight )) +
  geom_bar(stat="identity", fill="darkgrey")+
  geom_text(aes(label=round(Weight,2)), vjust=-0.3, size=3.5)+
  xlab("") +
  scale_fill_manual(values=c("grey60","grey80"))+
  ggtitle('Optimal Portfolio weights without a riskfree asset') +
  theme_minimal() +
  scale_y_continuous(breaks=seq(-2,1.5,1/2))+
  theme_minimal() +
  theme(axis.text.x=element_text(angle=45, hjust = 1))
```



Q 3.iv.

Michaud's method or re-sampeling

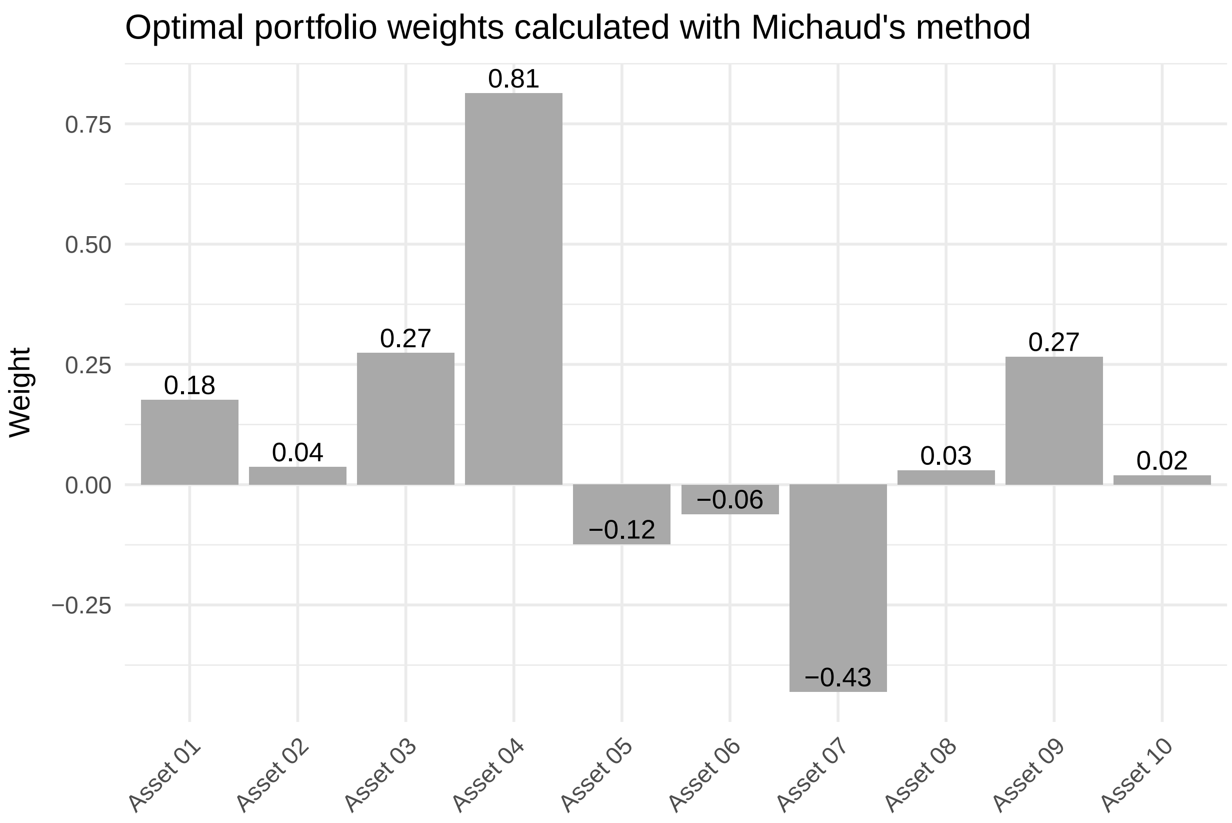
In order to perform Michaud's method we need to first sample the returns

```r
target_new = 0.2/300
N <- 100 # Number of samples
set.seed(1000)

w_matrix <- matrix(, N ,10) # settig up a matrix for the different weight vectors

Index <- seq(1,nrow(returns)) # generating an index to sample N numbers from

for (i in 1:N) {
  sample_index <- sample(Index, 100, replace= TRUE)   # sampling N numbers
  out = estimate_mu_S(returns[sample_index,])         # calculating mu and S
  w_matrix[i,1:10] = weights(out$S, out$mu, target)   # calculating the weights
}

w_michaud <- colSums(w_matrix)/N  # averaging the N weight vectors to get the final weight vector
names(w_michaud) <- colnames(dd)

print(round(w_michaud,4))
```

```
## Asset 01 Asset 02 Asset 03 Asset 04 Asset 05 Asset 06 Asset 07 Asset 08
##   0.1764   0.0368   0.2737   0.8137  -0.1239  -0.0613  -0.4309   0.0299
## Asset 09 Asset 10
##   0.2660   0.0197
```

Displaying the portfolio

```r
df <- data.frame(Weight = w_michaud,      # generating a dataframe for ease of handling
                Assets = names(w_michaud))

library(ggplot2)    # producing the plot
ggplot(data=df,aes(x=Assets,y=Weight )) +
  geom_bar(stat="identity", fill="darkgrey")+
  geom_text(aes(label=round(Weight,2)), vjust=-0.3, size=3.5)+
  xlab("") +
  ggtitle('Optimal portfolio weights calculated with Michaud\'s method') +
  theme_minimal() +
  scale_y_continuous(breaks=seq(-1,1.5,1/4))+
  theme_minimal() +
  theme(axis.text.x=element_text(angle=45, hjust = 1))
```


Optimal portfolio weights calculated with Michaud's method
Weight
0.75
0.50
0.25
0.00
−0.25
0.18
0.04
0.27
0.81
−0.12
−0.06
−0.43
0.03
0.27
0.02
Asset 01
Asset 02
Asset 03
Asset 04
Asset 05
Asset 06
Asset 07
Asset 08
Asset 09
Asset 10

Q 3.v.

No risk free asset and no short-selling

Now we want to find the optimal portfolio weights without a risk free asset in the presence of a no short sale constraint. We can calculate the optimal weights in this scenario with the package quadprog.

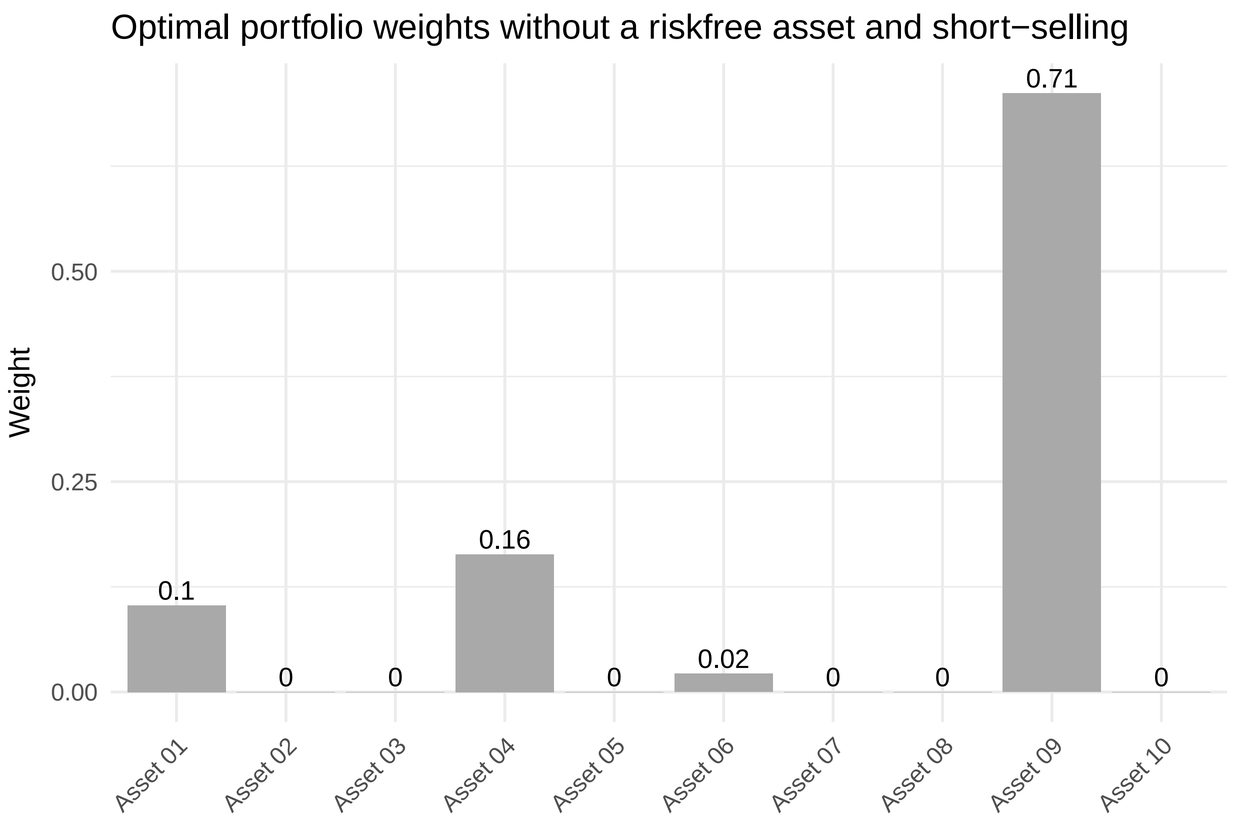
```r
require(quadprog)
```

```
## Loading required package: quadprog
```

```r
target_new   = 0.2/300

# Recalculating for safety
out <-  estimate_mu_S(returns)
mu <-  out$mu
S <-  out$S
u <-  rep(1, length(mu))  # vector with just one's we need for the constraint matrix
n <- ncol(returns)

# Constraint matrix
Con <- cbind( # One constraint per column
  mu,                           # Target return constraint
  matrix( rep(1,n), nrow=n ),   # The weights have to sum up to 1
  diag(n)                       # No short-selling constraint
)

# Matrix with the 3 constraint conditions
ci <- c(target, 1, rep(0,n))

out = solve.QP(S, rep(0, length(mu)), Con, ci,meq=2)
w_noshort = out$solution
```

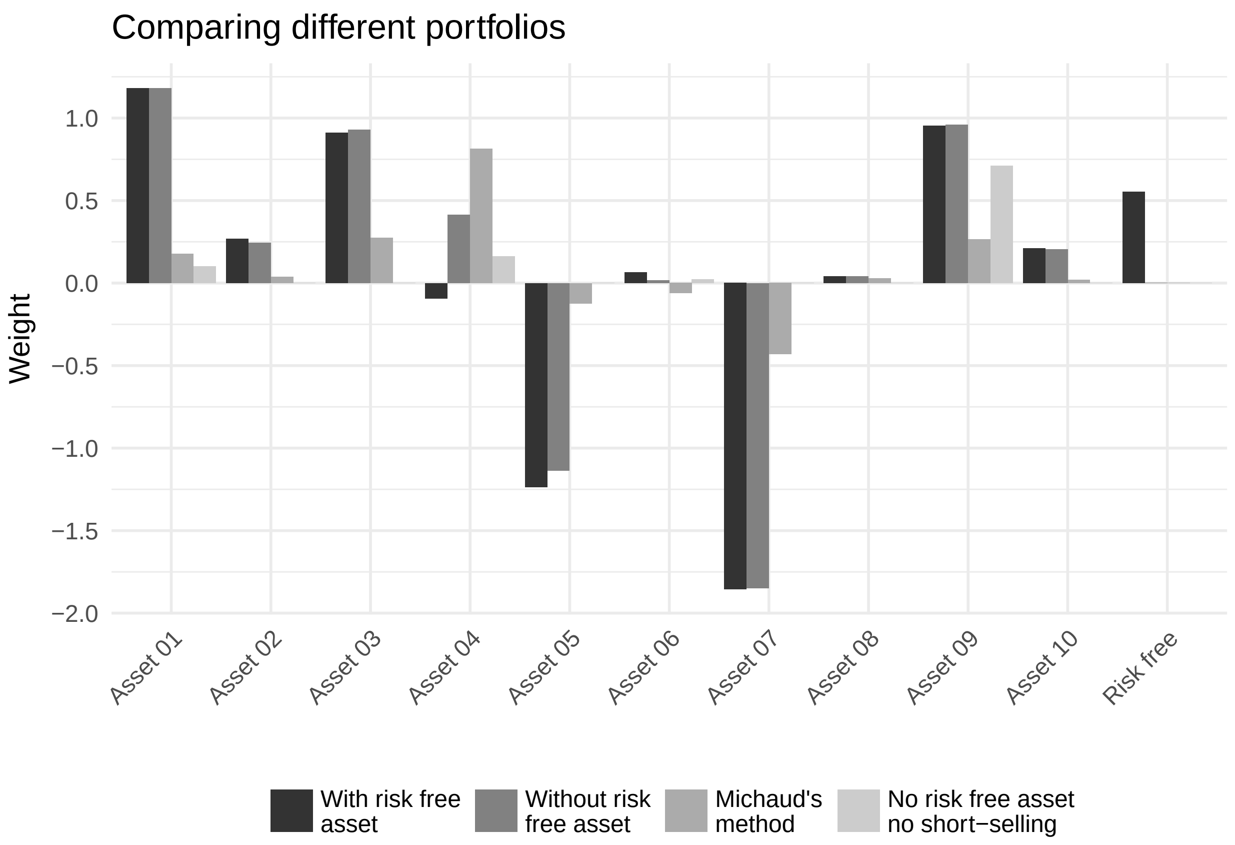
Visualize the portfolio

```r
asset_names <- colnames(dd)

df <- data.frame(Weight = w_noshort,
                 Assets = asset_names)


ggplot(data=df,aes(x=Assets,y=Weight )) +
  geom_bar(stat="identity", fill="darkgrey")+
  geom_text(aes(label=round(Weight,2)), vjust=-0.3, size=3.5)+
  xlab("") +
  ggtitle('Optimal portfolio weights without a riskfree asset and short-selling') +
  theme_minimal() +
  scale_y_continuous(breaks=seq(0,1,1/4))+
  theme_minimal() +
  theme(axis.text.x=element_text(angle=45, hjust = 1))
```



Compare all the different portfoilios with each other

```r
w_no_rfa[11] <- 0
w_michaud[11] <- 0
w_noshort[11] <- 0

my_weights <- data.frame(
                AssetID = c(colnames(dd),"Risk free"),
                w_with_rfa = w_with_rfa,
                w_no_rfa = w_no_rfa,
                w_michaud = w_michaud,
                w_noshort = w_noshort, row.names = c(names(dd),"Risk free")
                )

my_weights <- melt(my_weights,id.vars = c("AssetID"))

ggplot(data=my_weights,aes(x=AssetID,y=value, fill=variable )) +
  geom_bar(stat="identity",position=position_dodge()) +
  ggtitle('Comparing different portfolios') +
  scale_fill_grey(name="",
      breaks=c("w_with_rfa", "w_no_rfa", "w_michaud", "w_noshort"),
      labels=c("With risk free \nasset", "Without risk \nfree asset", "Michaud's \nmethod","No risk fre
  xlab("") +
  ylab("Weight")+
  scale_y_continuous(breaks=seq(-2,1.5,1/2))+
  theme_minimal() +
  theme(axis.text.x=element_text(angle=45, hjust = 1)) +
  theme(legend.position="bottom")
```


Comparing different portfolios
Weight
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Asset 01
Asset 02
Asset 03
Asset 04
Asset 05
Asset 06
Asset 07
Asset 08
Asset 09
Asset 10
Risk free
With risk free asset
Without risk free asset
Michaud's method
No risk free asset no short−selling

Part III: Blockchain

Question 1

The three proporties of cryptographic hash functions

Definition

Collision resistance - Collision resistance means that it is computationally infeasible to find two inputs for a cryptographic hash function that produce the same output. Mathematically speaking it is practically impossible to find two inputs $x \in X$ and $y \in X$ with $x \neq y$ such that $H(x) = H(y)$. However, until today no known hashing function is provable collision resistant

Concealing - Concealing means that one cannot guess in a reasonable time which input produced a specific output. Even if he or she knows the hashing function and how it works. The output, therefore, does not reveal any information about the input. That also means that a specific output does not increase the probability that the input lies in a certain range or area. Formally, a hash function H is concealing if, for some secret r drawn from p(r) and given H(r || x), it is infeasible to find x.

Puzzle friendliness - This property of a hashing function is similar and yet different to the concealing property. A hash function H is puzzle friendly if it is unfeasible to find an x such that $H(k || x) = y$ in time significantly less than O(2n). The following applies: k is chosen from a distribution with high min-entropy and n is the number of elements in the output set. This has to be true for every possible n-bit output value y.

Application

Collision resistance - One application of this property is the SHA-256 hash function. It assigns a distinct 256bit hash to every possible input sequence. Since there are endless possible input sequences and only 2ˆ256 possible output hashes, there must be many examples where two inputs produce the same output hash. In reality, 2ˆ256 is such a large number that we will probably never find a pair of inputs that produce the same output. Therefore the SHA-256 hash function is collision free.

Concealing - Consider a blind bid auction. To achieve the highest possible price, it is important that the participants of the auction do not know the bids of each other. If they would know them, they could simply bid one dollar more then the highest bid. In addition to that, it is also important that one can not change his bid after the deadline. In such a situation a concealing hash function can be applied. Every party has to calculate the hash of their bid with a given hash function and publish that hash before the deadline. After the deadline the clear bids are also published and with the hash function it can be verified that these are in fact the bids given before the deadline.

One problem is that there is often only a limited range of reasonable bids. One could simply calculate all the hashes of all the bids in a certain range and compare them to the hashes of the competing parties to find out their bid. To make this nearly impossible hashes are calculated with a combination of a random number and the bid itself. This procedure makes it impossible to check all reasonable bids.

Puzzle friendliness - Puzzle friendliness is primarily important for the search puzzles / lottery that is at the core of the blockchain technology. Miners are required to perform a brute force search to find x such that $H(k || x) = y \in Y$, where H is the hashing function, and Y is a subset of the domain of H (i.e. $Y \in Z$). The puzzle friendliness property ensures that H(x || k) are distributed “uniformly at random” over the domain of Z. This means one can not infer that x must lie in a certain range of the information where Y is. Thereby a perfect lottery can be generated.

Question 2

How to ensure that the blockchain has not been corrupted

Q2.i

One important property of the blockchain is that it is immutable. This means that if one has the hash of the last block of the blockchain he can always check with 100% accuracy if the blockchain has been tampered with. Even in the slightest of ways and even if something was changed at the beginning of the chain. This is of great importance since the concept of a distributed ledger gives everyone the ability to possibly corrupt the blockchain to gain personal benefits. However, such a change would be immediately noticed because of the very way the blockchain is interlinked.

Every block incorporates the hash of the previous block. Due to the properties of a good hashing function that we already described above, it is impossible to change a block without changing its hash. Since every block is interlinked with the previous block one can check and ensure the consistency of the blockchain by simply holding the final hash at a secure place.

So if someone would like to change something in block 30 that would also change the hash of block 30. Because this hash is incorporated in block 31 the hash of block 31 would not fit anymore. So a potential attacker would have to recompute the hash of block 30, block 31, block 32 and so one. Due to the proof-of-work-concept and similar mechanisms it is nearly impossible to do this in a feasible amount of time.

Q2.ii

It is important to store a distributed ledger in a tamper evident data structure such as a block chain. This is extremely important due to the very nature of a distributed ledger. Because the data is not stored at a central authority and everyone has access to the complete history of transactions it is vitally important to have a simple and reliable mechanism that ensures the immutability of the blockchain and thereby the database. Thereby the necessary trust into the distruibuted ledger is generated. Since no one can change the date without someone noticing everyone can have access to it. This makes the idea of a distributed ledger possible in the first place.

Question 3

When is a bitcoin transaction valid?

First, we have to define what a Bitcoin transaction consists of. Let's look at an example. Alice sends 1 Bitcoin to Bob. The corresponding transaction consists of the following three parts:

1. **The input** - A record of the bitcoin address that sent the Bitcoin to Alice in the first place. Remember that the Blockchain does not store balances but only past transactions.
2. **The amount** - How many Bitcoins does Alice send to Bob?
3. **The output** - Bob's Bitcoin address

Alice has to verify this transaction with her private key. The public key (Also called the bitcoin address) and the private key form a cryptographic key pair. Alice can publish her public key so others can send Bitcoin to her and she can prove that she is the owner of the address by signing transactions with her private key. Miners can verify the validity of transactions signed by Alice private key without actually knowing her private key. This works because of the specific properties of the cryptographic key pair consisting of the public and the private key. It is important to remember that anyone who has access to the private key of an account can send Bitcoins from this account.

So to sum up, a transaction is valid if miners verify the following points:

- The transaction is formally correct
- Alice has received enough unspent Bitcoins in the past
- Alice has signed the transaction with her private key so that miners can check it with her public key (The address of the sending account)

Once the transaction is verified, it gets bundled into a new block of transactions with all valid transactions of the past 10 minutes. Miners then engage in the proof-of-work process and try to find a string that when added to the block causes the block to have a hash with a specific number of leading zeros. The number of leading zeros determines the difficulty of the puzzle and depends on the size of the network engaging in the search puzzle.

Question 4

Bitcoin consensus protocol

Q. 4.i

Many experts say that distributed trustless consensus is the primary innovation of the blockchain technology. Usually, trust is needed to reach consensus. Not so in the blockchain universe.

The basic premise of the blockchain technology is that one cannot trust anyone else. Reason being that in the digital realm millions of entities can be created with a few clicks. Majority opinions are therefore useless and not a trustworthy way to reach consensus about the state of a specific system like a digital currency. The only practical solution is that one confirms the validity of data by himself. To do so, one, of course, needs access to the respective data. In the case of digital currency, one needs to check the complete transaction history to confirm the validity of the current supposed state. This leads to the concept of a distributed ledger. Everyone has full access to all data to verify the correctness of it. The problem arises when new data like for example a transaction is added to the blockchain. Two different entities could both add new data to the chain and propose their transactions to the network at the same moment. How can the system agree on one chain and how can it ensure that once it decides it will not change its decision in the future?

The way the Bitcoin blockchain solves this problem is called proof-of-work. This concept adds artificial computational difficulty to the ledger. It thereby makes it extremely difficult to change something that already got included in the blockchain while keeping the validity of the chain intact. To do so would take an extremely large amount of computational power since one would have to recalculate every hash of every block that follows the now changed one. The computational effort or work that the so-called miners put into the blockchain thereby ensures that its immutability. In other words, mining is the process of generating proof-of-work.

It workes as follows. When a new block is proposed as an addition to the chain miners try to find a number that in combination with the rest of the block causes the hash of the block to have certain rare characteristics. The key here is that finding such a number is hard while verifying it is a valid number is easy. Every block has to include such a rare number, and the difficulty of these rare numbers has to depend on the size of the network searching for it (Measured through the Hash-rate). The miner who finds one of those numbers gets awarded a reward for this efforts. Thereby the network is incentivized to participate in the proof-of-work process.

Q. 4.ii

A Sybil attack is a specific form of an attack on a peer-to-peer network. The idea is that in a network where consensus is formed based on how many entities share a specific belief, one can create pseudo identities to gain a disproportionally large influence. This influence can then be used to convince the network of any particular truth that benefits the attacker. For example in a distributed ledger that merely relies on the majority opinion to reach consensus about the current state of the system one could apply a Sybil attack to steal money from others. The attacker would start generating fake identities under his controls until he controls a majority of the entities in the network. He can then present a corrupt state of the ledger to the network with all the entities under his control. Other entities will accept this new state as the truth because they think the majority proposes it.

Such an attack is not possible in the Bitcoin system due to Bitcoins proof-of-work consensus protocol. Instead of merely using the majority of entities as a way to find consensus Bitcoin uses the majority of computational power to do so. It is therefore much more difficult to perform a similar attack on the Bitcoin blockchain. One would have to control more than 50% of the entire hashing power of the bitcoin network which is nearly impossible. Besides that, if someone would corrupt the network Bitcoin would probably loose its value anyways.

Q. 4.iii

Due to the nature of the blockchain, a transaction cannot be considered final just because it was added to a block that was published to the network. This is because there could emerge a longer blockchain than the one where this particular transaction was appended to. If all miners agree on this longer blockchain, the transaction has never happened. Because of this problem, it is said that the transaction has been mined at a depth of 1 block when it has just been added to a blockchain. The more blocks that are added to this chain the more secure we can be that our transaction is incorporated in the consensus blockchain. To ensure against double spending one should not consider a transaction as confirmed until it is a few blocks deep incorporated in the chain.

Q. 4.iv

The proof-of-work consensus algorithm requires a lot of computational power. Miners have to solve the search puzzle and can only do this through a brute force method. On the one hand, this algorithm ensures the reliability of the Bitcoin blockchain. Yet, on the other hand, it consumes an incredible amount of energy that could be put to better use.

In fact, according to the Bitcoin Energy Consumption Index, the estimated current energy consumption of Bitcoin is already at 25.25 TWh. This number represents 0.12% of the worlds electricity consumption. A single Bitcoin transaction requires approximately 100 kWh of energy. With this amount of energy 3 US household could still their energy consumption of one day. Due to the high energy consumption, most Bitcoin miners are located in countries with low energy costs like China.

The proof-of-work consensus algorithm is only one of many consensus algorithms. Because of its simplicity, it was the first one that was widely adopted by users. In the future however, it will probably not be the most widely used one. More energy efficient alternatives like proof-of-stake, demonstrate that a distributed ledger technology based cryptocurrency is also possible without such a high amount of “useless” energy consumption. To gain broad acceptance Bitcoin and cryptocurrencies, in general, have to find a way to become more energy efficient.

Question 5

The Etherum platform

Q.5.i.

There are two types of accounts in Ethereum. Namely, Externally Owned Accounts (EOA) and Contracts Accounts (CA). EOA’s are accounts controlled by a private key. The person how has the private key associated with an EOA has the ability to send ether and messages from this account. CA, on the other hand, are not owned by anyone. These accounts have some code attached to them, and they are controlled by this code.

The characteristics of the two accounts are as follows:

Externally owned accounts (EOAs)	Contract accounts
Has an ether balance	Has an ether balance
Can send transactions (Ether transfer or trigger contract code)	Code execution is triggered by transactions or messages from other contracts
Controlled by private keys	Persistent storage - can have a permanent state
No associated code	Has associated code
-	Can send message to other contract accounts

Everything that happens on the Etheru blockchain is set in motion by EOAs. EOA can creat CA and can send messages to them so that they perform their code. CA can also send messages to other CA to perform more complex tasks. The code of these CA is evaluated on each node participating in the Etherum network every time a new block is verified. Due to its self-fulfilling property, the code associated with CA is often also called a smart contract.

Q.5.ii.

The Etherum blockchain enables users to perform arbitrary programs "on the blockchain". This is the main advantage of Etherum over Bitcoin since Bitcoin only allows users to execute non-turning and straightforward scripts of code. Such scripts that run automatically if a specific predefined condition is met can be incredibly useful in many different situations.

One setting which could be substantially disrupted by the Etherum platform in the future is a situation in where two parties need a trustee in the middle to perform a transaction. For example, consider the following case. A wants to buy a house from B. Ideally, A would hand the purchase price over to B, and then B would transfer the ownership of the house to A. In reality, however, such transactions often require a trustee as a third party because A and B don't trust each other enough. Of course, B could take the money from A without giving up his ownership status. In precisely such a situation an Etherum CA or often called smart contract could solve the trust issue and take the place of the third party. The CA could be a short script that acts similar to a trustee. Both A and B send their money/ownership certificate to the CA, and if the CA has received both, he will forward them. If the CA only receives one and not the other, he will send it back to its original owner.

Of course, it is most important that the network can fully trust on the correct execution of these scripts. The network has to reach consensus about the current state of the system similar to the Bitcoin blockchain. It currently achieves this through a proof-of-work mechanism. Like in the Bitcoin network Etherum miners execute transactions (This includes running the code of CA) and solve a proof of work puzzle. The winning miner of the proof-of-work puzzle proposes a new block containing all transactions and therefore a representation of the new state. Through this mechanism, Ethereum provides trust on the results of the execution of code on CA.

Question 6

Comparing a centralized and a decentralized ledger

Pros:

Centralized	Decentralized
System that has proven itself over centuries	No trust in central authority needed
Requires no consensus to take action	Data is complete, consistent and widely available on every node
Lower electricity consumption (eco-friendly)	No single database and system on which the network relies
There is a central authority that can be contacted in case of problems	Immutability (tamper-proof)
Transactions can be repealed	Lower transaction fees (can also become quite high)

Cons:

Centralized	Decentralized
One Institution has complete control over the network	Draws a lot of energy (proof-of-work)
Needs trust in the third party	Codebase can be corrupted
Single point of failure	Anonymity attracts people with bad intentions
Possibly higher fees	Transactions are irreversible
No editing restrictions	Append-only
asfsdvsfdv	Has not proven itself over a long period of time

Conclusion:

There is no clear winner because both systems have significant advantages and disadvantages compared to each other. In some situations, a centralized system is the better choice in other situations a decentralized one is more beneficial. Ith highly depends on the setting which system is optimal.

By Alexander Steeb - alexander.steeb@gmx.de - 10.11.2017 - St.Gallen