Project 3: Ovarian Cancer

Contents

1. Project Overview

1.1 General facts about Ovarian Cancer

Ovarian cancer (OC) is the one of the leading cancer death causes in women. One of 71 women will be diagnosed with this disease at some point during her lifetime. When treated in early stages, OC has relatively high survival rates. However, symptomes mostly start to develop when the tumour has already progressed to advanced stages, which is why the overall 5-year survival rate for OC lies under 50%.

Ovarian cancer is a very heterogenous disease and mostly represented by epithelial tumours. There are four main subtypes: serous, endometrioid, mucinous and clear cell. Furthermore, tumours can be classified dependent on their state of differentiation. Low grade (LG) Ovarian cancer is well differentiated and therefore not invasive; high grade (HG) tumours are often aggressive and frequently metastasize.

Standard therapy approaches include surgery and chemotherapy with DNA-damaging agents. Personalized immunotherapeutic approaches often target special features of cancer cell genome and proteome such as over-expressed surface receptors.

1.2 Common mutations in OC

Literature on Ovarian cancer states that the most common mutations are found in the DNA repair-related genes BRCA1 and BRCA2. Abnormalities in other DNA and mismatch repair genes have been found too, as well as TP53, which is also known as “guardian of the genome”. According to Testa et al., TP53 accounts for most mutations found in high grade serous Ovarian cancer. Allegedly, another very common mutation is ARID1A.

1.3 Project outline

After the data cleanup, our first task will be to verify the information about the common driver mutations and to gain further insight on potential driver mutations in Ovarian cancer. Further analysis will be revolving around these driver mutations. Among other things, co-existing mutations will be investigated in order to assess cancer development.

The next step, which aims at the development of potential future cancer treatments, focuses on the identification of so-called synthetic lethality interaction partners. Those are genes whose knockout specifically leads to cell death in cancer cells, depending on the cells' genotype and therefore also their phenotype. This analysis will be conducted with respect to ocurring driver mutations and cancer subtype. Finally, a regression analysis will be performed in order to gain more basic insight on the influence of a gene's copy number on gene expression. Obtained results will allow a more secure assessment of meaningfulness of cancer-related gene amplifications.

2. Data Cleanup

At first the dataset needs to be downloaded into R-Studio.

```
allDepMapData <- readRDS("~/GitHub/project-01-group-
03/DepMap19Q1_allData.RDS")
```

A re-naming of the initial matrices facilitates the following workflow.

```
copynumber = allDepMapData[["copynumber"]]
mutation = allDepMapData[["mutation"]]
kd.ceres = allDepMapData[["kd.ceres"]]
kd.prob = allDepMapData[["kd.prob"]]
annotation = allDepMapData[["annotation"]]
expression = allDepMapData[["expression"]]
```

Next, the Ovarian Cancer cell lines will be extracted, since the remaining cell lines do not pose relevance for our project.

```
annotation = annotation[which(annotation$Primary.Disease == "Ovarian
Cancer"), ] # The annotation matrix now only consists of Ovarian Cancer cell
lines
ID = annotation$DepMap_ID # DepMap_ID is the column name for the Ovarian
Cancer cell lines
expression = expression[ , which(colnames(expression) %in% ID)] # Extract the
ovarian cancer cell lines from expression, copynumber, kd.ceres, kd.prob and
mutation
copynumber = copynumber [ , which(colnames(copynumber) %in% ID)]
kd.ceres = kd.ceres [ , which(colnames(kd.ceres) %in% ID)]
kd.prob = kd.prob [ , which(colnames(kd.prob) %in% ID)]
mutation = mutation [ ID]
```

The dataframes with information about copynumbers, gene expression and knockdown data, whose rows consist of the genes, will now be ordered alphabetically. This step will be useful later on to compare different variables for certain genes.

```
copynumber <- copynumber[order(rownames(copynumber)),] # order the rownames
alphabetically
expression <- expression[order(rownames(expression)),]
kd.ceres <- kd.ceres[order(rownames(kd.ceres)),]
kd.prob <- kd.prob[order(rownames(kd.prob)),]
```

Further work with the annotation matrix is made substantially easier by naming the rows with respect to the cell lines' names. So far, the rownames comprise of numbers without apparent relevance. What is more, unnecessary columns are removed from the annotation matrix and the main data is removed from the workspace.

```
rownames(annotation) = annotation$DepMap_ID # The column "DepMap_ID" contains
the cell line' names

# removing unecessary columns
annotation = annotation[, -which(colnames(annotation) %in% c("DepMap_ID",
"Aliases", "Primary.Disease", "Gender", "Source"))]

#removing the main data, because we already extracted the OC Data
rm(allDepMapData)
```

NA values often pose problems for functions in R and cannot always be automatically coerced. For this reason, rows containing NA values will be removed from the matrices.

Install packages

To perform our analysis, several R packages with useful functions need to be installed and loaded.

```
install.packages("reshape", repos = "http://cran.us.r-project.org") # hier
ein if else machen! Wenn package installiert, nur in der library

## Installing package into 'C:/Users/pssst/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)

## package 'reshape' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\pssst\AppData\Local\Temp\Rtmp8Y8sNt\downloaded_packages

install.packages("ggplot2", repos = "http://cran.us.r-project.org") # laden,
nicht nochmal herunterladen. Vllt hat David sowas in seinen

## Installing package into 'C:/Users/pssst/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)

## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\pssst\AppData\Local\Temp\Rtmp8Y8sNt\downloaded_packages

install.packages("data.table", repos = "http://cran.us.r-project.org") #
Beispielen mal gemacht

## Installing package into 'C:/Users/pssst/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)

## package 'data.table' successfully unpacked and MD5 sums checked
##
```

```
## The downloaded binary packages are in
##  C:\Users\pssst\AppData\Local\Temp\Rtmp8Y8sNt\downloaded_packages
```

```
install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/pssst/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## package 'gridExtra' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\pssst\AppData\Local\Temp\Rtmp8Y8sNt\downloaded_packages
```

```
library(reshape)
library(ggplot2)
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:reshape':
##
##     melt
```

```
library(gridExtra)
```

3. Driver Mutations

Fuse mutation lists to one matrix.

```
mutation.all = as.data.frame(rbindlist(mutation))
```

Our analyses just need the information in certain columns of our data frame, for example gene name and location (chromosome), cell line, the kind of mutation (missense, frame shift, etc.). Thus, we extract these columns and put them in the data frame "mutation.all".

```
 mutation.all = mutation.all[, which(colnames(mutation.all) %in%
c("Hugo_Symbol", "DepMap_ID", "Variant_Classification", "Variant_annotation",
"isTCGAhotspot", "Chromosome", "isDeleterious"))]
```

There are different types of mutations. Some of them are silent, which means that the amino acid sequence is not altered and the protein structure is not affected. Other mutations lead to an amino acid exchange, but do not provoke dramatic conformational changes in the encoded protein. Whether or not a mutation has an impact on protein function is noted in the column "isDeleterious". We want to extract all mutations that are TRUE for isDeleterious, since these might have something to do with cancer cell development.

```
mutation.all = mutation.all[which(mutation.all$isDeleterious == "TRUE"), ] #
only include rows (=genes) that have deleterious mutations in data frame
"mutation.all"
```

To simplify work the matrix mutation.all was ordered alphabetically.

```
mutation.all <- mutation.all[order(mutation.all$Hugo_Symbol),]
```

Find most frequently mutated genes by summing up all “mutation events” found in all cell lines. To do this, the names of the genes were converted to factors.

```
mutation.all$Hugo_Symbol = factor(mutation.all$Hugo_Symbol)
```

The most common mutation mutation found is ARID1A (16 times). Surprisingly, BRCA1 and BRCA2 are only mutated twice and once, respectively.

Problem: Many mutations occur several times in one cell line, but in different DNA loci. If we want to identify the most frequent mutations among all cell lines, these duplicates should not be included. We want to extract mutations that occur frequently in different cell lines. Therefore, we use the “duplicated” function from the dplyr package and define a new data frame called new_uniq that only contains one mutation of a gene per cell line for simplicity (since the type of mutation is not relevant here).

```
duplicates <- which(duplicated(mutation.all[c('Hugo_Symbol', 'DepMap_ID')]),
) #find all identical combinations of mutated gene and cell line

new_uniq <- mutation.all[!duplicated(mutation.all[c('Hugo_Symbol',
'DepMap_ID')]),] #do not include duplicates in data frame new_uniq
summary(new_uniq)
```

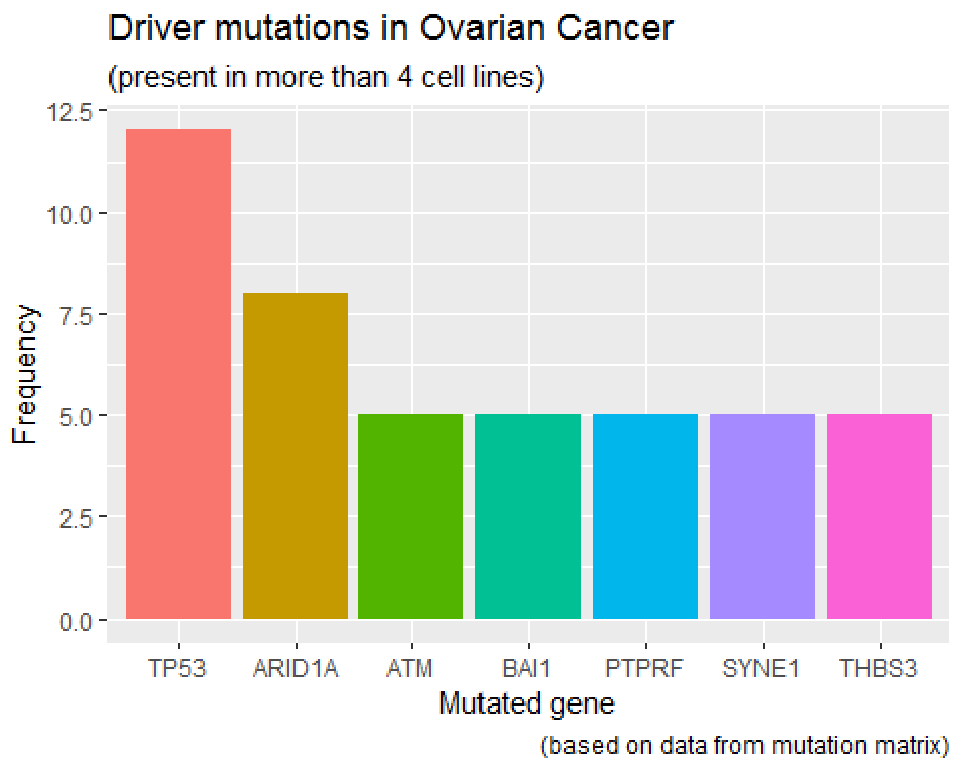
```
##   Hugo_Symbol    Chromosome         Variant_Classification isDeleterious
##  TP53    :  12   Length:3016        Length:3016            Mode:logical
##  ARID1A  :   8   Class :character   Class :character       TRUE:3016
##  ATM     :   5   Mode  :character   Mode  :character
##  BAI1    :   5
##  PTPRF   :   5
##  SYNE1   :   5
##  (Other):2976
##  isTCGAhotspot   Variant_annotation  DepMap_ID
##  Mode :logical   Length:3016        Length:3016
##  FALSE:2841      Class :character   Class :character
##  TRUE :175       Mode  :character   Mode  :character
##
##
##
##
```

As we can see, ARID1A is only the second most common mutation after TP53, which is mutated in 12 of 34 different cell lines. The most common mutations among our cell lines are thought to be our driver mutations. We will be conducting further analyses with these genes.

For a better overview, we solely plot genes that are mutated more than 4 times in total. Further analyses will focus on these designated driver mutations.

```
# Create a new data frame that contains the mutated genes (but just once per
cell line, as defined in new_uniq)
DriverMutation <- as.data.frame(table(new_uniq$Hugo_Symbol))

#we extract all mutations that occur at least one time for a later step
```

```
dm_null = DriverMutation[which(DriverMutation$Freq > 0), ]
dm_null = dm_null[order(dm_null$Freq, decreasing = TRUE),]
rownames(dm_null) <- dm_null$Var1

# extract all mutations that occur more than 4 times among our cell lines in
total
DriverMutation = DriverMutation[which(DriverMutation$Freq > 4), ]

DriverMutation$sub <- with(DriverMutation, reorder(DriverMutation$Var1, -
DriverMutation$Freq, mean, na.rm = T))

ggplot(data = DriverMutation) +
geom_col(mapping = aes(x = DriverMutation$sub, y = DriverMutation$Freq, fill
= DriverMutation$sub), show.legend = F) +
labs(x = "Mutated gene", y = "Frequency", title = "Driver mutations in
Ovarian Cancer", subtitle = "(present in more than 4 cell lines)", caption =
"(based on data from mutation matrix)") # create a barplot of the most
frequent mutations, displaying their occurence
```



4. Co-existing Mutations

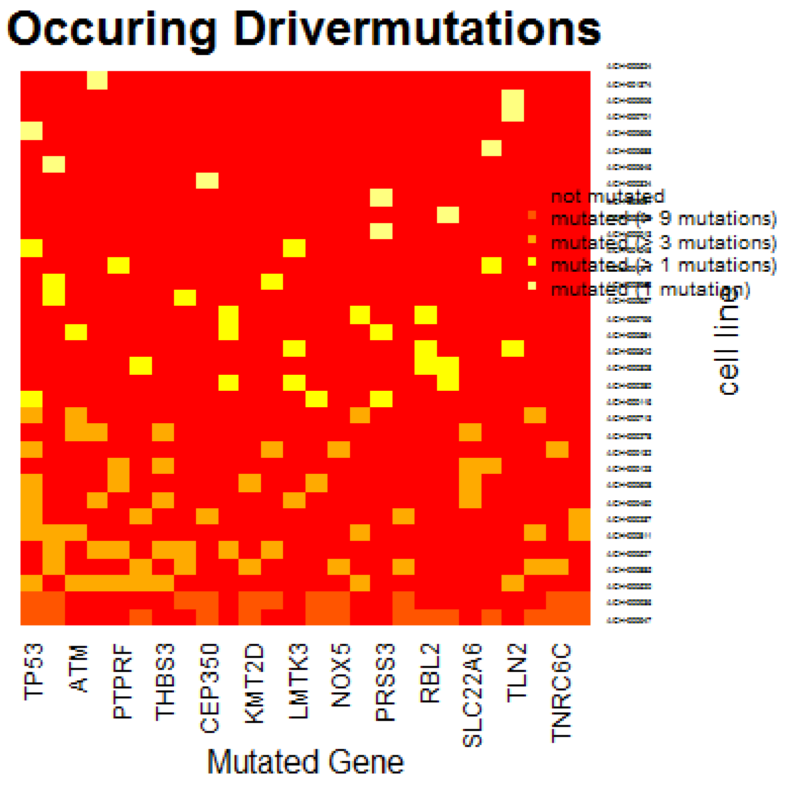
4.1 Investigation of co-existing driver mutations via heatmap

Here, we want to investigate whether there are certain mutations that often occur together and are therefore possibly linked to each other in their genesis. The goal will be to create a symmetrical matrix whose columns AND rows are our genes; the cells' content will be the counts of how often two corresponding genes turn up in the same cell line together. Next, maximum values can be extracted and tested for significance.

To get a short overview about how many co existing drivermutations are common in the data we perform a heatmap:

```r
#creating a new matrix containing every cell line that has at least one of
our 7 drivermutations. We check if the name of our Drivermutation occurs in
the Drivermutation matrix
dm <- new_uniq[which(new_uniq$Hugo_Symbol %in% dm_null$Var1), ]

#function to find out if the Mutation occurs in the cell lines: "1" means yes
and "0" means no
anno <- apply(dm_null, 1, function(x) {annotation[x]<-
ifelse(rownames(annotation) %in% dm[which(dm$Hugo_Symbol %in% x),
]$DepMap_ID, 1, 0)})

#we have to transform the matrix to a data frame for further steps
anno <- as.data.frame(anno)

#we add the names of the cell lines to the matrix
rownames(anno) <- rownames(annotation)

#now we sum up how many TRUE occur in each cell line
anno$summe <- apply(anno, 1, function(x) { sum(x)})

#to get a better overview we use only the drivermutations that occur in more
than 3 cell lines (the other two are needed in step 5)
dm_drei <- dm_null[which(dm_null$Freq > 3), ]
dm_zwei <- dm_null[which(dm_null$Freq > 2), ]
dm_eins <- dm_null[which(dm_null$Freq > 1), ]

#these two are extracted for step 5
anno_zwei <- anno[, which(colnames(anno) %in% rownames(dm_zwei))]
anno_zwei <- as.data.frame(anno_zwei)
anno_eins <- anno[, which(colnames(anno) %in% rownames(dm_eins))]
anno_eins <- as.data.frame(anno_eins)

anno <- anno[, which(colnames(anno) %in% rownames(dm_drei))]
#we have to bring the cell lines in a better order: so we put the cell lines
with the most mutations together and decrease it upwards.
anno$summe <- apply(anno, 1, function(x) { sum(x)})
anno <- anno[order(anno$summe, decreasing = TRUE),]

#we remove columns which should not be plotted
anno <- anno[, -which(colnames(anno) == "summe")]

#and bring the matrix in a format the function "heatmap" can plot
anno <- data.matrix(anno)
```

```r
#we create a color palette containing the colors we want to use
cols <- heat.colors(5)
mypalette <- colorRampPalette(cols)(2)

#we also add a second column to the matrix to include color associated
Explanations for the Legend
co <- as.data.frame(cols)
co$kat <- c("not mutated", "mutated (> 9 mutations)", "mutated (> 3
mutations)", "mutated (> 1 mutations)", "mutated (1 mutation)")

#plotting the heatmap
heatmap(anno, Rowv = NA, Colv = NA, main = "Occuring Drivermutations",xlab =
"Mutated Gene", ylab = "cell line", col = heat.colors(5),  cexCol = 0.9,
cexRow = 0.3)

#adding a legend
legend("topright", pch = 15, col = cols, legend = co$kat, bty = 'n', cex =
0.6)
```



4.2 Driver mutations in literature

4.2.1 Validation of drivermutations in literature

In the heatmap it is visible that many drivermutations, especially the ones that occur less than five times in our cell lines, occur very randomly along the cell lines. Caused by this fact another way to find SL partners is needed.To find out if the main drivermutations is important for a possible SL partner, the main driver muations have to be validated by checking which of them were mentioned in an Ovarian cancer context in literature.

5. Synthetic Lethality Interactions

After identifying our most common driver mutations, this information was used to find possible synthetic lethality interaction partners. First of all, the variance of the estimated cell survival probability (kd.prob) when knocking out specific genes was calculated. It was hereby assumed that genes that display high variances when knocked out are more likely to have specific synthetic lethality interactions since genes that broadly lead to cell death in most cell lines are probably essential for survival on their own. Therefore, the search for synthetic lethality interaction partners will be conducted with genes whose variances are greater than the 75% quantile.

It remains to be found out which difference in survival probability the SL interaction partner for ARID1A mutations mentioned in the literature, BRD2, displays.

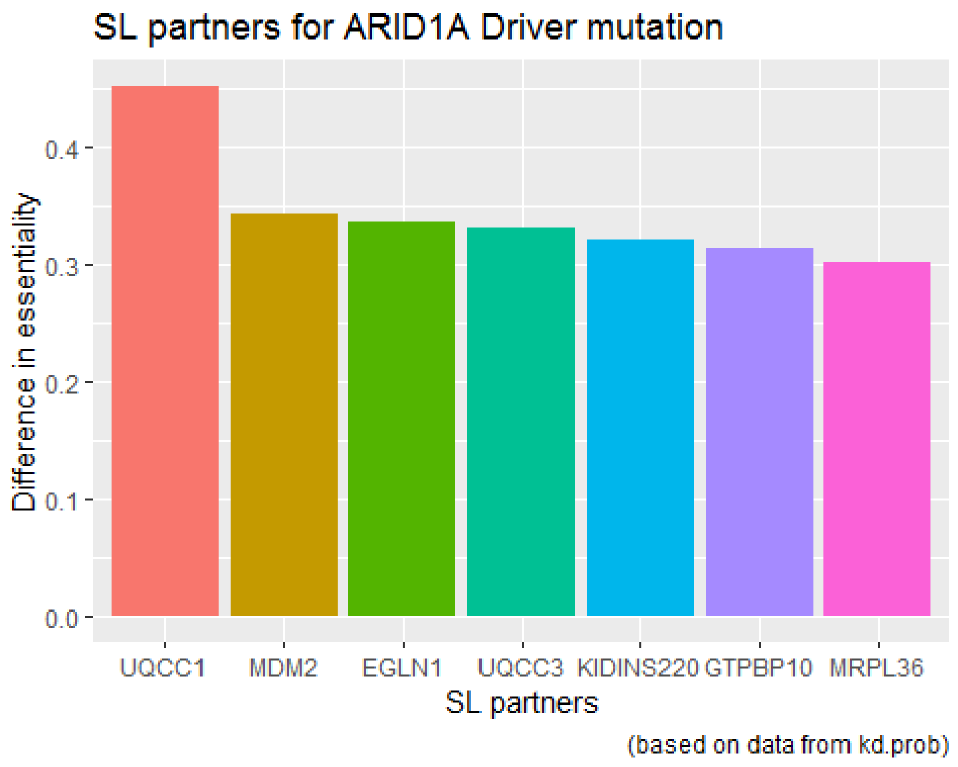
```
dmARID1A = dm[which(dm$Hugo_Symbol == "ARID1A"), ]
nonARID1A <- kd.prob[, -which(colnames(kd.prob) %in% dmARID1A$DepMap_ID)] #
prob Matrix with cell lines that do not have the ARID1A mutation
yesARID1A <- kd.prob[, which(colnames(kd.prob) %in% dmARID1A$DepMap_ID)] #
prob matrix with cell lines that do have driver mutations


no <- nonARID1A[which(row.names(nonARID1A) == "BRD2"), ] # select BRD2 from
submatrix comprising of cell lines without ARID1A mutations
apply(no, 1, mean) # calculate mean of BRD2 survival probability

##     BRD2
## 0.521879

yes <- yesARID1A[which(row.names(yesARID1A) == "BRD2"), ]
apply(yes, 1, mean)

##      BRD2
## 0.6395188

rm(no, yes)
```

The output says that BRD2 is about 10% more essential in ARID1A mutated cells… This seems like a very low difference to be considered a “real” SL partner. Perhaps this is due to the fact that in the literature, BRD2 was mentioned as a SL partner explicitly in clear cell carcinomas. Our data comprises of different types of Ovarian cancer.

An other idea to find SL partners: calculate mean gene essentiality probability for both initially defined groups with/without driver mutations (TP53, ARID1A, ATM, BAI1, PTPRF, SYNE1, THBS3), plot them and then ask for significant differences (with statistical test!).

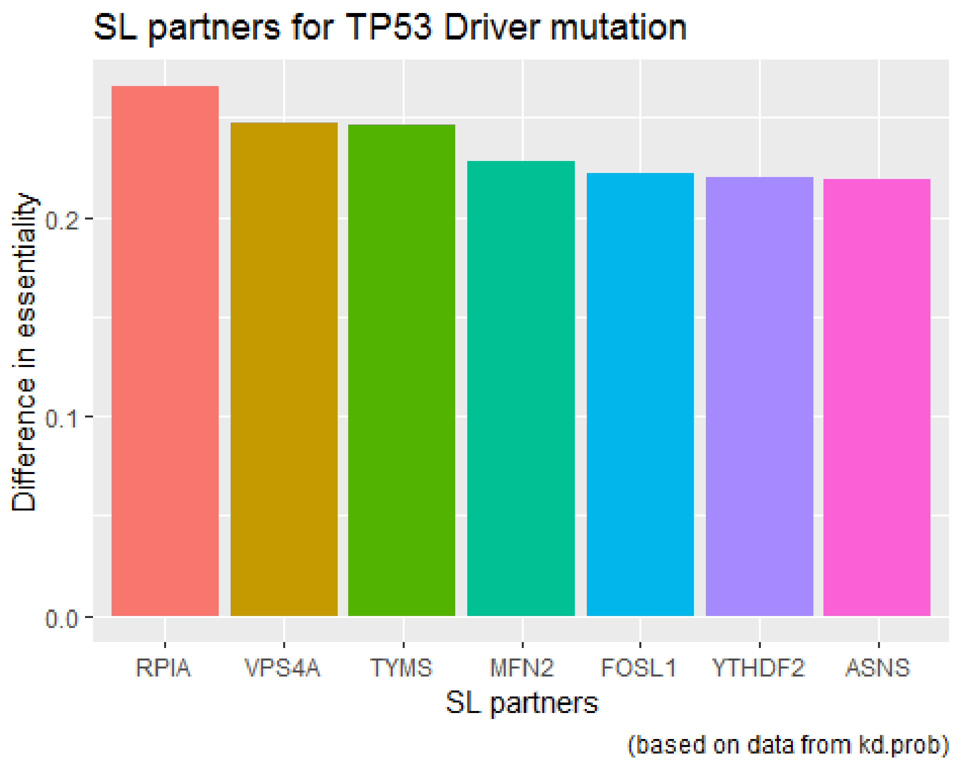
```
dmARID1A = dm[which(dm$Hugo_Symbol == "ARID1A"), ]
nonARID1A <- kd.prob[, -which(colnames(kd.prob) %in% dmARID1A$DepMap_ID)] #
prob Matrix with cell lines that do not have the ARID1A mutation
```

```r
yesARID1A <- kd.prob[, which(colnames(kd.prob) %in% dmARID1A$DepMap_ID)] #
prob matrix with cell lines that do have driver mutation

# now calculate difference between gene essentiality

m.ARID1A <- as.data.frame(apply(yesARID1A, 1, mean))
m.nonARID1A <- as.data.frame(apply(nonARID1A, 1, mean))
diffARID1A <- m.ARID1A - m.nonARID1A # calculate difference in essentiality
diffARID1A$Difference <- diffARID1A$`apply(yesARID1A, 1, mean)` # wie mach
ich das schlauer?
diffARID1A <- diffARID1A[which(diffARID1A$Difference > 0.3), ]


diffARID1A$sub <- with(diffARID1A, reorder(rownames(diffARID1A), -
diffARID1A$Difference, mean, na.rm = T))

ggplot(data = diffARID1A) +
geom_col(mapping = aes(x = diffARID1A$sub, y = diffARID1A$Difference, fill =
diffARID1A$sub), show.legend = F) +
labs(x = "SL partners", y = "Difference in essentiality", title = "SL
partners for ARID1A Driver mutation", caption = "(based on data from
kd.prob)")
```



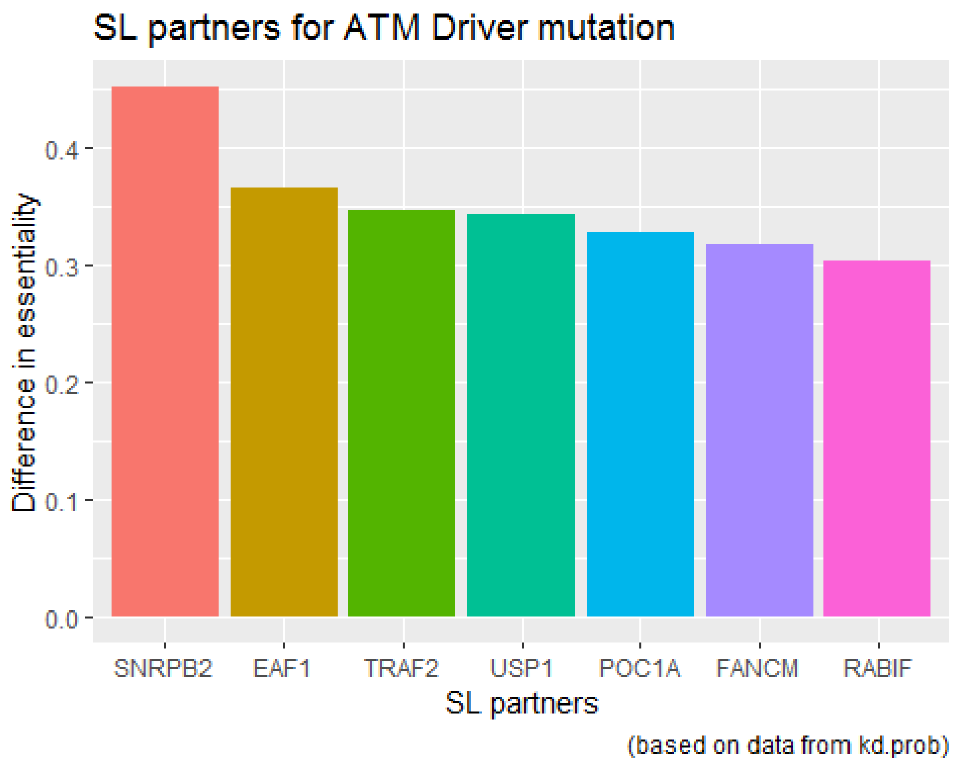
```r
# Muss dann nebeneinander am besten angezeigt werden. Dann diskussion!
```

```r
dmTP53 = dm[which(dm$Hugo_Symbol == "TP53"), ]
nonTP53 <- kd.prob[, -which(colnames(kd.prob) %in% dmTP53$DepMap_ID)] # prob
Matrix with cell lines that do not have the TP53 mutation
yesTP53 <- kd.prob[, which(colnames(kd.prob) %in% dmTP53$DepMap_ID)] # prob
matrix with cell lines that do have driver mutation

m.TP53 <- as.data.frame(apply(yesTP53, 1, mean))
m.nonTP53 <- as.data.frame(apply(nonTP53, 1, mean))
diffTP53 <- m.TP53 - m.nonTP53 # calculate difference in essentiality
diffTP53$Difference <- diffTP53$`apply(yesTP53, 1, mean)`
diffTP53 <- diffTP53[which(diffTP53$Difference > 0.219), ]

# make a barplot for the genes whose knockout leads to biggest difference in
survival probability

diffTP53$sub <- with(diffTP53, reorder(rownames(diffTP53), -
diffTP53$Difference, mean, na.rm = T))

ggplot(data = diffTP53) +
geom_col(mapping = aes(x = diffTP53$sub, y = diffTP53$Difference, fill =
diffTP53$sub), show.legend = F) +
labs(x = "SL partners", y = "Difference in essentiality", title = "SL
partners for TP53 Driver mutation", caption = "(based on data from kd.prob)")
```



```r
dmATM = dm[which(dm$Hugo_Symbol == "ATM"), ]
nonATM <- kd.prob[, -which(colnames(kd.prob) %in% dmATM$DepMap_ID)] # prob
Matrix with cell lines that do not have the ATM mutation
yesATM <- kd.prob[, which(colnames(kd.prob) %in% dmATM$DepMap_ID)] # prob
```

```r
matrix with cell lines that do have driver mutation

m.ATM <- as.data.frame(apply(yesATM, 1, mean))
m.nonATM <- as.data.frame(apply(nonATM, 1, mean))
diffATM <- m.ATM - m.nonATM # calculate difference in essentiality
diffATM$Difference <- diffATM$`apply(yesATM, 1, mean)` # wie mach ich das
schlauer?
diffATM <- diffATM[which(diffATM$Difference > 0.3), ]

# make a barplot for the genes whose knockout leads to biggest difference in
survival probability

diffATM$sub <- with(diffATM, reorder(rownames(diffATM), -diffATM$Difference,
mean, na.rm = T))

ggplot(data = diffATM) +
geom_col(mapping = aes(x = diffATM$sub, y = diffATM$Difference, fill =
diffATM$sub), show.legend = F) +
labs(x = "SL partners", y = "Difference in essentiality", title = "SL
partners for ATM Driver mutation", caption = "(based on data from kd.prob)")
```



Now: test significance of obtained values with a statistical test (which one?)

5.2 Performing Principal Component Analysis

5.2.1 Creating groups with similar drivermutations

In step 4.2.1 the drivermutations were validated with literature. And the ones which were common are: TP53, ARID1A, ATM and PTPRF. In the next step the cell lines are investigated if these Mutations occur in them and which groups can be defined:

```
#to investigate which cell lines contain which drivermutation we create a new
data frame containing only the colums about our four Drivermutations
annodm <- as.data.frame(anno[, which(colnames(annotation) %in% c("nix"))])



#to seperate the cell lines in groups with similar Mutations we give every
Drivermutation a specific value: If TP53 is True in the cell line it gets a
"1000" , if it isnt ther will be a "0"
annodm$TP53 <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo_Symbol %in%
"TP53"), ]$DepMap_ID, 1000, 0)


#same here, ARID1A gets a "100" for True and a "0" for FALSE
annodm$ARID1A <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo_Symbol
%in% "ARID1A"), ]$DepMap_ID, 100, 0)

#and the same for the next Mutations
annodm$ATM <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo_Symbol %in%
"ATM"), ]$DepMap_ID, 10, 0)
annodm$PTPRF <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo_Symbol %in%
"PTPRF"), ]$DepMap_ID, 1, 0)


#we can now sum up these values in a new column
annodm$summe <- apply(annodm, 1, function(x) { sum(x)})
```

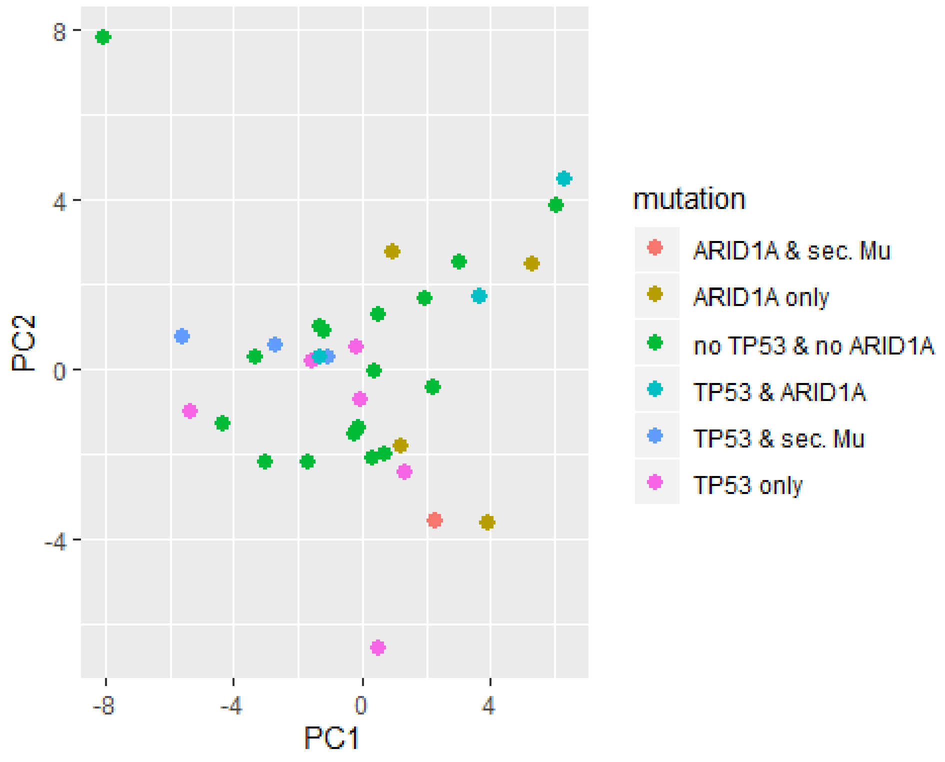
Many cell lines do not even have one of our chosen mutations because their sum is zero. At many others we can see that they have the same combination of Drivermutations because they have the same value at “annodm$summe”. In the following step we gave the different combinations a specific category

```
#we picked the most common combinations and gave them different categories by
their sums
annodm$kat <- ifelse(annodm$summe == 1000, "TP53 only", ifelse(annodm$summe >
1099, "TP53 & ARID1A", ifelse(annodm$summe == 100, "ARID1A only",
ifelse(annodm$summe == 110, "ARID1A & sec. Mu", ifelse(annodm$summe == 101,
"ARID1A & sec. Mu", ifelse(annodm$summe < 100, "no TP53 & no ARID1A",
ifelse(annodm$summe == 0, "no TP53 & no ARID1A", "TP53 & sec. Mu")))))))
```

5.2.2 PCA combined with cluster after drivermutations

To reduce the information of the kd.prob matrix we perform a PCA. The analysis creates groups of cell lines with similar lethality partners.

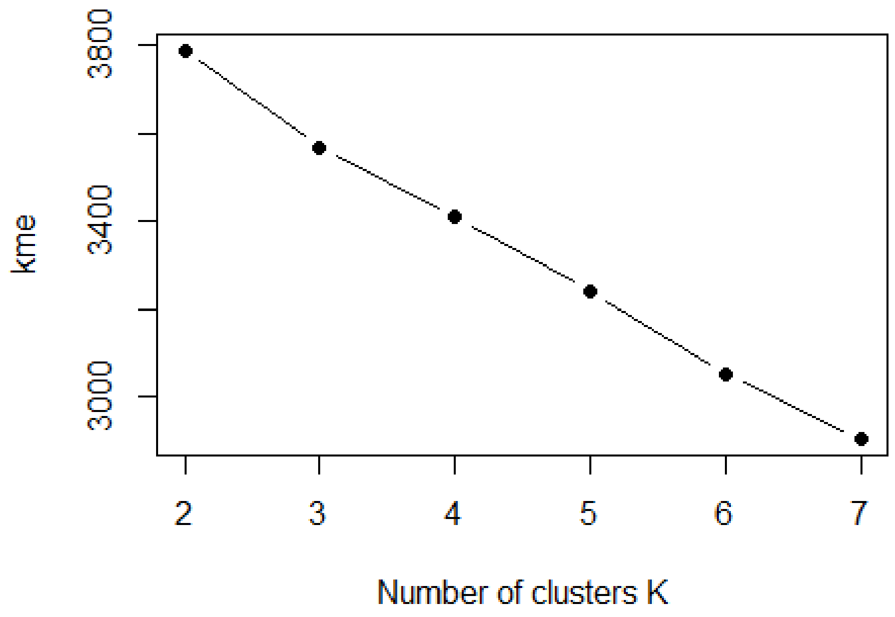
```
# calculate variance over all rows (genes)
topVar = apply(kd.prob, 1, var)

# new data frame with just the genes whose knockout leads to highly variant
cell death events -> perhaps our synthetic lethality interaction partners?
kd.prob.topVar = kd.prob[topVar > quantile(topVar, probs = 0.75), ]

#we have to transpose the data frame
kd.prob.top <- t(kd.prob.topVar)

#we use the colors defined in step 4 for our co existing mutations
mutation <- annodm$kat

#this function calculates the PCA with the variance calculated before
data<-kd.prob.top
df<-kd.prob.top
PC<-prcomp(df)
PCi<-data.frame(PC$x)

#we plot the pca with the different colors for cell lines with similar Driver
Mutations
ggplot(PCi,aes(x=PC1,y=PC2,col=mutation))+geom_point(size=2.5)
```



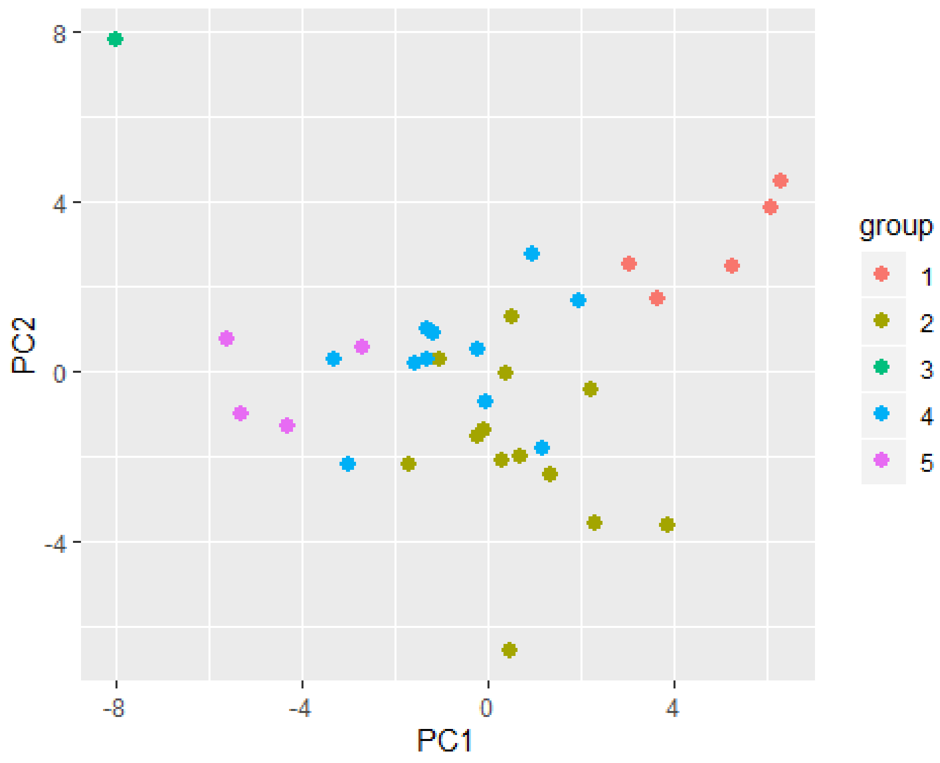
5.2.3 PCA combined with kmeans cluster

We can see that the cluster does not really fit to the results of the PCA. In the next step, we perform a kmeans cluster and check if that fits better

```
#at first we need to find out how many centers we use for the clustering
kme = sapply(2:7, function(k) {
    kmeans(kd.prob.top, centers = k)$tot.withinss
})
plot(2:7, kme, type = "b", pch = 19, xlab = "Number of clusters K")
```



```
#the elbow plot does not give clear information about the ideal number of
clusters, but we will use five centers (silhouette plot is in progress)

#performing clustering
km <- kmeans(kd.prob.top, 5)

#extracting the colors
group <- as.factor(km$cluster)


#plotting PCA as before
ggplot(PCi,aes(x=PC1,y=PC2,col=group))+geom_point(size=2.5)
```



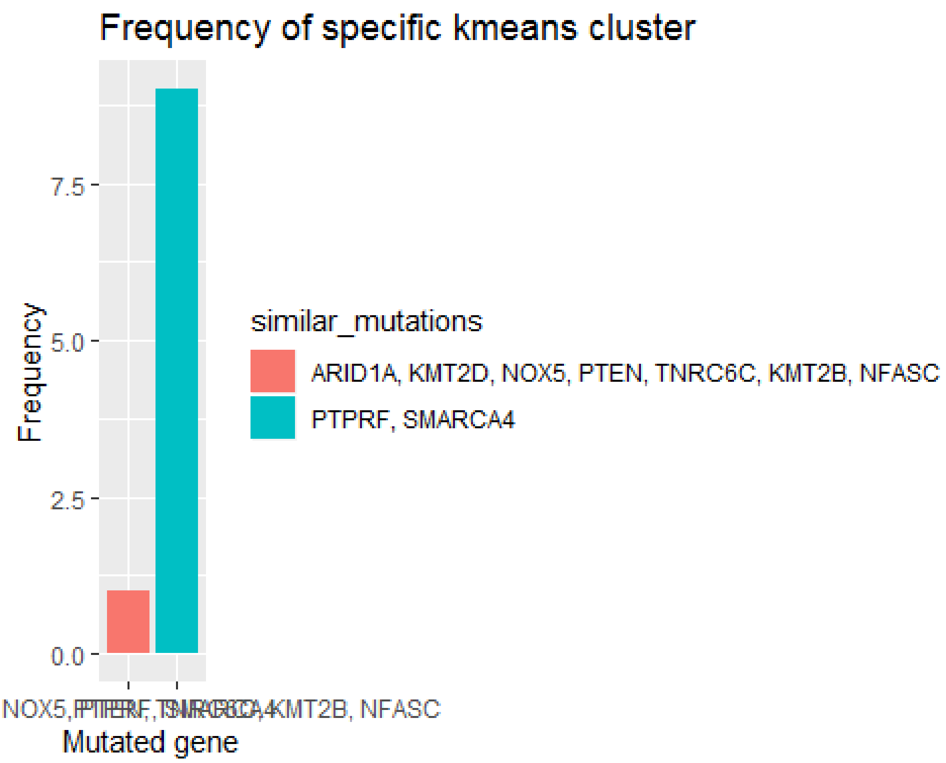
5.2.4 PCA results

clustered by kmeans there are five groups which fit to the pca very well. In the next step it is investigated which cell lines are grouped by kmeans and what they have in common. But there comes up a problem: every time our code runs, the function kmeans will create a new cluster. Caused by that it in impossible to interpret exacly the groups which were formed in the chunk before. To get a good interpretation of the kmeans data a loop is created which forms a new kmeans cluster every single round that will be analysed.

```
#a data frame is created to print the results in
po <- as.data.frame(1)

#the loop will create 10000 kmeans cluster
for (j in 0:300) {   #!!!!!!! hier bevor ihrs abgebt 10000 eintragen, dauert
dann nur bissl lÃ¤nger zu knitten!!!!!

  #a kmeans cluster is formed
  km <- kmeans(kd.prob.top, 5)

  #the number of the cluster is extracted (1 to 5)
  kem <- (km[["cluster"]])
  kem <- as.data.frame(kem)

  #we add the rownames from the annotation matrix to validate the cell lines
have the right assignment
  kem$id <- rownames(annotation)
```

```r
  #we need another loop to check each of the five clusters
  for(i in 1:5) {

    #at first the variable has to be loaded with the Drivermutaions that
occur two or more times in the cell lines
    kp <- anno_eins[which(rownames(anno_eins) %in% kem[which(kem$kem %in%
i),]$id),];

    #we transpose kp for a easier workflow and transform it to a data frame
    kp <- t(kp);
    kp <- as.data.frame(kp);

    #and we count the number how often the mutation occurs in that group
    kp$summe <- apply(kp, 1, function(x) { sum(x)});

    #now we delete genes that are mutated in less than 67% of the cell lines
    kp <- kp[which(kp$summe > 0.666*NCOL(kp)),];

    #we remove the column "summe""
    kp <- kp[, -which(colnames(kp) %in% "summe")]

    #at this point the data is extracted from the loop into the data frame
created before

    #the following function counts the number of the round from 1 to 10000
    h <- j*5+i

    #we don't need groups that contain only one cell line because the
mutations would not be significant
    if (NCOL(kp) > 1) {

      #groups with only one or without any gene in common aren't usefull as
well
      if (NROW(kp) > 1) {

    #the data is transformed to a string and written into the data frame
    po[h] <- toString(rownames(kp))
      }
    #the next two step have to be done otherwise there would be empty colums
in between and an error comes up
      else {po[h] <- "nein"}
    }
    else {po[h] <- "nein"}
  }
}

#the data frame is transposed to get a better overview and to perform further
analysis
po <- t(po)
```

```r
#the combinations are counted and converted to a data frame
ko <- table(po)
ko <- as.data.frame(ko)

#they are order after the most frequency
ko <- ko[order(ko$Freq, decreasing = T),]

#and the "nein" row is deleted
ko <- ko[-which(ko$po %in% "nein"),]

#a barplot is created to visualize the data
similar_mutations <- ko$po
ggplot(data = ko) +
geom_col(mapping = aes(x = ko$po, y = ko$Freq, fill = similar_mutations),
show.legend = T) +
labs(x = "Mutated gene", y = "Frequency", title = "Frequency of specific
kmeans cluster")
```



```r
#we add the column subtype desease to the matrix
kem$sd <- annotation$Subtype.Disease
```

There is a group that comes up very often which contains two cell lines (ACH-001278 and ACH-000123). These cell lines have two Mutations in common: “PTPRF”" and “SMARCA4”. With the number of the cluster it is possible to check the cell lines in the “kem” matrix. At the two examples the same Subtype desease cancer (Small Cell Carcinoma of the Ovary Hypercalcemic Type (SCCOHT)) can be seen. These two cell lines may have the same SL partners.

6. Linear Regression

Independent of the previous analysis, this step will focus on the relationship between the gene expression and the copynumber. This approach will be performed with the totality of the given genes. Blub Literatur noch nutzen To investigate this issue a linear regression model will be applied to the question: If it is possible to estimate the gene expression based on the given value for the copy number of the certain gene.

6.1 Checking the distribution

The data frames expression and copy number are bound to the list list_all.genes. This process ensure that the primary dataset is not changed.

```
list_all.genes <- list(expression,copynumber) # bind the two data frames in
one list
names(list_all.genes) <- c("expression","copynumber") # rename the elements
of the list
```

Before using the copy number as predictor for the regression model, certain criteria have to be checked.As a requirement for the model the predictor variable should show a normal distribution and only a low number of outliers is preferred. First all the copy number values of the different cell lines are fused to one long column. The new matrices cn1 only contains only two columns.The copy number data indicates a high deviation for vales smaller than -1 in comparison to the gaussian distribution.

```
cn1b <- melt.data.frame(list_all.genes$copynumber, variable_name =
"cellline") # fuse all cell lines

## Using  as id variables
```

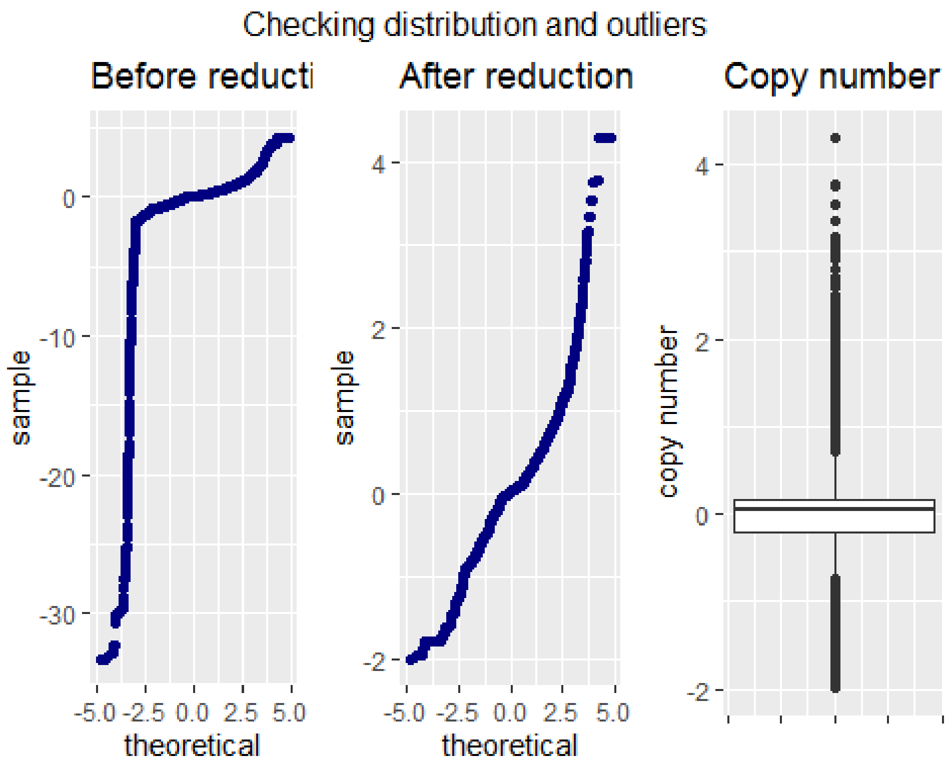
The cn1b data indicates a high number of values smaller than -1, which lead to a distribution with a right skew. To reduce the influence of this circumstance, all rows that obtain a value smaller than -2 or equal -2 are removed of the data set. This process leads to a limitation of the predictor variable of the regression models. Predictions will only be possible for genes with a copy number higher than -2.

```
rmv.rows = apply(list_all.genes$copynumber, 1, function(x) {sum(x <= -2)}) #
find all genes with at least one values <= -2
list_all.genes$copynumber <- list_all.genes$copynumber[-which(rmv.rows > 0),
] # remove all rows, those contain more than one value <= -2
```

After the reduction process the copy number values are fused to one column once again with the melt function. A Q-Q-plot is peformed to check the normality of the copy number data set once again and determine if the reduction process shows an improved result for the distribution.

```
cn1 <- melt.data.frame(list_all.genes$copynumber, variable_name = "cellline")
# fuse all cell lines to one long column

## Using  as id variables
```

```
plotcnb <- ggplot() + geom_qq(aes(sample = cn1b$value), color= "navy blue") +
ggtitle("Before reduction") # plot the quantils of the copynumber with the
quantils of a gaussian distribution (before reduction process)

plotcn <- ggplot() + geom_qq(aes(sample = cn1$value), color= "navy blue") +
ggtitle("After reduction") # plot the quantils of the copynumber with the
quantils of a gaussian distribution (after reduction process)

boxplotcn <- qplot(y=cn1$value, x= 1, geom = "boxplot")+ ggtitle("Copy
number") + xlab("") + ylab("copy number") +
theme(axis.text.x=element_blank()) # create boxplot + name the graph + define
axis + remove the scaling of the x-axis

grid.arrange(plotcnb, plotcn,boxplotcn, ncol=3, top = "Checking distribution
and outliers") #arrange the to plots next to each other
```



In general a Q-Q-plot compares the quantiles of a theoretical normal distribution (x-axis) with the distribution of a sample data set (y-axis) and can therefore be used to check normality of the expression and copy number data. Before the reduction process the Q-Q-plot (left) shows a higher number of values smaller than -1, which leads to a right skew of the distribution of data set. After the reduction process the Q-Q-plot (middle) reveals a better fit for normality, although for copy numbers higher than one a strong aberration is revealed. The boxplot (right) illustrate the issue of a high number of outliers for each side of the 1,5 IQR (Interquantile range).Having a large number of outliners in the predictor variable can effect the slope for the regression model and may lead to a low accuracy of the predictions based on the model. Also the anomaly of the samples distribution can effect the output of the model. This circumstances has to be considered as a possible error for the regression model.

6.2 Identify common genes of both variables

Only genes,those are present in both dataframes, can be included for the linear regression model.For each copy number value a corresponding expression value must exists, which can later be plotted in a univariate linear regression model.

```
common_names = Reduce(intersect, lapply(list_all.genes, row.names)) #
identify shared rownames in both data frames
list_all.genes <- lapply(list_all.genes, function(x) {x[row.names(x) %in%
common_names,]}) # only keep the common rownames in the list
```

After both matrices only consists of shared genes, the gene expression values and copy number values are once again merged to one long column and are combined in a new object

```
exp1 <- melt.data.frame(list_all.genes$expression, variable_name = "cell
line") #fuse all expression values to one long column

## Using  as id variables

cn1 <- melt.data.frame(list_all.genes$copynumber, variable_name = "cellline")
# fuse all cell lines

## Using  as id variables

regression.all.genes <- as.data.frame(cbind(exp1$value, cn1$value)) # merge
the colums

colnames(regression.all.genes) <- c("expression", "copynumber") # rename the
colums
```

In the object regression.all.genes the copy number value in row one correspond to the gene expression value in row one. This circumstance leads to the possibility of plotting the corresponding values of the two variables in the next step.

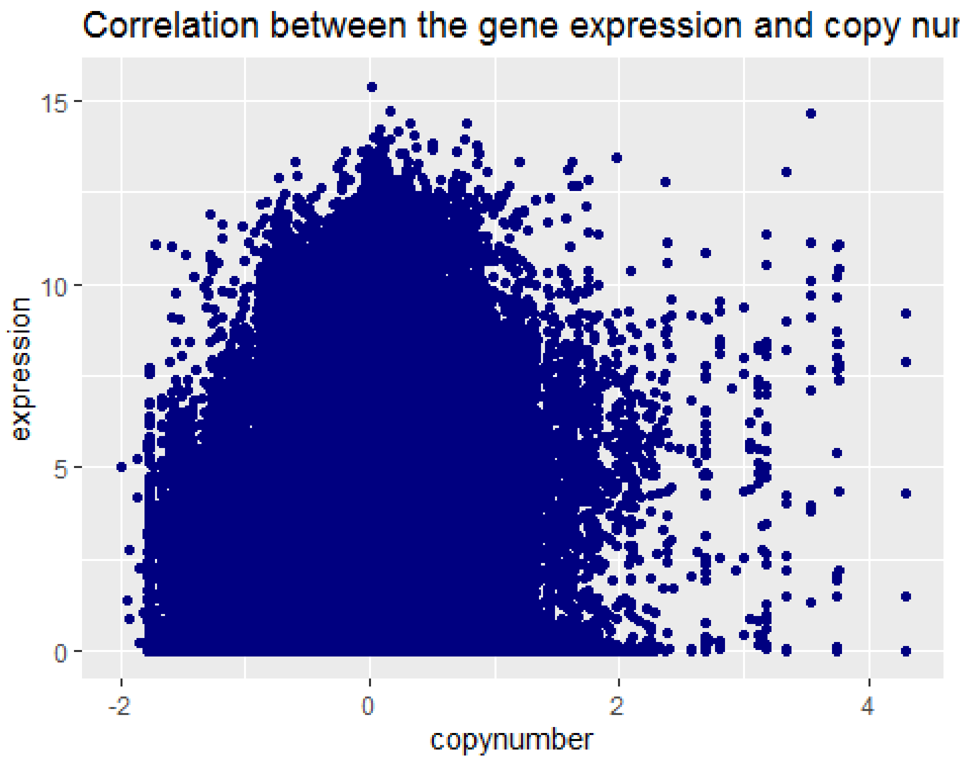
6.3 Correlation between the two variables

In this step the correlation between the gene expression and copy number was tested, to decide wether a linear regression model can be applied for the problem. First. the gene expression values and copy number values are fused a single data frame.

```
regression.all.genes <- as.data.frame(cbind(exp1$value, cn1$value)) # merge
the colums
colnames(regression.all.genes) <- c("expression", "copynumber") # rename the
colums
```

To obtain a visual overview the correlation between the two variables are illustrated in a scatter plot.

```
ggplot(regression.all.genes, aes(x = copynumber, y = expression)) +
geom_point(color = "navy blue")+ labs(title = "Correlation between the gene
expression and copy number") # create scatter plot for expression and copy
number + include the data points in blue + ingtegrate title
```



The scatter plot reveals only a low visual relationship between the two variables and do not indicated a linear trendline between the gene expression and the copy number. To analyse the correlation futhermore, the person and spearman correlation coefficient is computed. The advantage of the spearman correlation is, that the values are transferred into rankings before performing the calculation of the correlation coefficient. This process decreases the effect of outliners for the coefficient.

```
cor(regression.all.genes$expression, regression.all.genes$copynumber) #
compute person correlation

## [1] 0.09071741

cor(regression.all.genes$expression, regression.all.genes$copynumber, method
= "spearman") # compute spearman correlation coefficient

## [1] 0.07229923
```

Both correlation coefficients show a low correlation between the two varibles of approximately 0.14 to 0.13. The sligthly lower spearman correlation concludes, that the person correlation is lightly influenced by outliners. To verify the correlation a t-test for both correlation coefficient is performed. The null hypothesis indicates that there is no correlation between the copy number and the gene expression, while the alternativ hypothesis confirms a correlation between the two variables. As a common level of significance = 0,05 is choosen and a p-value, which is smaller than the level of significance, will lead to a denial of the null hypothesis. First the the person correlation coefficient is verified

```r
cor.test(regression.all.genes$expression, regression.all.genes$copynumber)
```

```
## 
##  Pearson's product-moment correlation
## 
## data:  regression.all.genes$expression and regression.all.genes$copynumber
## t = 74.176, df = 663066, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.08832973 0.09310404
## sample estimates:
##        cor 
## 0.09071741
```

The p-value shows the possibility that an equal or higher t-value is observed under the null hypothesis. With a p-value < 2.2e-16 the null hypothesis can be rejected, becaue the p-value shows smaller value than the level of significance and therefore it is unlikely to observe the calculated correlation coefficient without a correlation between the two variables. Only in the case of a p-value that is higher than the level of significance the null hypothesis would be retained. The same analysis is performed for the spearman correlation coefficient.

```r
cor.test(regression.all.genes$expression, regression.all.genes$copynumber,
method = "spearman")
```

```
## 
##  Spearman's rank correlation rho
## 
## data:  regression.all.genes$expression and regression.all.genes$copynumber
## S = 4.5074e+16, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##        rho 
## 0.07229923
```

The test also shows a p-value < 2.2e-16, concluding the rejection of the null hypothesis. Both test lead to the conclusion, that the variables expression and copynumber have a weak correlation and therefore can be applied for a linear regression model.

To ensure a clean workspace the no longer needed data frames are removed

```r
remove(exp1, cn1, cn1b, boxplotcn, plotcn, plotcnb, common_names, rmv.rows) #
remove objects
```

6.5 Univariate linear regression

Before using the regression.all.genes data for a regression model, a small amount of values have to be seperated for training purposes. 200 rows are taken randomly of the regression.all.genes matrix and will later be used as trainings data for the model.

```
testing.all.genes <-
regression.all.genes[sample(1:nrow(regression.all.genes), 200),] # new
dataframe with 200 randomly selected rows of dataframe regression.all.genes
```

Next the 200 rows are removed from the original to exclude them from the regression model.

```
regression.all.genes <- regression.all.genes[ -
sample(1:nrow(regression.all.genes), 200),] # remove the testing data of the
primary data set
```

After the reduction process the data is integrated in the regression model lrm.all.genes. The variable copynumber is used as a predictor for the variable gene expression.

```
lrm.all.genes <- lm(expression ~ copynumber, data = regression.all.genes)
#establish univariate linear regression model
summary(lrm.all.genes) # show regression model

##
## Call:
## lm(formula = expression ~ copynumber, data = regression.all.genes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0012 -2.3566 -0.3796  1.8692 12.8403
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.529057   0.003024  836.22   <2e-16 ***
## copynumber  0.575544   0.007762   74.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.462 on 662866 degrees of freedom
## Multiple R-squared:  0.008227,   Adjusted R-squared:  0.008226
## F-statistic:  5499 on 1 and 662866 DF,  p-value: < 2.2e-16
```

With the calculated intercept of 3.874921 and a slope of 0,780919 the equation for the regression is:

gene expression = 3,874921 + 0,780919*copynumber

With a p-value smaller that 2e-16 the t-test indicates that the slope of the regression line has a significant aberration from zero. Even with a level of significance of 0,01 (one per mill) the regression line shows a aberration, which leads to the denial of the null-hypothesis.

The standard error for the residuals indicates an average aberration of 2,147 between the predicted expression value and the expression value of the dataset. For a good fit of the model a error equal to zero is preferred.

The R-squared statistic shows the proportion of variance and measures
the linear relationship between the predictor variable (copynumber) and the response variable (expression). While a value near 1 would indicated a good fit of the model, the regression model for all genes has a value of 0.01951 –> near 0 represents a regression that does not explain the variance in the response variable well. (Adjusted R-squared: 0.01951 –> nur fÃ�Â¼r multiple regression relevant)

For a good fit the residuals have to be distributed symmetric around a median of approximately 0. With a calculated value of 0.0177 the median is just above zero. Futhermore the first and third quantil indicate a similar value with reverse algebraic sign. The interquantile range is 2,8359

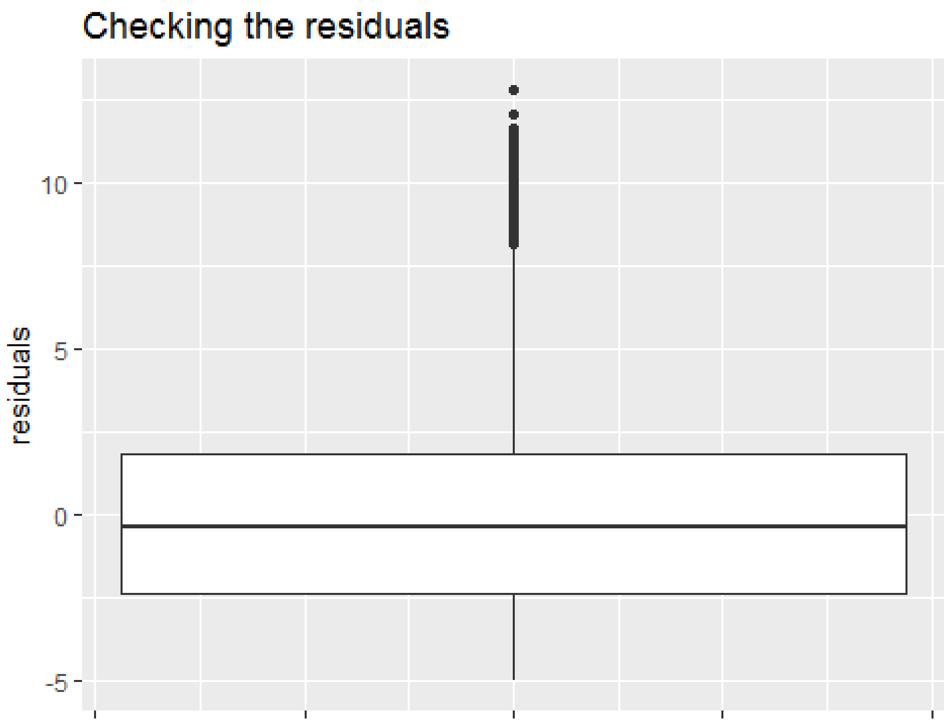
IQR = Q3-Q1 = 1,3571-(-1,4788) = 2,8359

Both extrema show a higher value than the IQR (min = 2IQR and max = 4IQR). To facilitate the futher analysis of the residuals, they are saved in the new object Resid

```
Resid <- as.data.frame(lrm.all.genes$residuals) # single data frame for the
residuals
names(Resid) <- c("residuals") # rename the column
```

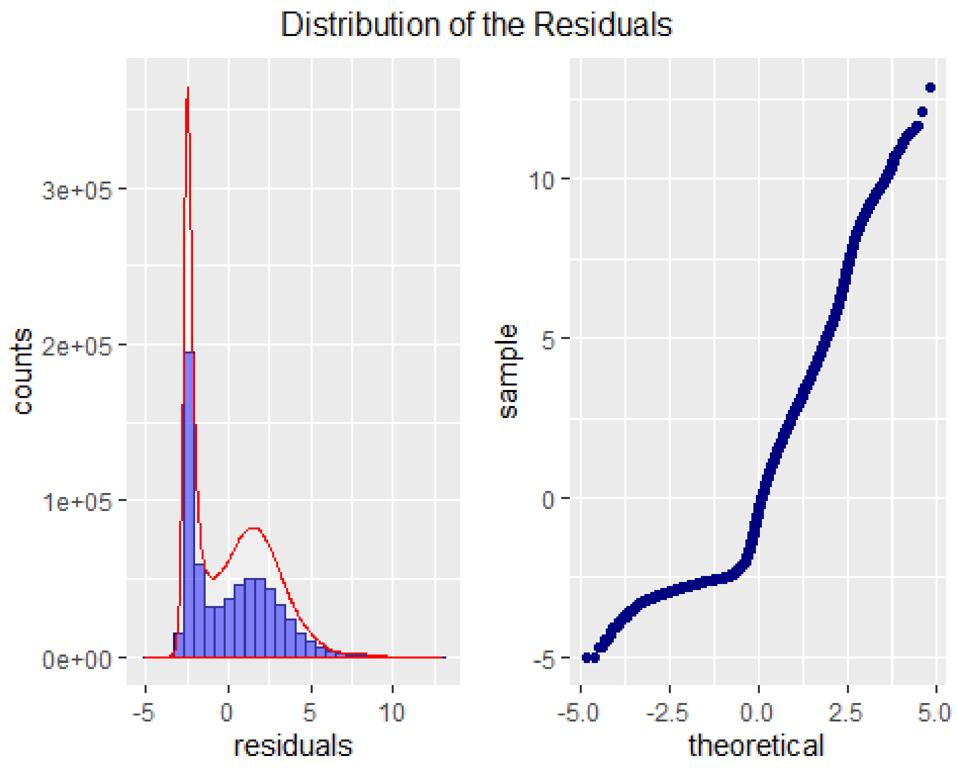
To check the circumstances that the minimum and maximum show a high aberration of the data points of the IQR a boxplot is applied

```
qplot(y=Resid$residuals, x= 1, geom = "boxplot") + ggtitle("Checking the
residuals") + xlab("") + ylab("residuals") +
theme(axis.text.x=element_blank()) # boxplot for residuals + title + name
axis + remove the scaling for the x-axies
```



While only one value lies beneath 1,5 IQR, over the 1,5 IQR a high number of outliers is indicated. This outliers will probably lead to a higher varation between the training values and the estimated values in a later step. Another requirement for the residuals is normality, which is checked with a histogramm and a Q-Q-plot

```
hist.re <-ggplot(data = Resid, aes(Resid$residuals)) + geom_histogram(bins =
30, col = "navy blue", fill="blue", alpha= .6) + labs( x="residuals",
y="counts") + stat_density(aes(y=..count..), col="red", alpha=.2, fill=
"white") # plot histogram for residuals(alpha = transparency of the plot) +
name axis + lay density curve over histogram

qqplot.re <- ggplot() + geom_qq(aes(sample = Resid$residuals), color= "navy
blue") # plot quantiles of the residuals against quantiles of gaussian
distribution

grid.arrange(hist.re, qqplot.re, ncol=2, top= "Distribution of the
Residuals") # show both plots side by side
```



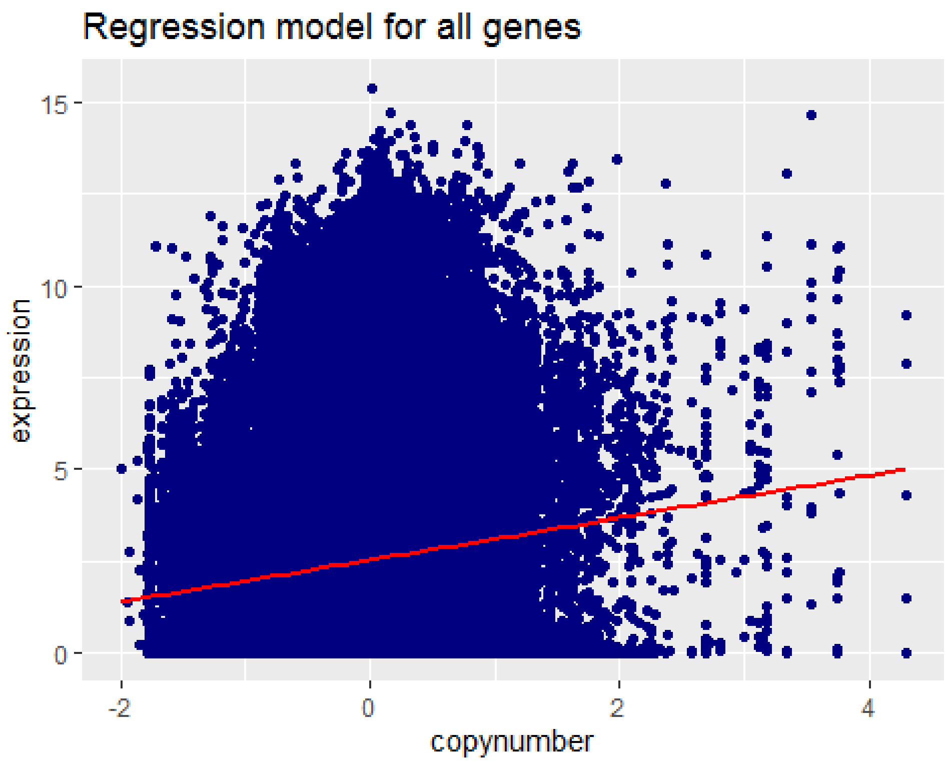
The histogram of the residuals show a good fit for normality, but a higher number of counts is shown for the lower tail. The quantile-quantile-plot also reveals a gaussian distribution with a right skew. Next the correlation between the predictor varibale (copy number) and the residuals is checked

```
cor(regression.all.genes$copynumber ,lrm.all.genes$residuals)

## [1] 2.100256e-16

cor.test(regression.all.genes$copynumber, lrm.all.genes$residuals)
```

```
##
##  Pearson's product-moment correlation
##
## data:  regression.all.genes$copynumber and lrm.all.genes$residuals
## t = 1.71e-13, df = 662866, p-value = 1
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.002407325  0.002407325
## sample estimates:
##          cor
## 2.100256e-16
```

The correlation between the two variables is equal to zero and therefore indicate no correlation between the copy number and the residuals. The t-Test with a p-value of 1 leads to the acceptance of the null hypothesis. After all requierments are checked and the regression model is established a scatter plot with the regression line is applied to gain a visual overview.

```
ggplot(regression.all.genes, aes(x=copynumber, y=expression)) +
geom_point(color = "navy blue")+ geom_smooth(method=lm, se=FALSE, color =
"red") + ggtitle("Regression model for all genes") # scatter plot for
expression and copy number + insert regression line + title
```



The scatter plot shows a high aberration of the predicted values for the gene expression from the regression line.

6.6 Testing the model

To check if the model can be applied to the issue, the testing data is used to examine the aberration between the predicted expression and the real expression of values, that were not used for the linear regression model. First the predicted values for gene expression are calculated base on the copy number values for the testing dataset.

```
prediction <- predict(lrm.all.genes, newdata = testing.all.genes) # calculate
the predicted expression values based on the copy number
```

Next the standard error of the residuals is caculated for difference between the predicted values and the real values of the testing data set. For the calculation of the error the following formula was used

–> insert formula

In the formula n displays the number of rows in the given data frame. To calculate the residuals the difference between the real value and the predicted value of the gene expression is used.

```
sqrt(1/nrow(testing.all.genes) * sum((testing.all.genes$expression-
prediction)^2)) # compute standard error for the 200 expression values

## [1] 2.591774
```

The value of 2.222435 is slightly higher than the residual standard error for the regression model, due to circumstance that the testing data did not account to the regression model.

6.7 Discussion

Although the results of the linear regression model are somewhat sobering, this is not very surprising from the biological point of view. Initially, it was our goal to predict the gene expression by looking at the gene copy numbers, since it is known that many upregulations in cancer such as VEGF are due to gene amplifications.
However, gene expression is a very strictly regulated process which is influenced by many different factors. The numerous steps in the pathway from DNA to RNA to protein comprise of
* transcriptional control * RNA processing * RNA transport and localization control * protein activity control.
Since the expression data from our dataset gives information about the quantity of RNA transcripts, the according control feature is transcriptional control, possibly also RNA processing and transport. Factors that influence the gene expression on DNA level are:
* Promotors, which can be active/inactive and whose affinity to transcription initiation factors can vary from gene to gene * Transcription factors and regulators: These proteins specifically bind the DNA with their amino acid residues and make it accessible or inaccessible for the transcription machinery. * Epigenetics: + DNA-methylation: Methylation of CpG nucleotides which are frequent in so-called CpG islands situated in genes' promotor regions leads to a transcriptional blockade in most cases (depending on binding characteristics of transcription regulator proteins). + Histone-methylation and acetylation: Chemical modification of the proteins that pack the DNA has a huge impact on

binding of transcription machinery proteins and transcription factors. * Non-coding RNA: There are approximately 9000 genes coding for RNA molecules that are not translated into proteins, but have regulatory functions. They often form complexes with other proteins and process or degrade other RNA transcripts. Other non-coding RNAs directly bind the DNA double helix and influence DNA conformation or the accessibility and binding affinity for other proteins such as transcription factors or RNA polymerases.
In each cell, a complex interplay of all these regulations takes place. It is therefore impossible to predict gene expression alone by knowing the gene copy number - to obtain a more accurate model, we would need access to further information concerning the mentioned regulatory elements and processes.

7. Conclusion