Project 3: Ovarian Cancer

## Contents

1. Project Overview
2. Basic Requirements
3. Driver Mutations
4. Co-existing Mutations
5. Synthetic Lethality Interactions
6. Linear Regression
7. Conclusion
8. Sources

## 1. Project Overview

### 1.1 General facts about Ovarian Cancer

Ovarian cancer (OC) is 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 Ovarian cancer is well differentiated and therefore not invasive; high grade 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

Occurence and frequency of certain mutations in Ovarian cancer largely depends on cancer type and whether it can be classified as hereditary or not. Literature on hereditary Ovarian cancer states that the most common mutations are found in the DNA repair-related genes BRCA1 and BRCA2. Abnormalities in other regulatory and tumor suppressor genes have been found too, such as TP53, which is also known as “guardian of the genome” and is responsible for induction of apoptosis when substantial DNA damage occurs. TP53 loss of function mutations are known to majorly contribute to cancer development due to evasion of cell death. According to Testa et al., TP53 accounts for most mutations found in high grade serous Ovarian cancer. Allegedly, another very common mutation is ARID1A. The gene ARID1A encodes a protein involved with epigenetic regulation: as part of a chromatin remodeling complex, it influences the expression of CDKN1A, SMAD3, MLH1 and PIK3IP1 (Takeda et al., 2016). Deleterious mutations of ARID1A lead to dysfunctional chromatin remodeling, which is associated with cancerogenesis and cell transformation. Furthermore, genes involved in DNA repair (CHEK2, RAD51, BRIP1, PALB2) and mismatch repair (MSH2, MSH6, MLH1, PMS2) are frequently mutated in Ovarian cancer . Other frequently defective pathways are those involved in cell survival, proliferation and growth; mutations often occur in NOTCH, RAS/MEK, PI3K, FOXM1, BRAF and RAS genes (Testa et al., 2018).

### 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. Basic Requirements

### 2.1 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")

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

install.packages("ggplot2", repos = "http://cran.us.r-project.org")

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

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

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

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

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

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

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

#install.packages("dplyr", repos = "http://cran.us.r-project.org")  
  
  
library(reshape)  
library(ggplot2)  
library(data.table)

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

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

library(gridExtra)  
library(grid)  
#library(dplyr)

### 2.2 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"]]

The whole dataset consists of a total of 512 cell lines, all of which are derived from a specific cancer type. The Ovarian Cancer cell lines will be extracted, since the remaining cell lines do not pose relevance for our project.

# Extract Ovarian Cancer cell lines  
annotation = annotation[which(annotation$Primary.Disease == "Ovarian Cancer"), ]  
  
# Define ID as vector containing all cell line names  
ID = annotation$DepMap\_ID  
  
# Expression, copy number, kd.ceres, kd.prob and mutation matrices now only contain OC cell lines:  
expression = expression[ , which(colnames(expression) %in% ID)]  
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.

# Order the rownames alphabetically   
copynumber <- copynumber[order(rownames(copynumber)),]  
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.

# Make rownames = genes  
rownames(annotation) = annotation$DepMap\_ID  
  
# Removal of unecessary columns  
annotation = annotation[, -which(colnames(annotation) %in% c("DepMap\_ID", "Aliases", "Primary.Disease", "Gender", "Source"))]  
  
# Removal of initial dataset, since relevant information has been extracted  
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.

# do not show output  
  
NAV = apply(copynumber, 1, function(x) {sum(is.na(x))})   
copynumber = copynumber[-which(NAV > 0), ]

## 3. Driver Mutations

To find out the main driver mutations in the Ovarian Cancer dataset, some re-structuring is required.

### 3.1 Re-structuring of mutation matrices

To look at overall frequency of mutations among all cell lines, the existing mutation lists are fused to one matrix.

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

For the following analysis, just the information in certain columns of the data frame is needed, 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.

# Only include rows (=genes) that have deleterious mutations in data frame "mutation.all"  
mutation.all = mutation.all[which(mutation.all$isDeleterious == "TRUE"), ]  
  
# Order alphabetically:  
mutation.all <- mutation.all[order(mutation.all$Hugo\_Symbol),]

Frequently mutated genes are found by summing up all “mutation events” found in all cell lines. However, the problem arises that many mutations occur several times in one cell line, but in different DNA loci. These duplicates should not be included when counting the most frequent mutations among all cell lines. Therefore, the “duplicated” function from the dplyr package is used to 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

As can be seen, TP53 is the most commonly mutated gene (12 of 34 different cell lines). ARID1A follows with 8 mutated cell lines. Our driver mutations are thought to be the most frequently mutated genes. Further analysis will be conducted with these genes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Driver mutation | TP53 | ARID1A | ATM | BAI1 | PTPRF | SYNE1 |
| Number of affected cell lines | 12 | 8 | 5 | 5 | 5 | 5 |

### 3.2 Validation of drivermutations in literature –> Emily

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.

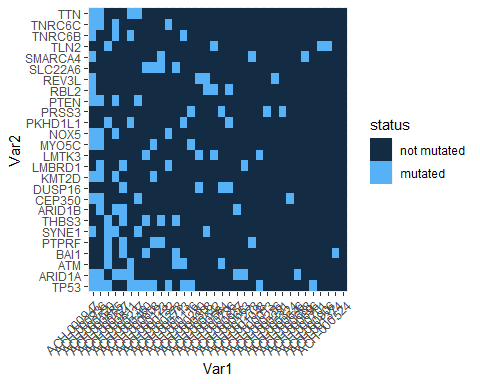
## 4. Co-existing Mutations

#### 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:

# Create a new data frame that contains the mutated genes (but just once per cell line, as definded in new\_uniq)  
dm\_null <- as.data.frame(table(new\_uniq$Hugo\_Symbol))   
  
#we extract all mutations that occur at least one time for a later step  
  
dm\_null = dm\_null[order(dm\_null$Freq, decreasing = TRUE),]  
rownames(dm\_null) <- dm\_null$Var1  
  
#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)  
  
#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)  
  
  
#the data is prepared to perform a heatmap  
mat <- matrix(anno, nrow = nrow(anno), ncol = ncol(anno), dimnames = list(rownames(anno), colnames(anno)))  
mat.melted <- melt(mat)  
  
  
ggplot(mat.melted, aes(x = Var1, y = Var2, fill = value)) +  
 scale\_fill\_gradient( name="status",  
 guide = "legend", limit = c(0,1), breaks=c(0,1), labels = c("not mutated","mutated")) +  
 geom\_tile( show.legend = T) +  
 theme(axis.text.x = element\_text(angle = 45, vjust = 1, size = 9, hjust = 1))



# Create a new data frame that contains the mutated genes (but just once per cell line, as definded in new\_uniq)  
#dm\_null <- as.data.frame(table(new\_uniq$Hugo\_Symbol))   
  
# Order matrix with mutations, rename rows  
#dm\_null = dm\_null[order(dm\_null$Freq, decreasing = TRUE),]  
#rownames(dm\_null) <- dm\_null$Var1  
  
# löschen, falls wirklich unnötig:  
# Creating a new matrix containing every cell line that has at least one of the 7 drivermutations. Then check if the Driver mutation names occur in the Driver mutation 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 <- as.data.frame(apply(dm\_null, 1, function(x) {annotation[x]<- ifelse(rownames(annotation) %in% dm[which(dm$Hugo\_Symbol %in% x), ]$DepMap\_ID, 1, 0)}))  
#anno <- as.matrix(apply(dm\_null, 1, function(x) {annotation[x]<- ifelse(rownames(annotation) %in% new\_uniq[which(new\_uniq$Hugo\_Symbol %in% x), ]$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 <- as.data.frame(anno[, which(colnames(anno) %in% rownames(dm\_zwei))]) # as.matrix  
#anno\_eins <- as.data.frame(anno[, which(colnames(anno) %in% rownames(dm\_eins))]) # as.matrix  
  
#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) # unnötig, wenn das mit as.matrix oben schon funktioniert  
  
#we create a color palette containing the colors we want to use  
#cols <- heat.colors(5)  
#mypalette <- colorRampPalette(cols)(2)  
  
# ACHTUNG EVTL ALLES LÖSCHEN:  
  
#we also add a second column to the matrix to include color associated Explanations for the legend  
#co <- as.data.frame(cols) # Zeilen 235 und diese kann man in einem Schritt: co <- as.data.frame(heat.colors(5))  
# co$kat <- c("not mutated", "mutated (> 9 mutations)", "mutated (> 3 mutations)", "mutated (> 1 mutations)", "mutated (1 mutation)") # braucht er nicht nochmal ???  
  
#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)  
  
#the data is prepared to perform a heatmap  
#mat <- matrix(anno, nrow = nrow(anno), ncol = ncol(anno), dimnames = list(rownames(anno), colnames(anno)))  
#mat.melted <- melt(mat)  
  
  
#ggplot(mat.melted, aes(x = Var1, y = Var2, fill = value)) +  
 #scale\_fill\_gradient( name="status",  
 #guide = "legend", limit = c(0,1), breaks=c(0,1), labels = c("not mutated","mutated")) +  
 #geom\_tile( show.legend = T) +  
 #theme(axis.text.x = element\_text(angle = 45, vjust = 1, size = 9, hjust = 1))

## 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)

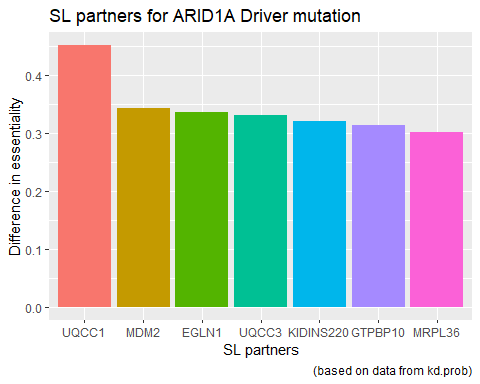
## BRD2   
## 0.6395188

rm(no, yes)

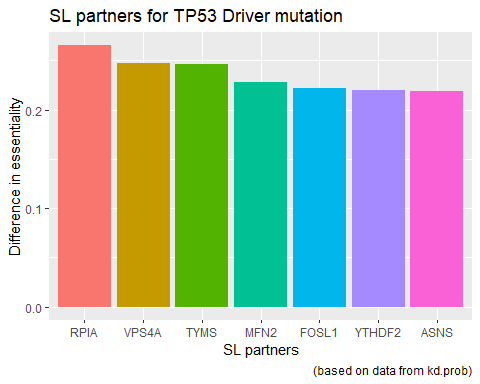
The output says that BRD2 is about 10% more essential in ARID1A mutated cells… This seems like a very low difference to be considered a “real” SL partner. Perhaps this is due to the fact that in the literature, BRD2 was mentioned as a SL partner explicitly in clear cell carcinomas. Our data comprises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

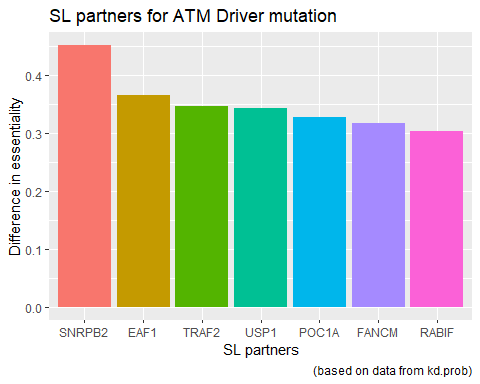
dmARID1A = dm[which(dm$Hugo\_Symbol == "ARID1A"), ]  
nonARID1A <- kd.prob[, -which(colnames(kd.prob) %in% dmARID1A$DepMap\_ID)] # prob Matrix with cell lines that do not have the ARID1A mutation  
yesARID1A <- kd.prob[, which(colnames(kd.prob) %in% dmARID1A$DepMap\_ID)] # prob matrix with cell lines 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)")



# Muss dann nebeneinander am besten angezeigt werden. Dann diskussion!  
  
  
  
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)")



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 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 driver mutations 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

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”

# Create an empty object consisting of 34 rows (cell lines) and 0 columns  
annodm <- as.data.frame(anno[, which(colnames(annotation) %in% c("nix"))])  
  
# Add a column for TP53 mutation; if a cell line contains the mutation, 1000 will be inserted into the cell of the matrix (otherwise 0)  
annodm$TP53 <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo\_Symbol %in% "TP53"), ]$DepMap\_ID, 1000, 0)  
  
# Add a column for ARID1A mutation; if a cell line contains the mutation, 100 will be inserted into the cell of the matrix (otherwise 0)  
annodm$ARID1A <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo\_Symbol %in% "ARID1A"), ]$DepMap\_ID, 100, 0)  
  
# Add a column for ATM mutation; if a cell line contains the mutation, 10 will be inserted into the cell of the matrix (otherwise 0)  
annodm$ATM <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo\_Symbol %in% "ATM"), ]$DepMap\_ID, 10, 0)  
  
# Add a column for PTPRF mutation; if a cell line contains the mutation, 1 will be inserted into the cell of the matrix (otherwise 0)  
annodm$PTPRF <- ifelse(rownames(annotation) %in% dm[which(dm$Hugo\_Symbol %in% "PTPRF"), ]$DepMap\_ID, 1, 0)  
  
# Sum up the designated values into a new column  
annodm$summe <- apply(annodm, 1, function(x) { sum(x)})

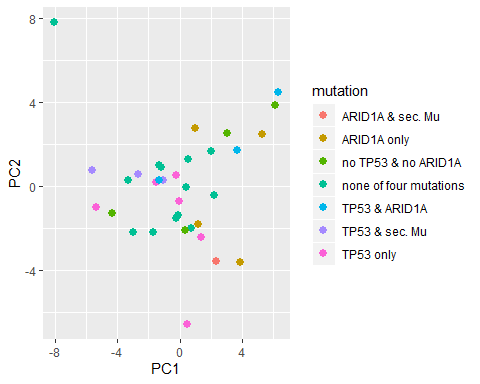
Many cell lines do not even have one of our chosen mutations because their sum is zero. At many others we can see that they have the same combination of Drivermutations because they have the same value at “annodm$summe”. In the following step we gave the different combinations a specific category

# Add new column saying which mutations a cell line has based on the calculated 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$summe == 0, "none of four mutations",  
 ifelse(annodm$summe < 100, "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.

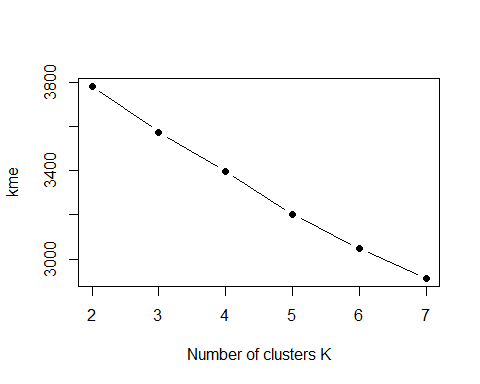
# calculate variance over all rows (genes)  
topVar = apply(kd.prob, 1, var)   
  
# new data frame with just the genes whose knockout leads to highly variant cell death events -> perhaps our synthetic lethality interaction partners?  
topVar = kd.prob[topVar > quantile(topVar, probs = 0.75), ]   
  
#we have to transpose the data frame  
topVar <- t(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  
PC<-prcomp(topVar)  
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)



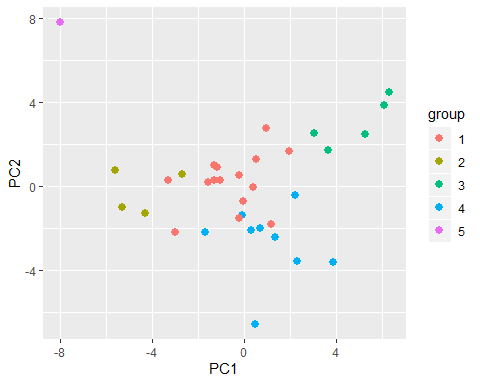
#### 5.2.3 PCA combined with kmeans cluster

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(topVar, centers = k)$tot.withinss  
})  
plot(2:7, kme, type = "b", pch = 19, xlab = "Number of clusters K")



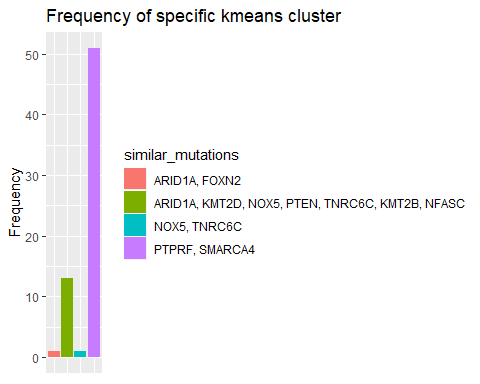
#the elbow plot does not give clear information about the ideal number of clusters, but we will use five centers (silhouette plot is in progress)  
  
#performing clustering  
km <- kmeans(topVar, 5)  
  
#extracting the colors  
group <- as.factor(km$cluster)  
  
  
#plotting PCA as before  
ggplot(PCi,aes(x=PC1,y=PC2,col=group))+geom\_point(size=2.5)



#### 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 1000 kmeans cluster  
for (j in 0:1000) {   
  
 #a kmeans cluster is formed  
 km <- kmeans(topVar, 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)  
   
 #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) {  
   
 #groups with too many cell lines arent interesting as well  
 if (NROW(kp) < 9) {  
 #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"}}  
   
 else {po[h] <- "nein"}  
 }   
}  
  
#the data frame is transposed to get a better overview and to perform further analysis  
po <- t(po)  
  
#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) +  
  
 theme(axis.title.x=element\_blank(),axis.text.x=element\_blank(),axis.ticks.x=element\_blank()) +  
 labs(x = "Mutated gene", y = "Frequency", title = "Frequency of specific kmeans cluster")



#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. The other group which contains the mutations: “ARID1A”, “KMT2D”, “NOX5”, “PTEN”, “TNRC6C” etc. is formed out of three cell lines (ACH-000947 ACH-000936 ACH-000885). Their Subtype desease is “Adenocarcinoma”, but there are many others with the same Subtype desease. But caused by the fact that those three are clustered together very often by their kd.prob data. It is possible that one of these mutated genes has a promising SL partner.

## 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. Several studies indicate that a variation of the gene copy number can affect the level of gene expression for the altered gene. Also a copy number variation can influence the gene expression positively or negatively (Stranger et al., 2007). To investigate if there is a linear relationship between the gene expression and the gene copy number, 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.

# Bind the two data frames together in one list  
list\_all.genes <- list(expression,copynumber)  
  
# Rename the elements of the new list  
names(list\_all.genes) <- c("expression","copynumber")

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.

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 and therefore a deviation to the gaussian distribution.  
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 model. Predictions will only be possible for genes with a copy number higher than -2.

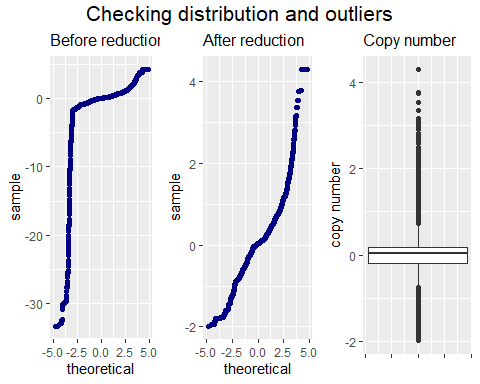
# Find all genes with at least one values <= -2  
rmv.rows = apply(list\_all.genes$copynumber, 1, function(x) {sum(x <= -2)})   
  
# Remove rows, that contain more than one value <= -2  
list\_all.genes$copynumber <- list\_all.genes$copynumber[-which(rmv.rows > 0), ]

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.

# Fuse all cell lines to one long column  
cn1 <- melt.data.frame(list\_all.genes$copynumber, variable\_name = "cellline")

## Using as id variables

# Plot the quantils of the copy numbers with the quantils of a gaussian distribution (before reduction process)  
plotcnb <- ggplot() + geom\_qq(aes(sample = cn1b$value), color= "navy blue") + ggtitle("Before reduction") + theme(plot.title = element\_text(size=12, lineheight = 2))   
  
# Plot the quantils of the copy numbers with the quantils of a gaussian distribution (after reduction process)  
plotcn <- ggplot() + geom\_qq(aes(sample = cn1$value), color= "navy blue") + ggtitle("After reduction") + theme(plot.title = element\_text(size=12, lineheight = 2))  
  
# Create boxplot for the copynumber data + remove the scaling of the x-axis  
boxplotcn <- qplot(y=cn1$value, x= 1, geom = "boxplot")+ ggtitle("Copy number") + xlab("") + ylab("copy number") + theme(axis.text.x=element\_blank(), plot.title = element\_text(size=12, lineheight = 2))   
  
# Arrange the two plots next to each other  
grid.arrange(plotcnb, plotcn,boxplotcn, ncol=3, top = textGrob("Checking distribution and outliers", gp=gpar(fontsize=16)))



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 affect 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.

# Identify shared rownames in both data frames  
common\_names = Reduce(intersect, lapply(list\_all.genes, row.names))   
  
# Only keep the common rownames in the list  
list\_all.genes <- lapply(list\_all.genes, function(x) {x[row.names(x) %in% common\_names,]})

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.

# Fuse all expression values to one long column  
exp1 <- melt.data.frame(list\_all.genes$expression, variable\_name = "cell line")

## Using as id variables

# Fuse all copy number values to one long column  
cn1 <- melt.data.frame(list\_all.genes$copynumber, variable\_name = "cell line")

## Using as id variables

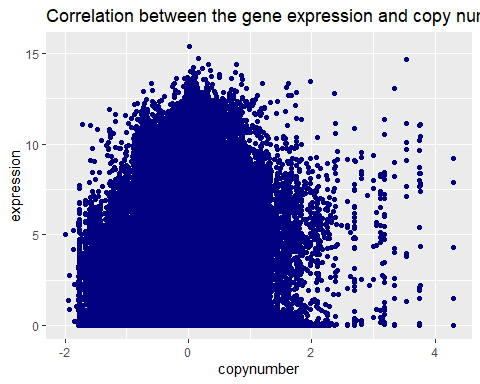
# Merge both columns in new object  
regression.all.genes <- as.data.frame(cbind(exp1$value, cn1$value))   
  
# Rename the columns of the new object  
colnames(regression.all.genes) <- c("expression", "copynumber")

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

# Create scatter plot for expression and copy number   
ggplot(regression.all.genes, aes(x = copynumber, y = expression)) + geom\_point(color = "navy blue")+ labs(title = "Correlation between the gene expression and copy number", xlab= "copy number", ylab="gene expression")



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 pearson 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.

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

## [1] 0.09071741

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

## [1] 0.07229923

Both correlation coefficients show a very weak correlation between the two varibles of approximately 0.09 to 0.08. The sligthly lower spearman correlation concludes, that the person correlation is lightly influenced by outliners. With a value smaller than 0,1 for both correlation coefficients only a very weak correlation between the gene expression and the gene copy number is indicated. To verify, that the correlation is not equal to zero and the calculated coefficient is due to a statistical error 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 person correlation coefficient is verified.

# t-Test for the pearson correlation coefficient  
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 a 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.

# t-Test for spearman correlation coefficient  
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 calculated correlation coefficients would not be obtained if the real correlation is equal zero. But still the small values for the coefficients shows a very weak correlation between the gene expression and the copy number. Therefore the copy number of a gene may not fit to predict the gene expression. To decide whether the gene copy number is helpful for prediction purposes, even if the two variables show a very weak linear relationship, a linear regression model will be applied.

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

# Remove objects  
remove(exp1, cn1, cn1b, boxplotcn, plotcn, plotcnb, common\_names, rmv.rows)

#### 6.5 Univariate linear regression

Before using the regression.all.genes data for a regression model, the data set has to be split in a training and testing data set. A common used ratio for the data splitting is 80/20 or 70/30, the regression.all.genes data set will be split in approximately 70% for the trainings data and 30% for validation process afterwards.  
198920 of 663068 rows are taken randomly of the regression.all.genes matrix and will later be used as trainings data for the model.

# Establish new dataframe with 198920 randomly selected rows of dataframe regression.all.genes  
testing.all.genes <- regression.all.genes[sample(1:nrow(regression.all.genes), 198920),]

Next the 198920 rows are removed from the original object to exclude them from the later performed linear regression model.

# Remove the testing data of the primary data set   
regression.all.genes <- regression.all.genes[ -sample(1:nrow(regression.all.genes), 198920),]

After the separation of the testing data, the training data is integrated in the linear regression model lrm.all.genes. The variable copynumber is used as a predictor for the variable gene expression.

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

##   
## Call:  
## lm(formula = expression ~ copynumber, data = regression.all.genes)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.9896 -2.3579 -0.3794 1.8721 12.0990   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.529347 0.003614 699.97 <2e-16 \*\*\*  
## copynumber 0.572766 0.009287 61.67 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.462 on 464146 degrees of freedom  
## Multiple R-squared: 0.008129, Adjusted R-squared: 0.008127   
## F-statistic: 3804 on 1 and 464146 DF, p-value: < 2.2e-16

With the calculated intercept of 2.522913 and a slope of 0.581562 the equation for the regression is:

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

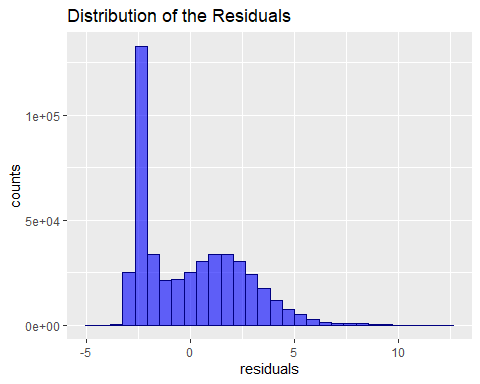
With a p-value smaller that 2e-16 the t-Test indicates, that the slope of the regression line has a significant aberration from zero.  
The standard error for the residuals indicates an average aberration of 2.459 between the predicted expression value and the expression value of the dataset. For a good 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 (copy number) 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 0.008392. That the value is near zero suggests, that the variable copy number does not fit to explain the variable gene expression. This possibly results of the limitation to only one predictor variable for the regression model and the complex molecular biological mechanisms for regulation auf the gene expression, which will be explained in a paragaph 6.7

For the integrity of the analysis the requierments for a good fit of the regression model will be checked. The first requierment is, that the residuals are distributed symmetric around a median of approximately 0. With a calculated value of -0.3827 a slightly deviation of zero is obtained. To facilitate the futher analysis of the residuals, they are saved in the new object Resid

# Save residuals in single data frame   
Resid <- as.data.frame(lrm.all.genes$residuals)   
  
# Rename the column of the new data frame  
names(Resid) <- c("residuals")

Another requirement for the residuals is normality, which is checked with a histogramm of the frequency of certain values for the residuals

# Plot histogram for residuals of the training data  
ggplot(data = Resid, aes(Resid$residuals)) + geom\_histogram(bins = 30, col = "navy blue", fill="blue", alpha= .6) + labs( x="residuals", y="counts", title = "Distribution of the Residuals")



The histogram of the residuals show a hugh abnormality of the gaussian distribution in the lower tail. Because a histogramm only helps to obtain a visual overview of the distribution a Kolmogorov–Smirnov test is performed to verify if the distribution of the residuals fits normality. In general the test verfies wether two data sets share the same distribution. For this issue the distribution of the residuals will be compared to a gaussian distribution. The null hypothesis of the statistical test proves a normal distribution for the residuals, while the alternative hypothesis leads to the assumption, that the data is not distributed normally.

# test the distribution of the residuals for normality  
ks.test(Resid$residuals, "pnorm")

##   
## One-sample Kolmogorov-Smirnov test  
##   
## data: Resid$residuals  
## D = 0.34838, p-value < 2.2e-16  
## alternative hypothesis: two-sided

With a p-value smaller than 0,05 the null hypothesis can be rejected, which concludes, that the distribution of the residuals does not fit normality.

Another requirement for the accuracy of a regression model is, that the residuals are independent of the predictor variable. To analyse this circumstance the correlation between the predictor varibale (copy number) and the residuals is checked.

# Estimate the pearson correlation coefficient  
cor(regression.all.genes$copynumber ,lrm.all.genes$residuals)

## [1] -5.52336e-16

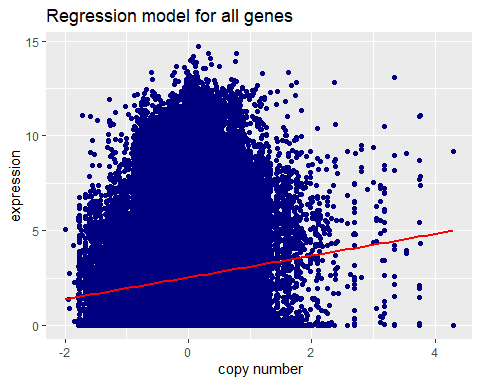
# Test the significance of the pearson correlation coefficient  
cor.test(regression.all.genes$copynumber, lrm.all.genes$residuals)

##   
## Pearson's product-moment correlation  
##   
## data: regression.all.genes$copynumber and lrm.all.genes$residuals  
## t = -3.763e-13, df = 464146, p-value = 1  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.002876869 0.002876869  
## sample estimates:  
## cor   
## -5.52336e-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 obtain a visual overview.

# Plot scatter plot for expression and copy number   
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") + xlab(" copy number")



The scatter plot shows a high aberration of the predicted values for the gene expression from the regression line. The previous checking process leads to the conclusion, that the model does not fit to predict the gene expression. To futher analysis this postulation the model will be applied for the testing data set.

### 6.6 Testing the model

In this step the testing data will be used to examine the aberration between the predicted expression and the real expression 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.

# Form new object with the predicted expression values for the testing data set.  
prediction.testing <- as.data.frame(predict(lrm.all.genes, newdata = testing.all.genes))   
  
# Name the column of the new object  
names(prediction.testing) <- c("pred.exp")

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

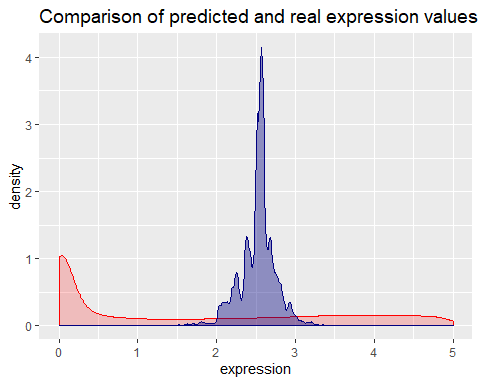
In the formula n displays the number of rows in the given data frame. To calculate the residuals the difference between the real value and the predicted value of the gene expression is used.

# Compute standard error for the expression values of the testing data set  
sqrt(1/nrow(testing.all.genes) \* sum((testing.all.genes$expression-prediction.testing)^2))

## [1] 2.464139

The value of 2.463125 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. For a concluding verification the distribution of the predicted gene expression and real values of the gene expression are plotted for the testing data.

# Plot the density for the predicted and real values (testing data)  
ggplot()+ geom\_density(data = testing.all.genes,aes(x=expression), color="red", alpha=.2, fill= "red") + geom\_density(data = prediction.testing, aes(x=pred.exp), color ="navy blue", alpha=.4, fill= "navy blue" ) + xlim(0,5) + ggtitle("Comparison of predicted and real expression values") + theme(plot.title = element\_text(size=14, lineheight = 2))



The plot illustrate the distribution of the real expression values in red, while the predicted values are indicated in blue. While the predicted values are normally distributed, the reals values show a high density between 0 and 0.5. Values higher than 0.5 hold steady at a frequency of approximately 0.2. To illustrate the aberration between the to distribution the maxima expression value is set to 5. It has to be considered that the real expression values have a maxima around 15, but values about 5 only represent a very small proportion of the expression data. The plots show once more that the copy number of a gene does not fit to predict the gene expression.

### 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.

, which would be integrated in a multivariate linear regression model. Futhermore the prediction process should only be applied to a certain gene and not the totality of all genes. This would decrease the different regulatory mechanisms between different genes, although regulation also differ in the cells. The approach of investigate the correlation between both variables for a single gene was not performed, due to the fact that each gene only contains 34 data points.

## 7. Conclusion

### 7.1 Driver mutation

### 7.2 Co-existing Mutations

### 7.3 Synthetic Lethality Interactions

### 7.4 Linear Regression

In the last step a linear regression model was applied to the question whether it is possible to predict the gene expression based on the copy number for the totality of the genes. Because the predictor variable copy number did not fit normality, all values smaller than -2 were removed, limiting the prediction to copy number values higher than -2. Furthermore, a very weak correlation was observed between the two variables, which was verified with a t-Test. This circumstance forecast a bad fit of the two variables for the regression model. While 70% of the data set were used as trainings data for the model, 30% were separated for testing purposes. The R-squared conclude, that only 0,8 % of the variation of the expression data can be explained by the gene copy number and therefore other variables should be considered as prediction variable. The examination of the residuals confirms, that they do not correlate with the predictor copy number, but the distribution of the residuals does not fit normality. The verification with the testing data set illustrate the huge aberration between the predicted and real values for the gene expression. The outcome of the model shows that the complex biological regulatory mechanisms, which influence the level of gene expression on different ways, can not be explained by a single variable. For an accurate predication a multivariate regression model, which includes several of the in 6.7 mentioned variables, is suggested.

## 8. Sources