Project 3: Ovarian Cancer

Nina Hahnen, Elena Kaube, Marco Rheinnecker, Emily Tropf

Contents

1. Project Overview

1.1 General facts about Ovarian Cancer

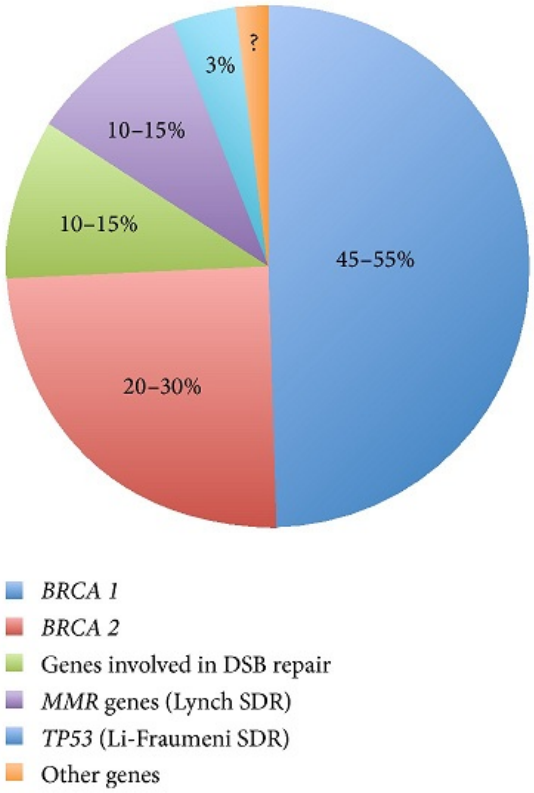
Ovarian cancer (OC) is one of the leading cancer death causes in women. One of 77 women will be diagnosed with this disease at some point during her lifetime (Testa *et al.*, 2018). When treated in early stages, OC has relatively high survival rates. However, symptomes mostly start to develop when the 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 (Testa *et al.*, 2018). 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, as described by Testa *et al.*. For example, literature on hereditary Ovarian cancer states that the most common mutations are found in the DNA repair-related genes BRCA1 and BRCA2 (Toss *et al.*, 2015). These are however rarely found in non-hereditary Ovarian cancer. 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 and carcinosarcomas. Another 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 (O'Neil *et al.*, 2017). 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 Load packages

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

```
# Load packages
library(reshape)
library(ggplot2)
library(data.table)
library(gridExtra)
library(grid)
```

2.2 Data Cleanup

First, the dataset needs to be loaded 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.

```
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.

```
# Bind individual objects of the list in one data frame
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".

```
# Extract needed colums of the data frame
 mutation.all = mutation.all[, which(colnames(mutation.all) %in% c("Hugo_Symbol", "DepMap_ID", "Varia
nt_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).

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

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

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 driver mutations with literature

To determine whether the six obsorved mutations are associated with cancer and can be seen as driver mutations for the data set, the function of the genes are look up in the literature

TP53 TP53 is a tumor suppressor gene, which is coding for the transcription factor p53. The protein regulates DNA repair mechanisms and apoptosis. An inactivation of the tumor suppressor gene, also leads to an increased genomic instability. This mutation is not only the most frequent mutation in ovarian cancer (especially in high serous carcinomas.), but also occur in a multiplicity of other cancer types (Mandilaras *et al.*, 2019).

ARID1A ARID1A is a member of SWI/SNF family, which influence the expression level via chromatin remodeling. This process regulates on the second base the cell proliferation and differentiation. The tumor suppressor gene is mutated in 50% of all ovarian clear cell carcinomas (Berns *et al.*, 2018).

ATM ATM encodes for a protein kinase and serve as a Checkpoint for the genome replication. A mutation in the repair enzyme leads to chromosomal fragility causing a higher risk of for breast and ovarian cancer (Thorstenson *et al.*, 2003).

BAI1 BAI1 inhibits the angiogenesis and is a cell growth suppressor. The neoplasm of blood vessels is essential for tumor proliferation and metastasis of tumors. BAI1 functions as tumor suppressor gene in human cells(Chao *et al.*, 2015).

PTPRF PTPRF is a tumor suppressor gene, which was found in different cancer types. The exact function of the gene and the relationship to cancer development is fully investigated. (Tian et al., 2018)

SYNE1 SYNE1 encodes a protein expressed in skeletal, smooth muscle and peripheral blood lymphocytes. (Doherty et al., 2010)

THBS3 THBS3 encodes a glycoprotein that mediates cell to cell and cell to matrix interactions and is also involved in the regulation of skeletal maturation. A mutation of this gene was found in different cancer types but is fully investigated.

The tumor suppressor genes TP53 and ARID1A, which both are associated with Ovarian cancer, will be used as main driver mutations for the following analysis. They also occur most frequently in ovarian cancer and inhibiting the inactivation of the two genes via synthetic lethality interaction will have a positive impact on the mortality rate for several subtypes of ovarian cancer. Since the relationship between ovarian cancer and SYNE1/ THBS3 is not fully explained, the mutations will not be included in further analysis. The mutations ATM, BAI1 and PTPRF will be considered as driver mutation of second order.

4. Coexisting Mutations

After identifying the most frequent mutations and verifying the driver mutations with the literature, this step aims to detect whether most cell lines only exhibit one driver mutation or show a coexistence of several driver mutations. Mutations that often occur together are possibly linked to each other in their genesis and thereby perhabs contribute to a deeper understanding of cancer cell development. First, a new object with one column for the mutated genes and a second column for the frequency of the specific mutation is defined. Again, the data frame new_uniq is used, since each mutation must not be considered more than once per cell line.

```
# Create new data frame with one column for the mutated gene at the corresponding frequency
dm_null <- as.data.frame(table(new_uniq$Hugo_Symbol))

# Order the genes with decreasing order for the frequency
dm_null = dm_null[order(dm_null$Freq, decreasing = TRUE),]

# Use the names of the genes as rownames
rownames(dm_null) <- dm_null$Var1
```

To later determine whether a specific mutation is present in a certain cell line, a new object is created. The cells of the new object contain either “1” or “0”, giving the information whether the respective mutation occurs in the cell line (“1” if TRUE) or not (“0” for FALSE). The single genes are be added as columns:

```
# Create new data frame for the analysis
dm <- new_uniq[which(new_uniq$Hugo_Symbol %in% dm_null$Var1), ]

# Apply function to dm_null, "1" indicates the mutation's presence, "0" its absence
anno <- apply(dm_null, 1, function(x) {annotation[x]<- ifelse(rownames(annotation) %in% dm[which(dm$H
ugo_Symbol %in% x), ]$DepMap_ID, 1, 0)})

# Transform new matrix to a data frame
anno <- as.data.frame(anno)

# Add the cell line numbers as row names
rownames(anno) <- rownames(annotation)
```

The new object can now be used to calculate the number of mutations in one cell line based on the sum of the values stored in a row’s cells.

```
# Sum up the values for each cell line to determine the total of mutated genes
anno$summe <- apply(anno, 1, function(x) { sum(x)})

# Show the characteristics for the column summe
summary(anno$summe)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   23.00   38.75   48.00   88.71   67.50  529.00
```

The minimum of 23 mutations present in one cell line proves that mutations are no single event in each cell, but that mutations tend to accumulate in cancer cells. The average cell line exhibits a total of 89 mutated genes. However, many of these mutations are so-called passenger mutations that randomly occur in cells with dysregulated cellular mechanisms and are often the result of genome instability due to the occurance of a driver mutation (Alberts *et al.*, 2017). They do not contribute to cancerogenesis and can therefore not be considered as driver mutations. To sort them out, only mutated genes that are present in more than three cell lines are used for the futher analysis.

```
# Create object with mutations that occur at least in 2 or 4 cell lines (dm_eins is required for
step 5)
dm_eins <- dm_null[which(dm_null$Freq > 1), ]
dm_drei <- dm_null[which(dm_null$Freq > 3), ]

# Use the above created objects to remove genes that do not occur at least two or four times
(anno_eins is required for step 5)
anno_eins <- as.data.frame(anno[, which(colnames(anno) %in% rownames(dm_eins))])
anno <- anno[, which(colnames(anno) %in% rownames(dm_drei))]
```

After the reduction process, the sum of the single mutations for each cell line is calculated again.

```
# Sum up the mutated genes for each cell line
anno$summe <- apply(anno, 1, function(x) { sum(x)})

# Order the cell lines in a decreasing manner
anno <- anno[order(anno$summe, decreasing = TRUE),]

# Show the characteristics for the column "summe"
summary(anno$summe)
```

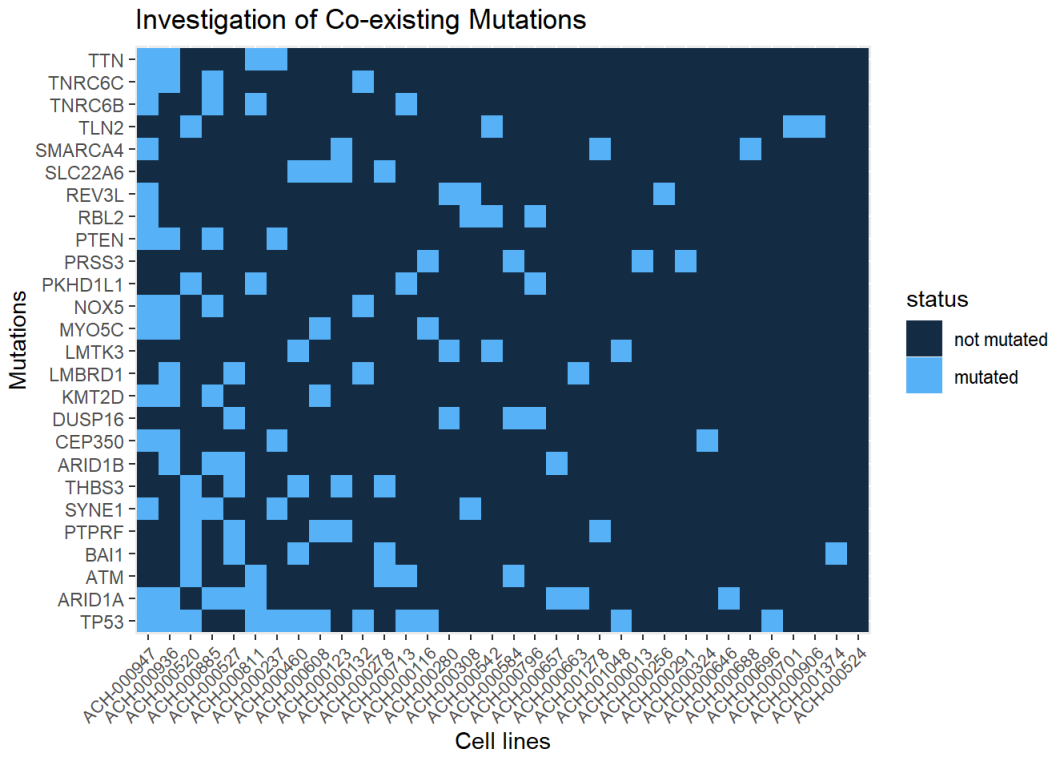
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   1.000   3.000   3.559   4.750  14.000
```

Including mutations that occur at least in four different cell lines limits the amount of observed mutations. Approximately three to four frequent mutations are observed in each cell line, while the previous results suggested an average number of 89 mutations. The minimum of 0 can be explained as follows - a cell line that only exhibits “rare” passenger mutations which were sorted out in one of the previous steps would appear to not have any mutations. To later use the data frame anno for a heat map to obtain a visual overview the column “summe” has to be removed and the data frame is converted in a matrix.

```
# Remove column summe to exclude it from the heat map
anno <- anno[, -which(colnames(anno) == "summe")]

# Convert object anno into matrix
anno <- data.matrix(anno)
```

Next the data is saved in another object which can then be used to create a heatmap. By means of color-coding, this heatmap indicates whether a specific gene is mutated in a cell line.

```
# Prepare data for application of a heatmap
mat <- matrix(anno, nrow = nrow(anno), ncol = ncol(anno), dimnames = list(rownames(anno),
colnames(anno)))

# Melt the values to one long column
mat.melted <- melt(mat)

# Plot cell lines and mutations in a heatmap
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 = 8, hjust = 1)) + ggtitle("Investigat
ion of Co-existing Mutations") + xlab("Cell lines") + ylab("Mutations")
```



The heatmap once more indicates that cell lines with only a single driver mutation are rare, while most cell lines consist of combinations of different mutations. The heatmap shows the cell line with the highest number of mutations on the left; the number of mutations decreases to the right side. Only cell line ACH-000524 does not contain any of the frequent mutations, and 10 of 34 cell lines only have one of the defined driver mutations. Obviously, a general mutation whose existence is associated with the development of overian cancer does not exist. Each cancer cell exhibits a high amount of genetic variation due to mutation, which most probably is the result of genomic instability. Studies state that 33 to 66 is the average number of deleterious mutations in cancer cells (including passenger mutations). It is possible that the heatmap shows driver and passanger mutations - it would have to be verified in further experiments whether respective mutation has an impact on cancer development. Nevertheless, the heatmap shows that most cell lines contain a complex combination of altered genes. These circumstances must be considered in the search for synthetic lethality interaction partners.

5. Synthetic Lethality Interactions

5.1 Identification of SLI partners with applied statistics

As mentioned in the introduction, it is our goal to seek out synthetic lethality interaction (SLI) partners for driver mutations occurring in Ovarian cancer. These SLI partners are genes that become essential in cancer cells due to the presence of mutations that lead to altered protein function and signaling pathways. The main idea behind identifying the SLI partners is finding a way to target cancer cells without affecting healthy cells (O'Neil *et al.*, 2017). A cancer drug leading to effective knockdown of SLI partner genes in a patient's tumour could specifically induce cell death in cancer cells. This kind of approach in combination with other therapies would hopefully increase treatment efficacy and reduce unwanted side effects, thus improving life quality.

The task is to find possible SLI partners for the most frequent driver mutations, these being TP53 and ARID1A. For every driver mutation, our batch of cell lines will be devided into two groups depending on presence or non-presence of the respective driver mutation.

Using the matrix kd.prob, a one-sided unpaired t-test will be performed in order to calculate the p values for gene essentiality in both groups respectively. It was hereby assumed that SLI partners exhibit higher essentiality values in cell lines that possess the respective driver mutation. By extracting the genes that show the lowest p values thus indicating highly significant differences concerning essentiality, potential SLI interaction partners can be identified. First, the SLI partner search will be performed for ARID1A:

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


# Remove TBC1D3 gene because it has only zeros as values
yesARID1A <- yesARID1A[-14997, ]
nonARID1A <- nonARID1A[-14997, ]

# Perform one-sided t test with all marker genes (unpaired)
p.values = sapply(1:nrow(yesARID1A), function(i) {

    # Compute t-test for all genes whose knockout leads to increased essentiality
    t.test(yesARID1A[i, ], nonARID1A[i, ], alternative = "greater")$p.value
})

p.values <- setNames(p.values, rownames(yesARID1A))

# Significance level is 0.1 %
p.values.sig <- p.values[p.values<0.001]
```

In the literature, the gene BRD2 is mentioned as a synthetic lethality interaction partner for ARID1A in Ovarian clear cell carcinomas (Berns *et al.*, 2018). It remains to be found out which significance the difference in survival probability this SLI partner displays.

```
# Check p-value for alleged SL partner BRD2
p.values[grep("\\bBRD2\\b", rownames(as.matrix(p.values)))]
```

```
##     BRD2
## 0.176329
```

The p value for BRD2 is 0.1763, which statistically is far away from being significant. It can be concluded that in Ovarian Cancer in general, BRD2 is not a significant SL partner in cell lines with ARID1A mutations. However, the most likely reason for this discrepancy is that Berns *et al.* focused on the Ovarian cancer subtype clear cell carcinomas. Since each transcriptional program is very complex, it is possible that SLI partners strongly depend on cell types.

The same statistical testing procedure as was performed for ARID1A is applied to TP53:

```
dmTP53 = dm[which(dm$Hugo_Symbol == "TP53"), ]

# Probability Matrix with cell lines that do not have the TP53 mutation
nonTP53 <- kd.prob[, -which(colnames(kd.prob) %in% dmTP53$DepMap_ID)]

# Probability matrix with cell lines that do have the TP53 driver mutation
yesTP53 <- kd.prob[, which(colnames(kd.prob) %in% dmTP53$DepMap_ID)]

# Remove TBC1D3 gene because it has only zeros as values
yesTP53 <- yesTP53[-14997, ]
nonTP53 <- nonTP53[-14997, ]

# Perform one-sided t test with all marker genes (unpaired)

p.val = sapply(1:nrow(yesTP53), function(i) {
# Compute t-test for all genes whose knockout leads to increased essentiality
t.test(yesTP53[i, ], nonTP53[i, ], alternative = "greater")$p.value
})

p.val <- setNames(p.values, rownames(yesTP53))

# significance level is 0.1 %
p.val.sig <- p.val[p.val<0.001]
```

Stored in p.values.sig for ARID1A and p.val.sig for TP53 are the most significant synthetic lethality interaction partners. These can later be consulted and for example be validated in further experiments (e.g. a larger group of Ovarian cancer cell lines, mouse models).

5.2 Grouping of the cell lines based on the combination of mutations

In step 4.2 the driver mutations were validated with literature, leading to the assumption of the four mutations TP53, ARID1A, ATM and PTPRF, which are associated with cancer development. As indicated in step 4, most driver mutations occur in a combination with other driver mutations in the cell lines. To investigate the specific interaction with possible synthetic lethality via PCA, the cell lines need to be grouped based on their mutational signatures first. In the following, each of the four driver mutations will receive a different decimal power, which are listed in the table below.

Driver mutation	Number
TP53	1000
ARID1A	100
ATM	10
PTPRF	1

Before using the decimal power as a “sign” for the present driver mutations in a cell line, a new object is created. In the rows, the 34 cell lines are listed and a column for each mutation indicates the combination of mutations for the specific cell line. Each cell with a “0” shows that the driver mutation (column name) is not present in the cell line, while “1”, “10”, “100” and/or “1000” indicate the existence of the driver mutation(s).

```
# 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 t
he 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, 1
0, 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)})
```

The sum of all values for each cell line indicates a specific combination of driver mutations and can be used to identify these. The column “summe” is now used to name the mutation combination and will highlight the different groups in the following Principal Component Analysis. A new column for written mutation combinations a is added to the object annodm.

```
# 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"
)))))))
```

5.3 PCA cluster by the driver mutation combination

To investigate possible SLI partners a Principle Component Analysis (PCA) is performed. In general a PCA can be applied for dimension reduction and identify a large data set’s Pinciple Components, which condense the essential information of the data set. Differences in the

data set exhibit the most information because the data points can be categorized by variables that shows the variance in the data set. As described in the paragraphs above, genes with a high variance are likely to be possible SLI partners. First the knockdown data is reduced to genes that have a variance greater than the 75% quantile.

```
# Calculate variance over all rows for the gene knockdown (probability)
topVar = apply(kd.prob, 1, var)

# Reduce data set to genes that indicate a variance greater than the 75% quantile
kd.prob.topVar = kd.prob[topVar > quantile(topVar, probs = 0.75), ]

# Transpose the data frame
kd.prob.topVar <- t(kd.prob.topVar)
```

Next the reduced knockdown data is exerted in a PCA. To obtain a clustering of the different driver mutation combinations, the defined groups of the previous step are coloured.

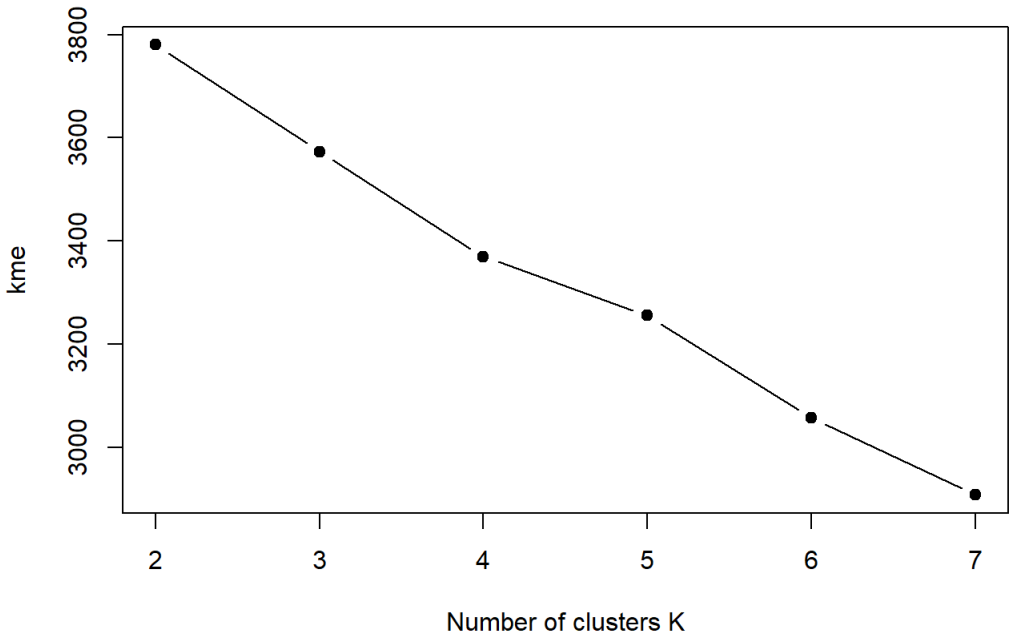
```
# Calculate the variance and Principle Components for the knock.down data
PC<-prcomp(kd.prob.topVar)
PCi<-data.frame(PC$x)

# Use different colours for the defined groups in step 5.2
Mutation <- annodm$kat

# Plot the PCA for and show defined groups in different colours
ggplot(PCi,aes(x=PC1,y=PC2,col=Mutation))+geom_point(size=2.5) + ggtitle("Cell lines clustered by
driver mutations")
```

As can be seen, no clustering of the groups, which are defined by their driver mutation pattern, is visible. In this form, the PCA does not support the significant SLI partners found by means of the statistical test. The reason for this could be that the observation of many driver mutations at once in a cell line poses a too great complexity of genetic characteristics and can therefore not be plotted this simply. A more direct way to find SLI partners would be to establish cell lines with one driver mutation only and to subsequently introduce further mutations to examine the complex interplay of several mutations.

5.4 PCA combined with kmeans cluster

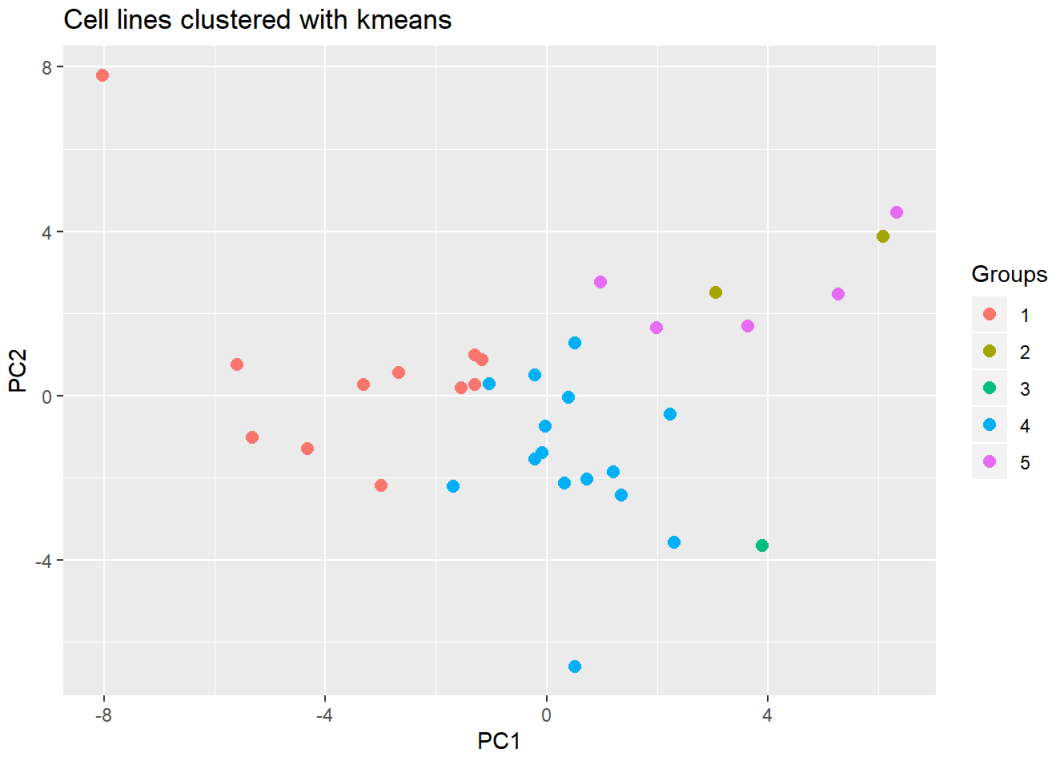
Obviously, the cell lines do not visibly cluster based on the combination of the most frequent driver mutations. As an alternative idea, the clustering could be based on the different OC subtypes. The problem hereby is that the subtype definitions are very vague and numerous, which is disadvantageous since our data only comprises of 34 cell lines. The high number of different subtypes can be seen when using the function `summary()`. For this reason, an elbow plot will be performed prior to the PCA to verify the quality of the clustering. A visible kink in the elbow plot indicates a more drastic decrease in WSS, which is a measure for the distance between the points of a cluster. Finally, a kmeans clustering and PCA can be performed.

```
# Find optimal number of cluster
kme = sapply(2:7, function(k) {

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



Unfortunately, the elbow plot does not give clear information about the ideal number of clusters. However, this is not surprinsing since the dataset comprises of various different Ovarian cancer subtypes that do not exhibit great relation as has already been shown in the analysis of occurring driver mutations. Five centers will be used in the following.

```
# Perform kmeans clustering
km <- kmeans(kd.prob.topVar, 5)

# Extract groups of cell lines for the plot
Groups <- as.factor(km$cluster)

# Plot PCA with different colour for each group
ggplot(PCi,aes(x=PC1,y=PC2,col=Groups))+geom_point(size=2.5)+ggtitle("Cell lines clustered with
kmeans")
```



As defined, there are five groups. Here, as well, there are no visibly seperated clusters. Nevertheless, the clusters are situated in distinct regions of the plot, suggesting a connection between the cell lines. The composition of the clusters is displayed in the following table showing the observed subtype for each cluster:

Cluster	Subtype
1	Adenocarcinoma (various types: serous, endometrioid, clear cell)
2	Teratocarcinoma
3	Adenocarcinoma (serous), Carcinoma, Cystadenocarcinoma
4	Carcinoma, Adenocarcinoma (clear cell, serous), Cystadenocarcinoma
5	Small cell carcinoma, Adenocarcinoma (mucinous, clear cell, endometrioid), Carcinoma

There does not seem to be a clear seperation of subtypes when clustering using the knockdown data. Against our expectations, these results suggest that SL interaction partners are not greatly dependent on cancer subtype. For more convincing results, a more explicit classification regarding the subtypes would have been required. Furthermore, a higher number of different cell lines would have been advantageous.
Another possible reason for the inconclusive results is due to possible inefficiency of RNA knockdown which was applied to the cell lines.

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 all given genes. Several studies state that a variation of the gene copy number can affect the level of gene expression for the affected gene. What is more, 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 copy number and the gene expression, a linear regression model will be administered.

6.1 Checking the distribution

The data frames expression and copy number are bound to form the list list_all.genes. This process ensures 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 preferentially, there should only a low number of outliers.
First all, the copy number values of the different cell lines are fused to one long column. The new matrix cn1b only contains two columns.

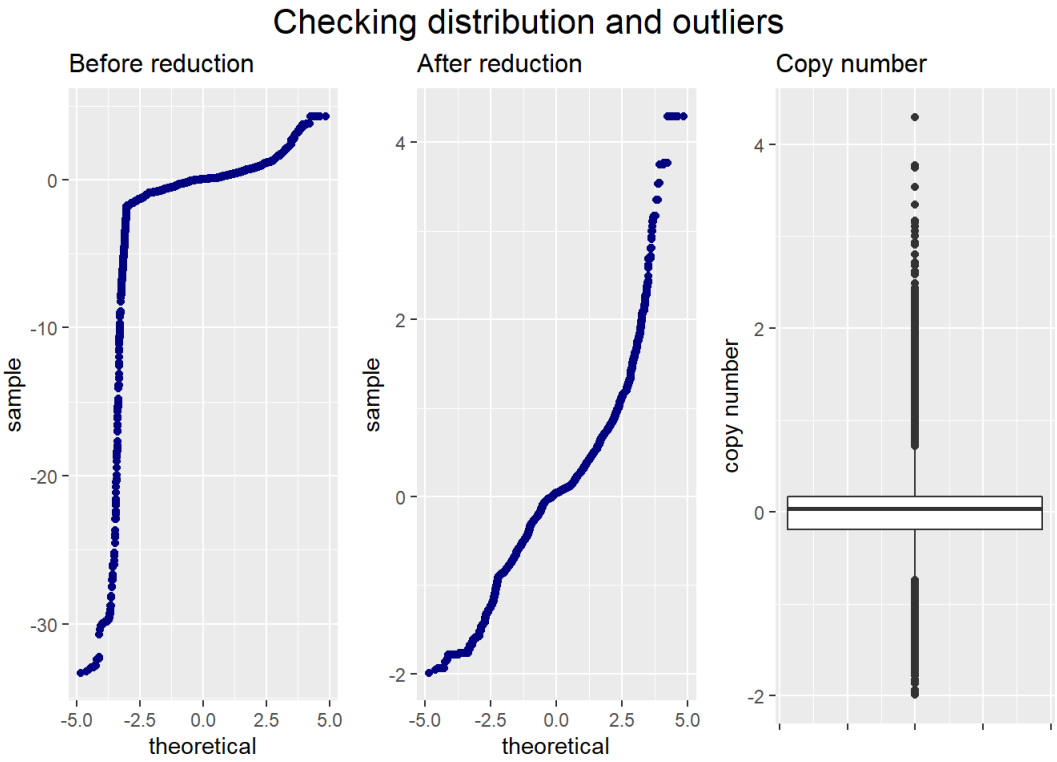
```
# Fuse all cell lines
cn1b <- melt.data.frame(list_all.genes$copynumber, variable_name = "cellline")
```

The cn1b data contains a high number of values smaller than -1, which lead to a distribution with a right skew and therefore imply a discrepancy between the gaussian distribution and the cn1b data.
To reduce this effect, all rows that obtain a value lower than or equal to -2 are removed from the data set. This limitates the prediction values' range of the regression model. The limit was set after assessing the Q-Q-plot shown below. 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 performed to check the normality of the copy number data set once again and it is determined 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")

# Plot the quantiles of the copy numbers with the quantiles 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 quantiles of the copy numbers with the quantiles 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("co
py 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 for 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 the data set.
After the reduction process, the Q-Q-plot (middle) reveals a better fit for normality, although for copy numbers higher than 1 a strong aberration is recognisable.
The boxplot (right) illustrates the issue of a high number of outliers for each side of the 1,5 IQR (Interquantile range). Having a large number of outliers 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. The anomaly of the samples' distribution can also effect the output of the model. This must be kept in mind as a possible error for the regression model.

6.2 Identification of common genes of both variables

Only genes that are present in both dataframes (list_all.genes, row.names) can be included in the linear regression model. For each copy number value a corresponding expression value must exist, 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,]})
```

Since both matrices only consist 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.

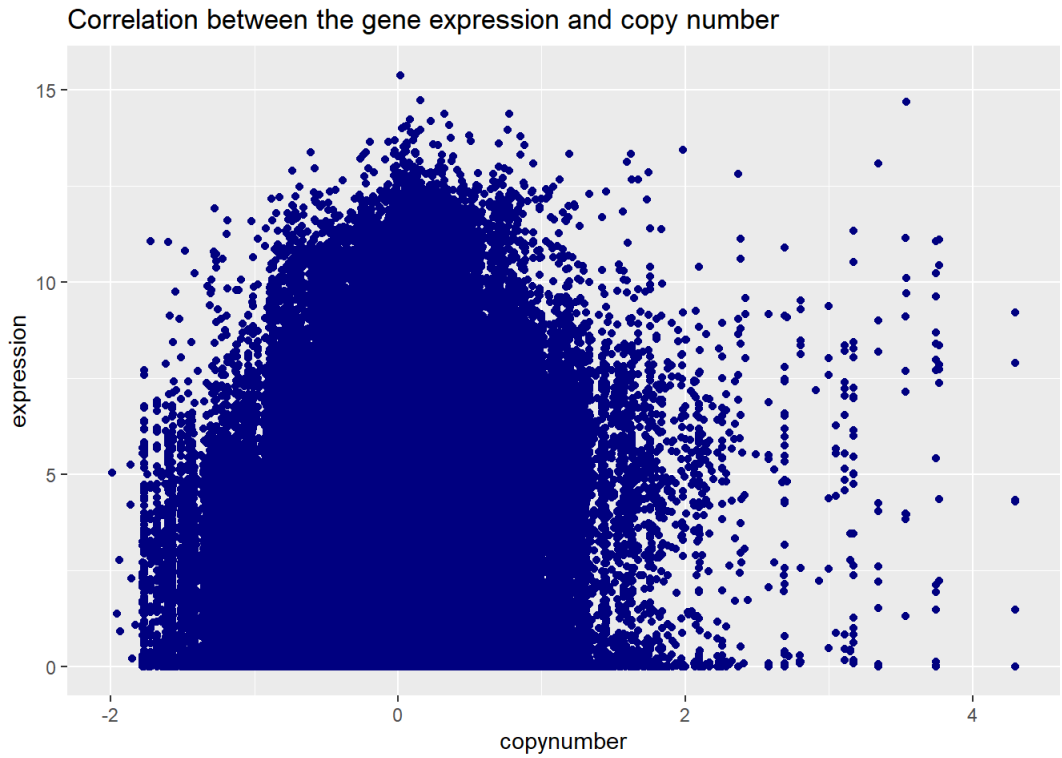
```
# Fuse all expression values to one long column
exp1 <- melt.data.frame(list_all.genes$expression, variable_name = "cell line")

# Fuse all copy number values to one long column
cn1 <- melt.data.frame(list_all.genes$copynumber, variable_name = "cell line")

# 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 the first row corresponds to the respective gene expression value. This way, it is possible to plot 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 whether a linear regression model can be applied for the problem. To obtain a visual overview, the correlation between the two variables is 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="ge
ne expression")
```



The scatter plot reveals only a low visual relationship between the two variables and does not indicate 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 calculating the correlation coefficient. This process decreases the effect of outliers for the coefficient.

```
# Compute Pearson 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 of approximately 0.09 to 0.08 between the two variables. The sligthly lower Spearman correlation shows that the Pearson correlation is possibly influenced by outliers. 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. In order to check that the correlation is not equal to zero and the calculated coefficient is due to a statistical error a t-test for both correlation coefficients is performed. The null hypothesis states that there is no correlation between the copy number and the gene expression, while the alternative hypothesis confirms a correlation between the two variables. As a common level of significance = 0.05 is chosen and a p-value, which is smaller than the level of significance, will lead to the denial of the null hypothesis. First the Pearson 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, because 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 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 tests lead to the conclusion that the calculated correlation coefficients would not be obtained if the real correlation equals zero. 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 be 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 into 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 into approximately 70% for the trainings data and 30% for validation process afterwards.
198920 of 663068 rows are taken randomly from 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
## -5.0190 -2.3536 -0.3857  1.8690 12.8440
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.525298   0.003613  698.97   <2e-16 ***
## copynumber  0.580554   0.009271   62.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.461 on 464146 degrees of freedom
## Multiple R-squared:  0.008378,   Adjusted R-squared:  0.008376
## F-statistic:  3921 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

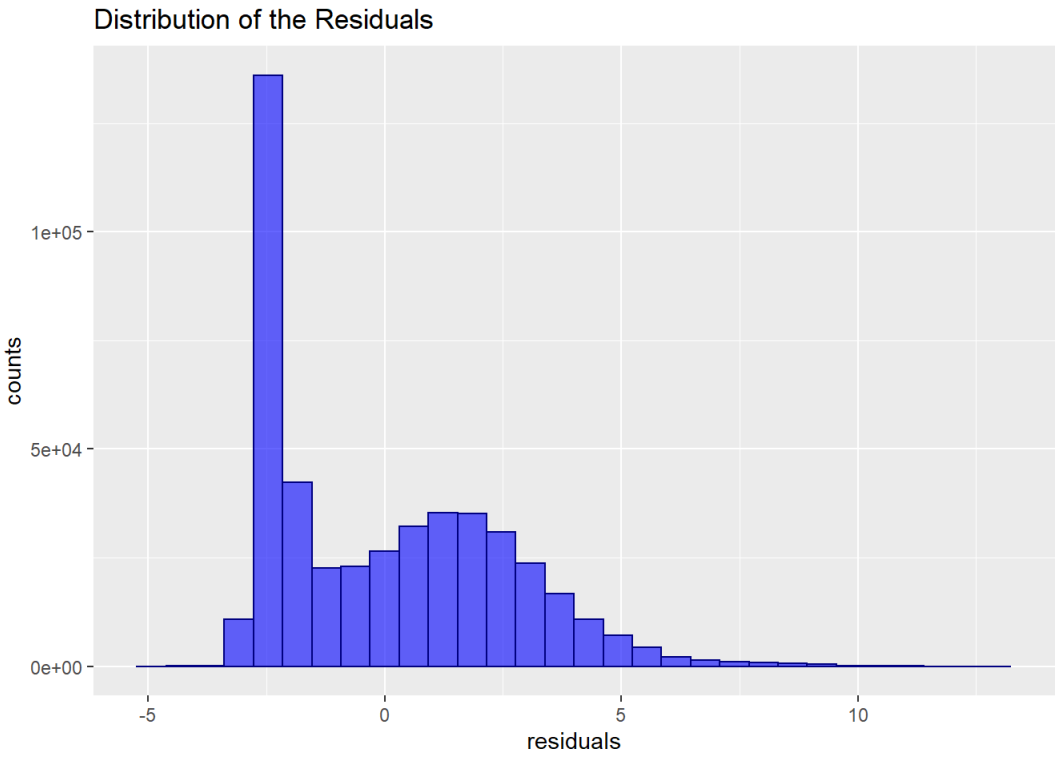
With a p-value smaller than 2e-16 the t-test indicates that the slope of the regression line has a significant aberration from zero.
The standard error for the residuals indicates an average aberration of 2.459 between the predicted expression value and the expression value of the dataset. For a good fit of the model an 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. The value near zero suggests that the variable copy number does not fit to explain the variable gene expression. This possibly results from 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 requirements for a good fit of the regression model will be checked. The first requirement is that the residuals are distributed symmetrically around a median of approximately 0. With a calculated value of -0.3827 a slight deviation from 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 shows a huge abnormality of the Gaussian distribution in the lower tail. Because a histogram 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 whether 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 = 0.34839, 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] -6.454092e-17
```

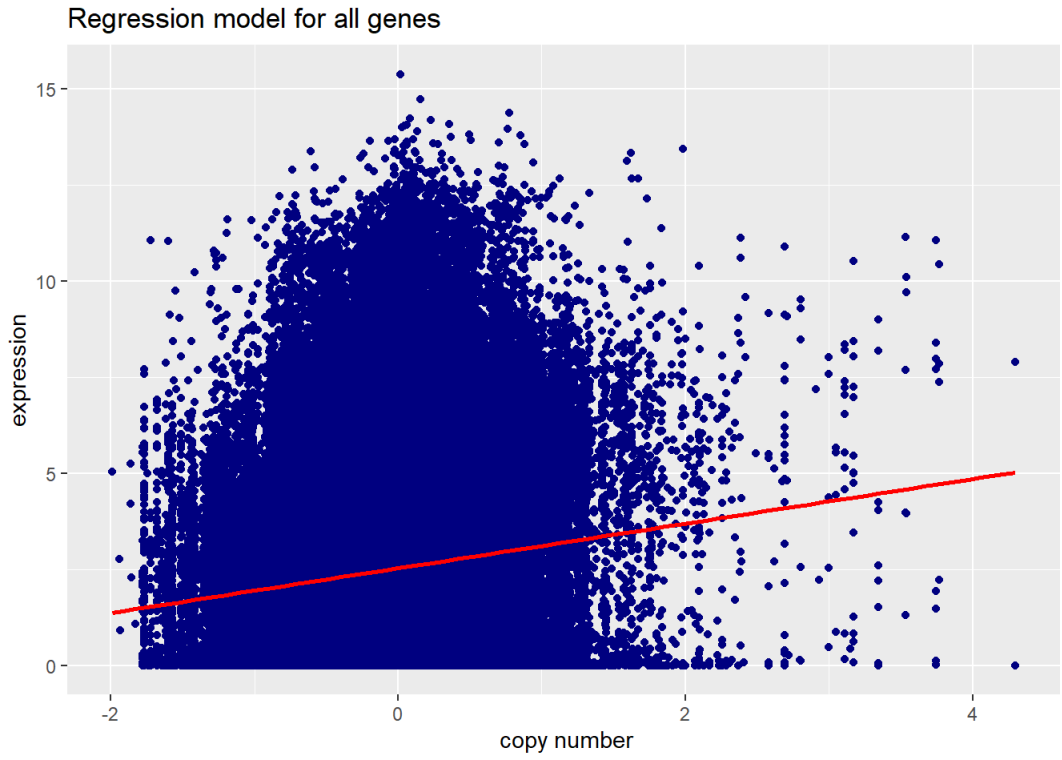
```
# Test the significance of the pearson correlation coefficient
cor.test(regression.all.genes$copynumber, lrm.all.genes$residuals)
```

```
##
##  Pearson's product-moment correlation
##
## data:  regression.all.genes$copynumber and lrm.all.genes$residuals
## t = -4.3971e-14, 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
## -6.454092e-17
```

The correlation between the two variables is equal to zero and therefore does not indicate a 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 analyse 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. In the formula of the standard error n displays the number of rows in the given data frame. To calculate the residuals the difference between

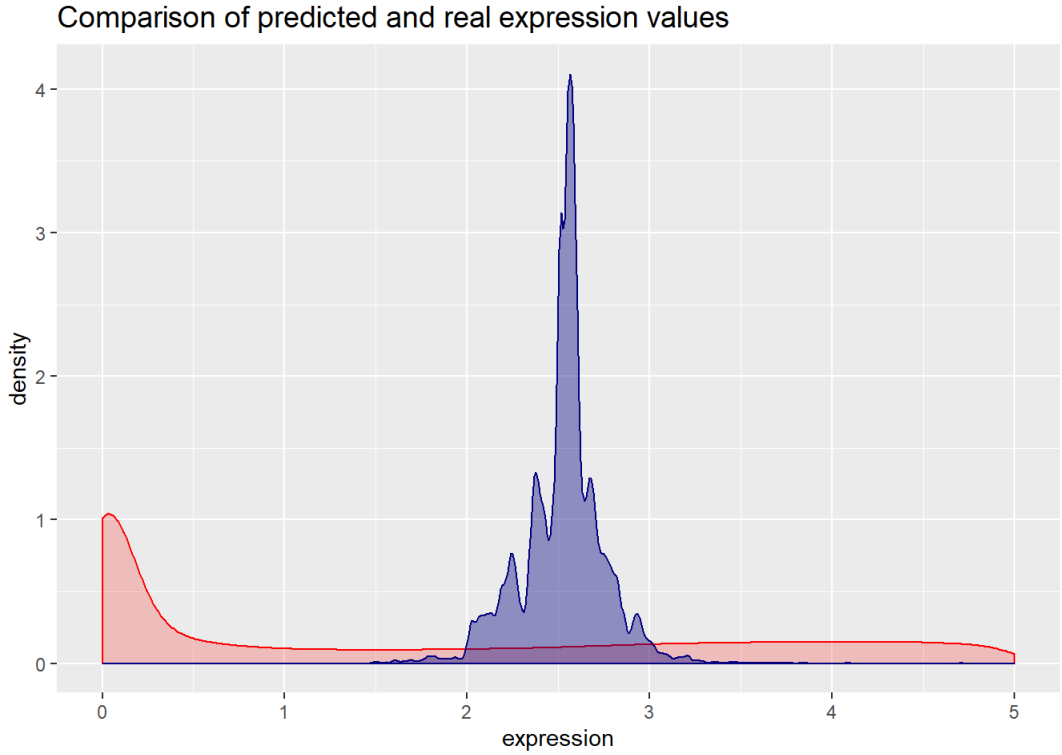
the real value and the predicted value of the gene expression is used.

```
# 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] 2.460661
```

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") + the
me(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

In the first step the most frequent mutations of the data set were identified. The analysis revealed that the mutation of a single gene often occurs more than once in one cell line. After duplicates were removed the tumor suppressor gene TP53 exhibited the highest mutation rate with 12 counts in total, followed by the gene ARID1A with 8 counts. Both tumor suppressors are linked to the development of Ovarian cancer and are known as common driver mutations for this cancer type. With a slightly lower frequency of 5 counts mutation in the genes ATM, BAI1 and PTPRF occurred in the cancer data set.

7.2 Co-existing Mutations

After identifying the driver mutations, it was analysed which mutations occur together most frequently. The results were shown in a heatmap. It was concluded that there is no typical pattern for certain mutation combinations.

7.3 Synthetic Lethality Interactions

By applying a t-test, the most significant SLI partners for the two most common driver mutations TP53 and ARID1A were identified. These are the most likely to exhibit positive results in further experiments and should be investigated thoroughly.

Next, a PCA was performed to determine possible SLI partners for the most common mutations in OC using the knockdown data. Since no clear results could be obtained, a kmeans clustering was performed prior to a second PCA aiming to find subtype-specific SLI partners.

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

Alberts, B., Johnson, A., Lewis, J., Morgan, D., Raff, M., Roberts, K., and Walter, P. (2017). Molekularbiologie der Zelle, 6. Auflage edn (Weinheim: Wiley-VCH).

Berns, K., Caumanns, J.J., Hijmans, E.M., Gennissen, A.M.C., Severson, T.M., Evers, B., Wisman, G.B.A., Jan Meersma, G., Lieftink, C., Beijersbergen, R.L., et al. (2018). ARID1A mutation sensitizes most ovarian clear cell carcinomas to BET inhibitors. Oncogene 37, 4611-4625.

Chao, A., Tsai, C.-L., Jung, S.-M., Chuang, W.-C., Kao, C., Hsu, A., Chen, S.-H., Lin, C.-Y., Lee, Y.-C., Lee, Y.-S., et al. (2015). BAI1-Associated Protein 2-Like 1 (BAIAP2L1) Is a Potential Biomarker in Ovarian Cancer. PloS one 10, e0133081-e0133081.

Doherty, J.A., Rossing, M.A., Cushing-Haugen, K.L., Chen, C., Van Den Berg, D.J., Wu, A.H., Pike, M.C., Ness, R.B., Moysich, K., Chenevix-Trench, G., et al. (2010). ESR1/SYNE1 polymorphism and invasive epithelial ovarian cancer risk: an Ovarian Cancer Association

Consortium study. Cancer Epidemiol Biomarkers Prev 19, 245-250.

Mandilaras, V., Garg, S., Cabanero, M., Tan, Q., Pastrello, C., Burnier, J., Karakasis, K., Wang, L., Dhani, N.C., Butler, M.O., et al. (2019). TP53 mutations in high grade serous ovarian cancer and impact on clinical outcomes: a comparison of next generation sequencing and bioinformatics analyses. International Journal of Gynecologic Cancer 29, 346-352.

O'Neil, N.J., Bailey, M.L., and Hieter, P. (2017). Synthetic lethality and cancer. Nat Rev Genet 18, 613-623.

Stranger, B.E., Forrest, M.S., Dunning, M., Ingle, C.E., Beazley, C., Thorne, N., Redon, R., Bird, C.P., de Grassi, A., Lee, C., et al. (2007). Relative Impact of Nucleotide and Copy Number Variation on Gene Expression Phenotypes. 315, 848-853.

Takeda, T., Banno, K., Okawa, R., Yanokura, M., Iijima, M., Irie-Kunitomi, H., Nakamura, K., Iida, M., Adachi, M., Umene, K., et al. (2016). ARID1A gene mutation in ovarian and endometrial cancers (Review). Oncol Rep 35, 607-613.

Testa, U., Petrucci, E., Pasquini, L., Castelli, G., and Pelosi, E. (2018). Ovarian Cancers: Genetic Abnormalities, Tumor Heterogeneity and Progression, Clonal Evolution and Cancer Stem Cells. Medicines (Basel) 5.

Toss, A., Tomasello, C., Razzaboni, E., Contu, G., Grandi, G., Cagnacci, A., Schilder, R.J., and Cortesi, L. (2015). Hereditary ovarian cancer: not only BRCA 1 and 2 genes. Biomed Res Int 2015, 341723.

Thorstenson, Y.R., Roxas, A., Kroiss, R., Jenkins, M.A., Yu, K.M., Bachrich, T., Muhr, D., Wayne, T.L., Chu, G., Davis, R.W., et al. (2003). Contributions of ***ATM*** Mutations to Familial Breast and Ovarian Cancer. Cancer Research 63, 3325-3333.

Tian, X.a., Yang, C., Yang, L., Sun, Q., and Liu, N. (2018). PTPRF as a novel tumor suppressor through deactivation of ERK1/2 signaling in gastric adenocarcinoma. Onco Targets Ther 11, 7795-7803.