Group 02 - Skin Cancer

Leonie Thomas, Isabel Potthof, Elif Tosun and Marlene Khin

21.07.2019

Preparations

1.Loading following packages

```
library(ggplot2)
library(relaimpo)
library(factoextra)
library(gridExtra)
library(reshape2)
library(data.table)
library(cluster)
library(rstudioapi)
library(pheatmap)
library(caret)
library(tidyverse)
library(dendextend)
library(factoextra)
library(devtools)
library(ggfortify)
library(rstudioapi)
library(data.table)
library(ggplot2)
library(scales)
library(stats)
library(caTools)
```

2. Setting the sys-path and loading the data

The sys-path was used in R, but markdown could not knit it so the data was loaded as explained in step 3.

```
root.dir = dirname(rstudioapi::getSourceEditorContext()$path)
data = readRDS(paste0(root.dir, "/DepMap19Q1_allData.RDS"))
```

3. Loading the data set

```
data = readRDS("C:/Users/LeoTh/Documents/GitHub/project-01-group-
02/DepMap19Q1_allData.RDS")
data$expression[1:10,1:5]
```

```
##          ACH-000004 ACH-000005 ACH-000007 ACH-000009 ACH-000011
## TSPAN6    2.6158871  3.0669502 4.06608919 6.50795317  4.5777309
## TNMD      0.0000000  0.0000000 0.00000000 0.09761080  0.0000000
```

```
## DPM1      5.3233701  5.7626146 5.88996020 7.98162436  5.5357419
## SCYL3     2.4059924  2.9927684 3.04963077 2.24792751  2.0874628
## C1orf112  3.9020736  5.3596617 3.76022095 4.49121176  2.6461627
## FGR       0.9259994  0.2387869 0.02856915 0.00000000  0.0000000
## CFH       4.8889867  5.7001623 0.01435529 0.02856915  0.3561438
## FUCA2     3.8962718  4.1432301 5.56193706 7.08852326  5.5251293
## GCLC      4.8359241  5.3805909 4.97407037 5.72737594  4.3540289
## NFYA      4.9868660  5.3391374 3.99004722 4.63343121  3.1554254

data$copynumber[1:10,1:5]

##            ACH-000004 ACH-000005 ACH-000007 ACH-000009 ACH-000011
## A1BG           0.4322   0.265953  -0.041971    -0.0352   0.131139
## NAT2           0.1197   0.215723  -0.043539    -0.0768  -0.332231
## ADA           -0.3483  -0.389113  -0.031292     1.0902   0.185270
## CDH2           0.1071   0.100735  -0.064250    -0.8614  -0.322219
## AKT3           0.1191   0.163029   0.541094     0.0862   0.284791
## GAGE12F        0.0375  -1.197690 -29.958527     0.4590  -0.299359
## ZBTB11-AS1     0.1022   0.164588  -0.036427    -0.0351  -0.360381
## MED6          -0.3972  -0.381579  -0.069094    -0.3454  -0.350737
## NR2E3          0.1294   0.228166  -0.050321    -0.3594   0.196947
## NAALAD2        0.1385   0.194863  -0.051092     0.9784  -0.375141

data$kd.ceres[1:10,1:5]

##          ACH-000004  ACH-000005   ACH-000007   ACH-000009    ACH-000011
## A1BG     0.13464536 -0.21244506  0.043317923  0.070512000  0.1909349984
## A1CF     0.07553627  0.23312358  0.066837574  0.008429764  0.0839524066
## A2M     -0.14020860  0.04436493 -0.036196515  0.027114196 -0.0007407878
## A2ML1    0.01392843  0.17383724  0.134781001  0.055926773  0.3533751560
## A3GALT2  0.02913103 -0.12438932  0.082995584  0.046325389 -0.0370438478
## A4GALT  -0.14728384 -0.29884901  0.119084008  0.015968267 -0.2058028668
## A4GNT    0.27582919  0.12025981  0.057116006  0.053502301  0.0712754121
## AAAS    -0.36363296 -0.33992528 -0.352541473 -0.498860059 -0.3173102098
## AACS     0.25016463 -0.01130946 -0.005799644  0.110794285  0.0998241033
## AADAC    0.12974343  0.01565951  0.241488251  0.066921220  0.1055390534

data$kd.prob[1:10,1:5]

##            ACH-000004  ACH-000005   ACH-000007  ACH-000009   ACH-000011
## A1BG    0.0024724805 0.106866767 0.0080037212 0.005476636 1.425907e-03
## A1CF    0.0061129164 0.002192881 0.0057571355 0.013869531 6.869958e-03
## A2M     0.0808199476 0.013620180 0.0225154331 0.010608210 2.070112e-02
## A2ML1   0.0143307481 0.003971270 0.0020673543 0.006886869 9.601914e-05
## A3GALT2 0.0117268791 0.057068492 0.0045482949 0.007972490 3.171446e-02
## A4GALT  0.0863246673 0.181176460 0.0026466956 0.012466221 1.576283e-01
## A4GNT   0.0001951991 0.006674315 0.0066110831 0.007143158 8.184101e-03
## AAAS    0.3857305749 0.226019789 0.3509659628 0.654158459 3.256387e-01
## AACS    0.0003231688 0.022459532 0.0154067787 0.002847221 5.510092e-03
## AADAC   0.0026802343 0.017681919 0.0003347122 0.005807022 5.072046e-03

data$annotation[1:10,1:5]
```

```
##     DepMap_ID                                    CCLE_Name
## 4  ACH-000004     HEL_HAEMATOPOIETIC_AND_LYMPHOID_TISSUE
## 5  ACH-000005 HEL9217_HAEMATOPOIETIC_AND_LYMPHOID_TISSUE
## 7  ACH-000007                      LS513_LARGE_INTESTINE
## 8  ACH-000009                    C2BBE1_LARGE_INTESTINE
## 10 ACH-000011                        253J_URINARY_TRACT
## 11 ACH-000012                              HCC827_LUNG
## 12 ACH-000013                           ONCODG1_OVARY
## 13 ACH-000014                             HS294T_SKIN
## 14 ACH-000015                           NCIH1581_LUNG
## 16 ACH-000017                            SKBR3_BREAST
##                   Aliases         Primary.Disease
## 4                     HEL                Leukemia
## 5             HEL 92.1.7                Leukemia
## 7                  LS513 Colon/Colorectal Cancer
## 8                C2BBe1 Colon/Colorectal Cancer
## 10                 253J          Bladder Cancer
## 11               HCC827             Lung Cancer
## 12            ONCO-DG-1          Ovarian Cancer
## 13 Hs 294T;A101D;Hs 294.T             Skin Cancer
## 14    NCI-H1581;NCI-H2077             Lung Cancer
## 16                SK-BR-3           Breast Cancer
##                                          Subtype.Disease
## 4    Acute Myelogenous Leukemia (AML), M6 (Erythroleukemia)
## 5    Acute Myelogenous Leukemia (AML), M6 (Erythroleukemia)
## 7                                          Colon Carcinoma
## 8                                    Colon Adenocarcinoma
## 10                                               Carcinoma
## 11       Non-Small Cell Lung Cancer (NSCLC), Adenocarcinoma
## 12                                         Adenocarcinoma
## 13                                               Melanoma
## 14 Non-Small Cell Lung Cancer (NSCLC), Large Cell Carcinoma
## 16                                               Carcinoma
```

```
data$mutation$`ACH-000004`[1:10,1:5]
```

```
##      V1 Hugo_Symbol Entrez_Gene_Id NCBI_Build Chromosome
##  1: 698      RNF207         388591         37          1
##  2: 699     PLEKHG5          57449         37          1
##  3: 700        PDPN          10630         37          1
##  4: 701       CASP9            842         37          1
##  5: 702     RAP1GAP           5909         37          1
##  6: 703        C1QC            714         37          1
##  7: 704      CNKSR1          10256         37          1
##  8: 705       AHDC1          27245         37          1
##  9: 706     COL16A1           1307         37          1
## 10: 707       CSMD2         114784         37          1
```

1. Data cleanup

1.1 Extracting and splitting our data

Defining a new matrix only containing the mutation data which is structured differently from the other matrices.

```
mut <- data$mutation
```

Additionally, to the mutation matrix another matrix is needed containing all matrices except the mutation data.

```
'%!in%' <- function(x,y)!('%in%'(x,y))  # defining an operator that will only pick the data that is NOT defined in the list; so the data that needs to be excluded
dt_new <- lapply(which(names(data) %!in% "mutation"), function(a) data[[a]])
# extracting the non-mutation data
names(dt_new) <- names(data)[which(names(data) %!in% "mutation")] # renaming the data with the original names
#our data now consists out of 2 lists
names(dt_new)
```

```
## [1] "expression" "copynumber" "kd.ceres"   "kd.prob"    "annotation"
```

```
head(mut[[1]])#just picking one cell line as an example
```

```
##     V1 Hugo_Symbol Entrez_Gene_Id NCBI_Build Chromosome Start_position
## 1: 698      RNF207         388591         37          1        6279339
## 2: 699     PLEKHG5          57449         37          1        6533165
## 3: 700        PDPN          10630         37          1       13940848
## 4: 701       CASP9            842         37          1       15819484
## 5: 702     RAP1GAP           5909         37          1       21924552
## 6: 703        C1QC            714         37          1       22974054
##    End_position Strand Variant_Classification Variant_Type
## 1:      6279339      +      Missense_Mutation          SNP
## 2:      6533165      +      Missense_Mutation          SNP
## 3:     13940848      +      Missense_Mutation          SNP
## 4:     15819484      +      Missense_Mutation          SNP
## 5:     21924552      +      Missense_Mutation          SNP
## 6:     22974054      +                 Silent          SNP
##    Reference_Allele Tumor_Seq_Allele1    dbSNP_RS dbSNP_Val_Status
## 1:                G                 C
## 2:                G                 A rs373198302
## 3:                A                 G
## 4:                C                 G
## 5:                A                 T        <NA>             <NA>
## 6:                G                 C rs369658525
##         Genome_Change Annotation_Transcript Tumor_Sample_Barcode
## 1:  g.chr1:6279339G>C     ENST00000377939.4           ACH-000004
## 2:  g.chr1:6533165G>A     ENST00000400915.3           ACH-000004
## 3: g.chr1:13940848A>G     ENST00000509009.1           ACH-000004
## 4: g.chr1:15819484C>G     ENST00000333868.5           ACH-000004
```

```
## 5: g.chr1:21924552A>T     ENST00000374765.4           ACH-000004
## 6: g.chr1:22974054G>C     ENST00000374639.3           ACH-000004
##    cDNA_Change         Codon_Change Protein_Change isDeleterious
## 1:   c.1777G>C c.(1777-1779)Gag>Cag        p.E593Q         FALSE
## 2:   c.1033C>T c.(1033-1035)Ccc>Tcc        p.P345S         FALSE
## 3:    c.409A>G   c.(409-411)Atc>Gtc        p.I137V         FALSE
## 4:   c.1205G>C c.(1204-1206)gGt>gCt        p.G402A         FALSE
## 5:   c.1885T>A c.(1885-1887)Tct>Act        p.S629T         FALSE
## 6:    c.516G>C   c.(514-516)gcG>gcC        p.A172A         FALSE
##    isTCGAhotspot TCGAhsCnt isCOSMIChotspot COSMIChsCnt   ExAC_AF VA_WES_AC
## 1:         FALSE         0           FALSE           0        NA      <NA>
## 2:         FALSE         0           FALSE           0 4.942e-05      <NA>
## 3:         FALSE         0           FALSE           0        NA      <NA>
## 4:         FALSE         0           FALSE           0        NA      <NA>
## 5:         FALSE         0           FALSE           0        NA      <NA>
## 6:         FALSE         0           FALSE           0 4.118e-05      <NA>
##    CGA_WES_AC SangerWES_AC SangerRecalibWES_AC RNAseq_AC  HC_AC RD_AC
## 1:       <NA>         <NA>               72:69      <NA>   <NA>  <NA>
## 2:       <NA>         <NA>               43:41      <NA>   <NA>  <NA>
## 3:       <NA>         <NA>               52:64      <NA>   <NA>  <NA>
## 4:       <NA>         <NA>              32:162     27:58 26:121  <NA>
## 5:       <NA>         <NA>                 7:8     22:13   <NA>  <NA>
## 6:       <NA>         <NA>               49:58      <NA>   <NA>  <NA>
##    WGS_AC   Variant_annotation  DepMap_ID
## 1:   <NA> other non-conserving ACH-000004
## 2:   <NA> other non-conserving ACH-000004
## 3:   <NA> other non-conserving ACH-000004
## 4:   <NA> other non-conserving ACH-000004
## 5:   <NA> other non-conserving ACH-000004
## 6:   <NA>               silent ACH-000004
```

The next step is to extract the cell lines of the skin cancer. For that we need to get to know the names of the cell lines from the skin cancer, this information we can get out of the annotation dataframe. Then we can create a new dataframe which only contains the data we will work with.

Defining which samples will be taken out of the original dataset.

```
sample_case = c("Skin Cancer")
```

Looking at the annotation matrix and searching only for the primary diseases matching the previous defined sample_case. A vector containing all the cell lines with skin cancer as the primary disease is obtained.

```
samples = data$annotation$DepMap_ID[which(data$annotation$Primary.Disease ==
sample_case)]
```

34 cell lines have the primary disease skin cancer.

Extracting all cell lines defined in the previous step out of the data (except the mutation matrix).

```r
processed_data <- lapply(1:length(dt_new), function(a) { # picking the data
for our sample
  dat_picker <- dt_new[[a]] # picking one file at each iteration
  if(names(dt_new[a])== "annotation"){ # treating the annotations differnetly
because the cell line names are in a colum and are not the columnames like in
the other matrices
    output <- dat_picker[which(dat_picker[,1] %in% samples),]
  } else {
  output <- dat_picker[,which(colnames(dat_picker) %in% samples)]# only
taking the skin cancer cell lines
  output <- output[complete.cases(output),] # only taking rows without NAs
  output <- output[order(rownames(output)),] # reordering the genes according
to their name
  }
  return(output)
})
names(processed_data) <- names(dt_new) # renameing the objects according to
the original data
rm(dt_new,sample_case) # removing objects which are not need anymore
#taking a look at the data:
processed_data$expression[1:10,1:5]
```

```
##           ACH-000014 ACH-000274 ACH-000304 ACH-000322 ACH-000348
## 7SK       0.32996228 0.12548079 0.00000000 0.00000000 0.61939343
## A1BG      6.08661395 0.79908731 5.21179100 5.24716800 5.67694436
## A1BG-AS1  4.33771109 0.34482850 3.82273000 3.76234882 3.92979100
## A1CF      0.05658353 0.08406426 0.01435529 0.04264434 0.00000000
## A2M       6.14425036 7.95006009 6.83605000 6.00360224 4.62993941
## A2M-AS1   0.50589093 1.12432814 0.46466830 0.46466827 0.28688115
## A2ML1     0.23878686 0.01435529 0.00000000 0.00000000 0.01435529
## A2ML1-AS1 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## A2ML1-AS2 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## A2MP1     0.04264434 0.01435529 0.00000000 0.01435529 0.00000000
```

```r
processed_data$copynumber[1:10,1:5]
```

```
##          ACH-000014 ACH-000274 ACH-000304 ACH-000322 ACH-000348
## A1BG         0.0989  -0.097469     0.0184     0.0536     0.2243
## A1BG-AS1     0.0989  -0.097469     0.0184     0.0536     0.2243
## A1CF        -0.3120   0.005770    -0.9286    -0.3620    -0.2004
## A2M          0.1110   0.014809     0.0080     0.0876    -0.1028
## A2M-AS1      0.1110   0.014809     0.0080     0.1893    -0.1028
## A2ML1        0.1110   0.014809     0.0080     0.1893    -0.1028
## A2MP1        0.1110   0.014809     0.0080     0.0876    -0.1028
## A4GALT       0.0580  -0.011931     0.0359     0.2368    -0.1255
## A4GNT        0.0961  -0.001092     0.0207     0.4689     0.2784
## AA06         0.0823   0.017090     0.0315     0.1839    -0.1232
```

```r
processed_data$kd.ceres[1:10,1:5]
```

```
##           ACH-000014  ACH-000274   ACH-000304  ACH-000322  ACH-000348
## A1BG     0.112978995  0.09000078  0.123999117  0.17552003  0.05979116
```

```
## A1CF    -0.035332501  0.13896177  0.274438862  0.08205882  0.07506780
## A2M      0.028806113 -0.07116082 -0.201401893  0.01866713 -0.05790399
## A2ML1    0.169333904  0.10695957  0.291635816  0.20971252  0.09614451
## A3GALT2 -0.003591934 -0.09866711 -0.056745475 -0.37374191 -0.07849708
## A4GALT  -0.084698165 -0.11183837 -0.164706825  0.07185597 -0.05414073
## A4GNT   -0.117547293  0.03983944 -0.005448509 -0.01545300  0.07601372
## AAAS    -0.371033490 -0.41828686 -0.195925758 -0.28411957 -0.32252314
## AACS    -0.045406257  0.02847436 -0.097559375 -0.08118755 -0.01892812
## AADAC    0.138287975  0.16723110  0.147066204  0.38636762  0.04059718
```

```
processed_data$kd.prob[1:10,1:5]
```

```
##          ACH-000014  ACH-000274  ACH-000304   ACH-000322  ACH-000348
## A1BG    0.002774072 0.004409101 0.002713543 9.701586e-04 0.006625887
## A1CF    0.023529328 0.002004631 0.000372957 4.262132e-03 0.005132902
## A2M     0.009873198 0.039221771 0.103075053 1.047207e-02 0.037397252
## A2ML1   0.001095265 0.003375634 0.000306326 5.420935e-04 0.003550176
## A3GALT2 0.015498205 0.053319884 0.025767742 3.809270e-01 0.048325382
## A4GALT  0.042860321 0.061256566 0.075245128 4.955556e-03 0.035654746
## A4GNT   0.061754873 0.009330674 0.014367402 1.639439e-02 0.005049978
## AAAS    0.417853626 0.491584887 0.098458357 2.246814e-01 0.386247431
## AACS    0.026723760 0.010955990 0.039643561 3.609931e-02 0.022159161
## AADAC   0.001837255 0.001242681 0.001980259 1.917526e-05 0.009061951
```

```
processed_data$annotation[1:10,1:5]
```

```
##      DepMap_ID     CCLE_Name                Aliases Primary.Disease
## 13  ACH-000014   HS294T_SKIN Hs 294T;A101D;Hs 294.T     Skin Cancer
## 269 ACH-000274   HS852T_SKIN               Hs 852.T     Skin Cancer
## 299 ACH-000304    WM115_SKIN                 WM-115     Skin Cancer
## 317 ACH-000322    HT144_SKIN                 HT-144     Skin Cancer
## 343 ACH-000348 RPMI7951_SKIN              RPMI-7951     Skin Cancer
## 393 ACH-000401  COLO800_SKIN               COLO-800     Skin Cancer
## 396 ACH-000404   K029AX_SKIN                 K029AX     Skin Cancer
## 418 ACH-000425   UACC62_SKIN                UACC-62     Skin Cancer
## 443 ACH-000450    MELHO_SKIN                 MEL-HO     Skin Cancer
## 451 ACH-000458      CJM_SKIN                    CJM     Skin Cancer
##     Subtype.Disease
## 13         Melanoma
## 269        Melanoma
## 299        Melanoma
## 317        Melanoma
## 343        Melanoma
## 393        Melanoma
## 396        Melanoma
## 418        Melanoma
## 443        Melanoma
## 451        Melanoma
```

```
processed_data$mutation$`ACH-000004`[1:10,1:5]
```

```
## NULL
```

Extracting the previously defined cell lines from the mutation data.

```
ids = which(names(mut) %in% samples)
allDepMap_mutation_SkinCancer = lapply(ids, function(a) {
  mut[[a]]})
rm(mut, ids, data) #tidying
```

Losing the mutations which are not deleterious meaning not interesting to us.

```
allDepMap_mutation_SkinCancer = lapply(1:34, function(a) {

allDepMap_mutation_SkinCancer[[a]][which(allDepMap_mutation_SkinCancer[[a]][,
"isDeleterious"]== TRUE), ]
    })
names(allDepMap_mutation_SkinCancer) <- samples
```

Losing all genes which are not in every data frame. First, all gene names have to be picked out of the data.

```
Genenames <-
unique(c(rownames(processed_data[[1]]),rownames(processed_data[[2]]),rownames
(processed_data[[3]]),rownames(processed_data[[4]])))
```

Then picking these genes which are in all 4 data frames which are needed for further analysis.

```
i <- 1
out <- vector("character", length(seq_along(1:16970)))# length of the matrix
depending on how many Genes we have which are in every data frame
for (x in seq_along(Genenames)) {
  if(Genenames[x] %in% rownames(processed_data$expression) & Genenames[x]
%in% rownames(processed_data$copynumber) & Genenames[x] %in%
rownames(processed_data$kd.ceres) & Genenames[x] %in%
rownames(processed_data$kd.prob))
  {out[i] <- Genenames[x]
  i <- i+1
  }
}

allDepMap_annotation_SkinCancer <- processed_data$annotation # saving the
annotation object in a seperate dataframe
# because it doesnt contain any information about the genes

processed_data <- lapply(processed_data[1:4], function(a) {
  a <- a[which(rownames(a) %in% out),]
  return(a)
})

processed_data$mutation <- allDepMap_mutation_SkinCancer
processed_data$annotation <- allDepMap_annotation_SkinCancer
rm(i,out, Genenames,x, allDepMap_annotation_SkinCancer, samples,
allDepMap_mutation_SkinCancer)
```

Looking at the processed data.

```
processed_data$expression[1:10,1:5]
```

```
##          ACH-000014 ACH-000274 ACH-000304 ACH-000322 ACH-000348
## A1BG     6.08661395 0.79908731 5.21179100 5.24716800 5.67694436
## A1CF     0.05658353 0.08406426 0.01435529 0.04264434 0.00000000
## A2M      6.14425036 7.95006009 6.83605000 6.00360224 4.62993941
## A2ML1    0.23878686 0.01435529 0.00000000 0.00000000 0.01435529
## A4GALT   0.97819563 0.02856915 0.12432810 0.98550043 3.09592442
## A4GNT    0.07038933 0.11103131 0.00000000 0.04264434 0.09761080
## AAAS     5.97613447 5.61087720 5.55642900 5.72737594 5.85424505
## AACS     5.13463167 5.08788710 4.21645500 4.14241344 4.17552460
## AADAC    0.01435529 0.21412481 0.05658353 0.07038933 0.33342373
## AADACL2 0.02856915 0.00000000 0.00000000 0.00000000 0.02856915
```

```
processed_data$copynumber[1:10,1:5]
```

```
##          ACH-000014 ACH-000274 ACH-000304 ACH-000322 ACH-000348
## A1BG         0.0989  -0.097469     0.0184     0.0536     0.2243
## A1CF        -0.3120   0.005770    -0.9286    -0.3620    -0.2004
## A2M          0.1110   0.014809     0.0080     0.0876    -0.1028
## A2ML1        0.1110   0.014809     0.0080     0.1893    -0.1028
## A4GALT       0.0580  -0.011931     0.0359     0.2368    -0.1255
## A4GNT        0.0961  -0.001092     0.0207     0.4689     0.2784
## AAAS         0.1107   0.014809     0.0115    -0.0250    -0.1559
## AACS         0.1120   0.014809     0.0340     0.0906    -0.1817
## AADAC        0.0768   0.015465     0.0304     0.4689     0.2784
## AADACL2      0.0768   0.015465     0.0304     0.4689     0.2784
```

```
processed_data$kd.ceres[1:10,1:5]
```

```
##           ACH-000014  ACH-000274   ACH-000304  ACH-000322  ACH-000348
## A1BG      0.11297899  0.09000078  0.123999117  0.17552003  0.05979116
## A1CF     -0.03533250  0.13896177  0.274438862  0.08205882  0.07506780
## A2M       0.02880611 -0.07116082 -0.201401893  0.01866713 -0.05790399
## A2ML1     0.16933390  0.10695957  0.291635816  0.20971252  0.09614451
## A4GALT   -0.08469817 -0.11183837 -0.164706825  0.07185597 -0.05414073
## A4GNT    -0.11754729  0.03983944 -0.005448509 -0.01545300  0.07601372
## AAAS     -0.37103349 -0.41828686 -0.195925758 -0.28411957 -0.32252314
## AACS     -0.04540626  0.02847436 -0.097559375 -0.08118755 -0.01892812
## AADAC     0.13828797  0.16723110  0.147066204  0.38636762  0.04059718
## AADACL2   0.07158946  0.08736997  0.084883216  0.23972258  0.09690599
```

```
processed_data$kd.prob[1:10,1:5]
```

```
##           ACH-000014  ACH-000274  ACH-000304   ACH-000322  ACH-000348
## A1BG     0.002774072 0.004409101 0.002713543 9.701586e-04 0.006625887
## A1CF     0.023529328 0.002004631 0.000372957 4.262132e-03 0.005132902
## A2M      0.009873198 0.039221771 0.103075053 1.047207e-02 0.037397252
## A2ML1    0.001095265 0.003375634 0.000306326 5.420935e-04 0.003550176
## A4GALT   0.042860321 0.061256566 0.075245128 4.955556e-03 0.035654746
## A4GNT    0.061754873 0.009330674 0.014367402 1.639439e-02 0.005049978
```

```
## AAAS    0.417853626 0.491584887 0.098458357 2.246814e-01 0.386247431
## AACS    0.026723760 0.010955990 0.039643561 3.609931e-02 0.022159161
## AADAC   0.001837255 0.001242681 0.001980259 1.917526e-05 0.009061951
## AADACL2 0.005275945 0.004588671 0.004600816 3.200669e-04 0.003501290
```

```
processed_data$annotation[1:10,1:5]
```

```
##      DepMap_ID     CCLE_Name                Aliases Primary.Disease
## 13  ACH-000014   HS294T_SKIN Hs 294T;A101D;Hs 294.T     Skin Cancer
## 269 ACH-000274   HS852T_SKIN               Hs 852.T     Skin Cancer
## 299 ACH-000304    WM115_SKIN                 WM-115     Skin Cancer
## 317 ACH-000322    HT144_SKIN                 HT-144     Skin Cancer
## 343 ACH-000348 RPMI7951_SKIN              RPMI-7951     Skin Cancer
## 393 ACH-000401  COLO800_SKIN               COLO-800     Skin Cancer
## 396 ACH-000404   K029AX_SKIN                 K029AX     Skin Cancer
## 418 ACH-000425   UACC62_SKIN                UACC-62     Skin Cancer
## 443 ACH-000450    MELHO_SKIN                 MEL-HO     Skin Cancer
## 451 ACH-000458      CJM_SKIN                    CJM     Skin Cancer
##     Subtype.Disease
## 13         Melanoma
## 269        Melanoma
## 299        Melanoma
## 317        Melanoma
## 343        Melanoma
## 393        Melanoma
## 396        Melanoma
## 418        Melanoma
## 443        Melanoma
## 451        Melanoma
```

```
processed_data$mutation$`ACH-000004`[1:10,1:5]
```

```
## NULL
```

2. Data visualization

2.1 Preparing our data for plotting

2.1.1 Extracting our data for plotting

Not all the data is needed for plotting so the data is prepared for the following plots.

```
generalPlottingData <- lapply(1:(length(processed_data)-2), function(a) { #
the annotation matrix is not needed
  dtPicker <- processed_data[[a]]
  out <- melt(dtPicker) # binding the data togehter that it has samples and
values as columns
  out$Gene <- rep(rownames(dtPicker), ncol(dtPicker)) # adding the genes;
probably this might be useful in a later stage
  out$Case <- names(processed_data)[1:(length(processed_data)-1)][a]# adding
a labelling column
  colnames(out) <- c("Sample", "Value", "Gene", "Case") # renameing the
columns
  return(out)
})

## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables

names(generalPlottingData) <-
names(processed_data)[1:(length(processed_data)-2)] # renameing the data
```

2.1.2 Plotting Data - Driver Mutations

Producing a vector encompasing every gene which at least mutated once.

```
singleGenes <-
as.vector(unique(as.data.frame(rbindlist(lapply(seq_along(processed_data$muta
tion), function(a) {
  out <-
as.data.frame(as.vector(unique(processed_data$mutation[[a]]$Hugo_Symbol)))}))
))[,1])
```

Creating a data frame containing the mutation rate of every gene.

```
geneCounts <- sapply(seq_along(singleGenes), function(a) {
  genePicker <- singleGenes[a] # picking one gene
  sumGene <- lapply(seq_along(processed_data$mutation), function(b) {
    mutPicker <- processed_data$mutation[[b]] # picking one of the 34
mutation lists
    out <- as.data.frame(length(which(mutPicker$Hugo_Symbol == genePicker)))
# looking how often an entry is in the mutation list
    return(out)
  })
```

```
  geneCount <- colSums(as.data.frame(rbindlist(sumGene))) # summing it up to
get the total count for each gene
  return(geneCount)
})
names(geneCounts) <- singleGenes # renameing
geneCounts <- as.data.frame(geneCounts) # creating a nice data frame
colnames(geneCounts) <- c("Value")
geneCounts <- geneCounts[order(-geneCounts$Value), , drop = FALSE] # sorting
the data frame
head(geneCounts)
```

```
##         Value
## TTN        13
## TP53        9
## HMCN1       8
## TMTC2       7
## RYR2        7
## CACNA1I     7
```

Extacting the data for the top 10 which will be our driver mutations in the further investigation.

```
dataTopDriverGenes <- lapply(1:(length(processed_data)-2), function(a) { #
picking the data for our sample
  dat_picker <- processed_data[[a]] # picking one file at each iteration
  output <- dat_picker[which(rownames(dat_picker) %in%
rownames(geneCounts)[1:10]),] # comparing the rownames of the picked data
with the names of the 10 most mutated genes
  return(output)
})
names(dataTopDriverGenes) <- names(processed_data)[1:4]

rm(singleGenes)
```

2.1.3 Extracting the drivermutations for every cell line

Putting all mutation data in one matrix.

```
oneMatrix <- data.frame()
for (i in c(1:34)) { # 34 is the number of cell lines of interest
  oneMatrix <-
rbind(oneMatrix,processed_data$mutation[[i]][,Hugo_Symbol:DepMap_ID])
}
```

Extracting just the column of the gene name and the cell line.

```
celllinesMutations <- oneMatrix[which(oneMatrix$Hugo_Symbol %in%
rownames(geneCounts)[1:10] ),]
celllinesMutations <- cbind(celllinesMutations$Hugo_Symbol,
celllinesMutations$DepMap_ID)
View(celllinesMutations)
```

Extracting the drivermuations for every cell line out of the data frame and putting it into another data frame so it can be used for plotting.

```
Genes <- c("COL11A1,TMTC2,TTN", " HMCN1", "COL11A1,HMCN1,SLC510",
"HMCN1,TMTC2", "COL11A1,TP53,TTN","none","ZNF292","RYR2","HMCN"
,"none2","none3", "TP53, TTN","HMCN1", "TTN,ZNF292","TMTC2,TP53,NEB","TP53",
"TMTC2,NEB","none4","TMTC2,TTN,ZNF292",
"none5","CACNA1I","HMCN1,TP53,ZNF292","none6","none7","HMCN1,TMTC2,ZNF292","R
YR2,TMTC2,NEB","RYR2,NEB,TTN,CACNA1I","HMCM1,TP53","TTN","COL11A1,SLC5A10","C
OL11A1,CACNA1I","TTN,CACNA1I","RYR2,CACNA1I,ZNF292","TP53,TTN,CACNA1I" )
celllines <- c(colnames(processed_data$expression))
cellinesMutations <- as.data.frame(cbind(celllines, Genes))

rm(oneMatrix, Genes,celllines,i)
```

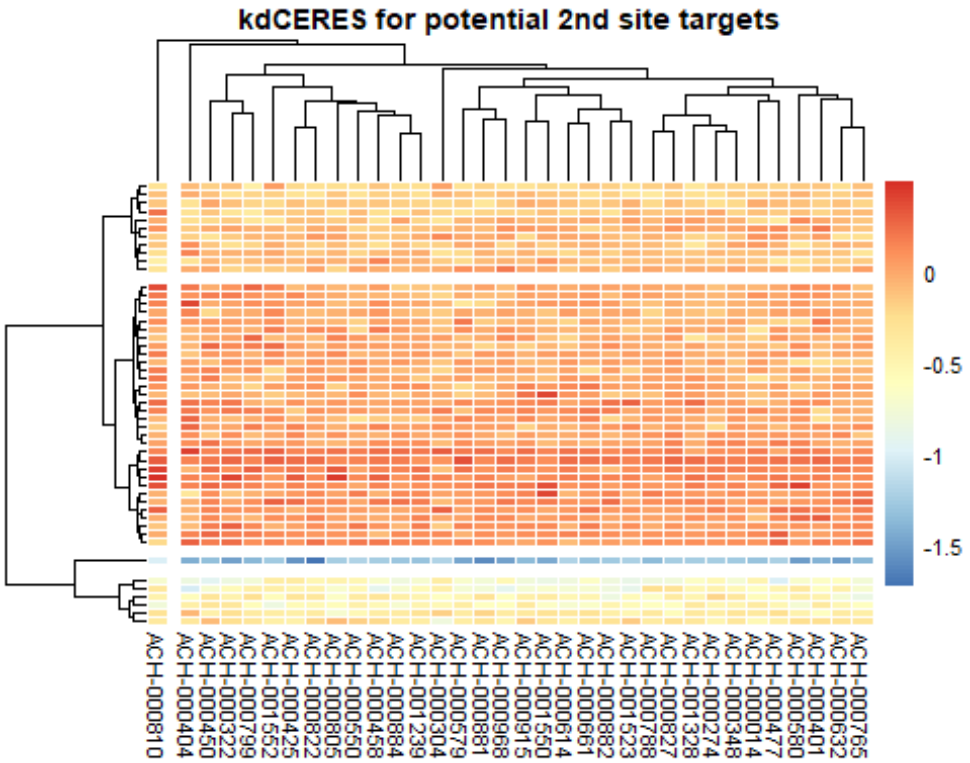
The explanation for the previous extraction will be outlined in the following visualization part.

2.2 Visualizing our data

2.2.1 Heatmap with the knock down data

Starting with a heatmap of the knock down data (the kd.ceres matrix). This matrix consists of gene knockdown scores. The impact of the knocked out gene on the cell survival is reflected by that score. The impact can be a reduction or an increase in proliferation. It could also mean that there is no change in cell proliferation at all. Smaller values refer to higher importance. Useing only the first 50 genes because otherwise the computer was overchallenged and could not produce the heatmap.

```
pheatmap(as.matrix(processed_data$kd.ceres[1:50,]), clustering_method =
"ward.D2",border_color = "white", fontsize = 8,
         main = paste0("kdCERES for potential 2nd site targets"),
         show_rownames = F, show_colnames = T,
         cutree_rows = 4,
         cutree_cols = 2,
         fontsize_row=8)
```



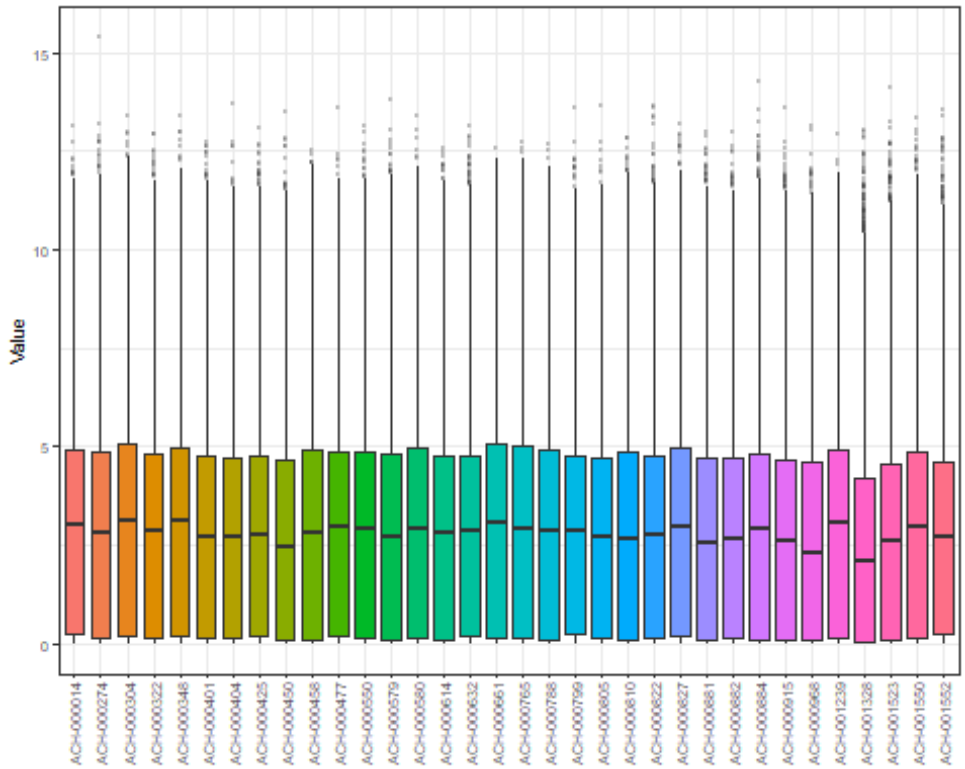
- *There are clear differences between the knockdown data depending on the knocked out gene in a specific cell line.*
- *The cell lines behave differently when the same gene is knocked out.*
- *This means there are genes that are important for cell proliferation and could play a role in cancer development.*

2.2.2 Distribution of the expression values between the different cell lines

Creating a boxplot with the expression matrix to see how the expression of the genes is distributed over the different cell lines.

```
data <- generalPlottingData$expression

ggplot(data, aes(x=Sample, y= Value)) +
  geom_boxplot(aes(fill = Sample), outlier.size = 0.1, outlier.alpha = 0.2) +
# reconstructing the outliers a bit (reduce them in size; because we are
interested in the boxplots and not the outliers)
  theme_bw(base_size = 7) + # formating the size of the theme nicely
  theme(legend.position= "none", # defining the legend position (here no
leghend will be needed)
        legend.direction="horizontal", #define the legend direction if one is
there
        plot.title = element_text(hjust = 0.5), # making the title of the
plot into the middle
```

```
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1), #
defining the orientation of the text on the x-axis
        legend.title= element_blank(), # no title of the legend should be
plotted
        axis.title.x = element_blank(), # no title of the x-axis is relevant;
because that would be samples and that is cleare due to the naming
        strip.text.y = element_text(angle = 90)) # defining the orientation
of the text of the y-axis
```



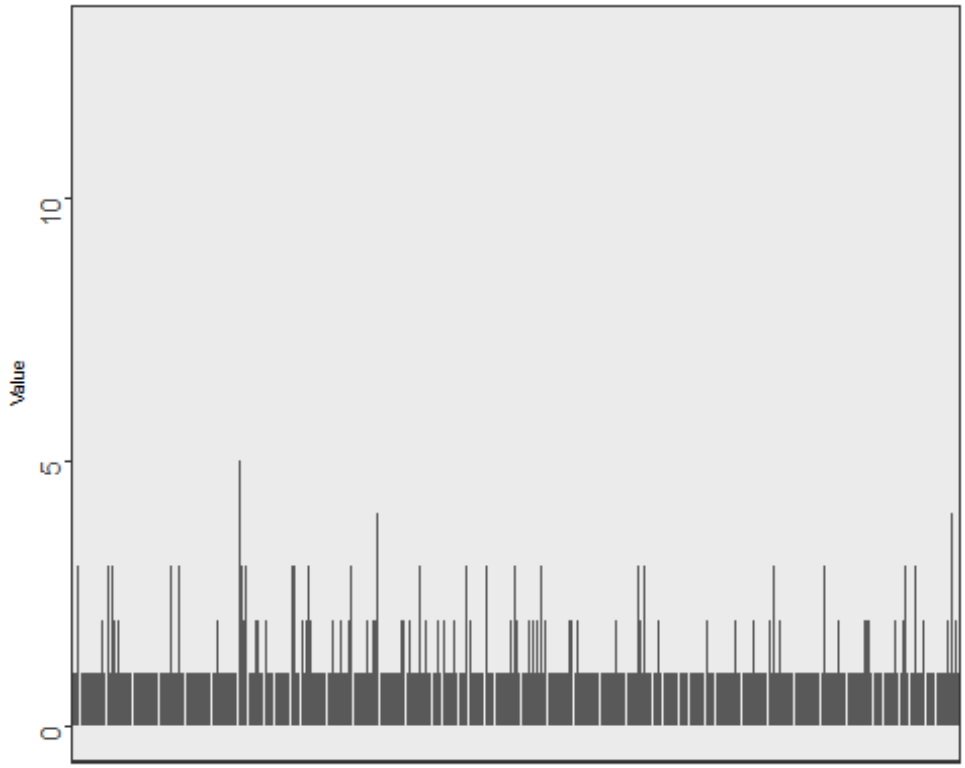
- *Many genes are distributed between the 25 and 75 quantile. But there are also some outliers which are of special interest in the following data analysis.*
- *For now we can say that the data is differnetly distributed between the celllines based on different mutations in the different cell lines.*

2.2.3 Top 10 mutated genes

In the Data extraction part we extracted the genecounts for every mutation. Firstly we want to take a general look at the distribution of the mutation number of every gene over all cell lines:

```
plotData <- geneCounts

plotData$Gene <- rownames(plotData)

ggplot(data = plotData) +
```

```
  (geom_bar(mapping = aes(x = Gene, y = Value), stat = "identity")) +
  theme_bw(base_size = 7) + # formating the size of the theme nicely
  theme(legend.position= "none", # defining the legend position (here no
legend will be needed)
        legend.direction="horizontal", # defining the legend direction if one
is there
        plot.title = element_text(hjust = 0.5), # making the title of the
plot into the middle
        axis.text.x = element_blank(), # defining the orientation of the text
on the x-axis
        axis.text.y = element_text(angle = 90, vjust = 0.5, hjust=1, size =
10), # defining the orientation of the text on the x-axis
        legend.title= element_blank(), # no title of the legend should be
plotted
        axis.title.x = element_blank(), # no title of the x-axis is relevant;
because that would be samples and that is cleare due to the naming
        strip.text.y = element_text(angle = 90)) # defining the orientation
of the text of the y-axis
```

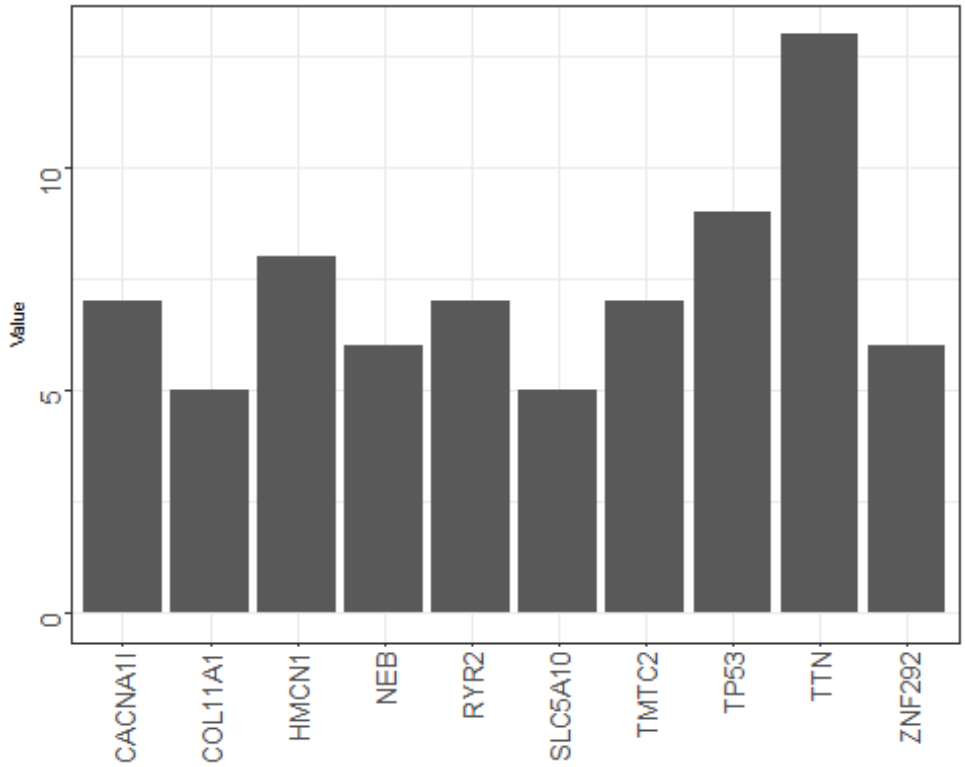



```
rm(plotData)
```

Now we want to see which mutations are the top 10 mutated genes.

```
plotData <- geneCounts[1:10, ,drop = FALSE]

plotData$Gene <- rownames(plotData)

ggplot(data = plotData) +
```

```
  (geom_bar(mapping = aes(x = Gene, y = Value), stat = "identity")) +
  theme_bw(base_size = 7) + # formating the size of the theme nicely
  theme(legend.position= "none", # defining the legend position (here no
legend will be needed)
        legend.direction="horizontal", # defining the legend direction if one
is there
        plot.title = element_text(hjust = 0.5), # making the title of the
plot into the middle
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, size =
10), # defining the orientation of the text on the x-axis
        axis.text.y = element_text(angle = 90, vjust = 0.5, hjust=1, size =
10), # defining the orientation of the text on the x-axis
        legend.title= element_blank(), # no title of the legend should be
plotted
        axis.title.x = element_blank(), # no title of the x-axis is relevant;
because that would be samples and that is cleare due to the naming
        strip.text.y = element_text(angle = 90)) # defining the orientation
of the text of the y-axis
```




```
rm(plotData)
```

- *The expected driver mutations are BRAF, RAS, NF1 and Triple-WT, because they are specific for cutaneous melanoma (1).*
- *The barplot does not mention any of the expected ones, so in the end an analysis of the biological background is needed.*

---

3. Dimensionality reduction

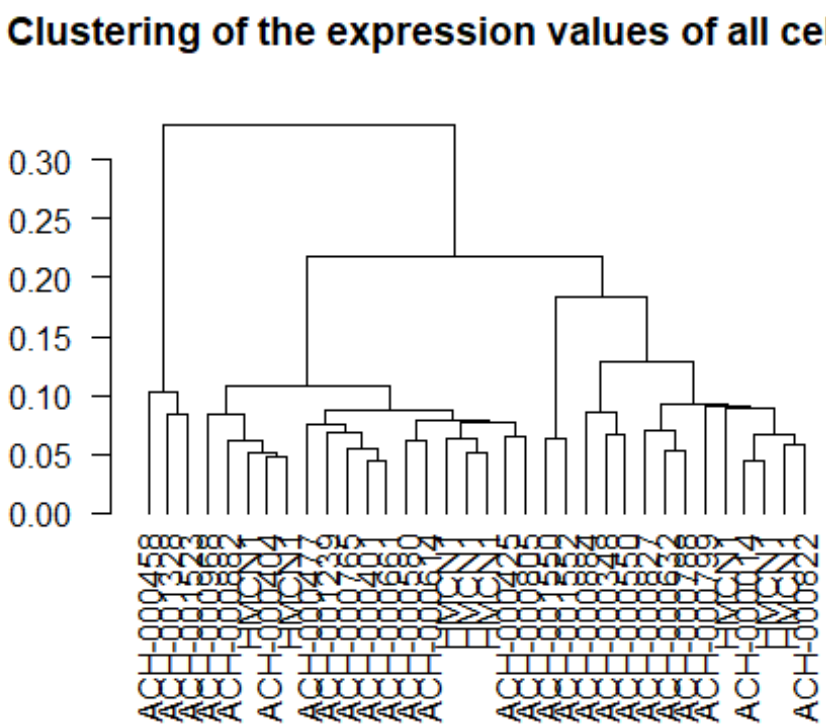
General questions:

- Can we group the different driver mutations together so that we can see in which other genes the cell lines with a specific driver mutation differentiate?
- With Dimnsionality reduction we could gain insight which other genes are our secound targets.

3.1 Hierachical clustering

Creating a hierachical cluster with our driver mutations.

```
drivergene <- 3
# determines which of the driver mutations will be seen in the cluster at the
x axis
dataset <- processed_data$expression # determines which dataset we use

colnames(dataset)[which(colnames(dataset) %in%
unique(celllinesMutations[which(celllinesMutations[,1] ==
rownames(geneCounts)[drivergene]),2]))] <- rownames(geneCounts)[drivergene]
# setting the colnames of the cell lines which have the drivermutation
entered in the dirvermutation variable , to this drivermutation so we can see
if these cell lines cluster together
# drivermutation 3 is just an example can set every drivergene of interest


cor.mat = cor(dataset, method = "spearman")
cor.dist = as.dist(1 - cor.mat)
cor.hc = hclust(cor.dist, method = "ward.D2")
cor.hc = as.dendrogram(cor.hc)
plot(cor.hc, las = 2, cex.lab = 1, main = "Clustering of the expression
values of all cell lines")
```




```
rm(drivergene, realcelllinenames, dataset, cor.hc, cor.mat, cor.dist)

## Warning in rm(drivergene, realcelllinenames, dataset, cor.hc, cor.mat,
## cor.dist): Objekt 'realcelllinenames' nicht gefunden
```

3.2 K-means

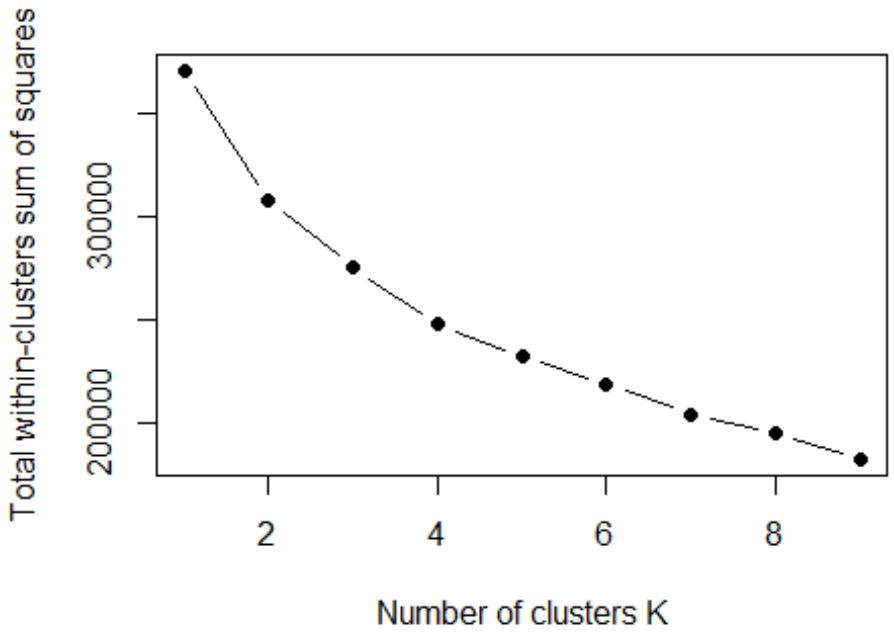
Performing a k-means to identify the structure of our clusters.

```
dataset <- t(processed_data$expression[-
which(rownames(processed_data$expression) %in% rownames(geneCounts)[1:10]),])
# determining which dataset we use
# trying to cluster the cell lines with the same driver mutations in the same
cluster according to the
# expression data without the expression of the driver mutations
# Searching for the cause of the diffences between the cell lines besides the
expression of the driver mutations

rownames(dataset) <- cellinesMutations$Genes

dataset <- dataset[,-which(apply(dataset, 2, function(x) {
  var(x)
}) == 0)]
```

For choosing the best number centers for the clusters the kink method was used.

```
wss = sapply(1:9, function(k) {
  kmeans(x = dataset, centers = k)$tot.withinss
```

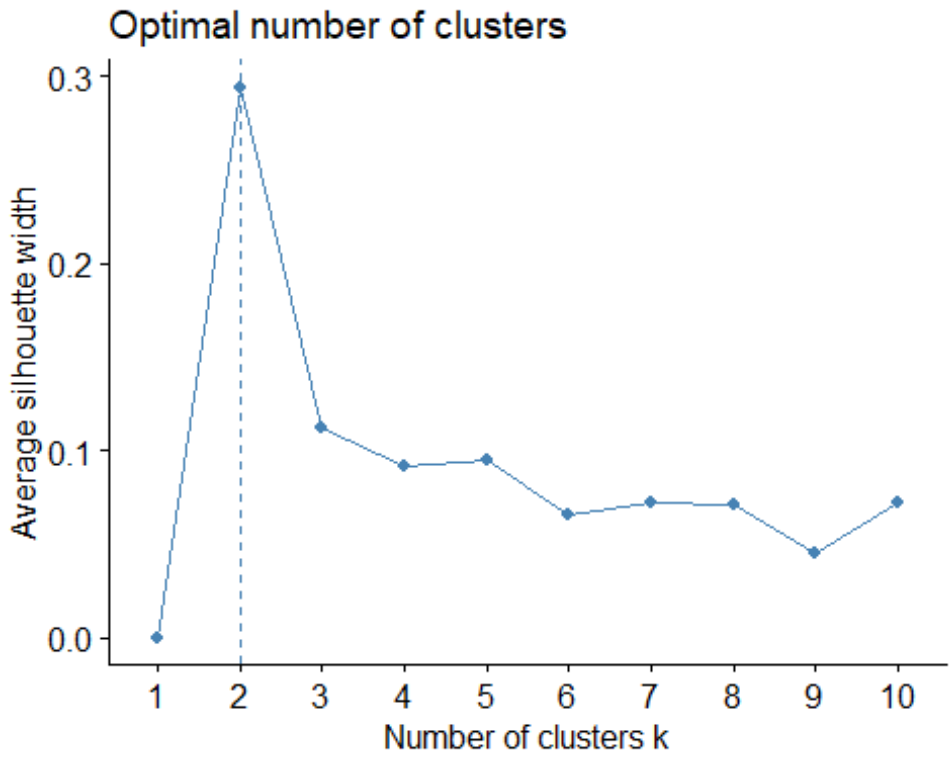
```
})
plot(1:9, wss, type = "b", pch = 19, xlab = "Number of clusters K", ylab =
"Total within-clusters sum of squares")
```



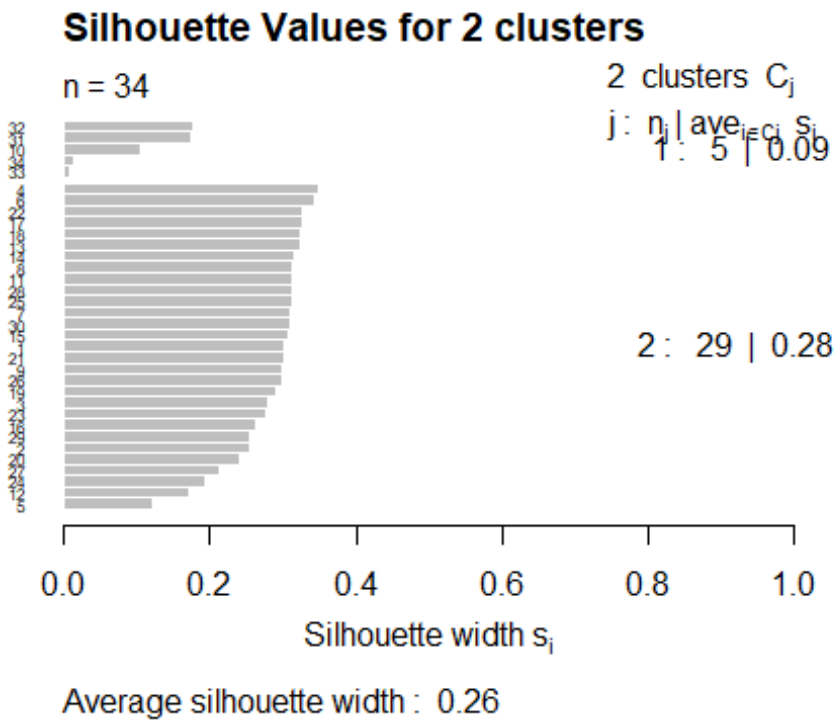
- *But theres no kink in this curve so we need to use other methods to tell us how much centers would be best to choose.*

Now we try the silouette method.

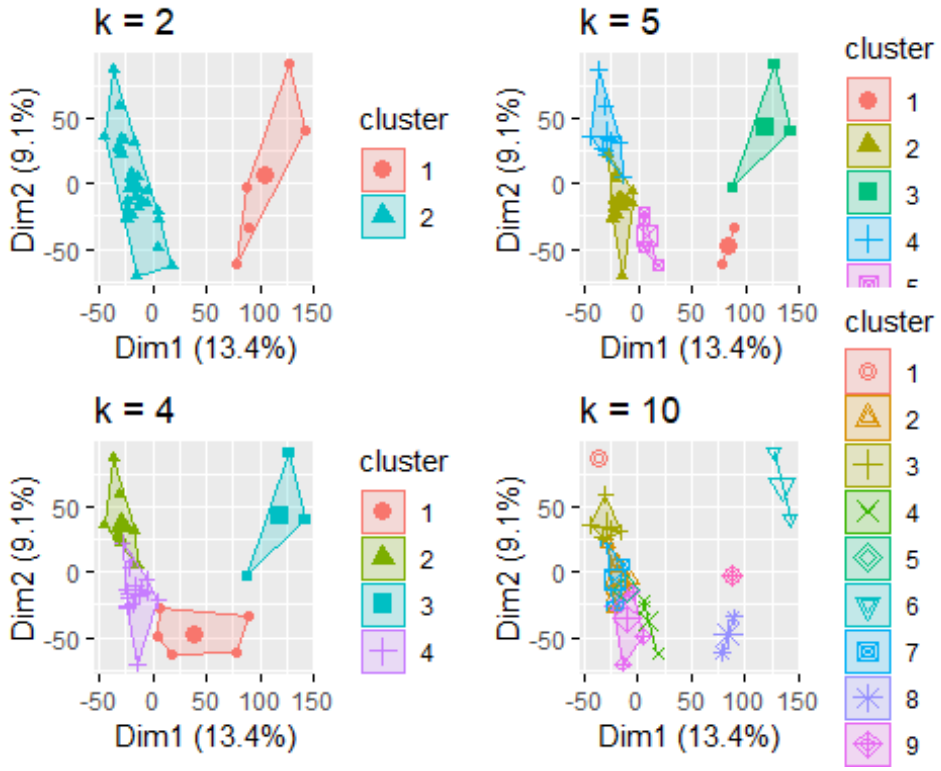
```
fviz_nbclust(dataset, kmeans, method = "silhouette")
```



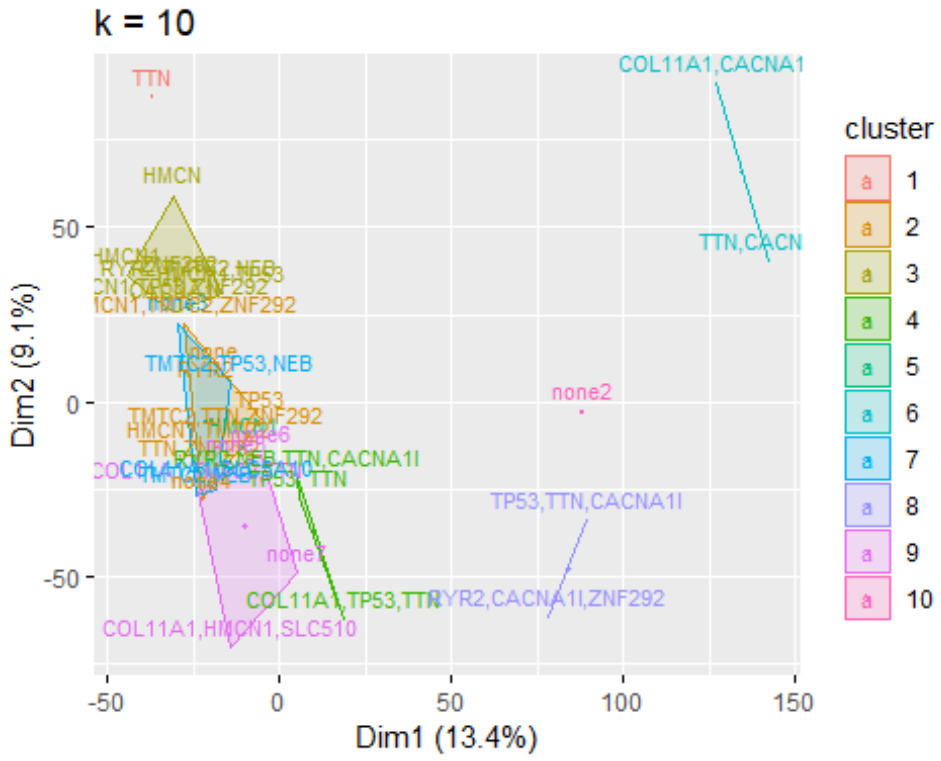

```r
# according to the silhouette method the clustering with two centers seems to be the best one

# taking a look at the clustering with different centers (2, 4, 5, 10)
km = kmeans(x =dataset, centers = 2, nstart = 100)
plot(silhouette(km$cluster,dist(dataset)), main = "Silhouette Values for 2 clusters", cex=0.5)
```



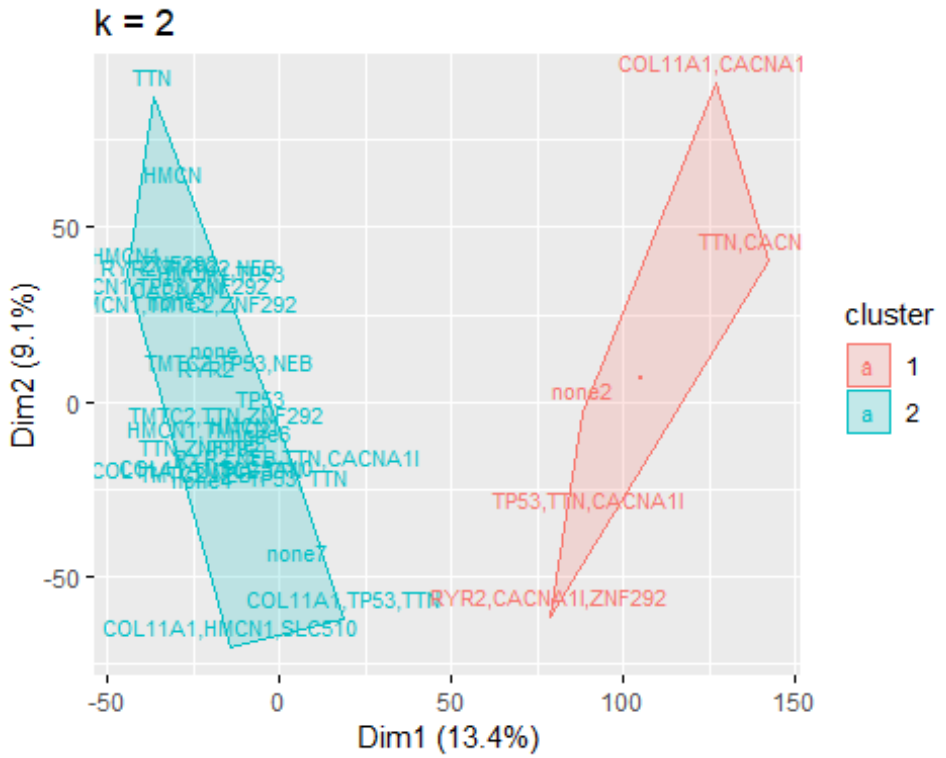

```r
km2 <- kmeans(dataset, centers = 2, nstart = 100)
km3 <- kmeans(dataset, centers = 5, nstart = 100)
km4 <- kmeans(dataset, centers = 4, nstart = 100)
km5 <- kmeans(dataset, centers = 10, nstart = 100)



p1 <- fviz_cluster(km2,geom = "point", data = dataset) + ggtitle("k = 2")
p2 <- fviz_cluster(km3, geom = "point", data = dataset) + ggtitle("k = 5")
p3 <- fviz_cluster(km4, geom = "point", data = dataset) + ggtitle("k = 4")
p4 <- fviz_cluster(km5, geom = "point", data = dataset) + ggtitle("k = 10")

grid.arrange(p1, p2, p3, p4, nrow = 2)
```




```
p4 <- fviz_cluster(km5, geom = "text", labelsize = 9, data = dataset) +
ggtitle("k = 10")
plot(p4) # clustering with 10 centers does not conclude in clusters with the
same driver mutations
```



```
# having more than one driver mutation in most cell lines may cause this
p1 <- fviz_cluster(km2,geom = "text", labelsize = 9, data = dataset) +
ggtitle("k = 2")
plot(p1)
```




```
rm(km,km2,km3,km4,km5,p1,p2,p3,p4, dataset,wss)
```

- *The clustering with two centers seems to be the best one.*
- *Our next step in the pca will be to see which of the genes drive the differentation of the celllines in this plot because they will be the most variable and thus most interesting ones.*

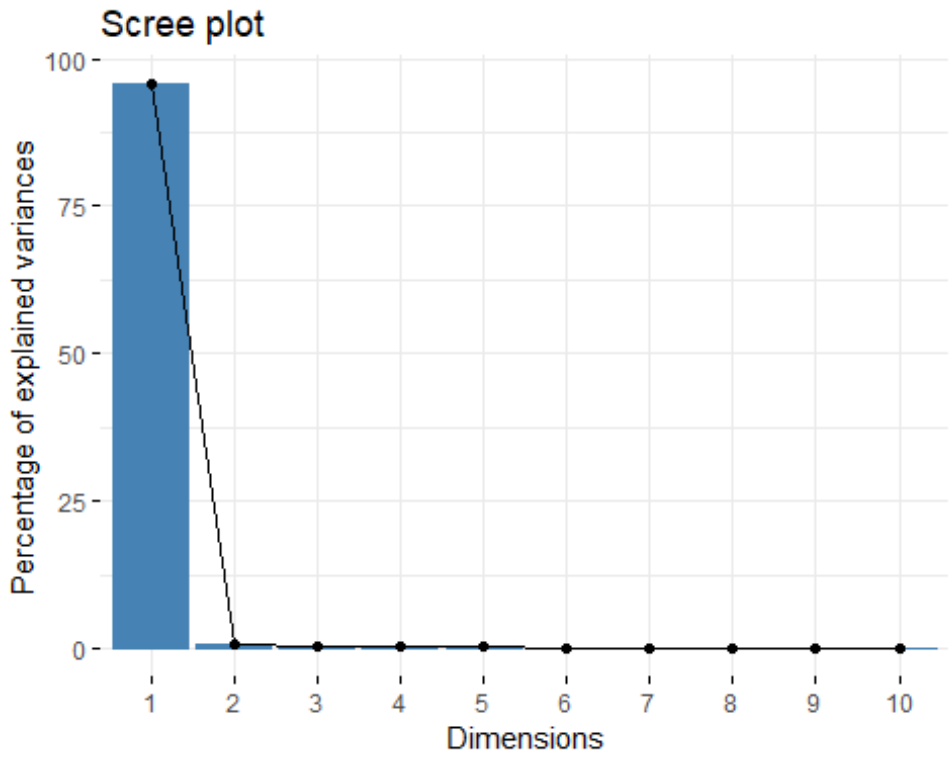
3.3 PCA

Investigating with a principal component analysis why the data clusters together the way it does. Looking at the first two principal components because they are the most interesting.

```
dataset <- processed_data$expression # determining which dataset will be used

colnames(dataset)<- cellinesMutations$Genes

pca = prcomp(t(dataset), center = F, scale. = F)
summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     495.3869 46.27533 35.11722 26.07913 24.64502
## Proportion of Variance   0.9573  0.00835  0.00481  0.00265  0.00237
## Cumulative Proportion    0.9573  0.96567  0.97048  0.97313  0.97550
##                             PC6      PC7      PC8      PC9     PC10
## Standard deviation     21.01126 20.73196 19.77350 19.31929 17.76823
## Proportion of Variance  0.00172  0.00168  0.00153  0.00146  0.00123
## Cumulative Proportion   0.97723  0.97890  0.98043  0.98188  0.98312
##                            PC11    PC12     PC13     PC14     PC15
## Standard deviation     17.71421 16.8245 16.18356 15.71160 15.43367
## Proportion of Variance  0.00122  0.0011  0.00102  0.00096  0.00093
## Cumulative Proportion   0.98434  0.9854  0.98647  0.98743  0.98836
##                            PC16     PC17     PC18     PC19     PC20
## Standard deviation     15.26658 14.96113 14.46341 13.90066 13.62804
## Proportion of Variance  0.00091  0.00087  0.00082  0.00075  0.00072
## Cumulative Proportion   0.98927  0.99014  0.99096  0.99171  0.99243
##                            PC21     PC22     PC23     PC24     PC25
## Standard deviation     13.48726 13.20117 13.08648 12.66417 12.34321
## Proportion of Variance  0.00071  0.00068  0.00067  0.00063  0.00059
## Cumulative Proportion   0.99314  0.99382  0.99449  0.99512  0.99571
##                            PC26     PC27     PC28     PC29     PC30
## Standard deviation     11.96792 11.74491 11.66818 11.40272 11.21416
## Proportion of Variance  0.00056  0.00054  0.00053  0.00051  0.00049
## Cumulative Proportion   0.99627  0.99681  0.99734  0.99785  0.99834
##                            PC31     PC32     PC33    PC34
## Standard deviation     10.83176 10.65546 10.26134 9.49512
## Proportion of Variance  0.00046  0.00044  0.00041 0.00035
## Cumulative Proportion   0.99879  0.99924  0.99965 1.00000
```

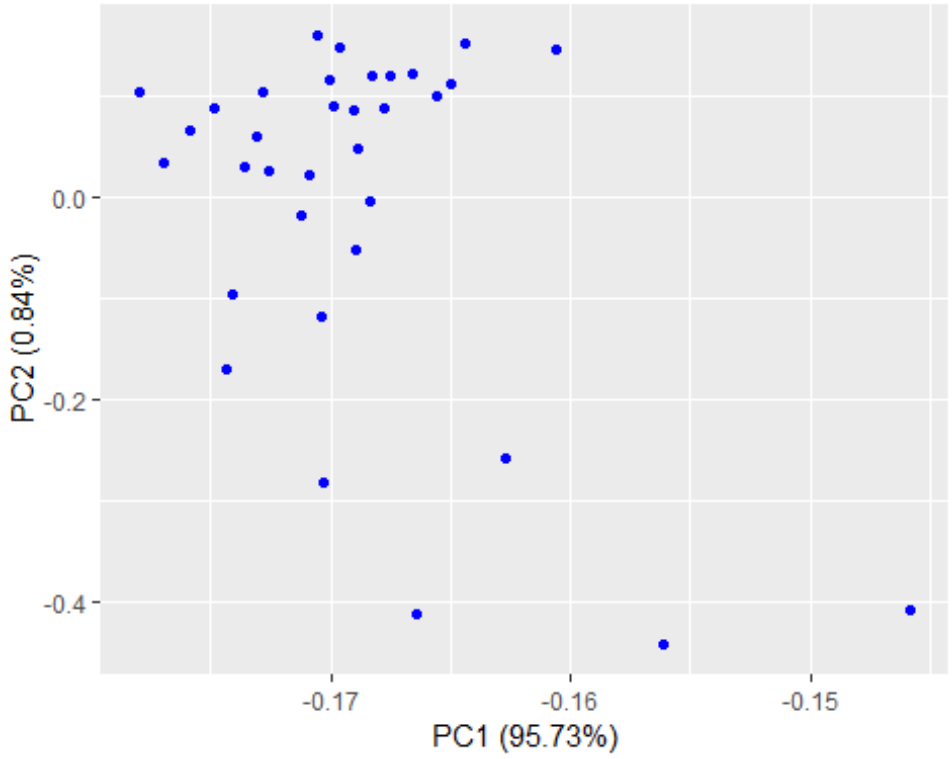
```
# showing labels (cell lines)


fviz_eig(pca)
```



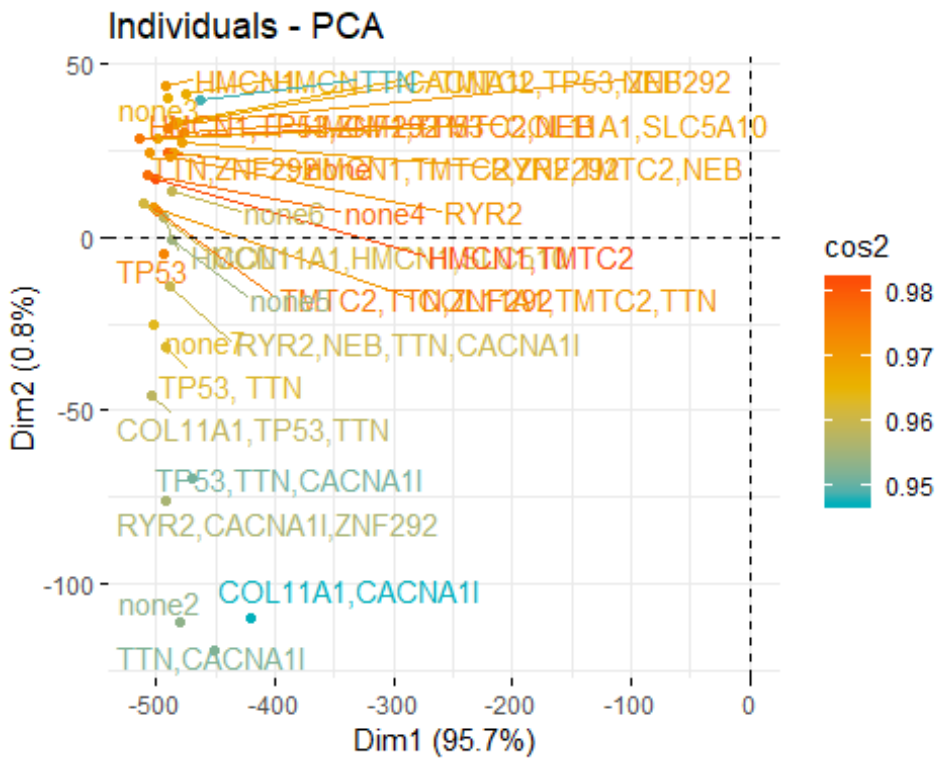

```
str(pca)
```

```
## List of 5
##  $ sdev    : num [1:34] 495.4 46.3 35.1 26.1 24.6 ...
##  $ rotation: num [1:16970, 1:34] -9.20e-03 -3.19e-05 -7.79e-03 -1.45e-04 -
2.30e-03 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:16970] "A1BG" "A1CF" "A2M" "A2ML1" ...
##   .. ..$ : chr [1:34] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : logi FALSE
##  $ scale   : logi FALSE
##  $ x       : num [1:34, 1:34] -502 -494 -511 -500 -504 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:34] "COL11A1,TMTC2,TTN" " HMCN1" "COL11A1,HMCN1,SLC510"
"HMCN1,TMTC2" ...
##   .. ..$ : chr [1:34] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
autoplot(pca, colour = 'blue')
```




```
fviz_pca_ind(pca,
             col.ind = "cos2", # color by the quality of representation
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE,     # Avoid text overlapping
)
```

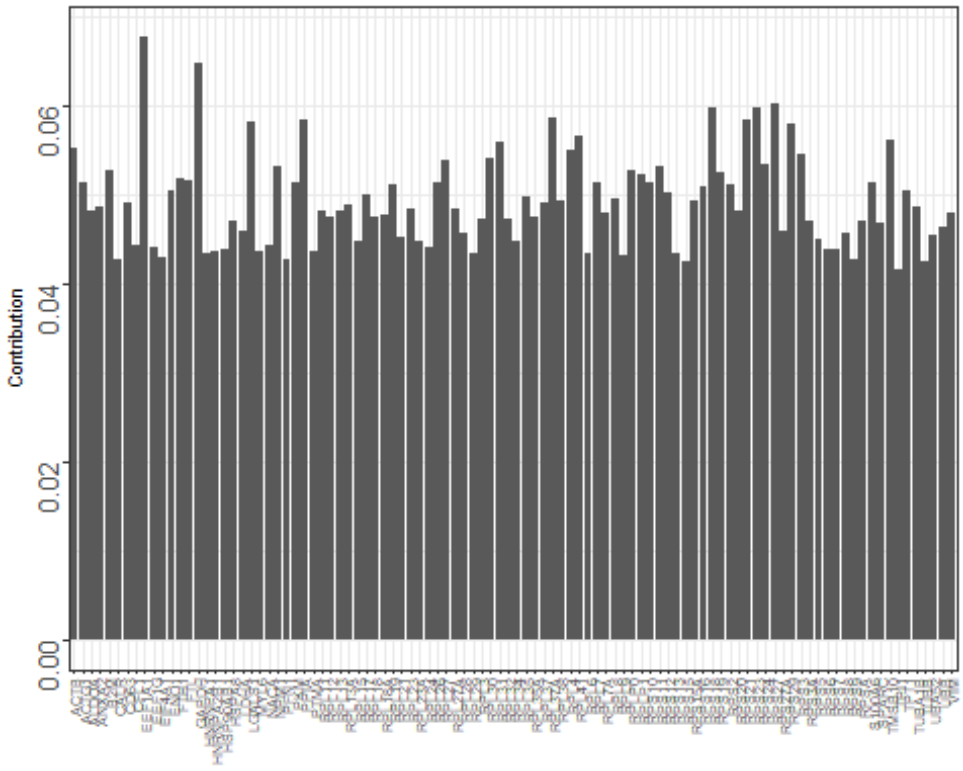


- *Again, there are two clusters.*
- *The first principal component contains the most information about the data.*

```
var_coord_func <- function(loadings, comp.sdev){
  loadings*comp.sdev
}

loadings <- pca$rotation
sdev <- pca$sdev
var.coord <- t(apply(loadings, 1, var_coord_func, sdev))

var.cos2 <- var.coord^2
comp.cos2 <- apply(var.cos2, 2, sum)
contrib <- function(var.cos2, comp.cos2){var.cos2*100/comp.cos2}

var.contrib <- t(apply(var.cos2,1, contrib, comp.cos2))
head(var.contrib[, 1:4])
```

```
##                  PC1          PC2          PC3          PC4
## A1BG   8.458140e-03 4.042140e-02 2.820464e-03 1.626858e-03
## A1CF   1.017925e-07 9.118405e-08 6.046034e-06 1.241073e-05
## A2M    6.063349e-03 4.405875e-02 8.758703e-02 1.918378e-02
## A2ML1  2.113426e-06 8.293155e-04 5.108695e-04 6.804066e-04
```

```
## A4GALT 5.277506e-04 5.244157e-02 2.770271e-03 6.278979e-03
## A4GNT  3.632784e-06 1.020123e-08 5.599425e-05 1.554282e-05

top100var.contrib <- var.contrib[,1]
top100var.contrib <- as.data.frame(top100var.contrib[order(-
top100var.contrib)])
top100var.contrib$Genes <- rownames(top100var.contrib)
top100var.contrib <- top100var.contrib[1:100,]
colnames(top100var.contrib)[1] <- "Contribution"



ggplot(data = top100var.contrib) +
  (geom_bar(mapping = aes(x = Genes, y = Contribution), stat = "identity")) +
  theme_bw(base_size = 7) + # formating the size of the theme nicely
  theme(legend.position= "none", # defining the legend position (here no
leghend will be needed)
        legend.direction="horizontal", # defining the legend direction if one
is there
        plot.title = element_text(hjust = 0.5), # making the title of the
plot into the middle
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, size =
5), # defining the orientation of the text on the x-axis
        axis.text.y = element_text(angle = 90, vjust = 0.5, hjust=1, size =
10), # defining the orientation of the text on the y-axis
        legend.title= element_blank(), # no title of the legend should be
plotted
        axis.title.x = element_blank(), # no title of the x-axis is relevant;
because that would be samples and that is cleare due to the naming
        strip.text.y = element_text(angle = 90)) # defining the orientation
of the text of the y-axis
```



- *These are the components which are contributing the most to our variation in the data. Maybe we will find some of these in our result of the p-test.*

```
rm(drivergene, realcelllinenames, dataset, loadings, pca, realcelllinenames,
var.contrib, var.coord, var.cos2, comp.cos2, sdev)

## Warning in rm(drivergene, realcelllinenames, dataset, loadings, pca,
## realcelllinenames, : Objekt 'drivergene' nicht gefunden

## Warning in rm(drivergene, realcelllinenames, dataset, loadings, pca,
## realcelllinenames, : Objekt 'realcelllinenames' nicht gefunden

## Warning in rm(drivergene, realcelllinenames, dataset, loadings, pca,
## realcelllinenames, : Objekt 'realcelllinenames' nicht gefunden
```

4. Statistical test

We want to perform a p-test and compare the p-values.

```
driverGenes <- rownames(geneCounts)[1:10] # only using the TOP 10 driver
genes
ttestgenes <- rownames(processed_data$kd.ceres)

potSecondSites <- lapply(seq_along(driverGenes), function(a) {
  genePicker <- driverGenes[a] # picking one driver gene
  print(paste0("I am doing driver mut: ", a))
  output <- sapply(seq_along(rownames(processed_data$kd.ceres)), function(b)
{ #the kdCERES matrix is of interest take its' rownames as refrence
    secondSitePicker <- rownames(processed_data$kd.ceres)[b] # picking a
potetnial 2nd site target
    if (secondSitePicker != genePicker) {
      drMUT <-
processed_data$kd.ceres[which(rownames(processed_data$kd.ceres) ==
genePicker),] # picking the driver mut data
      sndMUT <-
as.vector(processed_data$kd.ceres[which(rownames(processed_data$kd.ceres) ==
secondSitePicker),]) # picking the 2nd site data
      cor.val <- cor.test(unlist(drMUT, use.names=FALSE) , unlist(sndMUT,
use.names=FALSE), method = "spearman") # making a spearman correlation
      return(cor.val$p.value) # returning the p-values
    } else {
      return(1)
    }
  })
  names(output) <- rownames(processed_data$kd.ceres) # renaming all
  output <- as.data.frame(output) # getting a nice data frame
  return(output)
})
```

```
## [1] "I am doing driver mut: 1"
## [1] "I am doing driver mut: 2"
## [1] "I am doing driver mut: 3"
## [1] "I am doing driver mut: 4"
## [1] "I am doing driver mut: 5"
## [1] "I am doing driver mut: 6"
## [1] "I am doing driver mut: 7"
## [1] "I am doing driver mut: 8"
## [1] "I am doing driver mut: 9"
## [1] "I am doing driver mut: 10"
```

```
names(potSecondSites) <- driverGenes # renaming the list of lists
lapply(potSecondSites, head) # looking at the nice data
```

```
## $TTN
##            output
## A1BG   0.70023480
## A1CF   0.39115670
```

```
## A2M    0.34286907
## A2ML1  0.11397865
## A4GALT 0.19132453
## A4GNT  0.01504808
##
## $TP53
##            output
## A1BG   0.28160340
## A1CF   0.70023480
## A2M    0.39697321
## A2ML1  0.64097590
## A4GALT 0.09015868
## A4GNT  0.60183071
##
## $HMCN1
##           output
## A1BG   0.4227534
## A1CF   0.8657359
## A2M    0.6534159
## A2ML1  0.7917565
## A4GALT 0.8725280
## A4GNT  0.3437615
##
## $TMTC2
##            output
## A1BG   0.45154526
## A1CF   0.43701759
## A2M    0.95872743
## A2ML1  0.75867863
## A4GALT 0.01716129
## A4GNT  0.62127398
##
## $RYR2
##             output
## A1BG   0.879329218
## A1CF   0.669727445
## A2M    0.002884766
## A2ML1  0.213302043
## A4GALT 0.088108676
## A4GNT  0.304196025
##
## $CACNA1I
##            output
## A1BG   0.93400823
## A1CF   0.09259686
## A2M    0.14278128
## A2ML1  0.61030460
## A4GALT 0.10401228
## A4GNT  0.53711418
##
## $ZNF292
```

```
##            output
## A1BG   0.27458108
## A1CF   0.75736391
## A2M    0.57435603
## A2ML1  0.07565286
## A4GALT 0.38922902
## A4GNT  0.08058396
##
## $NEB
##            output
## A1BG   0.42275339
## A1CF   0.07595393
## A2M    0.36190890
## A2ML1  0.33314716
## A4GALT 0.04869324
## A4GNT  0.05084604
##
## $COL11A1
##            output
## A1BG   0.07535272
## A1CF   0.49777121
## A2M    0.80910520
## A2ML1  0.26539892
## A4GALT 0.18653536
## A4GNT  0.68237842
##
## $SLC5A10
##             output
## A1BG   0.991742072
## A1CF   0.467398656
## A2M    0.834622511
## A2ML1  0.453641927
## A4GALT 0.916192275
## A4GNT  0.007613915
```

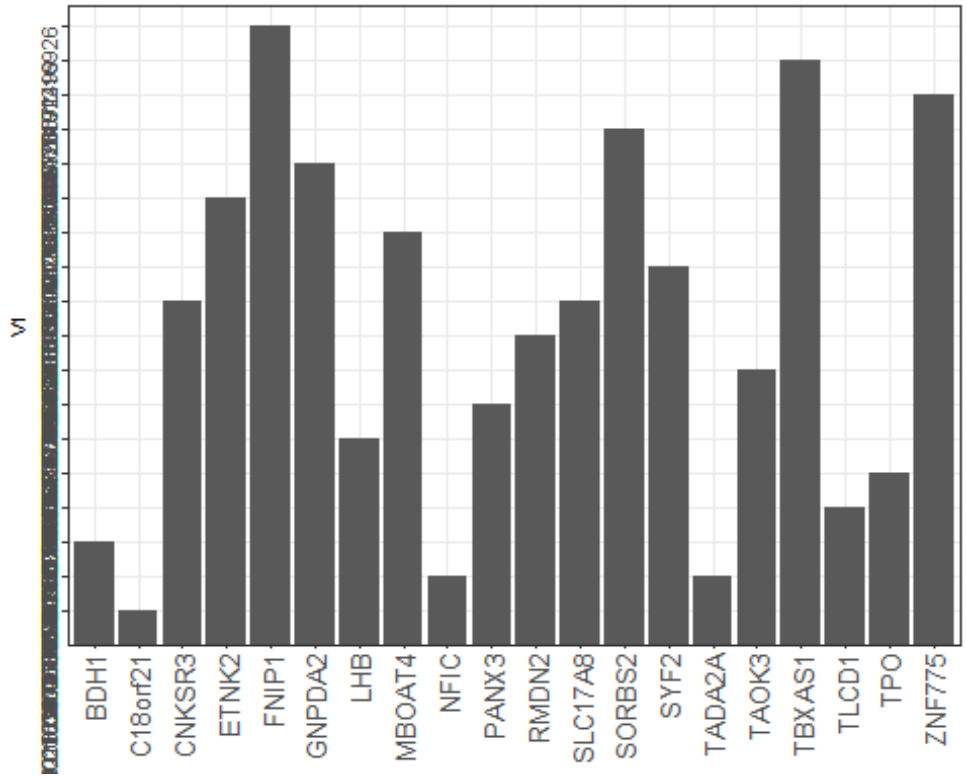
Now that we got all those p-values we want to order the data according to their p-values. So we can see the smallest ones which are the most important ones.

```
potSecondSites <- lapply(potSecondSites, function(a){
  a <- as.data.frame(cbind(a$output, rownames(a)))
  a <- a[order(a[1]), ]
})
```

Selecting the 20 genes out of every DriverGene List with the lowest p score.

```
potSecondSitestop20 <- lapply(seq_along(potSecondSites), function (a){
  output <- potSecondSites[[a]][1:20,]
  return(output)
})
names(potSecondSitestop20) <- driverGenes

ggplot(data = potSecondSitestop20$TTN) +
```

```
  (geom_bar(mapping = aes(x = V2, y = V1), stat = "identity")) +
  theme_bw(base_size = 7) + # formating the size of the theme nicely
  theme(legend.position= "none", # defining the legend position (here no
legend will be needed)
        legend.direction="horizontal", # defining the legend direction if one
is there
        plot.title = element_text(hjust = 0.5), # making the title of the
plot into the middle
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, size =
10), # defining the orientation of the text on the x-axis
        axis.text.y = element_text(angle = 90, vjust = 0.5, hjust=1, size =
8), # defining the orientation of the text on the y-axis
        legend.title= element_blank(), # no title of the legend should be
plotted
        axis.title.x = element_blank(), # no title of the x-axis is relevant;
because that would be samples and that is cleare due to the naming
        strip.text.y = element_text(angle = 90)) # defining the orientation
of the text of the y-axis
```




```
rm(potSecondSites, ttestgenes)
```

5. Multiple linear regression analysis

5.1 Predicting the expression of our driver genes with all the data

Creating a data frame for the multiple linear regression. In this data frame all the columns are the data frames and the rows represent the genes in every cell line.

With this data frame the linear regression is performed. After that the predicted values are compared with the real values of the data_set by a spearman correlation. Performing this with every driver gene.

```
# Building the dataframe for the linear Regression
a <- generalPlottingData$expression[,1:3]
a <-a[,c(1,3,2)]
copynumber <- generalPlottingData$copynumber[,2]
kd.ceres <- generalPlottingData$kd.ceres[,2]
kd.prob <- generalPlottingData$kd.prob[,2]

RegData <- cbind(a,copynumber,kd.ceres,kd.prob)

# doing the multiple linear regression
# comparing the predicted values of our model with the real values of the
test_data by spearman correlaton
# doing this for every driver gene

Regressionanalysis <-lapply(1:10, function(x){
  RegData <- cbind(a,copynumber,kd.ceres,kd.prob)
  Driverexpression <- c()
  for (i in 1:34) { # 34 = te skin cancer cell lines
    a <- 16970*i # 16970 = number of genes
    c <- (16970* (i-1))+1
    b <- colnames(processed_data$expression)[i]
    Driverexpression[c:a] <-
processed_data$expression[rownames(geneCounts)[x],b]
  }
  print(paste0("I am doing driver mut: ", rownames(geneCounts)[x]))
  RegData <- cbind(RegData,Driverexpression)
  RegData <-as.data.frame(RegData)
  colnames(RegData) <- as.vector(colnames(RegData))
  set.seed(123) # initializing the random numbers
  split = sample.split(RegData, SplitRatio = 0.8) # splitting the dataset
into 4/5 Training and 1/5 Testing dataset
  training_set = subset(RegData, split == TRUE) # using the labels to get the
training data
  test_set = subset(RegData, split == FALSE)
  rm(RegData)
  # fitting the multiple linear regression to the Training set
  regressor = lm(Driverexpression ~ Value + copynumber + kd.ceres + kd.prob ,
data = training_set) # predicting profit based on ALL (=.) the input
variables for one company
```

```
  # predicting the test set results
  y_pred = predict(regressor, newdata = test_set, se.fit = TRUE) # predicting
the expression based on the testing data
  test_set$Prediction = y_pred$fit # adding the predictions to the dataset
  # comparing the predictions (last column) with the real values of the
startups (2nd last column)
  Results <- cor.test(test_set$Driverexpression, test_set$Prediction, method
= "spearman", exact=FALSE)
  return(Results)
})
```

```
## [1] "I am doing driver mut: TTN"
## [1] "I am doing driver mut: TP53"
## [1] "I am doing driver mut: HMCN1"
## [1] "I am doing driver mut: TMTC2"
## [1] "I am doing driver mut: RYR2"
## [1] "I am doing driver mut: CACNA1I"
## [1] "I am doing driver mut: ZNF292"
## [1] "I am doing driver mut: NEB"
## [1] "I am doing driver mut: COL11A1"
## [1] "I am doing driver mut: SLC5A10"
```

```
names(Regressionanalysis) <- rownames(geneCounts)[1:10]
Regressionanalysis <- as.vector(Regressionanalysis)
rm(RegData,kd.ceres,kd.prob,copynumber,a)
```

```
ResultsRegression <- melt(lapply(1:length(Regressionanalysis), function(x){
  return(Regressionanalysis[[x]][3])
}))
ResultsRegression <-
cbind(ResultsRegression,melt(lapply(1:length(Regressionanalysis),
function(x){
  return(Regressionanalysis[[x]][1])
})))

ResultsRegression$L2 <- rownames(geneCounts)[1:10]
ResultsRegression <- ResultsRegression [,c(2,1,4)]
colnames(ResultsRegression) <- c("DriverGene", "pvalue", "Svalue" )

print(ResultsRegression)
```

```
##    DriverGene       pvalue       Svalue
## 1         TTN 5.509211e-16 7.317806e+14
## 2        TP53 4.997652e-09 7.359220e+14
## 3       HMCN1 5.739113e-37 7.233205e+14
## 4       TMTC2 1.486016e-36 7.234577e+14
## 5        RYR2 1.671884e-30 7.255677e+14
## 6     CACNA1I 7.949257e-20 7.299161e+14
## 7      ZNF292 9.481412e-12 7.341441e+14
## 8         NEB 5.286592e-14 7.328378e+14
```

```
## 9     COL11A1 1.548789e-77 7.124150e+14
## 10    SLC5A10 4.727922e-25 7.276653e+14
```

- *With these low p-values we can say with confidence that our Model is able to reproduce and predict the expression values of our driver genes.*
- *Using just our top 20 out of the statistical testing we hoped to see that the p values would not increase that much. This would verify our these that these genes are the essential components which drive the different expression of the Driver Gene.*
- *As you can see below this ist not the case and the p values are very much increased.*

```
Regressionanalysistop20 <-lapply(1:10, function(x){
  a <-
generalPlottingData$expression[which(generalPlottingData$expression[,3] %in%
as.character(potSecondSitestop20[[x]][,2])),1:3]
  a <-a[,c(1,3,2)]
  copynumber <-
generalPlottingData$copynumber[which(generalPlottingData$copynumber[,3] %in%
as.character(potSecondSitestop20[[x]][,2])),2]
  kd.ceres <-
generalPlottingData$kd.ceres[which(generalPlottingData$kd.ceres[,3] %in%
as.character(potSecondSitestop20[[x]][,2])),2]
  kd.prob <-
generalPlottingData$kd.prob[which(generalPlottingData$kd.prob[,3] %in%
as.character(potSecondSitestop20[[x]][,2])),2]
  RegData <- cbind(a,copynumber,kd.ceres,kd.prob)
  h <-
length(generalPlottingData$expression[which(generalPlottingData$copynumber[,3
] %in% as.character(potSecondSitestop20[[x]][,2])),2])
  Driverexpression <- c()
  for (i in 1:34) {
    a <- h*i
    c <- (h* (i-1))+1
    b <- colnames(processed_data$expression)[i]
    Driverexpression[c:a] <-
processed_data$expression[rownames(geneCounts)[x],b]
  }
  print(paste0("I am doing driver mut: ", rownames(geneCounts)[x]))
  RegData <- cbind(RegData,Driverexpression)
  RegData <-as.data.frame(RegData)
  colnames(RegData) <- as.vector(colnames(RegData))
  set.seed(123) #initialize the random numbers
  split = sample.split(RegData, SplitRatio = 0.8) #split the dataset into 4/5
Training and 1/5 Testing dataset
  training_set = subset(RegData, split == TRUE) #use the labels to get the
training data
  test_set = subset(RegData, split == FALSE)
  rm(RegData)
  # Fitting Multiple Linear Regression to the Training set
```

```
  regressor = lm(Driverexpression ~ Value + copynumber + kd.ceres + kd.prob ,
data = training_set) #predict profit based on ALL (=.) the input variables
for one company
  # Predicting the Test set results
  y_pred = predict(regressor, newdata = test_set, se.fit = TRUE) #predict the
expression based on your testing data
  test_set$Prediction = y_pred$fit #add your predictions to the dataset
  #Now compare the Predictions (last column) with the real values of the
startups (2nd last column)
  Results <- cor.test(test_set$Driverexpression, test_set$Prediction, method
= "spearman", exact=FALSE)
  return(Results)
})
```

```
## [1] "I am doing driver mut: TTN"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: TP53"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: HMCN1"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: TMTC2"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: RYR2"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: CACNA1I"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: ZNF292"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: NEB"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
## [1] "I am doing driver mut: COL11A1"
```

```
## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded

## [1] "I am doing driver mut: SLC5A10"

## Warning in data.frame(..., check.names = FALSE): row names were found from
## a short variable and have been discarded
```

```
names(Regressionanalysistop20) <- rownames(geneCounts)[1:10]
Regressionanalysistop20 <- as.vector(Regressionanalysistop20)

ResultsRegressiontop20 <- melt(lapply(1:length(Regressionanalysistop20),
function(x){
  return(Regressionanalysistop20[[x]][3])
}))
ResultsRegressiontop20 <-
cbind(ResultsRegressiontop20,melt(lapply(1:length(Regressionanalysistop20),
function(x){
  return(Regressionanalysistop20[[x]][1])
})))

ResultsRegressiontop20$L2 <- rownames(geneCounts)[1:10]
ResultsRegressiontop20 <- ResultsRegressiontop20 [,c(2,1,4)]
colnames(ResultsRegressiontop20) <- c("DriverGene", "pvalue", "Svalue" )

print(ResultsRegressiontop20)
```

```
##    DriverGene     pvalue      Svalue
## 1         TTN 0.09746992 49026556057
## 2        TP53 0.04382470 49238451647
## 3       HMCN1 0.88944564 48128985972
## 4       TMTC2 0.12381370 48956640123
## 5        RYR2 0.10252586 49012065541
## 6     CACNA1I 0.42022190 48523361801
## 7      ZNF292 0.07410218 49102647128
## 8         NEB 0.79788512 48198209845
## 9     COL11A1 0.80113984 48195718948
## 10    SLC5A10 0.20397571 48797830898
```

- *With this result we can not define confidently the second targets.*

---

6. Results

```
Resultspresentation <- lapply(1:length(potSecondSitestop20), function(x){
  return(potSecondSitestop20[[x]][2])
})

names(Resultspresentation) <- rownames(geneCounts)[1:10]
print(Resultspresentation)
```

```
## $TTN
##             V2
## 1619  C18orf21
## 9311      NFIC
## 14283   TADA2A
## 1308      BDH1
## 14708    TLCD1
## 15183      TPO
## 7750       LHB
## 10332    PANX3
## 14321    TAOK3
## 12208    RMDN2
## 2967    CNKSR3
## 13171  SLC17A8
## 14213     SYF2
## 8296    MBOAT4
## 4571     ETNK2
## 5693    GNPDA2
## 13698   SORBS2
## 16853   ZNF775
## 14435   TBXAS1
## 5161     FNIP1
##
## $TP53
##             V2
## 2499    CDKN1A
## 11816    RAD50
## 14298     TAF4
## 14764    TMCC1
## 4371      ELP5
## 2900    CLNS1A
## 3317    CSNK1E
## 15153  TP53BP1
## 1038      ATE1
## 5012     FEM1B
## 10303    PAGR1
## 12378    RPL23
## 16904   ZNF862
## 11548    PSME3
## 11231  PPP1R42
## 4149    DYNLT1
## 11199 PPP1R12A
## 9363     NIPBL
```

```
## 16084    WDR83
## 14889  TMEM207
##
## $HMCN1
##           V2
## 3855    DLEC1
## 7868      LPA
## 14241    SYT1
## 2696   CHI3L2
## 1086  ATP13A4
## 15195   TPRX1
## 3970     DNM1
## 997     ASGR1
## 5454     GCKR
## 321       ADO
## 2893     CLMN
## 564      AMTN
## 12134  RHBDD2
## 14043   STBD1
## 5219     FPGT
## 8695  MRGPRX1
## 289     ADCY2
## 11622  PTPN13
## 16185    XPO4
## 6637    IFNW1
##
## $TMTC2
##            V2
## 6823     INPP1
## 6415    HPCAL4
## 6838     INSL4
## 10807     PIM1
## 1630  C19orf44
## 13439   SLC6A5
## 12357     RPF2
## 13018     SHBG
## 14524    TENM4
## 9418      NME1
## 12112    RGS13
## 11239  PPP2R2A
## 13478  SLCO2A1
## 12557    RWDD3
## 4841    FAM78A
## 15160   TP53RK
## 7610     LAMC1
## 14552    TEX33
## 5893     GRIK4
## 16316   ZC3H7A
##
## $RYR2
##            V2
```

```
## 5170       FOS
## 10974   PLSCR1
## 4126     DUSP7
## 13270  SLC26A8
## 4693   FAM118B
## 13662    SNX21
## 12493     RRP8
## 6148     HERC4
## 2600   CEP57L1
## 15432     TSPO
## 13161  SLC16A7
## 6609    IFITM2
## 10984   PLXNA2
## 4233   EFCAB14
## 8288      MBIP
## 16664   ZNF501
## 12256    RNF14
## 13123  SLC10A1
## 4750  FAM177A1
## 173      ACSF3
##
## $CACNA1I
##              V2
## 13544     SMDT1
## 8832       MSX2
## 7726    LGALS12
## 9657      NUCB1
## 7536  KRTAP21-3
## 4033       DPRX
## 3564       DAOA
## 10200    OSBPL3
## 10618      PENK
## 4608    EXOC3L4
## 11025    PNPLA5
## 12802     SEH1L
## 760    APOBEC3F
## 9013      MYO1H
## 15604     UBA52
## 12857    SERHL2
## 1598   C16orf97
## 3660       DDR1
## 39        ABCA7
## 15108    TNRC6B
##
## $ZNF292
##              V2
## 7630    LARP4B
## 10984   PLXNA2
## 13213  SLC24A1
## 13373  SLC39A9
## 10203   OSBPL7
```

```
## 15231 TRAPPC10
## 12266   RNF152
## 1824   C6orf99
## 5586      GLB1
## 8021      LSM5
## 10214     OSMR
## 14875  TMEM190
## 15768   UNC93A
## 10717    PHF5A
## 3094      COPA
## 11354   PRKCSH
## 16138    WNT5A
## 8275       MAX
## 2703      CHL1
## 4649     F2RL2
##
## $NEB
##            V2
## 3318   CSNK1G1
## 3931   DNAJC10
## 7082    KCNAB3
## 12359 RPGRIP1L
## 2675    CHCHD5
## 15311   TRIM61
## 9390    NKX6-1
## 11828    RAD9B
## 16674   ZNF514
## 3429     CWC25
## 2332      CD22
## 13376  SLC40A1
## 8895  MTRNR2L2
## 4583     EVA1A
## 5809    GPR183
## 14900  TMEM219
## 1213     BAALC
## 11643    PTPRH
## 15120   TOM1L2
## 8621     MOCOS
##
## $COL11A1
##            V2
## 3713  DEFB108B
## 5233       FRK
## 10256      OXT
## 16705   ZNF560
## 2882     CLIC6
## 9029     MYOM1
## 14943   TMEM37
## 930      ARMS2
## 11982     RBP1
## 1922    CACNG7
```

```
## 7372    KLHL36
## 4932    FBXO11
## 15437   TSPYL6
## 1078    ATP10D
## 8008    LRTOMT
## 14339  TAS2R13
## 1898    CACHD1
## 5332    GABBR2
## 15616  UBASH3B
## 11202 PPP1R13B
##
## $SLC5A10
##              V2
## 8177     MAP2K4
## 13850     SPNS2
## 16525    ZNF256
## 6648      IFT52
## 13815     SPEM1
## 7249  KIAA1324L
## 4226      EEF2K
## 8987       MYL4
## 10888   PLAC8L1
## 14539     TEX10
## 16738    ZNF593
## 15072   TNFSF12
## 13018      SHBG
## 3795     DHRS11
## 11358     PRKD3
## 11415   PRPSAP2
## 10410     PCBP4
## 13153  SLC16A11
## 14303     TAF6L
## 4107     DUSP10
```

```
print(top100var.contrib[1:20,2])
```

```
##  [1] "EEF1A1" "GAPDH"  "RPS27"  "RPS23"  "RPS18"  "RPL37A" "PPIA"
##  [8] "RPS21"  "LGALS1" "RPS29"  "RPL41"  "TMSB10" "RPL31"  "ACTB"
## [15] "RPL4"   "RPS3"   "RPL30"  "RPL27"  "RPS24"  "RPS11"
```

```
which(top100var.contrib[1:20,2] %in%
as.character(melt(Resultspresentation)[,1]))
```

```
## Using V2 as id variables
## Using V2 as id variables
## Using V2 as id variables
## Using V2 as id variables
## Using V2 as id variables
## Using V2 as id variables
## Using V2 as id variables
## Using V2 as id variables
```

```
## Using V2 as id variables
## Using V2 as id variables



## integer(0)
```

- *Our top 10 driver mutations are : TTN, TP53, HMCN1, TMTC2, RYR2, CACNA1I, ZNF292, NEB, COL11A1, SLC5A10. We defined them with a barplot at the beginning.*
- *Our goal was to find possible second side targets interacting with these driver mutations more often than other mutations and which are of a greater importance than other interactin genes.*
- *We tried to define these targets with a PCA and a p-test.But the 2nd targets from the pca and the regression are not overlapping. This could be due to the data or mistakes we made in the skript. Also the kmeans and the PCA do not reproduce the same second side targets.*
- *However we decided to present the targets we defined with the p-test. As shown above we have listed the top 20 second side targets from the p-test for each gene.*
- *The regression model shows a really low p-value at which leads us to the conclucion that the above listed genes could be taken in account as targets for drug development in skin cancer.*

---

7. Biological background

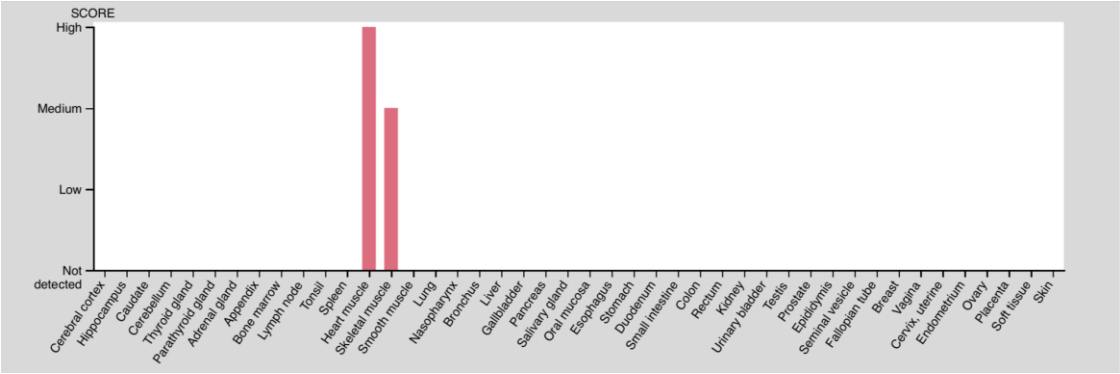
Firstly, the literature identifies BRAF, RAS, NFI and Triple WT as the leading mutations for skin cancer (1). Unfortunatly, mutations of these genes are not emphasized as our top 10 driver mutations.

However, we will now take a look at our results. Could the driver mutation extracted from our data be related to skin cancer? And are there biological interactions between the driver mutations and the second targets?

In the following, we will be investigating our top mutated genes TTN, TP53 and their second targets and their relevence in skin cancer therapy.

TTN, which is our top mutated gene, encodes for the largest protein in the human genome and is a part of the sacromere in the steriated muscle (2). Its complex structure and size, which leads to the sacromeric organization during a contraction, is due to its composition of 364 exons (2). That creats a protein, which is approximatly composed of 38,000 amino acid residues(2). The mentioned sacromere gene is stated to be a major human disease gene, for instance, truncating types of TTN are reprotedly the main reason for dilated cardiomyopathy, which is a common cause of heart failure and cardiac death (2).

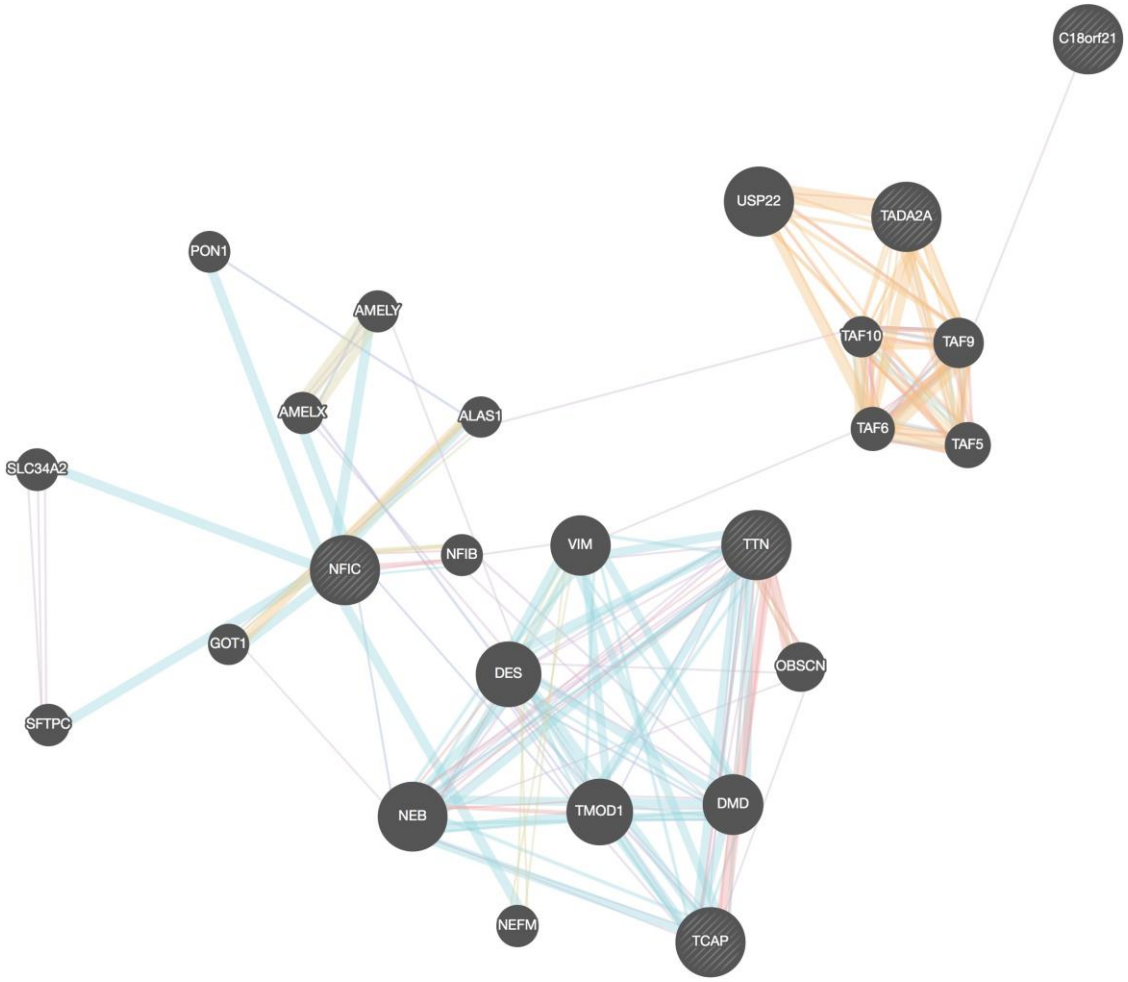
Tissue expression of TTN in heart muscle and skeletal muscle (3):

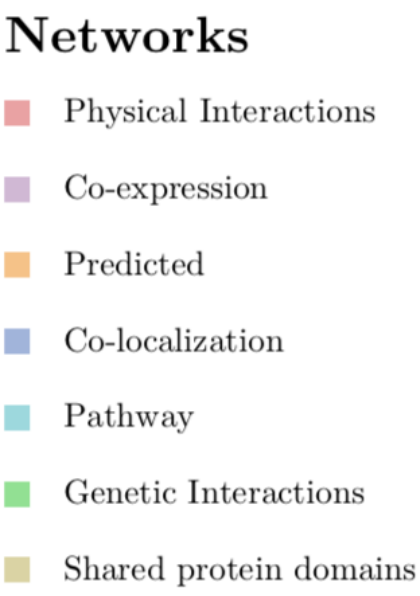


It is possible to see in the chart, that TTN is not detected in skin tissue, consequently, it will not play a leading role in the developement of skin cancer. Nevertheless, in the following we will investigate in how well our analysis was able to predict possible second targets.

In that matter we used a website called GeneMANIA, which is a server for biological network integration and predicting gene functions (4). The website enables us to visually see how genes are connected and to what extend physical interactions, co-expression, co-localiation, pathway, genetic interactions and shared protein domains can be found (4). It is possible to download a full report to gain fruther inside, for instance, which pathway in particular is connecting these genes (4). In our results the three major second targets for TTN are C18orf21, NFIC, TADA2A. Our driver mutation and the calculated second targets are indicated with stripes.

Network of TTN and second targets:





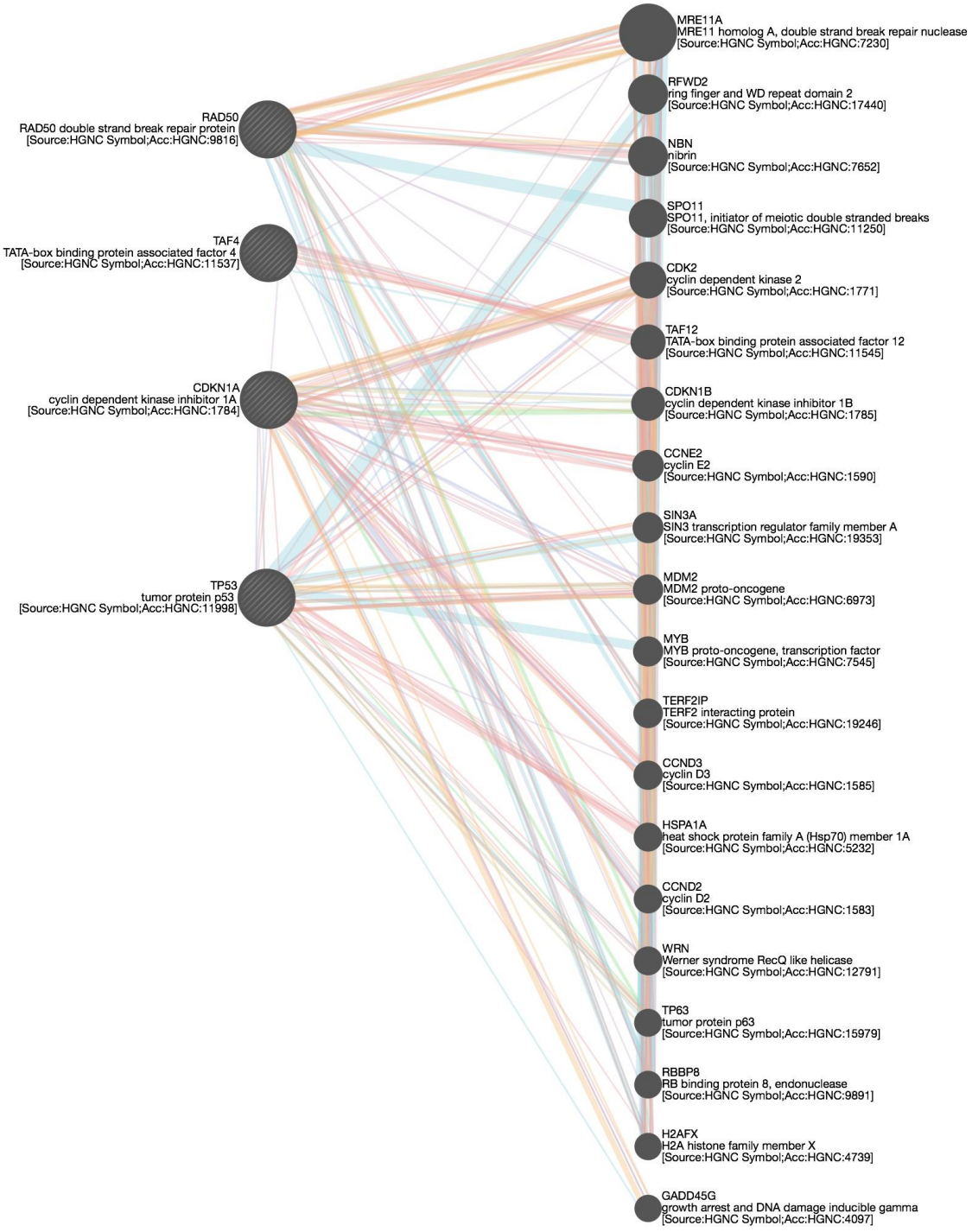
In the report it is stated that 67,64% of physical interaction and 13,5% of co-expression can be found, whereas, the probabilty for shared pathways and genetic interactions are significantyl low (4). Physical interaction indicates that these proteins are likely to form bonds between complexes and interact with each other (5). TTN is not directly connected with any second target, however, a link is to be found between TTN and NFIC which are co-expressed with NEB (nebulin),which is giant protein component of the cytoskeletal matrix that coexists with the thick and thin filaments within the sarcomeres of skeletal muscle (6).

As we mentioned in our results, because of our regression analysis it is not possible to state C18orf21, NFIC, TADA2A confidently as second targets, thus, the computed network propably does not match the underlying interactions in our data set.

In the following we will take a look at TP53, which is our second most mutated gene, and its second targets. This gene is a tumor suppressor protein containing transcriptional activation, DNA binding, and oligomerization domains, additionally, it plays a major role in the induction of cell cycle arrest, apoptosis, senescence, DNA repair, and changes in metabolism (7). A loss of function in TP53 through a mutation can lead to multiple types of human cancer (7). UV irridation appears to play a significant role in the mutation of TP53, according to the article " p53 and the Pathogenesis of Skin Cancer" a significant number of TP53 mutants are to be found on sun-exposed skin (8). As a result, our second targets for TP53 could be a lead for possible new drug targets in skin cancer therapy.

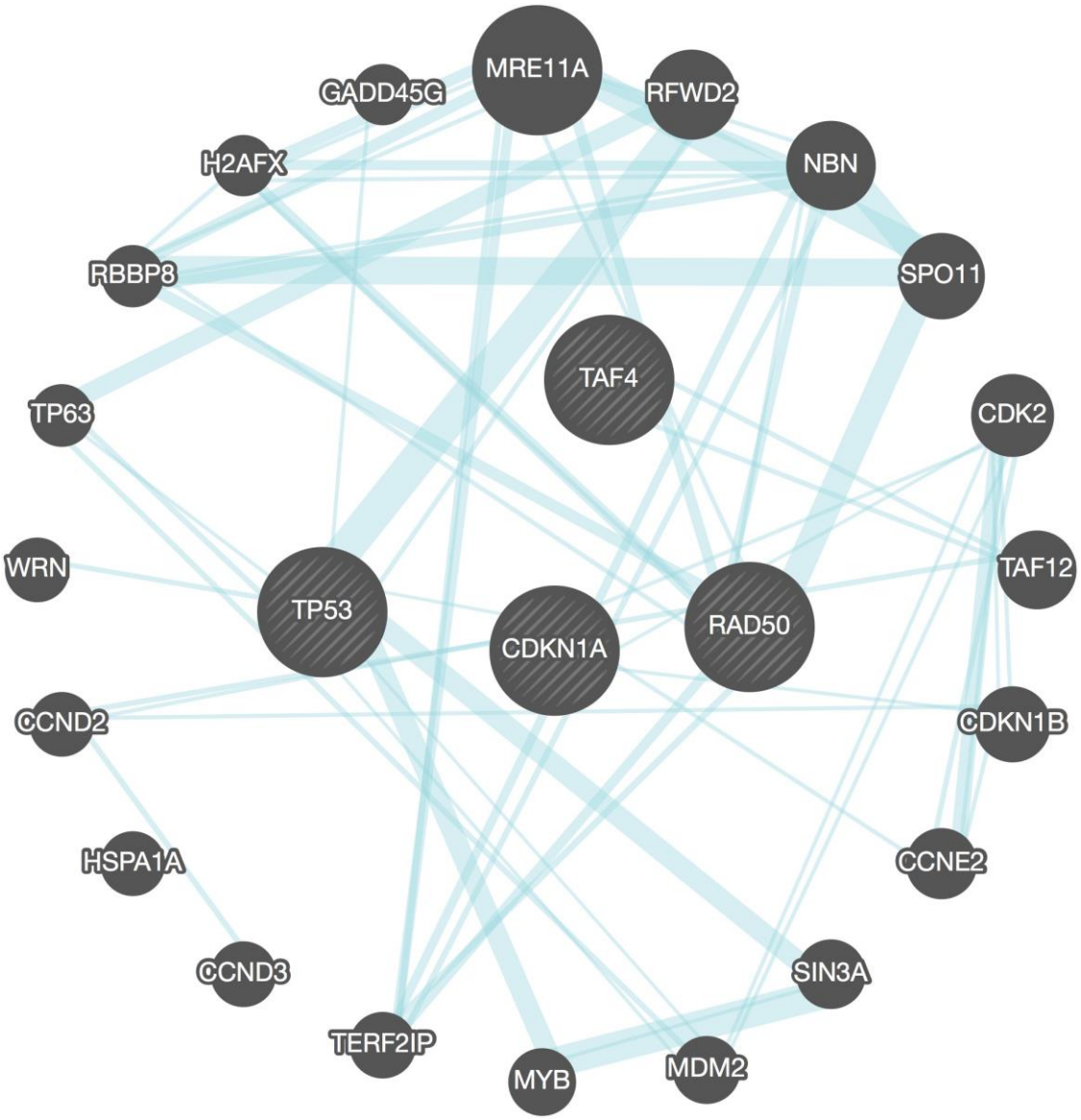
The top three second targets for TP53 ,which were extracted from our data base, are CDKN1A, RAD50, TAF4. We perform an anlysis with GeneMANIA to detect possible interactions (4). On the left side is our driver mutation and second targets and on the right side are all possible interaction partners are listed.

Network of TP53 and its second targets:



In this case its is noticeable that the correlation between the pathways of TP53 and the second targets is significantly high with 4.35% , which is illustrated below.

Network of pathways: TP53 and second targets:



According to our analysis, CDKN1A is predicted most likely to be a second target for TP53. Due to the correlation of driver mutation and the second targets pathways ,we will take a fruther look into a possible link between CDKN1A and TP53. CDKN1A is a cyclin dependent kinase inhibitor and expresses proteins which function as regulators of cell cycle progression at G1 (7). The expression of CDKN1A is controlled by TP53, which means that CDKN1A has a key role in the p53-dependent cell cycle G1 phase arrest caused by stress stimuli (7). It is reported that this second target can interact with proliferating cell nuclear antigen and has a regulatory role in S phase DNA repilcation and DNA damage repair (7). G1/S checkpoint defects are mentioned to be significant factors in melanoma tumorgenesis, although, the results implicate that cyclindependent kinases and TP53 are not major drivers, but can be taken into consideration (9).

To sum it up, in spite of the fact that the driver mutation from the literature and the calculated driver mutation from our data set do not match up, we were able to detect

possible second targets for our driver mutations. In the case of TP53, CDKN1A gene has clearly biological interactions with TP53 and our mathematical model was able to predict this interaction.

8. References

(1) Akbani et al. (2015), Genomic Classification of Cutaneous Melanoma. Cell 161, 1681-1696

(2) Gigli M, Begay RL, Morea G, et al. A Review of the Giant Protein Titin in Clinical Molecular Diagnostics of Cardiomyopathies. Front Cardiovasc Med. 2016;3:21. Published 2016 Jul 21. doi:10.3389/fcvm.2016.00021

(3) Uhlén M et al, 2015. Tissue-based map of the human proteome. Science PubMed: 25613900 DOI: 10.1126/science.1260419, Human Protein Atlas

(4) Warde-Farley D, Donaldson SL, Comes O, Zuberi K, Badrawi R, Chao P, Franz M, Grouios C, Kazi F, Lopes CT, Maitland A, Mostafavi S, Montojo J, Shao Q, Wright G, Bader GD, Morris Q Nucleic Acids Res. 2010 Jul 1;38 Suppl:W214-20 PubMed Abstract

(5) Clancy T, Rødland EA, Nygard S, Hovig E. Predicting physical interactions between protein complexes. Mol Cell Proteomics. 2013;12(6):1723-1734. doi:10.1074/mcp.O112.019828

(6) Pruitt KD, Tatusova T, Brown GR, Maglott DR. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. Nucleic Acids Res. 2012;40(Database issue):D130-D135. doi:10.1093/nar/gkr1079

(7) O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, McVeigh R, Rajput B, Robbertse B, Smith-White B, Ako-Adjei D, Astashyn A, Badretdin A, Bao Y, Blinkova O, Brover V, Chetvernin V, Choi J, Cox E, Ermolaeva O, Farrell CM, Goldfarb T, Gupta T, Haft D, Hatcher E, Hlavina W, Joardar VS, Kodali VK, Li W, Maglott D, Masterson P, McGarvey KM, Murphy MR, O'Neill K, Pujar S, Rangwala SH, Rausch D, Riddick LD, Schoch C, Shkeda A, Storz SS, Sun H, Thibaud-Nissen F, Tolstoy I, Tully RE, Vatsan AR, Wallin C, Webb D, Wu W, Landrum MJ, Kimchi A, Tatusova T, DiCuccio M, Kitts P, Murphy TD, Pruitt KD. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation.

Nucleic Acids Res. 2016 Jan 4;44(D1):D733-45 PubMed Tatusova T, DiCuccio M, Badretdin A, Chetvernin V, Nawrocki EP, Zaslavsky L, Lomsadze A, Pruitt KD, Borodovsky M, Ostell J. NCBI prokaryotic genome annotation pipeline. Nucleic Acids Res. 2016 Aug 19;44(14):6614-24 PubMed

Brister JR, Ako-Adjei D, Bao Y, Blinkova O. NCBI viral genomes resource. Nucleic Acids Res. 2015 Jan;43(Database issue):D571-7 PubMed

(8) Benjamin CL, Ananthaswamy HN. p53 and the pathogenesis of skin cancer. Toxicol Appl Pharmacol. 2007;224(3):241-248. doi:10.1016/j.taap.2006.12.006

(9) Soto JL, Cabrera CM, Serrano S, López-Nevot MA. Mutation analysis of genes that control the G1/S cell cycle in melanoma: TP53, CDKN1A, CDKN2A, and CDKN2B. BMC Cancer. 2005;5:36. Published 2005 Apr 8. doi:10.1186/1471-2407-5-36

finally done :)