HW3: Transcriptome Analysis

Homework 3 - to be done as groups

Names:

Group:

For deadlines etc., see Absalon.

You have to supply both the answer (whatever it is: numbers, a table, plots or combinations thereof), as well as the R or Linux code you used to make the plots. This should be done using this R markdown template: we want both the R markdown file and a resulting PDF. For PDF output, you may have to install some extra programs - RStudio will tell you.

Note that:

1. If the R code gives different results than your results, you will get severe point reductions or even 0 points for the exercise
2. Some questions may request you to use R options we have not covered explicitly in the course: this is part of the challenge
3. While this is a group work, we expect that everyone in the group will have understood the group solution: similar or harder question might show up in the individual homework. So, if something is hard, it means you need to spend more time on it
4. The results should be presented on a level of detail that someone else could replicate the analysis.

For statistical tests, you have to:

1) Motivate the choice of test
2) State exactly what the null hypothesis is (depends on test!)
3) Comment the outcome: do you reject the null hypothesis or not, and what does this mean for the actual question we wanted to answer (interpretation)?

Intro

As you already know how to quantify RNAseq data (see the quantification exercise from RNAseq lecture) this homework is about the downstream anlaysis of such quantifications.

Please use `knitr::kable()` to produce nicely formatted tables when you are asked to provide a table.

Part1: Data analysis and clustering

Use the supplied subset of Salmon quantifications stored in the ???salmon_result_part1.zip??? folder. These files contain the Salmon quantifications of 6 samples ??? 3 biological replicates of non-treated cells (WT) and 3 biological replicates of cells treated with a cancer promoting drug called TPA (WTTPA). Salmon was run with the `-seqBias` option.

Question 1.1

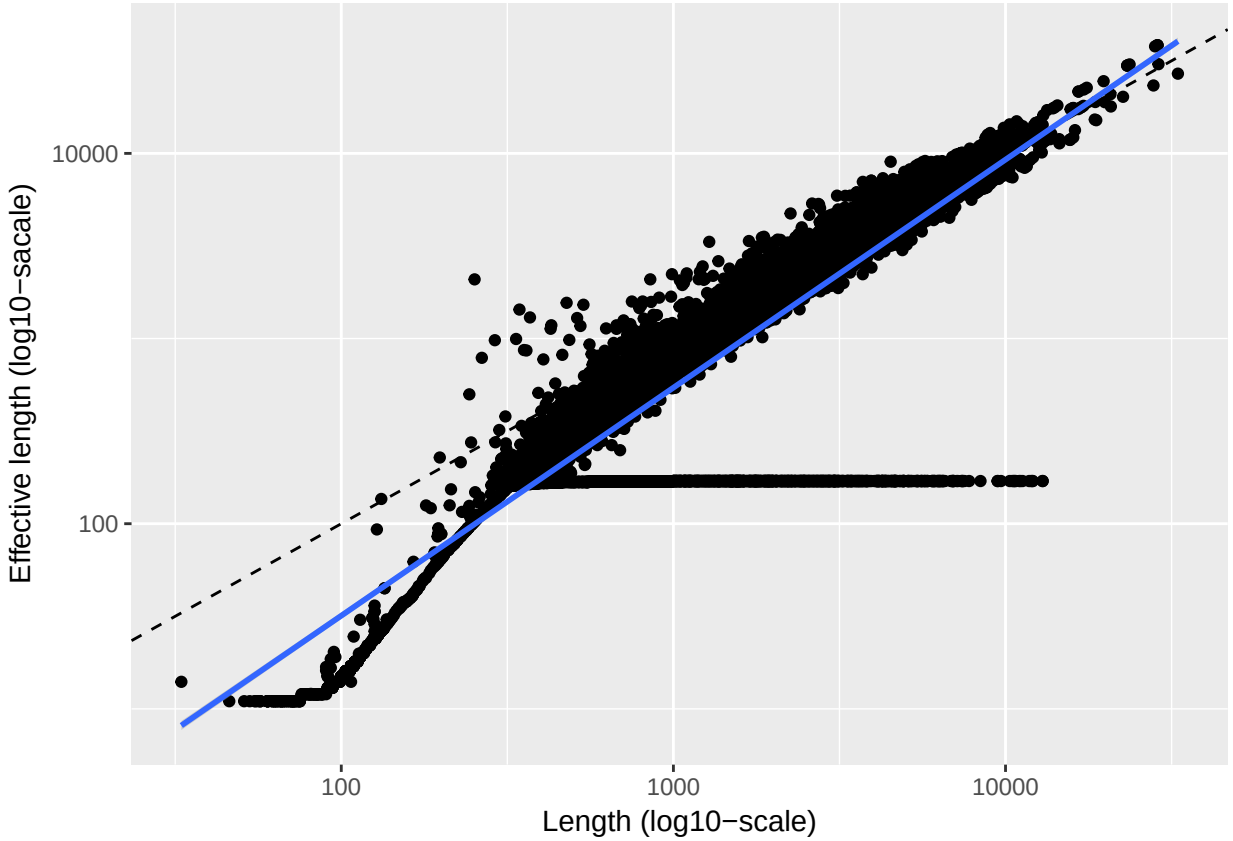
Read the ???quant.sf??? file from the Salmon result folder for WT1 into R with `read_tsv()`. Plot the isoform lengths versus the effective lengths as a scatter plot, add a smoothed line and a dashed line along

the diagonal. Scale both axis using log10-scaling in ggplot2. Comment on the comparison og the differences between the trend line and the diagonal line with respect to what is expected. **Use max 100 words.**

```
wt1_quant = read_tsv('salmon_result_part1/WT1/quant.sf')
```

```
## Parsed with column specification:
## cols(
##   Name = col_character(),
##   Length = col_integer(),
##   EffectiveLength = col_double(),
##   TPM = col_double(),
##   NumReads = col_double()
## )
```

```
wt1_quant %>% ggplot(aes(x=Length, y=EffectiveLength)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed') +
  geom_smooth(method='lm', fullrange=T) +
  scale_x_continuous(trans=log10_trans()) +
  scale_y_continuous(trans=log10_trans()) +
  xlab('Length (log10-scale)') +
  ylab('Effective length (log10-sacale)')
```

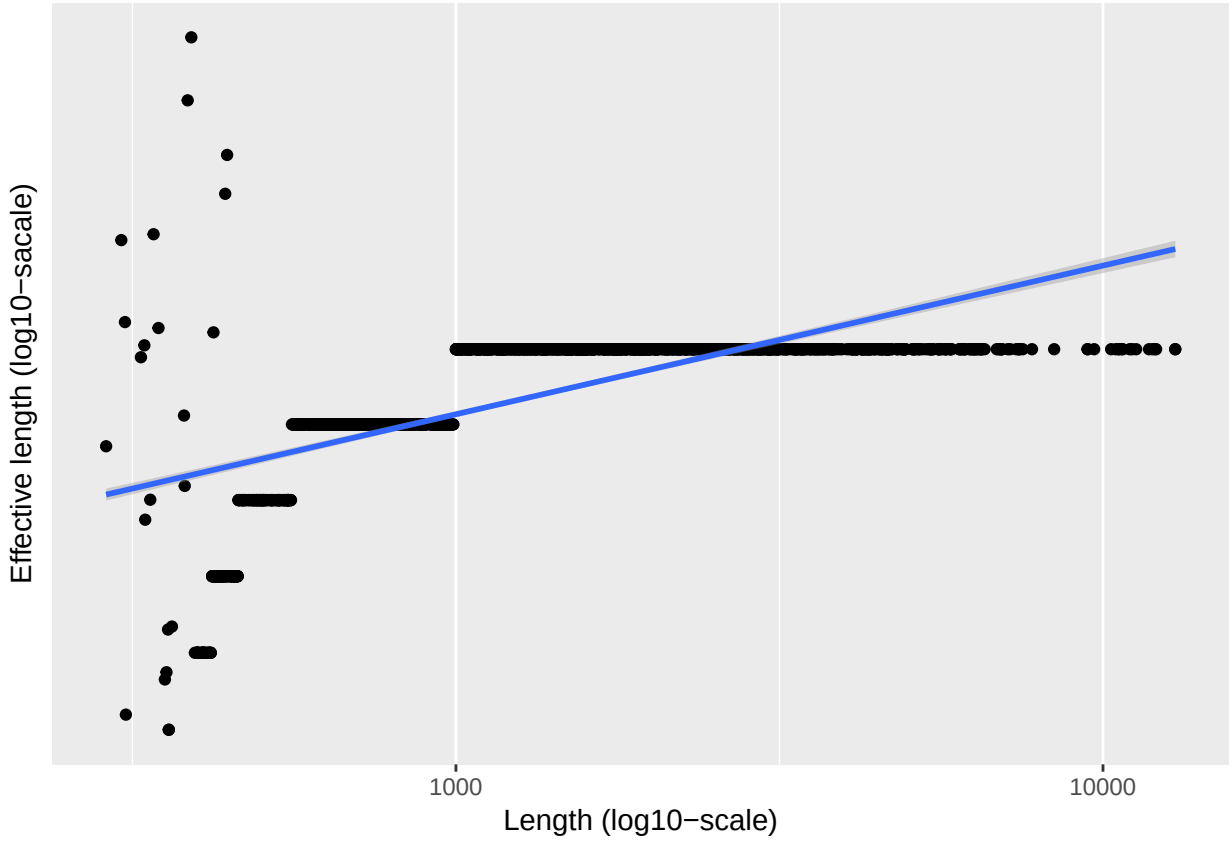


We would expect the diagonal and the smooth line to be approximately equal, since we would expect the effective length to be equal to the real one. But what we see is that for short sequences, both lines diverge more than for long sequences. This is because the ration between the removed part and the length is larger in short sequences than in long ones, i.e. if you remove 10 bp form a 20 bp sequences, only half remains, but if you remove 10 bp from a 100 bp sequences, 90% remains.

Question 1.2

Analyze and comment on the strange outliers in the plot from Question 1.1. **Use max 100 words.**

```
wt1_quant %>% filter(EffectiveLength<175 & EffectiveLength>165) %>%
ggplot(aes(x=Length, y=EffectiveLength)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, linetype = 'dashed') +
  geom_smooth(method='lm', fullrange=T) +
  scale_x_continuous(trans=log10_trans()) +
  scale_y_continuous(trans=log10_trans()) +
  xlab('Length (log10-scale)') +
  ylab('Effective length (log10-sacale)')
```




```
filterLength = filter(wt1_quant, EffectiveLength<175 & EffectiveLength>165)
knitr::kable(head(filterLength, 10))
```

Name	Length	EffectiveLength	TPM	NumReads
TCONS_00000003	636	169	0	0
TCONS_00003953	6950	170	0	0
TCONS_00003954	6493	170	0	0
TCONS_00003955	2347	170	0	0
TCONS_00003956	3130	170	0	0
TCONS_00003957	1512	170	0	0
TCONS_00003981	670	169	0	0
TCONS_00000030	1588	170	0	0
TCONS_00000032	4705	170	0	0

Name	Length	EffectiveLength	TPM	NumReads
TCONS_00000033	4675	170	0	0

We observe from the plot and the table, that there are some sequences that are always trimmed to about 170 bp effective length, no matter how long they are. By filtering these sequences we observe that they are the ones with no reads mapped to them, so the coverage information cannot be used to estimate the effective length. So 170 bp would be a default effective length.

Question 1.3

Use IsoformSwitchAnalyzeR???s `importIsoformExpression()` to import all the data into R. Convert the abundancies imported by `importIsoformExpression()` into a log2 transformed abundance matrix (using a pseudocount of 1) where columns are samples and isoform ids are stored as rownames. Report the first 4 rows as a table and discuss the advantage of a pseudocount of 1. **Use max 100 words.**

```
salmon_data = importIsoformExpression('salmon_result_part1/')
```

```
## Step 1 of 3: Identifying which algorithm was used...
##     The quantification algorithm used was: Salmon
## Step 2 of 3: Reading data...
## reading in files with read_tsv
## 1 2 3 4 5 6
## Step 2 of 3: Normalizing TxPM values via edgeR...
## Removing 2435 rows with all zero counts
## Done
```

```
salmon_data$abundance$isoform_id = NULL
pseudo_log2 = function(x){
  t = log2(x + 1)
  return(t)
}
transformed_matrix = apply(salmon_data$abundance, 2,pseudo_log2)
knitr::kable(head(salmon_data$abundance, 4))
```

	WT1	WT2	WT3	WTTPA1	WTTPA2	WTTPA3
TCONS_00000001	0.1906449	0.0000000	0.0000000	0.2498816	0.0000000	0
TCONS_00000002	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
TCONS_00000003	0.0000000	0.2208792	0.1366764	0.8380143	0.0000000	0
TCONS_00003946	0.0229655	0.0000000	0.1146470	0.0000000	0.0569531	0

```
knitr::kable(head(data.frame(transformed_matrix), 4))
```

	WT1	WT2	WT3	WTTPA1	WTTPA2	WTTPA3
TCONS_00000001	0.2517432	0.0000000	0.0000000	0.3217914	0.0000000	0
TCONS_00000002	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0
TCONS_00000003	0.0000000	0.2879205	0.1848216	0.8781480	0.0000000	0
TCONS_00003946	0.0327575	0.0000000	0.1565869	0.0000000	0.0799114	0

Pseudocounts are useful because they do not affect large numbers but help smooth the data when there are small values. The use of a pseudocount of 1 followed by a log2 transformation has the advantage of keeping values equal to 0 the same, because they might be meaningful, while affecting low values.

Question 1.4

Use tidyverse to extract the 100 most variable isoforms (aka those with highest variance) from the log2-transformed expression matrix. Provide a table with top five most variable isoforms.

```
variances = apply(transformed_matrix, 1, var)
transformed_matrix = cbind(transformed_matrix, variances)
transformed_tibble = as.tibble(transformed_matrix, rownames='isoform_id')
top_var = head(arrange(transformed_tibble, desc(variances)),100)
knitr::kable(head(top_var, 5))
```

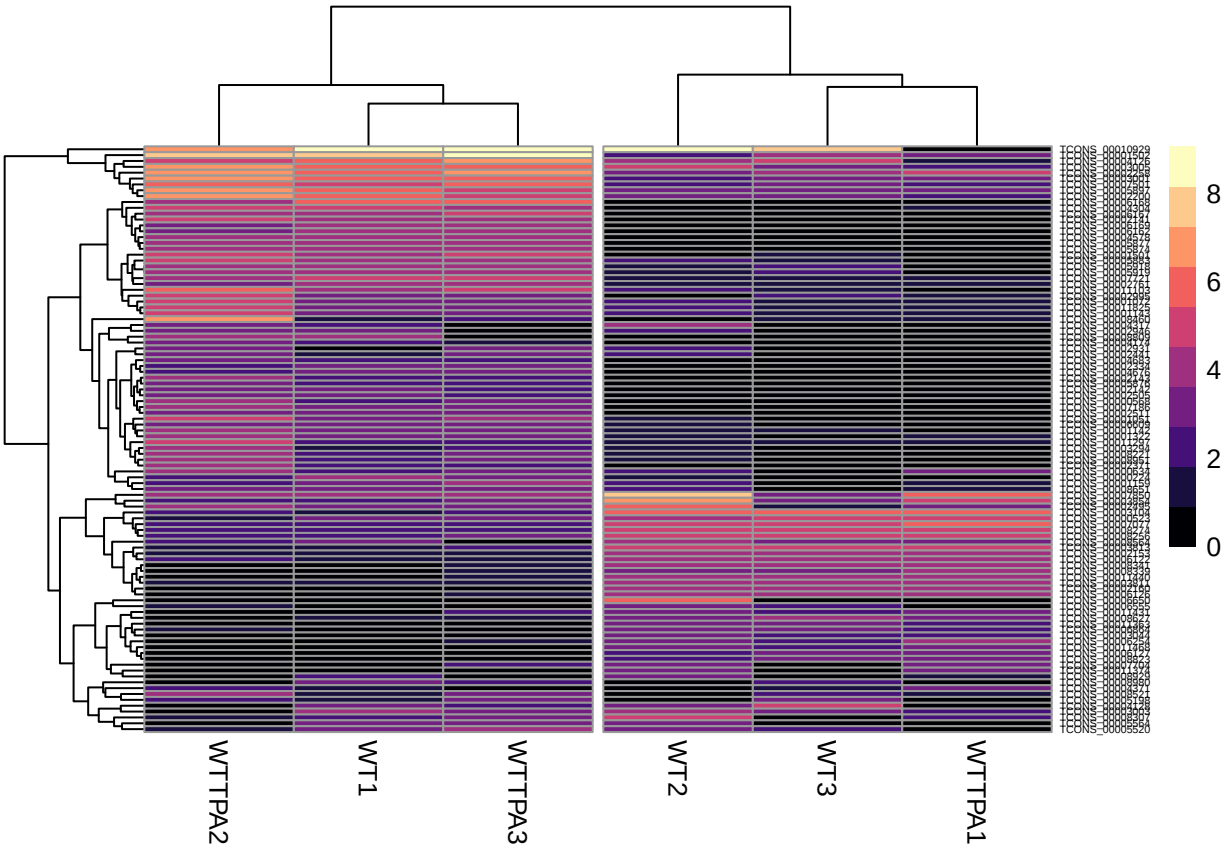
isoform_id	WT1	WT2	WT3	WTTPA1	WTTPA2	WTTPA3	variances
TCONS_00010929	8.400678	8.272262	7.776498	0.0000000	6.772383	9.057851	11.395361
TCONS_00006168	5.723089	0.000000	0.000000	0.0000000	4.359412	5.822059	8.699085
TCONS_00001502	7.803495	2.393165	4.204382	3.1846450	7.772890	8.498318	7.206314
TCONS_00008460	1.638561	0.000000	1.197683	0.9994763	7.108136	1.957551	6.346966
TCONS_00006650	0.000000	6.148895	0.000000	0.0000000	0.000000	0.000000	6.301484

These are the top 5 isoforms with the highest variance.

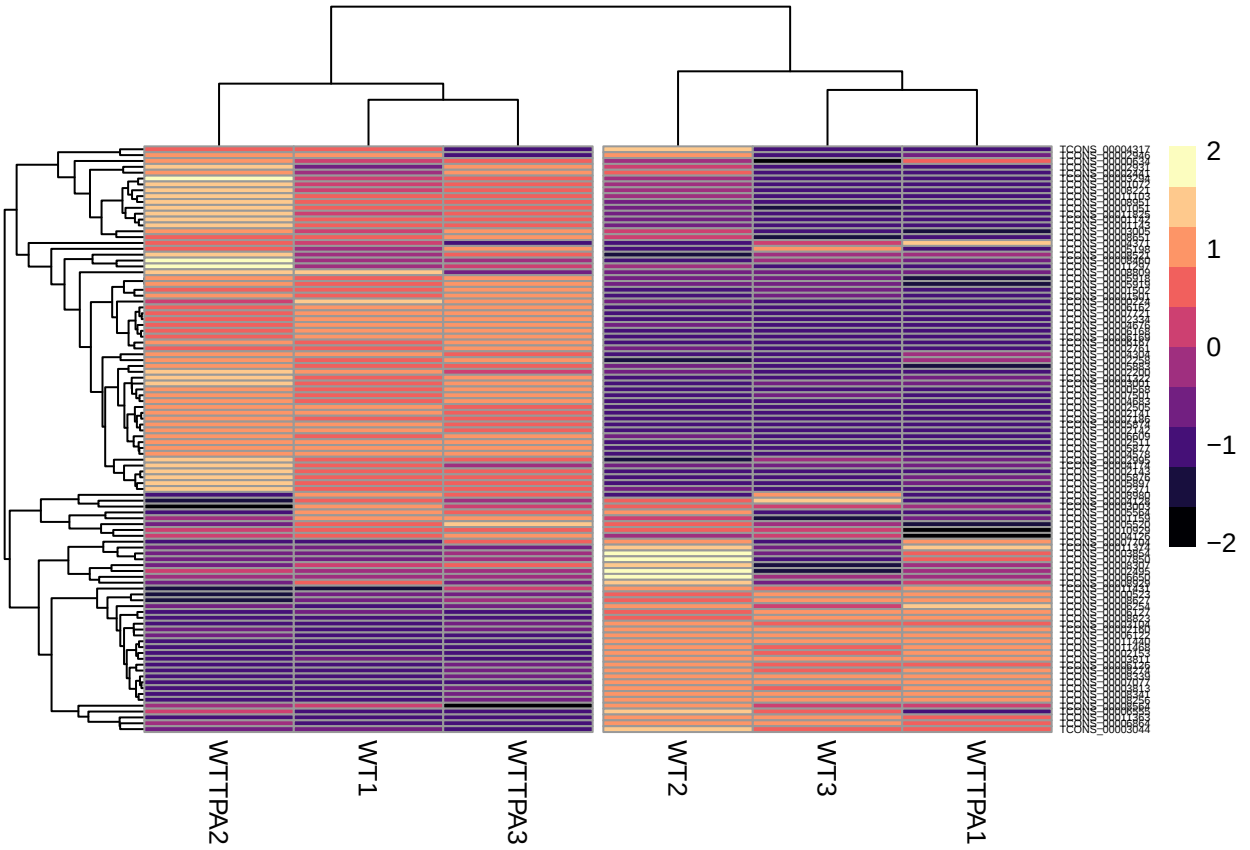
Question 1.5

Use the pheatmap package to make one visually appealing heatmap of the isoforms from 1.4 and comment on the result. Columns should be samples and rows isoforms. Furthermore, discuss pros and cons of the argument `scale = "row"` vs `scale = "none"`. **Use max 100 words.**

```
top_var_matrix = as.matrix(top_var[,2:7])
rownames(top_var_matrix) = top_var$isoform_id
pheatmap(top_var_matrix, color=magma(10), cutree_cols=2, show_rownames=T,
         scale='none', fontsize_row = 4, fontsize_col = 10)
```




```
pheatmap(top_var_matrix, color=magma(10), cutree_cols=2, show_rownames=T,
         scale='row', fontsize_row = 4, fontsize_col = 10)
```



We can observe that the samples are not clustered as expected, WT vs WTTPA, since WT1 and WTTPA1 are switched. There is a clear difference between the two groups: WTTPA2, WTTPA3, WT1 and WT2, WT3, WTTPA1. Most of the isoforms that are present in high abundance in one group are not in the other. We think that there might have been an error while labeling WT1 and WTTPA1 so that they appear in the other cluster. When using 'scale='row" the difference between the isoforms and sample clusters are visually clearer. Thanks to this normalization it is possible and safer to make comparison across rows, i.e. between different samples.

Part2: Isoform switch analysis with IsoformSwitchAnalyzeR

Use the supplied Salmon quantification subsets stored in the ???salmon_result_part2.zip??? folder (Different than the one you used in part 1!). These files contain the Salmon quantifications of 6 samples ??? 3 biological replicates of wildtype (WT) and 3 biological replicates of a knock out (KO) of a suspected splice factor ??? let us call it *factor X* for the sake of drama. Salmon was run with the `-seqBias` option.

Please note that you need IsoformSwitchAnalyzeR version > 1.1.10. You might need to update it first.

Question 2.1

Use the `importIsoformExpression` and `importRdata(addAnnotatedORFs=FALSE)` functions to create a switchAnalyzeRList object from the Salmon output supplied in the ???salmon_result_part2.zip??? folder. Use the GTF file also included in the zip file. Report the summary statistics of the resulting switchAnalyzeRList. What does the `addAnnotatedORFs=FALSE` argument do and why do you think it is enabled here?

```
packageVersion("IsoformSwitchAnalyzeR")
```

```
## [1] '1.2.0'
```

```
IsoformList1 <-  importIsoformExpression(parentDir = "./salmon_result_part2")
```

```
## Step 1 of 3: Identifying which algorithm was used...
##     The quantification algorithm used was: Salmon
## Step 2 of 3: Reading data...
## reading in files with read_tsv
## 1 2 3 4 5 6
## Step 2 of 3: Normalizing TxPM values via edgeR...
## Removing 2433 rows with all zero counts
## Done
```

```
head(data.frame(IsoformList1$counts))
```

```
##                     isoform_id      KO1      KO2 KO3          WT1       WT2
## TCONS_00000001 TCONS_00000001 1.003786 2.890696   0 4.400760e+00  0.000000
## TCONS_00000002 TCONS_00000002 0.000000 0.000000   0 0.000000e+00  0.000000
## TCONS_00000003 TCONS_00000003 0.000000 0.000000   0 1.069622e+01  2.199315
## TCONS_00003946 TCONS_00003946 0.000000 0.000000   0 0.000000e+00  0.000000
## TCONS_00003947 TCONS_00003947 0.000000 8.972704   0 1.398578e+02 81.796689
## TCONS_00003948 TCONS_00003948 0.000000 0.000000   0 1.196270e-06  0.000000
##                       WT3
## TCONS_00000001  0.000000
## TCONS_00000002  0.000000
## TCONS_00000003  1.378978
## TCONS_00003946 14.541497
## TCONS_00003947 32.194612
## TCONS_00003948  0.000000
```

```
repCount1 <- IsoformList1$counts[,2:7]
#colnames(repCount1) <- c('isoform_id','KO1','KO2','KO3','WT1','WT2','WT3')

#rownames(repCount1)

repCount1$isoform_id <- rownames(repCount1)

#head(IsoformList1$counts,4)

OurDesignMatrix <- data.frame(sampleID = c('KO1','KO2','KO3','WT1','WT2','WT3'),condition=c('KO','KO','

colnames(OurDesignMatrix) <- c('sampleID','condition')


My.SwitchAnalyzerRList <- importRdata(isoformCountMatrix = repCount1,
                                      addAnnotatedORFs = FALSE,
                                      isoformExonAnnoation = "./salmon_result_part2/subset.gtf",
                                      designMatrix = OurDesignMatrix)
```

```
## Step 1 of 5: Obtaining annotation...
## importing GTF (this may take a while)
## Step 2 of 5: Calculating gene expression...
##
  |
  |                                                                      |   0%
```

```
  |
  |=============                                                         |  20%
  |
  |=========================                                             |  40%
  |
  |=====================================                                 |  60%
  |
  |=================================================                     |  80%
  |
  |=============================================================================| 100%
```

```
## Step 3 of 5: Merging gene and isoform expression...
```

```
##
  |
  |                                                                      |   0%
  |
  |================================                                      |  50%
  |
  |=============================================================================| 100%
```

```
## Step 4 of 5: Making comparisons...
```

```
##
  |
  |                                                                      |   0%
  |
  |=============================================================================| 100%
```

```
## Step 5 of 5: Making switchAnalyzeRlist object...
## Done
```

```
# head(My.SwitchAnalyzerRList$isoformFeatures,2)
summary(My.SwitchAnalyzerRList)
```

```
## This switchAnalyzeRlist list contains:
##  10000 isoforms from 3825 genes
##  1 comparison from 2 conditions
```

```
#?importRdata

#cuffDB <- prepareCuffExample()

#designMatrix <- cummeRbund::replicates(IsoformList1$counts)[,c('rep_name','sample_name')]

#isoRepCount <- repCountMatrix(isoforms(cuffDB))
```

The addAnnotatedORFs will add annotated open reading frames, but since our data is not a quantification of known annotated transcripts, we can not use this option.

Question 2.2

Why is it essential the annotation stored in the GTF file is the exact annotation quantified with Salmon (in the context of IsoformSwitchAnalyzeR functionalities)? **Use max 100 words.**

It is essential that the GTF file is the exact genomic annotation that was quantified with Salmon, since it otherwise won't be possible to match exactly which isoforms match the same genes.

Salmon uses a annotation file to determine which part of the genome is expressed so that then it matches the reads to the specific transcribed feature. Then, IsoformSwitchAnalyzeR uses the same file to estimate which isoform (in terms of different reads) is used by each sample, so it cannot have a different gtf file with different features and coordinates.

Question 2.3

Load the supplied ???switchList.Rdata??? object into R with the `readRDS()` function. This is the result of running the whole IsoformSwitchAnalyzeR workflow on the full dataset. Make a table with the Top 10 switching genes with predicted consequences when sorting on q-values.

```
My.SwitchList <- readRDS("./hw3switchList.Rdata")


My.SwitchList.select <- select(as.tibble(My.SwitchList$isoformFeatures),
                               iso_ref, gene_ref, isoform_id, gene_id,
                               condition_1, condition_2, gene_name, gene_q_value,
                               gene_switch_q_value) %>%
  group_by(gene_name) %>%
  summarise(gene_q_value = min(gene_q_value),
            gene_switch_q_value = min(gene_switch_q_value)) %>%
  arrange(gene_switch_q_value)

knitr::kable(My.SwitchList.select[1:10,],digits=100, caption="2.3 \n Sorted genes of lowest gene switch
```

Table 5: 2.3 Sorted genes of lowest gene switch q-value

gene_name	gene_q_value	gene_switch_q_value
5830418K08Rik	0.000662452	3.175544e-64
Serbp1	0.905557000	1.478945e-19
Ablim1	0.035215400	1.155042e-15
Tef	0.039535100	4.686282e-15
Map4k4	0.000662452	4.770904e-15
Postn	0.005988670	9.818809e-14
Myo9a	0.896290000	8.903610e-13
Xrcc6	0.786097000	9.951012e-13
Snx14	1.000000000	4.031854e-12
Slmap	0.000662452	6.992658e-11

Question 2.4

Show code for how to produce switchPlot for these 10 genes and save them to your own computer. The plots should not be included in the report (only the code for how to produce it)!

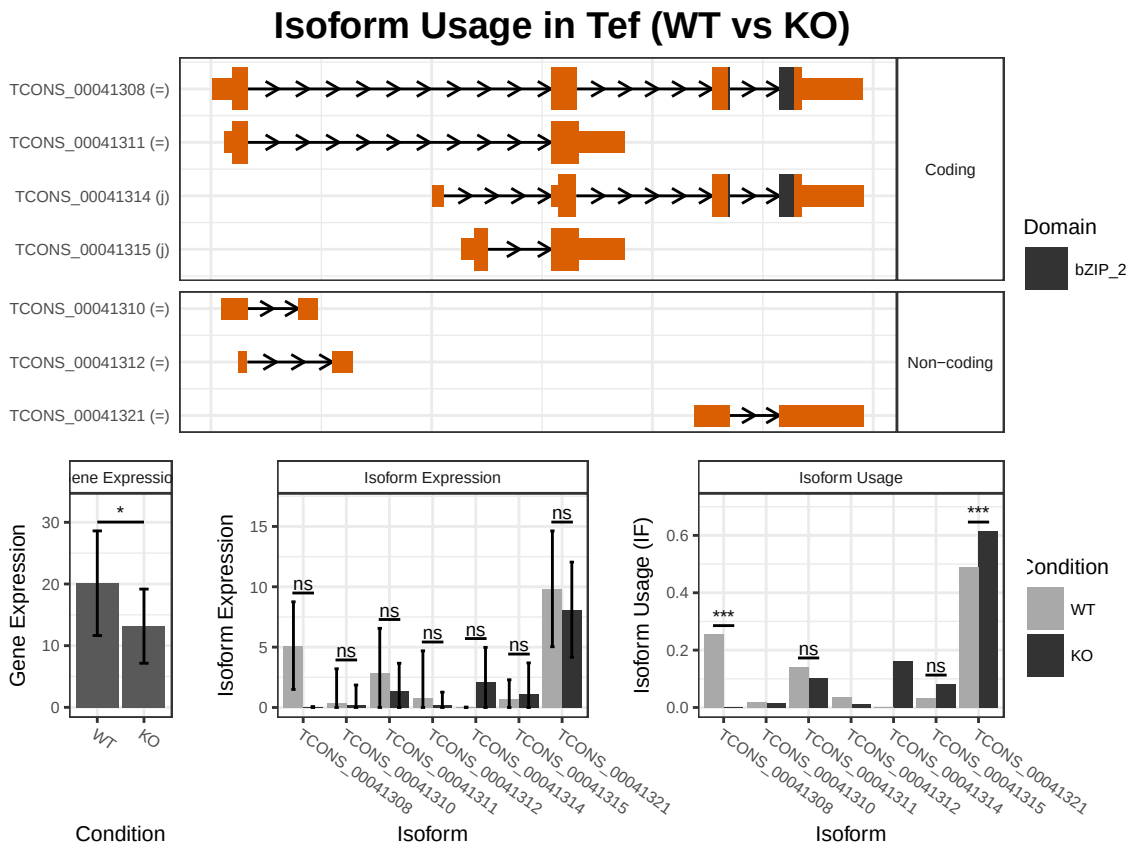
```
gene.names <- pull(My.SwitchList.select[1:10,], gene_name)

#switchPlot(switchAnalyzeRlist = My.SwitchList, gene='Tef')

plotter <- function(name){
  switchPlot(switchAnalyzeRlist = My.SwitchList, gene=name)
}
```

```
sapply(gene.names, FUN = plotter)
```

Question 2.5

Which of the top 10 genes with switches do you think is the most important? Include/produce the switchPlot for that particular gene in the report and provide references if necessary. **Use max 100 words.**

```
plotter('Tef')
```

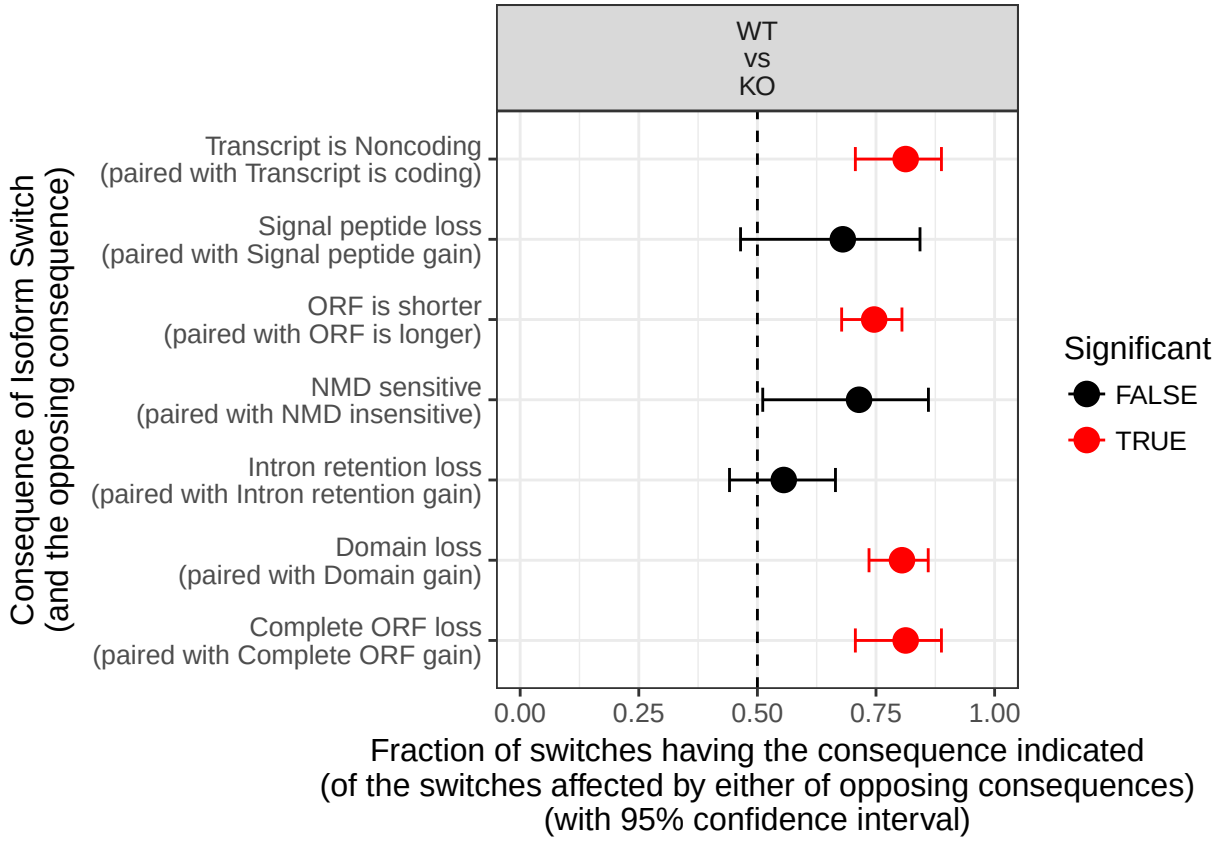


We believe that the most critical gene that is affected upon KO of *factor X* is the Tef-gene. From the Isoform Usage plot we observe that the expression of the coding mRNA is completely diminished upon KO of *factor X*. The switch will then mean that only the noncoding isoforms of the gene is expressed. In addition we can see that the gene expression is significantly different between the two conditions. The Tef gene encodes for Thyrotroph embryonic factor, that is a transcription factor involved in embryonic development of the pituitary gland.

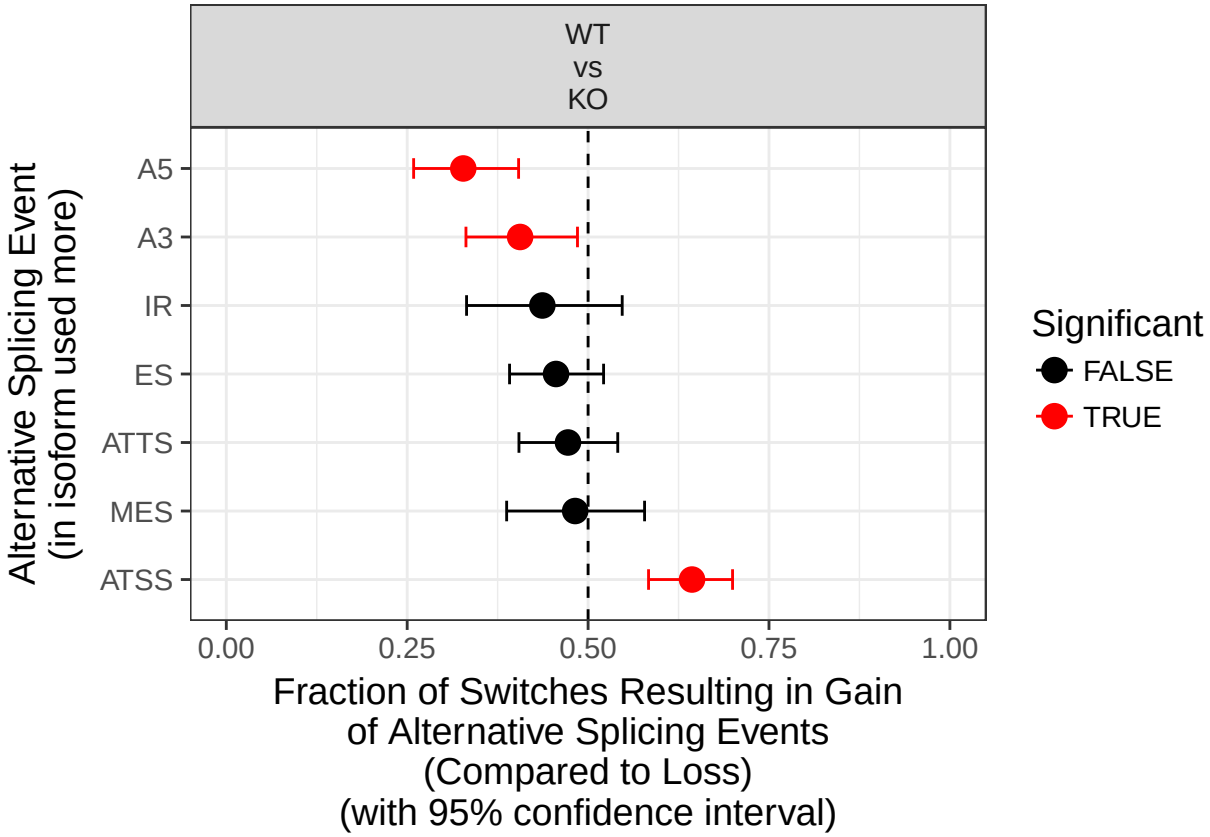
Question 2.6

Plot the global enrichment of switch consequences and alternative splicing and comment on it. What are the general patterns? **Use max 100 words.**

```
extractConsequenceEnrichment(
    My.SwitchList,
    consequencesToAnalyze='all',
    analysisOppositeConsequence = TRUE
)
```




```
extractSplicingEnrichment(My.SwitchList)
```



From the Consequence enrichment plot, we see that by KO of *factor X* there is a significant decrease in coding transcripts. We also see a general tendency for shorter open reading frames, loss of known domains and complete open reading frame loss. *Factor X* seems to be a very important factor for correct splicing, and from the Splicing enrichment plot, we observe that KO of *factor X* leads to a decrease in alternative 5' donor-sites and a decrease in alternative 3' acceptor-sites. Probably most importantly we see a significant increase in alternative transcription start sites.