

homework3

```
library(IsoformSwitchAnalyzeR)

## Loading required package: limma
## Loading required package: DEXSeq
## Loading required package: BiocParallel
## Loading required package: Biobase
## Loading required package: BiocGenerics
## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##   clusterExport, clusterMap, parApply, parCapply, parLapply,
##   parLapplyLB, parRapply, parSapply, parSapplyLB
## The following object is masked from 'package:limma':
##
##   plotMA
## The following objects are masked from 'package:stats':
##
##   IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##   anyDuplicated, append, as.data.frame, basename, cbind,
##   colnames, dirname, do.call, duplicated, eval, evalq, Filter,
##   Find, get, grep, grepl, intersect, is.unsorted, lapply, Map,
##   mapply, match, mget, order, paste, pmax, pmax.int, pmin,
##   pmin.int, Position, rank, rbind, Reduce, rownames, sapply,
##   setdiff, sort, table, tapply, union, unique, unsplit, which,
##   which.max, which.min
## Welcome to Bioconductor
##
##   Vignettes contain introductory material; view with
##   'browseVignettes()'. To cite Bioconductor, see
##   'citation("Biobase)"', and for packages 'citation("pkgname)".

## Loading required package: SummarizedExperiment
## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
```

```

## The following object is masked from 'package:base':
##
##     expand.grid
## Loading required package: IRanges
## Loading required package: GenomeInfoDb
## Loading required package: DelayedArray
## Loading required package: matrixStats
##
## Attaching package: 'matrixStats'
## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians
##
## Attaching package: 'DelayedArray'
## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges
## The following objects are masked from 'package:base':
##
##     aperm, apply, rowsum
## Loading required package: DESeq2
## Registered S3 methods overwritten by 'ggplot2':
##   method      from
##   [.quosures   rlang
##   c.quosures   rlang
##   print.quosures rlang
## Loading required package: AnnotationDbi
## Loading required package: RColorBrewer
## Loading required package: ggplot2
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.2.1 --

## v tibble  2.1.1    v purrr   0.3.2
## v tidyr   0.8.3    v dplyr   0.8.1
## v readr   1.3.1    v stringr 1.4.0
## v tibble  2.1.1    v forcats 0.4.0

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::collapse() masks IRanges::collapse()
## x dplyr::combine() masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count() masks matrixStats::count()
## x dplyr::desc() masks IRanges::desc()
## x tidyr::expand() masks S4Vectors::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::first() masks S4Vectors::first()
## x dplyr::lag() masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()

```

```
## x purrr::reduce()      masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()      masks S4Vectors::rename()
## x dplyr::select()      masks AnnotationDbi::select()
## x purrr::simplify()    masks DelayedArray::simplify()
## x dplyr::slice()       masks IRanges::slice()

library(pheatmap)
```

Question 1.1

```
wt1_quant <- read_tsv("./salmon_result_part1/salmon_result_part1/WT1/quant.sf")
```

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

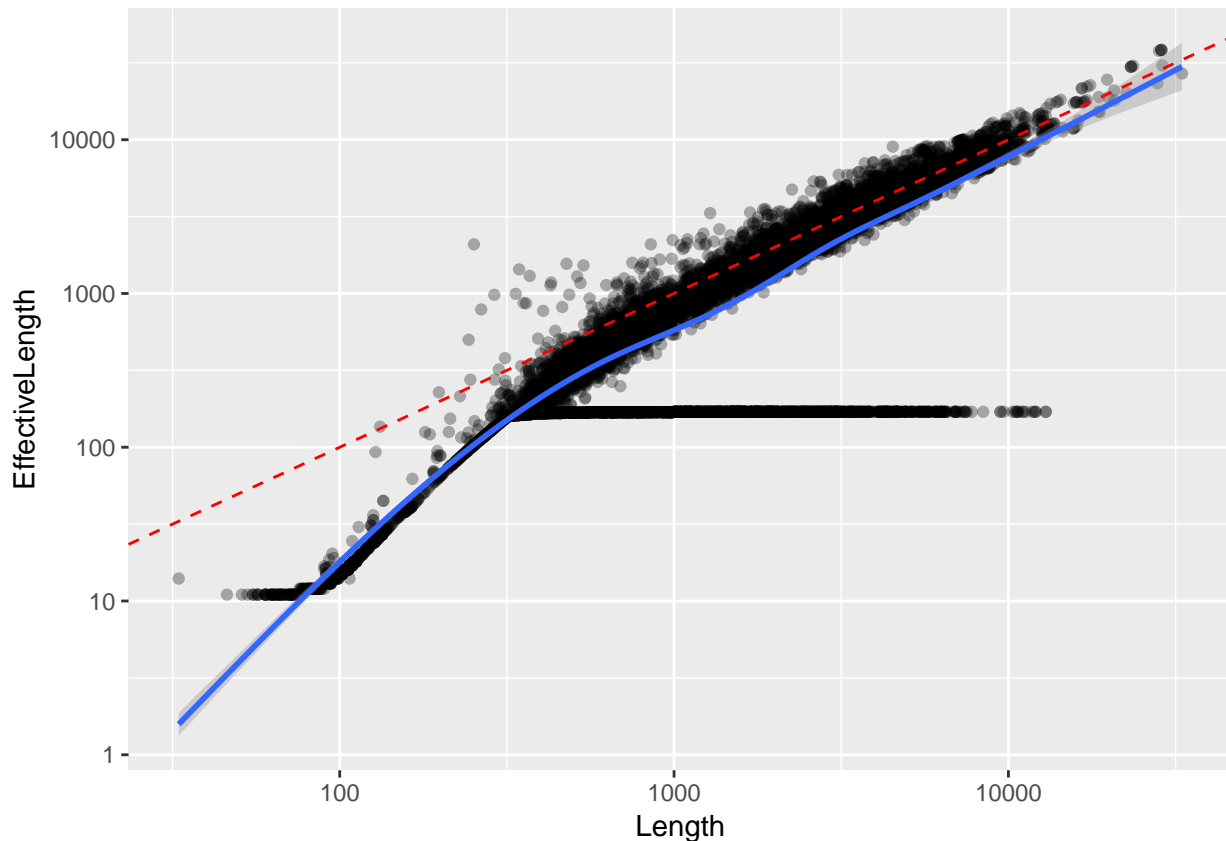
Here we plotted the isoform length versus effective length, and used geom smooth to show the trend line of the relationship between isoform length and effective length. The dash line shows the relationship where isoform lengths are equal to effective lengths.

From the plot, we can see that the actual relationship of isoform length and effective length was not linear relationship where isoform lengths are equal to effective lengths, but the effective lengths were actually shorter than the isoform length. This is due to the fact that during the cDNA library preparation step we fragmented the cDNA and filtered out the fragments that were smaller than certain threshold. These short fragments were mostly from the both end of cDNA; therefore the effective length is shorter than actual isoform length.

Moreover, we can see that the deviation from identity line for the shorter isoform was much greater than the longer isoform. This is because the filtered out fragment make up to greater proportion based pair in shorter isoform than longer isoform given the certain filtering threshold.

```
wt1_quant %>% ggplot(aes(x=Length, y=EffectiveLength)) +
  scale_x_continuous(trans='log10') + scale_y_continuous(trans='log10') +
  geom_point(alpha=0.3) +
  geom_smooth() + geom_abline(color = "red", linetype=2)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Question 1.2

From the figure in question 1.1, we can see the the set of outliers that form the sigmoidal-like curve in the plot. ##### I DON'T KNOW WHAT TO WRITE HERE #####

Question 1.3

Here we imported salmon data and transformed the abundance matrix by the function $\log_2(x + 1)$ when x was abundance value. We can see the first 4 transcripts in transformed abundance matrix below.

```
all_salmons <- importIsoformExpression(parentDir =
                                     "./salmon_result_part1/salmon_result_part1/")

## Step 1 of 3: Identifying which algorithm was used...
##   The quantification algorithm used was: Salmon
##   Found 6 quantification file(s) of interest
## Step 2 of 3: Reading data...
## reading in files with read_tsv
## 1 2 3 4 5 6
## Step 3 of 3: Normalizing FPKM/TxPM values via edgeR...
## Done

salmon_matrix <- as.matrix(all_salmons$abundance[,2:ncol(all_salmons$abundance)])
rownames(salmon_matrix) <- all_salmons$abundance[,1]

transformed_salmon <- log2(salmon_matrix+1)
```

```
transformed_salmon[1:4,]
```

```
##           WT1      WT2      WT3      WTTA1      WTTA2 WTTA3
## TCONS_00000001 0.2973299 0.0000000 0.0000000 0.3822156 0.0000000    0
## TCONS_00000002 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000    0
## TCONS_00000003 0.0000000 0.2984888 0.2253968 1.0124265 0.0000000    0
## TCONS_00003946 0.0392366 0.0000000 0.1913649 0.0000000 0.0564598    0
```

The reason why we added pseudocount = 1 to the abundance before apply \log_2 to the abundance value is that if the abundance of the transcripts in some condition were 0, when we applied \log_2 to 0 the value would become infinity and would not be applicable for further down stream analysis. Adding pseudocount = 1 to the abundance before transform by \log_2 will prevent this problem, and also \log_2 of 1 is 0 which is make sense for further analysis as the starting value before transform is already 0.

Question 1.4

We used tidyverse to extract top 100 most variable isoforms as shows by code below. We first converted abundance matrix to tibble. Then we used mutate() to apply the variance function in a rowwise manner by called rowwise() function before mutate(). After that we sorted the tibble by variance and sliced out top 100 transcript with highest variance between samples.

```
salmon_tibble <- as_tibble(transformed_salmon, rownames=NA)

top100var <- salmon_tibble %>% rownames_to_column() %>% rowwise() %>%
  mutate(variance=var(c(WT1, WT2, WT3, WTTA1, WTTA2, WTTA3))) %>%
  arrange(desc(variance)) %>% slice(1:100)

head(top100var, 5)
```

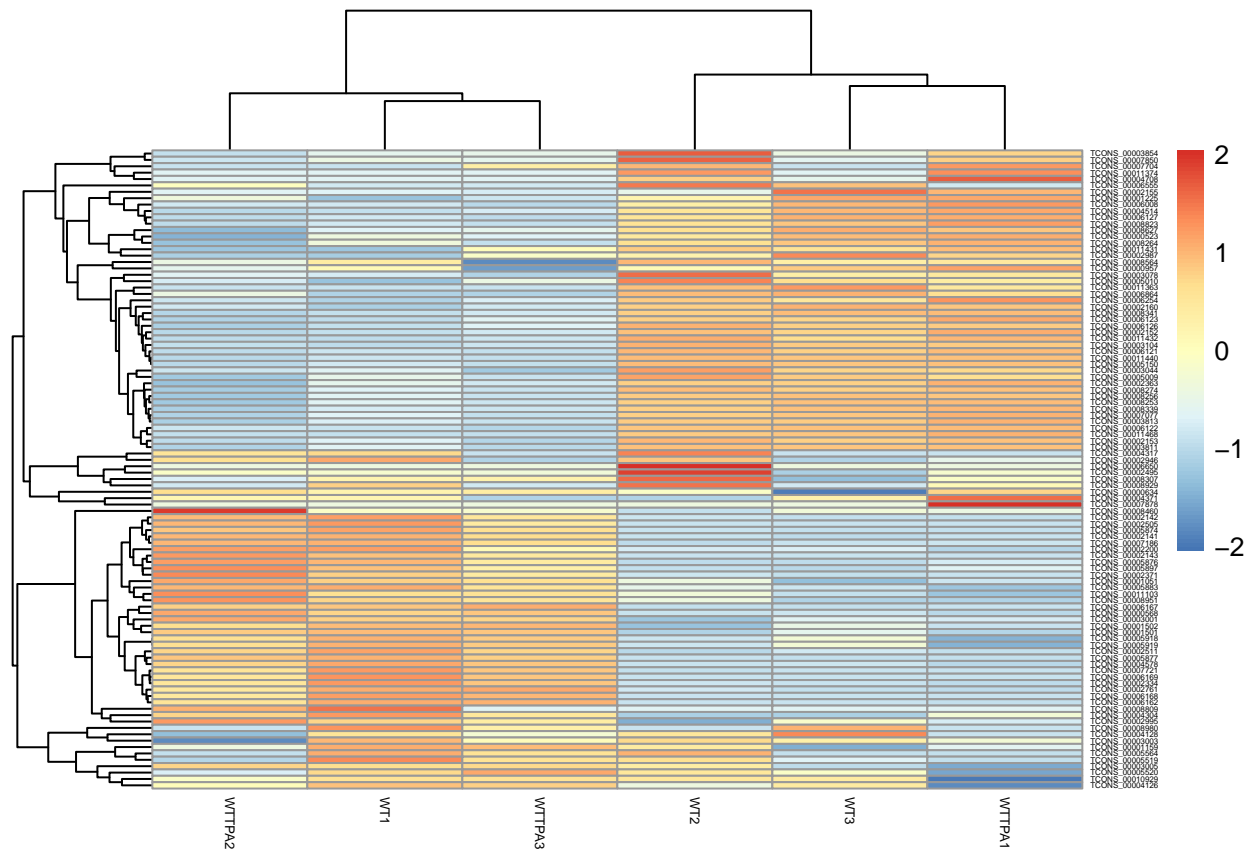
```
## Source: local data frame [5 x 8]
## Groups: <by row>
##
## # A tibble: 5 x 8
##   rowname      WT1      WT2      WT3 WTTA1 WTTA2 WTTA3 variance
##   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 TCONS_00010929 8.66  8.33  8.08  0     6.27  8.66   11.5
## 2 TCONS_00006168 5.98  0     0     0     3.88  5.44    8.27
## 3 TCONS_00006650 0     6.21  0     0     0     0     6.42
## 4 TCONS_00001502 8.07  2.44  4.50  3.44  7.26  8.10    6.20
## 5 TCONS_00003104 1.54  6.33  5.91  5.88  1.66  1.91    5.68
```

Question 1.5

We plotted the heatmap of matrix contain the transformed abundance value of top 100 transcripts with highest variance among samples using pheatmap function.

From the heatmap we can see that the samples could be clustered by the expression profile into 2 groups. The first group contained WT2, WT3 and WTTA1, and the second group contained WT1, WTTA2 and WTTA3). We can also see that those 2 groups clearly had different expression profile. The genes in top clusters of first group were expressed more than the bottom cluster, while the genes in top clusters of second group were expressed less than the bottom cluster.

```
top100var_mat <- as.matrix(top100var[,2:7])
rownames(top100var_mat) <- as.data.frame(top100var)[,1]
pheatmap(top100var_mat, show_rownames = TRUE, scale = "row", fontsize_row = 2.5,
  fontsize_col = 5)
```



Moreover, we can see that in our code the argument scale was changed from the default which is “none” to “row”. This would tell the function to normalize the \log_2 abundance value of all genes in each sample. The advantage of normalization is that it will put the \log_2 abundance value on the same scale. This will enable us to compare the abundance value of the same gene between samples, and it also helps us to clearly see the difference between high and low expression genes across all samples when plotting a heatmap.

Question 2

```
all_salmons2 <- importIsoformExpression(parentDir =
                                         "./salmon_result_part2/salmon_result_part2/")

## Step 1 of 3: Identifying which algorithm was used...
##      The quantification algorithm used was: Salmon
##      Found 6 quantification file(s) of interest
## Step 2 of 3: Reading data...
## reading in files with read_tsv
## 1 2 3 4 5 6
## Step 3 of 3: Normalizing FPKM/TxPM values via edgeR...
## Done
```

Question 2.1

First we loaded data into `switchAnalyzeRList` object using `importRdata`. The summary statistics of resulting `switchAnalyzeRList` were as below.

```

designMat <- data.frame(sampleID=colnames(all_salmons2$abundance)[-1],
                      condition=
                        str_replace(colnames(all_salmons2$abundance)[-1], "[0-9]", ""))

switchAnalyzeRlist <- importRdata(isoformCountMatrix = all_salmons2$counts,
                                isoformRepExpression = all_salmons2$abundance,
                                designMatrix = designMat,
                                isoformExonAnnotation =
                                  "./salmon_result_part2/salmon_result_part2/subset.gtf",
                                addAnnotatedORFs=FALSE)

```

```

## Step 1 of 6: Checking data...
## Step 2 of 6: Obtaining annotation...
##      importing GTF (this may take a while)
## Step 3 of 6: Calculating gene expression and isoform fraction...
##      2433 ( 24.33%) isoforms were removed since they were not expressed in any samples.
## Step 4 of 6: Merging gene and isoform expression...
##
|
|
|
|=====| 50%
|
|=====| 100%
## Step 5 of 6: Making comparisons...
##
|
|
|
|=====| 100%
## Step 6 of 6: Making switchAnalyzeRlist object...
## Done
switchAnalyzeRlist

```

```

## This switchAnalyzeRlist list contains:
## 7567 isoforms from 3304 genes
## 1 comparison from 2 conditions (in total 6 samples)

```

In total there were 7567 isoforms in the data set, and those isoforms came from 3304 genes. The samples came from two groups including the organism that splice factor X were knocked out and the wild type organism.

The options `addAnnotatedORFs` will be used only when we have the GTF file that specified the ORF regions of each isoform in our data set, and want to add that annotation about the position of ORF on our transcript data. This ORF information can directly be used by `IsoformSwitchAnalyzeR` for downstream analysis such as for analyzing the consequences of isoform switch instead of having to predict ORF from `analyzeORF()` function. However, we disabled this option here by setting it to `FALSE` as we did not have GTF file contain ORF information.

Question 2.2

The reason why the annotation in the GTF file must be the exact annotation quantified with Salmon is because the IsoformSwitchAnalyzeR will annotate the isoform in isoformRepExpression data frame by matching the isoform_id in data frame with the isoform_id in GTF file. The isoform_id is not the fixed id for each transcript isoform unlike GenBank accession number/ENSEMBL id for transcripts. Therefore, the isoform_id can be changed between experiment. That is why we need to make sure that the annotation in GTF file is the exact annotation of the same data set from salmon in order to get correct annotation.

Question 2.3

The table of top 10 switching genes with consequences sorted by q-values could be made by using below code. The top 10 genes were 5830418K08Rik, Ablim1, Tef, Xrcc6, Snx14, Slmap, Rac1, Fbxw7, Pld2 and Rrbp1, respectively.

```
switchList <- readRDS("./hw3switchList.Rdata")
top10switch <- extractTopSwitches(switchList, filterForConsequences = TRUE,
                                sortByQvals=TRUE, n=10)
top10switch
```

##	gene_ref	gene_id	gene_name	condition_1
## 1	geneComp_00100550	XL0C_047302	5830418K08Rik	WT
## 2	geneComp_00076087	XL0C_023295	Ablim1	WT
## 3	geneComp_00068215	XL0C_015573	Tef	WT
## 4	geneComp_00068223	XL0C_015581	Xrcc6	WT
## 5	geneComp_00101368	XL0C_048111:Snx14	Snx14	WT
## 6	geneComp_00066816	XL0C_014190	Slmap	WT
## 7	geneComp_00089842	XL0C_036766	Rac1	WT
## 8	geneComp_00081221	XL0C_028310	Fbxw7	WT
## 9	geneComp_00058485	XL0C_006025	Pld2	WT
## 10	geneComp_00080160	XL0C_027267	Rrbp1	WT

##	condition_2	gene_switch_q_value	switchConsequences	Gene	Rank
## 1	KO	3.175544e-64	TRUE	1	1
## 2	KO	1.155042e-15	TRUE	2	2
## 3	KO	4.686282e-15	TRUE	3	3
## 4	KO	9.951012e-13	TRUE	4	4
## 5	KO	4.031854e-12	TRUE	5	5
## 6	KO	6.992658e-11	TRUE	6	6
## 7	KO	8.587909e-10	TRUE	7	7
## 8	KO	7.331074e-09	TRUE	8	8
## 9	KO	1.277562e-08	TRUE	9	9
## 10	KO	1.922603e-08	TRUE	10	10

Question 2.4

To made switch plot for top10 genes we used code below. The conditions were “KO” and “WT”.

```
top10genes_name <- top10switch[, "gene_name"]
switchCondition <- switchList$conditions[,1]

for(gene in top10genes_name){
  # best resolution pics that are not too big
  png(paste0(gene, ".png", collapse = ""), width = 1200, height = 1200, res=160)
  switchPlot(switchList, gene=gene, condition1 = switchCondition[1],
            condition2 = switchCondition[2])
  dev.off()
```



```
}
```

Question 2.5

We suspected that the knocked out gene coded for splicing factor, so we expected that the genes that were relevant to this knocking out should show clear splicing changes not other possible structural change such as transcription start site changes. This narrowed genes down from 10 genes to Rac1 where the switch between isoform TCONS_00104606 and TCONS_00104607 was clearly can happened only by splicing change. While in other genes the changes in isoform structure could also be caused by TSS change. Moreover, the isoform with significantly increase in usage in knocked out mouse was non-coding isoform, while the isoform with significantly higher isoform usage in wild type mouse was the normal coding isoform that could be translated into functional protein. This means that this gene loss function in knocked out mouse due to isoform switch. Therefore, we picked Rac1 as the most important gene.

Question 2.6

From the global enrichment plot the general, changes in isoform structure that were significant ($FDR < 0.05$) were ORF loss, domain loss, shortening of ORF, and conversion of transcript to non-coding transcript. All of these general changes can be introduced by both change in splicing pattern or change in transcription start site. So, we could not clearly say whether these pattern supported our original hypothesis that factor X might be splicing factor or not.

If factor X was actually splicing factor, we expected to observe that majority of switching would show alternative splicing. However, from alternative splicing analysis, we can see that less than half of switching showed the sign of MES loss, IR gain, ES loss and ATTs gain also with FDR higher than 0.05. Only ATSS was found in more than half of switching events with FDR < 0.05 .

So, in related to our original hypothesis, we could not say yet that factor X is splicing factor as the results here not firmly supported this hypothesis.

```
extractConsequenceEnrichment(  
  switchList,  
  consequencesToAnalyze='all',  
  analysisOppositeConsequence = TRUE,  
  returnResult = FALSE  
)
```

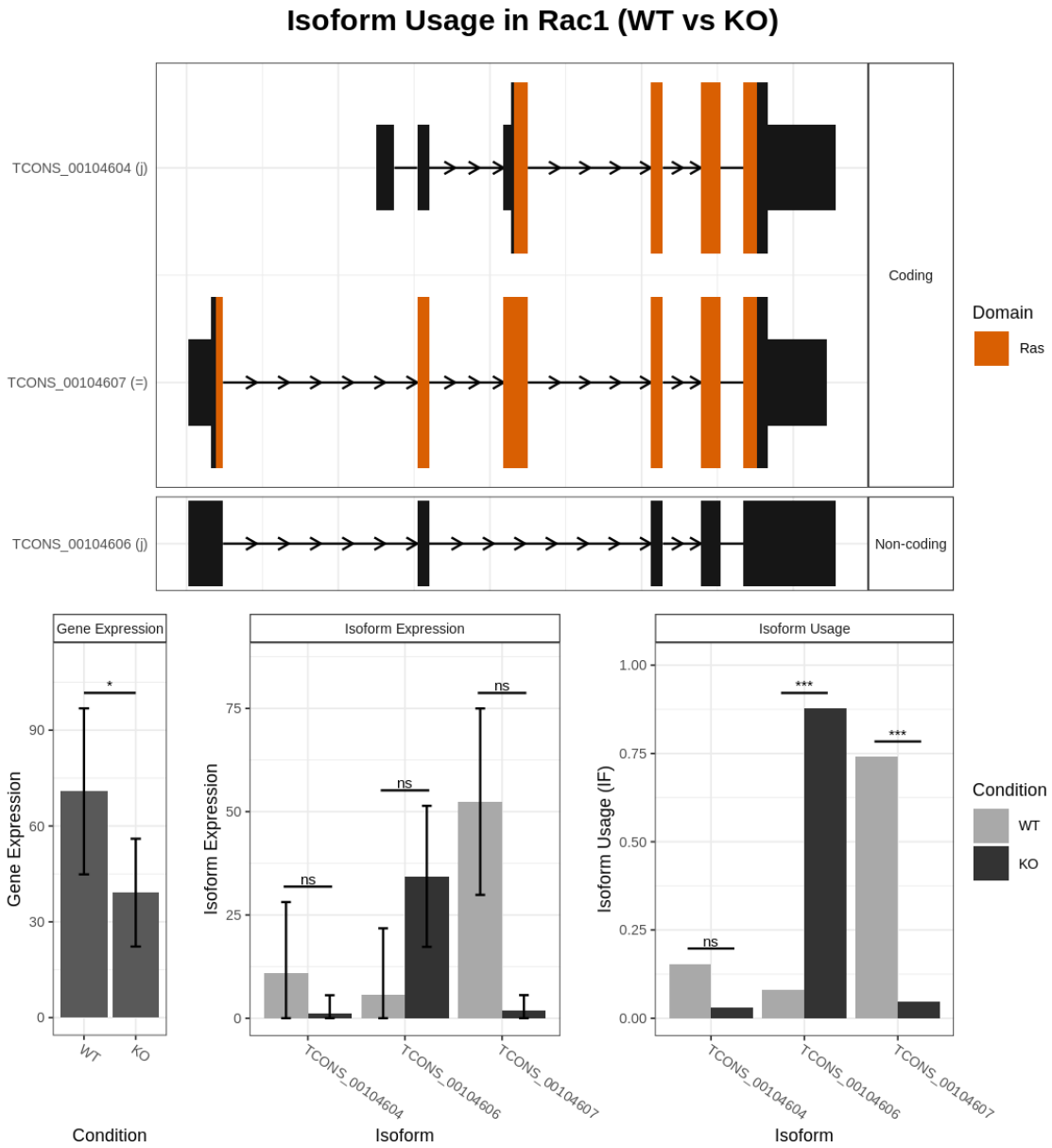
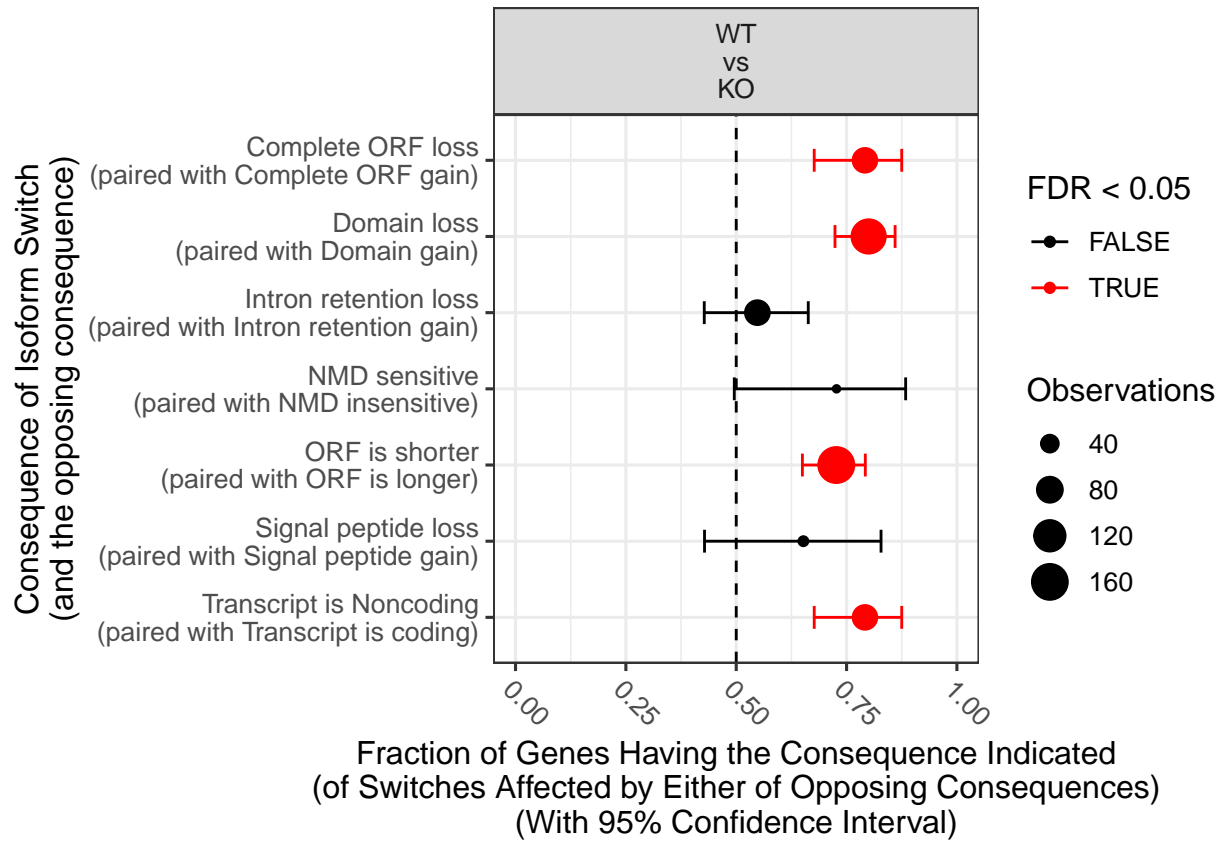


Figure 1: switchPlot of Rac1



```
extractSplicingEnrichment(
  switchList,
  returnResult = FALSE
)
```

