Assignment 3 group 6

Adham Khaled, Juan Manuel Medina, Antonio Ortega, Isabella Skandorff & Andreas Vincent

May 30, 2017

```
library("ggplot2")
library("cowplot")
library("reshape2")
library("dplyr")
library("gridExtra")
theme_set(theme_bw())
```

Part 1

(1)

Just because IRX3 shows a higher signal than RXR in both microarray experiments, it does not necessarily means that it is more expressed, as this signal can also be influenced by two main sources of variations unrelated to the gene expression: a global variation (a differential RNA quality of both genes, a different sample preparation process, etc) and a gene-specific variation (a differential tendency of both genes of dying, their potential inherent biological variations, etc). Both variations can be reduced by taking replicates, and the systematic variation can be specifically tackled by performing a systematic analysis of normalized data.

(2)

```
df <- read.table("data_for_part_1/normalized_data.txt", h = F)
names(df) <- paste(rep(c("HIV", "Control"), each = 5), 1:5, sep = "-")
# t-test p-values between HIV and Control
pvals <- apply(df, 1, function (x) t.test(x[1:5], x[6:10])$p.value)
```

(3)

```
# Genes with a p-value less than 0.05
diff_genes <- sum(pvals < 0.05) # 1911

# Expected FP = threshold% used in the t.test -> 0.05% of the t. tests that reject the H0 are wrong
FP1 <- round(0.05 * diff_genes, 0) # 96
```

The expected number of false positives is 5% of the set of significant results which is 96, or 5% of the 1911 significant differentially expresses genes (p-value$<$ 0.05).

(4)

```
# Bonferroni multiple testing correction
bonferroni_pvals <- p.adjust(pvals, method = "bonferroni")
```

```r
# Number of genes with p-value < 0.2 with Bonferroni correction
sum(bonferroni_pvals < 0.2) # 0
```

```
## [1] 0
```

```r
# BH multiple testing correction
BH_pvals <- p.adjust(pvals, method = "BH")

# Number of genes with p-value < 0.2 with BH correction
sum(BH_pvals < 0.2) # 12
```

```
## [1] 12
```

```r
# Expected FP = threshold% used in the t.test -> 0.2% of the t. tests are expected to be FP
FP2 <- round(0.2 * sum(BH_pvals < 0.2), 0) # 2
```

The number of significant bonferroni adjusted genes was 0, while the FDR method yielded 12 significant genes. The expected number of false positives under FDR was 2, or 2% of the significant FDR adjusted genes (p-value <0.2).

(5)

```r
#storing the means in the same dataframe as the original gene expression set
df$D_mean <- rowMeans(df[, c(1:5)])
df$C_mean <- rowMeans(df[, c(6:10)])

my_foldchanges <- log2(df[, 11]) - log2(df[, 12])
```

(6)

```r
# Report the fold changes for the genes with a FDR < 0.2 (using the BH method)
genes_FDR_lower_0.2 <- log2(df[which(BH_pvals < 0.2), 11]) -
                       log2(df[which(BH_pvals < 0.2), 12])

genes_FDR_lower_0.2
```

```
##  [1] 0.03680802 0.33677927 0.06387247 0.15993496 0.60119418 0.33087497
##  [7] 0.08063546 0.09060348 0.08200741 0.17300186 0.09105046 0.16164736
```

It is interesting to point out that the 12 genes with a significative fold change are all upregulated in the HIV patients, where the gene with most fold change has a 0.6 fold change (it is 60% more expressed than the same gene in the control condition) and the gene with the lower fold change has a 0.04 fold change # Part 2

Part 3

```r
gene <- read.table("data_for_part_3/cuffdiff_gene_differential_expression.txt", h = T)
transcript <- read.table("data_for_part_3/cuffdiff_transcript_differential_expression.txt", h = T)
splicing <- read.table("data_for_part_3/cuffdiff_splicing_differential_expression.txt", h = T)
names(transcript)[1] <- "transcript_id" # changing test_id column name to transcript_id
```

3.2

```
# Make two new dataframes from gene and transcript containing all rows with expression in at least one
# of the conditions, that is, with an expression value < 0 in column 8 or 9 (value 1 or value2)
gene_expressed <- gene[gene$value_1 | gene$value_2 != 0, ]
transcript_expressed <- transcript[which(transcript$value_1 | transcript$value_2 != 0), ]
```

3.3

```
# Number of genes and transcripts that were expressed and how many
# were significantly differentially expressed between conditions (column 14 or significant = "yes")
nrow(gene_expressed)        # 26
```

```
## [1] 26
```

```
nrow(transcript_expressed)  # 98
```

```
## [1] 98
```

```
# dplyr incorporation
gene_expressed %>% filter(significant == "yes") %>% nrow                          # 12
```

```
## [1] 12
```

```
nrow(transcript_expressed[which(transcript_expressed$significant == "yes"), ])  # 11
```

```
## [1] 11
```

3.4

```
# Make two reduced dataframes from gene_expressed and transcript_expressed and merge them based on the
# gene ids
gene_reduced <- gene_expressed[, c(2, 3, 8, 9)]
transcript_reduced <- transcript_expressed[, c(1, 2, 8, 9)]

gen_tr <- merge(gene_reduced, transcript_reduced, by = "gene_id",
                              suffixes = c(".GENE", ".TRANSCRIPT"))

dim(gen_tr)    # 98 rows, 7 columns
```

```
## [1] 98  7
```

3.5

```
# Calculate the IF for each transcript in both conditions (IF = isoform_exp / gene_exp) and the dIF
# (IF2 - IF1)

gen_tr$IF1 <- gen_tr$value_1.TRANSCRIPT /
                              gen_tr$value_1.GENE

gen_tr$IF2 <- gen_tr$value_2.TRANSCRIPT /
  gen_tr$value_2.GENE

gen_tr$dIF <- gen_tr$IF2 - gen_tr$IF1

# Overwrite the IF1 and IF2 columns of the dataframe
gen_tr$IF1 <- replace(gen_tr[, 8], is.na(gen_tr[, 8]), 0)
```

```
gen_tr$IF2 <- replace(gen_tr[, 9], is.na(gen_tr[, 9]), 0)

# Recalculate the dIF with the corrected IF1 and IF2 values and overwrite
gen_tr$dIF <- gen_tr$IF2 - gen_tr$IF1
```

If the expression value of a certain gene in condition is 0, when we try to obtain the IF dividing the isoform expression values by it, we obtain a non-sense mathematical result due to the fact that we are dividing by 0, and hence, R traduces this non-sense result into a NaN.

However, these NaN values could be reconverted to 0, because we can assume that, as there is no gene expression under the specified condition, and subsequently there is no isoform expression under that condition, the IF in that condition is also null.

So if we change the NaNs of columns 8 and 9 to 0, we can obtain proper results of dIF in column 10.

3.6

```
# Calculate the mean and median dIF value and discuss
mean(gen_tr$dIF)
```

```
## [1] -5.503846e-09
```

```
median(gen_tr$dIF)
```

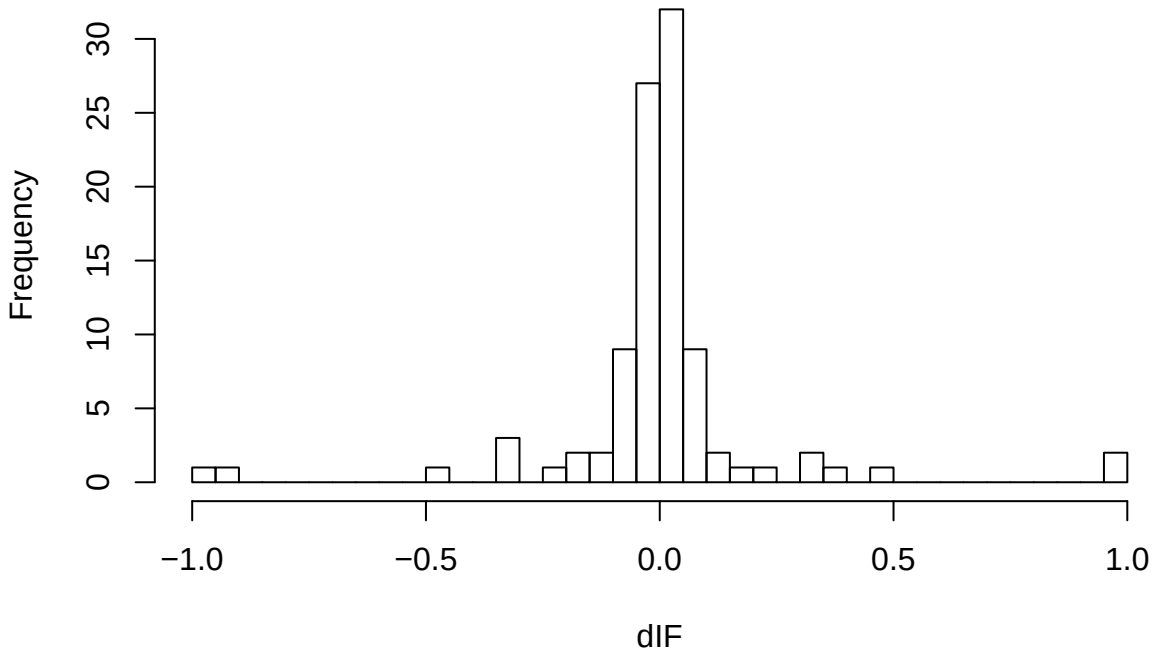
```
## [1] 1.720652e-05
```

Regardless of performing the process taking out the NaNs rows (6) or keeping them as 0, we obtain a mean and a median very close to 0. This agrees with an expected mean dIF of 0 since adding up all the dIF values for the different isoforms of the same gene is 0 by definition (although the median does not have to be 0). However, we are not obtaining a 0 mean due to numerical errors in R

Mean and median dIF values keeping the NaNs ds 0 * Mean -> -5.503846e-09 * Median -> 1.720652e-05

Mean and median dIF values taking out the NaNs * Mean -> 0.0092 * Median -> 0.0015

```
# Histogram of the distribution of dIF values
hist(gen_tr$dIF, breaks = 40, main = "Distribution of dIF values", xlab = "dIF")
```



3.7

```
# Subset the merged dataframe to only contain genes with dIF > +0.25 and dIF < -0.25
# Then, add the p-value from the splicing data frame using the match() function
gen_tr_pot <- gen_tr[which(gen_tr$dIF
                                        > 0.25 | gen_tr$dIF < -0.25), ]
# p-values from splicing dataframe which same gene_id as the ones in gen_tr_potential_
# _switch dataframe
gen_tr_pot$pvals <- splicing[match(gen_tr_pot$gene_id, splicing$gene_id), 12]
```

3.8

```
# TRANSCRIPT ID of the gene with the lowest p-value in the gen_tr_pot DF
filter(gen_tr_pot, gene_id == gen_tr_pot %>% arrange(pvals) %>% .$gene_id %>%
         as.character() %>% unique %>% .[1]) %>%  .$transcript_id %>% as.character
```

```
## [1] "TCONS_00000021" "TCONS_00000022"
```

```
 # TCONS_00000021 #

# GENE NAME of the gene with the lowest p-value in the gen_tr_pot DF
gen_tr_pot[which.min(gen_tr_pot$pvals), 2] %>% as.character
```

```
## [1] "uc001aeo.3"
```

```
 # uc001aeo.3 #

# dIF VALUE of the gene with the lowest p-value in the gen_tr_pot DF
# The dIF values.
 filter(gen_tr_pot, gene_id == gen_tr_pot %>% arrange(pvals) %>% .$gene_id %>%
```

```
          as.character() %>% unique %>% .[1]) %>% .$dIF
```

```
## [1] -0.3426094  0.3426083
```

```
# P-VALUE of the gene with the lowest p-value in the gen_tr_pot DF
gen_tr_pot[which.min(gen_tr_pot$pvals), 11]
```

```
## [1] 0.00075
```

```
 # 0.00075 #
```

```
# Reducing both dataframes
gene_reduced <- gene_expressed[, c(2, 3, 8, 9)]
transcript_reduced <- transcript_expressed[, c(1, 2, 8, 9)]

# Suffix attaches a defined string to same-name columns in combined dataframes to tell them apart
gen_tr <- merge(gene_reduced, transcript_reduced, by = "gene_id",
                              suffixes = c(".GENE", ".TRANSCRIPT"))

dim(gen_tr)    # 98 rows, 7 columns
```

```
## [1] 98  7
```

The dimension of the new dataframe was 98 rows x 7 columns

Gene id is uc001aeo.3. It is named CPTP in the RefSeq track, while in the UCSC genes track it's GLTPD1. The full name of the gene is ceramide-1-phosphate transfer protein. From Uniprot (http://www.uniprot.org/uniprot/Q5TA50): mediates the intracellular transfer ofceramide-1-phosphate between organelle membranes and the cell membrane. Required for normal structure of the Golgi stacks. Can bind phosphoceramides with a variety of aliphatic chains, but has a preference for lipids with saturated C16:0 or monounsaturated C18:1 aliphatic chains, and is inefficient with phosphoceramides containing lignoceryl (C24:0). Plays a role in the regulation of the cellular levels of ceramide-1-phosphate, and thereby contributes to the regulation of phospholipase PLA2G4A activity and the release of arachidonic acid. Has no activity with galactosylceramide, lactosylceramide, sphingomyelin, phosphatidylcholine, phosphatidic acid and ceramide.

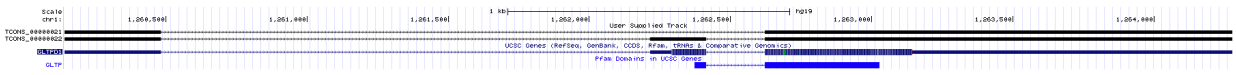


Figure 1: Capture of the UCSC browser at the GLTPD1 gene region. The tracks display the genomic position, a custom gtf file of transcripts, showing the two isoforms of the gene (TCONS_0000021 and TCONS_0000022), the UCSC genes and the PFam domains.

As can be seen in figure 1, this gene lies in the forward strand and comprises 3 exons. 2 of them are homologous to the GLTP PFam domain. This is interesting since one of the isoforms (TCONS_0000021) lacks exon 2, therefore missing the N-terminus residues of this domain. TCONS_0000022 features all 3 exons.

The structure of this protein is available at the PDB (under entry 4K84). It displays an all alpha fold, where most of the residues lie within an alpha helix. In figure 2 one can see that this second exon is spliced in TCONS_0000021, consists of almost two full helices. Only the C-terminus residues of the second helix are encoded in the third exon. This means that the protein product originating from the translation of TCONS_0000021 would lack the Nα helix as well as most of the next helix.

The CPTP domain consists of a double layer of alpha helices generating a hydrophobic pocket that allows for the transfer of lipids across membranes. This pocket is sealed by two Leu residues within the Nα helix (Simanshu **et al**). According to this, this hydrophobic pocket would be severely affected in protein product encoded by TCONS_0000021.

Since TCONS_0000021 expression decreases one order of magnitude when transitioning to condition 2, this analysis provides evidence for an inhibition of the expression of its truncated protein in condition 2. This

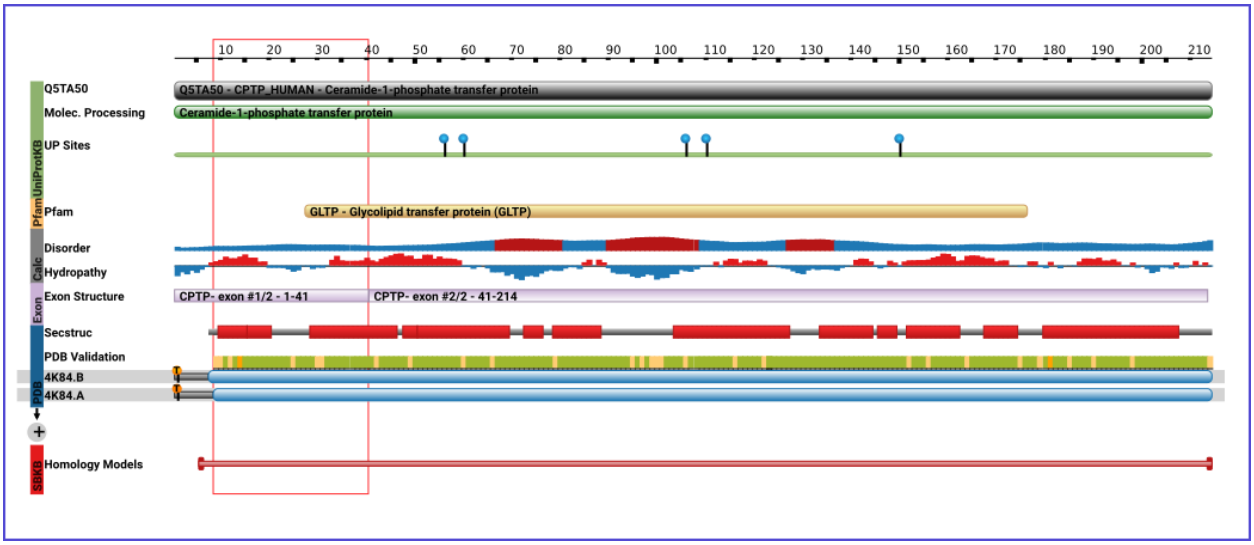


Figure 2: Organization of the GLTP domain in the GLTPD1 structure at the PDB under key 4K84. Several helices are distributed across the domain encompassing 2 exons. The two helices lying in the N-terminus of the domain are almost fully encoded by exon 2, whereas the rest of the domain lies in exon 3.

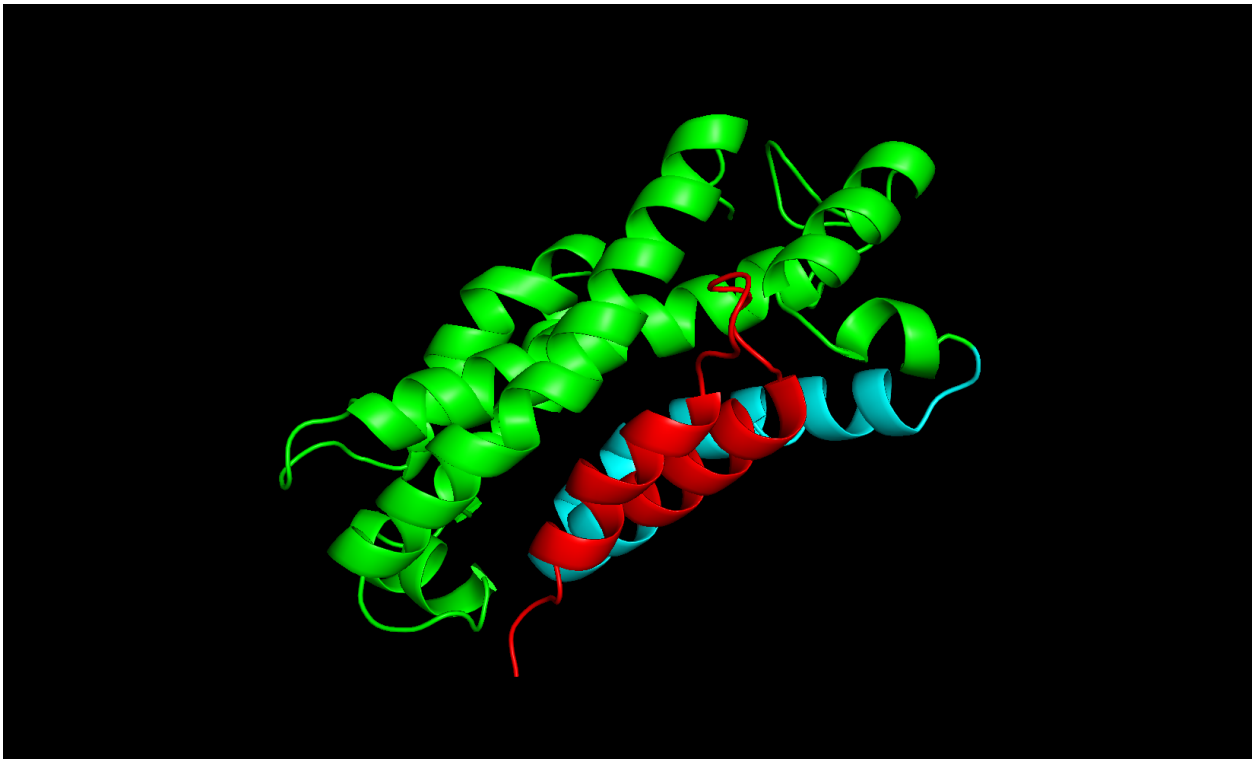


Figure 3: Pocket generated by the alpha helices in the GLTP domain. Green helices shape one side of the pocket. The blue and red helices enclose the pocket on the other side. The red helices are coded in exon 2 and are lost in the protein encoded by the TCONS_0000021 transcript.

is quite interesting since TCONS_0000022 does not change its expression that much. This means that TCONS_0000021 regulation's does not act at the promoter level but at another downstream process. Even if TCONS_0000022 expression does not change that much, this could either mean that the full protein remains produced in a similar amount, or that there other inhibitory mechanisms acting postranscriptionally.

Transcript	Expression 1	Expression 2	Pocket
TCONS_0000021	3299.4300	345.5850	No
TCONS_0000022	4623.5900	4335.4500	Yes

Table 1: Summary table

In conclusion, this analysis on has revealed that the GLTPD1 gene, producing to isoforms experiences an isoform switch. TCONS_0000021 expression decreases by almost 90%, while TCONS_0000022 remains similar. The structural difference of the proteins encoded by two isoforms, where the first isoforms codes for a protein with a truncated domain, provides an interesting insight in to the biology of lipid transfer across membranes and the function of the GLTPD1 gene.

References

Simanshu, D. K., Kamlekar, R. K., Wijesinghe, D. S., Zou, X., Zhai, X., Mishra, S. K., ... Patel, D. J. (2013). Non-vesicular trafficking by a ceramide-1-phosphate transfer protein regulates eicosanoids. Nature, 500(7463), 463–467. http://doi.org/10.1038/nature12332

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.2 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=es_ES.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=es_ES.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=es_ES.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] gridExtra_2.2.1 dplyr_0.5.0     reshape2_1.4.2  cowplot_0.7.0
## [5] ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.11     knitr_1.16       magrittr_1.5     munsell_0.4.3
##  [5] colorspace_1.3-2 R6_2.2.1         rlang_0.1.1      stringr_1.2.0
##  [9] plyr_1.8.4       tools_3.4.0      grid_3.4.0       gtable_0.2.0
## [13] DBI_0.6-1        htmltools_0.3.6  assertthat_0.2.0 yaml_2.1.14
```

```
## [17] lazyeval_0.2.0   rprojroot_1.2    digest_0.6.12    tibble_1.3.1
## [21] evaluate_0.10    rmarkdown_1.5    stringi_1.1.5    compiler_3.4.0
## [25] scales_0.4.1     backports_1.1.0
```