Assignment 2 group 6

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

May 15, 2017

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

Part 1

(a) What are the first five genomic nucleotides from the first exon of this transcript?

AAAGG

The DICER1 mRNA molecule should have the same sequence as the DNA genomic sequences in the sense strand (substituting T by U). As AK002007 is transcribed on the reverse strand and the default genomic sequence presented by the browser is the antisense, we have to reverse it. Therefore, the first five nucleotides of the exon in the AK002007 cDNA are AAAGG.

(b) Look at the raw mRNA sequence of AK002007, from the database it actually comes from. What are the first five nucleotides?

When we check the raw mRNA sequence of AK002007, it can be seen that the first five nucleotides are GAAGC.

(c) How do you explain the discrepancy (maximum 5 lines)?

AK002007 is possibly a truncated version of the DICER1 mRNA found in Genbank and refseq. The first 11 nts of AK002007 are found in the last 11 nt of the previous exons in the other mRNA sequences. The first 4 nts of these are predicted to be part of the 5' UTR of AK002007 by the aligner, confirming a misalignment. Otherwise, those 11 nt would be aligned as a separate exon in alignment with the other mRNA sequences.

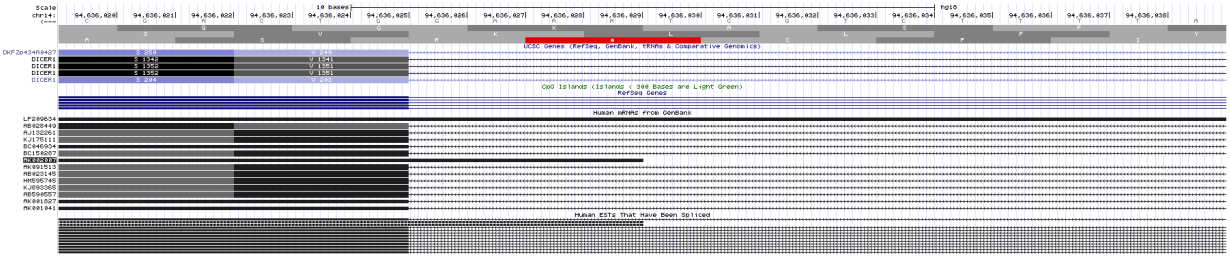


Figure 1: *Screenshot of the start of AK002007. The first 7 letters are discarded by the aligner, while the following 4 are aligned to the end of the intron.*

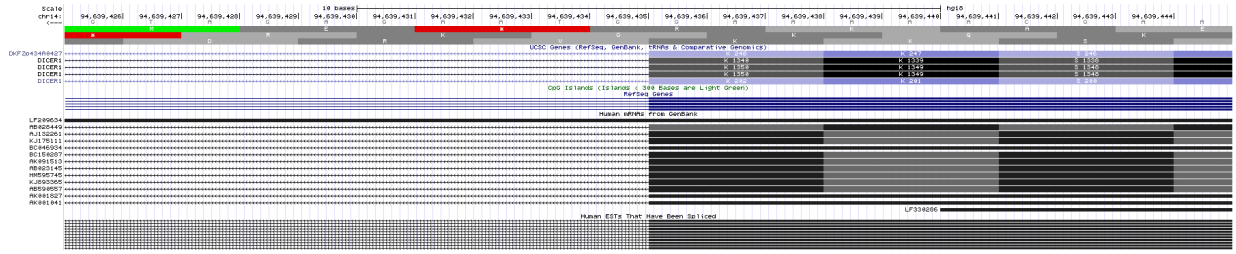


Figure 2: *Screenshoot of the upstream exon. As we can see, the last letters of the exon match the letters aligned to the intron.*

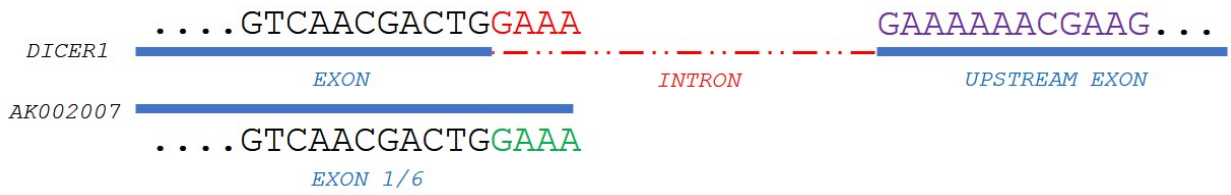


Figure 3: *Shows the mRNA sequence of AK002007 and the general DICER1. These mRNA sequences are read from the right to the left, because they are transcribed from the minus strand. Remarkably, the final 4 letters of the upstream exon and the intron are the exact same GAAA. The 11 nucleotides that match between the upstream exon and the raw mRNA sequence of AK002007 are written in pruple.*

Part 2

(a)

Bedtools commands to preprocess data before R

```
cat ERa_hg18.bed ERb_hg18.bed > full.bed
bedtools sort -i full.bed > full.sorted.bed
bedtools merge -i full.sorted.bed > merged.bed
# Calculates the percent coverage by chromosome
bedtools genomecov -i merged.bed -g hg18_chrom_sizes.txt > coverage.txt
bedtools genomecov -i ERa_hg18.bed -g hg18_chrom_sizes.txt > ERa.txt
bedtools genomecov -i ERb_hg18.bed -g hg18_chrom_sizes.txt > ERb.txt
```

R

```
df <- rbind(cbind(read.table(file = "coverage.txt"), protein = "Merged"),
            cbind(read.table(file = "ERa.txt"), protein = "ERa"),
            cbind(read.table(file = "ERb.txt"), protein = "ERb"))
colnames(df) <- c("chr", "bit", "start", "end", "fraction", "protein")
df$chr <- factor(as.character(df$chr),
                 levels = c(paste("chr", c(as.character(1:22), "X", "Y", "M"), sep = ""), "genome"))
sorted.index <- order(as.numeric(substring(df$chr, 4)))
df <- df[sorted.index,]
ggplot(data = df[(df$bit == 1 & df$chr != "genome"),],
       mapping = aes(x = chr, y = fraction, fill = protein)) +
  geom_bar(stat = "identity", position = "dodge") +
  scale_y_continuous(limits = c(0, 6e-4)) +
   theme(legend.title = element_blank(),
         legend.position = c(0.75, 0.87),
```

```
        legend.key.size = unit(0.5, "cm"),
        legend.direction="horizontal")
```

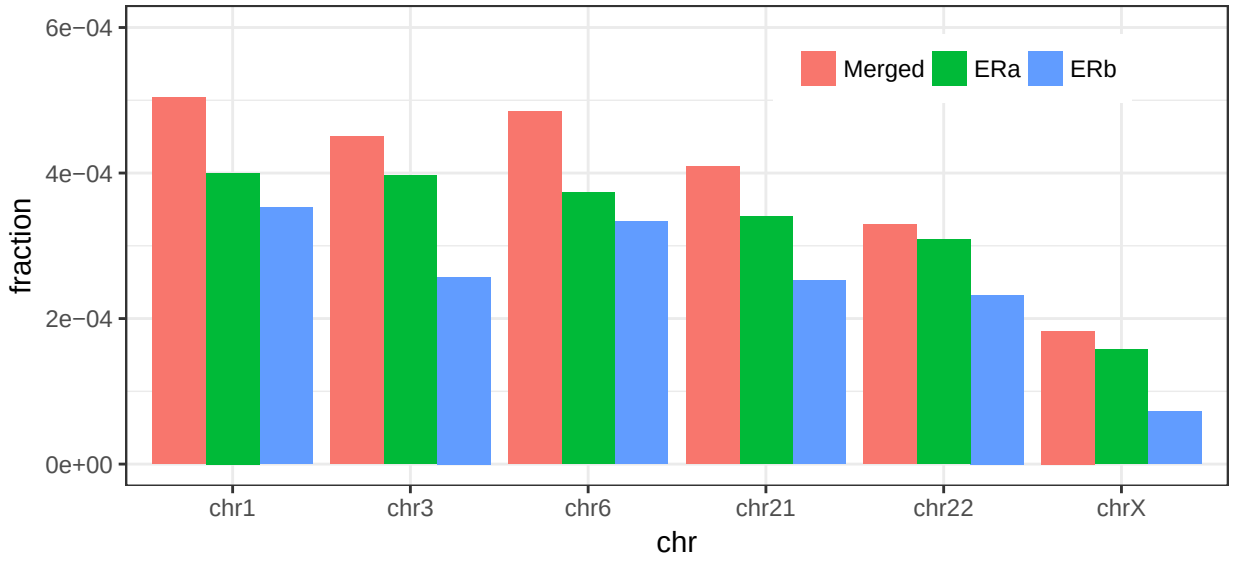


Figure 4. *Barplot of the coverage across the genome for ERa, Erb and the intervals resulting for a total merge of both.*

We also considered running a Wilcoxon test on the distribution of the fractions to determine whether there is a significant differences between the coverages of ERa and ERb, given an apparent decline in coverage in ERb in relation to ERa, but there was not ($p > 0.05$).

```
wilcox.test(df[df$bit == 1 & df$chr != "genome" & df$protein== "ERa", 5], df[df$bit == 1 & df$chr != "g
```

```
##
##  Wilcoxon rank sum test
##
## data:  df[df$bit == 1 & df$chr != "genome" & df$protein == "ERa", 5] and df[df$bit == 1 & df$chr != "
## W = 28, p-value = 0.132
## alternative hypothesis: true location shift is not equal to 0
```

As shown in the plot, the ERa and ERb ChIP sites are overlapping in the same chromosomes, these being 1, 3, 6, 21, 22, and X. In addition, after merging both interval sets, the binding fraction is more or less the same for both receptors (Figure 4). A biological explanation for observing that many overlapping ChIP sites for ERa and ERb could be that these receptors not only form homodimers, but also heterodimers with each other (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3389841/). Another possible explanation could be that ERa and ERb have similar or even contradicting actions on some of the same genes. A more basic explanation for getting such high overlap could be that in the laboratory process, the antibody used for detecting ERa and ERb in the ChIP analysis was not specific for either of them. Notably, ERa and ERb only bind to 6 out of 22 chromosomes. It seems highly unlikely that these receptors only bind in these 6 chromosomes, so this data set may had been filtered down to these 6 chromosome, or we may be only seeing these truncated results due to a technicality of the tilling assay.

(b)

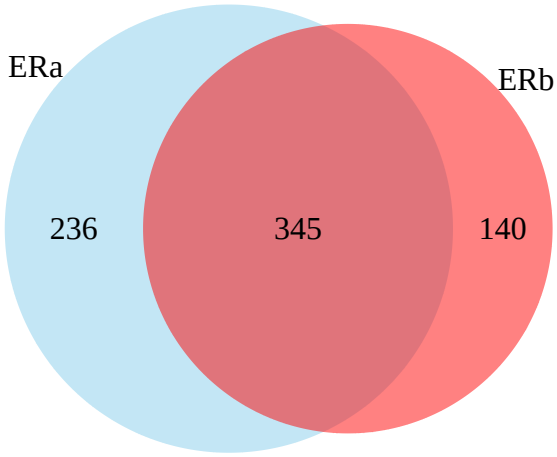
Bedtools commands to preprocess data before R

```
# Calculates number of interval overlaps between ERa and ERb
# Reports multiple overlaps as a single occurrence
```

```
bedtools intersect -a ERa_hg18.bed -b ERb_hg18.bed -c > AtoBoverlap.bed
bedtools intersect -a ERb_hg18.bed -b ERa_hg18.bed -c > BtoAoverlap.bed
```

R

```
df2 <- read.table(file ="AtoBoverlap.bed")
df3 <- read.table(file ="BtoAoverlap.bed")
overlap <- sum(df3$V4)
draw.pairwise.venn(nrow(df2), nrow(df3),
                   cross.area = overlap,
                   category = c("ERa", "ERb"),
                   lty = rep("blank", 2),
                   fill = c("skyblue", "red"),
                   alpha = rep(0.5, 2))
```




```
## (polygon[GRID.polygon.60], polygon[GRID.polygon.61], polygon[GRID.polygon.62], polygon[GRID.polygon.
```

Figure 5. *Venn diagram of the ERa and Erb interval sets, showing in the cross area how many of them overlap at least one base.*

As shown in Figure 5, there are 236 ERa ChIP intervals that do not overlap with ERb. On the other hand, there are 140 ERb ChIP sites that do not overlap with ERa Finally, ERa and ERb have 345 overlapping CHIP intervals

Part 3.

How can this be?

The fly mRNA sequence, which aligns to the mouse genome at chr9:24,851,809-24,851,889 (defined as "input region") looks different from the mRNA sequence that we get when we BLAT this to the fly genome because the gene in the fly genome includes the full fly's *rpl41* exome as well as introns (as seen in figure 6)

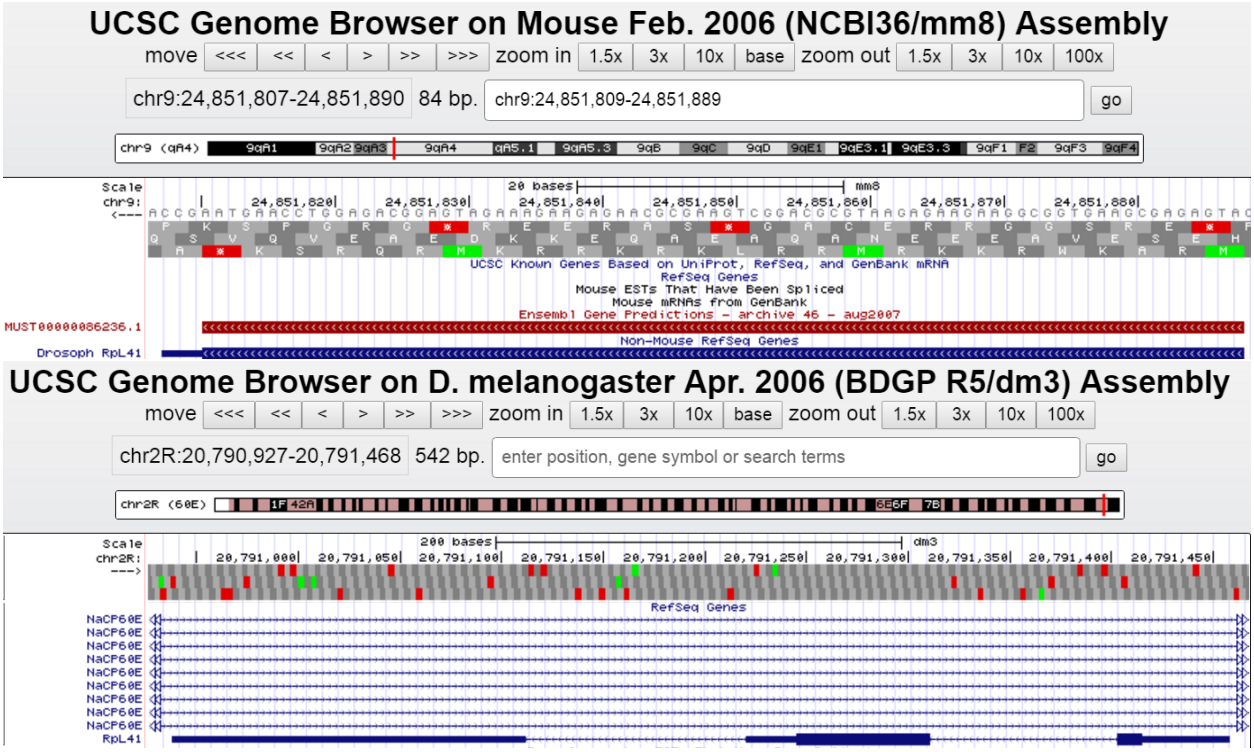


Figure 6. *Captures of UCSC browser of the mouse genome (upper) and fly genome (bottom). Upper figure shows the chr9:24,851,809-24,851,889 region in the mouse genome. Bottom figure shows the conserved fly mRNA sequence BLATed to the fly genome. This sequence corresponds to the rpl41 gene, encoding a ribosomal protein. The capture in the mouse genome shows that a 78 bp region in the fly genome aligns to the mouse genome at chr9. This corresponds to a subset of exons 2 and 3 in the fly rpl41 gene*

Is the mouse gene likely to be real?

We do not find it likely that the mouse input region is a real gene, based on the following three arguments:

- First of all, we find no support from the UCSC track or the Refseq track of any transcription of the input region, which makes us doubt that it represents a real gene.
- Secondly, if we compare the input region/conservation track to the functional Rpl41 gene in mouse (chr10:127,951,059-127 ,952,117), it can be seen that the functional gene includes introns and the input region/conservation track does not. Most eukaryotic genes include introns, so this indicates that the input region is not a real functional gene.
- Thirdly, we looked at the fly mRNA sequence in the conservation track from the input region and saw that there are 5 other exact copies of it in the mouse genome. These are 81.5 % identical to the NM_001014551 gene in the fly. We doubt that it is likely for a gene to have 5 identical and functional copies in a genome. Furthermore, our input region appears to be the complete last exon of the official rlp41 gene and part of the intronic region as well (FIGURE).It is more likely that these are remnants of an original gene that was once retrotransposed and then duplicated 5 times generating 5 copies in the genome (if they had been generated independently, they would show some variation between them, and they are 100% identical). If these copies are pseudogenes, one would often expect them to have acquired mutations, but it could be that the 5 copies were retrotransposed relatively recently and have not mutated enough.

Therefore, we conclude that the mouse input region is not likely to represent a real gene.

```
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] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] VennDiagram_1.6.17  futile.logger_1.4.3 gridExtra_2.2.1
## [4] dplyr_0.5.0         reshape2_1.4.2      cowplot_0.7.0
## [7] ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.10         knitr_1.16           magrittr_1.5
##  [4] munsell_0.4.3        colorspace_1.3-2     R6_2.2.1
##  [7] rlang_0.1.1          stringr_1.2.0        plyr_1.8.4
## [10] tools_3.4.0          gtable_0.2.0         DBI_0.6-1
## [13] lambda.r_1.1.9       htmltools_0.3.6      assertthat_0.2.0
## [16] yaml_2.1.14          lazyeval_0.2.0       rprojroot_1.2
## [19] digest_0.6.12        tibble_1.3.1         futile.options_1.0.0
## [22] evaluate_0.10        rmarkdown_1.5        labeling_0.3
## [25] stringi_1.1.5        compiler_3.4.0       scales_0.4.1
## [28] backports_1.0.5
```