Visualizing the alignment stats of STAR using R

This report was generated with R and the help of the `knitR` package.

The data set used here contains mRNA from 48 replicates of two *S. cerevisiae* populations: wildtype and *snf2* knock-out mutants. All 96 samples were sequenced on one flow cell (Illumina HiSeq 2000) The SRA accession number for the entire data set (consisting of 2 x 48 samples) is ERP004763.

```
library(knitr) # for turning the original script into a nicely formatted pdf
opts_chunk$set(echo = TRUE, message = FALSE) # knitr options
opts_knit$set(self.contained = FALSE) # knitr options
library(ggplot2)
library(gridExtra) # for composite plotting of ggplots
options(stringsAsFactors = FALSE)
```

Reading in data

```
# listing files to be read in
infiles <- list.files(path="01_STAR_logs/", # folder with log files
                      pattern="Log.final.out",
                      full.names = TRUE)

# iterating over the file list generates a list of data frames
align.results <- lapply(infiles, function(x)
  read.table(x, sep="|", strip.white=TRUE, stringsAsFactor=FALSE,
             skip=3, fill = TRUE, header = FALSE))
typeof(align.results)
```

```
## [1] "list"
```

```
head(align.results[[1]])
```

```
##                                          V1       V2
## 1 Mapping speed, Million of reads per hour  1038.94
## 2                    Number of input reads 12121025
## 3                Average input read length       51
## 4                            UNIQUE READS:
## 5             Uniquely mapped reads number 10850859
## 6                  Uniquely mapped reads %   89.52%
```

```
# removing "%" from some of the values to keep just the numeric parts
align.results <- lapply(align.results, function(x)
  transform(x, V2 = as.numeric(gsub("%", "", x$V2) )))

# some cosmetics of each data frame's name, specific for the sample names
# of the files used here
names(align.results) <-  gsub(".*(SNF2|WT)(\\_[0-9]*).*", "\\1\\2", infiles)
```

Generating a long data frame for ggplot2

```
# concatenating all data frames of align.results
align.results.df <- as.data.frame(do.call(rbind, align.results))

# removing lines without values
align.results.df <- align.results.df[complete.cases(align.results.df),]

# adding additional columns with information about sample and replicate ID,
# using the information from the row names (which are, in turn, based on the
# names of the individual data frames that were stored in the original
# list, align.results)
align.results.df$sample <- gsub("(.*)\\_.*", "\\1", row.names(align.results.df))
align.results.df$replicate <- as.factor(as.numeric(
  gsub(".*\\_([0-9]*)\\.[0-9]*", "\\1", row.names(align.results.df))
  ))

# check the result - we should have a data frame with 4 columns
head(align.results.df)
```

```
##                                                V1          V2 sample
## SNF2_1.1 Mapping speed, Million of reads per hour     1038.94   SNF2
## SNF2_1.2                    Number of input reads 12121025.00   SNF2
## SNF2_1.3                Average input read length       51.00   SNF2
## SNF2_1.5             Uniquely mapped reads number 10850859.00   SNF2
## SNF2_1.6                  Uniquely mapped reads %       89.52   SNF2
## SNF2_1.7                    Average mapped length       50.72   SNF2
##          replicate
## SNF2_1.1         1
## SNF2_1.2         1
## SNF2_1.3         1
## SNF2_1.5         1
## SNF2_1.6         1
## SNF2_1.7         1
```

Making plots

Numbers are usually more easily digestible and comparable if they are represented visually, especially if one has many samples (although you will probably rarely encounter 48 samples per replicate as for this example data used here). However, we don't need to visualize every entry from the `STAR` output since some are redundant, and some even not applicable in our case:

```
unique(align.results.df$V1)
```

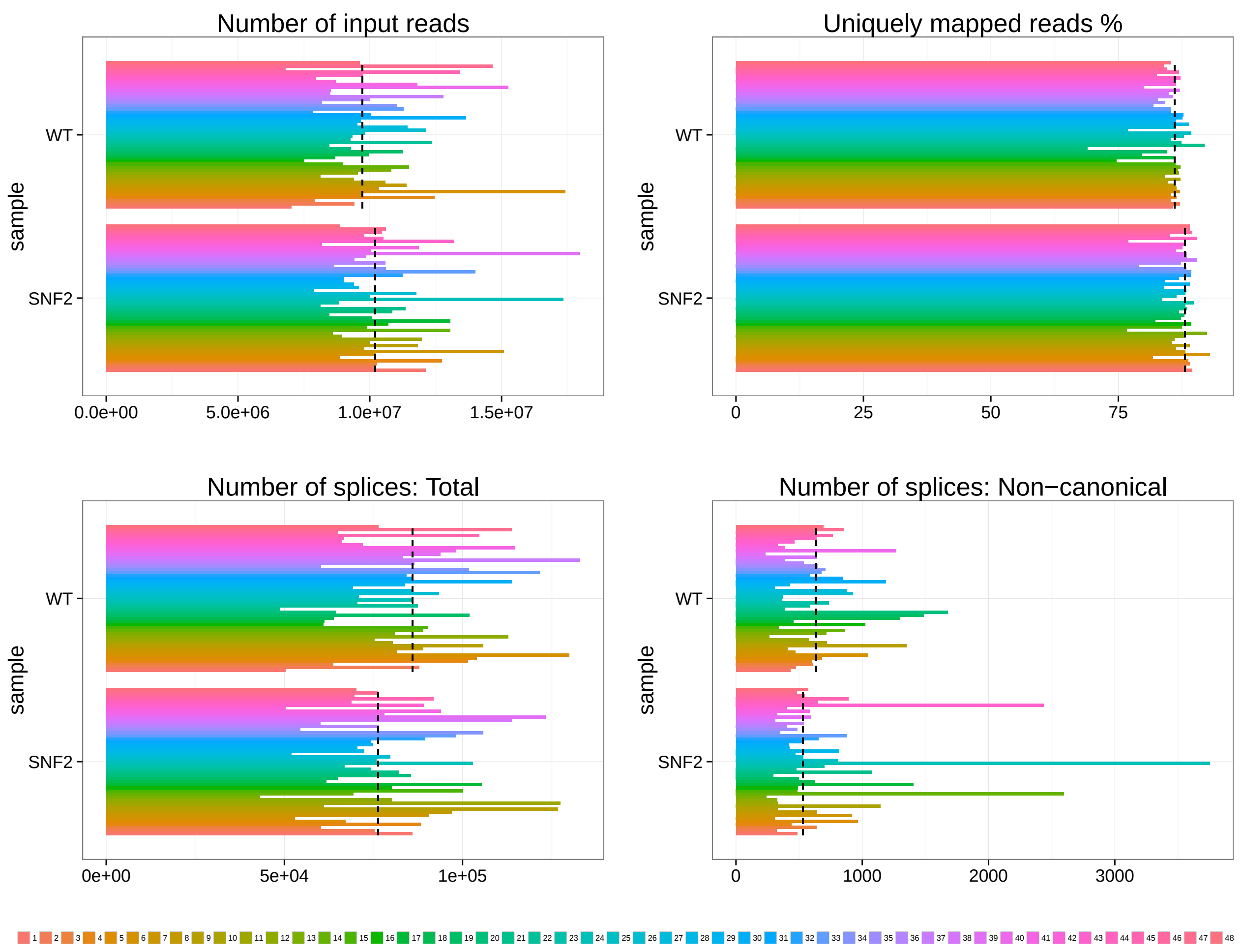
```
##  [1] "Mapping speed, Million of reads per hour"
##  [2] "Number of input reads"
##  [3] "Average input read length"
##  [4] "Uniquely mapped reads number"
##  [5] "Uniquely mapped reads %"
##  [6] "Average mapped length"
##  [7] "Number of splices: Total"
##  [8] "Number of splices: Annotated (sjdb)"
##  [9] "Number of splices: GT/AG"
## [10] "Number of splices: GC/AG"
## [11] "Number of splices: AT/AC"
## [12] "Number of splices: Non-canonical"
## [13] "Mismatch rate per base, %"
## [14] "Deletion rate per base"
## [15] "Deletion average length"
## [16] "Insertion rate per base"
## [17] "Insertion average length"
## [18] "Number of reads mapped to multiple loci"
## [19] "% of reads mapped to multiple loci"
## [20] "Number of reads mapped to too many loci"
## [21] "% of reads mapped to too many loci"
## [22] "% of reads unmapped: too many mismatches"
## [23] "% of reads unmapped: too short"
## [24] "% of reads unmapped: other"
```

Thus, we first define those entries that we are interested in:

```
filters = c("Number of input reads", "Uniquely mapped reads %",
            "Number of splices: Total", "Number of splices: Non-canonical")
```

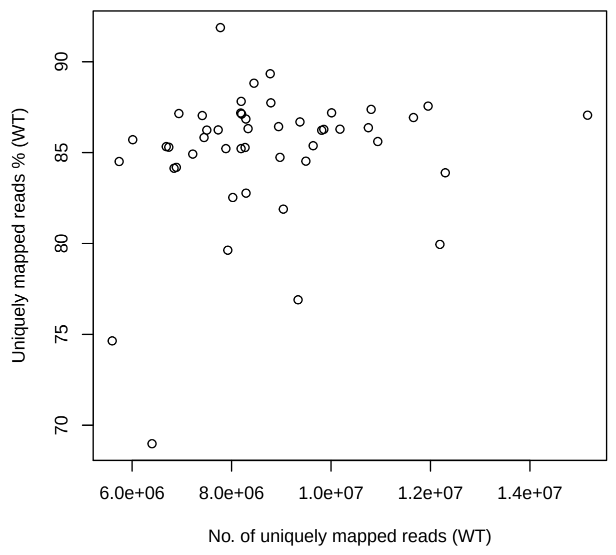
Now, let's generate some bar plots. The following chunk of code is optimized for returning a composite image with four plots and one legend. Note that the functions `PlottingAligmentResults` and `ExtractLegend` are part of the script that we sourced in the beginning.

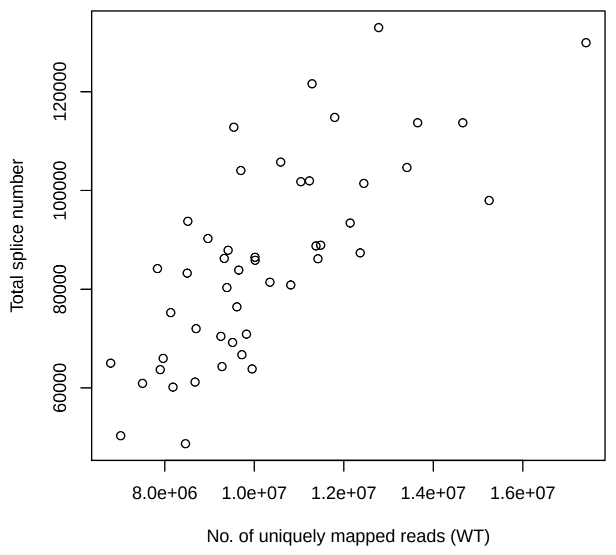
```
# for each entry in "filters", generate a bar chart
plots <- lapply(filters, function(x)
  PlottingAlignmentResults(x, align.results.df, Legend = FALSE))

# getting the legend is a bit tricky (kudos to Luce for figuring it out)
my.legend <- ExtractLegend(PlottingAlignmentResults(align.results.df,
                                             Filter = filters[1],
                                             Legend=TRUE))

# combining plots and legend
grid.arrange(arrangeGrob(plots[[1]], plots[[2]], plots[[3]], plots[[4]], nrow=2),
             my.legend$legend, nrow=2,
             heights= unit.c(unit(1, "npc") - my.legend$lheight, my.legend$lheight)
             )
```

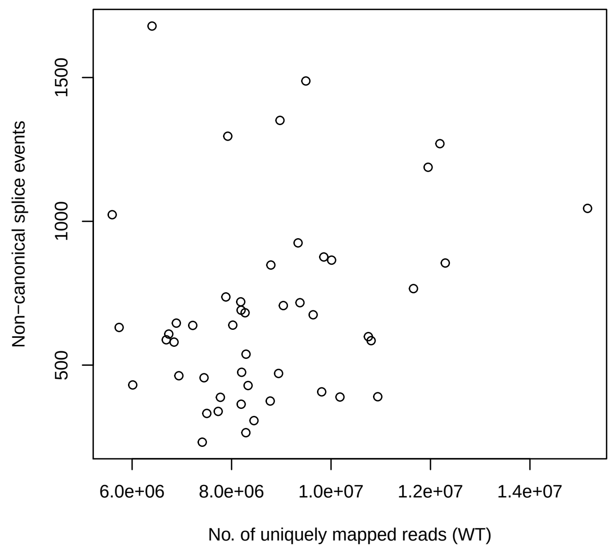


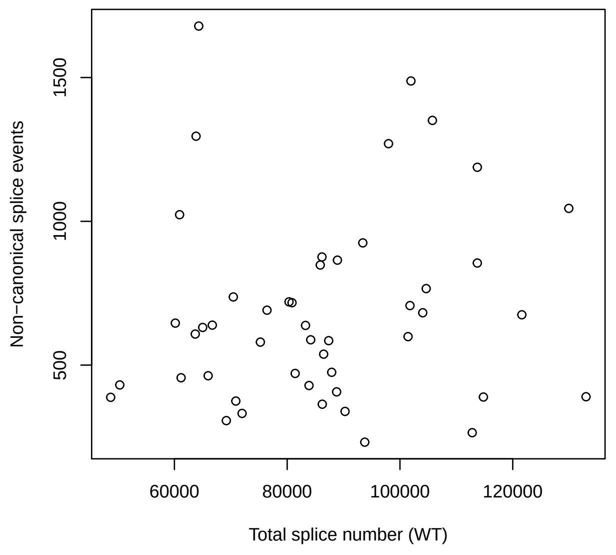
The dashed black line is the median value across all replicates of one condition. There is quite a bit of variation for all four stats between the replicates. Although the median sequencing depth of the SNF2 samples is slightly higher than for the WT samples, the median number of splice sites shows a reverse trend. This is counter-intuitive as the number of splice sites should correlate positively with the number of mapped reads. Let's check whether that relationship is simply confounded by just looking at the median values.

WT

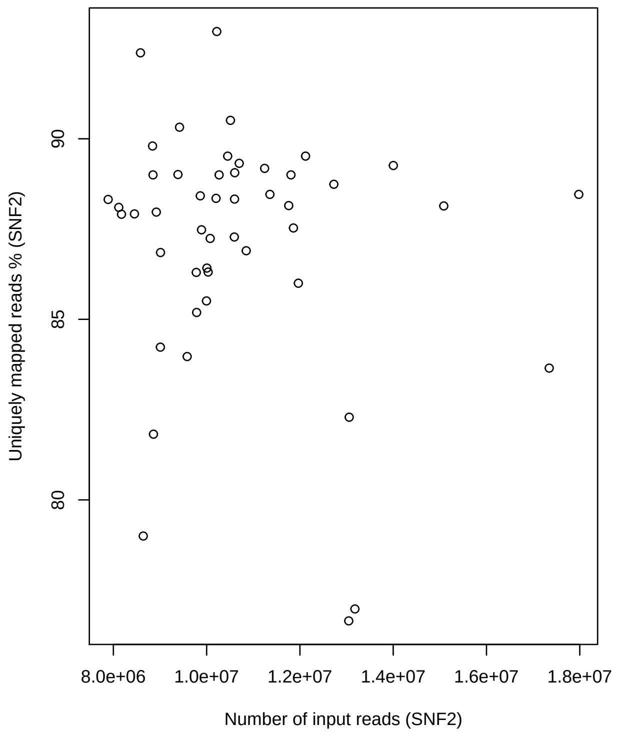

Uniquely mapped reads % (WT)
No. of uniquely mapped reads (WT)
70
75
80
85
90
6.0e+06
8.0e+06
1.0e+07
1.2e+07
1.4e+07

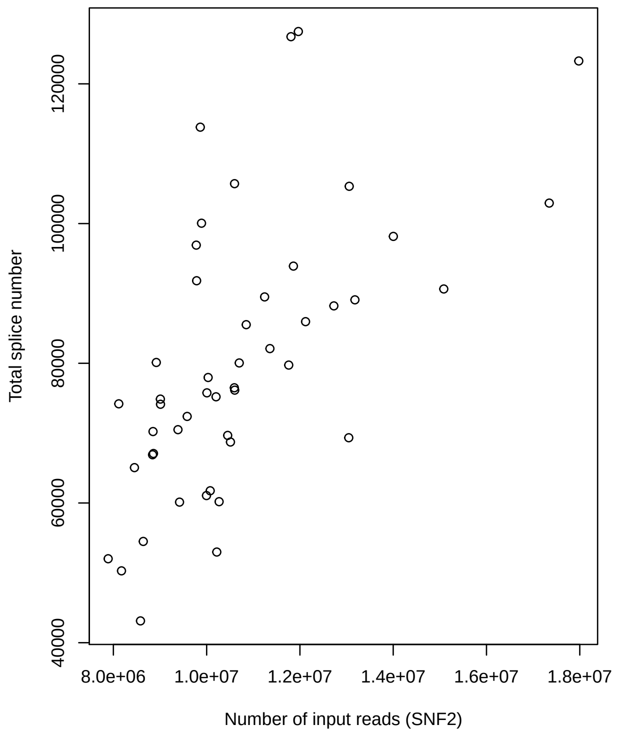

Total splice number
No. of uniquely mapped reads (WT)
60000
80000
100000
120000
8.0e+06
1.0e+07
1.2e+07
1.4e+07
1.6e+07

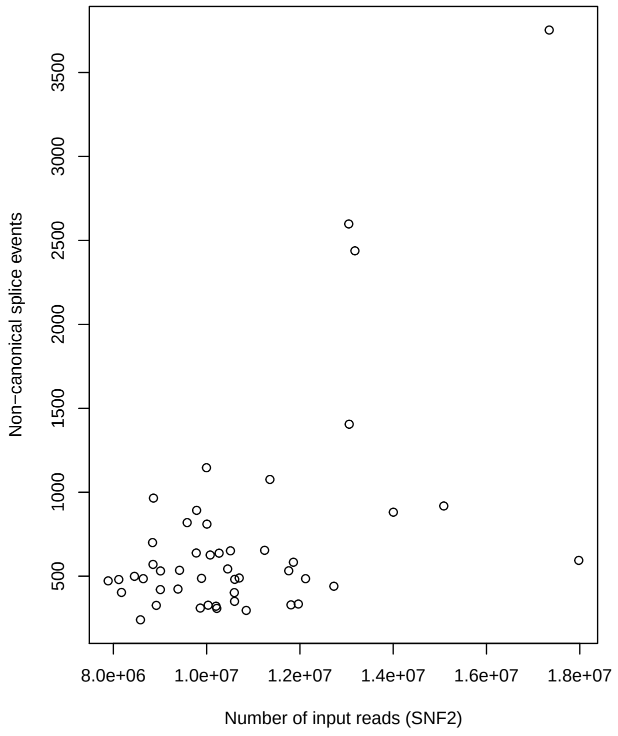

Non−canonical splice events
No. of uniquely mapped reads (WT)
500
1000
1500
6.0e+06
8.0e+06
1.0e+07
1.2e+07
1.4e+07

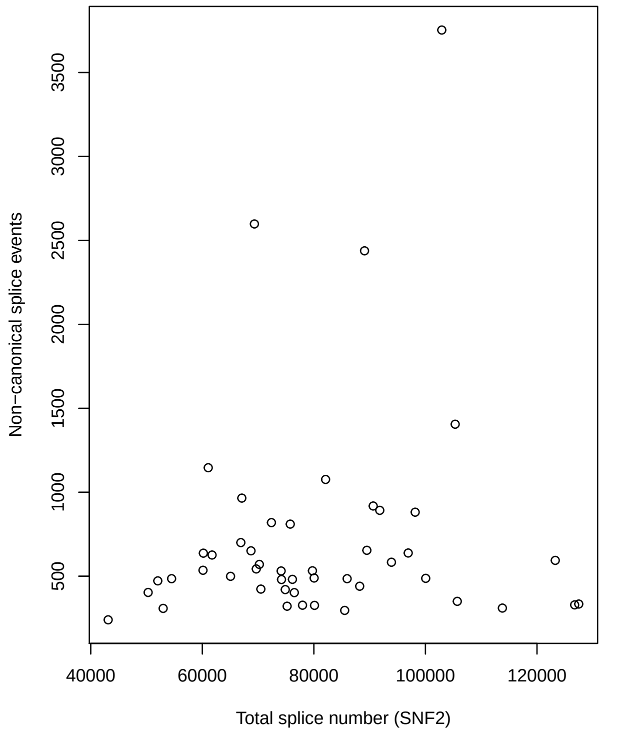

Non−canonical splice events
Total splice number (WT)
500
1000
1500
60000
80000
100000
120000

SNF2









In both sample types, the number of detected splice sites indeed seems to correlate with the number of input reads.

Apart from that, the samples almost all aligned well (70-80% alignment rate is typically seen for RNA-seq experiments).

For future references, let's get those samples that seem to deviate the most:

- samples with low mapping rate (<75%)

```
subset(align.results.df, V1 == filters[2] & V2 < 75)
```

```
##                               V1    V2 sample replicate
## WT_16.6 Uniquely mapped reads % 74.64     WT        16
## WT_20.6 Uniquely mapped reads % 68.98     WT        20
```

- samples with high numbers of non-canonical splice events (>1,000)

```
subset(align.results.df, V1 == filters[4] & V2 > 1000)
```

```
##                                         V1   V2 sample replicate
## SNF2_10.13 Number of splices: Non-canonical 1146   SNF2        10
## SNF2_14.13 Number of splices: Non-canonical 2598   SNF2        14
## SNF2_17.13 Number of splices: Non-canonical 1405   SNF2        17
## SNF2_21.13 Number of splices: Non-canonical 1076   SNF2        21
## SNF2_24.13 Number of splices: Non-canonical 3753   SNF2        24
## SNF2_43.13 Number of splices: Non-canonical 2438   SNF2        43
## WT_16.13   Number of splices: Non-canonical 1023     WT        16
## WT_18.13   Number of splices: Non-canonical 1296     WT        18
## WT_19.13   Number of splices: Non-canonical 1488     WT        19
## WT_20.13   Number of splices: Non-canonical 1679     WT        20
## WT_30.13   Number of splices: Non-canonical 1188     WT        30
## WT_40.13   Number of splices: Non-canonical 1270     WT        40
## WT_6.13    Number of splices: Non-canonical 1045     WT         6
## WT_9.13    Number of splices: Non-canonical 1351     WT         9
```