Alternative-based splicing analysis of *Apis mellifera* queens and drones - Part 2: Gonads

Joe Colgan

Contents

1. Introduction:

The purpose of this tutorial is to take output files generated by rRMATs and analysis using the R package maser, which filters and visualises differentially spliced genes. This script generates output files in the form of tables and plots in PNG format.

2. Install libraries:

For this tutorial, we need the following R packages installed:
- maser
- tidyverse
- ggpubr

```
## For maser:
#if (!require("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#BiocManager::install("maser")

## For tidyverse:
#install.packages("tidyverse")

## For ggpubr:
#install.packages("ggpubr")
```

3. Load libraries:

Once packages are installed, we can load the packages using the base R function 'library()'. By loading each package, we gain access to custom functions that have been written and compiled by other researchers. These functions are written in R and save us having to write functions ourselves.

```
library(maser)
library(tidyverse)
library(ggpubr)
```

4. Load data:

For the analysis using maser, we provide a path to our directory containing the output of our rRMATs-based analysis. Then using the 'maser()' function, we specify our comparison groups, as well as the output files from rRMATs, which we want to use in our analysis.

For the present analysis, we will load the output of rMATs, which uses counts that overlap splice junctions and exons.

First, we create an object called *path* where we assign the relative path to a directory that contains the output files of rMATs.

```
# path to folder:
path <- "data/gonad/"
```

Using this path, we next load files that contain 'JC' in their file name, and store this information in the object *samples.*

```
samples <- maser(path,
                 c("drone",
                   "queen"),
                 ftype = "JC")
```

Once successfully loaded, we can check the contents of our files.
To do this, we can use the base R function 'summary()' and specify which type of splicing event we want to examine. **Remember**: With rMATS, we gain information on five different splicing events, which using their respective abbreviated terms, we can use to specify and examine.

For the abbreviations:
- SE: Skipped Exons
- A5SS: Alternative 5' splice site
- A3SS: Alternative 3' splice site
- MXE: Mutually exclusive exons
- RI: Retained intron

Here, using the 'head()' function, we can visualise a subset of our maser-generated object. By default, 'head()' prints the first six lines while in our command below, the inclusion of '[, 1;8]' after the 'summary()' function means that we want to examine the first eight columns present in the maser-generated object.

```
head(summary(samples,
             type = "SE")[, 1:8])
```

```
##   ID       GeneID geneSymbol       PValue          FDR IncLevelDifference
## 1  0 LOC113219365         NA 1.000000e+00 1.000000e+00              0.010
## 2  1    LOC551434         NA 4.599084e-02 1.651595e-01              0.018
## 3  2    LOC551434         NA 1.000000e+00 1.000000e+00             -0.013
## 4  3    LOC409880         NA 1.000000e+00 1.000000e+00             -0.002
## 5  4 LOC100578661         NA 1.554312e-15 1.882457e-14              0.087
## 6  5 LOC100578661         NA 5.678820e-01 1.000000e+00             -0.080
##              PSI_1             PSI_2
## 1        1,1,1,1,1          0.97,1,1
## 2        1,1,1,1,1 0.995,0.961,0.991
## 3 0.95,0.977,1,1,1         1,1,0.995
## 4    1,0.991,1,1,1             1,1,1
## 5        1,1,1,1,1   0.83,0.98,0.928
## 6      0.6,1,1,1,1             1,1,1
```

Here, we can see information related to: - 'ID', which corresponds to the event ID
- 'GeneID', these are IDs taken from an General Transfer Format (GTF) file generated for *Apis mellifera*,

which was obtained from the Ensembl Metazoa database
- 'geneSymbol', coded as the same as 'GeneID'
- 'Pvalue', raw p-value calculated by rMATS
- 'FDR', false discovery rate, a p-value corrected for multiple testing
- 'IncLevelDifference',
- 'PSI_1',
- 'PS2_2'

4. Filter data:

rMATs using read counts to help identify and quantify differences in splicing events within and between treatment groups. Using the function 'filterByCoverage()', we can specify the number of reads, on average, that we wish to use to consider a splice event as being true. We can do this by using the argument 'avg_reads' within the 'filterByCoverage()' function. In our example below, we will use the value 5 but if wanted to be more conservation, we could increase this value.

```
## However, we can also test:
samples_counts <- as.data.frame(as.matrix(counts(samples)))

## Calculate mean rows:
samples_counts$mean_rows <-  rowMeans(samples_counts)

## We can calculate the mean and median using summary():
summary(samples_counts$mean_rows)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     0.12    19.31    76.69   262.21   244.09 34499.62
```

A better approach would be to use the base R function 'quantile()', where we provide a vector of values, such as the entries found in a single column of a dataframe. With the quantile() function, we can ask about specific cut-offs in our distribution. For example, say we wanted to know the number of reads supporting 90% of calls, we can set a percentile threshold of 0.1 (i.e., 10%).

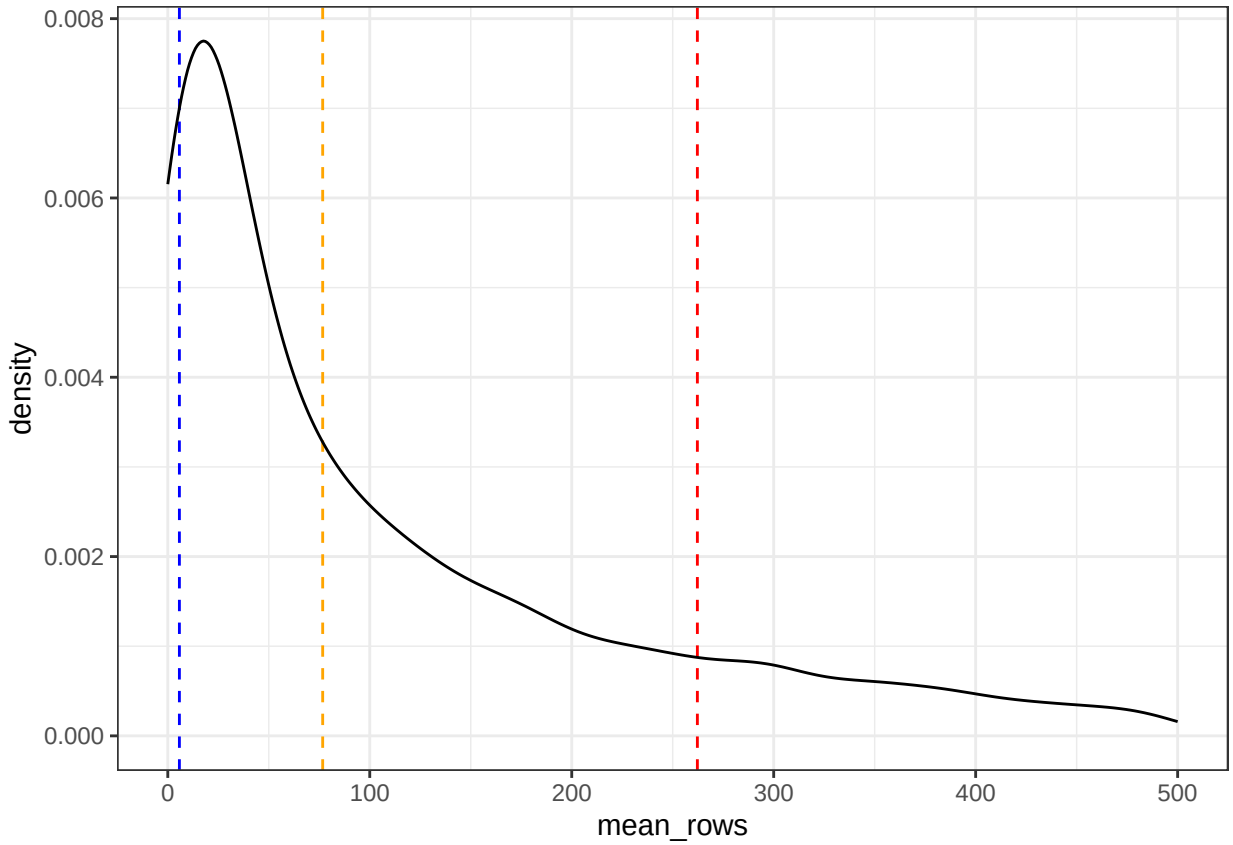
```
quantile(samples_counts$mean_rows,
         0.1)
```

```
##  10%
## 5.75
```

We can plot the distribution:

```
ggplot(data = samples_counts,
       aes(x = mean_rows)) +
  xlim(0, 500) +
  geom_vline(xintercept = mean(samples_counts$mean_rows),
             linetype = 'dashed',
             colour = 'red') +
  geom_vline(xintercept = median(samples_counts$mean_rows),
             linetype = 'dashed',
             colour = 'orange') +
  geom_vline(xintercept = quantile(samples_counts$mean_rows,
                                   0.1),
             linetype = 'dashed',
             colour = 'blue') +
  geom_density() +
  theme_bw()
```

```
## Warning: Removed 293 rows containing non-finite values (`stat_density()`).
```



Task: What would we a good threshold to filter?

```
samples_filt <- filterByCoverage(samples,
                                 avg_reads = 5)

## We can also extract counts and examine these:
samples_counts <-  counts(samples)

nrow(summary(samples_filt,
             type = "SE")[, 1:8])
```

```
## [1] 15934
```

```
head(summary(samples_filt,
             type = "SE")[, 1:8])
```

```
##   ID       GeneID geneSymbol       PValue          FDR IncLevelDifference
## 1  0 LOC113219365         NA 1.000000e+00 1.000000e+00              0.010
## 2  1    LOC551434         NA 4.599084e-02 1.651595e-01              0.018
## 3  2    LOC551434         NA 1.000000e+00 1.000000e+00             -0.013
## 4  3    LOC409880         NA 1.000000e+00 1.000000e+00             -0.002
## 5  4 LOC100578661         NA 1.554312e-15 1.882457e-14              0.087
## 6  5 LOC100578661         NA 5.678820e-01 1.000000e+00             -0.080
##              PSI_1             PSI_2
## 1        1,1,1,1,1          0.97,1,1
## 2        1,1,1,1,1 0.995,0.961,0.991
## 3 0.95,0.977,1,1,1         1,1,0.995
## 4    1,0.991,1,1,1             1,1,1
```

```
## 5        1,1,1,1,1   0.83,0.98,0.928
## 6      0.6,1,1,1,1             1,1,1
```

Task: What many skipped exon (SE) events are kept?

Task: Repeat this task but examine how many other spliced events are retained for the other types:
- A5SS:
- A3SS:
- RI:
- MXE:

Task: Which event is most common?

5. Filter events by significance:

The next step allows for the identification of significantly spliced events.
Using the function 'topEvents()', we can filter based on significance with the argument fdr = 0.05, meaning an event has a false discovery rate less than 0.05. We can also filter based on deltaPSI.

By running 'topEvents()' on our already filtered object, we store our more filtered dataset in a new object called samples_top.

```
samples_top <- topEvents(samples_filt,
                         fdr = 0.05,
                         deltaPSI = 0.1)
```

If we want to count how many events, have been retained, we can use the base R function 'nrow()', which counts the *number* of *rows* present in a dataset. First, to give the maser object to 'nrow()' in a format it recognises, we run our object through the function 'summary()', first.

```
nrow(summary(samples_top))
```

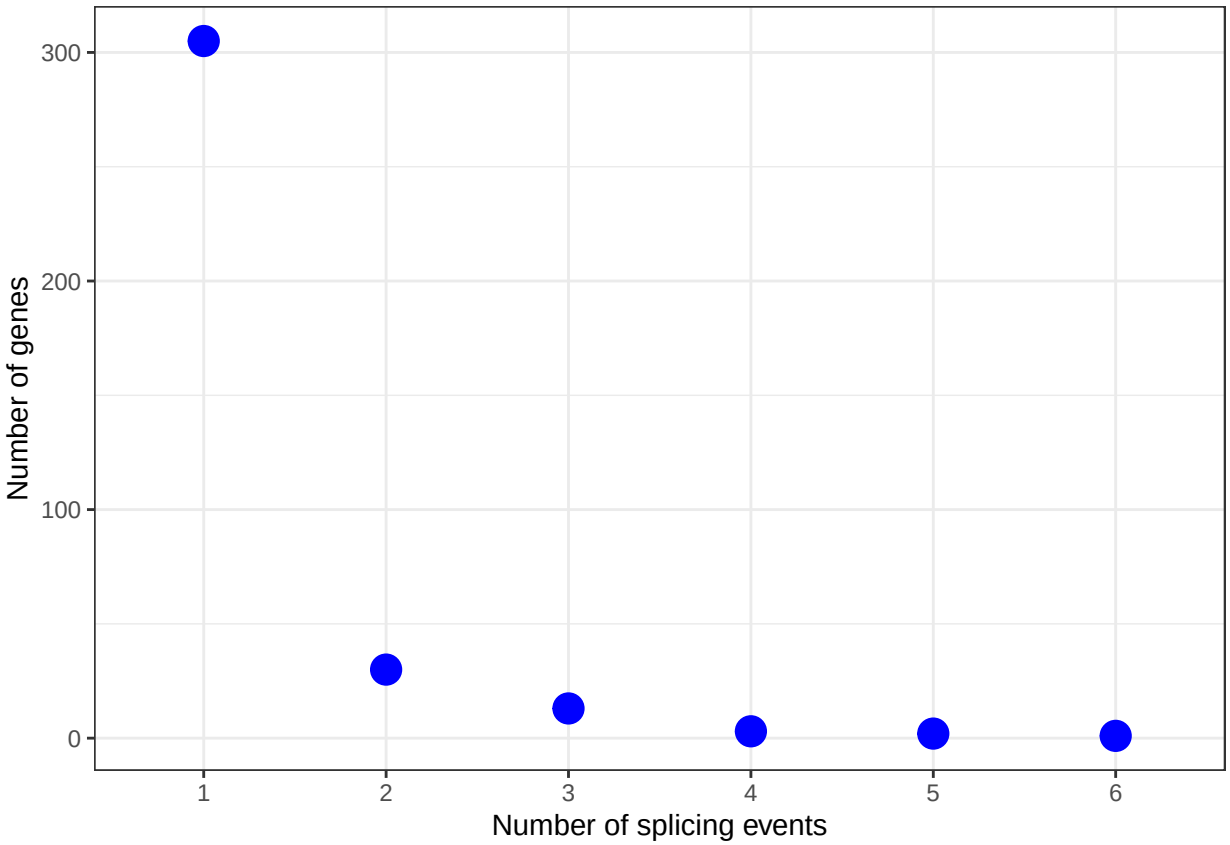
```
## [1] 432
```

Task: How many splicing events are kept after filtering?

We can also ask how many genes have at least one significantly spliced event.

```
splice_events_df <- as.data.frame(table(table(summary(samples_top)$GeneID)))
colnames(splice_events_df) <- c("Number_of_events",
                                "Count")

## Plot:
ggplot(data = splice_events_df,
       aes(x = Number_of_events,
           y = Count)) +
  geom_point(size = 5,
             colour = "blue") +
  xlab(label = "Number of splicing events") +
  ylab(label = "Number of genes") +
  theme_bw()
```



We can save this information, which we will use later in another script:

First, we convert the maser-based object into a dataframe.

```
samples_top_df <- as.data.frame(summary(samples_top))
```

If we want to save this object, we can first create a output directory using the base R function 'dir.create()' and with the argument 'path', we can set the name of the directory (e.g., a directory named 'results'), we want to create.

```
dir.create(path = "results")
```

Then we use a base R function 'saveRDS()', which allows us to save an object, such as our dataframe, and save it. This saved file can be used later in another script.

```
saveRDS(object = samples_top_df,
        file = "results/splice_events_gonad_jc.rds")
```

6. Plotting of significant events:

The *maser* package also provides custom functions for the visualisation of differences in splicing between treatments.

Using the function 'geneEvents()', we can specify a gene to subset by using the argument 'geneS'. Here, we will subset the gene 'LOC412608'.

As above, we can specify that for other arguments, we only keep events that reach certain criteria (e.g., fdr = 0.05 and deltaPSI = 0.1).

We run this function and store the output in a new object (samples_LOC412608).

```
head(samples_top_df)
```

```
##   ID    GeneID geneSymbol       PValue          FDR IncLevelDifference
## 1 12 LOC412608         NA 8.674262e-04 3.424051e-03              0.249
## 2 31 LOC411646         NA 6.830674e-04 2.755230e-03             -0.125
## 3 36 LOC412796         NA 0.000000e+00 0.000000e+00             -0.261
## 4 37 LOC412796         NA 2.776311e-05 1.402768e-04             -0.230
## 5 38 LOC412796         NA 8.968626e-11 9.482248e-10              0.169
## 6 43 LOC409713         NA 0.000000e+00 0.000000e+00             -0.187
##                            PSI_1             PSI_2            Chr Strand
## 1         1,0.914,0.658,0.539,1 0.616,0.543,0.561 chrNC_037653.1      -
## 2 0.565,0.705,0.657,0.659,0.685 0.785,0.794,0.758 chrNC_037652.1      +
## 3  0.57,0.547,0.643,0.755,0.745 0.869,0.948,0.922 chrNC_037652.1      -
## 4 0.495,0.375,0.542,0.569,0.667 0.639,0.874,0.765 chrNC_037652.1      -
## 5  0.42,0.323,0.395,0.301,0.414 0.182,0.254,0.169 chrNC_037652.1      -
## 6 0.519,0.614,0.684,0.659,0.647 0.832,0.821,0.782 chrNC_037652.1      -
##         exon_long      exon_short   exon_flanking
## 1 4859093-4859222 4859093-4859216 4859322-4859383
## 2 9395384-9395613 9395393-9395613 9394321-9394656
## 3 8208332-8208405 8208332-8208381 8208784-8208871
## 4 8208332-8208513 8208332-8208381 8208784-8208871
## 5 8208332-8208513 8208332-8208405 8208784-8208871
## 6 7855701-7855844 7855701-7855829 7856940-7857007
```

```
#samples_gene <- geneEvents(samples_top,
#                           geneS = "LOC724417",
#                           fdr = 0.05,
#                           deltaPSI = 0.1)
```

We can then check the number of splicing events found in this gene using the base R function 'print()':

```
#print(samples_gene)
```

Using the maser function 'plotGenePSI()', we can specify the type of splicing event we wish to plot, as well as whether to show replicates (i.e., an individual data point for each sample).

```
#plotGenePSI(samples_LOC412608,
#            type = "A3SS",
#            show_replicates = FALSE)
```

Here, we get a violin plot with our 'condition' ('drone' or 'queen') on the x-axis and PSI (percent spliced-in) on the y-axis for the specific splicing event being visualised. Each individual sample in our analysis is presented on this plot.

If we change the argument 'show_replicates' to FALSE, then we end up getting the average value per group.

```
#plotGenePSI(samples_LOC412608,
#            type = "A3SS",
#            show_replicates = FALSE)
```

We can slso investigate global differences across the entire transcriptome using volcano plots, which can also be generated using a custom function from maser.

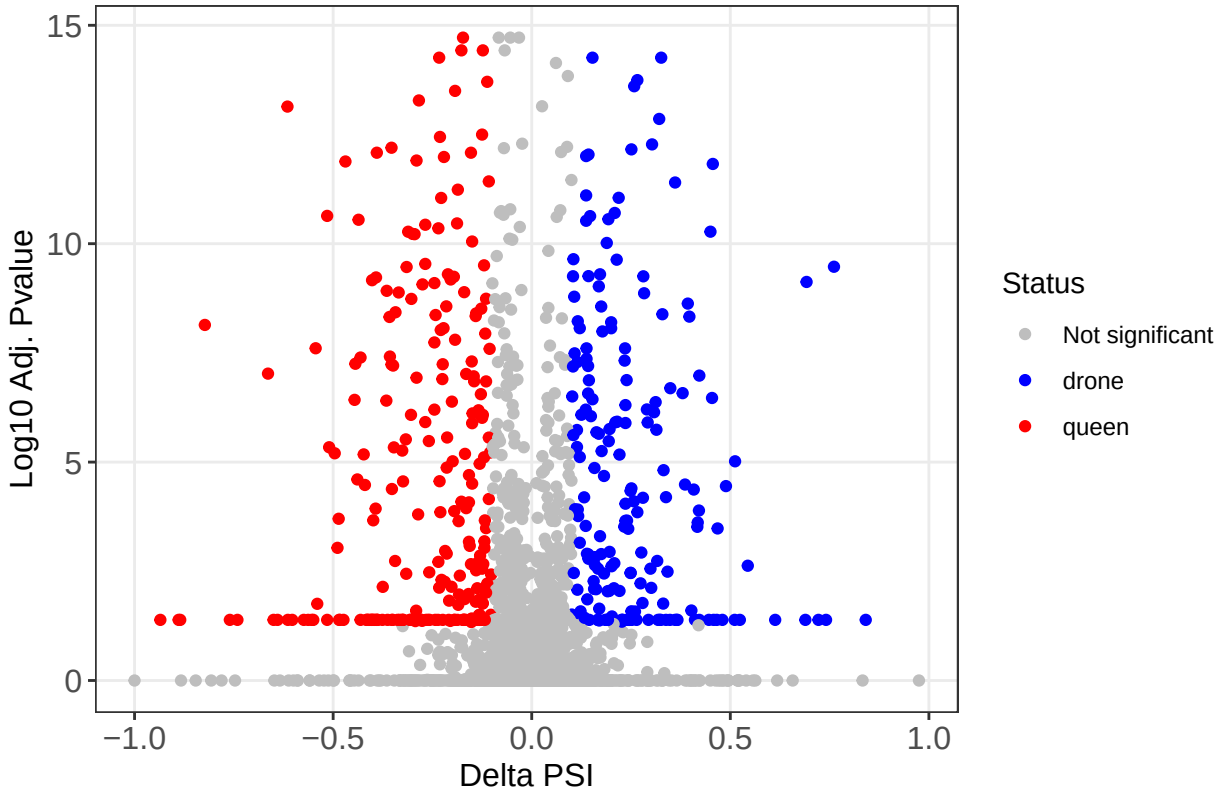
A volcano plot is a modified scatter plot where we plot deltaPSI on the x-axis and log10 transformed adjusted p value on the y-axis. FDR is a form of adjusted p value but we log10 transform the data so it is on a scale that is easier to plot and visualise patterns. If we did not, the scale of values would go from 0 to 1 and as many adjusted p values for significantly spliced events may be very small (e.g., 0.000000001), it could make

them difficult to visualise on a plot. Therefore, we log10 transform the values to have them at a scale that is easier to visualise and see patterns.

For calling the 'volcano()' function from maser, we need to give it:
- our filtered dataset
- an FDR threshold (e.g., 0.05)
- a deltaPSI threshold (e.g., 0.1)
- the type of event we want to plot

Below, we plot splicing events from type 'A3SS':

```
volcano(samples_filt,
        fdr = 0.05,
        deltaPSI = 0.1,
        type = "A3SS")
```



If we wish to visualise all splicing events together, we can run the function for each splicing event (we just change the argument 'type') and can store each in an individual object.

```
## Generate plot for A3SS splicing events:
A3SS_v_plot <- volcano(samples_filt,
        fdr = 0.05,
        deltaPSI = 0.1,
        type = "A3SS")

## Generate plot for A5SS splicing events:
A5SS_v_plot <- volcano(samples_filt,
        fdr = 0.05,
        deltaPSI = 0.1,
        type = "A5SS")
```

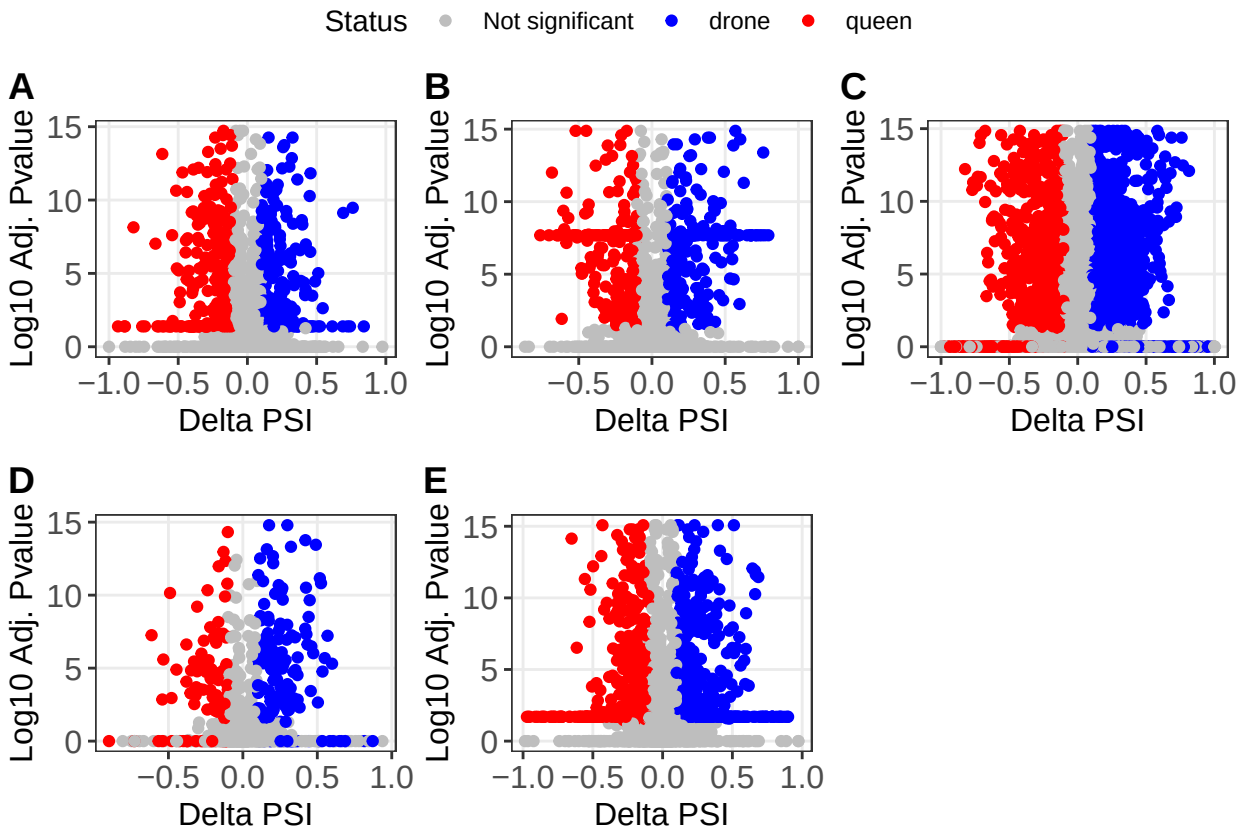
```
## Generate plot for skipped exons splicing events:
SE_v_plot <- volcano(samples_filt,
        fdr = 0.05,
        deltaPSI = 0.1,
        type = "SE")

## Generate plot for retained introns splicing events:
RI_v_plot <- volcano(samples_filt,
        fdr = 0.05,
        deltaPSI = 0.1,
        type = "RI")

## Generate plot for mutually exclusive exons splicing events:
MXE_v_plot <- volcano(samples_filt,
        fdr = 0.05,
        deltaPSI = 0.1,
        type = "MXE")
```

We can either look at these individually or using an R package, such as ggpubr, we can create a multi-panel plot, which allows us to visualise all plots together at the same time.

```
## Generate a combined plot:
ggarrange(A3SS_v_plot,
          A5SS_v_plot,
          SE_v_plot,
          RI_v_plot,
          MXE_v_plot,
          ncol = 3,
          nrow = 2,
          labels = c("A",
                     "B",
                     "C",
                     "D",
                     "E"),
          common.legend = TRUE)
```




```
combined_v_plot <- ggarrange(A3SS_v_plot,
                             A5SS_v_plot,
                             SE_v_plot,
                             RI_v_plot,
                             MXE_v_plot,
                             ncol = 3,
                             nrow = 2,
                             labels = c("A",
                                        "B",
                                        "C",
                                        "D",
                                        "E"),
                             common.legend = TRUE)
```

We can then save out plot in this newly created directory using the 'ggsave()' function from the tidyverse R package, which we installed at the start of the practical. The 'ggsave()' function is highly flexible and can create outputs in different file formats (e.g., PDF, SVG, PNG, JPEG) and the format of the output file is specified by the file extension at the end of the file name you are creating.

For example: - If you named your output file 'output.pdf', the plot will be saved as a PDF.
- If you named your output file 'output.png', the plot will be saved as a PNG.

```
ggsave(filename = "results/combined_volcano_splice_events_gonad.png",
       plot = combined_v_plot)
```

Similarly, if we want to extract filtered information held in the maser-created object, we can use the base R 'summary()' function to extract it and then the base R function 'as.data.frame()' to convert it into a dataframe object that we can then save to a text file.

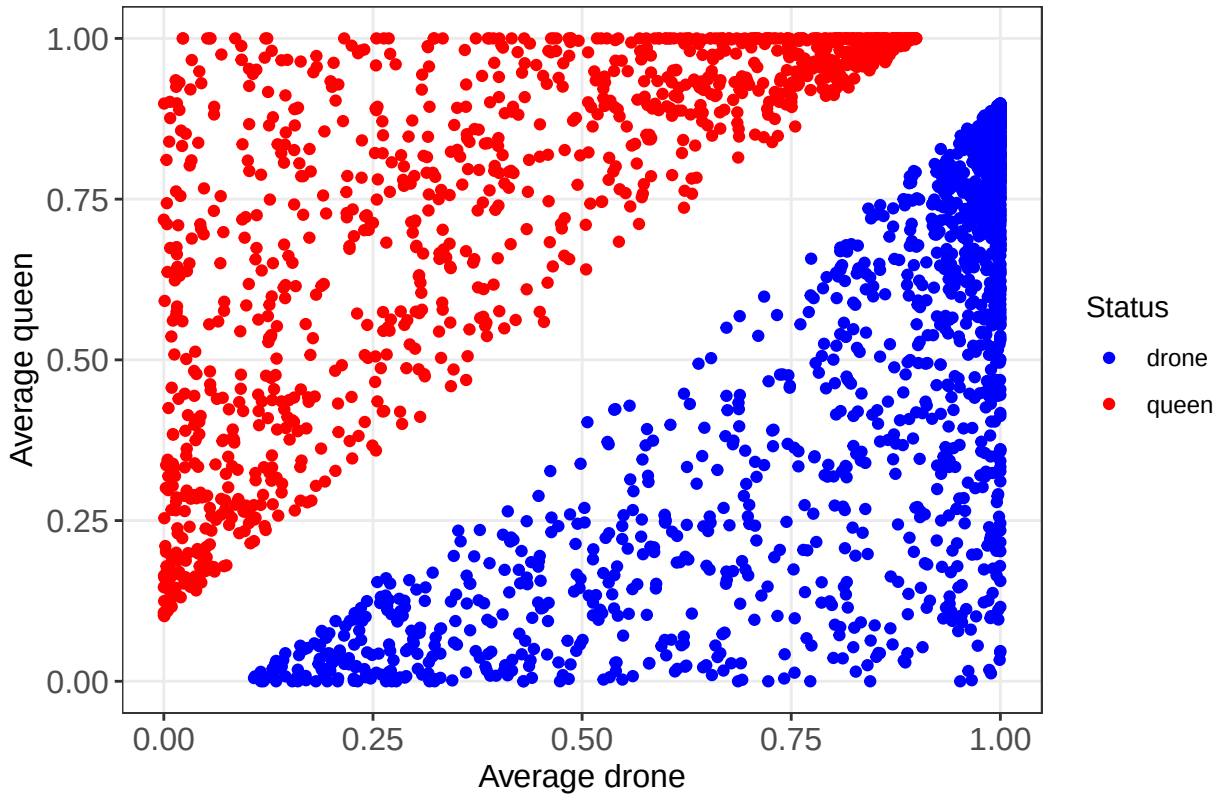
```
samples_filt_df <- as.data.frame(summary(samples_filt))
```

We can then save it to a file using the base R function 'write.table()'.

```
write.table(x = samples_filt_df,
            file = "results/filtered_maser_events_sig_gonad.txt",
            sep = "\t",
            quote = FALSE,
            row.names = FALSE,
            col.names = TRUE)
```

An alternative way of examining splicing differences between groups for each of the splicing events is to use dot plots using a custom function from maser called 'dotplot()'. By default, the 'dotplot()' function sets filters on significance (fdr = 0.05) and variation (deltaPSI = 0.1) so we do not need to set these. All we need to do is specify the type of splicing event we want to visualise.

```
dotplot(samples_top,
        type = "SE")
```



However, as above, if we want to plot all splicing events together, we can again use the function 'ggarrange()' from ggpubr.

```
## Generate plot for A3SS splicing events:
A3SS_d_plot <- dotplot(samples_filt,
                       type = "A3SS")

## Generate plot for A5SS splicing events:
A5SS_d_plot <- dotplot(samples_filt,
                       type = "A5SS")
```

```
## Generate plot for skipped exons splicing events:
SE_d_plot <- dotplot(samples_filt,
                     type = "SE")

## Generate plot for retained introns splicing events:
RI_d_plot <- dotplot(samples_filt,
                     type = "RI")

## Generate plot for mutually exclusive exons splicing events:
MXE_d_plot <- dotplot(samples_filt,
                      type = "MXE")
```

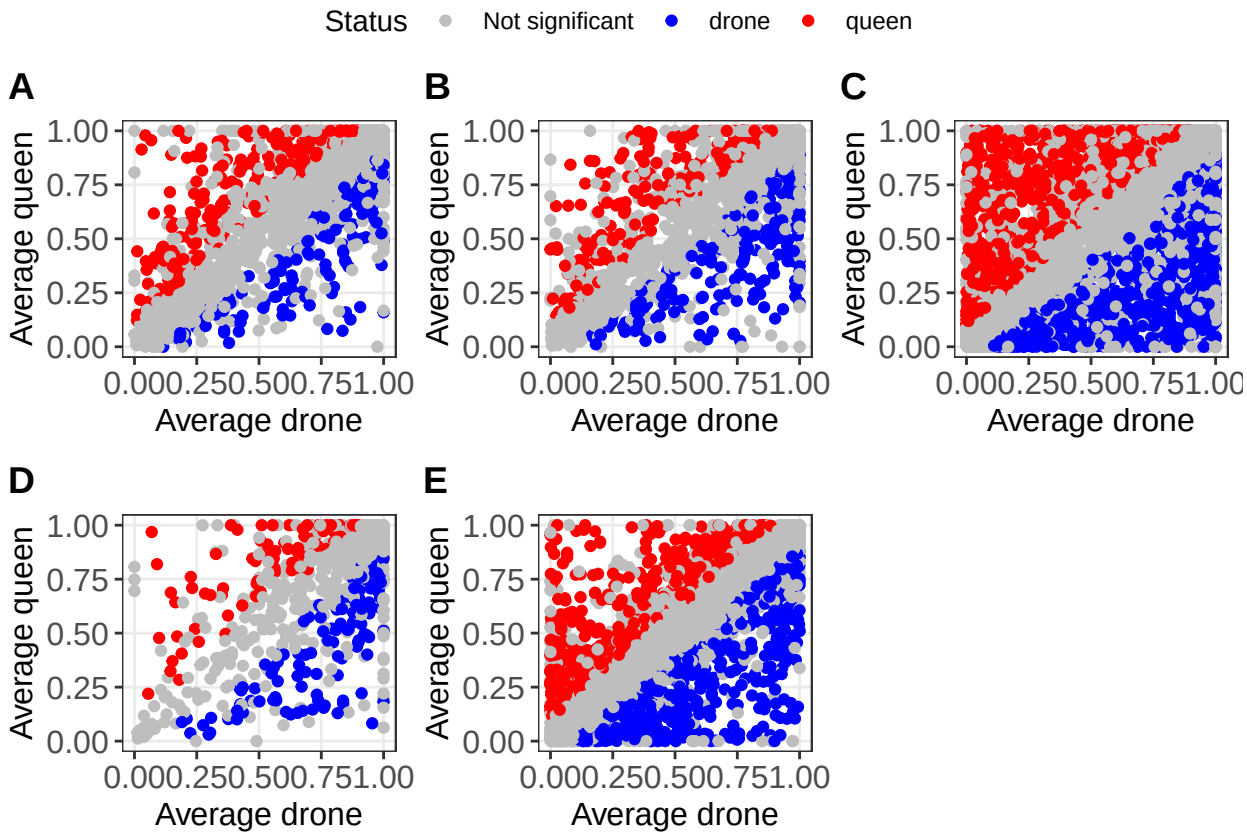
```
combined_d_plot <- ggarrange(A3SS_d_plot,
                             A5SS_d_plot,
                             SE_d_plot,
                             RI_d_plot,
                             MXE_d_plot,
                             ncol = 3,
                             nrow = 2,
                             labels = c("A",
                                        "B",
                                        "C",
                                        "D",
                                        "E"),
                             common.legend = TRUE)
```

As before with the volcano plots, we can save this.

```
ggsave(filename = "results/combined_dotplot_splice_events_gonad.png",
       plot = combined_d_plot)
```

We can also plot the multi-panel plot by printing the contents of the plot object to the console by running:

```
combined_d_plot
```


Status
Not significant
drone
queen
A
B
C
D
E
Average queen
Average drone
1.00
0.75
0.50
0.25
0.00
0.000.250.500.751.00