# Alluvial plot for SFQ and GFQ modules

*Carlos Martinez Ruiz*

*25 July 2019*

```
## [1] "Package ggalluvial version 0.9.1"
## [1] "Package ggplot2 version 3.2.0"
## [1] "Package stats version 3.6.1"
## [1] "Package graphics version 3.6.1"
## [1] "Package grDevices version 3.6.1"
## [1] "Package utils version 3.6.1"
## [1] "Package datasets version 3.6.1"
## [1] "Package methods version 3.6.1"
## [1] "Package base version 3.6.1"
```

The aim of this script is to generate an alluvial plot showing the potential overlap of genes within the gene expression modules detected between Single Founding Queens (SFQ) and Multiple Founding Queens (MFQ). The resulting plot should show that the larger modules in MFQ should contain genes in the multiple smaller SFQ modules. The unique ID per gene was previously extracted, the next step will generate a table with unique gene ID, module and group (SFQ vs GFQ).

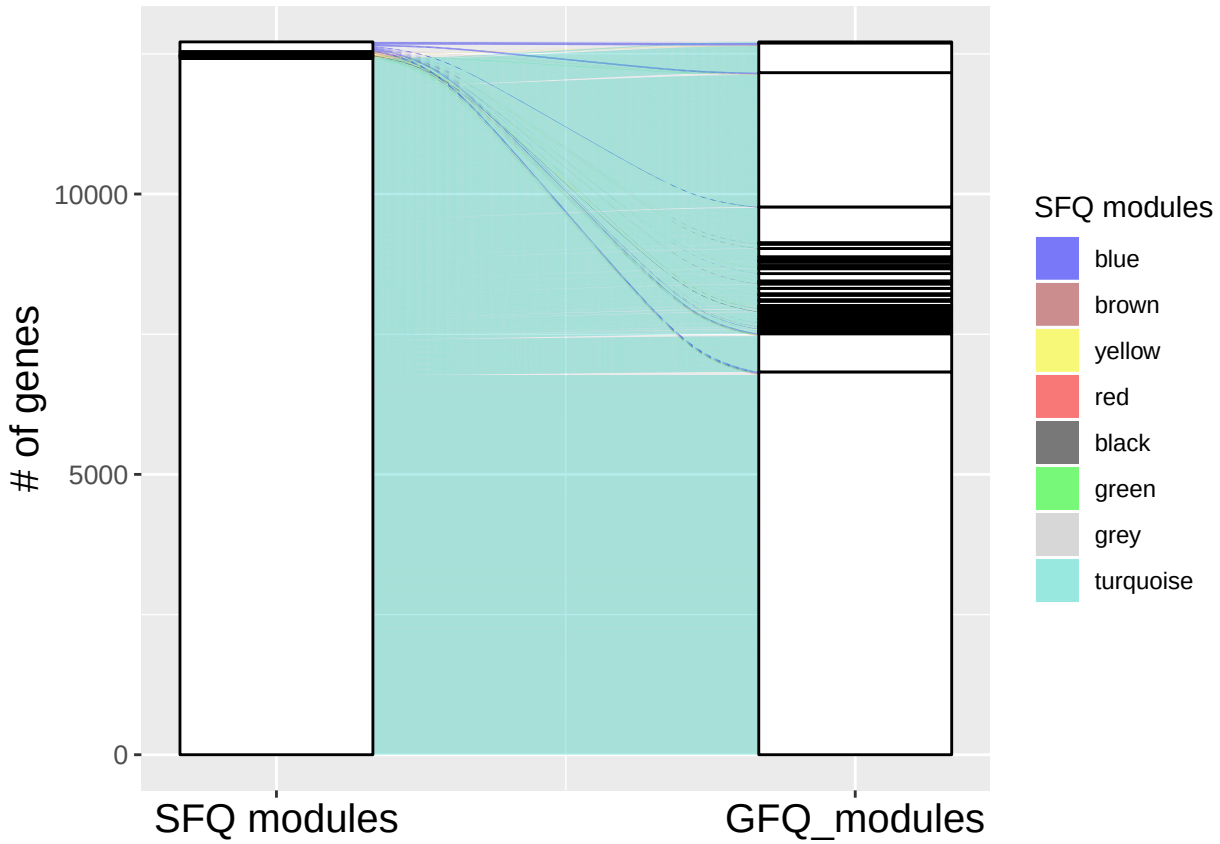
```
#Get the path for all modules
gene_id_files <- list.files("tmp", recursive = TRUE, pattern = "FQ", full.names = TRUE)

#Generate an empty data frame for the output
genes_per_module <- matrix(NA, ncol = 3)
colnames(genes_per_module) <- c("gene", "module", "group")

#Run a loop to generate the data frame
for(this_module in gene_id_files){
  #Load the gene ids for module
  gene_ids <- scan(file = this_module, what = "character")
  #Get the module and group name
  module_name <- gsub(pattern = "(tmp/[GS]FQ_)(.+)(_unique.+)",
                      replacement = "\\2",
                      x = this_module)
  group_name <- gsub(pattern = "(tmp/)([GS]FQ)(_.+)",
                     replacement = "\\2",
                     x = this_module)
  #Generate a matrix with the gene names and the module name
  whole_module <- cbind(gene_ids,
                        rep(module_name, length(gene_ids)),
                        rep(group_name, length(gene_ids)))
  #Stitch the new module to the joint matrix
  genes_per_module <- rbind(genes_per_module, whole_module)
}
#Remove NAs and turn into a data frame
genes_per_module <- as.data.frame(na.omit(genes_per_module))
#Set gene id as character
genes_per_module$gene <- as.character(genes_per_module$gene)
```

The data needs to be parsed to be adapted to a Snakey diagram

```
#Keep genes present in both groups
genes_to_keep <- names(which(table(genes_per_module$gene) == 2))

#Subset data to these genes only
genes_per_module_subset <- subset(x = genes_per_module, subset = gene %in% genes_to_keep)

#Separate the dataframe by groups and rename the 'module' variable to make it group specific
genes_per_module_sfq <- subset(x = genes_per_module_subset, subset = group == "SFQ")
colnames(genes_per_module_sfq)[2] <- "sfq"
genes_per_module_sfq$sfq <- paste0(genes_per_module_sfq$sfq, "_sfq")
genes_per_module_gfq <- subset(x = genes_per_module_subset, subset = group == "GFQ")
colnames(genes_per_module_gfq)[2] <- "gfq"
genes_per_module_gfq$gfq <- paste0(genes_per_module_gfq$gfq, "_gfq")

#Merge both dataframes into one
to_sankey <- merge(genes_per_module_gfq, genes_per_module_sfq, by = "gene")
to_sankey$group.x <- NULL
to_sankey$group.y <- NULL
#Add frequencies for the Sankey plot, in this case, all frequencies are 1 (each gene is only present on
to_sankey$freq <- 1
#Change the order of the modules so that Grey (genes without modules) and turquoise (the biggest module
#at the end
groups <- c("gfq", "sfq")
for(this_group in groups){
  #Get the modules for the group of interest
  modules <- unique(to_sankey[, this_group])
  #Get the correct order of modules (grey and turquoise at the end)
  correct_order <- c(modules[!grepl(pattern = "(^grey_)|(^turquoise_)", x = modules)],
                     paste0("grey_", this_group), paste0("turquoise_", this_group))
  #Generate a factor with the levels in the right order
  to_sankey[, this_group] <- factor(to_sankey[, this_group], levels = correct_order)
}
```

Generate Sankey diagram

```
mod_cols <- gsub(pattern = "_sfq", replacement = "", levels(to_sankey$sfq))
ggplot(to_sankey, aes(y = freq, axis1 = sfq, axis2 = gfq)) +
  geom_alluvium(aes(fill = sfq)) +
  scale_fill_manual(values = mod_cols, labels = mod_cols, name = "SFQ modules") +
  geom_stratum() +
  scale_x_continuous(breaks = c(1,2), labels = c("SFQ modules", "GFQ_modules")) +
  ylab("# of genes") + theme(axis.text.x = element_text(colour = "black", size = 15),
                             axis.text.y = element_text(size = 10),
                             axis.title.y = element_text(size = 15))
```




Generate another Sankey diagram, removing the grey and turquoise modules to have a better look at smaller modules

```
#Subset the data, removing the turquoise and grey modules in both sfq and gfq
genes_per_module_gfq_subset <- subset(x = genes_per_module_gfq,
                                      subset = gfq != "grey_gfq" & gfq != "turquoise_gfq")

genes_per_module_sfq_subset <- subset(x = genes_per_module_sfq,
                                      subset = sfq != "grey_sfq" & sfq != "turquoise_sfq")

#Merge the two datasets
to_sankey_subset <- merge(genes_per_module_gfq_subset, genes_per_module_sfq_subset, by = "gene")
to_sankey_subset$group.x <- NULL
to_sankey_subset$group.y <- NULL
#Add frequencies for the Sankey plot, in this case, all frequencies are 1 (each gene is only present on
to_sankey_subset$freq <- 1
#Make sfq and gfq factors
to_sankey_subset$gfq <- as.factor(to_sankey_subset$gfq)
to_sankey_subset$sfq <- as.factor(to_sankey_subset$sfq)

#Get the colors for the sankey plot
mod_cols_subset <- gsub(pattern = "_sfq", replacement = "", levels(to_sankey_subset$sfq))
#Order the modules to prevent entanglement
ordered_sankey <- to_sankey_subset[order(to_sankey_subset$sfq, to_sankey_subset$gfq), ]
to_sankey_subset$sfq <- factor(ordered_sankey$sfq,
                               levels = unique(ordered_sankey$sfq))

to_sankey_subset$gfq <- factor(ordered_sankey$gfq,
```

```
                              levels = unique(ordered_sankey$gfq))

ggplot(to_sankey_subset, aes(y = freq, axis1 = sfq, axis2 = gfq)) +
  geom_alluvium(aes(fill = sfq)) +
  scale_fill_manual(values = mod_cols_subset, labels = mod_cols_subset, name = "SFQ modules") +
  geom_stratum() +
  scale_x_continuous(breaks = c(1,2), labels = c("SFQ modules", "GFQ_modules")) +
  ylab("# of genes") + theme(axis.text.x = element_text(colour = "black", size = 15),
                             axis.text.y = element_text(size = 10),
                             axis.title.y = element_text(size = 15))
```