ase_comparison_north_america.Rmd

Carlos Martinez Ruiz

23 November 2018

```
## [1] "Package ggplot2 version 2.2.1"
## [1] "Package GenomicFeatures version 1.26.4"
## [1] "Package AnnotationDbi version 1.36.2"
## [1] "Package DESeq2 version 1.14.1"
## [1] "Package SummarizedExperiment version 1.4.0"
## [1] "Package Biobase version 2.34.0"
## [1] "Package GenomicRanges version 1.26.4"
## [1] "Package GenomeInfoDb version 1.10.3"
## [1] "Package IRanges version 2.8.2"
## [1] "Package S4Vectors version 0.12.2"
## [1] "Package BiocGenerics version 0.20.0"
## [1] "Package parallel version 3.4.3"
## [1] "Package stats4 version 3.4.3"
## [1] "Package readr version 1.1.1"
## [1] "Package tximport version 1.2.0"
## [1] "Package stats version 3.4.3"
## [1] "Package graphics version 3.4.3"
## [1] "Package grDevices version 3.4.3"
## [1] "Package utils version 3.4.3"
## [1] "Package datasets version 3.4.3"
## [1] "Package methods version 3.4.3"
## [1] "Package base version 3.4.3"
```

Introduction

This script will analyse the variant specific expression for the supergene in six samples of *Solenopsis invicta* polygyne queens (SbSB genotype) from (Wurm et al., 2011). To do so, the RNAseq reads were aligned to the *S.invicta* reference genome (gnG assembly) and the variant specific read counts obtained by running GATK's ASEreadCounter with a VCF file specific for North American populations.

Import variant specific reads into R

The read counts generated by GATK are stored in a csv file, where one of the columns has the read counts for the reference allele (SB) and another one for the alternative (Sb).

```r
#Loads the counts by variant generated by GATK. All samples!!
ase_files_ppq <- grep("^SRR.*csv",
                      list.files("DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/"
                      value = TRUE)
ase_files_ppq <- paste("DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/",
                       ase_files_ppq, sep = "" )

#This is HARD CODED!!!!!!!!!!!!!!!!!!! Always check!!!!
names(ase_files_ppq) <- c("Queen_1", "Queen_2", "Queen_3",
                          "Queen_4", "Queen_5", "Queen_6")
ase_files_ppq
```

```
##                                                                                  Queen_1
## "DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/SRR619836.na_neutral.csv"
##                                                                                              Queen_2
## "DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/SRR619837.na_neutral.csv"
##                                                                                              Queen_3
## "DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/SRR619838.na_neutral.csv"
##                                                                                              Queen_4
## "DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/SRR619849.na_neutral.csv"
##                                                                                              Queen_5
## "DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/SRR619956.na_neutral.csv"
##                                                                                              Queen_6
## "DC_sinvicta_2016-17/11_2016_ase_analysis/2018-11-23_neutral_ref_ase/SRR619976.na_neutral.csv"
```

```
#Load all the files in R as a single table
ase <- do.call(rbind, lapply(ase_files_ppq, read.table, header = TRUE))

#The table loaded contains all ase counts for Queens from the Morandin data. Each file has different le
#have the same variants.
```

Load variant specific read counts into R

The data loaded into R gives read counts per SNP, rather than per gene. Every SNP needs to be associated to its gene, and the number of read counts per gene averaged across all its SNPs using the median. The position of the genes was taken from the *S.invicta* annotation file (gnG assembly, version GCF_000188075.1). First step, load into R using GRanges the position of all *S.invicta* transcripts.

```
#Transforms the table loaded into a GRanges object, readable by GenomicRanges and GenomicFeatures.
#The first two parameters define the chromosome (scaffold in this case) and the position of the feature
#(in this case, variants). The other parameters are metadata (they could be whatever) in this case,
#the counts for the reference, the alternative allele and total and the sample to which each read belon

ase_gr <- GRanges(Rle(ase$contig), IRanges(start = ase$position, width = 1),
                  count_ref = ase$refCount, count_alt = ase$altCount,
                  tot_count = ase$totalCount,
                  sample = gsub("([A-z]+_[0-9])(\\.[0-9]+)", "\\1", rownames(ase)))


#Loads the 'position_transcripts.txt' file into R. This file contains simply the transcript name,
#the gene to which they are associated and their position in the gnG assembly. This file was generated
# from the actual gnG gff annotation file: 'GCF_000188075.1_Si_gnG_genomic.gff' (more details available
#'~/2016-05-09_DC_S.invicta_project/2016-11-29_rna_id_gnG/')
#The file needs some modifications first, as transcripts without gene associated need to have a blank s

#Position of the transcripts
position_transcripts <- read.table("DC_sinvicta_2016-17/11_2016_ase_analysis/position_transcripts.txt")

colnames(position_transcripts) <- c("contig", "start", "end")

#Transcript and gene names
gene_transcripts <- read.table("DC_sinvicta_2016-17/11_2016_ase_analysis/transcripts_and_genes.txt", fi

colnames(gene_transcripts) <- c("transcript", "gene")

gene_transcripts$transcript <- as.vector(gene_transcripts$transcript)
gene_transcripts$gene <- as.vector(gene_transcripts$gene)

#The blanks are replaced by the name of the transcript
```

```r
for (row in 1:nrow(gene_transcripts)) {
  if (gene_transcripts[row, 2] == "") {
    gene_transcripts[row, 2] <- gene_transcripts[row, 1]
  }
}

position_transcripts <- cbind(gene_transcripts, position_transcripts)

#Generates a GRanges object. Same as before, but this time the metadata refers to the gene name directl
position_transcripts_gr <- GRanges(Rle(position_transcripts$contig),
                                   IRanges(start = position_transcripts$start,
                                           end = position_transcripts$end),
                                   gene = position_transcripts$gene)
```

We also want to want which of these transcripts loaded belong to the supergene, for this the regions of the supergene in the gnG assembly need to be loaded into R. GRanges will be then used to overlap the position of the transcripts with that of the supergene.

```r
#Load the regions for gnG

regions_gng <- read.table("gng_regions.txt", header = TRUE)

#Generate a GRanges object from regions_gng

regions_ranges <- GRanges(Rle(regions_gng$scaffold),
                          IRanges(start = regions_gng$start,
                                  end = regions_gng$end),
                          recomb = regions_gng$region)
```

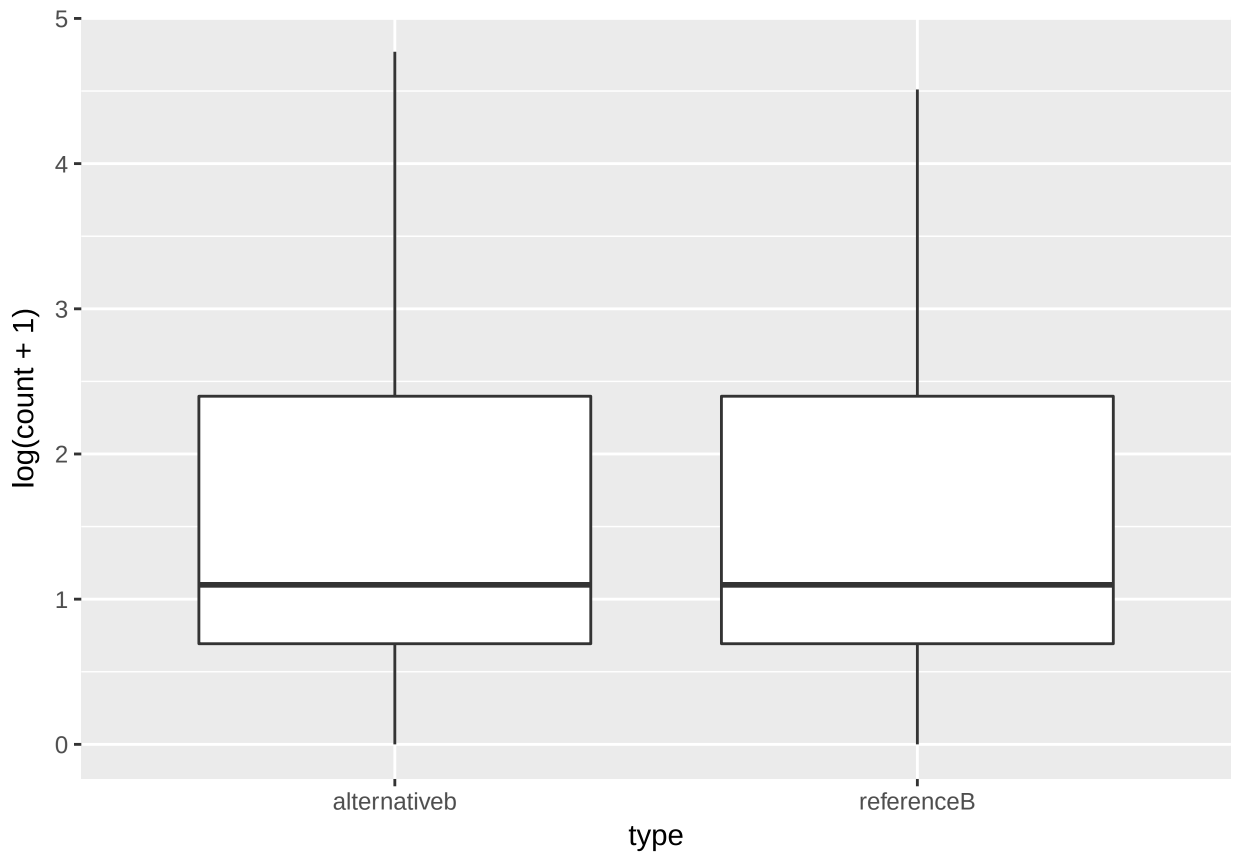
Once the positions have been generated, it is just a matter of overlapping all GRanges objects, to obtain variant specific data per gene for loci in the supergene region.

```r
#This function merges both GRanges objects (the one with the allele counts
# and the other with the transcript anf gene names) according to their overlapping positions.
#This returns a GRanges object with the allele counts per transcript.

ase_by_transcript <- mergeByOverlaps(ase_gr, position_transcripts_gr,
                                     ignore.strand = TRUE)

#Extract Transcript positions and ase counts from ase_by_transcript

ase_by_transcript_df <- as.data.frame(ase_by_transcript)


#Select only relevant fields (keep the position of the variants only, those of the transcripts will yie
ase_by_transcript_df <-  ase_by_transcript_df[, c("ase_gr.seqnames", "ase_gr.start",
                                              "ase_gr.count_ref", "ase_gr.count_alt",
                                              "ase_gr.tot_count", "ase_gr.sample",
                                              "position_transcripts_gr.gene")]

#Remove duplicates (arising from different transcripts belonging to the same gene):
ase_by_transcript_df <- ase_by_transcript_df[!duplicated(ase_by_transcript_df), ]


#Generate another GRanges object with the information (includes gene position and name)
```

```r
ase_by_transcript_pos <- GRanges(Rle(ase_by_transcript_df$ase_gr.seqnames),
                                 IRanges(start = ase_by_transcript_df$ase_gr.start,
                                         width = 1),
                                 count_ref = ase_by_transcript_df$ase_gr.count_ref,
                                 count_alt = ase_by_transcript_df$ase_gr.count_alt,
                                 countTot = ase_by_transcript_df$ase_gr.tot_count,
                                 gene = ase_by_transcript_df$position_transcripts_gr.gene,
                                 sample = ase_by_transcript_df$ase_gr.sample)

#Now, interesct this GRanges object with the recombination positions

regions_ase <- mergeByOverlaps(ase_by_transcript_pos,
                               regions_ranges, ignore.strand = TRUE)

#The GRanges object is clunky and contains a lot of information which is redundant and that I don't nee
#This generates a simplified dataframe with only the position, the allele counts and the transcript and

regions_ase_df <- cbind(as.data.frame(regions_ase$ase_by_transcript_pos),
                        as.data.frame(regions_ase$regions_ranges@elementMetadata))


#The position is not needed, so this is an even more simplified version of the final dataframe.

regions_ase_df <- regions_ase_df <- regions_ase_df[, c("count_ref", "count_alt",
                                                       "countTot", "gene",
                                                       "sample","recomb")]
```

Plot the raw reads per SNP to have a crude look at the variant differences in expression.

```r
#This is essentially the same dataset in this case
to_plot_ase_Bb <- subset(regions_ase_df, recomb == "supergene")

to_plot_ase_Bb_ref <- as.data.frame(to_plot_ase_Bb$count_ref)
to_plot_ase_Bb_ref$type <- rep("referenceB", nrow(to_plot_ase_Bb_ref))
colnames(to_plot_ase_Bb_ref) <- c("count", "type")

to_plot_ase_Bb_alt <- as.data.frame(to_plot_ase_Bb$count_alt)
to_plot_ase_Bb_alt$type <- rep("alternativeb", nrow(to_plot_ase_Bb_alt))

colnames(to_plot_ase_Bb_alt) <- c("count", "type")

to_plot_ase_Bb <- rbind(to_plot_ase_Bb_alt, to_plot_ase_Bb_ref)


ggplot(to_plot_ase_Bb, aes(x = type, y = log(count + 1))) + geom_boxplot()
```



Now the data is ready to be prepared for the DESeq2 analysis. In the next step, the counts per SNP will be aggregated per gene, and the read counts per read summarised using the median. That is, if a gene has 3 SNPs, the read counts per variant will be estimated as the median read count per variant of all 3 SNPs.

```
#This is essentially the same dataset in this ase, however a gene does get excluded.
#I think this is because they might be on the edges of some of the scaffolds,
#So the SNP may fall within the supergene, but most of the gene must be outside.
#In any case, the gene is present only in 4 samples, so it would be filtered out
#in the next step.
to_analyse <- subset(regions_ase_df, recomb == "supergene")

#Collapse the read counts per gene (instead of by variant):
to_analyse_agg     <- aggregate(cbind(count_ref, count_alt) ~ gene + sample,
                                data = to_analyse, median)

#Make sure that all samples have the same genes in the dataframe (some samples have genes that are not
id.table <- table(to_analyse_agg$gene)
to_analyse_agg <- subset(to_analyse_agg,
                         gene %in% names(id.table[id.table == 6]))
```

The data needs to be parsed to be used in DESeq2. Only genes which have counts for all individuals are included in downstream analyses.

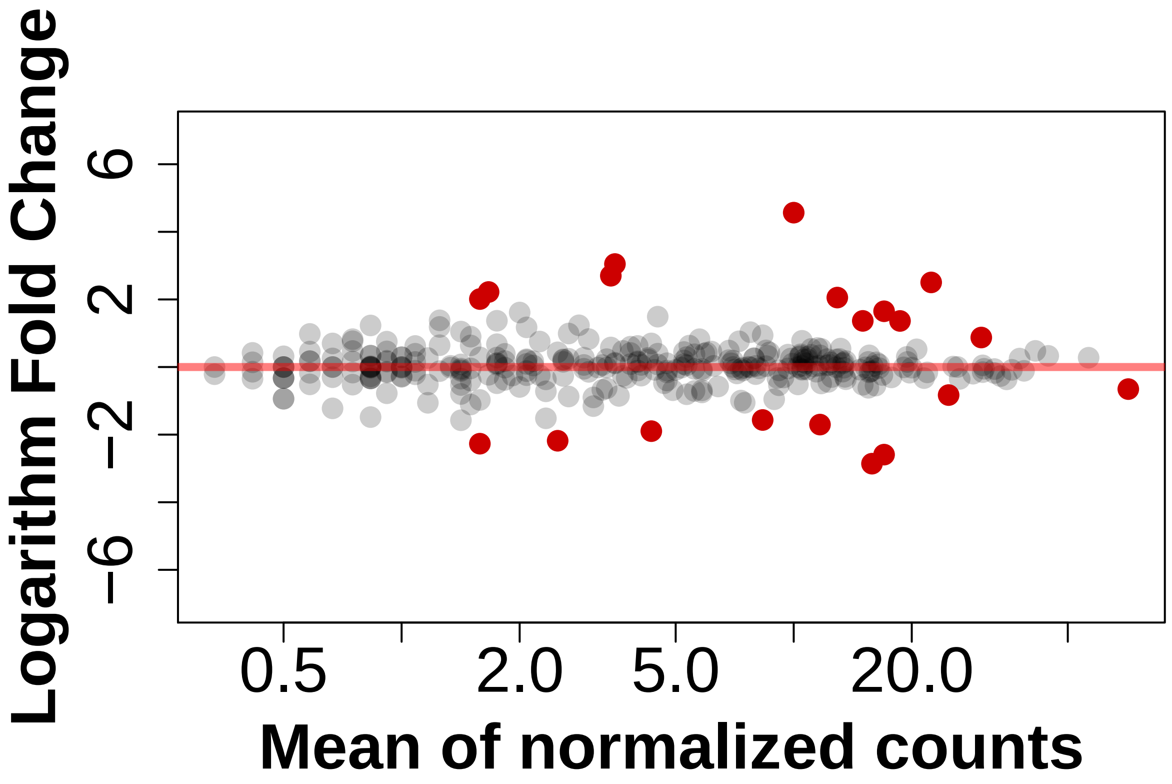
```
#Generate matrix for B read counts:

to_analyse_B <- matrix(data = to_analyse_agg$count_ref, ncol = length(unique(to_analyse_agg$sample)))
```

```r
colnames(to_analyse_B) <- paste(unique(to_analyse_agg$sample), "_B", sep = "")
rownames(to_analyse_B) <- unique(to_analyse_agg$gene)

#Generate matrix for b read counts:
to_analyse_b <- matrix(data = to_analyse_agg$count_alt,
                       ncol = length(unique(to_analyse_agg$sample)))

colnames(to_analyse_b) <- paste(unique(to_analyse_agg$sample),
                                "_b", sep = "")
rownames(to_analyse_b) <- unique(to_analyse_agg$gene)

#Merge the two matrixes into a single one, get data ready for DESeq2:-----------------------------------

to_analyse_DESeq2_polyq <- cbind(to_analyse_b, to_analyse_B)

#Generate the colData data frame for DESeq2

allele <- gsub(x = colnames(to_analyse_DESeq2_polyq),
               pattern = "(Queen_[0-9]_)([A-z])", replacement = "\\2")
colData_Bb <- data.frame(allele)
colData_Bb$sample <- rep(unique(to_analyse_agg$sample), 2)
rownames(colData_Bb) <- colnames(to_analyse_DESeq2_polyq)
```

DESeq2 analysis

The analysis for the allele specific expression between the SB and Sb variants is performed in DESeq2. Because the variant specific counts that are being compared always come from the same sample (and therefore, the same library), performing normalisation would flatten actual differences. The analysis therefore needs to be performed ensuring that the samples are not normalised (more info herehttp://rpubs.com/mikelove/ase).

```r
#Make sure all count values are integers:
to_analyse_DESeq2_polyq <- round(to_analyse_DESeq2_polyq)
#Load the data into DESeq2
dds_Bb <- DESeqDataSetFromMatrix(countData = to_analyse_DESeq2_polyq,
                                 colData = colData_Bb, design = ~ sample + allele)
sizeFactors(dds_Bb) <- rep(1, ncol(to_analyse_DESeq2_polyq))
#Perform the analysis:
dds_Bb_DE <- DESeq(dds_Bb)

#Get results:
res_Bb <- results(dds_Bb_DE, contrast = c("allele", "B", "b"))

#Let's have a look:

#Create a 'transparent' black color:
trans_black <- adjustcolor("black", alpha.f = 0.2)

par(mar = c(5, 4.6, 4, 2) + 0.1, mfrow = c(1, 1))
plotMA(res_Bb, alpha = 0.05,  ylim = c(-7, 7), xlab = "Mean of normalized counts",
       colNonSig = trans_black, ylab = "Logarithm Fold Change", font.lab = 2,
       cex.axis = 2, cex.lab = 2, cex.main = 2.5, cex = 1.5)
```



Enrichment analyses

Is there a biase towards SB? To check for this, we should expect the median of the ASE LFCs to be significantly positive. THat, is to find more highly expressed genes in SB than in Sb in general.

```
median(res_Bb$log2FoldChange)
```

```
## [1] 0.002138616
```

```
wilcox.test(res_Bb$log2FoldChange)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  res_Bb$log2FoldChange
## V = 28940, p-value = 0.2141
## alternative hypothesis: true location is not equal to 0
```

Additional tests to check for enrichment: Binomial test for significantly DE genes, and chi squared tests for DE genes in SB and Sb.

```
#How many DE genes over-expressed in Sb?
over_e_lb <- length(which(res_Bb$log2FoldChange < 0 & res_Bb$padj < 0.05))
#8

#How many DE genes over-expressed in SB?
over_e_bb <- length(which(res_Bb$log2FoldChange > 0 & res_Bb$padj < 0.05))
#21
```

```r
#Total genes in the supergene region:
total_genes_Bb <- length(res_Bb$pvalue)

#Chi2 to check wether any of the variants in enriched

to_test_chi           <- matrix(c(over_e_lb, total_genes_Bb - over_e_lb, over_e_bb,
                                  total_genes_Bb - over_e_bb), nrow = 2, ncol = 2)
colnames(to_test_chi) <- c("Bb", "BB")
rownames(to_test_chi) <- c("Over-expresed", "Non-DE")

chisq.test(to_test_chi)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  to_test_chi
## X-squared = 0.051567, df = 1, p-value = 0.8204
```

Variant specific data vs social form comparisons

It has been established that the SB alleles tend to be more highly expressed than their Sb counterparts. But of the few genes whose Sb allele is more highly expressed, are those genes also more highly expressed in general in polygyne queens? To test for this, the results from the comparison between social forms needs to be loaded into R again.

```r
load("DC_sinvicta_2016-17/robjects/morandin_analysis/2018-10-31_results_deseq2_queens_mor.RData")
load("DC_sinvicta_2016-17/robjects/morandin_analysis/2018-10-31_results_deseq2_workers_mor.RData")
```

Some transcripts with no genes associated to them have different names in the different result objects. Before the comparison, all objects need to have exactly the same names:

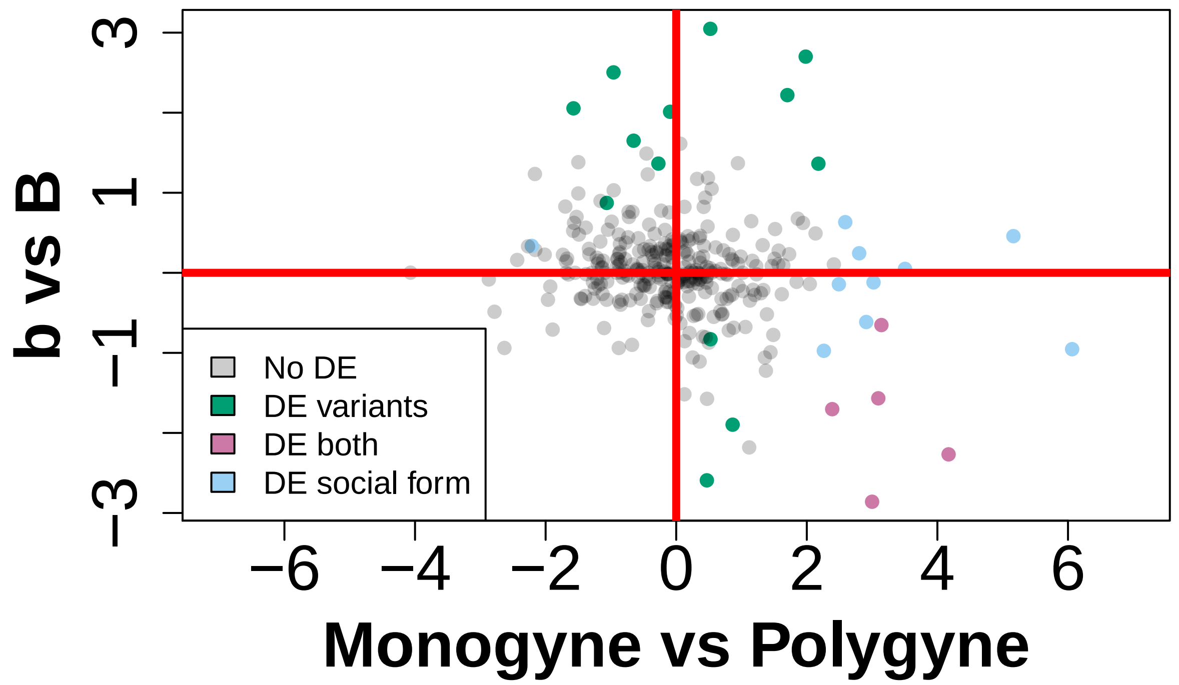
```r
ase_genes <- rownames(res_Bb)

#Load tx names with XP format:
#This file was generated in the directory /data/home/btx076/2016-05-09_DC_S.invicta_project/rna_ID in A
rna_tx <- read.table("rna_genes_xp.txt", fill = TRUE)
colnames(rna_tx) <- c("transcript", "gene_xp")
rna_tx$transcript <- as.character(rna_tx$transcript)
rna_tx$gene_xp <- as.character(rna_tx$gene_xp)

#Gene ID has a blank space for those transcripts that do not have an associated gene.
#These blanks are replaced by the name of the transcript
for (row in 1:nrow(rna_tx)) {
  if (rna_tx[row, 2] == "") {
    rna_tx[row, 2] <- rna_tx[row, 1]
  }
}

#Change the rna name by its tx (XP) name in the ase comparison
ase_genes[ase_genes %in% rna_tx$transcript] <- rna_tx[rna_tx$transcript %in% ase_genes, ]$gene_xp
rownames(res_Bb) <- ase_genes
```

Next, a dataset with both comparisons needs to be prepared, ensuring that all the genes are the same in both datasets and are in the same order.

```r
#Genes names equivalent in the Morandin data
genes_morandin <- rownames(res_de_txi_q)

#Get LFCs for each comparison, making sure that all genes are in the same order
lfcs_morandin           <- data.frame(res_de_txi_q[rownames(res_de_txi_q) %in% rownames(res_Bb), ])
lfcs_morandin$gene      <- rownames(res_de_txi_q[rownames(res_de_txi_q) %in% rownames(res_Bb), ])
lfcs_morandin$gene      <- as.factor(lfcs_morandin$gene)
lfcs_morandin           <- data.frame(lfcs_morandin$gene, lfcs_morandin$log2FoldChange, lfcs_morandin$pa
colnames(lfcs_morandin) <- c("gene", "lfcs_mor", "padj_mor")

lfcs_ase           <- data.frame(res_Bb[rownames(res_Bb) %in% lfcs_morandin$gene, ])
lfcs_ase$gene      <- rownames(res_Bb[rownames(res_Bb) %in% lfcs_morandin$gene, ])
lfcs_ase$gene      <- as.factor(lfcs_ase$gene)
lfcs_ase           <- data.frame(lfcs_ase$gene, lfcs_ase$log2FoldChange, lfcs_ase$padj)
colnames(lfcs_ase) <- c("gene", "lfcs_ase", "padj_ase")

#Make sure everything has the same order
lfcs_ase   <- lfcs_ase[order(lfcs_ase$gene), ]
lfcs_morandin <- lfcs_morandin[order(lfcs_morandin$gene), ]
if (!identical(lfcs_ase$gene, lfcs_morandin$gene)) {
  warning("The genes are not the same in both dataframes!")
}
```

Plot the LFCs for both comparisons. This plot shows the LFCs for the comparisons between social forms in queens (from the Morandin et al., 2016 dataset) in the x axis and the LFCs for the comparisons between the SB and Sb variants in polygyne queens (from the Wurm et al., 2011 dataset) in the y axis. The plot also shows the genes which are significantly differentially expressed (according to DESeq2) between social forms only (light blue dots), between variants only (green dots) or in both comparisons (purple dots).

```r
#Get a colour vector for all values based on the significance of the padj
lfcs_joint <- data.frame(cbind(lfcs_ase, lfcs_morandin), stringsAsFactors = FALSE)


lfcs_joint$cols <- trans_black
lfcs_joint$cols[lfcs_joint$padj_ase < 0.05 & lfcs_joint$padj_mor >= 0.05] <- "#009E73"
lfcs_joint$cols[lfcs_joint$padj_ase >= 0.05 & lfcs_joint$padj_mor < 0.05] <- "#9ad0f3"
lfcs_joint$cols[lfcs_joint$padj_ase < 0.05 & lfcs_joint$padj_mor < 0.05] <- "#CC79A7"


#Plot the thing(and save in png):
#png("DC_sinvicta_2016-17 / 05-0216-CD-analyses / ase_plot.png",
#    width = 10.6, height = 6.3, units = "in", res = 800)
par(mfrow = c(1, 1), mar = c(5, 4.6, 4, 2) + 0.1)
plot(lfcs_morandin$lfcs_mor, lfcs_ase$lfcs_ase, xlab = "Monogyne vs Polygyne",
     ylab = "b vs B", xlim = c(-7, 7), font.lab = 2, cex.axis = 2, cex.lab = 2,
     cex.main = 2.5, pch = 16, col = lfcs_joint$cols)
legend("bottomleft", legend = c("No DE", "DE variants", "DE both", "DE social form"),
       fill = unique(lfcs_joint$cols))
abline(h = 0, col = "red", lwd = 4)
abline(v = 0, col = "red", lwd = 4)
```



Test for enrichment in Sb for polygyne biased genes, via a chi squared test, and by comparing the LFCs distributions across variants in highly and lowly expressed genes in polygyne queens through a Kolmogorov-Smirnov test. The plot of the distributions can be generated by un-commenting lines 417 to 424.

```
####Compare the ASE distributions across a Mono vs Poly gradient
###### Select the Morandin LFCs from 2 sections of the dataset
#Get rid of the NAs first
lfcs_joint_nona <- lfcs_joint[!is.na(lfcs_joint$lfcs_mor), ]
#Sort the new dataset by morandin LFCs
lfcs_joint_mor_sorted <- lfcs_joint_nona[order(lfcs_joint_nona$lfcs_mor), ]

#Using only two halves (M vs P)
lfcs_joint_mor_sorted$mor_halves <- NA

lfcs_joint_mor_sorted$mor_halves[1:round((1 / 2 * nrow(lfcs_joint_mor_sorted)))] <- "HiM"
lfcs_joint_mor_sorted$mor_halves[(round(1 / 2 * nrow(lfcs_joint_mor_sorted)) + 1):nrow(lfcs_joint_mor_s
#lfcs_joint_mor_sorted$mor_halves[lfcs_joint_mor_sorted$lfcs_mor < 0] <- "HiM"
#lfcs_joint_mor_sorted$mor_halves[lfcs_joint_mor_sorted$lfcs_mor > 0] <- "HiP"

#Now it's just a matter of comparing the B vs b distributions of LFCs between quarters
#ggplot(lfcs_joint_mor_sorted, aes(x = lfcs_ase, fill = mor_halves)) + geom_density(alpha = 0.3) + them
#  labs(x = "Logarithm fold change between variants", y = "Density") +
#  theme(panel.border = element_blank(), panel.grid.major = element_blank(),
#        panel.grid.minor = element_blank(), axis.line = element_line(colour = "black")) +
#  theme(axis.title.x = element_text(face = "bold", size = 30),
#        axis.text.x  = element_text(angle = 90, vjust = 0.5, size = 25)) +
#  theme(axis.title.y = element_text(face = "bold", size = 30),
#        axis.text.y  = element_text(angle = 90, vjust = 0.5, size = 25))

#Kolmogorov-Smirnov tests between quarters to see how different the distributions are
ks.test(lfcs_joint_mor_sorted$lfcs_ase[lfcs_joint_mor_sorted$mor_halves == "HiM"],
        lfcs_joint_mor_sorted$lfcs_ase[lfcs_joint_mor_sorted$mor_halves == "HiP"])
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  lfcs_joint_mor_sorted$lfcs_ase[lfcs_joint_mor_sorted$mor_halves ==  and lfcs_joint_mor_sorted$
## D = 0.20497, p-value = 0.002309
## alternative hypothesis: two-sided
```

```
#Compare medians using Mann Whitney
wilcox.test(lfcs_joint_mor_sorted$lfcs_ase[lfcs_joint_mor_sorted$mor_halves == "HiM"],
            lfcs_joint_mor_sorted$lfcs_ase[lfcs_joint_mor_sorted$mor_halves == "HiP"])
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  lfcs_joint_mor_sorted$lfcs_ase[lfcs_joint_mor_sorted$mor_halves ==  and lfcs_joint_mor_sorted$
## W = 15816, p-value = 0.0006323
## alternative hypothesis: true location shift is not equal to 0
```

```
#From all Sb over-expressed genes, what is the proportion of polygyne genes? Is it different in SB?
#You need to use the clean df without NAs in the Morandin LFCs. NA in the Morandin LFCs indicate
#that the count for that gene was 0 for Morandin but not for the 2011 queen data, it is thus
#an incongruence and those genes should be removed.

#Total over-expressed genes in Sb
tot_lb <- length(which(lfcs_joint_mor_sorted$lfcs_ase < 0 & lfcs_joint_mor_sorted$padj_ase < 0.05))
#Polygyne biased genes in Sb
pol_lb <- length(which(lfcs_joint_mor_sorted$lfcs_ase < 0 & lfcs_joint_mor_sorted$padj_ase < 0.05 &
                         lfcs_joint_mor_sorted$lfcs_mor > 0 & lfcs_joint_mor_sorted$padj_mor < 0.05))

#Total over-expressed genes in SB
tot_bb <- length(which(lfcs_joint_mor_sorted$lfcs_ase > 0 & lfcs_joint_mor_sorted$padj_ase < 0.05))
#Polygyne biased genes in SB
pol_bb <- length(which(lfcs_joint_mor_sorted$lfcs_ase > 0 & lfcs_joint_mor_sorted$padj_ase < 0.05 &
                         lfcs_joint_mor_sorted$lfcs_mor > 0 & lfcs_joint_mor_sorted$padj_mor < 0.05))

#Chi2 test

to_test_chi_sb           <- matrix(c(pol_lb, tot_lb, pol_bb, tot_bb), nrow = 2, ncol = 2)
colnames(to_test_chi_sb) <- c("Sb", "SB")
rownames(to_test_chi_sb) <- c("Polygyne_only", "Total")

chisq.test(to_test_chi_sb)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  to_test_chi_sb
## X-squared = 2.6057, df = 1, p-value = 0.1065
```

```
#Significant, p-value = 0.01193
#Try with Fisher test
fisher.test(to_test_chi_sb)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  to_test_chi_sb
## p-value = 0.05303
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.7524778       Inf
## sample estimates:
## odds ratio
##        Inf
```

```
#Significant, p-value = 0.005423
```