Comparison social forms *Solenopsis invcita* using the Morandin et al., 2016 data

Carlos Martinez Ruiz

31 October 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

The Morandin et al., 2016https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0902-7 data consits in 3 samples from workers and queens from the two social forms of *Solenopsis invicta*. The aim of this analysis is to obtain differentially expressed genes between social forms in queens and workers independently and check whether or not the social chromosome is enriched in DEGs. The counts used for this analysis were generated by Kalisto (v0.43.0), using the available *S.invicta* coding sequences (gnG assembly, version GCF_000188075.1). Two samples were removed from the analysis due to poor alignment, a polygyne queen and a monogyne worker.

Import data from Kallisto into R using 'tximport'

The data from Kallisto first needs to be loaded into R, removing any problematic samples. In this case two samples were removed, a polygyne queen (sample DRR042171) and a monogyne worker (DRR042169), due to poor alignment to the reference. Tha samples then have a more workable name assigned and stored into colData, a matrix that will be then used in DESeq. This matrix assigns factors to each sample, in this case, caste and social form. Both the names and the assigned factors were hard coded, so it is worth checking the table once generated.

```
# Load Kallisto abundance files

paths <- list()
paths$data <- "DC_sinvicta_2016-17/2017-09-TEs_sinvicta/cds_from_genomic_simple/"
```

```r
paths$kallisto_output <- file.path(paths$data)

paths$kallisto_files_relative <- grep(x       = list.files(paths$kallisto_output,
                                                           recursive = TRUE),
                                      pattern = ".tsv",
                                      value   = TRUE)

paths$kallisto_files <- file.path(paths$kallisto_output,
                                  paths$kallisto_files_relative)



#Extract the sample names:
samples <- gsub(x           = paths$kallisto_files_relative,
                pattern     = "/.*",
                replacement = "")

samples <- gsub(x           = samples,
                pattern     = "no_tes_simple",
                replacement = "")

#Label the files:
names(paths$kallisto_files) <- samples
#Generate the colData table for DESeq2 analysis:
soc_form <- rep(c("Mono", "Poly"), each = 6)
caste    <- rep(rep(c("Queen", "Worker"), each = 3), 2)
sample   <- paste0(rep(rep(c("_Q", "_W"), each = 3), 2), rep(c(1:3), 3))
sample   <- paste0(soc_form, sample)


colData           <- matrix(c(sample, caste, soc_form), ncol = 3)
rownames(colData) <- samples
colnames(colData) <- c("sample", "caste", "social_form")

#Remove problematic samples Poly_Q1 and Mono_W2
colData <- colData[-grep("Poly_Q1|Mono_W2", colData[, 1]), ]

paths$kallisto_files <- paths$kallisto_files[-grep("DRR042171|DRR042169",
                                                   paths$kallisto_files)]
#Check the resulting matrix
colData
```

```
##           sample    caste    social_form
## DRR042165 "Mono_Q1" "Queen"  "Mono"
## DRR042166 "Mono_Q2" "Queen"  "Mono"
## DRR042167 "Mono_Q3" "Queen"  "Mono"
## DRR042168 "Mono_W1" "Worker" "Mono"
## DRR042170 "Mono_W3" "Worker" "Mono"
## DRR042172 "Poly_Q2" "Queen"  "Poly"
## DRR042173 "Poly_Q3" "Queen"  "Poly"
## DRR042174 "Poly_W1" "Worker" "Poly"
## DRR042175 "Poly_W2" "Worker" "Poly"
## DRR042176 "Poly_W3" "Worker" "Poly"
```

Once the data has been loaded into R, it needs to be ran through tximport to make it compatible with DESeq2. The analysis in DESeq2 needs to be performed independently in workers and queens, so the data will be divided in two.The analysis will be performed in genes rather than transcripts (the original input). Tximport will assign the counts per transcript to their corresponding genes. For this to work, a table with each transcript and its corresponding gene needs to be loaded into R. This table was generated based on the gene annotation from RefSeq (gnG assembly, version GCF_000188075.1).

```
#Separate data into workers and queens:
info <- list()
info$w <- colData[colData[, 2] == "Worker", ]
info$q <- colData[colData[, 2] == "Queen", ]

kallisto_files_w <- paths$kallisto_files[rownames(info$w)]
kallisto_files_q <- paths$kallisto_files[rownames(info$q)]

#Load a table with the transcript ID and the gene to which they belong.
#This file was generated in the directory
#/data/home/btx076/2016-05-09_DC_S.invicta_project/rna_ID in Apocrita
tx_gene <- read.table("xp_loc.txt", fill = TRUE)
colnames(tx_gene) <- c("transcript", "gene")
tx_gene$transcript <- as.character(tx_gene$transcript)
tx_gene$gene <- as.character(tx_gene$gene)

#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(tx_gene)) {
  if (tx_gene[row, 2] == "") {
    tx_gene[row, 2] <- tx_gene[row, 1]
  }
}

#Import the abundance files into a DESeq2 object

txi_reads_w <- tximport(kallisto_files_w, type = "kallisto", reader = read_tsv,
                        tx2gene = tx_gene)
txi_reads_q <- tximport(kallisto_files_q, type = "kallisto", reader = read_tsv,
                        tx2gene = tx_gene)
```

Perform the DESeq2 analysis

Once the data is ready, the analysis with DESeq2 can be performed by simply running `DESeq()`. **NOTE:** Because this version of DESeq2 is prior to the 1.16 update, the LFCs are automatically shrinked. In newer versions, `betaPrior = TRUE` needs to be added to the `DESeq()` function to get the same results.

```
#FOR QUEENS:
ddsTxi_q <- DESeqDataSetFromTximport(txi_reads_q, colData = info$q,
                                     design = ~ social_form)

#Performs DESeq2 test for detecting DE loci
de_txi_q <- DESeq(ddsTxi_q)

#summary of the results:
res_de_txi_q <- results(de_txi_q)

#FOR WORKERS:
```

```
ddsTxi_w <- DESeqDataSetFromTximport(txi_reads_w, colData = info$w,
                                     design = ~ social_form)

#Performs DESeq2 test for detecting DE loci
de_txi_w <- DESeq(ddsTxi_w)

#summary of the results:
res_de_txi_w <- results(de_txi_w)
#Save the results
res_q_name <- paste0("DC_sinvicta_2016-17/robjects/morandin_analysis/", Sys.Date(),
                     "_results_deseq2_queens_mor.RData")
save(res_de_txi_q, file = res_q_name)
res_w_name <- paste0("DC_sinvicta_2016-17/robjects/morandin_analysis/", Sys.Date(),
                     "_results_deseq2_workers_mor.RData")
save(res_de_txi_w, file = res_w_name)
```

How many loci were detected by DESeq as differentially expressed? **In workers**

```
sum(res_de_txi_w$padj < 0.05, na.rm = TRUE)
```

```
## [1] 318
```

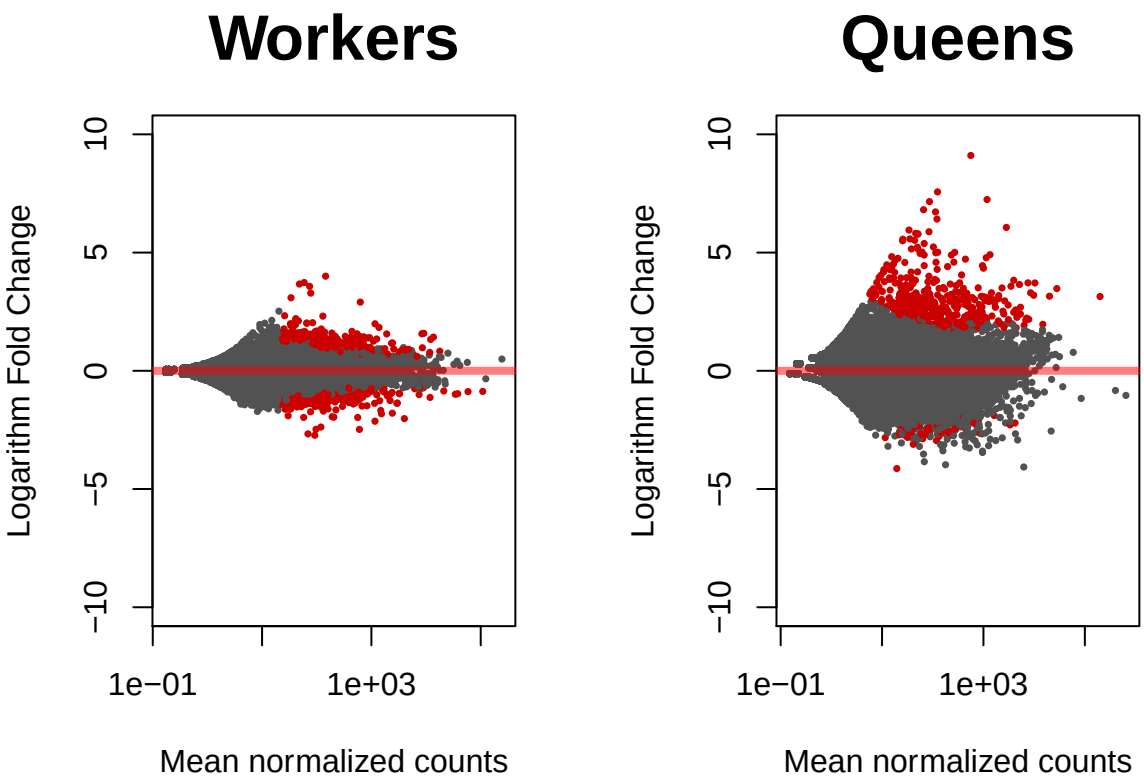
In queens

```
sum(res_de_txi_q$padj < 0.05, na.rm = TRUE)
```

```
## [1] 310
```

Plot the results in an MA plot, where the y axis is the LFCs between Polygyne and Monogyne individuals (i.e., positive values imply higher expression in polygyne colonies and vice-versa) and the x axis is the mean read count per gene. Red dots indicate genes which have been detected as significant (alpha = 0.05) according to DESeq. The differences between social forms are plotted independently for queens and workers.

```
par(mfrow = c(1, 2), mar = c(5, 4.6, 4, 2) + 0.1)
plotMA(res_de_txi_w, main = "Workers", ylim = c(-10, 10), xlab = "Mean normalized counts",
       ylab = "Logarithm Fold Change", font.lab = 1, cex.axis = 1, cex.lab = 1,
       cex.main = 2, cex = 0.5)
plotMA(res_de_txi_q, main = "Queens", ylim = c(-10, 10), xlab = "Mean normalized counts",
       ylab = "Logarithm Fold Change", font.lab = 1, cex.axis = 1, cex.lab = 1,
       cex.main = 2, cex = 0.5)
```



Genome location analysis

Once the differentially expressed genes between social forms have been detected, the next question to answer is whether there is an enrichment of socially biased loci in the supergene region. For this, a table with the position of the supergene in the *Solenopsis invicta* genome (gnG assembly) needs to be loaded. The overlap between the position of the supergene and the location of the differentially expressed genes in then calculated using the packages GenomeRanges and IRanges. First step, get the positions of the supergene and of all the genes in the genome into R

```
#Load the file containing the last known position of the inversion in
#the S.invicta genome (for gnG genome ref)
regions <- read.table("gng_regions.txt", header = TRUE)

#Load annotation file
annot <- makeTxDbFromGFF(file = "GCF_000188075.1_Si_gnG_genomic.gff")
#Returns coding regions by transcript (option "gene", otherwise it would extract features
#(transcripts in this case) by transcript using the option "tx").
ann_ranges <- transcriptsBy(annot, "gene")
ann_ranges <- unlist(ann_ranges)
#Create a GRanges object from the 'regions' table
regions_ranges <- GRanges(Rle(regions$scaffold), IRanges(start = regions$start,
                                                         end = regions$end),
                          lg = regions$lg, region = regions$region)

#Create a GRanges object intersecting each gene to its corresponding region
gene_region <- mergeByOverlaps(regions_ranges, ann_ranges, ignore.strand = TRUE)
```

```
#Parse the 'gene_region' GRanges object to make it useful
loc_genes_region           <- data.frame(gene_region@listData$region,
                                         gene_region@listData$ann_ranges@ranges@NAMES)
colnames(loc_genes_region) <- c("region", "loc_id")
loc_genes_region           <- unique(loc_genes_region)
```

Once the position of all genes within the genome (within or outside of the supergene) has been established, we can finally calculate the number of socially biased loci (defined as those detected as differentially expressed between social forms by DESeq2) within and outside the supergene. It would be also interesting to look at the number of genes significantly more highly expressed in polygyne (and vice-versa). All analyses are performed in queens and workers independently.

```
#Make sure that only genes present in the annotation file are present
# in the DEseq2.
res_de_w <- res_de_txi_w[rownames(res_de_txi_w) %in% loc_genes_region$loc_id, ]
res_de_q <- res_de_txi_q[rownames(res_de_txi_q) %in% loc_genes_region$loc_id, ]

#Get gene names of DE loci in workers and queens
de_loci_w <- rownames(res_de_w[which(res_de_w$padj < 0.05), ])
de_loci_q <- rownames(res_de_q[which(res_de_q$padj < 0.05), ])

#Get gene names of loci over-expressed in polygyne workers and queens
overp_loci_w <- rownames(res_de_w[which(res_de_w$log2FoldChange > 0 &
                                        res_de_w$padj < 0.05), ])
overm_loci_w <- rownames(res_de_w[which(res_de_w$log2FoldChange < 0 &
                                        res_de_w$padj < 0.05), ])

overp_loci_q <- rownames(res_de_q[which(res_de_q$log2FoldChange > 0 &
                                        res_de_q$padj < 0.05), ])
overm_loci_q <- rownames(res_de_q[which(res_de_q$log2FoldChange < 0 &
                                        res_de_q$padj < 0.05), ])

#Add columns for de genes in workers and queens to the regions df
loc_genes_region$de_w <- "non-significant"
loc_genes_region$de_q <- "non-significant"

#Label de genes in workers and queens
loc_genes_region[match(de_loci_q, loc_genes_region$loc_id), ]$de_q <- "significant"
loc_genes_region[match(de_loci_w, loc_genes_region$loc_id), ]$de_w <- "significant"

#Add columns the direction of over-expression in workers and queens
loc_genes_region$expr_dir_q <- "Non-diff"
loc_genes_region$expr_dir_w <- "Non-diff"

#Label over expressed genes in polygyne workers and queens
loc_genes_region[match(overp_loci_q, loc_genes_region$loc_id), ]$expr_dir_q <- "Poly"
loc_genes_region[match(overp_loci_w, loc_genes_region$loc_id), ]$expr_dir_w <- "Poly"

loc_genes_region[match(overm_loci_q, loc_genes_region$loc_id), ]$expr_dir_q <- "Mono"
loc_genes_region[match(overm_loci_w, loc_genes_region$loc_id), ]$expr_dir_w <- "Mono"

#Workers
de_rec_w <- length(which(loc_genes_region$region == "recombining" &
                         loc_genes_region$de_w == "significant"))
```

```
non_de_rec_w <- length(which(loc_genes_region$region == "recombining" &
                               loc_genes_region$de_w == "non-significant"))

de_non_rec_w <- length(which(loc_genes_region$region == "supergene" &
                               loc_genes_region$de_w == "significant"))

non_de_non_rec_w <- length(which(loc_genes_region$region == "supergene" &
                                   loc_genes_region$de_w == "non-significant"))

#Queens
de_rec_q <- length(which(loc_genes_region$region == "recombining" &
                           loc_genes_region$de_q == "significant"))

non_de_rec_q <- length(which(loc_genes_region$region == "recombining" &
                               loc_genes_region$de_q == "non-significant"))

de_non_rec_q <- length(which(loc_genes_region$region == "supergene" &
                               loc_genes_region$de_q == "significant"))

non_de_non_rec_q <- length(which(loc_genes_region$region == "supergene" &
                                   loc_genes_region$de_q == "non-significant"))
```

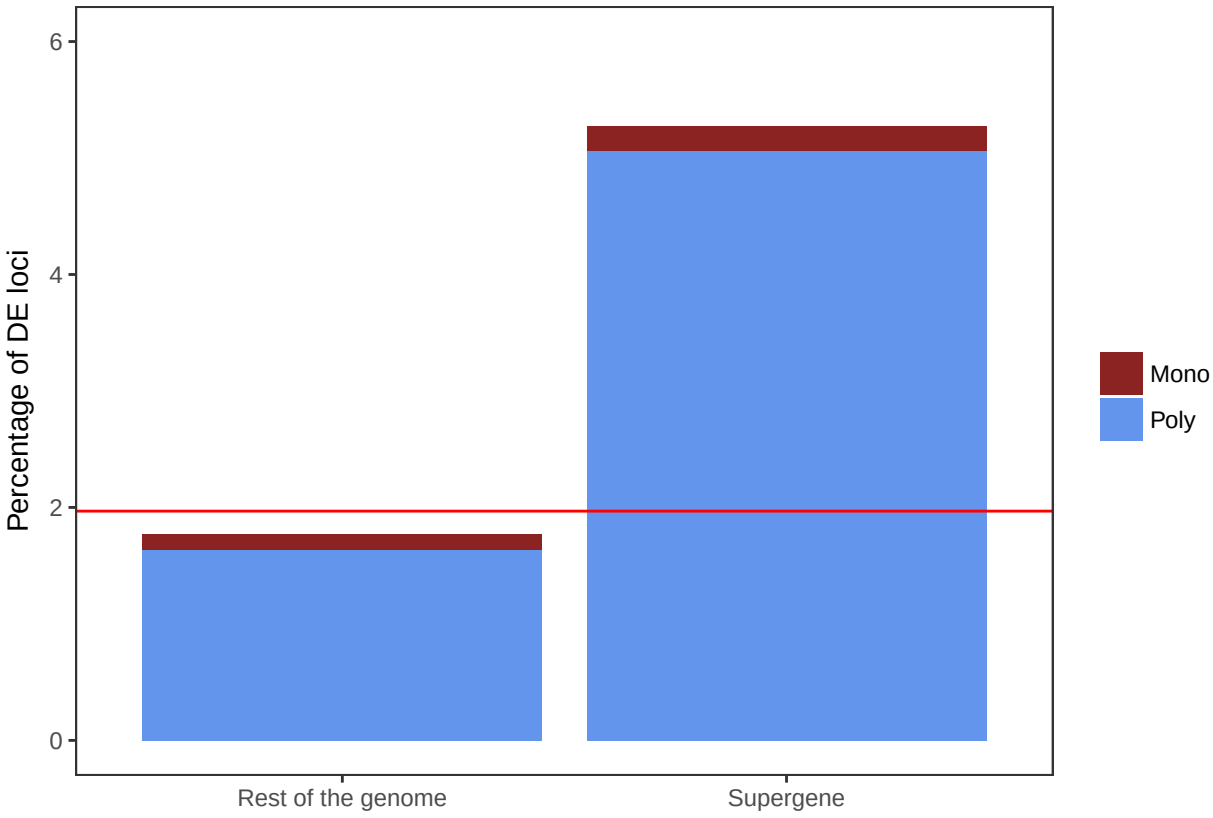
Generate a barplot showing both the distribution of socially biased loci within or outside the supergene region and the porportion of highly to lowly expressed loci in the different social forms

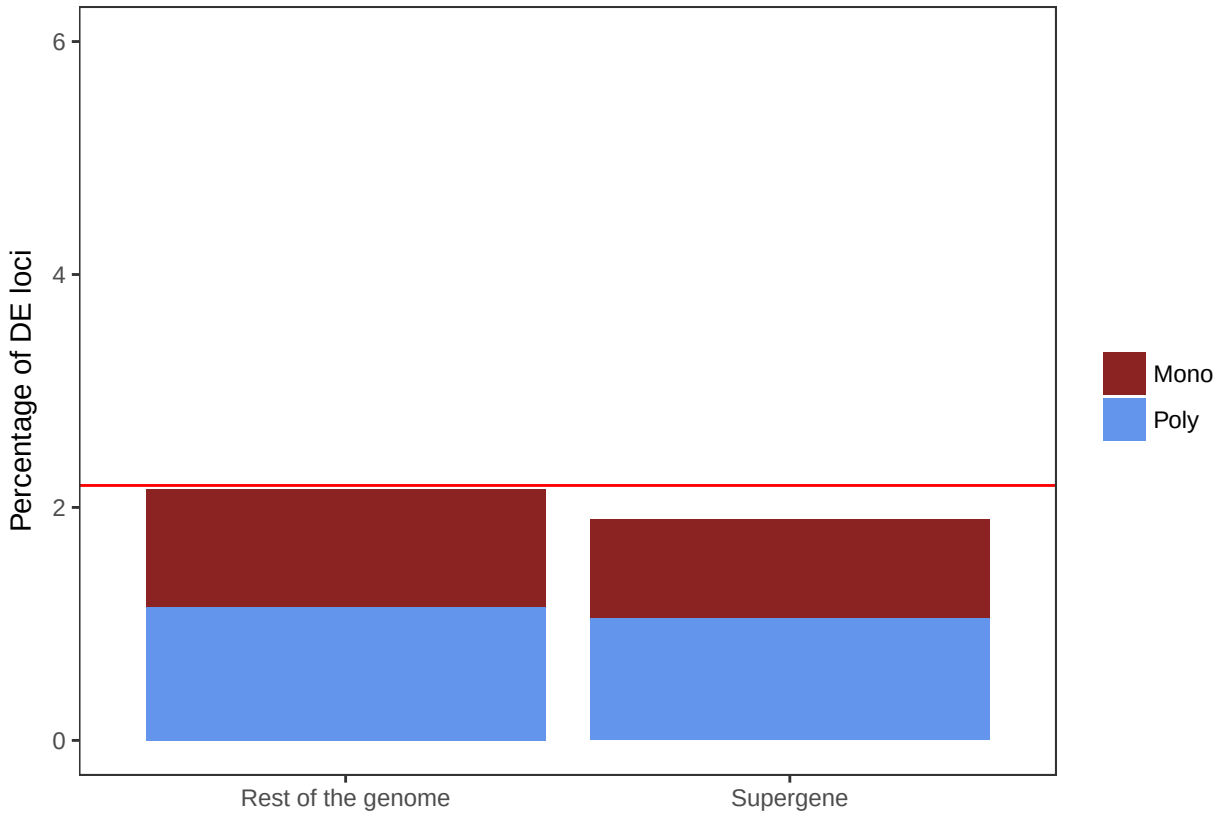
```
#####Barplot to show proportion of socially biased genes within and
# outside the supergene region ######


de_rec_m_q <- length(which(loc_genes_region$region == "recombining" &
                             loc_genes_region$de_q == "significant" &
                             loc_genes_region$expr_dir_q == "Mono"))

de_rec_p_q <- length(which(loc_genes_region$region == "recombining" &
                             loc_genes_region$de_q == "significant" &
                             loc_genes_region$expr_dir_q == "Poly"))

non_de_rec_m_q <- length(which(loc_genes_region$region == "recombining" &
                                 loc_genes_region$de_q == "non-significant" &
                                 loc_genes_region$expr_dir_q == "Mono"))

non_de_rec_p_q <- length(which(loc_genes_region$region == "recombining" &
                                 loc_genes_region$de_q == "non-significant" &
                                 loc_genes_region$expr_dir_q == "Poly"))

de_non_rec_m_q <- length(which(loc_genes_region$region == "supergene" &
                                 loc_genes_region$de_q == "significant" &
                                 loc_genes_region$expr_dir_q == "Mono"))

de_non_rec_p_q <- length(which(loc_genes_region$region == "supergene" &
                                 loc_genes_region$de_q == "significant" &
                                 loc_genes_region$expr_dir_q == "Poly"))

non_de_non_rec_m_q <- length(which(loc_genes_region$region == "supergene" &
```

```
                                    loc_genes_region$de_q == "non-significant" &
                                    loc_genes_region$expr_dir_q == "Mono"))

non_de_non_rec_p_q <- length(which(loc_genes_region$region == "supergene" &
                                    loc_genes_region$de_q == "non-significant" &
                                    loc_genes_region$expr_dir_q == "Poly"))

#Barplot for queens
de_percent_nrec_m_q <- de_non_rec_m_q / (non_de_non_rec_q + de_non_rec_q) * 100
de_percent_nrec_p_q <- de_non_rec_p_q / (non_de_non_rec_q + de_non_rec_q) * 100
de_percent_rec_m_q <- de_rec_m_q / (de_rec_q + non_de_rec_q) * 100
de_percent_rec_p_q <- de_rec_p_q / (de_rec_q + non_de_rec_q) * 100
expected_percent_q <- (de_non_rec_q + de_rec_q) / (non_de_non_rec_q + non_de_rec_q) * 100

to_plot_q           <- data.frame(c(de_percent_nrec_m_q, de_percent_nrec_p_q,
                                    de_percent_rec_m_q, de_percent_rec_p_q),
                                  rep(c("Supergene", "Rest of the genome"), each = 2),
                                  rep(c("Mono", "Poly"), n = 2))
colnames(to_plot_q) <- c("percentage", "region", "form")


p1 <- ggplot(data = to_plot_q, aes(x = region, y = percentage, fill = form)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = expected_percent_q, colour = "red") +
  theme_bw() + theme(panel.grid.minor = element_blank(), panel.grid.major = element_blank(),
                     legend.title = element_blank()) +
  scale_x_discrete(name = "") + scale_y_continuous(name = "Percentage of DE loci",
                                                   limits = c(0, 6)) +
  scale_fill_manual(values = c("brown4", "cornflowerblue"))


#Same plot, for workers
de_rec_m_w <- length(which(loc_genes_region$region == "recombining" &
                             loc_genes_region$de_w == "significant" &
                             loc_genes_region$expr_dir_w == "Mono"))

de_rec_p_w <- length(which(loc_genes_region$region == "recombining" &
                             loc_genes_region$de_w == "significant" &
                             loc_genes_region$expr_dir_w == "Poly"))

non_de_rec_m_w <- length(which(loc_genes_region$region == "recombining" &
                                 loc_genes_region$de_w == "non-significant" &
                                 loc_genes_region$expr_dir_w == "Mono"))

non_de_rec_p_w <- length(which(loc_genes_region$region == "recombining" &
                                 loc_genes_region$de_w == "non-significant" &
                                 loc_genes_region$expr_dir_w == "Poly"))

de_non_rec_m_w <- length(which(loc_genes_region$region == "supergene" &
                                 loc_genes_region$de_w == "significant" &
                                 loc_genes_region$expr_dir_w == "Mono"))

de_non_rec_p_w <- length(which(loc_genes_region$region == "supergene" &
```

```
                                  loc_genes_region$de_w == "significant" &
                                  loc_genes_region$expr_dir_w == "Poly"))

non_de_non_rec_m_w <- length(which(loc_genes_region$region == "supergene" &
                                     loc_genes_region$de_w == "non-significant" &
                                     loc_genes_region$expr_dir_w == "Mono"))

non_de_non_rec_p_w <- length(which(loc_genes_region$region == "supergene" &
                                     loc_genes_region$de_w == "non-significant" &
                                     loc_genes_region$expr_dir_w == "Poly"))

de_percent_nrec_m_w <- de_non_rec_m_w / (non_de_non_rec_w + de_non_rec_w) * 100
de_percent_nrec_p_w <- de_non_rec_p_w / (non_de_non_rec_w + de_non_rec_w) * 100
de_percent_rec_m_w <- de_rec_m_w / (de_rec_w + non_de_rec_w) * 100
de_percent_rec_p_w <- de_rec_p_w / (de_rec_w + non_de_rec_w) * 100
expected_percent_w <- (de_non_rec_w + de_rec_w) / (non_de_non_rec_w + non_de_rec_w) * 100

to_plot_w           <- data.frame(c(de_percent_nrec_m_w, de_percent_nrec_p_w,
                                    de_percent_rec_m_w, de_percent_rec_p_w),
                                  rep(c("Supergene", "Rest of the genome"), each = 2),
                                  rep(c("Mono", "Poly"), n = 2))
colnames(to_plot_w) <- c("percentage", "region", "form")


p2 <- ggplot(data = to_plot_w, aes(x = region, y = percentage, fill = form)) +
  geom_bar(stat = "identity") +
  geom_hline(yintercept = expected_percent_w, colour = "red") +
  theme_bw() + theme(panel.grid.minor = element_blank(), panel.grid.major = element_blank(),
                     legend.title = element_blank()) +
  scale_x_discrete(name = "") + scale_y_continuous(name = "Percentage of DE loci",
                                                   limits = c(0, 6)) +
  scale_fill_manual(values = c("brown4", "cornflowerblue"))
```

The red lines in the plots show the proportion of socially biased loci expected by randomness in workers or queens. The blue portion of each bar represents the proportion of those socially biased loci that are more highly expressed in polygyne individuals and vice-versa for the burgundy portion. Plot for queens


Percentage of DE loci
6
4
2
0
Rest of the genome
Supergene
Mono
Poly



Plot for workers

To check whether number of socially biased genes is different within and outside the supergene than what would be expected by randomness, the numbers are tested via a Chi squared test. A binomial test is run to check for differences between highly vs lowly expressed genes in polygyne/monogyne individuals. All analyses are performed in queens and workers independently.

Chi squared tests for queens:

```
to_test_q
```

```
##                     Recombinant Non recombinant
## Diff expressed               25             176
## Non diff expressed          449            9762
```

```
chisq.test(to_test_q)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  to_test_q
## X-squared = 27.508, df = 1, p-value = 1.565e-07
```

For workers:

```
to_test_w
```

```
##                     Recombinant Non recombinant
## Diff expressed                9             214
## Non diff expressed          465            9724
```

```r
chisq.test(to_test_w)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  to_test_w
## X-squared = 0.044823, df = 1, p-value = 0.8323
```

Binomial tests in queens:

```r
#In the genome
binom.test(x = de_rec_m_q, n = de_rec_m_q + de_rec_p_q, p = 0.5)
```

```
##
##  Exact binomial test
##
## data:  de_rec_m_q and de_rec_m_q + de_rec_p_q
## number of successes = 13, number of trials = 176, p-value <
## 2.2e-16
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.03991371 0.12299707
## sample estimates:
## probability of success
##             0.07386364
```

```r
#In the supergene
binom.test(x = de_non_rec_m_q, n = de_non_rec_m_q + de_non_rec_p_q, p = 0.5)
```

```
##
##  Exact binomial test
##
## data:  de_non_rec_m_q and de_non_rec_m_q + de_non_rec_p_q
## number of successes = 1, number of trials = 25, p-value = 1.55e-06
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.0010122 0.2035169
## sample estimates:
## probability of success
##                   0.04
```

and in workers:

```r
#In the genome
binom.test(x = de_rec_m_w, n = de_rec_m_w + de_rec_p_w, p = 0.5)
```

```
##
##  Exact binomial test
##
## data:  de_rec_m_w and de_rec_m_w + de_rec_p_w
## number of successes = 100, number of trials = 214, p-value =
## 0.3742
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.3989732 0.5365250
## sample estimates:
## probability of success
```

```
##              0.4672897
```

```
#In the supergene
binom.test(x = de_non_rec_m_w, n = de_non_rec_m_w + de_non_rec_p_w, p = 0.5)
```

```
##
##  Exact binomial test
##
## data:  de_non_rec_m_w and de_non_rec_m_w + de_non_rec_p_w
## number of successes = 4, number of trials = 9, p-value = 1
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.1369957 0.7879915
## sample estimates:
## probability of success
##              0.4444444
```