# Hot and Cold Thermal Challenges Astrangia poculata

Daniel Wuitchik

Here, we leverage the temperate stony coral, Astrangia poculata, which naturally exhibits a facultative symbiosis with Symbiodiniaceae, to explicitly examine how thermal challenges influence coral hosts in isolation from their symbionts. Aposymbiotic A. poculata were collected from Woods Hole, MA, the northern range limit for this species. Corals were thermally challenged in two independent common garden experiments (Heat challenge: 31C, 10 days; Cold challenge: 6C, 16 days) to determine the effects of divergent thermal stressors. Behavioural responses to food stimuli were monitored throughout the thermal challenges and genome-wide gene expression profiling (TagSeq) was used to characterize molecular underpinnings of the coral's response to stress in its aposymbiotic state.

## Contents

# Libraries

```
library(plyr)
library(tidyverse)
library(DESeq2)
library(ggplot2)
library(affycoretools)
library(arrayQualityMetrics)
library(genefilter)
library(DESeq)
library(cowplot)
library(readr)
library(RColorBrewer)
library(gplots)
library(knitr)
library(plotly)
library(vegan)
library(kableExtra)
library(reshape2)
library(prettydoc)
library(VennDiagram)
library(MASS)
library(ggrepel)
library(stringr)
```

# Behavioural Analysis

Coral polyp behaviours were observed every 3-4 days throughout the experiment. The total surface area of the coral that had extended vs retracted polyps was estimated and then scored between 1-5 based on polyp activity

| Score | Percent.of.colony.with.extended.polyps |
|---|---|
| 1 | 0 |
| 2 | 25 |
| 3 | 50 |
| 4 | 75 |
| 5 | 100 |

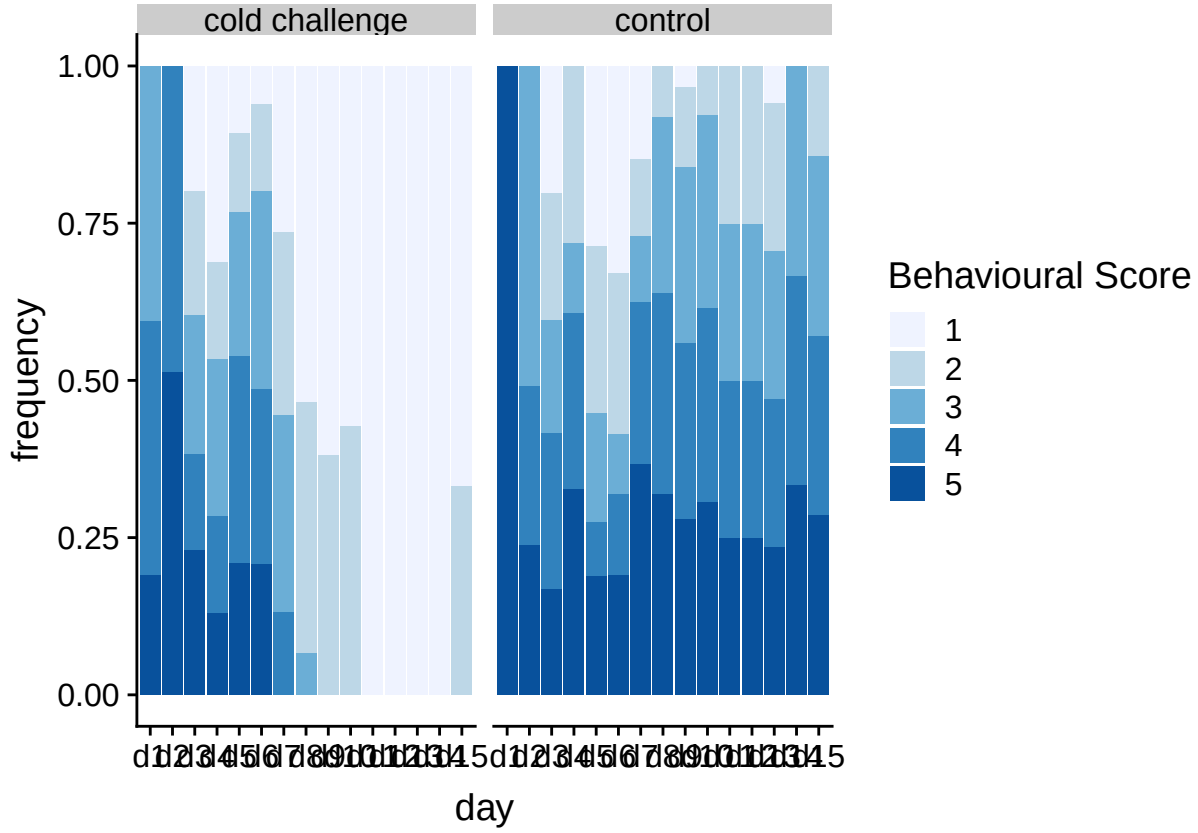
## Cold

Read in data, transform to long form and organize.

```
cold_behaviour = read.csv("cold_behaviour.csv") %>%
  melt() %>%
  rename(day = variable) %>%
  rename(polyp_behaviour = value) %>%
  rename(treatment = group)
```

This is how we plotted the behavioural figure in the manuscript.

```
cold_stacked = cold_behaviour %>%
  group_by(polyp_behaviour, day, treatment) %>%
  summarise(n = n()) %>%
  mutate(frequency = n / sum(n))

ggplot(cold_stacked, aes(y = frequency, x = day, fill = as.factor(polyp_behaviour))) +
  geom_bar(stat = "identity", position = "fill") +
  facet_grid(. ~ treatment) +
  scale_fill_brewer(palette = "Blues", direction=1) +
  labs(fill = "Behavioural Score") +
  theme_cowplot()
```




Now to determine if there is any statistical differences between treatment groups.

```
cold_model = polr(as.factor(polyp_behaviour) ~ treatment + day, data = cold_behaviour)
summary(cold_model)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = as.factor(polyp_behaviour) ~ treatment + day,
##     data = cold_behaviour)
##
## Coefficients:
##                    Value Std. Error t value
## treatmentcontrol  2.405     0.1829  13.149
## dayd2            -1.594     0.8460  -1.885
```

```
## dayd3            -4.218     0.7989  -5.280
## dayd4            -3.094     0.8088  -3.825
## dayd5            -4.697     0.8054  -5.832
## dayd6            -5.269     0.8070  -6.529
## dayd7            -5.028     0.7933  -6.338
## dayd8            -5.587     0.7960  -7.019
## dayd9            -6.050     0.8061  -7.505
## dayd10           -5.402     0.8020  -6.736
## dayd11           -5.893     0.8065  -7.307
## dayd12           -5.815     0.8063  -7.212
## dayd13           -7.392     0.8414  -8.785
## dayd14           -5.666     0.8044  -7.043
## dayd15           -5.560     0.8068  -6.892
##
## Intercepts:
##     Value   Std. Error t value
## 1|2 -5.2553  0.7513    -6.9950
## 2|3 -4.2170  0.7459    -5.6533
## 3|4 -3.3799  0.7422    -4.5542
## 4|5 -2.3505  0.7365    -3.1914
##
## Residual Deviance: 1488.85
## AIC: 1526.85
```

```
cold_table = coef(summary(cold_model))
```

```
##
## Re-fitting to get Hessian
```

```
cold_p = pnorm(abs(cold_table[,"t value"]), lower.tail = FALSE) * 2
cold_p
```

```
## treatmentcontrol            dayd2            dayd3            dayd4
##     1.730361e-39     5.949383e-02     1.291971e-07     1.305300e-04
##            dayd5            dayd6            dayd7            dayd8
##     5.490019e-09     6.632830e-11     2.333955e-10     2.234788e-12
##            dayd9           dayd10           dayd11           dayd12
##     6.121332e-14     1.626206e-11     2.722466e-13     5.533555e-13
##           dayd13           dayd14           dayd15              1|2
##     1.565671e-18     1.877490e-12     5.515670e-12     2.651799e-12
##              2|3              3|4              4|5
##     1.574125e-08     5.259516e-06     1.415736e-03
```

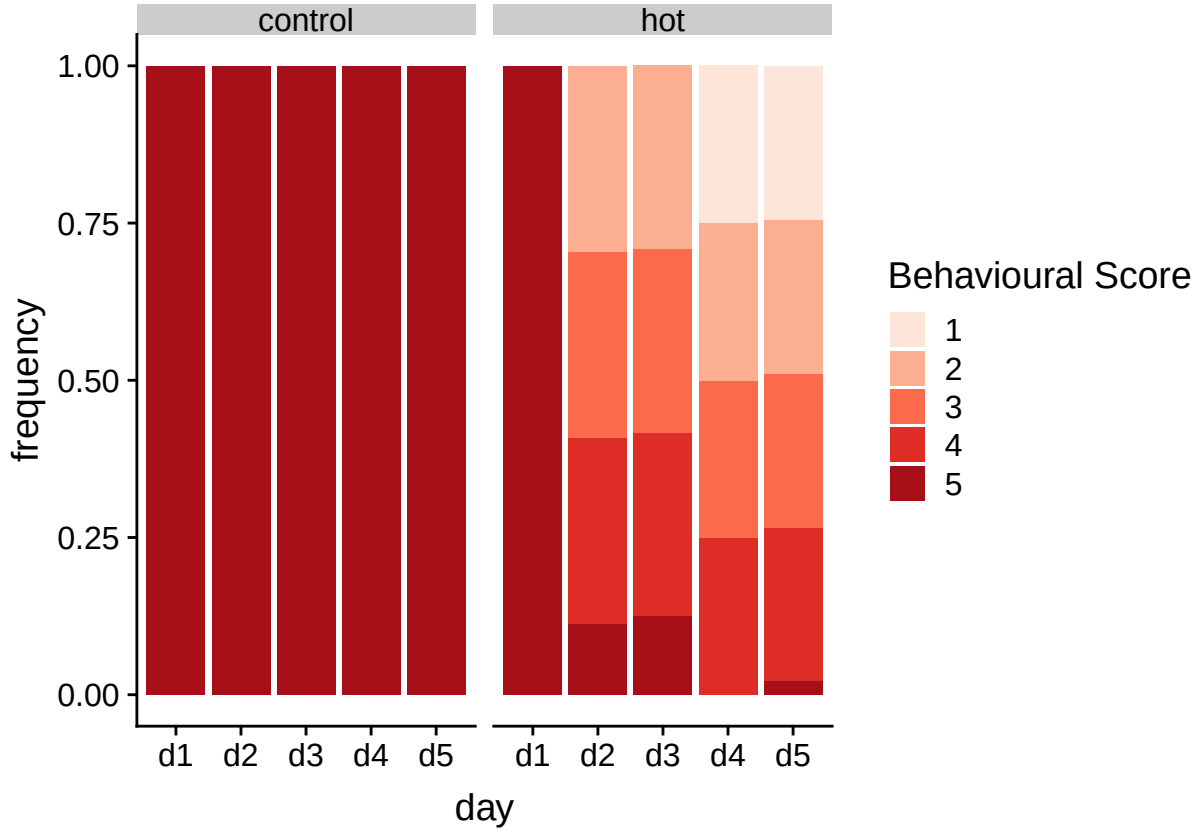
## Hot

Read in data, transform to long form and organize.

```
hot_behaviour = read.csv("hot_behaviour.csv") %>%
  melt() %>%
  rename(day = variable) %>%
  rename(polyp_behaviour = value) %>%
  mutate(genotype = sapply(strsplit(as.character(individual), split = ""), "[[", 2))

hot_behaviour$day = as.character(hot_behaviour$day)
hot_data = hot_behaviour %>%
  mutate(temperature = recode(day, d1 = "16", d2 = "23", d3 = "25", d4 = "28", d5 = "31"))
```

Displaying the data as a proportion of overall score with given phenotypes.

```
hot_stacked = hot_behaviour %>%
  group_by(polyp_behaviour, day, treatment) %>%
  summarise(n = n()) %>%
  mutate(frequency = n / sum(n))

ggplot(hot_stacked, aes(y = frequency, x = day, fill = as.factor(polyp_behaviour))) +
  geom_bar(stat = "identity", position = "fill") +
  facet_grid(. ~ treatment) +
  scale_fill_brewer(palette = "Reds", direction=1) +
  labs(fill = "Behavioural Score") +
  theme_cowplot()
```




Behavioural scores after feeding. The total surface area of the coral with extended polyps was estimated and then scored between 1-5 based on polyp activity

```
hot_model = polr(as.factor(polyp_behaviour) ~ treatment + day, data = hot_behaviour)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(hot_model)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = as.factor(polyp_behaviour) ~ treatment + day,
##     data = hot_behaviour)
##
```

```
## Coefficients:
##                Value Std. Error t value
## treatmenthot -19.22     0.1819 -105.69
## dayd2        -16.93     0.4107  -41.21
## dayd3        -16.43     0.4671  -35.17
## dayd4        -19.93     0.3726  -53.48
## dayd5        -19.59     0.3611  -54.26
##
## Intercepts:
##     Value     Std. Error t value
## 1|2  -40.3389    0.1819  -221.8048
## 2|3  -38.8912    0.2566  -151.5488
## 3|4  -37.8752    0.2909  -130.2071
## 4|5  -36.6582    0.3596  -101.9315
##
## Residual Deviance: 209.8631
## AIC: 227.8631
```

```
hot_table = coef(summary(hot_model))
```

```
##
## Re-fitting to get Hessian
```

```
hot_p = pnorm(abs(hot_table[,"t value"]), lower.tail = FALSE) * 2
hot_p
```

```
##  treatmenthot         dayd2         dayd3         dayd4         dayd5
##  0.000000e+00  0.000000e+00 5.902959e-271  0.000000e+00  0.000000e+00
##           1|2           2|3           3|4           4|5
##  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
```

## Data Cleanup

Here we want to filter out contaminant reads. These could be from various taxa, and so we first get NCBI taxonomic info

```
wget https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/new_taxdump/new_taxdump.tar.gz
tar -zxvf new_taxdump.tar.gz
```

Set up rank lineage file

```
tax <- read_tsv("rankedlineage.dmp",
                col_names = c("id", "name", "s", "g", "f", "o", "c","p", "k", "d"),
                col_types=("c-c-c-c-c-c-c-c-c-c-"))
```

From this I can make a list of potential contaminants to remove called filter_list

```
filter_list = tax %>%
  filter(k == "Archaea" | k == "Bacteria" | k =="Plantae"| k == "Protozoa" | k == "Chromista" | k == "Fu
  dplyr::select(id) %>%
  rename(speciesID = id) %>%
  as.data.frame()
```

Compare this with the iso2gene. First I have to break apart the delimiters to get the taxonomic id's into their own column. *136 genes filtered out with this method

```
iso2gene <- read_tsv("astrangia_iso2gene.tab")
```

```
iso2species = iso2gene %>%
  mutate(speciesID = gsub(".* OX=", "", Gene)) %>%      # Remove everything before OX=
  mutate(speciesID = gsub(" .*", "", speciesID)) %>%    # Remove everything after species ID
  filter(speciesID %in% filter_list$speciesID)         # Filter to only keep species ID that are in the

dirty = iso2species %>%
  inner_join(filter_list) %>%
  dplyr::select(Iso)

dirty_list = dirty %>%
  inner_join(iso2gene)

write.csv(dirty_list, "dirty.csv")
```

Now we can filter out the contamination

```
cold_counts = read.csv("cold_raw_assembled_transcriptome.csv") %>%
  rename(Iso = X) %>%
  filter(!Iso %in% dirty$Iso) %>%
  column_to_rownames(var = "Iso")
hot_counts = read.csv("hot_raw_counts_assembled_transcriptome.csv") %>%
  rename(Iso = X) %>%
  filter(!Iso %in% dirty$Iso) %>%
  column_to_rownames(var = "Iso")
```

# Outlier Detection

I utilized old school DESeq to look for any samples that didn't sequence properly.

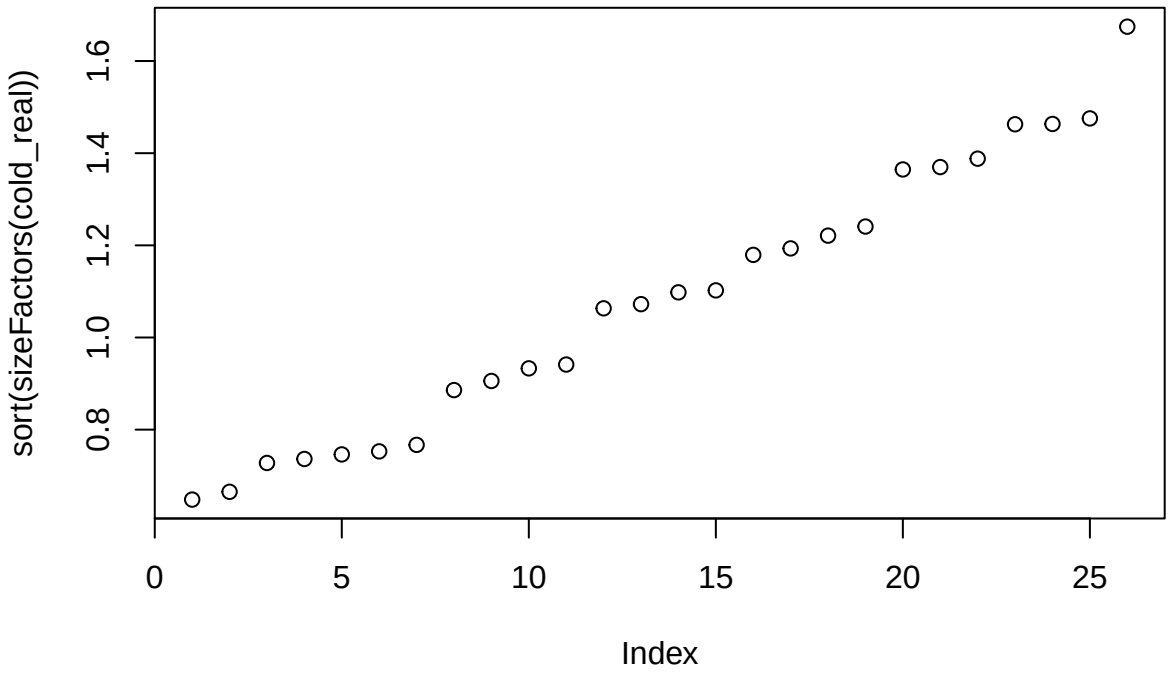
## Cold

First I set up experimental designs

```
cold_treatment = as.factor(sapply(strsplit(colnames(cold_counts), split = ""), "[[", 1)) %>%
            revalue(c("C" = "control", "F" = "cold"))
cold_genotype  = as.factor(sapply(strsplit(colnames(cold_counts), split = ""), "[[", 2))

cold_expDesign = data.frame(cold_treatment, cold_genotype)
            cold_expDesign$sample = colnames(cold_counts)
            write.csv(cold_expDesign, "cold_expDesign.csv", row.names = F)
```

Plotting the library size factors

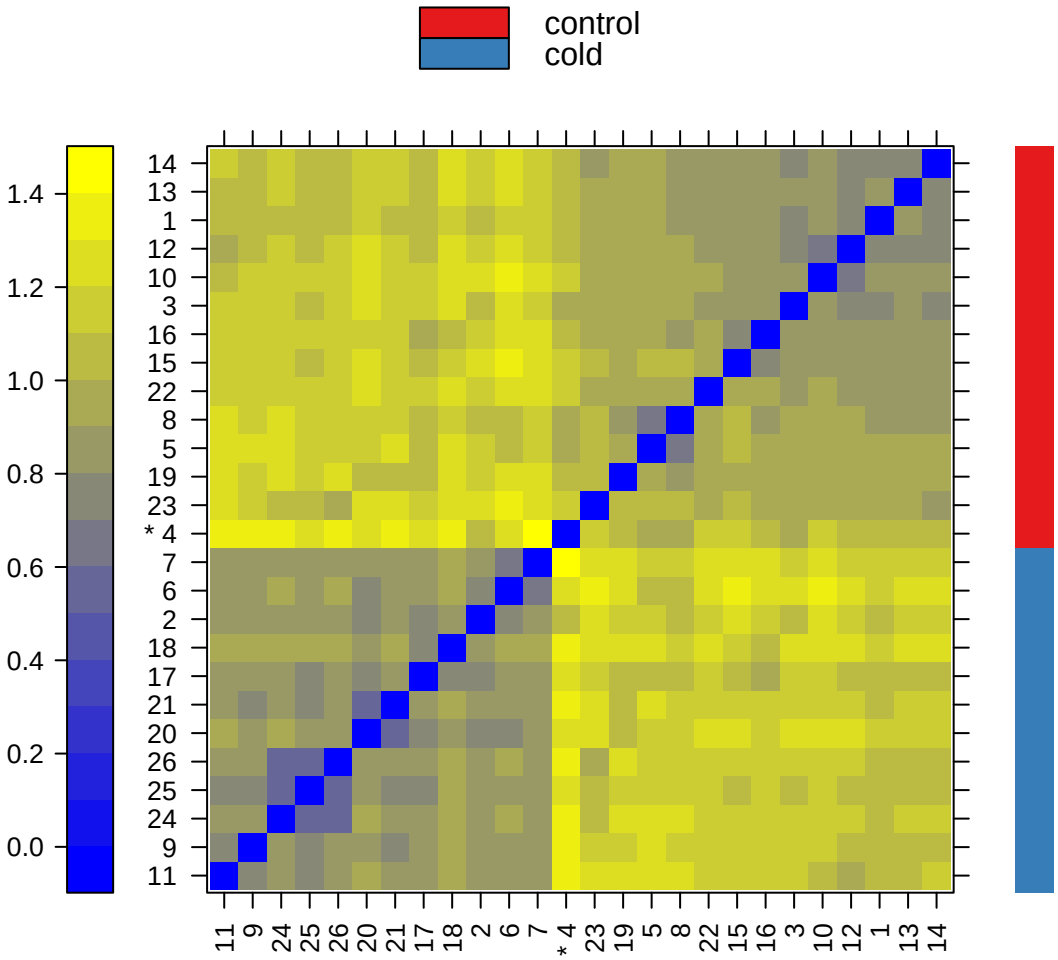
```
cold_real=newCountDataSet(cold_counts,cold_expDesign)
cold_real=estimateSizeFactors(cold_real)
plot(sort(sizeFactors(cold_real)))
```



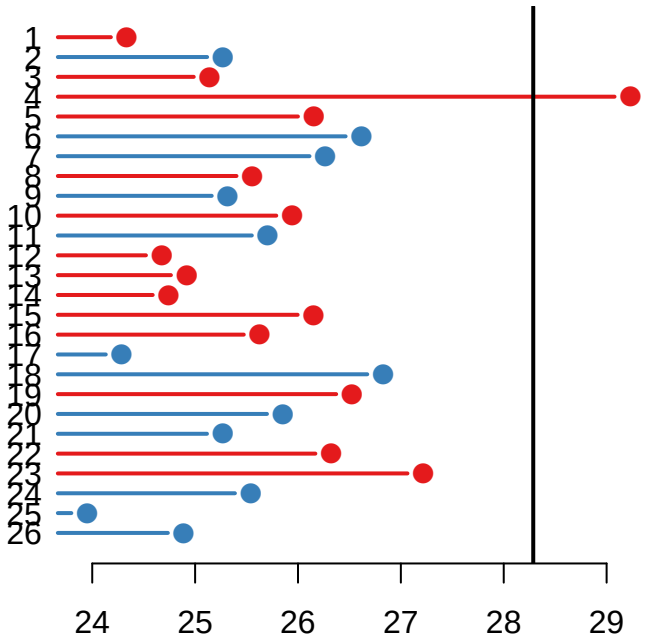

Outliers - here you have to manually inspect the html output.

```
cold_cds=estimateDispersions(cold_real,method="blind")
cold_vsdBlind=DESeq::varianceStabilizingTransformation(cold_cds)
arrayQualityMetrics(cold_vsdBlind,intgroup=c("cold_treatment"), force=TRUE, outdir = "cold_arrayQualityl
```

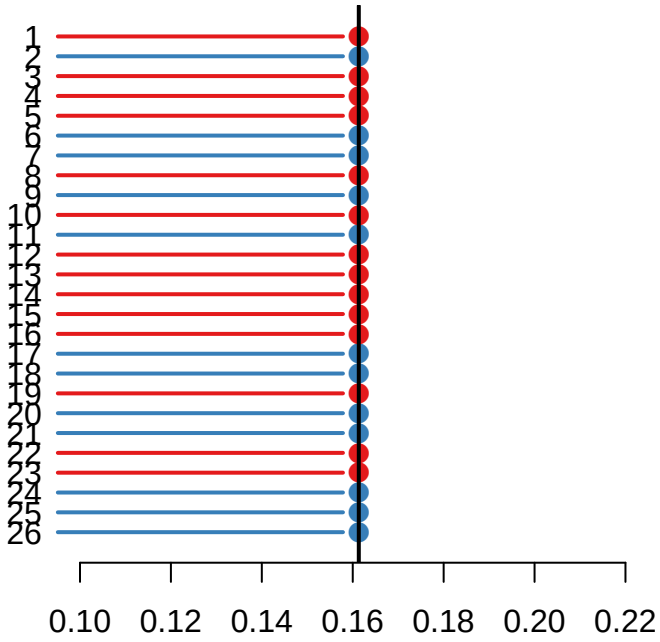
We can see the cold and control group nicely based on distances between arrays

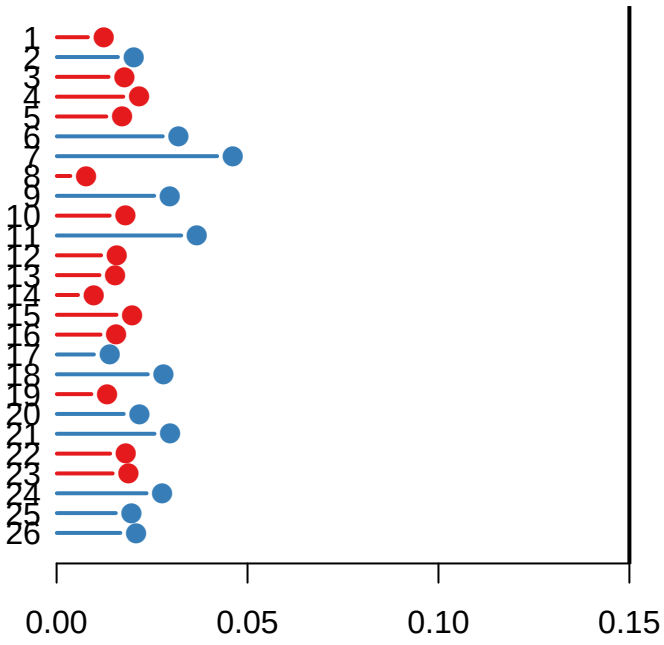



That said, there was one outlier detected based on the previous figures. The bars are shown in the original order of the arrays. Based on the distribution of the values across all arrays, a threshold of 28.3 was determined, which is indicated by the vertical line. One array exceeded the threshold and was considered an outlier. We can see the cold and control group nicely based on distances between arrays




Despite this showing an outlier, all other outlier tests tid not suggest that this sample was an outlier.






So, this sample was not deemed an outlier and it was kept in for the remainder of the analysis.

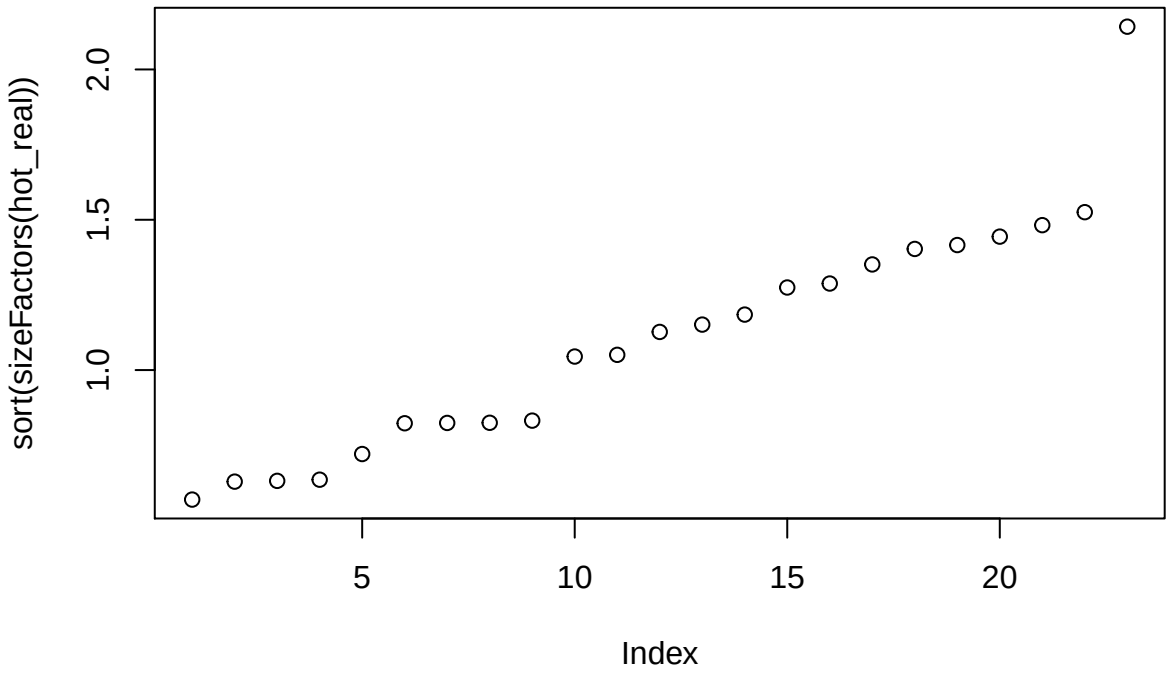
## Hot

First I set up experimental designs

```
hot_treatment = as.factor(sapply(strsplit(colnames(hot_counts), split = ""), "[[", 1)) %>%
            revalue(c("C" = "control", "H" = "hot"))
hot_genotype  = as.factor(sapply(strsplit(colnames(hot_counts), split = ""), "[[", 2))

hot_expDesign = data.frame(hot_treatment, hot_genotype)
            hot_expDesign$sample = colnames(hot_counts)
            write.csv(hot_expDesign, "hot_expDesign.csv", row.names = F)
```

Plotting the library size factors

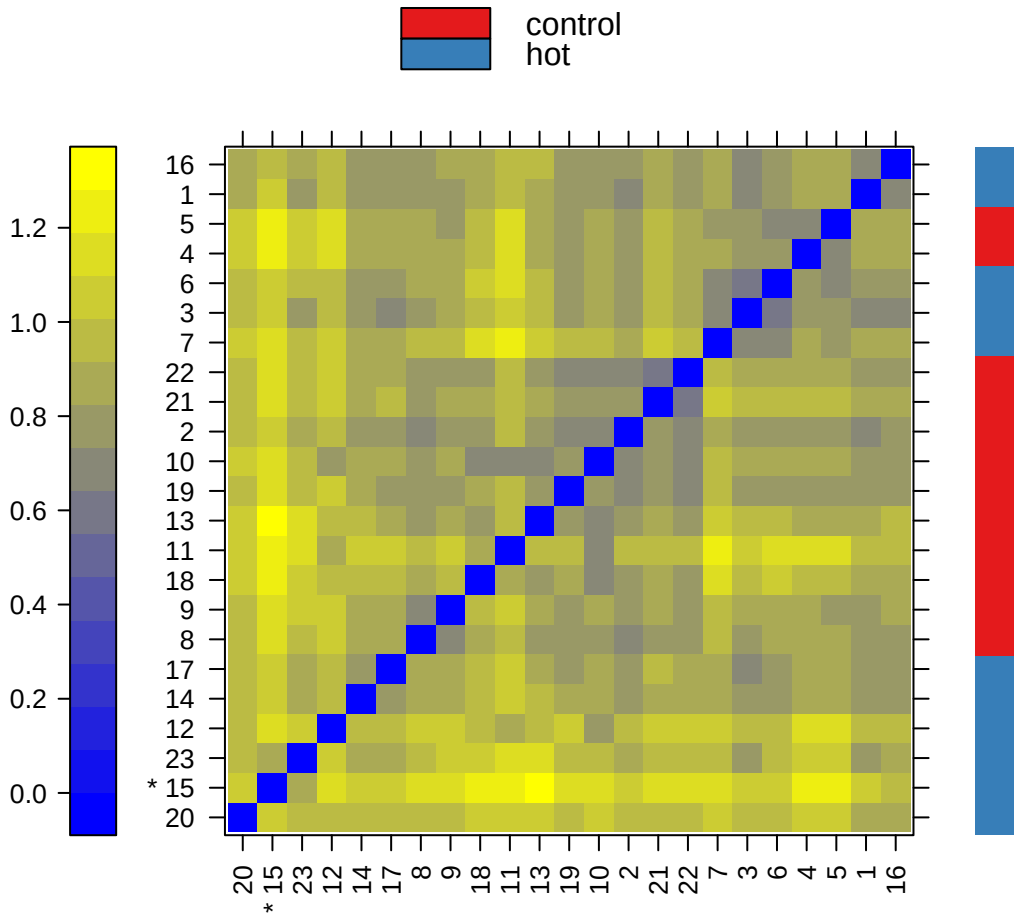
```
hot_real=newCountDataSet(hot_counts,hot_expDesign)
hot_real=estimateSizeFactors(hot_real)
plot(sort(sizeFactors(hot_real)))
```



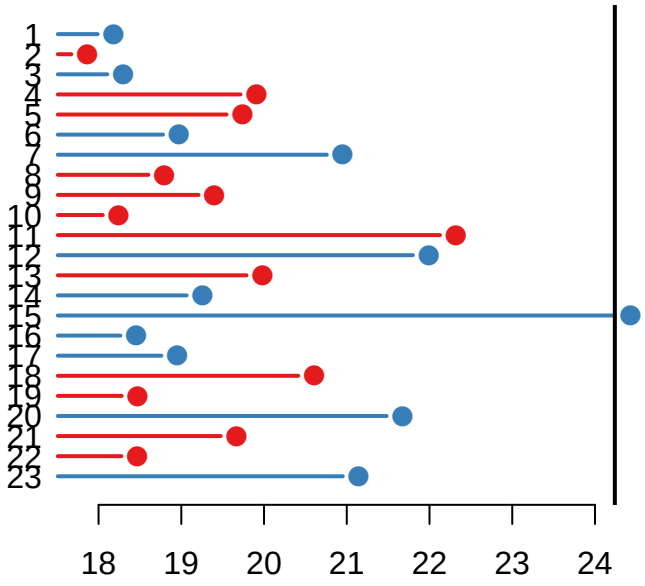

Outliers - here you have to manually inspect the html output.

```
hot_cds=estimateDispersions(hot_real,method="blind")
hot_vsdBlind=DESeq::varianceStabilizingTransformation(hot_cds)
arrayQualityMetrics(hot_vsdBlind,intgroup=c("hot_treatment"), force=TRUE, outdir = "hot_arrayQualityMet
```

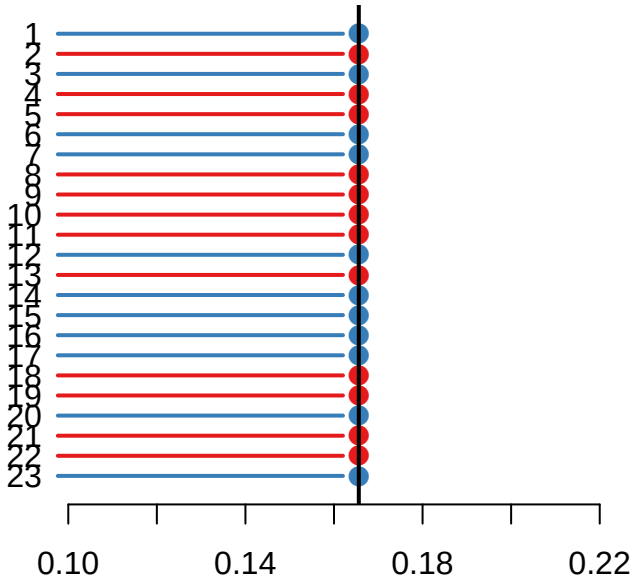
We can see the hot and control group mostly together, however not as strong a discrimination as the cold experiment based on distances between arrays

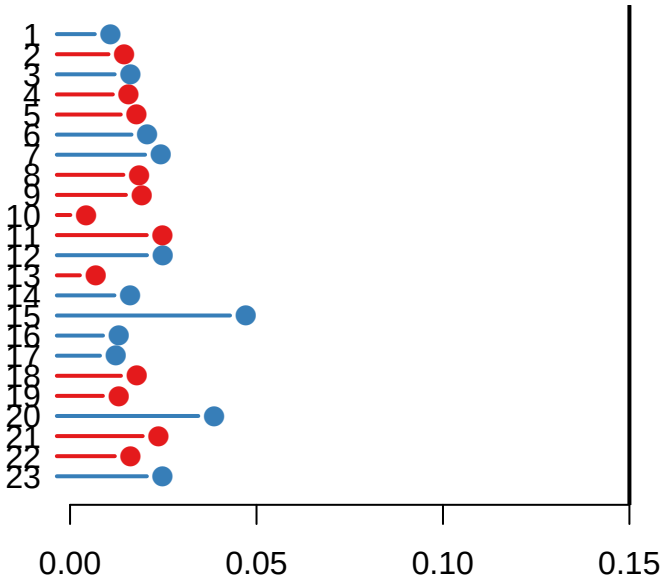



Much like in the cold, there was one sample that failed a single outlier quality metric. The bars are shown in the original order of the arrays. Based on the distribution of the values across all arrays, a threshold of 24.6 was determined, which is indicated by the vertical line. One array exceeded the threshold and was considered an outlier.




Despite this showing an outlier, all other outlier tests tid not suggest that this sample was an outlier.






All samples were kept for differential expression analysis.

# Differential Expression

## DESeq2 model

Cold

```
cold_dds = DESeqDataSetFromMatrix(countData = cold_counts, colData = cold_expDesign, design = ~ cold_ge
cold_dds = DESeq(cold_dds)
cold_rlogged = DESeq2::rlog(cold_dds, blind = TRUE) #for use later on
write.csv(assay(cold_rlogged), "cold_rlogged.csv")
cold_results = results(cold_dds, contrast = c("cold_treatment", "cold", "control"))
summary(cold_results)
```

```
##
## out of 13108 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 2569, 20%
## LFC < 0 (down)     : 3544, 27%
## outliers [1]       : 1, 0.0076%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
write.csv(cold_results, "cold_results.csv", row.names = TRUE)
```

Hot

```
hot_dds = DESeqDataSetFromMatrix(countData = hot_counts, colData = hot_expDesign, design = ~ hot_genoty
hot_dds = DESeq(hot_dds)
hot_rlogged = rlogTransformation(hot_dds, blind = TRUE)
write.csv(assay(hot_rlogged), "hot_rlogged.csv")
hot_results = results(hot_dds, contrast = c("hot_treatment", "hot", "control"))
```

```r
summary(hot_results)
```

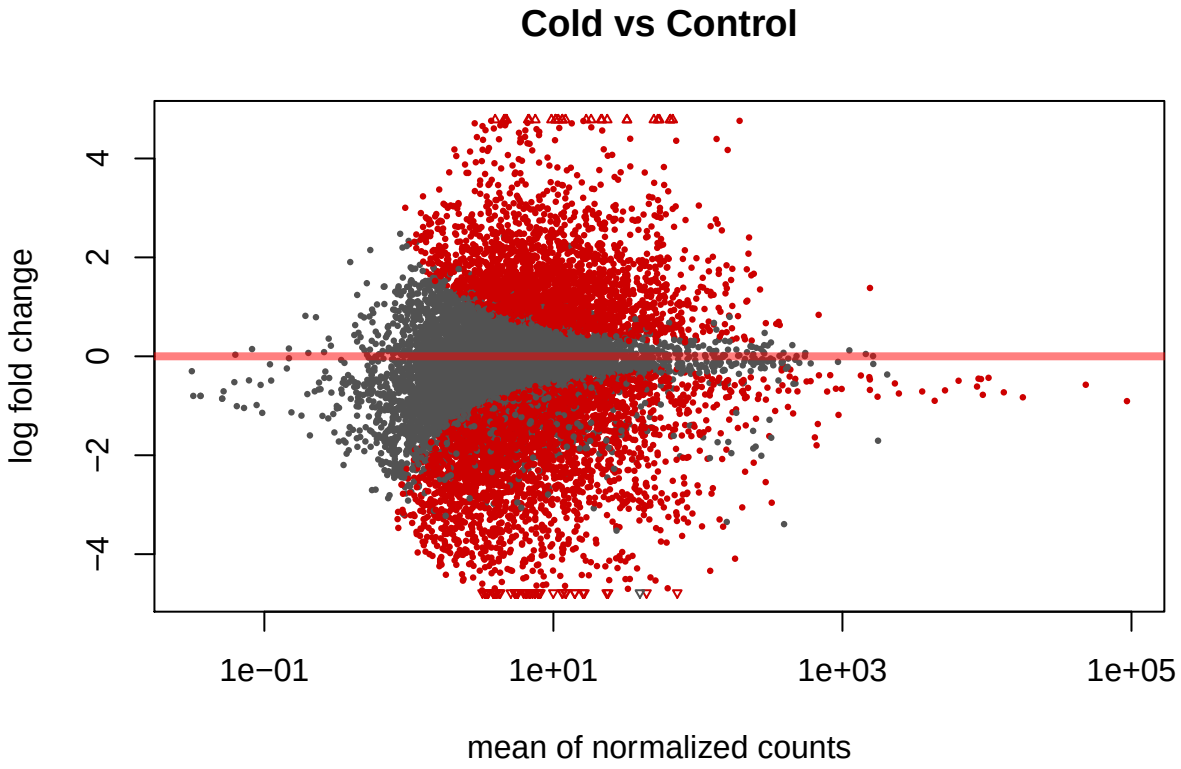
```
##
## out of 13109 with nonzero total read count
## adjusted p-value < 0.1
## LFC > 0 (up)       : 574, 4.4%
## LFC < 0 (down)     : 880, 6.7%
## outliers [1]       : 10, 0.076%
## low counts [2]     : 3050, 23%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```r
write.csv(hot_results, "hot_results.csv", row.names = TRUE)
```

### Volcano Plots

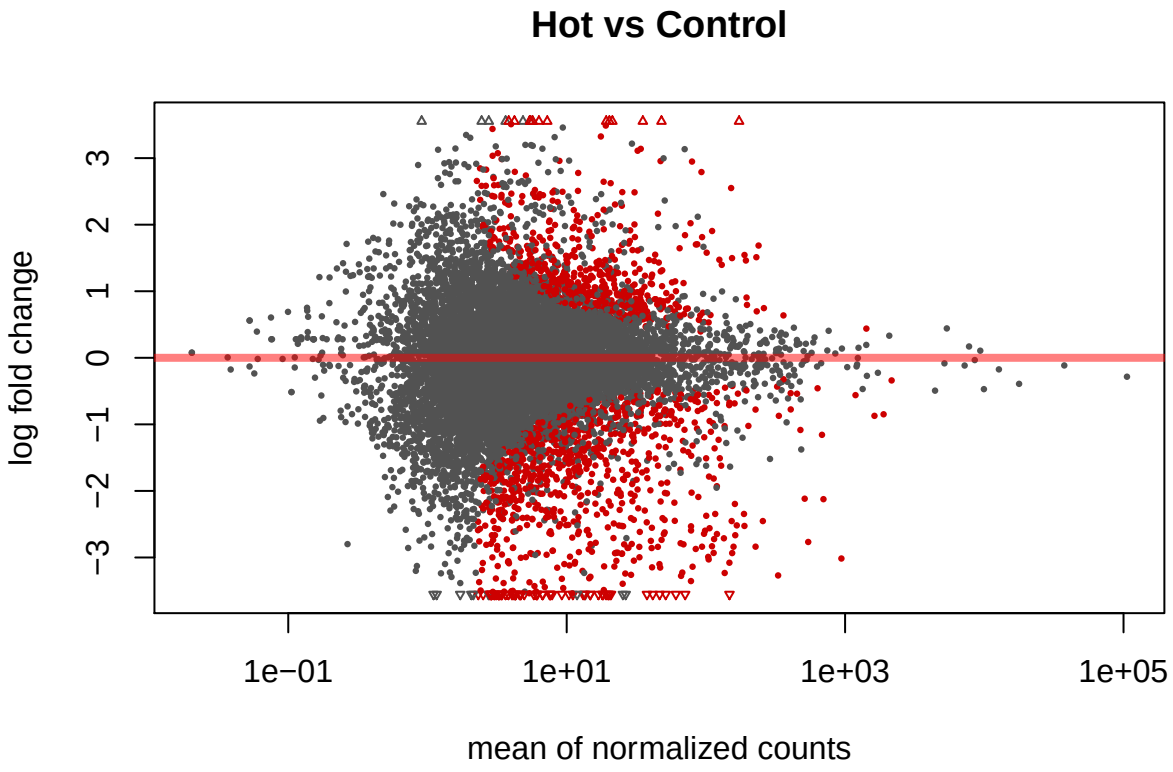
Cold

```r
DESeq2::plotMA(cold_results, main = "Cold vs Control")
```




Hot

```r
DESeq2::plotMA(hot_results, main = "Hot vs Control")
```



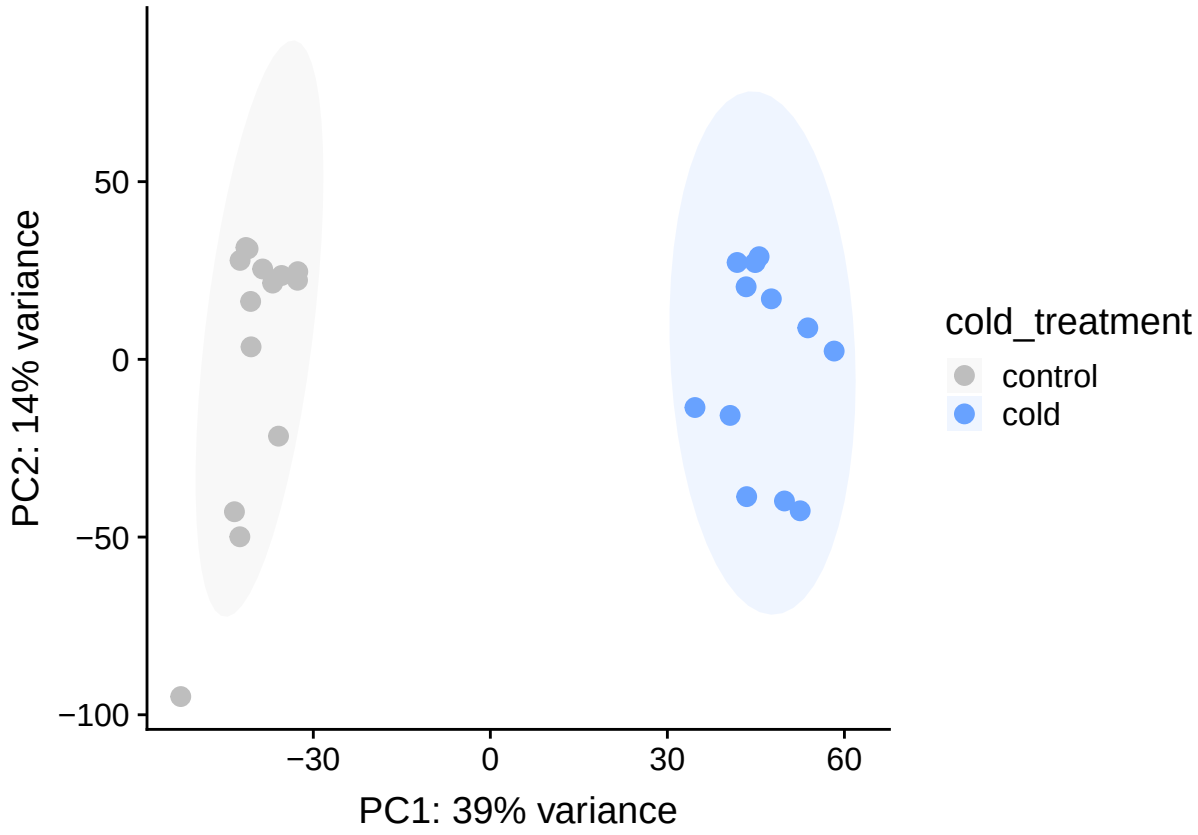

## PCAs

### Cold

```
cold_pcadata = DESeq2::plotPCA(cold_rlogged, intgroup = c( "cold_treatment", "cold_genotype"), returnDa
cold_percentVar = round(100 * attr(cold_pcadata, "percentVar"))
cold_pca = prcomp(t(assay(cold_rlogged)), center = TRUE, scale. = FALSE)

PCA_cold = as.data.frame(cold_pca$x)%>%
  dplyr::select(PC1, PC2) %>%
  rownames_to_column("sample") %>%
  left_join(cold_expDesign)
```

```
## Joining, by = "sample"
```

```
cold_cols = c("control" = "grey", "cold" = "#68a2ff")
ggplot(PCA_cold, aes(PC1, PC2)) +
  geom_point(aes(colour = cold_treatment), size = 3) +
  stat_ellipse(geom = "polygon", alpha = 1/10, aes(fill = cold_treatment)) +
  scale_colour_manual(values = cold_cols) +
  scale_fill_manual(values = cold_cols) +
  xlab(paste0("PC1: ",cold_percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",cold_percentVar[2],"% variance"))
```




Significance

```
adonis(cold_pca$x ~ cold_treatment + cold_genotype, method = 'eu')
```

```
##
## Call:
## adonis(formula = cold_pca$x ~ cold_treatment + cold_genotype,      method = "eu")
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##                Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## cold_treatment  1     48062   48062 12.3591 0.25605  0.001 ***
## cold_genotype   8     77423    9678  2.4887 0.41247  0.001 ***
## Residuals      16     62221    3889         0.33148
## Total          25    187706                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

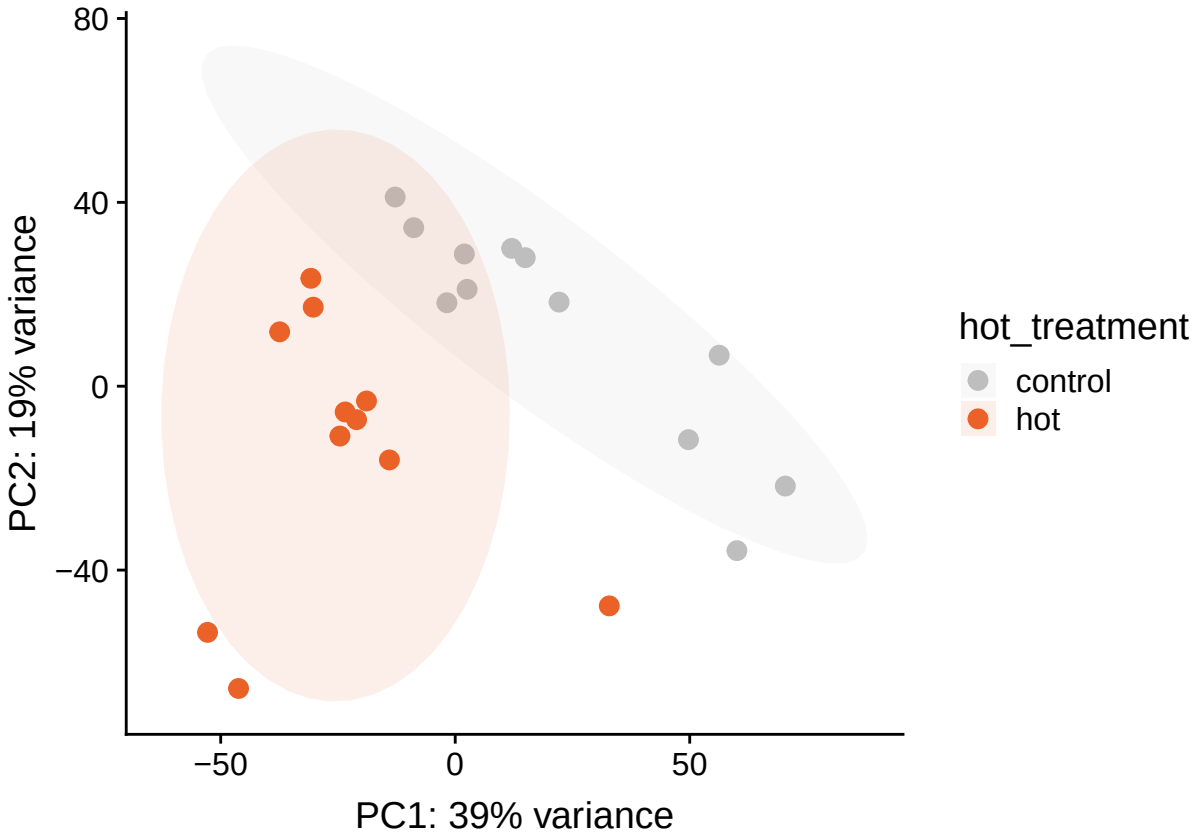
**Hot**

```
hot_pcadata = DESeq2::plotPCA(hot_rlogged, intgroup = c( "hot_treatment", "hot_genotype"), returnData =
hot_percentVar = round(100 * attr(hot_pcadata, "percentVar"))
hot_pca = prcomp(t(assay(hot_rlogged)), center = TRUE, scale. = FALSE)
PCA_hot = as.data.frame(hot_pca$x)%>%
  dplyr::select(PC1, PC2) %>%
```

```
  rownames_to_column("sample") %>%
  left_join(hot_expDesign)
```

```
## Joining, by = "sample"
```

```
hot_cols = c("control" = "grey", "hot" = "#ea6227")
ggplot(PCA_hot, aes(PC1, PC2)) +
  geom_point(aes(colour = hot_treatment), size = 3) +
  stat_ellipse(geom = "polygon", alpha = 1/10, aes(fill = hot_treatment)) +
  scale_colour_manual(values = hot_cols) +
  scale_fill_manual(values = hot_cols) +
  xlab(paste0("PC1: ",hot_percentVar[1],"% variance")) +
  ylab(paste0("PC2: ",hot_percentVar[2],"% variance"))
```




Significance

```
adonis(hot_pca$x ~ hot_treatment + hot_genotype, method = 'eu')
```

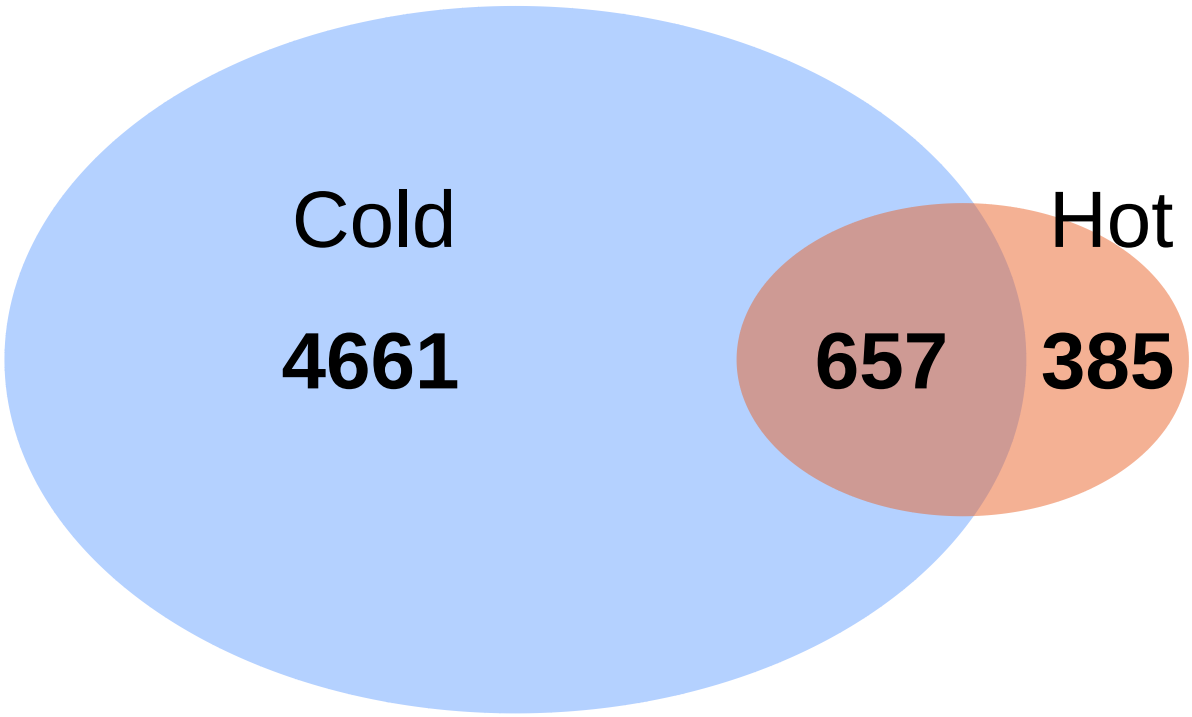
```
##
## Call:
## adonis(formula = hot_pca$x ~ hot_treatment + hot_genotype, method = "eu")
##
## Permutation: free
## Number of permutations: 999
##
## Terms added sequentially (first to last)
##
##               Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## hot_treatment  1     19385 19385.3  4.5248 0.13748  0.001 ***
```

```
## hot_genotype   8     65922  8240.3  1.9234 0.46753  0.001 ***
## Residuals     13     55695  4284.2         0.39499
## Total         22    141002                 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Shared Response

```
hot = read.csv("hot_results.csv", row.names = 1)
hot = row.names(hot[hot$padj<0.05 & !is.na(hot$padj),])
cold = read.csv("cold_results.csv", row.names = 1)
cold = row.names(cold[cold$padj<0.05 & !is.na(cold$padj),])
```

```
all_shared = list("Hot" = hot, "Cold" = cold)
    prettyvenn=venn.diagram(
      x = all_shared,
      filename=NULL,
      col = "transparent",
      fill = c("#ea6227", "#68a2ff"),
      alpha = 0.5,
      # label.col = c("darkred", "white", "darkgreen", "white", "white", "white", "blue4"),
      cex = 2.5,
      fontfamily = "sans",
      fontface = "bold",
      cat.default.pos = "text",
      cat.col = "black",
      cat.cex = 2.5,
      cat.fontfamily = "sans",
      cat.dist = c(0.08, 0.08),
      cat.pos = 1
    );
grid.draw(prettyvenn)
```




Hypergeometric test

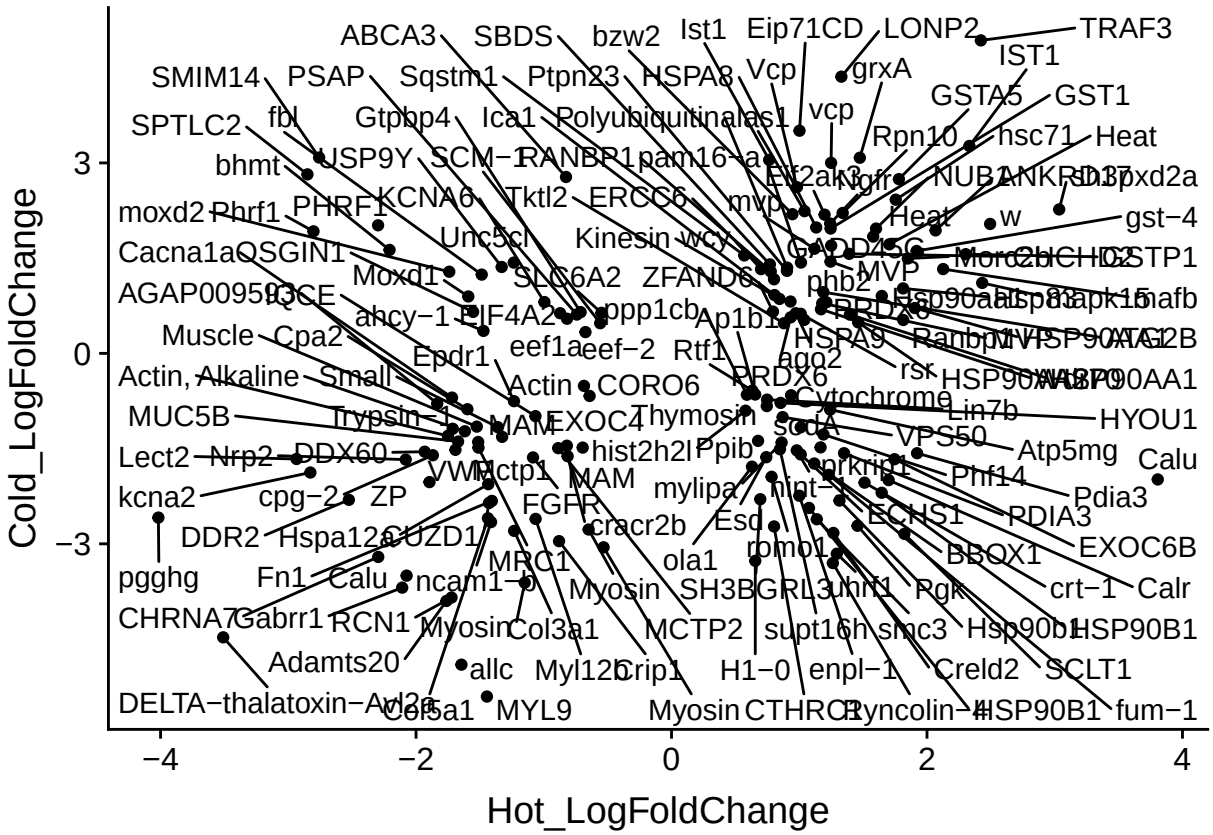
```
a = read.csv("hot_results.csv")
h = read.csv("hot_results.csv") %>%
  filter(padj < 0.05)

c = read.csv("cold_results.csv") %>%
  filter(padj < 0.05)

overlap = inner_join(h, c, by = "X")

phyper((nrow(overlap)-1), nrow(h), (nrow(a)-nrow(h)), nrow(c), lower.tail = F, log.p = FALSE)
```

```
## [1] 1.075883e-52
```

```
gene = read.delim("astrangia_iso2gene.tab", sep = "\t") %>%
  mutate(gene_symbol = gsub(".* GN=", "", Gene)) %>%      # Remove everything before OX=
  mutate(gene_symbol = gsub(" .*", "", gene_symbol))    # Remove everything after species ID

plot = overlap %>%
  rename(Iso = X) %>%
  inner_join(gene) %>%
  dplyr::select(gene_symbol, log2FoldChange.x, log2FoldChange.y) %>%
  rename(Hot_LogFoldChange = log2FoldChange.x) %>%
  rename(Cold_LogFoldChange = log2FoldChange.y)
```

Plotting delta ranks, this figure does not appear like this in the main manuscript. Only select genes are highlighted, for interactive gene plot visit my website: www.wuitchik.weebly.com/bioinformatics.html

```
delta_ranks = ggplot(plot, aes(Hot_LogFoldChange, Cold_LogFoldChange, label = gene_symbol)) +
  geom_point() +
  geom_text_repel()
delta_ranks
```




# GO Analysis

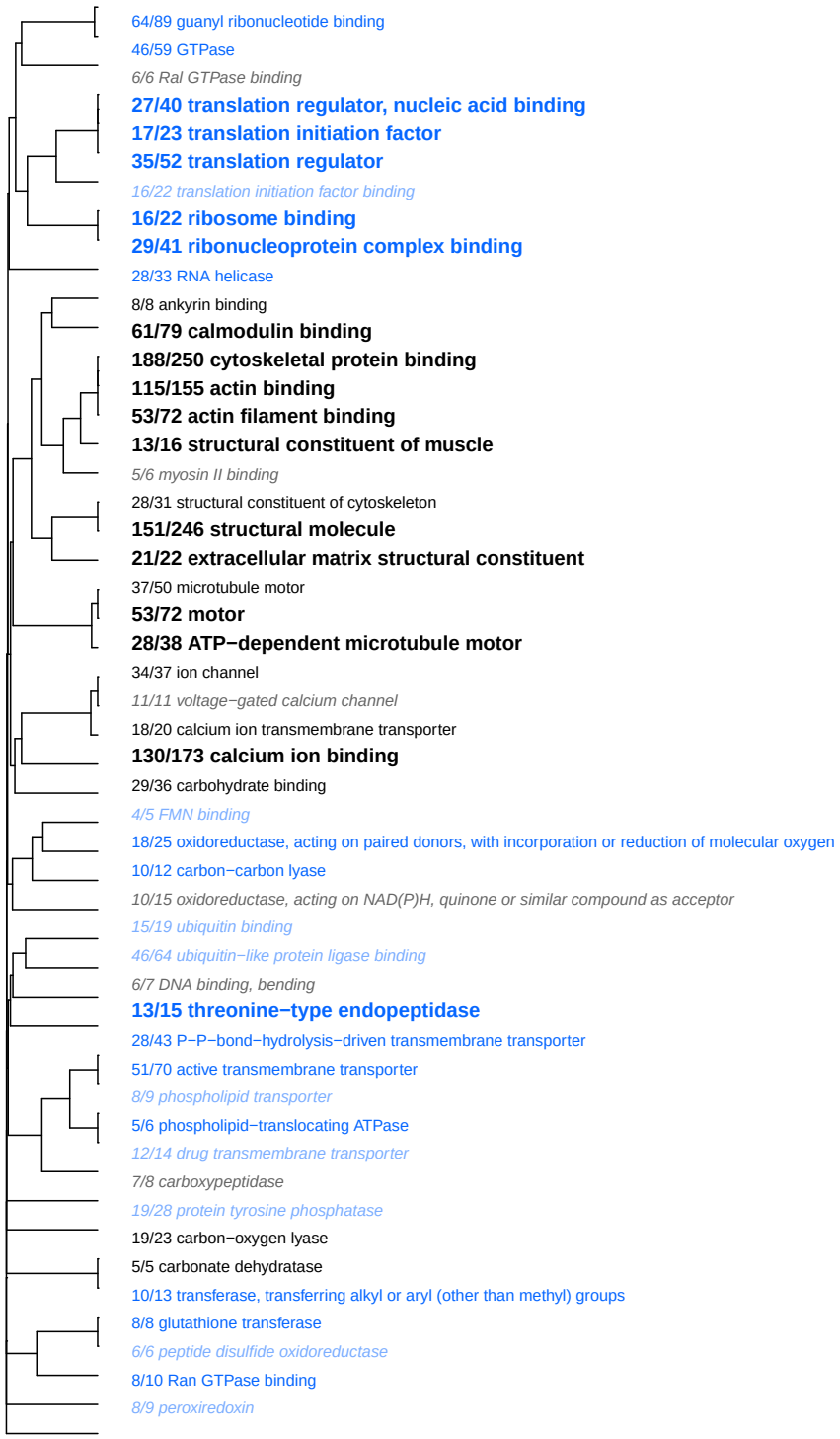
We are going to do a Mann-Whitney U test which requries that we first -logged the pvalue, and multiply it by -1 if it's less than zero or by 1 if it's greater than zero. Note that this needs to be opened and saved in excel (no other changes) as I believe the way R is saving the file the unicode does not work with Misha's script.

```
cold_go_input = read.csv("cold_results.csv") %>%
  mutate(mutated_p = -log(pvalue)) %>%
  mutate(mutated_p_updown = ifelse(log2FoldChange < 0, mutated_p*-1, mutated_p*1)) %>%
  dplyr::select(X, mutated_p_updown) %>%
  na.omit()

colnames(cold_go_input) = NULL

write.csv(cold_go_input, "cold_go_input.csv", row.names = FALSE)
```

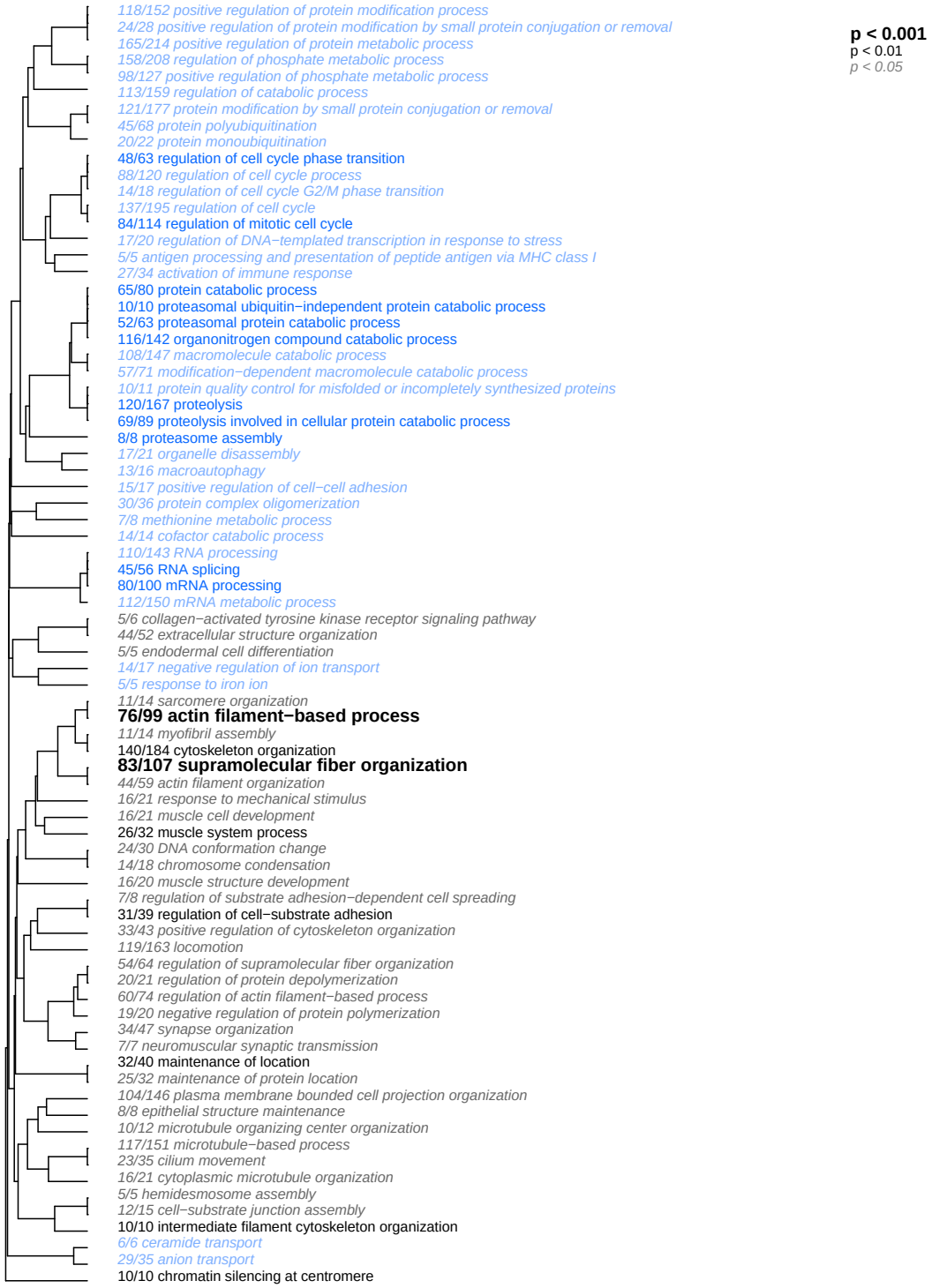
## Cold

### Molecular functions

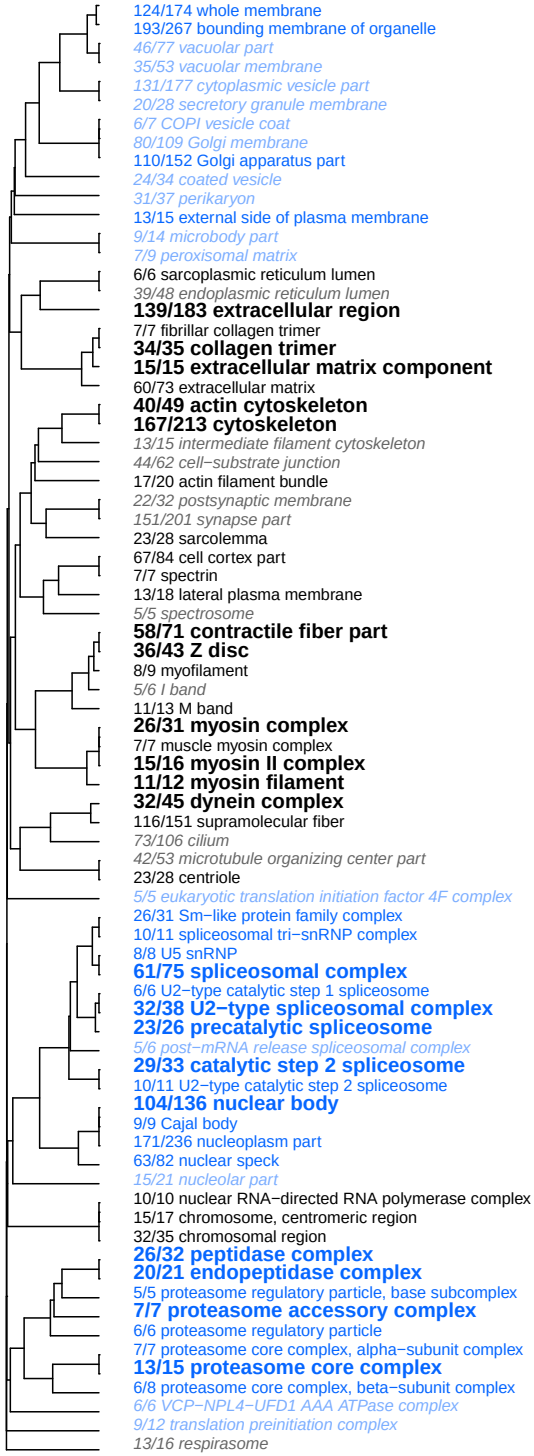
These code are all adapted from Dr. Matz and can be found GO_MWU https://github.com/z0on/GO_MWU This will be broken down into three sections


64/89 guanyl ribonucleotide binding
46/59 GTPase
6/6 Ral GTPase binding
27/40 translation regulator, nucleic acid binding
17/23 translation initiation factor
35/52 translation regulator
16/22 translation initiation factor binding
16/22 ribosome binding
29/41 ribonucleoprotein complex binding
28/33 RNA helicase
8/8 ankyrin binding
61/79 calmodulin binding
188/250 cytoskeletal protein binding
115/155 actin binding
53/72 actin filament binding
13/16 structural constituent of muscle
5/6 myosin II binding
28/31 structural constituent of cytoskeleton
151/246 structural molecule
21/22 extracellular matrix structural constituent
37/50 microtubule motor
53/72 motor
28/38 ATP−dependent microtubule motor
34/37 ion channel
11/11 voltage−gated calcium channel
18/20 calcium ion transmembrane transporter
130/173 calcium ion binding
29/36 carbohydrate binding
4/5 FMN binding
18/25 oxidoreductase, acting on paired donors, with incorporation or reduction of molecular oxygen
10/12 carbon−carbon lyase
10/15 oxidoreductase, acting on NAD(P)H, quinone or similar compound as acceptor
15/19 ubiquitin binding
46/64 ubiquitin−like protein ligase binding
6/7 DNA binding, bending
13/15 threonine−type endopeptidase
28/43 P−P−bond−hydrolysis−driven transmembrane transporter
51/70 active transmembrane transporter
8/9 phospholipid transporter
5/6 phospholipid−translocating ATPase
12/14 drug transmembrane transporter
7/8 carboxypeptidase
19/28 protein tyrosine phosphatase
19/23 carbon−oxygen lyase
5/5 carbonate dehydratase
10/13 transferase, transferring alkyl or aryl (other than methyl) groups
8/8 glutathione transferase
6/6 peptide disulfide oxidoreductase
8/10 Ran GTPase binding
8/9 peroxiredoxin
p < 0.01
p < 0.05
p < 0.1

## Biological Process



## Cellular Component

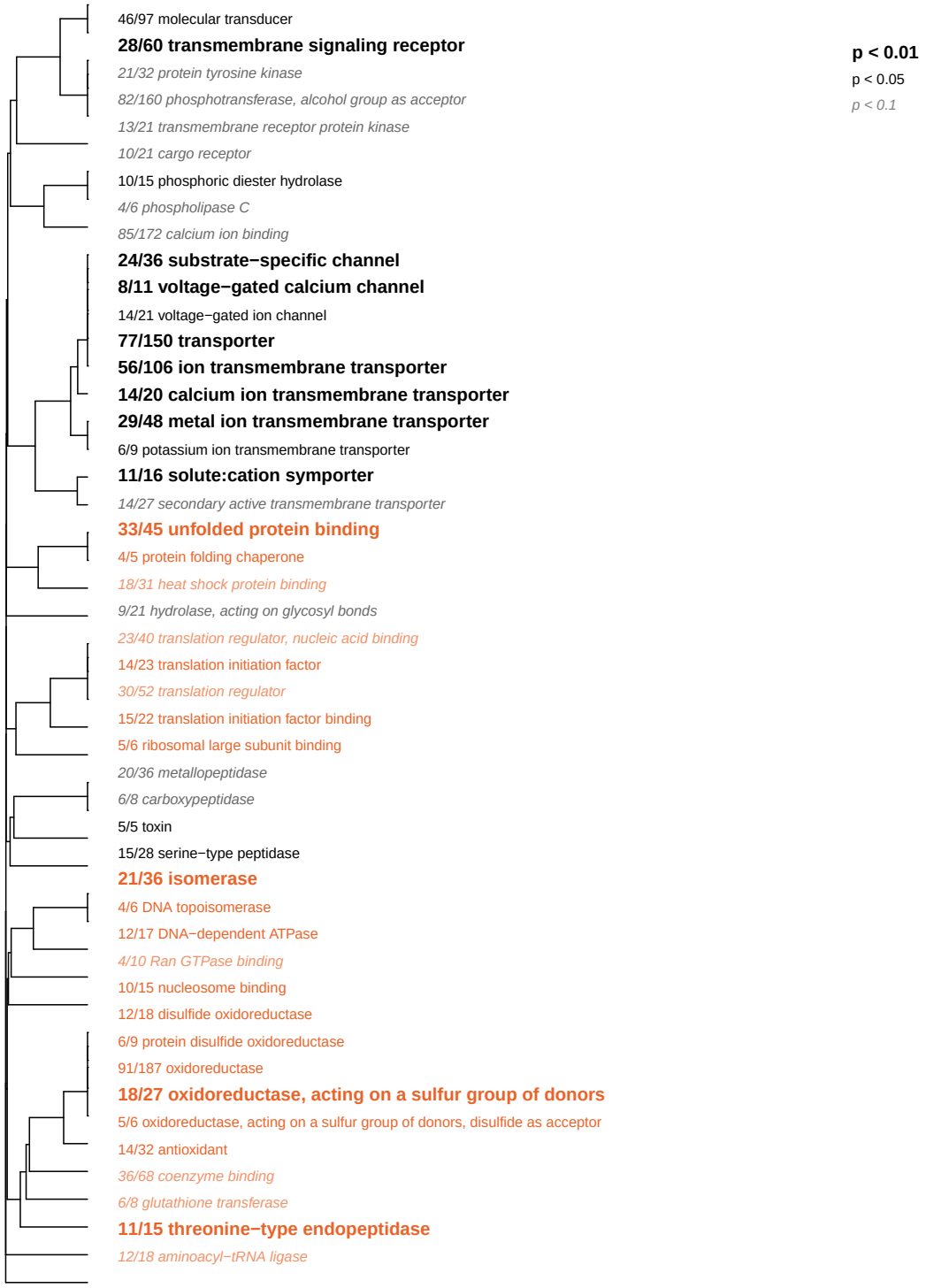



**p < 0.001**
p < 0.01
*p < 0.05*

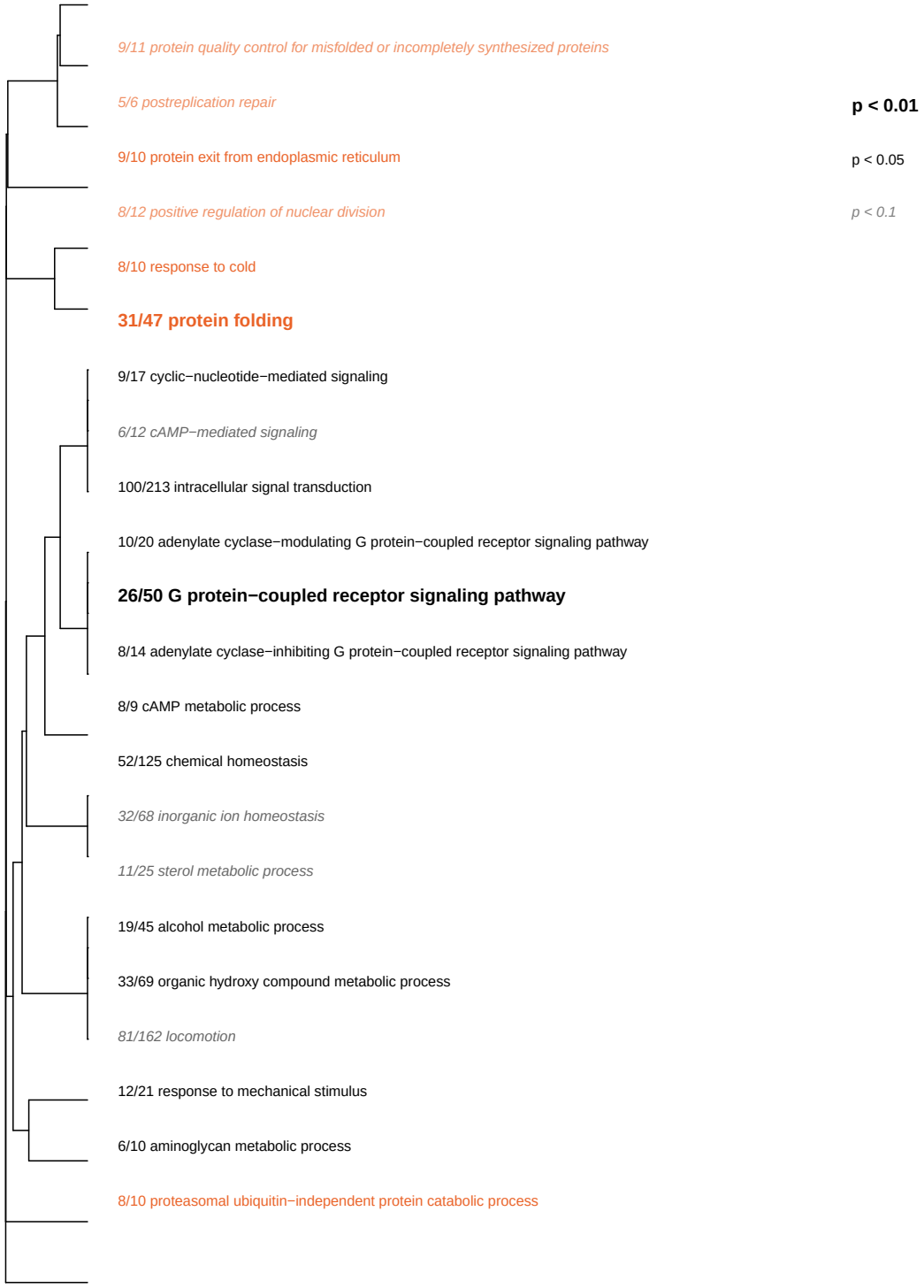
### Hot

```
hot_go_input = read.csv("hot_results.csv") %>%
  mutate(mutated_p = -log(pvalue)) %>%
  mutate(mutated_p_updown = ifelse(log2FoldChange < 0, mutated_p*-1, mutated_p*1)) %>%
  dplyr::select(X, mutated_p_updown) %>%
  na.omit()

colnames(hot_go_input) = NULL

write.csv(hot_go_input, "hot_go_input.csv", row.names = FALSE)
```

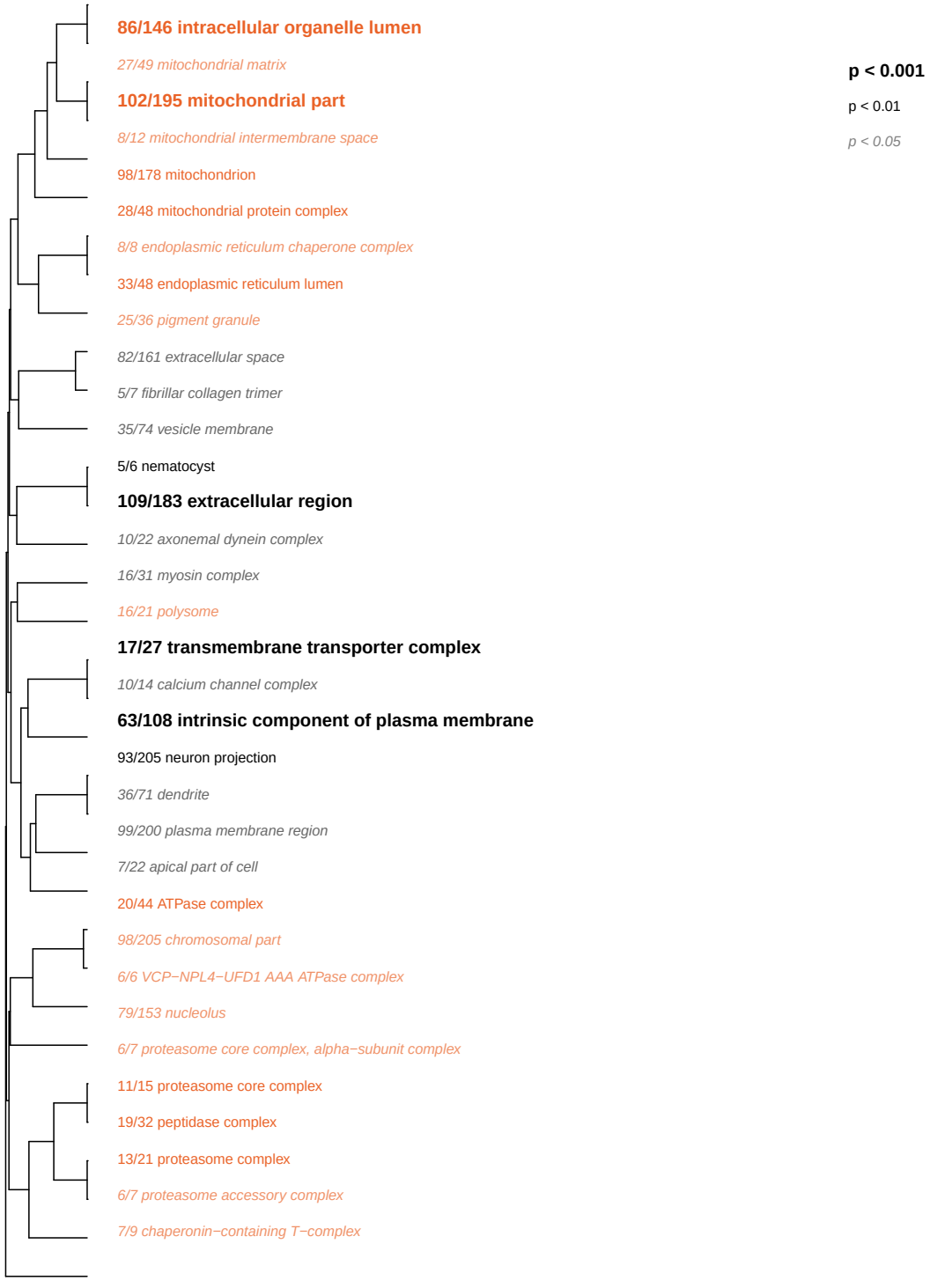
## Molecular functions



## Biological Process



## Cellular Component

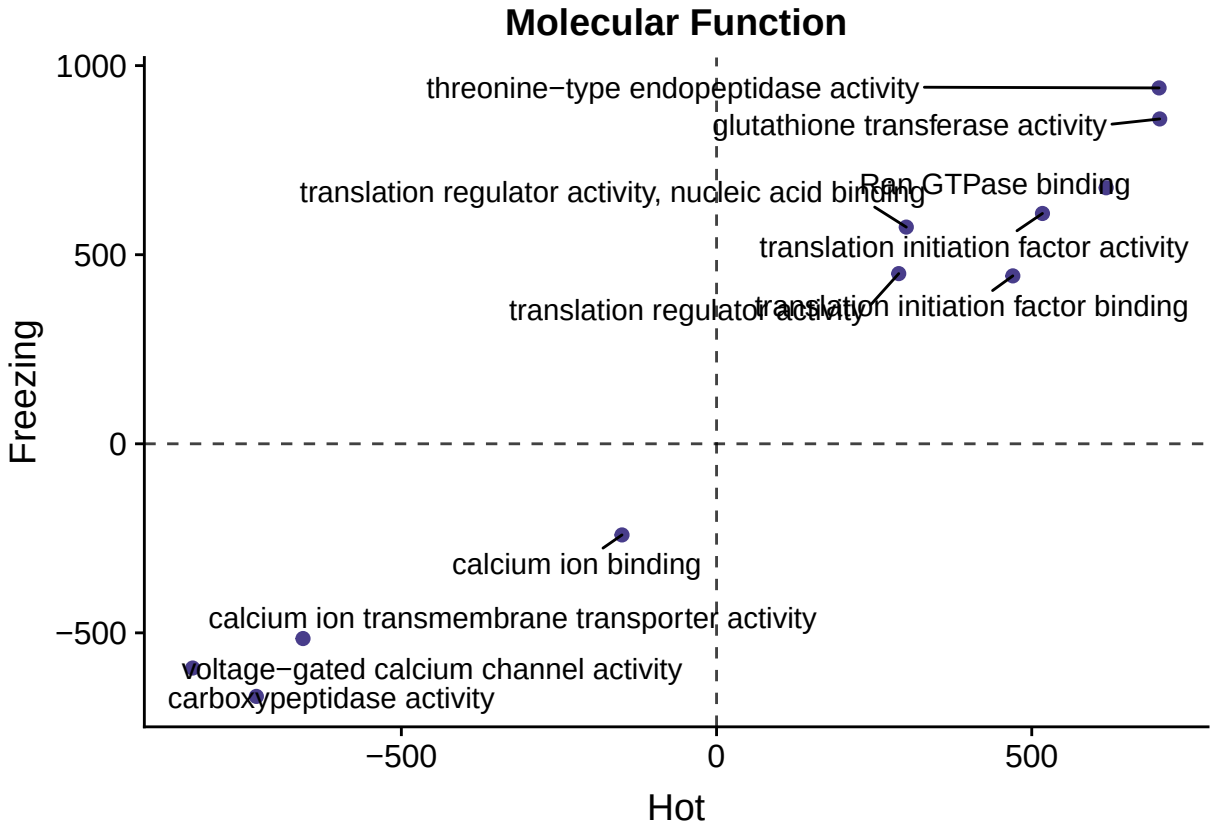


## Comparing Hot and Cold

### Molecular Functions

```
mf_hotMWU =read.table("MWU_MF_hot_go_input_excel_saved.csv",header=T)
mf_coldMWU =read.table("MWU_MF_cold_go_input_excel_saved.csv",header=T)

# Terms in both sets
mf_goods=intersect(mf_hotMWU$term,mf_coldMWU$term)
data1=mf_hotMWU[mf_hotMWU$term %in% mf_goods,]
data2=mf_coldMWU[mf_coldMWU$term %in% mf_goods,]

# Combine them
ress=merge(data1,data2,by="term")

plot = ress %>%
  mutate(colour =
           case_when( p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x > 0 & delta.rank.y > 0 ~ 'red',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x < 0 & delta.rank.y < 0 ~ 'blue',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x > 0 & delta.rank.y < 0 ~ 'purple',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x < 0 & delta.rank.y > 0 ~ 'green')) %
  replace_na(list(colour = "black"))

# This is to manually look for interesting go terms, and you can play with it in excel
mf_interest = plot %>%
  filter(p.adj.x <0.1) %>%
  filter(p.adj.y < 0.1)

write.csv(mf_interest, "mf_interesting.csv")

# Read back in your manipulated csv for those that you want to use as labels
mf_interest = read.csv("mf_interesting.csv")

# Here is the actual plot, lots of it is redundant

mf_plot = ggplot(mf_interest, aes(delta.rank.x, delta.rank.y, label = name.y)) +
  geom_point(aes(color = colour), size = 2, show.legend = FALSE) +
  scale_color_manual(values = c(red = "darkslateblue",
                                blue = "darkslateblue",
                                green = "darkslateblue",
                                red = "darkslateblue",
                                purple = "darkslateblue",
                                black = alpha("black", 0.15))) +
  scale_fill_manual(values = c(red = "orangered",
                               blue = "dodgerblue2",
                               green = "seagreen3",
                               red = "orangered2",
                               purple = "plum2",
                              black = "black")) +
    geom_text_repel(data = mf_interest, aes(),
                    segment.alpha = 1,
                    box.padding = .5,
                    direction = "both") +
  scale_size("size") +
```

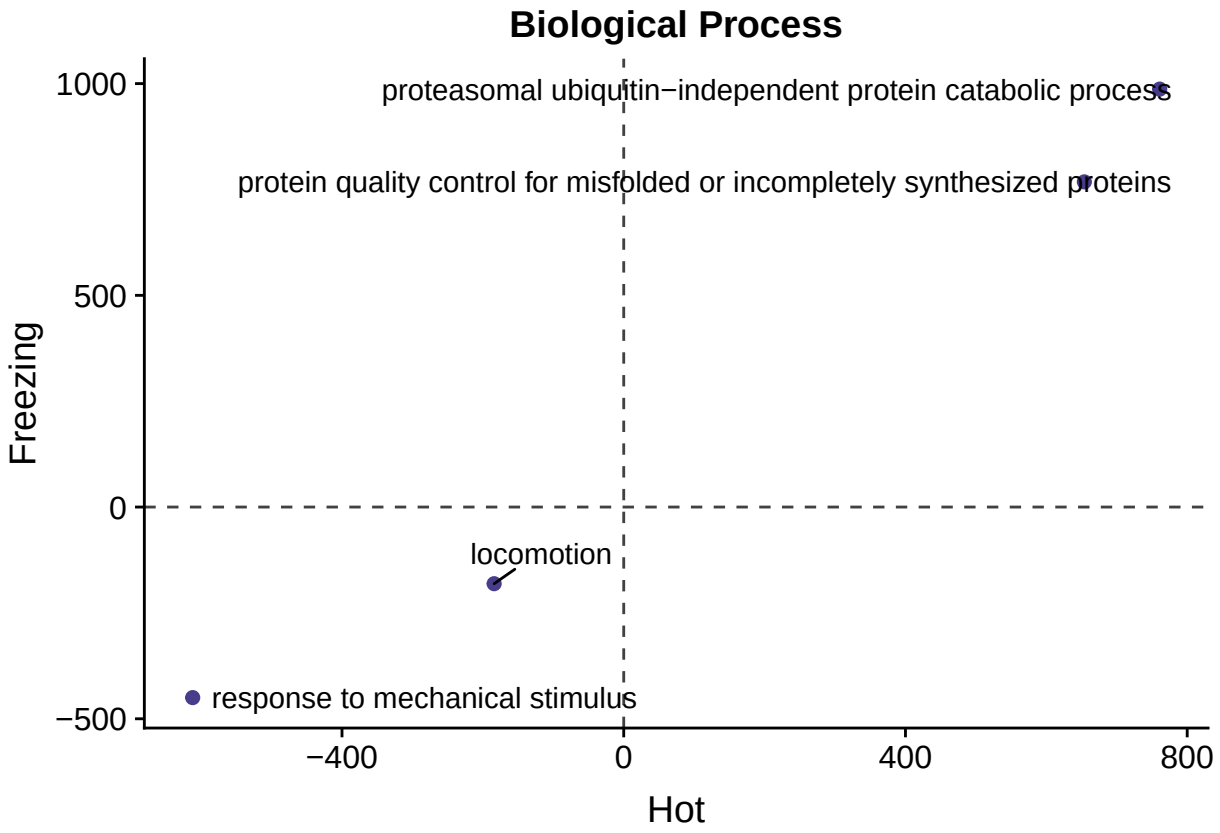
```
  labs( x = "Hot",
        y = "Freezing") +
  labs(title = "Molecular Function") +
  geom_vline(xintercept = 0, linetype = 2, alpha = 0.75) +
  geom_hline(yintercept = 0, linetype = 2, alpha = 0.75) +
  theme_cowplot()
mf_plot
```




**Biological Process**

```
bp_hotMWU =read.table("MWU_bp_hot_go_input_excel_saved.csv",header=T)
bp_coldMWU =read.table("MWU_bp_cold_go_input_excel_saved.csv",header=T)

# Terms in both sets
bp_goods=intersect(bp_hotMWU$term,bp_coldMWU$term)
data1=bp_hotMWU[bp_hotMWU$term %in% bp_goods,]
data2=bp_coldMWU[bp_coldMWU$term %in% bp_goods,]

# Combine them
ress=merge(data1,data2,by="term")

plot = ress %>%
  mutate(colour =
           case_when( p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x > 0 & delta.rank.y > 0 ~ 'red',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x < 0 & delta.rank.y < 0 ~ 'blue',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x > 0 & delta.rank.y < 0 ~ 'purple',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x < 0 & delta.rank.y > 0 ~ 'green')) %
```

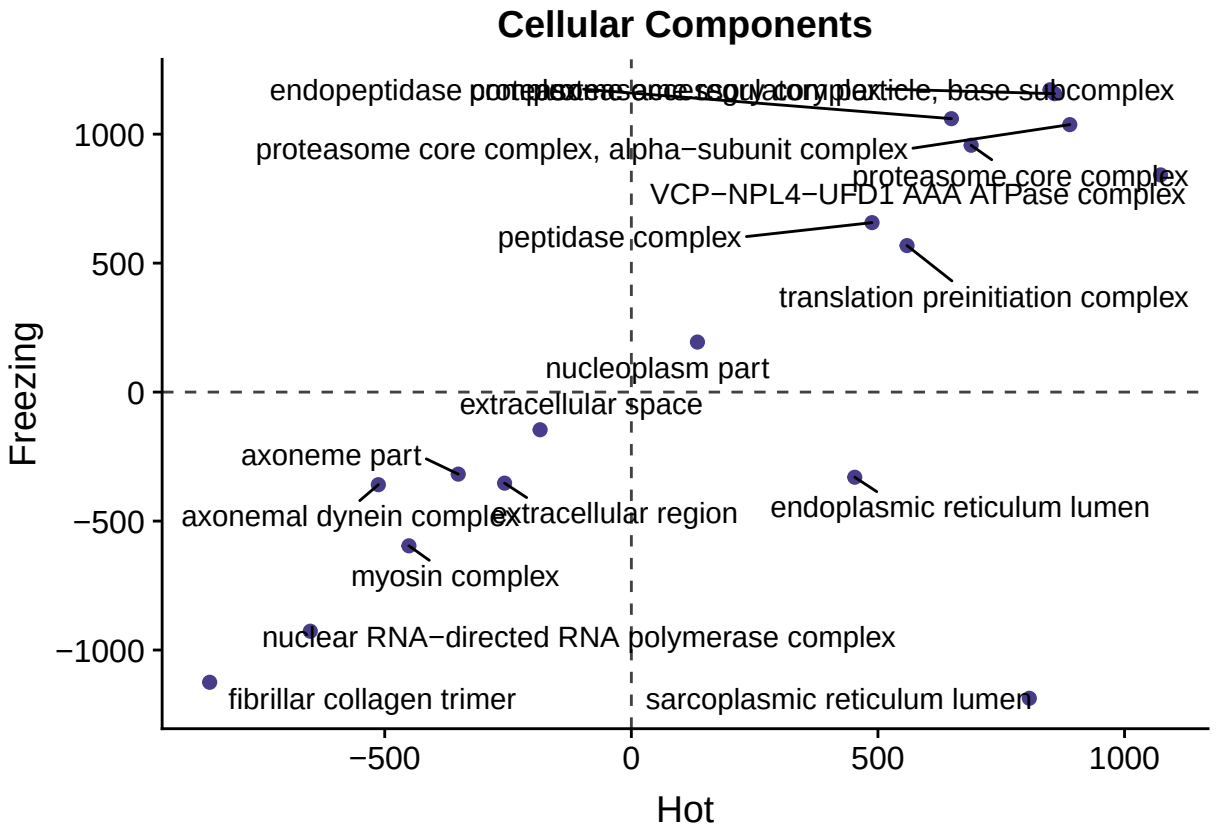
```
  replace_na(list(colour = "black"))

# This is to manually look for interesting go terms, and you can play with it in excel
bp_interest = plot %>%
  filter(p.adj.x <0.1) %>%
  filter(p.adj.y < 0.1)

write.csv(bp_interest, "bp_interesting.csv")

# Read back in your manipulated csv for those that you want to use as labels
bp_interest = read.csv("bp_interesting.csv")

# Here is the actual plot, lots of it is redundant

bp_plot = ggplot(bp_interest, aes(delta.rank.x, delta.rank.y, label = name.y)) +
  geom_point(aes(color = colour), size = 2, show.legend = FALSE) +
  scale_color_manual(values = c(red = "darkslateblue",
                                blue = "darkslateblue",
                                green = "darkslateblue",
                                red = "darkslateblue",
                                purple = "darkslateblue",
                                black = alpha("black", 0.15))) +
  scale_fill_manual(values = c(red = "orangered",
                               blue = "dodgerblue2",
                               green = "seagreen3",
                               red = "orangered2",
                               purple = "plum2",
                              black = "black")) +
    geom_text_repel(data = bp_interest, aes(),
                    segment.alpha = 1,
                    box.padding = .5,
                    direction = "both") +
  scale_size("size") +
  labs( x = "Hot",
        y = "Freezing") +
  labs(title = "Biological Process") +
  geom_vline(xintercept = 0, linetype = 2, alpha = 0.75) +
  geom_hline(yintercept = 0, linetype = 2, alpha = 0.75) +
  theme_cowplot()
bp_plot
```




**Cellular Components**

```
cc_hotMWU =read.table("MWU_cc_hot_go_input_excel_saved.csv",header=T)
cc_coldMWU =read.table("MWU_cc_cold_go_input_excel_saved.csv",header=T)

# Terms in both sets
cc_goods=intersect(cc_hotMWU$term,cc_coldMWU$term)
data1=cc_hotMWU[cc_hotMWU$term %in% cc_goods,]
data2=cc_coldMWU[cc_coldMWU$term %in% cc_goods,]

# Combine them
ress=merge(data1,data2,by="term")

plot = ress %>%
  mutate(colour =
           case_when( p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x > 0 & delta.rank.y > 0 ~ 'red',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x < 0 & delta.rank.y < 0 ~ 'blue',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x > 0 & delta.rank.y < 0 ~ 'purple',
                      p.adj.x < 0.1 & p.adj.y < 0.1 & delta.rank.x < 0 & delta.rank.y > 0 ~ 'green')) %
  replace_na(list(colour = "black"))

# This is to manually look for interesting go terms, and you can play with it in excel
cc_interest = plot %>%
  filter(p.adj.x <0.1) %>%
  filter(p.adj.y < 0.1)

write.csv(cc_interest, "cc_interesting.csv")
```

```
# Read back in your manipulated csv for those that you want to use as labels
cc_interest = read.csv("cc_interesting.csv")

# Here is the actual plot, lots of it is redundant

cc_plot = ggplot(cc_interest, aes(delta.rank.x, delta.rank.y, label = name.y)) +
  geom_point(aes(color = colour), size = 2, show.legend = FALSE) +
  scale_color_manual(values = c(red = "darkslateblue",
                                blue = "darkslateblue",
                                green = "darkslateblue",
                                red = "darkslateblue",
                                purple = "darkslateblue",
                                black = alpha("black", 0.15))) +
  scale_fill_manual(values = c(red = "orangered",
                               blue = "dodgerblue2",
                               green = "seagreen3",
                               red = "orangered2",
                               purple = "plum2",
                              black = "black")) +
    geom_text_repel(data = cc_interest, aes(),
                    segment.alpha = 1,
                    box.padding = .5,
                    direction = "both") +
  scale_size("size") +
  labs( x = "Hot",
        y = "Freezing") +
  labs(title = "Cellular Components") +
  geom_vline(xintercept = 0, linetype = 2, alpha = 0.75) +
  geom_hline(yintercept = 0, linetype = 2, alpha = 0.75) +
  theme_cowplot()
cc_plot
```




## Heatmaps

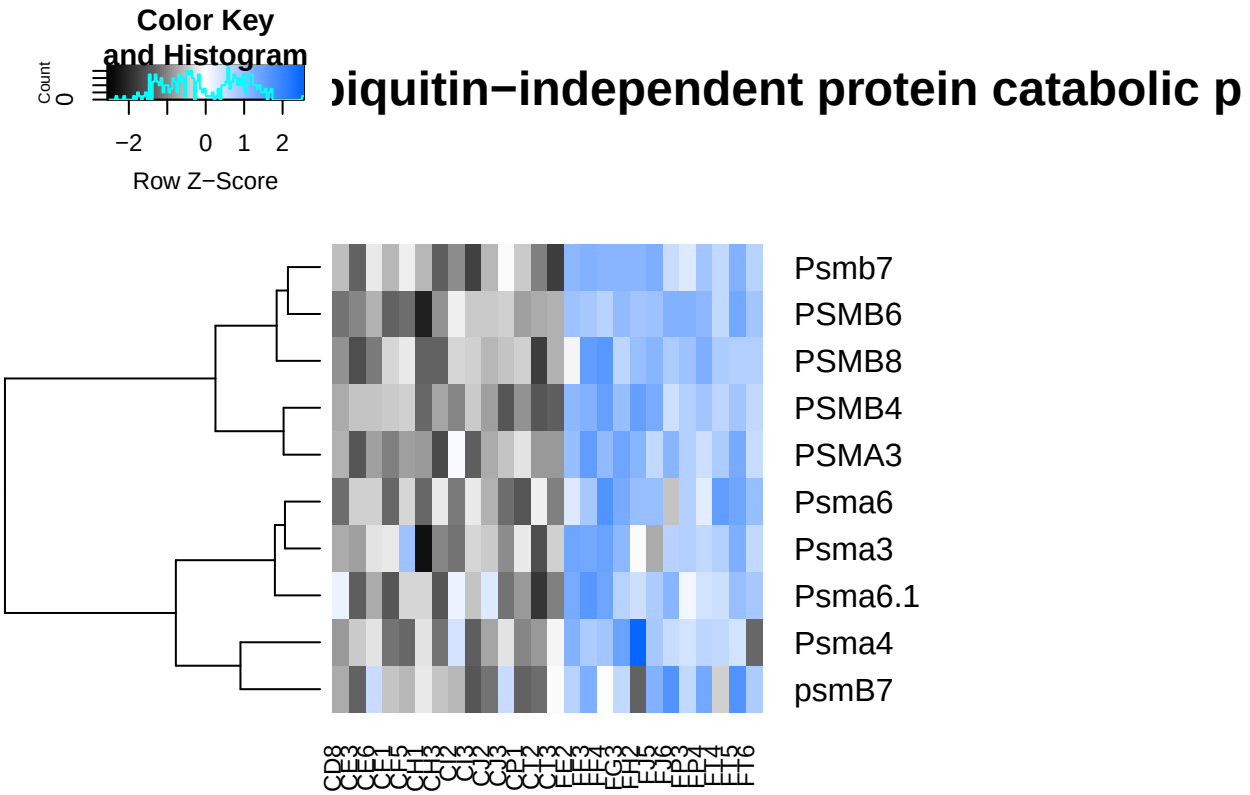
The purpose of these heatmaps is not to provide comprehensive heatmaps used in the manuscript. Rather, it is to highlight the code used to form a basis on how it was applied for each individual heatmap.

```
iso2go = read_tsv("astrangia_iso2go.tab") %>%
  rename(Iso = Gene_id)

cold_results_df = read.csv("cold_results.csv") %>%
  rename("Iso" = "X")

hot_results_df = read.csv("hot_results.csv") %>%
  rename("Iso" = "X")

cold_rlog = read.csv("cold_rlogged.csv") %>%
  rename("Iso" = "X") %>%
  left_join(cold_results_df) %>%
  filter(padj < 0.1) %>%
  dplyr::select(-baseMean, -log2FoldChange, -lfcSE, -stat, -pvalue, -padj)

hot_rlog = read.csv("hot_rlogged.csv") %>%
  rename("Iso" = "X") %>%
  left_join(hot_results_df) %>%
  filter(padj < 0.1) %>%
  dplyr::select(-baseMean, -log2FoldChange, -lfcSE, -stat, -pvalue, -padj)

hot_colour = colorRampPalette(rev(c("#ea6227","#f09167","white", "grey40","black")))(100)
cold_colour = colorRampPalette(rev(c("#0666ff","#7caeff","white", "grey40","black")))(100)
```

### GO:0010499 proteasomal ubiquitin-independent protein catabolic process

Cold

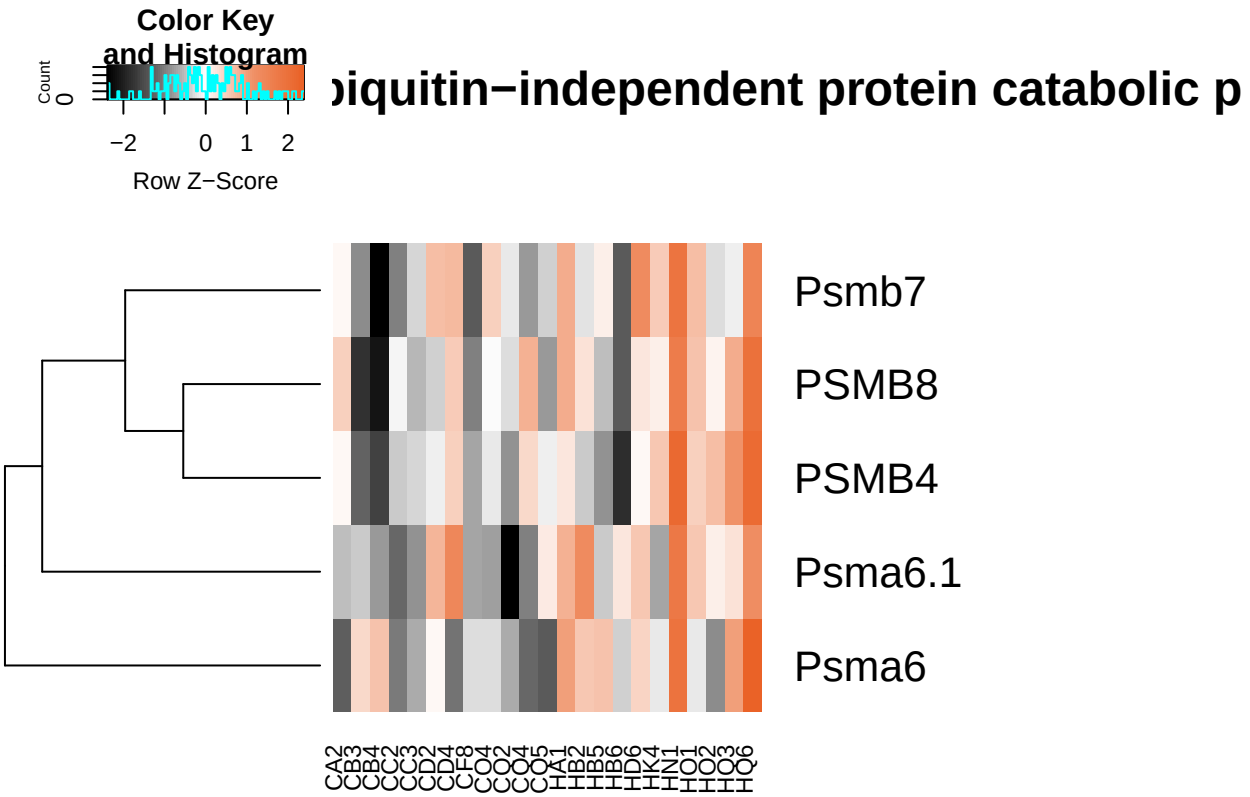
```
GO_0010499_cold = iso2go %>%
  filter(str_detect(GO_id, "GO:0010499")) %>%
  left_join(gene) %>%
  left_join(cold_rlog) %>%
  mutate(gene_symbol = make.names(gene_symbol, unique = TRUE)) %>%
  column_to_rownames(var = "gene_symbol") %>%
  dplyr::select(-GO_id, -Gene, -Iso) %>%
  drop_na()%>%
  dplyr::select(sort(current_vars()))

GO_0010499_cold_means=apply(GO_0010499_cold,1,mean) # means of rows
explc=GO_0010499_cold-GO_0010499_cold_means # subtracting them

heatmap.2(as.matrix(GO_0010499_cold), col = cold_colour, Rowv = TRUE, Colv = FALSE, scale = "row",
          dendrogram = "both",
          trace = "none",
          main = "GO:0010499 proteasomal ubiquitin-independent protein catabolic process",
          margin = c(5,15))
```




Hot

```
GO_0010499_hot = iso2go %>%
  filter(str_detect(GO_id, "GO:0010499")) %>%
  left_join(gene) %>%
  left_join(hot_rlog) %>%
```

```
  mutate(gene_symbol = make.names(gene_symbol, unique = TRUE)) %>%
  column_to_rownames(var = "gene_symbol") %>%
  dplyr::select(-GO_id, -Gene, -Iso) %>%
  drop_na() %>%
  dplyr::select(sort(current_vars()))


GO_0010499_hot_means=apply(GO_0010499_hot,1,mean) # means of rows
explc=GO_0010499_hot-GO_0010499_hot_means # subtracting them


heatmap.2(as.matrix(explc), col = hot_colour, Rowv = TRUE, Colv = FALSE, scale = "row",
          dendrogram = "both",
          trace = "none",
          main = "GO:0010499 proteasomal ubiquitin-independent protein catabolic process",
          margin = c(5,15))
```




## Session Info

```
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.15.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
```

```
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_CA.UTF-8/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] ggrepel_0.8.0               MASS_7.3-51.1
##  [3] VennDiagram_1.6.20          futile.logger_1.4.3
##  [5] prettydoc_0.2.1             reshape2_1.4.3
##  [7] kableExtra_1.1.0            vegan_2.5-4
##  [9] permute_0.9-4               plotly_4.9.0
## [11] knitr_1.28                  gplots_3.0.1.1
## [13] RColorBrewer_1.1-2          cowplot_0.9.4
## [15] DESeq_1.34.1                lattice_0.20-38
## [17] locfit_1.5-9.1              genefilter_1.64.0
## [19] arrayQualityMetrics_3.38.0  affycoretools_1.54.0
## [21] DESeq2_1.22.2               SummarizedExperiment_1.12.0
## [23] DelayedArray_0.8.0          BiocParallel_1.16.5
## [25] matrixStats_0.54.0          Biobase_2.42.0
## [27] GenomicRanges_1.34.0        GenomeInfoDb_1.18.1
## [29] IRanges_2.16.0              S4Vectors_0.20.1
## [31] BiocGenerics_0.28.0         forcats_0.4.0
## [33] stringr_1.4.0               dplyr_0.8.5
## [35] purrr_0.3.0                 readr_1.3.1
## [37] tidyr_1.0.2                 tibble_2.1.3
## [39] ggplot2_3.1.0               tidyverse_1.2.1
## [41] plyr_1.8.4
##
## loaded via a namespace (and not attached):
##   [1] R.utils_2.7.0             tidyselect_0.2.5
##   [3] RSQLite_2.1.1             AnnotationDbi_1.44.0
##   [5] htmlwidgets_1.3           beadarray_2.32.0
##   [7] munsell_0.5.0             codetools_0.2-16
##   [9] preprocessCore_1.44.0     withr_2.1.2
##  [11] colorspace_1.4-0          Category_2.48.0
##  [13] OrganismDbi_1.24.0        rstudioapi_0.9.0
##  [15] setRNG_2013.9-1           GenomeInfoDbData_1.2.0
##  [17] hwriter_1.3.2             bit64_0.9-7
##  [19] vctrs_0.2.4               generics_0.0.2
##  [21] lambda.r_1.2.3            xfun_0.13
##  [23] biovizBase_1.30.1         R6_2.4.0
##  [25] illuminaio_0.24.0         gridSVG_1.6-0
##  [27] AnnotationFilter_1.6.0    bitops_1.0-6
##  [29] reshape_0.8.8             assertthat_0.2.0
##  [31] scales_1.0.0              nnet_7.3-12
##  [33] gtable_0.2.0              Cairo_1.5-9
##  [35] affy_1.60.0               ggbio_1.30.0
##  [37] ensembldb_2.6.5           rlang_0.4.5
##  [39] splines_3.5.1             rtracklayer_1.42.1
##  [41] lazyeval_0.2.1            acepack_1.4.1
```

```
##  [43] dichromat_2.0-0           broom_0.5.2
##  [45] checkmate_1.9.1           BiocManager_1.30.4
##  [47] yaml_2.2.0                modelr_0.1.4
##  [49] GenomicFeatures_1.34.3    backports_1.1.3
##  [51] Hmisc_4.2-0               RBGL_1.58.1
##  [53] tools_3.5.1               affyio_1.52.0
##  [55] ff_2.2-14                 Rcpp_1.0.1
##  [57] base64enc_0.1-3           progress_1.2.0
##  [59] zlibbioc_1.28.0           RCurl_1.95-4.11
##  [61] prettyunits_1.0.2         openssl_1.2.1
##  [63] rpart_4.1-13              haven_2.1.0
##  [65] cluster_2.0.7-1           magrittr_1.5
##  [67] futile.options_1.0.1      data.table_1.12.0
##  [69] ProtGenerics_1.14.0       hms_0.4.2
##  [71] evaluate_0.14             xtable_1.8-3
##  [73] XML_3.98-1.16             gcrma_2.54.0
##  [75] readxl_1.3.1              gridExtra_2.3
##  [77] compiler_3.5.1            biomaRt_2.38.0
##  [79] KernSmooth_2.23-15        crayon_1.3.4
##  [81] ReportingTools_2.22.1     R.oo_1.22.0
##  [83] htmltools_0.3.6           GOstats_2.48.0
##  [85] mgcv_1.8-26               Formula_1.2-3
##  [87] geneplotter_1.60.0        lubridate_1.7.4
##  [89] DBI_1.0.0                 formatR_1.5
##  [91] Matrix_1.2-15             cli_1.0.1
##  [93] vsn_3.50.0                R.methodsS3_1.7.1
##  [95] gdata_2.18.0              pkgconfig_2.0.2
##  [97] GenomicAlignments_1.18.1  foreign_0.8-71
##  [99] xml2_1.2.0                foreach_1.4.4
## [101] annotate_1.60.0           BeadDataPackR_1.34.0
## [103] affyPLM_1.58.0            webshot_0.5.1
## [105] XVector_0.22.0            AnnotationForge_1.24.0
## [107] rvest_0.3.4               VariantAnnotation_1.28.10
## [109] digest_0.6.18             graph_1.60.0
## [111] Biostrings_2.50.2         rmarkdown_2.1
## [113] base64_2.0                cellranger_1.1.0
## [115] htmlTable_1.13.1          edgeR_3.24.3
## [117] GSEABase_1.44.0           curl_3.3
## [119] Rsamtools_1.34.1          gtools_3.8.1
## [121] lifecycle_0.2.0           nlme_3.1-137
## [123] jsonlite_1.6              PFAM.db_3.7.0
## [125] viridisLite_0.3.0         askpass_1.1
## [127] limma_3.38.3              BSgenome_1.50.0
## [129] pillar_1.3.1              GGally_1.4.0
## [131] httr_1.4.0                survival_2.43-3
## [133] GO.db_3.7.0               glue_1.3.0.9000
## [135] iterators_1.0.10          bit_1.1-14
## [137] Rgraphviz_2.26.0          stringi_1.2.4
## [139] blob_1.1.1                oligoClasses_1.44.0
## [141] latticeExtra_0.6-28       caTools_1.17.1.1
## [143] memoise_1.1.0
```