1.2_Gene_expression_analysis

Fay

2022-07-27

1. Gene expression in the laboratory infections - Heatmap

```
field <- read.csv("output_data/gene_expression/data_products/field_imputed_gene_expression.csv")

Genes_field   <- c("IFNy",  "CXCR3", "IL.6", #"GBP2", "IL.12", "IRG6",
                    "IL.10", "IL.13", "IL1RN",
                    "CXCR3", "CASP1", "CXCL9",
                    "IDO1", "IRGM1", "MPO", "MUC2", "MUC5AC", "MYD88",
                    "NCR1", "PRF1", "RETNLB", "SOCS1", "TICAM1", "TNF")
```

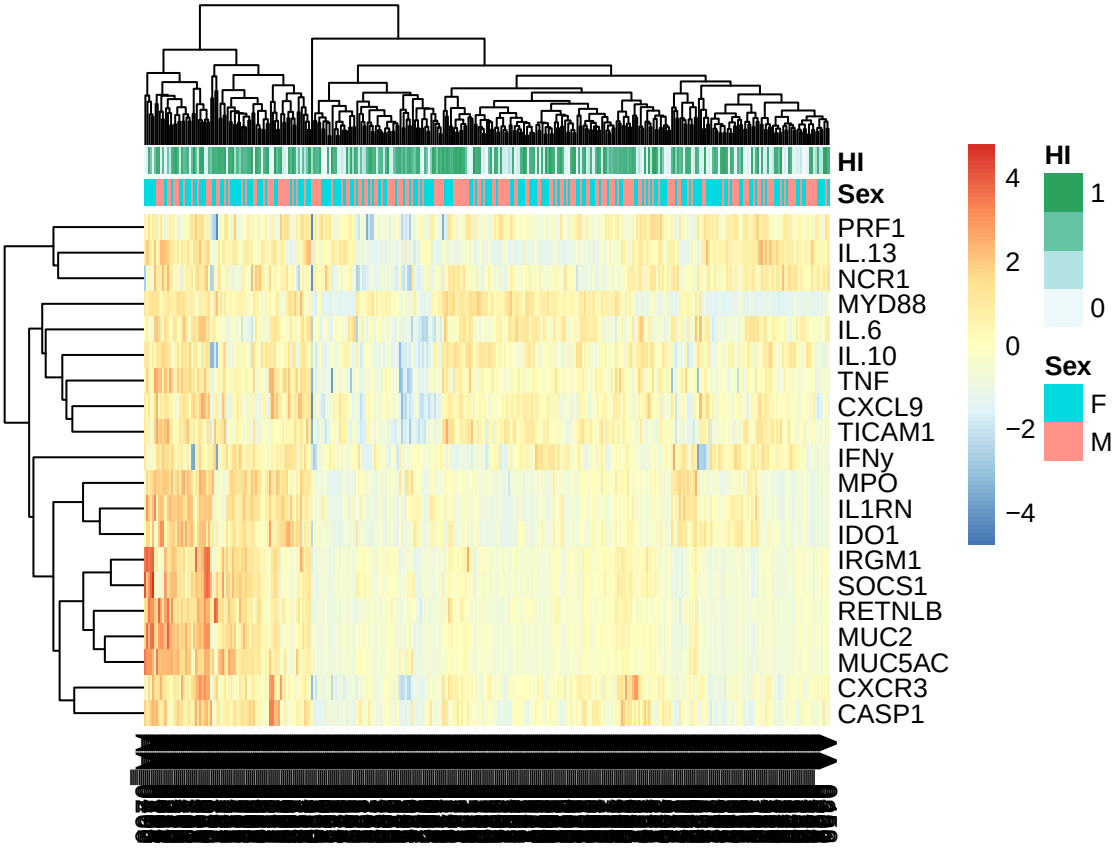
```
#filter out weird ifn y values
field <- field %>%
  filter(!IFNy >=40)

# turn the data frame into a matrix and transpose it. We want to have each cell
 # type as a row name
 gene <- t(as.matrix(field %>% dplyr::select(c(Mouse_ID, all_of(Genes_field)))))

 #switch the matrix back to a data frame format
 gene <- as.data.frame(gene)

 # turn the first row into column names
 gene %>%
     row_to_names(row_number = 1) -> heatmap_data

 table(rowSums(is.na(heatmap_data)) == nrow(heatmap_data))
```

```
##
## FALSE
##    20
```

```
 # turn the columns to numeric other wise the heatmap function will not work
 heatmap_data[] <- lapply(heatmap_data, function(x) as.numeric(as.character(x)))

 # remove columns with only NAs
 heatmap_data <- Filter(function(x)!all(is.na(x)), heatmap_data)

 #remove rows with only Nas
 heatmap_data <-  heatmap_data[, colSums(is.na(heatmap_data)) !=
                                 nrow(heatmap_data)]

 ### Prepare the annotation data frame for the heatmap
```

```
annotation_df <- as_tibble(field) %>%
  dplyr::group_by(Mouse_ID) %>%
  dplyr::select(c("Mouse_ID", "Sex", "HI")) %>%
  dplyr::filter(Mouse_ID %in% colnames(heatmap_data))

annotation_df <- unique(annotation_df)

annotation_df <- as.data.frame(unique(annotation_df)) %>%
  dplyr::select(-Mouse_ID)


### Prepare the annotation columns for the heatmap
rownames(annotation_df) <- annotation_df$Mouse_ID
# Match the row names to the heatmap data frame
rownames(annotation_df) <- colnames(heatmap_data)

heatmap_data <- as.matrix(heatmap_data)
```

Heatmap on gene expression data:



Heatmap with EIm mc

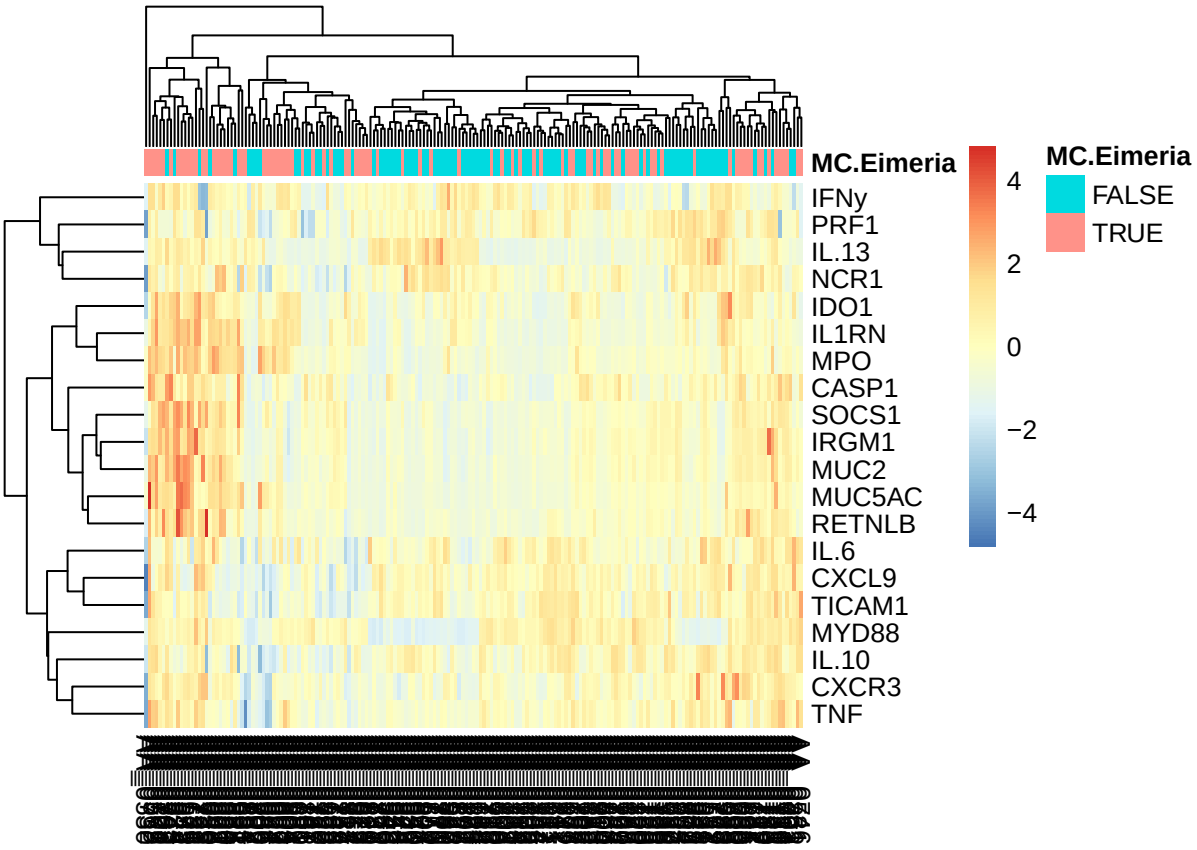
```
# turn the data frame into a matrix and transpose it. We want to have each cell
 # type as a row name
 gene <- t(as.matrix(field %>%
```

```
                      filter(!MC.Eimeria == "NA") %>%
                      dplyr::select(c(Mouse_ID, all_of(Genes_field)))))

 #switch the matrix back to a data frame format
 gene <- as.data.frame(gene)

 # turn the first row into column names
 gene %>%
     row_to_names(row_number = 1) -> heatmap_data

 table(rowSums(is.na(heatmap_data)) == nrow(heatmap_data))
```

```
##
## FALSE
##    20
```

```
 # turn the columns to numeric other wise the heatmap function will not work
 heatmap_data[] <- lapply(heatmap_data, function(x) as.numeric(as.character(x)))

 # remove columns with only NAs
 heatmap_data <- Filter(function(x)!all(is.na(x)), heatmap_data)

 #remove rows with only Nas
 heatmap_data <-  heatmap_data[, colSums(is.na(heatmap_data)) !=
                               nrow(heatmap_data)]

 ### Prepare the annotation data frame for the heatmap
 field$MC.Eimeria <- as.character(field$MC.Eimeria)

 annotation_df <- as_tibble(field) %>%
  filter(!MC.Eimeria == "NA") %>%
  dplyr::group_by(Mouse_ID) %>%
  dplyr::select(c("Mouse_ID", "MC.Eimeria")) %>%
  dplyr::filter(Mouse_ID %in% colnames(heatmap_data))

annotation_df <- unique(annotation_df)

annotation_df <- as.data.frame(unique(annotation_df)) %>%
  dplyr::select(-Mouse_ID)


### Prepare the annotation columns for the heatmap
rownames(annotation_df) <- annotation_df$Mouse_ID
# Match the row names to the heatmap data frame
rownames(annotation_df) <- colnames(heatmap_data)


heatmap_data <- as.data.frame(heatmap_data)
```

```
pheatmap(heatmap_data, annotation_col = annotation_df, scale = "row")
```



#

Heatmap with eimeria species

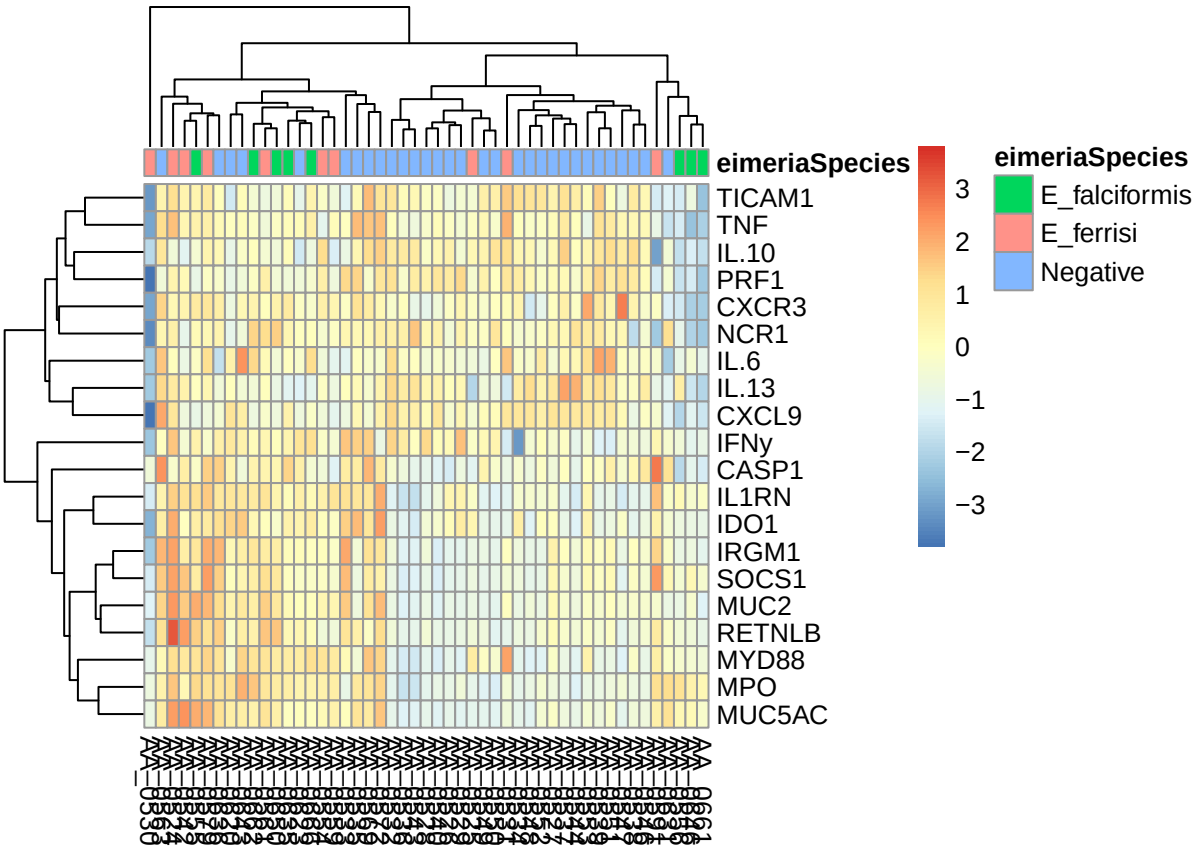
```
# turn the data frame into a matrix and transpose it. We want to have each cell
 # type as a row name
 gene <- t(as.matrix(field %>%
                       filter(!eimeriaSpecies == "NA") %>%
                       dplyr::select(c(Mouse_ID, all_of(Genes_field)))))

 #switch the matrix back to a data frame format
 gene <- as.data.frame(gene)

 # turn the first row into column names
 gene %>%
     row_to_names(row_number = 1) -> heatmap_data

 table(rowSums(is.na(heatmap_data)) == nrow(heatmap_data))
```

```
##
## FALSE
##    20
```

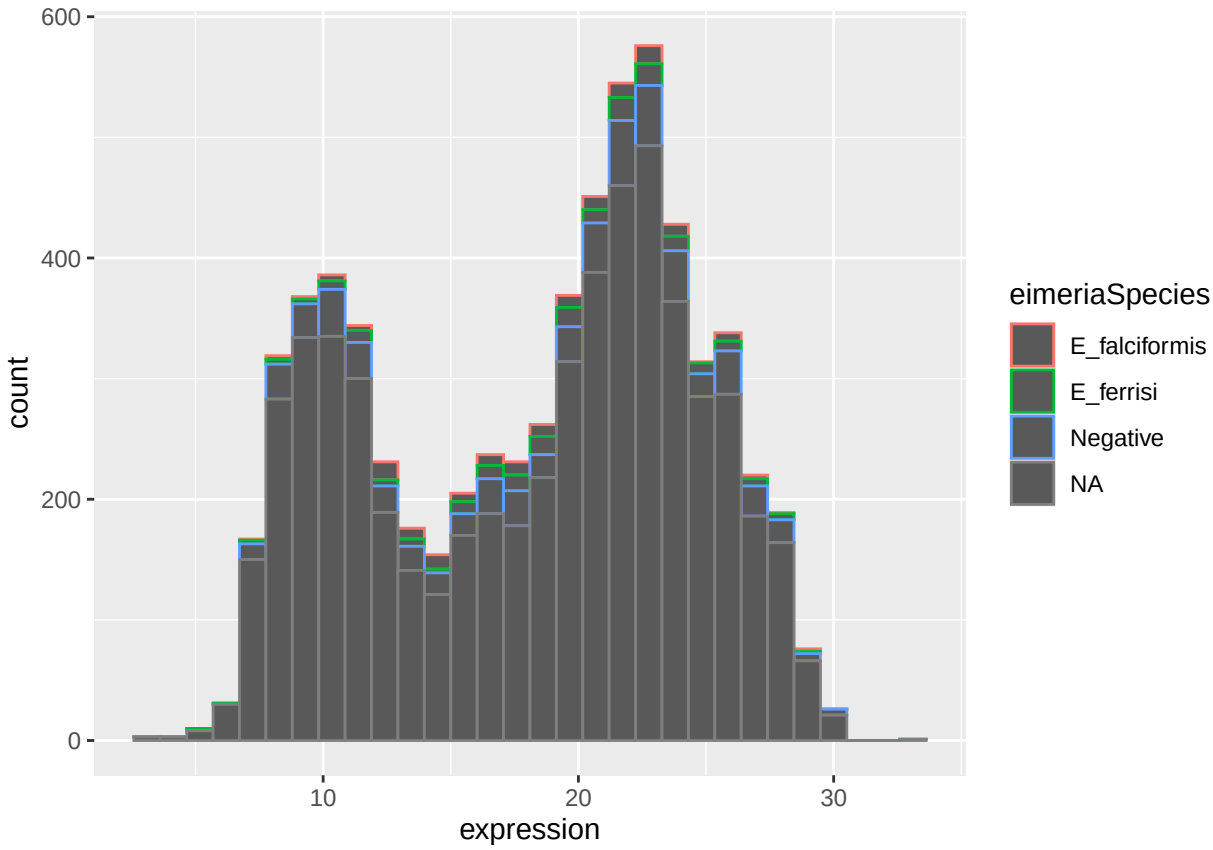
```
 # turn the columns to numeric other wise the heatmap function will not work
 heatmap_data[] <- lapply(heatmap_data, function(x) as.numeric(as.character(x)))

 # remove columns with only NAs
 heatmap_data <- Filter(function(x)!all(is.na(x)), heatmap_data)

 #remove rows with only Nas
 heatmap_data <-  heatmap_data[, colSums(is.na(heatmap_data)) !=
```

```
                                  nrow(heatmap_data)]

 annotation_df <- as_tibble(field) %>%
  filter(!MC.Eimeria == "NA") %>%
  dplyr::group_by(Mouse_ID) %>%
  dplyr::select(c("Mouse_ID", "eimeriaSpecies")) %>%
  dplyr::filter(Mouse_ID %in% colnames(heatmap_data))

annotation_df <- unique(annotation_df)

annotation_df <- as.data.frame(unique(annotation_df)) %>%
  dplyr::select(-Mouse_ID)


### Prepare the annotation columns for the heatmap
rownames(annotation_df) <- annotation_df$Mouse_ID
# Match the row names to the heatmap data frame
rownames(annotation_df) <- colnames(heatmap_data)


heatmap_data <- as.data.frame(heatmap_data)
```

```
pheatmap(heatmap_data, annotation_col = annotation_df, scale = "row")
```

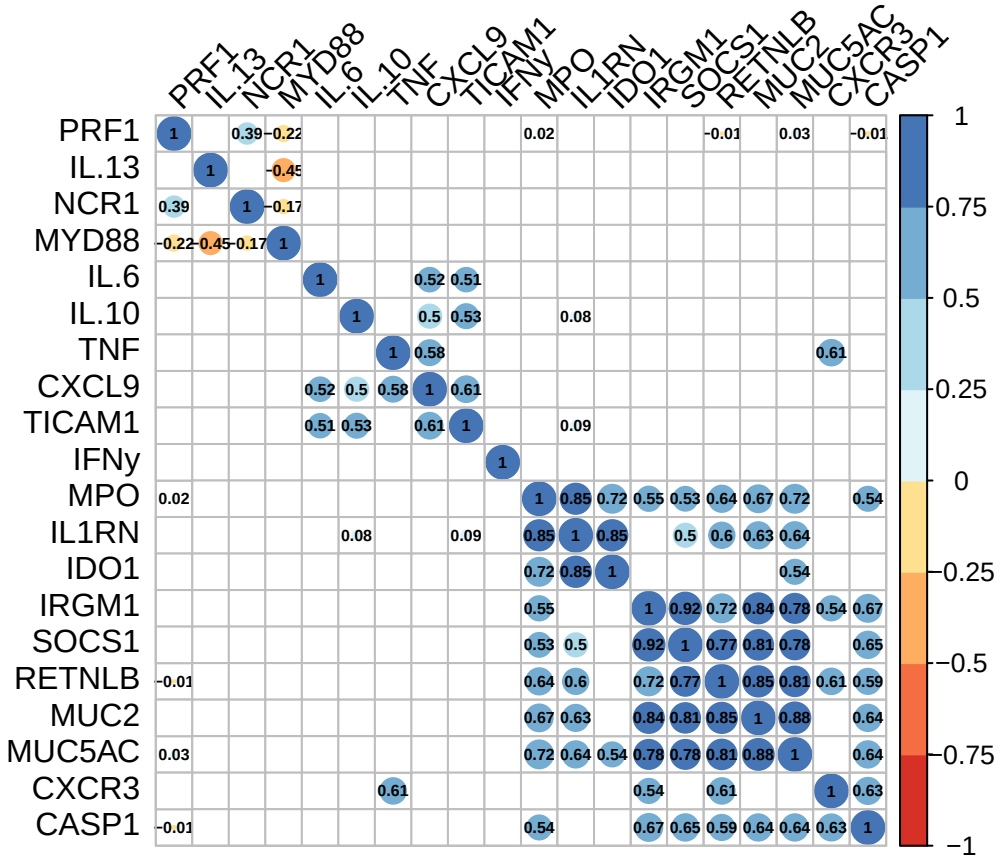



```
## Warning: Ignoring unknown parameters: echo
```



Corrplot of correlations

Here is a corrplot of the correlations between the genes. I am using the non-normalized genes

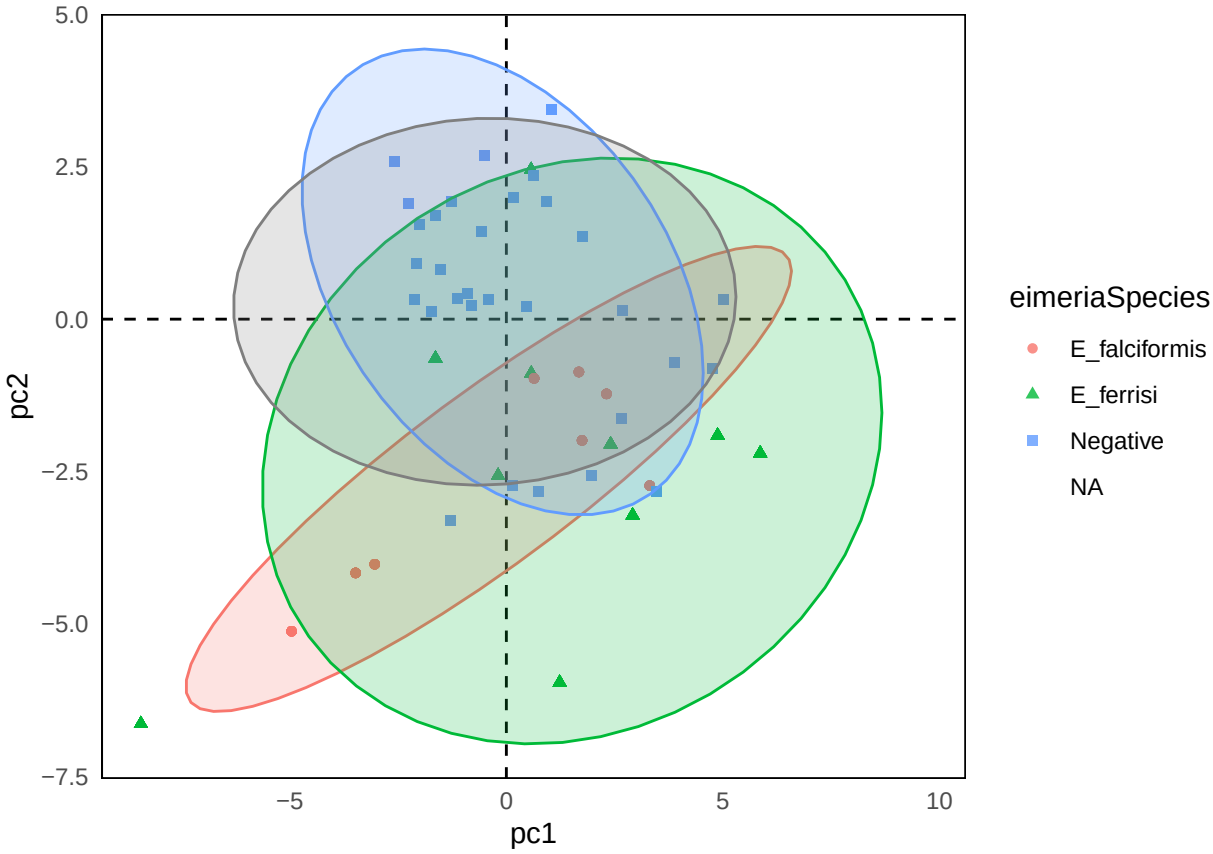


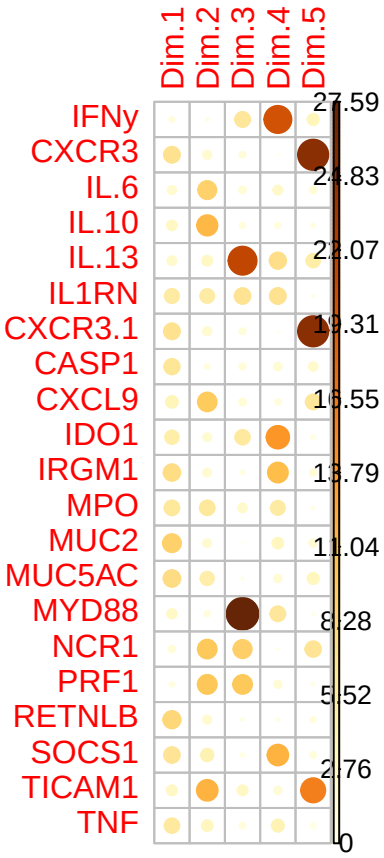

```
res.pca <- PCA(field[,all_of(Genes_field)], scale.unit = TRUE, graph = FALSE)
```

Biplot of the gene pca

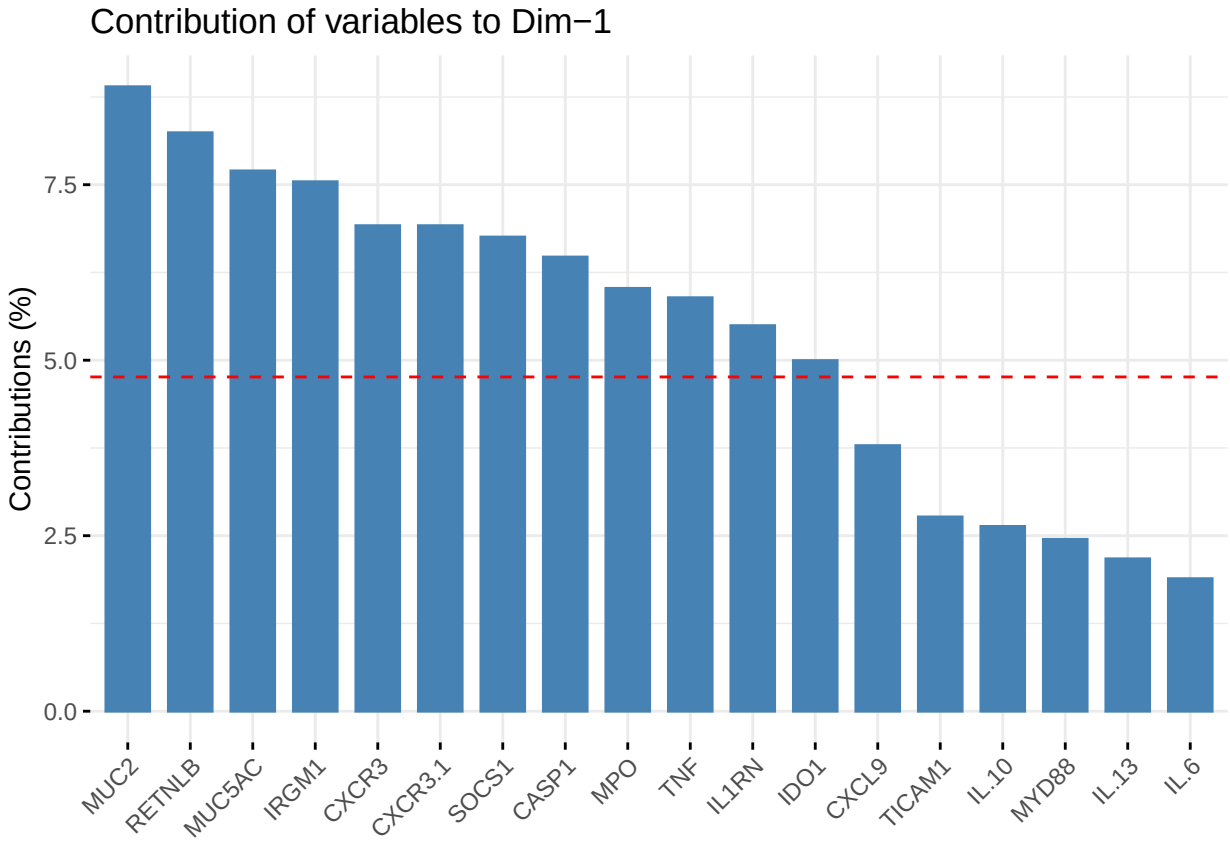
```
#Now we can make our initial plot of the PCA.
field %>%
  pivot_longer(cols = all_of(Genes_field), names_to = "Gene", values_to = "gene_expression")  %>%
  ggplot(aes(x = pc1, y = pc2, color = eimeriaSpecies, shape = eimeriaSpecies)) +
  geom_hline(yintercept = 0, lty = 2) +
  geom_vline(xintercept = 0, lty = 2) +
  geom_point(alpha = 0.8) +
  stat_ellipse(geom="polygon", aes(fill = eimeriaSpecies), alpha = 0.2, show.legend = FALSE,
               level = 0.95) +
  theme_minimal() +
  theme(panel.grid = element_blank(), panel.border = element_rect(fill= "transparent"))
```

```
## Warning: Removed 5680 rows containing missing values (geom_point).
```

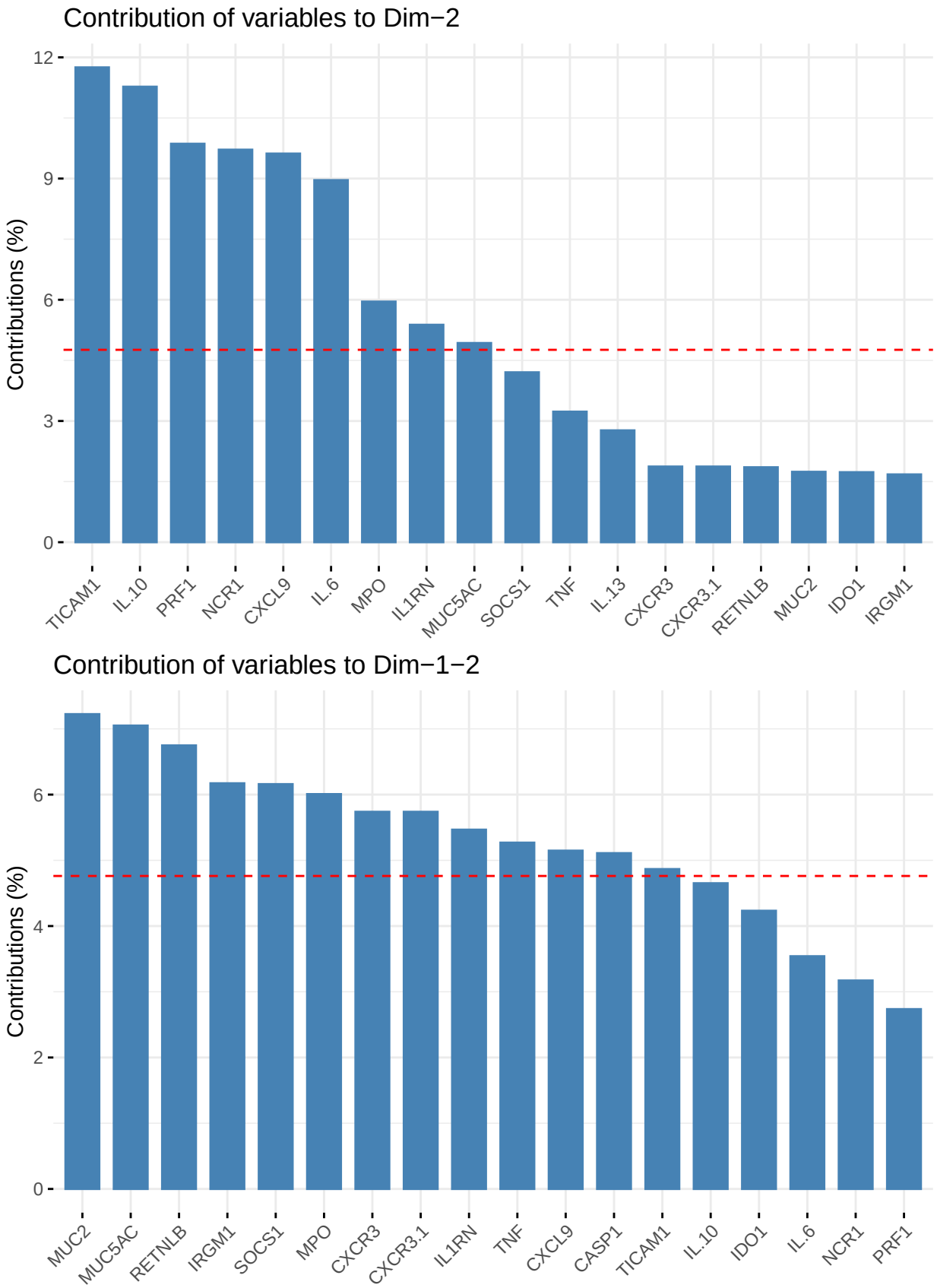

5.0
2.5
0.0
−2.5
−5.0
−7.5
pc2
−5
0
5
10
pc1
eimeriaSpecies
E_falciformis
E_ferrisi
Negative
NA


Dim.1
Dim.2
Dim.3
Dim.4
Dim.5
IFNy
CXCR3
IL.6
IL.10
IL.13
IL1RN
CXCR3.1
CASP1
CXCL9
IDO1
IRGM1
MPO
MUC2
MUC5AC
MYD88
NCR1
PRF1
RETNLB
SOCS1
TICAM1
TNF
27.59
24.83
22.07
19.31
16.55
13.79
11.04
8.28
5.52
2.76
0

The function fviz_contrib() [factoextra package] can be used to draw a bar plot of variable contributions. If your data contains many variables, you can decide to show only the top contributing variables. The R code below shows the top 10 variables contributing to the principal components:




```
# Contributions of variables to PC2
fviz_contrib(res.pca, choice = "var", axes = 2, top = 18)
```

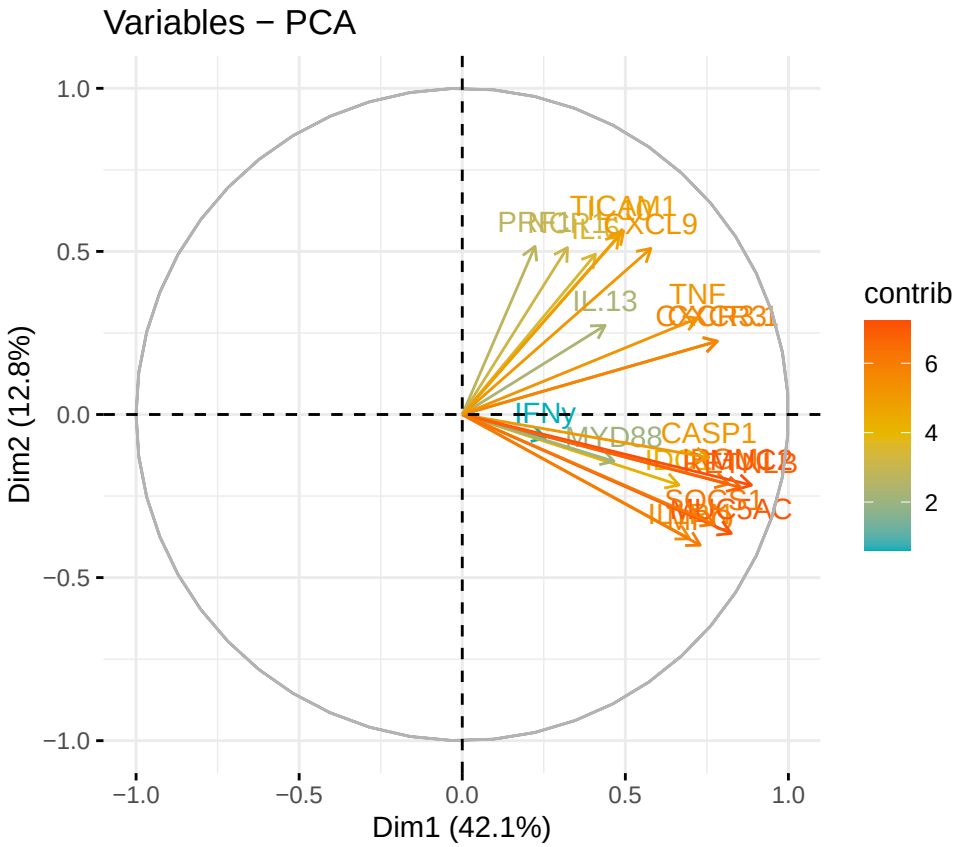


The red dashed line on the graph above indicates the expected average contribution. If the contribution of

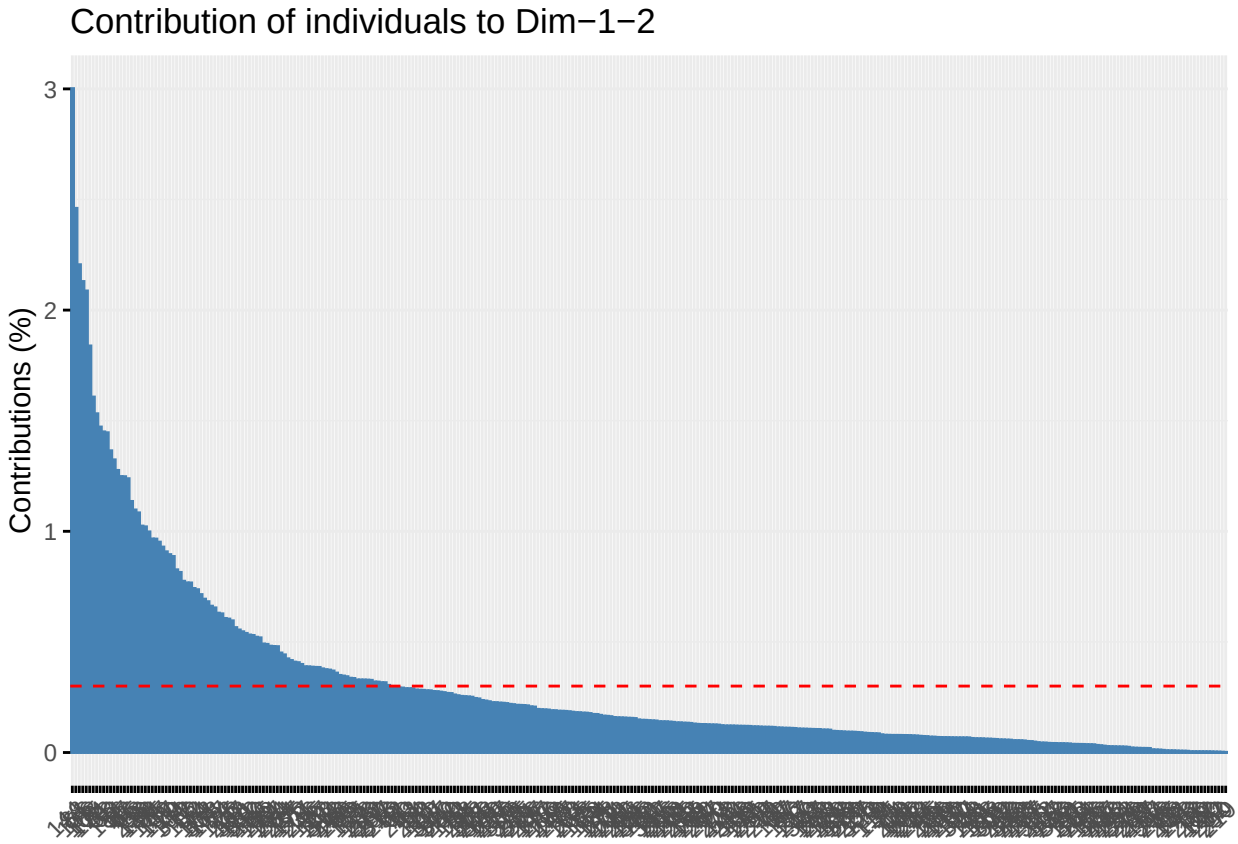
the variables were uniform, the expected value would be 1/length(variables) = 1/10 = 10%. For a given component, a variable with a contribution larger than this cutoff could be considered as important in contributing to the component.

Note that, the total contribution of a given variable, on explaining the variations retained by two principal components, say PC1 and PC2, is calculated as contrib = [(C1 * Eig1) + (C2 * Eig2)]/(Eig1 + Eig2), where

C1 and C2 are the contributions of the variable on PC1 and PC2, respectively Eig1 and Eig2 are the eigenvalues of PC1 and PC2, respectively. Recall that eigenvalues measure the amount of variation retained by each PC. In this case, the expected average contribution (cutoff) is calculated as follow: As mentioned above, if the contributions of the 10 variables were uniform, the expected average contribution on a given PC would be 1/10 = 10%. The expected average contribution of a variable for PC1 and PC2 is : [(10* Eig1) + (10 * Eig2)]/(Eig1 + Eig2)



To visualize the contribution of individuals to the first two principal components:

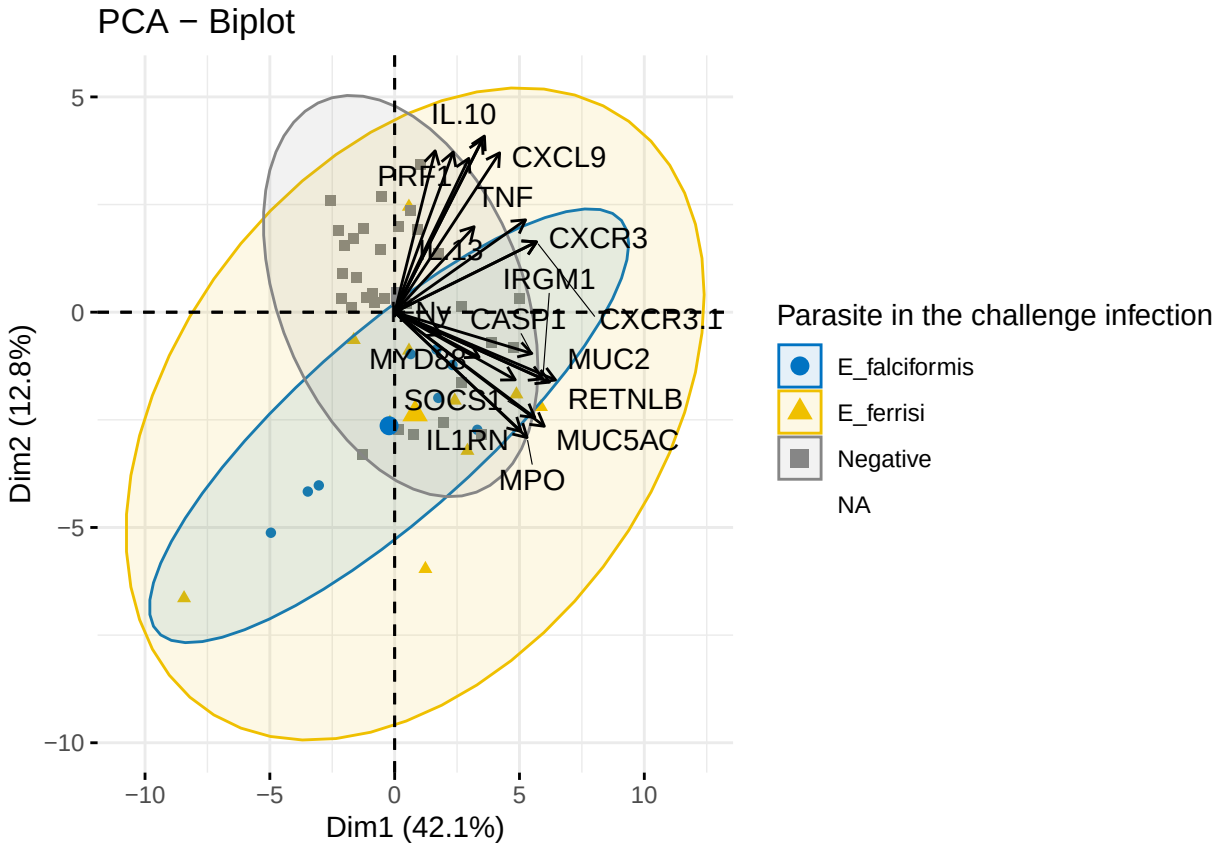


PCA + Biplot combination

```
## Warning: Removed 284 rows containing missing values (geom_point).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

```
## Warning: ggrepel: 4 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



In the following example, we want to color both individuals and variables by groups. The trick is to use pointshape = 21 for individual points. This particular point shape can be filled by a color using the argument fill.ind. The border line color of individual points is set to "black" using col.ind. To color variable by groups, the argument col.var will be used.