

5. PCA genes - field

Fay

2022-10-08

Load libraries

```
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr  0.3.5
## v tibble  3.1.8      v dplyr  1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(dplyr)
library(stringr)
library(FactoMineR)
library(reshape2)

##
## Attaching package: 'reshape2'
##
## The following object is masked from 'package:tidyr':
##
##     smiths

library(corrplot)

## corrplot 0.92 loaded

library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(lmtest)

## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

library(ggpubr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##   chisq.test, fisher.test
library(pheatmap)
```

Load data

```
hm <- read.csv("output_data/2.imputed_MICE_data_set.csv")
```

vectors for selecting

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

#add a suffix to represent changes in data file
Gene_field_imp <- paste(Gene_field, "imp", sep = "_")

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

Genes_wild_imp <- paste(Genes_wild, "imp", sep = "_")

Facs_field <- c("Position", "CD4", "Treg", "Div_Treg", "Treg17", "Th1",
               "Div_Th1", "Th17", "Div_Th17", "CD8", "Act_CD8",
               "Div_Act_CD8", "IFNy_CD4", "IFNy_CD8", "Treg_prop",
               "IL17A_CD4")

Facs_field_imp <- paste(Facs_field, "imp", sep = "_")

Facs_wild <- c("Treg", "CD4", "Treg17", "Th1", "Th17", "CD8",
               "Act_CD8", "IFNy_CD4", "IL17A_CD4", "IFNy_CD8")

Facs_wild_imp <- paste(Facs_wild, "imp", sep = "_")
```

PCA on the field genes *-imputed*

```
# somehow the field samples have the origin na,
# fix that
hm$origin[is.na(hm$origin)] <- "Field"

field <- hm %>%
  dplyr::filter(origin == "Field")
```

```

field <- unique(field)

#make a factor out of the melting curves (important for later visualization)
field <- field %>%
  dplyr::mutate(MC.Eimeria = as.factor(MC.Eimeria))

genes_mouse <- field %>%
  dplyr::select(c(Mouse_ID, all_of(Genes_wild)))

genes <- genes_mouse %>%
  dplyr::select(-Mouse_ID)

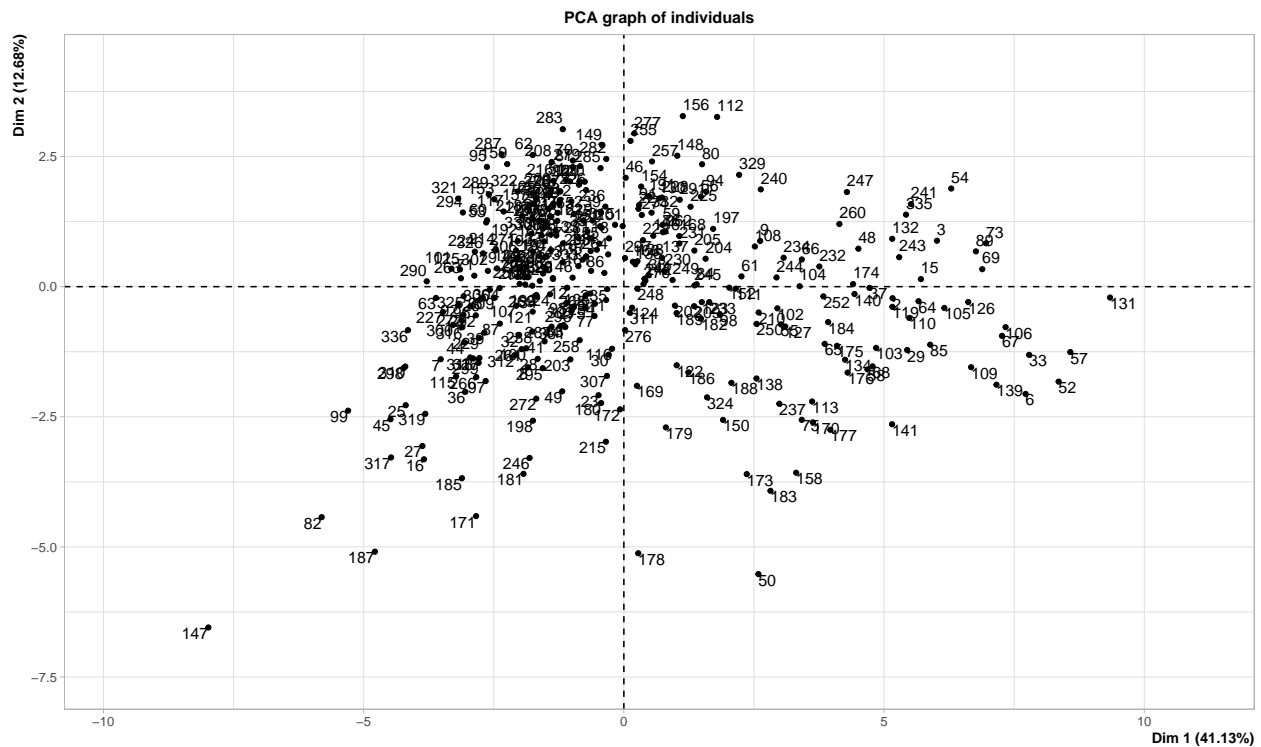
#remove rows with only nas
genes <- genes[,colSums(is.na(genes))<nrow(genes)]

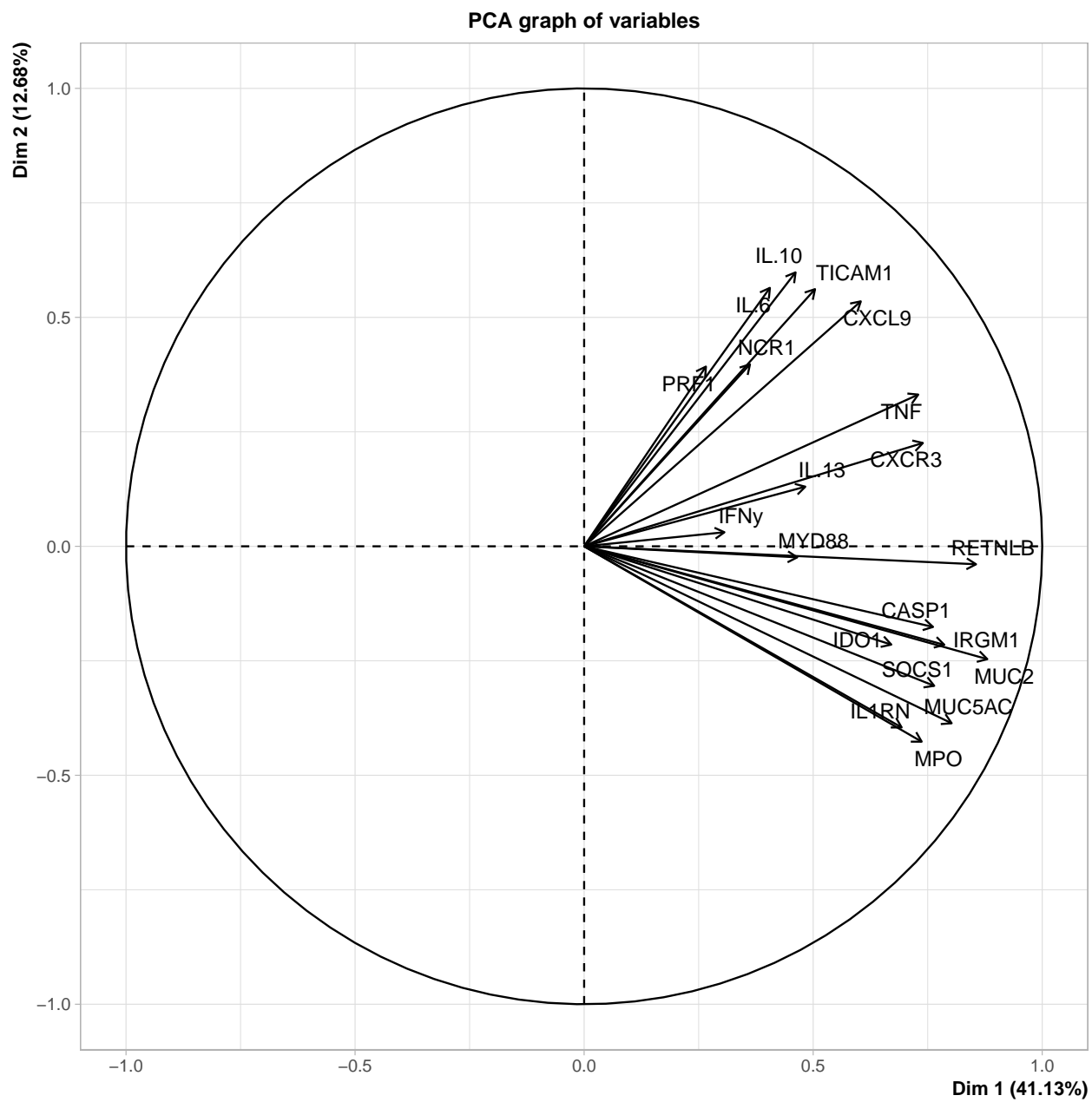
#remove columns with only nas
genes <- genes[rowSums(is.na(genes)) != ncol(genes), ]

##select same rows in the first table
field <- field[row.names(genes), ]

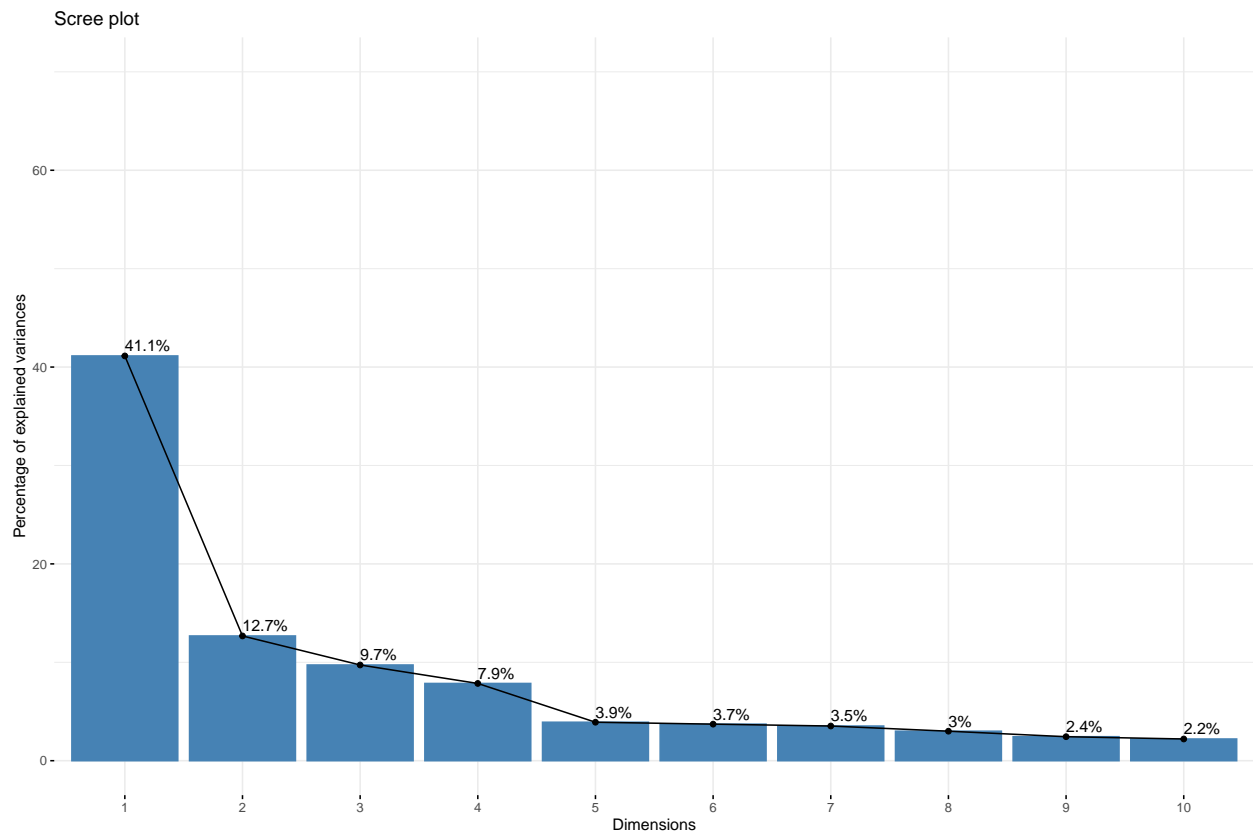
# we can now run a normal pca on the complete data set
res.pca <- PCA(genes)

```

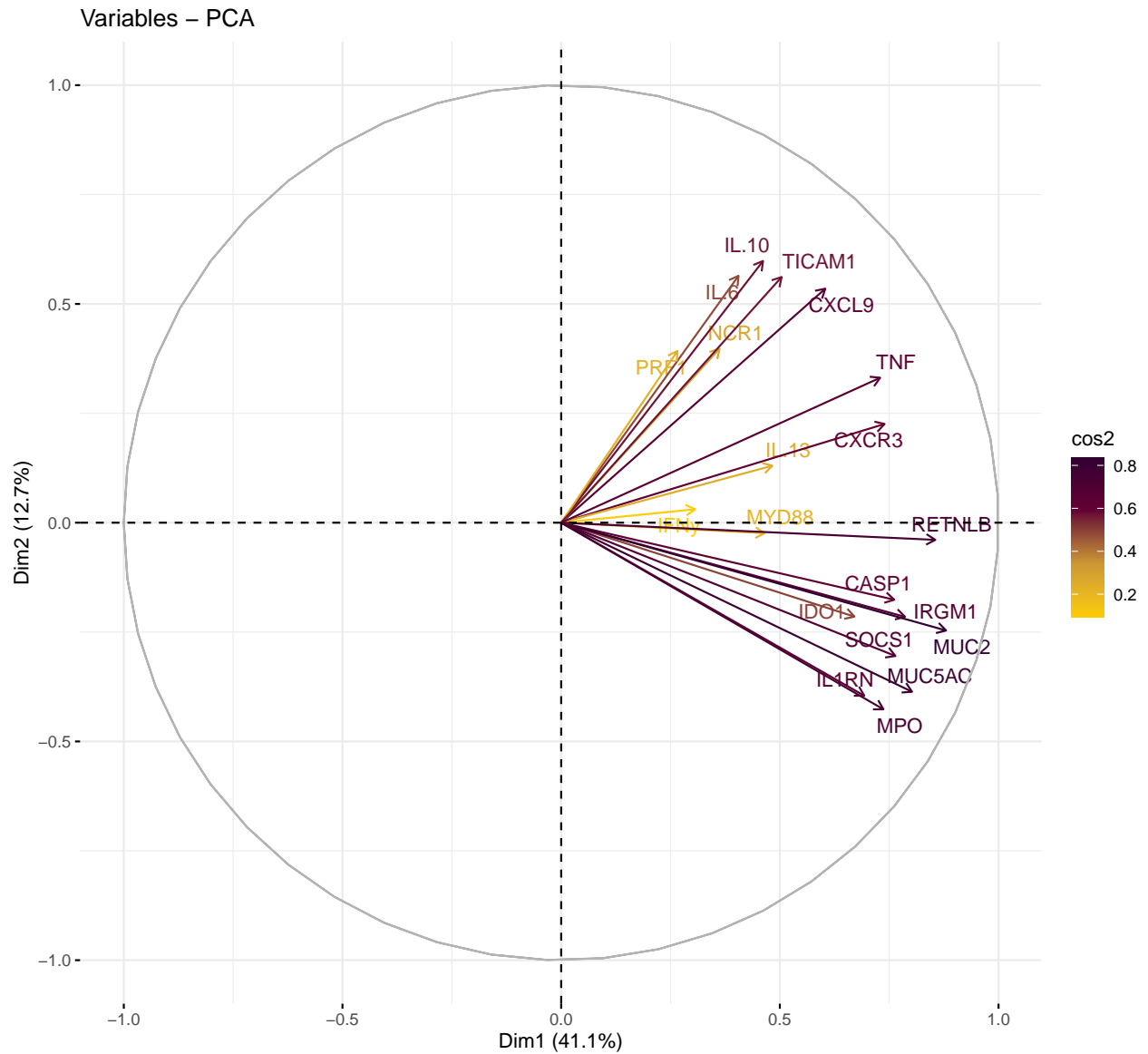




```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 70))
```

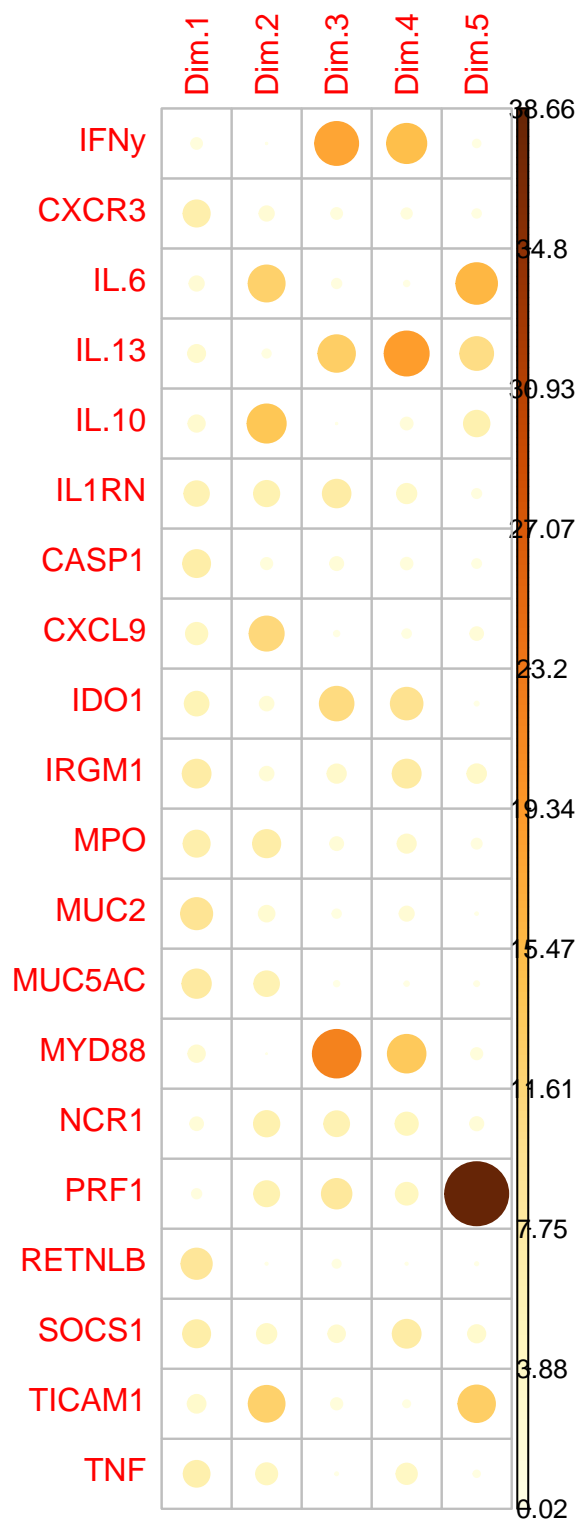


```
fviz_pca_var(res.pca, col.var = "cos2",  
              gradient.cols = c("#FFCC00", "#CC9933", "#660033", "#330033"),  
              repel = TRUE)
```

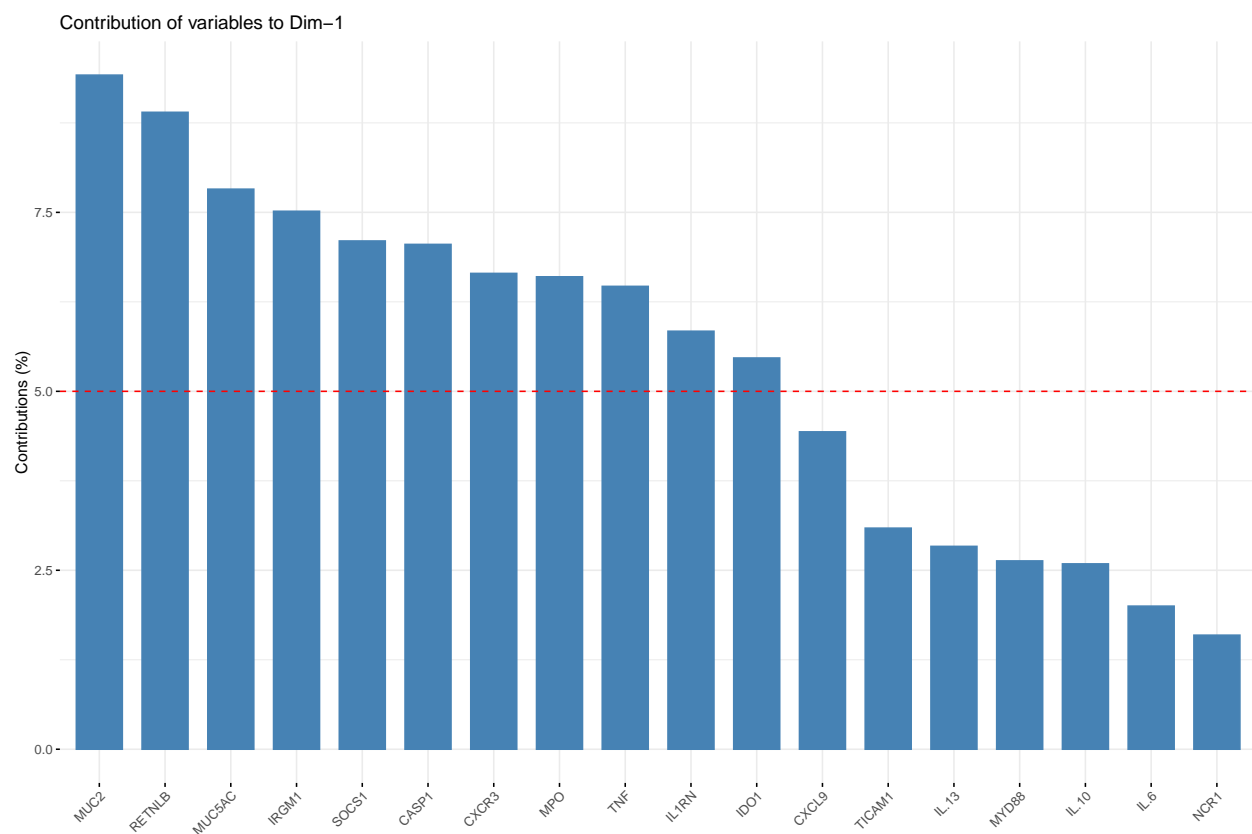


```
fviz_pca_ind(res.pca, col.ind = "cos2",
             gradient.cols = c("#FFCC00", "#CC9933", "#660033", "#330033"),
             repel = TRUE)
```

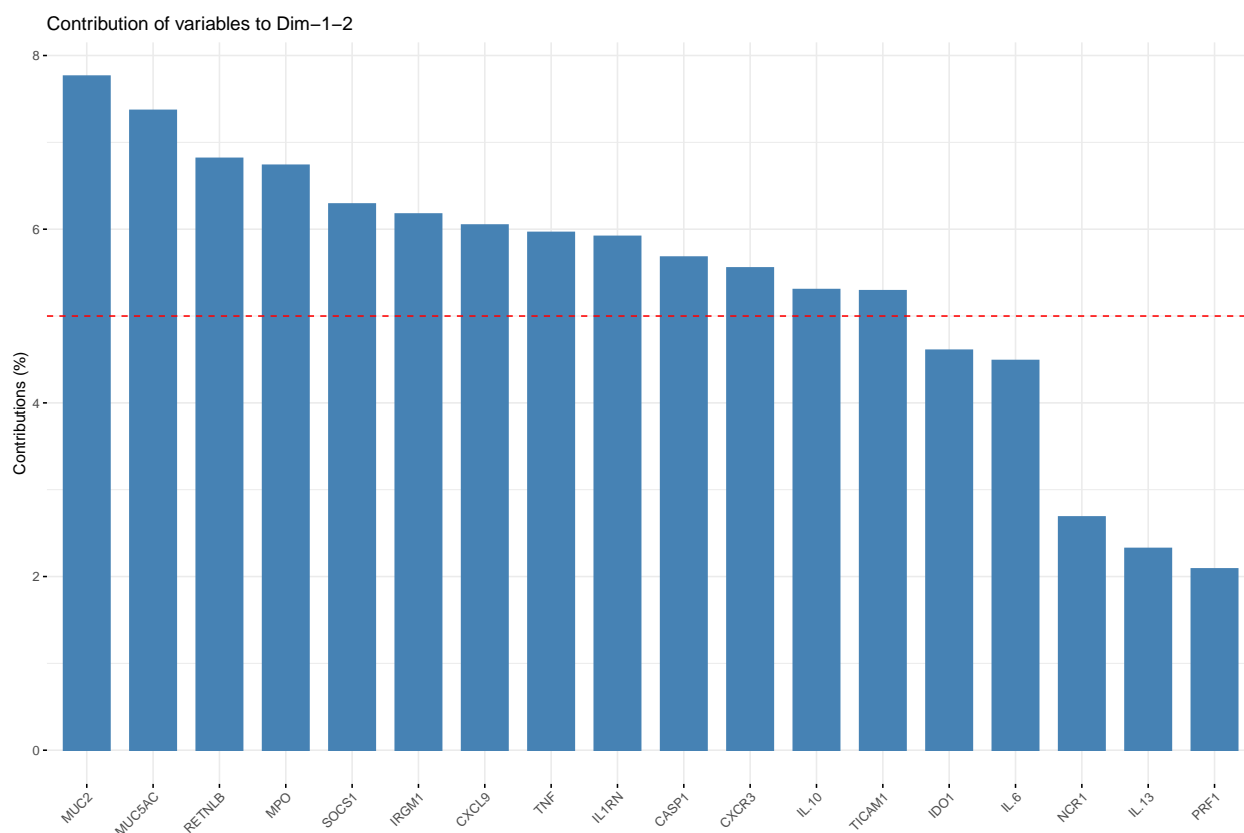
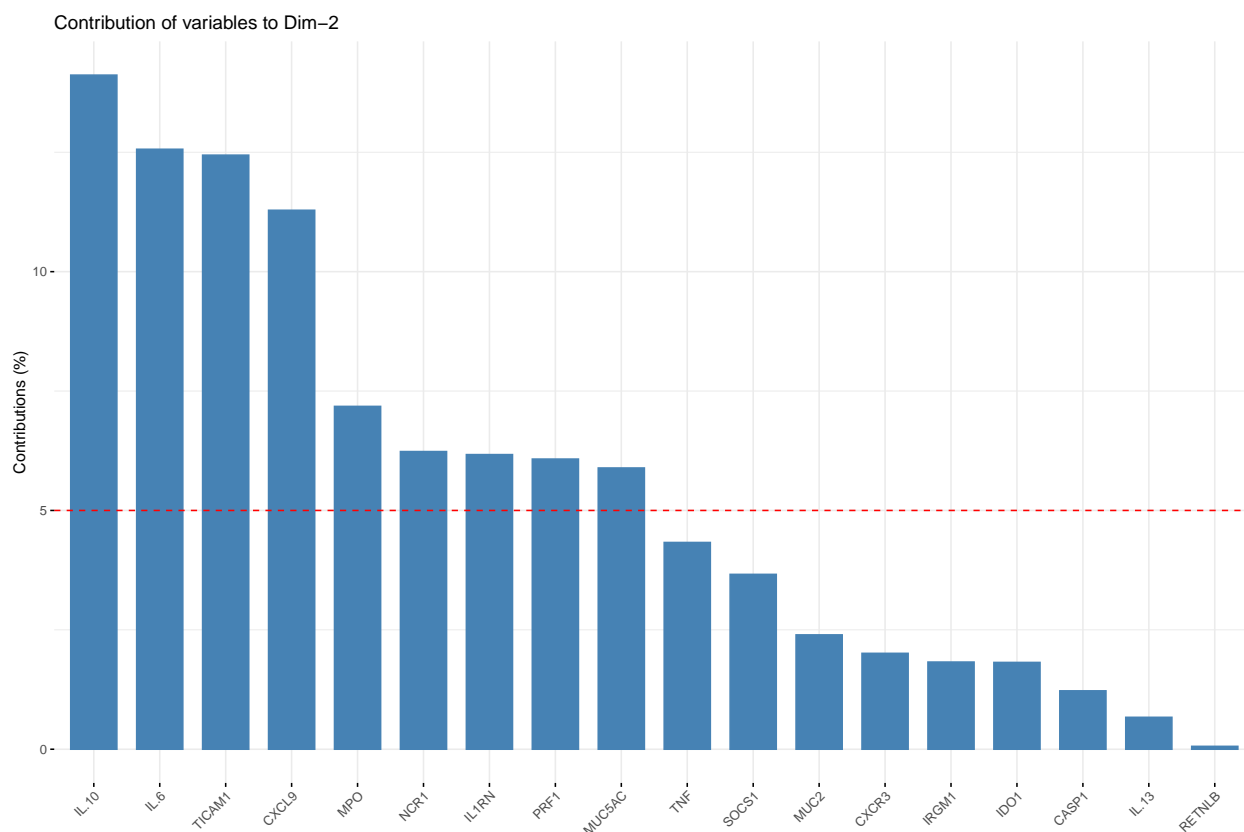
```
## Warning: ggrepel: 132 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

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)
```

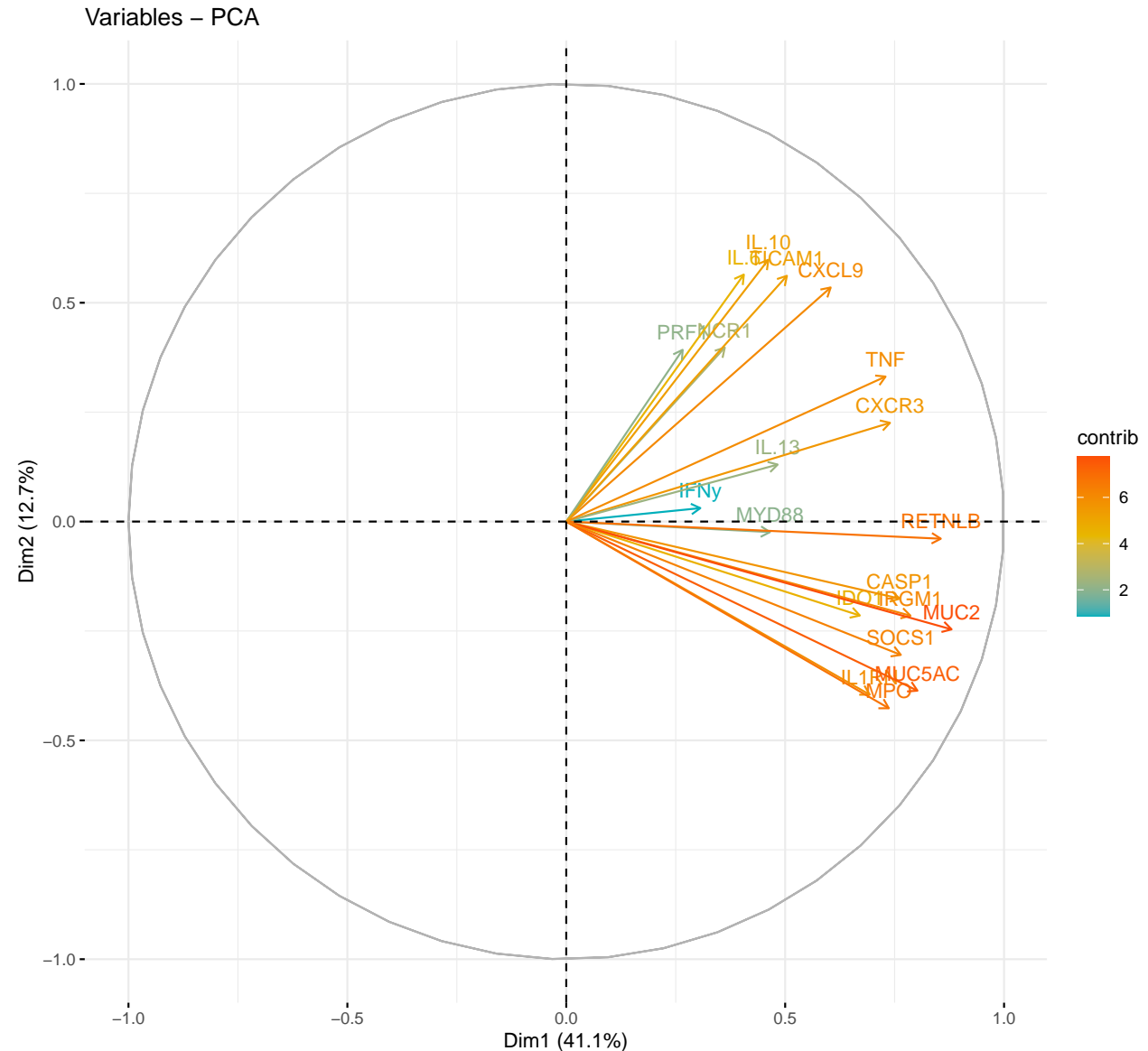


The red dashed line on the graph above indicates the expected average contribution. If the contribution of the variables were uniform, the expected value would be $1/\text{length}(\text{variables}) = 1/10 = 10\%$. For a given

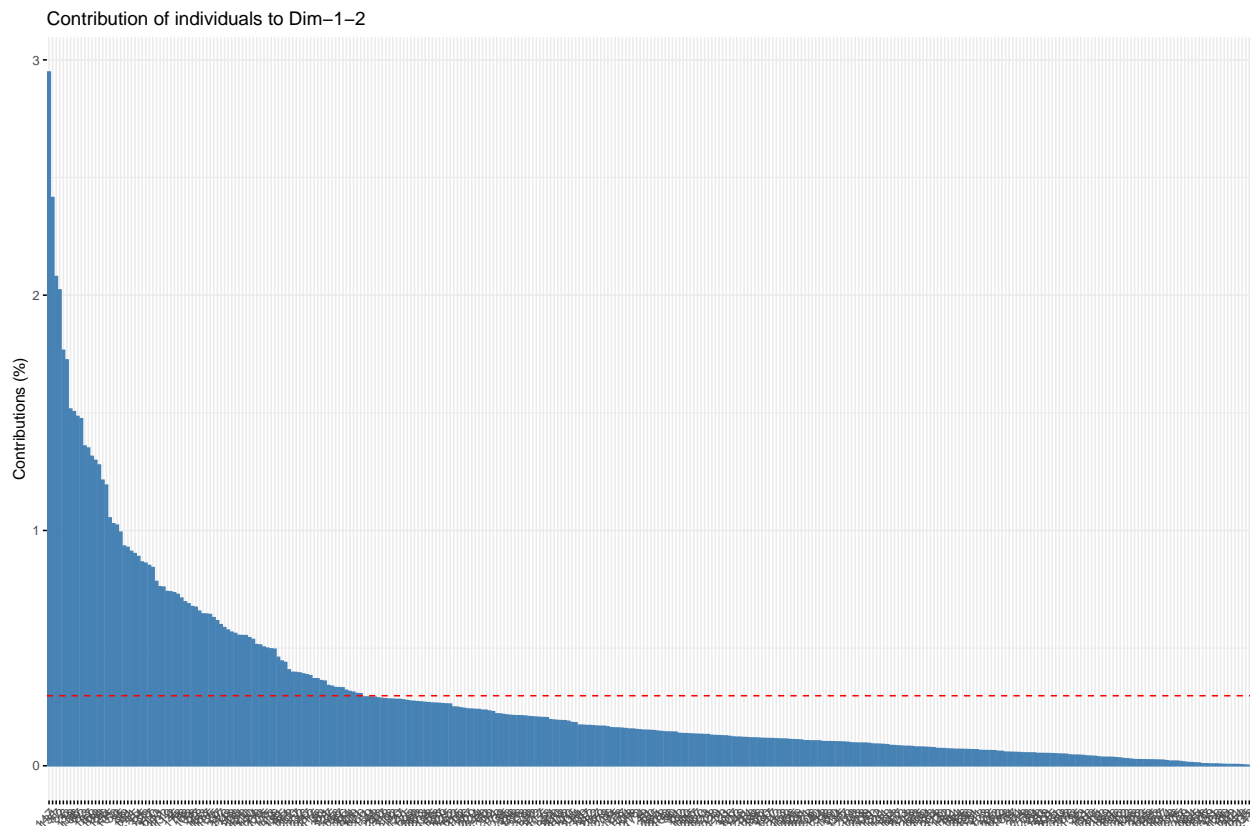
component, a variable with a contribution larger than this cutoff could be considered as important in contributing to the component.

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 $\text{contrib} = [(C1 * \text{Eig1}) + (C2 * \text{Eig2})]/(\text{Eig1} + \text{Eig2})$, 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 : $[(10 * \text{Eig1}) + (10 * \text{Eig2})]/(\text{Eig1} + \text{Eig2})$

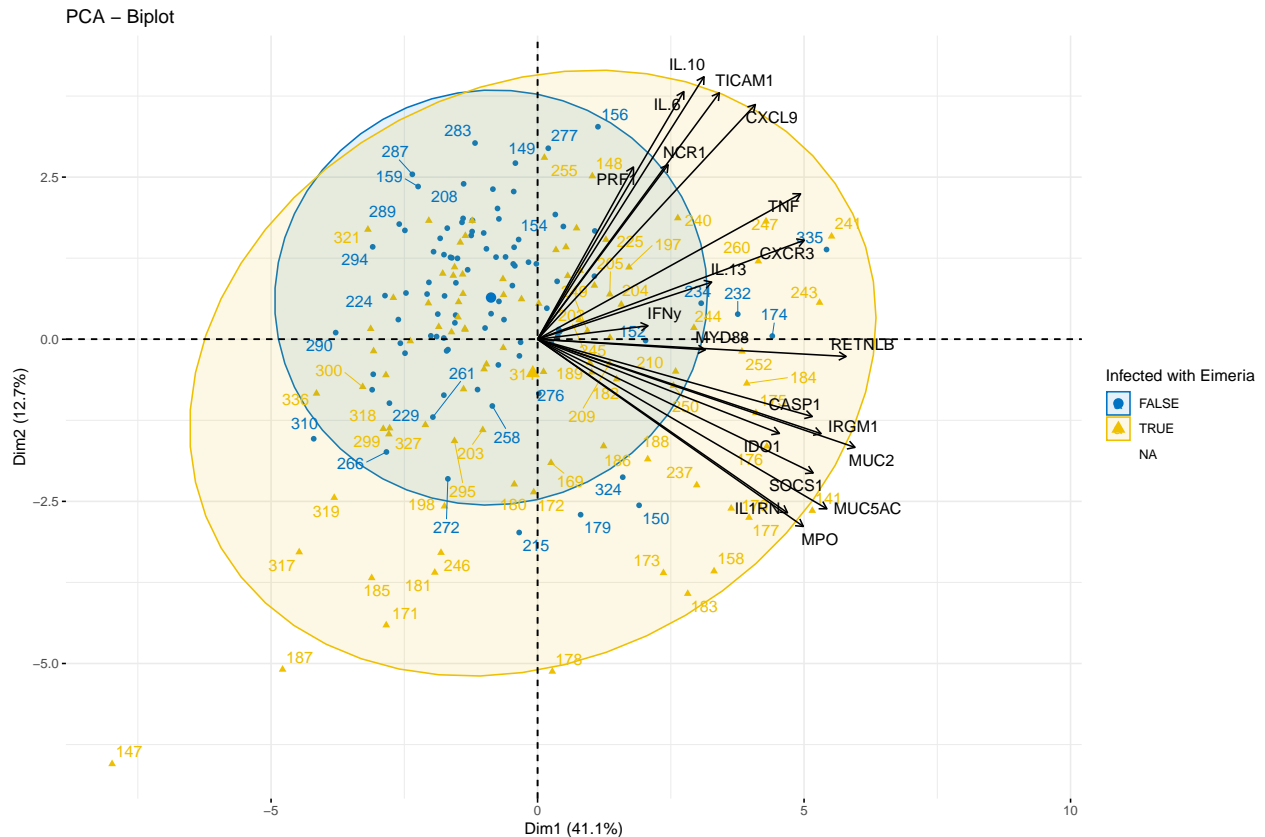


To visualize the contribution of individuals to the first two principal components:



PCA + Biplot combination

```
## Warning: Removed 151 rows containing missing values (`geom_point()`).  
## Warning: Removed 1 rows containing missing values (`geom_point()`).  
## Warning: ggrepel: 160 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.

repeating the heatmap on the now imputed data

```
genes_mouse <- genes_mouse %>%
  rename_with(~str_remove(., '_imp'))

#select same rows in the first table
genes_mouse <- genes_mouse[row.names(genes), ]

# turn the data frame into a matrix and transpose it. We want to have each cell
# type as a row name
genes_mouse <- t(as.matrix(genes_mouse))

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

heatmap_data <- as.data.frame(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
annotation_df <- as_tibble(field) %>%
  dplyr::select(c("Mouse_ID", "HI",
                 "Sex"))

annotation_df <- unique(annotation_df)

annotation_df <- as.data.frame(annotation_df)

### 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)

#remove the unnecessary column
annotation_df <- annotation_df %>% dplyr::select(-Mouse_ID)

```

Heatmap on gene expression data:

