

5. PCA genes - Lab

Fay

2022-10-08

Always change the knitting directory to the working directory! # Load libraries

```
library(tidyverse)
library(dplyr)
library(stringr)
library(FactoMineR)
library(reshape2)
library(corrplot)
library(factoextra)
library(lmtest)
library(ggpubr)
library(janitor)
library(pheatmap)
library(visdat)
```

Load data

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

vectors for selecting

```
Gene_lab <- c("IFNy", "CXCR3", "IL.6", "IL.13",
              "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_lab_imp <- paste(Gene_lab, "imp", sep = "_")

Genes_wild <- c("IFNy", "CXCR3", "IL.6", "IL.13",
               "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_lab <- 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_lab_imp <- paste(Facs_lab, "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 lab genes -*imputed*

```

#select the genes and lab muce
lab <- hm %>%
  dplyr::filter(origin == "Lab", Position == "mLN") #selecting for mln to avoid
# duplicates

lab <- unique(lab)

gene <- lab %>%
  dplyr::select(c(Mouse_ID, all_of(Gene_lab)))

genes <- unique(gene)

genes <- genes[, -1]

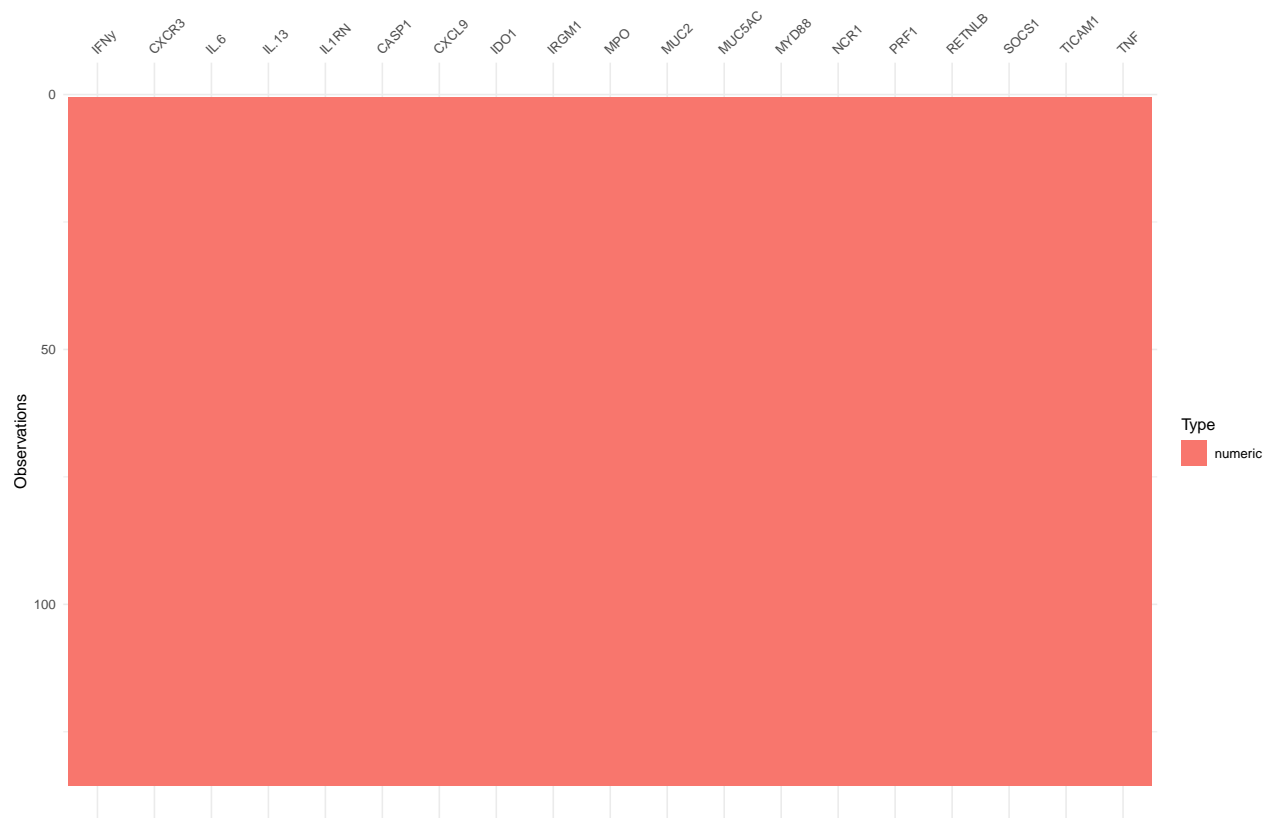
#remove rows with only nas
genes <- genes[, colSums(is.na(genes)) < nrow(genes)]

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

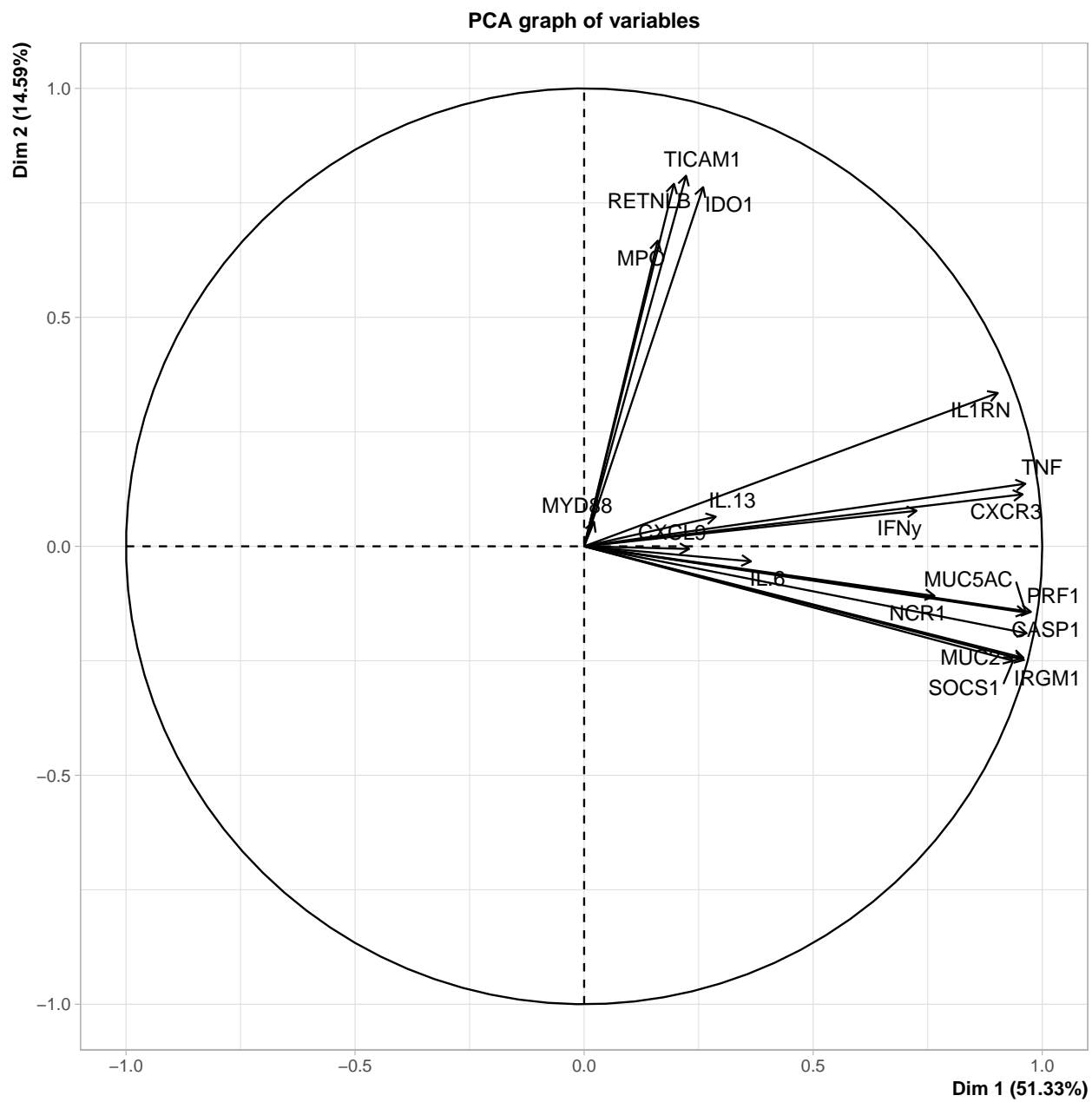
vis_dat(genes)

## Warning: `gather()` was deprecated in tidyr 1.2.0.
## i Please use `gather()` instead.
## i The deprecated feature was likely used in the visdat package.
##   Please report the issue at <https://github.com/ropensci/visdat/issues>.

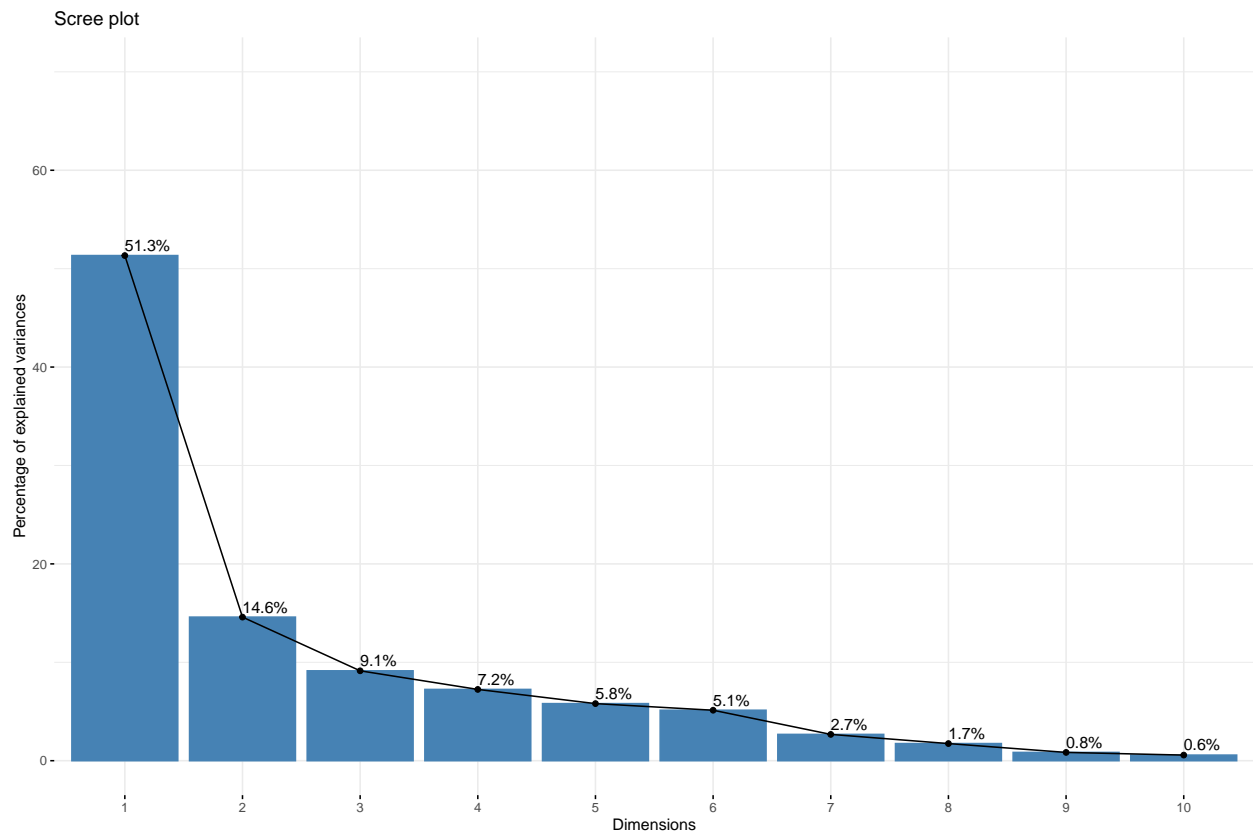
```



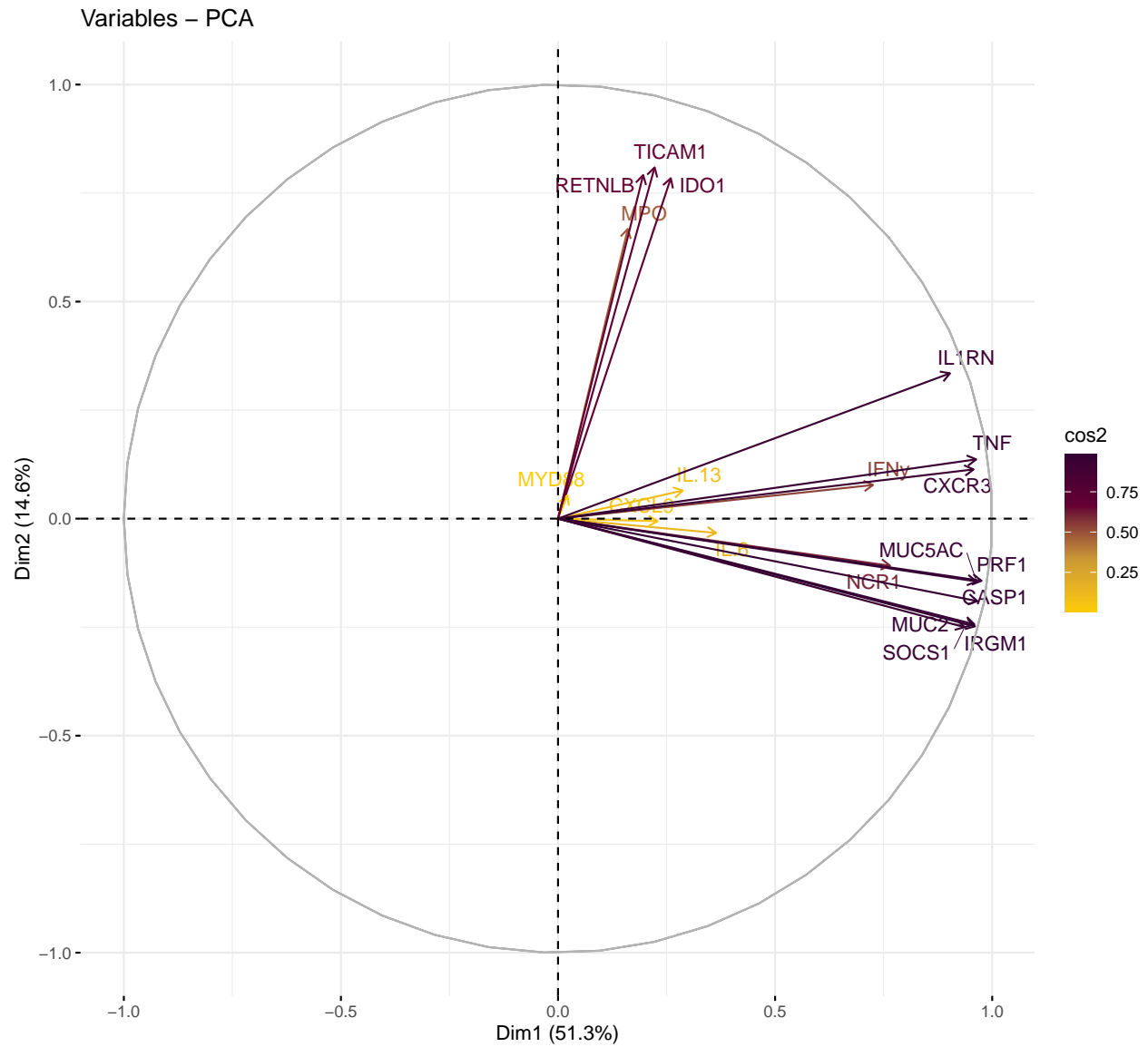
```
#select same rows in the first table  
gene <- gene[row.names(genes), ]  
  
# we can now run a normal pca on the complete data set  
res.pca <- PCA(genes)
```



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

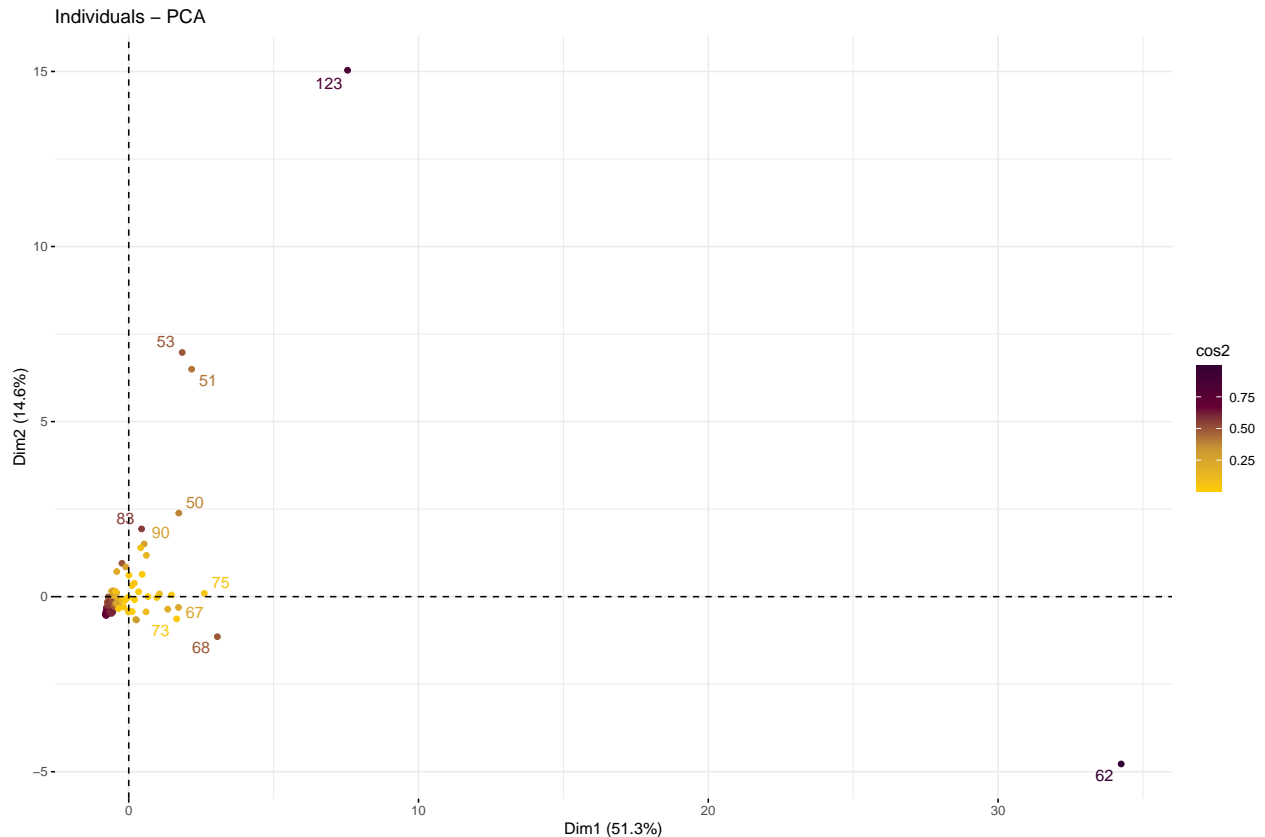


```
fviz_pca_var(res.pca, col.var = "cos2",  
              gradient.cols = c("#FFCC00", "#CC9933", "#660033", "#330033"),  
              repel = TRUE)
```



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

```
## Warning: ggrepel: 124 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Caution: When imputing data, the percentages of inertia associated with the first dimensions will be overestimated.

Another problem: the imputed data are, when the pca is performed considered like real observations. But they are estimations!!

Visualizing uncertainty due to missing data:

→ multiple imputation: generate several plausible values for each missing data point

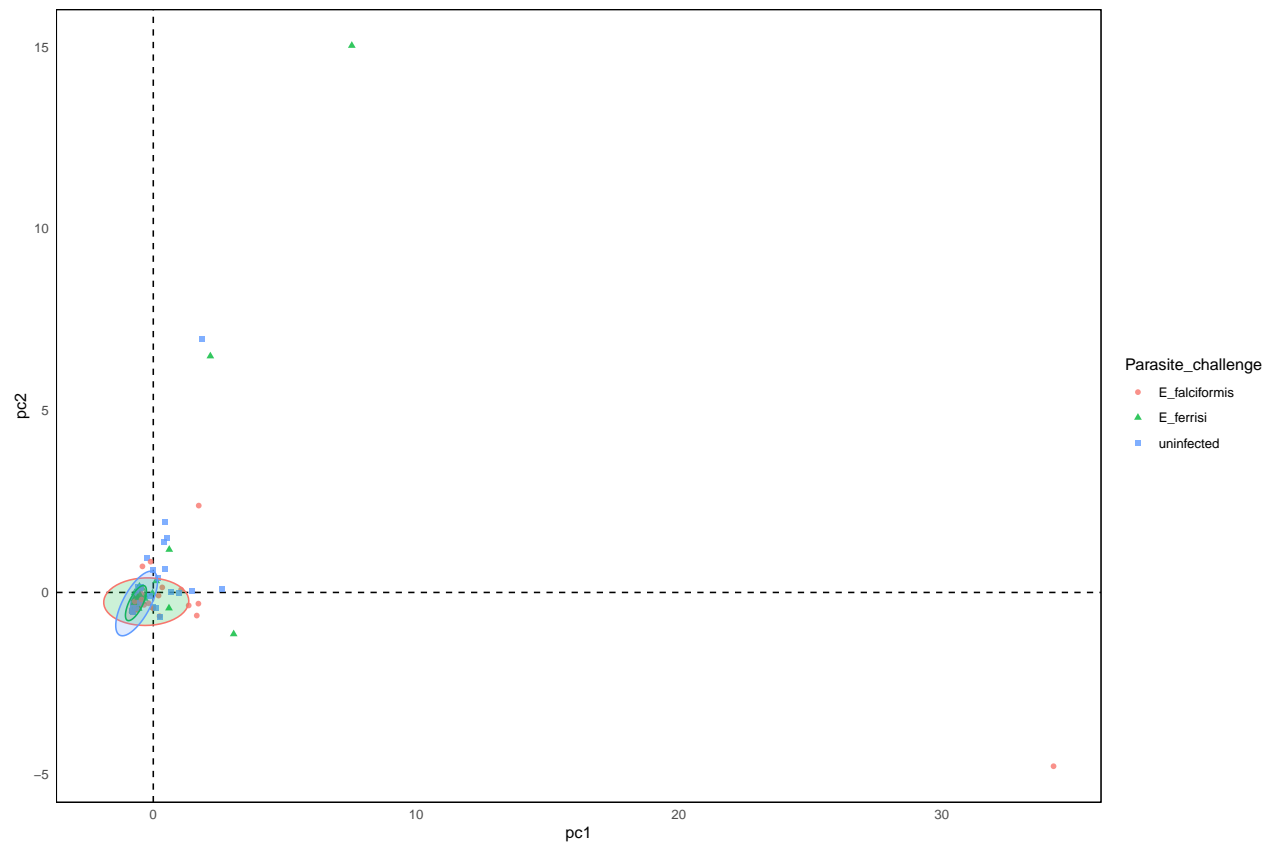
We here visualize the variability, that is uncertainty on the plane defined by two pca axes.

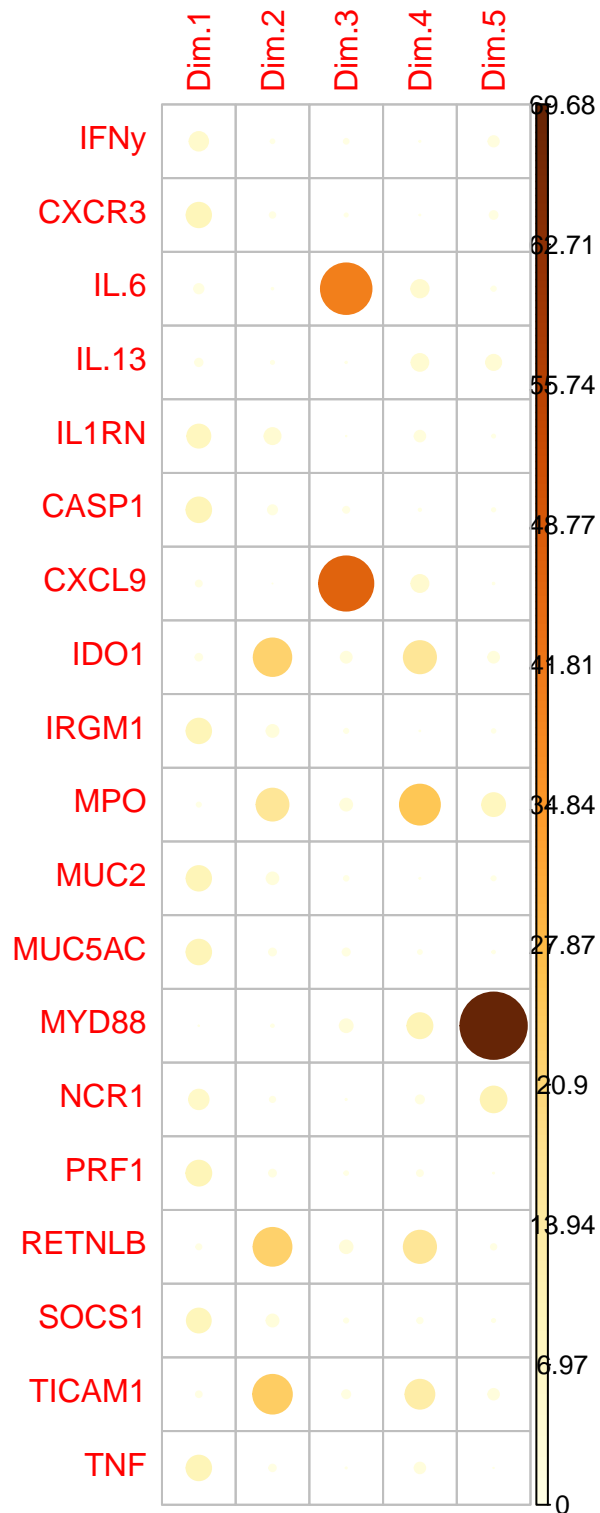
Biplot of the imputed gene pca

#Now we can make our initial plot of the PCA.

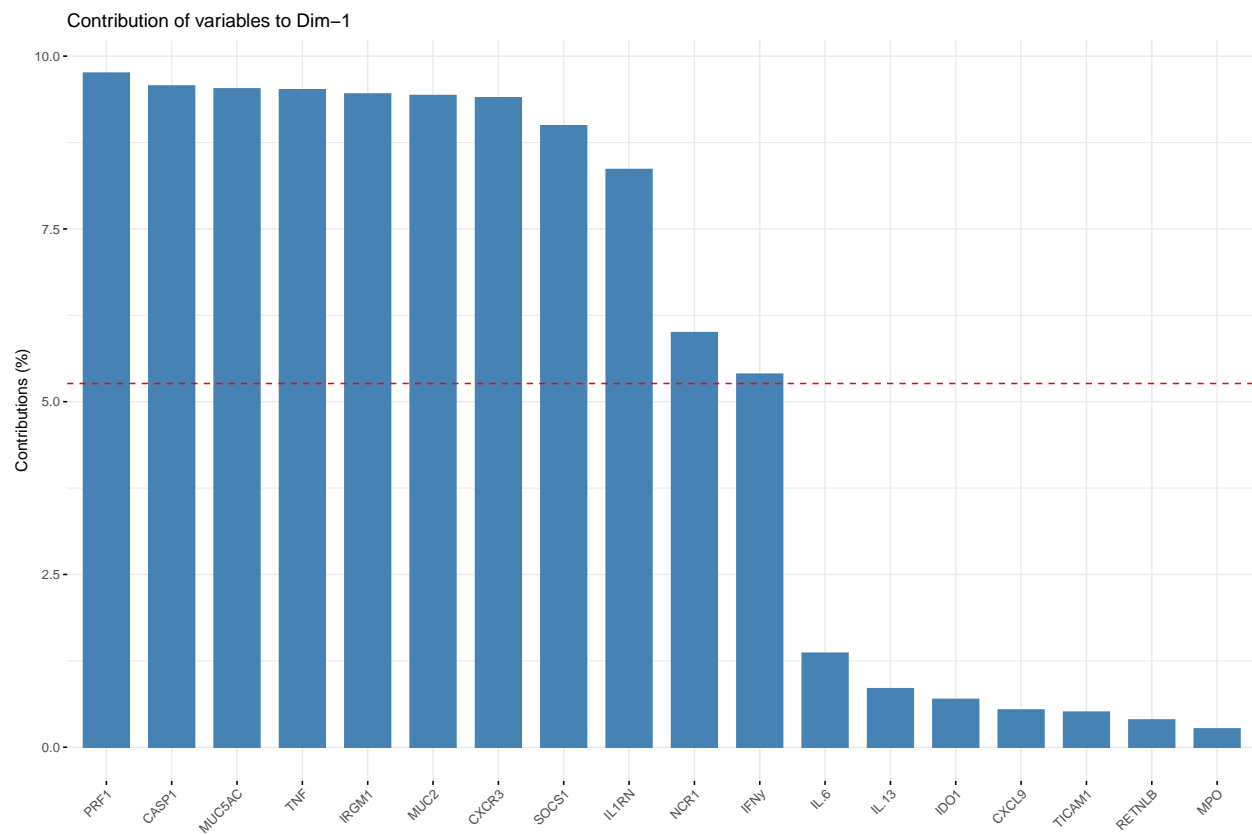
```
lab %>%
  ggplot(aes(x = pc1, y = pc2,
             color = Parasite_challenge,
             shape = Parasite_challenge)) +
  geom_hline(yintercept = 0, lty = 2) +
  geom_vline(xintercept = 0, lty = 2) +
  geom_point(alpha = 0.8) +
  stat_ellipse(geom="polygon",
              aes(fill = challenge_infection),
              alpha = 0.2, show.legend = FALSE,
              level = 0.95) +
  theme_minimal() +
  theme(panel.grid = element_blank(),
        panel.border = element_rect(fill= "transparent"))
```

```
## Warning: Removed 5 rows containing non-finite values (`stat_ellipse()`).  
## Warning: Removed 5 rows containing missing values (`geom_point()`).
```

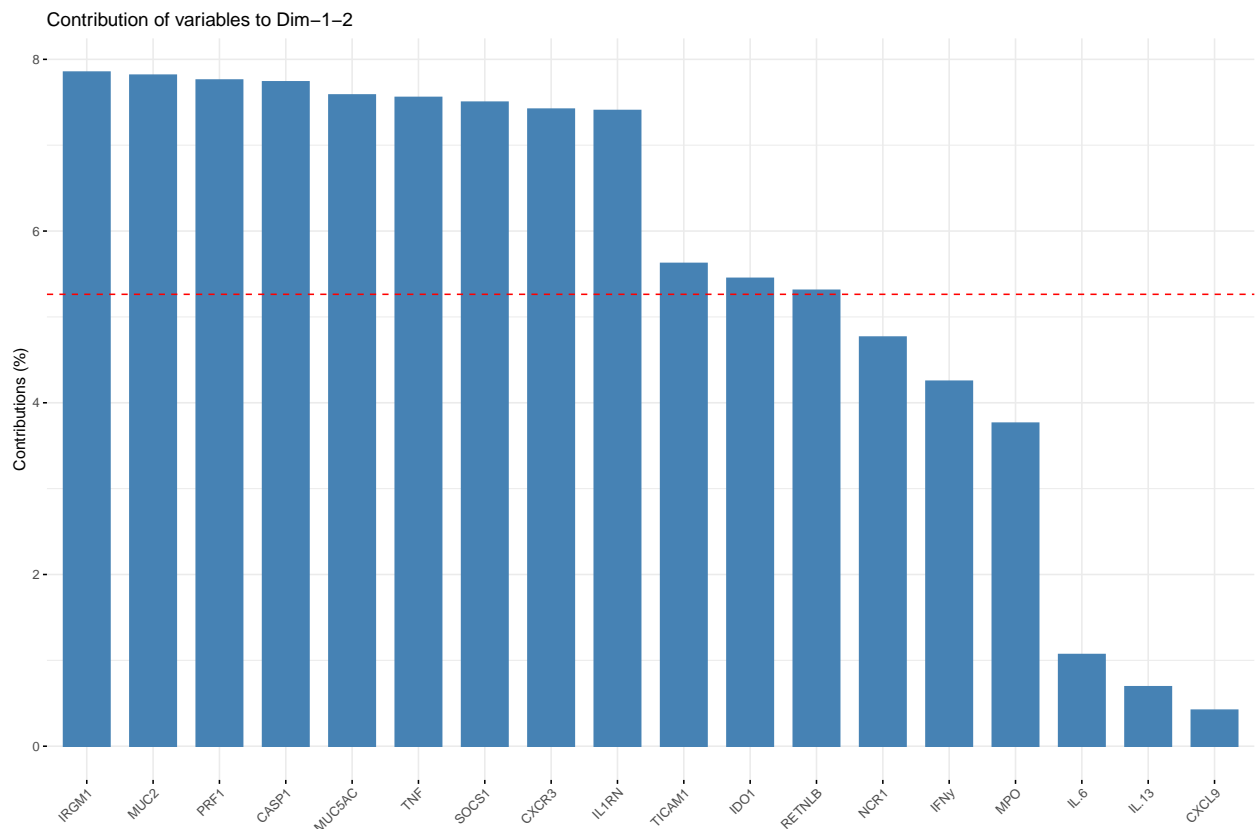
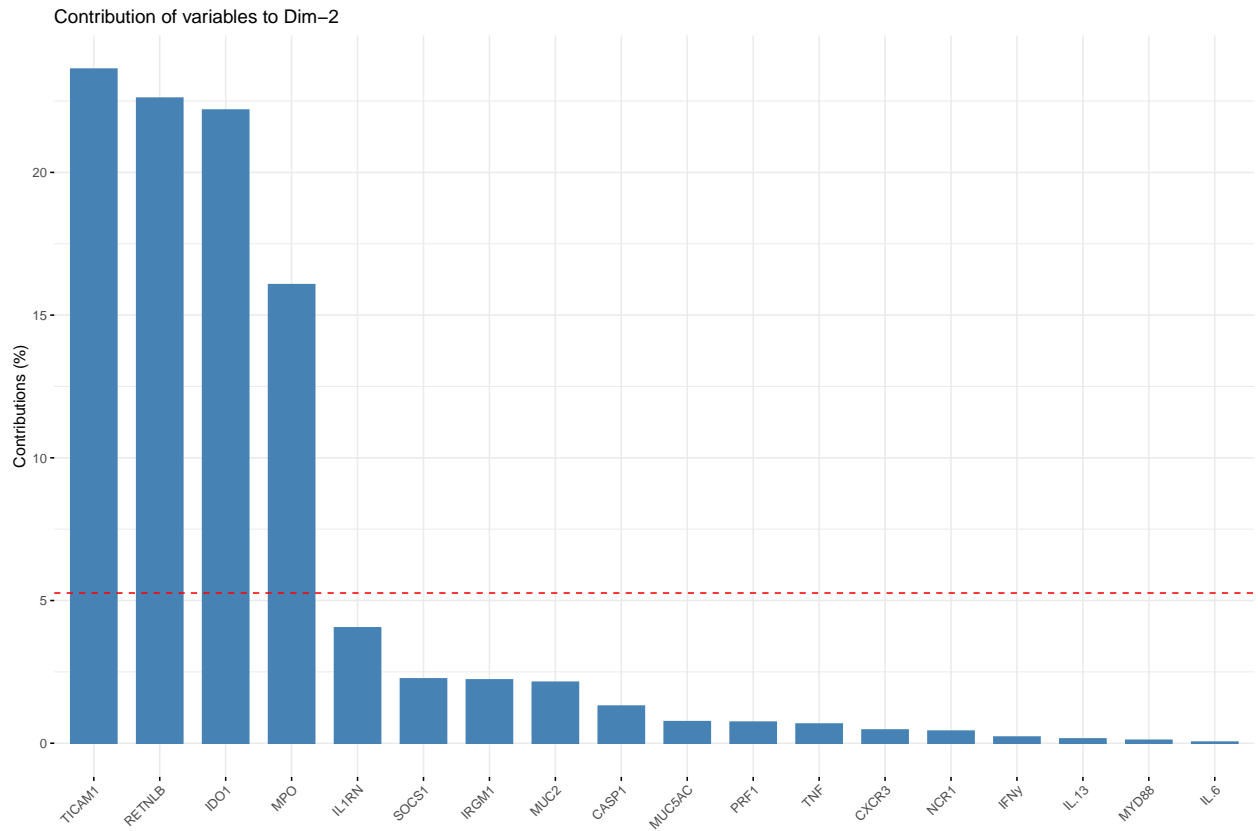




The function `fviz_contrib()` [factoextra package] can be used to draw a bar plot of variable contributions. If your data contains many variables, you can decide to show only the top contributing variables. The R code below shows the top 10 variables contributing to the principal components:



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

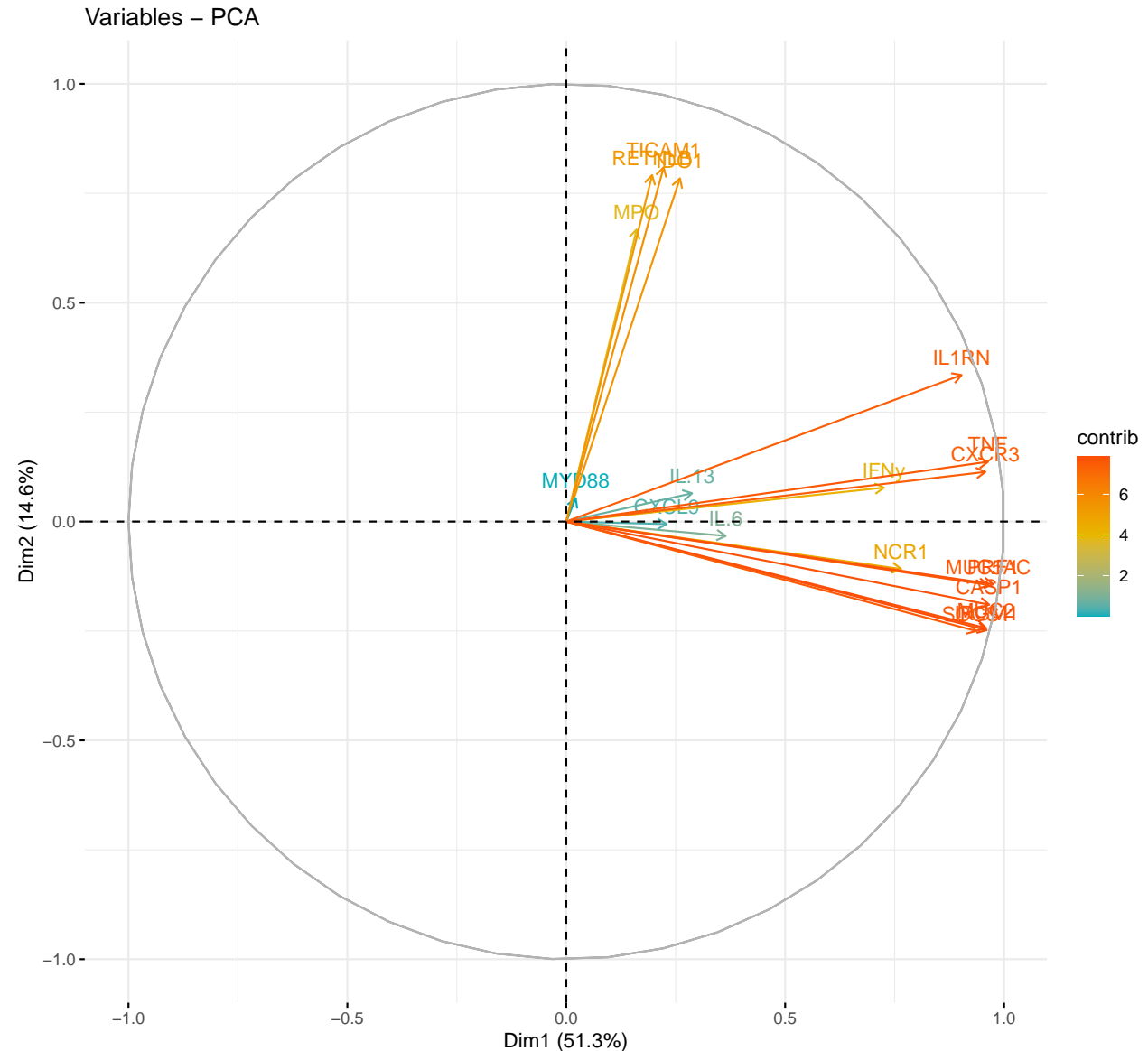


The red dashed line on the graph above indicates the expected average contribution. If the contribution of the variables were uniform, the expected value would be $1/\text{length}(\text{variables}) = 1/10 = 10\%$. For a given

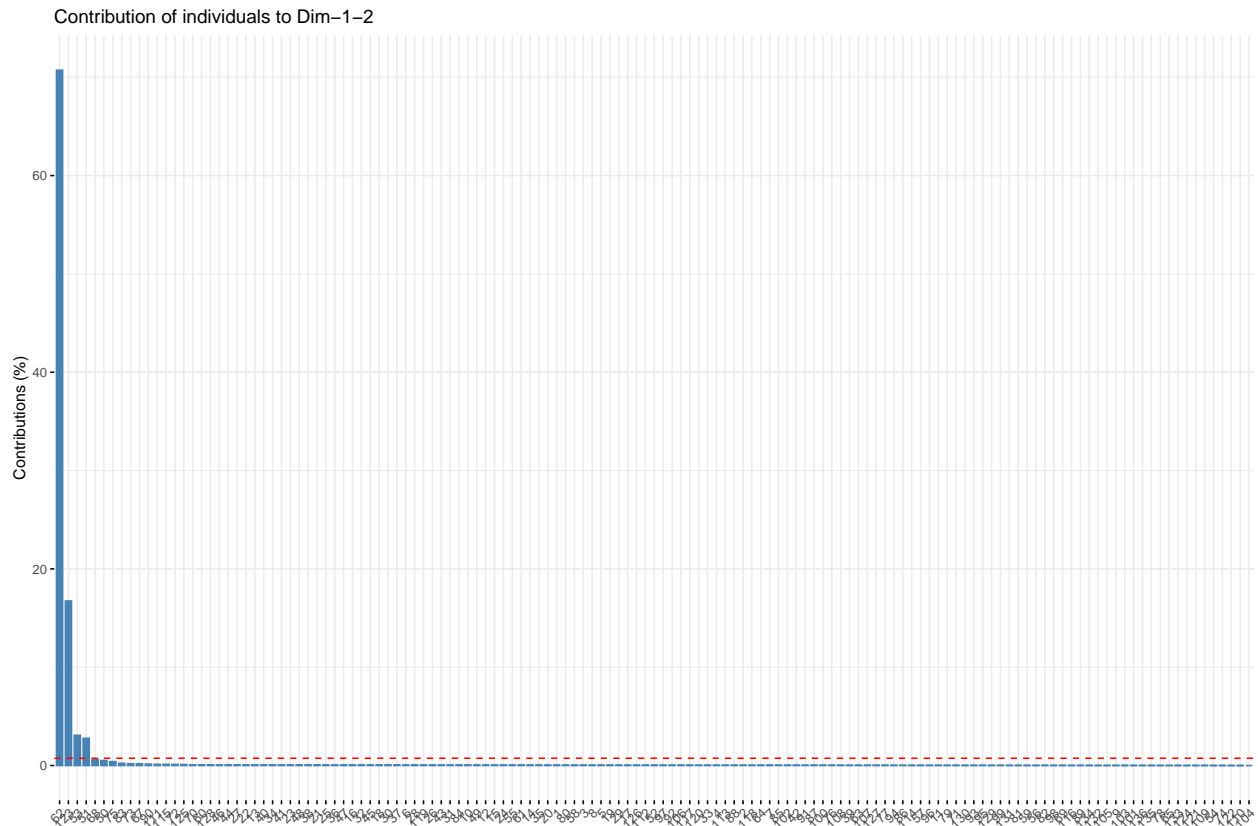
component, a variable with a contribution larger than this cutoff could be considered as important in contributing to the component.

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

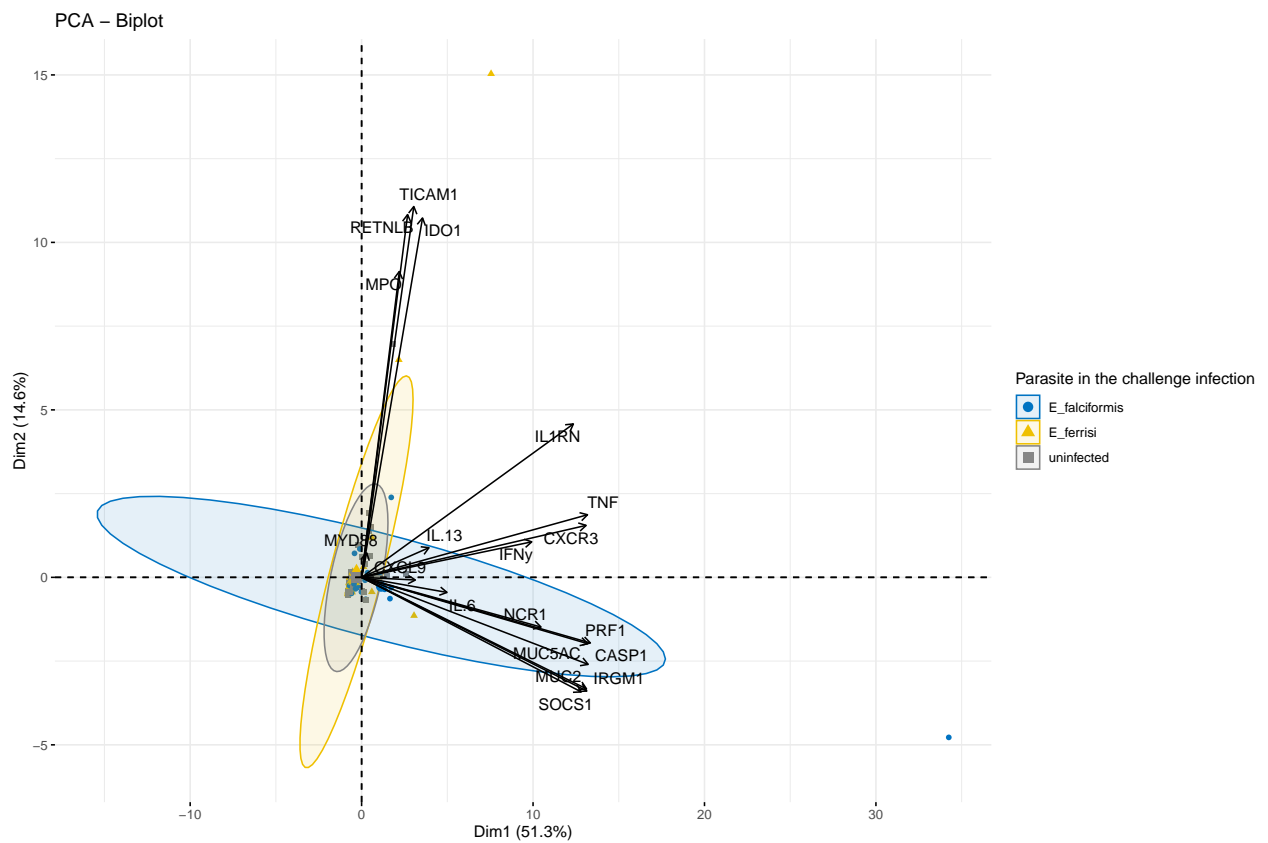
C1 and C2 are the contributions of the variable on PC1 and PC2, respectively Eig1 and Eig2 are the eigenvalues of PC1 and PC2, respectively. Recall that eigenvalues measure the amount of variation retained by each PC. In this case, the expected average contribution (cutoff) is calculated as follow: As 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



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.

Linear models:

```
##
## Call:
## lm(formula = max_WL ~ pc1 + pc2 + Parasite_challenge, data = lab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.608  -3.405   1.117   4.891  17.175
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    82.8079     1.3039   63.510 < 2e-16 ***
## pc1             0.1561     0.1937    0.806   0.422
## pc2            -0.2811     0.3583   -0.784   0.434
## Parasite_challengeE_ferrisi  8.0805     1.6309   4.955 2.21e-06 ***
## Parasite_challengeuninfected 10.9587     1.6163   6.780 3.80e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.898 on 130 degrees of freedom
## Multiple R-squared:  0.2643, Adjusted R-squared:  0.2416
## F-statistic: 11.67 on 4 and 130 DF, p-value: 3.945e-08
## [1] 911.4328
##
## Call:
## lm(formula = max_WL ~ pc1 + pc2, data = lab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18.466  -6.906   2.299   6.217   9.912
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  90.30385     0.68624 131.591 <2e-16 ***
## pc1          -0.08583     0.21974  -0.391   0.697
## pc2          -0.11327     0.41214  -0.275   0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.973 on 132 degrees of freedom
## Multiple R-squared:  0.001725, Adjusted R-squared: -0.0134
## F-statistic: 0.1141 on 2 and 132 DF, p-value: 0.8923
## [1] 948.631
```

Try instead: LLR test (likelihood ration) (LM4 package)?

<https://www.rdocumentation.org/packages/lmtest/versions/0.9-38/topics/lrtest>

In this way you compare each model, with the different variables used to predict.

Another way is to compare the AIC. (function : step)

```
weight_lm3 <- lm(max_WL ~ pc1 + pc2 + hybrid_status, data = lab)
weight_no_pc1 <- lm(max_WL ~ pc2 + hybrid_status, data = lab)
weight_no_pc2 <- lm(max_WL ~ pc1 + hybrid_status, data = lab)
weight_no_hybrid <- lm(max_WL ~ pc1 + pc2, data = lab)
lrtest(weight_lm3, weight_no_pc1)

## Likelihood ratio test
##
## Model 1: max_WL ~ pc1 + pc2 + hybrid_status
## Model 2: max_WL ~ pc2 + hybrid_status
##   #Df LogLik Df  Chisq Pr(>Chisq)
## 1    9 -455.98
## 2    8 -456.02 -1 0.0828    0.7736

lrtest(weight_lm3, weight_no_pc2)

## Likelihood ratio test
##
## Model 1: max_WL ~ pc1 + pc2 + hybrid_status
## Model 2: max_WL ~ pc1 + hybrid_status
##   #Df LogLik Df  Chisq Pr(>Chisq)
## 1    9 -455.98
## 2    8 -456.00 -1 0.0344    0.8528

lrtest(weight_lm3, weight_no_hybrid)

## Likelihood ratio test
##
## Model 1: max_WL ~ pc1 + pc2 + hybrid_status
## Model 2: max_WL ~ pc1 + pc2
##   #Df LogLik Df  Chisq Pr(>Chisq)
## 1    9 -455.98
## 2    4 -470.32 -5 28.672  2.689e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

##
## Call:
## lm(formula = max_WL ~ pc1 + pc2 + hybrid_status, data = lab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.838  -5.377   0.848   5.166  14.431
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    91.02398    1.11873   81.364 < 2e-16 ***
## pc1           -0.05886    0.21094   -0.279  0.78066
## pc2           -0.06913    0.38409   -0.180  0.85745
## hybrid_statusF0 M. m. musculus -5.31901    1.62219  -3.279  0.00134 **
## hybrid_statusF1 hybrid          5.22336    2.15503   2.424  0.01677 *
## hybrid_statusF1 M. m. domesticus 1.31341    2.98953   0.439  0.66116
## hybrid_statusF1 M. m. musculus  4.96625    3.46462   1.433  0.15420
## hybrid_statusother -0.29504    1.89858  -0.155  0.87675
## ---
```

```

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.31 on 127 degrees of freedom
## Multiple R-squared:  0.1927, Adjusted R-squared:  0.1482
## F-statistic: 4.332 on 7 and 127 DF,  p-value: 0.0002476
## [1] 929.9589
##
## Call:
## lm(formula = max_WL ~ pc1 + pc2 + infection_history, data = lab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.2184  -3.8061   0.6971   5.1086  14.4152
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)      83.1305      1.8741  44.358 < 2e-16
## pc1              0.2120      0.1951   1.087 0.279354
## pc2             -0.2821      0.3626  -0.778 0.438080
## infection_historyfalciformis_ferrisi    7.3768      2.4003   3.073 0.002603
## infection_historyfalciformis_uninfected  8.1921      2.4081   3.402 0.000901
## infection_historyferrisi_falciformis   -2.5995      2.9127  -0.892 0.373856
## infection_historyferrisi_ferrisi       9.6029      2.3525   4.082 7.94e-05
## infection_historyferrisi_uninfected    11.7391      2.3029   5.098 1.25e-06
## infection_historyuninfected            13.2847      3.0416   4.368 2.62e-05
## infection_historyuninfected_falciformis  2.4637      3.3359   0.739 0.461570
## infection_historyuninfected_ferrisi     3.5631      3.0401   1.172 0.243431
##
## (Intercept)          ***
## pc1
## pc2
## infection_historyfalciformis_ferrisi    **
## infection_historyfalciformis_uninfected ***
## infection_historyferrisi_falciformis
## infection_historyferrisi_ferrisi        ***
## infection_historyferrisi_uninfected     ***
## infection_historyuninfected             ***
## infection_historyuninfected_falciformis
## infection_historyuninfected_ferrisi
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.756 on 124 degrees of freedom
## Multiple R-squared:  0.3267, Adjusted R-squared:  0.2724
## F-statistic: 6.017 on 10 and 124 DF,  p-value: 2.094e-07
## [1] 911.4576
##
## Call:
## lm(formula = max_WL ~ pc1 + pc2, data = lab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max

```



```
## -18.466 -6.906 2.299 6.217 9.912
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) 90.30385 0.68624 131.591 <2e-16 ***
## pc1 -0.08583 0.21974 -0.391 0.697
## pc2 -0.11327 0.41214 -0.275 0.784
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.973 on 132 degrees of freedom
## Multiple R-squared: 0.001725, Adjusted R-squared: -0.0134
## F-statistic: 0.1141 on 2 and 132 DF, p-value: 0.8923

## df AIC
## weight_lm 6 911.4328
## weight_lm_exp_only 4 948.6310
```

repeating the heatmap on the now imputed data

```
# 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(gene))

# turn the first row into column names
gene %>%
  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
## 19

# 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(lab) %>%
  dplyr::select(c("Mouse_ID", "max_WL", "Parasite_challenge"))

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:

