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

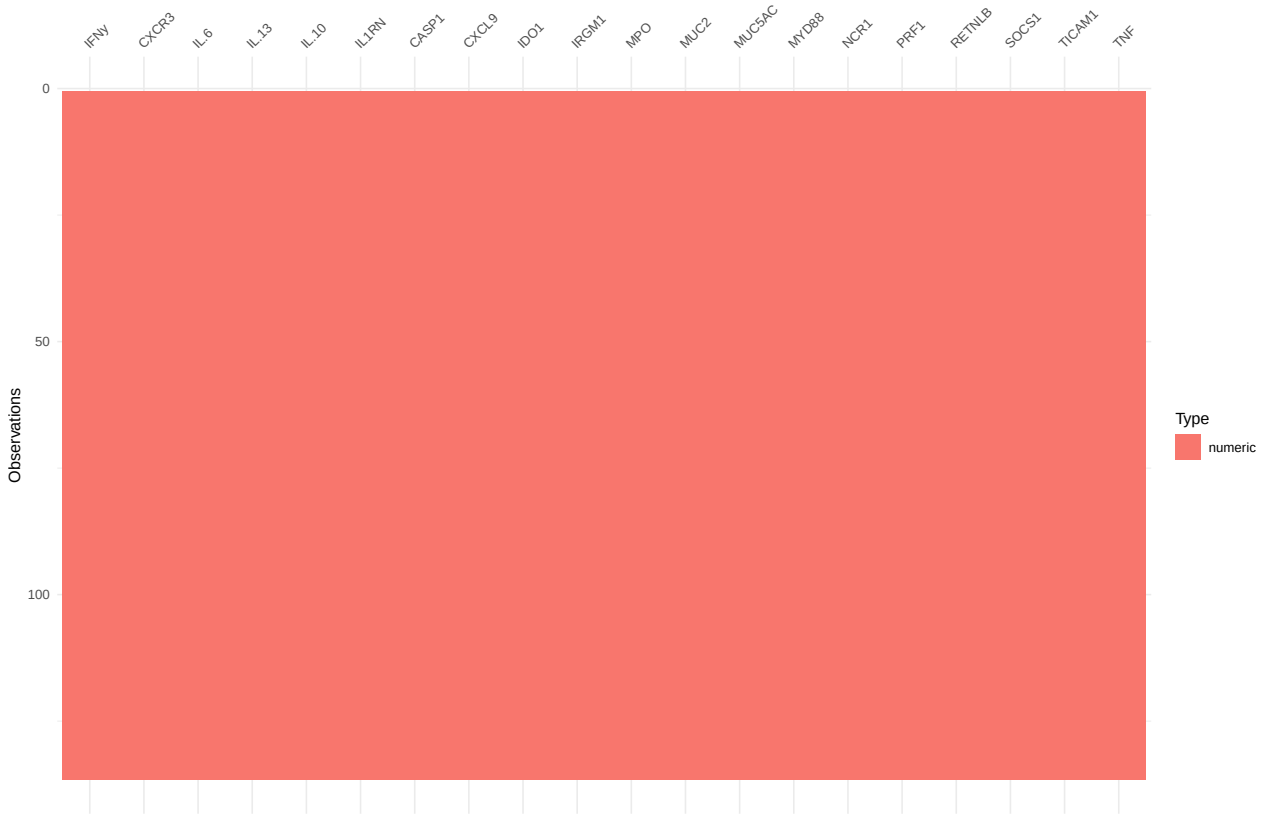
```
Gene_lab   <- c("IFNy", "CXCR3", "IL.6", "IL.13", "IL.10",
                "IL1RN","CASP1", "CXCL9", "IDO1", "IRGM1", "MPO",
                "MUC2", "MUC5AC", "MYD88", "NCR1", "PRF1", "RETNLB", "SOCS1",
                "TICAM1", "TNF") # "IL.12", "IRG6")

#add a suffix to represent changes in data file
Gene_lab_imp <- paste(Gene_lab, "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_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 = "_")
```

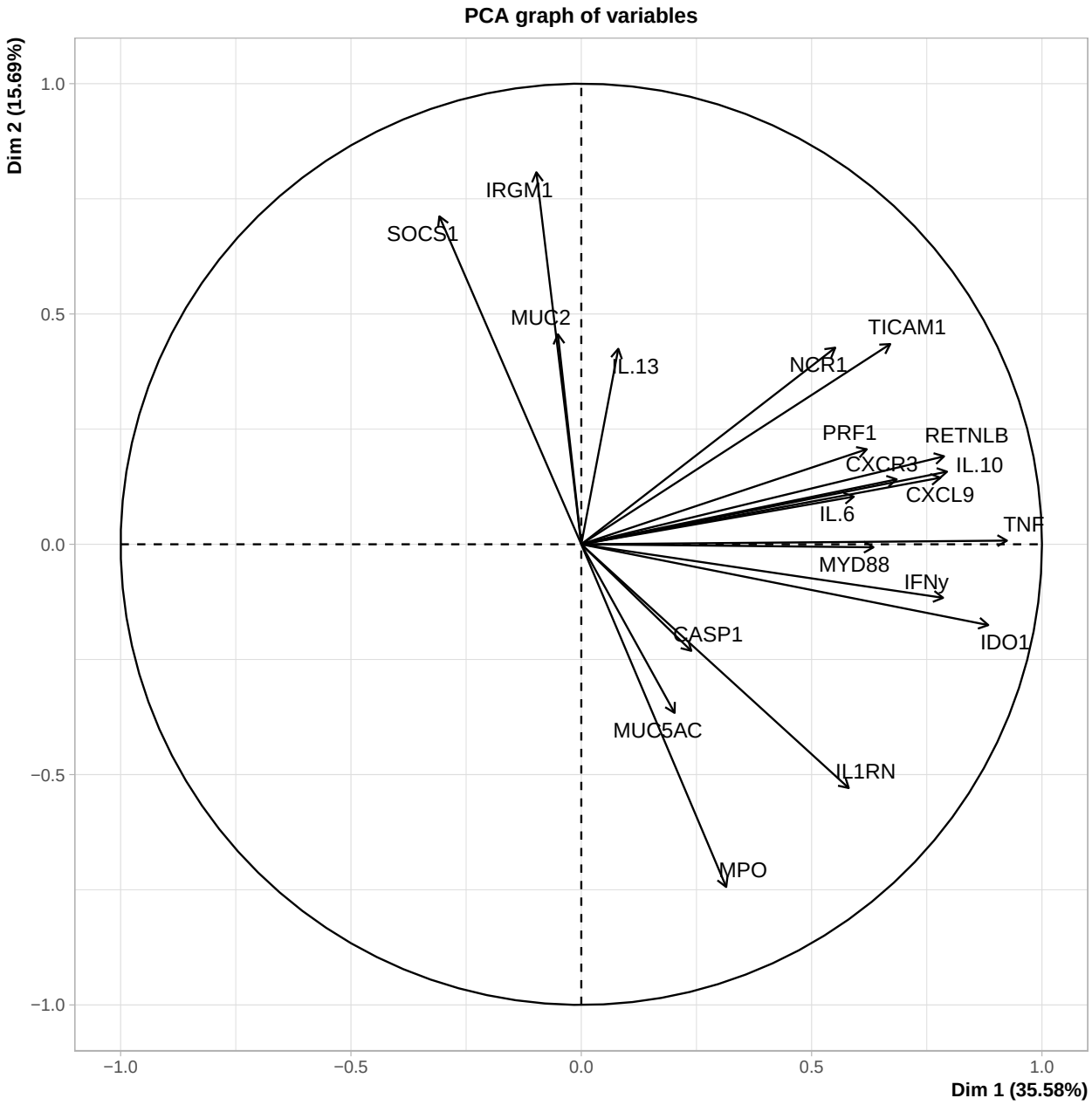
PCA on the lab genes -*imputed*

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

vis_dat(genes)
```

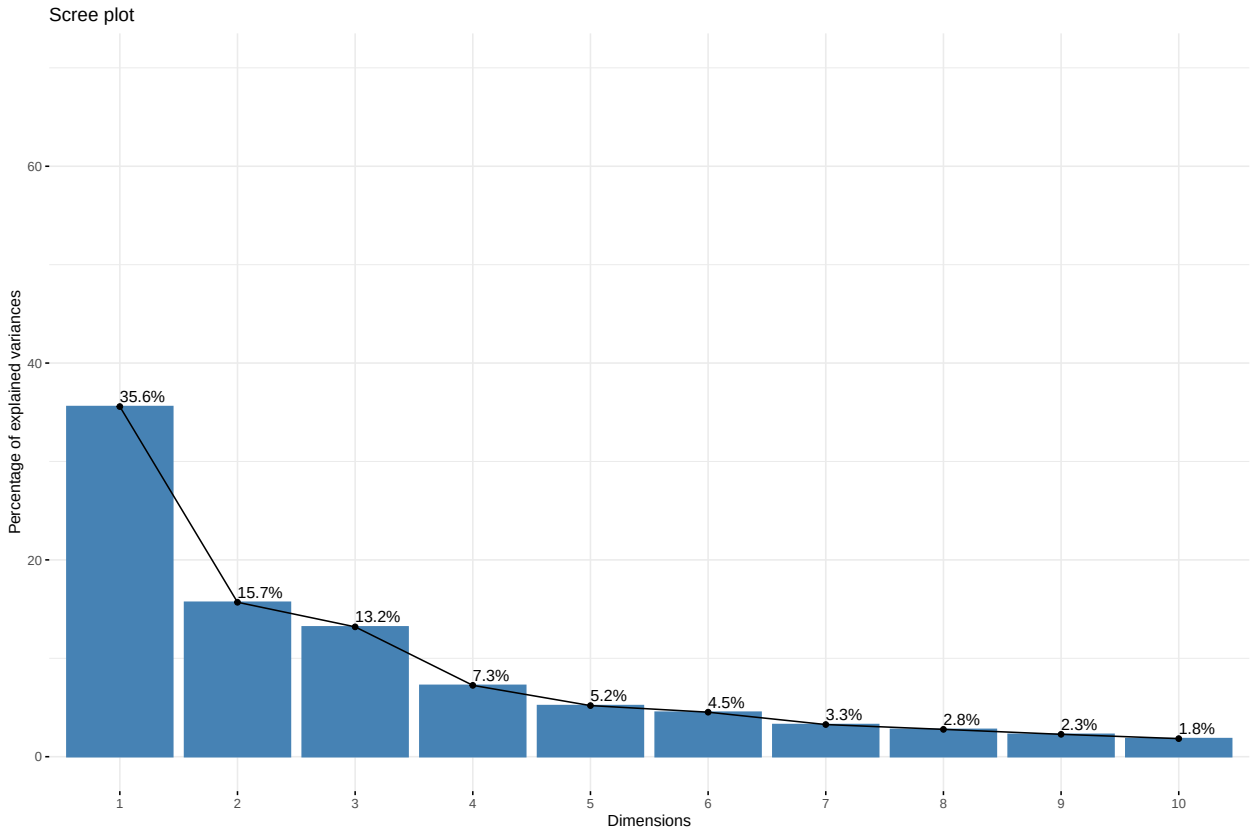
```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## i Please use `gather()` instead.
## i The deprecated feature was likely used in the visdat package.
##   Please report the issue at <https://github.com/ropensci/visdat/issues>.
```



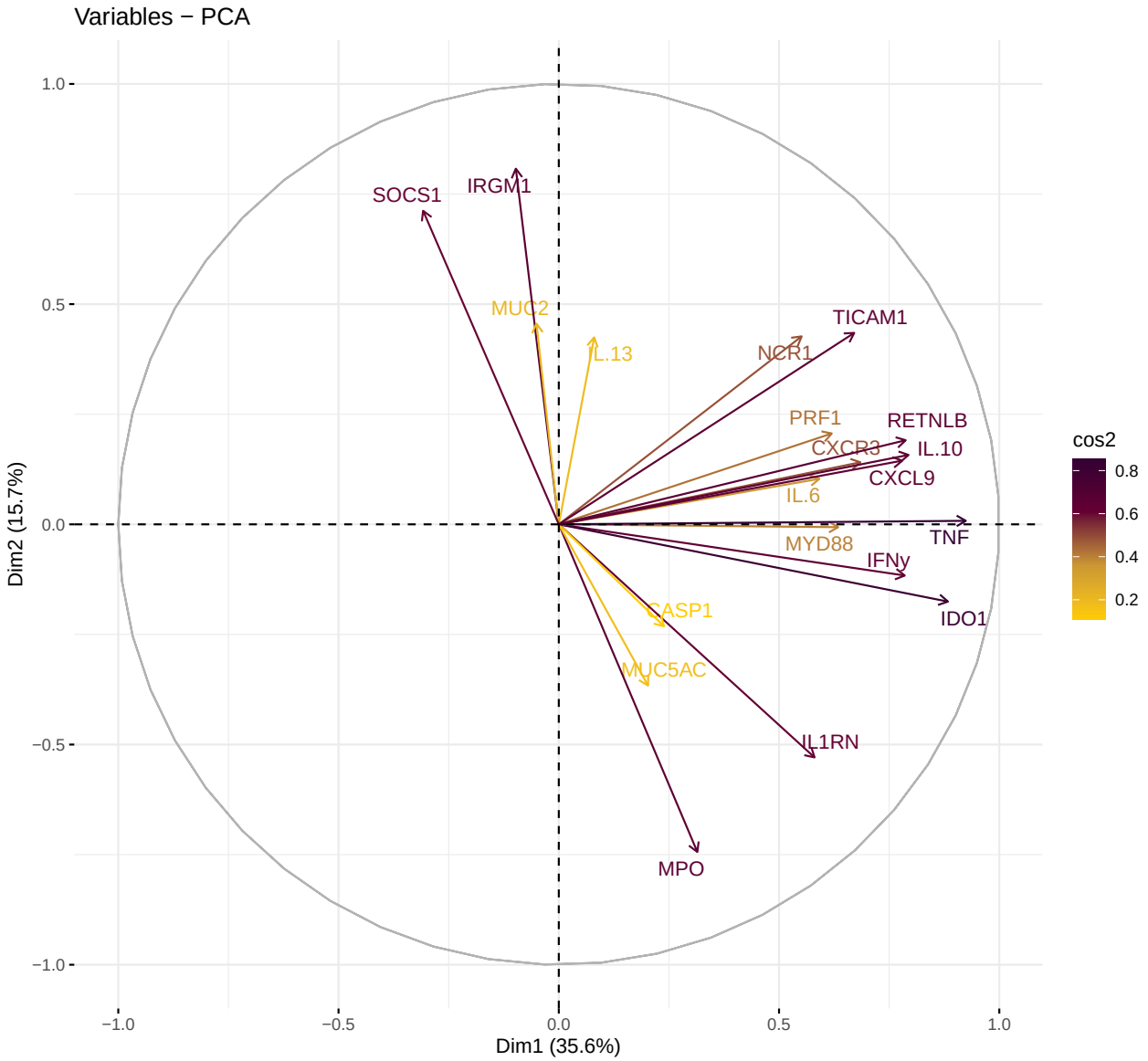

```
#select same rows in the first table
gene <- gene[row.names(genes), ]


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



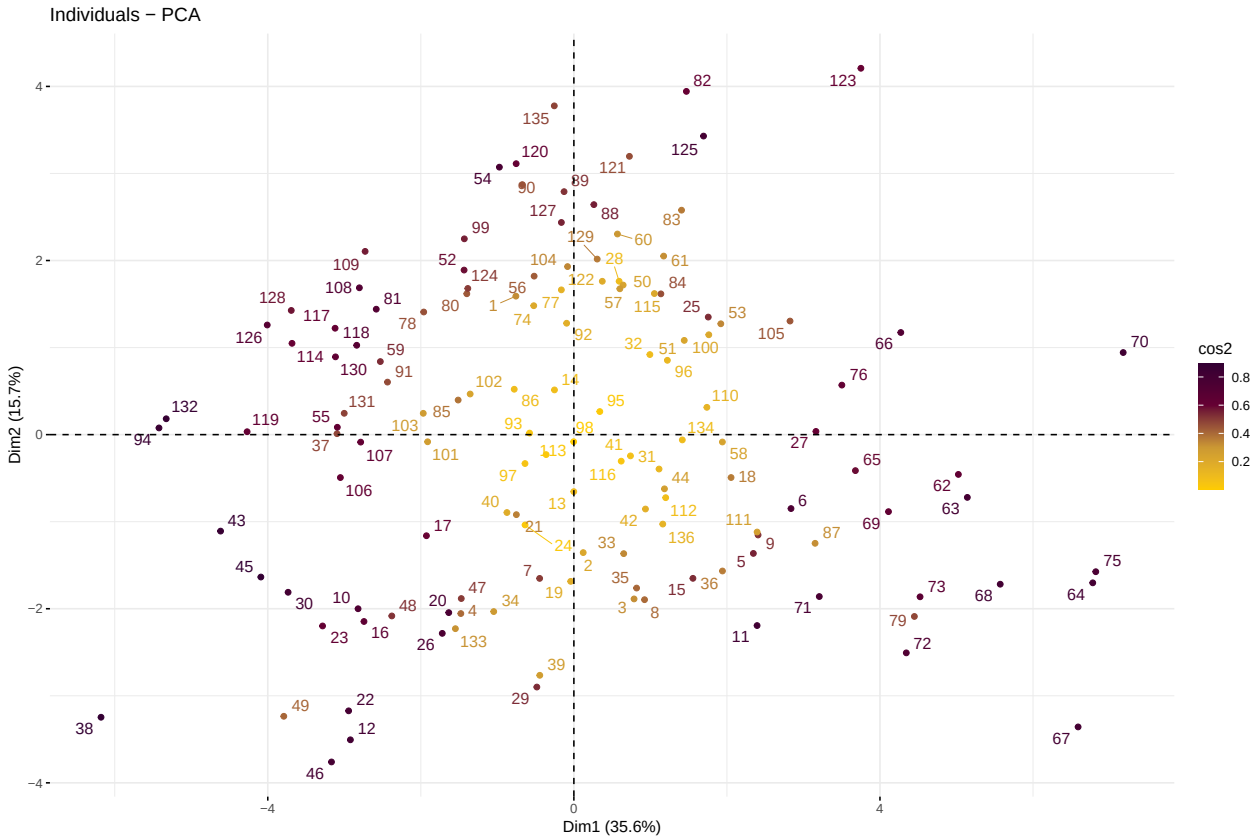

```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 70))
```




```
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)
```



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 issing data:

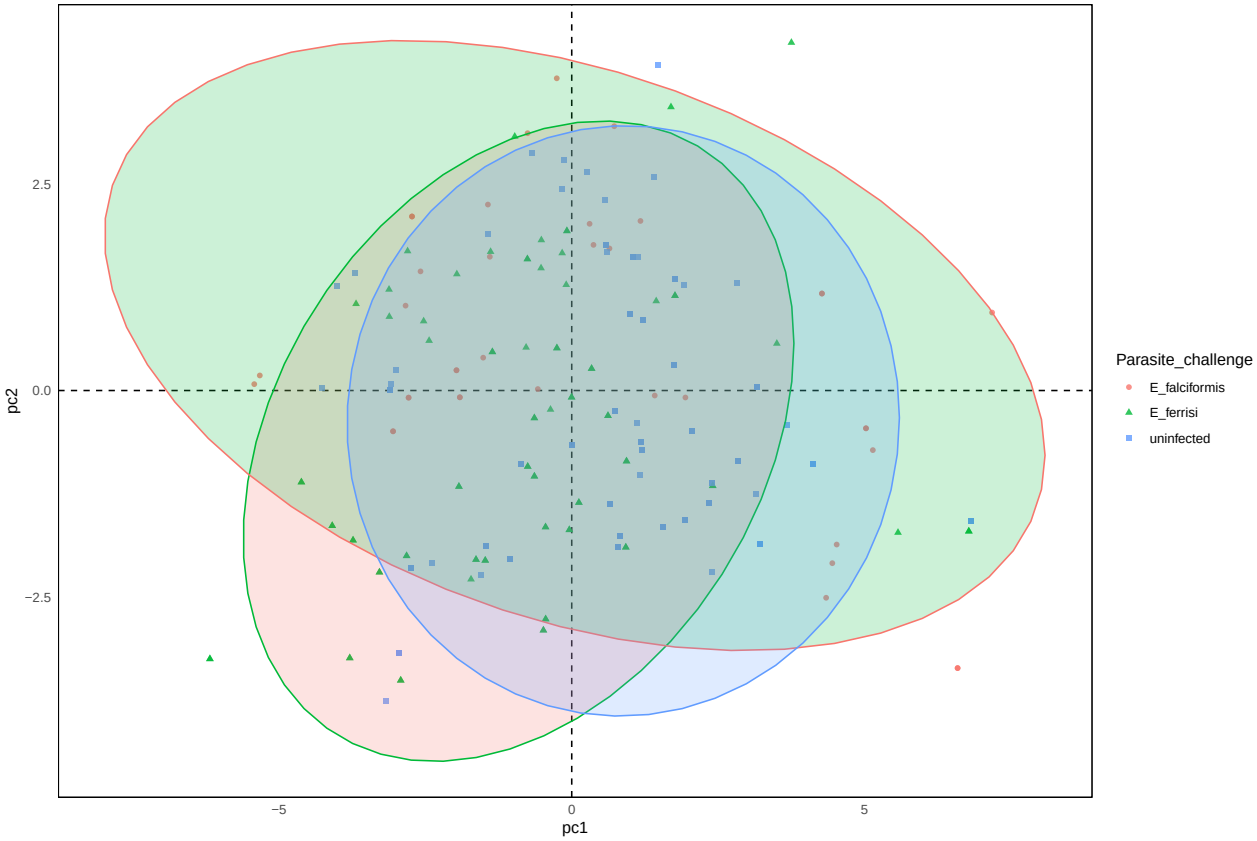
–> mulrimple imputation: generate several plausible values for each missing data point

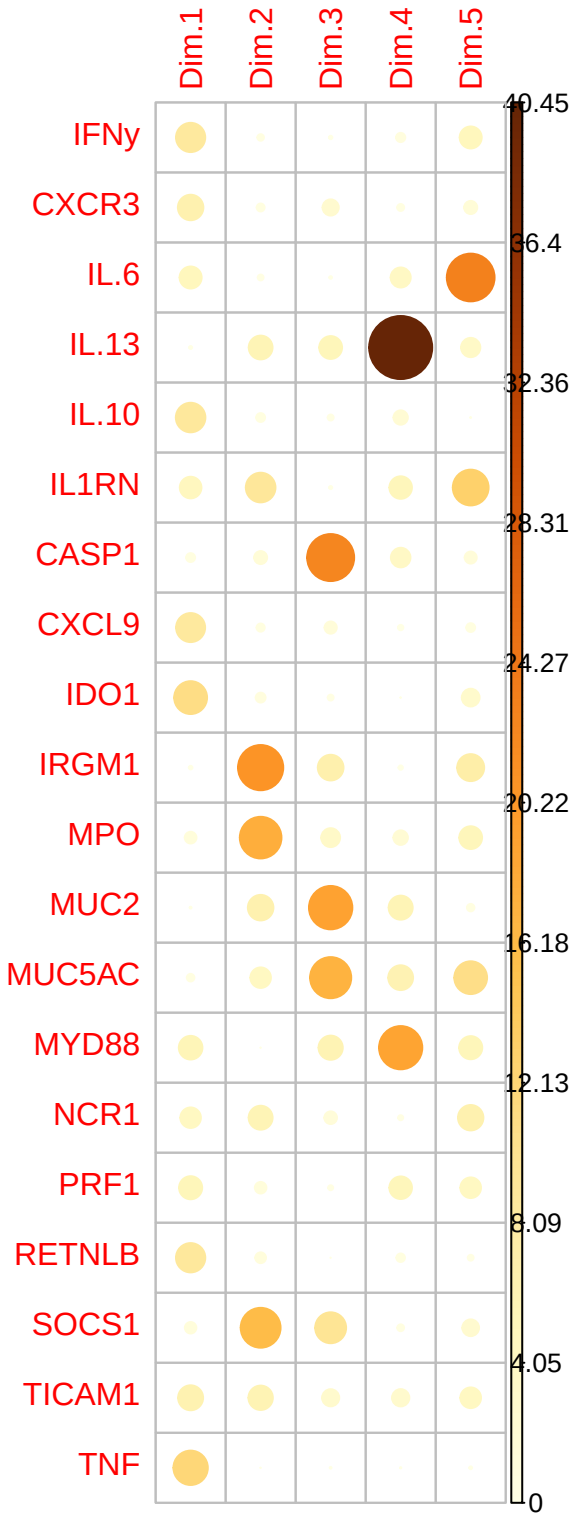
We here visualize the variability, that is uncertainty on the plane defined by two pca axes.

Biplot of the imputed gene pca

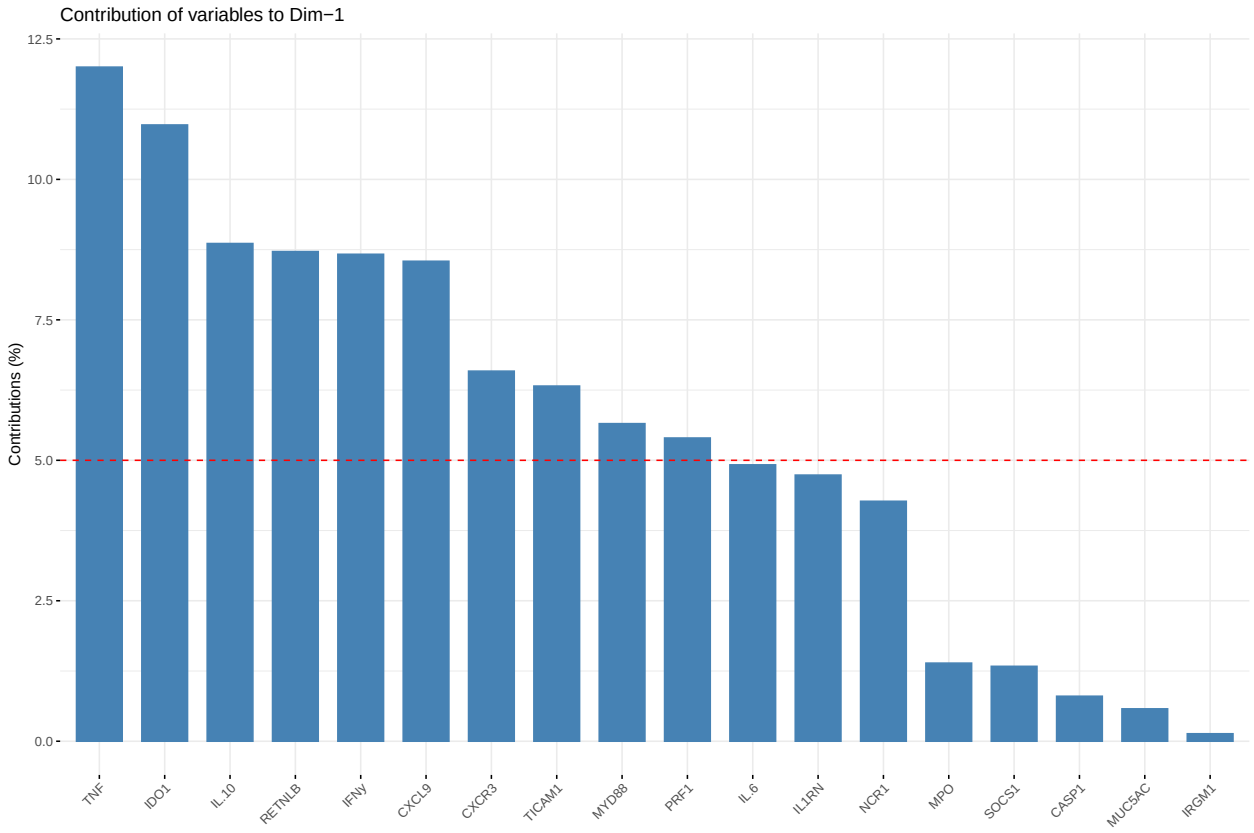
```
#Now we can make our initial plot of the PCA.
lab %>%
  ggplot(aes(x = pc1, y = pc2,
             color = Parasite_challenge,
             shape = Parasite_challenge)) +
  geom_hline(yintercept = 0, lty = 2) +
  geom_vline(xintercept = 0, lty = 2) +
  geom_point(alpha = 0.8) +
  stat_ellipse(geom="polygon",
               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 4 rows containing non-finite values (`stat_ellipse()`).
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```

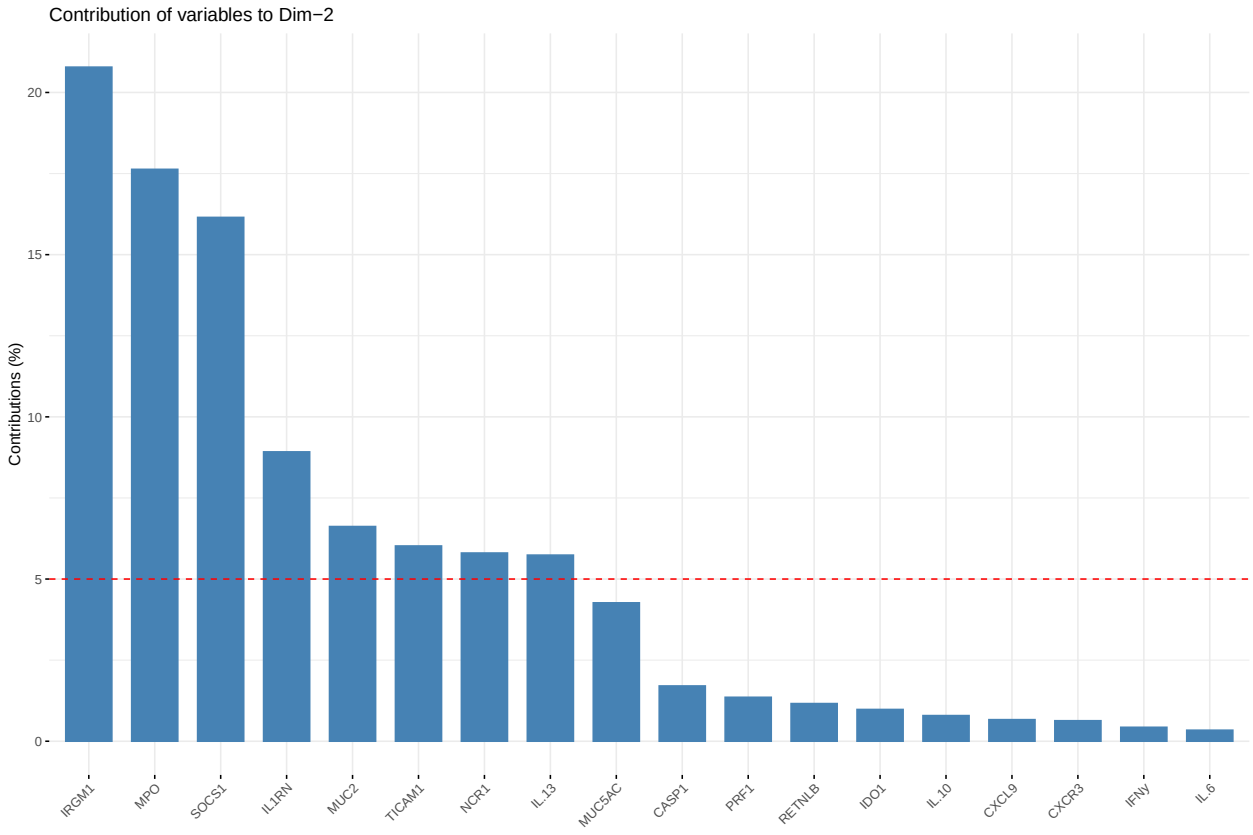


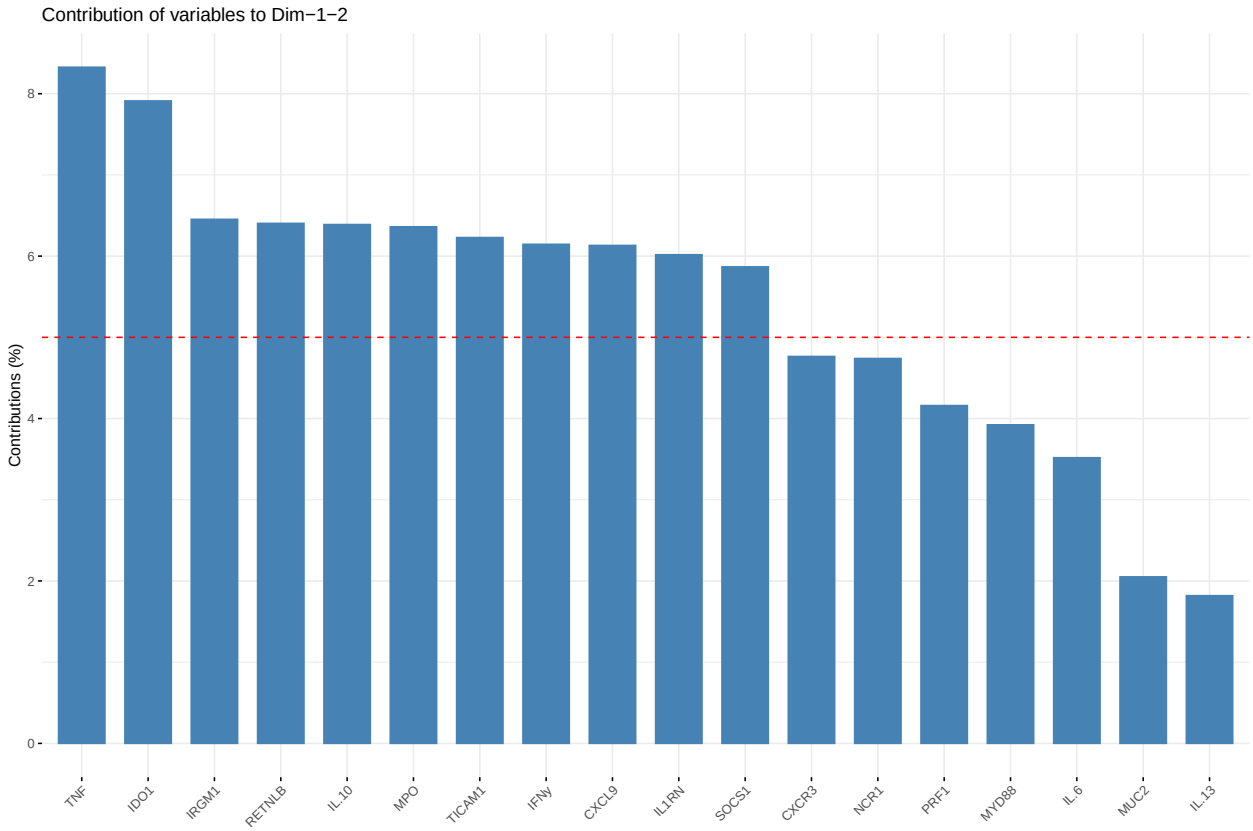


The function fviz_contrib() [factoextra package] can be used to draw a bar plot of variable contributions. If your data contains many variables, you can decide to show only the top contributing variables. The R code below shows the top 10 variables contributing to the principal components:




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



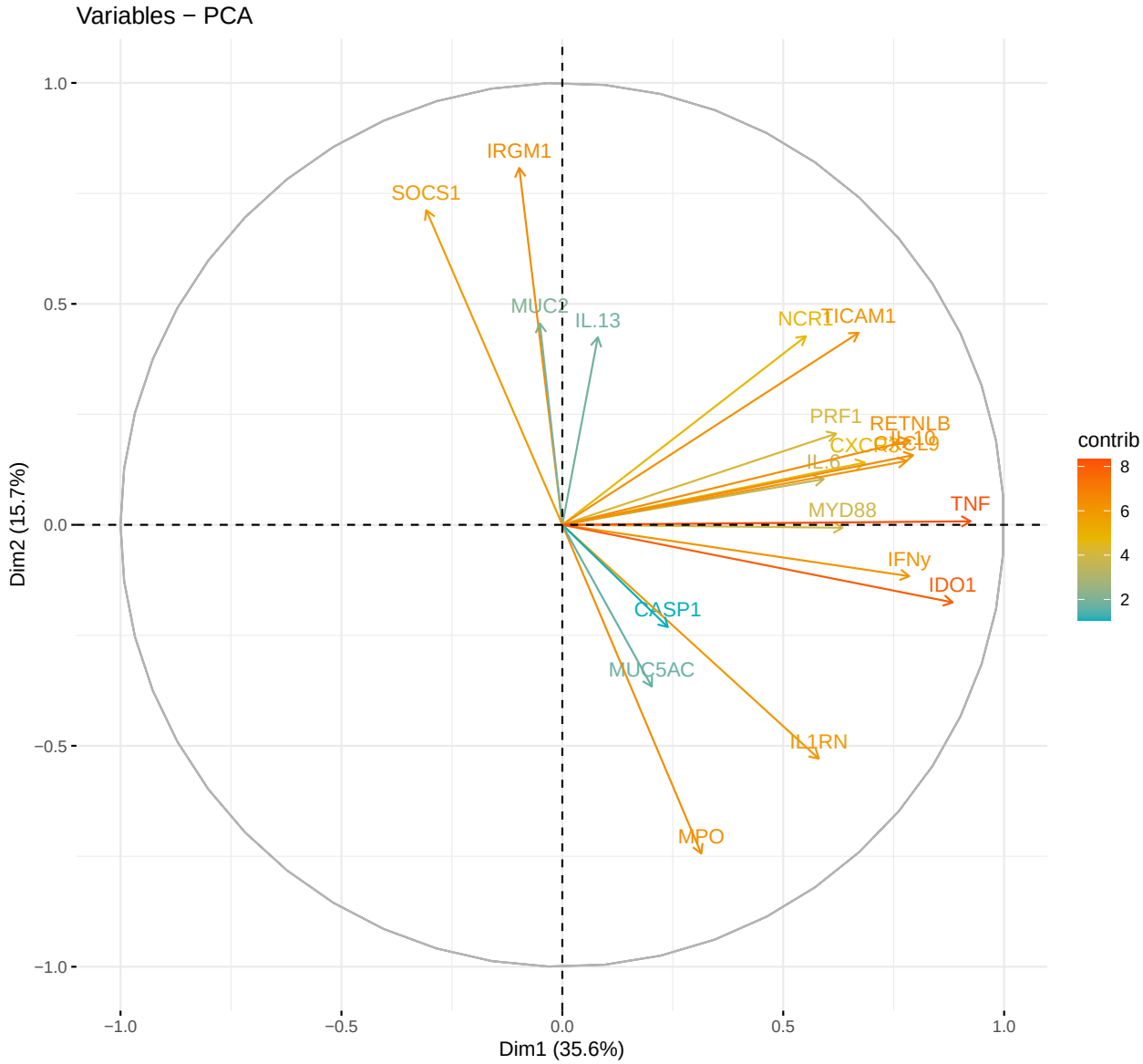


The red dashed line on the graph above indicates the expected average contribution. If the contribution of the variables were uniform, the expected value would be 1/length(variables) = 1/10 = 10%. For a given

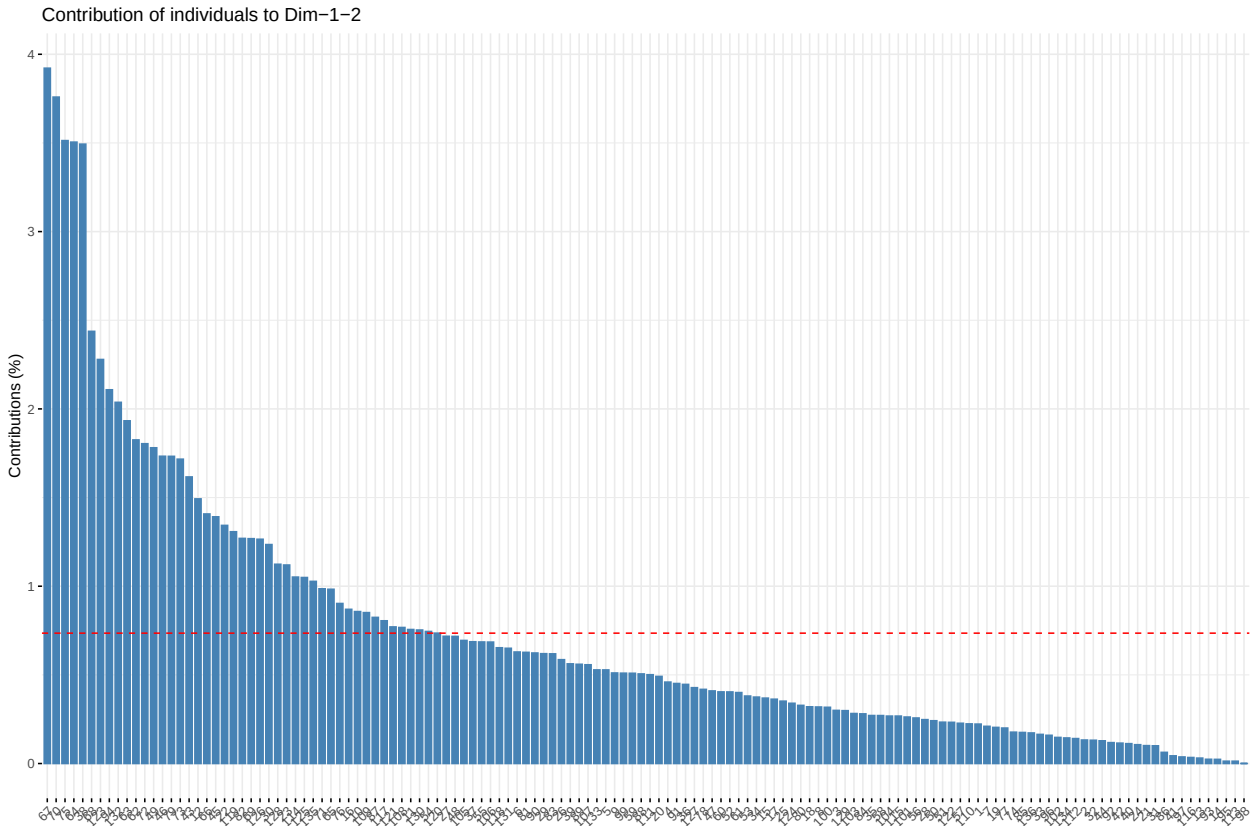
component, a variable with a contribution larger than this cutoff could be considered as important in contributing to the component.

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

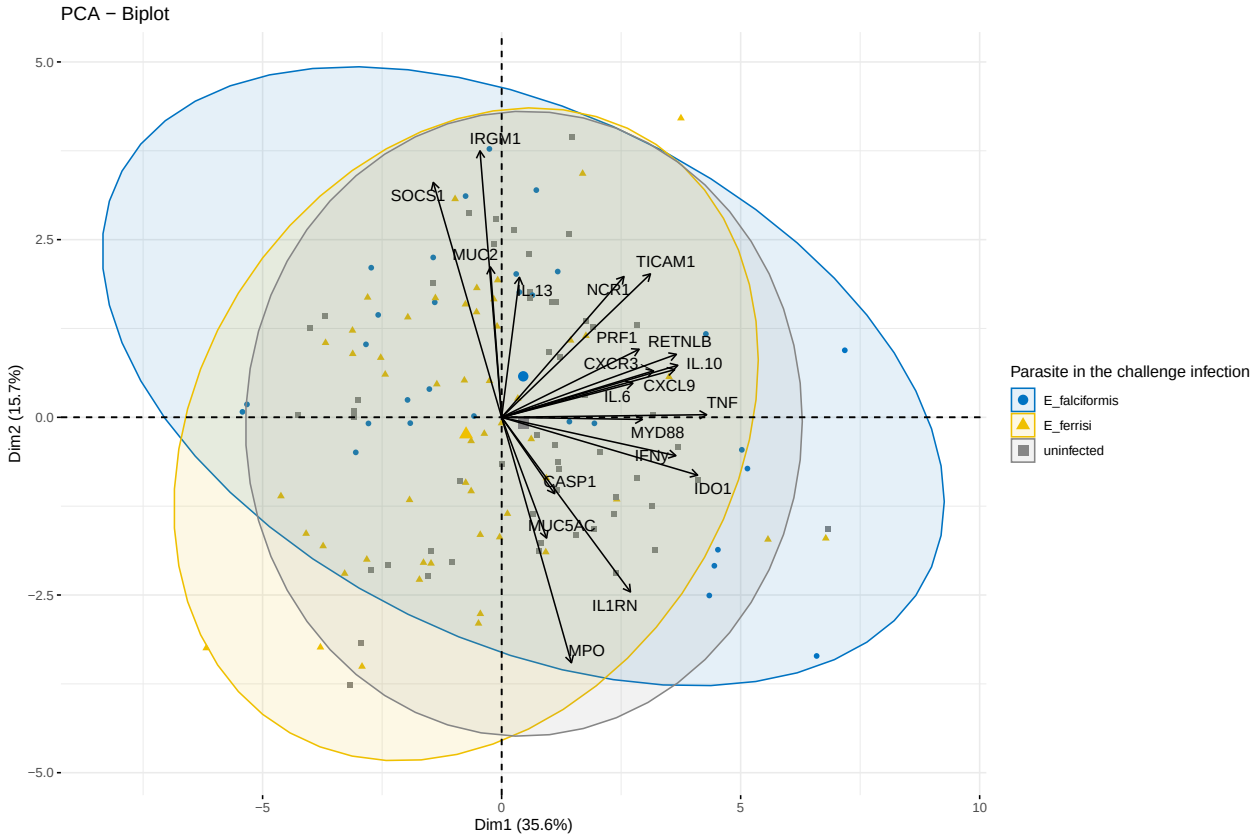
C1 and C2 are the contributions of the variable on PC1 and PC2, respectively Eig1 and Eig2 are the eigenvalues of PC1 and PC2, respectively. Recall that eigenvalues measure the amount of variation retained by each PC. In this case, the expected average contribution (cutoff) is calculated as follow: As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



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
## -19.1359  -3.8711   0.6575   4.4795  16.4814
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    83.8790     1.1978  70.029  < 2e-16 ***
## pc1                             0.5859     0.2129   2.752 0.006763 **
## pc2                            -1.1828     0.3178  -3.722 0.000292 ***
## Parasite_challengeE_ferrisi     7.0692     1.5260   4.632 8.61e-06 ***
## Parasite_challengeuninfected    9.4606     1.4861   6.366 3.01e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.46 on 131 degrees of freedom
## Multiple R-squared:  0.3514, Adjusted R-squared:  0.3316
## F-statistic: 17.75 on 4 and 131 DF,  p-value: 1.159e-11
```

```
## [1] 900.2979
```

```
##
## Call:
## lm(formula = max_WL ~ pc1 + pc2, data = lab)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.675  -5.338   1.571   5.787  14.756
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  90.3383     0.6301 143.380  < 2e-16 ***
## pc1           0.5489     0.2362   2.324   0.0216 *
## pc2          -1.4966     0.3556  -4.208  4.7e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.348 on 133 degrees of freedom
## Multiple R-squared:  0.1481, Adjusted R-squared:  0.1352
## F-statistic: 11.56 on 2 and 133 DF,  p-value: 2.358e-05
```

```
## [1] 933.3911
```

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 usesd 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 -454.02
## 2   8 -455.40 -1 2.7699    0.09605 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
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 -454.02
## 2   8 -457.65 -1 7.2749   0.006993 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
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 -454.02
## 2   4 -462.70 -5 17.36   0.003865 **
## ---
## 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
## -16.6524  -4.2005   0.7392   5.2417  16.1614
##
## Coefficients:
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      91.1939     1.0840  84.130  < 2e-16 ***
## pc1                               0.4439     0.2735   1.623  0.10708
## pc2                              -0.9912     0.3737  -2.652  0.00901 **
## hybrid_statusF0 M. m. musculus   -4.2109     1.6492  -2.553  0.01185 *
## hybrid_statusF1 hybrid            4.0624     2.1630   1.878  0.06264 .
```

```
## hybrid_statusF1 M. m. domesticus   0.0549     2.9193   0.019  0.98502
## hybrid_statusF1 M. m. musculus     3.8138     3.5054   1.088  0.27864
## hybrid_statusother                -1.4069     1.8454  -0.762  0.44725
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.027 on 128 degrees of freedom
## Multiple R-squared:  0.2501, Adjusted R-squared:  0.2091
## F-statistic:   6.1 on 7 and 128 DF,  p-value: 3.695e-06
```

```
## [1] 926.0309
```

```
##
## Call:
## lm(formula = max_WL ~ pc1 + pc2 + infection_history, data = lab)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.1307  -4.2117   0.8972   4.1483  15.4110
##
## Coefficients:
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                           85.02155    1.74525  48.716  < 2e-16
## pc1                                    0.55854    0.21130   2.643 0.009261
## pc2                                   -1.25850    0.33444  -3.763 0.000257
## infection_historyfalciformis_ferrisi   5.39629    2.22911   2.421 0.016923
## infection_historyfalciformis_uninfected  5.26691    2.27576   2.314 0.022280
## infection_historyferrisi_falciformis  -3.20755    2.64146  -1.214 0.226919
## infection_historyferrisi_ferrisi       7.16891    2.29882   3.119 0.002257
## infection_historyferrisi_uninfected    9.49348    2.14870   4.418 2.13e-05
## infection_historyuninfected           12.16571    2.79839   4.347 2.83e-05
## infection_historyuninfected_falciformis -0.08671    3.12254  -0.028 0.977891
## infection_historyuninfected_ferrisi    3.49689    2.82122   1.239 0.217485
##
## (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: 6.307 on 125 degrees of freedom
## Multiple R-squared:   0.41,  Adjusted R-squared:  0.3628
## F-statistic: 8.686 on 10 and 125 DF,  p-value: 1.1e-10
```

```
## [1] 899.4269
```

```
##
## Call:
```

```
## lm(formula = max_WL ~ pc1 + pc2, data = lab)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.675  -5.338   1.571   5.787  14.756
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  90.3383     0.6301 143.380  < 2e-16 ***
## pc1           0.5489     0.2362   2.324   0.0216 *
## pc2          -1.4966     0.3556  -4.208  4.7e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.348 on 133 degrees of freedom
## Multiple R-squared:  0.1481, Adjusted R-squared:  0.1352
## F-statistic: 11.56 on 2 and 133 DF,  p-value: 2.358e-05
```

```
##                    df      AIC
## weight_lm           6 900.2979
## weight_lm_exp_only  4 933.3911
```

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
##    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(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 unecessary column
annotation_df <- annotation_df %>% dplyr::select(-Mouse_ID, )
```

Heatmap on gene expression data: