PCA

by Steph and Ale

Contents

Large datasets as microbiome data are increasingly common and are often difficult to interpret. Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. Finding such new variables, the principal components, reduces to solving an eigenvalue/eigenvector problem(Jolliffe and Cadima 2016)

In this tutorial , Let's use a test data[1]. For this purpose Let's load the data that it is will be used.

```
microbiome_data<- read.delim("../Data/soil_species_bacteria.txt",
                             check.names = F, row.names = 1)
metadata<- read.delim("../Data/soil_maize_metadata.txt", check.names = F)
```

"Check.names argument is used to avoid column names from being altered"

Now Let's load the libraries to use.

```
#Libraries to data wrangling
library(tidyverse)

#libraries to perform clr transformations
library(ALDEx2)
library(compositions)
library(CoDaSeq)

#libraries to perform PCA
library(FactoMineR)
library(vegan)
#prcomp function is in stats packaged (load by default)

#libraries to plot or visualize results
library(ggplot2)
library(ggpubr)
```

[1]Data provided from Ph D Alejandra Miranda of her experiment (real data)

Let's explore different methods to transform (clr-transform/compositional analysis) and perform PCA analysis.

We are going to use 3 compositional methods of transformation:

Method 1:

ttps://www.bioconductor.org/packages/release/bioc/manuals/ALDEx2/man/ALDEx2.pdf

Method 2 :

https://github.com/ggloor/Frontiers_2017

Method 3:

https://github.com/ggloor/CJM_Supplement

Transformations

```
#Method 1

aldex.clr.transform <- aldex.clr(microbiome_data, mc.samples = 999, denom="all",
                                 verbose = FALSE, useMC=FALSE)
aldex.clr.transform.data<-  t(getMonteCarloSample(aldex.clr.transform,1) )


#Method 2

#setting unrelated conditions just to generate te input
conds_soil <- c(rep("cond1", 18), rep("cond2", 18))

aldex.clr.transform2 <- aldex.clr(microbiome_data, mc.samples=256, verbose=FALSE,
                                  conds_soil)
aldex.clr.transform.effect <- aldex.effect(aldex.clr.transform2,
                                  include.sample.summary = TRUE, verbose = FALSE)
#formatting rownames and colnames
clr.samples <- t(aldex.clr.transform.effect[,grep("rab.sample",
                                  colnames(aldex.clr.transform.effect))])
rownames(clr.samples) <- gsub("rab.sample.", "", rownames(clr.samples))
#exponential and clr function
exp <- apply(clr.samples,1,function(x)2^x)
aldex.clr.transform.data2<- t(apply(exp,2,function(x)log2(x)-mean(log2(x))))

#Method 3

#changing zeros
zero_function<- t(cmultRepl(t(microbiome_data), method="CZM", output="p-counts"))

#applying clr transformation
clr.transform.data<- t(codaSeq.clr(zero_function, samples.by.row = F))
```

Let's format metadata and set functions to visualize PCA's

```
#Formatting metadata variables setting as factors
metadata$Tillage<- factor(metadata$Tillage,
                  levels = c( "Yes", "No"),
                  labels = c("Tillage", "No-Tillage"))
```

```
metadata$Nitrogen<- factor(metadata$Nitrogen,
                  levels = c( "Yes", "No"),
                  labels = c("Nitrogen", "No-Nitrogen"))

#PCa functions and construction

#LABELS
PC1.f<- function(x,y){paste("PC1", round(sum(x$sdev[1] ^ 2) / mvar(y) * 100, 1), "%")}
PC2.f <- function(x,y){paste("PC2", round(sum(x$sdev[2] ^ 2) / mvar(y) * 100, 1), "%")}

pca_plots<- function(tab){ggplot() +
    geom_segment(data=data.frame(tab$rotation) %>%   #arrows
                  rownames_to_column(var = "FeatureID")%>%
                  mutate(a=sqrt(PC1^2+PC2^2)) %>%
                  # calculate the distance from the origin
                  top_n(10, a) %>% #keep 10 furthest away points
                  mutate(PC1=PC1*100, PC2=PC2*100),
                  aes(x=0, xend=PC1, y=0, yend=PC2),
                  arrow = arrow(length = unit(0.3,"cm")))+
    geom_point(data=data.frame(tab$x) %>% #individuals or points
                  rownames_to_column(var = "Sample")%>%
                  left_join(metadata, by = "Sample"),
                  aes(x=PC1, y=PC2, color=Tillage, shape= Nitrogen), size=4) +
   geom_vline(xintercept = 0, linetype = 2) +   #lines-cross
   geom_hline(yintercept = 0, linetype = 2) +
    theme_bw()+
   theme(axis.text = element_text(colour = "black", size = 12),#theme changes
                  axis.title = element_text(colour = "black", size = 12),
                  legend.text = element_text(size = 10),
                  legend.title = element_text(size = 12),
                  legend.position = "right",
                  legend.box = "vertical") }
```

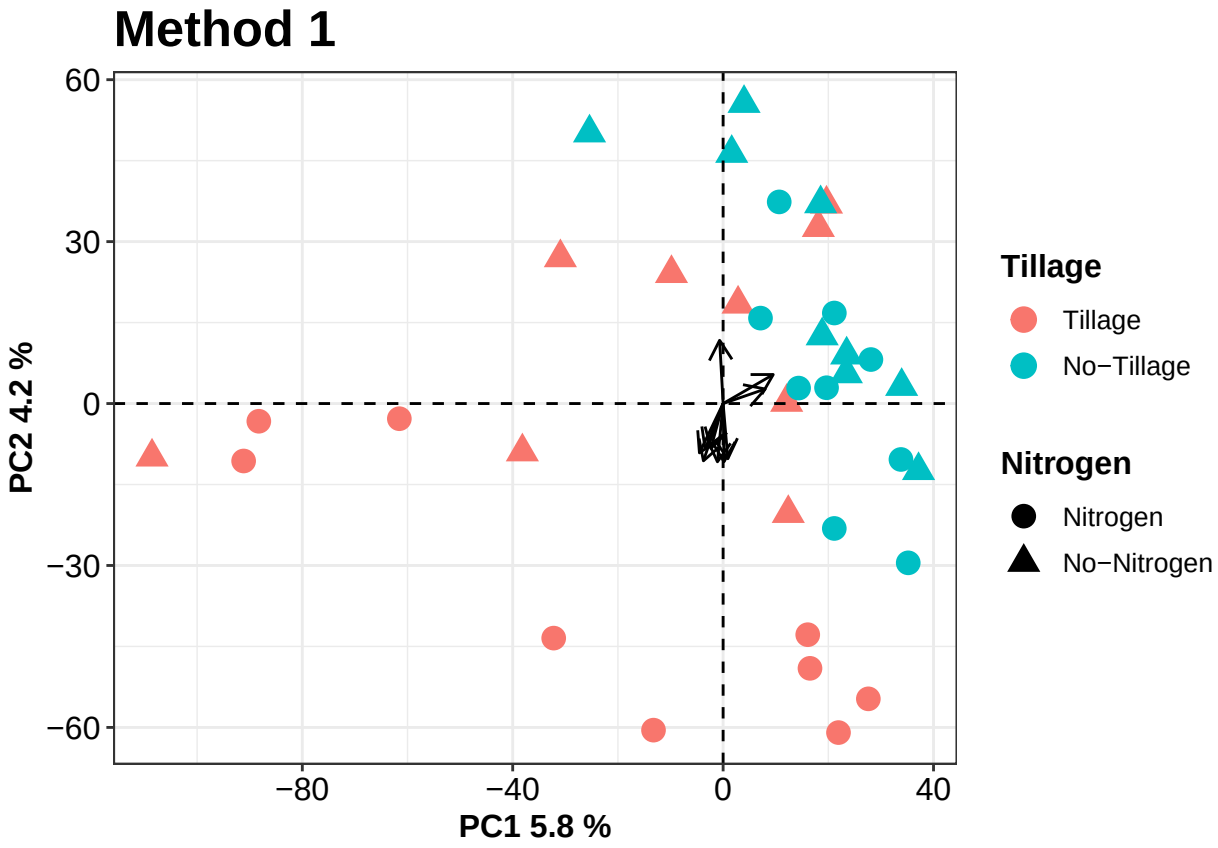
PCA Analysis, comparing transformation methods

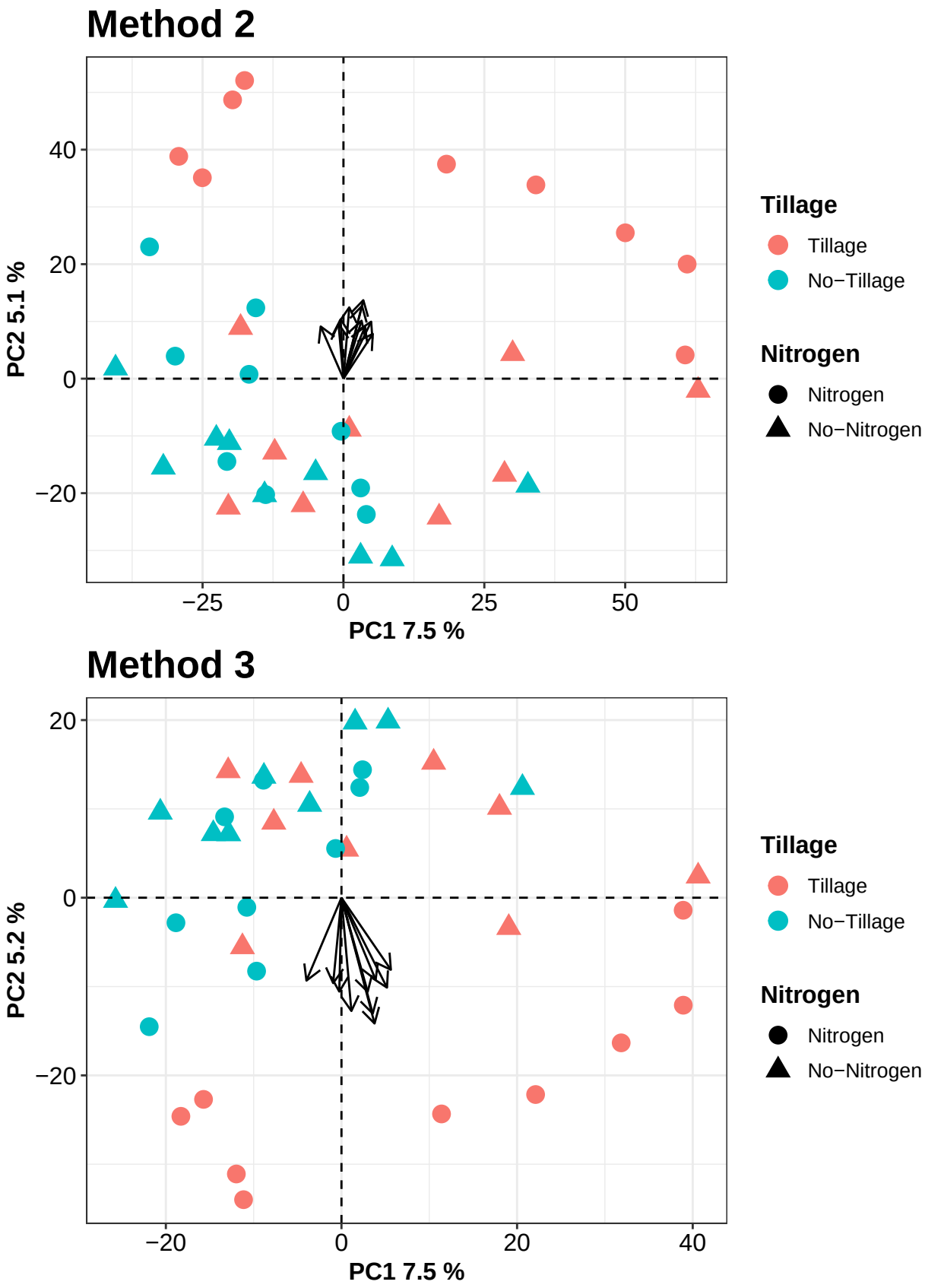
Now Let's perform PCA analysis and plot:

Let's use one of the functions to PCA to evaluate the different methods to clr transformation, Let's take 'prcomp' function from 'stats' r-base package

```
pca_method1<- prcomp(aldex.clr.transform.data)
pca_method2<- prcomp(aldex.clr.transform.data2)
pca_method3<- prcomp(clr.transform.data)



PC1_method1<-PC1.f(pca_method1, aldex.clr.transform.data)
PC2_method1<-PC2.f(pca_method1, aldex.clr.transform.data)

PC1_method2<-PC1.f(pca_method2, aldex.clr.transform.data2)
PC2_method2<-PC2.f(pca_method2, aldex.clr.transform.data2)

PC1_method3<-PC1.f(pca_method3, clr.transform.data)
```

```
PC2_method3<-PC2.f(pca_method3, clr.transform.data)

pca_m1<-pca_plots(pca_method1)+ xlab(PC1_method1)+ylab(PC2_method1)+ ggtitle(
  "Method 1")+theme(title=element_text(size = 16, face="bold"))
pca_m2<-pca_plots(pca_method2)+ xlab(PC1_method2)+ylab(PC2_method2)+ ggtitle(
  "Method 2")+theme(title=element_text(size = 16, face="bold"))
pca_m3<-pca_plots(pca_method3)+ xlab(PC1_method3)+ylab(PC2_method3)+ ggtitle(
  "Method 3")+theme(title=element_text(size = 16, face="bold"))

pca_m1; pca_m2; pca_m3
```




Method 2
PC2 5.1 %
PC1 7.5 %
40
20
0
−20
−25
0
25
50
Tillage
Tillage
No−Tillage
Nitrogen
Nitrogen
No−Nitrogen
Method 3
PC2 5.2 %
PC1 7.5 %
20
0
−20
−20
0
20
40
Tillage
Tillage
No−Tillage
Nitrogen
Nitrogen
No−Nitrogen

PCA Analysis, comparing PCA methods

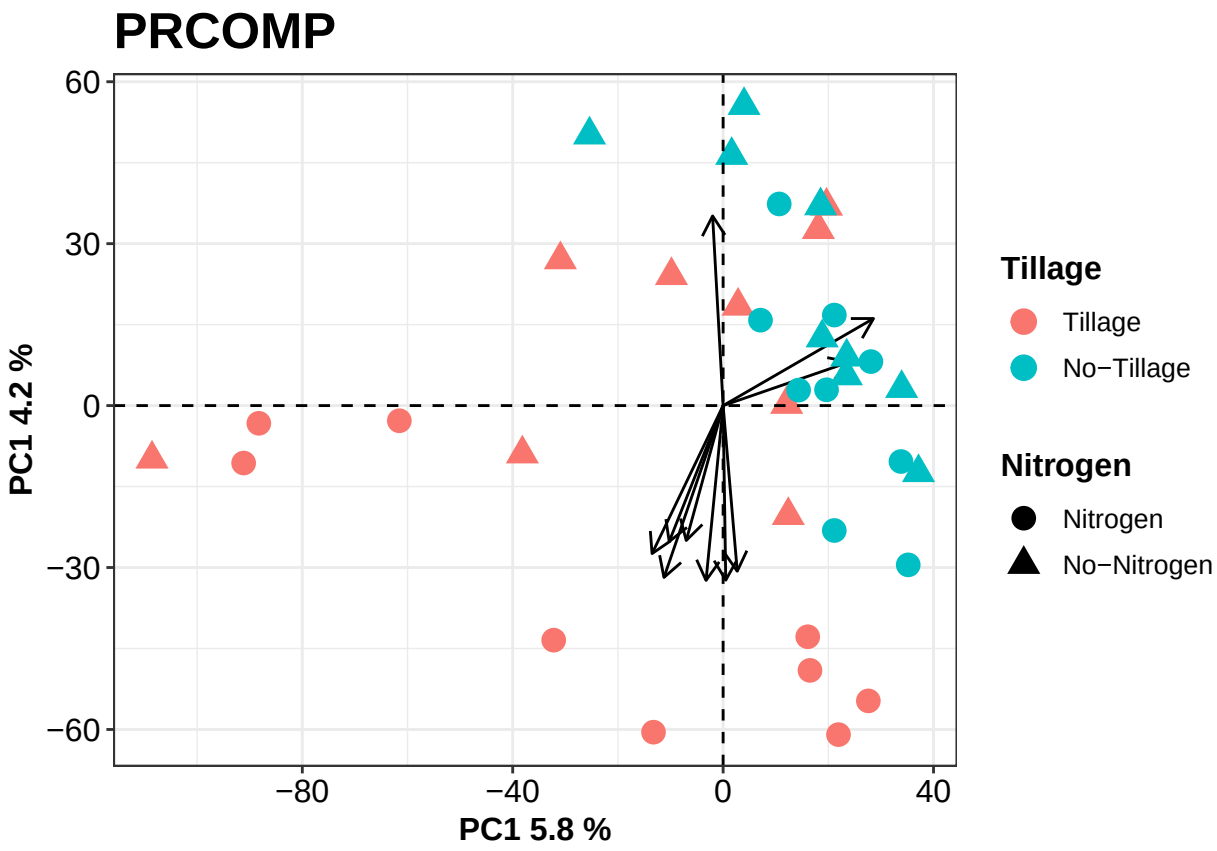
Now Let's perform PCA analysis and plot but with the different methods of doing PCA's

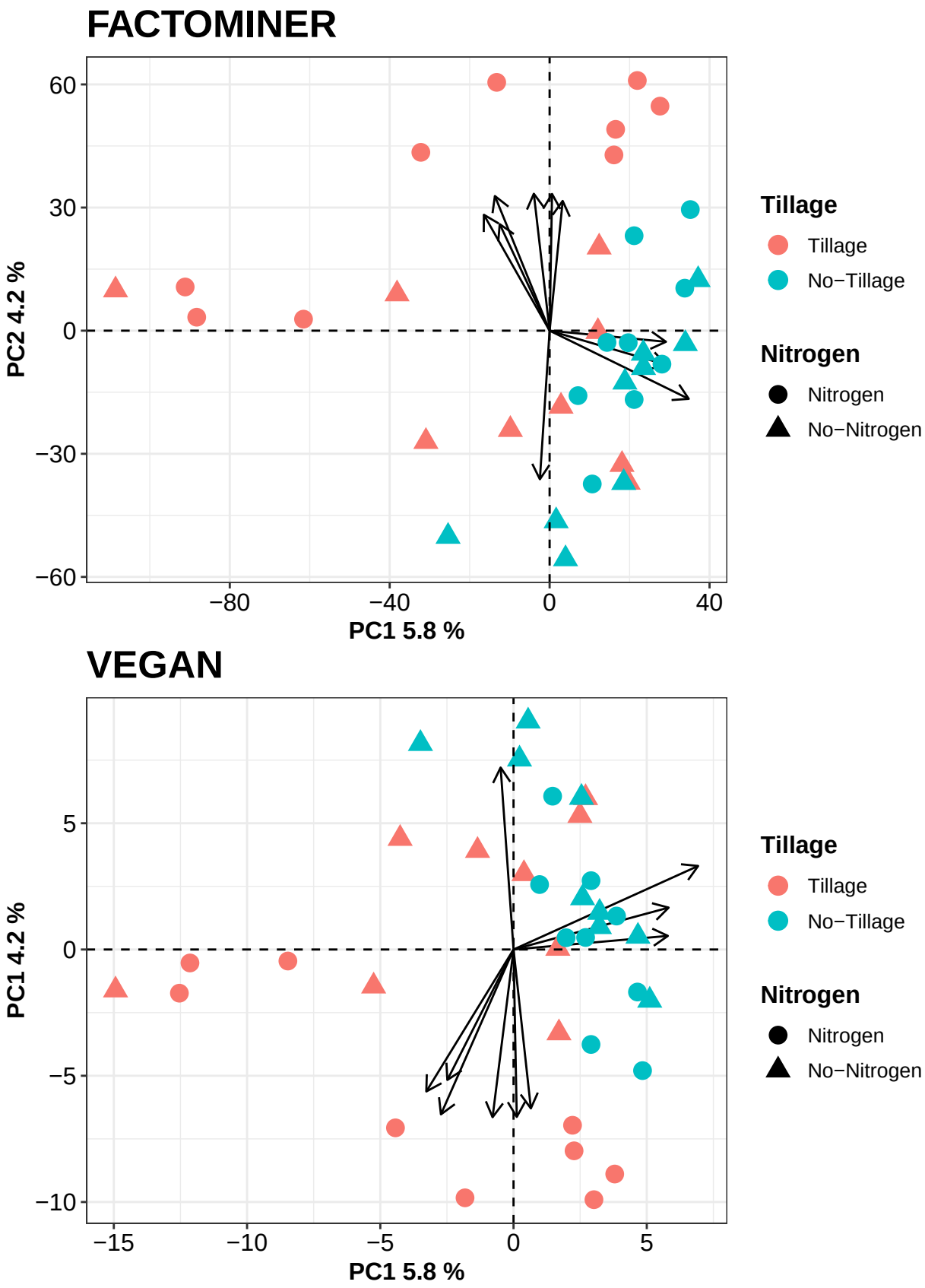
Let's use one of the methods of transformations to evaluate the different methods of PCA analysis , Let's take method 1

```
pca_prcomp<- prcomp(aldex.clr.transform.data)
pca_factominer<- PCA(aldex.clr.transform.data, scale.unit = F, graph = F)
pca_vegan<- rda(aldex.clr.transform.data)
smry <- summary(pca_vegan)


PC1_prcomp<- paste("PC1", round(sum(pca_prcomp$sdev[1] ^ 2) /
                                  mvar(aldex.clr.transform.data) * 100, 1), "%")
PC2_prcomp<- paste("PC1", round(sum(pca_prcomp$sdev[2] ^ 2) /
                                  mvar(aldex.clr.transform.data) * 100, 1), "%")

PC1_factorminer<- paste("PC1", round(pca_factominer$eig[1,2], 1), "%")
PC2_factorminer<- paste("PC2", round(pca_factominer$eig[2,2], 1), "%")

a<- round(100*pca_vegan$CA$eig/sum(pca_vegan$CA$eig),1)[1]
b<- round(100*pca_vegan$CA$eig/sum(pca_vegan$CA$eig),1)[2]


PC1_rda<- paste("PC1", a, "%")
PC2_rda<- paste("PC2",  b, "%")
```

"scale.unit= F is an argument that it's used to avoid scaling due to a normalization was previous done and graph = F in order to no visualize de default plot from PCA function"

Let's plot!


PRCOMP
60
30
0
−30
−60
PC1 4.2 %
−80
−40
0
40
PC1 5.8 %
Tillage
Tillage
No−Tillage
Nitrogen
Nitrogen
No−Nitrogen


FACTOMINER
PC2 4.2 %
PC1 5.8 %
60
30
0
−30
−60
−80
−40
0
40
VEGAN
PC1 4.2 %
PC1 5.8 %
5
0
−5
−10
−15
−10
−5
0
5
Tillage
Tillage
No−Tillage
Nitrogen
Nitrogen
No−Nitrogen

References

Jolliffe, Ian T., and Jorge Cadima. 2016. "Principal Component Analysis: A Review and Recent Developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (2065): 20150202. https://doi.org/10.1098/rsta.2015.0202.