PCA

by Steph and Ale

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]](#footnote-2). 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)

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](https://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

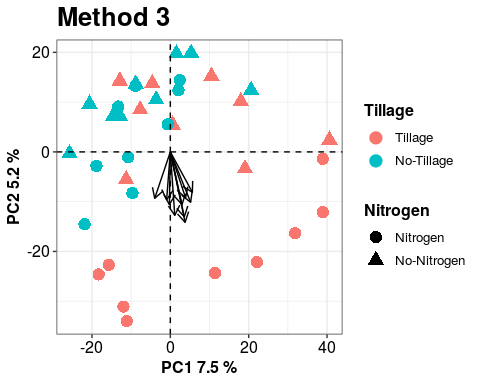
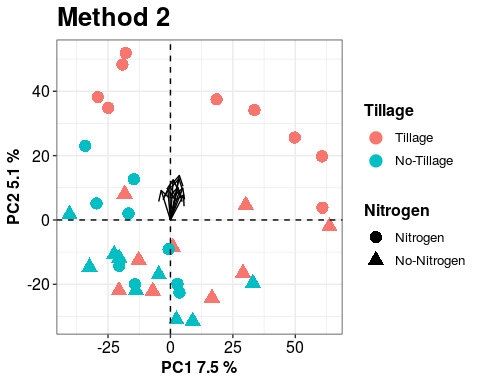
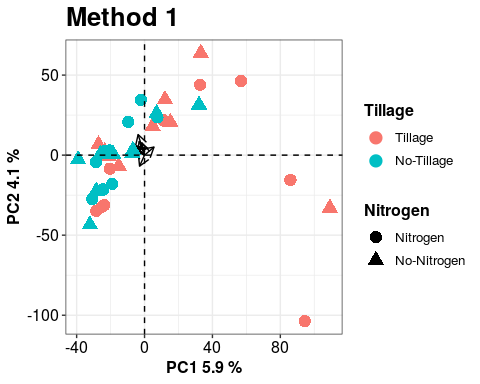
#Formatting metadata variables setting as factors  
metadata$Tillage<- factor(metadata$Tillage,  
 levels = c( "Yes", "No"),  
 labels = c("Tillage", "No-Tillage"))  
  
metadata$Nitrogen<- factor(metadata$Nitrogen,  
 levels = c( "Yes", "No"),  
 labels = c("Nitrogen", "No-Nitrogen"))  
  
#PCa functions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



## **PCA Analysis, comparing PCA methods**

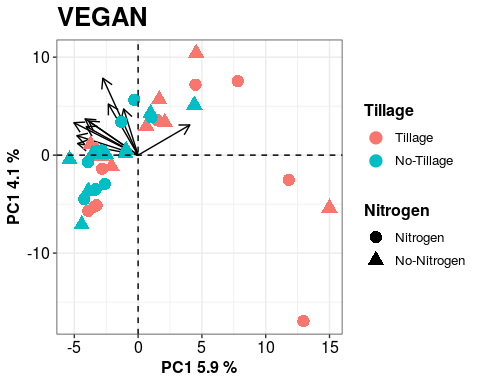
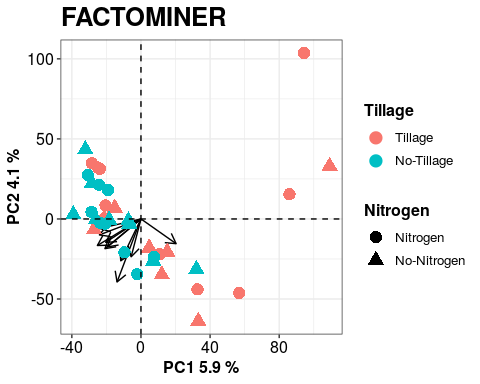
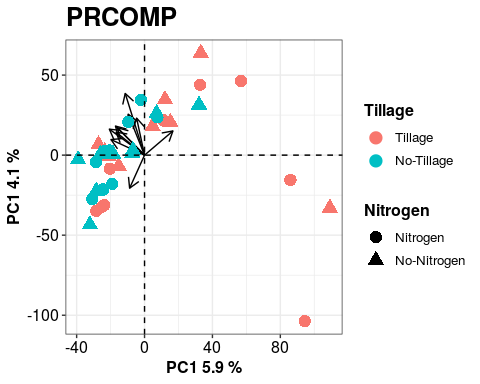
Now Let’s perform PCA analysis and plot but with the different methods of doing PCA’s

###### 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!



## **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>.

1. Data provided from Ph D Alejandra Miranda of her experiment (real data) [↑](#footnote-ref-2)