Multivariate statistical analysis of metagenomic data

Boris Shilov

Introduction

The aim of this analysis is to explore the functional space of the metagenomic dataset assembled by Forslund et al. (2015). My primary objective here is to determine, using dimensionality reduction and ordination methods, whether there is some structure in the data, and to see whether this structure is in any way associated with the treatment/disease status of the samples, and to do this without taking taxonomic information into account (hence the limitation to functional space).

Data format

```r
library(cowplot)
library(tidyverse)
set.seed(100)
data = read_tsv("data.r")
```

The dataset consists of six columns. The first is sample IDs, which are not unique keys since every sample may have multiple features. The second is the dataset of origin, which is either Danish (MHD), Swedish (SWE) or Han Chinese (CHN). The third column is treatment status of a particular patient, where they are either healthy (ND CTRL), are having type 2 diabetes treated with metformin included (T2D metformin+), are having type 2 diabetes treated without metformin (T2D metformin-) or are having type 1 diabetes treated (T1D). The fourth column specifies the type of feature in the fifth column. This is either gut microbial/metabolic module (GMM), SEED database annotation, bacterial family, bacterial genus or metagenomic operational taxonomic units (Motu). The fifth column contains the feature names which are coded differently depending on the feature type. Finally, the sixth column contains the abundance of a given feature, normalised and filtered from low-abundance samples..

The feature types are split into two spaces: the taxonomic space represented by the genuses, families and taxonomic units, and the functional space as annotated by SEED and GMM. The full SEED and GMM feature code annotations can be found in the appendix files, taken from Supplementary Table 10 of Forslund et al. (2015). I am only interested in exploring the structure of the functional space.

```r
functional_space = filter(data, FeatureType %in% c("GMM", "SEED"))
functional_space
```

```
## # A tibble: 413,168 x 6
##    Sample Dataset Status  FeatureType Feature Abundance
##    <chr>  <chr>   <chr>   <chr>       <chr>       <dbl>
##  1 MH0443 MHD     ND CTRL GMM         MF0119        902
##  2 MH0443 MHD     ND CTRL GMM         MF0090        302
##  3 MH0443 MHD     ND CTRL GMM         MF0097          0
##  4 MH0443 MHD     ND CTRL GMM         MF0092          0
##  5 MH0443 MHD     ND CTRL GMM         MF0011       1076
##  6 MH0443 MHD     ND CTRL GMM         MF0083          3
##  7 MH0443 MHD     ND CTRL GMM         MF0110          0
##  8 MH0443 MHD     ND CTRL GMM         MF0063          4
##  9 MH0443 MHD     ND CTRL GMM         MF0042          1
## 10 MH0443 MHD     ND CTRL GMM         MF0024        185
## # ... with 413,158 more rows
```

In order to perform analysis, I transform the data into the data matrix X, with every row being a sample and every column a particular feature. First I combine redundant columns. Some ecological statistical functions will assume the transpose of this format, so I will stick to using a data frame after initial processing here - tibbles are not transposed easily due to not implementng row names. This is known as a community or abundance matrix in ecology.

```
united_functional_space = functional_space %>%
  unite(SampleInfo, Sample, Dataset, Status, sep="~") %>%
  unite(FeatureInfo, FeatureType, Feature, sep="~")
X = united_functional_space %>% group_by(FeatureInfo) %>% spread(FeatureInfo, Abundance)
```

There is significant missingness in 25 of the individuals, and we exclude them from further analysis.

```
X = drop_na(X)
separated_X = X %>% separate(SampleInfo, into=c("Sample", "Dataset", "Status"), sep="~")
X.df = X %>% column_to_rownames("SampleInfo") %>% as.data.frame()
```

```
sum(X==0)/(dim(X)[1] * dim(X)[2])
```

```
## [1] 0.3054907
```

30% of our matrix is sparse.

Analysis

Log transformation of the data

As we have an abudance matrix, we need to transform the data to be linear. We can do this using a log transformation, in particular the centered log ratio transform. In order to log transform, we first need to get rid of the zeroes, which we use Bayesian multiplicative replacement for. Normally we would have to normalise and filter the data for low-abundance samples, but this has already been done for us.

```
library(zCompositions)
X.df.czm = cmultRepl(X.df,  label=0, method="CZM")
```

```
## No. corrected values:  29
```

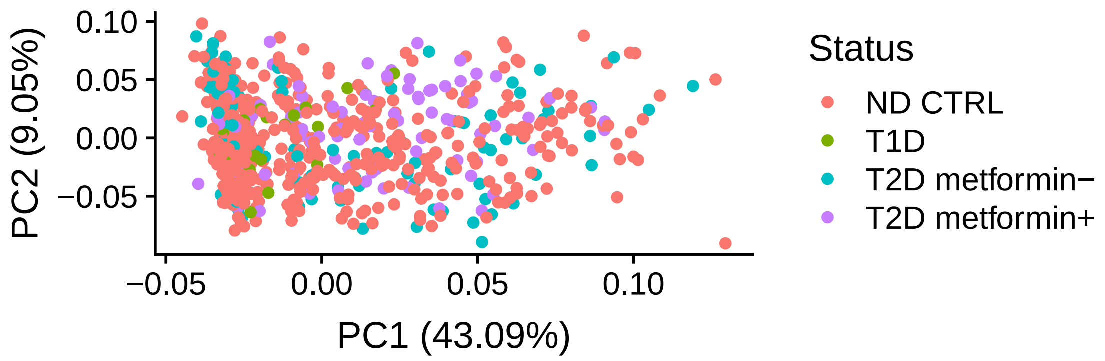
```
X.df.clr = t(apply(X.df.czm, 1, function(x) {log(x) - mean(log(x))} ))
annotated_X.clr = bind_cols(separated_X[1:3], as.tibble(X.df.clr))
```

PCA

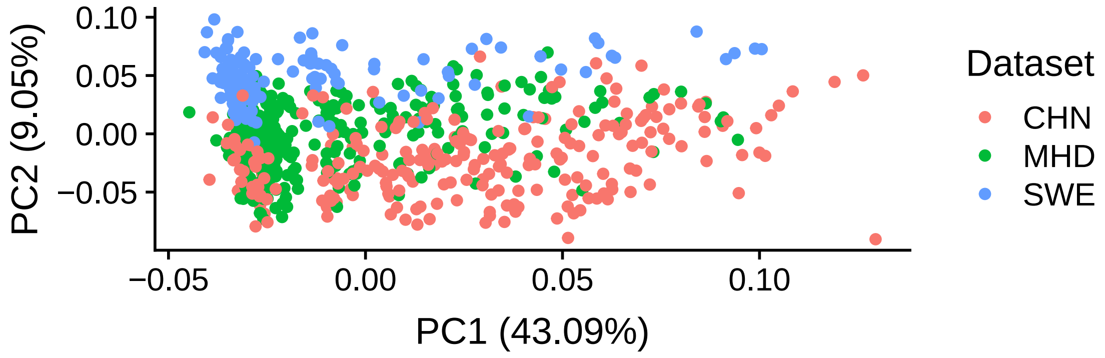
I attempt PCA on the transformed data, scaling to unit variance.

```
library(ggfortify)
library(nFactors)
library(GGally)
X.df.clr.PCA1 = prcomp(X.df.clr, scale=T)
```

```
autoplot(X.df.clr.PCA1, data=annotated_X.clr, colour="Status", x=1, y=2)
```

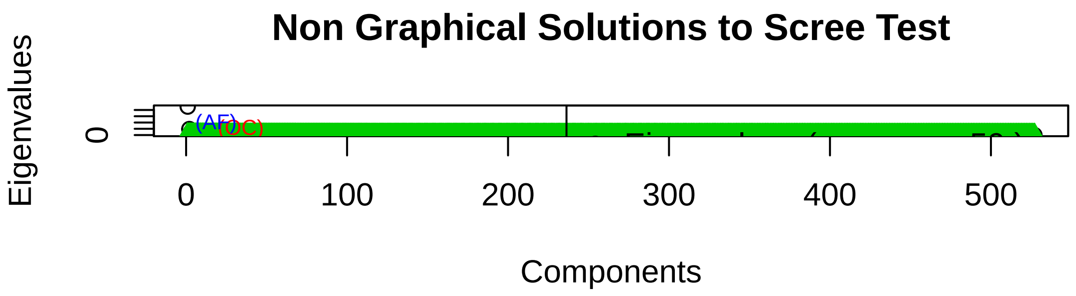


```
autoplot(X.df.clr.PCA1, data=annotated_X.clr, colour="Dataset", x=1, y=2)
```



The first component accounts for a lot of variance. It would be informative to attempt parallel analysis to determine the number of principal components to retain. This technique is a formalisation of the Scree plot and works by simulating a random data frame with the same number of of samples and variables as in the original data frame, computing a correlation matrix for this random d.f. We use the eigenvalues from this simulation to decide if the components in our real analysis are likely to be random noise.

```
parallelAnal1 = parallel(subject=nrow(X.df.clr), var=ncol(X.df.clr), rep=100)
screeTest1 = nScree(x=eigen(cor(X.df.clr))$values, aparallel=parallelAnal1$eigen$qevpea)
plotnScree(screeTest1)
```



This analysis suggests around 16 components would capture most of the non-random variability.

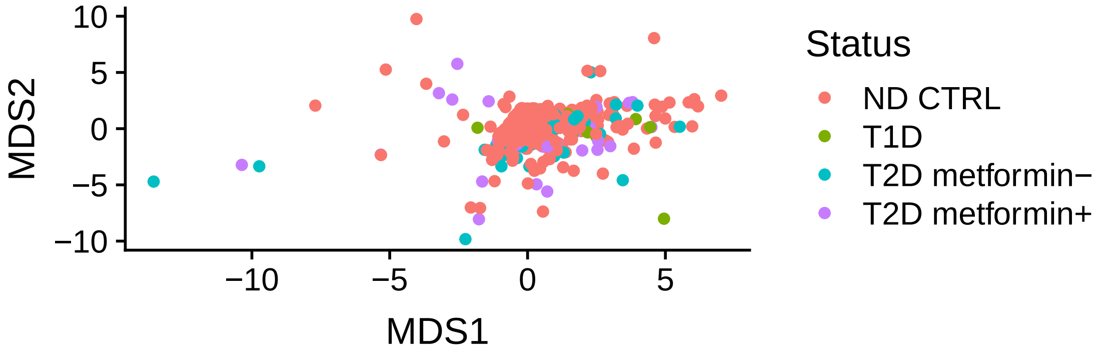
Non-Metric Multidimensional Scaling

I use the Canberra distance, a weighted Manhattan distance, as originally suggested by Forslund et al. (2015). metaMDS required original community matrices and does not accept negative values, hence we do not use the log transformed matrix.

```
library(vegan)
mds1 = metaMDS(t(X.df), k=2, trymax=50, distance="canberra")
mds1_results = as.tibble(mds1$species, rownames="SampleInfo")
```

```
mds1_results_separated = mds1_results %>% separate(SampleInfo, into=c("Sample", "Dataset", "Status"), s
```

```
ggplot(mds1_results_separated, aes(x=MDS1, y=MDS2,colour=Status)) + geom_point()
```

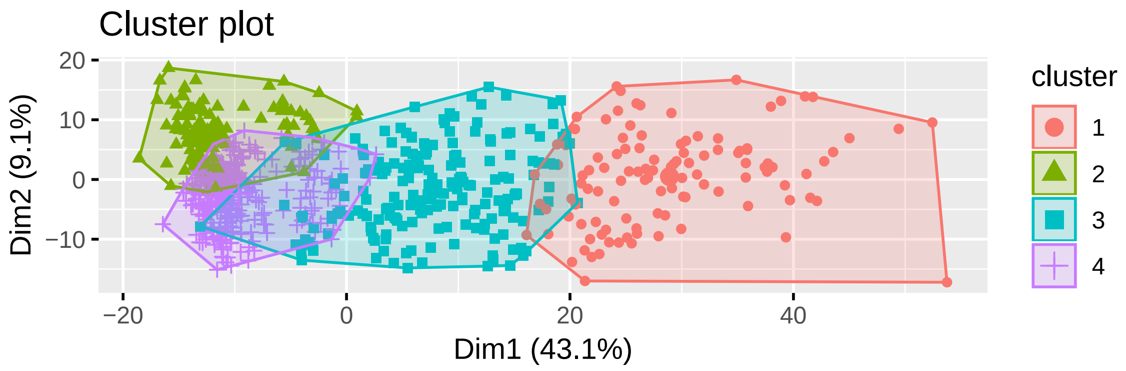


Unfortunately there is no convergence, so I cap the number of runs at 50.

k-Means Clustering

I am interested in finding four clinical groups, hence I specify four clusters to be found via k-Means. I then visualise the clusters in two dimensions using a PCA.

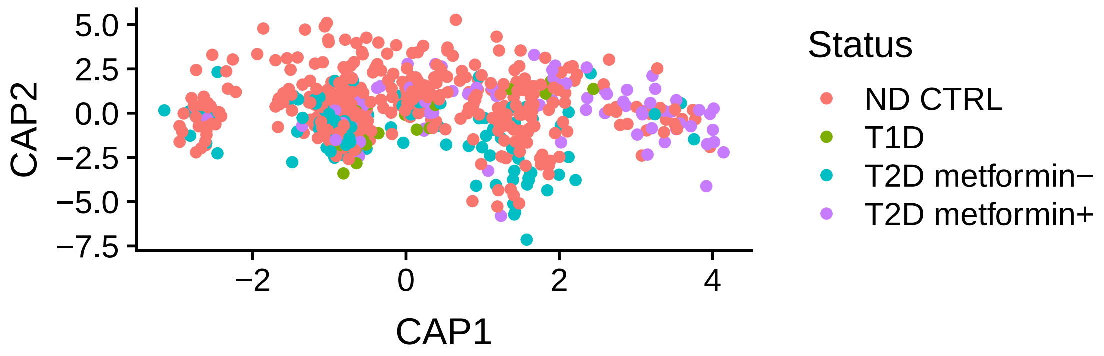
```
library(cluster)
library(factoextra)
fourMeansLog = kmeans(X.df.clr, centers=4)
fviz_cluster(fourMeansLog, data=X.df.clr, geom="point")
```



Canonical Analysis of Principal Coordinates

It should perhaps be possible to account for the considerable batch effect (as seen in the Dataset variable correlating with most of the observed structure) by conditioning on Dataset in a constrained ordination.

```
constrainedModel3 = capscale(X.df.clr~ Status + Condition(Dataset),
                             data=annotated_X.clr, distance="canberra")
constrainedModel3Summary = summary(constrainedModel3)
CAP3_results=bind_cols(annotated_X.clr[,1:3],as.tibble(scores(constrainedModel3)$sites))
ggplot(CAP3_results, aes(x=CAP1, y=CAP2,colour=Status)) + geom_point()
```



Conclusion

It is quite easy to see that there is indeed structure in this community matrix. It is perhaps not unexpected that ecology would exert some pressure on the SEED and GMM genetic elements. Despite my best efforts, I have been unable to correlate this structure with clinical status. The PCA clearly reveals Dataset structure, but not Status. Multidimensional scaling creates a very neat separation - on Dataset, however it does not converge. 4-means clustering reveals clusters that roughly correspond to separate Dataset levels on the PCA. To my surprise, even implementing a constrained ordination did not reveal non-random clinical Status structure between the CAPs, despite the fact that I conditioned on the Dataset of origin.

This was to be expected, to some extent. Microbiome data is quite infamous for having strong batch effects merely from contamination during sample handling alone. Here, the Dataset variable reflected different collection centers, so this played a role. In addition to that, even in these collection centers, samples were taken from individuals of vastly different demographics, differing in age, sex, ethnicity, and other important variables that have not been provided. It is highly likely that the Dataset is so important because all of these have an influence.

I must cautiously conclude that, given my present knowledge, I cannot discern any association between the genetic structure of the microbiome and diabetes status.

Bibliography

Forslund, Kristoffer, Falk Hildebrand, Trine Nielsen, Gwen Falony, Le ChatelierEmmanuelle, Shinichi Sunagawa, Edi Prifti, et al. 2015. "Disentangling Type 2 Diabetes and Metformin Treatment Signatures in the Human Gut Microbiota." *Nature* 528. https://doi.org/10.1038/nature15766.