Multivariate statistical analysis of metagenomic data

Boris Shilov

Introduction

The aim of this analysis is to explore the metagenomic dataset assembled by Forslund et al. (n.d.).

Data format

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

```
## # A tibble: 1,178,352 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 1,178,342 more rows
```

We can see that 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.

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. (n.d.). We are only interested in exploring the structure of the functional space.

```
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 actually perform analysis, we transform the data into the data matrix X, with every row being a sample and every column a particular feature. First we combine redundant columns.

```
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="~")
```

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

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

30% of our matrix is sparse.

Analysis

CoDa

```
library(zCompositions)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: NADA
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'NADA'
```

```
## The following object is masked from 'package:stats':
##
##     cor
```

```
## Loading required package: truncnorm
```

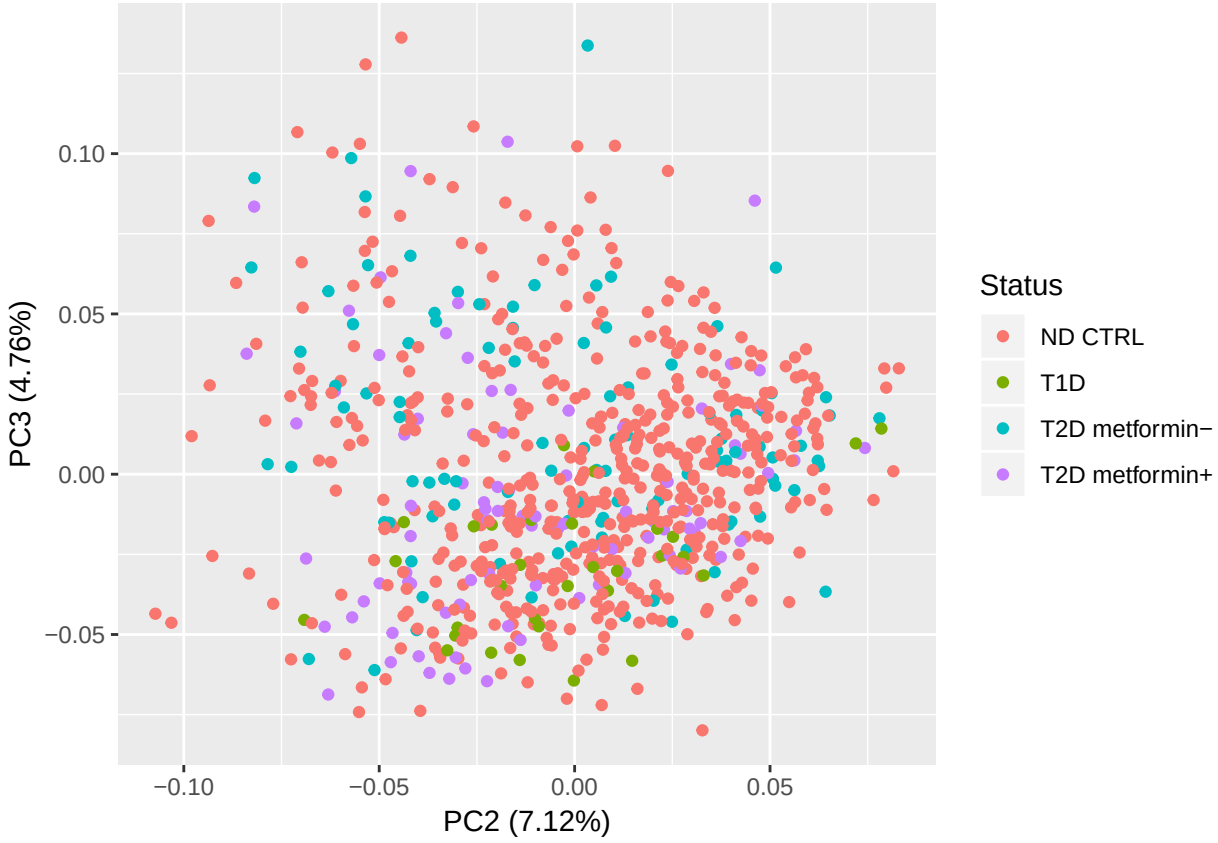
```
library(ggfortify)
X.df.transposed = X %>% column_to_rownames("SampleInfo") %>% as.data.frame()
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
X.df = t(X.df.transposed)
X.df.czm = cmultRepl(t(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, as.tibble(X.df.clr))
X.df.clr.PCA = prcomp(X.df.clr)
PCA.rotation.transpose = t(X.df.clr.PCA$rotation)
autoplot(X.df.clr.PCA, data=annotated_X.clr, colour="Status", x=2, y=3)
```



PCA

```
train = X %>% sample_frac(0.8)
test = anti_join(X, train, by="SampleInfo")
```

```
library(nFactors)
```

```
## Loading required package: psych
```

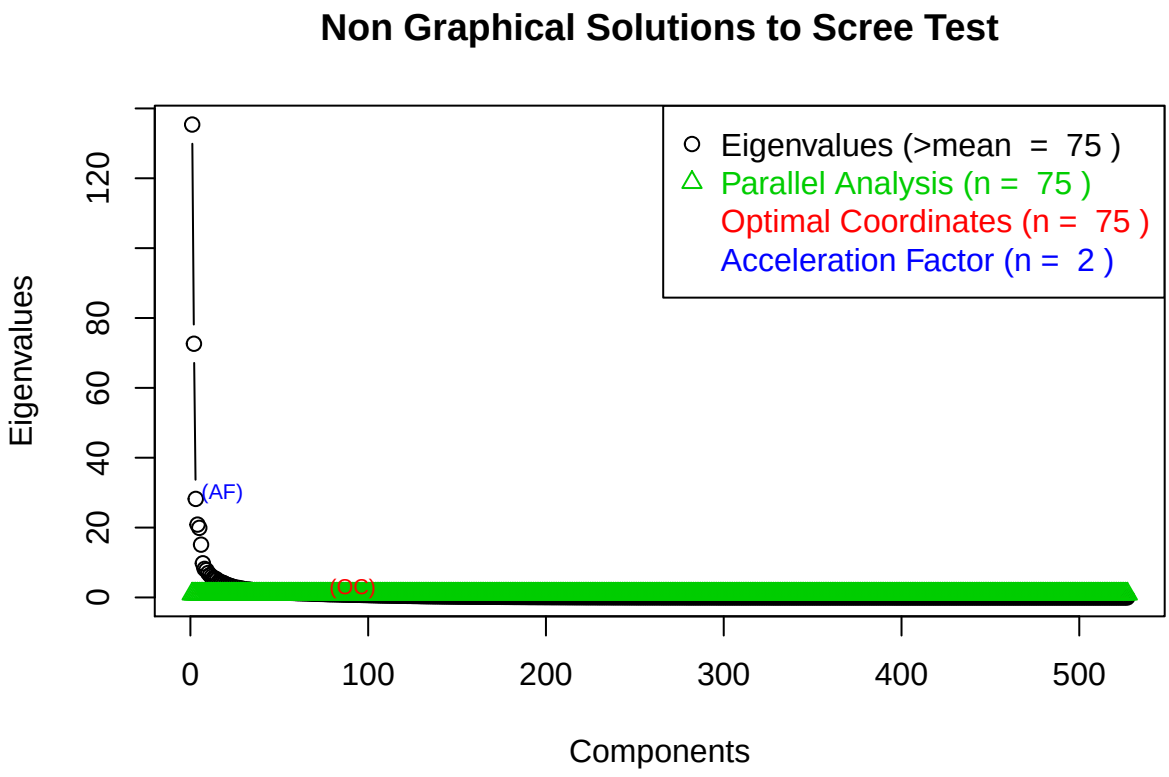
```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
## Loading required package: boot
```

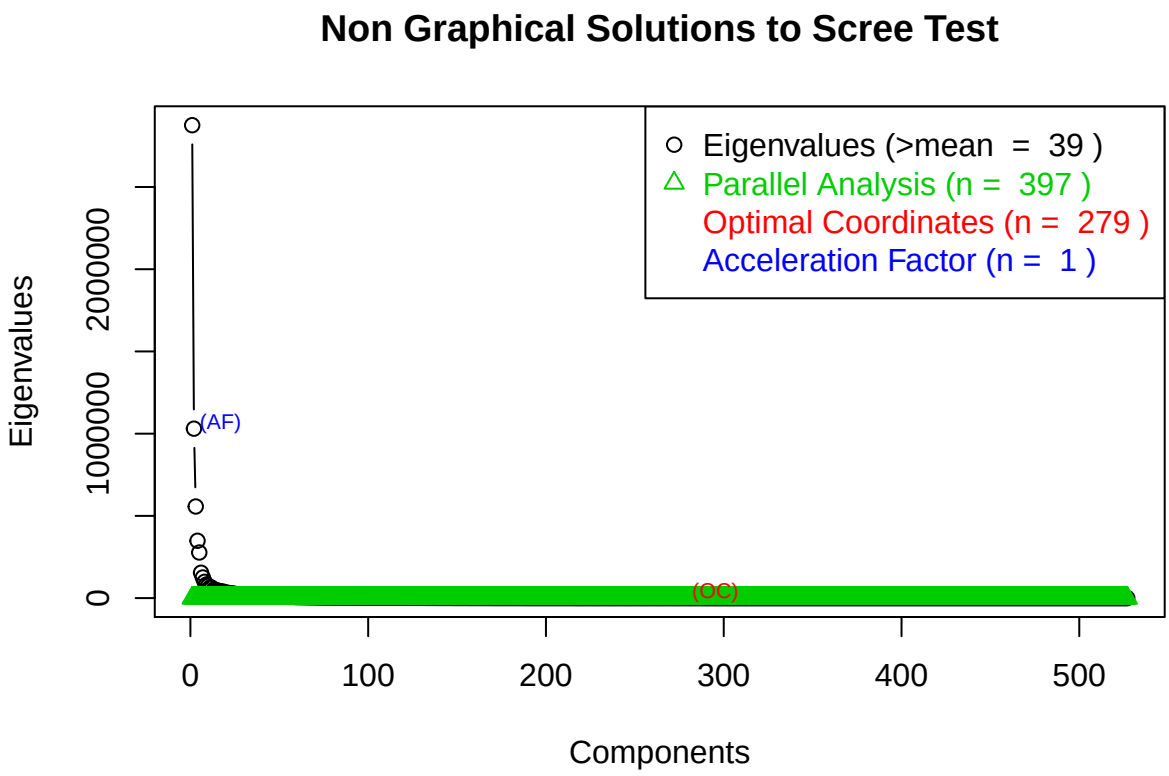
```
##
## Attaching package: 'boot'

## The following object is masked from 'package:psych':
##
##     logit

## The following object is masked from 'package:survival':
##
##     aml

## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':
##
##     melanoma

##
## Attaching package: 'nFactors'

## The following object is masked from 'package:lattice':
##
##     parallel
```

```
library(ggfortify)
eigenvals = eigen(cor(X[-1]))
eigenvaldist = parallel(subject=nrow(X[-1]), var=ncol(X[-1]), rep=100)
nS = nScree(x=eigenvals$values, aparallel=eigenvals$eigen$qevpea)
plotnScree(nS)
```



```
cov1 = cov(train[-1])
cor1 = cov2cor(cov1)
eigens = eigen(cov1)
PCA1 = prcomp(train[-1], scale=F, center=T)
```

```
PCA1_parallel = parallel(subject=nrow(train[-1]), var=ncol(train[-1]), rep=100)
PCA1_scree = nScree(x=PCA1$sdev^2, aparallel=PCA1_parallel$eigen$qevpea)
plotnScree(PCA1_scree)
```



This suggests it would be optimal to retain in the hundreds of components to explain most of the variance. Instead, we can use the Kaiser-Guttman rule to select the first 40 or so components, which explain about 93% - a large reduction in factors in exchange for a small reduction in the explained variance.

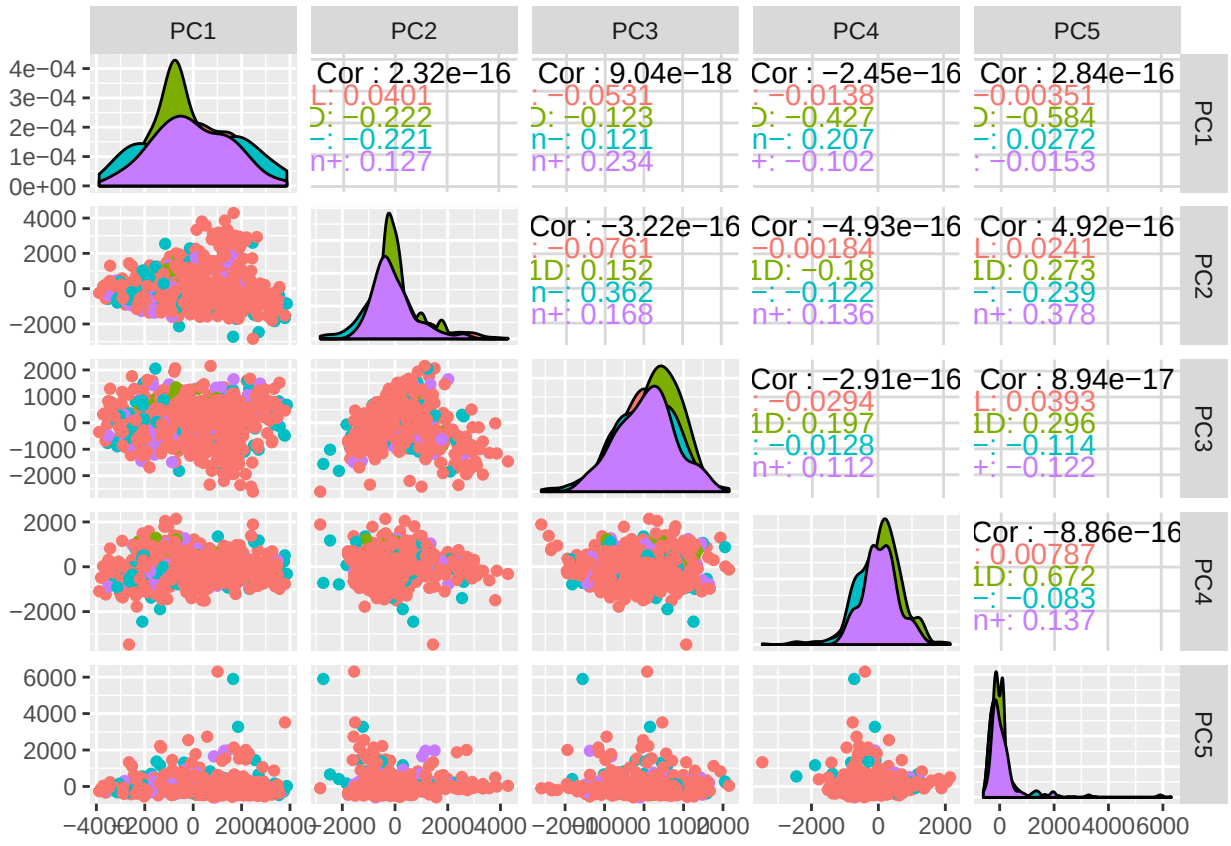
```
PCA2 = prcomp(X[-1], scale=F, center=T, tol=0.07)
```

```
library(GGally)
```

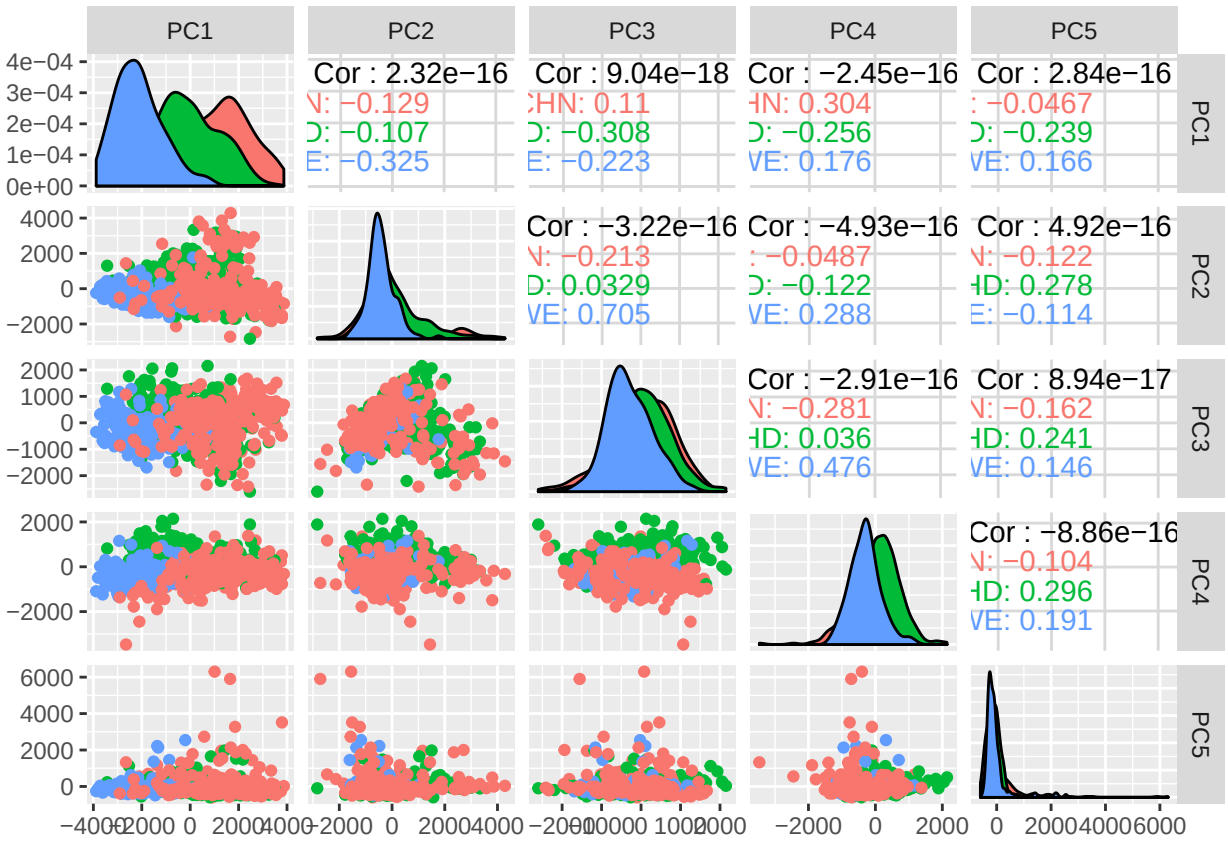
```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
PCA2_and_desc = bind_cols(as_tibble(PCA2$x), separated_X)
ggpairs(PCA2_and_desc, aes(colour=Status), columns=1:5, progress = FALSE)
```



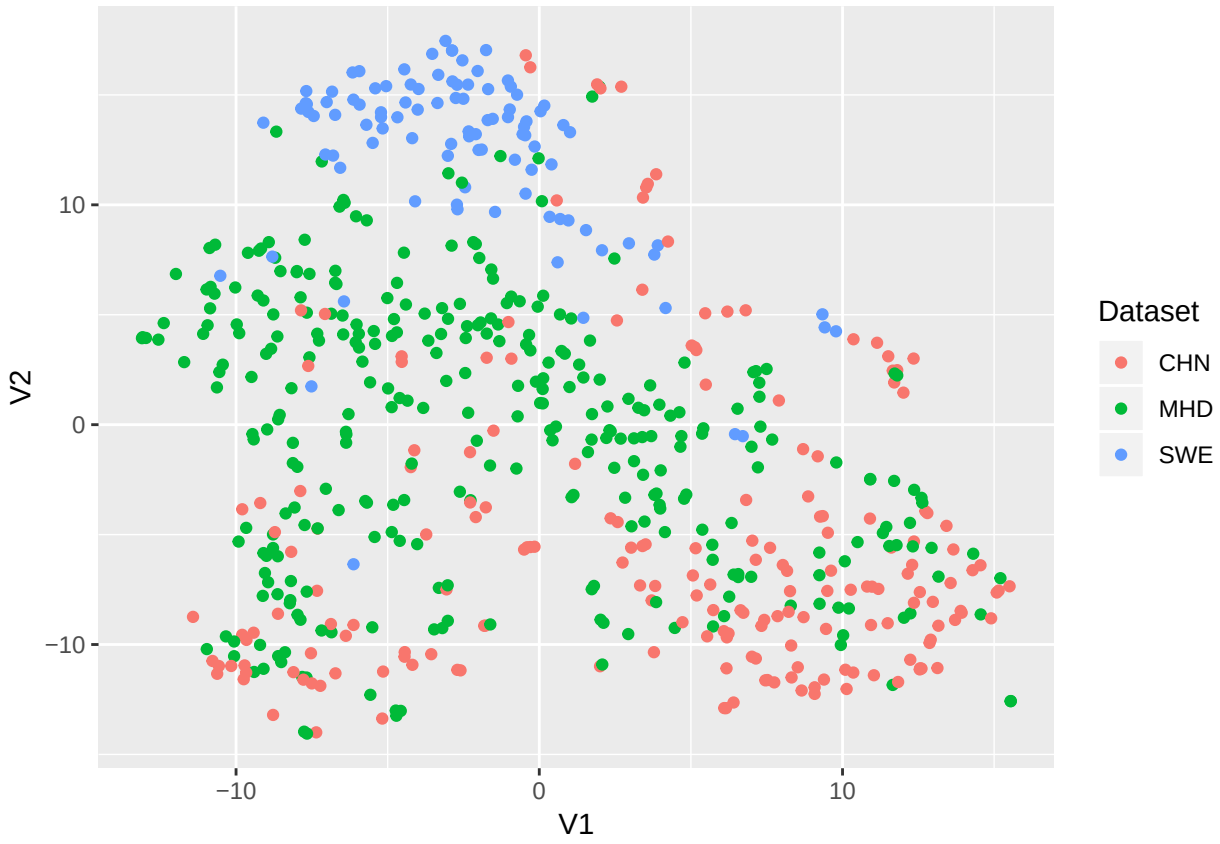

```
ggpairs(PCA2_and_desc, aes(colour=Dataset), columns=1:5, progress = FALSE)
```



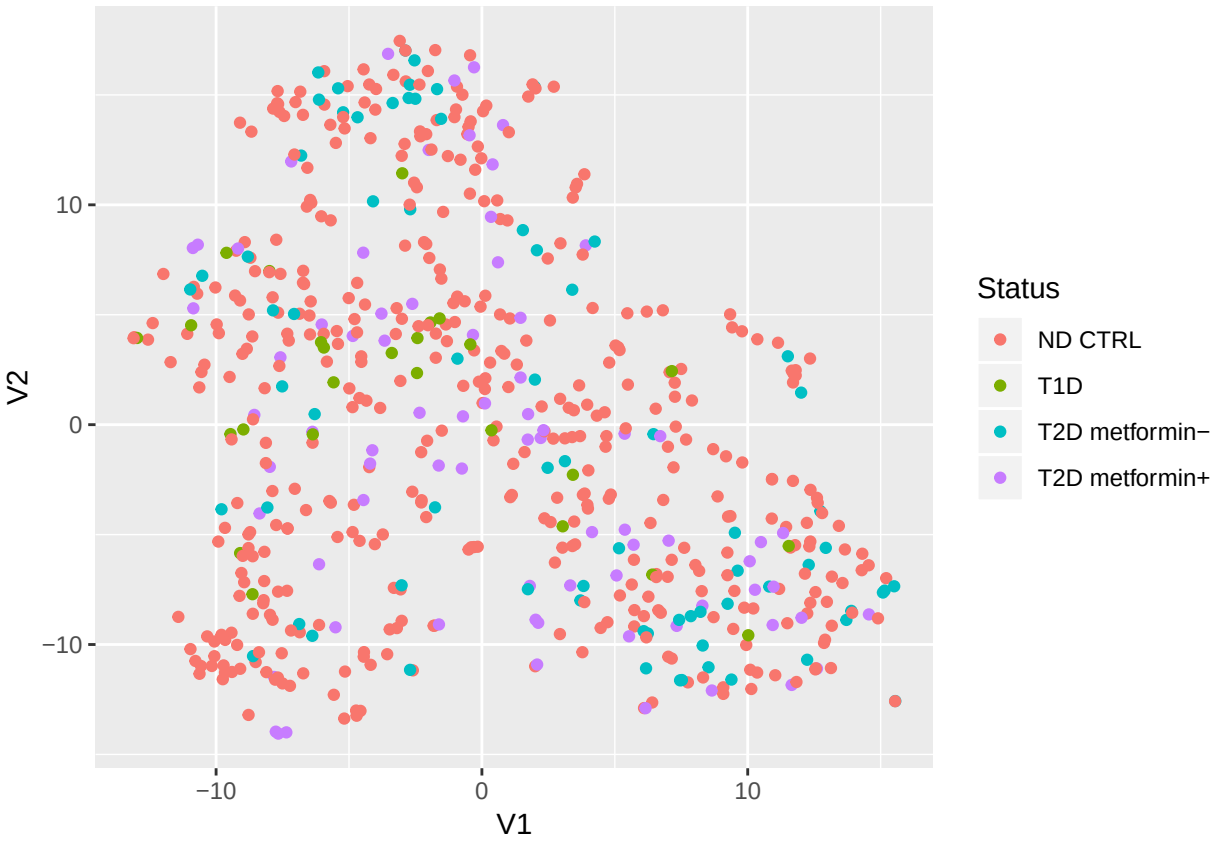
PCA appears to mostly separate based on the geographical origin rather than clinical status, which is suboptimal.

tSNE

```
library(Rtsne)
tsne1 = Rtsne(train[,-1], dims=2, perplexity=30, max_iter=400)
tSNE1_res = as_tibble(tsne1$Y)
separated = train[,1] %>% separate(SampleInfo, into=c("Sample", "Dataset", "Status"), sep="~")
tsne_with_additionals = bind_cols(tSNE1_res, separated)
ggplot(tsne_with_additionals, aes(x=V1, y=V2,colour=Dataset)) + geom_point()
```




```
ggplot(tsne_with_additionals, aes(x=V1, y=V2,colour=Status)) + geom_point()
```



t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction method. We embed the data into two dimensions. There is very clearly some structure to the data, we see that tSNE mostly discovers the geographic origin, and treatment status does not seem to explain much of the structure.

Constrained ordination

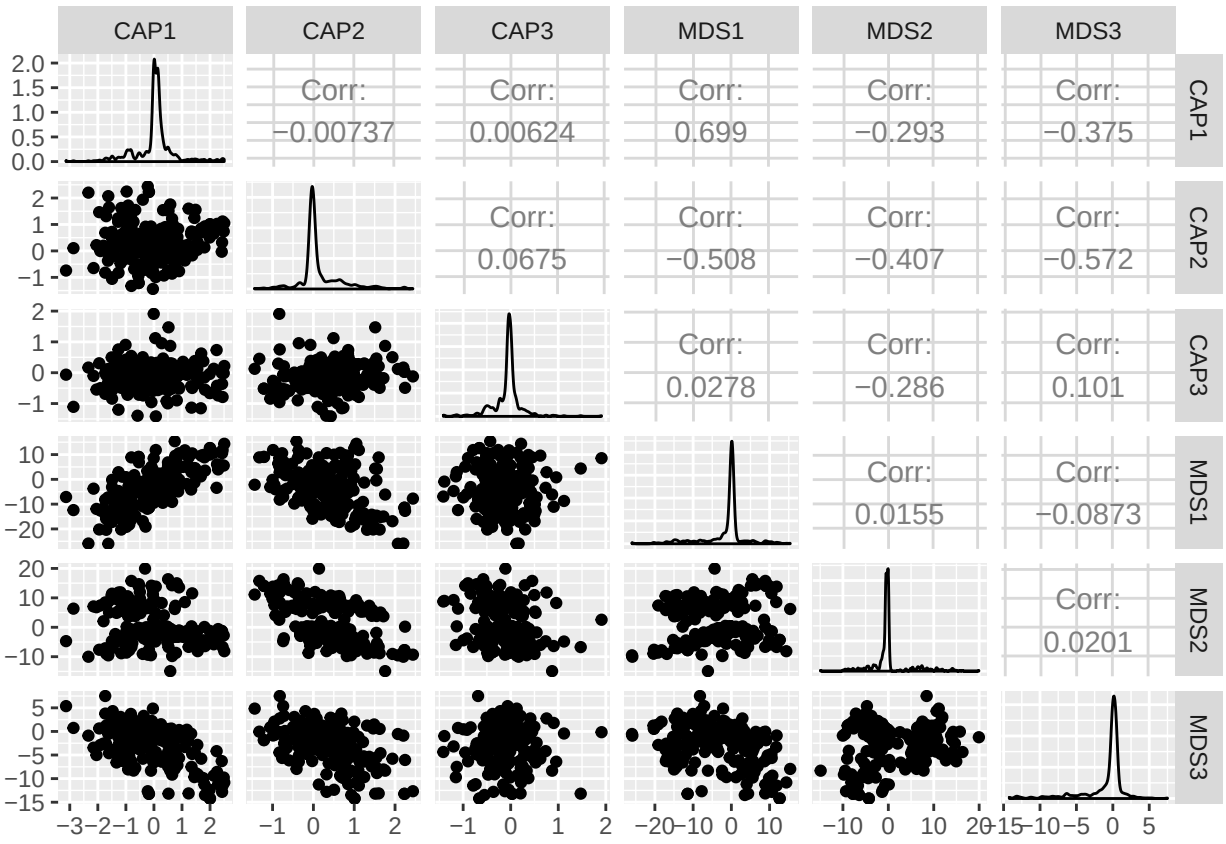
It is clear from our previous attempts that running unconstrained ordination on this dataset merely reveals considerable bias in collection methods accross different country sites, with also perhaps some country-specific structure to the microflora of the patient guts. We are really, however, interested in seeing the clinical structure in this data. Thus we have to substract or account for the collection and geographic bias. We do this using constrained ordination, conditioning on the Dataset variable.

```
library(vegan)
```

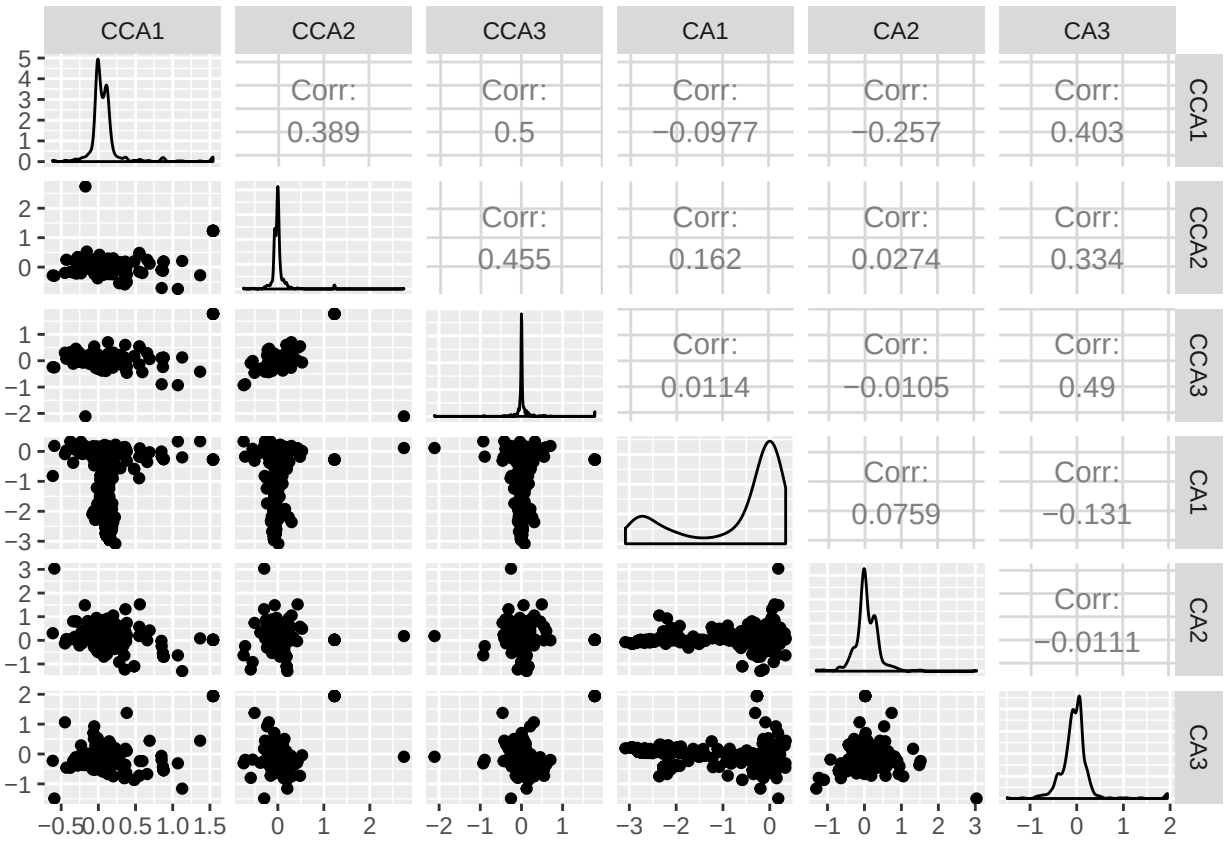
```
## Loading required package: permute
```

```
## This is vegan 2.5-3
```

```
constrainedModel1 = capscale(separated_X[, -c(1,2,3)] ~ Status + Condition(Dataset), data=separated_X)
constrainedModel1Summary = summary(constrainedModel1)
ggpairs(as.tibble(constrainedModel1Summary$species), progress = FALSE)
```




```
constrainedModel2 = cca(separated_X[, -c(1,2,3)] ~ Status + Condition(Dataset), data=separated_X)
constrainedModel2Summary = summary(constrainedModel2)
ggpairs(as.tibble(constrainedModel2Summary$species), progress = FALSE)
```



Bibliography

Forslund, Kristoffer, Falk Hildebrand, Trine Nielsen, Gwen Falony, Le ChatelierEmmanuelle, Shinichi Sunagawa, Edi Prifti, et al. n.d. "Disentangling Type 2 Diabetes and Metformin Treatment Signatures in the Human Gut Microbiota." *Nature* 528: 262 EP. https://doi.org/10.1038/nature15766.