Package **CompSign**

Lena Morrill

October 2017

CompSign is a toolkit for the analysis of mutational signatures with an emphasis on the compositional analysis of exposures. An overview of the compositional nature of the exposures to mutational signatures, which has been often overlooked, is found elsewhere[1].

Contents

1 Installation **1**

2 Datasets **2**

3 Create a `CompSign` object **3**

4 Battery of tests **5**
4.1 Test for normality 5
4.2 Test for equality 5

5 Clustering of samples **6**

6 Symmetric balances for correlation between signatures **8**

7 Optimal selection of partition **10**

8 Dimensionality reduction: PCA **13**

9 Session info **15**

1 Installation

`CompSign` can be installed as usual from github:

```
library(devtools)
devtools::install_github("lm687/CompSign")
```

[1] cite myself

```r
library(CompSign)
library(compositions)
library(MCMCpack)      ## for sampling from Dirichlet
library(ggplot2)
library(gridExtra)
library(ComplexHeatmap)
library(circlize)
```

2 Datasets

```
## if the folder data/ is not in github
for(i in list.files("../data/", pattern = "*rda", full.names = TRUE)){load(i)}
```

The package contains several datasets of exposures to mutational signatures and metadata of the corresponding samples. These datasets are:

- SNV signatures
  - Data for 560 breast cancer patients

```
# data("Breast560")
metadataBreast560 <- metadata(Breast560)
exposuresBreast560 <- count_matrix(Breast560)
dim(metadataBreast560); dim(exposuresBreast560)

## [1] 560  47
## [1] 560  12
```

  - Pan-cancer from EMu[2] (cite dataset)
- Copy Number signatures
  - Ovarian cancer-derived signatures, as described in [4]. It contains data for 12k TCGA samples.

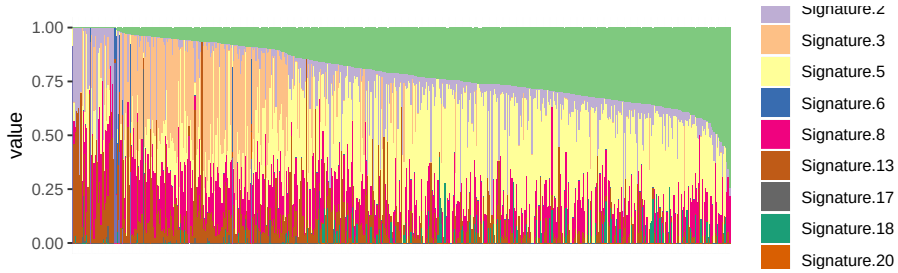
```
#data("CNA_12K_TCGA")
metadataCNA_12K_TCGA <- metadata(CNA_12K_TCGA)
exposuresCNA_12K_TCGA <- count_matrix(CNA_12K_TCGA)
```

  - Pan-cancer copy number signatures
- Synthetic data

```
dim(metadata(two_normal_pops))

## [1] 2000    1

dim(count_matrix(two_normal_pops))

## [1] 2000    3

#data(two_normal_pops)
```

Data can be visualised as follows add this to the functions of the package

```
source("../../CDA_in_Cancer/code/functions/meretricious/pretty_plots/prettySignatures.R", pr
createBarplot(count_matrix(Breast560), remove_labels = TRUE,
              order = names(sort(count_matrix(Breast560)[,1])))

## Creating plot... it might take some time if the data are large. Number of samples: 560
```



3 Create a CompSign object

This is a minimal example for transforming a matrix into a *sign* object

```
basic_matrix <- matrix(runif(12), nrow = 4)
colnames(basic_matrix) <- paste0('s', 1:3)
rownames(basic_matrix) <- paste0('Sample ', 1:4)
basic_sign <- to_sign(basic_matrix)
basic_sign

## An object of class "sign"
## Slot "id":
## [1] "basic_matrix"
##
## Slot "id_samples":
## [1] "Sample 1" "Sample 2" "Sample 3" "Sample 4"
##
## Slot "id_signatures":
## [1] "s1" "s2" "s3"
##
## Slot "count_matrix":
##                  s1        s2        s3
## Sample 1 0.3038001 0.5133714 0.9359289
## Sample 2 0.3554818 0.4216384 0.8309738
## Sample 3 0.5588399 0.9809409 0.1313172
## Sample 4 0.5800957 0.2018036 0.1685508
##
## Slot "modified":
## [1] FALSE
```

A *sign* object can be summarised as follows:
add_together_matrix?? what is this?

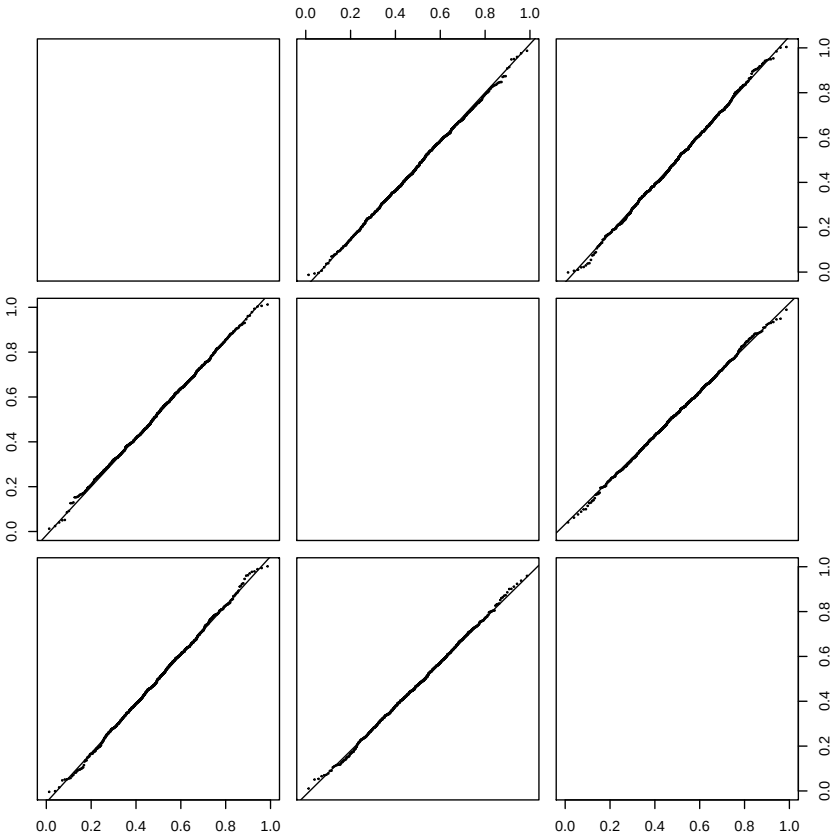
```
results_sumarise <- summarise(add_together_matrix(basic_sign))
results_sumarise

## $General
## [1] "Object of class sign"
##
## $`Number of signatures`
## [1] 3
##
## $`Number of samples`
## [1] 4
##
## $`Geometric means of signatures`
##        s2        s1        s3
## 0.4549733 0.4325617 0.3622185
##
## $Covariance
##              s1           s2          s3
## s1  0.019692778  0.006351886 -0.05941119
## s2  0.006351886  0.107692674 -0.03351665
## s3 -0.059411195 -0.033516650  0.18141621
```

4 Battery of tests

This section takes largely from Aitchison's pioneering work[1] and its succession[5].

4.1 Test for normality

```
par(mfrow=c(1,2))
# qqnorm.acomp(acomp(two_normal_pops@count_matrix), pch=19, cex=0.2)
qqnorm.acomp(acomp(two_normal_pops@count_matrix[1:1000,]), pch=19, cex=0.2)
```



4.2 Test for equality

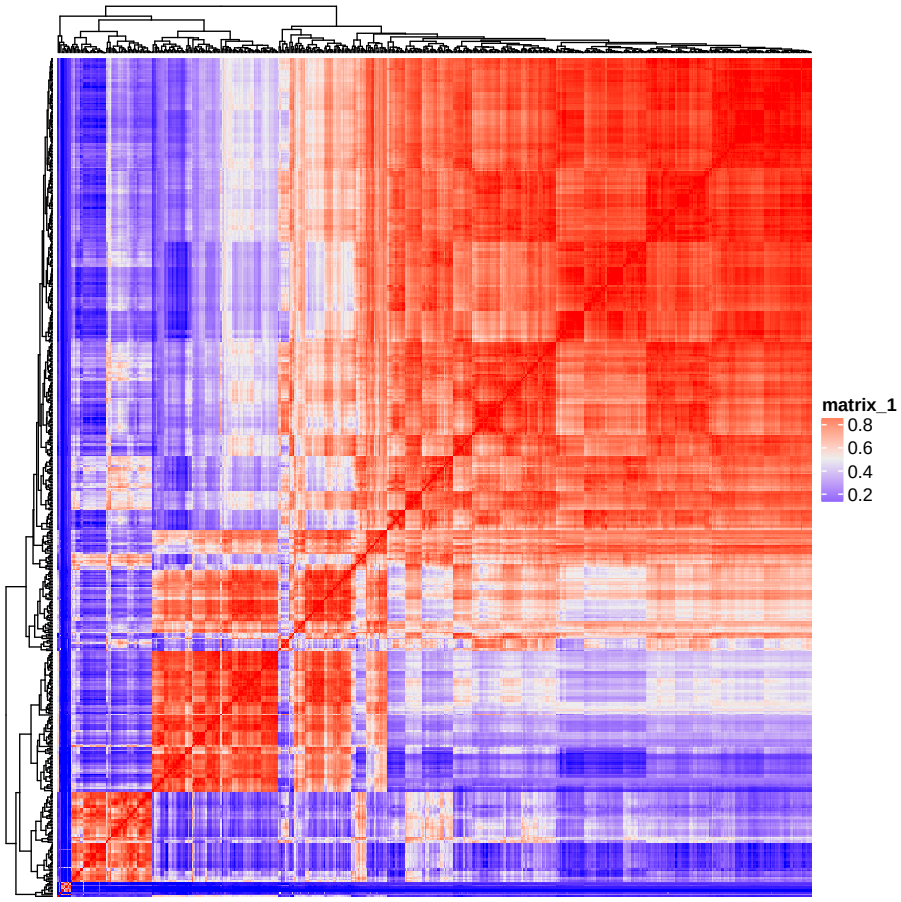
Test for equality of means.

```
compare_populations(predictors = count_matrix(Breast560),
                    response = as.numeric(as.factor(metadata(Breast560)$final.ER)))

## $note
## [1] "James test"
##
## $mesoi
##                   X1         X2       X3           X4        X5       X6
## Sample 1 -1.6969917 -0.9470633 1.923296 -0.103998542 0.2707288 1.528197
## Sample 2  0.8022291 -3.6033765 2.143807  0.002721535 0.6369184 1.607875
##                 X7       X8       X9      X10
## Sample 1 1.271388 1.248926 1.016479 1.017358
## Sample 2 1.086473 1.283400 1.083742 1.048809
##
## $info
##               test            p-value         correction
##       2.359067e+02       5.324172e-43       1.042820e+00
## corrected.critical
##       1.909095e+01
```

5 Clustering of samples

Samples can simply be clustered by the cosine similarity of their exposures, as done in Ren et al.

```
res_outerCosineSimilaritySNV <- outerCosineSimilarity(exposuresBreast560, exposuresBreast560

## [1] 560  12
## [1] 560  12

ComplexHeatmap::Heatmap(res_outerCosineSimilaritySNV)
```




```
# res_outerCosineSimilarityCNA <- outerCosineSimilarity(exposuresCNA_12K_TCGA[metadataCNA_12
#                                                       exposuresCNA_12K_TCGA[metadataCNA_12
#                                                       verbose=FALSE)
# ComplexHeatmap::Heatmap(res_outerCosineSimilarityCNA)
```

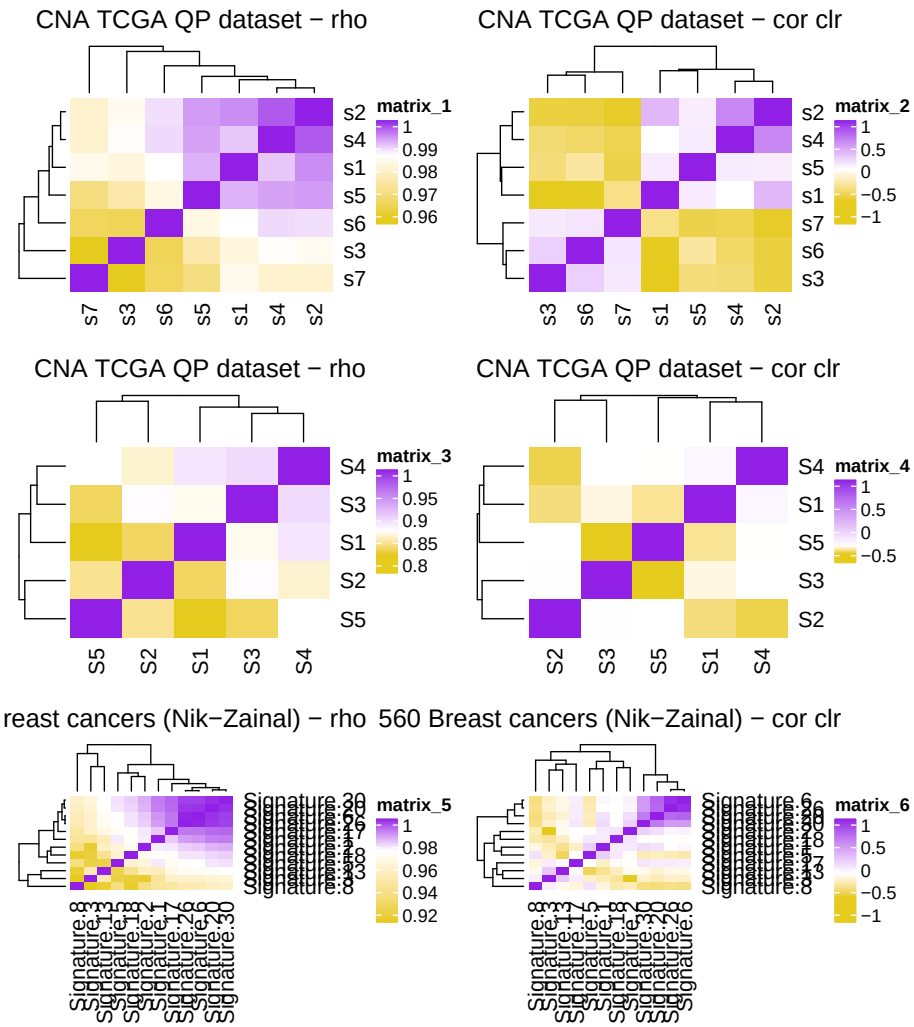
6 Symmetric balances for correlation between signatures

Pearson pointed out ([6]) that spurious correlations appear when closed, compositional, data, are analysed. Therefore normal correlation coefficients can be ruled out of our analysis.

I implement a correlation coefficient based on symmetric balances as introduced in [3].

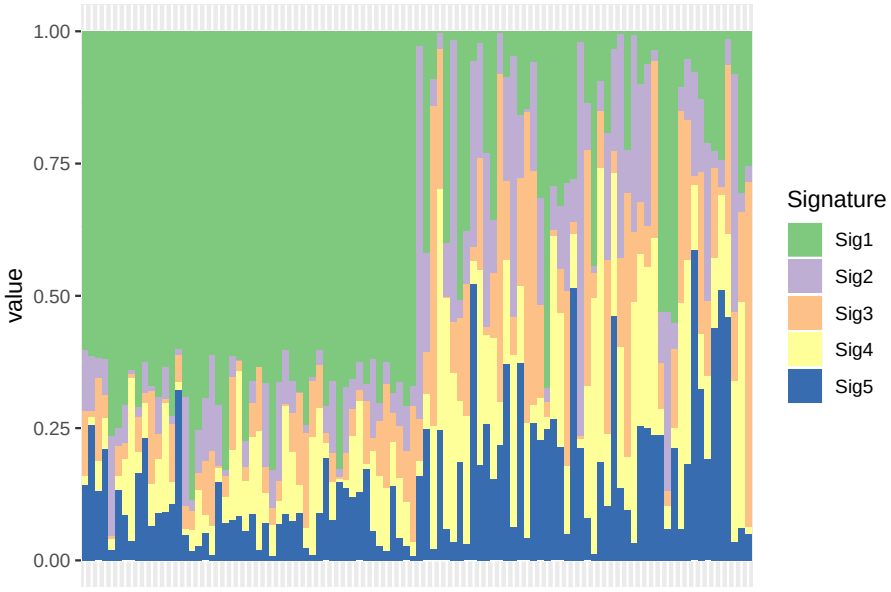
Code not shown because it's quite messy and under development, but check function `plotcomputeRho()`

```
## Pseudocount of 1e-07 added
## Pseudocount of 1e-07 added
## Pseudocount of 0 added
## Pseudocount of 0 added
## Pseudocount of 1e-07 added
## Pseudocount of 1e-07 added
```

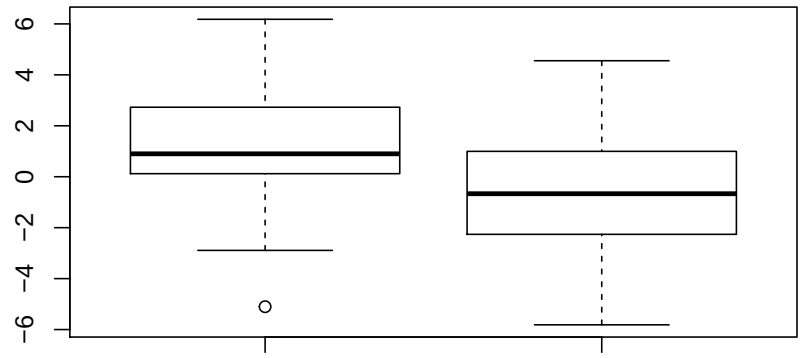

CNA TCGA QP dataset − rho
matrix_1
1
0.99
0.98
0.97
0.96
s2
s4
s1
s5
s6
s3
s7
s7
s3
s6
s5
s1
s4
s2
CNA TCGA QP dataset − cor clr
matrix_2
1
0.5
0
−0.5
−1
s2
s4
s5
s1
s7
s6
s3
s3
s6
s7
s1
s5
s4
s2
CNA TCGA QP dataset − rho
matrix_3
1
0.95
0.9
0.85
0.8
S4
S3
S1
S2
S5
S5
S2
S1
S3
S4
CNA TCGA QP dataset − cor clr
matrix_4
1
0.5
0
−0.5
S4
S1
S5
S3
S2
S2
S3
S5
S1
S4
reast cancers (Nik−Zainal) − rho
matrix_5
1
0.98
0.96
0.94
0.92
560 Breast cancers (Nik−Zainal) − cor clr
matrix_6
1
0.5
0
−0.5
−1

7 Optimal selection of partition

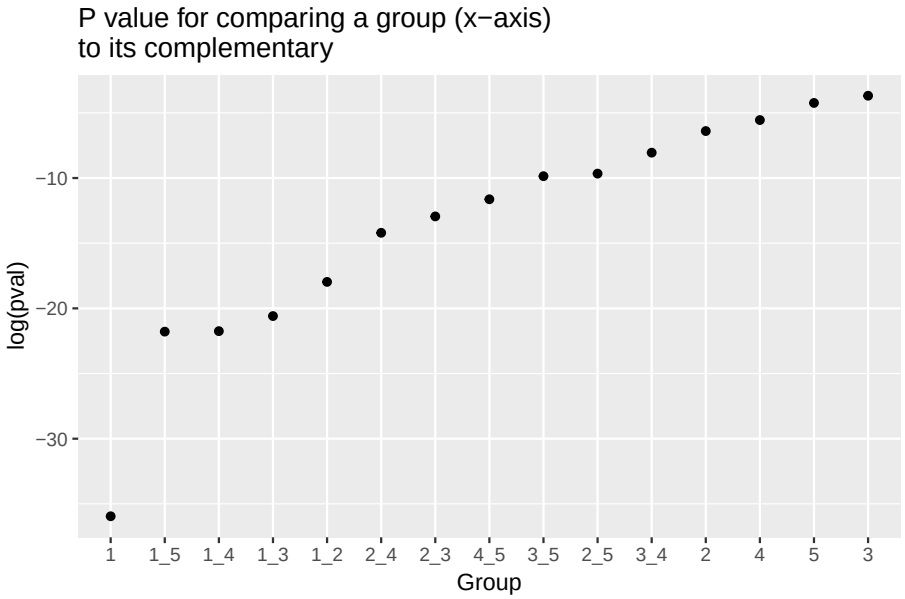
```
set.seed(1234)

## simulated data
Nsig <- 5; Nsamp <- 100
props <- MCMCpack::rdirichlet(Nsamp, c(rep(1,Nsig)))
colnames(props) <- paste0('Sig', 1:Nsig)
rownames(props) <- paste0('Sam ', 1:Nsamp)
## increase exposure to signature 1 in the first
## Nsamp/2, and re-normalise (two groups)
props[1:floor(Nsamp/2), 1] <- props[1:floor(Nsamp/2), 1] + 1.5
props <- sweep(props, 1, rowSums(props), '/')
createBarplot(props, remove_labels = TRUE)

## Creating plot... it might take some time if the data are large. Number of samples: 100
```




```
## corresponds to partitioning as follows:
##(s1, s2, s3) vs (s4, s5)
V <- c(1, 1, 1, -1, -1)
# plot(density(ilr(props, V = V)))
boxplot(ilr(props, V = V)[1:floor(Nsamp/2)],
        ilr(props, V = V)[(floor(Nsamp/2)+1):Nsamp], main = 'Comparison of ilr of (s1, s2, s
```




```
t.test(ilr(props, V = V)[1:floor(Nsamp/2)],
       ilr(props, V = V)[(floor(Nsamp/2)+1):Nsamp])$p.value

## [1] 8.913039e-06

it_partitions <- c()
if(Nsig > 8){warning('Large number of signatures')}
for(k in 1:floor(Nsig/2)){
  it_partitions <- c(it_partitions, lapply(1:ncol(combn(1:Nsig, k)), function(x) combn(1:Nsi
}
pvals <- rep(NA, length(it_partitions))
ict <- 1
for(i in it_partitions){
  V <- rep(-1, Nsig)
  V[i] <- 1
  pvals[ict] <- t.test(ilr(props, V = V)[1:floor(Nsamp/2)],
         ilr(props, V = V)[(floor(Nsamp/2)+1):Nsamp])$p.value
  ict <- ict + 1
}

groups <- sapply(it_partitions, paste0, collapse='_')
ggplot(data.frame(pval=pvals, group=groups),
       aes(x=factor(group, levels=groups[order(pvals)]), y=log(pval)))+
  geom_point() + ggtitle('P value for comparing a group (x-axis)\nto its complementary')+
  labs(x='Group')
```

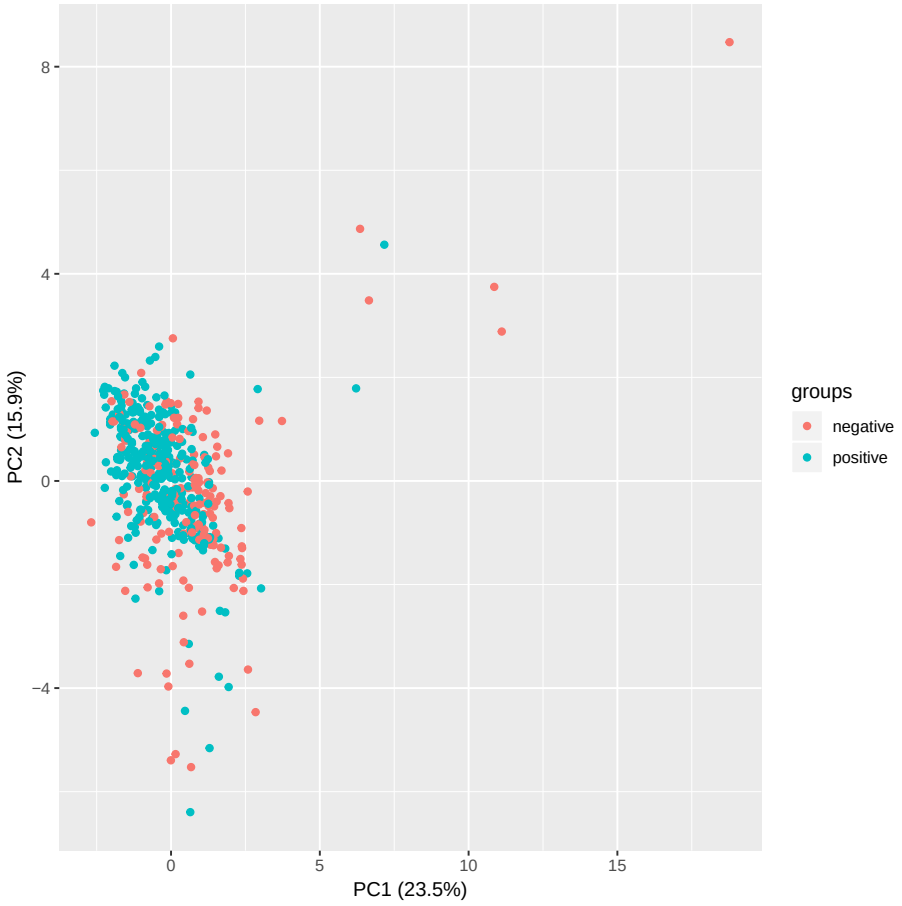



```
##' We expected 1 to be the one with lowest p-val
##' as this is how we created the dataset.
##' Groups containing signature 1 follow.
```

8 Dimensionality reduction: PCA

PCA is insensitive to the irl basis, so one can just use the default contrast matrix and run it with ilr, e.g.

```
plotPCA(compositions::ilr(exposuresBreast560),
        groups = metadataBreast560$final.ER)
```



References

[1] John Aitchison. "The statistical analysis of compositional data". In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1982), pp. 139–177.

[2] Andrej Fischer, Christopher JR Illingworth, Peter J Campbell, and Ville Mustonen. "EMu: probabilistic inference of mutational processes and their localization in the cancer genome". In: *Genome biology* 14.4 (2013), R39.

[3] Petra Kynčlová, Karel Hron, and Peter Filzmoser. "Correlation between compositional parts based on symmetric balances". In: *Mathematical Geosciences* 49.6 (2017), pp. 777–796.

[4] Geoff Macintyre, Teodora E Goranova, Dilrini De Silva, Darren Ennis, Anna M Piskorz, Matthew Eldridge, Daoud Sie, Liz-Anne Lewsley, Aishah Hanif, Cheryl Wilson, et al. "Copy number signatures and mutational processes in ovarian carcinoma". In: *Nature genetics* (2018), p. 1.

[5] Vera Pawlowsky-Glahn, Juan José Egozcue, and Raimon Tolosana-Delgado. *Modeling and analysis of compositional data*. John Wiley & Sons, 2015.

[6] Karl Pearson. "Mathematical contributions to the theory of evolution.—on a form of spurious correlation which may arise when indices are used in the measurement of organs". In: *Proceedings of the royal society of london* 60.359-367 (1897), pp. 489–498.

9 Session info

```
  sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] circlize_0.4.6        ComplexHeatmap_1.20.0 gridExtra_2.3
##  [4] RColorBrewer_1.1-2    reshape2_1.4.3        ggthemr_1.1.0
##  [7] ggplot2_3.1.1         MCMCpack_1.4-4        MASS_7.3-51.4
## [10] coda_0.19-2           Compositional_3.4     compositions_1.40-2
## [13] bayesm_3.1-1          energy_1.7-5          robustbase_0.93-4
## [16] tensorA_0.36.1        CompSign_0.1.0        usethis_1.4.0
## [19] devtools_2.0.2        knitr_1.22
##
## loaded via a namespace (and not attached):
##  [1] mcmc_0.9-6          fs_1.2.7            doParallel_1.0.14
##  [4] rprojroot_1.3-2     numDeriv_2016.8-1   tools_3.5.1
##  [7] backports_1.1.3     mixture_1.5         R6_2.4.0
## [10] lazyeval_0.2.2      colorspace_1.4-1    sn_1.5-3
## [13] GetoptLong_0.1.7    withr_2.1.2         tidyselect_0.2.5
## [16] prettyunits_1.0.2   mnormt_1.5-5        processx_3.3.0
## [19] compiler_3.5.1      cli_1.1.0           quantreg_5.38
## [22] SparseM_1.77        desc_1.2.0          labeling_0.3
## [25] scales_1.0.0        DEoptimR_1.0-8      callr_3.2.0
## [28] stringr_1.4.0       digest_0.6.18       pkgconfig_2.0.2
## [31] sessioninfo_1.1.1   highr_0.8           maps_3.3.0
## [34] rlang_0.3.4         GlobalOptions_0.1.0 rstudioapi_0.10
## [37] shape_1.4.4         dplyr_0.8.0.1       magrittr_1.5
## [40] dotCall64_1.0-0     Matrix_1.2-17       Rcpp_1.0.1
## [43] munsell_0.5.0       RcppZiggurat_0.1.5  stringi_1.4.3
## [46] pkgbuild_1.0.3      plyr_1.8.4          parallel_3.5.1
```

```
## [49] crayon_1.3.4        lattice_0.20-38     emplik_1.0-4.3
## [52] ps_1.3.0            pillar_1.3.1        boot_1.3-20
## [55] rjson_0.2.20        codetools_0.2-16    stats4_3.5.1
## [58] pkgload_1.0.2       glue_1.3.1          evaluate_0.13
## [61] remotes_2.0.2       spam_2.2-2          foreach_1.4.4
## [64] testthat_2.0.1      MatrixModels_0.4-1  gtable_0.3.0
## [67] purrr_0.3.2         assertthat_0.2.1    xfun_0.6
## [70] Rfast_1.9.3         tibble_2.1.1        iterators_1.0.10
## [73] memoise_1.1.0       fields_9.7
```