Package **CompSign**

Lena Morrill

October 2017

Important note: `V=V` needs to be changed to `V=gsi.buildilrBase(V)` everywhere.

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]cite myself

1 Installation

`CompSign` can be installed as usual from github:

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

```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[**fischer2013emu**] (cite dataset)
- Copy Number signatures
  - Ovarian cancer-derived signatures, as described in [**macintyre2018copy**]. 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])))

## Error in createBarplot(count_matrix(Breast560), remove_labels = TRUE,
:  could not find function "createBarplot"
```

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 0.5688626 0.3500838 0.8218299
## Sample 2 0.8763556 0.5482661 0.4639388
## Sample 3 0.9012522 0.6999470 0.3318919
## Sample 4 0.9117360 0.3541932 0.9619492
##
## Slot "modified":
## [1] FALSE
```

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

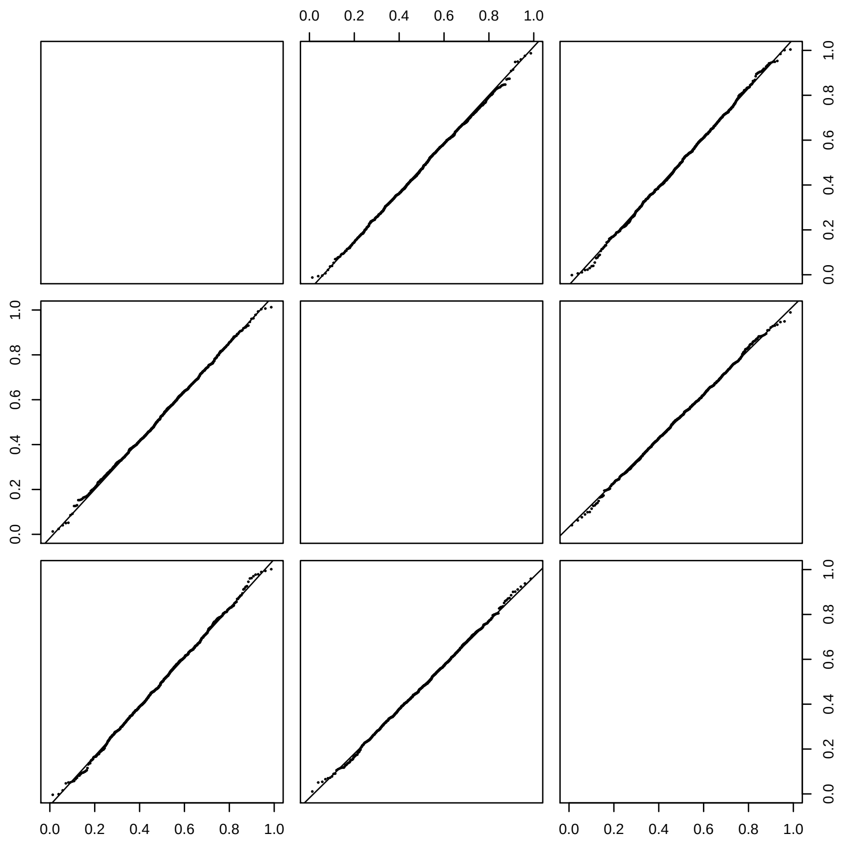
```
results_sumarise <- summarise(add_together_matrix(basic_sign))
results_sumarise

## $General
## [1] "Object of class sign"
##
## $`Number of signatures`
## [1] 3
##
## $`Number of samples`
## [1] 4
##
## $`Geometric means of signatures`
##        s1        s3        s2
## 0.8000199 0.5906740 0.4670541
##
## $Covariance
##             s1          s2          s3
## s1  0.02704821  0.01432705 -0.01699320
## s2  0.01432705  0.02849286 -0.04802392
## s3 -0.01699320 -0.04802392  0.08751513
```

4 Battery of tests

This section takes largely from Aitchison's pioneering work[**aitchison1982statistical**] and its succession[**pawlowsky2015modeling**].

4.1 Test for normality

```r
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

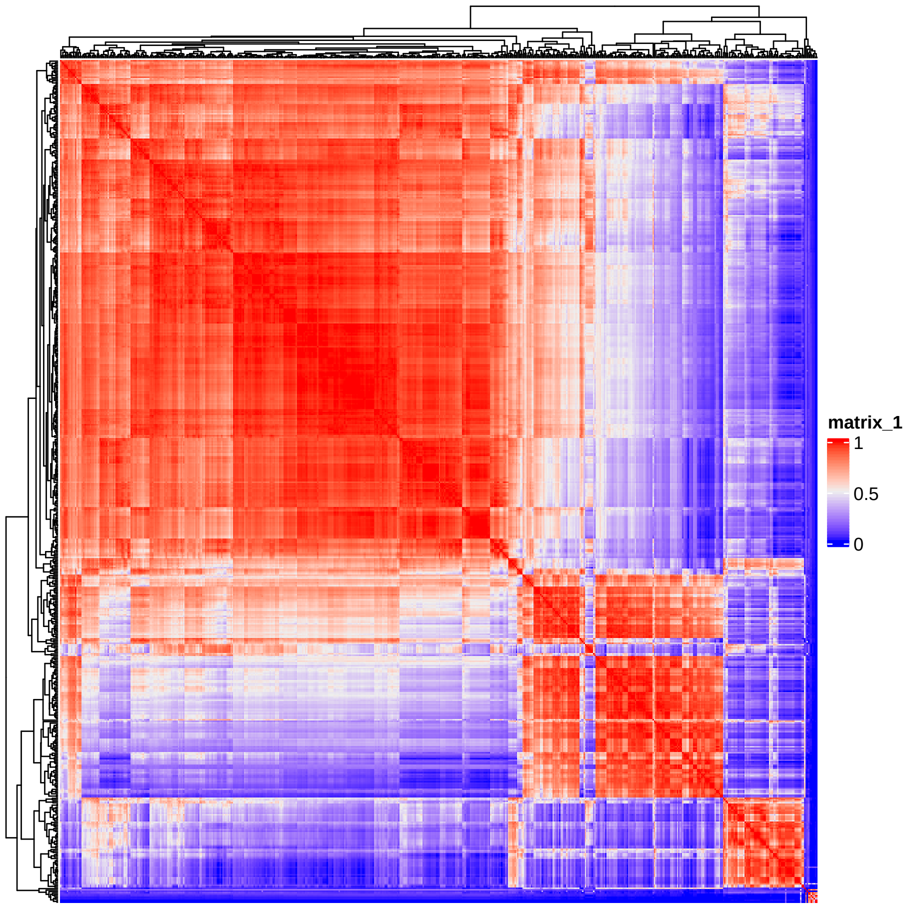
Test for equality of means.

```
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 corrected.critical
##       2.359067e+02       5.324172e-43       1.042820e+00       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, exposuresBreast56

## [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)
```

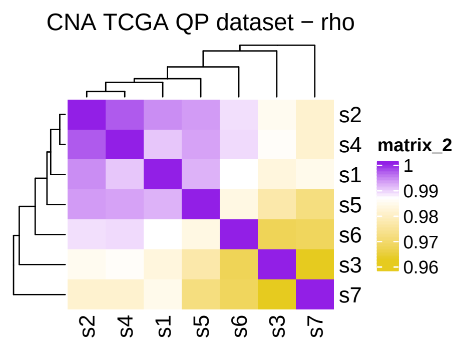
6 Symmetric balances for correlation between signatures

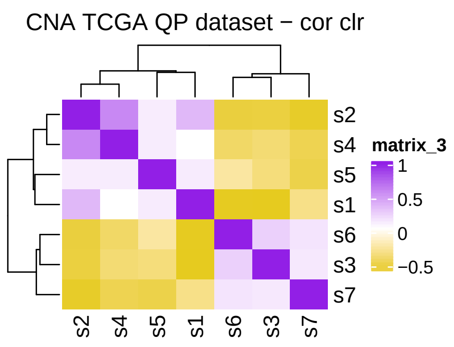
Pearson pointed out ([**pearson1897mathematical**]) that spurious correlations appear when closed, compositional, data, are analysed. Therefore normal correlation coefficients can be ruled out of our analysis.

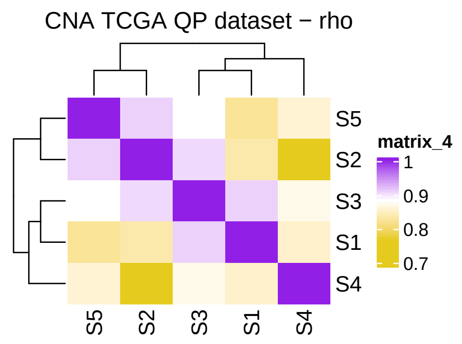
I implement a correlation coefficient based on symmetric balances as introduced in [**kynvclova2017correlation**].

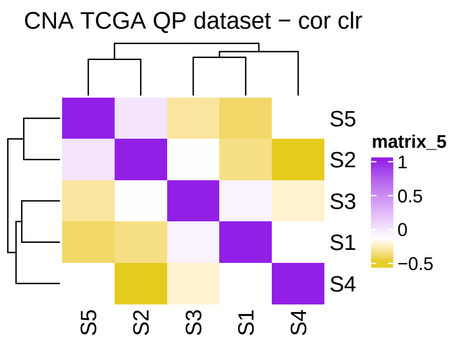
Code not shown because it's quite messy and under development, but check function `plotcomputeRho()`

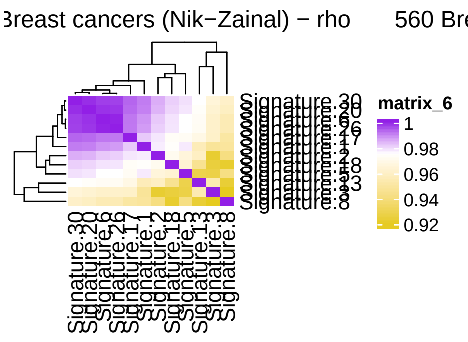
```
## Pseudocount of 1e-07 added
## Pseudocount of 1e-07 added
## Pseudocount of 0 added
## Pseudocount of 0 added
## Pseudocount of 1e-07 added
## Pseudocount of 1e-07 added
```

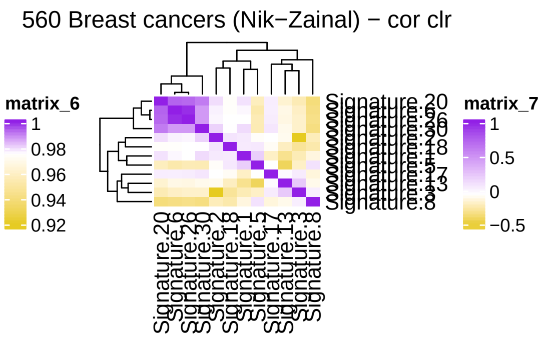












ADD THE RAW VERSION

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)

## Error in createBarplot(props, remove_labels = TRUE): could not find
function "createBarplot"

## 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

## Error in unclass(x)[i, , drop = drop]:  (subscript) logical subscript
too long

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

## Error in gsi.orig(x, y):  argument "y" is missing, with no default

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
}
```

```
## Error in gsi.orig(x, y):  argument "y" is missing, with no default
```

```
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')
```

```
## Warning:  Removed 15 rows containing missing values (geom_point).
```

```
##' 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)

## Error in plotPCA(compositions::ilr(exposuresBreast560), groups =
metadataBreast560$final.ER): could not find function "plotPCA"
```

9 Plotting

9.1 Barplots

As seen before

9.2 Ternary diagrams

```
plot_ggtern(close_data(exposuresCNA_12K_TCGA, c('s1', 's2', 's6')),
            colours = factor(metadataCNA_12K_TCGA$project_id))

## Loading required package:  ggtern
## Warning in library(package, lib.loc = lib.loc, character.only =
TRUE, logical.return = TRUE, :  there is no package called 'ggtern'
## Error in ggtern(data = tmp_df, aes(x = s1, y = s2, z = s3, grp =
as.numeric(grp)), :  could not find function "ggtern"
```

10 Session info

```r
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] Compositional_5.8    reshape2_1.4.4       ggthemr_1.1.0
##  [4] circlize_0.4.15      ComplexHeatmap_2.6.2 gridExtra_2.3
##  [7] ggplot2_3.3.6        MCMCpack_1.6-2       MASS_7.3-56
## [10] coda_0.19-4          compositions_2.0-4   CompSign_0.1.0
## [13] knitr_1.39
##
## loaded via a namespace (and not attached):
##   [1] colorspace_2.0-3    rjson_0.2.21        ellipsis_0.3.2
##   [4] class_7.3-20        emplik_1.1-1        GlobalOptions_0.1.2
##   [7] proxy_0.4-26        clue_0.3-60         rstudioapi_0.13
##  [10] listenv_0.8.0       MatrixModels_0.5-0  gsl_2.1-7.1
##  [13] fansi_1.0.3         splines_4.0.3       codetools_0.2-18
##  [16] extrafont_0.18      mnormt_2.0.2        doParallel_1.0.17
##  [19] robustbase_0.95-0   jsonlite_1.8.0      mda_0.5-2
##  [22] mixture_2.0.4       Cairo_1.5-15        mcmc_0.9-7
##  [25] Rttf2pt1_1.3.10     cluster_2.1.3       bigstatsr_1.5.6
##  [28] png_0.1-7           compiler_4.0.3      bigparallelr_0.3.2
##  [31] assertthat_0.2.1    RcppZiggurat_0.1.6  Matrix_1.4-1
##  [34] fastmap_1.1.0       cli_3.2.0           FlexDir_1.0
##  [37] htmltools_0.5.2     quantreg_5.88       tools_4.0.3
##  [40] gtable_0.3.0        glue_1.6.2          RANN_2.6.1
##  [43] dplyr_1.0.8         Rcpp_1.0.8.3        vctrs_0.4.0
##  [46] extrafontdb_1.0     iterators_1.0.14    tensorA_0.36.2
##  [49] xfun_0.30           stringr_1.4.0       pchc_0.8
##  [52] globals_0.15.1      lifecycle_1.0.1     future_1.26.1
##  [55] codalm_0.1.2        NlcOptim_0.6        DEoptimR_1.0-11
##  [58] scales_1.2.0        flock_0.7           parallel_4.0.3
##  [61] SparseM_1.81        Rfast2_0.1.3        RColorBrewer_1.1-3
##  [64] stringi_1.7.6       SQUAREM_2021.1      highr_0.9
##  [67] S4Vectors_0.28.1    foreach_1.5.2       energy_1.7-9
##  [70] e1071_1.7-9         bigassertr_0.1.5    BiocGenerics_0.36.1
##  [73] boot_1.3-28         shape_1.4.6         rlang_1.0.2
##  [76] pkgconfig_2.0.3     matrixStats_0.61.0  rgl_0.108.3
##  [79] evaluate_0.15       lattice_0.20-45     sf_1.0-7
##  [82] purrr_0.3.4         htmlwidgets_1.5.4   cowplot_1.1.1
##  [85] Rfast_2.0.6         tidyselect_1.1.2    parallelly_1.32.0
##  [88] plyr_1.8.7          magrittr_2.0.3      R6_2.5.1
##  [91] magick_2.7.3        IRanges_2.24.1      generics_0.1.3
##  [94] DBI_1.1.3           pillar_1.8.0        withr_2.5.0
##  [97] sn_2.0.2            units_0.8-0         survival_3.3-1
## [100] sp_1.4-6            nnet_7.3-17         tibble_3.1.6
## [103] future.apply_1.9.0  bayesm_3.1-4        dcov_0.1.1
## [106] bnlearn_4.7.1       crayon_1.5.1        KernSmooth_2.23-20
## [109] utf8_1.2.2          tmvnsim_1.0-2       GetoptLong_1.0.5
## [112] rnaturalearth_0.1.0 classInt_0.4-3      digest_0.6.29
## [115] Directional_5.5     numDeriv_2016.8-1.1 glmnet_4.1-3
## [118] stats4_4.0.3        munsell_0.5.0       quadprog_1.5-8
```