Package **CompSign**

Lena Morrill

October 2017

CompSign is a package for yadayada... overlooked that mutational signatures are compositional in nature yadayada. The reference manual can be found here.

```
knitr::opts_chunk$set(cache = FALSE)
```

```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:09:57 2018 ------##

## install latest version
library(devtools)
devtools::install_github("lm687/CompSign")

## Skipping install of 'CompSign' from a github remote, the SHA1 (d0c7784d)
has not changed since last install.
##  Use `force = TRUE` to force installation

library(CompSign)
library(compositions)

## Loading required package:  tensorA
##
## Attaching package:  'tensorA'
## The following object is masked from 'package:base':
##
##    norm
## Loading required package:  robustbase
## Loading required package:  energy
## Loading required package:  bayesm
## Welcome to compositions, a package for compositional data analysis.
## Find an intro with "?  compositions"
##
## Attaching package:  'compositions'
## The following objects are masked from 'package:stats':
##
##     cor, cov, dist, var
## The following objects are masked from 'package:base':
##
##     %*%, scale, scale.default
```

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

```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:10:00 2018 ------##

##########################
####### Dummy data #######
##########################

### Example of matrix transformed into sign object
input_dummy <- matrix(runif(100), 4)
colnames(input_dummy) <- paste0('s', 1:25); rownames(input_dummy) <- paste0('sam', 1:4)
sign_dummy <- to_sign(input_dummy)
```

1 Summarise the signature matrix

```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:10:00 2018 ------##

add_together_matrix(sign_dummy)

## An object of class "sign"
## Slot "id":
## [1] "input_dummy"
##
## Slot "id_samples":
## [1] "sam1" "sam2" "sam3" "sam4"
##
## Slot "id_signatures":
##  [1] "s1"  "s2"  "s3"  "s4"  "s5"  "s6"  "s7"  "s8"  "s9"  "s10" "s11"
## [12] "s12" "s13" "s14" "s15" "s16" "s17" "s18" "s19" "s20" "s21" "s22"
## [23] "s23" "s24" "s25"
##
## Slot "count_matrix":
##             s1         s2        s3        s4         s5        s6
## sam1 0.7784261 0.01899545 0.2832337 0.7634464 0.47690738 0.4932939
## sam2 0.8647305 0.54011930 0.7992423 0.8853640 0.82286212 0.9339099
## sam3 0.6502008 0.44315965 0.3580316 0.9311900 0.43560483 0.9687104
## sam4 0.1163561 0.54584555 0.1325613 0.1515711 0.04704713 0.5303048
##             s7         s8        s9       s10        s11       s12
## sam1 0.4447834 0.84069656 0.5091814 0.2069675 0.08193922 0.9733937
## sam2 0.7258888 0.00198996 0.1395706 0.9629527 0.03750356 0.6127755
## sam3 0.9675364 0.08431608 0.5187020 0.6895406 0.03911379 0.3846920
## sam4 0.8527858 0.23624631 0.5573887 0.1506753 0.71586751 0.9165366
##             s13        s14       s15        s16       s17        s18
## sam1 0.53997656 0.18754798 0.7340085 0.73489632 0.3042449 0.07395421
## sam2 0.67711458 0.20379231 0.7464848 0.55030052 0.2577085 0.35628649
## sam3 0.02900615 0.22309559 0.3796148 0.05790401 0.7853138 0.10672071
## sam4 0.85884574 0.03870028 0.6201593 0.24404868 0.6390459 0.71745306
##            s19       s20       s21       s22       s23        s24
## sam1 0.8204762 0.6920064 0.5414644 0.6441391 0.2410165 0.68262420
## sam2 0.1553688 0.3970055 0.1269795 0.1329946 0.5596698 0.61172885
## sam3 0.1293228 0.3754686 0.5026743 0.5166404 0.4180655 0.39536817
## sam4 0.5733194 0.9927708 0.9557972 0.3777508 0.2864715 0.03293686
##            s25
## sam1 0.3438330
## sam2 0.4738645
## sam3 0.8433543
## sam4 0.2386754
##
## Slot "modified":
## [1] TRUE
```

```
results_sumarise <- summarise(add_together_matrix(sign_dummy))
results_sumarise$General
```

```
## [1] "Object of class sign"
```

2 Linear model for numerical predictors

```
## This chunk was last ran in
timestamp()
```

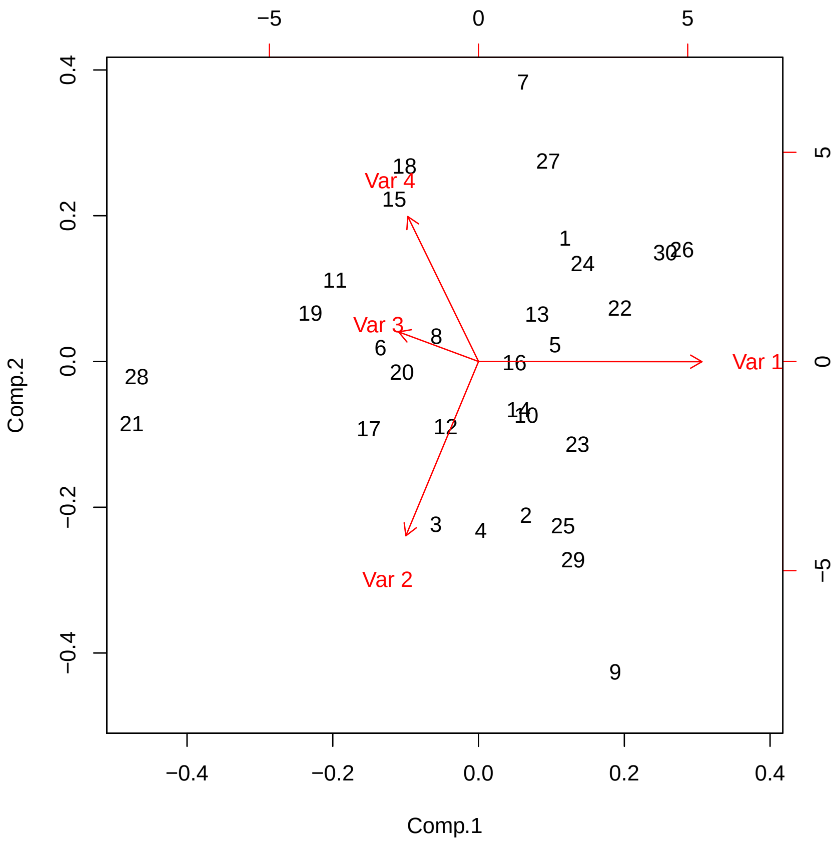
```
## ##------ Wed Nov  7 10:10:00 2018 ------##
```

```
tmp_merged_compositional <- new("merged_compositional",
                                id='adas',
                                id_samples=paste0("sam", 1:30),
                                id_signatures= c('s1', 's2', 's3', 's4'), ## signature name
                                count_matrix=MCMCpack::rdirichlet(30, c(1,1,1,1)),
                                df=data.frame(a=sample(1:1e4, 30), b=rep(10, 30)))
comp_lm(tmp_merged_compositional)
```

```
## [[1]]
## Response Y1 :
##
## Call:
## lm(formula = Y1 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0235 -0.3696 -0.1226  0.2959  1.9088
##
## Coefficients: (1 not defined because of singularities)
##                                           Estimate Std. Error t value
## (Intercept)                             -1.069e-01  3.268e-01  -0.327
## as.matrix((x@df)[, indices_predictor])a  3.724e-06  5.745e-05   0.065
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                                0.746
## as.matrix((x@df)[, indices_predictor])a    0.949
## as.matrix((x@df)[, indices_predictor])b       NA
##
## Residual standard error: 0.8785 on 28 degrees of freedom
## Multiple R-squared:  0.00015,Adjusted R-squared:  -0.03556
## F-statistic: 0.0042 on 1 and 28 DF,  p-value: 0.9488
##
##
## Response Y2 :
##
## Call:
## lm(formula = Y2 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8314 -0.8996 -0.1323  0.9301  1.8544
##
## Coefficients: (1 not defined because of singularities)
##                                           Estimate Std. Error t value
## (Intercept)                              5.205e-01  3.918e-01   1.328
## as.matrix((x@df)[, indices_predictor])a -2.985e-05  6.888e-05  -0.433
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                                0.195
## as.matrix((x@df)[, indices_predictor])a    0.668
## as.matrix((x@df)[, indices_predictor])b       NA
##
## Residual standard error: 1.053 on 28 degrees of freedom
## Multiple R-squared:  0.006661,Adjusted R-squared:  -0.02881
## F-statistic: 0.1878 on 1 and 28 DF,  p-value: 0.6681
##
##
## Response Y3 :
##
## Call:
## lm(formula = Y3 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4651 -0.4461  0.1964  0.5655  2.0742
##
## Coefficients: (1 not defined because of singularities)
##                                           Estimate Std. Error t value
## (Intercept)                              3.875e-01  4.612e-01   0.840
## as.matrix((x@df)[, indices_predictor])a -8.795e-05  8.109e-05  -1.085
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                                0.408
## as.matrix((x@df)[, indices_predictor])a    0.287
## as.matrix((x@df)[, indices_predictor])b       NA
##
## Residual standard error: 1.24 on 28 degrees of freedom
## Multiple R-squared:  0.04032,Adjusted R-squared:  0.006049
## F-statistic: 1.176 on 1 and 28 DF,  p-value: 0.2873
```

3 Importing data

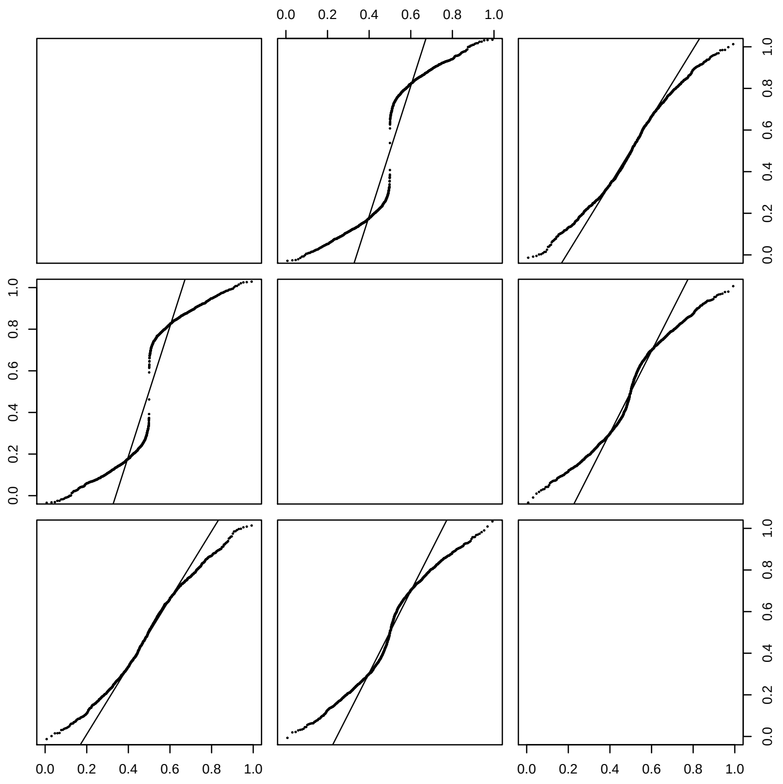
```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:10:02 2018 ------##

biplot(princomp(acomp(MCMCpack::rdirichlet(30, rep(1, 4)))))
```



4 Other

1. Test for normality as follows:

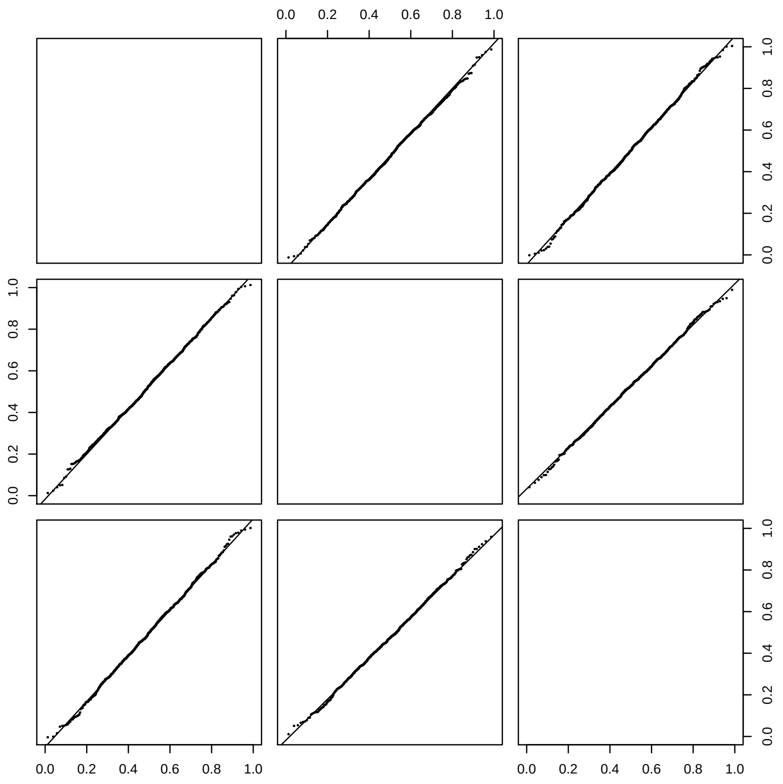
```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:10:02 2018 ------##

data(two_normal_pops)

## Warning in data(two_normal_pops):  data set 'two_normal_pops'
not found

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, plot.it=FAL

## Warning in plot.window(...):  "plot.it" is not a graphical
parameter

## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter

## Warning in title(...):  "plot.it" is not a graphical parameter

## Warning in plot.window(...):  "plot.it" is not a graphical
parameter

## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter

## Warning in title(...):  "plot.it" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...):
"plot.it" is not a graphical parameter

## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "plot.it"
is not a graphical parameter
```

```
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...):
"plot.it" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "plot.it"
is not a graphical parameter
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...):
"plot.it" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "plot.it"
is not a graphical parameter
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "plot.it"
is not a graphical parameter
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...):
"plot.it" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "plot.it"
is not a graphical parameter
```

```
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "plot.it"
is not a graphical parameter
## Warning in plot.window(...):  "plot.it" is not a graphical
parameter
## Warning in plot.xy(xy, type, ...):  "plot.it" is not a graphical
parameter
## Warning in title(...):  "plot.it" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...):
"plot.it" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...):
"plot.it" is not a graphical parameter
```



5 Clustering of samples

5.1 Testing hypotheses about two populations

We might have our samples split into two categories; e.g. sex. As in Aithison 1986[], I follow a hierarchy of alternative hypotheses, from least to most complex.

Our first question is whether two populations have the same covariance and structure and center (i.e. if there is any distributional difference)

```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:10:02 2018 ------##

##TODO!!
```

The next is whether the populations have a different center:

```
## This chunk was last ran in
timestamp()

## ##------ Wed Nov  7 10:10:02 2018 ------##

## This dataset includes the two components above, as well as four others
## (a total of seven)
data("two_normal_pops_extended")

## Warning in data("two_normal_pops_extended"):  data set 'two_normal_pops_extended'
not found

## Data from the Landscape... paper
data("Breast560")

## Warning in data("Breast560"):  data set 'Breast560' not found

wrapper_compare_populations <- function(predictors, response, ...){
  if(length(unique(response)) == 2){
    tmp <- compare_populations(predictors, response, ...)
    tmp <- tmp$info[1:2]
    tmp
  }
}

x <- do.call('rbind', lapply(1:ncol(metadata(Breast560)),
       function(k){
         wrapper_compare_populations(predictors = count_matrix(Breast560),
                                     response = metadata(Breast560)[,k])
         }
       ))
```

```
## Loading required package:  Compositional
##
## Attaching package:  'Compositional'
## The following object is masked from 'package:compositions':
##
##     alr
```

```
x
```

```
##          test      p-value
## [1,] 223.6681 6.334800e-26
## [2,] 237.6260 6.270514e-29
## [3,] 237.4362 2.457445e-43
## [4,]  78.3811 9.584122e-12
```

6 Data for 560 breast cancer patients

Data from 560 breast cancer patients is available as part of the document as well:

```
data("Breast560")
```

```
## Warning in data("Breast560"):  data set 'Breast560' not found
```

```
metadata(Breast560)[1:4,1:5]
```

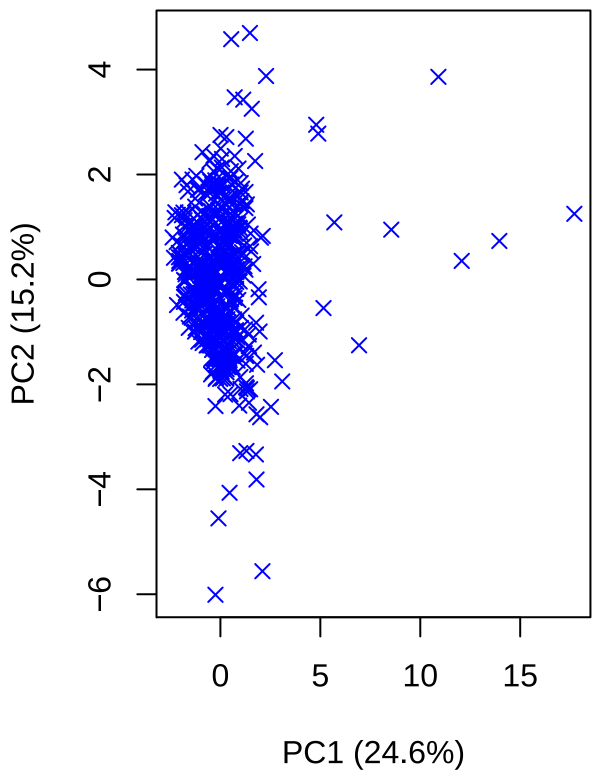
```
##         donor_gender donor_age_at_diagnosis donor_age_at_last_follow.up
## PD10010       female                     56            no_data_supplied
## PD10011       female                     75            no_data_supplied
## PD10014       female                     64            no_data_supplied
## PD11326       female                     38                          47
##          specimen_type donor_vital_status
## PD10010 tumour_primary              alive
## PD10011 tumour_primary              alive
## PD10014 tumour_primary           deceased
## PD11326 tumour_primary              alive
```

```
count_matrix(Breast560)[1:4,1:5]
```

```
##          Signature.1  Signature.2  Signature.3 Signature.5 Signature.6
## PD10010 0.0013656127 0.0002299146 0.0009201629 0.001224842           0
## PD10011 0.0020433984 0.0000000000 0.0025729837 0.005577513           0
## PD10014 0.0010016166 0.0000000000 0.0060192540 0.005093551           0
## PD11326 0.0009765135 0.0005635162 0.0059827947 0.002310769           0
```

Not sure if this is correct

```
source("../../CDA_in_Cancer/code/functions/basic_functions.R")
plotPCA(ilr(count_matrix(Breast560)), pch=4, col='blue')

## Loading required package:  ggplot2
```



6.1 (ongoing) test for equality

```
comp.test(x = count_matrix(Breast560),
                                ina = as.numeric(as.factor(metadata(Breast560)$final.ER)),
                                test = "james", R = 0)

## $note
## [1] "James test"
##
```

```
## $mesoi
##                X1        X2         X3       X4         X5        X6
## Sample 1 0.583829 -1.336900 -0.1776487 1.467427 -0.5977703 0.3018865
## Sample 2 1.410313  1.626698 -1.0779649 1.773800 -0.0129786 0.7226163
##                 X7         X8       X9       X10       X11
## Sample 1 0.9095453 0.59145173 1.170214 0.9935606 0.9807331
## Sample 2 1.2939339 0.04855843 1.241930 1.0600895 1.0444659
##
## $info
##               test            p-value         correction
##       2.797486e+02       6.279768e-51       1.046173e+00
## corrected.critical
##       2.058360e+01
```

7 Data for 12k TCGA samples, with ovarian cancer-derived CNA signatures

```
timestamp()

## ##------ Wed Nov  7 10:10:04 2018 ------##

data("CNA_12K_TCGA")

## Warning in data("CNA_12K_TCGA"):  data set 'CNA_12K_TCGA' not found

dim(metadata(CNA_12K_TCGA))

## [1] 10899    37

dim(count_matrix(CNA_12K_TCGA))

## [1] 10899     7
```