

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()

## ##----- Tue Nov  6 16:18:24 2018 -----##

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

## Skipping install of 'CompSign' from a github remote, the SHA1 (2f8715ef)
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
```

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

## ##----- Tue Nov  6 16:18:25 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()

## ##----- Tue Nov  6 16:18:25 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 0.07272969 0.9140327 0.4968223 0.5184914 0.4145388 0.1969778
## sam2 0.50024880 0.8342635 0.5119216 0.7972444 0.1921037 0.1753118
## sam3 0.38248952 0.3448541 0.5881578 0.1442695 0.3637460 0.1938514
## sam4 0.41510238 0.2023308 0.5460272 0.7285554 0.7078948 0.6262570
##           s7           s8           s9           s10          s11          s12
## sam1 0.3525339 0.4488783 0.4694396 0.6789127 0.2526726 0.7649207
## sam2 0.4708593 0.9900759 0.6184483 0.1181695 0.5974123 0.5006406
## sam3 0.4106082 0.8226040 0.8818504 0.3184051 0.6330861 0.2058611
## sam4 0.6488849 0.4297723 0.2651439 0.3967617 0.5279068 0.4998687
##           s13          s14          s15          s16          s17          s18
## sam1 0.79002042 0.3092334 0.96584926 0.41697016 0.3454698 0.53132524
## sam2 0.09006762 0.9700768 0.04151968 0.91852417 0.4029363 0.68407133
## sam3 0.77174884 0.9976588 0.59072908 0.52178241 0.6780246 0.05903725
## sam4 0.56109056 0.5114064 0.05022707 0.08896006 0.1888010 0.06147117
##           s19          s20          s21          s22          s23          s24
## sam1 0.2332226 0.5597038 0.73399792 0.3430649 0.6219140 0.8483383
## sam2 0.4414435 0.7630962 0.05136377 0.5325759 0.7028786 0.8466368
## sam3 0.6789870 0.8356054 0.96469319 0.7010786 0.8402231 0.5877767
## sam4 0.8066790 0.2289996 0.10104237 0.1947610 0.9714809 0.1344074
##           s25
## sam1 0.8395067
## sam2 0.4334580
## sam3 0.9234258
## sam4 0.6953657
##
## 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()

## ##----- Tue Nov  6 16:18:25 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
## -2.0267 -0.6400  0.1642  0.6065  2.3522
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value
## (Intercept)      5.607e-01   3.580e-01   1.566
## as.matrix((x@df)[, indices_predictor])a -9.669e-05   6.908e-05  -1.400
## as.matrix((x@df)[, indices_predictor])b          NA          NA          NA
##              Pr(>|t|)
## (Intercept)          0.129
## as.matrix((x@df)[, indices_predictor])a    0.173
## as.matrix((x@df)[, indices_predictor])b          NA
##
## Residual standard error: 1.067 on 28 degrees of freedom
## Multiple R-squared:  0.0654, Adjusted R-squared:  0.03202
## F-statistic: 1.959 on 1 and 28 DF,  p-value: 0.1726
##
##
## Response Y2 :
##
## Call:
## lm(formula = Y2 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.6978 -0.4500  0.0375  0.7130  2.2381
##
## Coefficients: (1 not defined because of singularities)
##              Estimate Std. Error t value
## (Intercept)     -3.687e-02   3.983e-01  -0.093
## as.matrix((x@df)[, indices_predictor])a -3.878e-05   7.686e-05  -0.505
## as.matrix((x@df)[, indices_predictor])b          NA          NA          NA
##              Pr(>|t|)
## (Intercept)          0.927

```

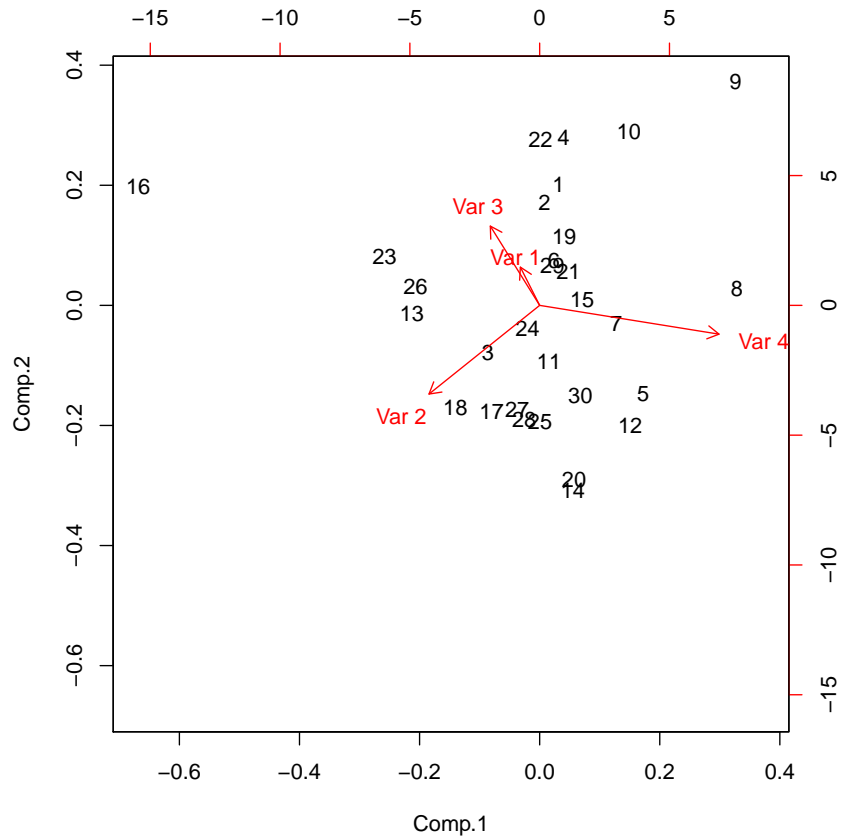
```
## as.matrix((x@df)[, indices_predictor])a    0.618
## as.matrix((x@df)[, indices_predictor])b      NA
##
## Residual standard error: 1.187 on 28 degrees of freedom
## Multiple R-squared:  0.00901, Adjusted R-squared:  -0.02638
## F-statistic: 0.2546 on 1 and 28 DF,  p-value: 0.6178
##
##
## Response Y3 :
##
## Call:
## lm(formula = Y3 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.9387 -0.5832 -0.1760  0.3872  1.8835
##
## Coefficients: (1 not defined because of singularities)
##                                Estimate Std. Error t value
## (Intercept)                   7.847e-01  2.663e-01   2.946
## as.matrix((x@df)[, indices_predictor])a -1.137e-04  5.139e-05 -2.213
## as.matrix((x@df)[, indices_predictor])b      NA      NA      NA
##                                Pr(>|t|)
## (Intercept)                   0.00641 **
## as.matrix((x@df)[, indices_predictor])a   0.03523 *
## as.matrix((x@df)[, indices_predictor])b      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7935 on 28 degrees of freedom
## Multiple R-squared:  0.1488, Adjusted R-squared:  0.1184
## F-statistic: 4.897 on 1 and 28 DF,  p-value: 0.03523
```

3 Importing data

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

## ##----- Tue Nov  6 16:18:26 2018 -----##

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



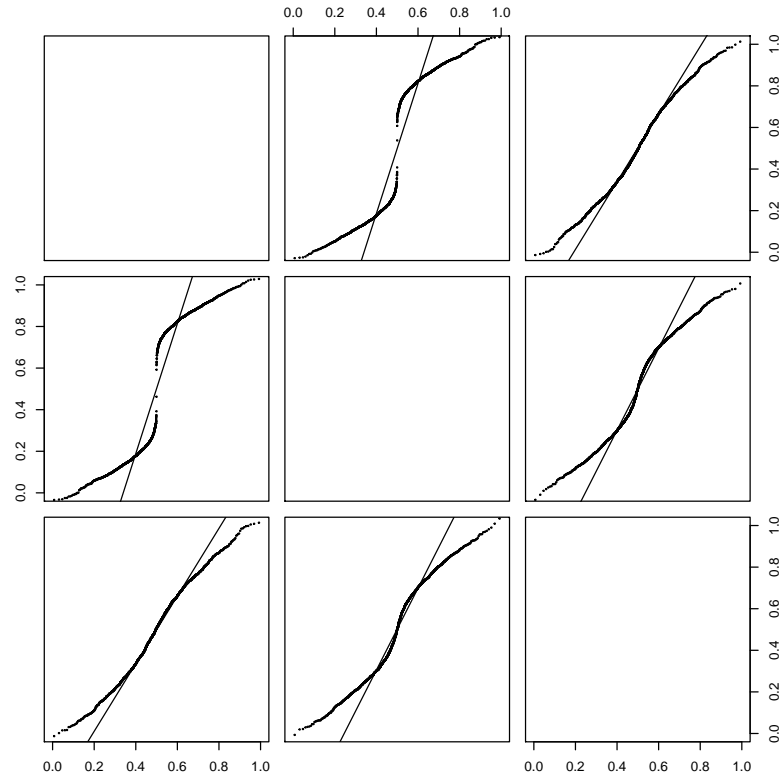
4 Other

1. Test for normality as follows:

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

## ##----- Tue Nov  6 16:18:26 2018 -----##

data(two_normal_pops)
par(mfrow=c(1,2))
qqnorm.acomp(acomp(two_normal_pops@count_matrix), pch=19, cex=0.2)
```



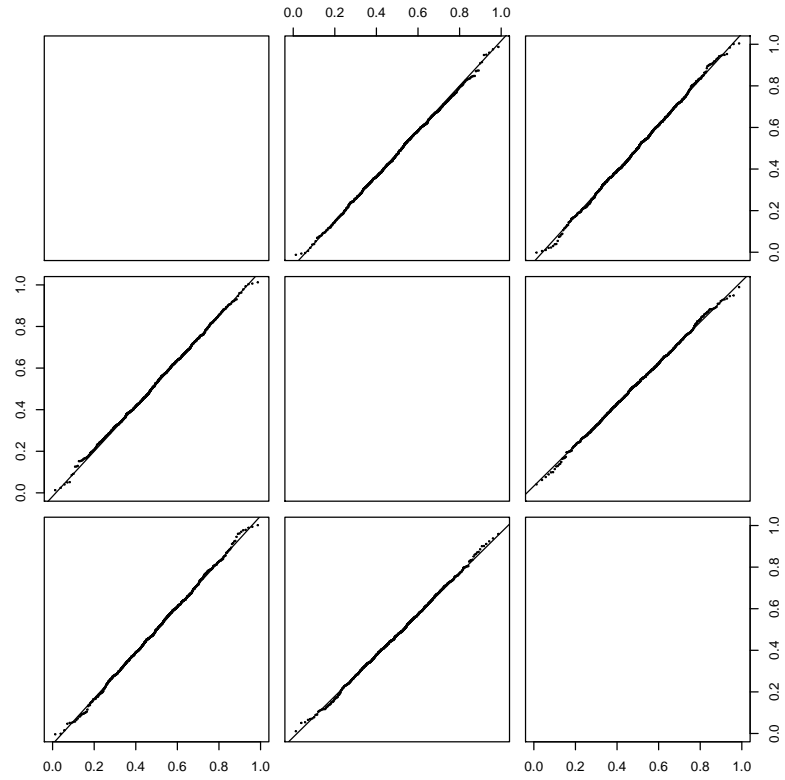
```
qqnorm.acomp(acomp(two_normal_pops@count_matrix[1:1000,]), pch=19, cex=0.2, plot.it=FALSE)

## 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()

## ##----- Tue Nov  6 16:18:27 2018 -----##

##TODO!!
```

The next is whether the populations have a different center:

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

## ##----- Tue Nov  6 16:18:27 2018 -----##

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

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

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")
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("~/Documents/CantabPhD/CDA_in_Cancer/code/functions/various_functions.R")

## Warning in file(filename, "r", encoding = encoding): cannot open
file '/Users/morril01/Documents/CantabPhD/CDA_in_Cancer/code/functions/various_functions.R':
No such file or directory
## Error in file(filename, "r", encoding = encoding): cannot open
the connection

plotPCA(ilr(count_matrix(Breast560)), pch=4, col='blue')
```

```
## Error in plotPCA(irlr(count_matrix(Breast560)), pch = 4, col = "blue"):
could not find function "plotPCA"
```

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()

## ##----- Tue Nov 6 16:18:28 2018 -----##

data("CNA_12K_TCGA")

dim(metadata(CNA_12K_TCGA))

## [1] 10899 37

dim(count_matrix(CNA_12K_TCGA))

## [1] 10899 7
```