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 = TRUE)
```

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

## ##------ Tue Oct 23 16:07:05 2018 ------##

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

## Downloading GitHub repo lm687/CompSign@master

## Rfast      (1.9.0  -> 1.9.1 ) [CRAN]
## rlang      (0.2.2  -> 0.3.0 ) [CRAN]
## robustbase (0.93-1 -> 0.93-3) [CRAN]

## Installing 3 packages:  Rfast, rlang, robustbase

##
##   There are binary versions available but the source versions are
##   later:
##             binary source needs_compilation
## Rfast         1.9.0  1.9.1              TRUE
## rlang         0.2.2  0.3.0              TRUE
## robustbase 0.93-1.1 0.93-3              TRUE

## installing the source packages 'Rfast', 'rlang', 'robustbase'
## Error in i.p(...):  (converted from warning) installation of package
'Rfast' had non-zero exit status

library(CompSign)
library(compositions)
```

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

## ##------ Tue Oct 23 16:07:17 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 Oct 23 16:07:17 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 0.6233436 0.4169807 0.42993493 0.05794318 0.5901997 0.4031462
## sam2 0.3413632 0.9537744 0.40388599 0.77977055 0.3459038 0.6192916
## sam3 0.7667998 0.5407284 0.41226984 0.52461156 0.8846929 0.0349309
## sam4 0.2600229 0.1852972 0.08326648 0.72768078 0.4195605 0.2982445
##             s7         s8        s9       s10       s11       s12
## sam1 0.5173039 0.03627578 0.6358517 0.3560672 0.4794818 0.8603726
## sam2 0.9682418 0.51997587 0.5762934 0.9463702 0.4856246 0.6810569
```

```
## sam3 0.5646822 0.19521899 0.5881210 0.4894135 0.0917544 0.7878934
## sam4 0.7432367 0.24150709 0.7345787 0.9173480 0.3879061 0.6024034
##            s13        s14        s15       s16        s17        s18
## sam1 0.9625107 0.64921463 0.51214372 0.2608192 0.01568284 0.07549878
## sam2 0.4402104 0.59216471 0.54887152 0.3266606 0.35872920 0.54298686
## sam3 0.9492627 0.24655105 0.44548164 0.2954519 0.90683456 0.54280131
## sam4 0.3617747 0.08594473 0.04314753 0.5781598 0.24789950 0.09762706
##            s19       s20        s21       s22        s23       s24
## sam1 0.6848625 0.3172088 0.65449068 0.9566853 0.87854259 0.2507065
## sam2 0.4230498 0.4200921 0.20150530 0.3688930 0.44524506 0.4808561
## sam3 0.7456224 0.3255062 0.84818795 0.1185494 0.07345417 0.8769561
## sam4 0.2665278 0.1624959 0.05967938 0.7779736 0.45255367 0.5115345
##            s25
## sam1 0.2770414
## sam2 0.9197584
## sam3 0.5491871
## sam4 0.8128649
##
## 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 Oct 23 16:07:17 2018 ------##

tmp_merged_compositional <- new("merged_compositional",
                                id='adas',
                                id_samples=paste0("sam", 1:30),
                                id_signatures= c('s1', 's2', 's3', 's4'), ## signature names
                                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
## -2.7931 -0.4757 -0.1486  0.7628  4.3697
##
## Coefficients: (1 not defined because of singularities)
##                                         Estimate Std. Error t value
## (Intercept)                             3.596e-02  5.387e-01   0.067
## as.matrix((x@df)[, indices_predictor])a -3.235e-05  8.739e-05  -0.370
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                                0.947
## as.matrix((x@df)[, indices_predictor])a    0.714
## as.matrix((x@df)[, indices_predictor])b       NA
##
## Residual standard error: 1.377 on 28 degrees of freedom
## Multiple R-squared:  0.004871,Adjusted R-squared:  -0.03067
## F-statistic: 0.1371 on 1 and 28 DF,  p-value: 0.714
##
##
## Response Y2 :
##
## Call:
## lm(formula = Y2 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.94194 -0.73269  0.01492  0.57745  2.14031
##
## Coefficients: (1 not defined because of singularities)
##                                         Estimate Std. Error t value
## (Intercept)                            -1.824e-01  4.317e-01  -0.422
## as.matrix((x@df)[, indices_predictor])a  6.315e-05  7.002e-05   0.902
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                                0.676
## as.matrix((x@df)[, indices_predictor])a    0.375
## as.matrix((x@df)[, indices_predictor])b       NA
##
## Residual standard error: 1.103 on 28 degrees of freedom
## Multiple R-squared:  0.02823,Adjusted R-squared:  -0.006474
## F-statistic: 0.8135 on 1 and 28 DF,  p-value: 0.3748
##
```

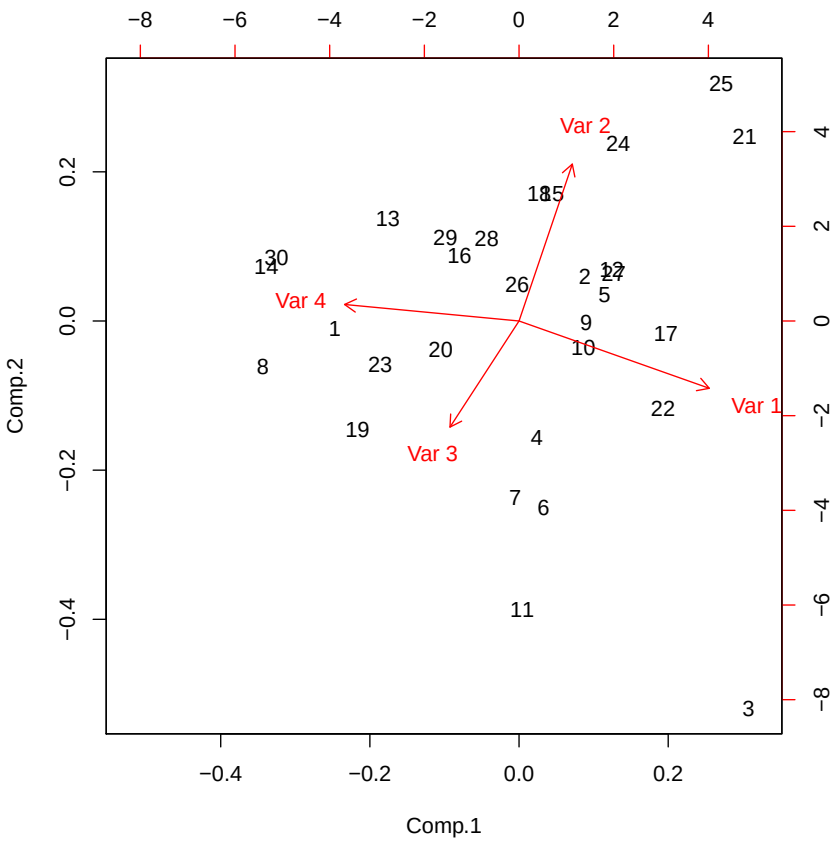
```
##
## Response Y3 :
##
## Call:
## lm(formula = Y3 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0491 -0.3923 -0.0201  0.4309  3.3966
##
## Coefficients: (1 not defined because of singularities)
##                                          Estimate Std. Error t value
## (Intercept)                             0.3396571  0.4531005   0.750
## as.matrix((x@df)[, indices_predictor])a -0.0001075  0.0000735  -1.463
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                                0.460
## as.matrix((x@df)[, indices_predictor])a    0.155
## as.matrix((x@df)[, indices_predictor])b       NA
##
## Residual standard error: 1.158 on 28 degrees of freedom
## Multiple R-squared:  0.07102,Adjusted R-squared:  0.03784
## F-statistic: 2.141 on 1 and 28 DF,  p-value: 0.1546
```

3 Importing data

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

## ##------ Tue Oct 23 16:07:17 2018 ------##

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



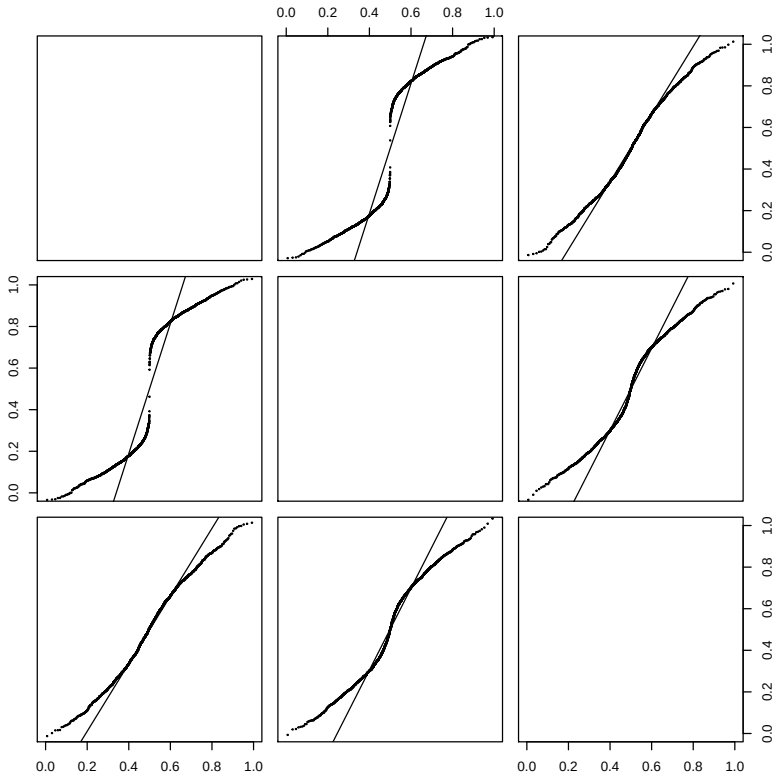
4 Other

1. Test for normality as follows:

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

## ##------ Thu Oct 25 12:14:21 2018 ------##

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

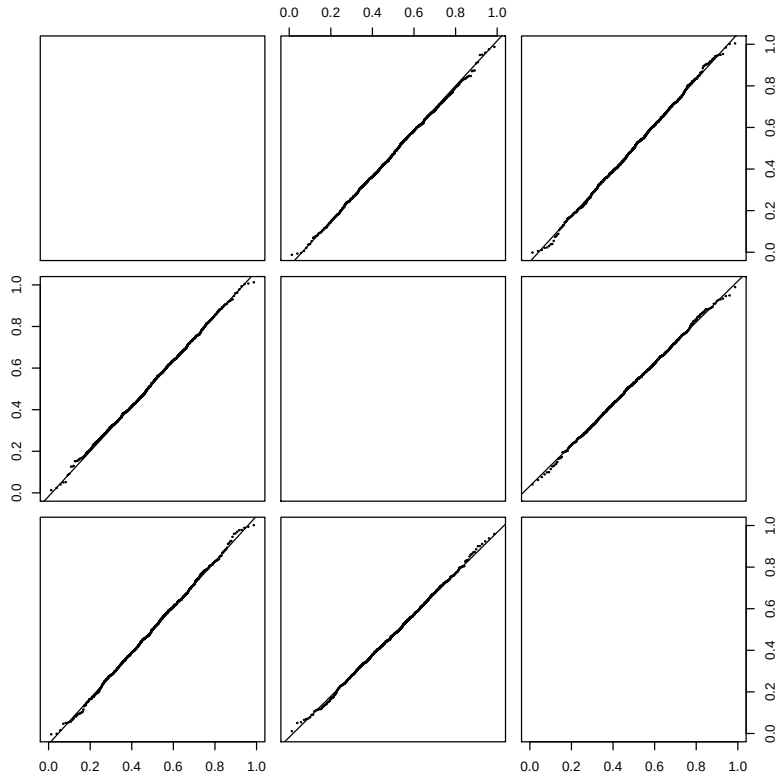



```
qqnorm.acomp(acomp(two_normal_pops@count_matrix[1:1000,]), pch=19, cex=0.2, plot.it=FAL

## 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()

## ##------ Thu Oct 25 12:14:22 2018 ------##

##TODO!!
```

The next is whether the populations have a different center:

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

## ##------ Thu Oct 25 12:14:22 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])
         }
       ))


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