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 https://github.com/lm687/CompSign/blob/master/CompSign.pdf

```
## knitr preferences
## no chache
```

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

## Skipping install of 'CompSign' from a github remote, the SHA1 (61cbadcb)
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
```

```
############################
####### 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

```
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 0.91779376 0.02385303 0.23876339 0.9712870 0.5408566 0.66646516
## sam2 0.01276465 0.34940536 0.63061196 0.6900336 0.3018326 0.60664123
## sam3 0.50539042 0.36147988 0.04297495 0.1056115 0.4906120 0.76086449
## sam4 0.70298904 0.03583288 0.65580700 0.4148143 0.5538498 0.07630959
##             s7        s8        s9        s10       s11       s12
## sam1 0.3374145 0.4773106 0.9515823 0.32474048 0.5656766 0.4174376
## sam2 0.9262440 0.7558482 0.6446747 0.54491706 0.5338909 0.8887019
## sam3 0.4919054 0.5217433 0.6016393 0.09456215 0.3792113 0.3280545
## sam4 0.8834992 0.1588780 0.4867555 0.19865778 0.3253180 0.3400624
##             s13       s14       s15       s16       s17       s18
## sam1 0.57315953 0.73772051 0.7100046 0.6773222 0.4430465 0.8818602
```

```
## sam2 0.82091263 0.49757104 0.7250833 0.7779467 0.8043129 0.3487173
## sam3 0.95243050 0.07723305 0.7091111 0.1253123 0.8933768 0.2195923
## sam4 0.09944258 0.17252315 0.4020018 0.2020037 0.2721947 0.3008867
##            s19         s20       s21       s22       s23       s24
## sam1 0.8509031 0.477927554 0.7759270 0.4968509 0.3611036 0.3978248
## sam2 0.3235608 0.008046405 0.2447500 0.5041135 0.3593709 0.2207983
## sam3 0.1355410 0.703507220 0.2849771 0.7882534 0.3600759 0.3927194
## sam4 0.8579977 0.757115631 0.3605919 0.8781320 0.8303557 0.4997538
##             s25
## sam1 0.11169879
## sam2 0.61510575
## sam3 0.78184170
## sam4 0.01722335
##
## 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

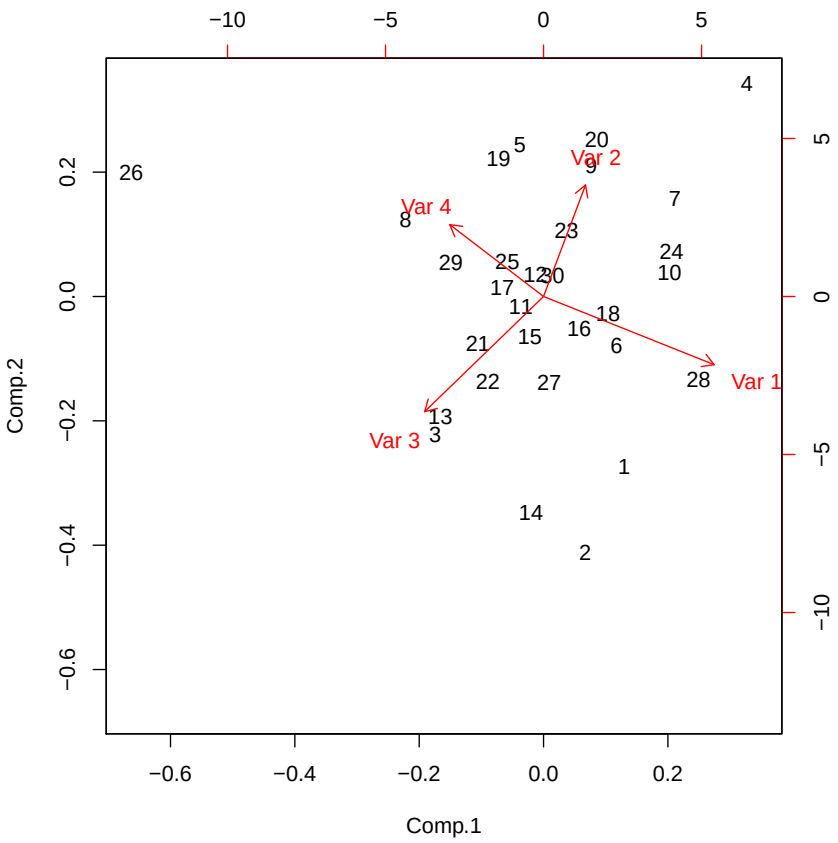
```
tmp_merged_compositional <- new("merged_compositional",
                                id='adas',
                                id_samples=paste0("sam", 1:30),
                                id_signatures= c('s1', 's2', 's3', 's4'), ## signature names
                                count_matrix=MCMCpack::rdirichlet(30, 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
## -2.9347 -0.6192  0.1741  0.8892  2.7811
##
## Coefficients: (1 not defined because of singularities)
```

```
##                                           Estimate Std. Error t value
## (Intercept)                              -2.794e-01  4.741e-01  -0.589
## as.matrix((x@df)[, indices_predictor])a  6.639e-05  7.658e-05   0.867
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                          Pr(>|t|)
## (Intercept)                                 0.560
## as.matrix((x@df)[, indices_predictor])a     0.393
## as.matrix((x@df)[, indices_predictor])b        NA
##
## Residual standard error: 1.333 on 28 degrees of freedom
## Multiple R-squared:  0.02614,Adjusted R-squared:  -0.008636
## F-statistic: 0.7517 on 1 and 28 DF,  p-value: 0.3933
##
##
## Response Y2 :
##
## Call:
## lm(formula = Y2 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1762 -0.8213 -0.0309  0.8489  2.9780
##
## Coefficients: (1 not defined because of singularities)
##                                           Estimate Std. Error t value
## (Intercept)                              -2.596e-02  5.195e-01  -0.050
## as.matrix((x@df)[, indices_predictor])a  1.697e-05  8.391e-05   0.202
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                          Pr(>|t|)
## (Intercept)                                 0.960
## as.matrix((x@df)[, indices_predictor])a     0.841
## as.matrix((x@df)[, indices_predictor])b        NA
##
## Residual standard error: 1.46 on 28 degrees of freedom
## Multiple R-squared:  0.001458,Adjusted R-squared:  -0.0342
## F-statistic: 0.0409 on 1 and 28 DF,  p-value: 0.8412
##
##
## Response Y3 :
##
## Call:
## lm(formula = Y3 ~ as.matrix((x@df)[, indices_predictor]))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -2.47089 -1.09373 -0.09707  0.98925  2.43186
##
## Coefficients: (1 not defined because of singularities)
##                                          Estimate Std. Error t value
## (Intercept)                              7.672e-01  4.930e-01   1.556
## as.matrix((x@df)[, indices_predictor])a -2.044e-04  7.963e-05  -2.566
## as.matrix((x@df)[, indices_predictor])b         NA         NA      NA
##                                         Pr(>|t|)
## (Intercept)                               0.1309
## as.matrix((x@df)[, indices_predictor])a   0.0159 *
## as.matrix((x@df)[, indices_predictor])b       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.386 on 28 degrees of freedom
## Multiple R-squared:  0.1904,Adjusted R-squared:  0.1615
## F-statistic: 6.586 on 1 and 28 DF,  p-value: 0.01591
```

3 Importing data

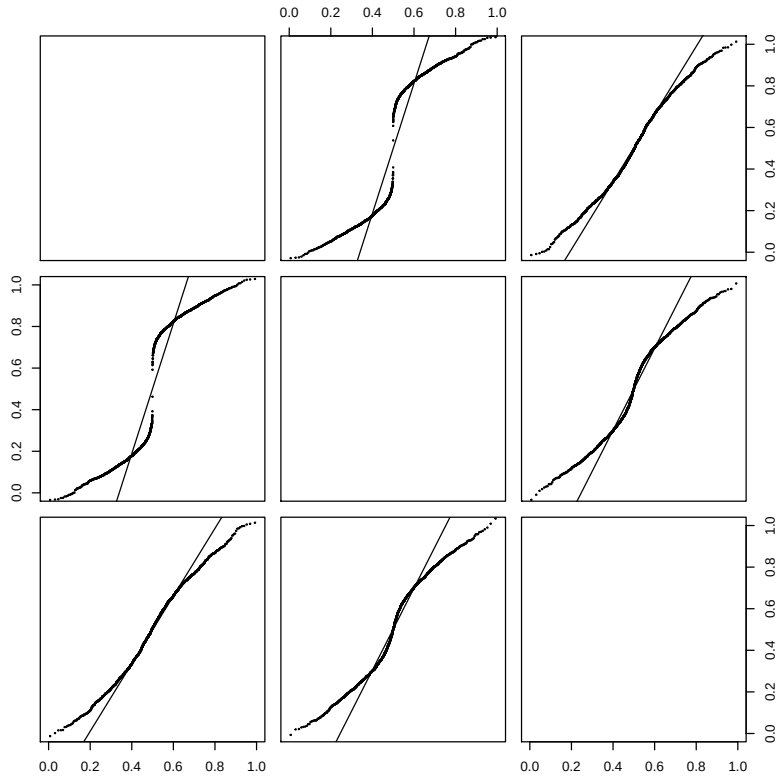
```
biplot(princomp(acomp(MCMCpack::rdirichlet(30, rep(1, 4)))))
```



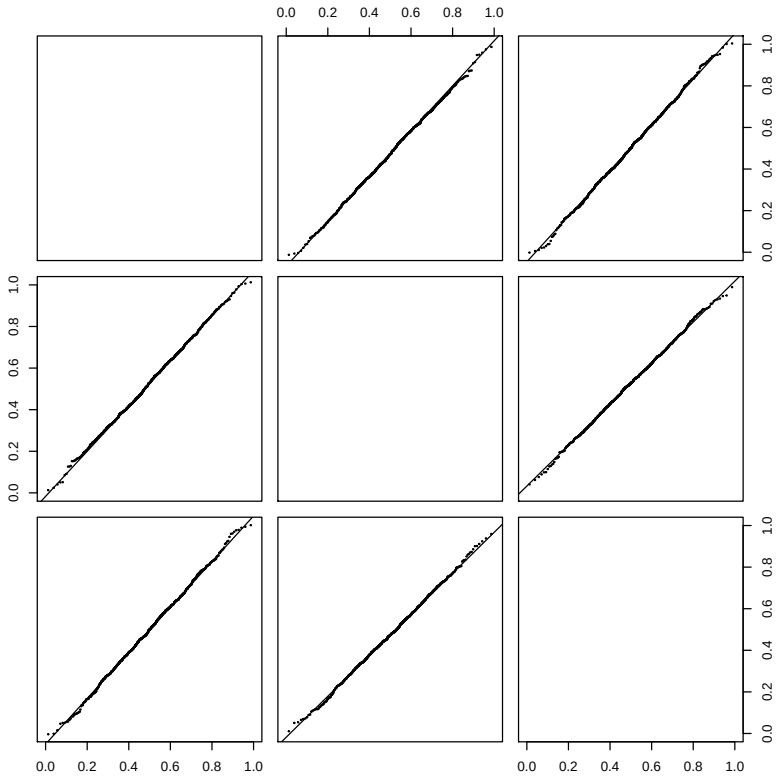
4 Other

1. Test for normality as follows:

```
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)
```


0.0 0.2 0.4 0.6 0.8 1.0