ioctitlebempty ldtitle
[1]ldtitle1

oifpackagelaterparnotes2016/07/26 arnote@real

parnote@real

otitle

oauthor

odate

Contents

1 Introduction

ccrepe is a package for analysis of sparse compositional data. Specifically, it determines the significance of association between features in a composition, using any similarity measure (e.g. Pearson correlation, Spearman correlation, etc.) The CCREPE methodology stands for Compositionality Corrected by Renormalization and Permutation, as detailed below. The package also provides a novel similarity measure, the N-dimensional checkerboard score (NC-score), particularly appropriate to compositions derived from microbial community sequencing data. This results in p-values and false discovery rate q-values corrected for the effects of compositionality. The package contains two functions `ccrepe` and `nc.score` and is maintained using the Benjamin-Hochberg-Yekutieli procedure. For greater detail, see Faust et al. [2012] and Schwager and Colleagues.

CCREPE employs several filtering steps before the data are processed. It removes any missing subjects using `na.omit`: in the two dataset case, any subjects missing in *either* dataset will be removed. Any subjects or features which are all zero are removed as well: an all-zero subject cannot be normalized (its sum is 0) and an all-zero feature has standard deviation 0 (in addition to being uninteresting biologically).

2.2 Arguments

`x` First *dataframe* or *matrix* containing relative abundances. Columns are features, rows are samples. Rows should therefore sum to a constant. Row names are used for identification if present.

`y` Second *dataframe* or *matrix* (optional) containing relative abundances. Columns are features, rows are samples. Rows should therefore sum to a constant. If both `x` and `y` are specified, they will be merged by row names. If no row names are specified for either or both datasets, the default is to merge by row number.

`sim.score` Similarity measure, such as `cor` or `nc.score`. This can be either an existing R function or user-defined. If the latter, certain properties should be satisfied as detailed below (also see examples). The default similarity measure is the Pearson correlation.

A user-defined similarity measure should mimic the interface of `cor`:

1. Take either two *vector* inputs one *matrix* or *dataframe* input.
2. In the case of two inputs, return a single number.
3. In the case of one input, return a matrix in which the (`i`,`j`)th entry is the similarity score for column `i` and column `j` in the original matrix.
4. The resulting matrix (in the case of one input) must be symmetric.
5. The inputs must be named `x` and `y`.

`sim.score.args` An optional list of arguments for the measurement function. When given, they are passed to the `sim.score` function directly. For example, in the case of `cor`, the following would be acceptable:

ototalleftmargin@ osetminipage

```
sim.score.args = list(method="spearman", use="complete.obs")
```

ototalleftmarginototalleftmargin ominipagefalse

`min.subj` Minimum number (count) of samples that must be non-missing in order to apply the similarity measure. This is to ensure that there are sufficient samples to perform a bootstrap (default: 20).

`iterations` The number of iterations for both bootstrap and permutation calculations (default: 1000).

`subset.cols.x` A vector of column indices from x to indicate which features to compare

`subset.cols.y` A vector of column indices from y to indicate which features to compare

`errthresh` If feature has number of zeros greater than $errthresh^{1/n}$, that feature is excluded

`verbose` If `TRUE`, print periodic progress of the algorithm through the dataset(s), as well as including more detailed debugging output. (default: `FALSE`).

`iterations.gap` If `verbose=TRUE`, the number of iterations between issuing status messages (default: 100).

`distributions` Optional output file for detailed log (if given) of all intermediate permutation and renormalization distributions.

`compare.within.x` A boolean value indicating whether to do comparisons given by taking all subsets of size 2 from `subset.cols.x` or to do comparisons given by taking all possible combinations of `subset.cols.x` and `subset.cols.y`. If `TRUE` but `subset.cols.y=NA`, returns all comparisons involving any features in `subset.cols.x`. This argument is only used when `y=NA`.

`concurrent.output` Optional output file to which each comparison will be written as it is calculated.

`make.output.table` A boolean value indicating whether to include table-formatted output.

2.3 Output

`ccrepe` returns a *list* containing both the calculation results and the parameters used:

`sim.score` *matrix* of simliarity scores for all requested comparisons. The (`i`,`j`)th element corresponds to the similarity score of column `i` (or the `i`th column of `subset.cols.1`) and column `j` (or the `j`th column of `subset.cols.1`) in one dataset, or to the similarity score of column `i` (or the `i`th column of `subset.cols.1`) in dataset `x` and column `j` (or the `j`th column of `subset.cols.2`) in dataset `y` in the case of two datasets.

`p.values` *matrix* of the corrected p-values for all requested comparisons. The (`i`,`j`)th element corresponds to the p-value of the (`i`,`j`)th element of `sim.score`.

`q.values` *matrix* of the Benjamini-Hochberg-Yekutieli corrected p-values. The (`i`,`j`)th element corresponds to the p-value of the (`i`,`j`)th element of `sim.score`.

`z.stat` *matrix* of the z-statistics used in generating the p-values for all requested comparisons. The (`i`,`j`)th element corresponds to the z-statistic generating the (`i`,`j`)th element of `p.values`.

2.4 Usage

ototalleftmargin@ osetminipage

```
ccrepe(
 x = NA,
 y = NA,
 sim.score = cor,
 sim.score.args = list(),
 min.subj = 20,
 iterations = 1000,
 subset.cols.x = NULL,
 subset.cols.y = NULL,
 errthresh  = 1e-04,
 verbose = FALSE,
 iterations.gap = 100,
 distributions = NA,
 compare.within.x = TRUE,
 concurrent.output = NA,
 make.output.table = FALSE)
```

ototalleftotalleftmargin ominipagefalse

2.5 Example 1

An example of how to use `ccrepe` with one dataset.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
data.rowsum <- apply(data,1,sum)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
test.input <- data.norm

dimnames(test.input) <- list(c(
 "Sample 1", "Sample 2","Sample 3","Sample 4","Sample 5",
 "Sample 6","Sample 7","Sample 8","Sample 9","Sample 10"),
 c("Feature 1", "Feature 2", "Feature 3","Feature 4"))

test.output <- ccrepe(x=test.input, iterations=20, min.subj=10)
```

ototalleftotalleftmargin ominipagefalse

ototalleftmargin@ osetminipage

```
par(mfrow=c(1,2))
plot(data[,1],data[,2],xlab="Feature 1",ylab="Feature 2",main="Non-normalized")
plot(data.norm[,1],data.norm[,2],xlab="Feature 1",ylab="Feature 2",
     main="Normalized")
```

ototalleftotalleftmargin

ototalleftotalleftmargin ominipagefalse

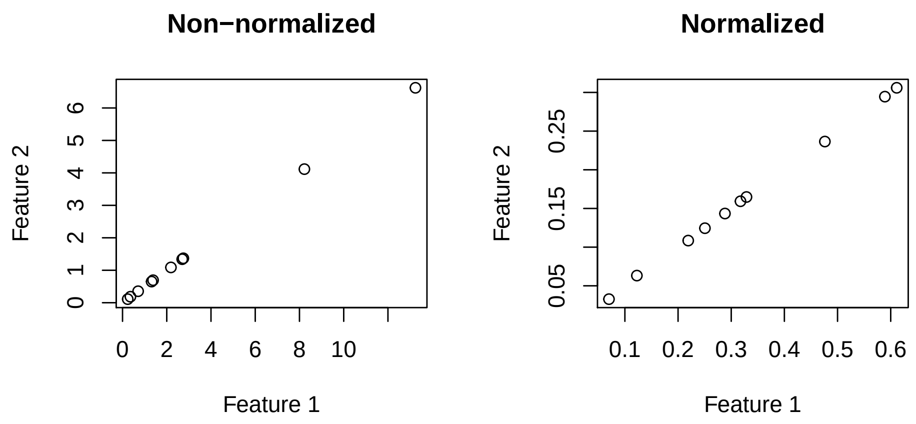

Figure 1: **Non-normalized and normalized associations between feature 1 and feature 2**
In this case we would expect feature 1 and feature 2 to be associated. In the output we see this by the positive sim.score value in the [1,2] element of test.output$sim.score and the small q-value in the [1,2] element of test.output$q.values.

ototalleftmargin@ osetminipage

```
test.output
```

```
## $p.values
##              Feature 1    Feature 2 Feature 3  Feature 4
## Feature 1           NA 1.756616e-05 0.5654526 0.16533044
## Feature 2 1.756616e-05           NA 0.2299822 0.04276288
## Feature 3 5.654526e-01 2.299822e-01        NA 0.32957253
## Feature 4 1.653304e-01 4.276288e-02 0.3295725         NA
##
## $z.stat
##            Feature 1 Feature 2  Feature 3  Feature 4
## Feature 1         NA  4.293774 -0.5747614 -1.3873652
## Feature 2  4.2937736        NA -1.2004047 -2.0260186
## Feature 3 -0.5747614 -1.200405         NA  0.9749753
## Feature 4 -1.3873652 -2.026019  0.9749753         NA
##
## $sim.score
##            Feature 1  Feature 2  Feature 3  Feature 4
## Feature 1         NA  0.9999119 -0.3876728 -0.7912952
## Feature 2  0.9999119         NA -0.3859728 -0.7923909
## Feature 3 -0.3876728 -0.3859728         NA -0.2568388
## Feature 4 -0.7912952 -0.7923909 -0.2568388         NA
##
## $q.values
##              Feature 1    Feature 2 Feature 3 Feature 4
## Feature 1           NA 0.0002496828 1.3395433 0.7833274
## Feature 2 0.0002496828           NA 0.8172331 0.3039126
## Feature 3 1.3395432651 0.8172331240        NA 0.9368990
## Feature 4 0.7833273932 0.3039125974 0.9368990        NA
```

ototalleftotalleftmargin

ototalleftmargin ominipagefalse

2.6 Example 2

An example of how to use `ccrepe` with two datasets.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
data.rowsum <- apply(data,1,sum)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized

##  [1] 1 1 1 1 1 1 1 1 1 1

test.input <- data.norm


data2 <- matrix(rlnorm(105,meanlog=0,sdlog=1),nrow=15,ncol=7)
aligned.rows <- c(seq(1,4),seq(6,9),11,12)  # The datasets dont need
                                            # to have subjects line up exactly
data2[aligned.rows,1] <-  2*data[,3] + rnorm(10,0,0.01)
data2.rowsum <- apply(data2,1,sum)
data2.norm <- data2/data2.rowsum
apply(data2.norm,1,sum)  # The rows sum to 1, so the data are normalized

##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

test.input.2 <- data2.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))
dimnames(test.input.2) <- list(paste("Sample",c(seq(1,4),11,seq(5,8),12,9,10,13,14,15)),paste('

test.output.two.datasets <- ccrepe(x=test.input, y=test.input.2, iterations=20, min.subj=10)

## Warning in preprocess_data(CA): Removing subjects
Sample 11, Sample 12, Sample 13, Sample 14, Sample 15
from dataset y because they are not in dataset x.
```

ototalleftmargin ominipagefalse

Please note that we receive a warning because the subjects don't match - only paired observations.

ototalleftmargin@ osetminipage

```
par(mfrow=c(1,2))
plot(data2[aligned.rows,1],data[,3],xlab="dataset 2: Feature 1",ylab="dataset 1: Feature 3",mai
plot(data2.norm[aligned.rows,1],data.norm[,3],xlab="dataset 2: Feature 1",ylab="dataset 1: Feat
     main="Normalized")
```

ototalleftmargin ominipagefalse

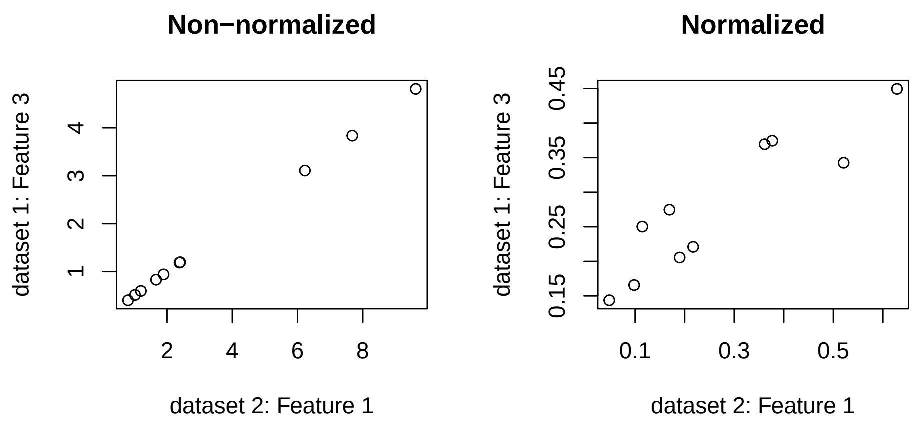

Figure 2: **Non-normalized and normalized associations between feature 1 and feature 2**
In this case we would expect feature 1 and feature 2 to be associated. In the output we see this by the positive sim.score value in the [1,2] element of test.output$sim.score and the small q-value in the [1,2] element of test.output$q.values.

ototalleftmargin@ osetminipage

```
test.output.two.datasets

## $p.values
##              Feature 1   Feature 2 Feature 3 Feature 4  Feature 5 Feature 6
## Feature 1 0.345872402 0.419383416 0.9432693 0.5006452 0.35064584 0.2216574
## Feature 2 0.265286949 0.371659168 0.8074678 0.4084374 0.37293944 0.1756130
## Feature 3 0.002050669 0.005449224 0.3733573 0.7184186 0.22331262 0.6460115
## Feature 4 0.740295138 0.511050407 0.7169565 0.3534291 0.07328915 0.3795741
##             Feature 7
## Feature 1 0.96084739
## Feature 2 0.89331951
## Feature 3 0.07237619
## Feature 4 0.53927351
##
## $z.stat
##             Feature 1  Feature 2   Feature 3  Feature 4  Feature 5  Feature 6
## Feature 1 -0.9426257  0.8074914  0.07116139  0.6734749  0.9333373 -1.2221327
## Feature 2 -1.1139819  0.8933698 -0.24369396  0.8266465  0.8909809 -1.3543869
## Feature 3  3.0827942 -2.7792043 -0.89020227  0.3605731  1.2177667  0.4593101
## Feature 4 -0.3314625  0.6572030  0.36252939 -0.9279585 -1.7910259  0.8786813
##              Feature 7
## Feature 1  0.04909023
## Feature 2  0.13410505
## Feature 3 -1.79674451
## Feature 4  0.61391195
##
## $sim.score
##              Feature 1   Feature 2   Feature 3    Feature 4  Feature 5
## Feature 1 -0.39577927  0.26949793 -0.07841653  0.196328360  0.3438109
## Feature 2 -0.39946820  0.26984242 -0.07808791  0.198252113  0.3426280
## Feature 3  0.91139207 -0.63408844 -0.25676915 -0.006946768  0.2964227
## Feature 4 -0.06748179  0.05391557  0.21966731 -0.204136706 -0.5205751
```

ototalleftototalleftmargin

```
##            Feature 6   Feature 7
## Feature 1 -0.4029356  0.17725578
## Feature 2 -0.4043042  0.18336066
## Feature 3  0.1025558 -0.46187735
## Feature 4  0.3709278  0.05731246
##
## $q.values
##           Feature 1 Feature 2 Feature 3 Feature 4 Feature 5 Feature 6
## Feature 1  4.206722 2.7004287  3.824215  3.044584  3.838301  4.043910
## Feature 2  3.629914 3.3902677  3.535539  2.794319  3.140258  3.844652
## Feature 3  0.224474 0.2982463  2.919221  3.419165  3.492092  3.367374
## Feature 4  3.376479 2.9442896  3.567307  3.517062  2.005626  2.769974
##           Feature 7
## Feature 1  3.756356
## Feature 2  3.761004
## Feature 3  2.640857
## Feature 4  2.951545
```

ototalleftotalleftmargin ominipagefalse

2.7 Example 3

An example of how to use `ccrepe` with `nc.score` as the similarity score.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
data.rowsum <- apply(data,1,sum)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized

##  [1] 1 1 1 1 1 1 1 1 1 1

test.input <- data.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))

test.output.nc.score     <- ccrepe(x=test.input, sim.score=nc.score, iterations=20, min.subj=1
```

ototalleftotalleftmargin ominipagefalse

ototalleftmargin@ osetminipage

```
par(mfrow=c(1,2))
plot(data[,1],data[,2],xlab="Feature 1",ylab="Feature 2",main="Non-normalized")
plot(data.norm[,1],data.norm[,2],xlab="Feature 1",ylab="Feature 2",
     main="Normalized")
```

ototalleftotalleftmargin ominipagefalse

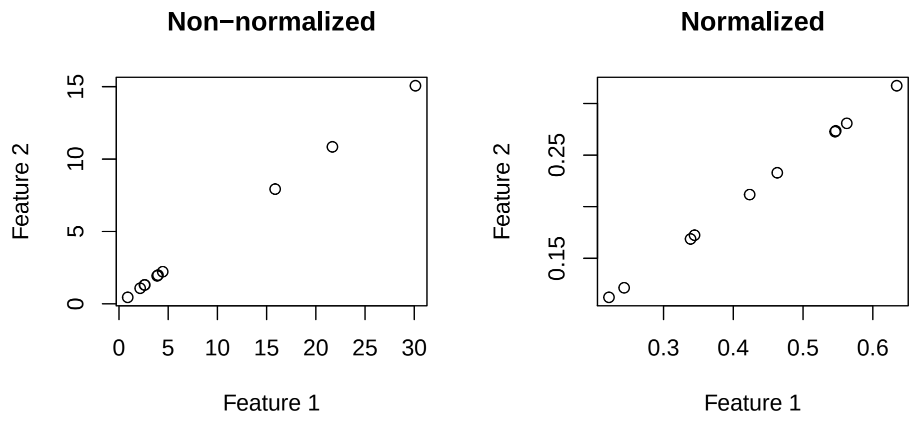

Figure 3: **Non-normalized and normalized associations between feature 1 and feature 2**
In this case we would expect feature 1 and feature 2 to be associated. In the output we see this by the positive sim.score value in the [1,2] element of test.output$sim.score and the small q-value in the [1,2] element of test.output$q.values. In this case, however, the sim.score represents the NC-Score between two features rather than the Spearman correlation.

ototalleftmargin@ osetminipage

```
test.output.nc.score

## $p.values
##               Feature 1    Feature 2 Feature 3 Feature 4
## Feature 1           NA 2.842233e-06 0.4112287 0.6724454
## Feature 2 2.842233e-06           NA 0.2225533 0.2176691
## Feature 3 4.112287e-01 2.225533e-01        NA 0.4681866
## Feature 4 6.724454e-01 2.176691e-01 0.4681866        NA
##
## $z.stat
##             Feature 1 Feature 2  Feature 3  Feature 4
## Feature 1          NA  4.681903 -0.8217333 -0.4227942
## Feature 2   4.6819035        NA -1.2197667 -1.2327499
## Feature 3 -0.8217333 -1.219767         NA  0.7254327
## Feature 4 -0.4227942 -1.232750  0.7254327         NA
##
## $sim.score
##            Feature 1 Feature 2 Feature 3 Feature 4
## Feature 1        NA   1.00000   -0.4375  -0.59375
## Feature 2   1.00000        NA   -0.4375  -0.59375
## Feature 3  -0.43750  -0.43750        NA   0.00000
## Feature 4  -0.59375  -0.59375    0.0000        NA
##
## $q.values
##               Feature 1    Feature 2 Feature 3 Feature 4
## Feature 1           NA 4.039908e-05  1.461286  1.593006
## Feature 2 4.039908e-05           NA  1.054447  1.546958
## Feature 3 1.461286e+00 1.054447e+00        NA  1.330947
## Feature 4 1.593006e+00 1.546958e+00  1.330947        NA
```

ototalleftmargin ototalleftmargin

ominipagefalse

2.8 Example 4

An example of how to use ccrepe with a user-defined sim.score function.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
data.rowsum <- apply(data,1,sum)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized

##  [1] 1 1 1 1 1 1 1 1 1 1

test.input <- data.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))

my.test.sim.score <- function(x,y=NA,constant=0.5){
       if(is.vector(x) && is.vector(y)) return(constant)
          if(is.matrix(x) && is.na(y)) return(matrix(rep(constant,ncol(x)^2),ncol=ncol(x)))
          if(is.data.frame(x) && is.na(y)) return(matrix(rep(constant,ncol(x)^2),ncol=ncol(x)))
          else stop('ERROR')
   }

test.output.sim.score    <- ccrepe(x=test.input, sim.score=my.test.sim.score, iterations=20, mi
```

ototalleftmargin ominipagefalse

ototalleftmargin@ osetminipage

```
par(mfrow=c(1,2))
plot(data[,1],data[,2],xlab="Feature 1",ylab="Feature 2",main="Non-normalized")
plot(data.norm[,1],data.norm[,2],xlab="Feature 1",ylab="Feature 2",
     main="Normalized")
```

ototalleftmargin ominipagefalse

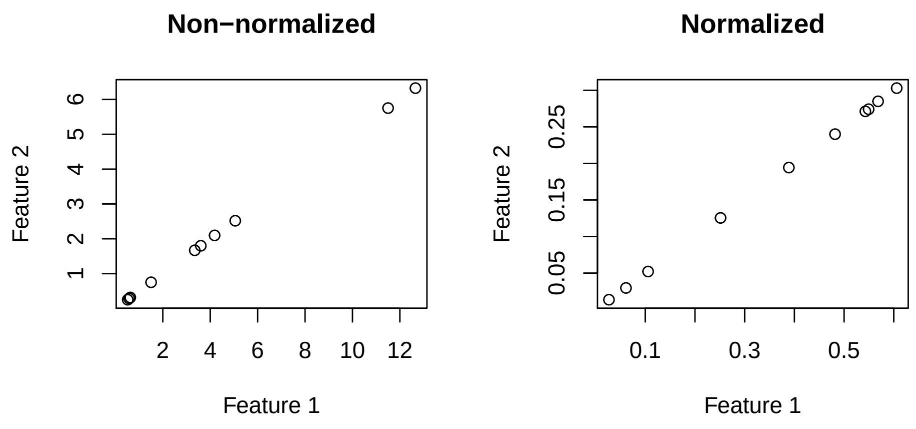

Figure 4: **Non-normalized and normalized associations between feature 1 and feature 2**
In this case we would expect feature 1 and feature 2 to be associated. Note that the values of sim.score are all 0.6 and none of the p-values are very small because of the arbitrary definition of the similarity score.

ototalleftmargin@ osetminipage

```
test.output.sim.score
```

```
## $p.values
##           Feature 1 Feature 2 Feature 3 Feature 4
## Feature 1        NA       NaN       NaN       NaN
## Feature 2       NaN        NA       NaN       NaN
## Feature 3       NaN       NaN        NA       NaN
## Feature 4       NaN       NaN       NaN        NA
##
## $z.stat
##           Feature 1 Feature 2 Feature 3 Feature 4
## Feature 1        NA       NaN       NaN       NaN
## Feature 2       NaN        NA       NaN       NaN
## Feature 3       NaN       NaN        NA       NaN
## Feature 4       NaN       NaN       NaN        NA
##
## $sim.score
##           Feature 1 Feature 2 Feature 3 Feature 4
## Feature 1        NA       0.6       0.6       0.6
## Feature 2       0.6        NA       0.6       0.6
## Feature 3       0.6       0.6        NA       0.6
## Feature 4       0.6       0.6       0.6        NA
##
## $q.values
##           Feature 1 Feature 2 Feature 3 Feature 4
## Feature 1        NA       NaN       NaN       NaN
## Feature 2       NaN        NA       NaN       NaN
## Feature 3       NaN       NaN        NA       NaN
## Feature 4       NaN       NaN       NaN        NA
```

ototalleftototalleftmargin ominipagefalse

2.9 Example 5

An example of how to use `ccrepe` when specifying column subsets.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data.rowsum <- apply(data,1,sum)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
test.input <- data.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))

test.output.1.3     <- ccrepe(x=test.input, iterations=20, min.subj=10, subset.cols.x=c(1,3))
test.output.1       <- ccrepe(x=test.input, iterations=20, min.subj=10, subset.cols.x=c(1), com
```

ototalleftototalleftmargin

```
test.output.12.3    <- ccrepe(x=test.input, iterations=20, min.subj=10, subset.cols.x=c(1,2),su
test.output.1.3$sim.score

##            Feature 1  Feature 3
## Feature 1         NA -0.2057674
## Feature 3 -0.2057674         NA

test.output.1$sim.score

##            Feature 1  Feature 2  Feature 3  Feature 4
## Feature 1         NA -0.4314052 -0.2057674 -0.5379471
## Feature 2 -0.4314052         NA         NA         NA
## Feature 3 -0.2057674         NA         NA         NA
## Feature 4 -0.5379471         NA         NA         NA

test.output.12.3$sim.score

##            Feature 1  Feature 2  Feature 3 Feature 4
## Feature 1         NA         NA -0.2057674        NA
## Feature 2         NA         NA -0.1505928        NA
## Feature 3 -0.2057674 -0.1505928         NA        NA
## Feature 4         NA         NA         NA        NA
```

ototalleftotalleftmargin ominipagefalse

3 nc.score

The `nc.score` similarity measure is an N-dimensional extension of the checkerboard score particularly suited to similarity score calculations between compositions derived from ecological relative abundance measurements. In such cases, features typically represent species abundances, and the NC-score discretizes these continuous values into one of N bins before computing a normalized similarity of co-occurrence or co-exclusion. This can be used as a standalone function or with `ccrepe` as above to obtain compositionality-corrected p-values.

3.1 General Functionality

The NC-score is an extension to Diamond's checkerboard score (see Cody and Diamond [1975]) to ordinal data, and simplifies to a calculation of Kendall's τ on binned data instead of ranked data. Let two features in a dataset with n subjects be denoted by

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{bmatrix}.$$

The binning function maps from the original data to b numbered bins in $\{1, ..., b\}$. Let the binning function be denoted by $B(\cdot)$. The co-variation and co-exclusion patterns are the same as concordant and discordant pairs in Kendall's τ. Considering a 2×2 submatrix of the form

$$\begin{bmatrix} B(x_i) & B(x_j) \\ B(y_i) & B(y_j) \end{bmatrix},$$

a co-variation pattern is counted when $(B(x_i) - B(x_j))(B(y_i) - B(y_j)) > 0$ and a co-exclusion pattern, conversely, when $(B(x_i) - B(x_j))(B(y_i) - B(y_j)) < 0$. The NC-score statistic for features x and y is then defined as

$$(\text{number of co-variation patterns}) - (\text{number of co-exclusion patterns}),$$

normalized by the Kendall's τ normalization factor accounting for ties described in Kendall [1970].

3.2 Arguments

`x` First numerical *vector*, or single *dataframe* or *matrix*, containing relative abundances. If the latter, columns are features, rows are samples. Rows should therefore sum to a constant.

`y` If provided, second numerical *vector* containing relative abundances. If given, `x` must be a *vector* as well.

`nbins` A non-negative integer of the number of bins to generate (cutoffs will be generated by the discretize function from the infotheo package).

`bin.cutoffs` A list of values demarcating the bin cutoffs. The binning is performed using the findInterval function.

`use` An optional character string givinga method for computing covariances in the presence of missing values. This must be (an abbreviaion of) on of the strings "everything", "all.obs", "complete.obs","na.or.complete", or "pairwise.complete.obs".

3.3 Output

`nc.score` returns either a single number (if called with two vectors) or a *matrix* of all pairwise scores (if called with a *matrix*) of normalized scores. This behaviour is precisely analogous to the cor function in R

3.4 Usage

ototalleftmargin@ osetminipage

```
nc.score(
 x,
 y = NULL,
 use = "everything",
 nbins = NULL,
 bin.cutoffs=NULL)
```

ototalleftototalleftmargin ominipagefalse

3.5 Example 1

An example of using `nc.score` to get a single similarity score or a matrix.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data.rowsum <- apply(data,1,sum)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
```

ototalleftototalleftmargin

```
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized

##  [1] 0.3920407 0.9723630 1.3591161 0.6423790 0.8270197 1.0181079 0.6733859
##  [8] 1.3173995 0.9984763 0.5751192

test.input <- data.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))

test.output.matrix <- nc.score(x=test.input)
test.output.num    <- nc.score(x=test.input[,1],y=test.input[,2])
```

ototalleftotalleftmargin ominipagefalse

ototalleftmargin@ osetminipage

```
par(mfrow=c(1, 2))
plot(data[,1],data[,2],xlab="Feature 1",ylab="Feature 2",main="Non-normalized")
plot(data.norm[,1],data.norm[,2],xlab="Feature 1",ylab="Feature 2",
     main="Normalized")
```

ototalleftotalleftmargin ominipagefalse

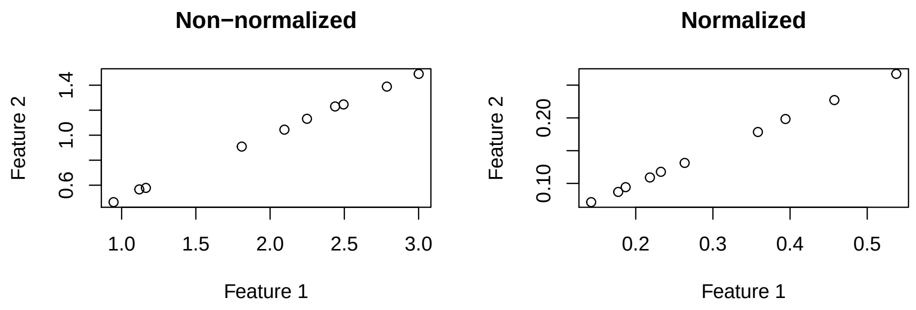

Figure 5: Non-normalized and normalized associations between feature 1 and feature 2 of the second example
Again, we expect to observe a positive association between feature 1 and feature 2. In terms of generalized checker-board scores, we would expect to see more co-variation patterns than co-exclusion patterns. This is shown by the positive and relatively high value of the [1,2] element of test.output.matrix (which is identical to test.output.num)

ototalleftmargin@ osetminipage

```
test.output.matrix

##           Feature 1 Feature 2 Feature 3 Feature 4
## Feature 1   1.00000   1.00000   0.21875   0.43750
## Feature 2   1.00000   1.00000   0.21875   0.43750
## Feature 3   0.21875   0.21875   1.00000   0.34375
## Feature 4   0.43750   0.43750   0.34375   1.00000

test.output.num

## [1] 1
```

ototalleftotalleftmargin ominipagefalse

3.6 Example 2

An example of using `nc.score` with an aribitrary bin number.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data.rowsum <- apply(data,1,sum)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized
```

```
##  [1] 1.7193961 0.4600418 2.2192763 1.3795590 1.9252256 0.5910881 0.4465658
##  [8] 0.7193439 1.2258060 1.1177629
```

```
test.input <- data.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))

test.output <- nc.score(x=test.input,nbins=4)
```

ototalleftototalleftmargin ominipagefalse

ototalleftmargin@ osetminipage

```
par(mfrow=c(1, 2))
plot(data[,1],data[,2],xlab="Feature 1",ylab="Feature 2",main="Non-normalized")
plot(data.norm[,1],data.norm[,2],xlab="Feature 1",ylab="Feature 2",
     main="Normalized")
```

ototalleftototalleftmargin ominipagefalse

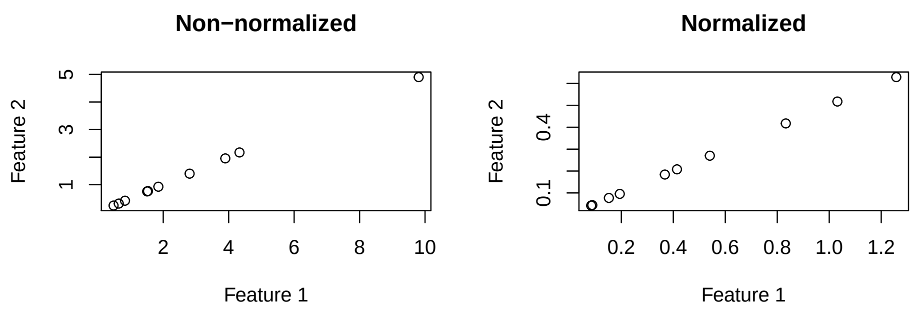

Figure 6: **Non-normalized and normalized associations between feature 1 and feature 2 of the second example**
Again, we expect to observe a positive association between feature 1 and feature 2. In terms of generalized checker-board scores, we would expect to see more co-variation patterns than co-exclusion patterns. This is shown by the positive and relatively high value in the [1,2] element of test.output. In this case, the smaller bin number yields a smaller NC-score because of the coarser partitioning of the data.

ototalleftmargin@ osetminipage

```
test.output
```

```
##            Feature 1  Feature 2  Feature 3  Feature 4
## Feature 1  1.0000000  1.0000000  0.2000000 -0.3142857
## Feature 2  1.0000000  1.0000000  0.2000000 -0.3142857
```

ototalleftototalleftmargin

```
## Feature 3  0.2000000  0.2000000  1.0000000 -0.2285714
## Feature 4 -0.3142857 -0.3142857 -0.2285714  1.0000000
```

ototalleftotalleftmargin ominipagefalse

3.7 Example 3

An example of using `nc.score` with user-defined bin edges.

ototalleftmargin@ osetminipage

```
data <- matrix(rlnorm(40,meanlog=0,sdlog=1),nrow=10,ncol=4)
data.rowsum <- apply(data,1,sum)
data[,1] = 2*data[,2] + rnorm(10,0,0.01)
data.norm <- data/data.rowsum
apply(data.norm,1,sum)  # The rows sum to 1, so the data are normalized
```

```
##  [1] 1.1963684 1.1872435 0.5714268 1.2940662 0.2371974 0.7576704 1.0687281
##  [8] 1.9319371 1.3125265 0.6846835
```

```
test.input <- data.norm

dimnames(test.input) <- list(paste("Sample",seq(1,10)),paste("Feature",seq(1,4)))

test.output <- nc.score(x=test.input,bin.cutoffs=c(0.1,0.2,0.3))
```

ototalleftotalleftmargin ominipagefalse

ototalleftmargin@ osetminipage

```
par(mfrow=c(1, 2))
plot(data[,1],data[,2],xlab="Feature 1",ylab="Feature 2",main="Non-normalized")
plot(data.norm[,1],data.norm[,2],xlab="Feature 1",ylab="Feature 2",
     main="Normalized")
```

ototalleftotalleftmargin ominipagefalse

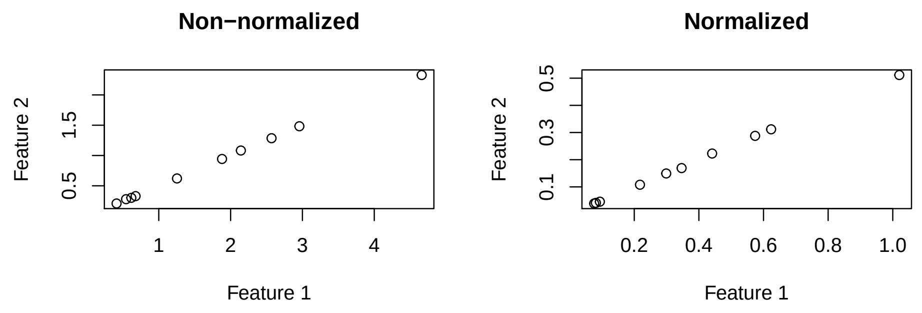

Figure 7: **Non-normalized and normalized associations between feature 1 and feature 2 of the second example** Again, we expect to observe a positive association between feature 1 and feature 2. In terms of generalized checker-board scores, we would expect to see more co-variation patterns than co-exclusion patterns. This is shown by the positive and relatively high value in the [1,2] element of test.output. The bin edges specified here represent almost absent ([0,0.001)), low abundance ([0.001,0.1)), medium abundance ([0.1,0.25)), and high abundance ([0.6,1)).

ototalleftmargin@ osetminipage

```
test.output
```

```
##             Feature 1   Feature 2   Feature 3  Feature 4
## Feature 1  1.00000000  0.85628096 -0.08858079  0.4724309
## Feature 2  0.85628096  1.00000000 -0.02702703  0.2162162
## Feature 3 -0.08858079 -0.02702703  1.00000000 -0.4324324
## Feature 4  0.47243088  0.21621622 -0.43243243  1.0000000
```

ototalleftmargin ototalleftmargin ominipagefalse

4 References

References

Martin Leonard Cody and Jared Mason Diamond. *Ecology and evolution of communities*. Harvard University Press, 1975.

Karoline Faust, J Fah Sathirapongsasuti, Jacques Izard, Nicola Segata, Dirk Gevers, Jeroen Raes, and Curtis Huttenhower. Microbial co-occurrence relationships in the human microbiome. *PLoS computational biology*, 8(7):e1002606, 2012.

M.G. Kendall. *Rank correlation methods*. Charles Griffin & Co., 1970.

Emma Schwager and Colleagues. Detecting statistically significant associtations between sparse and high dimensional compositional data. In Progress.