dispRity demo for ecological data

Thomas Guillerme

October 6, 2015

This is a quick demo for using the `dispRity` package (v.0.1.1) in ecological analysis. See the other dispRity demos for a general demo of the `dispRity` package.

1 Before starting

`dispRity` is a package for calculating disparity in `R`. To keep it short, this package allows to summarise ordinated matrices (e.g. MDS, PCA, PCO, PCoA) into single values. These ordinated matrices can represent the any-o-space of possibilities of your analyses (e.g. for morphological data, the morphospace represents all the possible morphologies).

1.1 Installation

You can install this package easily if you use the latest version of `R` and `devtools`.

```
install.packages("devtools")
library(devtools)
install_github("TGuillerme/dispRity", ref = "master")
library(dispRity)
```

Note that we use the `release` branch here which is version 0.1.1. If you want the piping-hot (and full of bugs) version, change the option `ref = "release"` to `ref = "master"`. This package depends mainly on the `ape` package and the `timeSliceTree::paleotree` function.

1.2 Data

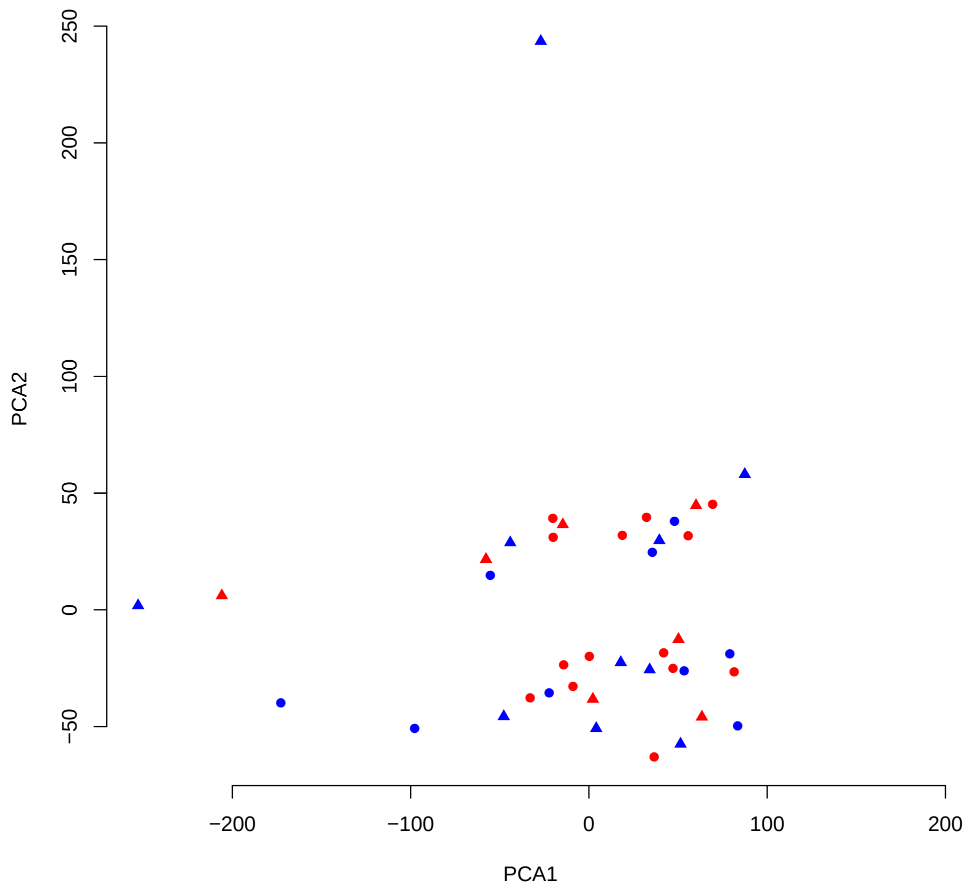
For this example with ecological data we are going to use data from McClean (unpubl.) that contains an ordination of a distance matrix between different experimental plots.

```
## Loading demo and the package data
library(dispRity)

## Loading required package:  paleotree

## let's use the matrix from McClean (unpubl.)
data(McClean_data)
## This dataset contains an ordinated matrix (in 20 dimensions) of the distance
## between experimental plots.
ord_matrix <- McClean_data$ordination
## As well as two list of different factors affecting each experimental plot:
## the treatment and the depth.
treatments <- McClean_data$treatment
depth <- McClean_data$depth
```

2 A classic PCA

A classical way to represent this ordinated data will be to use PCA plots.

```r
## The x and y axis represent the two first dimensions of the PCA
x <- ord_matrix[, 1]
y <- ord_matrix[, 2]
## The colors will represent the treatments
cols <- sub("treat1", "red", treatments)
cols <- sub("treat2", "blue", cols)
## The symbols will represent the depth
pchs <- sub(1, 16, depth)
pchs <- as.numeric(sub(2, 17, pchs))
## Graphical option
par(bty = "n")
## A classic PCA plot
plot(x, y, col = cols, pch = pchs, xlab = "PCA1", ylab = "PCA2",
    xlim = range(x) + c(0, 100))
```



This shows the distribution of the experimental plots along the two first axis of variance of the ordinated distance matrix. However, the problem, is it ignores the 18 other axis of the ordination and the PCA axis 1 and 2 do not represent a biological reality *per se* but more some ordinations of correlations between the data and some factors. Therefore, one might want to approach this problem without getting stuck in only two dimensions and consider the whole dataset as a n-dimensional object. In practice, we might be interested in looking how some experimental treatment for example, will affect the position of our experimental plots in this n-dimensional object. For example: do the experimental plots shift in some specific space of the n-dimensional object when depth increases?

3 Splitting the data

As a first split, one might want to split the data (i.e. the n-dimensional object) into subsamples that we want to compare. First let's make a factor table:

```
## Making the factor table
factors <- as.data.frame(matrix(data = c(treatments, depth), nrow = nrow(ord_matrix),
    ncol = 2, byrow = FALSE, dimnames = list(rownames(ord_matrix))))
names(factors)<-c("Treat", "Depth")
## And here is what it looks like
head(factors)

##      Treat Depth
## 1   treat1     1
## 1.1 treat1     2
## 2   treat2     1
## 2.1 treat2     2
## 3   treat1     1
## 3.1 treat1     1
```

Second, let's split the data according to these factors to create the subsamples of the ordinated space by using the `cust.series` function:

```
## Splitting the ordinated space into four subsamples
customised_series <- cust.series(ord_matrix, factors)
## Note that the output of dispRity functions are dispRity objects
class(customised_series)

## [1] "dispRity"

## These objects are automatically printed in a summary method (calling S3 print.dispRity)
## giving information about the object
customised_series

## 4 custom series for 40 taxa.
## Series:
## Treat.treat1, Treat.treat2, Depth.1, Depth.2.
```

For more details on the `dispRity` objects, see the other dispRity demos. Basically the idea is to avoid jamming the `R` console such as when using:

```
## Summarise the object
str(customised_series)
```

3.1 Calculating disparity

Now we're going to see the functionalities of the core function of this package: the `dispRity` function. This function is a modulable function that allow to simply (and quickly!) calculate disparity from a matrix. Disparity can be calculated in many ways, this function is a tool to measure disparity *as defined by the user* (and here's where the modulable part comes in). For more details on disparity, see the other dispRity demos.

One can usually decompose the disparity metrics into two elements:

1. the **class metric** that is a descriptor of the matrix. For example describing the ranges of each column in the matrix or the euclidean distances between each row and the centroid of the matrix.

2. the **summary metric** that is a summary of the class metric values. For example, the sum of the ranges or the median of the euclidean distances.

Basically the combination can be infinite between the class and summary metrics. For example, people might want to measure the median variances of the axis or the product of the distances from the centroid. However, it is probable that some metrics are better to reflect some biological aspects of the any-o-space than others...

In practice, the `dispRity` function intakes a pair of class and summary metrics as a definition of disparity. Several of these metrics are implemented in other packages (like `stats::median`, `base::sum`, etc.) and this package proposes several metrics listed in `dispRity.metric` (see `?dispRity.metric`). It is even possible to use your very own class and summary metrics! This will be actually heavily encouraged and facilitate with the `make.metric` function in a future version.

To use these metrics pairs in the `dispRity` function, it's pretty easy:

```
## For example, let's calculate the median distance between each plot and the
## centroid of the ordinated space
disparity <- dispRity(customised_series, metric = c(median, centroids))
## Note that disparity is a dispRity object and printing it just gives details
## on the object, not the results. We need to use summary.dispRity (S3) to get
## the results.
summary(disparity)

##          series  n  mean
## 1 Treat.treat1 21 82.38
## 2 Treat.treat2 19 94.68
## 3      Depth.1 23 83.98
## 4      Depth.2 17 89.72
```

Note that we calculated the median distance between plots and the centroid but the output displays mean values. This is because summary will, by default, summarise the data using the mean value. Here the mean represents the mean of the median distance for each series which is a bit useless (i.e. the mean of one value is that same value). We can display the median as well using:

```
summary(disparity, cent.tend=median)

##          series  n median
## 1 Treat.treat1 21  82.38
## 2 Treat.treat2 19  94.68
## 3      Depth.1 23  83.98
## 4      Depth.2 17  89.72

## Or even the product! It won't affect the results
summary(disparity, cent.tend=prod)

##          series  n  prod
## 1 Treat.treat1 21 82.38
## 2 Treat.treat2 19 94.68
## 3      Depth.1 23 83.98
## 4      Depth.2 17 89.72
```

This is because we did calculate the noise within our data. We can classically do so by bootstrapping the data!

3.2 Bootstrapping the data

The `dispRity` pacakge also provides easy way to bootstrap the data via the `boot.matrix` function. We can even rarefy the data to see the effect of the number of plots per series: For more details on the bootstrapping options, see the other dispRity demos.

```
bootstrapped_data <- boot.matrix(customised_series, bootstraps=100)
rarefied_data <- boot.matrix(customised_series, bootstraps=100, rarefaction=TRUE)
## Note that the output is a dispRity object giving some details on the series and the bootstraps
bootstrapped_data

## Bootstrapped ordinated matrix with 40 taxa.
## Series:
## Treat.treat1, Treat.treat2, Depth.1, Depth.2.
## Data was split using custom method.
## Data was bootstrapped 100 times, using the full bootstrap method.
```

We can now rerun a more robust disparity analysis:

```
disparity_BS <- dispRity(bootstrapped_data, metric = c(median, centroids))
disparity_rare <- dispRity(rarefied_data, metric = c(median, centroids))
## Note that calculation time is increased!
```

3.3 Summarising the data

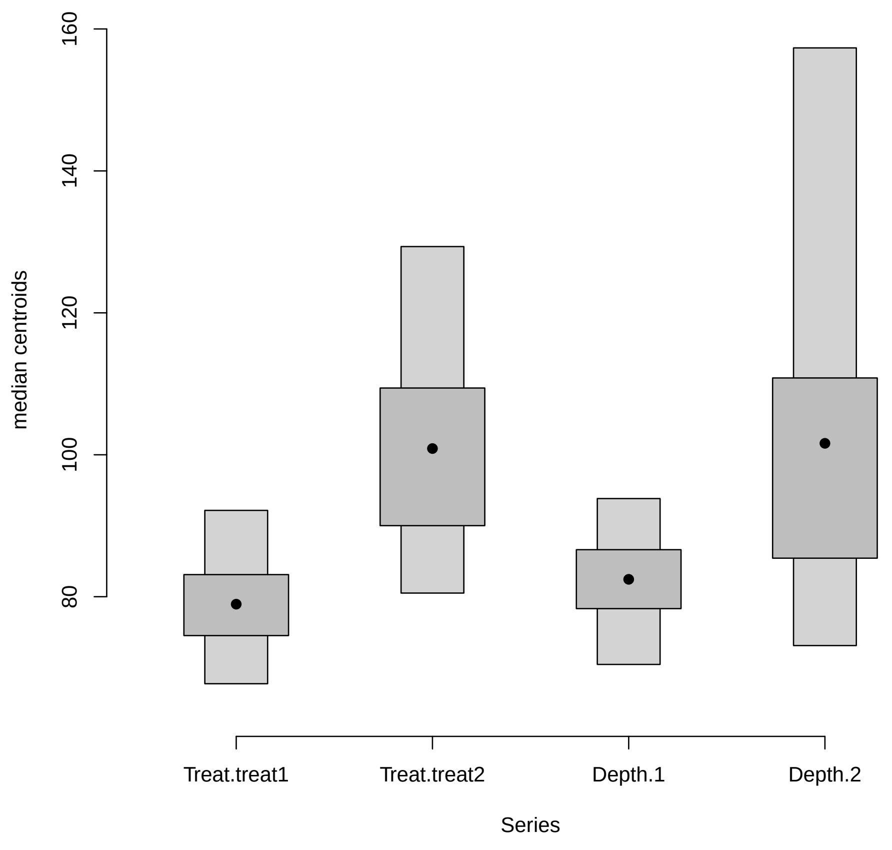
We can now summarise the data using various options such as the confidence intervals levels and the central tendency.

```
## The default:
summary(disparity_BS)

##          series  n  mean  2.5%   25%   75% 97.5%
## 1 Treat.treat1 21  79.0 67.74 74.52  83.1  92.2
## 2 Treat.treat2 19 100.9 80.51 90.01 109.4 129.3
## 3      Depth.1 23  82.5 70.46 78.32  86.6  93.8
## 4      Depth.2 17 101.6 73.12 85.43 110.8 157.3

## The CIs are calculated as 50 and 95 and the central tendency is the mean by default.
## But we can specify different options
summary(disparity_BS, CI=90, cent.tend=median)

##          series  n median    5%   95%
## 1 Treat.treat1 21  78.90 69.15  90.1
## 2 Treat.treat2 19  98.85 81.90 126.9
## 3      Depth.1 23  82.29 72.19  92.2
## 4      Depth.2 17  97.38 74.90 152.0

## Finally we can see the results of the rarefaction analysis:
head(summary(disparity_rare))
```

```
##          series n mean  2.5%   25%  75% 97.5%
## 1 Treat.treat1 3 67.4 29.20 54.16 77.0 128.8
## 2 Treat.treat1 4 74.9 49.65 61.05 85.0 115.8
## 3 Treat.treat1 5 74.6 50.72 64.49 82.0 115.0
## 4 Treat.treat1 6 75.1 55.62 65.21 82.2 103.4
## 5 Treat.treat1 7 76.1 53.77 68.14 83.3 107.9
## 6 Treat.treat1 8 77.8 58.43 69.23 83.4 109.8

## This outputs a longer table with all the variations of plots down to 3 plots per series.
```

Finally we can also plot the results using the simple `plot.dispRity` S3 method:

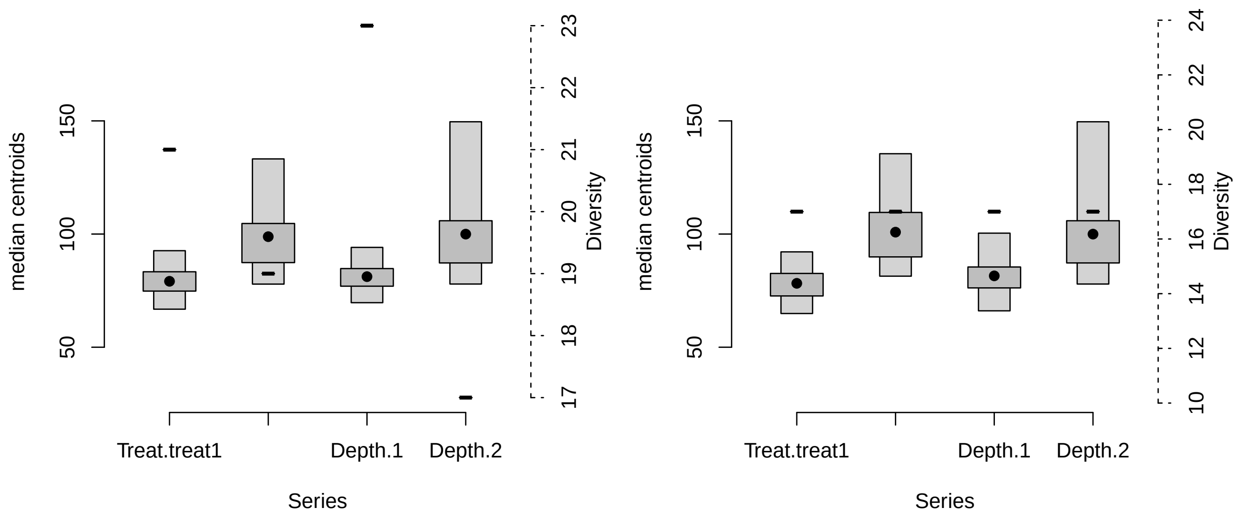
```
## Graphical option
par(bty = "n")
## Plotting the score for each groups
plot(disparity_BS)
```



Or have a look at the effect of the number of experimental plots:

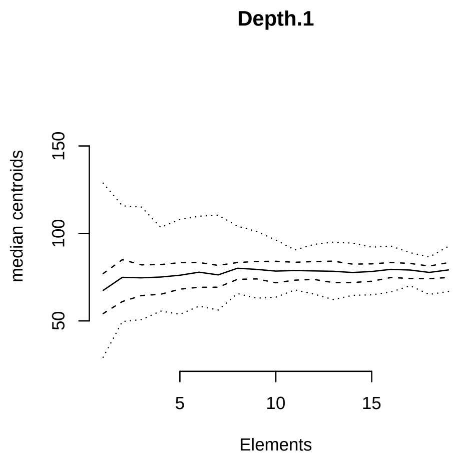
```
## Graphical options
quartz(width = 10, height = 5) ; par(mfrow = (c(1,2)), bty = "n")
## The same but looking at the number of plots
plot(disparity_rare, diversity = TRUE)
```

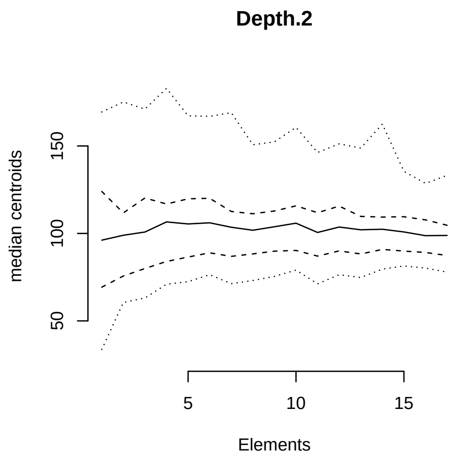
```r
## With the same number of plots per group
plot(disparity_rare, diversity = TRUE, rarefaction = 17)
```

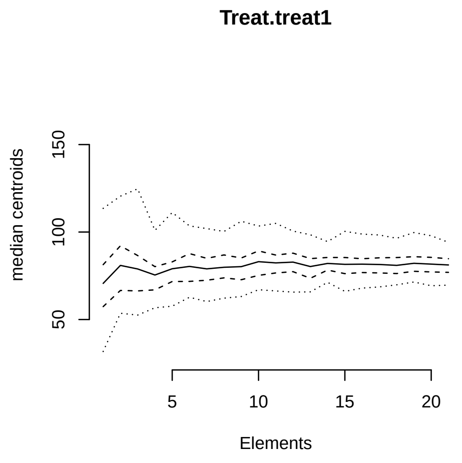


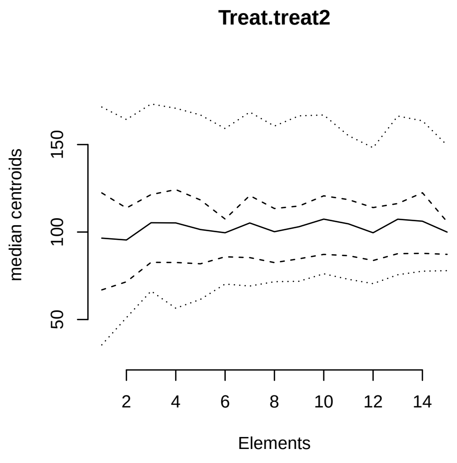
Or event have a look at the rarefaction curves:

```r
## Graphical option
par(bty = "n")
## Plotting the rarefaction curves
plot(disparity_rare, rarefaction = "plot")
```


Depth.1
median centroids
50
100
150
5
10
15
Elements


Depth.2
median centroids
50
100
150
5
10
15
Elements


Treat.treat1
median centroids
50
100
150
5
10
15
20
Elements


Treat.treat2
median centroids
50
100
150
2
4
6
8
10
12
14
Elements