`dispRity` manual

Thomas Guillerme

November 25, 2015

`dispRity` is a package for calculating disparity in `R`. It allows to summarise ordinated matrices (e.g. MDS, PCA, PCO, PCoA) into single values.

Contents

1 Before starting

1.1 Glossary

Because this package is aimed to be multidisciplinary, many names or terms used in this tutorial might be non-familiar to certain fields. Here is a list of what are the exact meaning of these term:

- **Ordinated space**: it designates the mathematical multidimensional object studied here. In morphometrics, this one is often referred as being the morphospace, or the cladisto-space for cladisticts or the eco-space for ecology, etc. In practice though, this term designates an ordinated matrix where the columns represent the dimensions of the ordinated space (often > 3!) and the rows represent the elements within this space.

- **Elements**: it designates the rows of the ordinated space, elements can be either taxa, field sites, countries, etc...
- **Dimensions**: it designates the columns of the ordinated space. The dimensions can also be referred to as axis.
- **Series**: it designates sub-samples of the ordinated space. Basically a series contain the same number of dimensions as the morphospace but might contain a smaller number of elements. For example, if our ordinated space is composed of birds and mammals (i.e. the elements) and 50 dimensions, we can create two series of just mammals or birds as elements (but the same 50 dimensions) to look at the difference in disparity between both groups.

1.2 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 = "release")
library(dispRity)
```

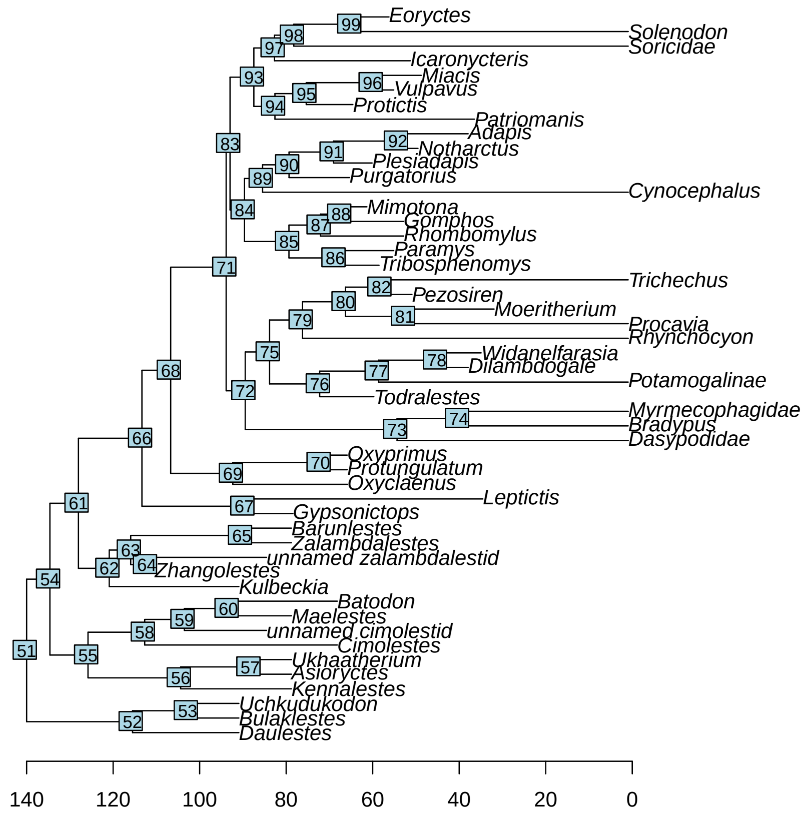
Note that we use the `release` branch here which is version 0.1.2. For the piping-hot (but potentially full of bugs) version, one can change the argument `ref = "release"` to `ref = "master"`. This package depends mainly on the `ape` package and the `timeSliceTree::paleotree` function.

1.3 Data

In this tutorial we are going to use a subset of the ordinated cladistic data from Beck and Lee (2014) that contains 50 taxa ordinated using their cladistic distance (i.e. the distance between their discrete morphological characters). Note that this data is more oriented towards palaeobiology analysis but that it can apply to other disciplines. Please refer to the GitHub page: github.com/TGuillerme/dispRity for other vignettes covering some specific example of the package with ecological data.

```
## Loading the package
library(dispRity)

## Loading required package:  paleotree

## Loading the ordinated matrix containing 50 taxa
data(BeckLee_mat50)
dim(BeckLee_mat50)

## [1] 50 48

head(BeckLee_mat50[,1:5])

##                    [,1]          [,2]        [,3]       [,4]      [,5]
## Cimolestes   -0.5319679  0.1117759259  0.09865194 -0.1933148 0.2035833
## Maelestes    -0.4087147  0.0139690317  0.26268300  0.2297096 0.1310953
## Batodon      -0.6923194  0.3308625215 -0.10175223 -0.1899656 0.1003108
## Bulaklestes  -0.6802291 -0.0134872777  0.11018009 -0.4103588 0.4326298
## Daulestes    -0.7386111  0.0009001369  0.12006449 -0.4978191 0.4741342
## Uchkudukodon -0.5105254 -0.2420633915  0.44170317 -0.1172972 0.3602273

## Loading another ordinated matrix containing 50 tips + 49 nodes
data(BeckLee_mat99)
dim(BeckLee_mat99)
```

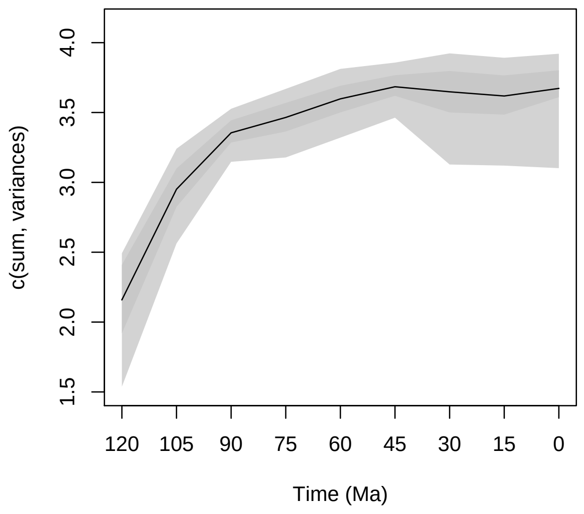
```
## [1] 99 97

head(BeckLee_mat99[,1:5], 2)

##                    [,1]       [,2]       [,3]       [,4]        [,5]
## Cimolestes -0.6082437 -0.0323683 0.08458885 -0.4338448 -0.30536875
## Maelestes  -0.5730206 -0.2840361 0.01308847 -0.1258848  0.06123611

tail(BeckLee_mat99[,1:5], 2)

##              [,1]      [,2]       [,3]       [,4]       [,5]
## n48 -0.05529018 0.4799330 0.04118477 0.04944912 -0.3558830
## n49 -0.13067785 0.4478168 0.11956268 0.13800340 -0.3222785

## Loading a list of first and last occurrence data
data(BeckLee_ages)
head(BeckLee_ages)

##             FAD  LAD
## Adapis     37.2 36.8
## Asioryctes 83.6 72.1
## Leptictis  33.9 33.3
## Miacis     49.0 46.7
## Mimotona   61.6 59.2
## Notharctus 50.2 47.0

## Loading the phylogeny
data(BeckLee_tree)
plot(BeckLee_tree) ; nodelabels(cex=0.8) ; axisPhylo(root=140)
```



1.4 A quick go through

Here is a really crude and quick go through the package to show some of the features of this package. Note that all these features will be discussed in more details below.

```
## Splitting the data
sliced_data <- time.series(BeckLee_mat99, BeckLee_tree, method = "continuous",
    model = "acctran", time = rev(seq(from=0, to=130, by=15)), FADLAD = BeckLee_ages)

## Some tips have no FAD/LAD and are assumed to be single points in time.

## Bootstrapping the data
bootstrapped_data <- boot.matrix(sliced_data, 100)
## Calculating disparity
sum_of_variances <- dispRity(bootstrapped_data, metric = c(sum, variances))
## Summarising the results
summary(sum_of_variances)

##   series  n observed  mean  2.5%   25%   75% 97.5%
## 1    120  5    2.699 2.159 1.538 1.919 2.409 2.493
## 2    105 11    3.275 2.951 2.563 2.825 3.099 3.241
## 3     90 18    3.534 3.355 3.147 3.286 3.444 3.527
## 4      75 19    3.660 3.465 3.178 3.364 3.569 3.669
## 5      60 20    3.805 3.598 3.320 3.501 3.691 3.812
## 6      45 14    3.956 3.684 3.463 3.620 3.767 3.857
## 7      30 10    4.055 3.648 3.128 3.501 3.797 3.923
## 8      15 10    4.055 3.618 3.120 3.485 3.765 3.892
## 9       0 10    4.055 3.672 3.102 3.610 3.802 3.921

plot(sum_of_variances)
## Testing some the effect of time on disparity
summary(test.dispRity(sum_of_variances, test = aov, comparisons = "all"))

##               Df Sum Sq Mean Sq F value Pr(>F)
## series         8 202.92  25.364   716.5 <2e-16 ***
## Residuals    891  31.54   0.035
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```



2 Package specificities

2.1 The `dispRity` objects

Disparity analysis can involve a lot of shuffling around with many matrices (especially when bootstrapping the data) which can be a bit impractical to visualise and quickly jam your `R` console. For example, we can have a look at the structure of the object created in the quick example:

```
str(sum_of_variances)
## That's a more than 4500 lines of output!
```

Therefore this package proposes a new class of object called `dispRity` objects. These objects allow to easily use a S3 method functions such as `summary.dispRity` (just called as `summary`; see section 3.5) or `plot.dispRity` (just called as `plot`; see section 3.6). But also, this allows to use the S3 method for printing `dispRity` objects via `print.dispRity` that allows to summarise the content of the objects similar to the `phylo` class objects (see `print.phylo::ape`).

```r
## Which class is the sum_of_variances object?
class(sum_of_variances)

## [1] "dispRity"

## What's in the object
names(sum_of_variances)

## [1] "data"      "disparity" "elements"  "series"    "call"

## We can summarise it using the S3 method print.dispRity
sum_of_variances

## Disparity measurements across 9 series for 99 elements
## Series:
## 120, 105, 90, 75, 60, 45 ...
## Disparity calculated as: c(sum, variances) for 97 dimensions.
## Data was split using continuous method.
## Data was bootstrapped 100 times, using the full bootstrap method.

## This displays some information about what's in the object
```

Note however, that it is always possible to recall the full object using the argument `all=TRUE`:

```r
## Displaying the full object
print(sum_of_variances, all=TRUE)
```

Finally, some utility functions such as `get.dispRity` or `extract.dispRity` allows to access to some specific content of the object:

```r
## Extracting some specific series from the disparity object
series_1_and_4 <- get.dispRity(sum_of_variances, what=c(1,4))
series_1_and_4

## Disparity measurements across 2 series for 24 elements
## Series:
## 120, 75.
## Disparity calculated as: c(sum, variances) for 97 dimensions.
## Data was split using continuous method.
## Data was bootstrapped 100 times, using the full bootstrap method.

## The observed disparity
extract.dispRity(sum_of_variances)

##      120      105       90       75       60       45       30       15        0
## 2.698895 3.274613 3.533596 3.660471 3.804800 3.955998 4.054575 4.054575 4.054575

## The list of bootstrapped scores of disparity
str(extract.dispRity(sum_of_variances, observed = FALSE))

## List of 9
##  $ 120: num [1:100] 2.4 2.36 2.36 2.48 2.1 ...
##  $ 105: num [1:100] 2.7 2.83 2.82 3.14 2.64 ...
##  $ 90 : num [1:100] 3.48 3.33 3.44 3.28 3.21 ...
##  $ 75 : num [1:100] 3.57 3.62 3.67 3.57 3.48 ...
##  $ 60 : num [1:100] 3.71 3.62 3.49 3.41 3.49 ...
##  $ 45 : num [1:100] 3.74 3.89 3.68 3.74 3.74 ...
##  $ 30 : num [1:100] 3.87 3.7 3.84 3.64 3.86 ...
##  $ 15 : num [1:100] 3.31 3.86 3.74 3.73 3.41 ...
##  $ 0  : num [1:100] 3.85 3.67 3.76 3.18 3.64 ...
```

2.2 Modular functions

This package aims to be a modular package where users can personalise some aspects of the package fairly easily. In this version 0.1.2 only one function is fully modular (`dispRity` ; see section 3.4) but more will be hopefully available in the near future (see section 4).

The modular idea is that users can use implemented tools to help facilitating their own personalised function. For example, the `dispRity` function intake a `metric` argument designating how disparity should be calculated. This argument can take some already implemented functions such as `mean` but also some completely new ones such as the sum divided by length:

```
## Disparity measured as the mean
summary(dispRity(sliced_data, metric = mean))

##   series  n observed
## 1    120  5   -0.005
## 2    105 11   -0.010
## 3     90 18   -0.010
## 4     75 19   -0.001
## 5     60 20    0.005
## 6     45 14    0.013
## 7     30 10    0.017
## 8     15 10    0.017
## 9      0 10    0.017

## A function for measuring the sum divided by the length (the mean!)
sum.by.length <- function(X)  sum(X)/length(X)
## Disparity measured as the mean divided by the length
summary(dispRity(sliced_data, metric = sum.by.length))

##   series  n observed
## 1    120  5   -0.005
## 2    105 11   -0.010
## 3     90 18   -0.010
## 4     75 19   -0.001
## 5     60 20    0.005
## 6     45 14    0.013
## 7     30 10    0.017
## 8     15 10    0.017
## 9      0 10    0.017

## Giving the exact same results!
```

For more information concerning this modularity, please refer to the GitHub page: github.com/TGuillerme/dispRity for the vignette about the metric implementation in `dispRity` .

3 Functions

3.1 `cust.series`

3.2 `time.series`

3.3 `boot.matrix`

3.4 `dispRity`

3.5 `summary`

3.6 `plot`

3.7 `test.dispRity`

3.8 utilities

`tree.age`

This function allows to calculate the age of each individual nodes and tips in a tree. It can either use the root age of the tree (if present as `$root.time`) or else calculate the age using a user defined root age via the `age` argument. Also, it is possible to decide whether the time is calculated towards the past (e.g. million years ago) or towards the present (e.g. in time since the origin).

```
## This tree has a root age
BeckLee_tree$root.time

## [1] 139.074

## So we can get the age of each tips and nodes directly
head(tree.age(BeckLee_tree))

##    ages     elements
## 1 90.00    Daulestes
## 2 90.00  Bulaklestes
## 3 90.00 Uchkudukodon
## 4 77.85  Kennalestes
## 5 77.85   Asioryctes
## 6 77.85 Ukhaatherium

## But we can also decide to make the age relative (between 1 and 0)
head(tree.age(BeckLee_tree, age = 1))

##    ages     elements
## 1 0.647    Daulestes
## 2 0.647  Bulaklestes
## 3 0.647 Uchkudukodon
## 4 0.560  Kennalestes
## 5 0.560   Asioryctes
## 6 0.560 Ukhaatherium

## Or even relative, but from the root (i.e. how far are the nodes/tips
## from the root)
head(tree.age(BeckLee_tree, age = 1, order = "present"))
```

```
##         ages     elements
## 1 0.3528637    Daulestes
## 2 0.3528637  Bulaklestes
## 3 0.3528637 Uchkudukodon
## 4 0.4402271  Kennalestes
## 5 0.4402271   Asioryctes
## 6 0.4402271 Ukhaatherium
```

4 Developments

As stated at the start of the demo, this version 0.1.2 is still in development and many parts are missing. Here are the new functionalities that will be implemented before the proper release version (v.1).

4.1 More user defined stuff

I intend also develop functions to help users to develop their own algorithms for the bootstrap method (via `make.boot`) or the evolutionary models (via `make.model`). Both functions will provide similar testing as the `make.metric` function.

4.2 Faster!

Finally, for long analysis, I intend to develop a parallel running version of the package. In fact, most of the internal functions are base on `lapply` functions that can be easily passed to `snow::parLapply` or similar parallel functions.

References

Beck, R. M. and M. S. Lee. 2014. Ancient dates or accelerated rates? Morphological clocks and the antiquity of placental mammals. Proceedings of the Royal Society B: Biological Sciences 281:1–10.