Traits data in R

Thomas Guillerme (t.guillerme@sheffield.ac.uk)

2024-08-07

Contents

Chapter 1

Preamble

1.1 Mega-disclaimer

English people have a horrible saying: "there's more than one way to skin a cat" (which, funnily enough, the origin of this idiom and taxidermists might disagree) which applies to this whole workshop. Here you'll have my way of looking at morphological traits and doing disparity analyses that is not by any means *THE* way to do it. Hopefully by the end of the workshop you'll feel comfortable enough exploring it your way! I've also tried to add contradiction throughout the workshop when my claims are not widely shared. Make sure you have a look at these!

1.2 Requirements

1.2.1 General computer level

In this workshop I will assume you are already familiar with how your computer works basically. You know how to do the following:

- Open a file with a specific software (sometimes that's not just double clicking)
- Locate a file in your computer (you know what a path is)

1.2.1.1 Required software

- `R` (the latest version)[1]
- A plain text editor (NOT WORD!)

[1] **Always update your software**. If you do that already, thank you very much. If you don't, have a thought for the software developers that try very hard to correct/improve/update your favourite software. They don't do it for no reason (and no, it's not that hard to click on "update" button).

- *recommended:* an IDE (integrated development environment). For example Rstudio or Sublime Text.

1.2.2 `R` level

I will assume you are already familiar with basic `R`. The basic notions that I'll assume you know are:

- What is a package (e.g. `ape` or `dispRity`)
- What is an object (e.g. `this_object <- 1`)
- What is an object's class (e.g. the class `"matrix"` or `"phylo"`)
- What is a function (e.g. the function `mean(c(1,2))`)
- How to access function manuals (e.g. `?mean`)

1.2.2.1 Required packages

For this tutorial we will be using the packages `ape` (Paradis and Schliep (2019)), `dispRity` (Guillerme (2018)) and `treats` (Guillerme (2024)):

```
required_packages <- c("ape", # for analysing phylogenetic data
                       "dispRity", # for disparity analyses
                       "treats") # for some trees and traits plots
install.packages(required_packages)
```

Note that you can also use the excellent following packages for more specific tasks not covered here:

- `Claddis` for analysing discrete morphological characters (Lloyd (2018))
- `paleotree` for analysing palaeo phylogenetic data (Bapst (2012))
- `strap` (Bell and Lloyd (2015)) or `deeptime` (Gearty (2024)) for plotting nice geological time scales
- `palaeoverse` (Jones et al. (2023)) for wrangling palaeo data
- `Morpho` (Schlager (2017)) or `geomorph` (Baken et al. (2021)) for analysing geometric morphometric data

Let's get into it.

1.3 Glossary

Trait data is going to be the generic term here to designate any type of data that describe some measurable features of a specimen, species, group, etc. Here is a glossary table from Guillerme, Puttick, et al. (2020) that specify what I mean throughout this book (and check Mammola et al. (2021) if you're more coming from ecology).

Mathematics	Ecology	Macroevolution	This book
Matrix ($n \times d$) with a structural relation between rows and columns	Functional space, morphospace, etc.	Morphospace, traitspace, etc.	traitspace
Rows (n)	Taxa, field sites, environments, etc.	Taxa, specimen, populations, etc.	observations
Columns (d)	Traits, Ordination scores, distances, etc.	Traits, ordination scores, distances, etc.	dimensions
Matrix subset ($m \times d$; $m \leq n$)	Treatments, phylogenetic group (clade), etc.	Clades, geological stratum, etc.	group
Statistic (i.e. a measure)	Dissimilarity index or metric, hypervolume, functional diversity, etc.	Disparity metric or index	space occupancy measure
Multidimensional analysis	Dissimilarity analysis, trait analysis, etc.	Disparity analysis, disparity-through-time, etc.	multidimensional analysis

Note that here traits are treated as unidimensional (i.e. one column = one trait) but it doesn't need to be the case! For example you can have a multidimensional trait that is described using multiple numbers (a multidimensional trait), for the trait "location" can be described using both latitude and longitude (a 2D trait).

1.4 How does this work?

Throughout the manuscript you will see some different coloured zoned that you can use to either to load your own data (in blue), catch up if you feel a bit behind (orange), or play around with different options (green).

1.4.1 Use your own data

USE YOUR DATA: load your own data!

```
## Read your favourite csv file (you'll need to set up the path yourself!)
continuous_data <- read.csv("path_to_my_super_serious_dataset.csv")
```

1.4.2 Catching up

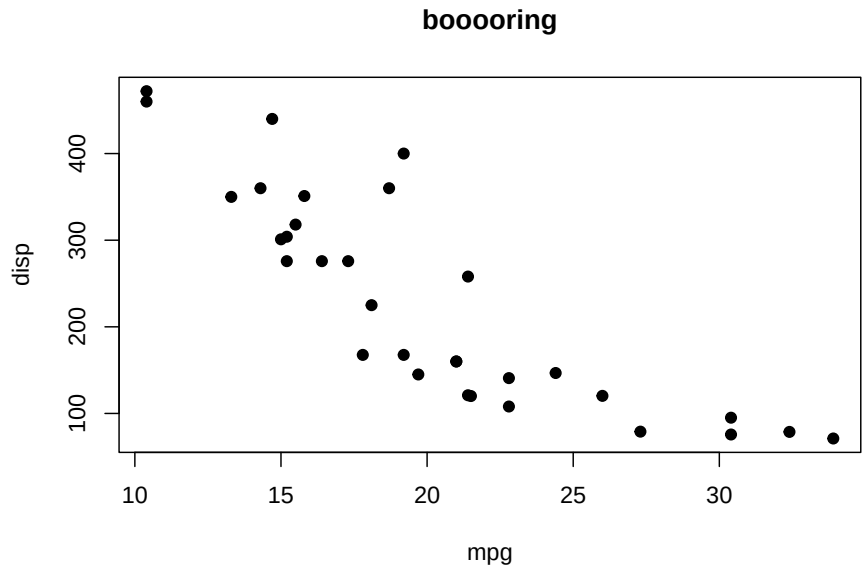
CATCHING UP ZONE: loading some example data!

```
## Loading some data
data(mtcars)
## Giving the data the correct name so that you can follow
my_data <- mtcars
```

1.4.3 Tinker time!

TINKER TIMES: play around with this very complicated plot.

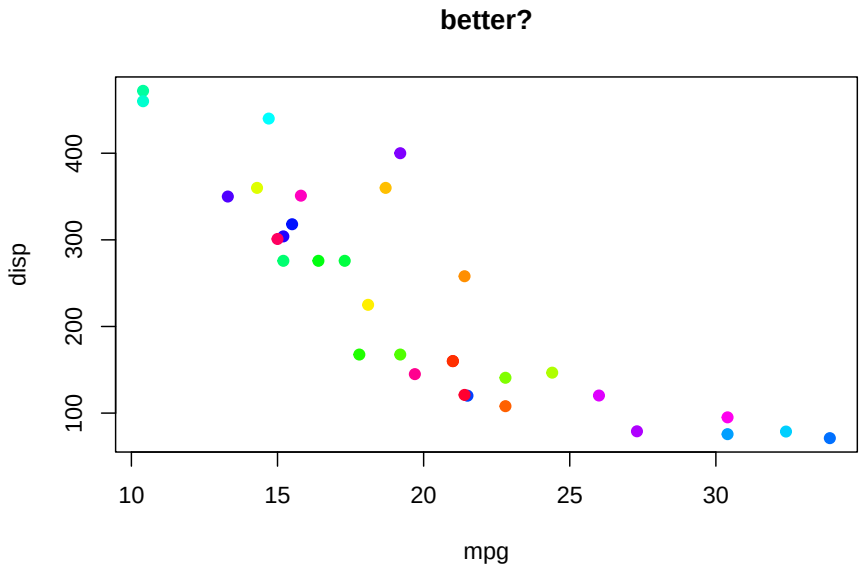
```
## Plot data
plot(my_data[, c(1,3)], col = "black", pch = 19, main = "booooring")
```



Can you find a solution to make it less boring?

[**Click to expand the solution**]:

```
## Plot data
plot(my_data[, c(1,3)], col = rainbow(nrow(my_data)), pch = 19, main = "better?")
```



1.5 Help me!

Whether you are doing this workshop online or in person, please help me improve this whole workshop:

- Something is not working
- Something is unclear
- There's a typo (realy)!

You can just shout at me or drop me an email!

1.6 References

```
## The packages to load for the section
library(dispRity)
library(ape)
library(treats)
```

Chapter 2

From observations to traitspaces

Here we will look how to go from some observations to a traitspace in `R`. This step allows us to go from observed/collected data into a multidimensional traitspace that we can then analyse further for our research.

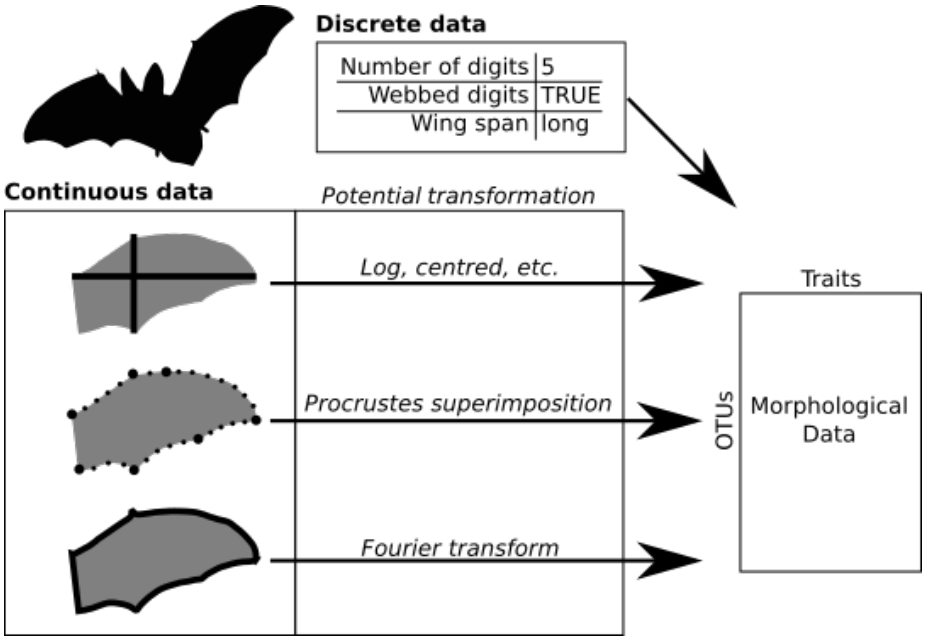


Figure 2.1: Types of morphological data - From Guillerme, Cooper, et al. (2020)

2.1 Morphological traits

2.1.1 Morphological traits in the age of molecular data?

In a recent paper I reviewed (hopefully to be published soon - it's going to be a super useful piece of work for the community!), the authors made a distinction between molecular characters being objective and morphological ones being subjective. The common idea is that you put some DNA in a machine and get some sequence of letters out of it (ACTG) which appears to be more objective than going into a museum and comparing specimens with your expert eyes. And here's my rant on why I disagree:

TL;DR: I think the comparison "molecular = objective" and "morphological = subjective" is a false comparison. This might be because of a dual meaning of these two words:

- Subjectivity and objectivity as a epistemoligical concept where something is subjective when it's link to a person's experience and something is objective when it's not. But both morphological and molecular characters are mostly objective in their description (i.e. the character is not linked to a person's experience) but mostly subjective in their use (i.e. the character's usage is linked to a person's experience)/
- But also in research we tend to associate these words to a moral judgment when referring to our work: objective = good and subjective = bad. Which again is both true and false for any character: we should use the method best adapted to a certain type of character (to our knowledge) so the character is always, in that sense *objectively* good (in a subjective sense; i.e. to the best of our knowledge).

Regarding the first meaning (epistemological): Although molecular characters are objective (there is a nucleotide "A" in position n on gene Y regardless of a person's experience), morphological characters can be equally objective (there are 4 cusps on this teeth; or in continuous terms: this bone is longer than another one). Furthermore, often authors are not only interested in the characters (molecular and morphological) but how they are used for a research purpose (e.g. building a tree). This comes with many arbitrary decisions in both cases (i.e. subjective choice - not bad choices, *cf* the second meaning of the words below). For example, workers building phylogenies based on molecular characters have to make a lot of arbitrary decisions on which genes to use based on time, cost and availability constraints. Maybe the comparison should be more along the lines of "molecular = easier to automatise" and "morphological = harder to automatise" (although both are probably not true when diving into it: this was true 20 years ago, less so nowadays).

Regarding the second meaning (moral judgment; objective = good, subjective = bad): Although continuous measurements appear to be more objective (*sensu* 1st meaning: "bone A is of length X") and this *better* for the aim of the worker (*sensu* 2nd meaning; "it's better to use continuous characters to build a morphological

tree" - see this one by the way: Parins-Fukuchi (2018)). It might not be the case when choosing another method (subjective; *sensu* 1st meaning) where discrete morphological characters are more subjective (*sensu* 1st meaning; an expert chose to spend time encoding character X rather than character Y) but it leads to *better* results depending on the method (*sensu* 2nd meaning; when using parsimony, trees are better when expert chose the characters - i.e. the trees end up being more useful to the community). This applies also to molecular characters when workers tend to use the GTR model with 4 categories of rates approximating a Gamma distribution (rarely another number). This can be equally seen as a subjective choice (1st meaning) leading to an objective improvement (2nd meaning).

2.1.2 Continuous trait data

This method is one of the easiest. Continuous data is often measured as standard physical units (e.g. meters, grams, seconds, etc.) and can be read directly from a data file in `R`. On easiest data file format is `csv` ("comma separated values") that is easily generated from most spreadsheet editor (usually they have an option "Save As... > .csv"). These files look something like this:

```
         ,variable 1, variable 2
species_a, 1.23, 4.56
species_b, 7.89, 0.12
```

and can be read in `R` using `read.csv`.

USE YOUR DATA: load your own continuous data

```
my_continuous_data <- read.csv("path_to_my_file.csv")
```

You can also use `read.csv(..., row.names = <column_number>` to directly read one column as the rownames:

```
my_continuous_data <- read.csv("path_to_my_file.csv", row.names = 1)
```

2.1.3 Discrete trait data

2.1.3.1 Glossary

There are many specific terms when coming to discrete morphological characters, here is a list of some that I use (though note that the definitions here are my own, there is no standard or correct one!). My version is heavily inspired by Brazeau (2011).

- **character**: this is a trait (e.g.: "snout length")
- **state**: this is one of the potential discrete value the state can have for a specific species (e.g. "snout length: short, long")
- **token**: this the symbol used to represent the state for a specific character (e.g. "snout length: short (0), long (1)")

- **ordered/unordered**: this designates whether the character is ordered (e.g. "snout length: short, medium, long") or whether it is unordered (e.g. "habitat: aquatic, arboreal, fossorial")
- **ambiguity**: this designates whether the state for a specific species is known or not (see possible list below):
- **missing**: this is when the ambiguity is unknown: e.g. the feature to describe the character is not available. This is often encoded with the token `"?"`.
- **inapplicable**: this is when the ambiguity is known but not relevant: e.g. the character "Wing colour" for a dog (it's not missing, it just doesn't exist). This is often encoded with the token `"-"`.
- **uncertain**: this is when the state is unknown but only between certain states. For example, for a character "snout length: short(0), medium(1), long(2)" for a fossil species with a damaged snout (but that doesn't look short), we could code it as either "medium OR long" (often using the tokens: `"1/2"`).
- **polymorphic**: this is when a state has multiple observed states. For example, for the same character as before, a species could have both mediums AND long snouts (depending on the specimens). It is then often encoded using the tokens `"1&2"`.

Discrete characters can be stored in a simple spreadsheet as for continuous traits:

```
         ,character 1, character 2
species_a, 0, ?
species_b, 0, 1
```

They can be input using the `read.csv` function as above. Easy.

But more commonly, discrete morphological characters are stored in `.nexus`/`.nex` files. These are standardised comparative methods files with meta data and a list of characters. They are very commonly used both for molecular and morphological data. You can recognise them easily, they always start on the first line with `#NEXUS` and can contain more or less a lot of metadata. I'm not going to go in the details here but most of the time you can read them easily with `ape::read.nexus.data` or `Claddis::read_nexus_matrix`.

USE YOUR DATA: load your own discrete data

```
discrete_traits <- read.nexus.data("path_to_my_file.nex")
```

Note that they can be a bit more complicated to input depending on how they were formated. Sometimes they require a bit of editing to avoid error messages. Here is a comprehensive list of things to look out for when importing discrete morphological characters:

- SPACES!

That's it. Basically all the problems come from extra spaces in the matrix chunk in the nexus file. To avoid any: only use spaces to separate the element name

(species, genera, specimen, etc.) and the string of characters. For example this is a good format:

```
species_name 0100010???{01}11(01)001
```

But all these are gonna create problems:

```
species name 0100010???{01}11(01)001
species_name 010001 0???{01}1 1(01)001
species_name 0100010???{0 1}11(0 1)001
```

2.2 Ordinations

Method	Variables	Distances	R functions
Principal component analyses (PCA)	Continuous	Euclidean (no need for a distance matrix)	`stats::prcomp`, `stats::princomp`
Principal coordinates analysis (PCO, PCoA), Multidimensional scaling (MDS), Non-metric MDS (NMDS)	Continuous, discrete, complex	Any distance (needs a distance matrix)	`stats::cmdscale`

(table adapted from Legendre and Legendre (2012)).

2.2.1 Principal components analyses (PCA)

SLIDES: PCA

CATCHING UP ZONE: here's some example of continuous data if you didn't import your own. This is a dataset of 24 continuous measurements from 60 observations (species of skinks) from Brennan et al. (2024) :

```
## Continuous data
continuous_data <- read.csv("../examples/continuous_characters/Brennan2024.csv",
                            row.names = 1)
```

To do a PCA, it's pretty simple, you can use the function `prcomp` or `princomp`. They are very different function in that they do the exact same things but their outputs have different names. To access the PCA matrix for `prcomp` you need to call `my_pca$x` and for `princomp` it's `my_pca$scores`. But they're fundamentally the same thing. Here's how to compute it as well as measuring the loadings, i.e. the amount of variance explained per dimensions:

```
## Ordinating some data with prcomp
my_ordination <- prcomp(continuous_data)
## Getting the trait space
my_traitspace <- my_ordination$x
## Measuring the "loading" (the variance per dimension)
my_loadings <- apply(my_traitspace, 2, var)/sum(apply(my_traitspace, 2, var))
```

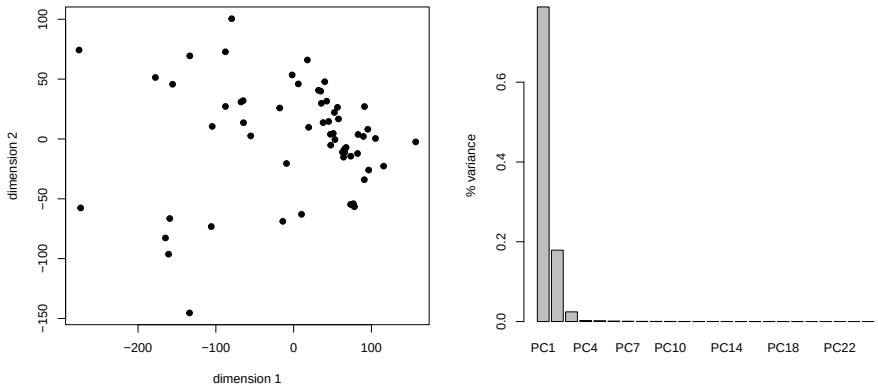
Alternatively you can calculate the loadings by using the `$sdev` component that is the square root of the eigenvalues (so the loadings are the scaled squared `$sdev` component) but personally I prefer calculating it using the variance (I find it easier to understand).

```
## Alternatively variance loading calculation
my_ordination$sdev^2/sum(my_ordination$sdev^2)
```

```
##  [1] 7.885555e-01 1.789772e-01 2.427367e-02 2.471291e-03 2.038708e-03
##  [6] 1.161455e-03 7.586403e-04 4.199369e-04 3.708986e-04 2.988461e-04
## [11] 1.917230e-04 1.296428e-04 9.967757e-05 7.058466e-05 5.630608e-05
## [16] 3.549546e-05 3.026064e-05 2.576774e-05 1.410035e-05 1.252596e-05
## [21] 7.728232e-06 6.153455e-32 1.940995e-32 1.770047e-33
```

Visualising the ordination of the data:

```
op <- par(mfrow = c(1,2))
## Plotting the original data
plot(my_traitspace[, c(1,2)],
     xlab = "dimension 1", ylab = "dimension 2", pch = 19)
## Adding another plot with the loadings
barplot(my_loadings, ylab = "% variance")
```




```
par(op)
```

2.2.2 Principal coordinates analysis (PCO)

CATCHING UP ZONE: here's some example of discrete data if you didn't import your own. This is a dataset from Beck and Lee (2014) of 106 mammal species (mainly fossils) and 421 characters.

```
## 421 discrete morphological characters
discrete_traits <- read.nexus.data("../examples/discrete_characters/Beck2014.nex")
## Combine the output list into a matrix (rows are species and columns are characters)
traits_matrix <- do.call(rbind, discrete_traits)
```

PCOs are also pretty easy but unlike PCAs they require one extra step before ordinating the variance-covariance: calculating the distance matrix.

2.2.2.1 Distance matrices

SLIDES: PCO

There are many ways to calculate distance matrices from discrete morphological characters. See this very good review for more information: Lloyd (2016). In short the type of distance metric should depend on the quality of your data (amount of missing or uncertain data) and the type of characters (binary, ordered or not, etc.).

The most comprehensive way to measure distance matrices can be done by using the `Claddis::calculate_morphological_distances` which allows for many many options to tailor the distance metric (and special behaviours) to your own dataset.

However, here we will be using the more generic `dispRity::char.diff` function that is faster but is designed for a more general purpose so lacks some of the

options from `Claddis`.

```
## Calculating the maximum observable distance
distance_matrix <- char.diff(traits_matrix, method = "mord", by.col = FALSE)
```

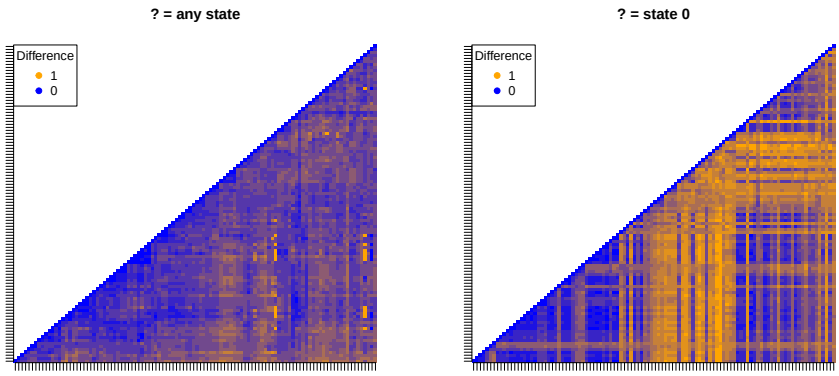
Note that here we are going all by default which is useful to get everyone on board for a workshop but is definitely not what you should do for your research! Always have a look at the default hidden options and what they're doing!

For example in this instance with the data from Beck and Lee (2014) we treated all the characters as unordered (you can change that with the option `order = c(TRUE, FALSE, TRUE, ...)` for every character if they are to be treated as ordered (`TRUE`) or not (`FALSE`)). Also, we have decided to treat encoded ambiguities in a standard default way where question marks (`"?"`) are treated as missing values and interpreted as all possible character states and `"\"` designated ambiguous characters and where treated as any ambiguous state marked here (e.g. `"0\2"` is either state `0` or `2` - but not any other state, e.g. `1`). You can personalise the way you want to interpret these characters by giving a different list of token definition and

```
## Calculating the maximum observable distance
distance_matrix2 <- char.diff(traits_matrix, method = "mord", by.col = FALSE,
                              # defining the special token and it's name
                              special.tokens = c("missing" = "\\?"),
                              # defining the behaviour (always returning 1, the first
                              special.behaviour = list(missing = function(x,y) return(
                              )
```

This change results in a very different distance matrix (and is probably not very pertinent for this specific dataset!).

```
op <- par(mfrow = c(1,2))
## Plotting the original data
plot(distance_matrix, main = "? = any state")
## Adding another plot with the loadings
plot(distance_matrix2, main = "? = state 0")
```

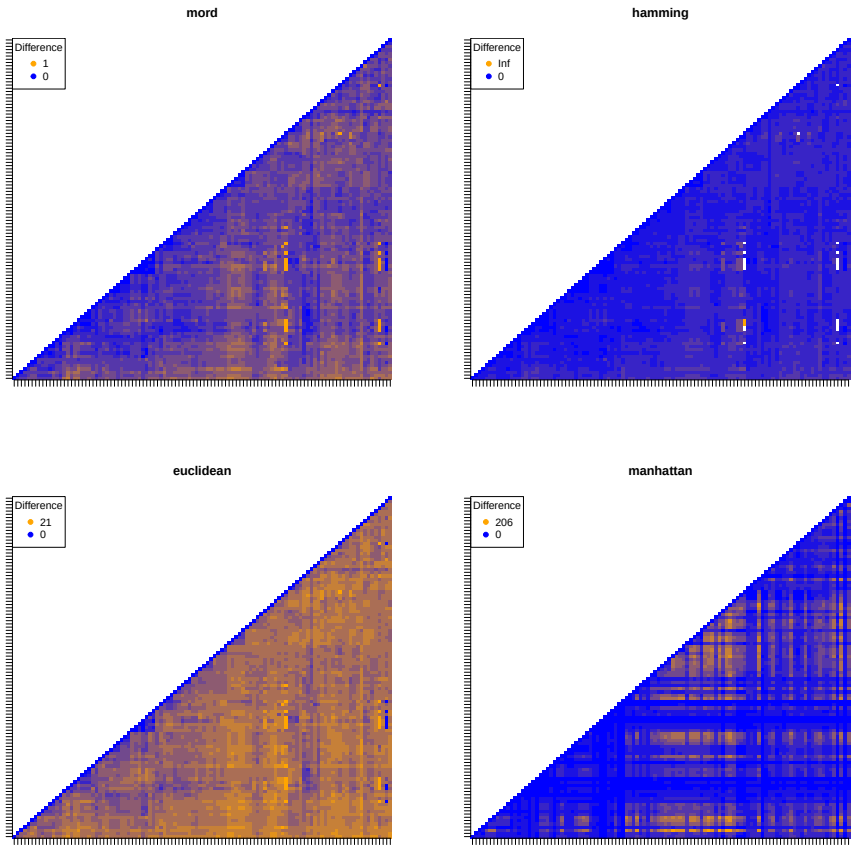



```
par(op)
```

TINKER TIMES: You can also play around with the different type of distance metrics and see how that affects the results by changing the option `method`. Have a read in the manual for the details (`?char.diff`).

[**Click to expand the solution**]:

```
## Four different metrics
distance_mord      <- char.diff(traits_matrix, method = "mord", by.col = FALSE)
distance_hamming   <- char.diff(traits_matrix, method = "hamming", by.col = FALSE)
distance_euclidean <- char.diff(traits_matrix, method = "euclidean", by.col = FALSE)
distance_manhattan <- char.diff(traits_matrix, method = "manhattan", by.col = FALSE)

## Plotting the results
op <- par(mfrow = c(2,2))
plot(distance_mord, main = "mord")
plot(distance_hamming, main = "hamming")
plot(distance_euclidean, main = "euclidean")
plot(distance_manhattan, main = "manhattan")
```




```
par(op)
```

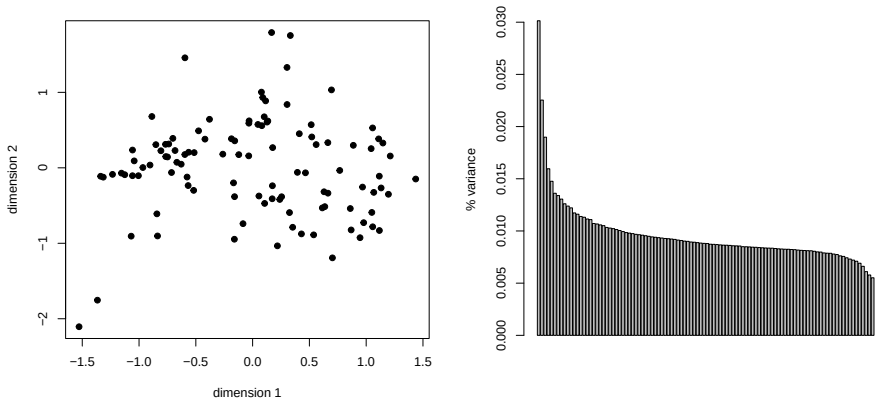
2.2.2.2 PCO

Once you have a distance matrix, you can ordinated it using the `cmdscale` function which works similarly to a Principal Components Analysis. One minor subtlety though is that you have to choose the number of dimensions (`k`) and might or might not use a correction to treat the distance matrix as Euclidean or not. Regarding the number of dimensions, a good practice is to use the number of species minus 2 to allow for the biggest spread of the variance (i.e. to avoid cramming the variance in the first dimensions). Regarding the correction, if you don't use a simple Euclidean distance metric, use the option `add = TRUE` to use the Cailliez (1983) correction (which adds an estimated constant to make the distances Euclidean).

```
## Ordinating a distance matrix
my_pco <- cmdscale(distance_matrix,
                   # the number of dimensions
                   k = ncol(distance_matrix)-2,
                   # the Cailliez correction
                   add = TRUE)

## Getting just the coordinates
my_traitspace <- my_pco$points
my_loadings <- apply(my_traitspace, 2, var)/sum(apply(my_traitspace, 2, var))
```

And we can visualise the first two dimensions and the distribution of the variance per dimensions:

```
op <- par(mfrow = c(1,2))
## Plotting the original data
plot(my_traitspace[, c(1,2)],
     xlab = "dimension 1", ylab = "dimension 2", pch = 19)
## Adding another plot with the loadings
barplot(my_loadings, ylab = "% variance")
```




```
par(op)
```

2.3 Estimating ancestral traits

2.3.1 Reading trees into R

Reading phylogenetic trees in R is rather easy: you can use either the `ape::read.tree` function for reading a tree in newick format (a text file starting with "(") or `ape::read.nexus` for reading a nexus tree (a text file starting with "#NEXUS").

`read.tree` or `read.nexus`? Can't decide which one to use? You can use this custom made function that will always decide for you:

```
read.any.tree <- function(file) {
    ## Check whether the first line of the file is "#NEXUS"
    if(scan(what = "#NEXUS", file = file, nlines = 1, quiet = TRUE) == "#NEXUS") {
        ## The tree is a true nexus
        return(read.nexus(file))
    } else {
        ## The tree is a true newick
        return(read.tree(file))
    }
}
```

```
## Reading a newick tree
my_tree <- read.tree("path_to_my_file.newick")
## Reading a nexus tree
my_tree <- read.nexus("path_to_my_file.nexus")
## Reading any tree
my_tree <- read.any.tree("path_to_my_file.dontknow")
```

CATCHING UP ZONE: here's some example of discrete and continuous data and tree if you didn't import your own.

This first chunk is a continuous dataset of 24 measurements from 60 observations (species of skinks) from Brennan et al. (2024) :

```
## Get the continuous
continuous_data <- read.csv("../examples/continuous_characters/Brennan2024.csv",
                            row.names = 1)
## Get the tree
continuous_tree <- read.tree("../examples/trees/Brennan2024.tre")

## Adding node labels
continuous_tree <- makeNodeLabel(continuous_tree)

## Cleaning both the tree and the data to match
cleaned_data    <- clean.data(data = continuous_data, tree = continuous_tree)
continuous_data <- cleaned_data$data
continuous_tree  <- cleaned_data$tree
## The following were dropped:
cleaned_data$dropped_rows
```

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

```
cleaned_data$dropped_tips
```

```
## [1] "Egernia_s.badia"
```

This second chunk is a discrete dataset from Beck and Lee (2014) of 106 mammal species (mainly fossils) and 421 characters and the associated tree.

```
## Get discrete data
discrete_traits <- read.nexus.data("../examples/discrete_characters/Beck2014.nex")
## Combine the output list into a matrix (rows are species and columns are characters)
discrete_data <- do.call(rbind, discrete_traits)
## Get the tree
discrete_tree <- read.nexus("../examples/trees/Beck2014.tre")

## Cleaning both the tree and the data to match
cleaned_data  <- clean.data(data = discrete_data, tree = discrete_tree)
discrete_data <- cleaned_data$data
discrete_tree <- cleaned_data$tree
## The following were dropped:
cleaned_data$dropped_rows
```

```
## [1] "Montanalestes"    "Lainodon"         "Kharmerungulatum" "Alymlestes"
```

```
cleaned_data$dropped_tips
```

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

2.3.2 Estimating ancestral traits

There are many ways to estimate ancestral trait data in which I will not go into to much details. I will not go into the details on *how* ancestral traits are calculated because 1) that would be a whole course on likelihood estimation and model fitting and 2) all that would just be to calculate "glorified" averages. So for this reason I'm just going to focus on on way to do it but note that there are many ways to do it and many options to consider. My favourites are:

- `ape::ace` for estimating ancestral traits for both continuous and discrete traits.
- `dispRity::multi.ace` an updated version of `ape::ace` dealing with continuous characters and discrete ones (but with more options) and that can handle tree distributions (we will look at this one here).
- `Claddis::estimate_ancestral_states` which is more specialized in estimating ancestral states for discrete characters. It noticeably has cool models for dealing with inapplicable data!

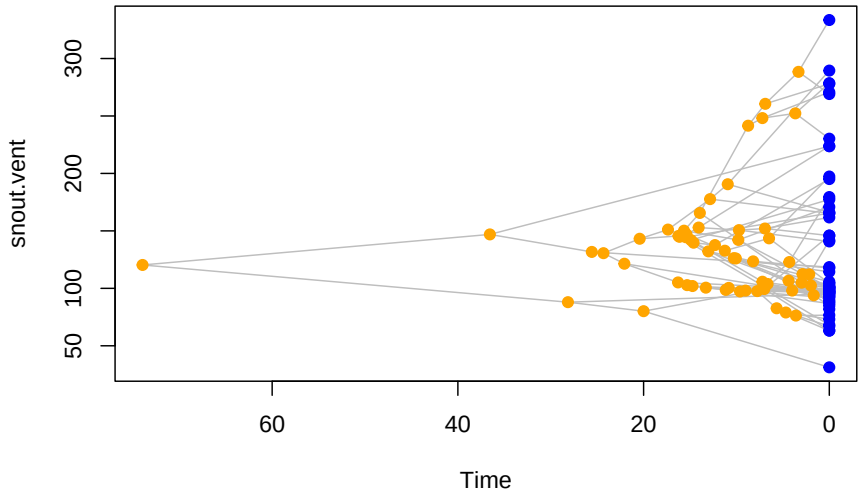
  Note that I have strong personal views on estimating ancestral traits: it is a useful tool for running some analyses (as we will see in next chapters) but it remains a very an impossible task, no matter how good the model and the data: unless we observe ancestral data (e.g. in the fossil record) we can only have a more or less vague idea of what the ancestral trait value could have been. If you use such methods for your research, I highly suggest you don't use point estimates

(i.e. single values) but rather distributions: both for the estimate and for the tree topology used for the estimation.

```
## Selecting a specific trait:
snout_vent <- continuous_data[,1]
names(snout_vent) <- rownames(continuous_data)

## Estimating the ancestral states for this trait
ancestral_snout_vent <- ace(snout_vent, continuous_tree)
```

```
## Warning in sqrt(1/out$hessian): NaNs produced
```

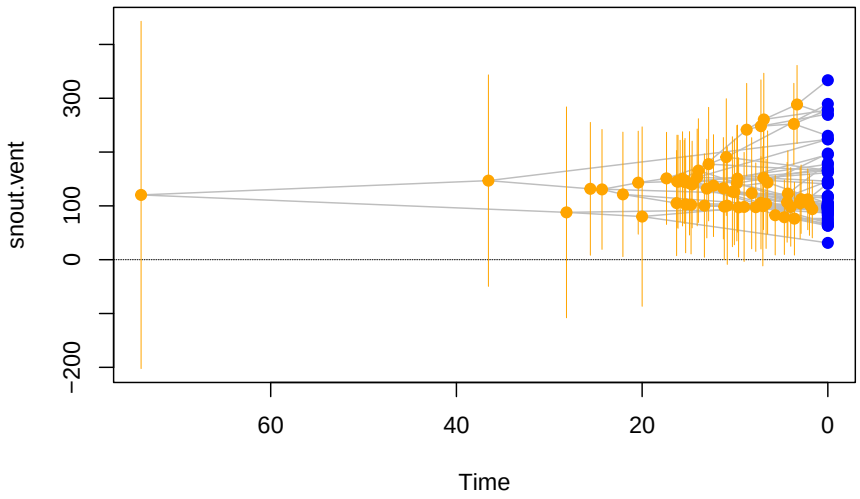
```
## Combining the tips and the ancestral values
ancestral_values <- ancestral_snout_vent$ace
snout_vent_data <- as.matrix(data.frame("snout vent" = c(snout_vent, ancestral_values)))

## Plotting the data (using the treats package)
treats_data <- make.treats(continuous_tree, data = snout_vent_data)
plot(treats_data)
```



This look all good and realistic (i.e. the ancestor of this group has a trait value that is a scaled average of the observed species). However, if we have a look at the errors around the estimations, we have a whole different story!

```
## Plotting the trait data
plot(treats_data, ylim = range(ancestral_snout_vent$CI95))
abline(h = 0, lty = 2, lwd = 0.2)
```

```
## Get the node ages
node_ages <- tree.age(continuous_tree)$age[-c(1:Ntip(continuous_tree))]
## Get the 95% confidence interval for each node estiamte
confidence_intervals <- ancestral_snout_vent$CI95
## Plotting each confidence interval
for(one_node in 1:Nnode(continuous_tree)) {
  lines(x = rep(node_ages[one_node], 2), y = confidence_intervals[one_node,], col = "orange", lwd
}
```



So that's not that great!

2.3.2.1 How to improve this mess?

- Be cautious about ancestral state estimations: they are just that (estimations! - not *reconstructions*).
- Whenever possible, use multiple trees to account for topological error.
- Whenever possible, use the distribution of estimates rather than point estimates.
- Whenever possible, use multiple traits rather than a single one.

But for now we can just go ahead and keep these caveats in mind. You can use the `dispRity::multi.ace` function to estimate ancestral trait values for large datasets and use tree distributions:

```
## Running the ancestral states for all the traits (it should take a couple of seconds)
ancestral_states <- multi.ace(continuous_data, continuous_tree)
```

```
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
## Warning in sqrt(1/out$hessian): NaNs produced
```

With the same caveats in mind, we can do the same for discrete traits using still `multi.ace`. Note that some of the syntax here is the same as we've seen for `char.diff`: there is a lot of room for personalisation to tailor the token interpretations and behaviour, the models, etc... Here we will use the option `output = "combined"` to get a combined matrix (tips + nodes) for ease of use.

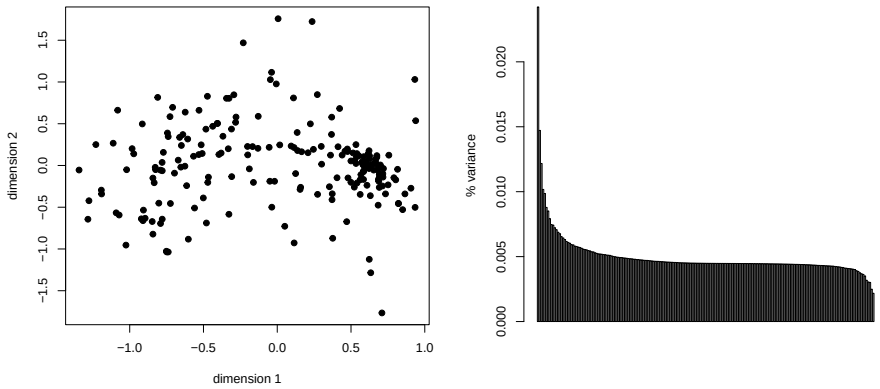
```
## Estimating ancestral traits for discrete characters characters
ancestral_states <- multi.ace(discrete_data, discrete_tree, output = "combined")
```

We can then use this new matrix with ancestral states to re-calculate distances (the same as before but including the nodes).

TINKER TIMES: see if you remember how to do it by just copy/pasting chunks from above.

[**Click to expand the solution**]:

```
## Calculating the distance matrix
distance_matrix <- char.diff(ancestral_states, method = "mord", by.col = FALSE)

## Ordinating the data
my_pco <- cmdscale(distance_matrix,
                   # the number of dimensions
                   k = ncol(distance_matrix)-2,
                   # the Cailliez correction
                   add = TRUE)

## Getting just the coordinates
my_traitspace <- my_pco$points
my_loadings <- apply(my_traitspace, 2, var)/sum(apply(my_traitspace, 2, var))

op <- par(mfrow = c(1,2))
## Plotting the original data
plot(my_traitspace[, c(1,2)],
     xlab = "dimension 1", ylab = "dimension 2", pch = 19)
## Adding another plot with the loadings
barplot(my_loadings, ylab = "% variance")
```




```
par(op)
```

2.4 References

Chapter 3

From traitspaces to disparity

Once you have created your traitspace, you can then measure disparity. But was is disparity? It depends.

There are many different definitions, some are more narrow that others but all agree on the same vague idea that disparity is a statistic that summaries a group's space occupancy. Basically a measurement of how elements are distributed in space based on their characteristics (e.g. traits) and very often used to compare groups between each other based on that measurement. For example is disparity higher in group X or Y? It can be often easily opposed to "taxonomic diversity" (i.e. number of species). Basically for this workshop we will define disparity as the diversity of traits combinations which can tell us something about the biology of a system.

This definition is not universal among palaeobiologists: for example, some colleagues use a more restricted definition of the term (e.g. in Hopkins and Gerber (2021) - not OA; sneaky download from here - I think more inspired by historical definitions). I use the more broad defintion that we discuss here Guillerme, Cooper, et al. (2020) (maybe more inspired by ecology).

> Note that in palaeobiology we call this statistic "disparity", "morphological disparity" or "trait disparity" but they are all the same concept. Even more noteworthy, this is a concept that is also routinely used in ecology but often called "trait dissimilarity" or "functional diversity". There is a very rich literature on it that I highly recommend to read to learn more about how this concept has been studied globally. Check out Petchey and Gaston (2006) and Mammola et al. (2021) as great starting points.

3.1 Measuring disparity

From this broad definition of disparity, there is many many ways to measure it. With some colleagues, we strongly argue that it should always be measured in regards to a biological question: i.e. measure it the way it will answer *your specific question*, not just the way other people have done it (Guillerme, Puttick, et al. (2020), Mammola et al. (2021), Guillerme, Cardoso, et al. (2024)), But we will see in more details how to do that later.

Let's use a super simple case where we want to measure the standard deviation in the traitspace as our disparity metric[1]. This could be used for answering a simple question "how spread is the trait combination in our dataset".

CATCHING UP ZONE: here's an example of a traitspace made from continuous data. But please, feel free to use your own!

```
## An inbuilt ordination directly from the dispRity package
data(BeckLee_mat50)
my_traitspace <- BeckLee_mat50
```

```
## The basics for measuring standard deviation
sd(my_traitspace)
```

```
## [1] 0.2268704
```

One other way to achieve the exact same results is through the `dispRity` function from the eponymous package. This is a bit pointless and cumbersome for now but it will become more obviously useful later on.

```
## Same but using the dispRity package
my_dispRity <- dispRity(my_traitspace, metric = sd)
summary(my_dispRity)
```

```
##   subsets  n   obs
## 1       1 50 0.227
```

So that's it! You've now measured disparity. Well done. We will talk more about what to measure in a bit but first I want to chat about... ***The multidimensional nightmare***

3.2 The curse of dimensionality

SLIDES: Dimensionality

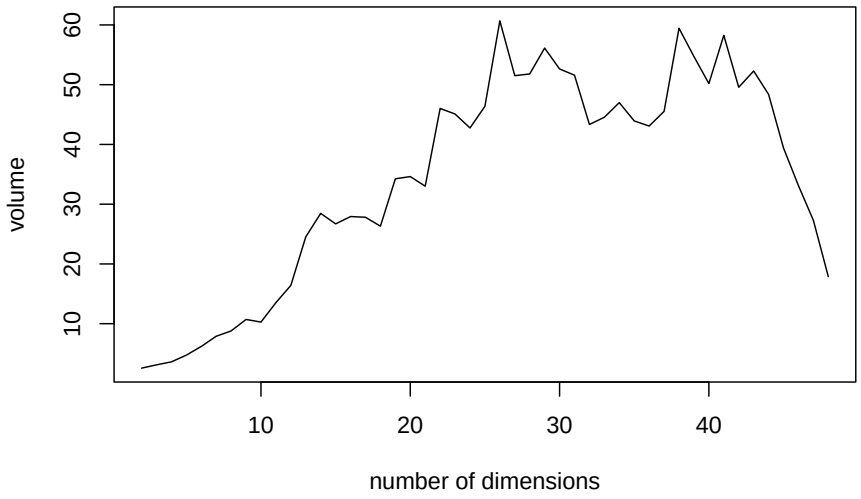
[1] **Disparity metric or disparity index?**. You might encounter both terms which designate the same thing: a disparity statistic. Some people prefer using the term "index" because a "metric" has a specific definition in mathematics (i.e. not all indices or statistics are metrics). Some other people prefer using the term "metric" (e.g. myself) because the term "index" has a specific definition in informatics (i.e. an index is a numbered element in a list). Tomato tomato.

Basically this is a problem you get very easily when you go beyond anything three dimensional. When increasing the number of dimensions in your dataset things become very quickly intuitive and headache inducing, even after working on it for many years. Depending on how your brain works, you will have to take some leap of faith here and assume that what you see and understand is not always correct when you pile up a higher number of dimensions!

TINKER TIMES: have a think of how the volume of your traitspace will change with the number of dimensions.

[**Click to visualise what your data is doing**]:

```
## Measuring the volume for each additional dimensions
## A placeholder for the loop results
volumes <- numeric()

for(dimension in 2:ncol(my_traitspace)) {
  ## Calculating for each number of column the product of ranges
  new_volume <- prod( # the product
    apply(my_traitspace[, 1:dimension], 2, # apply a function for the selected columns (2)
      function(x) diff(range(x)))) ## calculate the difference between the range of the column (
  ## Add it to our volume placeholder
  volumes <- c(volumes, new_volume)
}

## Plot the volumes per number of dimensions
plot(x = 2:ncol(my_traitspace),
     y = volumes,
     type = "l",
     xlab = "number of dimensions",
     ylab = "volume")
```



Congratulations, you now know about the curse of multidimensionality! We can see this excellent illustration by Toph Tucker to have a grasp of what is happening here.

3.2.1 Reducing the number of dimensions

CATCHING UP ZONE: loading some example data!

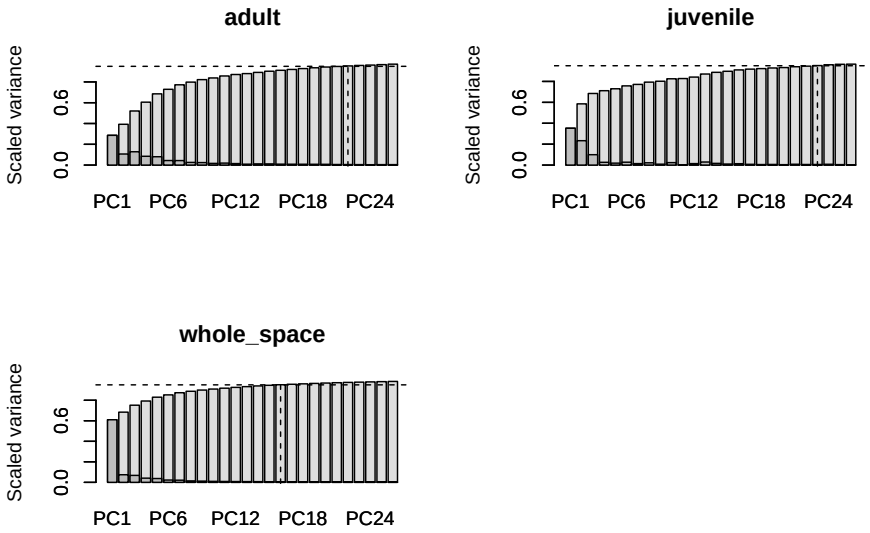
```
## Loading some example data
data(demo_data)
traitspace <- demo_data$hopkins
```

For very large datasets, it can be useful to reduce the number of dimensions to avoid the curse as much as possible. This is especially the case if you use a PCA to build your traitspace. Because it ordinates your data per ranking variance, you end up with most variance in the first couple of dimensions. Often, you can reduce the number of dimensions so you have a cut-off of the n first dimensions that include X percentage of the variance in your traitspace. Usually this X is 95% or 99%.

One thing to keep in mind though when you do this is that if you have groups in your traitspace (we will see how to do this later), you might want to check whether they are all equally distributed along those axes. For example, a reduced traitspace with 95% of the total variance does not automatically means a traitspace with at least 95% of the variance for each group.

SLIDES: The duck axis

```
## How many axes are needed
axes_selection <- select.axes(traitspace, threshold = 0.95)
plot(select.axes(demo_data$hopkins))
```



3.3 Which disparity metric to use?

So which disparity metric should you be using? It's very easy: ***IT DEPENDS***. In the next section we will be looking in more details what kind of analyses we can do with disparity and what kind of questions we can ask with such approach. But basically, it's all about tailoring the disparity metric to your own specific question.

We have done some work on which metrics to choose with colleagues (Guillerme, Puttick, et al. (2020), Mammola et al. (2021) - but other great one exist e.g. Ciampaglio, Kemp, and McShea (2001), Petchey and Gaston (2006), Stewart et al. (2023)). Usually, most of these consist in comparing the effect of different metrics at capturing the correct pattern or their sensitivity to missing data. This is very important of course but I think it's lacking a didactic aspect that would help people with no idea what to measure on how to measure. We discuss this in this preprint Guillerme, Cardoso, et al. (2024).

Basically we believe that for choosing a metric, rather than starting with a metric's property, it might be easier to start with the question at hand. You can separate part of your statistical research into three components:

- The *mechanism*: that is the biological question at hand and usually has the word *"why"* in it. For example: "why do mass extinction change the disparity of the groups that survive the mass extinction?". The *why* here is the *mechanism* by which disparity can change in groups due to mass extinctions (i.e. some biology).
- The *process*: this is the dynamic change that makes your question interesting and usually has the word *"how"* in it. For example: "how does disparity change before and after the mass extinction?". The *how* here is the *process* by which disparity changes through time. This is not especially a biological question (although it ultimately will be) because you can just ask yourself *how* without asking *why* (even though that might make it harder to get your paper accepted or your project funded ;)).
- The *pattern*: this is the actual change involved in the process and usually has the work *"what"* in it. For example what is changing before and after the extinction event? This is effectively the statistic that will allow you to look at the process and understand the mechanism. Basically it's the statistic, value or... disparity!

So once you have your mechanism and process in mind you can choose your metric.

SLIDES: What? How? Why?

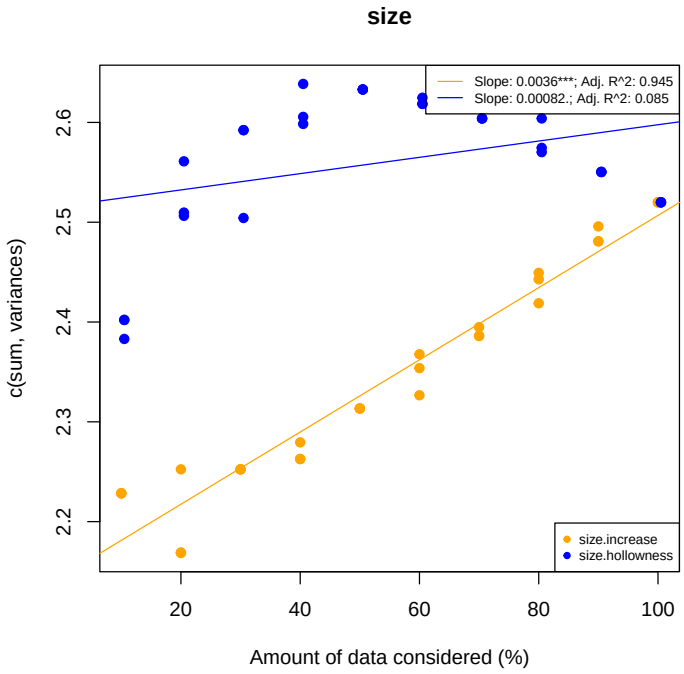
3.3.1 Choosing and testing disparity metrics

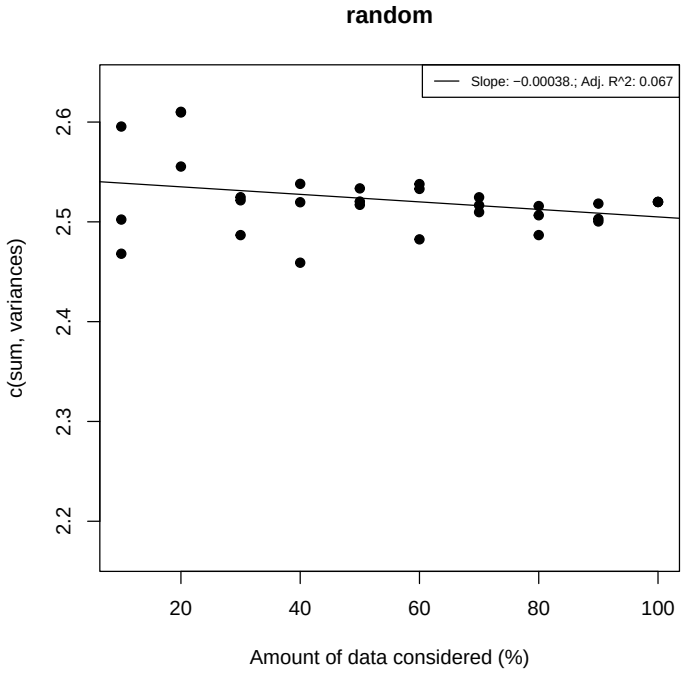
SLIDES: Types of metrics

You can use the `dispRity` and `moms` pipeline to test your metric.

`moms`.

```
## Testing the sum of variance
my_test <- test.metric(my_traitspace, metric = c(sum,variances), shifts = c("size", "ra
plot(my_test)
```


size
Slope: 0.0036***; Adj. R^2: 0.945
Slope: 0.00082.; Adj. R^2: 0.085
size.increase
size.hollowness
c(sum, variances)
2.2
2.3
2.4
2.5
2.6
20
40
60
80
100
Amount of data considered (%)


random
Slope: −0.00038.; Adj. R^2: 0.067
c(sum, variances)
2.2
2.3
2.4
2.5
2.6
20
40
60
80
100
Amount of data considered (%)

TINKER TIMES: Have a go and test multiple metrics and multiple processes to see which one would work better!

[**Click to expand the solution**]:

```
## Does the sum of variance captures changes in size or in position
my_test1 <- test.metric(my_traitspace, metric = c(sum,variances), shifts = c("size", "
## What about the sum of ranges?
my_test2 <- test.metric(my_traitspace, metric = c(sum,ranges), shifts = c("size", "posi
```

3.4 References

```
## Loading required package: phytools
```

```
## Loading required package: maps
```

Chapter 4

From disparity to macroevolution

CATCHING UP ZONE: here's some example discrete dataset from Beck and Lee (2014) with ancestral states estimations and

[**Click to expand the data loading part**]:

```
## Get discrete data
discrete_traits <- read.nexus.data("../examples/discrete_characters/Beck2014.nex")
## Combine the output list into a matrix (rows are species and columns are characters)
discrete_data <- do.call(rbind, discrete_traits)
## Get the tree
discrete_tree <- read.nexus("../examples/trees/Beck2014.tre")

## Cleaning both the tree and the data to match
cleaned_data  <- clean.data(data = discrete_data, tree = discrete_tree)
discrete_data <- cleaned_data$data
discrete_tree <- cleaned_data$tree
## The following were dropped:
cleaned_data$dropped_rows
```

```
## [1] "Montanalestes"    "Lainodon"         "Kharmerungulatum" "Alymlestes"
```

```
cleaned_data$dropped_tips
```

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

```
## Adding node labels
discrete_tree <- makeNodeLabel(discrete_tree, prefix = "n")
## Adding the root time
discrete_tree$root.time <- max(tree.age(discrete_tree)$age)
```

```r
## Estimating ancestral states
ancestral_states <- multi.ace(discrete_data, discrete_tree, output = "combined")

# Calculating the distance matrix
distance_matrix <- char.diff(ancestral_states, method = "mord", by.col = FALSE)

## Ordinating the data
my_pco <- cmdscale(distance_matrix,
                   # the number of dimensions
                   k = ncol(distance_matrix)-2,
                   # the Cailliez correction
                   add = TRUE)

## Getting just the coordinates
my_traitspace <- my_pco$points
```

Once we have our question at hand (and an idea of how to measure disparity) we can finally start doing some cool stuff and use disparity to tell macroevolutionary (or ecological!) stories.

4.1 Testing hypothesis with disparity data

One first aspect for these type of analyses, is to split the data in a meaningful way that will illustrate the process in question. We can split it through time (we'll cover that very soon) or by groups using the `dispRity::custom.subsets` function.

4.1.1 Making subsets

This function is relatively easy to use, you can just provide your trait space and a specific way to group you data. For example that could be two different clades or some lists of species with specific characteristics (e.g. diet).

USE YOUR DATA: set up your groups based on your own data.

Either load a data frame (.csv) with different categories, for example:

```
  , diet, group
species_a, omnivore, group A
species_b, omnivore, group B
species_c, carnivore, group A
```

```r
## Read your favourite csv file (you'll need to set up the path yourself!)
my_data_groups <- read.csv("path_to_my_super_serious_dataset.csv")
```

or else you can just create your list manually

```
## Making a list of different groups
my_data_groups <- list("omnivores"  = c("species_a", "species_b", ...),
                       "carnivores" = c("species_c", ...))
```

Here we will just group the species from the example dataset into crown and stem mammals using the `dispRity::crown.stem` function:

CATCHING UP ZONE: Creating two groups from the Beck and Lee (2014) dataset: the crown and the stem mammals

```
## Crown and stem mammals
my_data_groups <- crown.stem(tree = discrete_tree, inc.nodes = TRUE)
```

And here's how we can create the groups into a `"dispRity"` object:

```
## Grouping the data into different subsets
my_groups <- custom.subsets(my_traitspace, group = my_data_groups)
```

Note from here on we will be heavily relying on `"dispRity"` objects. They are a specific `R` class of objects (S3 - for those that know what that is more nerding here) that are attached to the `dispRity` package. It allows for easier visualisation (plotting, printing, summarising) and neater easier to reproduce analysis: each step is configured by you and is sharable and savable.

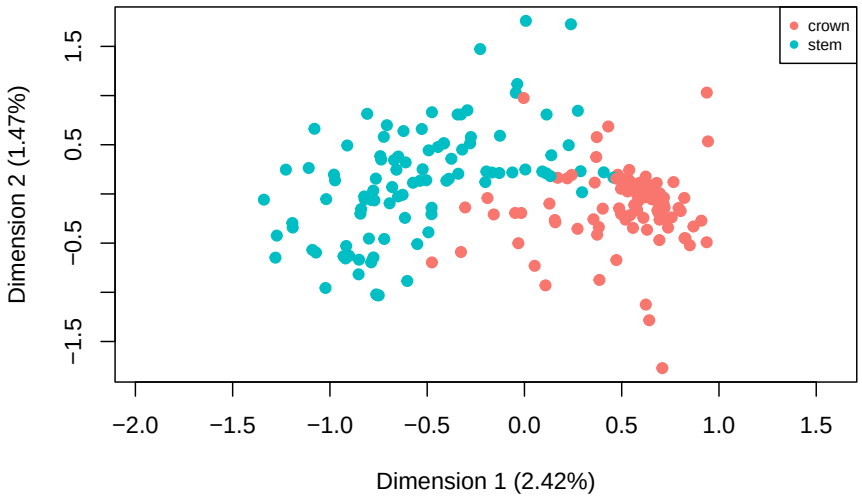
```
## What is the class of my_groups?
class(my_groups)
```

```
## [1] "dispRity"
```

```
## What's in it?
my_groups
```

```
##  ---- dispRity object ----
## 2 customised subsets for 203 elements in one matrix:
##     crown, stem.
```

```
## What does it look like in 2D
plot(my_groups)
```



Once we have this groups sorted, we can easily measure disparity:

```
## The sum of variance of our groups
my_disparity <- dispRity(my_groups, metric = c(sum, variances))
my_disparity
```

```
##  ---- dispRity object ----
## 2 customised subsets for 203 elements in one matrix with 201 dimensions:
##     crown, stem.
## Disparity was calculated as: c(sum, variances).
```

```
## What's the difference in variance?
summary(my_disparity)
```

```
##   subsets   n   obs
## 1   crown 103 16.50
## 2    stem  99 16.37
```

We can compare the values of disparity we have between our two groups and tell a story about it. For example, with the example dataset we have a slight increase in variance in the crown mammals compared to the stem ones indicating maybe a increase in trait space occupancy through the phylogeny? But that's very anecdotal and needs more convincing. We probably want to do some statistical test!

> A personal note on statistical testing for statophobes: we are now entering the zone when there's going to be p-values and other statistical bits that can be a bit scary. But fear not: although statistics

(and researchers) often present this as an objective measure of the truth that has to be rigorous and final, I see it much more as a methodical way to convince our colleagues that our hypothesis is justifiable. I think that statistics are not the truth but a very rigorous and shareable way to explain what is happening in the natural world. A great paper to this effect is Dushoff, Kain, and Bolker (2019) that neatly suggests changing the word "significant" (i.e. a magical black box word that makes your research correct or not) into the more commonly english word "clearly" (i.e. a normal word that does call any magic: "disparity in group A is clearly/not clearly different than in group B"). Hopefully this approach will make you more relaxed about the following session and just see it as a *rigorously subjective* way to convince your comrades rather than some *objective* dark magic! Note that note all my comrades (e.g. Muff et al. (2022)) share this view of statistics though ;).

4.1.2 Bootstrapping and rarefying

One way to check whether these two samples are truly different and not only different by chance is to bootstrap the data. Bootstrapping is an easy way to look at the quality of your data (Efron and Tibshirani (1994) [^3]). Basically it consists of randomly resampling your data (with replacement) a certain number of times. Effectively this means randomly increasing the weight of some of your data points (the ones resampled) and reducing the weight of some other (the ones not resampled). [^3] If you're ever struggling with stats and R, they made an excellent YouTube playlist of 68 short stats courses. Do one per day after lunch and you'll be a stats expert in 3.5 months!

```
## Bootstrapping the data
my_disparity <- dispRity(boot.matrix(my_groups),
                         metric = c(sum, variances))
my_disparity
```

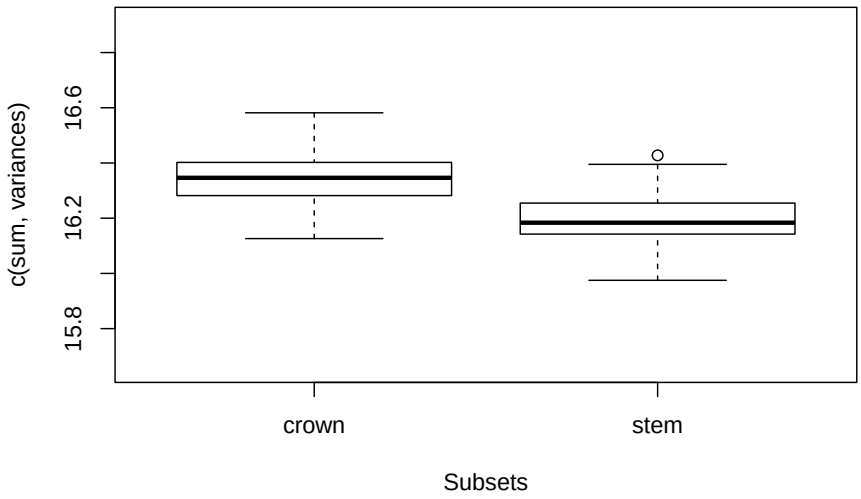
```
##  ---- dispRity object ----
## 2 customised subsets for 203 elements in one matrix with 201 dimensions:
##     crown, stem.
## Data was bootstrapped 100 times (method:"full").
## Disparity was calculated as: c(sum, variances).
```

```
## What the difference in variance after bootstrapping?
summary(my_disparity)
```

```
##   subsets   n   obs bs.median  2.5%   25%   75% 97.5%
## 1   crown 103 16.50     16.35 16.13 16.28 16.40 16.52
## 2    stem  99 16.37     16.18 16.04 16.14 16.25 16.35
```

With the example dataset we see again a very small difference but with small-ish overlap in the confidence intervals:

```
## Visualising the difference in distribution
plot(my_disparity)
```



This difference though might be due to the difference in number of species (more species in the crown group means more disparity?). You can check that by *rarefying* your data. This is a variant of the bootstrapping procedure but instead of resampling the same number of elements, you can set a specific number of elements to resample (or several numbers):

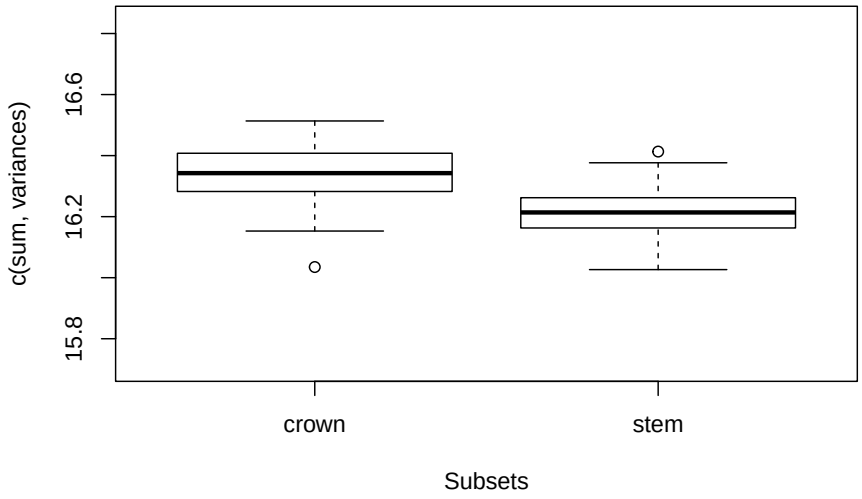
```
## Get the size of the smallest group@
n_small <- min(size.subsets(my_groups))

## Bootstrapping the data
my_disparity <- dispRity(boot.matrix(my_groups, rarefaction = n_small),
                         metric = c(sum, variances))
my_disparity
```

```
##  ---- dispRity object ----
## 2 customised subsets for 203 elements in one matrix with 201 dimensions:
##     crown, stem.
## Data was bootstrapped 100 times (method:"full") and rarefied to 99 elements.
## Disparity was calculated as: c(sum, variances).
```

```
## What the difference in variance after bootstrapping?
summary(my_disparity)
```

```
##   subsets   n   obs bs.median  2.5%   25%   75% 97.5%
## 1   crown 103 16.50     16.34 16.16 16.28 16.41 16.51
## 2   crown  99    NA     16.33 16.15 16.28 16.41 16.50
## 3    stem  99 16.37     16.21 16.06 16.16 16.26 16.36
```

```
## Visualising the difference in distribution
plot(my_disparity)
```



Here we can see it doesn't make a big difference.

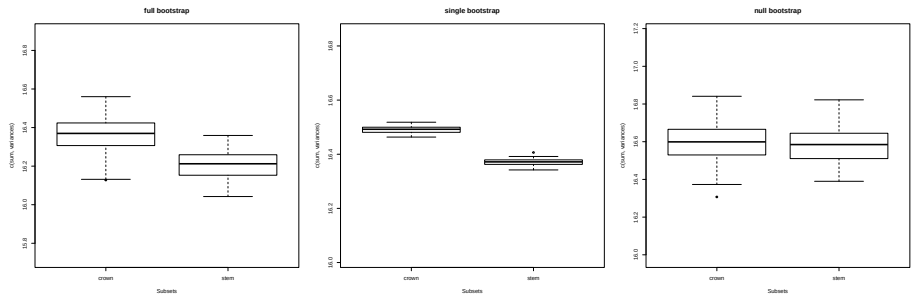
TINKER TIMES: here we just looked at the `"full"` bootstrapping algorithm (that resamples every elements or every number of elements with the `rarefaction` option). You can also change it to the more conservative `"single"` bootstrapping that resamples only one element every time; or the `"null"` algorithm that resamples *ALL* the elements in the trait space (i.e. not only the elements in the group).

Think about what the difference each algorithm could make to your data and test it!

[**Click to expand the solution**]:

```
## Bootstrapping the data 3 ways
full_boot   <- dispRity(boot.matrix(my_groups, boot.type = "full"),
                        metric = c(sum, variances))
single_boot <- dispRity(boot.matrix(my_groups, boot.type = "single"),
                        metric = c(sum, variances))
null_boot   <- dispRity(boot.matrix(my_groups, boot.type = "null"),
```

```
                          metric = c(sum, variances))

## Plotting the differences
op <- par(mfrow = c(1,3))
plot(full_boot, main = "full bootstrap")
plot(single_boot, main = "single bootstrap")
plot(null_boot, main = "null bootstrap")
```




```
par(op)
```

4.1.3 Testing hypotheses 1

Once we have a distribution of disparity values (here from our bootstrap), we can actually calculate the probability of our groups sharing the same distribution or not (i.e. calculating a p-value). Because our distributions are not independent data (i.e. the values come from bootstraps, not observations), we can use a non-parametric t-test that doesn't assume independence in the data: the Wilcoxon rank test (`wilcox.test`). We can do that directly by extracting the bootstrapped disparity values (using get disparity):

```
## Extracting the disparity values
bootstrapped_values <- get.disparity(my_disparity, observed = FALSE)
## Testing the differences
wilcox.test(bootstrapped_values$crown[[1]], bootstrapped_values$stem[[1]])
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  bootstrapped_values$crown[[1]] and bootstrapped_values$stem[[1]]
## W = 8479, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Or, more nicely, by using the `test.dispRity` function:

```
## Running the test automatically on the dispRity object
test.dispRity(my_disparity, test = wilcox.test)
```

```
## [[1]]
##              statistic: W
## crown : stem         8479
##
## [[2]]
##                   p.value
## crown : stem 1.90691e-17
```

> Note that you can also use disparity statistics that are distributions rather than point estimates. E.g. all the `variances` distribution instead of the `c(sum, variances)` point estimate. More info and examples here.

If you had multiple groups, you can try doing pairs of tests using `comparisons` (but think about correcting for p-value inflation using the `correction` option), or, more neatly by using an `aov`.

> Note also that if you're just interested in comparing the matrices for each subset (not their disparity), you can use the `adonis.dispRity` which is a PERMANOVA wrapper function for `vegan::adonis2`.

4.2 Measuring disparity through time

One other obvious way to look at changes in disparity is to look at disparity through time. There are two main ways to look at it - which weirdly are both called "disparity through time":

- Measuring disparity at different points in time. We are going to cover that first and call it simply **disparity through time**.
- Measuring disparity through time and comparing it to models of change in disparity. We are going to cover that later and call it `dtt`.

And we'll chat about the differences between both methods of course!

USE YOUR DATA: get the first and last occurrence data (FAD/LAD) for your own data (as a .csv file)

```
  ,FAD ,LAD
species_a, 37.20, 36.80
species_b, 83.60, 72.10
```

```
## Read your favourite csv file (you'll need to set up the path yourself!)
my_FADLAD <- read.csv("path_to_my_super_serious_dataset.csv")
```

CATCHING UP ZONE: Using the occurrence dataset from Beck and Lee (2014)

```
## Crown and stem mammals
my_FADLAD <- read.csv("../examples/FADLAD/Beck2014.csv", row.names = 1)
```

4.2.1 Time binning

One first and relatively obvious way is to calculate disparity between groups but where groups are different time bins. This is easily done in `dispRity` with the `chrono.subsets` function using the `method = "discrete"`. Here for example, we can create 5 equally sized time bins across the temporal distribution of our data:

```
## Creating discrete bin subsets
time_subsets <- chrono.subsets(my_traitspace, method = "discrete",
                               tree = discrete_tree, FADLAD = my_FADLAD,
                               time = 5)
```

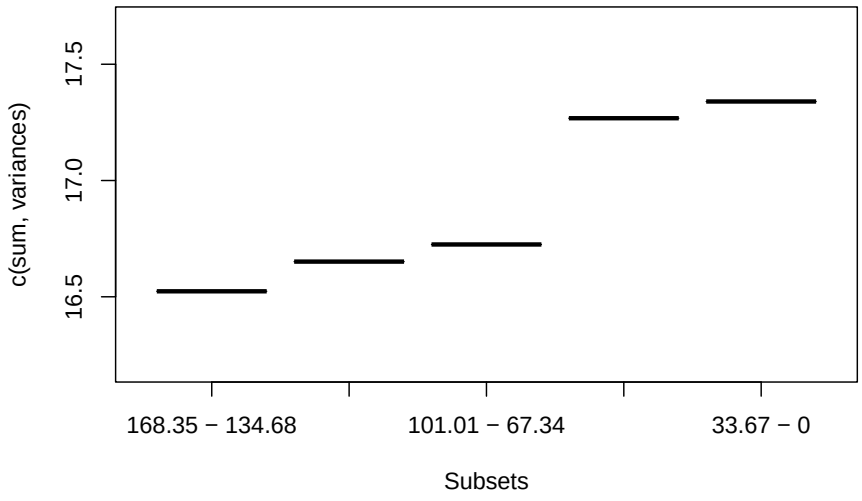
> Note that if you have occurrence data for all your species you don't need a `tree` argument.

We can then calculate and visualise disparity

```
## Calculating disparity per bin
time_disparity <- dispRity(time_subsets, metric = c(sum, variances))
summary(time_disparity)
```

```
##           subsets  n   obs
## 1 168.35 - 134.68  3 16.52
## 2 134.68 - 101.01  7 16.65
## 3  101.01 - 67.34 31 16.72
## 4   67.34 - 33.67 50 17.27
## 5       33.67 - 0 18 17.34
```

```
plot(time_disparity)
```



Tada! That was super easy!

TINKER TIMES: you can also make it a little bit more interesting or at least a bit more related to geology by specifying time bins that correspond to geological layers. You can do that by specifying a vector for time in the format of `time = c(145, 66, 23)` for the boundaries of the Cretaceous (from 145 million years - mya - to 66 mya) and the Palaeogene (66 to 23 mya). See if you can do that for your dataset.

[**Click to expand the solution**]:

```
## Option 1: enter the boundaries ages manually
## boooooooring! - and prone to errors

## Option 2: use the palaeoverse package!
## Getting the geological time from the palaeoverse package
period_bins <- palaeoverse::time_bins(rank = "period")
```

```
## Registered S3 method overwritten by 'e1071':
##   method       from
##   print.fclust GET
```

```
## Get the boundaries from Jurassic to present
period_bins_bounds <- c(period_bins[c(8:12), "max_ma"], 0)

## Creating the subsets
time_subsets_geo <- chrono.subsets(my_traitspace, method = "discrete",
```

```
                                    tree = discrete_tree, FADLAD = my_FADLAD,
                                    time = period_bins_bounds)
```

```
## Warning: The interval 23.03 - 2.58 is empty.
```

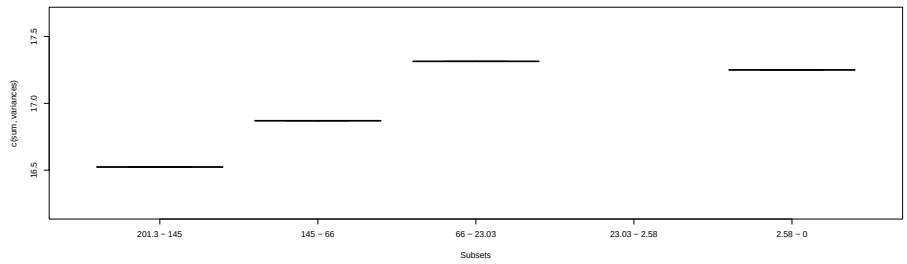
```
## Calculating disparity per bin
time_disparity2 <- dispRity(time_subsets_geo, metric = c(sum, variances))
```

```
## Warning in dispRity(time_subsets_geo, metric = c(sum, variances)): Disparity
## not calculated for subset 23.03 - 2.58 (not enough data).
```

```
summary(time_disparity2)
```

```
##        subsets  n   obs
## 1  201.3 - 145  3 16.52
## 2     145 - 66 40 16.87
## 3   66 - 23.03 50 17.31
## 4 23.03 - 2.58  0    NA
## 5     2.58 - 0 14 17.25
```

```
plot(time_disparity2)
```



This is all well and good but there can be some problems with the time binning method covered just above. We discuss most of the potential problems here: Guillerme and Cooper (2018). Two of the main ones are that:

1. It assumes punctuated equilibrium disparity evolution (i.e. disparity is uniform within bins and changes only - and drastically - at bins boundary)
2. It can artificially increase the effect of extinction events by forcing changes at a bin boundary.

4.2.2 Time slicing!

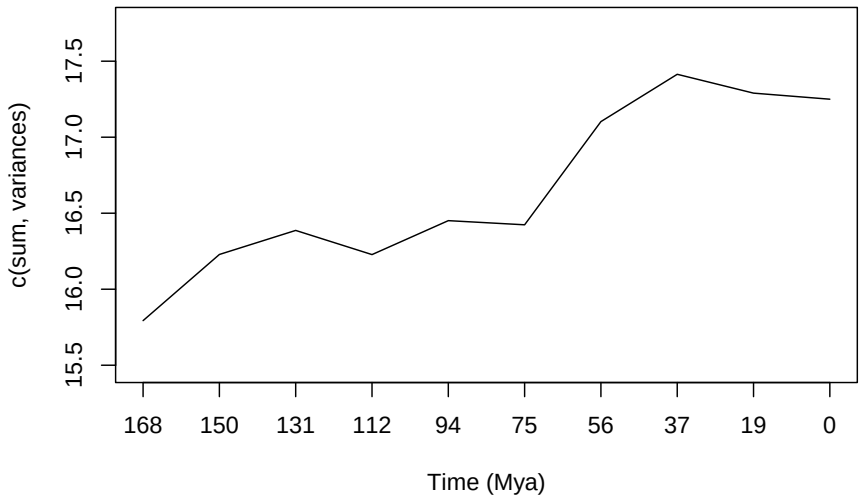
SLIDES: Time slicing

Let's try to do a simple ACCTRAN model (accelerated transition) on ten time slices:

```
## Creating continuous time slices subsets
time_subsets <- chrono.subsets(my_traitspace, method = "continuous",
                               tree = discrete_tree, FADLAD = my_FADLAD,
                               time = 10, model = "acctran")
## Calculating disparity per bin
time_disparity <- dispRity(time_subsets, metric = c(sum, variances))
summary(time_disparity)
```

```
##    subsets  n   obs
## 1   168.35  3 15.79
## 2   149.64  6 16.23
## 3   130.94 14 16.39
## 4   112.23 18 16.23
## 5    93.53 26 16.45
## 6    74.82 36 16.42
## 7    56.12 38 17.10
## 8    37.41 24 17.41
## 9    18.71 15 17.29
## 10       0 14 17.25
```

```
plot(time_disparity)
```

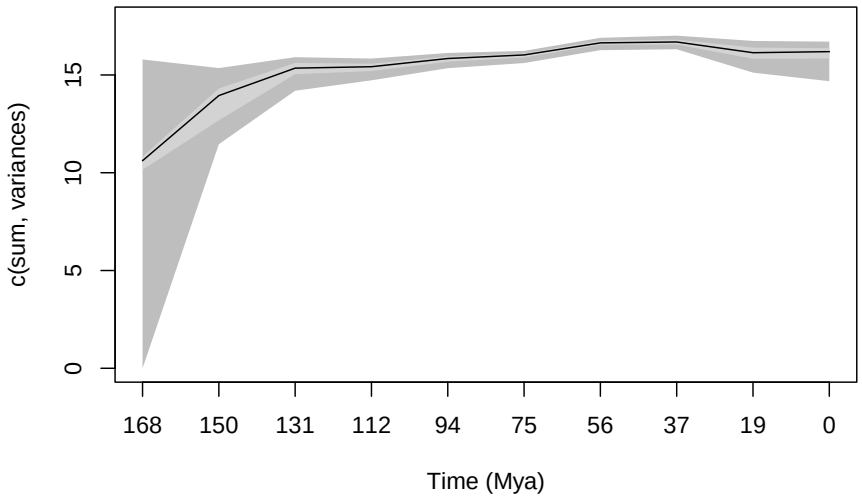


Note that so far the model we use is not probabilistic, that means that every time you run it you'll have the same solution. However, some of the models (called the `*.split` models) are probabilistic in nature and would require you bootstrapping the data to show the range of estimated solutions:

```
## Creating continuous time slices subsets using a split model
gradual_split <- chrono.subsets(my_traitspace, method = "continuous",
                                tree = discrete_tree, FADLAD = my_FADLAD,
                                time = 10, model = "gradual.split")
## Calculating disparity per bin
time_disparity <- dispRity(boot.matrix(time_subsets), metric = c(sum, variances))
summary(time_disparity)
```

```
##    subsets  n   obs bs.median  2.5%   25%   75% 97.5%
## 1   168.35  3 15.79     10.62  0.00 10.15 10.82 15.79
## 2   149.64  6 16.23     13.95 11.46 12.68 14.31 15.35
## 3   130.94 14 16.39     15.35 14.19 15.04 15.62 15.91
## 4   112.23 18 16.23     15.42 14.73 15.20 15.58 15.84
## 5    93.53 26 16.45     15.84 15.35 15.68 15.95 16.13
## 6    74.82 36 16.42     16.02 15.61 15.90 16.11 16.23
## 7    56.12 38 17.10     16.64 16.27 16.54 16.75 16.90
## 8    37.41 24 17.41     16.69 16.32 16.56 16.81 17.01
## 9    18.71 15 17.29     16.14 15.12 15.83 16.40 16.74
## 10       0 14 17.25     16.19 14.68 15.84 16.37 16.71
```

```
plot(time_disparity)
```



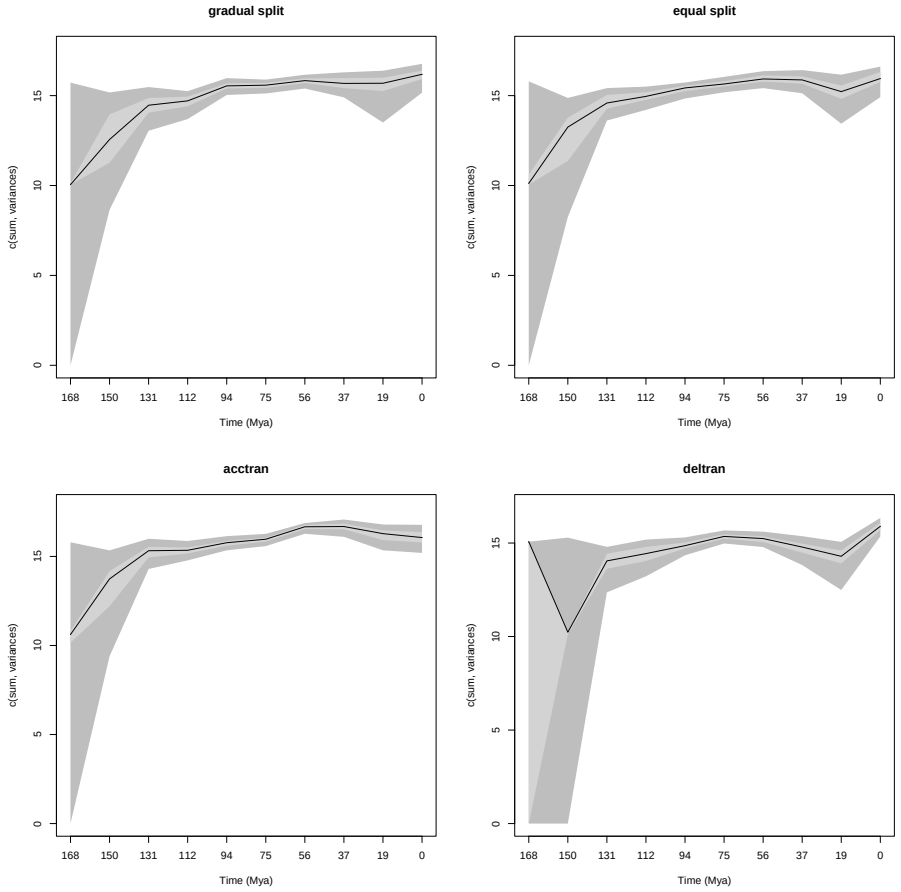
TINKER TIMES: have a look at comparing the different models of time-slicing and see how it changes your disparity curves (if it changes them at all!).

[**Click to expand the solution**]:

```
## Testing different models
gradual_split <- chrono.subsets(my_traitspace, method = "continuous",
                                tree = discrete_tree, FADLAD = my_FADLAD,
                                time = 10, model = "gradual.split")
equal_split   <- chrono.subsets(my_traitspace, method = "continuous",
                                tree = discrete_tree, FADLAD = my_FADLAD,
                                time = 10, model = "equal.split")
acctran       <- chrono.subsets(my_traitspace, method = "continuous",
                                tree = discrete_tree, FADLAD = my_FADLAD,
                                time = 10, model = "acctran")
deltran       <- chrono.subsets(my_traitspace, method = "continuous",
                                tree = discrete_tree, FADLAD = my_FADLAD,
                                time = 10, model = "deltran")
## Calculating disparity per bin
gradual_split_disp <- dispRity(boot.matrix(gradual_split), metric = c(sum, variances))
equal_split_disp   <- dispRity(boot.matrix(equal_split), metric = c(sum, variances))
acctran_disp       <- dispRity(boot.matrix(acctran), metric = c(sum, variances))
deltran_disp       <- dispRity(boot.matrix(deltran), metric = c(sum, variances))
```

```
## Warning in boot.matrix(deltran): The following subsets have less than 3 elements: 168.35.
## This might effect the bootstrap/rarefaction output.
```

```
op <- par(mfrow = c(2,2))
plot(gradual_split_disp, main = "gradual split")
plot(equal_split_disp, main = "equal split")
plot(acctran_disp, main = "acctran")
plot(deltran_disp, main = "deltran")
```




```
par(op)
```

4.2.3 `dtt`: disparity through time

`dtt` analyses are disparity through time analyses where the disparity is calculated for each clade in a given tree and then compared to a simulated disparity curve (usually a Brownian motion one) using the same tree. This results in a comparison between an observed clade disparity curve scaled by the number of species and a simulated trait. Note that this is quiet different than disparity through time how we measured it previously.

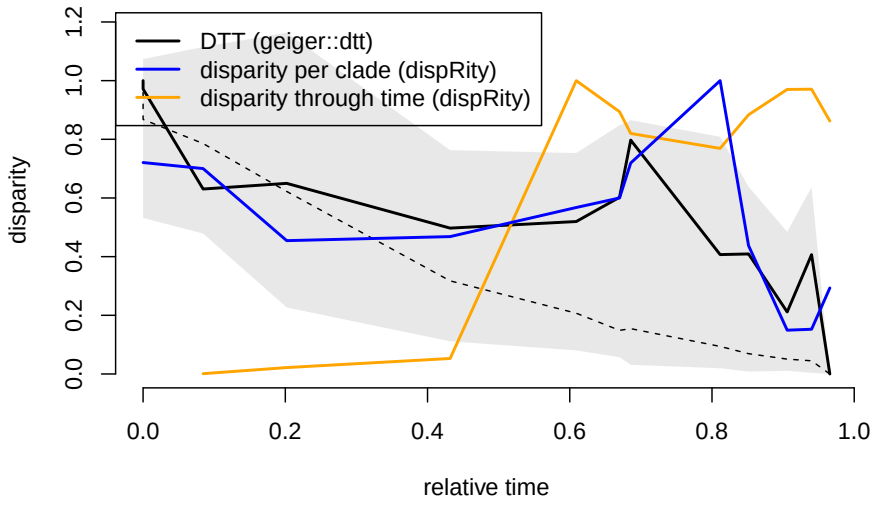
Here is a comparison with the three methods with:

- DTT being the disparity-through-time as described in Harmon et al. (2003) and calculated in Pennell et al. (2014);
- the raw disparity per clade calculated using `dispRity` (and plotted through time).

- the disparity through time using `dispRity` (time-slicing using the proximity model).

  Note that I'm not going to go in the details here but you can see and follow the code in the raw markdown file if you want.

```
## Warning in dispRity(time_subsets, metric = average.sq): Disparity not
## calculated for subset 0.5833 (not enough data).
```



You can see there is some difference between all the curves here! The ones between the DTT and the clade disparity (black and blue) can be probably explained by algorithmic choices between both approaches (the scaling procedure is a bit more complex in `geiger:dtt` and the clade disparity values are time slice averages - from what I understood). But in general both curves are similar.

This is not the case though with the disparity through time analyses `dispRity` style where disparity is measured as snapshots of disparity in the trait space at any specific time point (here the nodes ages, corresponding to the clade ages). So keep these two differences in mind and sorry about the confusing name similarities!

So which method is better? I have some opinion here but I'm biased and I like to remind you to choose the method best adapted to your question, not especially because of somebody's opinion (e.g. mine).

That said, one of the major drawbacks I see with the `dtt` method is that, unfortunately, the implementation is not really modular. It currently doesn't allow trees with fossil data and it only works with three inbuilt disparity metrics.

In general though, both methods (`dtt` and disparity through time) both have the problem on ways to test null expectations that are easily to calculate but also make sense biologically. For example, in `dtt`, comparing the observed disparity to a simple Brownian Motion can often be a bit naive and unsurprising (i.e. it's often easily expected that some trait have a complex evolutionary history and are not just following a BM).

Stay tuned for some potential solution with the Guillerme (2024) package allowing to simulate more complex data and evolutionary histories to compare observed and simulated disparity. I'm still working on it but if you feel explorative, please contact me or check out this looooooong tutorial video and vignette.

4.3 Measuring disparity with trees

Finally, note that here we haven't looked much at phylogenetic hypotheses and phylogenetic comparative methods. This is mainly because I didn't want to add to much stuff and that we'll have a great session dedicated to it tomorrow!

But here's a brief example on how to look at disparity using standard phylogenetic generalised least square (pgls) method. Note that this type of analysis is only relevant if you calculated disparity using a dimension level 2 metric that is linked to the rows (for example the distance of each element to the centroid using `centroids`) and if you have only data for your tips:

```
## Simple example
data(BeckLee_mat50)
data(BeckLee_tree)
disparity <- dispRity(BeckLee_mat50, metric = centroids, tree = BeckLee_tree)

## Running a simple PGLS
model <- pgls.dispRity(disparity)
summary(model)
```

```
##
## Call:
## [1] "dispRity interface of phylolm using: formula = disparity ~ 1 and model = BM"
##
##     AIC  logLik
## -158.71   81.35
##
## Raw residuals:
##      Min       1Q   Median       3Q      Max
## -0.06247 -0.02075  0.03188  0.07433  0.14999
##
## Mean tip height: 88.13181
## Parameter estimate(s) using ML:
## sigma2: 5.776737e-05
```

```
##
## Coefficients:
##             Estimate   StdErr t.value   p.value
## (Intercept) 1.541096 0.024797  62.148 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-squared:     0 Adjusted R-squared:     0
```

We will be covering that in much more details tomorrow!

4.4 Telling disparity stories

We've looked at many methods but now it's time to work on telling a story!

Here are some great examples:

- disparity through time in crocodiles by Godoy (2020)
- relation between changes in disparity and temperature in turtles by Farina et al. (2023)
- morphospace changes in brachiopods by Guo et al. (2024)
- diversification of sabertooth tigers by Chatar et al. (2024)
- Mezosoic archosaurs disparity by Shipley et al. (2024)
- convergence between mosasaurs and early cetacean skulls by Bennion et al. (2023)
- niche partitioning in marine reptiles by Foffa, Young, and Brusatte (2024)

TINKER TIMES: Do your very own disparity analysis!

Think about what the difference each algorithm could make to your data and test it!

[**Click to expand the solution**]:

Question: is there an effect of the the K-Pg extinction on mammalian disparity?

Here we will be looking at how the

1- From data to trait spaces

Here we will again use the data from Beck and Lee (2014) to test our hypothesis. We will transform the data by doing:

- Ancestral states estimations on discrete characters
- Calculating a distance matrix using the MORD distance (Lloyd (2016))
- Ordinating the distance matrix using a PCO with the Cailliez correction (Cailliez (1983))

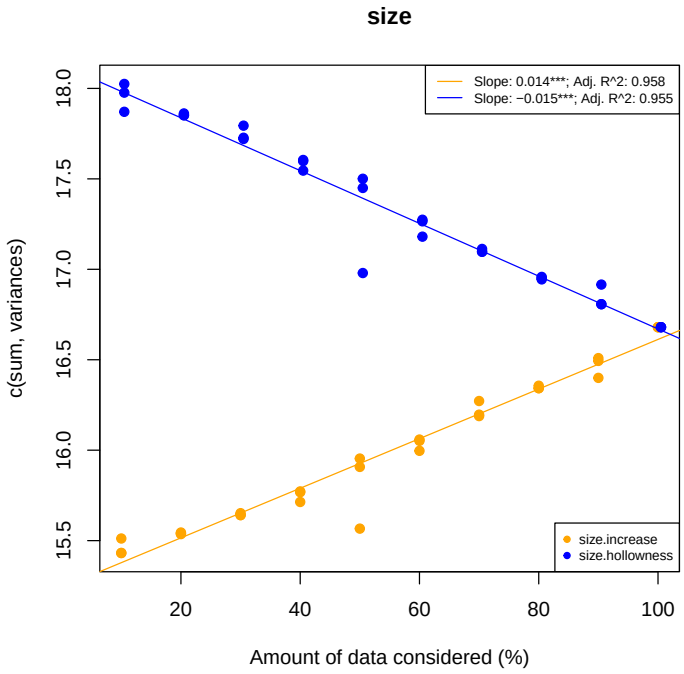
```
## Get discrete data
discrete_traits <- read.nexus.data("../examples/discrete_characters/Beck2014.nex")
## Combine the output list into a matrix (rows are species and columns are characters)
```

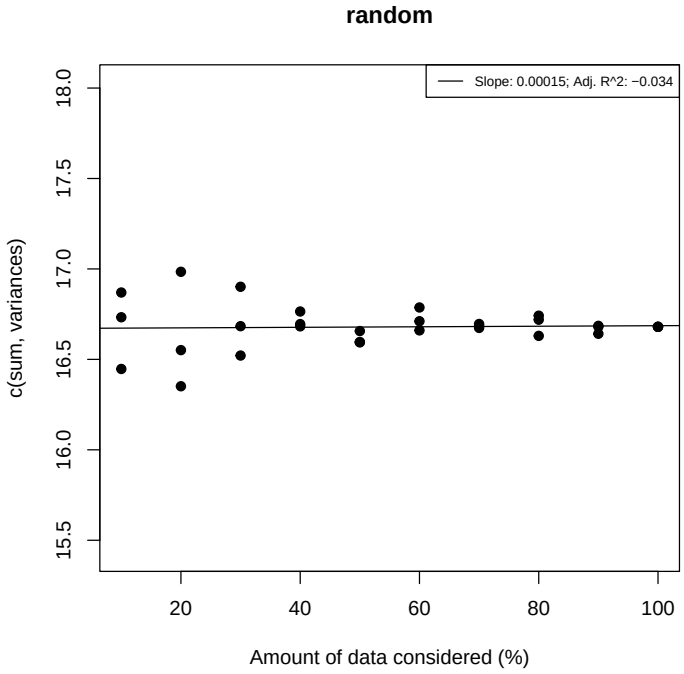
```
discrete_data <- do.call(rbind, discrete_traits)
## Get the tree
discrete_tree <- read.nexus("../examples/trees/Beck2014.tre")

## Cleaning both the tree and the data to match
cleaned_data  <- clean.data(data = discrete_data, tree = discrete_tree)
discrete_data <- cleaned_data$data
discrete_tree <- cleaned_data$tree

## Adding node labels
discrete_tree <- makeNodeLabel(discrete_tree, prefix = "n")
## Adding the root time
discrete_tree$root.time <- max(tree.age(discrete_tree)$age)

## Estimating ancestral states
ancestral_states <- multi.ace(discrete_data, discrete_tree, output = "combined")

# Calculating the distance matrix
distance_matrix <- char.diff(ancestral_states, method = "mord", by.col = FALSE)

## Ordinating the data
my_pco <- cmdscale(distance_matrix,
                   # the number of dimensions
                   k = ncol(distance_matrix)-2,
                   # the Cailliez correction
                   add = TRUE)

## Getting just the coordinates
my_traitspace <- my_pco$points
```

2- From traitspace to disparity

From this traitspace we will choose one metric looking at the change in disparity trait space size using the sum of each dimension's variance.

```
## Testing the sum of variances as a proxy of traitspace size
my_test <- test.metric(my_traitspace, metric = c(sum, variances), shifts = c("size", "
plot(my_test)
```

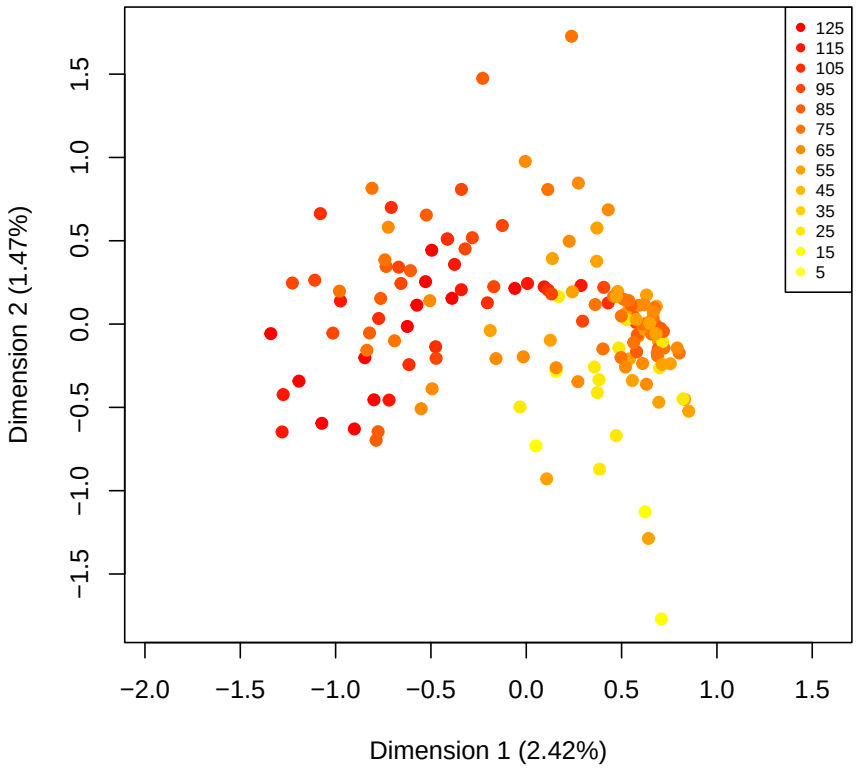

size
Slope: 0.014***; Adj. R^2: 0.958
Slope: −0.015***; Adj. R^2: 0.955
size.increase
size.hollowness
c(sum, variances)
15.5
16.0
16.5
17.0
17.5
18.0
20
40
60
80
100
Amount of data considered (%)


random
Slope: 0.00015; Adj. R^2: −0.034
c(sum, variances)
15.5
16.0
16.5
17.0
17.5
18.0
20
40
60
80
100
Amount of data considered (%)

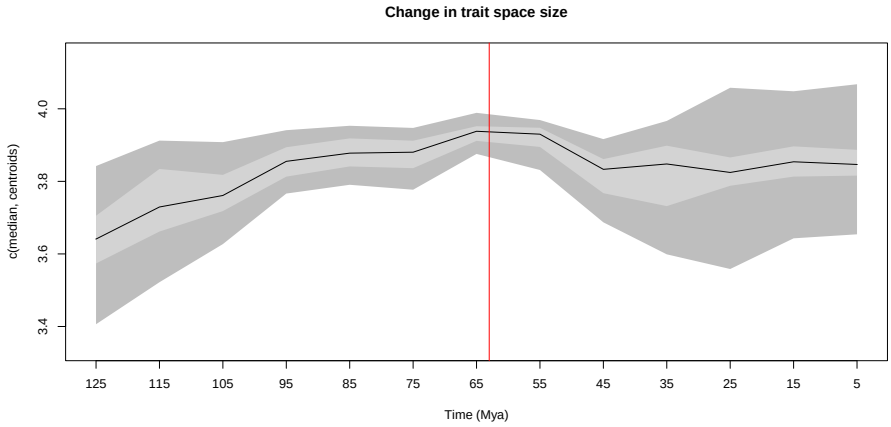
3 - From disparity to macroevolution

With the trait space and the chosen disparity metrics we can now test whether there is a change it trait space size or density before and after the K-Pg extinction?

```
## Creating the time subsets
time_subsets <- chrono.subsets(my_traitspace, tree = discrete_tree,
                               method = "continuous", model = "proximity",
                               time = rev(seq(from = 5, to = 125, by = 10)))
## Visualising the traitspace
plot(time_subsets)
```




```
## Measuring disparity through time
disparity_size <- dispRity(boot.matrix(time_subsets), metric = c(median, centroids))
plot(disparity_size, main = "Change in trait space size")
abline(v = 7.2, col = "red")
```



Testing the change in disparity in terms of of size after the K-Pg extinction with a lag of up to 30 million years:

```
## Selecting the two subsets after the mass extinction
extinction_effect <- list(c("65", "55"), c("65", "45"))

## Testing the differences in disparity
results <- test.dispRity(disparity_size, comparisons = extinction_effect,
                         test = wilcox.test, correction = "bonferroni")
results
```

```
## [[1]]
##         statistic: W
## 65 : 55         5798
## 65 : 45         9759
##
## [[2]]
##               p.value
## 65 : 55 1.026858e-01
## 65 : 45 6.018892e-31
```

Here we see now clear difference between before the K-Pg extinction (65 mya) and right after (55 mya) but some clear affect 20 million years later (45 mya). This suggest more an effect of the PETM 55 mya than the K-Pg on this dataset!

4.5 References

Baken, E. K., M. L. Collyer, A. Kaliontzopoulou, and D. C. Adams. 2021. "Geomorph V4.0 and gmShiny: Enhanced Analytics and a New Graphical Interface for a Comprehensive Morphometric Experience." *Methods in Ecology and Evolution* 12: 2355–63.

Bapst, David W. 2012. "Paleotree: An R Package for Paleontological and Phylogenetic Analyses of Evolution." *Methods in Ecology and Evolution* 3 (5): 803–7.

Beck, Robin MD, and Michael SY Lee. 2014. "Ancient Dates or Accelerated Rates? Morphological Clocks and the Antiquity of Placental Mammals." *Proceedings of the Royal Society B: Biological Sciences* 281 (1793): 20141278.

Bell, Mark A., and Graeme T. Lloyd. 2015. "Strap: An R Package for Plotting Phylogenies Against Stratigraphy and Assessing Their Stratigraphic Congruence." *Palaeontology* 58: 379–89. https://doi.org/10.1111/pala.12142.

Bennion, Rebecca F, Jamie A MacLaren, Ellen J Coombs, Felix G Marx, Olivier Lambert, and Valentin Fischer. 2023. "Convergence and Constraint in the Cranial Evolution of Mosasaurid Reptiles and Early Cetaceans." *Paleobiology* 49 (2): 215–31.

Brazeau, Martin D. 2011. "Problematic Character Coding Methods in Morphology and Their Effects." *Biological Journal of the Linnean Society* 104 (3): 489–98.

Brennan, Ian G, David G Chapple, J Scott Keogh, and Stephen Donnellan. 2024. "Evolutionary Bursts Drive Morphological Novelty in the World's Largest Skinks." *bioRxiv*, 2024–06.

Cailliez, Francis. 1983. "The Analytical Solution of the Additive Constant Problem." *Psychometrika* 48 (2): 305–8.

Chatar, Narimane, Margot Michaud, Davide Tamagnini, and Valentin Fischer. 2024. "Evolutionary Patterns of Cat-Like Carnivorans Unveil Drivers of the Sabertooth Morphology." *Current Biology* 34 (11): 2460–73.

Ciampaglio, Charles N, Matthieu Kemp, and Daniel W McShea. 2001. "Detecting Changes in Morphospace Occupation Patterns in the Fossil Record: Characterization and Analysis of Measures of Disparity." *Paleobiology* 27 (4): 695–715.

Dushoff, Jonathan, Morgan P Kain, and Benjamin M Bolker. 2019. "I Can See Clearly Now: Reinterpreting Statistical Significance." *Methods in Ecology and Evolution* 10 (6): 756–59.

Efron, Bradley, and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. Chapman; Hall/CRC.

Farina, Bruna M, Pedro L Godoy, Roger BJ Benson, Max C Langer, and Gabriel S Ferreira. 2023. "Turtle Body Size Evolution Is Determined by Lineage-Specific Specializations Rather Than Global Trends." *Ecology and Evolution* 13 (6): e10201.

Foffa, Davide, Mark T Young, and Stephen L Brusatte. 2024. "Comparative Functional Morphology Indicates Niche Partitioning Among Sympatric Marine Reptiles." *Royal Society Open Science* 11 (5): 231951.

Gearty, William. 2024. *Deeptime: Plotting Tools for Anyone Working in Deep Time*. https://CRAN.R-project.org/package=deeptime.

Godoy, Pedro L. 2020. "Crocodylomorph Cranial Shape Evolution and Its Relationship with Body Size and Ecology." *Journal of Evolutionary Biology* 33 (1): 4–21.

Guillerme, Thomas. 2018. "DispRity: A Modular R Package for Measuring Disparity." *Methods in Ecology and Evolution* 9 (7): 1755–63.

———. 2024. "Treats: A Modular R Package for Simulating Trees and Traits." *Methods in Ecology and Evolution* 15 (4): 647–56.

Guillerme, Thomas, Pedro Cardoso, Maria Wagner Jørgensen, Stefano Mammola, and Thomas J Matthews. 2024. "The What, How and Why of Trait-Based Analyses in Ecology." *bioRxiv*, 2024–06.

Guillerme, Thomas, and Natalie Cooper. 2018. "Time for a Rethink: Time Sub-Sampling Methods in Disparity-Through-Time Analyses." *Palaeontology* 61 (4): 481–93.

Guillerme, Thomas, Natalie Cooper, Stephen L Brusatte, Katie E Davis, Andrew L Jackson, Sylvain Gerber, Anjali Goswami, et al. 2020. "Disparities in the Analysis of Morphological Disparity." *Biology Letters* 16 (7): 20200199.

Guillerme, Thomas, Mark N Puttick, Ariel E Marcy, and Vera Weisbecker. 2020. "Shifting Spaces: Which Disparity or Dissimilarity Measurement Best Summarize Occupancy in Multidimensional Spaces?" *Ecology and Evolution* 10 (14): 7261–75.

Guo, Zhen, Michael J Benton, Thomas L Stubbs, and Zhong-Qiang Chen. 2024. "Morphological Innovation Did Not Drive Diversification in Mesozoic–Cenozoic Brachiopods." *Nature Ecology & Evolution*, 1–11.

Harmon, Luke J, James A Schulte, Allan Larson, and Jonathan B Losos. 2003. "Tempo and Mode of Evolutionary Radiation in Iguanian Lizards." *Science* 301 (5635): 961–64.

Hopkins, Melanie J, and Sylvain Gerber. 2021. "Morphological Disparity." *Evolutionary Developmental Biology: A Reference Guide*, 965–76.

Jones, Lewis A., William Gearty, Bethany J. Allen, Kilian Eichenseer, Christopher D. Dean, Sof<U+00ED>a Galv<U+00E1>n, Miranta Kouvari, et al. 2023. "Palaeoverse: A Community-Driven R Package to Support Palaeobiological Analysis." *Methods in Ecology and Evolution*, 1–11. https://doi.org/10.1111/2041-210X.14099.

Legendre, Pierre, and Louis Legendre. 2012. *Numerical Ecology*. Vol. 24. Elsevier.

Lloyd, Graeme T. 2016. "Estimating Morphological Diversity and Tempo with Discrete Character-Taxon Matrices: Implementation, Challenges, Progress,

and Future Directions." *Biological Journal of the Linnean Society* 118 (1): 131–51.

———. 2018. "Journeys Through Discrete-Character Morphospace: Synthesizing Phylogeny, Tempo, and Disparity." *Palaeontology* 61 (5): 637–45.

Mammola, Stefano, Carlos P Carmona, Thomas Guillerme, and Pedro Cardoso. 2021. "Concepts and Applications in Functional Diversity." *Functional Ecology* 35 (9): 1869–85.

Muff, Stefanie, Erlend B Nilsen, Robert B O'Hara, and Chloé R Nater. 2022. "Rewriting Results Sections in the Language of Evidence." *Trends in Ecology & Evolution* 37 (3): 203–10.

Paradis, Emmanuel, and Klaus Schliep. 2019. "Ape 5.0: An Environment for Modern Phylogenetics and Evolutionary Analyses in R." *Bioinformatics* 35: 526–28. https://doi.org/10.1093/bioinformatics/bty633.

Parins-Fukuchi, Caroline. 2018. "Use of Continuous Traits Can Improve Morphological Phylogenetics." *Systematic Biology* 67 (2): 328–39.

Pennell, M. W., J. M. Eastman, G. J. Slater, J. W. Brown, J. C. Uyeda, R. G. Fitzjohn, M. E. Alfaro, and L. J. Harmon. 2014. "Geiger V2.0: An Expanded Suite of Methods for Fitting Macroevolutionary Models to Phylogenetic Trees." *Bioinformatics* 30: 2216–8. https://doi.org/10.1093/bioinformatics/btu181.

Petchey, Owen L, and Kevin J Gaston. 2006. "Functional Diversity: Back to Basics and Looking Forward." *Ecology Letters* 9 (6): 741–58.

Schlager, Stefan. 2017. "Morpho and Rvcg – Shape Analysis in R." In *Statistical Shape and Deformation Analysis*, edited by Guoyan Zheng, Shuo Li, and Gabor Szekely, 217–56. Academic Press.

Shipley, Amy E, Armin Elsler, Suresh A Singh, Thomas L Stubbs, and Michael J Benton. 2024. "Locomotion and the Early Mesozoic Success of Archosauromorpha." *Royal Society Open Science* 11 (2): 231495.

Stewart, Kerry, Carlos P Carmona, Chris Clements, Chris Venditti, Joseph A Tobias, and Manuela González-Suárez. 2023. "Functional Diversity Metrics Can Perform Well with Highly Incomplete Data Sets." *Methods in Ecology and Evolution* 14 (11): 2856–72.