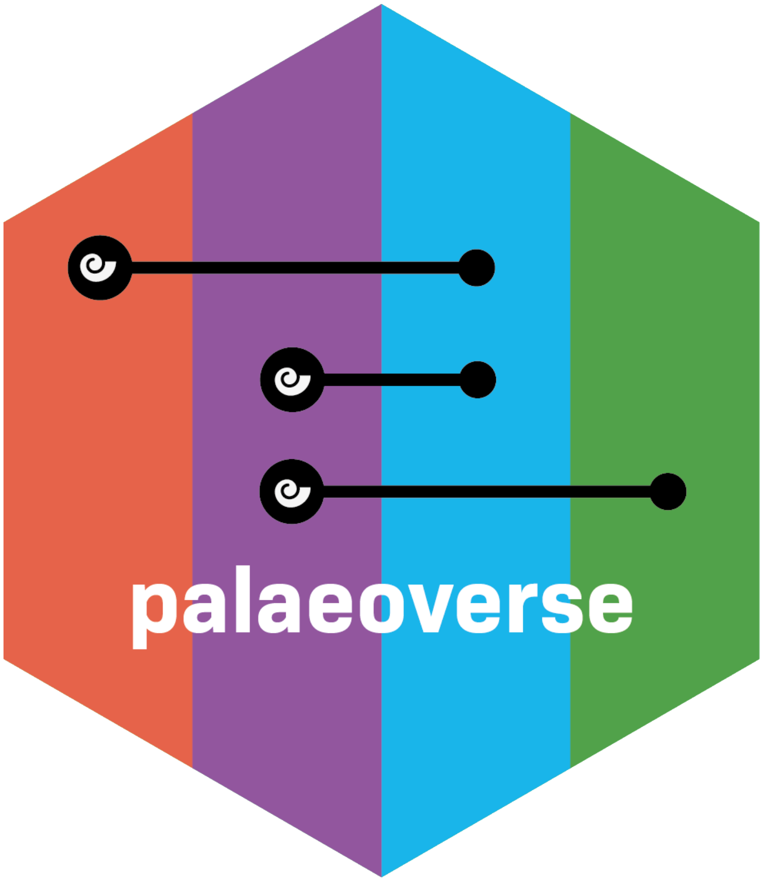
Introduction to palaeoverse: Structure and Conventions



Lewis A. Jones

Last updated: 30 May, 2022

1 Introduction

The 'palaeoverse' *R* package is a community-driven software library providing generic tools for palaeobiological analysis. The core principles of palaeoverse are to: (1) streamline analyses, (2) enhance code readability, and (3) improve reproducibility of results.

This document describes the essential structure and conventions of the palaeoverse R package. Naturally, there are always disagreements regarding best practices and conventions and palaeoverse is no exception. Despite this, all the essentials in palaeoverse are encouraged to make the lives of both the developer, and the user easier. It is worth noting that the core structure and conventions adopted in palaeoverse are heavily influenced by Hadley Wickham and Jenny Bryan's R Packages and the tidyverse style guide, which is currently also Google's guide.

"Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread." *–tidyverse style guide*

2 Files

2.1 Names

File names should always be concise, meaningful and end in `'.R'`. Avoid using special characters whenever possible (i.e. stick to letters and numbers), and use `'_'` or `'-'` instead of spaces in file names. The use of lowercase is also strongly encouraged, and **never** have file names that only differ in capitalization. Note: the preferred separator is `'_'` to consist with the 'snake_case' convention for developing functions (more on that later!).

```
# Nein!
Sup3r Aw3sum Functi0n.r

# Besser
super_awesome_function.R

# Das ist gut!
time_binning.R
```

2.2 Organisation

2.3 Internal structure

If your script makes use of other packages, load them all at the beginning. This is more transparent for the user than having various packages sprinkled throughout the code.

Ensure that your code is easily readable for other developers and users. For example, when you are commenting code, you might want to break your code into chunks using lines of `'-'` or `'='`.

```
# Load packages =======================
library(palaeoverse)
library(AwesomePackage)

# Clean up taxonomic data =============

# Run analyses ========================
```

3 Syntax

4 Data

Often we will want to include data in palaeoverse. This might come in the form of example datasets for testing functions (e.g. fossil occurrences), reference datasets such as the Geological Timescale 2020, or even data that is fundamental for a function to run. In palaeoverse's structure, we currently recognize three main ways of including data in the package depending on its usage.

- Raw data
- Internal data
- Exported data

4.1 Raw data

Raw data should always be included in `inst/extdata`. If you want to include cleaned data in `data/`, it is generally a good idea to include the code used to process the raw data. If you ever need to reproduce or update your cleaned data, this will save you precious time. The code for processing your data should be included in `data-raw/`. Strictly speaking, raw data does not need to be documented. However, it is a good idea to include the original source and version (including a download date) in your code.

4.2 Exported data

Package data you wish to make available to the user should be stored in `data/`. Each file in this directory should be either a `.rda` or `.RData`. This file type is fast, small and explicit. The most appropriate way to include exported data is to use `usethis::use_data()`.

```
fossils <- readRDS("./inst/extdata/raw_fossil_data.RDS")
usethis::use_data(fossils)
```

When using large datasets, we want to ensure that our files are not bloated and taking up too much space on our user's machine. As such, you may want to experiment with the compression settings in `usethis::use_data()`. Generally, `xz` and `gzip` can create smaller files than the default `bzip2`. You can also implement several 'hacks' to generate smaller files which you may want to consider for large datasets (depending on whether your data is sensitive to such changes). Data with many decimal places consume a lot of memory, consider how many significant figures are relevant for your data, and `round()` accordingly. You can also experiment with your file size by multiplying your data by X (e.g. 1,000) to remove decimal places altogether. **Note:** Remember to undo any transformations when calling or working with the data.

4.3 Internal data

Data you do not wish to directly make available to the user should be saved as `R/sysdata.rda`. This is the best option for pre-computed data tables that are needed for a function to run. For example, in palaeoverse we have pre-rotated spatial grids rotated we use in the `palaeorotate()`. Objects in `R/sysdata.rda` are not exported, so again, strictly speaking raw data does not need to be documented. However, it is a good idea to document the internal data in the function documentation.

```
Merdith2021 <- readRDS("./inst/extdata/Merdith2021.RDS")
usethis::use_data(Merdith2021, internal = TRUE)
```

4.4 Data documentation

Objects in `data/` are always exported by default, and should be documented accordingly. In order to document data, you document the name of the dataset and save it in `R/data`. For example, the documentation block used to document GTS2020 is saved as `R/data.R` and is similar to the following (simplified here):

```
#' Geological Time Scale 2020
#'
#' A dataset of the Geological Time Scale 2020. Age data from:
#'  \url{https://stratigraphy.org/timescale/}.
#' Definitions of relative climate states are also included in the dataset, and were
#' compiled from various resources (see item descriptions).
#'
#' @format A data frame with 189 rows and 20 variables:
#' \describe{
#'   \item{stage_name}{Name of stratigraphic stage}
#'   ...
#' }
#' @section References:
#' Gradstein, F.M., Ogg, J.G., Schmitz, M.D. and Ogg, G.M. eds., 2020.
#' Geologic time scale 2020. Elsevier.
#' @source Compiled by Lewis A. Jones.
"GTS2020"
```

4.4.1 What does it mean to document your data?

Include section here. . .

4.5 Data size

Include section here about reducing data size. . .

5 Functions

5.1 Function documentation

5.2 Error messages

5.3 Tests

6 Contributing

6.1 GitHub

6.2 Direct