Introduction to fossilbrush (v1.0.0), an R package for automated cleaning and error flagging of fossil occurrence data

Joe Flannery Sutherland

17/03/2022

Introduction

`fossilbrush` is a package for automated cleaning and error flagging of fossil occurrence data. The functions were developed using datasets from the Paleobiology Database (PBDB), but can be applied to any ocurrence dataset with taxonomic information and chronostratigraphic ranges present, along with locality information if available. Any number of taxonomy columns are allowed (including missing values) and locality information is only necessary for the function `revise_ranges` (see **Error flagging against a range database**).

Family	Genus	First Appearence Datum	Last Appearence Datum	Locality
Examplidae	Joebloggsia	250.2	247.1	Atlantis
Examplidae	Joebloggsia	250.2	247.1	Atlantis
Taxonidae	Facsimilus	243.6	241.1	Xanadu
NA	Imperfectella	243.6	241.5	Xanadu
Taxonidae	Automaton	243.6	241.5	Xanadu
Examplidae	NA	218.5	210.8	Narnia

Data acquisition and chronostratigraphy

To begin, we will need to install the package from the repository on Github: `devtools::install_github("jf15558/fossilbru`
For this vignette, we will use an example dataset of Palaeozoic brachiopods downloaded from the PBDB, along with the taxonomic ranges of brachiopods taken from the Sepkoski Compendium.

```
# load the package
library(fossilbrush)
# load the example datasets
data(brachios)
data(sepkoski)
# trim the Sepkoski Compendium to the relevant entries
sepkoski <- sepkoski[which(sepkoski$PHYLUM == "Brachiopoda"),]
```

It is important that the chronostratigraphic information in a dataset is up-to-date and consistent between data and analytical strategy. The Sepkoski Compendium included here has dating from A Geologic Timescale 2020 (GTS2020) pre-applied. Paleobiology Database data uses outdated International Chronostratigraphic Commission on Stratigraphy 2013 (ICS2013) dates, so we will use a lookup function, `GTS2020_scale` to convert chronostratigraphic interval names to GTS2020 values. This is useful to update existing datasets, but new data can als be downloaded using `get_pbdb`, with the option to automatically apply GTS2020 redating.

```
# update chronostratigraphy
brachios <- GTS2020_scale(brachios, srt = "early_interval", end = "late_interval",
                          max_ma = "max_ma", min_ma = "min_ma", verbose = FALSE)
# brachios <- get_pbdb("Brachiopoda", interval = "Palaeozoic, tscale = "GTS2020")
```

Data cleaning

Next, we will scan the occurrence dataset for taxonomic inconsistencies and errors, as well as harmonising the taxonomic schemes between the PBDB dataset and the Sepkoski Compendium. This latter step is not necessary for error analysis, but demonstrates how a dataset compiled from multiple databases may be cleaned and harmonised. First, we will combine the PBDB occurrences and Sepkoski Compendium into one object using basic R, then apply the cleaning function. This function implements a four-step cleaning routine which scans for formatting irregularities, spelling inconsistencies, re-use of names at different taxonomic ranks, and conflicting higher classification schemes. At each step, potential errors can be cleaned automatically or simply flagged. It is advisable to start by flagging errors only, then investigating them manually as the automatic cleaning is performed heuristically and is not guaranteed to be correct.

```
# combine the datasets
occs <- cbind.data.frame(phylum = c(brachios$phylum, sepkoski$PHYLUM),
                         class = c(brachios$class, sepkoski$CLASS),
                         order = c(brachios$order, sepkoski$ORDER),
                         family = c(brachios$family, rep(NA, nrow(sepkoski))),
                         genus = c(brachios$genus, sepkoski$GENUS),
                         max_ma = c(brachios$GTS_FAD, sepkoski$RANGE_BASE),
                         min_ma = c(brachios$GTS_LAD, sepkoski$RANGE_TOP),
                         coll_no = c(brachios$collection_no, rep(NA, nrow(sepkoski))))
# define the taxonomic ranks used in the dataset (re-used elsewhere)
b_ranks <- c("phylum", "class", "order", "family", "genus")
# define a list of suffixes to be used at each taxonomic level when scanning for synonyms
b_suff = list(NULL, NULL, NULL, NULL, c("ina", "ella", "etta"))
# scan for errors
occs_c <- check_taxonomy(occs, suff_set = b_suff, ranks = b_ranks)
```

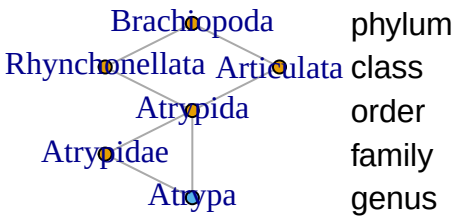
```
## Checking formatting [1/4]

##  - formatting errors detected (see $formatting in output)

## Checking spelling   [2/4]

##  - potential synonyms detected (see $synonyms in output)

## Checking ranks      [3/4]

##  - no cross-rank names detected

## Checking taxonomy   [4/4]

##  - conflicting classifications detected (see $duplicates in output)
```

The function output contains a cleaned dataset (`occs_c$data`) which could be taken directly, but we will instead examine the flagged errors as this is best practise. Multiple formatting irregularities are detected and their indexes provided. Examining these (e.g. `occs$genus[occs_c$formatting$genus]`) shows that most of these cases arise from inclusion of a subgenus name in brackets, or use of the PBDB-specific format for a lack of taxon assignment (e.g. NO_FAMILY_SPECIFIED). Only a few synonyms are flagged so we can inspect these manually (`occs_c$synonyms`) to see which ones we want to resolve.

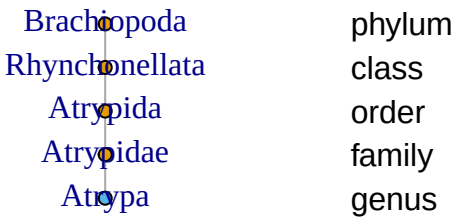
```
# manually resolve the clear synonymous spellings
occs$family[which(occs$family == "Disciniidae")] <- "Discinidae"
occs$genus[which(occs$genus == "Ptychomalotoechia")] <- "Ptychomaletoechia"
occs$genus[which(occs$genus == "Hipparionix")] <- "Hipparionyx"
occs$genus[which(occs$genus == "Sphaenospira")] <- "Sphenospira"
# strip out the PBDB 'missing taxon' format
for(i in c("phylum", "class", "order", "family", "genus")) {
  occs[grep("^NO_", occs[,i]),i] <- NA
}
```

We will now reapply the cleaning function to automatically resolve conflicting classifications and any remaining formatting errors, then plot the same taxon to confirm that its classification has been resolved. The jump parameter determines for how many levels a conflicting classification must diverge in order to be considered a homonym - taxa which are taxonomically distinct but which have inadvertantly been given the same name. We know that all classifications are the same at phylum level - Brachiopoda - and that four other levels of classification are present so we will set the jump parameter to 5. For other datasets, a more conservative jump parameter will permit strongly divergent classifications to treated as homonyms.

```
# clean the data, this time resolving classifications
occs_c <- check_taxonomy(occs, suff_set = b_suff, ranks = b_ranks, verbose = FALSE,
                         clean_name = TRUE, resolve_duplicates = TRUE, jump = 5)
# plot a taxon before and after cleaning to confirm that it is correct
par(mfrow = c(1, 2))
plot_taxa(occs, "Atrypa", trank = "genus", ranks = b_ranks, mode = "parent")
```




```
plot_taxa(occs_c$data, "Atrypa", trank = "genus", ranks = b_ranks, mode = "parent")
```



We can now be confident that that dataset is clean and taxonomically harmonised, so we will extract the PBDB and Sepkoski datasets from the cleaned object.

```
# extract PBDB
sepkoski_c <- occs_c$data[(nrow(brachios) + 1):nrow(occs_c$data),]
# extract Sepkoski
brachios_c <- occs_c$data[1:nrow(brachios),]
```

Error flagging against a range database

We will revise the occurrence ages in the PBDB dataset (`x`) using the taxon ranges given in the Sepkoski Compendium (`y`) with the `revise_ranges` function. Ages are revised locality-wise (`brachios_c$coll_no`) so that the stratigraphic consistency of the assemblages are preserved, using a consensus of ages from any taxa which also have entries in the range dataset. To expedite the process we will also enable error flagging (`do.flag = TRUE`) so that only assemblages which contain error-flagged occurrences are revised. This flagging is performed internally by `flag_ranges` which compares occurrence ages against their taxon ranges individually, rather than locality-wise. After revision the new occurrence ages, which remain consistent locality-wise, are appended to the dataset and are available for analysis.

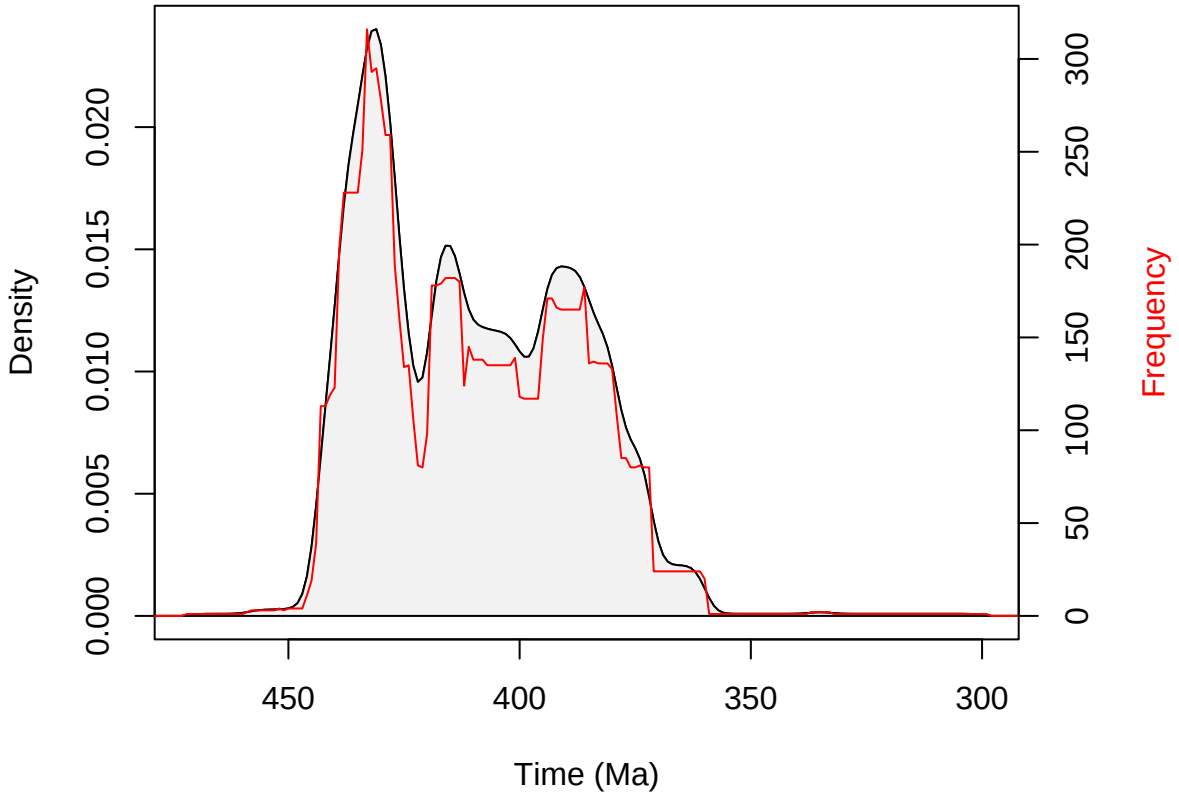
```
# drop occurrences with older LADs than FADs
brachios_c <- brachios_c[brachios_c$max_ma > brachios_c$min_ma,]
# chunk to a small size for better run time
set.seed(1)
samp <- sample(1:nrow(brachios_c), 1000)
# flag and resolve against the Sepkoski Compendium, collection-wise
```

```
revrng <- revise_ranges(x = brachios_c[samp,], y = sepkoski_c, do.flag = TRUE, verbose = F,
                        taxon = "genus", assemblage = "coll_no",
                        srt = "max_ma", end = "min_ma")
# append the revised occurrence ages and error codes to the dataset
brachios_c$newfad <- brachios_c$newlad <- brachios_c$errcode <- NA
brachios_c$newfad[samp] <- revrng$occurrence$FAD
brachios_c$newlad[samp] <- revrng$occurrence$LAD
brachios_c$errcode[samp] <- revrng$occurrence$tax_flag
```

Error flagging using pacmacro

The other functions in this package use the stratigraphic density of occurrences to flag taxa with anomalous ranges and occurrences. The core of these methods is the construction of a taxon-density matrix, which has its own dedicated function to allow density profiles to be constructed then plotted. We will construct the density matrix for the Palaeozoic brachiopods, where the columns are genera and the rows are their densities at regular time intervals from the start of the dataset to the end of the dataset (i.e. the matrix rows begins at the first appearence datum of the oldest occurrence and proceed step-wise to the last appearence datum of the youngest occurrence).

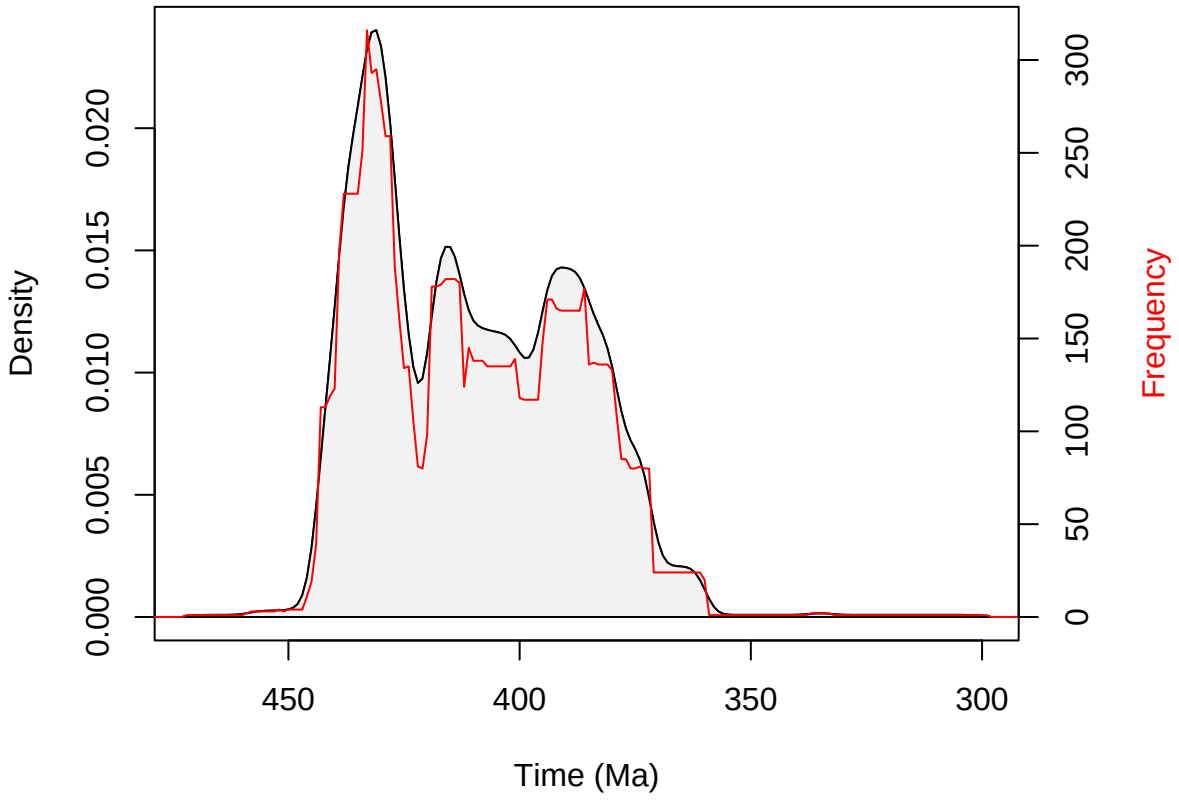
```
# densify ranges
dens <- densify(brachios_c)
# plot an example taxon
plot_dprofile(dens, "Atrypa")
```



We can see that the tails of the density profile for *Atrypa* are rather long, suggesting that some occurrences have been misidentified outside of its true stratigraphic range. We can scan for anomalously long tails using its stratigraphic density profile, based on the proportion of the stratigraphic duration occupied by a given proportion of total occurrence density contained in the tails. This is performed by `pacmacro_ranges` which takes the occurrence dataset directly and constructs the density profiles internally before analysing the tail

proportions and determining the truncated stratigraphic range after tail removal. Multiple tail proportions can be investigated at once to permit sensitivity testing (`tail.flag = c(0.3, 0.35, 0.4)`), but we will focus on results from the function default of 0.35. In the output, the `tflag*` column (multiple columns if multiple tail proportions are tested) marks long-tailed taxa with a 1.

```
# pacmacro trimming
pacm <- pacmacro_ranges(brachios_c, tail.flag = c(0.3, 0.35, 0.4),
                        rank = "genus", srt = "max_ma", end = "min_ma")
# replot the taxon and mark its truncated ranges
plot_dprofile(dens, "Atrypa")
abline(v = pacm$kdensity["Atrypa","FAD95"], col = "blue")
abline(v = pacm$kdensity["Atrypa","LAD95"], col = "blue")
```



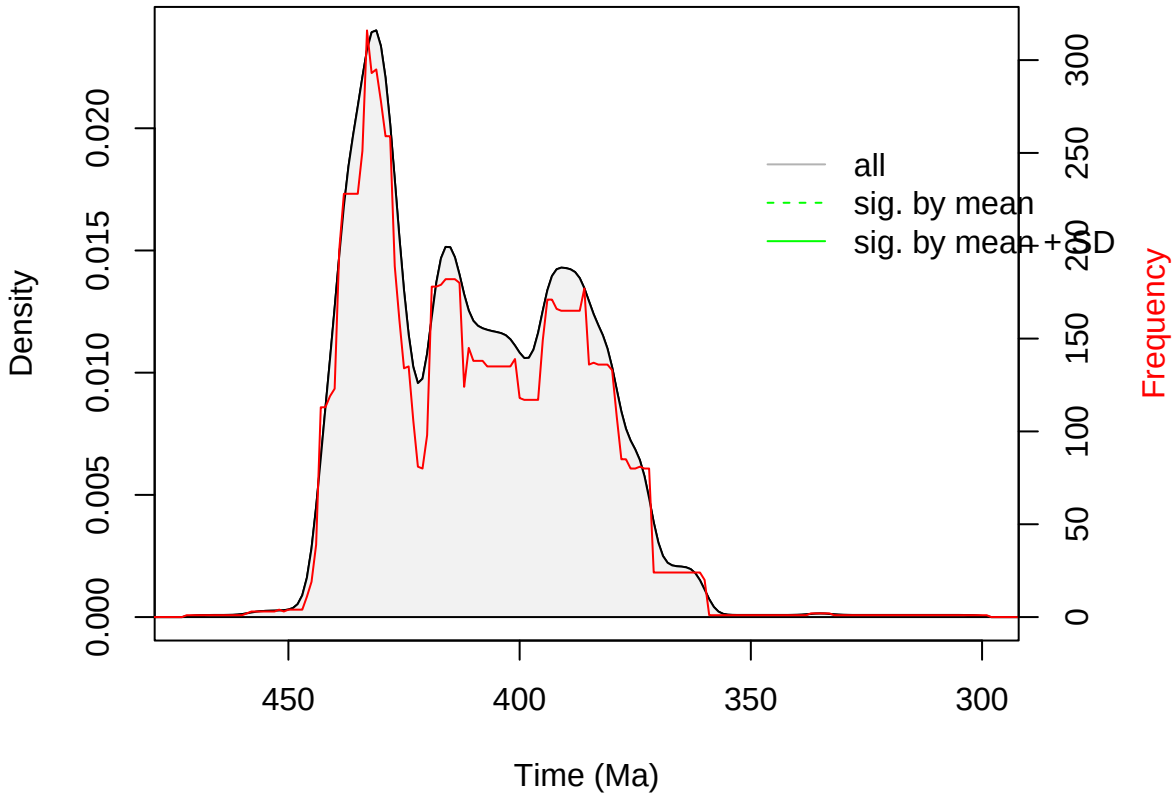
The analysis confirms that *Atrypa* shows a long-tailed stratigraphic density profile. For all the taxa with anomalously long tails, we can then reflag the database against the truncated ranges to identify the anomalous occurrences, appending the flag codes to the dataset. In this way, taxa with error flags (anything other than R1R) can be dropped from the dataset if desired or scrutinised for expert cleaning prior to analysis.

```
# extract the truncated ranges
pranges <- pacm$kdensity
# for those identified as anomalous (tflag = 1), update the range values to the 95% trim
pranges$FAD[which(pranges$tflag0.35 == 1)] <- pranges$FAD95[which(pranges$tflag0.35 == 1)]
pranges$LAD[which(pranges$tflag0.35 == 1)] <- pranges$LAD95[which(pranges$tflag0.35 == 1)]
# format the range table
pranges <- cbind.data.frame(genus = rownames(pranges),
                            max_ma = pranges$FAD,
                            min_ma = pranges$LAD)
# perform the flagging and append to the dataset
pflags <- flag_ranges(brachios_c, pranges, verbose = FALSE)
brachios_c$pflag <- pflags$occurrence$status
```

Error flagging using interpeak thresholding

Another way to detect suspect data is to query the stratigrapic plausibility of the density profiles. In idealised terms, a density profile should be unimodal. In reality, it may contain multiple peaks due to incomplete sampling in the fossil record, or from conflation of taxonomically and stratigraphically distinct taxa into a single morphospecies due to misidentification or wastebasket taxon concepts. We can scan a density spectrum to identify the peaks and determine if they are separated by enough time to indicate conflation of records or otherwise using `threshold_ranges`. Again, this takes an occurrence dataset directly and constructs the density profile internally, but there are two settings we will adjust. We will increase the window size used to assess whether peaks are well differentiated from background noise, and increase the interpeak threshold to reflect the typical duration of marine genera along with the coarser dating intervals present in the Palaeozoic. We will then plot *Atrypa* again to see which peaks have been taken as representing distinct taxa, appending the new taxon groups to the dataset.

```
# interpeak thresholding
itp <- threshold_ranges(brachios_c, win = 8, thresh = 10,
                        rank = "genus", srt = "max_ma", end = "min_ma")
# append the stratigraphically thresholded taxon names to the dataset
brachios_c$newgen <- itp$data
# plot the taxon, now identifying the peaks
plot_dprofile(dens, "Atrypa")
add_itp(itp, "Atrypa")
```



In summary we now have a cleaned dataset with additional information which can help us to omit stratigraphically suspect data from downstream analyses, along with revised stratigraphic ages on which to perform the actual analyses - for example diversity through time.