Phanerozoic-scale global marine biodiversity analysis with the R package `divDyn v0.8`

Adam T. Kocsis, John Alroy, Carl J. Reddin, Wolfgang Kiessling

2019-06-11

1. Introduction

In this text we describe how to use the `divDyn` R package for Phanerozoic scale global biodiversity analysis with data acquired from the Paleobiology Database (PaleoDB, `http://www.paleobiodb.org/`, `http://fossilworks.org`) until the final results. The aim of this vignette is to describe a fully reproducible workflow. This text does not explain all the capabilities of the package. For a full treatment, please refer to the more detailed vignette 'Handout to the R package `divDyn v0.8.x` for diversity dynamics from fossil occurrence data'.

The code presented in this document will work with `divDyn` versions `0.8.x`, but it will be updated with every major release of the package. The most up-to-date version is available from

`https://github.com/divDyn/ddPhanero/blob/master/doc/dd_phanero.pdf`

in .pdf, and from

`https://github.com/divDyn/ddPhanero/blob/master/doc/dd_phanero.html`

In .html format.

1.1. Necessary files and installation

The package is available from the CRAN servers (`https://CRAN.R-project.org/package=divDyn`), which means that you can run

```
install.packages("divDyn")
```

from the R console to install `divDyn`. The package will be updated regularly, but it usually takes a week until all binaries are available from CRAN. To make sure that you have the most up-to-date version, you can check the package GitHub repository

`http://www.github.com/divDyn/r_package`

and install from there.

The files necessary to replicate the analyses presented here are deposited in another dedicated GitHub repository

`http://www.github.com/divDyn/ddPhanero`

and are referenced directly from the vignette. Along with the vignette, the data will be updated with every major release of the package and will be archived in Zenodo (`https://doi.org/10.5281/zenodo.2545982`). The analyses presented here are also available in a regular `.R` script file that was used to generate the display items (`scripts/1.0.1/2019-05-31_marineAnimals_ddPhanero.R`).

2. Getting the data

The occurrence data used in the study were retrieved from the Paleobiology Database. We used all occurrences from the Ediacaran to Holocene interval, with several additional variables (lithology, environment) to enable further analyses.

2.1. Our downloaded data

To ensure the exact reproducibility of the material presented here, the raw dataset that was used to calculate the results (downloaded on May 31, 2019) is available for download and experimentation. If you do not wish to carry out a PaleoDB download yourself, you can use our file.

```
load(url(
"https://github.com/divDyn/ddPhanero/raw/master/data/PaleoDB/2019-05-31_paleoDB.RData"
))
```

This download has

```
nrow(dat)
```

```
## [1] 1397110
```

rows. The command above will import a single `data.frame` class object `dat` to the global workspace. The metadata are available from the GitHub repository of the example analysis.

2.2. New download

In case you do want to carry out a new download, the download query is reproducible with the link that we used to access the data. The download will commence automatically if you copy and paste this URL into your browser.

```
https://paleobiodb.org/data1.2/occs/list.csv?interval=Ediacaran,Holocene&
show=class,classext,genus,subgenus,abund,coll,coords,loc,paleoloc,strat,stratext,lith,
env,ref,crmod, timebins,timecompare
```

It will take a considerable amount of time (30+ minutes) to download this set, and the resulting file is very large (ca. `1.4 GB`). Nevertheless, it can be stored effectively by omitting some variables and saving it in a binary `.RData` format. You can load the file (saved as `allData_2019-05-31.csv` in this case) with the `read.csv()` function.

```
# assuming that the file is in your current working directory
dat <- read.csv("allData_2019-05-31.csv", header=TRUE, stringsAsFactors=FALSE)
```

Then you can define the required columns. We used the following set so the final file can be retrieved from GitHub without timeout problems.

```
need<-c("collection_no", "collection_name","accepted_name",
"accepted_rank", "identified_name","identified_rank",
"early_interval", "late_interval", "max_ma","min_ma",
"reference_no", "phylum", "class", "order", "family",
"genus", "lng", "lat", "paleolng", "paleolat", "formation",
"lithology1", "lithification1", "environment", "created",
"zone")
dat <- dat[,need]
```

Then you can save everything as an `.RData` file,

```
save(dat, file="allData_2019-05-31.RData")
```

and use `load()` to read the file whenever you need it.

```
load(file="allData_2019-05-31.RData")
```

All additional filtering and transformation will be explained in the following section.

3. Data processing

Processing the raw download consists of steps that can have a huge effect on the results. These transformations include basic taxonomic and environmental filtering, as well as assigning the occurrences to stratigraphic bins and environmental categories.

3.1. Taxonomic filtering

This dataset is filtered to only comprise marine animal taxa and heterotrophic protists, i.e. the same taxonomic groups listed in Sepkoski's (2002) compendium. As data downloads can be tedious when the taxonomic filters of the PaleoDB are used, we decided to omit taxa procedurally using the available higher-level taxonomic fields: `phylum`, `class`, `order` and `family`.

Filtering started with omission of occurrences that were not identified to the level of genus, which is indicated in the `accepted_rank` field or with non-informative genus names (empty quotes):

```
dat <- dat[dat$accepted_rank %in% c("genus", "species"), ]
dat <- dat[dat$genus!="", ]
```

Omitting these poorly identified fossils decreases the sample size to

```
nrow(dat)
```

```
## [1] 1224822
```

occurrences. The filtering process continues with the selection of phyla that have or had recorded marine species. Phyla that have considerable terrestrial and marine records (chordates and arthropods) are included based on lower taxonomic ranks.

```
# sampled phyla
# levels(factors(dat$phylum))
#A. phyla
marineNoPlant <- c("",
  "Agmata",
  "Annelida",
  "Bilateralomorpha",
  "Brachiopoda",
  "Bryozoa",
  "Calcispongea",
  "Chaetognatha",
  "Cnidaria",
  "Ctenophora",
  "Echinodermata",
  "Entoprocta",
  "Foraminifera",
  "Hemichordata",
  "Hyolitha",
```

```
  "Mollusca",
  "Nematoda",
  "Nematomorpha",
  "Nemertina",
  "Onychophora",
  "Petalonamae",
  "Phoronida",
  "Platyhelminthes",
  "Porifera",
  "Problematica",
  "Rhizopodea",
  "Rotifera",
  "Sarcomastigophora",
  "Sipuncula",
  "Uncertain",
  "Vetulicolia",
  ""
)
```

Then a logical vector was defined for each row, suggesting whether the phylum of that occurrence is present or not in the above defined set `marineNoPlant`.

```
# logical vector of rows indicating these
bByPhyla <- dat$phylum %in% marineNoPlant
```

The rest of the data were saved for further filtering. Classes that are likely to be marine animals were extracted from this subset.

```
# noNeed <- dat[!bByPhyla,]
#B. classes
#levels(factor(noNeed$class))
needClass <- c(
    "Acanthodii",
    "Actinopteri",
    "Actinopterygii",
    "Agnatha",
    "Cephalaspidomorphi",
    "Chondrichthyes",
    "Cladistia",
    "Coelacanthimorpha",
    "Conodonta",
    "Galeaspida",
    "Myxini",
    "Osteichthyes",
    "Petromyzontida",
    "Plagiostomi",
    "Pteraspidomorphi",
    "Artiopoda",
    "Branchiopoda",
    "Cephalocarida",
    "Copepoda",
    "Malacostraca",
    "Maxillopoda",
    "Megacheira",
    "Merostomoidea",
```

```
    "Ostracoda",
    "Paratrilobita",
    "Pycnogonida",
    "Remipedia",
    "Thylacocephala",
    "Trilobita",
    "Xiphosura"
)
# logical vector of rows indicating occurrences
bNeedClass <- dat$class %in% needClass
```

The mammalian orders Sirenia and Cetacea were also included, as well as pinniped families from the order Carnivora.

```
#C. mammals
# mammals <- dat[dat$class=="Mammalia", ]
# levels(factor(mammals$order))
  needMammalOrd <- c("Cetacea", "Sirenia")
  bMammalOrder <- dat$order %in% needMammalOrd

# the carnivores
# carnivores <- dat[dat$order=="Carnivora", ]
# levels(factor(carnivores$family))
  needFam <- c("Otariidae", "Phocidae", "Desmatophocidae")
  bNeedMamFam <- dat$family %in% needFam
```

Some reptile orders were also included:

```
# D. Reptiles
# reptiles <- dat[dat$class=="Reptilia", ]
# levels(factor(reptiles$order))

needReptOrd<-c(
  "Eosauropterygia",
  "Hupehsuchia",
  "Ichthyosauria",
  "Placodontia",
  "Sauropterygia",
  "Thalattosauria"
)
# the logical vector for the total data
bRept <- dat$order %in% needReptOrd
```

Families of sea turtles are also added to the analyzed set.

```
# E. Sea turtles
# turtles <- dat[dat$order=="Testudines", ]
# levels(factor(turtles$family))

needTurtleFam <- c(
  "Cheloniidae",
  "Protostegidae",
  "Dermochelyidae",
  "Dermochelyoidae",
  "Toxochelyidae",
  "Pancheloniidae"
```

```
)
bTurtle <- dat$family%in%needTurtleFam
```

And then, we subsetted the data with these multiple filters. In this sense, the logical OR | works as a union operator between the taxonomic groups.

```
dat <- dat[bByPhyla | bNeedClass | bMammalOrder | bNeedMamFam | bRept | bTurtle, ]
```

The entire procedure decreases the number of occurrences to around 900,000.

```
# the number of rows after taxonomic filtering
nrow(dat)
```

```
## [1] 885799
```

After the filtering by taxonomy was finished, we can make sure that potential homonymies will not affect the results by combining the class names and genus names to create individual entries:

```
# resolve the potential homonymy problem
dat$clgen <- paste(dat$class, dat$genus)
```

3.2. Filtering by sedimentary environment

Some of the taxa above contain freshwater and/or terrestrial groups. These are expected to occur in such sedimentary environments, which is recorded in the `environment` field. We can list the sampled environments with this simple line of code.

```
# filter by environment
levels(factor((dat$environment)))
```

The following environments are expected to contain terrestrial taxa.

```
omitEnv <- c(
  "\"floodplain\"", "alluvial fan", "cave", "\"channel\"", "channel lag" ,
  "coarse channel fill", "crater lake", "crevasse splay", "dry floodplain",
  "delta plain", "dune", "eolian indet.", "fine channel fill", "fissure fill",
  "fluvial indet.", "fluvial-lacustrine indet.", "fluvial-deltaic indet.",
  "glacial", "interdune", "karst indet.", "lacustrine - large",
  "lacustrine - small", "lacustrine delta front", "lacustrine delta plain",
  "lacustrine deltaic indet.", "lacustrine indet.",
  "lacustrine interdistributary bay", "lacustrine prodelta", "levee", "loess",
  "mire/swamp", "pond", "sinkhole", "spring", "tar", "terrestrial indet.",
  "wet floodplain")
```

We can omit occurrences with these entries with a similar command as above.

```
# omit the occurrences
dat <- dat[!dat$environment%in%omitEnv, ]
```

Although this filtering is not perfect, we believe that it is adequate for answering large-scale questions. The remaining non-marine and non-animal occurrences are unlikely to influence the results. After this filtering step

```
nrow(dat)
```

```
## [1] 870560
```

occurrences remain in the dataset. Collections that came from unlithified sediments can yield fossils with unusually good preservation. As these are more frequent in younger sites and occur heterogeneously, sampling

bias can be reduced by omitting such collections from the data.

```
dat <- dat[dat$lithification1!="unlithified", ]
```

This last step leaves

```
nrow(dat)
```

```
## [1] 805994
```

occurrences for the analyses.

For additional calculations, the environmental and lithological information can be categorized to either `"reef"` or `"non-reef"` (environment type),`"shallow"` or `"deep"` (bathymetry), `"siliciclastic"` or `"carbonate"` (substrate type) and `"coarse"` or `"fine"` (grain size) entries. The `keys` object contains the information in a relevant form, which can be used by a grouping function `categorize()` that replaces groups of entries in a vector with single group names (see `?categorize`).

```
data(keys)
# using the others will be included in an appendix to this vignette
names(keys)
```

```
## [1] "tenInt" "stgInt" "reefs"  "lith"   "lat"    "bath"   "grain"  "depenv"
```

Then the grouping can be applied with:

```
# siliciclastic or carbonate?
dat$lith <- categorize(dat$lithology1,keys$lith)

# batyhmetry
dat$bath <- categorize(dat$environment,keys$bath)

# grain size
dat$gra <- categorize(dat$lithology1,keys$grain)

# reef or not?
dat$reef <- categorize(dat$environment, keys$reef)
dat$reef[dat$lith=="clastic" & dat$environment=="marine indet."] <- "non-reef"
```

Although these variables will not be used in the following analyses, they are useful for answering many questions regarding habitat preferences, macroevolutionary differences between environments and extinction selectivity.

3.3. Stratigraphic binning

To use the occurrences in the `divDyn` package in an efficient way, the entries must be assigned to a discrete time scale. We included two of these time-scales in the package: the widely-used 10 myr time scale of the Paleobiology Database, and another one based on the stratigraphic stages of Ogg et al. (2016). These can loaded with the `data()` function.

```
# 10 million year timescale
data(tens)
# stage-level timescale
data(stages)
```

You can learn more about these time scales if you type in `?tens` or `?stages`. The first time scale (`tens`) has 49 entries identifying roughly 10-million year bins. The second one (`stages`) has almost double the stratigraphic resolution, but some of the ICS stages are clumped to ensure a more even distribution of durations (cf. Miocene and Pliocene).

It is easier to handle the two timescales with the same functions if the time bin names have identical column names in both tables.

```
# names of the bins
colnames(tens)[colnames(tens)=="X10"] <- "name"
# names of the bins
colnames(stages)[colnames(stages)=="stage"] <- "name"
```

The original data in the Paleobiology Database get their stratigraphic information based on the `early_interval` and `late_interval` values. The valid entries in these variables come from a list of interval names that convert them to numeric ages and establish connections between the different entries. In the dynamic timescale of Fossilworks (J. Alroy), they are also linked to the 10 million year timescale and the ICS stages (Ogg et al. 2016), without the changes in the Neogene. The `early_interval` and `late_interval` values designate stratigraphic position in a straightforward way: the `early_interval` marks the oldest possible age and the `late_interval` the youngest. If a single name suffices to describe the inherent uncertainty of an interval, the `late_interval` remains empty.

Using a complete download of collections from Fossilworks, we compiled a table that resolves these 'interval' entries to the timescales of our interest. This table is the `stratkeys` object, which can be attached by running `data(stratkeys)`. The rows in this table were then transformed to `list` type entries that can be used by the `categorize()` function. Similarly to the environmental variables, this can be found in the `keys` object.

The stratigraphic binning starts with figuring out which numbered bins the `early_interval` and `late_interval` entries are assigned to. Let's start with the 10 million year bins.

```
# categorize entries as they are in the lookup table
tenMin <- categorize(dat[ ,"early_interval"], keys$tenInt)
tenMax <- categorize(dat[ ,"late_interval"], keys$tenInt)
```

Then the entries have to converted to simple numeric values.

```
tenMin <- as.numeric(tenMin)
tenMax <- as.numeric(tenMax)
```

This code creates two vectors of numeric bin numbers, where `NA` entries indicate that the names were not found in the table, and `-1` entries indicate empty character strings (where no `late_interval` entry is given). As our goal is to retain only those occurrences that have precise enough stratigraphic assignments, we only want to consider those occurrences that have either the same `tenMin` or `tenMax` number or where `tenMax` is `-1`. This is accomplished by the following steps.

First, a final, empty vector is defined.

```
dat$ten <- rep(NA, nrow(dat))
```

Then the condition above is expressed indicating which rows have only a single assigned bin number.

```
tenCondition <- c(
  # the early and late interval fields indicate the same bin
  which(tenMax==tenMin),
  # or the late_interval field is empty
  which(tenMax==-1))
```

Finally, those values are copied, where the condition is true.

```
# in these entries, use the bin indicated by the early_interval
  dat$ten[tenCondition] <- tenMin[tenCondition]
```

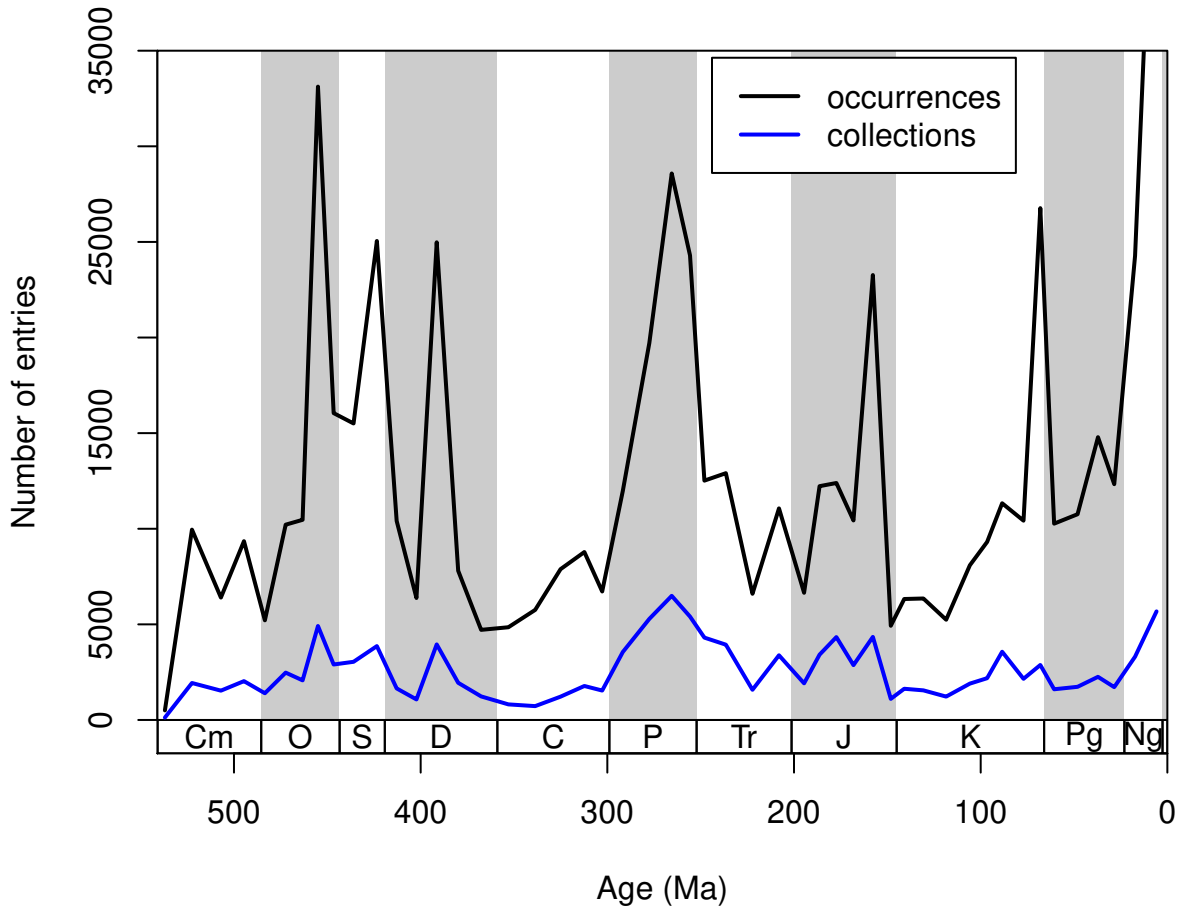
The final object is a column in the data table, where `dat$ten` is a single variable of integers that have `NA` entries where the collection/occurrence cannot be assigned to a single bin in the time scale. The number of occurrences and collections in each bin can be calculated with `table()` or in a single step with the `binstat()`

function.

```
sampTen <- binstat(dat, tax="clgen", bin="ten", coll="collection_no", duplicates=FALSE)
```

As every row in `dat` represents a species-level occurrence entry, multiple species of the same genus can be registered in every collection. These are only counted as one occurrence, when the `duplicates` argument of `binstat()` is set to `FALSE`. The resulting `data.frame` can be used to plot the number of occurrences and collections through time.

```
# the plot
  tsplot(stages, boxes="sys", shading="sys", xlim=4:95, ylim=c(0,35000),
    ylab="Number of entries", xlab="Age (Ma)")
# occurrences
  lines(tens$mid, sampTen$occs, lwd=2)
# collections
  lines(tens$mid, sampTen$colls, lwd=2, col="blue")
# legend
  legend("topright", bg="white", legend=c("occurrences", "collections"),
    col=c("black", "blue"), lwd=2, inset=c(0.15,0.01), cex=1)
```



The following code repeats the entire process for the stratigraphic stages:

```
# the 'stg' entries (lookup)
stgMin <- categorize(dat[ ,"early_interval"], keys$stgInt)
stgMax <- categorize(dat[ ,"late_interval"], keys$stgInt)

# convert to numeric
stgMin <- as.numeric(stgMin)
stgMax <- as.numeric(stgMax)

# empty container
dat$stg <- rep(NA, nrow(dat))

# select entries, where
stgCondition <- c(
# the early and late interval fields indicate the same stg
  which(stgMax==stgMin),
# or the late_intervar field is empty
  which(stgMax==-1))

# in these entries, use the stg indicated by the early_interval
  dat$stg[stgCondition] <- stgMin[stgCondition]
```

But beware! Stage-level assignments have been a problem in the earliest Paleozoic (Cambrian and Ordovician periods). Therefore, the assignments above are not perfect in these periods: numerous entries are not properly processed based on the interval lookup table. To make use of these collections in the analyses that follow, we processed these data separately. The necessary files to do these corrections are added to the example GitHub repository and will be updated for future use. Cambrian collections were assigned one-by-one by Na Lin for the analysis of studying the diversity dynamics of the Cambrian (Na and Kiessling, 2015). You can download these from:

```
load(url(
"https://github.com/divDyn/ddPhanero/raw/master/data/Stratigraphy/2018-08-31/cambStrat.RData"))
```

And apply them with the following script:

```
source(
"https://github.com/divDyn/ddPhanero/raw/master/scripts/strat/2018-08-31/cambProcess.R")
```

For the Ordovician assignments, Wolfgang Kiessling compiled tables on formations and biozones.

```
load(url(
"https://github.com/divDyn/ddPhanero/raw/master/data/Stratigraphy/2018-08-31/ordStrat.RData"))
```

You can use these with the following script.

```
source(
"https://github.com/divDyn/ddPhanero/raw/master/scripts/strat/2019-05-31/ordProcess.R")
```

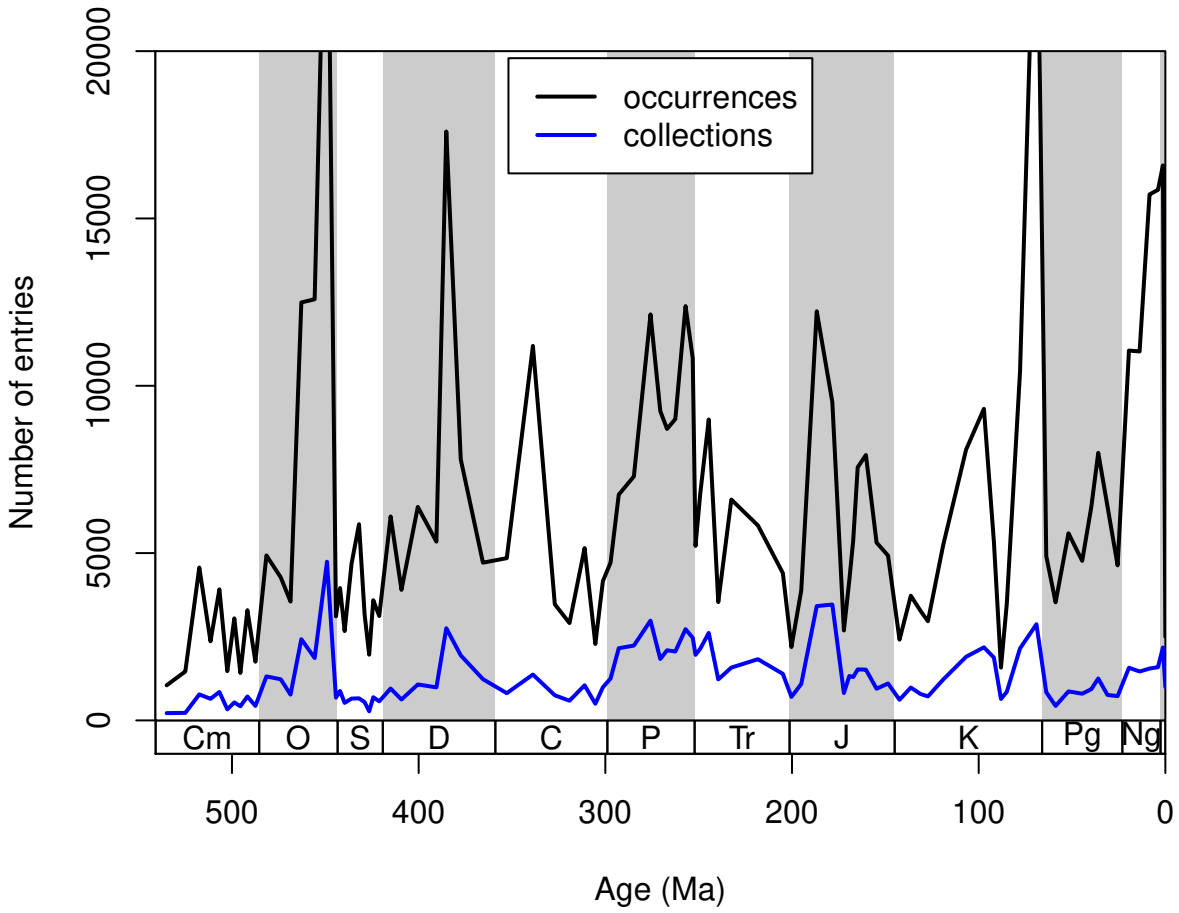
Now that we assigned the occurrences to stages, as above, the number of sampled occurrences and collections can be calculated with `binstat()`.

```
sampStg <- binstat(dat, tax="clgen", bin="stg", coll="collection_no", duplicates=FALSE)
```

And then can be plotted in a similar way.

```
# the plot
  tsplot(stages, boxes="sys", shading="sys", xlim=4:95, ylim=c(0,20000),
    ylab="Number of entries", xlab="Age (Ma)")
# occurrences
```

```
  lines(stages$mid, sampStg$occs, lwd=2)
# collections
  lines(stages$mid, sampStg$colls, lwd=2, col="blue")
# legend
  legend("top", bg="white", legend=c("occurrences", "collections"),
    col=c("black", "blue"), lwd=2, inset=c(0.15,0.01), cex=1)
```



4. Calculating richness (Figure 1)

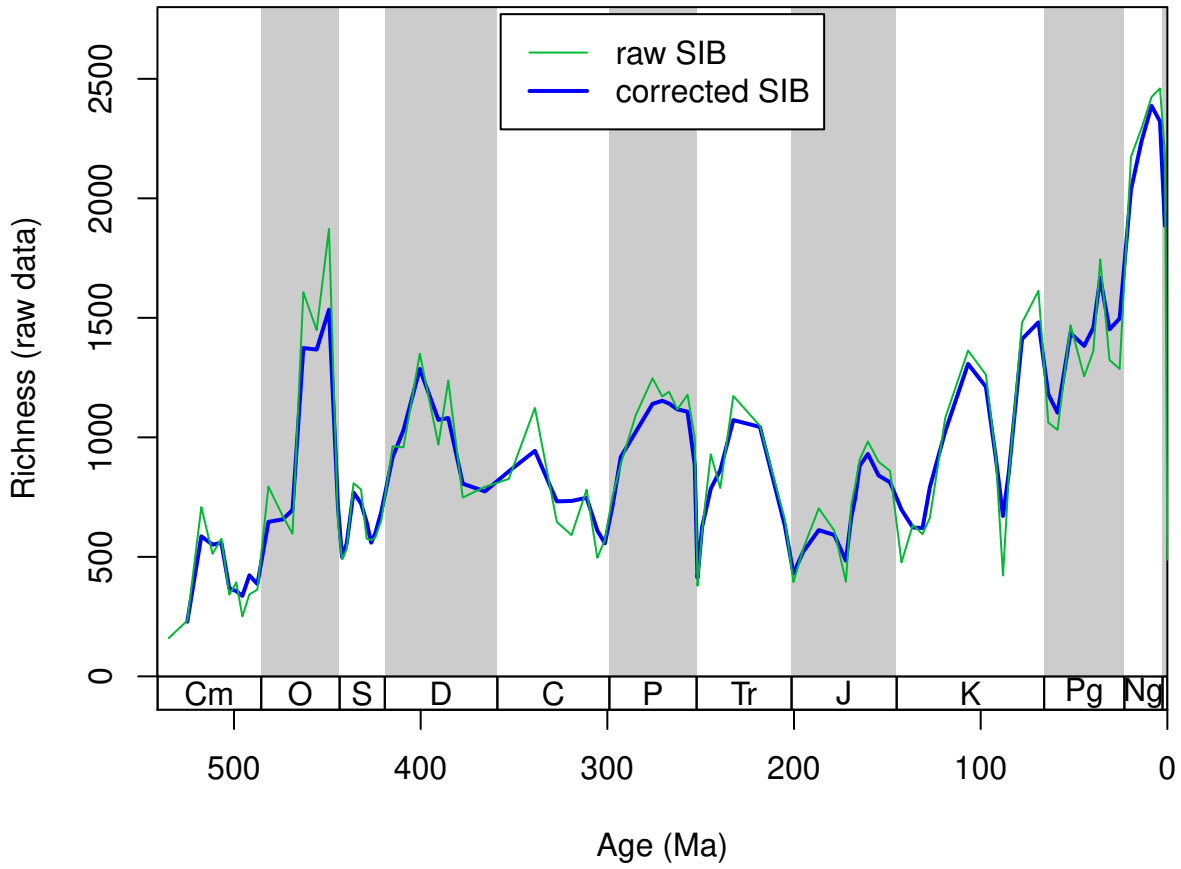
The first figure of the paper depicts changes in genus richness over the Phanerozoic, using the 10 million year intervals with raw and subsampled data. You can calculate the raw results with the basic `divDyn()` function.

```
# raw patterns
ddStages <- divDyn(dat, bin="stg", tax="clgen")
```

This function call produces a `data.frame` class object, with the calculated metrics as columns. The variables are explained in the function help file `?divDyn`. Because we have been using positive integer bin numbers, the indices of rows in this table match the bin identifier numbers. Therefore, the corresponding age values are in the same index rows in the `bins` table. To visualize the results, we need to plot the timescale first with

the `tsplot()` function (otherwise it is difficult to match results with time intervals visually) and then we can use `lines()` to show the variable in question.

```
tsplot(stages, boxes="sys", shading="sys", xlim=4:95, ylim=c(0,2800),
  ylab="Richness (raw data)", xlab="Age (Ma)")
lines(stages$mid, ddStages$divCSIB, col="blue", lwd=2)
lines(stages$mid, ddStages$divSIB, col="#00BB33", lwd=1)
legend("top", inset=c(0.01,0.01), legend=c("raw SIB", "corrected SIB"),
  col=c("#00BB33", "blue"), lwd=c(1,2), bg="white", cex=1)
```



Sampling-standardized values can be calculated with a single function called `subsample()`.

```
sqsStagesPlot <- subsample(dat, bin="stg", tax="clgen", coll="collection_no", q=0.7,
  iter=100, ref="reference_no",singleton="ref", type="sqs", duplicates=FALSE,
  excludeDominant=TRUE, largestColl =TRUE, output="dist", na.rm=TRUE)
```

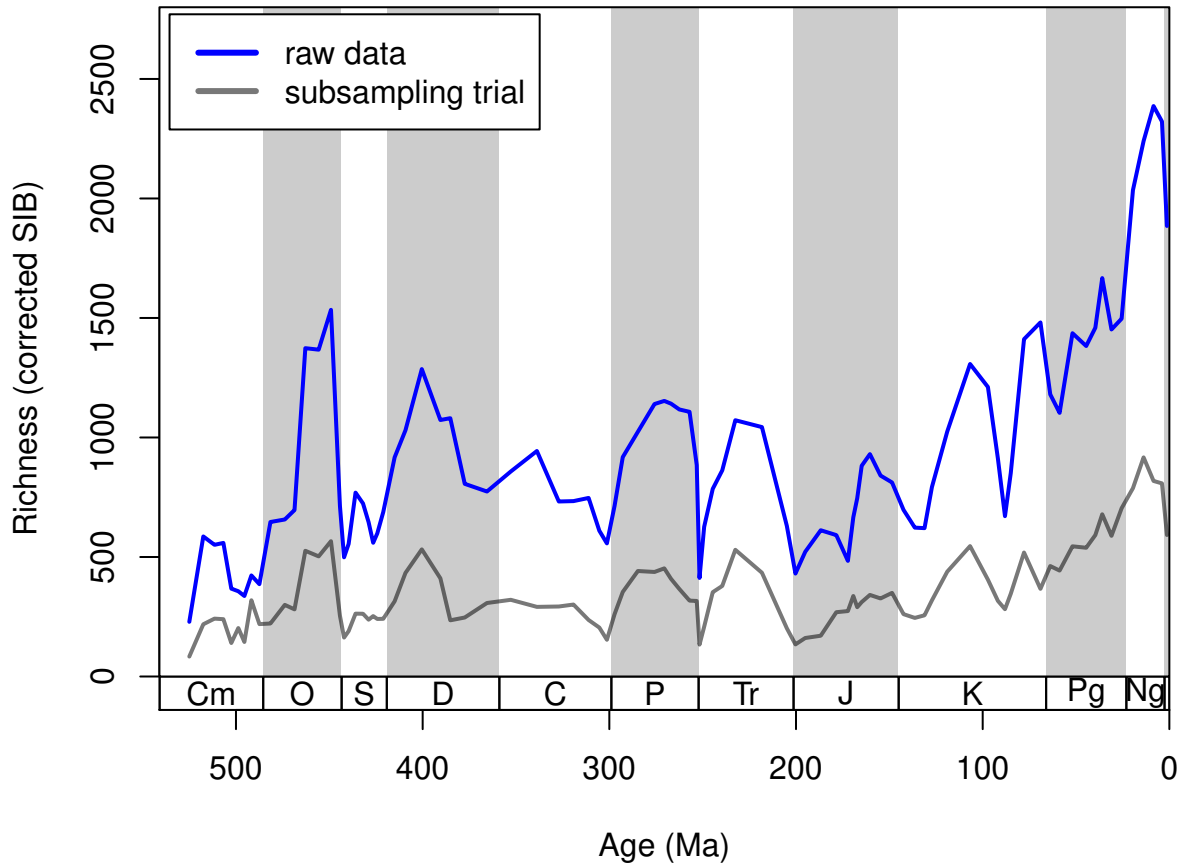
The function above is configured to use the 'Shareholder Quorum Subsampling' (Alroy, 2010a, 2010b) or 'coverage-based rarefaction' (Chao and Jost, 2012) method (`type="sqs"`) that subsamples the data down to an even level of sample coverage (Good, 1953). The former name is usually used to refer to the algorithmic version of the method; latter one to refer to the analytical version. We only implemented the algorithmic version as the complexities of paleontological data make the analytical version less useful in this context. The function is configured to use a reference-based 'singleton' treatment (`singleton="ref"`) for overall sampling

correction, excluding dominant taxa from all calculations involving frequencies (`excludeDominant=TRUE`) and with the separate treatment of the largest collection in the time slice (`largestColl=TRUE`), as indicated by Alroy (2010a). Setting `output` to `"dist"` makes the function return the results of individual subsampling trials. Please take a look at the help files of the functions `?subsample` or `?subtrialSQS` if you want to know more.

You can compare one trial result (in this case the 51st) with the original time series with the following code:

```
tsplot(stages, boxes="sys", shading="sys", xlim=4:95, ylim=c(0,2800),
  ylab="Richness (corrected SIB)" , xlab="Age (Ma)")

lines(stages$mid, ddStages$divCSIB, col="blue", lwd=2)
lines(stages$mid, sqsStagesPlot$divCSIB[ ,51], col="#00000088", lwd=2)
legend("topleft", bg="white", legend=c("raw data", "subsampling trial"),
  col=c("blue", "#00000088"), lwd=3, inset=c(0.01, 0.01))
```

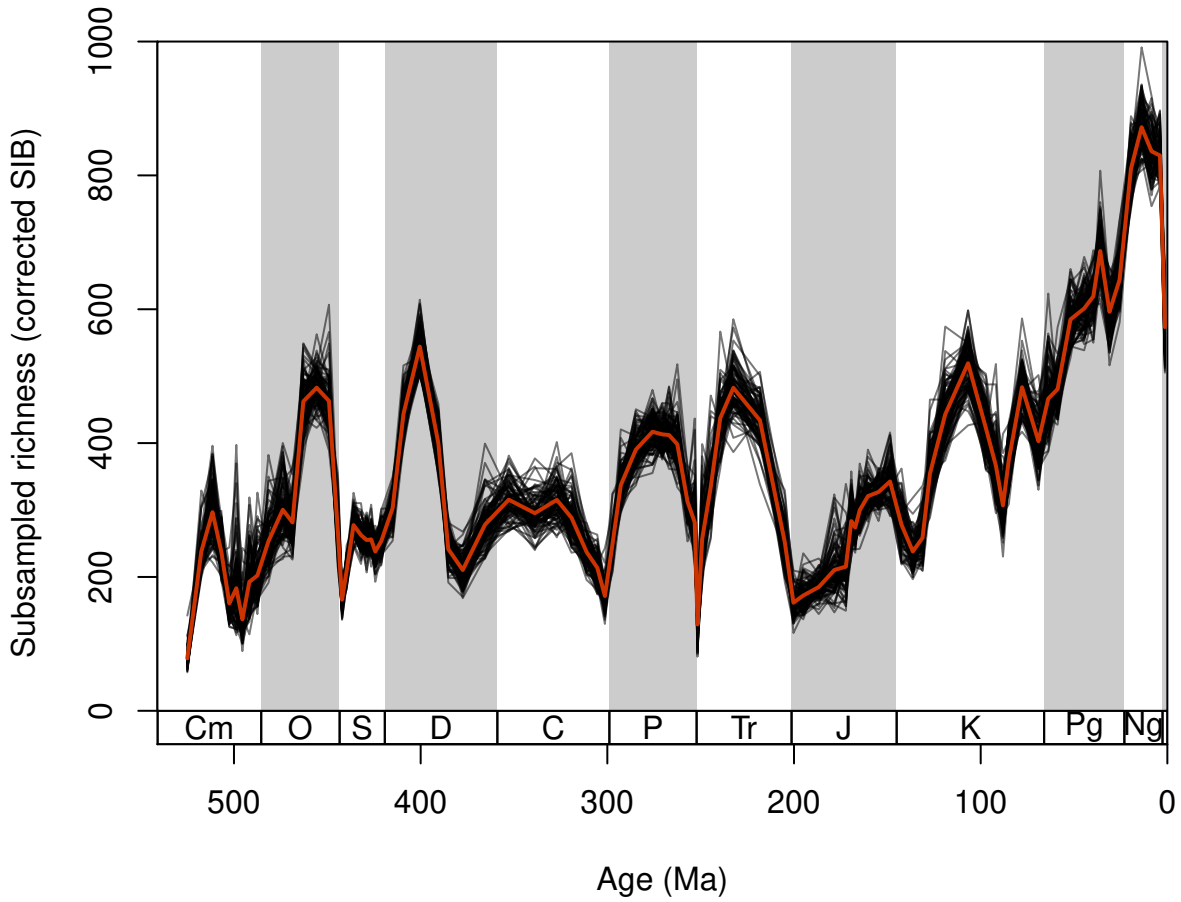


You can either let the function average the trial results by setting the `output` argument accordingly, or you can do it yourself. Here are the results of the 100 trials calculated above.

```
# only plot
tsplot(stages, boxes="sys", shading="sys", xlim=4:95, ylim=c(0,1000),
  ylab="Subsampled richness (corrected SIB)", xlab="Age (Ma)")
```

```
# loop through all trial results
for(i in 1:ncol(sqsStagesPlot$divCSIB)){
  lines(stages$mid, sqsStagesPlot$divCSIB[,i], col="#00000088")
}

# the mean of the trial results
meanRes <- apply(sqsStagesPlot$divCSIB, 1, mean, na.rm=T)
lines(stages$mid, meanRes, col="#CC3300", lwd=2)
```



5. Analyzing the taxonomic rates

This study focuses on turnover rates rather than richness estimates. The prototypes for these are the analyses presented by Alroy (2008). Here we provide the detailed analytical code to reproduce these using the per capita rates (Foote, 1999). These are the most used in the literature, even though they were critiqued by Alroy (2014) for their accuracy. The stratigraphic resolution in this is example was at the level of stages. All necessary functions are included in the `phanDyn.R` file, which is deposited on GitHub.

```
# load the necessary functions
source("https://github.com/divDyn/ddPhanero/raw/master/scripts/1.0.1/phanDyn.R")
```

5.1. Questions addressed with original rate values

The analyses in the following part of this section can be repeated with any set of extinction rate, origination rate and richness time series. To prepare the generalized implementation of the following code in a function (Section 5.3), the necessary variables are renamed so the same analyses can be rerun on different series if they are named appropriately. Variables from the timescale tables are also necessary.

```
# the name of the overall data frame (already created)
ext <- ddStages[ , "extPC"] # extinction rates
ori <- ddStages[ , "oriPC"] # origination rates
div <- ddStages[ , "divCSIB"] # diversity (richness) series
dur <- stages[ , "dur"] # durations of time intervals
age <- stages[ , "mid"] # time interal midpoints
name <- stages[ , "name"] # names of the time intervals
```

A. Is the pulsed model of turnover supported by the instantaneous rates?

The taxonomic rates currently implemented in the package converge on the per capita rates (Foote, 1999) as sampling completeness approaches 1. The original definition of the per capita rates expresses the intensity of turnover as per lineage-myr, which means that the log proportions are divided (normalized) by the durations of the time intervals. Implicit in these equations is that taxonomic turnover happened continuously in the bin, which has been suggested to be an invalid assumption, at least for extinctions (known as the pulsed model: Foote, 2005). If a 'pulsed' model is correct, the normalization of rates by bin duration is not only unnecessary but plain wrong. A 'pulsed' model is supported by the absence of correlations between turnover rates and bin durations (Alroy, 2008). With normalization a negative correlation between rate and duration may arise (Alroy, 2008). Rates calculated this way are sometimes characterized as 'instantaneous'.

Using the `pulseCont()` function we prepared tests for correlations between interval durations and rate values with and without the normalization. It uses tests of Spearman's rank-order correlations to assess these relationships. The function can be found in the `phanDyn.R` file, and it takes a rate series and vector of bin durations as arguments.

```
pulseCont(ext, dur, alpha=0.01)
```

```
## $est
## not-normalized     normalized
##      0.2322265     -0.4496332
##
## $p
## not-normalized     normalized
##   2.762984e-02   8.756357e-06
##
## $sig
## normalized
## -0.4496332
```

The `est` element of the output list contains the correlation coefficients, `p` the p-values, and the `sig` element indicates significant results at the $p < 0.01$ level. As only the normalized rates are significantly correlated with bin durations, it is likely that normalization strengthens the association between the two variables. This suggests that for extinctions (based on the raw, stage-level per capita rates) the 'pulsed' model turnover is more appropriate.

B. Are the taxonomic rates declining?

The decline of taxonomic turnover rates is one of the most robust findings of paleobiology and has been reported many times in the literature (e.g. Raup and Sepkoski, 1982; Sepkoski, 1998, Bambach et al, 2004; Alroy, 2008). Therefore, we expect both origination and extinction rates to decline over time. In general, we have to be cautious about the earliest and latest parts of the time series. First, these rates cannot be estimated for the first and last bins (some even not for the second and the second last), as it is impossible to count taxa with certain temporal patterns of occurrences in these intervals (e.g. three-timers). The so-called edge-effects (Foote 2000) also bias the per-capita rates systematically. The stratigraphy of the Cambrian and the Ordovician intervals remains problematic, while we also know that higher taxonomic entities originated in these intervals that separate them from the rest of the Phanerozoic. Therefore, we decided to copy the analytical parameters of Alroy (2008), and the declining pattern was also assessed separately for the post-Ordovician interval. The following intervals were applied in the assessment:

```
# total decline interval in Ma
    maxAgeDecline <- 540
    minAgeDecline <- 20
# mid dates of post-Ordovician time bins (both time scales) are younger than
    postDate <- 443
```

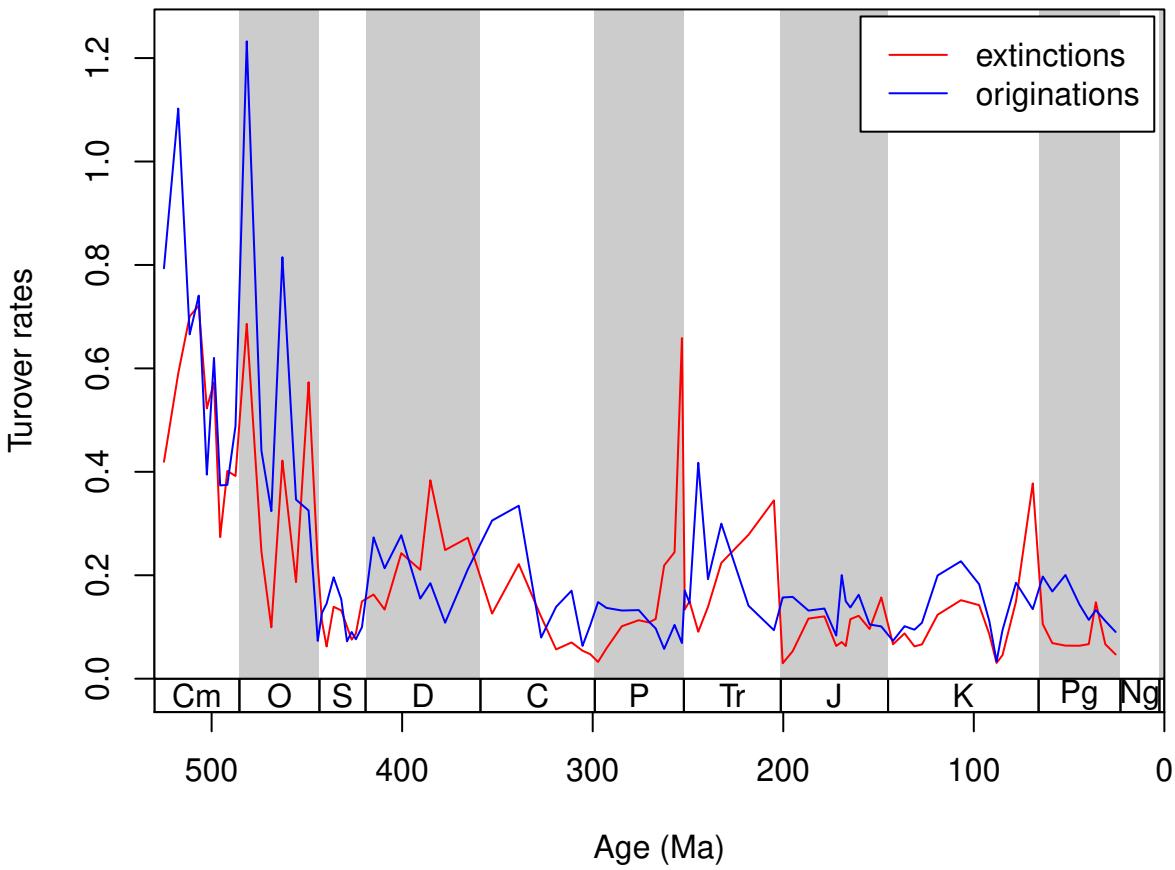
To retain the match between the bin numbers in the time scale table and the indices of the results, we defined logical vectors that replace values that are out of this interval with `NA` entries.

```
# logical vectors indicating what is necessary
# Cambrian included
indNAlong <- !(age<=maxAgeDecline & age>=minAgeDecline)
# post Ordovician
indNApostord <- !(age<=postDate & age>= minAgeDecline)
```

You can relate these values to the estimated ages in `bins` and `stages` objects. It is good practice to visualize the variables before carrying out the correlation tests.

```
# actual variables used in the correlation tests:
  extVarLong <- ext
  oriVarLong <- ori
  extVarLong[indNAlong] <- NA
  oriVarLong[indNAlong] <- NA

# plotting
  # maximum y value
  nMaxRate<-max(c(extVarLong, oriVarLong), na.rm=T)
  # actual plot
  tsplot(stages,  boxes="sys", shading="sys", ylim=c(0, nMaxRate*1.05),
    ylab="Turover rates", xlim=c(530,0), xlab="Age (Ma)")
  lines(age, extVarLong, col="red")
  lines(age, oriVarLong, col="blue")
  legend("topright", legend=c("extinctions", "originations"), col=c("red", "blue"),
    lwd=c(1,1), bg="white", inset=c(0.01, 0.01))
```



Some patterns of a decline are evident from viewing these series, but we need to test them. We can create a 2 by 4 matrix to hold results of correlations against time (coefficients and p-values) that will be calculated for each of the series.

```
# create a table from the decline information
decMat <- matrix(NA, ncol=2, nrow=4)
  colnames(decMat)<- c("correlation","p-value")
  rownames(decMat)<- c("extinction", "post-Ordovician extinction",
  "origination", "post-Ordovician origination")
```

First, the total declines (`extVarLong` and `oriVarLong`) are assessed.

```
# is the extinction rate declining
  extDecline <- cor.test(age, extVarLong, method="spearman")
  decMat[1, 1] <- extDecline$estimate
  decMat[1, 2] <- extDecline$p.value

# is the origination rate declining
  oriDecline <- cor.test(age, oriVarLong, method="spearman")
  decMat[3, 1] <- oriDecline$estimate
  decMat[3, 2] <- oriDecline$p.value
```

Then the post-Ordovician declines are assessed.

```
# extinction
  extPostOrd <- extVarLong
  extPostOrd[indNApostord] <- NA
  extDeclinePostOrd <- cor.test(age, extPostOrd, method="spearman")
  # save
  decMat[2, 1] <- extDeclinePostOrd$estimate
  decMat[2, 2] <- extDeclinePostOrd$p.value

# origination
  oriPostOrd <- oriVarLong
  oriPostOrd[indNApostord] <- NA
  oriDeclinePostOrd <- cor.test(age, oriPostOrd, method="spearman")
  # save
  decMat[4, 1] <- oriDeclinePostOrd$estimate
  decMat[4, 2] <- oriDeclinePostOrd$p.value
```

The results of the test above are in the `decMat` object, clearly suggesting patterns of an overall decline for both extinctions and originations, although this might not be true for the post-Ordovician parts of the series.

```
round(decMat, 4)
```

```
##                              correlation p-value
## extinction                        0.5377  0.0000
## post-Ordovician extinction        0.2439  0.0437
## origination                       0.4450  0.0000
## post-Ordovician origination       0.0606  0.6202
```

5.2. Detrending the series

The definition of the rate distribution, identification of outlying values and implementation of accurate tests of cross correlations between the time series assume that the series are stationary. To get closer to meeting this assumption, the rate series have to be detrended. There are multiple ways of fitting a trend to the time series. You can fit a polynomial function, model it as an ARIMA process, or apply a scatterplot smoother (e.g. LOESS) to the values.

Among these options, the last is the simplest one and it has the fewest assumptions about the underlying processes. This is a very useful property, as it decreases the chance of inappropriate fit when the same analytical code is run on different time series, or when long-term trends are not significant. This section starts by going through how this procedure can be applied to the time series of originations and extinctions without transforming the data (see section 5.3 for other options).

LOESS relies on a bandwidth parameter (or `span`) that describes how much smoothing should be applied to the series, which is usually considered to be an arbitrary parameter. Luckily, there are ways to search for an optimal setting. An easily applicable implementation is included in the `fANCOVA` package (Wang, 2010), which you can download from the CRAN servers with the following line of code:

```
install.packages("fANCOVA")
```

The relevant function requires the time series to be free of missing values with the corresponding ages. You can get rid of these in a structured way by executing these commands:

```
# extinctions
   extMiss <- !is.na(extVarLong)
   transExtNoNA <- extVarLong[extMiss]
   ageExt <- age[extMiss]
```

```
# originations
   oriMiss <- !is.na(oriVarLong)
   transOriNoNA <- oriVarLong[oriMiss]
   ageOri <- age[oriMiss]
```

After running this snippet, `transExtNoNA` includes the extinction rate values and `ageExt` has the corresponding ages. The same applies to `transOriNoNA` and `ageOri` but for originations. Doing the actual smoothing is just a single step with `fANCOVA`.

```
# the models
extModel <- fANCOVA::loess.as(ageExt, transExtNoNA, degree = 1,
  criterion = "aicc", user.span = NULL, plot = FALSE)
```

```
## fANCOVA 0.5-1 loaded
```

```
oriModel <- fANCOVA::loess.as(ageOri, transOriNoNA, degree = 1,
  criterion = "aicc", user.span = NULL, plot = FALSE)
```

These settings will make the function use corrected AICs to define the optimal smoothing of the curves. To get the actual values, predictions must be calculated from these model objects. Running the `predict()` function on them will output the predicted values (i.e. the trend).

```
# predictions for extinctions
predict(extModel)
```

```
##  [1] 0.62325673 0.58552102 0.55534469 0.53144031 0.51002727 0.49110464
##  [7] 0.47483598 0.45634028 0.43531586 0.40640234 0.36769122 0.34072877
## [13] 0.31075909 0.27532159 0.24552070 0.22642213 0.21966458 0.21433174
## [19] 0.20668676 0.20130734 0.20011972 0.19830945 0.19603717 0.19375088
## [25] 0.19213373 0.19296649 0.19616201 0.19764743 0.19661934 0.19295288
## [31] 0.18260631 0.16567759 0.14814499 0.13971782 0.13719641 0.13714356
## [37] 0.13864752 0.13658665 0.13189898 0.13199185 0.13942305 0.14635715
## [43] 0.15321349 0.15858871 0.16406136 0.17007275 0.17457318 0.17648189
## [49] 0.17944167 0.18317396 0.18487283 0.18352600 0.17043911 0.15402125
## [55] 0.14866778 0.14264883 0.13316227 0.12340286 0.11629935 0.11334492
## [61] 0.11143325 0.10952288 0.10668156 0.10389752 0.10186044 0.10096756
## [67] 0.10110914 0.10196667 0.10303353 0.10505987 0.10691976 0.10842763
## [73] 0.10786080 0.10669645 0.10626053 0.10587401 0.10443185 0.10333510
## [79] 0.10229616 0.10111632 0.09969432 0.09856992 0.09783309 0.09659208
## [85] 0.09493763
```

There is a problem with this result, however, namely that certain predictions where the original time series had `NA` values are not present. This leads to misalignment with the other time series and numerous future problems. Therefore, it is better practice to coerce the output of missing values in the prediction, by specifying the `x` (time) coordinates for the predictions. These are in the original `age` vector and can be used this way:

```
predExt <- predict(extModel, newdata=data.frame(x=age)) # extinctions
predOri <- predict(oriModel, newdata=data.frame(x=age)) # originations
```

The resulting objects have the same number of values as the original series.

```
length(predExt)
```

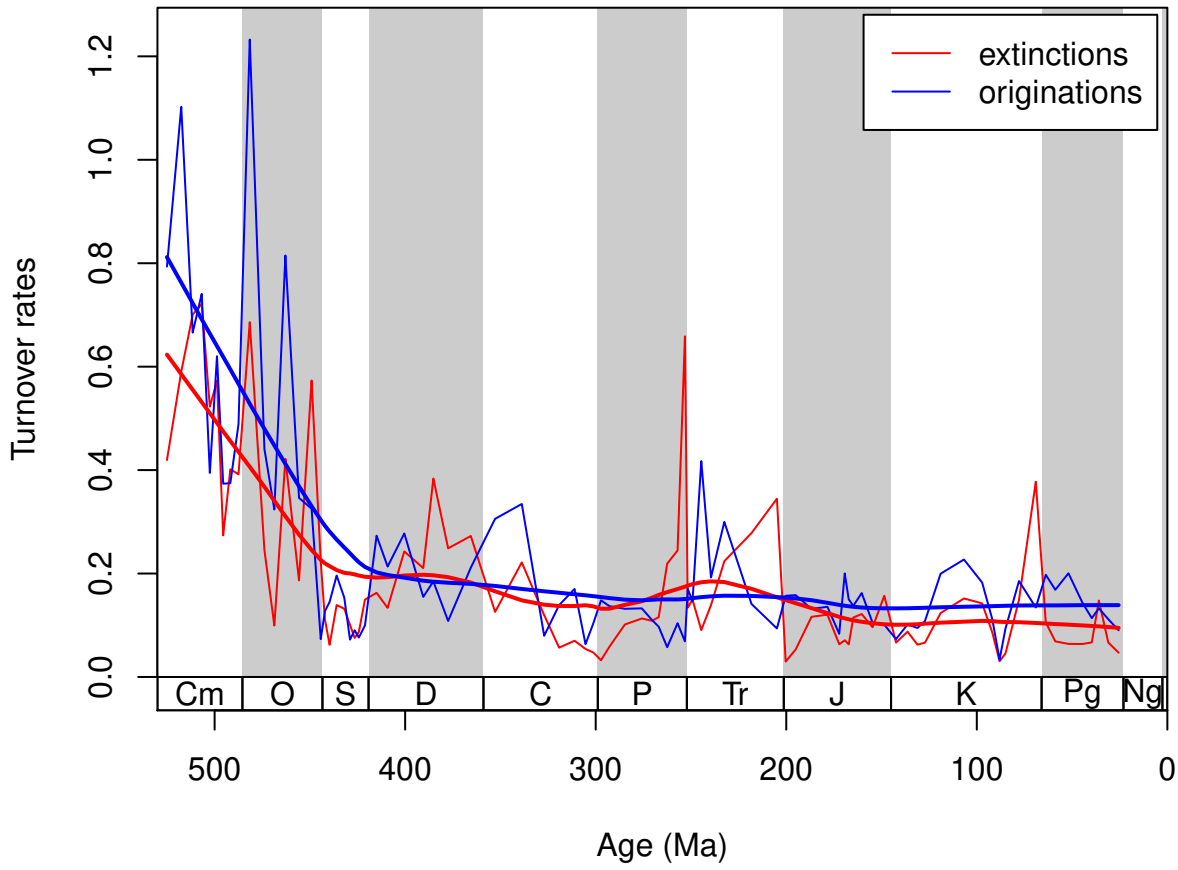
```
## [1] 95
```

```
length(predOri)
```

```
## [1] 95
```

You can draw the predicted values with `lines()`.

```
lines(age, predExt,col="red", lwd=2) # extinctions
lines(age, predOri,col="blue", lwd=2) # originations
```



The detrended values are then the residuals with multiplicative decomposition.

```
detExt <- extVarLong/predExt
detOri <- oriVarLong/predOri
```

Following Alroy (2008), only the post-Cambrian values were used in further analyses, as the Cambrian rates appear to be unusually high in all cases.

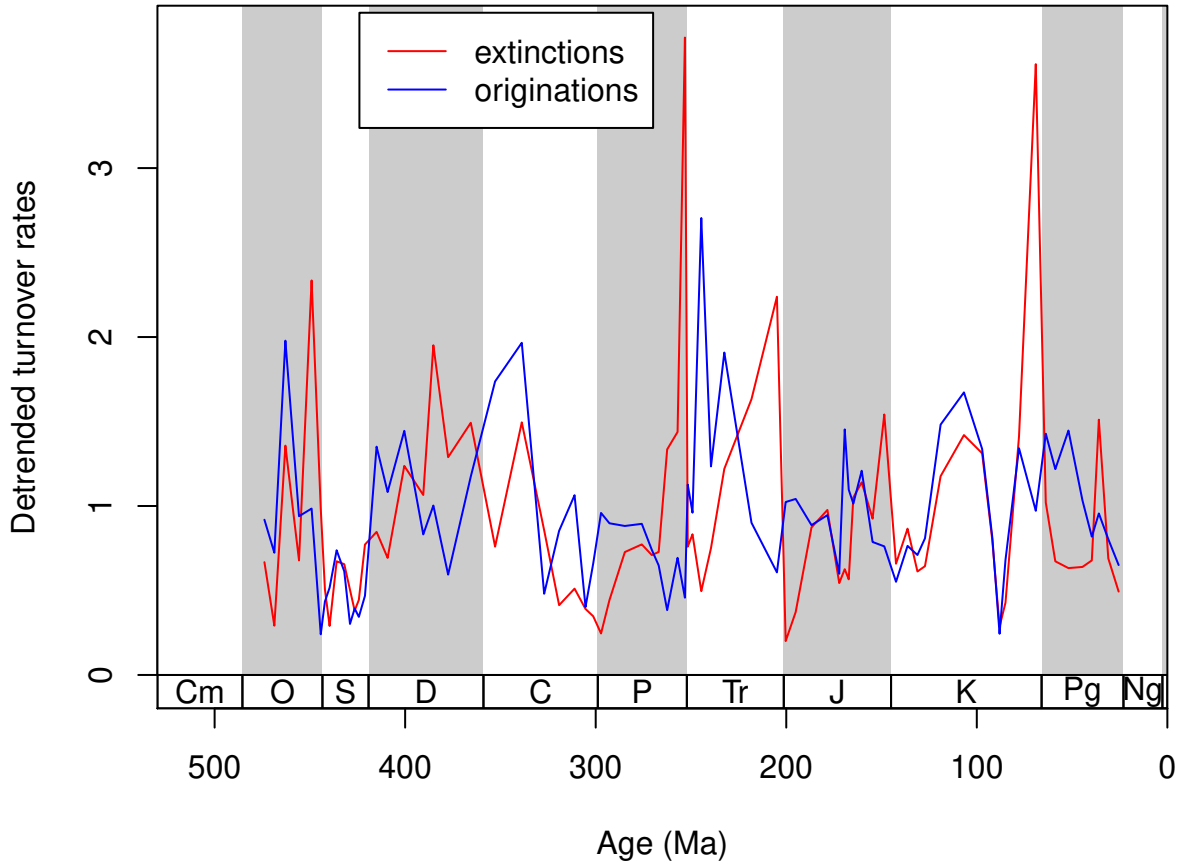
```
# interval for detrending in ma
    maxAgeDetrend <- 475
    minAgeDetrend <- 20
# indices and variables
indNAshort<- !(age<=maxAgeDetrend & age>=minAgeDetrend)

# recreate the variables
  detExtShort <- detExt
  detOriShort <- detOri
```

```
  detExtShort[indNAshort] <- NA
  detOriShort[indNAshort] <- NA
```

You can plot the residuals with `lines()`.

```
# for setting the y-axis range
detMax <- max(c(detExtShort,detExtShort), na.rm=T)
# plot
tsplot(stages,  boxes="sys", shading="sys", ylim=c(0,detMax*1.05),
  ylab="Detrended turnover rates", xlim=c(530,0), xlab="Age (Ma)")
lines(age, detExtShort, col="red" ) # extinctions
lines(age, detOriShort, col="blue" ) # originations
legend("topleft", legend=c("extinctions", "originations"), col=c("red", "blue"),
  lwd=c(1,1), bg="white", inset=c(0.2, 0.01))
```

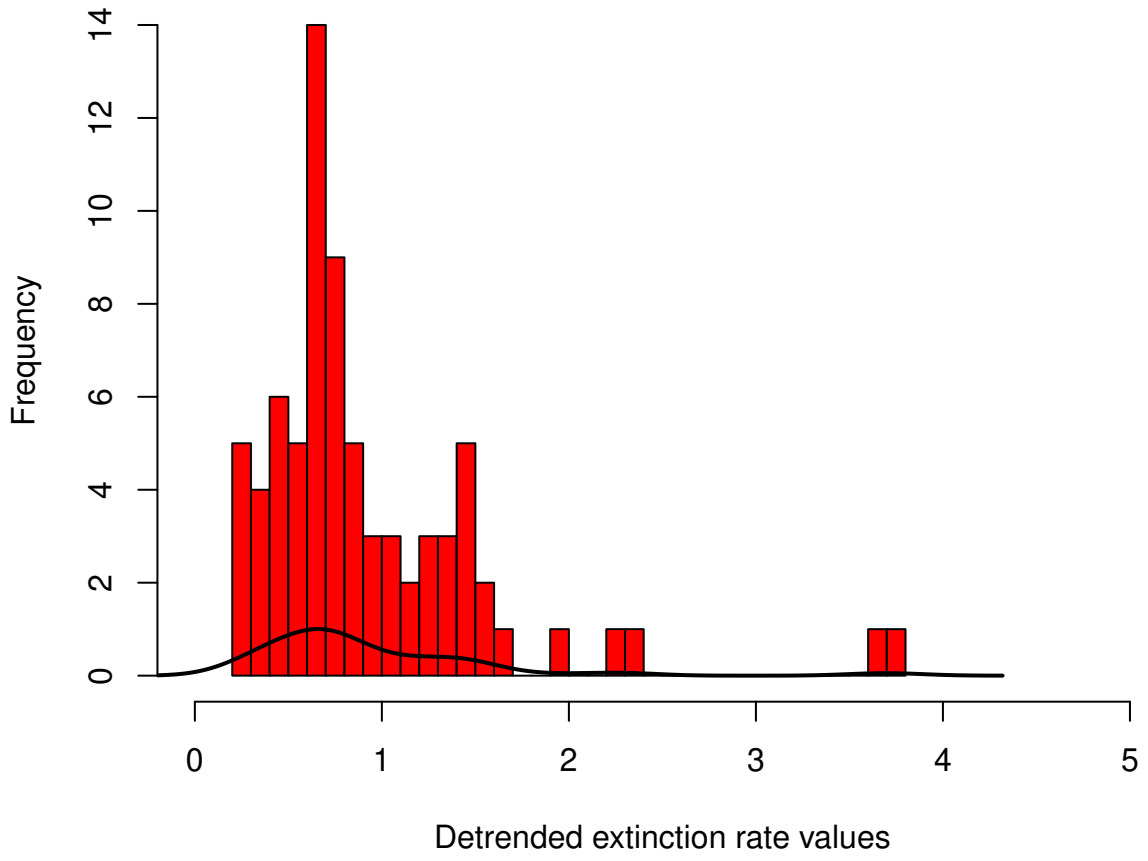


As mentioned before, there are other ways to do the detrending of the series, some of which we also implemented (see section 5.3 for options). You are invited to inspect these within the `analyzedMetrics()` function that can be found in the `phanDyn.R` file.

C. Is the distribution of rates lognormal?

Mass extinctions have always been diagnosed by analyzing the distributions of the rates. Originally, these were just compared to the point estimation confidence-interval of a linear decline model (Raup and Sepkoski, 1982), which was already criticized back then (Quinn, 1983). Bambach et al. (2004) fitted a LOESS model to the extinction rates, and analyzed the distribution of its residuals (similar to what we just did, assuming a local background process). Alroy (2008) suggested detrending the rates with exponential functions. He analyzed the distribution of the rates first and then pointed to potential outliers. Here is the distribution of the detrended extinction rate series with a kernel density estimator:

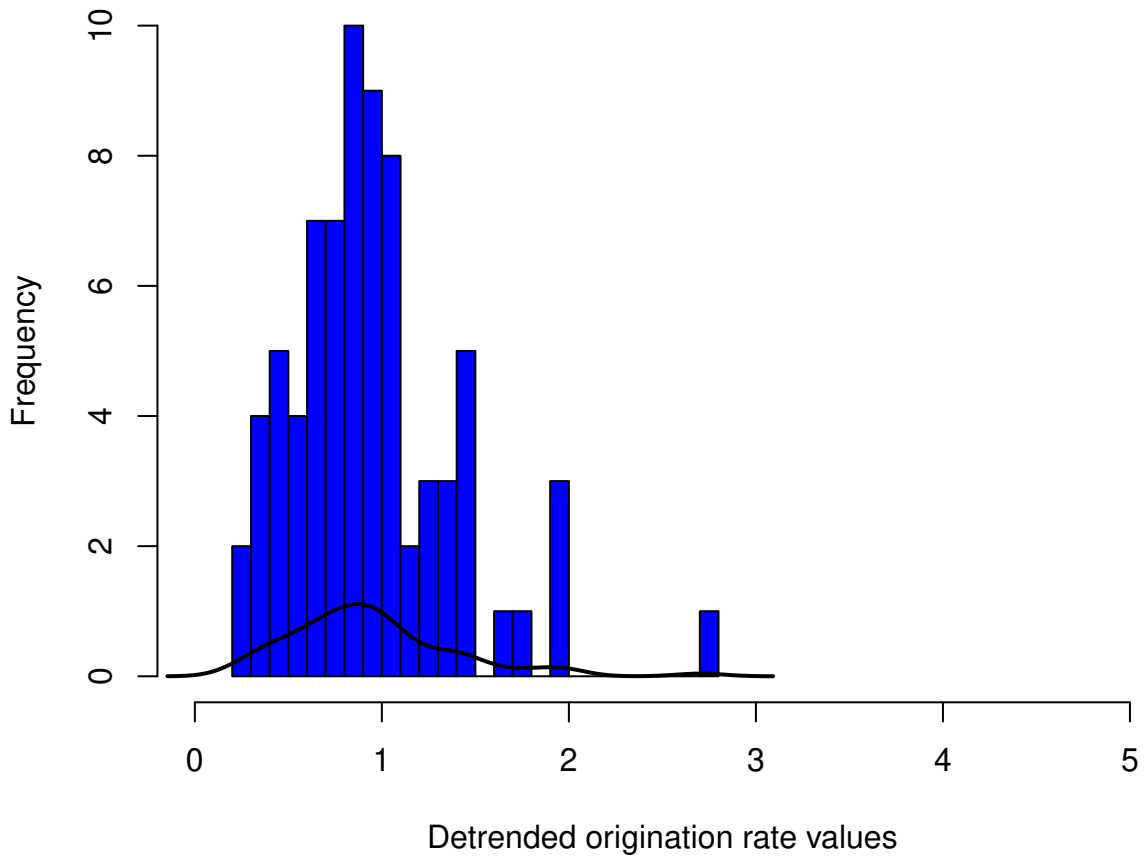
```
# extinctions histogram
hist(detExtShort , breaks=30, xlim=c(0,5), col="red",
  xlab="Detrended extinction rate values", main="")
# kernel density estimator
den <- density(detExtShort [!is.na(detExtShort )])
lines(den$x, den$y, lwd=2)
```



And here is the same plot for originations.

```
# histogram
hist(detOriShort, breaks=30, xlim=c(0, 5), col="blue",
```

```
  xlab="Detrended origination rate values", main="")
# kernel density estimator
den<-density(detOriShort[!is.na(detOriShort)])
lines(den$x, den$y, lwd=2)
```



As the extinction rate distributions tend to be right-skewed and they are 0-bounded, analyzing the logarithms of the rates, or their square roots make more sense. The question behind the analysis of rate distributions is whether we can distinguish between two processes of extinction in terms of mass extinction episodes and background extinctions, or whether the two just at different positions along a spectrum. A lognormal distribution suggests that there is only a difference in magnitude between these intervals, which has to be assessed. Therefore, the function checks whether the rates can come from a lognormal distribution or not, which is implemented by using Shapiro-Wilk tests.

```
# are logged data normal? - extinction
extVar <- log(detExtShort)
extVar[is.infinite(extVar)] <- NA # omit infinites
pShap <- shapiro.test(extVar)$p.value
names(pShap)[1] <- "ext"

# are logged data normal? - origination
```

```
oriVar <- log(detOriShort)
oriVar[is.infinite(oriVar)] <- NA   # omit infinites
pShap <- c(pShap,shapiro.test(oriVar)$p.value)
names(pShap)[2] <- "ori"

# the p-values of the tests:
pShap
```

```
##       ext       ori
## 0.6204032 0.3704712
```

These results indicate that at the stage-level resolution and using the per capita rates (Foote 1999), both origination and extinction rate series have lognormal distributions, suggesting that mass extinction intervals are not qualitatively different from background turnover.

D. Which time slices are outliers from the distribution?

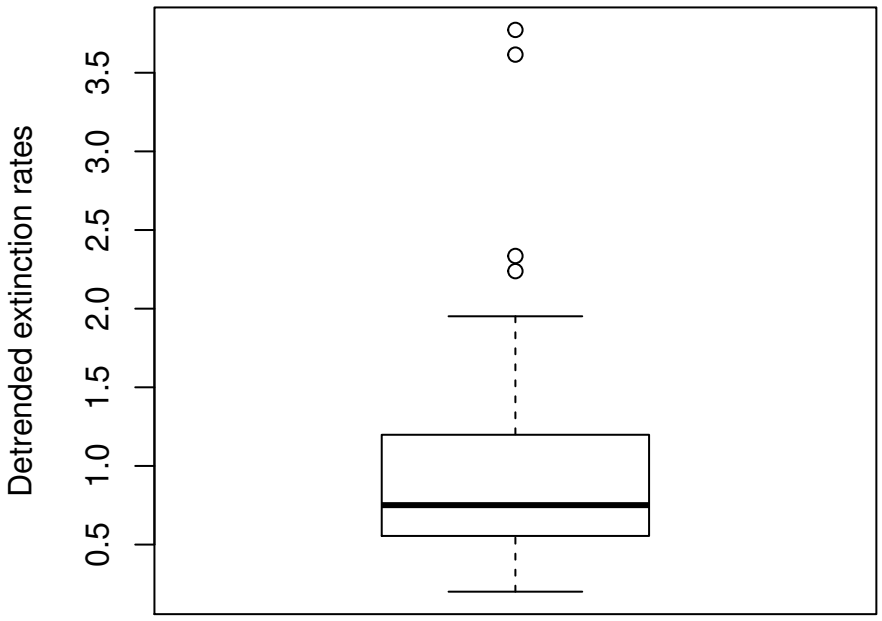
The exact definition of a 'mass extinction interval' has become somewhat blurry over the years. As rates fit a lognormal distribution quite well, it appears likely that background and mass extinction intervals are similar in qualitative terms, they are just at different positions along a spectrum. Even if this is the case, cataclysmic events (i.e. mass extinctions) will fall in the upper tail of the extinction rate distribution, which we can separate with non-parametric methods. The literature generally agrees that the largest 'mass extinction' was at the Permian/Triassic boundary, which we can reassess with a following line of code.

```
largest <- name[which(max(detExtShort,  na.rm=T)==detExtShort)]
largest
```

```
## [1] "Changhsingian"
```

Which is the latest Permian interval, and is indistinguishable from the end-Permian mass extinction. We can also identify and contrast potential outliers with R's `boxplot()` function.

```
boxp <- boxplot(detExtShort, ylab="Detrended extinction rates")
```




```
outliers <- name[as.numeric(names(boxp$out))]
outliers
```

```
## [1] "Katian"        "Changhsingian" "Rhaetian"      "Maastrichtian"
```

This suggests that without making any assumptions about the extinction rate distribution, four mass extinctions can be identified in the raw, stage-resolution data and the per capita rates. Among these, the end-Permian (Changhsingian), end-Triassic (Rhaetian) and end-Cretaceous (Maastrichtian) mass extinctions ('Big Three' in Alroy, 2008) are the most studied and usually represent the highest values in detrended extinction rate series. The presences of these three events are assessed with different rate calculation methods using the function presented in section 5.3. The Katian value is most likely associated with the end-Ordovician event.

E. Can we find traces of equilibrial dynamics in the series?

The quest to detect equilibrial patterns of diversity dynamics is rooted in island biogeography. Equilibrial dynamics means that diversity is bounded, but it does not necessarily mean that carrying capacity is stable through time. It rather points to the fact that there is an attractor in the richness dimension, pulling diversity up, when it is relatively low, and depressing it when it is relatively high (Alroy, 2010b). Alroy (2008) argued that equilibrial dynamics should manifest in three cross-correlations:

- Higher origination rates in time slice i should lead to higher diversity in time slice i+1.
- Higher diversities in time slice i in general should lead to higher extinction rates in time slice i+1

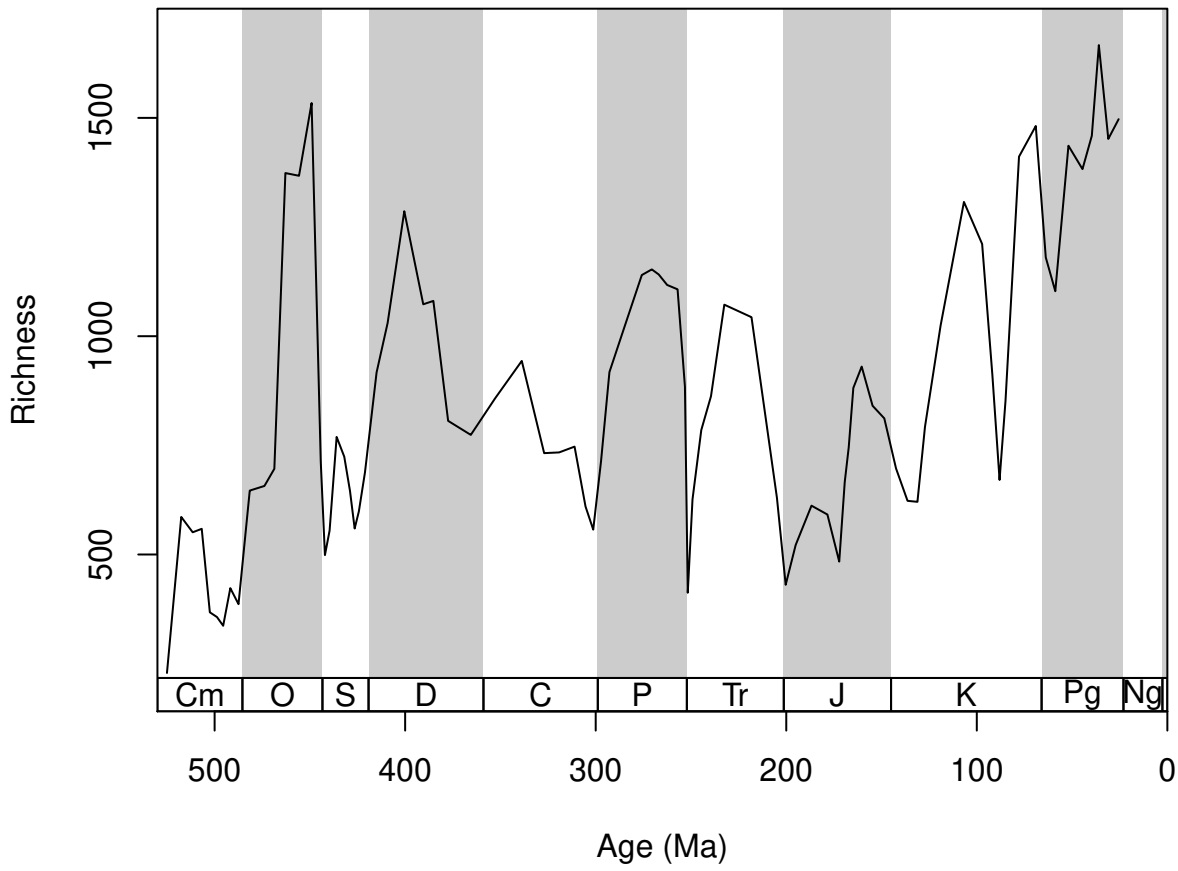
- Higher extinction rates in time slice i should lead to higher origination rate in slice i+1.

We have already detrended the turnover rates, but we still need to detrend the richness values. We follow the same basic procedure.

```
# select the same interval as for turnover rates
divVarLong <- div
divVarLong[indNAlong] <- NA
```

This can be plotted with:

```
# plotting the logged series
nMaxDiv <- max(divVarLong, na.rm=T)
nMinDiv <- min(divVarLong, na.rm=T)
tsplot(stages,  boxes="sys", shading="sys", ylim=c(nMinDiv*0.95, nMaxDiv*1.05),
  ylab="Richness", xlim=c(530,0), xlab="Age (Ma)")
lines(age, divVarLong, col="black")
```



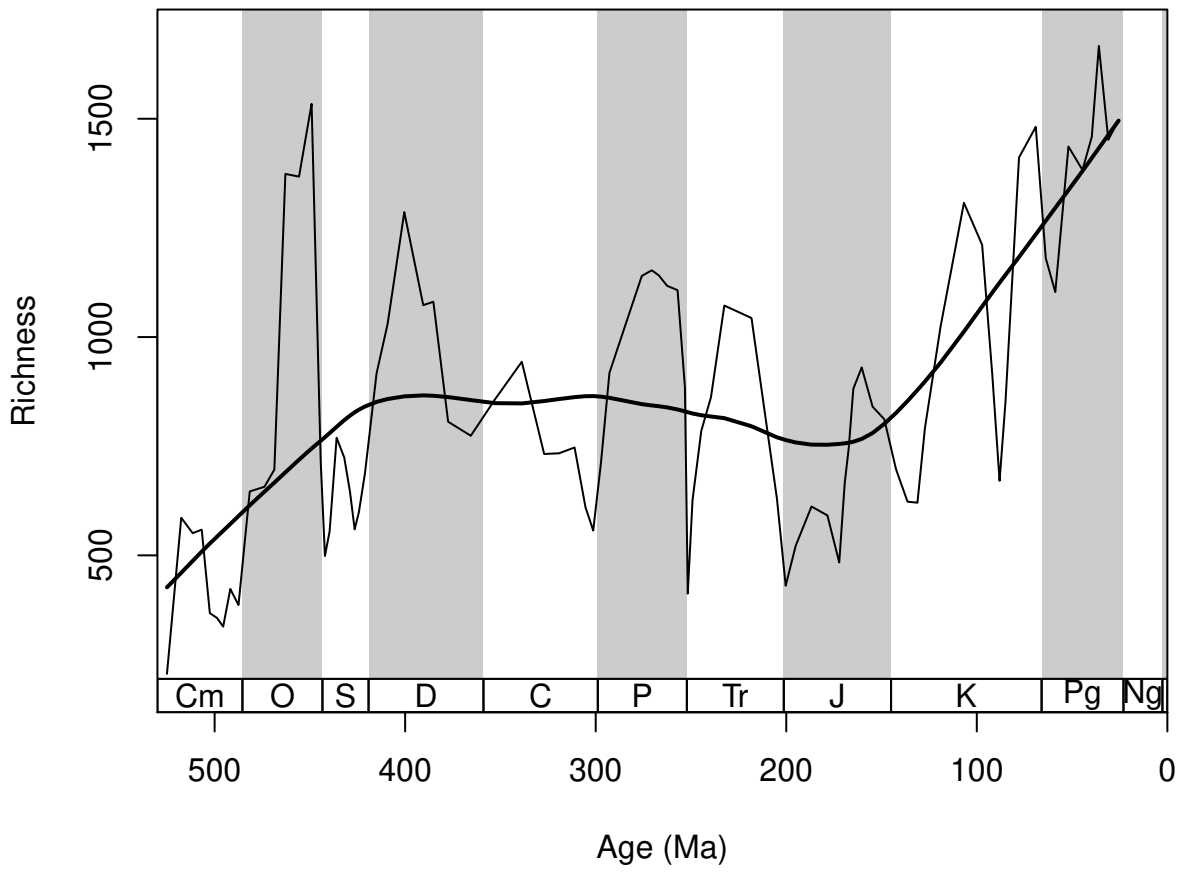
After this step, the LOESS regression is applied to the data.

```
divMiss <- !is.na(divVarLong)
  transDivNoNA <- divVarLong[divMiss]
  ageDiv <- age[divMiss]
```

```
divModel <- fANCOVA::loess.as(ageDiv, transDivNoNA, degree = 1,
  criterion = "aicc",  user.span = NULL, plot = FALSE)
```

The predictions of this model are then calculated.

```
# predicted
 transPredict <- predict(divModel, newdata=data.frame(x=age))
 lines(age, transPredict, lwd=2)
```



The residuals are taken and are rescaled to the original magnitude, using the mean of the original series.

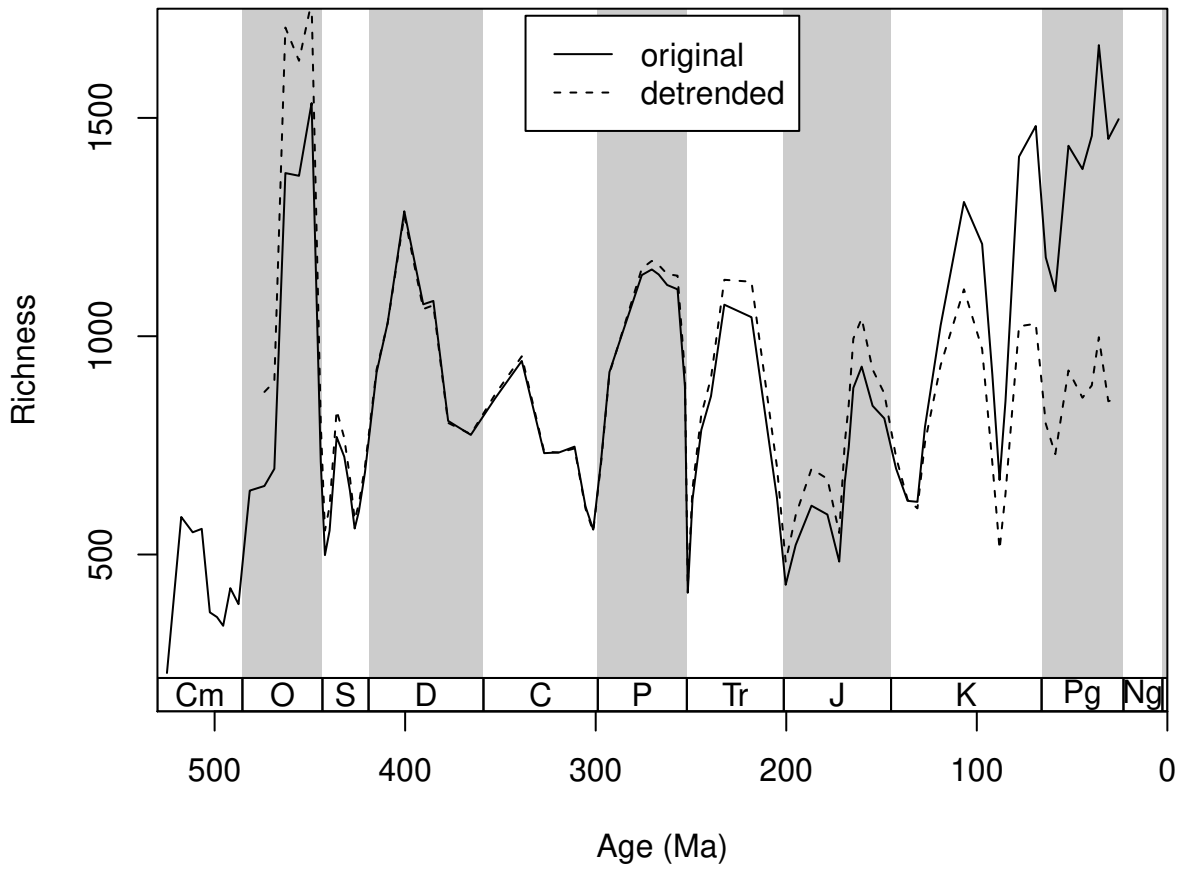
```
 transResid <- divVarLong/transPredict  # multiplicative decomposition
 detDiv<- mean(divVarLong, na.rm=T)*transResid # rescaling
```

The series was then subsetted to include only those values that are in the detrended turnover rate series.

```
# limit to the same part as the turnover
  detDivShort <- detDiv
  detDivShort[indNAshort] <- NA
```

The resulting series is comparable to the original series, but without the increasing trajectory.

```
# y ranges
tsplot(stages,  boxes="sys", shading="sys", ylim=c(nMinDiv*0.95, nMaxDiv*1.05),
  ylab="Richness", xlim=c(530,0), xlab="Age (Ma)")
lines(age, divVarLong, lty=1)
lines(age, detDivShort, col="black", lty=2)
legend("top", inset=c(0.01, 0.01), legend=c("original", "detrended"), lty=c(1,2),
  bg="white")
```



After the diversity series is detrended, another small function `dynamics()` is applied to calculate cross correlation patterns at the chosen lag between the detrended extinction rate, origination rate and diversity variables. You can find this in the `phanDyn.R` file. `ext`, `ori` and `div` are the original detrended series (extinction, origination and diversity, respectively), and `extPlus`, `oriPlus` and `divPlus` denote the shifted series. For instance, a positive correlation between `ext` and `oriPlus` indicates that high extinction rate values are usually followed by high origination rate values in the next bin. Correlations between `ext` and `extPlus` indicate autocorrelations. A single run of this function produces the following results, with the alpha level of `0.01`.

```
dyn <- dynamics(ori=detOriShort, ext=detExtShort, div=detDivShort, method="spearman",
  l=1, alpha=0.01)
# the $sig element returns significant components
```

```
dyn$sig
```

```
##       Var1    Var2  estimate            p
## 2      div divPlus 0.6658127 0.000000e+00
## 3      div     ori 0.3913798 5.679593e-04
## 5      div     ext 0.6021622 2.194412e-08
## 6      div extPlus 0.6178008 8.651692e-09
## 9  divPlus     ori 0.3816512 8.619459e-04
## 10 divPlus oriPlus 0.3913798 5.679593e-04
## 12 divPlus extPlus 0.6021622 2.194412e-08
## 16     ori oriPlus 0.3995113 4.672974e-04
## 30     ext extPlus 0.4104110 3.164707e-04
```

5.3. Single analysis function

After the basic analytical script was completed to answer the questions in sections 5.1 and 5.2 (above) a single function was prepared that takes any origination rate, extinction rate and richness time series and performs the analyses that we presented above. This function `analyzeMetrics()` can be found in the `phanDyn.R` file. The function requires a single `data.frame` object that has to include variables with the three time series, the bin durations, bin midpoint ages and names. All this information can be compiled by concatenating the time scale object and the output `data.frame` of the `divDyn()` function.

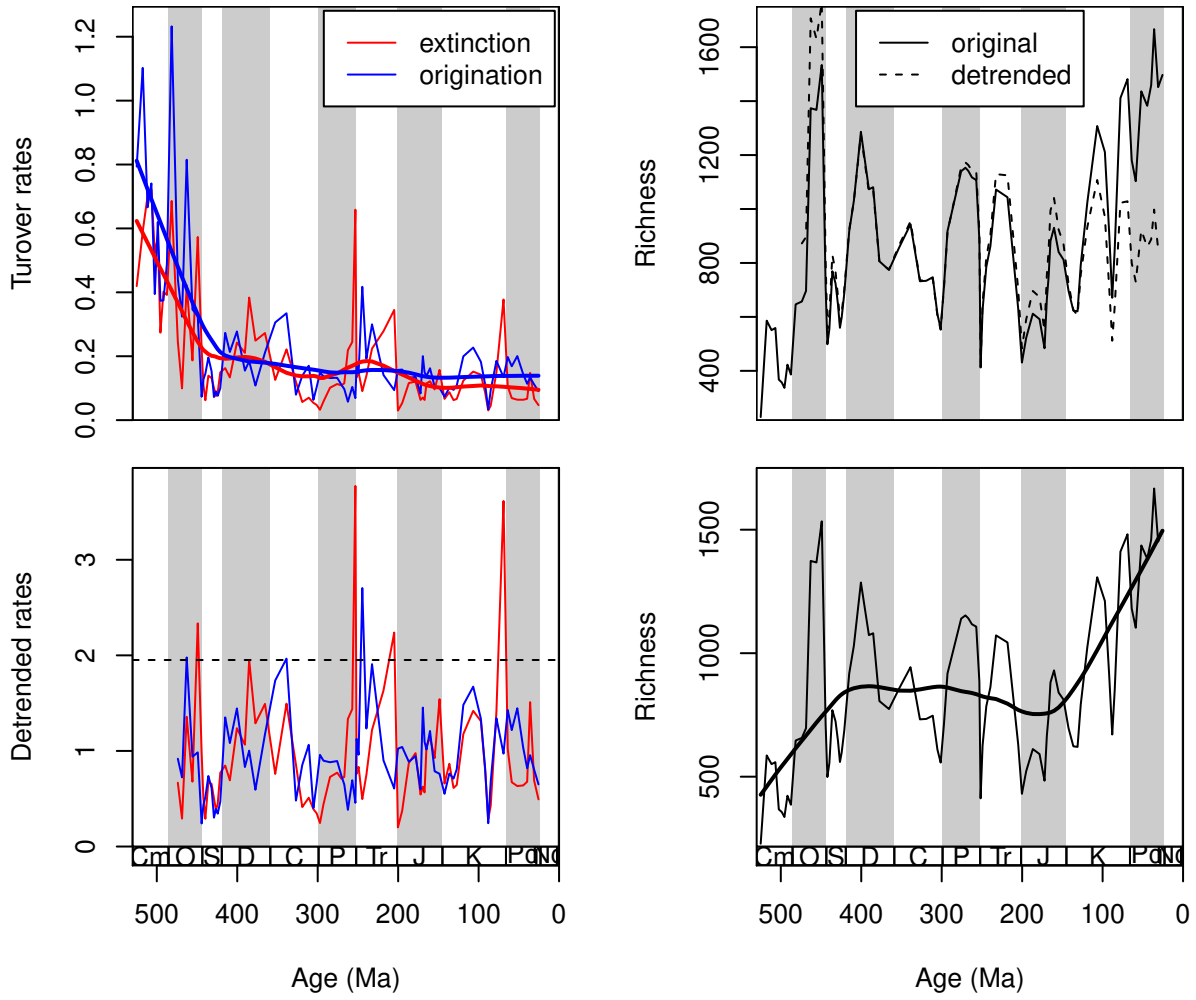
```
metrics <- cbind(ddStages, stages)
```

The arguments `ext` (extinction), `ori` (origination), `div` (richness/diversity), `age`, `dur` (duration) and `name` refer to the column names of the required variables. The function was implemented to return whether the 'pulsed' or 'continuous' model is supported, and to apply normalization with bin durations, if it is chosen so by the user. Normalized rates are only used in further analyses, when a correlation between rates and bin durations was evident in the non-normalized data and when the argument `normalize` is set to `TRUE`. In case both correlation tests are significant, the pulsed model is used as the default. Setting this argument to `FALSE` will coerce the calculations to use the input time series, using the 'pulsed' model throughout.

As mentioned in section 5.2, there are multiple ways to detrend the series. We implemented some of these methods, the procedure can be configured by adjusting the `detrend`, `transform` and `additive` arguments of the `analyzedMetrics()` function. The `detrend` argument specifies a switch between the different detrending options. Setting this to `"loess"` will run the detrending process shown in section 5.2 based on LOESS regression with the `fANCOVA` package (Wang, 2010). The option `"linear"` will fit a linear model to all three series. Setting the argument to `"arima"` will install and use the `forecast` package (Hyndman and Khandakar, 2008) to fit ARIMA models with the `auto.arima()` function. The argument `transform` specifies which transformation should be applied to the time series before the decomposition takes place. Setting this argument to `FALSE` will use the original series, `"log"` applies logarithm transformation and then exponentiation, `"sqrt"` applies square root transformation and then the squaring of the predicted and detrended values. The logical argument `additive` indicates whether additive or multiplicative decomposition is used to remove the trend.

The function also has additional arguments that enable the plotting of time series (`plot=TRUE`) and the output of messages and warnings (`feedback=TRUE`). If `plot` is set to `TRUE` a single four-panel plot is produced, where the panels depict: 1. rate values, 2. richness values, 3. detrended rates , 4. transformed richness values (identical to 3., if no transformation happens).

```
res <- analyzeMetrics(metrics, ext="extPC", ori="oriPC", div="divCSIB",
  age="mid", dur="dur", name="name", normalize=FALSE,
  plot=TRUE, feedback=FALSE)
```



The `list` class output of the function includes the objects that were introduced earlier in sections 5.1 and 5.2.

```
res
```

```
## $`whichModel is indicated, pulsed or continuous`
## [1] TRUE
##
## $`pulsed-continuous significance`
##                                      est            p
## extinction (not-normalized)   0.2322265 2.762984e-02
## extinction (normalized)      -0.4496332 8.756357e-06
## origination (not-normalized)  0.3398184 1.049805e-03
## origination (normalized)     -0.5185042 1.651254e-07
##
## $declines
##                               correlation      p-value
## extinctions                    0.53769787 1.825059e-07
## post-Ordovician extinctions    0.24391670 4.365518e-02
```

```
## originations                   0.44500684 2.419816e-05
## post-Ordovician originations  0.06057727 6.202256e-01
##
## $`largest extinction`
## [1] "end-Permian"
##
## $`extinction outliers (boxplot)`
## [1] "other"          "end-Permian"    "end-Triassic"   "end-Cretaceous"
##
## $`log-normality (Shapiro-Wilk p-value)`
##       ext       ori
## 0.6204032 0.3704712
##
## $dyn
## $dyn$est
##                 div   divPlus       ori      oriPlus          ext
## div      1.00000000 0.6658127 0.3913798 -0.031943725  0.602162162
## divPlus  0.66581266 1.0000000 0.3816512  0.391379801  0.085879304
## ori      0.39137980 0.3816512 1.0000000  0.399511292  0.270753912
## oriPlus -0.03194372 0.3913798 0.3995113  1.000000000 -0.009433543
## ext      0.60216216 0.0858793 0.2707539 -0.009433543  1.000000000
## extPlus  0.61780081 0.6021622 0.2754683  0.270753912  0.410410959
##           extPlus
## div     0.6178008
## divPlus 0.6021622
## ori     0.2754683
## oriPlus 0.2707539
## ext     0.4104110
## extPlus 1.0000000
##
## $dyn$p
##                  div      divPlus          ori      oriPlus          ext
## div     0.000000e+00 0.000000e+00 0.0005679593 0.7866069247 2.194412e-08
## divPlus 0.000000e+00 0.000000e+00 0.0008619459 0.0005679593 4.660626e-01
## ori     5.679593e-04 8.619459e-04 0.0000000000 0.0004672974 1.906588e-02
## oriPlus 7.866069e-01 5.679593e-04 0.0004672974 0.0000000000 9.363599e-01
## ext     2.194412e-08 4.660626e-01 0.0190658833 0.9363599436 0.000000e+00
## extPlus 8.651692e-09 2.194412e-08 0.0177947295 0.0190658833 3.164707e-04
##              extPlus
## div     8.651692e-09
## divPlus 2.194412e-08
## ori     1.779473e-02
## oriPlus 1.906588e-02
## ext     3.164707e-04
## extPlus 0.000000e+00
##
## $dyn$sig
##       Var1    Var2  estimate            p
## 2      div divPlus 0.6658127 0.000000e+00
## 3      div     ori 0.3913798 5.679593e-04
## 5      div     ext 0.6021622 2.194412e-08
## 6      div extPlus 0.6178008 8.651692e-09
## 9  divPlus     ori 0.3816512 8.619459e-04
## 10 divPlus oriPlus 0.3913798 5.679593e-04
```

```
## 12 divPlus extPlus 0.6021622 2.194412e-08
## 16     ori oriPlus 0.3995113 4.672974e-04
## 17     ext     ori 0.2707539 1.906588e-02
## 18 extPlus     ori 0.2754683 1.779473e-02
## 24 extPlus oriPlus 0.2707539 1.906588e-02
## 30     ext extPlus 0.4104110 3.164707e-04
```

6. Applying the analyses to multiple series

The best thing about the rate and diversity estimators is also the worst thing: they are in continuous development. Although there is evidence to support the better applicability of some methods (Alroy, 2014; 2015), different researchers may favor different solutions to answer scientific hypotheses. Therefore, we are not trying to impose the use of any methods, but rather intend to make these calculations available to everybody and demonstrate the effects of methodological choices on the results.

For instance, results depend on the chosen equations to calculate the taxonomic turnover series. Although the per capita (Foote, 1999) rates were the most frequently used method in the past two decades, using these equations for the estimation of the rates can lead to systematic errors close to the edges of the time series (i.e. edge effects; Foote, 2000). Other methods (Alroy, 2008; 2014; 2015) were developed to be unaffected by this phenomenon. Nevertheless, even within the series, the different methods will lead to considerably varying values that can have an effect on the results used to test overarching hypotheses.

Therefore, we decided to redo the analysis presented in section 5 using different metrics of turnover and methods of sampling standardization, to assess how robust the general findings are in the face of different methodologies. Sets of time series were drafted based on two different resolutions (10my bin and stages), three different data treatments (using raw data, Classical Rarefaction and Shareholder Quroum Subsampling) and four different rate calculation methods (per capita rates, corrected three-timer rates, gap-filler equations and second-for-third substitution rates). Metrics published earlier were not considered in the analyses as they have known problems with accuracy (Foote, 1994). Equilibrial dynamics were tested using the corrected SIB diversity values.

6.1. Calculating time series with multiple methods

The candidate metrics are calculated with the different resolutions and data treatment options. First, the calculations with the 10 my time scale are implemented.

Before the calculation of the actual values can take place, the desired level of sampling intensities has to be determined for the sampling standardization processes. Classical Rarefaction (CR) is the oldest of the subsampling methods (Raup, 1975). Diversity curves drafted with this method are strongly dependent on sampling intensity: Alroy (2010b) has shown that the method leads to richness curves that flatten as the quota (desired sampling intensity in occurrences) decreases. Despite the fact that CR has its own limitations for richness estimation, it is still one of the most widely used sampling standardization protocol in the literature, and it produces comparable results to SQS with our data.

The quota for CR is set so that a complete time series can be produced without creating time slices with 'failed' subsampling, where there are not enough sampled occurrences. These were already calculated and are present in the `sampTen` and `sampStages` objects.

Let's first look at the number of occurrences in the 10 myr stage-results. We need to find out which bins are the least sampled to figure out a good subsampling quota. To do this, it is very useful to assign the bin identifiers to the number of occurrences first (names attribute), and then we can put the occurrences in ascending order.

```
binOccs <- sampTen$occs
names(binOccs) <- paste("bin", rownames(sampTen), sep="")
# in ascending order
sort(binOccs)
```

```
##  bin1 bin16 bin17 bin35  bin5 bin38 bin18 bin36 bin37 bin13  bin3 bin28
##   501  4713  4849  4926  5211  5243  5756  6326  6352  6379  6401  6599
## bin30 bin21 bin15 bin19 bin39 bin20 bin40  bin4  bin2  bin6 bin44 bin12
##  6651  6719  7803  7887  8090  8782  9311  9349  9954 10213 10267 10409
## bin42 bin33  bin7 bin45 bin29 bin41 bin22 bin31 bin47 bin32 bin26 bin27
## 10423 10434 10466 10759 11065 11332 11981 12226 12331 12393 12517 12906
## bin46 bin10  bin9 bin23 bin34 bin48 bin25 bin14 bin11 bin43 bin24  bin8
## 14789 15506 16048 19712 23267 24273 24294 24980 25050 26767 28580 33122
## bin49
## 50516
```

This indicates that the first bin is disproportionately poorly sampled. This is unlikely to convey important information for most of the time series. At the same time, subsampling the whole series to this low sampling level would destroy a huge amount of information. However, the second lowest level of about 4,800 occurrences (`bin16`, Devonian 5) looks like a decent level of sampling and can be applied to almost the whole time series.

The same reasoning can be applied to the stage-level data.

```
stgOccs <- sampStg$occs
names(stgOccs) <- paste("stg", rownames(sampStg), sep="")
sort(stgOccs)
```

```
##  stg4 stg11  stg5  stg9 stg78 stg13 stg26 stg59 stg41  stg7 stg70 stg95
##  1049  1418  1462  1472  1579  1755  1964  2193  2283  2363  2413  2499
## stg22 stg63 stg39 stg73 stg10 stg20 stg28 stg25 stg72 stg12 stg38 stg79
##  2670  2687  2909  2963  3049  3109  3120  3164  3274  3293  3470  3477
## stg83 stg55 stg16 stg27 stg71 stg60 stg30  stg8 stg21 stg42 stg64 stg15
##  3528  3536  3551  3592  3726  3885  3901  3916  3954  4162  4188  4272
## stg58  stg6 stg89 stg23 stg35 stg43 stg85 stg36 stg82 stg69 stg14 stg40
##  4402  4569  4633  4681  4713  4716  4767  4849  4911  4926  4930  5150
## stg52 stg74 stg68 stg77 stg32 stg65 stg84 stg57 stg24 stg29 stg31 stg86
##  5214  5243  5315  5340  5343  5385  5592  5825  5863  6095  6379  6384
## stg88 stg56 stg44 stg53 stg45 stg66 stg34 stg67 stg87 stg75 stg48 stg54
##  6396  6599  6748  6750  7290  7564  7803  7931  7996  8090  8710  8992
## stg49 stg47 stg76 stg62 stg80 stg51 stg91 stg90 stg37 stg46 stg61 stg50
##  9005  9243  9311  9514 10423 10822 11025 11052 11191 12133 12226 12381
## stg17 stg18 stg92 stg93 stg94 stg33 stg81 stg19
## 12490 12586 15718 15854 16585 17597 26767 27872
```

In this case, the number of occurrences in a bin increases gradually, and multiple poorly sampled bins are in the middle of the series. Using different quotas would create different results, for the sake of simplicity, it is probably best to restrain ourselves to a lower quota of 1,100 occurrences for the stage-level analyses.

For SQS, the subsampling configuration discussed in Section 4 stands here, too. A major advantage of SQS to CR is that it gives you more or less the same curve regardless of the quorum, as long as the quorum isn't extremely low (< 0.4 or so, Alroy 2010b).

The large quantity of data and the relatively small subsample sizes indicate that one must carry out a high number of iterations for the estimates to stabilize. The results presented in this section are based on 500 iterations, which will take a considerable amount of time to run, but you can decrease this parameter at will by setting the `iter` argument of the `subsample()` function.

```
# 1. raw patterns
ddTen <- divDyn(dat, bin="ten", tax="clgen")
# 2. CR
crTen <- subsample(dat, bin="ten", tax="clgen", coll="collection_no", q=4700, iter=500,
  duplicates=FALSE, na.rm=TRUE)
# 3. SQS
sqsTen <- subsample(dat, bin="ten", tax="clgen", coll="collection_no", q=0.7, iter=500,
  ref="reference_no", singleton="ref", type="sqs", duplicates=FALSE, excludeDominant=TRUE,
  largestColl =TRUE, na.rm=TRUE)
```

Then the calculations were repeated for the stage-level resolution.

```
# 1. raw patterns
  ddStg <- divDyn(dat, bin="stg", tax="clgen")
# 2. CR
crStg <- subsample(dat, bin="stg", tax="clgen", coll="collection_no", q=1000,
  iter=500, duplicates=FALSE, na.rm=TRUE)
# 3. SQS
sqsStg <- subsample(dat, bin="stg", tax="clgen", coll="collection_no", q=0.7,
  iter=500, ref="reference_no", singleton="ref", type="sqs", duplicates=FALSE,
  excludeDominant=TRUE, largestColl=TRUE, na.rm=TRUE)
```

In total, we have 6 different result objects.

6.2. Running the analysis script on the different sets of time series

The next thing is to organize the results (i.e., create identifiers of the different time series) so the analyses described in section 5 can be iterated. Each run of the analytical script needs three time series (origination, extinction, richness), which can be identified with the appropriate result table names (stratigraphic resolution, data treatment) and the variable names (metric type). The following object `comb` is created to organize these combinations.

```
# combination table
# types of rates
comb <- matrix(
  c(
    "extPC", "oriPC", "divCSIB", # per capita rates
    "extC3t", "oriC3t", "divCSIB", # corrected 3t rates
    "extGF", "oriGF", "divCSIB", # gap-filler rates
    "ext2f3", "ori2f3", "divCSIB" # second-for-third-substitution rates
  ),
  ncol=3, nrow=4, byrow=T)
  rownames(comb) <- c("PC", "C3t", "GF", "2f3")
comb <- comb[rep(1:4, 6), ]

# the result matrices
sourceVar<-rep(c("ddTen", "crTen", "sqsTen", "ddStg", "crStg", "sqsStg"),each=4)

# timescale objects
scale <- rep(c("tens", "stages"), each=12)
# combine everything together
comb <- cbind(sourceVar, scale, comb)
colnames(comb) <- c("source", "timescale", "ext", "ori", "div")
```

```
# the names of the results (rownames)
subtype <- rep(rep(c("raw", "cr", "sqs"), each=4),2)
rownames(comb) <- paste(subtype, rownames(comb), sep="")
rownames(comb) <- paste(rownames(comb), rep(c("10my", "stages"), each=12), sep="_")
comb
```

```
##                source   timescale ext      ori      div
## rawPC_10my     "ddTen"  "tens"    "extPC"  "oriPC"  "divCSIB"
## rawC3t_10my    "ddTen"  "tens"    "extC3t" "oriC3t" "divCSIB"
## rawGF_10my     "ddTen"  "tens"    "extGF"  "oriGF"  "divCSIB"
## raw2f3_10my    "ddTen"  "tens"    "ext2f3" "ori2f3" "divCSIB"
## crPC_10my      "crTen"  "tens"    "extPC"  "oriPC"  "divCSIB"
## crC3t_10my     "crTen"  "tens"    "extC3t" "oriC3t" "divCSIB"
## crGF_10my      "crTen"  "tens"    "extGF"  "oriGF"  "divCSIB"
## cr2f3_10my     "crTen"  "tens"    "ext2f3" "ori2f3" "divCSIB"
## sqsPC_10my     "sqsTen" "tens"    "extPC"  "oriPC"  "divCSIB"
## sqsC3t_10my    "sqsTen" "tens"    "extC3t" "oriC3t" "divCSIB"
## sqsGF_10my     "sqsTen" "tens"    "extGF"  "oriGF"  "divCSIB"
## sqs2f3_10my    "sqsTen" "tens"    "ext2f3" "ori2f3" "divCSIB"
## rawPC_stages   "ddStg"  "stages"  "extPC"  "oriPC"  "divCSIB"
## rawC3t_stages  "ddStg"  "stages"  "extC3t" "oriC3t" "divCSIB"
## rawGF_stages   "ddStg"  "stages"  "extGF"  "oriGF"  "divCSIB"
## raw2f3_stages  "ddStg"  "stages"  "ext2f3" "ori2f3" "divCSIB"
## crPC_stages    "crStg"  "stages"  "extPC"  "oriPC"  "divCSIB"
## crC3t_stages   "crStg"  "stages"  "extC3t" "oriC3t" "divCSIB"
## crGF_stages    "crStg"  "stages"  "extGF"  "oriGF"  "divCSIB"
## cr2f3_stages   "crStg"  "stages"  "ext2f3" "ori2f3" "divCSIB"
## sqsPC_stages   "sqsStg" "stages"  "extPC"  "oriPC"  "divCSIB"
## sqsC3t_stages  "sqsStg" "stages"  "extC3t" "oriC3t" "divCSIB"
## sqsGF_stages   "sqsStg" "stages"  "extGF"  "oriGF"  "divCSIB"
## sqs2f3_stages  "sqsStg" "stages"  "ext2f3" "ori2f3" "divCSIB"
```

In this table, every row corresponds to the arguments of a certain set of time series. For instance, the first row points to the raw per capita rates with the 10 myr stratigraphic resolution. Now, all that remain is to initialize and iterate the analyses using the different sets of time series. In order to display the rates together, they have to be saved in separate containers.

```
# matrices to hold rates for later plotting
extDatTen <- data.frame(ten=1:49) # extinctions, 10my
oriDatTen <- data.frame(ten=1:49) # originations, 10my
extDatStg <- data.frame(stg=1:95) # extinctions, stages
oriDatStg <- data.frame(stg=1:95) # originations, stages
```

The first two `data.frame` objects correspond to the 10my bin resolution and the second one to the stage-level resolution. The following code applies the function to the different time series in a `for` loop (for transparency).

```
#! # initialize pdf, if you want to
#! pdf("2018-08-21_marineAnimals2_redo.PDF", 17,14)
# iterate through all cases
for(i in 1:nrow(comb)){
  # name of the set
  case <- rownames(comb)[i]
  # the results matrix and the timescale object combined
  metrics <- cbind(get(comb[i, "timescale"]), get(comb[i, "source"]))

  # save rates for later, depending on the resolution
```

```
  # 10 my
  if(comb[i, "timescale"] == "tens"){
    oriDatTen[[case]] <- metrics[, comb[i, "ori"]]
    extDatTen[[case]] <- metrics[, comb[i, "ext"]]
  }
  # stages
  if(comb[i, "timescale"] == "stages"){
    oriDatStg[[case]] <- metrics[, comb[i, "ori"]]
    extDatStg[[case]] <- metrics[, comb[i, "ext"]]
  }
  # the analytical function
  res <- analyzeMetrics(metrics, ext=comb[i, "ext"], ori=comb[i, "ori"],
    div=comb[i, "div"], age="mid", dur="dur", name="name", normalize=TRUE,
    plot=FALSE, feedback=FALSE, detrend="loess", transform=FALSE, additive=FALSE)

  # save in global namespace with unique name
  assign(case, res)

  # add the name to the plot
  #! par(mfrow=c(1,1))
  #! mtext(side=3, text=case, line=-2, cex=3)
}

#! dev.off()
```

With this snippet, each result will be stored as a list class object in the global namespace, with the name of the row in `comb` that contains the arguments.

All lines commented with `#!` are part of the embedded plotting functionality. If you uncomment these lines (delete `!` too!) and set `plot=TRUE` in the `analyzeMetrics()` function call, a single .pdf file will be produced that will show the four-panel figure presented in section 5.3 for every set of time series (row in `comb`) on a separate page. You can take look at this `.pdf` file by following this link:

`https://github.com/divDyn/ddPhanero/blob/master/export/1.0.1/detrending.pdf`

6.3. Summarizing the results (Table 3)

To get an idea of the generality of the results, the support for the individual hypotheses can be summarized in a single display item. We prepared a function to extract correlation coefficient estimates and p-values from the individual objects. Two temporary tables were prepared - one for the estimates and another one for the significance values. The idea behind this approach is to use the first table to present the actual values and use the table of p-values to format the first table (using conditional formatting in MS Excel). The function to gather the relevant information from the lists can be inspected in the `phanDyn.R` file (`extractVals()` function).

```
# the shown values
values <- extractVals(rownames(comb), pvals=FALSE)
```

This is a fairly large table that includes every case-related results in a column. For correlation tests, the table denotes the coefficient estimates. For the presence of mass-extinction intervals, the table shows binary values. As the iteration through the output object is the same for the p-values and the estimates, it was a straightforward choice to use the same function to extract the p-values. Setting the `pvals` argument of this function to `TRUE` will extract the p-values, where another they are also present.

```
# the p values
pvals <- extractVals(rownames(comb), pvals=TRUE)
```

Note that the entries for the presences of mass extinctions in this table there are no significance values. The table was compiled to be used for conditional formatting and 0.002 values indicate the presence of a mass extinction, 0.02 indicates its absence.

The support or rejection of hypotheses depend on the p-values. The results based on the 10 myr-scale results have the following values (rounded and transposed for better readability):

```
round(t(pvals[1:12,]),3)
```

```
##                                   rawPC_10my rawC3t_10my rawGF_10my
## ext. rates with durations              0.995       0.671      0.847
## norm. ext. rates with durations        0.004       0.002      0.012
## orig. rates with durations             0.044       0.122      0.047
## norm. orig. rates with  durations      0.058       0.005      0.058
## extinctions                            0.000       0.000      0.000
## post-Ordovician extinctions            0.035       0.015      0.011
## originations                           0.002       0.002      0.008
## post-Ordovician originations           0.862       0.127      0.824
## end-Permian ME                         0.002       0.002      0.002
## end-Triassic ME                        0.002       0.002      0.002
## end-Cretaceous ME                      0.002       0.002      0.002
## end-Permian is highest                 0.002       0.020      0.002
## number of mass extinctions             0.002       0.002      0.002
## extinctions log-normal (p-values)      0.151       0.434      0.077
## originations log-normal (p-values)     0.398       0.455      0.795
## origination and lagged diversity       0.089       0.399      0.300
## diversity and lagged extinction        0.010       0.079      0.119
## extinction and lagged origination      0.903       0.000      0.067
##                                   raw2f3_10my crPC_10my crC3t_10my
## ext. rates with durations               0.959     0.652      0.166
## norm. ext. rates with durations         0.028     0.000      0.000
## orig. rates with durations              0.039     0.138      0.465
## norm. orig. rates with  durations       0.306     0.016      0.006
## extinctions                             0.000     0.000      0.000
## post-Ordovician extinctions             0.156     0.038      0.007
## originations                            0.011     0.000      0.003
## post-Ordovician originations            0.784     0.040      0.100
## end-Permian ME                          0.002     0.002      0.002
## end-Triassic ME                         0.002     0.002      0.002
## end-Cretaceous ME                       0.002     0.002      0.002
## end-Permian is highest                  0.002     0.002      0.002
## number of mass extinctions              0.002     0.002      0.002
## extinctions log-normal (p-values)       0.603     0.123      0.055
## originations log-normal (p-values)      0.863     0.719      0.952
## origination and lagged diversity        0.189     0.025      0.111
## diversity and lagged extinction         0.148     0.049      0.424
## extinction and lagged origination       0.548     0.453      0.000
##                                   crGF_10my cr2f3_10my sqsPC_10my
## ext. rates with durations             0.575      0.356      0.600
## norm. ext. rates with durations       0.000      0.000      0.008
## orig. rates with durations            0.232      0.262      0.037
## norm. orig. rates with  durations     0.073      0.089      0.151
```

```
## extinctions                            0.000      0.000      0.000
## post-Ordovician extinctions            0.015      0.044      0.050
## originations                           0.001      0.001      0.002
## post-Ordovician originations           0.057      0.038      0.797
## end-Permian ME                         0.002      0.002      0.002
## end-Triassic ME                        0.002      0.002      0.002
## end-Cretaceous ME                      0.002      0.002      0.020
## end-Permian is highest                 0.002      0.002      0.002
## number of mass extinctions             0.002      0.002      0.002
## extinctions log-normal (p-values)      0.144      0.074      0.862
## originations log-normal (p-values)     0.991      0.795      0.426
## origination and lagged diversity       0.094      0.103      0.003
## diversity and lagged extinction        0.108      0.159      0.006
## extinction and lagged origination      0.014      0.027      0.385
##                                    sqsC3t_10my sqsGF_10my sqs2f3_10my
## ext. rates with durations                0.537      0.446       0.954
## norm. ext. rates with durations          0.002      0.037       0.018
## orig. rates with durations               0.281      0.124       0.153
## norm. orig. rates with  durations        0.014      0.165       0.460
## extinctions                              0.000      0.000       0.000
## post-Ordovician extinctions              0.019      0.005       0.078
## originations                             0.004      0.001       0.002
## post-Ordovician originations             0.240      0.154       0.165
## end-Permian ME                           0.002      0.002       0.002
## end-Triassic ME                          0.002      0.002       0.002
## end-Cretaceous ME                        0.002      0.020       0.002
## end-Permian is highest                   0.002      0.002       0.002
## number of mass extinctions               0.002      0.002       0.002
## extinctions log-normal (p-values)        0.231      0.815       0.473
## originations log-normal (p-values)       0.798      0.013       0.158
## origination and lagged diversity         0.040      0.058       0.090
## diversity and lagged extinction          0.206      0.112       0.121
## extinction and lagged origination        0.002      0.312       0.658
```

The stage-level p-values can also be extracted in a similar way:

```
round(t(pvals[13:24,]),3)
```

```
##                                    rawPC_stages rawC3t_stages rawGF_stages
## ext. rates with durations                 0.028         0.160        0.013
## norm. ext. rates with durations           0.000         0.001        0.003
## orig. rates with durations                0.001         0.063        0.006
## norm. orig. rates with  durations         0.000         0.000        0.000
## extinctions                               0.000         0.000        0.000
## post-Ordovician extinctions               0.044         0.325        0.250
## originations                              0.000         0.001        0.001
## post-Ordovician originations              0.620         0.788        0.660
## end-Permian ME                            0.002         0.002        0.002
## end-Triassic ME                           0.002         0.002        0.002
## end-Cretaceous ME                         0.002         0.002        0.002
## end-Permian is highest                    0.002         0.002        0.002
## number of mass extinctions                0.002         0.002        0.002
## extinctions log-normal (p-values)         0.620         0.031        0.232
## originations log-normal (p-values)        0.370         0.011        0.000
## origination and lagged diversity          0.001         0.014        0.054
```

```
## diversity and lagged extinction           0.000         0.000        0.001
## extinction and lagged origination         0.936         0.379        0.165
##                                    raw2f3_stages crPC_stages crC3t_stages
## ext. rates with durations                  0.016       0.439        0.166
## norm. ext. rates with durations            0.030       0.000        0.000
## orig. rates with durations                 0.004       0.111        0.154
## norm. orig. rates with  durations          0.002       0.000        0.000
## extinctions                                0.000       0.000        0.000
## post-Ordovician extinctions                0.228       0.006        0.445
## originations                               0.003       0.000        0.000
## post-Ordovician originations               0.961       0.023        0.698
## end-Permian ME                             0.002       0.002        0.002
## end-Triassic ME                            0.002       0.002        0.002
## end-Cretaceous ME                          0.002       0.002        0.002
## end-Permian is highest                     0.002       0.002        0.002
## number of mass extinctions                 0.002       0.002        0.002
## extinctions log-normal (p-values)          0.000       0.064        0.003
## originations log-normal (p-values)         0.000       0.873        0.905
## origination and lagged diversity           0.085       0.002        0.097
## diversity and lagged extinction            0.005       0.000        0.260
## extinction and lagged origination          0.874       0.442        0.000
##                                    crGF_stages cr2f3_stages sqsPC_stages
## ext. rates with durations                0.076        0.103        0.215
## norm. ext. rates with durations          0.000        0.000        0.000
## orig. rates with durations               0.204        0.294        0.023
## norm. orig. rates with  durations        0.000        0.000        0.000
## extinctions                              0.000        0.000        0.000
## post-Ordovician extinctions              0.342        0.268        0.008
## originations                             0.000        0.000        0.000
## post-Ordovician originations             0.584        0.454        0.213
## end-Permian ME                           0.002        0.002        0.002
## end-Triassic ME                          0.002        0.002        0.002
## end-Cretaceous ME                        0.002        0.002        0.002
## end-Permian is highest                   0.002        0.002        0.002
## number of mass extinctions               0.002        0.002        0.002
## extinctions log-normal (p-values)        0.002        0.216        0.873
## originations log-normal (p-values)       0.001        0.000        0.465
## origination and lagged diversity         0.225        0.580        0.000
## diversity and lagged extinction          0.086        0.278        0.000
## extinction and lagged origination        0.000        0.000        0.901
##                                    sqsC3t_stages sqsGF_stages
## ext. rates with durations                  0.122        0.019
## norm. ext. rates with durations            0.000        0.000
## orig. rates with durations                 0.100        0.044
## norm. orig. rates with  durations          0.000        0.000
## extinctions                                0.000        0.000
## post-Ordovician extinctions                0.478        0.464
## originations                               0.001        0.000
## post-Ordovician originations               0.875        0.805
## end-Permian ME                             0.002        0.002
## end-Triassic ME                            0.002        0.002
## end-Cretaceous ME                          0.002        0.002
## end-Permian is highest                     0.002        0.002
## number of mass extinctions                 0.002        0.002
```

```
## extinctions log-normal (p-values)          0.000        0.001
## originations log-normal (p-values)         0.542        0.000
## origination and lagged diversity           0.004        0.005
## diversity and lagged extinction            0.028        0.001
## extinction and lagged origination          0.157        0.281
##                                    sqs2f3_stages
## ext. rates with durations                  0.025
## norm. ext. rates with durations            0.000
## orig. rates with durations                 0.071
## norm. orig. rates with  durations          0.000
## extinctions                                0.000
## post-Ordovician extinctions                0.366
## originations                               0.001
## post-Ordovician originations               0.693
## end-Permian ME                             0.002
## end-Triassic ME                            0.002
## end-Cretaceous ME                          0.002
## end-Permian is highest                     0.002
## number of mass extinctions                 0.002
## extinctions log-normal (p-values)          0.003
## originations log-normal (p-values)         0.007
## origination and lagged diversity           0.015
## diversity and lagged extinction            0.008
## extinction and lagged origination          0.641
```

The tables of p-values and estimates can be combined using conditional formatting to render a comprehensible table. The following formatting steps were applied to these tables that were later fed to an MS Excel spreadsheet.

```
# round most values
valuesRounded <- round(values,2)
# except for p-values
valuesRounded[,
  c("extinctions log-normal (p-values)",
    "originations log-normal (p-values)")] <-
  round(values[,c(
    "extinctions log-normal (p-values)",
    "originations log-normal (p-values)")],3)

#transform to character
#valuesRounded <- as.data.frame(valuesRounded)
for(i in 9:13){
    valuesRounded[ ,i] <- as.character(valuesRounded[,i])
    valuesRounded[valuesRounded[ ,i]=="1",i] <- "yes"
    valuesRounded[valuesRounded[ ,i]=="0",i] <- "no"
}

# use inequality signs where values are very low
valuesRounded[
  valuesRounded[,"extinctions log-normal (p-values)"]=="0",
  "extinctions log-normal (p-values)"] <- "<0.001"
valuesRounded[
  valuesRounded[,"originations log-normal (p-values)"]=="0",
  "originations log-normal (p-values)"] <-  "<0.001"
```

In the actual paper, Table 3 was compiled after the transposition of the estimate and p-value tables, and the separation of both tables to 10 myr time scale and stage-level timescale subsets.

```
# transpose everything
tValues<- t(valuesRounded)
tP <- t(pvals)

# separate based on resolution
# 10 myr
  valBin <- tValues[,1:12]
  pBin <- tP[,1:12]
# stages
  valStage <- tValues[,13:24]
  pStage <- tP[,13:24]
```

These four tables were then fed to the Excel spreadsheet we prepared to do the conditional formatting of the `valBin` and `valStage` tables. This MS Excel file (`conditionalTable.xlsx`) can also be found at the GitHub repository of the example.

6.4. Figure of rates with multiple methods (Figure 2)

To show the total variation in the taxonomic rate series, the origination and extinction rates were rendered at the two different resolutions. Four panels were drawn, each panel having 12 time series. The quantiles of the value distributions in the independent time slices were connected with the `shades()` function to provide backgrounds.

Then the panels in sequence:

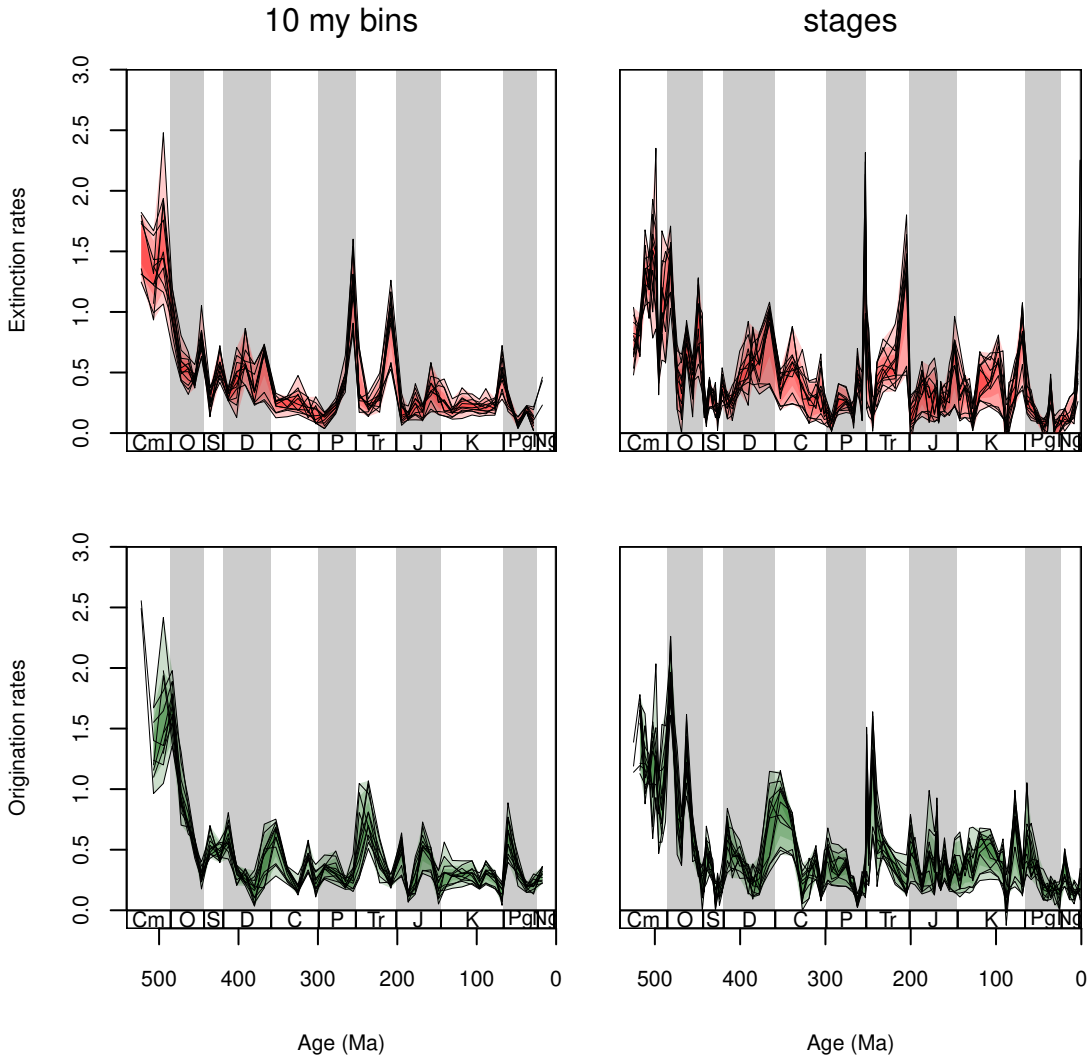
```
par(mfrow=c(2,2))
# extinctions
  # 1st panel - 10myr bins
  par(mar=c(2, 5.1, 4.1, 1)) # margin adjustment
  # plot
  tsplot(stages, boxes="sys", shading="sys", ylim=c(0, 3), ylab="Extinction rates",
    xlim=4:95, plot.args=list(axes=F, cex.lab=0.8), xlab="",labels.args=list(cex=0.8))
  axis(2, cex.axis=0.8)
  shades(tens$mid, as.matrix(extDatTen[,-1]), col="red") # background
  for(i in 2:ncol(extDatTen))
    lines(tens$mid, extDatTen[,i] , lwd=0.5) # extinctions - BIN
  mtext(line=1, text="10 my bins", side=3, cex=1) # show resolution

# 2nd panel - stages
  par(mar=c(2, 1, 4.1, 4.1)) # margin adjustment
  # plot
  tsplot(stages, boxes="sys", shading="sys", ylim=c(0, 2), ylab="", xlim=4:95,
    plot.args=list(axes=F, cex.lab=0.8), xlab="",labels.args=list(cex=0.8))
  shades(stages$mid, as.matrix(extDatStg[,-1]), col="red") # background
  for(i in 2:ncol(extDatStg))
    lines(stages$mid, extDatStg[ ,i] , lwd=0.5) # extinction - stages
  mtext(line=1, text="stages", side=3, cex=1)

# originations
  # 3rd panel - 10 myr bins
  par(mar=c(5.1, 5.1, 1, 1)) # margin adjustment
  # plot
```

```
tsplot(stages, boxes="sys", shading="sys", ylim=c(0, 3), ylab="Origination rates",
  xlim=4:95, plot.args=list(axes=F, cex.lab=0.8),labels.args=list(cex=0.8),
  xlab="Age (Ma)")
axis(1, cex.axis=0.8)
axis(2, cex.axis=0.8)
shades(tens$mid, as.matrix(oriDatTen[,-1]), col="darkgreen") # background
for(i in 2:ncol(oriDatTen))
  lines(tens$mid, oriDatTen[ ,i], lwd=0.5) # originations - BIN

# 4th panel - stages
par(mar=c(5.1, 1, 1, 4.1))# margin adjustment
# plot
tsplot(stages, boxes="sys", shading="sys", ylim=c(0, 2), ylab="", xlim=4:95,
  plot.args=list(axes=F, cex.lab=0.8),labels.args=list(cex=0.8), xlab="Age (Ma)")
axis(1, cex.axis=0.8)
shades(stages$mid, as.matrix(oriDatStg[,-1]), col="darkgreen") # background
for(i in 2:ncol(oriDatStg))
  lines(stages$mid, oriDatStg[ ,i] , lwd=0.5) # originations - stages
```



7. Conclusions

As you can see in the plots above, the estimated rates vary considerably from trial to trial, hence the variation in the final table (Table 3) summarizing the results. The different metrics have different advantages and disadvantages, although some are expected to work better than others [for instance, gap-fillers (Alroy, 2014) completely supersede the corrected three-timer rates (Alroy, 2008)]. The different subsampling procedures and stratigraphic resolutions also contribute to the variation that ultimately reflects the inherent uncertainty in our estimates.

It is apparent that some patterns are more robust than others and their uncertainty has to be taken into account when evaluating large-scale scientific hypotheses. It is highly probable that extinction and origination declined in the whole Phanerozoic, but the post-Ordovician likely featured no decline or a much shallower (lower rates in the Cenozoic are probably biologically meaningful). The most studied 'Big Three' (Alroy, 2008) mass extinction intervals (end-Permian, end-Triassic, end-Cretaceous) stand out clearly from the distribution of extinction rates. Equilibrial dynamics of global richness (Alroy, 2008, 2010b), on the other hand, have to be reevaluated, based on the number of different approaches we have and the increasing quantity of data.

Acknowledgments

Work on the package was funded by the Deutsche Forschungsgemeinschaft (Ko 5382/1-1, Ko 5382/1-2 and Ki 806/16-1) and is part of the Research Unit TERSANE (FO 2332). We thank all contributors of the Paleobiology Database, especially M. Clapham, A. Hendy, M. Carrano, A. Miller, M. Uhen, M. Aberhan B. Krger, P. Wagner, M. Patzkowsky, M. Foote and J. Plfy. We are also thankful to Na Lin, for assigning the Cambrian collections to the stages.

References

Alroy, J. 2008. Dynamics of origination and extinction in the marine fossil record. Proceedings of the National Academy of Science 105:11536-11542.

Alroy, J. 2010a. The shifting balance of diversity among major marine animal groups. Science 329:1191-1194.

Alroy, J. 2010b. Geographical, environmental and intrinsic biotic controls on Phanerozoic marine diversification. Palaeontology 53:1211-1235.

Alroy, J. 2014. Accurate and precise estimates of origination and extinction rates. Paleobiology, 40, 374-397.

Alroy, J. 2015. A more precise speciation and extinction rate estimator. Paleobiology, 41, 633-639.

Bambach, R. K., A. H. Knoll, and S. C. Wang. 2004. Origination, extinction, and mass depletions of marine diversity. Paleobiology 30:522-542.

Chao, A., and L. Jost. 2012. Coverage-based rarefaction and extrapolation: standardizing samples by completeness rather than size. Ecology 93:2533-2547.

Foote, M. 1994. Temporal variation in extinction risk and temporal scaling of extinction metrics. Paleobiology, 20, 424-444.

Foote, M. 1999. Morphological diversity in the evolutionary radiation of Paleozoic and post-Paleozoic crinoids. Paleobiology, 25, 1-115.

Foote, M. 2000. Origination and extinction components of taxonomic diversity: General Problems. Paleobiology 26:74-102.

Foote, M. 2005. Pulsed origination and extinction in the marine realm. Paleobiology 31:6-20.

Good, I. J. 1953. The popoulation frequencies of species and the estimation of population parameters. Biometrika 40:237-264.

Hyndman, R. J., & Khandakar, Y. 2008. Automatic time series forecasting: the forecast package for R. Journal of Statistical Software, 26:1-22.

Na, L., and W. Kiessling. 2015. Diversity partitioning during the Cambrian radiation. Proceedings of the National Academy of Sciences 112:4702-4706.

Ogg, J. G., G. Ogg, and F. M. Gradstein. 2016. A concise geologic time scale: 2016. Elsevier.

Quinn, J. F. 1983. Mass extinctions in the fossil record. Science 219:1239-1240.

Raup, D. M. 1975. Taxonomic diversity estimation using rarefaction. Paleobiology, 1, 333-342.

Raup, D. M., and J. J. J. Sepkoski. 1982. Mass extinctions in the marine fossil record. Science 215:1501-1503.

Sepkoski, J. J. J. 1998. Rates of speciation in the fossil record. Philosophical Transactions of the Royal Society of London B Biological Sciences, 353:315-326.

Sepkoski Jr, J. J. 2002. A compendium of fossil marine animal genera. Bulletins of American Paleontology, 363, 1-560.

Wang, X.-F. 2010. fANCOVA: Nonparametric analysis of covariance. Retrieved from https://CRAN.R-project.org/package=fANCOVA