Example usage in R

Rutger Vos (@rvosa)

31-10-2018

Introduction

This document gives a first introduction in querying the TGD database such that occurrences and phylogenetic data are properly linked despite input data sets potentially using different taxon labels and different backbone taxonomies The overarching design of the schema is such that there is a table of `taxa`, which uses the names accepted by either the Mammal Species of the World 3rd edition, the GBIF backbone taxonomy, or both. As such, each record in that table has at least one of the identifiers used by these resources. Subsequently, all variations on these accepted names (that is, alternative spellings, synonyms, etc.) are records in the `taxonvariants` table, each of which have a foreign key that links back to the accepted name version. Whenever a data set (a tree, a set of occurrences, a trait data set) is imported, the record labels that are used in that data set are looked up in the taxon variants table. Any existing exact match is re-used, and new ones are created as needed. Then, an attempt is made to link new variants to accepted taxon names using taxonomic name reconciliation (as implemented in the `taxize` package). The upshot of this design is that, if you want to get all occurrences (say) for a given species, you first have to look up that species in the taxa table, then get all the variations on that name in the taxonvariants table, and then all occurrences that link to any of these variants. And, if you want to combine, for example, occurrences and phylogeny, you have to look for the intersection between "taxa that have variants that have occurrences", and "taxa that have variants that have branches in the given tree".

Without further ado, here's what the steps might be. First we connect to the database:

```
library(DBI)
```

```
## Warning: package 'DBI' was built under R version 3.4.4
```

```
db_file <- "../data/sql/tgd.db"
tgd <- dbConnect(RSQLite::SQLite(), db_file)
```

Then we need to look up the ID of the mammal supertree of Bininda-Emonds et al. (2007), which is named ../data/phylogeny/Bininda-emonds_2007_mammals.tsv. We query as follows:

```
tree_name <- "../data/phylogeny/Bininda-emonds_2007_mammals.tsv"
query <- 'select tree_id from trees where tree_name="%s"'
tree_id <- dbGetQuery(tgd, sprintf(query, tree_name))$tree_id
```

Given this `tree_id`, we are now going to look for taxa that have taxonvariants that have branches in the tree and that have taxonvariants that have occurrences. This is a fairly complex query all in all because there's a number of joins that need to be done. As follows:

```
query <- 'select distinct
taxa.taxon_name,occurrences.decimal_latitude,occurrences.decimal_longitude
from (((taxonvariants
inner join taxa on taxonvariants.taxon_id = taxa.taxon_id)
inner join occurrences on occurrences.taxonvariant_id = taxonvariants.taxonvariant_id)
inner join branches on branches.taxonvariant_id = taxonvariants.taxonvariant_id)
where branches.tree_id=12 and
taxa.taxon_level="SPECIES"
order by taxon_name'
results <- dbGetQuery(tgd, query)
```

Building a data frame

Now that we have a great big table, we should be able to do some potentially interesting analyses. For example, does Rapoport's rule apply to our data? This is an ecogeographical rule that states that latitudinal ranges of plants and animals are generally smaller at lower latitudes than at higher latitudes. So, for this we need the absolute average latitude and the latitudinal range.

The data structure that we are going to want thus needs to have one observation for each taxon, recording 2 variables (absolute average latitude, latitudinal range, longitudinal range). Here now we populate a data frame to do this:

```
# we will use the distinct taxon names (these are the clean, accepted names) as row names
names <- unique(results$taxon_name)

# we start with a matrix, populate this, and then make it into a data.frame
df <- matrix(ncol=2, nrow=length(names))

# use i index for the matrix
for ( i in 1:length(names) ) {

    # average absolute latitude
    aal <- mean(abs(results[which(results$taxon_name==names[[i]]),"decimal_latitude"]))

    # latitudinal range
    lar <- range(results[which(results$taxon_name==names[[i]]),"decimal_latitude"])
    lar <- lar[[2]] - lar[[1]]

    # add row to matrix
    df[i,] <- c( aal, lar )
}

# make data frame, assign taxa as row names, variables as col names
df <- data.frame(df, row.names = names)
colnames(df) <- c("average_absolute_latitude", "latitudinal_range")

# remove observations with ranges of 0, i.e. single observation
row_sub = apply(df, 1, function(row) all(row !=0 ))
df <- df[row_sub,]
names <- row.names(df)
```

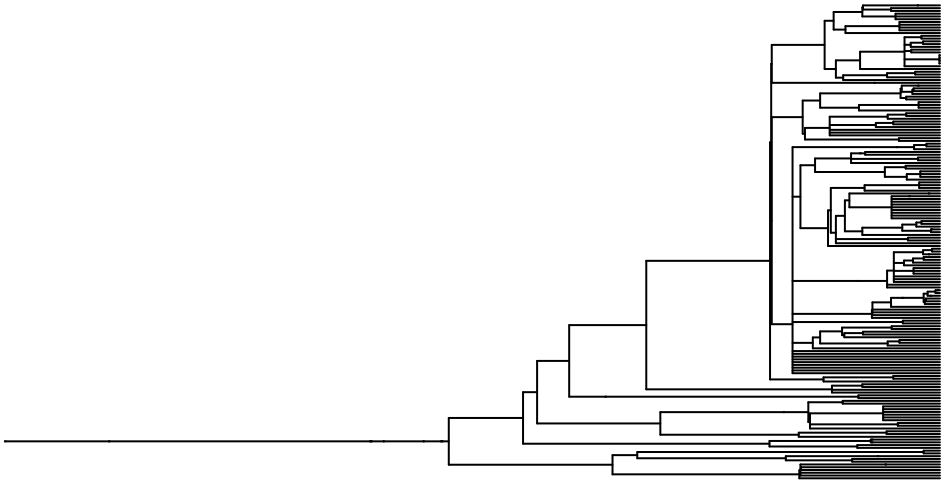
Importing phylogenies

We might now be tempted to simply do some correlation tests, but this is actually not a valid approach. Our observations are of species that are related to one another, so part of the commonality in observation values might be due to shared descent. For example, two sister species are probably near each other on the earth, and have a similar range: pseudo-replication. To account for this we need take phylogeny into account. Hence, we need to extract a tree from the database. We will do this by extracting a `phylo` object (as originally introduced by the `ape` package). As follows:

```
library(ape)
source('../R/make_phylo.R')
tree <- make.phylo(db_file, tree_id, names)

# at this scale, the tip labels will be cluttered.
```

```r
# let's just make sure it's an ultrametric tree.
plot(tree, show.tip.label=F)
```



Doing a comparative analysis

We now have everything in place to do a phylogenetic comparative analysis, e.g. `pic` in ape. We follow this tutorial:

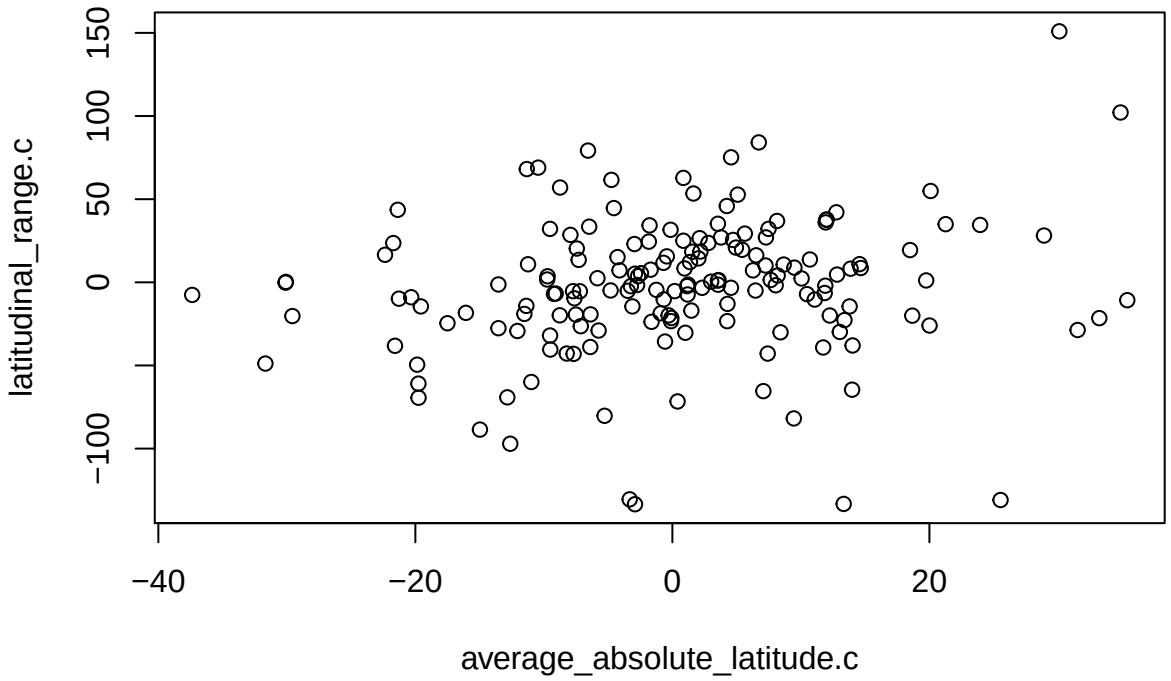
```r
# Because we extract a subtree from the database, it may
# have interior nodes with just one child. Here we
# collapse these.
tree <- collapse.singles(tree)

# The tree may not be fully bifurcating, althout PIC needs
# this. Hence we arbitrarily resolve it.
tree <- multi2di(tree)

# create separate vectors
average_absolute_latitude <- df$average_absolute_latitude
names(average_absolute_latitude) <- row.names(df)
latitudinal_range <- df$latitudinal_range
names(latitudinal_range) <- row.names(df)

# calculate contrasts
average_absolute_latitude.c <- pic(average_absolute_latitude, tree, scaled=F)
latitudinal_range.c <- pic(latitudinal_range, tree, scaled=F)

# produce scatterplot
plot(average_absolute_latitude.c, latitudinal_range.c)
```

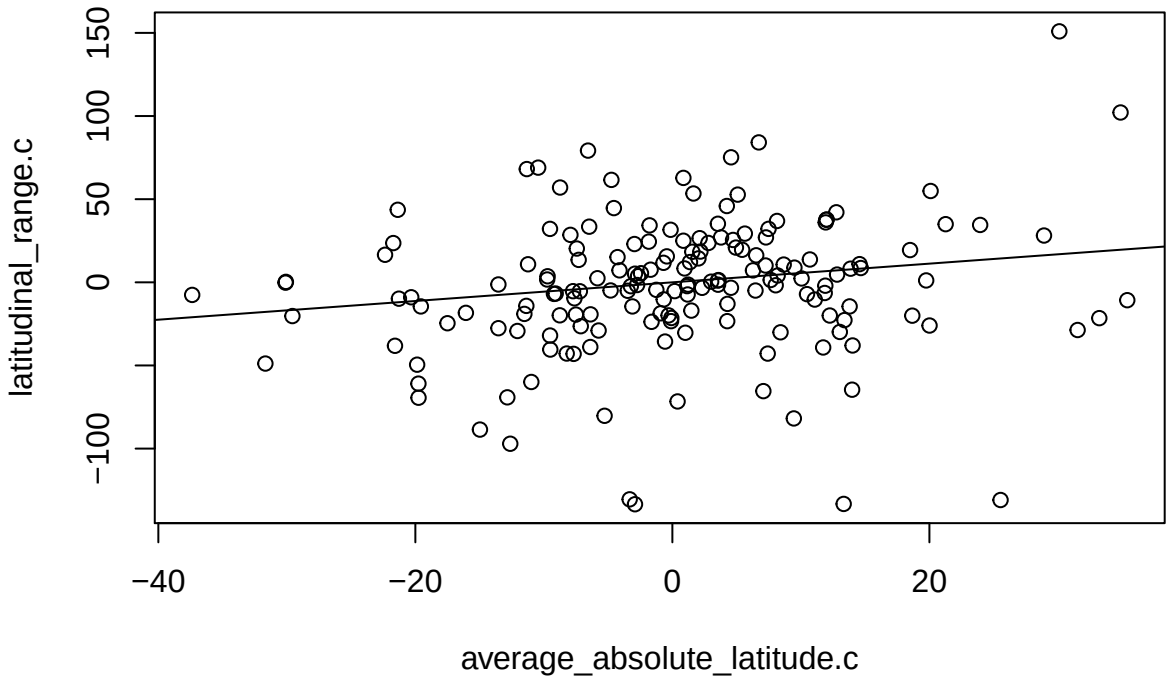


The raw plot is suggestive of a positive correlation, so let's do a linear model:

```
regress <- lm(latitudinal_range.c~average_absolute_latitude.c-1)
summary.lm(regress) # p < 0.05
```

```
##
## Call:
## lm(formula = latitudinal_range.c ~ average_absolute_latitude.c -
##     1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -145.265  -22.312   -1.296   17.851  134.005
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## average_absolute_latitude.c   0.5618     0.2394   2.347   0.0201 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.3 on 170 degrees of freedom
## Multiple R-squared:  0.03138,    Adjusted R-squared:  0.02568
## F-statistic: 5.507 on 1 and 170 DF,  p-value: 0.02009
```

```
# plot with regression line
plot(average_absolute_latitude.c, latitudinal_range.c)
abline(regress)
```



It appears that Rapoport's rule indeed applies ($p<0.05$*)