An Introduction to working with BIEN3

Alton Barbehenn

October 20, 2015

Introduction:

There are good examples of how to search for species ranges and locations, traits, and species at the locations in the BIEN_API_Examples.R file. This will show you how to use each function for a specific entries as well as how to use the functions with vectors. The actual code that should be modified VERY carefully, is called BIEN_API_FILE_Compilation.R. These files were written by Brian Maitner, from the Enquist Lab group at the University of Arizona, as part of his work on the BIEN3 project.

Packages:

The following packages are recommended, some are needed, for working with BIEN3:

```
#most important
  library(RPostgreSQL)

  library(rgeos)
  library(rgdal)
  library(maptools)
  library(sp)
  library(maps)

  library(scales)
  library(grDevices)
#least important
```

These packages are divided into order of importance. The top package RPostgreSQL is used in every function that accesses the BIEN3 servers and it allows you to use SQL while remaining in R enviroment. The middle chunk is used in the BIEN.ranges.species function. The packages allow us to generate a range polygon that can then be plotted onto a map of the world. The final packages apply to the manipulating of color and transparency of the species range polygons. Specifically, scales allows us to use the alpha command to retroactively modify the transparency value of a color and grDevices allows us to generate an arbitrarily large color pallet without repeating colors.

SQL:

SQL is a programming laguage designed for managing a database. One of the basic procedures you can make is a query. A query is made with a SELECT statement which retrieves specific data that matches the specified criteria; in our case, we use the SELECT statement to tell the BIEN3 servers which data fields we want returned. When using a SELECT statement, the statement is followed directly by the column headers that will be retrieved, so any data fields you need should be entered there. The FROM clause is another key part of a SELTECT statement as it directs the search to the correct data table. Finally the WHERE clause further narrows down what is returned by making comparisons and only propogating the data that evaluates as TURE. This helps to remove any data that is irrelevant to our needs. By limiting the range of our data with our FROM and WHERE clauses we are getting exactly what we want, thus optimising the time our queries take and the data retrieved.

It is pretty easy to modify the SQL code in each search to get the data you want. Any valid parameter that is placed in the SELECT section of the query should be returned. In general it is recommended that you use the code closest to what you want to search, i.e. if you're searching for traits, use a trait search command as your base. Take the basic queries for BIEN.gis.family and BIEN.gis.genus for instance. If you wanted

the family search to return the genus of each specimen as well as its family and species, you can change the below given code from:

```
occQuery <- paste("SELECT scrubbed_family, scrubbed_species_binomial, latitude,
                  longitude,date_collected,datasource,custodial_institution_codes,
                  collection_code", paste(cultivated_select,newworld_select),"FROM
                  view_full_occurrence_individual WHERE scrubbed_family in (",
                  paste(shQuote(family, type = "sh"),collapse = ','),")",
                  paste(cultivated_query,newworld_query), " AND higher_plant_group
                  IS NOT NULL AND (is_geovalid = 1 OR is_geovalid IS NULL) ORDER BY
                  scrubbed_species_binomial;")
```

to:

```
occQuery <- paste("SELECT scrubbed_family, scrubbed_genus, scrubbed_species_binomial,
                  latitude,longitude,date_collected,datasource,custodial_institution_codes
                  ,collection_code", paste(cultivated_select,newworld_select),"FROM
                  view_full_occurrence_individual WHERE scrubbed_family in (",
                  paste(shQuote(family, type ="sh"),collapse = ','),")",
                  paste(cultivated_query,newworld_query), " AND higher_plant_group IS NOT
                  NULL AND (is_geovalid = 1 OR is_geovalid IS NULL) ORDER BY
                  scrubbed_species_binomial;")
```

Note that the only difference is the addition of scrubbed_genus to the search. The paste command converts objects into character vectors so that they can be inserted into the SQL string. Most of the string after the SELECT statement probably shouldn't be modified, but if you want to expand your search you can use the expanded function definition to change the cultivated and new world parameters.

My Code:

I wrote two files: PlotMultiple.R and FamilyToSpecies.R. The first file takes a species vector (can be manually entered, or derived as seen in the FamilyToSpecies.R) and plots it onto a map of the world. The species polygons are colored as per the rainbow function which guarantees unique colors for each species. Each polygon has a transparency that is set in the alpha function, this makes it so we can see if polygons overlap and where. The bulk of the function is in this code:

```
  #first gets the ranges for each species in species_vector
  color_vector <- rainbow(len)

  #set up world map
  require(maps)#easy source of maps
  #plots a world map (WGS84 projection), in forest grey
  map('world', fill = TRUE, col = "grey")

#for each species, plot it onto map in different color
for(i in 1:len) {
#you will need to make sure to set your working
#directory or to specify the full filepath
  species_poly<-readShapePoly(species_vector[i])
  plot(species_poly,col=alpha(color_vector[i], 0.3),add=TRUE)
}
```

The second file, FamilyToSpecies.R, takes a set of families and creates a list of species that fall in the family that have been found. It uses the BIEN.gis.family function to get the information about each specimen in the

family's location then takes only the species column and makes a vector containing only one if each species and no NA values. To accomplish this I used the unique and na.omit functions as seen below.

```
#note that the scrubbed_species_binomial is the second column
#when using this version of BIEN.gis.family().
species_vector = family_loc_data[2]
species_vector = unique(species_vector)
species_vector = na.omit(species_vector)
```

We want to remove duplicates because removing redundancies before searching saves us a lot of time given the runtime of many of our search functions. We also need to remove NA values becuase searching for a NA value causes our search function to crash. From this it searches for the traits of each species listed. When using the species vector that is created in FamilyToSpecies.R, it is important to use the following syntax:

```
trait_data <- BIEN.trait.traitbyspecies(trait=trait_vector, species=species_vector[,1])
```

This line searches for the traits in trait vector for each of the species in species vector. The syntax used in species vector is a way of isolating the column of names of species. The general syntax is [row,column].

Sources:

"Selecting Data." SQLCourse. ITBusinessEdge, 2015. Web. 21 Oct. 2015. - I used this page to help write up the select basics.

I used a lot of http://stackoverflow.com/ to help with learning how to use R and with debugging my code.

Created with R-Markdown