Using the RNeotoma API and Package.

by: Simon Goring

RNeotoma is the beginning of an R package that uses an API (or Application Programming Interface). An API allows different applications to talk to one another. In most cases you will see an API being used in a web environment, where commands are passed through the URL using particular, fixed, variables.

The really great thing about R and web-based APIs is that it improves our ability to perform reproducible research. At the end of this excersise you will have the R code required to do the analysis and, because we are interfacing with Neotoma, you and I will both have the same base data sets. If these were exciting and amazing results (and even if they were mediocre, ho-hum results) this would mean that reviewers could be sure the results were accurate, and people hoping to learn from or build on the results could do so more easily. There is an excellent paper by Roger Peng in *Science* (PDF here) that talks more about the principles and importance of reproducible research in the computational sciences.

What does a web API look like?

You use APIs on a daily basis and you might not even know it. Take a second right now to go to Google and type in a search term, then hit enter. You should see something like this:

```
https://www.google.com/#hl=en&safe=on&sclient=psy-
ab&q=why+is+a+horse's+head+in+my+bed%3F&oq=why+is+a+horse's+head+in+my+bed%3F&fp=c5ac5d05622acf2c
```

This is the API that interfaces between your web browser and Google's databases.

You can see I searched for the string “why is a horse's head in my bed”, it is prefaced by “q=”. Google's servers know that when they are reading the URL everything followed by the hash sign (#) is part of a set of variables.

Google has published the Custom Search API on the web here so for your first exercise you should build your own search.

Exercise 1: Build a search request using the Google search API, try to use at least three variables. What did you search for? Did it work?

Neotoma API

Neotoma has the same kind of API, although it is specialized to serve up the paleoecological records stored in the database. We'll discuss the API in more detail, but if you want to spend time on your own, you can check out the reference documents here.

The Neotoma API is just a way for applications on your computer (such as R) to interact with the Neotoma database. It allows you to search for sites, publications, authors and data sets using the Neotoma URL. For example the *sites* command:

```
http://api.neotomadb.org/v1/data/sites?altmin=0&altmax=100
```

will return a crazy wall of text. If you look at it more closely you will see that it starts with the word *success*, and then contains site data for all sites in the Neotoma database that have elevations between 0 and 100ft. Try changing *altmax=100* to *altmax=high*. What happens then? Now, try to figure out that the highest elevation site in Neotoma is (hint, it shows up as 2 separate sites), what is the site name and what is the elevation?

There are functions to return data sets, publications and other database tables from within Neotoma. I've put some R functions up on dropbox, here that are starting to explore these functions. Save the file to your work folder, and then "source" the file from your scratch file:

```
# You should have base.functions.R and your work file in the same
# directory, make sure that your 'working Directory' is set to the same
# place, and then run:

source("base.functions.R")
```

This will run all the functions and place them into memory. From there you can start to use them directly. For this exercise you don't need to know how they all work. I've tried to mirror the Neotoma API as best I can, so you can take a look at the functions and the Neotoma API reference to get a better sense of what's going on, but I will walk you though one function in particular, *get.datasets*. get.datasets is intended to search Neotoma for a number of variables. This:

```
get.datasets <- function(siteid, datasettype, piid, altmin, altmax, loc, gpid,
    taxonids, taxonname, ageold, ageyoung, ageof, subdate) {
    ...
}
```

Is the call that declares the function. What do you think the variable names mean? You can look at the function or the Neotoma API resource for datasets.

If I wanted to get all data sets that had *Picea* pollen in them and then plot them, I could use the command:

```
test <- get.datasets(taxonname = "Picea*")
```

```
## The API call was successful, you have returned  3273 records.
```

but how do I plot them? Use the command to create a variable *test* and then look at how big the variable is, look at the column names. Can you think of some ways to plot the data? What do you see?

```

```

Now we're going to do something awesome. Let's see if we can track the migration of Spruce in North America following deglaciation.

```
# We want only North America.  The bounding box for North America is: LonW
# -167.3, LatS 5.4, LonE -52,2, LatN 83.2 Let's see how many total records
# we could get:

all.NA <- get.datasets(loc = c(-167, 5.4, -83, 52))

# Now limit it to sites in North America with Spruce pollen:
all.NA.pic <- get.datasets(loc = c(-167, 5.4, -83, 52), taxonname = "Picea*")

# We're going to divide up the late-glacial/Holocene into 1000 year bins
# from 14000ybp to the present.  This would make:
year.bins <- seq(14000, -1000, by = -1000)
output.table <- data.frame(no.records = rep(NA, length(year.bins) - 1), average.lat =
rep(NA,
    length(year.bins) - 1), sd.lat = rep(NA, length(year.bins) - 1), average.long =
rep(NA,
    length(year.bins) - 1), sd.long = rep(NA, length(year.bins) - 1))


for (i in 1:(length(year.bins) - 1)) {
    # You may notice that I've clipped part of western North America.  This is
    # because spruce occupies a broad latitudinal gradient in the west, and I
    # don't want it to mess up our results.

    good.sites <- get.datasets(loc = c(-100, 5.4, -52, 83), taxonname = "Picea*",
        ageold = year.bins[i], ageyoung = year.bins[i + 1])

    # good.sites will return a large table, similar to the table you got from
    # test earlier.  We really only want some of the data, the average
    # latitude and longitude of the sites with Spruce in them.
    output.table[i, ] <- c(nrow(good.sites),
mean(as.numeric(good.sites$Site.LatitudeNorth)),
        sd(as.numeric(good.sites$Site.LatitudeNorth)),
mean(as.numeric(good.sites$Site.LongitudeWest)),
        sd(as.numeric(good.sites$Site.LongitudeWest)))

}
```

See if you can plot out the average latitude and longitude of the Spruce pollen records yourself, either on their own (just plotting latitude) or together (just longitude). What is happening? Why?

Your results should look like this:

```
## Warning: package 'maps' was built under R version 2.14.2
```