Neotoma paper

Simon Goring

13 July, 2014

# neotoma: A Programmatic Interface to the Neotoma Paleoecological Database

## Abstract:

Paleoecological data provide an opportunity to study vegetation and climate interactions at time scales that cannot be observed through modern field studies, and allows us to observe changes in so-called 'slow' processes associated with centennial and millennial scale changes in climate.

Here we describe the R package neotoma, to be used to obtain and manipulate paleoecological data from the Neotoma Paleoecological Database. neotoma searches the Neotoma Database for datasets associated with location, taxa, or dataset types using the database's Application Programming Interface. The package can return full datasets or metadata associated with paloecological sites and provides the ability to standardize taxonomies using one of several recognized standard taxonomies from the published literature.

To assist with the use of the neotoma package we provide examples of key functions using examples from the published literature, for both plant and mammal taxa.

## Introduction

Paleoecological data are used to understand patterns and drivers of biogeographical, climatic and evolutionary change at multiple spatial and temporal scales. Paleoecoinformatics ([@brewer2012paleo; @uhen2013card]) is increasingly providing tools to researchers across disciplines to access and use large datasets spanning thousands to millions of years. These datasets may be used to provide better insight into patterns of biomass burning (Blarquez et al, 2013; Power et al.), global vegetation change ([@blois2013modeling; @blarquez2014disentangling]) or changes in physical processes over time ([@goring2012depo]). The increasing interest in uniting ecological and paleoecological data to understand modern ecological patterns and responses to future climate change [@fritz2013diversity; @behrensmeyer2012building; @dietl2011conservation] means that efforts to unite these two, seemingly independednt data-streams will rely, in part, on more robust tools to access and synthesize paleoecological data.

[INSERT NEOTOMA PARAGRAPHS HERE]

The statistical software R ([@RCoreTeam2014]) is commonly used for analysis of paleoecological data and several packages in R exist for analysis (analogue: [@analogue2013; @analogue2007]; rioja: [@rioja2013], Bchron: [@bchron2014], paleofire: [@paleofire2014]). Notwithstanding these packages, the use of extensive paleoecological resources within R has traditionally relied on *ad hoc* methods of obtaining and importing data. This has meant reliance on static online datasets such as the NOAA Paleoclimate repository or North American Modern Pollen Database, and on the distribution of individual datasets from author to analyst.

With an increasing push to provide publications that include numerically reproducible results (e.g., [@goring2012depo; @gill2013linking; @goring2013pollen]) it is important to provide tools that allow analysts to directly access dynamic datasets, and to provide tools to support reproducible workflows. The rOpenSci project has provided a number of tools that can directly interact with Application Programmatic Interfaces (APIs) to access data from a number of databases including rfishbase (FishBase: [@boettiger2012rfishbase]) and taxize (Encyclopedia of Life, iPlant/Taxosaurus and others: [@chamberlain2013taxize]) among others.

(URL) To illustrate use cases for the neotoma package we present examples drawn from the paleoecological literature to illustrate how neotoma provides the tools to perform research that is critical to understanding paleoecological change in the Pleistocene in an open and reproducible manner.

## The neotoma package

## Examples

Macdonald and Cwynar ([@macdonald1991post]) used relative pollen abundances for *Pinus*, expressed as percentages, to map the northward migration of lodgepole pine (*Pinus contorta* var *latifolia*) following deglaciation. In their study a cutoff of 15% Pinus pollen was used to indicate presence of Pinus trees at pollen sample sites (*Pinus* pollen is produced prolifically and is dispersed widely, so a high threshold was used). Recent work by Strong and Hills ([@strong2013holocene]) remapped the migration front using a lower pollen proportion (5%) and more sites. Here we perform a simplified version of the analysis to illustrate the capabilities of the package.

To begin we define a spatial bounding box and a set of taxa of interest. Strong and Hills ([@strong2013holocene]) use a region approximately bounded by 54^oN to the south and 65^oN to the North, and from 110^oW to 130^oW. The command get\_site is used to find all sites within a bounding box:

library(neotoma, ggmap, ggplot2, reshape2, plyr, Bchron, gridExtra)  
all.sites <- get\_site(loc = c(-140, 50, -110, 65))

#> The API call was successful, you have returned 97 records.

**Code Block 1**. *Obtain site data.*

The get\_sites command returns a site data.frame, with siteID, latitude, longitude, altitude, SiteName, and SiteDescription. Each row represents a unique site.

We can see that this returns a total of R nrow(all.sites) sites. Sites are effectively containers for datasets though. Generally it's better to search for datasets, because datasets hold data on species occurrences, abundances, or other measurements of interest. When you search for a dataset you can limit the type of dataset, either by looking for specific taxa, or by describing the dataset type. Here we will restrict our search to the pollen dataset in Neotoma and all taxa beginning with *Pinus*. We use the \* wildcard to indicate any and all taxa beginning with *Pinus* in their name:

all.datasets <- get\_dataset(loc = c(-140, 50, -110, 65),

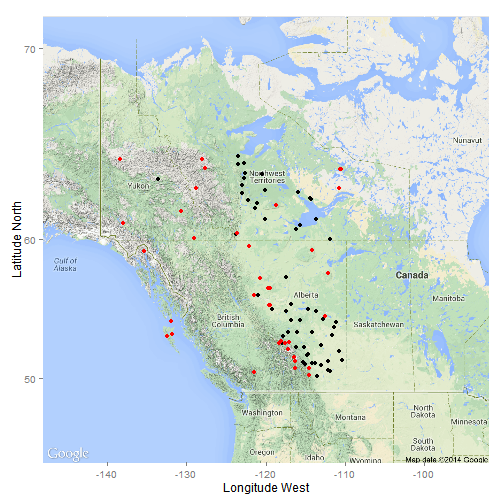
datasettype = "pollen",   
 taxonname = "Pinus\*")

**Code Block 2**. *Obtain datasets.*

A dataset is a larger data object. The dataset has site information, but it also has information about the specific dataset.

Here the API tells us we now have only 42 datasets compared to the 97 sites returned by the get\_sites example. The other sites contain other datasets, e.g. pollen surface samples, or vertebrate fauna, and datasets from fossil pollen cores comprise less than half of the records. We can plot these sites over our original 97.

bc.map <- get\_map(location = c(-120, 60), zoom = 4)  
ggmap(bc.map) + geom\_point(data = all.sites, aes(x = long, y = lat)) + geom\_point(data = get\_site(dataset = all.datasets),   
 aes(x = long, y = lat), color = 2) + xlab("Longitude West") + ylab("Latitude North")



**Figure 1**. Sites returned from Neotoma. Sites in black are returned by get\_sitesand include multiple dataset types; sites in red have fossil pollen datasets.

Let's get the data for each of the sites in our fossil pollen dataset:

# This step may be time consuming, particularly if you have  
# a slow internet connection.  
all.downloads <- suppressMessages(get\_download(sapply(all.datasets, function(x) x$DatasetID)))

In most cases the get\_download command will return a message for an individual core such as:

API call was successful. Returned record for Cottonwood Slough.

The download object is a list with six objects: metadata, sample.meta, chronology, taxon.list, counts, lab.data. The metadata is again a list with a dataset, similar to the one returned by get\_dataset and then pi.data, information about the investigator. The sample.meta is where the depth and age information is stored. The actual chronologies are stored in the chronology list. If a core has a single record the list has a length of one. Some cores have multiple chronologies and these are added to the list. The default chronology is always represented in sample.meta, and is always the first chronology.

If you choose to build your own chronology using Bacon ([@blaauw2011flexible]) or another method you can obtain the chronological controls for the core using the get\_chroncontrol function and the chronology ID in either sample.meta or any one of the chronology objects. Note that the list of chronological controls used to build a chronology for a site may differ among the chronologies for a single site, the default model often contains the most accurate chronological control data.

The taxon.list is a critical component of the download object. It lists the taxa found in the core, as well as any laboratory data, along with the units of measurement and taxonomic grouping. This is important information for determining which taxa make it into pollen percentages. The counts are the actual count or percentage data recorded for the core. The lab.data contains information about any spike used to determine concentrations, sample quantities and, in some cases, charcoal counts.

The pollen taxonomy can vary substantially across sites depending on differences among researchers in identification, or changing taxonomies for species, genera or families over time. Intersite differences in taxonomy is poses a challenge to data synthesis, but can be addressed by using standardized taxonomies (REFS). The neotoma package implements a taxonomic standardizer to attempt to standardize to one of four published taxonomies for the United States and Canada. While this function can be helpful in many cases it should also be used with care. The aggregation table is accessible using data(pollen.equiv) and the function to compile the data is called compile\_list.

For this example we are interested only in the percentage of *Pinus* in the fossil pollen datasets, so we can compile the taxa to the most straightforward taxonomy, 'P25' from Gavin *et al*. ([@gavin2003statistical]). The first record downloaded is Andy Lake, published by Szeicz ([@szeicz1995late]). We can see in the download object that the taxon.table has 5 columns:

``{r, results='as is'} kable(head(all.downloads[[1]]$taxon.list))

Once we apply the compile\_list function to the dataset using the 'P25' compiler:

compiled.cores <- lapply(all.downloads, function(x) compile\_list(x, "P25"))

we can see that the taxon.table now has an extra column (we've removed several columns to improve readability).

``{r, results = 'as.is'} kable(head(compiled.cores[[1]]$taxon.list[,c(1, 5, 6)]))

The function compile\_list returns an object that looks exactly like the download passed to it, however, the taxon.list data.frame gains a column named compressed that links the original taxonomy to the revised taxonomy. This acts as an important check for researchers who choose to use this package for large-scale analysis. Here we see that taxa such as *Potentilla* is lumped into Other, along with spores and other taxa. The compile\_list function can also accept user-defined tables for aggregation if the provided compilations are not acceptable.

In this case the counts look reasonable, and the synonomy appears to have been applied correctly (although we're really only interested in *Pinus*). We now transform our counts into percentages to standardize across cores. We can see what a single core looks like:

# Get the percentage data for the first core:  
core.pct <- as.data.frame(compiled.cores[[1]]$counts /

rowSums(compiled.cores[[1]]$counts)) \*   
 100

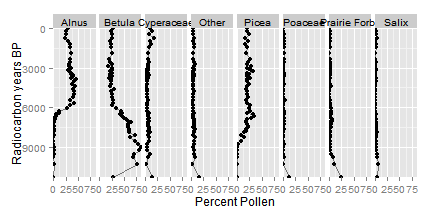
core.pct$depth <- compiled.cores[[1]]$sample.meta$depths  
core.pct$age <- compiled.cores[[1]]$sample.meta$Age  
  
# Eliminate taxa with no samples greater than 5%.  
core.pct <- core.pct[, colSums(core.pct > 5) > 0]  
  
core.data <- melt(core.pct, id = c("depth", "age"))  
  
ggplot(data = core.data, aes(x = value, y = age)) +

geom\_path(alpha = 0.5) +   
geom\_point() + facet\_wrap(~variable, nrow = 1) +

scale\_y\_reverse(expand = c(0, 0)) +

scale\_x\_continuous(breaks = c(0, 25, 50, 75), expand = c(0, 0)) +

xlab("Percent Pollen") + ylab(all.downloads[[1]]$chronologies[[1]]$AgeType[1])



Andy Lake ([@szeicz1995late]) shows major changes in pollen abundances through time, particularly for *Betula* and *Alnus*, but little *Pinus* pollen.

Pollen data is found in the counts slot. We can figure out which sample in a site has the first local *Pinus* presence using a cutoff of 5% ([@strong2013holocene]). Programmatically we can find which rows in the *Pinus* column have presence over 5% and then find the highest row number since age increases with row number.

top.pinus <- function(x) {  
 # Convert the core data into proportions by dividing counts by the sum of  
 # the row.  
 x.pct <- x$counts/rowSums(x$counts)  
   
 # Find the highest row index associated with Pinus presence over 5%  
 oldest.row <- max(which(x.pct[, "Pinus"] > 0.05))  
   
 # return a data.frame with site name and locations, and then the age and  
 # date type associated with the oldest Pinus presence. We preserve  
 # date type since some records have ages in radiocarbon years.

# We will convert these ages to calendar years in a next step.  
   
 data.frame(site = x$metadata$site.data$SiteName,

lat = x$metadata$site.data$LatitudeNorth,

long = x$metadata$site.data$LongitudeWest,

age = x$sample.meta$Age[oldest.row],

date = x$sample.meta$AgeType[oldest.row])  
}  
  
# Apply this function to each core (here we use the plyr functions so we can  
# return a data.frame instead of a list).  
summary.pinus <- ldply(compiled.cores, top.pinus)  
  
# We need to calibrate dates that are recorded in radiocarbon years. In  
# most cases we have no idea what the radiocarbon uncertainty was. For this example I  
# am simply assuming a 100 year SD for calibration. This is likely too  
# conservative.  
radio.years <- summary.pinus$date %in% "Radiocarbon years BP"  
  
calibrated <- BchronCalibrate(summary.pinus$age[radio.years],

ageSds = rep(100, sum(radio.years, na.rm = TRUE)),

calCurves = rep("intcal13", sum(radio.years, na.rm = TRUE)))  
  
wmean.date <- function(x) sum(x$ageGrid \* x$densities/sum(x$densities))  
summary.pinus$age[radio.years] <- sapply(calibrated, wmean.date)  
  
# Can be improved by assuming a monotone smooth spline.  
regress <- ggplot(summary.pinus, aes(x = lat, y = age)) +

geom\_point(aes(color = long), size = 2) +

scale\_y\_reverse(expand = c(0, 100)) +

xlab("Latitude North") +

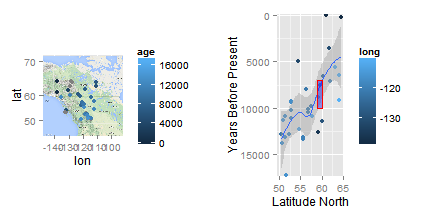
ylab("Years Before Present") +

geom\_smooth(n = 40, method = "loess") +

geom\_rect(aes(xmin = 59, xmax = 60,

ymin = 7000, ymax = 10000), color = 2, fill = "blue", alpha = 0.01)

mapped <- ggmap(bc.map) + geom\_point(data = summary.pinus, aes(x = long, y = lat,   
 color = age), size = 2)  
  
grid.arrange(mapped, regress, nrow = 1)



**Figure 2.** *Panel A shows the date of earliest colonization by Pinus in western North America, in calibrated radiocarbonyears. Panel B shows the same figure with a GAM smooth. The highlighted box indicated the estimate proposed by Strong and Hills [@strong2013holocene] based on their data set.*

And so we see a clear pattern of migration by *Pinus* in northwestern North America. These results match up well with the findings of Strong and Hills ([@strong2013holocene]) who suggest that *Pinus* reached a northern extent between 59 and 60oN at approximately 7 - 10kyr.

### Mammal Distributions in the Pleistocene

Grahm et al. [@graham1996spatial] looked for patterns of change in mammal distributions through the Pleistocene to modern era using fossil assemblages assembled from FAUNMAP. The paper used multiple multivariate analyses to show in part, that mammal species have responded in a Gleasonian manner to climate change since the late-Pleistocene. Their paper shows some species migrating northward in response to warming climates, others staying relatively stable and some moving southward. FAUNMAP has been incorporated into Neotoma, so here we conduct simple analyses of species distributional changes.

First we need to obtain all fossil assemblages from Neotoma for verterbrate fauna,

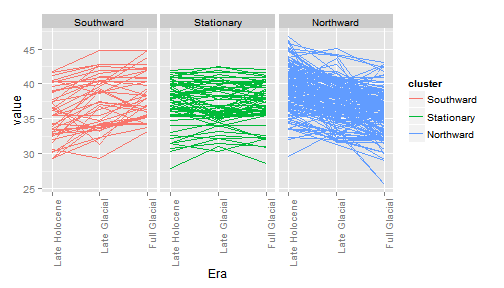
# Bounding box is effectively the continental USA, excluding Alaska.  
mam.set <- get\_dataset(datasettype = "vertebrate fauna", loc = c(-125, 24, -66,   
 49.5))  
  
# Calling this many sites can be very time consuming. It takes  
# approximately an hour to run fully.  
mam.dl <- get\_download(sapply(mam.set, function(x) x$DatasetID))

So, now we have all the sites, we need to bin them into time periods as in Graham et al. [@graham1996spatial]. To do that we first need to build a large table with time and xy coordinates for each site. Time data in sample.meta is not the same as for for pollen data, where many pollen sites contain an age (often mean age) and upper and lower bounds. Most mammal sites have younger and older bounds, but no estimates of exact age. In this case we take a short-cut and simply calculate the age of a sample as the average the younger and older bounds to save the reader from having to examine too much code.

library(plyr)  
  
# To be moved into the neotoma package.  
source("R/compile\_it.R")  
  
compiled.mam <- ldply(mam.dl, .fun = function(x) compile\_it(x), .progress = "text")

# We assign time bins to the data. The command findInterval should tell us  
# if it is in an inteval equivalent to the Modern (0 - 500ybp), Late  
# Holocene (500 - 4000ybp), Early-Mid Holocene (4kyr - 10kyr), Late Glacial  
# (10kyr - 15kyr), Full Glacial (15kyr - 20kyr) or Late Pleistocene  
# (20kyr+).  
time.bins <- c(500, 4000, 10000, 15000, 20000)  
  
# This is not the best option, age bounds cross our pre-defined bins,  
# however solving this is more complex than this example requires.  
mean.age <- apply(compiled.mam[, c("ageold", "ageyoung", "age")], 1, mean, na.rm = TRUE)  
interval <- findInterval(mean.age, time.bins)  
  
compiled.mam$ageInterval <- c("Modern", "Late Holocene", "Early-Mid Holocene",   
 "Late Glacial", "Full Glacial", "Late Pleistocene")[interval + 1]  
  
mam.melt <- melt(compiled.mam, measure.vars = 10:(ncol(compiled.mam) - 1), na.rm = TRUE,   
 factorsAsStrings = TRUE)  
  
mam.melt$ageInterval <- factor(mam.melt$ageInterval, levels = c("Modern", "Late Holocene",   
 "Early-Mid Holocene", "Late Glacial", "Full Glacial", "Late Pleistocene"))  
  
mam.lat <- dcast(data = mam.melt, variable ~ ageInterval, value.var = "lat",   
 fun.aggregate = mean, drop = TRUE)[, c(1, 3, 5, 6)]  
  
# We only want taxa that appear at all time periods:  
mam.lat <- mam.lat[rowSums(is.na(mam.lat)) == 0, ]  
  
# Group the samples based on the range & direction (N vs S) of migration.  
mam.lat$grouping <- factor(findInterval(mam.lat[, 2] - mam.lat[, 4], c(-11,   
 -1, 1, 20)), labels = c("Southward", "Stationary", "Northward"))  
  
  
mam.lat.melt <- melt(mam.lat)  
colnames(mam.lat.melt)[2:3] <- c("cluster", "Era")

ggplot(mam.lat.melt, aes(x = Era, y = value)) + geom\_path(aes(group = variable,   
 color = cluster)) + facet\_wrap(~cluster) + scale\_x\_discrete(expand = c(0.1,   
 0)) + theme(axis.text.x = element\_text(angle = 90, hjust = 1))



So we can see that at this simple test species are not uniformly responding to climatic warming following deglaciation. These findings echo those of Graham et al. [@graham1996science] who showed that taxon response is largely individualistic. While we do see the pre-ponderance of migration is northward, a number of taxa show little migratory response and a number show southward migration. In this example we fail to include movement to the west or east, and ignore the issues that may be associated with the topography of the mountainous west.

# Conclusion

The increasing pressure to develop large-scale databases requires the development of tools that can access the data and can leave reproducible analyses so that others can build from and verify results.

Here we present the neotoma package for R [@RCoreTeam2014] and use examples from the literature to show its utility. neotoma joins a number of other existing packages that are designed either to exploit exisiting paleoecological datasets [@paleofire2014] or to manipulate paleoecological data [@analogue2013; @analogue2007; @rioja2013]. The neotoma package itself is available either from the CRAN repository, or from GitHub where ongoing development continues with help from the public.

The use of the Neotoma database continues to expand, and here we provide researchers with the tools to move analytics to an open framework using R [@RCoreTeam2014] so that methods can be more fully visible.