LinkedEarth - Neotoma P418 Use Case

Step 1: Create a polygon to search for our site.

```
locpoly <- matrix(c(-70,-71,-71,-70,45,45,44,44),ncol=2,byrow = F)
```

Step 2: Convert array to GeoJSON FeatureCollection.

```
#install.packages("geojson")
library(geojson)
```

```
## Warning: package 'geojson' was built under R version 3.4.2
```

```
##
## Attaching package: 'geojson'
```

```
## The following object is masked from 'package:graphics':
##
##     polygon
```

```
poly <- geojson::polygon(paste0('{ "type": "Polygon", "coordinates": [[', "[",locpoly[1,1],", ",locpoly

feature <- geojson::feature(poly)
feature_collection <- geojson::featurecollection(feature)
feature_collection_str <- geo_pretty(feature_collection)
```

Step 3: Set type of search for spatial search.

```
search_type <- "spatial/search/object"
```

Step 4: Set the request domain URL for the geodex web service. Please see http://geodex.org/swagger-ui/ for a complete description of the web service call formats.

```
domain_url <- "http://geodex.org/api/v1/"
request_url <- paste(domain_url, search_type, sep="")
```

Step 5: Create an R list data structure to hold URL submission parameters

```
params_list <- list(geowithin = feature_collection_str)
```

Step 6: Make call to the Geodex RESTful web service using the requests package.

```
library(httr)
r <- GET(request_url, query = params_list)
results <- content(r, "text", encoding = "ISO-8859-1")
```

Step 7: Expore results.

```
library(rjson)
results_json <- fromJSON(results)
features <- results_json$features
number_of_results <- length(features)
first_result <- features[[1]]

url <- first_result$properties$URL
```

Sort results by distance to polygon

```
library(geosphere)
```

```
## Warning: package 'geosphere' was built under R version 3.4.2
```

```
#find nearest coord in each feature
nearestInFeature <- function(feat){
  if(is.list(feat$geometry$coordinates)){
    coords <- matrix(unlist((feat$geometry$coordinates)),ncol=2,byrow = T)
  }else{
    coords <- matrix((feat$geometry$coordinates),ncol=2,byrow = T)
  }
  md =try(min(geosphere::distGeo(coords,locpoly[1,])))
  if(is.numeric(md)){
    return(md)
  }else{
      return(NA)
    }

}
nearestDist <- sapply(results_json$features,nearestInFeature)

#sort by nearest
results_json$features <- results_json$features[order(nearestDist)]



allUrls <- sapply(results_json$features,FUN = function(x){x$properties$URL})
geometry <- first_result$geometry$type
coordinates <- first_result$geometry$coordinates[[1]]


#print first 10 results
print(allUrls[1:10])
```

```
##  [1] "http://data.neotomadb.org/datasets/1490/"
##  [2] "http://data.neotomadb.org/datasets/190/"
##  [3] "http://data.neotomadb.org/datasets/1509/"
##  [4] "http://data.neotomadb.org/datasets/1489/"
##  [5] "http://data.neotomadb.org/datasets/1493/"
##  [6] "http://data.neotomadb.org/datasets/1492/"
##  [7] "http://wiki.linked.earth/NAm-BasinPond.Gajewski.1988"
##  [8] "http://data.neotomadb.org/datasets/237/"
##  [9] "http://data.neotomadb.org/datasets/1494/"
## [10] "http://wiki.linked.earth/NAm-ElephantMountain.Conkey.1994"
```

Numbers 2 & 7 are what we're looking for.

Explore the metadata for the LinkedEarth result

```
dataset_url <- allUrls[7]
```

Step 13: Set the web service call for extracting a set of metadata.

```
search_type <- "graph/details"
```

Step 14: Create the URL for the RESTful web service call.

```
domain_url <- "http://geodex.org/api/v1/"

request_url <- paste(domain_url, search_type, sep="")
```

Step 15: Create the GET parameter for this call.

```
params_list <- list(r = dataset_url)
```

Step 16: Execute the RESTful web service call.

```
r <- GET(request_url, query = params_list)
results <- content(r, "text", encoding = "ISO-8859-1")
```

Step 17: Print the results in a user friendly way.

```
results_json <- fromJSON(results)
print(paste("URI =", results_json$S))
```

```
## [1] "URI = t3522624"
```

```
print(paste("Alternate Name =", results_json$Aname))
```

```
## [1] "Alternate Name = "
```

```
print(paste("Name =", results_json$Name))
```

```
## [1] "Name = "
```

```
print(paste("URL =", results_json$URL))
```

```
## [1] "URL = http://wiki.linked.earth/NAm-BasinPond.Gajewski.1988"
```

```
print(paste("Description =", results_json$Description))
```

```
## [1] "Description = "
```

```
print(paste("Citation =", results_json$Citation))
```

```
## [1] "Citation = t3522626"
```

```
print(paste("Data Published =", results_json$Datepublished))
```

```
## [1] "Data Published = "
```

```
print(paste("Dataset Download Link =", results_json$Curl))
```

```
## [1] "Dataset Download Link = http://wiki.linked.earth/wiki/index.php/Special:WTLiPD?op=export&lipdid=
```

```
print(paste("Keywords =", results_json$Keywords))
```

```
## [1] "Keywords = paleoclimate, climate"
```

```
print(paste("License =", results_json$License))
```

```
## [1] "License = "
```

Load in LinkedEarth LiPD

```
library(lipdR)
#L2 <- readLipd(allUrls[7])
L2 <- readLipd(results_json$Curl)
```

```
## Please enter the dataset name for this file (Name.Location.Year) :
## [1] "reading: Downloads.lpd"
```

Explore the metadata for the Neotoma result

```
dataset_url <- allUrls[2]
```

Step 15: Create the GET parameter for this call.

```
params_list <- list(r = dataset_url)
```

Step 16: Execute the RESTful web service call.

```
r <- GET(request_url, query = params_list)
results <- content(r, "text", encoding = "ISO-8859-1")
```

Step 17: Print the results in a user friendly way.

```
results_json <- fromJSON(results)
print(paste("URI =", results_json$S))
```

```
## [1] "URI = t3611777"
```

```
print(paste("Alternate Name =", results_json$Aname))
```

```
## [1] "Alternate Name = "
```

```
print(paste("Name =", results_json$Name))
```

```
## [1] "Name = "
```

```
print(paste("URL =", results_json$URL))
```

```
## [1] "URL = http://data.neotomadb.org/datasets/190/"
```

```
print(paste("Description =", results_json$Description))
```

```
## [1] "Description = "
```

```
print(paste("Citation =", results_json$Citation))
```

```
## [1] "Citation = "
```

```
print(paste("Data Published =", results_json$Datepublished))
```

```
## [1] "Data Published = "
```

```
print(paste("Dataset Download Link =", results_json$Curl))
```

```
## [1] "Dataset Download Link = "
```

```
print(paste("Keywords =", results_json$Keywords))
```

```
## [1] "Keywords = "
```

```
print(paste("License =", results_json$License))
```

```
## [1] "License = https://creativecommons.org/licenses/by/4.0/deed.en_US"
```

Load in Neotoma file as a LiPD

```
library(geoChronR)
library(neotoma)
```

```
##
## Attaching package: 'neotoma'
```

```
## The following object is masked from 'package:lipdR':
```

```
##
##     get_table
```

```
site = get_site("Basin Pond")
```

```
## The API call was successful, you have returned  1 records.
```

```
site2 = data.frame(site.id = 234)
L = neotoma2Lipd(site = site)
```

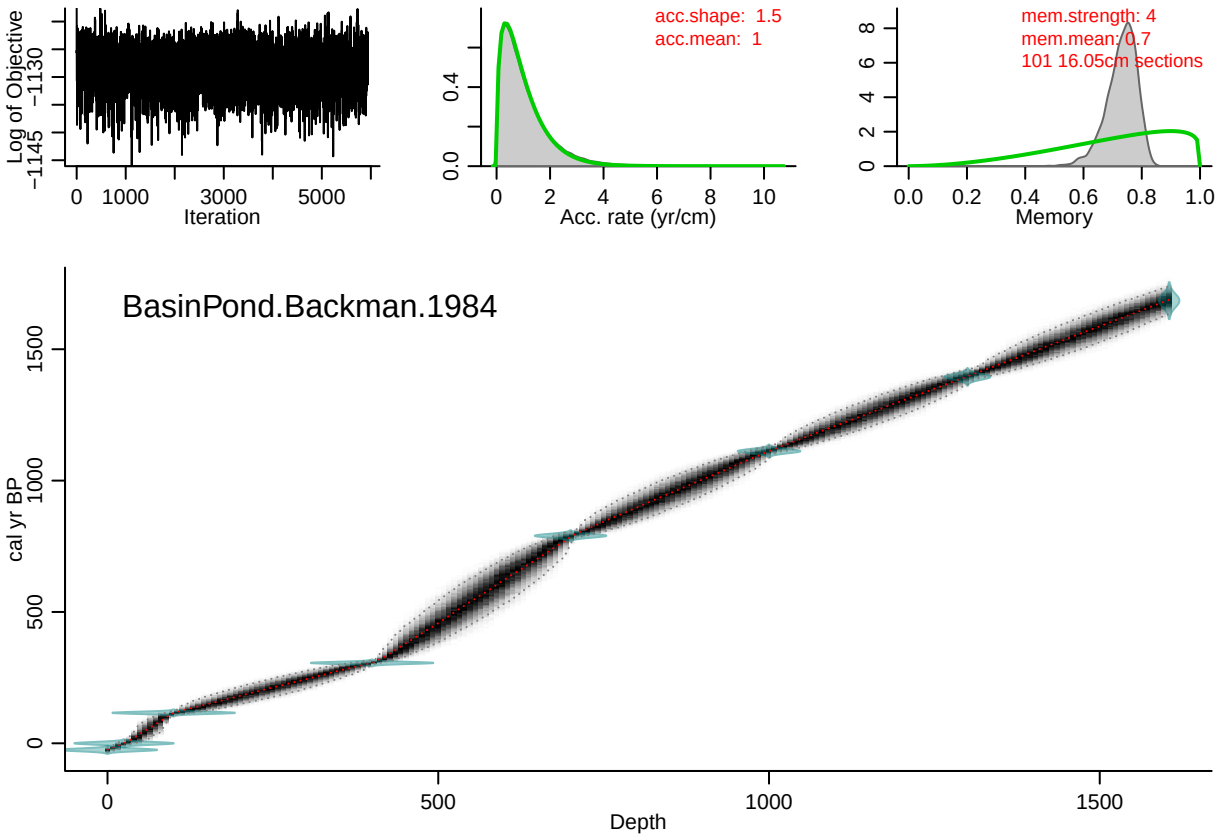
```
## API call was successful. Returned record for Basin Pond
```

```
## The API call was successful, you have returned  6 records.
## API call was successful. Returned chronology.
## API call was successful. Returned chronology.
## API call was successful. Returned chronology.
```

```
#estimate uncertainty from range
L$chronData[[2]]$measurementTable[[1]] = estimateUncertaintyFromRange(L$chronData[[2]]$measurementTable
```

```
detach("package:geojson",character.only = TRUE)
```

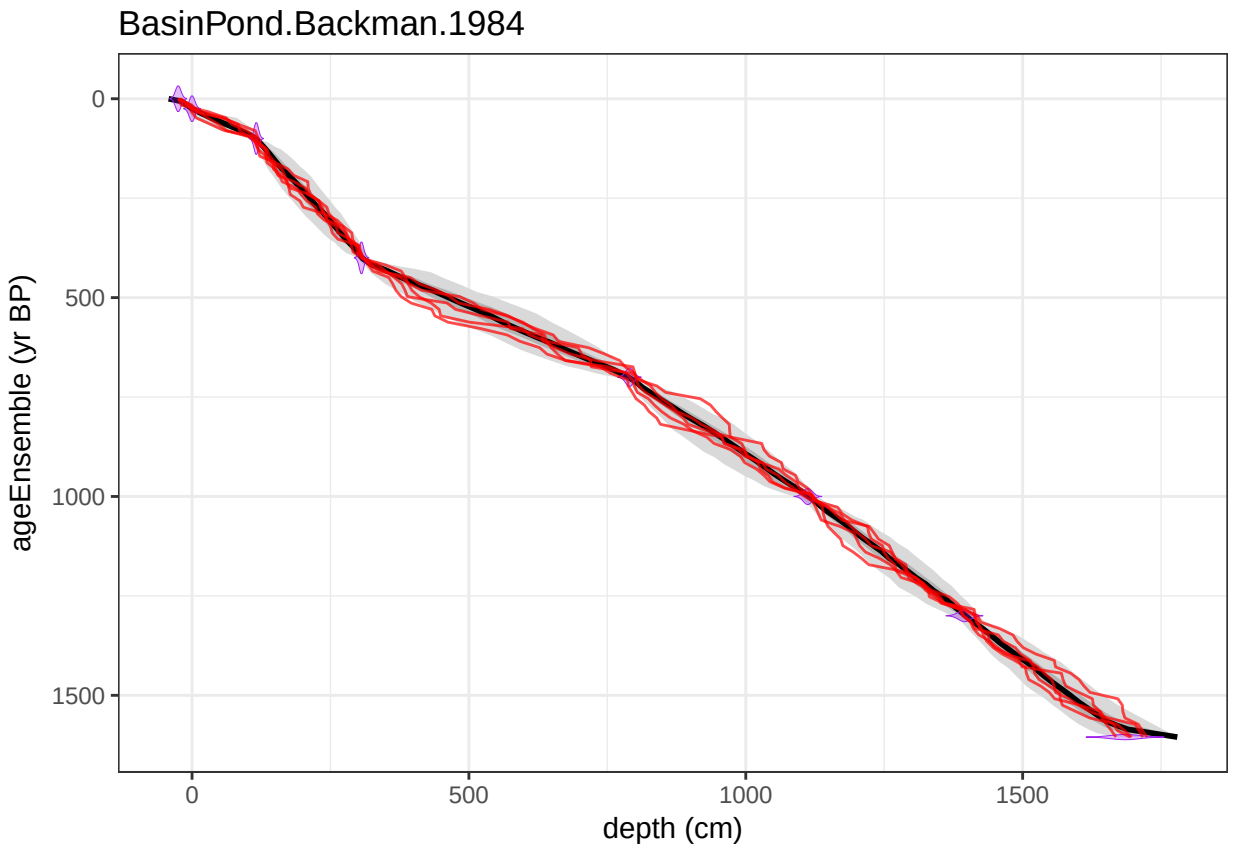
```
## [1] "Looking for laboratory ID..."
## [1] "Found it! Moving on..."
## [1] "Looking for radiocarbon ages..."
## [1] "Looking for radiocarbon age uncertainty..."
## [1] "Looking for calibrated ages..."
## [1] "Found it! Moving on..."
## [1] "Looking for calibrated age uncertainty..."
## [1] "Found it! Moving on..."
## [1] "Looking for depth..."
## [1] "Found it! Moving on..."
## [1] "Looking for radiocarbon reservoir age offsets (deltaR)..."
## [1] "can also use radiocarbon reservoir ages if need be..."
## [1] "Looking for radiocarbon reservoir age uncertainties..."
## [1] "Looking for column of reject ages, or ages not included in age model"
## [1] 8
##    id  age error depth cc dR dSTD ta tb
## 1 330  -25     5     0  0  0    0 33 34
## 2 331    0     5    25  0  0    0 33 34
## 3 332  116     4   100  0  0    0 33 34
## 4 333  306     4   400  0  0    0 33 34
## 5 334  790     7   700  0  0    0 33 34
## 6 335 1112     8  1000  0  0    0 33 34
## 7 336 1395    11  1300  0  0    0 33 34
## 8 337 1685    26  1605  0  0    0 33 34
## Hi there, welcome to Bacon for Bayesian age-depth modelling
##  Using calibration curve specified within the .csv file,
```




```
##  Warning, this will take quite some time to calculate. I suggest increasing d.by to, e.g. 10
## Mean 95% confidence ranges 92.5 yr, min. 16.7 yr at 404 cm, max. 179.7 yr at 578 cm
##
## [1] "taking a short break..."
```

```
plotChron(L,chron.number = 2,model.num = 2)
```

```
## [1] "Found it! Moving on..."
## [1] "Found it! Moving on..."
```




```
L$paleoData[[1]]$measurementTable[[1]]$temperature <- L2$paleoData[[1]]$measurementTable[[1]]$temperatu
```

```
L <- mapAgeEnsembleToPaleoData(L = L,which.chron = 2,which.paleo = 1,which.model = 2,which.ens = 1,whic
```

```
## [1] "BasinPond.Backman.1984"
## [1] "Looking for age ensemble...."
## [1] "Found it! Moving on..."
## [1] "Found it! Moving on..."
## [1] "getting depth from the paleodata table..."
## [1] "Found it! Moving on..."
```

Now plot temperature with age uncertainty

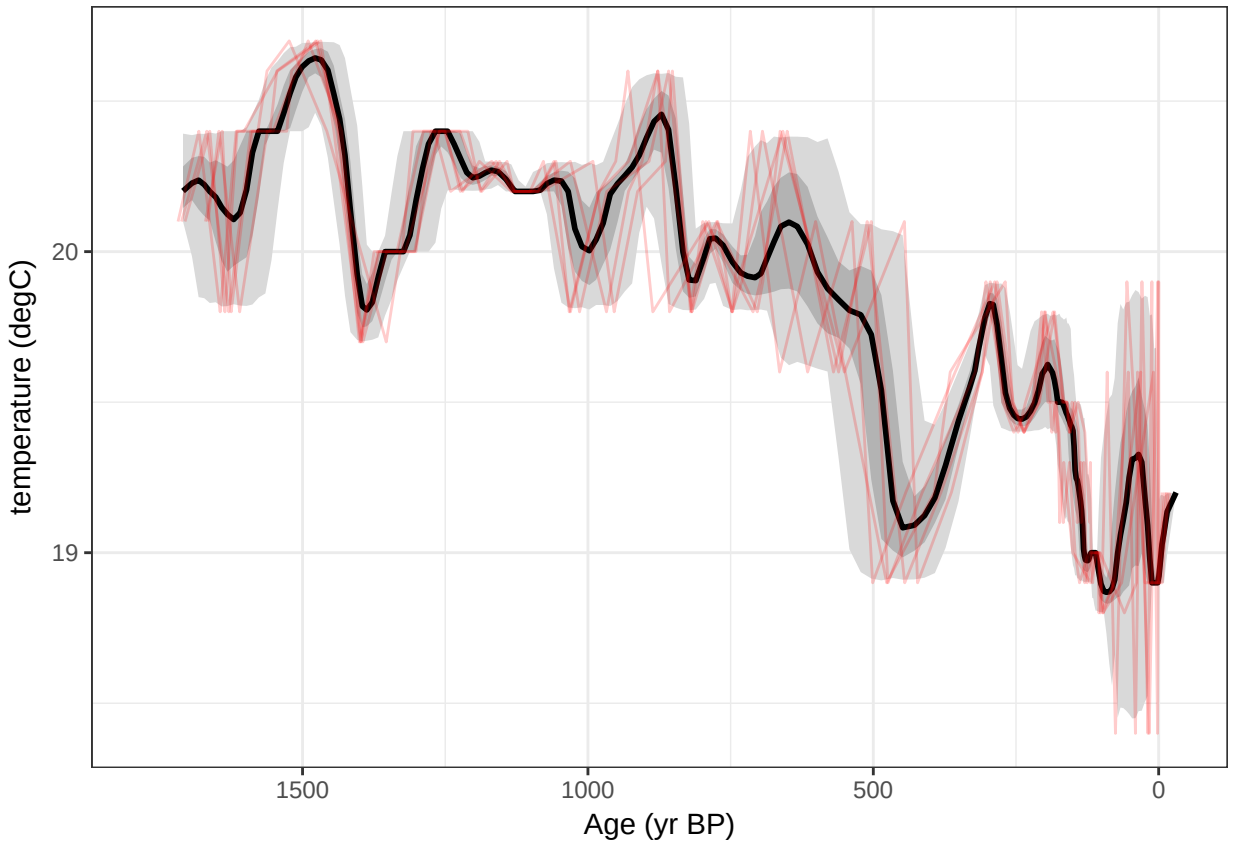
```
library(ggplot2)
temp <- selectData(L,"temperature")
```

```
## [1] "Found it! Moving on..."
```

```
ageEnsemble <- selectData(L,"ageEnsemble")
```

```
## [1] "Found it! Moving on..."
```

```
plot <- plotTimeseriesEnsRibbons(ageEnsemble,temp)
plot <- plotTimeseriesEnsLines(ageEnsemble,temp,color = "red",add.to.plot = plot,maxPlotN = 5)+
  scale_x_reverse("Age (yr BP)")
print(plot)
```


temperature (degC)
20
19
1500
1000
500
0
Age (yr BP)