

1 rgbif: R client for working with GBIF species occurrence data

2 Scott Chamberlain^{*,a}

3 ^a*University of California, Berkeley, CA, USA*

4 **Abstract**

- 5 1. xxx
- 6 2. xxx
- 7 3. xxx
- 8 4. xxxx

9 **Introduction**

10 Perhaps the most fundamental element in many fields of ecology is the individual. The number of
11 individuals of each species in a given location forms the basis for many sub-fields of ecology and evolution.
12 Some research questions necessitate collecting new data, while others can easily take advantage of
13 existing data. In fact, some ecology fields are built largely on existing data, e.g., macro-ecology (Brown,
14 1995; Beck et al., 2012).

15 Data on individuals, including which species, and where they're found, can be used for a large number of
16 research questions. Biodiversity records have been used for a suite of other use cases: validating habitat
17 suitability models with real occurrence data (Ficetola et al., 2014); ancestral range reconstruction
18 (Ferretti et al., 2015; María Mendoza et al., 2015); development of invasive species watch lists (Faulkner
19 et al., 2014); evaluating risk of invasive species spread (Febbraro et al., 2013); and effects of climate
20 change on future biodiversity (Brown et al., 2015).

21 In addition to wide utility, this data is important for conservation. Biodiversity loss is one of the greatest
22 challenges of our time (Pimm et al., 2014), and some have called this the sixth great mass extinction
23 (Ceballos et al., 2015). Given this challenge there is a great need for data on specimen records, whether
24 collected from live sightings in the field or specimens in museums.

25 There are many online services that collect and maintain specimen records. However, Global Biodiversity
26 Information Facility (hereafter, GBIF, <http://www.gbif.org>) is the largest collection of biodiversity

*Corresponding author

Email address: `scott(at)ropensci.org` (Scott Chamberlain)

27 records globally, currently with 640 million records, 1.6 million taxa, 15,000 datasets from 780 publishers
28 (current as of 2015-11-25). Many large biodiversity warehouses such as iNaturalist (<http://www.inaturalist.org>), VertNet (<http://vertnet.org>), and USGS's Biodiversity Information Serving Our Nation
29 (BISON; <http://bison.usgs.ornl.gov>) all feed into GBIF.
30

31 Herein, we describe the `rgbif` software library (Chamberlain et al.) for working with GBIF data in
32 the R programming environment (R Core Team, 2014). R is a widely used language in academia, and
33 in non-profit and private sectors. Importantly, R makes it easy to do all of the steps of the research
34 process, including data management, data manipulation and cleaning, statistics, and vizualization.
35 Thus, an R client for getting GBIF data is a powerful tool to facilitate reproducible research.

36 **The `rgbif` package**

37 The `rgbif` package is nearly completely written in R (a small Javascript library is included for reading
38 well known text), uses an [MIT license](#) to maximize use everywhere. `rgbif` is developed publicly
39 on GitHub at <https://github.com/ropensci/rgbif>, where development versions of the package can be
40 installed, and bugs and feature requests reported. Stable versions of `rgbif` can be installed from [CRAN](#),
41 the distribution network for R packages. `rgbif` is part of the rOpenSci project (<http://ropensci.org>), a
42 developer network making R software to facilitate reproducible research.

43 *Package interface*

44 `rgbif` is designed following the [GBIF Application Programming Interface](#), or API. The GBIF API has
45 four major components: registry, taxonomic names, occurrences, and maps. We also include functions to
46 interface with the OAI-PMH GBIF service; only dataset information is available vis this service, however.
47 We ignore maps in `rgbif` as it is concerned with generating maps primarily for web applications. `rgbif`
48 has a suite of functions dealing with each of registry, taxonomic names, and occurrences - we'll go
49 through each in turn describing design and example usage.

50 *GBIF feedback loop*

51 With each request `rgbif` makes to GBIF's API, we send request headers that tell GBIF that the request
52 is coming from `rgbif`, including what version of the package. This helps GBIF know what proportion
53 of requests are coming from this package, and therefore R, as most requests likely will come from `rgbif`;
54 this information is helpful for them in thinking about how people are using GBIF data.

55 *Registry*

56 The GBIF registry API services are spread across five sets of functions via the main API:

- 57 • Datasets
- 58 • Installations
- 59 • Networks
- 60 • Nodes
- 61 • Organizations

62 And dataset information in general is available via the OAI-PMH service, functions in `rgbif` prefixed
63 with `gbif_oai_`.

64 Datasets are owned by organizations. Organizations are endorsed by nodes to share datasets with GBIF.
65 Datasets are published through institutions, which may be hosted at another organization. A network
66 is a group of datasets (managed by GBIF). Datasets are the units that matter the most with respect
67 to registry information, while installations, networks, nodes, and organizations are simply higher level
68 organizational structure.

69 *Datasets*

70 Dataset functions include search, dataset metadata retrieval, and dataset metrics. Searching for datasets
71 is an important part of the discovery process. One can search for datasets on the GBIF web portal.
72 However, programmatic searching using this package is much more powerful. Identifying datasets
73 appropriate for a research question is helpful as you can get metadata for each dataset, and track down
74 dataset specific problems, if any.

75 The `dataset_search()` function is one way to search for datasets. Here, we search for the query term
76 “oregon”, which finds any datasets that have terms matching that term.

```
res <- dataset_search(query = "oregon")
res$data$datasetTitle[1:10]
#> [1] "SDNHM Birds Collection"
#> [2] "CM Birds Collection"
#> [3] "condoncollection"
```

```
#> [4] "Taxonomy in Flux Checklist"
#> [5] "Wool carder bees of the genus Anthidium in the Western Hemisphere"
#> [6] "Bryophyte Collection - University of Washington Herbarium (WTU)"
#> [7] "University of British Columbia Herbarium (UBC) - Bryophytes Collection"
#> [8] "UWFC Ichthyology Collection"
#> [9] "Lichen Collection - University of Washington Herbarium (WTU)"
#> [10] "UWBM Mammalogy Collection"
```

77 See also `datasets()` and `dataset_suggest()` for searching for datasets.

78 *Dataset metrics.* Dataset metrics are another useful way of figuring out what datasets you may want to
 79 use. One drawback is that these metrics data are only available for datasets of type *checklist*, but there
 80 are quite a lot of them (2580).

81 Here, we search for dataset metrics for a single dataset, with uuid `ec93a739-1681-4b04-b62f-3a687127a17f`,
 82 a checklist of the ants (Hymenoptera: Formicidae) of the World.

```
res <- dataset_metrics(uuid='ec93a739-1681-4b04-b62f-3a687127a17f')
data.frame(rank = names(res$countByRank),
           count = unname(unlist(res$countByRank)))
```

rank	count
SPECIES	13710
SUBSPECIES	3234
GENUS	726
TRIBE	53
SUBFAMILY	20
FAMILY	2
KINGDOM	1
PHYLUM	1
CLASS	1
ORDER	1

83 *Networks, nodes, and installations*

84 Networks, nodes and installations are at a higher level of organization above datasets, but can be useful
85 if you want to explore data from given organizations. Here, we search for the first 10 GBIF networks,
86 returning just the title field.

```
networks(limit=10)$data$title
#> [1] "GBIF Backbone Sources"
#> [2] "Canadensys"
#> [3] "Southwest Collections of Arthropods Network (SCAN)"
#> [4] "VertNet"
#> [5] "Dryad"
#> [6] "GBIF Network"
#> [7] "The Knowledge Network for Biocomplexity (KNB) "
#> [8] "Online Zoological Collections of Australian Museums (OZCAM)"
#> [9] "Catalogue of Life"
#> [10] "Ocean Biogeographic Information System (OBIS)"
```

87 *Taxonomic names*

88 The GBIF taxonomic names API services are spread across five functions:

- 89 • Search GBIF name backbone - `name_backbone()`
- 90 • Search across all checklists - `name_lookup()`
- 91 • Quick name lookup - `name_suggest()`
- 92 • Name usage of a name according to a checklist - `name_usage()`
- 93 • GBIF name parser - `parsenames()`

94 The goal of these name functions is often to settle on a taxonomic name known to GBIF's database.
95 This serves two purposes: 1) when referring to a taxonomic name, you can point to a URI on the
96 internet, and 2) you can search for metadata on a taxon, and occurrences of that taxon in GBIF.

97 Taxonomic names are particularly tricky. Many different organizations have their own unique codes for
98 the same taxonomic names, and some taxonomic groups have preferred sources for the definitive names

99 for that group. That's why it's best to determine what name GBIF uses, and its associated identifier,
100 for the taxon of interest instead of simply searching for occurrences with a taxonomic name.

101 When searching for occurrences (see below) you can search by taxonomic name (and other filters, e.g.,
102 taxonomic rank), but you're probably better off figuring out the taxonomic key in the GBIF backbone
103 taxonomy, and using that to search for occurrences. The **taxonkey** parameter in the GBIF occurrences
104 API expects a GBIF backbone taxon key.

105 *GBIF Backbone*

106 The GBIF backbone taxonomy is used in GBIF to have a consistent way to refer to taxonomic
107 names throughout their services. The backbone has 4410899 unique names and 2497114 species
108 names. The backbone taxonomy is also a dataset with key d7dddbf4-2cf0-4f39-9b2a-bb099caae36c
109 (<http://www.gbif.org/dataset/d7dddbf4-2cf0-4f39-9b2a-bb099caae36c>).

110 We can search the backbone taxonomy with the function **name_backbone()**. Here, we're searching for
111 the name *Poa*, restricting to genera, and the family *Poaceae*.

```
res <- name_backbone(name='Poa', rank='genus', family='Poaceae')
res[c('usageKey', 'kingdom')]
#> $usageKey
#> [1] 2704173
#>
#> $kingdom
#> [1] "Plantae"
```

112 *Name searching*

113 One of the quickest ways to search for names is using **name_suggest()**, which does a very quick search
114 and returns minimal data. Here, we're searching for the query term *Pum*, and we get back many names:

```
name_suggest(q='Pum', limit = 6)
```

key	canonicalName	rank
3269133	Pumilus	GENUS

key	canonicalName	rank
4407849	Pumilinura	GENUS
4323990	Pumiliopsis	GENUS
4324083	Pumiliopes	GENUS
4161281	Pumilea	GENUS
4312370	Pumilopagurus	GENUS

With these results, you can then proceed to search for occurrences with the taxon key(s), or drill down further with other name searching functions to get the exact taxon of interest.

Occurrences

GBIF provides two ways to get occurrence data: through the `/occurrence/search` route (see `occ_search()`), or via the `/occurrence/download` route (many functions, see below). `occ_search()` is the main function for the search route, and is more appropriate for smaller data, while `occ_download*()` functions are more appropriate for larger data requests.

Large is of course a subjective term. When you hit a “large dataset” will depend primarily on the size of the your data request. GBIF imposes for any given search a limit of 200,000 records in the search service, after which point you can’t download any more records for that search. However, you can download more records for different searches.

We think the search service is still quite useful for many people even given the 200,000 limit. For those that need more data, we have created a similar interface in the `download_*()` functions, that should be easy to use. Users should take note that using the download service has a few extra steps to get data into R, but is straight-forward.

The download service, like the occurrence search service, is rate-limited. That is, you can only have one to three downloads running simultaneously for your user credentials. However, simply check when a download job is complete, then you should be able to start a new download request.

Download API

The download API syntax is similar to the occurrence search API in that the same parameters are used, but the way in which the query is defined is different. For example, in the download API you can do

136 greater than searches (i.e., `latitude > 50`), whereas you can not do that in the occurrence search API.
137 Thus, unfortunately, we couldn't make the query interface exactly the same for both search and download
138 functions.

139 Using the download service can consist of as few as three steps: 1) Request data via a search; 2)
140 Download data; 3) Import data into R.

141 Request data download given a query. Here, we search for the taxon key 3119195, which is the key for
142 *Helianthus annuus* (<http://www.gbif.org/species/3119195>).

```
occ_download('taxonKey = 3119195')  
#> <<gbif download>>  
#> Username: xxxx  
#> E-mail: xxxx  
#> Download key: 0000840-150615163101818
```

143 You can check on when the download is ready using the functions `occ_download_list()` and
144 `occ_download_meta()`. When it's ready use `occ_download_get()` to download the dataset to your
145 computer.

```
(res <- occ_download_get("0000840-150615163101818", overwrite = TRUE))  
#> <<gbif downloaded get>>  
#> Path: ./0000840-150615163101818.zip  
#> File size: 3.19 MB
```

146 What's printed out above is a very brief summary of what was downloaded, the path to the file, and its
147 size (in human readable form).

148 Next, read the data in to R using the function `occ_download_import()`.

```
library("dplyr")  
dat <- occ_download_import(res)  
dat %>%  
  select(gbifID, decimalLatitude, decimalLongitude)  
#>      gbifID decimalLatitude decimalLongitude
```



```

#> 1 725767384      61.01005      24.41740
#> 2 725767447      59.82923      23.13550
#> 3 725767450      60.38505      25.17449
#> 4 725767513      68.37648      23.51963
#> 5 725767546      67.19203      24.85820
#> 6 725767579      60.21607      24.67412
#> 7 725767609      66.49260      25.70471
#> 8 725767645      61.36634      24.76218
#> 9 725767678      62.29174      27.96500
#> 10 725767681      60.28615      22.38489
#> ..      ...      ...

```

149 *Downloaded data format.* The downloaded dataset from GBIF is actually a Darwin Core Archive
 150 (DwC-A), an internationally recognized biodiversity informatics standard (<http://rs.tdwg.org/dwc/>).
 151 The DwC-A downloaded is a compressed folder with a number of files, including metadata, citations for
 152 each of the datasets included in the download, and the data itself, in separate files for each dataset as
 153 well as one single `.txt` file. In `occ_download_import()`, we simply fetch data from the `.txt` file. If you
 154 want to dig into the metadata, citations, etc., it is easily accessible from the folder on your computer.

155 *Search API*

156 The search API follows the GBIF API and is broken down into the following functions:

- 157 • Get a single numeric count of occurrences - `occ_count()`
- 158 • Search for occurrences - `occ_search()`
- 159 • A simplified and optimized version of `occ_search()` - `occ_data()`
- 160 • Get occurrences by occurrence identifier - `occ_get()`
- 161 • Get occurrence metadata - `occ_metadata()`

162 *Search for occurrences.* The main search work-horse is `occ_search()`. This function allows very flexible
 163 search definitions. In addition, this function does paging internally, making it such that the user does
 164 not have worry about the 300 records per request limit - but of course we can't go over the 200,000
 165 maximum limit.

166 The output of `occ_search()` borrows the tidy `data.frame` idea from the `dplyr` R package, so that no
 167 matter how large the `data.frame`, the output is easily assessed because only a few of the records (rows)
 168 are shown, only a few columns are shown (with others shown in name only), and metadata is shown on
 169 top of the `data.frame` to indicate data found and returned, media records found, unique taxonomic
 170 hierarchies returned, and the query executed.

171 The output of these examples, except one, aren't shown, but all run correctly.

172 Search by species name, using `name_backbone()` first to get key

```
(key <- name_suggest(q = 'Helianthus annuus', rank = 'species')$key[1])
#> [1] 3119195
occ_search(taxonKey = key, limit = 2)
#> Records found [21577]
#> Records returned [2]
#> No. unique hierarchies [1]
#> No. media records [2]
#> Args [taxonKey=3119195, limit=2, offset=0, fields=all]
#> First 10 rows of data
#>
#>           name           key decimalLatitude decimalLongitude
#> 1 Helianthus annuus 1143516596          35.42767         -105.0688
#> 2 Helianthus annuus 1095851641           0.00000           0.0000
#> Variables not shown: issues (chr), datasetKey (chr), publishingOrgKey
#> (chr), publishingCountry (chr), protocol (chr), lastCrawled (chr),
#> lastParsed (chr), extensions (chr), basisOfRecord (chr), taxonKey
#> (int), kingdomKey (int), phylumKey (int), classKey (int), orderKey
#> (int), familyKey (int), genusKey (int), speciesKey (int),
#> scientificName (chr), kingdom (chr), phylum (chr), order (chr),
#> family (chr), genus (chr), species (chr), genericName (chr),
#> specificEpithet (chr), taxonRank (chr), dateIdentified (chr), year
#> (int), month (int), day (int), eventDate (chr), modified (chr),
#> lastInterpreted (chr), references (chr), identifiers (chr), facts
#> (chr), relations (chr), geodeticDatum (chr), class (chr), countryCode
```

```
#>      (chr), country (chr), rightsHolder (chr), identifier (chr),
#>      verbatimEventDate (chr), datasetName (chr), gbifID (chr),
#>      verbatimLocality (chr), collectionCode (chr), occurrenceID (chr),
#>      taxonID (chr), recordedBy (chr), catalogNumber (chr),
#>      http...unknown.org.occurrenceDetails (chr), institutionCode (chr),
#>      rights (chr), occurrenceRemarks (chr), identificationID (chr),
#>      elevation (dbl), elevationAccuracy (dbl), stateProvince (chr),
#>      recordNumber (chr), locality (chr), municipality (chr), language
#>      (chr), type (chr), ownerInstitutionCode (chr), identifiedBy (chr)
```

173 Instead of getting a taxon key first, you can search for a name directly

```
occ_search(scientificName = 'Ursus americanus')
```

174 Search for many species

```
splist <- c('Cyanocitta stelleri', 'Junco hyemalis', 'Aix sponsa')
keys <- sapply(splist, function(x) name_suggest(x)$key[1], USE.NAMES = FALSE)
occ_search(taxonKey = keys, limit = 5, return = 'data')
```

175 Spatial search, based on well known text format, or a bounding box set of four coordinates

```
# well known text
occ_search(geometry = 'POLYGON((30.1 10.1, 10 20, 20 40, 40 40, 30.1 10.1))', limit = 20)
# bounding box
occ_search(geometry = c(-125.0,38.4,-121.8,40.9), limit = 20)
```

176 Get only occurrences with lat/long data

```
occ_search(hasCoordinate = TRUE, limit = 5)
```

177 Get only those occurrences with spatial issues

```
occ_search(hasGeospatialIssue = TRUE, limit = 5)
```

178 *Data cleaning.* GBIF provides optional data issues with each occurrence record. These issues fall into
179 many different pre-defined classes, covering issues with taxonomic names, geographic data, and more
180 (see `occ_issues_lookup()` to find out more information on GBIF issues; and the same data on [GBIF's](#)
181 [development site](#)).

182 `occ_issues()` provides a way to easily filter data downloaded via `occ_search()` based on GBIF issues.

```
out <- occ_search(issue = 'DEPTH_UNLIKELY', limit = 500)
NROW(out)
#> [1] 4
out %>% occ_issues(-cudc) %>% .$data %>% NROW
#> [1] 2
```

183 *Mapping*

184 An obvious downstream use case for species occurrence data is to map the data. `rgbif` per se is largely
185 not concerned with making this easier, although we do have a simple wrapper around `ggplot2` to make
186 it easy to get a quick plot of occurrence data. For example, here we plot 100 occurrences for *Puma*
187 *concolor*.

```
key <- name_backbone(name='Puma concolor')$speciesKey
dat <- occ_search(taxonKey = key, limit = 100, hasCoordinate = TRUE)
gbifmap(dat$data)
```

188 Another package, [spoccutils](#)

189 *GBIF data in other R packages*

190 We discuss usage of GBIF data in other R packages throughout the manuscript, but provide a synopsis
191 here for clarity.

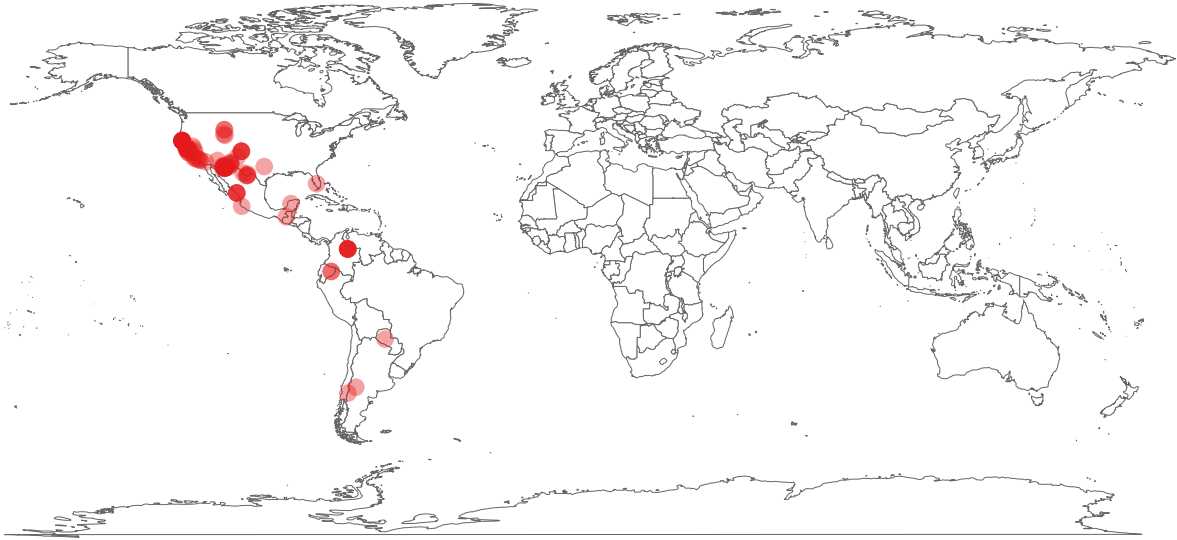


Figure 1:

192 *taxize*

193 Some of the GBIF taxonomic services are also available in [taxize](#), an R package that focuses on getting
 194 data from taxonomic data sources on the web. For example, with `get_gbifid()` one can get GBIF IDs
 195 used for a set of taxonomic names - then use those IDs in other functions in *taxize* to get additional
 196 information, like taxonomically downstream children.

197 *spocc*

198 GBIF occurrence data is available in the R package [spocc](#) via `rgbif`. *spocc* is a unified interface
 199 for fetching species occurrence data from many sources on the web. For example, a user can collect
 200 occurrence data from GBIF, iDigBio, and iNaturalist, and easily combine them, then use other packages
 201 to clean and visualize the data.

202 Use cases

203 The following are three use cases for `rgbif`: niche modelling, spatial change in biodiversity, and
 204 distribution mapping.

205 *Ecological niche modelling*

206 In this example, we plot actual occurrence data for *Bradypus* species against a single predictor variable,
207 BIO1 (annual mean temperature). This is only one step in a species distribution modelling workflow.
208 This example can be done using BISON data as well with our rbison package.

209 *Load libraries*

```
library("rgbif")
library("dismo")
library("maptools")
library("plyr")
```

210 *Raster files*

211 Make a list of files that are installed with the dismo package, then create a rasterStack from these

```
files <- list.files(paste(system.file(package = "dismo"), "/ex", sep = ""),
                   "grd", full.names = TRUE)
predictors <- stack(files)
```

212 *Get world boundaries*

```
data(wrld_simpl)
```

213 *Get GBIF data using the rOpenSci package rgbif*

```
nn <- name_lookup("bradypus*", rank = "species")
nn <- na.omit(unique(nn$data$subKey))
df <- occ_search(taxonKey = nn, hasCoordinate = TRUE, limit = 500)
df <- df[ apply(df, function(x) class(x$data)) %in% "data.frame" ]
df <- ldply(lapply(df, "[", "data"))
df2 <- df[,c('decimalLongitude', 'decimalLatitude')]
```

214 *Plot*

215 (1) Add raster data, (2) Add political boundaries, (3) Add the points (occurrences)

```
plot(predictors, 1)
plot(wrld_simpl, add = TRUE)
points(df2, col = "blue")
```

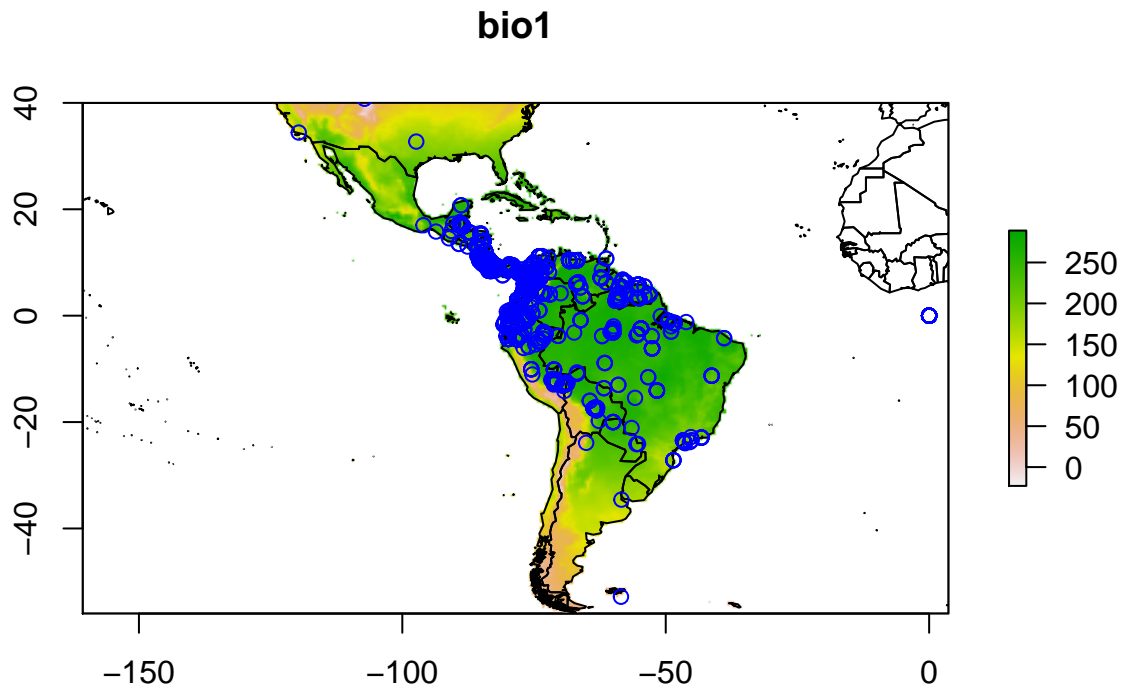


Figure 2:

216 *Biodiversity in big cities*

217 In this example, we collect specimen records across different cities using GBIF data from the `rgbif`
218 package.

219 *Load libraries*

```
library("rgbif")
library("ggplot2")
```



```

scale_color_continuous(low = "#60E1EE", high = "#0404C8") +
labs(x="", y="") +
theme_grey(base_size=14) +
theme(legend.position = "bottom", legend.key = element_blank()) +
guides(color = guide_legend(keywidth = 2))

```

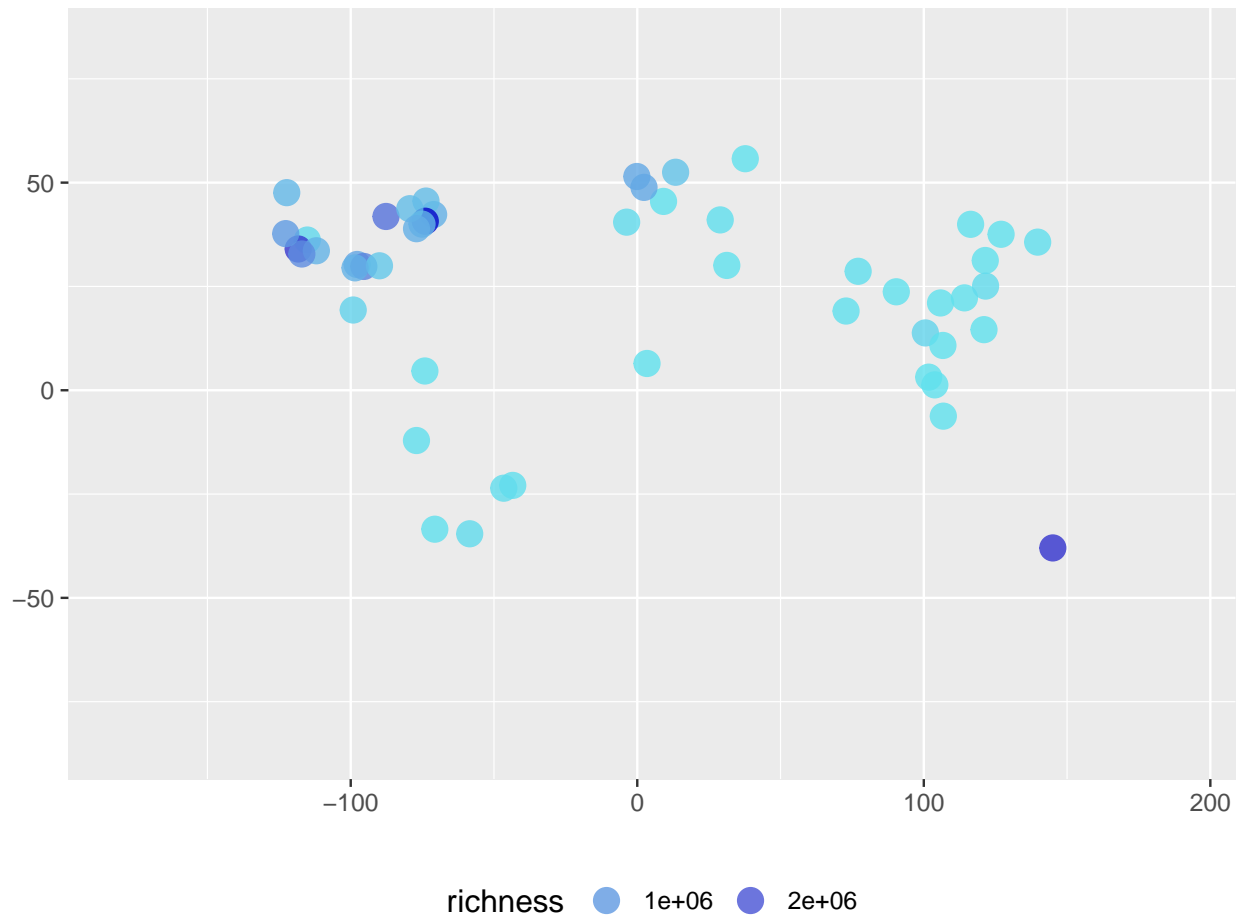


Figure 3:

225 *Valley oak occurrence data comparison*

226 This example comes from [Antonio J. Perez-Luque](#) who [shared his plot on Twitter](#). Antonio compared
 227 the occurrences of Valley Oak (*Quercus lobata*) from [GBIF](#) to the distribution of the same species from
 228 the [Atlas of US Trees](#).

229 *Load libraries*

```
library('rgbif')
library('raster')
library('sp')
library('maptools')
library('rgeos')
library('scales')
```

230 *Get GBIF Data for Quercus lobata*

```
keyQ1 <- name_backbone(name='Quercus lobata', kingdom='plants')$speciesKey
dat.Q1 <- occ_search(taxonKey=keyQ1, return='data', limit=50000)
```

231 *Get Distribution map of Q. lobata Atlas of US Trees (Little, E.)*

232 From <http://esp.cr.usgs.gov/data/little/>. And save shapefile in same directory

```
url <- 'http://esp.cr.usgs.gov/data/little/querloba.zip'
tmp <- tempdir()
download.file(url, destfile = "~/querloba.zip")
unzip("~/querloba.zip", exdir = tmp)
q1 <- readShapePoly(file.path(tmp, "querloba.shp"))
```

233 *Get Elevation data of US*

```
alt.USA <- getData('alt', country = 'USA')
```

234 *Create Hillshade of US*

```
alt.USA <- alt.USA[[1]]
slope.USA <- terrain(alt.USA, opt = 'slope')
aspect.USA <- terrain(alt.USA, opt = 'aspect')
hill.USA <- hillShade(slope.USA, aspect.USA, angle = 45, direction = 315)
```

235 *Plot map*

```

plot(hill.USA, col = grey(0:100/100), legend = FALSE, xlim = c(-125, -116), ylim = c(32, 42), ma
# add shape from Atlas of US Trees
plot(ql, add = TRUE, col = alpha("white", 0.6), border = FALSE)
# add Gbif presence points
points(dat.Ql$decimalLongitude, dat.Ql$decimalLatitude, cex = .7, pch = 19, col = alpha("darkgre
legend(x = -121, y = 40.5, "GBIF Data", pch = 19, col = 'darkgreen', bty = 'n', pt.cex = 1, cex :
legend(x = -121, y = 41.5, "Atlas of United States Trees \n (Little, E. 1971)", pt.cex = 1.5, ce

```

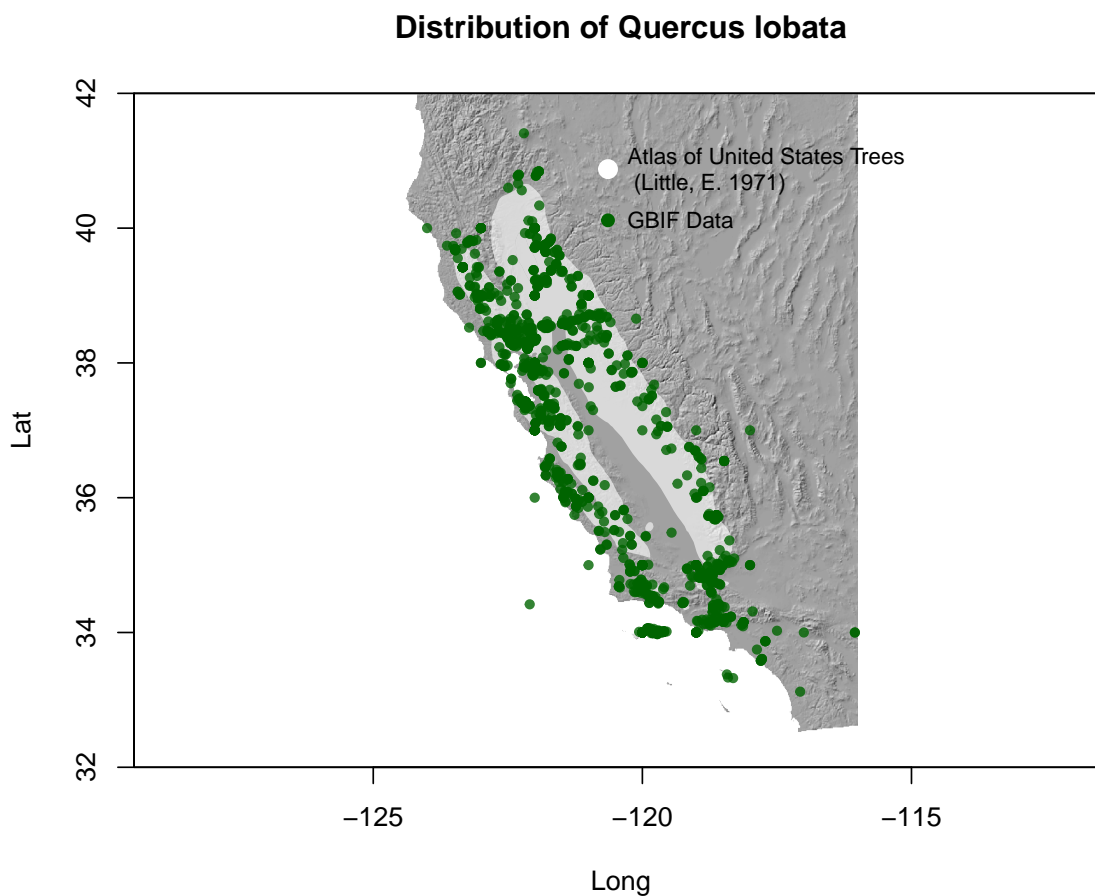


Figure 4:

Conclusions and future directions

The `rgbif` R package provides a programmatic interface to GBIF's application programming interface (API) - a powerful tool for making research using species occurrence data reproducible. In fact, the

239 `rgbif` package has already been used in 22 scholarly publications (as of 2015-11-14).

240 The `rgbif` package is relatively stable, and should not have many breaking changes unless necessitated
241 due to changes in the GBIF API.

242 One area of focus in the future is to attempt to solve many use cases that have been brought up with
243 respect to GBIF data. For example, some specimens are included in GBIF that are located in botanical
244 gardens. For many research questions, researchers are interested in “wild” type occurrences, not those
245 in human curated scenarios. Making removal of these occurrences easy would be very useful, but is
246 actually quite a hard problem.

247 **Acknowledgements**

248 This project was supported in part by the Alfred P Sloan Foundation (Grant 2013-6-22), and the
249 Helmsley Foundation (Grant XXXXXX).

250 **Data Accessibility**

251 All scripts and data used in this paper can be found in the permanent data archive Zenodo under
252 the digital object identifier (DOI). This DOI corresponds to a snapshot of the GitHub repository at
253 github.com/sckott/msrgbif. Software can be found at <https://github.com/ropensci/rgbif>, under an
254 MIT license.

255 **References**

256 Beck J., Ballesteros-Mejia L., Buchmann CM., Dengler J., Fritz SA., Gruber B., Hof C., Jansen
257 F., Knapp S., Kreft H., Schneider A-K., Winter M., Dormann CF. 2012. Whats on the horizon for
258 macroecology? *Ecography* 35:673–683.

259 Brown JH. 1995. *Macroecology*. University of Chicago Press.

260 Brown KA., Parks KE., Bethell CA., Johnson SE., Mulligan M. 2015. Predicting plant diversity patterns
261 in madagascar: Understanding the effects of climate and land cover change in a biodiversity hotspot.
262 *PLOS ONE* 10:e0122721.

263 Ceballos G., Ehrlich PR., Barnosky AD., Garcia A., Pringle RM., Palmer TM. 2015. Accelerated
264 modern human-induced species losses: Entering the sixth mass extinction. *Science Advances* 1:e1400253–
265 e1400253.

266 Chamberlain S., Ram K., Barve V., Mcglinn D. *Rgbif: Interface to the global 'biodiversity' information*
267 *facility 'aPI'*.

268 Faulkner KT., Robertson MP., Rouget M., Wilson JR. 2014. A simple, rapid methodology for developing
269 invasive species watch lists. *Biological Conservation* 179:25–32.

270 Febbraro MD., Lurz PWW., Genovesi P., Maiorano L., Girardello M., Bertolino S. 2013. The use of
271 climatic niches in screening procedures for introduced species to evaluate risk of spread: A case with
272 the american eastern grey squirrel. *PLoS ONE* 8:e66559.

273 Ferretti F., Verd GM., Seret B., Šprem JS., Micheli F. 2015. Falling through the cracks: The fading
274 history of a large iconic predator. *Fish and Fisheries*:n/a–n/a.

275 Ficetola GF., Rondinini C., Bonardi A., Baisero D., Padoa-Schioppa E. 2014. Habitat availability for
276 amphibians and extinction threat: A global analysis. *Diversity and Distributions* 21:302–311.

277 María Mendoza., Ospina OE., Cárdenas-Henao H., García-R JC. 2015. A likelihood inference of
278 historical biogeography in the world's most diverse terrestrial vertebrate genus: Diversification of
279 direct-developing frogs (craugastoridae: Pristimantis) across the neotropics. *Molecular Phylogenetics*
280 *and Evolution* 85:50–58.

281 Pimm SL., Jenkins CN., Abell R., Brooks TM., Gittleman JL., Joppa LN., Raven PH., Roberts CM.,
282 Sexton JO. 2014. The biodiversity of species and their rates of extinction, distribution, and protection.
283 *Science* 344:1246752–1246752.

284 R Core Team. 2014. *R: A language and environment for statistical computing*. Vienna, Austria: R
285 Foundation for Statistical Computing.