Guide to using the ecoengine R package

The Berkeley Ecoengine (http://ecoengine.berkeley.edu) provides an open API to a wealth of museum data contained in the Berkeley natural history museums. This R package provides a programmatic interface to this rich repository of data allowing for the data to be easily analyzed and visualized or brought to bear in other contexts. This vignette provides a brief overview of the package's capabilities.

The API documentation is available at http://ecoengine.berkeley.edu/developers/. As with most APIs it is possible to query all the available endpoints that are accessible through the API itself. Ecoengine has something similar.

```
library(ecoengine)
ee_about()
```

Table 1: Table continues below

type
wieslander_vegetation_type_mapping
wieslander_vegetation_type_mapping
wieslander_vegetation_type_mapping
wieslander_vegetation_type_mapping
data
data
data
data
actions
meta-data
meta-data
meta-data

endpoint
https://ecoengine.berkeley.edu/api/vtmplots_trees/
https://ecoengine.berkeley.edu/api/vtmplots/
https://ecoengine.berkeley.edu/api/vtmplots_brushes/
https://ecoengine.berkeley.edu/api/vtmveg/
https://ecoengine.berkeley.edu/api/checklists/
https://ecoengine.berkeley.edu/api/sensors/
https://ecoengine.berkeley.edu/api/observations/
https://ecoengine.berkeley.edu/api/photos/
https://ecoengine.berkeley.edu/api/search/
https://ecoengine.berkeley.edu/api/layers/
https://ecoengine.berkeley.edu/api/rstore/
https://ecoengine.berkeley.edu/api/sources/

The ecoengine class

The data functions in the package include ones that query obervations, checklists, photos, and vegetation records. These data are all formatted as a common `S3` class called `ecoengine`. The class includes 4 slots.

- [`Total results on server`] A total result count (not necessarily the results in this particular object but the total number available for a particlar query)
- [`Args`] The arguments (So a reader can replicate the results or rerun the query using other tools.)
- [`Type`] The type (`photos`, `observation`, or `checklist`)
- [`Number of results retrieved`] The data. Data are most often coerced into a `data.frame`. To access the data simply use `result_object$data`.

The default `print` method for the class will summarize the object.

Notes on downloading large data requests

For the sake of speed, results are paginated at `1000` results per page. It is possible to request all pages for any query by specifying `page = all` in any function that retrieves data. However, this option should be used if the request is reasonably sized. With larger requests, there is a chance that the query might become interrupted and you could lose any data that may have been partially downloaded. In such cases the recommended practice is to use the returned observations to split the request. You can always check the number of requests you'll need to retreive data for any query by running `ee_pages(obj)` where `obj` is an object of class `ecoengine`.

```
request <- ee_photos(county = "Santa Clara County", quiet = TRUE, progress = FALSE)
# Use quiet to suppress messages. Use progress = FALSE to suppress progress
# bars which can clutter up documents.
ee_pages(request)
```

```
#>  [1] 1
```

```
# Now it's simple to parallelize this request You can parallelize across
# number of cores by passing a vector of pages from 1 through the total
# available.
```

Specimen Observations

The database contains over 2 million records (3386177 total). Many of these have already been georeferenced. There are two ways to obtain observations. One is to query the database directly based on a partial or exact taxonomic match. For example

```
pinus_observations <- ee_observations(scientific_name = "Pinus", page = 1, quiet = TRUE,
    progress = FALSE)
pinus_observations
```

```
#>  [Total results on the server]: 58875
#>  [Args]:
#>  country = United States
#>  scientific_name = Pinus
#>  extra = last_modified
#>  georeferenced = FALSE
#>  page_size = 1000
#>  page = 1
#>  [Type]: FeatureCollection
#>  [Number of results retrieved]: 1000
```

For additional fields upon which to query, simply look through the help for `?ee_observations`. In addition to narrowing data by taxonomic group, it's also possible to add a bounding box (add argument `bbox`) or request only data that have been georeferenced (set `georeferenced = TRUE`).

```
lynx_data <- ee_observations(genus = "Lynx", georeferenced = TRUE, quiet = TRUE,
    progress = FALSE)
lynx_data
```

```
#>  [Total results on the server]: 725
#>  [Args]:
#>  country = United States
#>  genus = Lynx
#>  extra = last_modified
#>  georeferenced = True
#>  page_size = 1000
#>  page = 1
#>  [Type]: FeatureCollection
#>  [Number of results retrieved]: 725
```

```
# Notice that we only for the first 1000 rows.  But since 795 is not a big
# request, we can obtain this all in one go.
lynx_data <- ee_observations(genus = "Lynx", georeferenced = TRUE, page = "all",
    progress = FALSE)
```

```
#>  Search contains 725 observations (downloading 1 of 1 pages)
```

```
lynx_data
```

```
#>  [Total results on the server]: 725
#>  [Args]:
#>  country = United States
#>  genus = Lynx
#>  extra = last_modified
#>  georeferenced = True
#>  page_size = 1000
#>  page = all
#>  [Type]: FeatureCollection
#>  [Number of results retrieved]: 725
```

Other search examples

```
animalia <- ee_observations(kingdom = "Animalia")
Artemisia <- ee_observations(scientific_name = "Artemisia douglasiana")
asteraceae <- ee_observationss(family = "asteraceae")
vulpes <- ee_observations(genus = "vulpes")
Anas <- ee_observations(scientific_name = "Anas cyanoptera", page = "all")
loons <- ee_observations(scientific_name = "Gavia immer", page = "all")
plantae <- ee_observations(kingdom = "plantae")
# grab first 10 pages (250 results)
plantae <- ee_observations(kingdom = "plantae", page = 1:10)
chordata <- ee_observations(phylum = "chordata")
# Class is clss since the former is a reserved keyword in SQL.
aves <- ee_observations(clss = "aves")
```

Additional Features

As of July 2014, the API now allows you exclude or request additional fields from the database, even if they are not directly exposed by the API. The list of fields are:

`id`, `record`, `source`, `remote_resource`, `begin_date`, `end_date`, `collection_code`, `institution_code`, `state_province`, `county`, `last_modified`, `original_id`, `geometry`, `coordinate_uncertainty_in_meters`, `md5`, `scientific_name`, `observation_type`, `date_precision`, `locality`, `earliest_period_or_lowest_system`, `latest_period_or_highest_system`, `kingdom`, `phylum`, `clss`, `order`, `family`, `genus`, `specific_epithet`, `infraspecific_epithet`, `minimum_depth_in_meters`, `maximum_depth_in_meters`, `maximum_elevation_in_meters`, `minimum_elevation_in_meters`, `catalog_number`, `preparations`, `sex`, `life_stage`, `water_body`, `country`, `individual_count`, `associated_resources`

To request additional fields

Just pass then in the `extra` field with multiple ones separated by commas.

```
aves <- ee_observations(clss = "aves", extra = "kingdom,genus")

#>  Search contains 237619 observations (downloading 1 of 238 pages)

#>    |                                                                              |

names(aves$data)

#>   [1] "longitude"                       "latitude"
#>   [3] "type"                            "state_province"
#>   [5] "coordinate_uncertainty_in_meters" "recorded_by"
#>   [7] "begin_date"                      "end_date"
#>   [9] "source"                          "url"
#>  [11] "country"                         "scientific_name"
#>  [13] "locality"                        "record"
#>  [15] "remote_resource"                 "last_modified"
#>  [17] "kingdom"                         "genus"
#>  [19] "observation_type"
```

Similarly use `exclude` to exclude any fields that might be returned by default.

```
aves <- ee_observations(clss = "aves", exclude = "source,remote_resource")

#>  Search contains 237619 observations (downloading 1 of 238 pages)

#>    |                                                                              |

names(aves$data)

#>   [1] "longitude"                       "latitude"
#>   [3] "type"                            "state_province"
#>   [5] "coordinate_uncertainty_in_meters" "recorded_by"
#>   [7] "begin_date"                      "end_date"
#>   [9] "locality"                        "url"
#>  [11] "country"                         "scientific_name"
#>  [13] "record"                          "last_modified"
#>  [15] "observation_type"
```

Mapping observations

The development version of the package includes a new function `ee_map()` that allows users to generate interactive maps from observation queries using Leaflet.js.

```
lynx_data <- ee_observations(genus = "Lynx", georeferenced = TRUE, page = "all",
    quiet = TRUE)
ee_map(lynx_data)
```

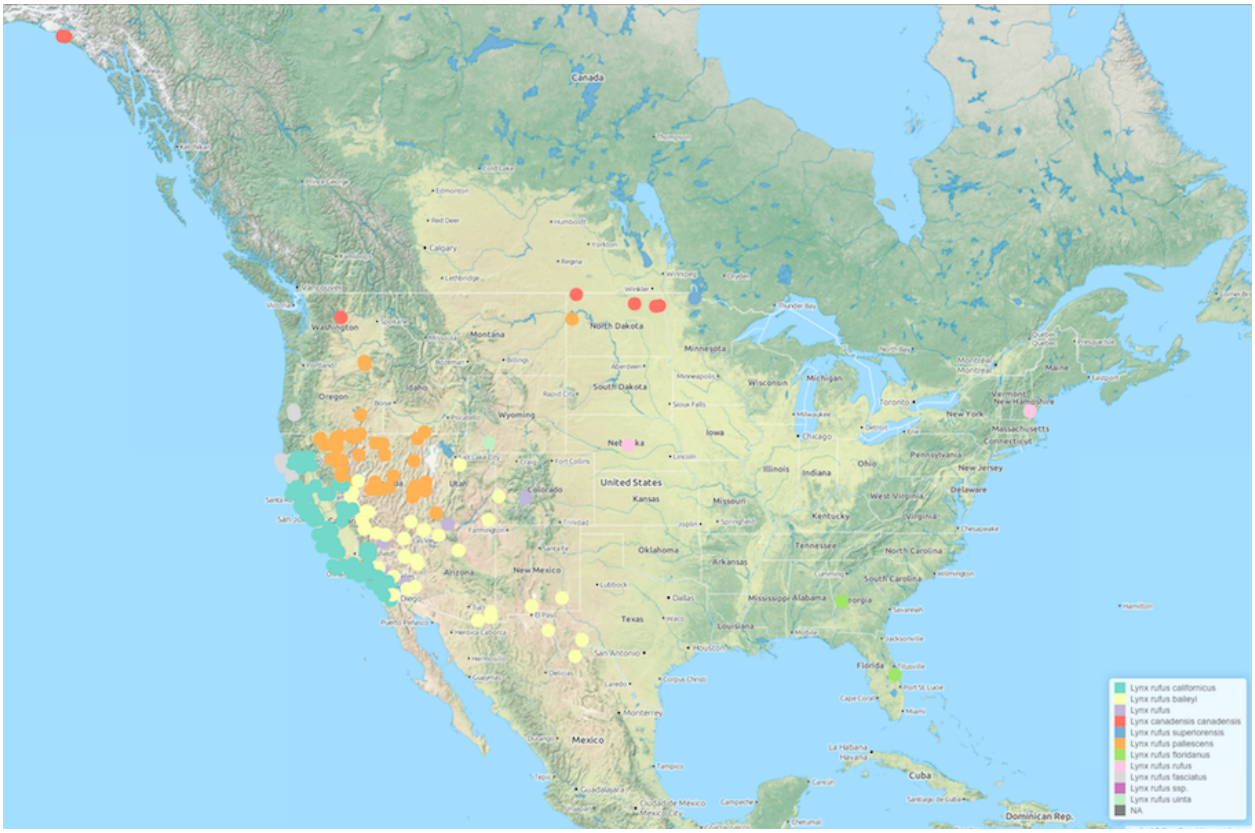

Figure 1: Map of Lynx observations across North America

Photos

The ecoengine also contains a large number of photos from various sources. It's easy to query the photo database using similar arguments as above. One can search by taxa, location, source, collection and much more.

```
photos <- ee_photos(quiet = TRUE, progress = FALSE)
photos
```

```
#>  [Total results on the server]: 72454
#>  [Args]:
#>  page_size = 1000
#>  georeferenced = 0
#>  page = 1
```

```
#>  [Type]: photos
#>  [Number of results retrieved]: 1000
```

The database currently holds 72454 photos. Photos can be searched by state province, county, genus, scientific name, authors along with date bounds. For additional options see `?ee_photos`.

Searching photos by author

```
charles_results <- ee_photos(author = "Charles Webber", quiet = TRUE, progress = FALSE)
charles_results
```

```
#>  [Total results on the server]: 3656
#>  [Args]:
#>  page_size = 1000
#>  authors = Charles Webber
#>  georeferenced = FALSE
#>  page = 1
#>  [Type]: photos
#>  [Number of results retrieved]: 1000
```

```
# Let's examine a couple of rows of the data
charles_results$data[1:2, ]
```

```
#>           authors                                  locality          county
#>  1 Charles Webber    Yosemite National Park, Badger Pass Mariposa County
#>  2 Charles Webber Yosemite National Park, Yosemite Falls Mariposa County
#>    photog_notes
#>  1      Tan Oak
#>  2         <NA>
#>                                                                                  url
#>  1 https://ecoengine.berkeley.edu/api/photos/CalPhotos%3A8076%2B3101%2B2933%2B0025/
#>  2 https://ecoengine.berkeley.edu/api/photos/CalPhotos%3A8076%2B3101%2B0667%2B0107/
#>    begin_date end_date geojson.type   longitude  latitude
#>  1       <NA>     <NA>        Point -119.657387 37.663724
#>  2       <NA>     <NA>        Point -119.597389 37.753851
#>                          record
#>  1 CalPhotos:8076+3101+2933+0025
#>  2 CalPhotos:8076+3101+0667+0107
#>                                                  remote_resource
#>  1 http://calphotos.berkeley.edu/cgi/img_query?seq_num=21272&one=T
#>  2 http://calphotos.berkeley.edu/cgi/img_query?seq_num=14468&one=T
#>    collection_code              scientific_name
#>  1      CalAcademy Notholithocarpus densiflorus
#>  2      CalAcademy     Rhododendron occidentale
#>                                                                                                  url
#>  1 https://ecoengine.berkeley.edu/api/observations/CalPhotos%3A8076%2B3101%2B2933%2B0025%3A1/
#>  2 https://ecoengine.berkeley.edu/api/observations/CalPhotos%3A8076%2B3101%2B0667%2B0107%3A1/
#>                                                        media_url
#>  1 http://calphotos.berkeley.edu/imgs/512x768/8076_3101/2933/0025.jpeg
#>  2 http://calphotos.berkeley.edu/imgs/512x768/8076_3101/0667/0107.jpeg
#>                                      source
#>  1 https://ecoengine.berkeley.edu/api/sources/9/
#>  2 https://ecoengine.berkeley.edu/api/sources/9/
```

Browsing these photos

```
view_photos(charles_results)
```

This will launch your default browser and render a page with thumbnails of all images returned by the search query. You can do this with any `ecoengine` object of type `photos`. Suggestions for improving the photo browser are welcome.

Figure 2:

Other photo search examples

```
# All the photos in the CDGA collection
all_cdfa <- ee_photos(collection_code = "CDFA", page = "all", progress = FALSE)
# All Racoon pictures
racoons <- ee_photos(scientific_name = "Procyon lotor", quiet = TRUE, progress = FALSE)
```

Species checklists

There is a wealth of checklists from all the source locations. To get all available checklists from the engine, run:

```
all_lists <- ee_checklists()
```

```
#>  Returning 52 checklists
```

```
head(all_lists[, c("footprint", "subject")])
```

```
#>                                                        footprint
#>  1   https://ecoengine.berkeley.edu/api/footprints/angelo-reserve/
#>  2   https://ecoengine.berkeley.edu/api/footprints/angelo-reserve/
#>  3   https://ecoengine.berkeley.edu/api/footprints/angelo-reserve/
#>  4 https://ecoengine.berkeley.edu/api/footprints/hastings-reserve/
#>  5   https://ecoengine.berkeley.edu/api/footprints/angelo-reserve/
#>  6 https://ecoengine.berkeley.edu/api/footprints/hastings-reserve/
#>       subject
#>  1    Mammals
#>  2     Mosses
#>  3    Beetles
#>  4    Spiders
#>  5 Amphibians
#>  6       Ants
```

Currently there are 52 lists available. We can drill deeper into any list to get all the available data. We can also narrow our checklist search to groups of interest (see `unique(all_lists$subject)`). For example, to get the list of Spiders:

```
spiders <- ee_checklists(subject = "Spiders")
```

```
#>  Returning 1 checklists
```

```
spiders
```

```
#>                  record
#>  4 bigcb:speciesslist:15
#>                                                        footprint
#>  4 https://ecoengine.berkeley.edu/api/footprints/hastings-reserve/
#>                                                                                 url
#>  4 https://ecoengine.berkeley.edu/api/checklists/bigcb%3Aspecieslist%3A15/
#>                                           source subject
#>  4 https://ecoengine.berkeley.edu/api/sources/18/ Spiders
```

Now we can drill deep into each list. For this tutorial I'll just retrieve data from the the two lists returned above.

```
library(plyr)
spider_details <- ldply(spiders$url, checklist_details)
names(spider_details)
```

```
#>   [1] "url"                              "observation_type"
#>   [3] "scientific_name"                  "collection_code"
#>   [5] "institution_code"                 "country"
#>   [7] "state_province"                   "county"
#>   [9] "locality"                         "begin_date"
#>  [11] "end_date"                         "kingdom"
#>  [13] "phylum"                           "clss"
#>  [15] "order"                            "family"
#>  [17] "genus"                            "specific_epithet"
```

```
#>  [19] "infraspecific_epithet"            "source"
#>  [21] "remote_resource"                  "earliest_period_or_lowest_system"
#>  [23] "latest_period_or_highest_system"
```

```
unique(spider_details$scientific_name)
```

```
#>   [1] "Holocnemus pluchei"        "Oecobius navus"
#>   [3] "Uloborus diversus"         "Neriene litigiosa"
#>   [5] "Theridion "                "Tidarren "
#>   [7] "Dictyna "                  "Mallos "
#>   [9] "Yorima "                   "Hahnia sanjuanensis"
#>  [11] "Cybaeus "                  "Zanomys "
#>  [13] "Anachemmis "               "Titiotus "
#>  [15] "Oxyopes scalaris"          "Zora hespera"
#>  [17] "Drassinella "              "Phrurotimpus mateonus"
#>  [19] "Scotinella "               "Castianeira luctifera"
#>  [21] "Meriola californica"       "Drassyllus insularis"
#>  [23] "Herpyllus propinquus"      "Micaria utahna"
#>  [25] "Trachyzelotes lyonneti"    "Ebo evansae"
#>  [27] "Habronattus oregonensis"   "Metaphidippus "
#>  [29] "Platycryptus californicus" "Calymmaria "
#>  [31] "Frontinella communis"      "Undetermined "
#>  [33] "Latrodectus hesperus"
```

Our resulting dataset now contains 33 unique spider species.

Searching the engine

The search is elastic by default. One can search for any field in `ee_observations()` across all available resources. For example,

```
# The search function runs an automatic elastic search across all resources
# available through the engine.
lynx_results <- ee_search(query = "genus:Lynx")
lynx_results[, -3]
# This gives you a breakdown of what's available allowing you dig deeper.
```

	field	results
state_province.2	California	282
state_province.21	Nevada	70
state_province.3	Alaska	51
state_province.4	British Columbia	34
state_province.5	Arizona	24
state_province.6	Montana	14
state_province.7	Baja California Sur	13
state_province.8	Baja California	12
state_province.9	Zacatecas	10
state_province.10	New Mexico	8
kingdom	animalia	562
genus	lynx	578
resource	Observations	578

	field	results
family	felidae	563
scientific_name.2	Lynx rufus californicus	232
scientific_name.21	Lynx rufus baileyi	93
scientific_name.3	Lynx canadensis canadensis	93
scientific_name.4	Lynx rufus pallescens	76
scientific_name.5	Lynx rufus	16
scientific_name.6	Lynx rufus peninsularis	15
scientific_name.7	Lynx canadensis	13
scientific_name.8	Lynx rufus rufus	11
scientific_name.9	Lynx rufus fasciatus	11
scientific_name.10	Lynx rufus escuinapae	10
country.2	United States	495
country.21	Mexico	45
country.3	Canada	34
country.4	None	4
clss	mammalia	562
order	carnivora	561
phylum	chordata	559
georeferenced.2	true	525
georeferenced.21	false	53
observation_type.2	specimen	557
observation_type.21	photo	16
observation_type.3	fossil	4
observation_type.4	checklist	1
decade.2	1931-1940	132
decade.21	1921-1930	91
decade.3	1911-1920	74
decade.4	1901-1910	66
decade.5	1971-1980	62
decade.6	1941-1950	46
decade.7	1961-1970	35
decade.8	1951-1960	34
decade.9	2001-2010	9
decade.10	1991-2000	4

Similarly it's possible to search through the observations in a detailed manner as well.

```
# all_lynx_data <- ee_search_obs(query = 'Lynx', page = 'all', progress =
# FALSE) all_lynx_data
```

Miscellaneous functions

Footprints

`ee_footprints()` provides a list of all the footprints.

```
footprints <- ee_footprints()
footprints[, -3]  # To keep the table from spilling over
```

Table 4: Table continues below

name
Angelo Reserve
Sagehen Reserve
Hastings Reserve
Blue Oak Ranch Reserve

url
https://ecoengine.berkeley.edu/api/footprints/angelo-reserve/
https://ecoengine.berkeley.edu/api/footprints/sagehen-reserve/
https://ecoengine.berkeley.edu/api/footprints/hastings-reserve/
https://ecoengine.berkeley.edu/api/footprints/blue-oak-ranch-reserve/

Data sources

`ee_sources()` provides a list of data sources for the specimens contained in the museum.

```
source_list <- ee_sources()
unique(source_list$name)
```

name
UCANR Drought images
LACM Vertebrate Collection
CAS Botany
CAS Entomology
MVZ Herp Collection
VTM plot coordinates
CAS Herpetology
VTM plot data
CAS Ornithology
Consortium of California Herbaria

```
devtools::session_info()
```

```
#>  Session info -------------------------------------------------------------

#>   setting  value
#>   version  R version 3.1.2 (2014-10-31)
#>   system   x86_64, darwin13.4.0
#>   ui       X11
#>   language (EN)
#>   collate  en_US.UTF-8
#>   tz       America/Los_Angeles

#>  Packages -----------------------------------------------------------------

#>   package    * version  date       source
```

```
#>   assertthat * 0.1      2013-12-06 CRAN (R 3.1.0)
#>   bitops     * 1.0-6    2013-08-17 CRAN (R 3.1.0)
#>   brew       * 1.0-6    2011-04-13 CRAN (R 3.1.0)
#>   codetools  * 0.2-11   2015-03-10 CRAN (R 3.1.3)
#>   colorspace * 1.2-6    2015-03-11 CRAN (R 3.1.3)
#>   coyote     * 0.1      2014-05-05 Github (karthik/coyote@4ed329d)
#>   DBI        * 0.3.1    2014-09-24 CRAN (R 3.1.1)
#>   devtools     1.7.0    2015-01-17 CRAN (R 3.1.2)
#>   digest     * 0.6.8    2014-12-31 CRAN (R 3.1.2)
#>   dplyr        0.4.1    2015-01-14 CRAN (R 3.1.2)
#>   ecoengine    1.9      2015-03-24 local
#>   evaluate   * 0.5.5    2014-04-29 CRAN (R 3.1.0)
#>   formatR    * 1.0      2014-08-25 CRAN (R 3.1.2)
#>   ggplot2      1.0.1    2015-03-17 CRAN (R 3.1.3)
#>   gtable     * 0.1.2    2012-12-05 CRAN (R 3.1.0)
#>   htmltools  * 0.2.6    2014-08-14 Github (rstudio/htmltools@fa3e0ab)
#>   httr         0.6.1    2015-01-01 CRAN (R 3.1.2)
#>   jsonlite   * 0.9.14   2014-12-01 CRAN (R 3.1.2)
#>   knitr        1.9      2015-01-20 CRAN (R 3.1.2)
#>   leafletR   * 0.3-1    2014-10-23 CRAN (R 3.1.2)
#>   lubridate  * 1.3.3    2013-12-31 CRAN (R 3.1.0)
#>   magrittr   * 1.5      2014-11-22 CRAN (R 3.1.2)
#>   MASS       * 7.3-40   2015-03-21 CRAN (R 3.1.3)
#>   memoise    * 0.2.1    2014-04-22 CRAN (R 3.1.0)
#>   munsell    * 0.4.2    2013-07-11 CRAN (R 3.1.0)
#>   pander       0.5.1    2014-10-29 CRAN (R 3.1.2)
#>   plyr         1.8.1    2014-02-26 CRAN (R 3.1.0)
#>   proto      * 0.3-10   2012-12-22 CRAN (R 3.1.0)
#>   Rcpp       * 0.11.5   2015-03-06 CRAN (R 3.1.3)
#>   RCurl      * 1.95-4.5 2014-12-06 CRAN (R 3.1.2)
#>   reshape2   * 1.4.1    2014-12-06 CRAN (R 3.1.2)
#>   rmarkdown    0.5.1    2015-01-26 CRAN (R 3.1.2)
#>   rstudioapi * 0.2      2014-12-31 CRAN (R 3.1.2)
#>   scales     * 0.2.4    2014-04-22 CRAN (R 3.1.0)
#>   stringr    * 0.6.2    2012-12-06 CRAN (R 3.1.0)
#>   whisker    * 0.3-2    2013-04-28 CRAN (R 3.1.0)
```

Please send any comments, questions, or ideas for new functionality or improvements to <karthik.ram@berkeley.edu>. The code lives on GitHub under the rOpenSci account. Pull requests and bug reports are most welcome.

Karthik Ram
Mar, 2015
Fremont, California