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()
```

```
## Loading required package: rjson
```

type	endpoint
meta-data	http://ecoengine.berkeley.edu/api/sources/
meta-data	http://ecoengine.berkeley.edu/api/footprints/
data	http://ecoengine.berkeley.edu/api/checklists/
data	http://ecoengine.berkeley.edu/api/sensors/
data	http://ecoengine.berkeley.edu/api/vtmveg/
data	http://ecoengine.berkeley.edu/api/observations/
data	http://ecoengine.berkeley.edu/api/photos/
actions	http://ecoengine.berkeley.edu/api/search/

The ecoengine class

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

- A total result count (not necessarily the results in this particular object but the total number available for a particlar query)
- The call (So a reader can replicate the results or rerun the query using other tools.)
- The type (`photos`, `observation`, `checklist`, or `sensor`)
- 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 `25` 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 (`1,000` or fewer records). 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")
```

```
##   |                                                                             |
```

```
## Search returned 754 photos (downloading page 1 of 31)
```

```
ee_pages(request)
```

```
## [1] 31
```

```
# 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. 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_exact = "Pinus", page = 1)
```

```
## Retrieving 1 pages (total: 25 records)
```

```
##   |                                                                             |
```

```
pinus_observations
```

```
##  [Type]: observations
##  [Number of results]: 25
##  [Call]: http://ecoengine.berkeley.edu/api/observations/?country=United+States&page=2&scientific_name
##  [Output dataset]: 25 rows
```

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)
```

```
##   |                                                                             |
## 795 observations found
```

```
lynx_data
```

```
##  [Type]: observations
##  [Number of results]: 795
##  [Call]: http://ecoengine.berkeley.edu/api/observations/?country=United+States&genus=Lynx&page=2&pag
##  [Output dataset]: 25 rows
```

```
# Notice that we only for the first 25 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")
```

```
## Retrieving 32 pages (total: 795 records)
```

```
##   |                                                                          |
```

```
lynx_data
```

```
##  [Type]: observations
##  [Number of results]: 795
##  [Call]: http://ecoengine.berkeley.edu/api/observations/?country=United+States&genus=Lynx&page=2&pag
##  [Output dataset]: 795 rows
```

Photos

```
photos <- ee_photos()
```

```
##   |                                                                          |
```

```
## Search returned 43708 photos (downloading page 1 of 1749)
```

```
photos
```

```
##  [Type]: photos
##  [Number of results]: 43708
##  [Call]: http://ecoengine.berkeley.edu/api/photos/?page=2&page_size=25&format=json
##  [Output dataset]: 25 rows
```

The database currently holds 43708 photos. Photos can be searched by state province, county, genus, scientific name, authors along with date bounds. For additional options see `?ee_photos_get`.

Searching photos by author

```
charles_results <- ee_photos(author = "Charles Webber")
```

```
##   |                                                                          |
```

```
## Search returned 4012 photos (downloading page 1 of 161)
```

```
charles_results
```

```
##  [Type]: photos
##  [Number of results]: 4012
##  [Call]: http://ecoengine.berkeley.edu/api/photos/?format=json&page=2&page_size=25&authors=Charles+W
##  [Output dataset]: 25 rows
```

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

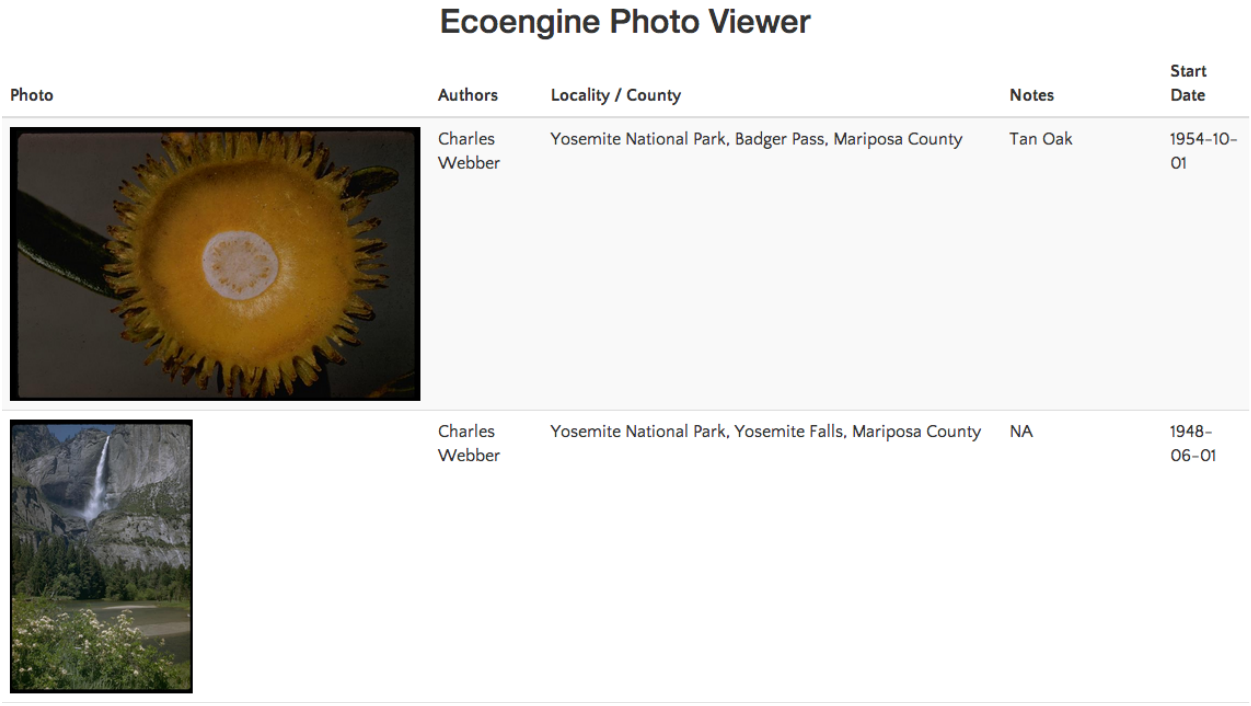
```
##          authors                               locality          county
## 1 Charles Webber    Yosemite National Park, Badger Pass Mariposa County
## 2 Charles Webber Yosemite National Park, Yosemite Falls Mariposa County
##   photog_notes
## 1      Tan Oak
## 2         <NA>
##                                                                                url
## 1 http://ecoengine.berkeley.edu/api/photos/CalPhotos%3A8076%2B3101%2B2933%2B0025/
## 2 http://ecoengine.berkeley.edu/api/photos/CalPhotos%3A8076%2B3101%2B0667%2B0107/
##   begin_date   end_date                        record
## 1 1954-10-01 1954-10-01 CalPhotos:8076+3101+2933+0025
## 2 1948-06-01 1948-06-01 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 observations.scientific_name
## 1      CalAcademy      Lithocarpus densiflorus
## 2      CalAcademy     Rhododendron occidentale
##                                                                             observations.url
## 1 http://ecoengine.berkeley.edu/api/observations/CalPhotos%3A8076%2B3101%2B2933%2B0025%3A1/
## 2 http://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 geojson.type
## 1 http://ecoengine.berkeley.edu/api/sources/9/         <NA>
## 2 http://ecoengine.berkeley.edu/api/sources/9/         <NA>
##   geojson.coordinates1 geojson.coordinates2
## 1                 <NA>                 <NA>
## 2                 <NA>                 <NA>
```

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.

Other photo search examples



Ecoengine Photo Viewer

Photo	Authors	Locality / County	Notes	Start Date
	Charles Webber	Yosemite National Park, Badger Pass, Mariposa County	Tan Oak	1954-10-01
	Charles Webber	Yosemite National Park, Yosemite Falls, Mariposa County	NA	1948-06-01

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

Species checklists

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

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

```
## Returning 57 checklists
```

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

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

Currently there are 57 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 2 checklists
```

```
spiders
```

```
##                   record
## 4  bigcb:specieslist:15
## 10 bigcb:specieslist:20
##                                                         footprint
## 4  http://ecoengine.berkeley.edu/api/footprints/hastings-reserve/
## 10   http://ecoengine.berkeley.edu/api/footprints/angelo-reserve/
##                                                                 url
## 4  http://ecoengine.berkeley.edu/api/checklists/bigcb%3Aspecieslist%3A15/
## 10 http://ecoengine.berkeley.edu/api/checklists/bigcb%3Aspecieslist%3A20/
##                                           source subject
## 4  http://ecoengine.berkeley.edu/api/sources/18/ Spiders
## 10 http://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"                         "coordinate_uncertainty_in_meters"
## [11] "begin_date"                       "end_date"
## [13] "kingdom"                          "phylum"
## [15] "clss"                             "order"
## [17] "family"                           "genus"
## [19] "specific_epithet"                 "infraspecific_epithet"
## [21] "source"                           "remote_resource"
## [23] "earliest_period_or_lowest_system" "latest_period_or_highest_system"
```

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

```
##  [1] "holocnemus pluchei"        "oecobius navus"
##  [3] "uloborus diversus"         "neriene litigiosa"
##  [5] "theridion sp. A"           "tidarren sp."
##  [7] "dictyna sp. A"             "dictyna sp. B"
##  [9] "mallos sp."                "yorima sp."
## [11] "hahnia sanjuanensis"       "cybaeus sp."
## [13] "zanomys sp."               "anachemmis sp."
## [15] "titiotus sp."              "oxyopes scalaris"
## [17] "zora hespera"              "drassinella sp."
## [19] "phrurotimpus mateonus"     "scotinella sp."
## [21] "castianeira luctifera"     "meriola californica"
## [23] "drassyllus insularis"      "herpyllus propinquus"
## [25] "micaria utahna"            "trachyzelotes lyonneti"
## [27] "ebo evansae"               "habronattus oregonensis"
## [29] "metaphidippus sp."         "platycryptus californicus"
## [31] "calymmaria sp."            "frontinella communis"
## [33] "undetermined sp."          "latrodectus hesperus"
## [35] "theridion sp. B"           "agelenopsis oregonensis"
## [37] "pardosa spp."              "schizocosa mccooki"
## [39] "hololena sp."              "callobius sp."
## [41] "pimus sp."                 "aliatypus sp."
## [43] "antrodiaetus sp."          "antrodiaetus riversi"
## [45] "anyphaena californica"     "aculepeira packardi"
## [47] "araneus bispinosus"        "araniella displicata"
## [49] "cyclosa conica"            "cyclosa turbinata"
## [51] "brommella sp."             "cicurina sp."
## [53] "dictyna sp."               "emblyna oregona"
## [55] "orodrassus sp."            "sergiolus sp."
## [57] "erigone sp."               "pityohyphantes sp."
## [59] "tachygyna sp."             "alopecosa kochi"
## [61] "oxyopes salticus"          "philodromus sp."
## [63] "tibellus oblongus"         "pimoa sp."
## [65] "undetermined spp."         "metaphidippus manni"
## [67] "thiodina sp."              "diaea livens"
## [69] "metellina sp."             "cobanus cambridgei"
## [71] "tetragnatha sp."           "tetragnatha versicolor"
## [73] "dipoena sp."               "theridion spp."
## [75] "misumena vatia"            "misumenops sp."
## [77] "tmarus angulatus"          "xysticus sp."
## [79] "hyptiotes gertschi"        "mexigonus morosus"
```

Our resulting dataset now contains 80 unique spider species.

Sensors

Sensor data come from the Keck HydroWatch Center.

You'll need a sensor's id to query the data for that particular metric and location. The `ee_list_sensors()` function will give you a condensed list with the location, metric, binning method and most importantly the `sensor_id`. You'll need this id for the data retrieval.

```
head(ee_list_sensors())
```

station_name	units
Angelo HQ WS	Kilojoules per square meter
Angelo Meadow WS	Watts per square meter
Angelo HQ SF Eel Gage	Percent
Angelo HQ WS	Degree
Cahto Peak WS	Meters per second
Angelo Meadow WS	Meters per second

Table 2: List of stations (continued below)

variable	method_name	record
Solar radiation total kj/m^2	Conversion to 30-minute timesteps	1625
Solar radiation total w/m^2	Conversion to 30-minute timesteps	1632
Rel humidity perc	Conversion to 30-minute timesteps	1641
Wind direction degrees	Conversion to 30-minute timesteps	1644
Wind speed avg ms	Conversion to 30-minute timesteps	1651
Wind speed max ms	Conversion to 30-minute timesteps	1654

Let's download solar radiation for the Angelo reserve HQ (sensor_id = 1625).

```
# First we can grab the list of sensor ids
sensor_ids <- ee_list_sensors()$record
# In this case we just need data for sensor with id 1625
angelo_hq <- sensor_ids[1]
results <- ee_sensor_data_get(angelo_hq, page = 2)
```

```
## Search returned 56779 observations (downloading page 2 of 2272)
```

```
results
```

```
##  [Type]: sensor
##  [Number of results]: 56779
##  [Call]: http://ecoengine.berkeley.edu/api/sensors/1625/data/?page=2&format=json
##  [Output dataset]: 25 rows
```

Notice that the query returned 56779 observations but has only retrieved the 25-50 since we requested records for page 2 (and each page by default retrieves 25 records). You can request page = "all" but remember that this will make 2271 requests. This could take a while and might make sense to parallelize the request or split the calls so as to avoid hammering the server all at once.

Now we can examine the data itself.

```
head(results$data)
```

```
##             local_date value data_quality_qualifierid
## 2  2010-01-06 02:00:00 -9999                       19
## 26 2010-01-06 02:30:00 -9999                       19
## 3  2010-01-06 03:00:00 -9999                       19
## 4  2010-01-06 03:30:00 -9999                       19
## 5  2010-01-06 04:00:00 -9999                       19
## 6  2010-01-06 04:30:00 -9999                       19
##               data_quality_qualifier_description data_quality_valid
## 2  Passed sanity check; see incident report IR_8              FALSE
## 26 Passed sanity check; see incident report IR_8              FALSE
## 3  Passed sanity check; see incident report IR_8              FALSE
## 4  Passed sanity check; see incident report IR_8              FALSE
## 5  Passed sanity check; see incident report IR_8              FALSE
## 6  Passed sanity check; see incident report IR_8              FALSE
```

We can also aggregate sensor data for any of the above mentioned sensors. We do this using the `ee_sensor_agg()` function. The function requires a sensor id and how the data should be binned. You can specify hours, minutes, seconds, days, weeks, month, and years. If for example you need the data binned every `15` days, simply add `days = 15` to the call. Once every `10` days and `2` hours would be `ee_sensor_agg(sensor_id = 1625, days = 10, hours = 2)`

```
stations <- ee_list_sensors()
# This gives you a list to choose from
sensor_df <- ee_sensor_agg(sensor_id = stations[1, c("record")], weeks = 2)
```

```
##   |                                                                         |
```

```
## Retrieving page 1 (85 observations total)
```

```
sensor_df
```

```
##  [Type]: sensor
##  [Number of results]: 85
##  [Call]: http://ecoengine.berkeley.edu/api/sensors/1625/aggregate/?interval=2W&page=2&page_size=25&f
##  [Output dataset]: 25 rows
```

```
head(sensor_df$data)
```

```
##    begin_date  mean min   max   sum count
## 2  2010-01-17 18.94   0 150.8  7613   402
## 26 2010-01-31 17.03   0 237.7 11444   672
## 3  2010-02-14 29.54   0 336.3 19852   672
## 4  2010-02-28 42.08   0 402.5 28276   672
## 5  2010-03-14 59.12   0 466.6 39730   672
## 6  2010-03-28 93.55   0 490.6 62678   670
```

As with other functions, the results are paginated. Since we only need `85` records in this case:

```
sensor_df <- ee_sensor_agg(sensor_id = 1625, weeks = 2, page = "all")
```

```
## Retrieving 4 pages (total: 85 records)
```

```
##   |                                                                              |
```

```
sensor_df
```

```
##  [Type]: sensor
##  [Number of results]: 85
##  [Call]: http://ecoengine.berkeley.edu/api/sensors/1625/aggregate/?interval=2W&page=2&page_size=25&f
##  [Output dataset]: 85 rows
```

To obtain data

Searching the engine

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

```
# A faceted search through all the resources
ee_search()
# A detailed search of the observations
ee_search_obs()
```

Miscellaneous functions

Footprints

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

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

id	name
12	Angelo Reserve
13	Sagehen Reserve
14	Hastings Reserve
15	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
LACM Vertebrate Collection
MVZ Birds
MVZ Herp Collection
MVZ Mammals
Wieslander Vegetation Map
CAS Herpetology
Consortium of California Herbaria
UCMP Vertebrate Collection
Sensor Data Qualifiers
Essig Museum of Entymology

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
January, 2014
Berkeley, CA