R project for statistical computing

Task Proposal

Integrating biodiversity data curation functionality

Thiloshon Nagarajah
4-1-2017

Project info

Project title: Integrating biodiversity data curation functionality

Project short title (30 characters): Data-cleaning Integration

URL of project idea page:
https://github.com/rstats-gsoc/gsoc2017/wiki/Integrating-biodiversity-data-curation-functionality

Bio of Student

I am Thiloshon Nagarajah a Software Engineering undergraduate at Informatics Institute of Technology, Colombo, Sri Lanka. My interests and aspirations lie in trying to yield the naturally occurring data and derive insights from it, that is to create order from chaos.

My experience with R begins with multiple online courses and I'm willing to learn and apply what I learned. I was able to learn R systematically from programming to reproducible research and documentation, the fundamental skills needed for this task. I have completed first five courses in Coursera Data Science Specialization by Johns Hopkins University namely,

1) The Data Scientist's Toolbox
2) R Programming
3) Getting and Cleaning Data
4) Exploratory Data Analysis
5) Reproducible Research

and currently doing the sixth course, Statistical Inference. Further my interest in Algorithms made me follow the Data Structures and Algorithms Coursera course by University of California, San Diego and finished first 3 courses in the specialization namely,

1) Algorithmic Toolbox
2) Data Structures
3) Algorithms on Graphs

I have experience working in open source projects, for Fedora Project in 2012 and Sahana Foundation in 2013 at the Google code-in competition. I was not able to win but finished more than three tasks both year which secured me certificates and T-shirts. I am a mentor in the university Peer to Peer Programming Club which helps students understand and improve programming. Throughout the time I was a mentor, I was able to observe and identify the common areas where students often lose the essence of programming and made a blog series to help students in these areas.

I have led a test driven university group project for three months following Agile methodologies which resulted in an IoT solution to regulate tooth brushing. From it I learned to build an industry standard product feature by feature, testing each phase as it's built and to keep track of the progress by utilizing task management tools like Trello, which are I think utmost necessity to this Task.

Other than R, my skills mainly lie in Java and I have been programming in java for the last two years. I am familiar with web development languages like HTML5, CSS3, JavaScript, SQL, PHP and hardware technologies like Arduino and IoT and also an android programmer. All my projects can be found in GitHub repositories.

Outside programming, I am a short film maker and graphics designer. I work on small projects whenever I get free time and my portfolio can be found here.

Contact Information

Student name:
Thiloshon Nagarajah

Student postal address:
A4, 37,
Sri Sumangala Mawatha,
Ratmalana,
Western Province,
Sri Lanka

Telephone(s):
+94774209947
+94112721583 (Home)
+94777793984 (Mother, In case of emergency)

Email(s):
thiloshon@gmail.com (Primary)
thiloshon.2015298@iit.ac.lk (Secondary)

Other communications channels:
Skype: thiloshon.nagarajah
Google+: +ThiloshonNagarajah

Student affiliation

Institution: Informatics Institute of Technology
Program: Software Engineering
Stage of completion: 2nd year of 4 years.

Contact to verify:
Aloka Fernando
Assistant Lecturer / Placement Coordinator
Informatics Institute of Technology,
57, Ramakrishna Road, Colombo 06,
Sri Lanka
E: aloka.f@iit.ac.lk
T: +94-112-360-212/107 (Ext)
M: +94-76-8209-748
www.iit.ac.lk

Schedule Conflicts

My university offers a sandwich degree which means I have to do a 11-month compulsory internship starting from first week of August. It will be a 40 hour/week internship. To compensate for that 3 weeks, I'm starting the coding three weeks earlier than the Official Coding Time. It will not interfere the Community Bonding since usually apart from communicating with the mentors and organization, few other preparations are expected to be done during this time like setting up environments and familiarizing with code base. But for this task no such things are needed since the task involves creating a new package from scratch. And I have already familiarized with the packages needed to build in the Test task period and proposal period. So as soon as Community Bonding starts I will be able to start the coding too.

Mentors

Mentor names: Tomer Gueta, Vijay Barve and Yohay Carmel

Mentor emails: tomer.gu@gmail.com, vijay.barve@gmail.com, yohay@technion.ac.il

Have you been in touch with the mentors? When and how?

Yes. On Mar 12 2017 by their mail ids. I introduced myself and notified them of finishing the test tasks and pointed out a problem with the API, task was based on. They acknowledged and asked to work with the proposal and suggested few resources.

And then on Mar 22 2017. I sent them the first draft of the proposal to get their feedback and subsequently requested feedback from Brian Peterson and Yohay Carmel.

On Mar 27th I contacted Tomer Gueta again and requested a clarification in the task.

On Mar 28th Tomer Gueta replied with a well-constructed feedback, clarified my doubt and gave revision on an image I had used in the proposal.

Coding Plan and Methods

Introduction to Problem:

> Biodiversity research is evolving rapidly, progressively changing into a more collaborative and data-intensive science. The integration and analysis of large amounts of data is inevitable, as researchers increasingly address questions at broader spatial, taxonomic and temporal scales than before. Until recently, biodiversity data was scattered in different formats in natural history collections, survey reports, and in literature. In the last fifteen years, lot of efforts are being made to establish standards in the biodiversity database structure (Darwin Core standard, DwC). However, none of the hundreds DwC fields are mandatory or impose strong rules on the content associated with any record; thus, data vary in precision and in quality. To-date, there are several centralized portals that aggregate large volumes of biodiversity records from around the world and publish them in a DwC format. These aggregators are prone to numerous data errors, due to incomplete or erroneous information at the publisher level, errors during the publishing processes (e.g. formatting of date information) as well as errors during the central harvesting and indexing procedures.
>
> (Task Page)

The importance of data in the biodiversity research has been repeatedly stressed in the recent times and various organizations have come together and followed each other to provide data for advancing biodiversity research. But, that is exactly where the main hiccup of biodiversity research lies. Since there are many such organizations, the data aggregated by these organizations vary in precision and in quality. Further, though in recent times more researchers have started to use R for their data analyses, since they need to retrieve, manage and assess data with complex (DwC) structure and high volume, only researchers with extremely sound R programming background have been able to attempt this.

The various R packages created so far have been focused on addressing some elements of the entire process. For example

- finch, rgbif and spocc for biodiversity data retrieval.
- Taxize, traits, rredlist for taxonomical enrichment and cleaning
- Biogeo, rgeospatialquality, scrubr, assertr for data cleaning

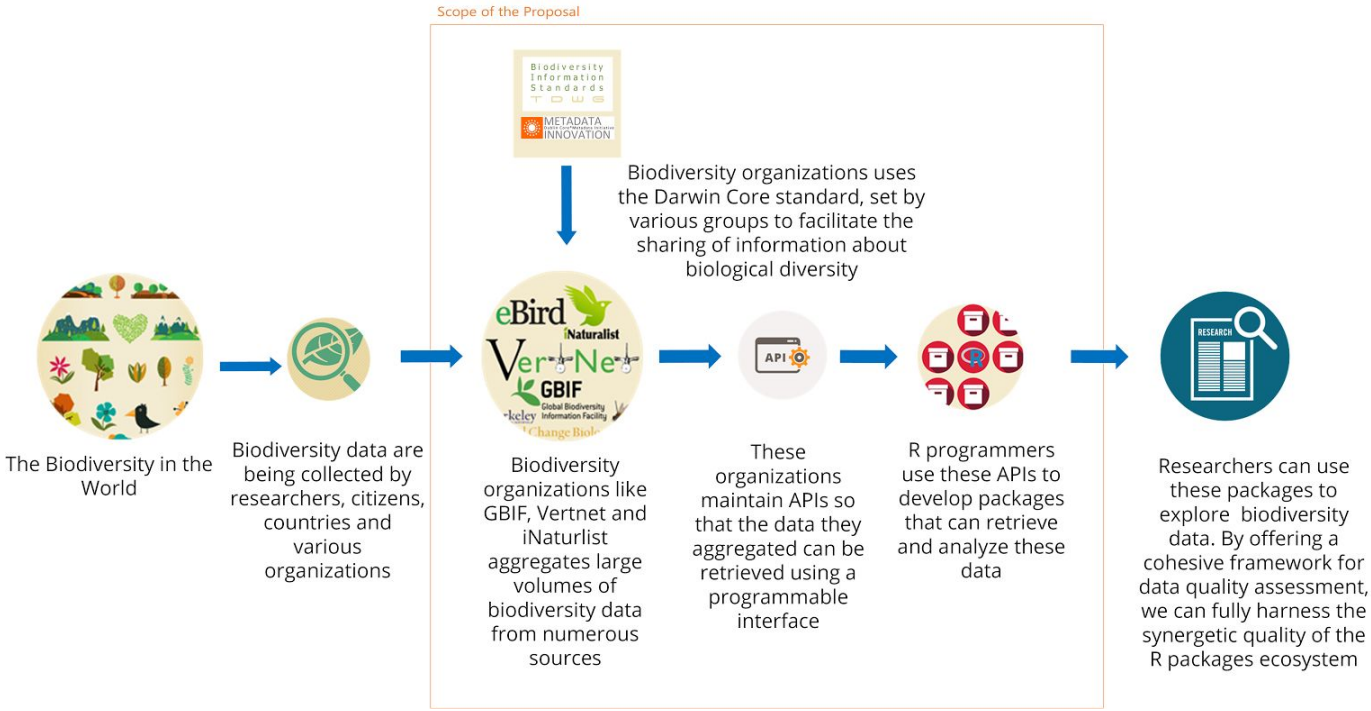
Thus when a researcher decides to use these tools, he needs to

1) Know these packages exist
2) Understand what each package does
3) Compare and contrast packages offering same functionalities and decide the best for his needs
4) Maintain compatibility between the packages and datasets transferred between packages.

What we propose:

We propose to create a R Package that will function as the main data retrieval, cleaning, management and backup tool to the researchers. The functionalities of the package are culmination of various existing packages and enhancements to existing functions rather than creating one from the scratch. This way we can cultivate the existing resources, collaborative knowledge and skills and also address the problem we identified efficiently. The package will also address the issue of researchers not having sound R programming skills.

Before we analyze solutions, it's important to understand the stakeholders and scope of the project.



The Darwin Core is a body of standards which includes a glossary of terms intended to facilitate the sharing of information about biological diversity by providing reference definitions, examples, and commentaries.

DWC Archive is a specific structure of data with meta, data, etc. Simple DwC is also another structure but with less complexity.

GBIF and similar (around 15 similar) organizations aggregate species occurrence data around the world and archive the data according to DwC standard.

Proposed Package:
The package will cover major processes in the research pipeline.

1) Getting Biodiversity data to the workspace
   The biodiversity data can be read from existing DwC archive files in various formats (DwCA, XML, CSV) or it can be downloaded from online sources (GBIF, Vertnet). In case the user doesn't know what data to retrieve the name suggestion functions will also be included. Converting common name to scientific name, scientific name to common name, getting taxon keys to names will also be covered.

2) Flagging the data
   The biodiversity data is aggregated by various different organizations. Thus these data vary in precision and in quality. It is highly necessary to first check the quality of the data and strip the records which lacks the quality expected before using it. Various packages built thus far have been able to check data for various discrepancies such as scrubr and rgeospatialquality. Integrating functionalities given by such packages to produce a better quality control will benefit the community greatly. The data will be checked for discrepancies

In spatial - coordinate, country and country code
In temporal - all time fields
In taxonomic - epithet, scientific name and common name
And duplicate records of data will be flagged.

3) Cleaning data
The process is done step by step to help user configure and control the cleaning process.
The data will be flagged first for various discrepancies. It can be any combination of spatial, temporal, taxonomic and duplicate flags as user specifies. Then the user can view the data he will be losing and decide if he wants to tweak the flags. In times when the data is high in volume, this procedural cleaning would help user for number of reasons.
    1) When user wants multiple flags, he can apply each quality check one by one and decide if he wants to remove flagged data once he applies one check. If he is to apply all flags at once the records to be removed will be high in number and he has to put much effort to go through all of it to decide if he wants that quality check
    2) When user views the flagged data, not all fields will be shown. Only the fields the quality check was done on and the flagged result will be shown. This saves user from having to deal with all the complex fields in the original data and having to go to and forth the data to check the flags.

    If he is satisfied with the records he will lose then, the data can be cleaned.

    Ex: Step 01
    ```
    biodiversityData + coordinateIncompleteFlag() + allFlags(taxonomic)
    ```
    Step 02
    ```
    viewFlaggedData()
    ```
    Step 03
    ```
    cleanAll()
    ```

4) Maintaining backups of data
The original data user retrieved and any subsequent resultant data of his process can be backed up with versioning to maintain reproducibility.

Underneath, the package will implement these,

1) Standardization of data
The data retrieved will be standardized according to DwC formats. To maintain consistency and feasibility I have decided to use GBIF fields as the standard.
The reasons are,
    a. The GBIF uses DwC as the standardization and the fields from GBIF backbone complies with DwC terms.
    b. GBIF is a well-established organization and using the fields from their backbone will assure consistency and acceptance by researchers.
    c. GBIF is a superset of all major biodiversity data available. Any data gathered from GBIF can be expected to also be in other sources too.

2) Unique fields grouping system for DwC fields
The fields in DwC will be grouped as GBIF recommends. (GBIF (2010). Darwin Core Quick Reference Guide, version 1.3, released on 10 Mar 2012)

    1) Record-Level Terms (Darwin Core)
    Terms under the Record-level Terms section apply to the whole record regardless of the record type.

2) Class Occurrence
   The category of information pertaining to evidence of an occurrence in nature, in a collection, or in a dataset (specimen, observation, etc.)
3) Class Event
   The category of information pertaining to an event (an action that occurs at a place and during a period of time)
4) Class Location
   A spatial region or named place. For Darwin Core, a set of terms describing a place, whether named or not.
5) Class Geological Context
   The category of information pertaining to a location within a geological context, such as stratigraphy.
6) Class Identification
   The category of information pertaining to taxonomic determinations (the assignment of a scientific name)
7) Class Taxon
   The category of information pertaining to taxonomic names, taxon name usages, or taxon concepts.
8) Auxiliary terms – Resource Relationship
9) Auxiliary terms – Measurement or Fact

   Refer Appendix for full list of grouping fields.

Functions in the pipeline of entire process

Functions are tagged as,

1) As is – The functions which use source functions as it is or with minor changes. Even though the functions are used as it is, I have decided to rename them. Reason being, the names in these packages are not intuitively understandable which creates an unpleasant experience for inexperienced users.
   For ex:
   - occ_data
   - occ_search
   - occ_get
   - occ_download are all part of `rgbif` package and used to retrieve data in

   different ways. So when an inexperienced user tries to use these packages, the names are misleading and baffling. But converting the names to
   - biodiv_get
   - biodiv_get_fast
   - biodiv_get_occurence
   - biodiv_get_download

   will help users easily identify the functions and intuitively understand their functionality. Besides this way of naming has a proper structure in the names of functions since all sister functions will have same suffix to be able to easily use them.
   Ex:
   All sister functions of `biodiv_read` deals with reading DwC data from local files.
   `biodiv_read_XML`
   `biodiv_read_DwCA`
   `biodiv_read_CSV`

All sister functions of `biodiv_get` deals with getting data from APIs.
`biodiv_get`
`biodiv_get_fast`
`biodiv_get_occurence`
`biodiv_get_download`

2) Enhancement - The functions which modify or enhance or merge source functions
3) New – The functions which are created for the task or use functions that are not related to biodiversity

The first part of process is retrieving data. There are two ways to do this.

1) **Reading archive data file into R.**

`biodiv_read_XML`	*Reading DarwinRecordSet and SimpleDarwinRecordSet files.*
Tag: As is	Uses: `finch::simple_read`

`biodiv_read_DwCA`	*Read Darwin Core Archive*
Tag: As is	Uses: `finch::dwca_read`

`biodiv_read_CSV`	*Read Darwin Core Archive*
Tag: New	It reads the DwC files downloaded as csv file from GBIF

2) **Downloading using an API.**

`biodiv_get`	*Search for occurrences data across many data sources*
Tag: Enhancement	Uses: `gbif::occ_search, spocc::occ`

This is one of the most important functions in the package. `gbif::occ_search` returns only the occurrence data in GBIF backbone whereas the `spocc::occ` returns data from various sources such as rgbif, rbison, rinat, rebird, AntWeb, ecoengine, rvertnet and ridigbio. The gbif data returned by the occ function has 5 less columns than the original occ_search function namely, identifiers, facts, relations, species and speciesKey. So the new function will be implemented as follows,

1) If the user queries data from GBIF, then `gbif::occ_search` will be used.
2) For other sources requested the `spocc::occ` will be used.

Further the `spocc::occ` can be enhanced to address few short comes in the function.

1) To query data from all the sources, user has to type all the names of the sources in the parameter list. To avoid this a new value for the parameter '`from`' will be introduced as 'all' which will call the `spocc::occ` with all names of the sources appended in the function call.
   Ex:

`occ("Macaca radiata", from = c("gbif", "bison", "rinat", "rebird", "AntWeb", "ecoengine", "rvertnet", "ridigbio"))`
will be replaced with
`occ("Macaca radiata", from = "all")`

2) When querying the occurrence records the `limit` parameter sets the number of records to be returned. The default is 500 and user can set how many records he wants. But when he wants all the data in the backbone, he has to first find out how many records of data is stored in the backbone using `occ_count` function and set that value as limit. To make this easier a new value for `limit` parameter will be introduced as 'all' which would return all data of the query.

3) A new parameter will be introduced such that If user can query only a percentage of the data in the source.
Eg: If he wants only 50% of the occurrence data in gbif for a species, then he will be able to call the function as, `occ("Macaca radiata", truncate = 0.5).` So if there are 200 occurrence data in the source, he will only get 100 records and in another call can retrieve the next 50%. The reason for introducing this parameter is to help user allocate 'train' and 'test' datasets easily for analysis or predictive purposes.

4) The data returned by these functions can be overwhelming for inexperienced R users. The fields returned are most of the times not necessary for most of the researches. To help users easily manage the data a new parameter will be introduced, such that he can retrieve only the fields in these categories.
Occurrence, Event, Location, Geological Context, Identification and Taxon
Eg: `occ("Macaca radiata", fields = "Occurrence")` will return only the fields that are related with occurrence.

`biodiv_get_fast` *Search for GBIF occurrences - simplified for speed*

Tag: As is Uses: `rgbif::occ_data`

`biodiv_get_occ` *Get data for specific GBIF occurrences*

Tag: Enhancement Uses: `rgbif::occ_get spocc::inspect`

`rgbif::occ_get` returns occurrence data only from GBIF backbone and `spocc::inspect` returns data from other sources too. So, the enhancement is to use `occ_get` to get GBIF data if user types a numeral ID and use spocc to get data from various other sources too if user inputs results from `spocc::occ`.
Reason why I'm not using spocc to get GBIF data too is, spocc doesn't support getting occurrence data from a numeral occurrenceID. It requires occurrence ID as an occkey class whereas the rgbif returns data for even just a numeral. So this enhancement will give user number of options to get occurrence data without having to change the type of data user has.

`biodiv_facet` *Facet GBIF occurrences*

Tag: As is Uses: `rgbif::occ_facet`

`biodiv_download_request` *Spin up a download request for GBIF occurrence data*

Tag: As is Uses: `rgbif::occ_download`

`biodiv_download_get` *Get a download from GBIF*

Tag: As is Uses: `rgbif::occ_download_get`

`biodiv_download_cancel` *Cancel a download cretion process.*

Tag: As is Uses: `rgbif::occ_download_cancel`

`biodiv_download_import` *Import a downloaded file from GBIF*

Tag: As is Uses: `rgbif::occ_download_import`

`biodiv_download_list` *Lists the downloads created by a user*

Tag: As is Uses: `rgbif::occ_download_list`

`biodiv_download_meta` *Retrieves the occurrence download metadata by its unique key*

Tag: As is Uses: `rgbif::occ_download_meta`

Getting data utility functions

`biodiv_comm2sci` *Conversion from common name to scientific name*

Tag: As is Uses: `taxize::comm2sci`

`biodiv_sci2comm` *Conversion from scientific name to common name*

Tag: As is Uses: `taxize::sci2comm`

`biodiv_taxon_conversion` *Get taxonkey from names and vice versa*

Tag: Enhancement Uses: `rgbif::name_suggest, taxize::get_ids`

`rgbif::name_suggest` only checks the GBIF backbone to get the taxonkey and retrieves all related names and keys. But `taxize::get_ids` checks more sources to get wide range of taxon keys. `taxize::get_ids` returns GBIF data too, but returns only the exactly matched taxon key.

So the function will be enhanced such that the user can get the exact taxon key of the queried search or else get more suggestions from many sources related with the queried term. This will be controlled by a parameter.

Ex: `biodiv_taxon_conversion("Puma concolor", exact = TRUE)` will return,

```
2435099
```

whereas `biodiv_taxon_conversion("Puma concolor")` will return,

```
# A tibble: 28 × 3
      key canonicalName                rank
    <int> <chr>                      <chr>
1 2435099 Puma concolor              SPECIES
2 8836300 Puma concolor discolor     SUBSPECIES
3 7193927 Puma concolor concolor     SUBSPECIES
4 6164624 Puma concolor costaricensis SUBSPECIES
5 6164590 Puma concolor couguar      SUBSPECIES
# ... with 23 more rows
```

`biodiv_dataset_suggest` *Suggest datasets in GBIF*

Tag: As is Uses: `rgbif::dataset_suggest`

`biodiv_isocodes` *Table of country two character ISO codes, and GBIF names*

Tag: As is Uses: `rgbif::isocodes`

`biodiv_name_suggest` *Search for species names across many data sources*

Tag: Enhancement Uses: `rgbif::name_suggest, spocc::occ_names, taxize::get_ids`

`spocc::occ_names` searches for names across many sources, and `rgbif::name_suggest` only in GBIF. But `name_suggest` suggest names that match with only scientific name and `occ_names` checks in many other fields too. `taxize::get_ids` checks for matches in common names.

So the enhancement is to get the name and suggestions that match with scientific name and also common name will be given.
Eg:
A query for “lion” can return

1) Based on common names,
   `Auckland sea lion, a sea lion`
   `fleshy dandelion , a flower`
2) Based on scientific names
   `Abudefduf melanopselion, a fish`
   `Accipiter hiogaster polionotus, a bird`

`biodiv_spellcheck` *Spell check search term for occurrence searches*

Tag: As is Uses: `rgbif::occ_spellcheck`

`biodiv_country_codes` *Look up 2 character ISO country codes*

Tag: As is Uses: `rgbif::rgb_country_codes`

`biodiv_species_key` *Get species keys for multiple species*

Tag: As is Uses: `rgbif::name_backbone`

`biodiv_name_lookup` *Lookup names in all taxonomies in GBIF*

Tag: As is Uses: `rgbif::name_lookup`

`biodiv_to_dataframe` *Combine results from occ calls to a single data.frame*

Tag: Enhancement Uses: `spocc::occ2df`

The function converts the results of occ call to a dataframe. But since the occ uses various sources to get data only common fields in these source data are converted as dataframe. Currently only six fields are converted into a dataframe namely, Name, longitude, latitude, prov, date and key.

The enhancement is to return all fields in the as the result dataframe. The columns which doesn't match across sources will be merged by columns.

Getting data miscellaneous functions

`biodiv_facet` *Facetted count occurrence search*

Tag: As is Uses: `rgbif::count_facet`

`biodiv_count` *Get number of occurrence records*

Tag: As is Uses: `rgbif::occ_count`

`biodiv_photos` *View photos from GBIF*

Tag: As is Uses: `rgbif::gbif_photos`

`biodiv_dataset_search` *Search datasets in GBIF*

Tag: As is Uses: `rgbif::dataset_search`

`biodiv_to.gbif` *Coerce occurrence keys to gbifkey/occkey objects*

Tag: As is Uses: `rgbif::as.gbif`

`biodiv_bbox2wkt` *Convert a bounding box to a WKT polygon, and vice versa*

Tag: As is Uses: `rgbif::bbox2wkt`

`biodiv_wkt_vis` *Visualize well-known text area on a map*

Tag: As is Uses: `rgbif::wkt_vis`

Cleaning
Spatial Cleaning

`biodiv_coord_incomplete` *Add incomplete coordinate flag*

Tag: Enhancement Uses: `scrubr::coord_incomplete`, `rgeospatialquality::addflags`

The coord_incomplete flag from `scrubr::coord_incomplete` and nonZeroCoordinates flag from `rgeospatialquality::addflags` will be applied.

`biodiv_coord_impossible` *Add impossible coordinate flag*

Tag: Enhancement Uses: `scrubr::coord_impossible`, `rgeospatialquality::addflags`

The coord_impossible flag from `scrubr::coord_impossible` and validCoordinates flag from `rgeospatialquality::addflags` will be applied.

`biodiv_coord_unlikely` *Add likely coordinate flag*

Tag: As is Uses: `scrubr::coord_unlikely`

`biodiv_coord_within` *Add coordinate out of bounds flag*

Tag: Enhancement Uses: `scrubr::coord_within`, `rgeospatialquality::addflags`

The coord_within flag from `scrubr::coord_incomplete` and coordinatesInsideCountry flag from `rgeospatialquality::addflags` will be applied.

`biodiv_hasCountry` *Add country flag*

Tag: As is Uses: `rgeospatialquality::addflags`

`biodiv_validCountry` *Add validCountry flag*

Tag: As is Uses: `rgeospatialquality::addflags`

`biodiv_validCountryCode` *Add validCountryCode flag*

Tag: As is Uses: `rgeospatialquality::addflags`

`biodiv_validCoords` *Add valid Coordinates flag*

Tag: As is Uses: `rgeospatialquality::addflags`

Temporal Cleaning

`biodiv_incorrectdate` *Add incorrect date flag*

Tag: As is Uses: `scrubr::date`

`biodiv_missingDate` *Add missing date flag*

Tag: As is Uses: `scrubr::date`

Taxonomic Cleaning

`biodiv_epithet` *Add epithet missing flag*

Tag: As is Uses: `scrubr::tax_no_epithet`

`biodiv_hasScientificName` *Add scientific name flag*

Tag: As is Uses: `rgeospatialquality::addflags`

`biodiv_fixScientificName` *Fix bad flagged data*

Tag: Enhancement `Uses: taxize::gnr_resolve`

`taxize::gnr_resolve` returns a list of possible matches but doesn't change anything in the original data. So user has to select one name and manually fix names. The enhancement is once dataframe or result of occ call is passed to the function, it finds the scientific name column, calls the API, returns the possible fixes with matching scores and asks user to select a name. Then it changes all the names in the data to user specified name.

Conditional Cleaning

`flagsCustom`	*Add incomplete records flag in user specified columns*

Tag: New
The data returned by the occ call often has rows with missing values. The rows with no records in user specified columns can be flagged for incomplete records.
Ex: `flagsCustom("spatial")` will search for all the columns that are related with spatial attributes and check if there are 'NA's or empty cells in those columns and flag accordingly.

Cleaning Miscellaneous Functions

`duplicate`	*Add duplicate data flag*

Tag: Enhancement Uses: `scrubr::dedup`

The occ function returns data from many sources including GBIF. But GBIF tends to have many records from other organizations too. So when we get data from occ call the data at most times have duplicate entries.
Ex: occ call for the elephant with has_coords = TRUE returns many duplicate records like this.

```
                    name    longitude     latitude     prov          date    key
1   Mammuthus primigenius -153.3333     65.10083     bison         <NA>   896896230
2   Elephas primigenius   -153.3333     65.10083     gbif   1989-08-16   896896230
```

`scrubr::dedup` makes intelligent choices when finding duplicates but few things can be improved.

1) Here these two records are the same but scientific name is different. So considering name fields while de duplicating will not give an optimal cleaning since the name fields tend to have different versions of scientific names which come from various sources. `dedup` is not able to find these duplicates.
2) When deleting duplicates, `dedup` keeps the first occurring record and deletes all other duplicates. But in this example that would not be optimal since the first record is not complete but second record is.

So the enhancement is to also check the duplications made by different variants of scientific names and to remove records which has less data.

`Issue_flag`	*Add gbif issues flag*

Tag: As is Uses `rgbif::occ_issues`

`formatqg`	*Format fieldnames to comply with DwC*

Tag: As is Uses: `rgeospatialquality::format_gq`

`biodiv_validate_DWCA`	*Validate a Darwin Core Archive*

Tag: As is Uses: `finch::dwca_validate`

Cleaning Utility Functions

`Clean`	*Remove records with specific flags*

Tag: New

`view_flaggedData`	*View all flagged data*

Tag: New

`viewFlags`	*View all flags added*

Tag: New The list of flags added by user thus far will be shown

Ex:
```
> viewFlags()
# 3 flags added
#
# coord_impossible  Flagged for coordinates out of the possible range
# 33 rows flagged
#
# valid_country     Flagged for invalid county
# 8 rows flagged
#
# incorrect_date    Flagged for incorrect dates
# 12 rows flagged
```

`dropFlags` *Drop indexed flags*

Tag: New

`CleanAll`	*Clean with all attributes*

Tag: New

How cleaning works,
The user can keep adding multiple flags data. The flags added can be viewed and dropped in the process. Once the flags are added user can delete all the data with bad flags.

If user doesn't want these stepped cleaning, the cleanAll function call does all the flagging and deletes bad data.

Backup and retrieval functions

`createLocalRepo` *Create an Empty backup directory*

Tag: As is Uses: `archivist::createLocalRepo`

`setLocalRepo` *Set Directory's Global Path*

Tag: As is Uses: `archivist::setLocalRepo`

`saveToLocalRepo` *Backup an R object*

Tag: As is Uses: `archivist::saveToLocalRepo`

`showLocalRepo` *View the List of backups*

Tag: As is Uses: `archivist::showLocalRepo`

`loadFromLocalRepo` *Load backups*

Tag: As is Uses: `archivist::loadFromLocalRepo`

`aread` *Read backups*

Tag: As is Uses: `archivist::aread`

`deleteLocalRepo` *Delete the Existing backups*

Tag: As is Uses: `archivist::deleteLocalRepo`

`zipLocalRepo` *Create a zip Archive from backup*

Tag: As is Uses: `archivist::zipLocalRepo`

Timeline

I have divided the timeline to smaller parts of weeks and each week has a specific objective. The week starts from Monday and ends on Sunday. Coding will be done in the weekday from Monday to Friday and testing, debugging and documentation of the component developed that week will be completed in the weekend. Each week this will be strictly followed to ensure the code meets the standards and documentations and vignettes are thoroughly documented.

Apart from weekly objectives, each month will have a part of proposal as objective.

Month 01: Data Retrieval and Standardization functions
Month 02: Data Cleaning and Enrichment functions
Month 03: Data Backup functions and Kurator project, rOpenSci Integration process.

In total I will have to create 7 new functions, enhance 10 functions and implement existing 45 functions. Implementing existing functions will not require much time or effort but needs to be thoroughly tested. Enhancing functions will require moderate amount of time and new functions needs to be given much effort.

Data Retrieval and Standardization functions are the very core of the package and the functions are very complex. Thus more time (6 Weeks) will be spent on this even though most are enhancements and As-is. There are 31 total functions to be implemented in the first month but only 1 is new function. 25 are existing functions which needs just to be wrapped in the package. But the 5 enhancements are very important functions such as occ_get and occ_search and these 5 functions will require more time than the rest.

Data Cleaning and Enrichment has 23 functions to be implemented but only 6 are new functions. All these new functions are to deal with managing the flags from other functions which is not that complex. 12 are existing functions and 5 are enhancements and I have allocated 4 weeks for all these.

Data Backup has only 8 existing functions to be implemented thus I have allocated only 2 weeks for that.

Further I have allocated a week as Retrospective Week and two weeks as Contingency Week.

Retrospective Week: This is week is for stopping coding for a while and think back what I have achieved in the previous weeks, what could have been done better, to discuss and develop new features I thought of while doing the task with the mentor and to finish any unfinished works as of that time.
Contingency Week: This week comes at the end of the coding period to allocate time for the unforeseeable issues or delays.

That concludes the 15 entire weeks in the task period.

Detailed Timeline

<Community Bonding>
May 1 Monday - May 7 Sunday
I start to interact with the Mentor I am assigned with, get to know about organization and works.

May 8 Monday - May 14 Sunday <Week 01>
Coding Begins. All readings functions will be implemented.
2 As-is and 1 New function

May 15 Monday - May 21 Sunday <Week 02>
occ_data, get and search functions will be implemented.
2 As-is and 2 Enhancement functions. These are the most important and time consuming methods.

May 22 Monday - May 28 Sunday <Week 03>
occ_download related functions will be implemented.
6 As-is Functions.
</Community Bonding>
<Coding>
<Month 01>
May 29 Monday - June 4 Sunday <Month 01 Week 01 Continued>
Utility and miscellaneous functions will be implemented.
10 As-is Functions. All are simple functions which would not need much time

June 5 Monday - June 9 Friday <Month 01 Week 02>
Retrospective days (4 days).

June 10 Saturday - June 18 Sunday <Month 01 Week 03>
Utility and miscellaneous functions continued.
5 As-is and 3 Enhancement functions

June 19 Monday - June 25 Sunday <Month 01 Week 04>
Spatial cleaning functions will be implemented.
5 As-is and 3 Enhancement functions
<Evaluations>
June 26 Monday - June 30 Friday <Month 01 Week 05>
Temporal, Taxonomic, Duplicate and Conditional cleaning functions
4 As-is, 1 New, 2 Enhancement Functions
</Evaluations>
July 1 Saturday - July 2 Sunday <Month 01 Week 05 Continued>
</Month01>

<Month 02>
July 3 Monday - July 12 Wednesday <Month 02 Week 01>
Miscellaneous cleaning functions will be implemented.
3 As-is, 5 new functions will be implemented

July 13 Thursday - July 16 Sunday <Month 02 Week 02>
Retrospective days. (4 days)

July 17 Monday - July 23 Sunday <Month 02 Week 03>
Backup Functions will be implemented
8 As-is Functions

<Evaluations>
July 24 Monday - July 28 Friday <Month 02 Week 04>
Documentations will be combined, finalized. Including diagnostic report template and vignettes
</Evaluations>
July 29 Saturday - July 30 Sunday <Month 02 Week 04 Continued>
</Month 02>

<Month 03>
July 31 Monday - August 6 Sunday <Month 03 Week 01>
Kurator project and rOpenSci integration process starts

August 7 Monday - August 13 Sunday <Month 03 Week 02>
Additional time allocation for unexpected delays and enhancements beyond proposal

August 14 Monday - August 20 Sunday <Month 03 Week 03>
Additional time allocation for unexpected delays

</Month 03>
</Coding>

<Students Submit Code and Final Evaluations>
August 21 Monday - August 29 Tuesday
</Students Submit Code and Final Evaluations>

Management of Coding Project

The code will be committed to a Github repository. Ideally code will be committed every two days but this might change according to the function I am working at that time. If the code needs a lot of background knowledge, then it might take longer than two days to commit whereas smaller functions will be needed to committed as soon as that function is developed regardless of days.

As I mentioned in the Bio of Student section I have worked in agile project before and used Trello task management tool. I hope to use it for this task to better manage the work and keep the deadline checked.

The Trello board can be viewed here.

Trello board will be shared with the mentor and my progress can be tracked by the mentor always.

The task involves with packages created by the programmers in the community like Vijay Barve who was part of `rgbif` package. So when working with enhancing or using their functions, getting guidance and feedback from the original creators will be invaluable. So, my plans are to connect with these and many other mentors of the organization. I will add the mentors who are interested in the task to my task Trello Board. They can review my work continuously and give feedbacks leisurely as they get free time. The proof reading and checking the quality of the code has to be done by someone other than me, so these materials will be shared in the Trello for mentors to take a look.

I don't expect much commit conflicts to occur as this is a new code base and not an existing one and I will be the only main programmer during the task period. But I have experience with Git and GitHub and I would be able to solve the conflicts.

My GitHub profile can be viewed here.

The main challenge I will have in this task is in testing. Since the task involves with huge amount of data, we will have to test new functions for numerous number of test cases. Luckily since the task is to integrate functionalities from other packages, stress testing new functions against the already existing functions with large datasets will be the best way to ensure quality.

Test

I completed all the test tasks required for the task and the solution is in this repository.

There was however a small issue with the rgeospatialquality package.

Rgeospatialquality::addflags() returned an error even after following the vignette of the function. So I manually downloaded the GBIF data from portal and compared with the data returned by the RGBIF API, and the only difference was the portal data had lowercase column names. And after changing the case of column names of API data, the rgeospatialquality API accepted the request. So I have changed the cases of the column names in all the tasks using rgeospatialquality.

1) *Easy: Install packages 'rgbif' and 'rgeospatialquality'. Execute the 'add_flags' function on 500 records of the species 'Perameles nasuta' (follow this vignette).*

```
# Required Libraries
library(rgbif)
library(rgeospatialquality)

# Getting taxon key of any one species with names Perameles nasuta
taxonKey<-name_suggest("Perameles nasuta")
key<-taxonKey$key[1]

# Getting occurrence data of that species
data<-occ_search(key) #default limit = 500
data<- data$data

# Making column names lowercase
colnames(data) <- tolower(colnames(data))

#Adding flag to a tidy dataframe
tidyData <- as.data.frame(data)
tidyData <- add_flags(tidyData)

#return the flagged data
tidyData
```

2) *Do the same but retrieve data from the GBIF portal.*

```
# Required Libraries
library(rgeospatialquality)

# Reading downloaded portal data
data <-
    read.csv(
        "0067842-160910150852091/0067842-160910150852091.csv",
        header = TRUE,
        sep = "\t",
        nrows = 500
    )

# Flagging
flaggedData <- add_flags(data)

flaggedData
```

3) *Medium: Write a function that retrieves from GBIF 5000 georeferenced records of Australian mammals, and then sends all of them successfully to the Geospatial Quality API.*

```
#' Function for Retrieving 5000 georeferenced records of Australian mammals from GBIF and
adding quality flags.
#'
#' @author Thiloshon Nagarajah
#' @description {
#' GSOC
#' Integrating Biodiversity Data Curation Functionality.
#' Test Medium 01.
#' }
#' @param country The 2-letter country code (as per ISO-3166-1)
#' @param hasCoordinate A number
#' @param limit Number of records to return
#' @param classKey Class classification key
#' @return A tidy dataframe of occurancedata and quality flags of fields as specified in
parameters.
#' @examples
#' addQualityFlag()
#' addQualityFlag("SL", False, 100, 148)

addQualityFlag <-
    function(country = "AU",
             hasCoordinate = TRUE,
             limit = 5000,
             classKey = 121) {
        # Required Libraries
        library(rgbif)
        library(plyr)
        library(rgeospatialquality)

        # Retrieving data
        rawData <-
            occ_data(
                classKey = classKey,
                country = country,
                hasCoordinate = hasCoordinate,
                limit = limit
            )
        data <- as.data.frame(rawData$data)

        # Making coloumn names simplecase
        colnames(data) <- tolower(colnames(data))

        # Splitting data into 5 components of 1000 records each since only 1000 records
        #   can be sent through rgeospatialquality API
        stripData <-
            split(data, rep(1:5, each = round(NROW(data) / 5)))

        # Adding flags to each component
        finalData <- lapply(stripData, add_flags)

        # Collapsing flags dataframe in data dataframe returned by the API.
        formatData <- lapply(finalData, function(df) {
            j <- df[, 1:NCOL(df) - 1]
            h <- df$flags
            df <- cbind(j, h)
        })

        # Combining the split data
        df <- ldply(formatData, rbind)[2:117]

        # Returning final data
        df    }
```

Other two tasks don't involve any coding, but documentation. It can be found in the linked GitHub repository.

Appendix

Unique Fields Grouping

Record-Level Terms (Darwin Core)

From: GBIF (2010). Darwin Core Quick Reference Guide, version 1.3, released on 10 Mar 2012, (contributed by Wieczorek , J., De Giovanni , R., Vieglais , D. Remsen D.P., Döring, M, Robertson, T.), Copenhagen: Global Biodiversity Information Facility, 41 pp,.

Terms under the Record-level Terms section apply to the whole record regardless of the record type.

institutionID
collectionID
datasetID
institutionCode
collectionCode
datasetName
ownerInstitutionCode
basisOfRecord
informationWithheld
dataGeneralizations
dynamicProperties

Class Occurrence

The category of information pertaining to evidence of an occurrence in nature, in a collection, or in a dataset (specimen, observation, etc.)

occurrenceID
catalogNumber
occurrenceDetails
occurrenceRemarks
recordNumber
recordedBy
individualID
individualCount
sex
lifeStage
reproductiveCondition
associatedTaxa
behavior
establishmentMeans
occurrenceStatus
preparations
disposition
otherCatalogNumbers
previousIdentifications
associatedMedia
associatedReferences
associatedOccurrences
associatedSequences

Class Event

The category of information pertaining to an event (an action that occurs at a place and during a period of time).

eventID
samplingProtocol
samplingEffort
eventDate
eventTime
startDayOfYear
endDayOfYear
year
month
day
verbatimEventDate
habitat
fieldNumber
fieldNotes
eventRemarks

Class Location

A spatial region or named place. For Darwin Core, a set of terms describing a place, whether named or not.

locationID
higherGeographyID
higherGeography
verbatimLongitude
verbatimCoordinateSystem
verbatimSRS

continent
waterBody
islandGroup
island
country
countryCode
stateProvince
county
municipality
locality
verbatimLocality
verbatimElevation
minimumElevationInMeters
maximumElevationInMeters
verbatimDepth
minimumDepthInMeters
maximumDepthInMeters
minimumDistanceAboveSurfaceInMeters
verbatimLatitude
decimalLatitude
decimalLongitude
geodeticDatum
coordinateUncertaintyInMeters
coordinatePrecision
pointRadiusSpatialFit
footprintWKT
footprintSRS
footprintSpatialFit
georeferencedBy
georeferencedDate
georeferenceProtocol
georeferenceSources
georeferenceVerificationStatus
georeferenceRemarks
maximumDistanceAboveSurfaceInMeters
locationAccordingTo
locationRemarks
verbatimCoordinates

Class Geological Context
The category of information pertaining to a location within a geological context, such as stratigraphy.

geologicalContextID
earliestEonOrLowestEonothem
latestEonOrHighestEonothem
earliestEraOrLowestErathem
latestEraOrHighestErathem
earliestPeriodOrLowestSystem
latestPeriodOrHighestSystem
earliestEpochOrLowestSeries
latestEpochOrHighestSeries
earliestAgeOrLowestStage
latestAgeOrHighestStage
lowestBiostratigraphicZone
highestBiostratigraphicZone
lithostratigraphicTerms
group
formation
member
bed

Class Identification
The category of information pertaining to taxonomic determinations (the assignment of a scientific name).

identificationID
identifiedBy
dateIdentified
identificationReferences
identificationVerificationStatus
identificationRemarks
identificationQualifier
typeStatus

Class Taxon
The category of information pertaining to taxonomic names, taxon name usages, or taxon concepts.

taxonID
scientificNameID
acceptedNameUsageID
parentNameUsageID
originalNameUsageID
nameAccordingToID
namePublishedInID
taxonConceptID
scientificName
acceptedNameUsage
parentNameUsage
originalNameUsage
nameAccordingTo
namePublishedIn
namePublishedInYear
higherClassification
kingdom
phylum
class
order
family
genus
subgenus
specificEpithet
infraspecificEpithet
taxonRank
verbatimTaxonRank
scientificNameAuthorship
vernacularName
nomenclaturalCode

Auxiliary terms – ResourceRelationship

Information about relationships between resources (instances of data records, such as Occurrences, Taxa, Locations, Events).

Resources can be thought of as identifiable records and may include, but need not be limited to Occurrences, Locations, Events, Identifications, or Taxon records. Auxiliary terms are meaningful in a relational database but not applicable for a flat database structure.

resourceRelationshipID
resourceID
relatedResourceID
relationshipOfResource
relationshipAccordingTo
relationshipEstablishedDate
relationshipRemarks
relationshipAccordingTo

Auxiliary terms – MeasurementOrFact
Information about measurements, facts, characteristics, or assertions about a resource (instance of data record, such as Occurrence, Taxon, Location, Event).
Auxiliary terms are meaningful in a relational database but not applicable for a flat database structure.

measurementID
measurementType
measurementValue
measurementAccuracy
measurementUnit
measurementDeterminedBy
measurementMethod
measurementRemarks