PPC_Pipline_Supplementary_Documentation

Johannes Nelson

2023-12-19

Overview

In addition to the documentation within the scripts in the form of comments, this file provides a higher level overview of each script, highlighting key points for functionality, as well as any key assumptions being made.

Extract_Main_Data.R

General

This scripts purpose is to take the data out of Kobo, process it and clean it, and produce organized, easily-legible files for further analysis.

Inputs/Requirements

Since this is the first stage, there are no local inputs. However, the user will be prompted for the Kobo username and password, so these need to be handy and that user's permissions need to be such that they can extract data in this way.

Outputs

The outputs of this script are files for submission data, tree data, photo data, and geolocation data. They will be stored in a folder called 'Main_Data'.

Assumptions

There are some assumptions hard-coded into this script. * The script assumes that the name mappings in the 'Kobo_Key' document are accurate. * The script assumes naming conventions within Kobo fields does not change with future form versions (e.g. it searches for geolcation data by looking for the strings 'Corner' and 'Centroid'. If these fields are changed in future form versions, it will not be able to find them. The same goes for key data fields like tree count, species, etc.) Here is a coded example:

```
    names(df)[grep("number", names(df), ignore.case = TRUE)] <- "Tree_Count"
    names(df)[grep("species", names(df), ignore.case = TRUE)] <- "Species"
    names(df)[grep("type", names(df), ignore.case = TRUE)] <- "Tree_Type"
```

This chunk of code conforms column labels by searching for key words within the column labels as they are in Kobo. If these change, this will not find them. Also, if a new column is later added with the word number, species, or type as a part of its name, this will try to label that column with the uniform labels incorrectly.

- Removal of empty columns: There are a number of entirely empty columns in the dataset that mostly correspond to fields that are not inputs but descriptions of parts of the Kobo form. The assumption here is that if there is not a single piece of information from any of the submissions, it is likely one of these technical fields.

Troubleshooting

- **'robotoolbox' R package issue:** At the time of creation, there was a bug in this package, and I needed the creator to go in and fix it, which he was kind enough to do. However, the fix is currently only applied to the development version, so installing this package in the conventional way will not install the version we need. If you get error that looks something like this:

Error in rbindlist(drop_nulls(survey), fill = TRUE): Column 7 of item 3 is length 3 inconsistent with column 1 which is length 5. Only length-1 columns are recycled.

Then you have the older version. Try this:

```
remove.packages("robotoolbox")
install.packages("remotes")
remotes::install_gitlab("dickoa/robotoolbox")
```

- **JSON error:** This happened rarely, but if you encounter an error that talks about JSON format/parsing, wait and try to run the script again. It has to do with a faulty connection/communication step between your machine and Kobo.
- **Table changes within Kobo:** While unlikely at this point in data collection, there is ever-present potential that Kobo will confusingly store things in opaque and mysterious ways that are slightly different than they were at the time I wrote the script. I have included some out puts int he console that should be able to help you see if this is happening, such as the number of expected and retreived tables. Printed statements are also created at each step of the script, so if an error does appear, you can locate which module created the error easily and address the issue.

Extract_Brazil_Data.R

General

This does the same thing as the above script, and everything above pertains to this one as well, with one note I would like to add.

Because of the relatively small amount of submissions present in the database at the time of creating the script, I feel less certain that the submissions (including my test submissions) cover all bases of what might get put into the database. Since the main data is full of strange tables, I assume that with time, some of these peculiarities may appear in the Brazil dataset as well. If this happens, the script will need to be altered to accomodate the changes.

Correct_Species_Names.R

General

This script uses the 'taxize' package to facilitate connection the the Global Names Resolver while attempting to map species names present in the Kobo database with accpeted species names in known scientific databases.

Inputs/Requirements

- It will take the extracted data from the previous scripts as input–primarily the file with "Tree_Data_Uncorrected" within its name.
- If one exists, which it always should since the repository contains one at the time of hand-over, a 'Taxonomic_Corrections' file in the 'Species_Data' folder will also be used to automatically make corrections that are already known. The script will work without this file, but it will take a very long time to run (20+ minutes just for the automated part–much longer for manual validation).

Outputs

- Tree_Data_Corrected_YYYY-MM-DD_HHMM.csv: tree data with corrected names.
- Taxonomic_Corrections_YYY-MM-DD_HHMM.csv: updated taxonomic corrections.
- Optionally, at the end the user can be prompted to get a file showing the species that need review.

Assumptions

- Extraction scripts have been run
- Two databases are used for name matching: Tropicos - Missouri Botanical Garden and The International Plant Names Index. Others are available if these are not desired.

Troubleshooting

- Runtime: as mentioned, there is a part of the script that takes a while to run, so if you think your R session is frozen, it probably has not.
- Manual validation: This can take a long time, since the script will always show you all the species that it was not able to resolve, and prompt you for the correct name. Hundreds of these are likely common names. If you've corrected all the species names recently and are running the script again, it will still show you all these common names one by one, which can be onerous. I included a 'hidden' option, where if you type the word 'save' as a response, it will move forward and assume you know no corrections for those names.

Add_Family_Names.R

General

This script takes the cleaned and corrected species names and attempts to find family names for them. It does this using the 'rgbif' package to interact with the GBIF database. There is also 'hidden' functionality to use the 'taxize' package to pull from the ncbi database. Results were only slightly different, and since the GBIF method was much faster and required no API token set up (which would have been an obnoxious step that each user would have to do before running the first time), I choose the GBIF route. It there is interest in the other way, I can facilitate that too.

Inputs

- Tree_Data_Corrected_YYYY-MM-DD_HHMM.csv: the output from the previous script with corrected species names.
- Family_Names_YYYY-MM-DD_HHMM.csv: the most up to date file with known family names for each species.

Outputs

- Final_Tree_Data_YYY-MM-DD_HHMM.csv: the final tree data file (before passing to analysis), with correct species names and family names added.
- Family_Names_YYYY-MM-DD_HHMM.csv: An up to date family names file.

Assumptions

- No major assumptions other than the reliability of the GBIF database.

Troubleshooting

- This script will ask the user if they want to manually add family names for those species which could not be automatically matched. If choosing yes, the user will then have to go through each unresolved species name and look up a family name for it. Only run this if you have time. Unlike species names, I did not have time to create a manually up-to-date family names file, so this will require some effort the first time it is run.

Analyze_Data.R

IMP_Invasive_Species_Scanner.R

IMP_Native_Alien_Classification.R