Darwin Core mapping

For: my_dataset_title

author_1
author_2

2018-08-10

Contents

1 Setup

Load libraries:

```
library(tidyverse)      # To transform data
library(magrittr)       # To use %<>% pipes
library(here)           # To find files
library(janitor)        # To clean input data
library(readxl)         # To read Excel files
library(digest)         # To generate hashes
library(rgbif)          # To use GBIF services
```

2 Read source data

Create a data frame `input_data` from the source data:

```
input_data <- read_excel(path = here("data", "raw", "checklist.xlsx"))
```

Preview data:

```
input_data %>% head(n = 5)
```

scientific_name	kingdom	country_code	locality	occurrence_status
Bassia laniflora (S.G. Gmel.) A.J. Scott	Plantae	BE	Flanders	present
Amaranthus macrocarpus Benth. var. pallidus Benth.	Plantae	BE	Walloon region	present
Amaranthus macrocarpus Benth. var. pallidus Benth.	Plantae	BE	French region	present
Euphorbia x pseudovirgata (Schur)	Plantae	FR	NA	present
Alytes obstetricans	Animalia	BE	NA	present

3 Process source data

3.1 Rows and columns

Clean data somewhat:

```
input_data %<>%
  remove_empty("rows") %>%  # Remove empty rows
  clean_names()             # Have sensible (lowercase) column names
```

Add prefix `input_` to all column names to avoid name clashes with Darwin Core terms:

```
colnames(input_data) <- paste0("input_", colnames(input_data))
```

3.2 Scientific names

3.2.1 Retrieve nomenclatural information

Use the GBIF nameparser to retrieve nomenclatural information for the scientific names in the checklist:

```
parsed_names <- input_data %>%
  distinct(input_scientific_name) %>%
  pull() %>% # Create vector from dataframe
  parsenames() # An rgbif function
```

Show scientific names with nomenclatural issues, i.e. not of `type = SCIENTIFIC` or that could not be fully parsed. Note: these are not necessarily incorrect.

```
parsed_names %>%
  select(scientificname, type, parsed, parsedpartially, rankmarker) %>%
  filter(!(type == "SCIENTIFIC" & parsed == "TRUE" & parsedpartially == "FALSE"))
```

scientificname	type	parsed	parsedpartially	rankmarker
Cotoneaster x 'Hybridus pendulus'	HYBRID	FALSE	FALSE	NA
Acmella agg.	INFORMAL	TRUE	FALSE	agg.
AseroÙ rubra	SCIENTIFIC	TRUE	TRUE	NA
Triticosecale x	SCIENTIFIC	TRUE	TRUE	NA

Correct names and reparse:

```
input_data %<>% mutate(input_scientific_name = recode(input_scientific_name,
  "AseroÙ rubra" = "Asero rubra"
))

# Redo parsing
parsed_names <- input_data %>%
  distinct(input_scientific_name) %>%
  pull() %>%
  parsenames()

# Show names with nomenclatural issues again
parsed_names %>%
  select(scientificname, type, parsed, parsedpartially, rankmarker) %>%
  filter(!(type == "SCIENTIFIC" & parsed == "TRUE" & parsedpartially == "FALSE"))
```

scientificname	type	parsed	parsedpartially	rankmarker
Cotoneaster x 'Hybridus pendulus'	HYBRID	FALSE	FALSE	NA
Acmella agg.	INFORMAL	TRUE	FALSE	agg.
Triticosecale x	SCIENTIFIC	TRUE	TRUE	NA

3.2.2 Add taxonRank information

The nameparser function also provides information about the rank of the taxon (in `rankmarker`). Here we join this information with our checklist. Cleaning these ranks will done in the Taxon Core mapping:

```
input_data %<>% left_join(
  select(parsed_names, scientificname, rankmarker),
  by = c("input_scientific_name" = "scientificname")) %>%
  rename(input_rankmarker = rankmarker)
```

3.2.3 Generate taxonID

To link taxa with information in the extension(s), each taxon needs a unique and relatively stable taxonID. Here we create one in the form of `dataset_shortname:taxon:hash`, where `hash` is unique code based on scientific name and kingdom (that will remain the same as long as scientific name and kingdom remain the same):

```
vdigest <- Vectorize(digest) # Vectorize digest function to work with vectors
input_data %<>% mutate(input_taxon_id = paste(
  "my_dataset_shortname", # e.g. "alien-fishes-checklist"
  "taxon",
  vdigest(paste(input_scientific_name, input_kingdom), algo = "md5"),
  sep = ":"
))
```

3.2.4 Show summary

Show the number of taxa and distributions per kingdom and rank:

```
input_data %>%
  group_by(input_kingdom, input_rankmarker) %>%
  summarize(
    `# taxa` = n_distinct(input_taxon_id),
    `# distributions` = n()
  ) %>%
  adorn_totals("row")
```

input_kingdom	input_rankmarker	# taxa	# distributions
Animalia	sp.	2	2
Fungi	sp.	1	1
Plantae	agg.	1	1
Plantae	infrasp.	1	1
Plantae	sp.	2	2
Plantae	var.	1	2
Plantae	NA	2	2
Total	-	10	11

3.3 Preview data

```
input_data %>% head()
```

input_scientific_name	input_kingdom	input_country_code	input_locality	input_
Bassia laniflora (S.G. Gmel.) A.J. Scott	Plantae	BE	Flanders	present
Amaranthus macrocarpus Benth. var. pallidus Benth.	Plantae	BE	Walloon region	present
Amaranthus macrocarpus Benth. var. pallidus Benth.	Plantae	BE	French region	present
Euphorbia x pseudovirgata (Schur)	Plantae	FR	NA	present
Alytes obstetricans	Animalia	BE	NA	present
Plebejus dardanus	Animalia	MK	NA	present

4 Create taxon core

Create a dataframe with unique taxa only (ignoring multiple distribution rows):

```
taxon <- input_data %>% distinct(input_taxon_id, .keep_all = TRUE)
```

4.1 Term mapping

Map the data to Darwin Core Taxon.

Start with record-level terms which contain metadata about the dataset (which is generally the same for all records).

4.1.1 language

```
taxon %<>% mutate(language = "my_language") # e.g. "en"
```

4.1.2 license

```
taxon %<>% mutate(license = "my_license") # e.g. "http://creativecommons.org/publicdomain/zero/1.0/"
```

4.1.3 rightsHolder

```
taxon %<>% mutate(rightsHolder = "my_rights_holder") # e.g. "INBO"
```

4.1.4 datasetID

```
taxon %<>% mutate(datasetID = "my_dataset_doi") # e.g. "https://doi.org/10.15468/xvuzfh"
```

4.1.5 institutionCode

```
taxon %<>% mutate(institutionCode = "my_institution_code") # e.g. "INBO"
```

4.1.6 datasetName

```
taxon %<>% mutate(datasetName = "my_dataset_title") # e.g. "Checklist of non-native freshwater fishes i
```

The following terms contain information about the taxon:

4.1.7 taxonID

```
taxon %<>% mutate(taxonID = input_taxon_id)
```

4.1.8 scientificName

```
taxon %<>% mutate(scientificName = input_scientific_name)
```

4.1.9 kingdom

Inspect values:

```
taxon %>%
  group_by(input_kingdom) %>%
  count()
```

input_kingdom	n
Animalia	2
Fungi	1
Plantae	7

Map values:

```
taxon %<>% mutate(kingdom = input_kingdom)
```

4.1.10 taxonRank

Inspect values:

```
taxon %>%
  group_by(input_rankmarker) %>%
  count()
```

input_rankmarker	n
agg.	1
infrasp.	1
sp.	5
var.	1
NA	2

Map values by recoding to the GBIF rank vocabulary:

```
taxon %<>% mutate(taxonRank = recode(input_rankmarker,
  "agg."      = "speciesAggregate",
  "infrasp."  = "infraspecificname",
  "sp."       = "species",
  "var."      = "variety",
  .missing    = ""
))
```

Inspect mapped values:

```
taxon %>%
  group_by(input_rankmarker, taxonRank) %>%
  count()
```

input_rankmarker	taxonRank	n
agg.	speciesAggregate	1
infrasp.	infraspecificname	1
sp.	species	5
var.	variety	1
NA		2

4.1.11 nomenclaturalCode

```
taxon %<>% mutate(nomenclaturalCode = "my_nomenclaturalCode")
```

4.2 Post-processing

Remove the original columns:

```
taxon %<>% select(-starts_with("input_"))
```

Preview data:

```
taxon %>% head()
```

language	license	rightsHolder	datasetID	institutionCode	datasetName	taxonID
my_language	my_license	my_rights_holder	my_dataset_doi	my_institution_code	my_dataset_title	my_data
my_language	my_license	my_rights_holder	my_dataset_doi	my_institution_code	my_dataset_title	my_data
my_language	my_license	my_rights_holder	my_dataset_doi	my_institution_code	my_dataset_title	my_data
my_language	my_license	my_rights_holder	my_dataset_doi	my_institution_code	my_dataset_title	my_data
my_language	my_license	my_rights_holder	my_dataset_doi	my_institution_code	my_dataset_title	my_data
my_language	my_license	my_rights_holder	my_dataset_doi	my_institution_code	my_dataset_title	my_data

Save to CSV:

```
write.csv(taxon, file = here("data", "processed", "taxon.csv"), na = "", row.names = FALSE, fileEncodin
```

5 Create distribution extension

Create a dataframe with all data:

```
distribution <- input_data
```

5.1 Term mapping

Map the data to Species Distribution.

5.1.1 taxonID

```
distribution %<>% mutate(taxonID = input_taxon_id)
```

5.1.2 locality

Inspect values:

```
distribution %>%
  group_by(input_country_code, input_locality) %>%
  count()
```

input_country_code	input_locality	n
BE	Flanders	1
BE	French region	1
BE	Walloon region	1
BE	NA	2
FR	NA	1
GB	NA	3
MK	NA	1
NL	NA	1

Map values to `input_locality` if provided, otherwise use the country name:

```
distribution %<>% mutate(locality = case_when(
  !is.na(input_locality) ~ input_locality,
  input_country_code == "BE" ~ "Belgium",
  input_country_code == "GB" ~ "United Kingdom",
  input_country_code == "MK" ~ "Macedonia",
  input_country_code == "NL" ~ "The Netherlands",
  TRUE ~ "" # In other cases leave empty
))
```

Inspect mapped values:

```
distribution %>%
  group_by(input_country_code, input_locality, locality) %>%
  count()
```

input_country_code	input_locality	locality	n
BE	Flanders	Flanders	1
BE	French region	French region	1
BE	Walloon region	Walloon region	1
BE	NA	Belgium	2
FR	NA		1
GB	NA	United Kingdom	3
MK	NA	Macedonia	1
NL	NA	The Netherlands	1

5.1.3 countryCode

Inspect values:

```
distribution %>%
  group_by(input_country_code) %>%
  count()
```

input_country_code	n
BE	5
FR	1
GB	3
MK	1
NL	1

Map values:

```
distribution %<>% mutate(countryCode = input_country_code)
```

5.1.4 occurrenceStatus

Inspect values:

```
distribution %>%
  group_by(input_occurrence_status) %>%
  count()
```

input_occurrence_status	n
doubtful	1
present	10

Map values:

```
distribution %<>% mutate(occurrenceStatus = input_occurrence_status)
```

5.1.5 threatStatus

Inspect values:

```
distribution %>%
  group_by(input_threat_status) %>%
  count()
```

input_threat_status	n
endangered	1
vulnerable	1
NA	9

Map values by recoding to the IUCN threat status vocabulary:

```
distribution %<>% mutate(threatStatus = recode(input_threat_status,
  "endangered" = "EN",
  "vulnerable" = "VU"
))
```

Inspect mapped values:

```
distribution %>%
  group_by(input_threat_status, threatStatus) %>%
  count()
```

input_threat_status	threatStatus	n
endangered	EN	1
vulnerable	VU	1
NA	NA	9

5.1.6 source

Inspect values:

```
distribution %>%
  group_by(input_source) %>%
  count() %>%
  head() # Remove to show all values
```

input_source
Bauwens D & Claus K (1996) Verspreiding van amfibieën en reptillen in Vlaanderen. De Wielewaal, Turnhout
https://doi.org/10.1007/s10530-016-1278-z
https://doi.org/10.3897/neobiota.23.5665
Verloove F (2018) Manual of Alien Plants of Belgium. Botanic Garden of Meise, Belgium. At: http://alienplantsbelgium.be
NA

Map values:

```
distribution %<>% mutate(source = input_source)
```

5.1.7 occurrenceRemarks

Inspect values:

```
distribution %>%
  group_by(input_remarks) %>%
  count() %>%
  head() # Remove to show all values
```

input_remarks	n
Absent according to Rudi Verovnik	1
data based solely on DAISIE portal	1
NA	9

Map values:

```
distribution %<>% mutate(occurrenceRemarks = input_remarks)
```

5.2 Post-processing

Remove the original columns:

```r
distribution %<>% select(-starts_with("input_"))
```

Preview data:

```r
distribution %>% head()
```

taxonID	locality	countryCode	occurrenceStatus	t
my__dataset__shortname:taxon:6c17452b1e85e24150e742cd5c32f353	Flanders	BE	present	N
my__dataset__shortname:taxon:6c341828da8ef55771f60db2bca5d530	Walloon region	BE	present	N
my__dataset__shortname:taxon:6c341828da8ef55771f60db2bca5d530	French region	BE	present	N
my__dataset__shortname:taxon:8c1b5d2f1230258c8aab8080de58adc8		FR	present	N
my__dataset__shortname:taxon:1e76cf83153eacf1bd350d2555445b1e	Belgium	BE	present	B
my__dataset__shortname:taxon:1e68c6f279e725acadb669262525a333	Macedonia	MK	present	V

Save to CSV:

```r
write.csv(distribution, file = here("data", "processed", "distribution.csv"), na = "", row.names = FALS
```