DASCO Manual

Ekin Kaplan, Hanno Seebens

17 Mayıs 2021

Introduction

The workflow "Downscaling Alien Species Checklists using Occurrence data" (DASCO) has been developed to assign alien species records available on regional checklists to individual populations and to deviating geographic delineations. Checklists represent lists of taxa, which have been recorded as being alien in a certain geographic region. Such checklists often contain taxa of a range of taxonomic groups in various habitats. The DASCO workflow can then be applied to e.g. differentiate marine and terrestrial taxa and assign the recorded populations to the respective geographic location within the marine or terrestrial biogeographic region. In this way, information obtained from checklists can be made available for spatial analyses or regional assessments with different geographic boundaries. This manual aims to explain the requirements and methods to use this workflow in detail.

The DASCO workflow takes advantage of the coordinate-based occurrences of species provided by the Global Biodiversity Information Facility (GBIF) and Ocen Biodiversity Information System (OBIS). For each taxon listed in the checklist, occurences are obtained from GBIF and OBIS. Obtained data were cleaned and standardised. To reduce mis-classifications, taxa were classified into marine and terrestrial taxa based on broad taxonomic information and on the categorisation provided by the World Register of Marine Species (WoRMS). Finally, occurrences were assigned to a set of geographic regions provided by the user.

All parts of the workflow are open source and can be modified by the user, but it is strongly recommend to report all modifications of the provided codes and tables together with the versions of the individual databases in the publication of the respective results to ensure transparency and reproducibility.

The workflow has been created in the R software, version 4.0.0 (R Core Team 2019).

Requirements:

Full execution of the workflow requires the following configuration: 1. An installed R software (version 4.0.0 or higher) and Rstudio, 2. installed R packages "data.table", "rgbif", "robis", "worrms", "CoordinateCleaner", "httr", "sf", and "R.utils", 3. at least one gbif.org account and 4. a stable internet connection. Note that the requirements 2-4 are specific to individual parts of the workflow. Parts of the workflow can therefore run in isolation without all requirements met.

The workflow has been tested on Linux and Windows.

1. The R environment:

R (version 4.0.0 or higher) and Rstudio needs to be installed on the computer, which can be freely obtained from https://cran.r-project.org and https://www.rstudio.com/, respectively. Eight required packages and their dependencies must be installed. Seven of these packages can be obtained from CRAN, and the remaining package "robis" can be obtained from GitHub. To download these packages, the following code can be executed in the terminal.

```
install.packages("data.table")
install.packages("rgbif")
install.packages("worrms")
install.packages("CoordinateCleaner")
install.packages("httr")
install.packages("sf")
install.packages("R.utils")
install_github("iobis/robis")
```

2. Workflow scripts and tables

The R scripts with the implemented workflow can be obtained from "https://github.com/hseebens/DASCOworkflow", which provides version control. In addition, releases of the workflow are stored at Zenodo () to obtain a digital unique identifier (DOI).

The downloaded zip file contains the required folder structure with the subfolders R/ and Data/. The zip file needs to be extracted to a folder defined by the user. The main folder contains the file "DASCOworkflow.Rproj", which is the Rstudio project file of the DASCO workflow. This file should be opened to use the workflow. The subfolder R/ contains all R scripts to run the workflow.

The subfolder Data/ contains two other subfolders named Input/ and Output/. The Input/ subfolder contains all files required to run the workflow such as location names and shapefiles, and the dataset that is going to be processed by the workflow. The file "AllLocations.csv" contains a list of location names and characteristics such as ISO3 codes, which is used as reference for the standardization of location names. This file can be modified by the user by e.g. adding new location names or alternative names of the region. Changes in the column 'Location' such as modification of the location name or the addition/removal of locations have to be applied to the shapefile in the subfolder Input/Shapefiles so that locations can be identified consistently.

All output of the workflow will be stored in the subfolder Output/. Output, which is produced within the course of the workflow application but only relevant for intermediate steps, are stored in the subfolder Output/Intermediate/

3. Data sets

Shapefile AllLocations Taxon_Region

Executing the DASCO workflow

To run the workflow, the Rstudio project file "DASCOworkflow.Rproj" has to be opened, which sets the folder, where it is located, as the working directory. The workflow can be run and all settings can be modified by using the file "run_DASCO_workflow.R" (see below).

Step 0. Preparation and modification of the default setting

The successful execution of the workflow requires some basic information, which needs to be provided in the file "run_DASCO_workflow.R". In this script, default settings of the workflow are provided under 'Global variables':

```
path_to_GBIFdownloads <- "/path/to/storage/GBIF/"
path_to_OBISdownloads <- "/path/to/storage/OBIS"

filename_inputData <- "Trial.csv"
column_scientificName <- "scientificName"
column_taxonName <- "TaxonName"
column_location <- "Region"
```

```
column_eventDate <- "FirstRecord"

name_of_shapefile <- "RegionsTerrMarine"

file_name_extension <- "FirstRecords"
```

The first two variables called "path_to_GBIFdownloads" and "path_to_OBISdownloads" define the folder to store the downloaded records from OBIS and GBIF. Depending on the list of species the files might be several GB large. Users should note that all files in these folders will be considered as being relevant files for further processing.

The following five variables are related to the data set, which contains the lists of taxon names from regional checklists. This file has to be stored in the folder Data/Input/. The preferred file format is csv with the following columns:

1. The name of the data set containing the taxon list has to be named in the variable "filename_inputData".
2. The variable "column_scientificName" specifies the column of the input data set with the scientific names of taxa with or without authority. The addition of the authority increases chances to obtain a match of taxon names at GBIF.
3. The "column_taxonName" variable denotes the column of the input data set with the binomial of the taxon without authority. This information is required to match taxon names at OBIS.
4. The "column_location" variable describes the name of the region where the species has been recorded.
5. The "column_eventDate" variable refers to the column containing the first record (i.e., the year when a species has been first described in a region). This variable is optional and missing values are allowed.

The workflow assigns occurrences of alien populations to regions as provided by polyons in a shapefile within the folder Data/Input/Shapefiles. The name of the shapefile can be specified in the variable "name_of_shapefile". As a default, a shapefile providing spatial boundaries of 517 marine and terrestrial regions is provided. Note that a modification of the shapefile should also be reflected in the file "AllLocations.csv".

The names of the files produced by the workflow can be extended by a term specified in the variable file_name_extension. That is, the content of this variable is added to each file name exported by the workflow. In this way, the user can e.g. determine the version of the workflow run, and ensures that previous output files are not overwritten.

source(file.path("R","load_functions.R")) # load all required functions

Users should start with loading the required packages and scripts that are specified in the "run_DASCO_workflow.R" main script. This can easily be done by running the main script until the "create_folders" function line (If users cannot load the packages, they should refer to former section 1. The R Environment).

Running the "create_folders" function checks if the necessary folder structure is present and creates the folder structure if there are problems.

The "prepare_dataset" function utilizes the former variables to run. This function standardizes the information that resides in the columns of the dataset, such as location and taxon names.

The following four variables are related to GBIF API. In the "n_accounts" variable, users can specify how many GBIF.org accounts they want to use. Using multiple accounts serves the purpose of downloading the occurrence data from GBIF in multiple chunks to process them much easier. "user", "pwd", and "email" variables require users to provide the information on their GBIF.org account. If users want to chunk their data and use multiple accounts, "user" and "email" variables should be created and specified in a structured manner, which means that the usernames and email accounts should have subsequent numbers at the end of the names, such as: user <- "hannoseebens1", email <- "hannoseebens1@outlook.com". "pwd" variable, which is specified for GBIF.org password, must be the same for every account.

After all these variables are defined and functions are executed, the preparation step is done and users are ready to download occurrence data from GBIF and OBIS APIs.

```
create_folders()

prepare_dataset(filename_inputData,column_scientificName,
                column_taxonName,column_location,column_eventDate,
                file_name_extension)

n_accounts <- 7

user <- "ekinhanno1"
pwd <- "******"
email <- "ekinhanno1@outlook.com"
```

##Step 1. Obtaining Occurrence Data

In this step, users can begin downloading the occurrence data for the desired species from GBIF and OBIS databases. Since the users provided the necessary information such as their dataset and account information, all they have to do is running the lines for executing "send_GBIF_request", "get_GBIF_download", "extract_GBIF_columns", and "get_OBIS_records" functions.

The "send_GBIF_request" function was created to prepare and send a JSON file with the records of species from the dataset to the GBIF API, which then lets the GBIF API prepare the files that contain occurrence data to be downloaded. This function first standardizes all species names through the "name_backbone" function from the "rgbif" package so that all synonyms or misspellings can be determined and then eliminated. Furthermore, the number of occurrence records for every species is determined through the "occ_count" function from the "rgbif" package. After this, according to the information that the user provided, the species are chunked nearly equally according to the number of occurrence records and the desired number of chunks (which is specified by the user in the preparation step). In the last step of this function, a JSON file for every chunk is created and sent to the GBIF API.

The "get_GBIF_download" function was created in order to download the files that contain occurrence data from GBIF.org automatically. Users must take into account three important subjects before running this function: 1) After running the "send_GBIF_request" function, users may need to wait between 15 minutes and 3 hours so that GBIF API can process the request and prepare the download files; 2) Users can check if the downloads are ready by logging to their GBIF.org account, and even download the occurrence data from the website. But, it must be noted that downloading the data manually from the website may cause inconveniences for the workflow; 3) If the directory for downloading GBIF occurrence data ("path_to_GBIFdownloads") is not empty, some problems may occur within the workflow. In this case, users may need to change the option in the function overwrite to "TRUE", allowing the function to overwrite the files in the directory. It should be noted that the files that are overwritten can not be retrieved later. This function utilizes the "occ_download_get" function from "rgbif" package to accomplish its task.

"extract_GBIF_columns" decompresses the zip files that were downloaded through "get_GBIF_download", eliminates fossil occurrence records, removes wrong (out of bound coordinates) or empty coordinates, and compiles the output in chunks. By running this function, the users are done with downloading occurrence data from the GBIF database and can move on to the OBIS phase.

The "get_OBIS_records" function is the only function that is required to download the necessary occurrence data from the OBIS database. This function utilizes the "occurrence" function from the "robis" package to download the occurrence data from the OBIS database. Furthermore, this function provides a couple of options to the users; these options include fields to download from the OBIS database ("fieldsobis"), removing fossil records (remove.fossils), and removing records with missing important data (omit.rows).

"get_OBIS_records" only downloads eight columns out of 114 ("scientificName","scientificNameID", "decimalLongitude","decimalLatitude", "basisOfRecord", "country", "speciesid", "marine") by default from OBIS API in order to decrease the clutter, but this can be arranged through changing "fieldobis" object according to the users needs. This function also removes fossil records by default; setting the "remove.fossils" object to

"FALSE" prevents that from happening. "omit.rows" object can also be changed according to users' needs if they want to remove some records based on missing information.

```
send_GBIF_request(file_name_extension,path_to_GBIFdownloads,
                  n_accounts,user=user,pwd=pwd,email=email)

get_GBIF_download(path_to_GBIFdownloads,file_name_extension)

extract_GBIF_columns(path_to_GBIFdownloads,file_name_extension)

get_OBIS_records(path_to_OBISdownloads,file_name_extension)
```

##Step 2. Cleaning Occurrence Data

As it is contemplated in our paper, the degree of quality of the data that is obtained from GBIF and OBIS may raise some concerns; in our workflow, these concerns are addressed in this step. "clean_GBIF_records" and "clean_OBIS_records" functions utilize the "CoordinateCleaner" package that is designed to clean wrong, imprecise, outlying and empty coordinates based on the information that is obtained from GBIF and OBIS databases. Both of these functions provide the option to thin the records. Since high numbers of records may cause memory issues, the number of records can be reduced by setting "thin_records" to TRUE (for both functions). Then, coordinates are rounded to the second digit, and duplicates are removed. For the single remaining record at this site, the original (not the rounded) record is kept. Thinning may result in imprecise results when the regions considered later in the workflow are small.

```
clean_GBIF_records(path_to_GBIFdownloads,file_name_extension,thin_records=TRUE)

clean_OBIS_records(path_to_OBISdownloads,file_name_extension,thin_records=FALSE)
```

##Step 3. Determining Alien Occurrences

In the last step, DASCO workflow utilizes two functions, "coords_to_regions_GBIF" and "coords_to_regions_OBIS" to accomplish the task of determining alien occurrences and produce a definitive last output. In order to utilize the coordinates, the "sf" package was used. As stated before, users can utilize the default shapefile that is presented in the DASCO workflow.

Both functions start with cross-checking if the habitat information for species on the WORMS database is matching with the coordinates from GBIF and OBIS databases. Then, utilizing the region information from the dataset that users provided, DASCO workflow identifies which downloaded coordinates match the regions from the dataset and keeps the occurrence records that are thought to be alien. After running both functions, users can execute the last function called "add_first_records" to compile the alien occurrences and produce the last output file named "AlienRegions_FinalDB_(file_name_extension)", into the "./Data/Output" directory, which contains all the alien occurrences from the dataset that is provided.

```
realm_extension <- TRUE

coords_to_regions_GBIF(name_of_shapefile,
                       path_to_GBIFdownloads,
                       realm_extension,
                       file_name_extension)

coords_to_regions_OBIS(name_of_shapefile,
                       path_to_OBISdownloads,
                       realm_extension,
                       file_name_extension)


dat <- add_first_records(file_name_extension,path_to_GBIFdownloads)
```