

DASCO Manual

Ekin Kaplan, Hanno Seebens

26 October 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 Ocean Biodiversity Information System (OBIS). For each taxon listed in the checklist, occurrences 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 recommended 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”, “woRMS”, “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.packages("rfishbase")
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

Executing the data sets requires having four data sets, which have to be stored in the folder Data/Input/:

1. Taxon-by-region list: The user has to provide a data set, which contains the records of the occurrences of taxa in a region. At minimum, the data set should consist of two columns for the taxon names and the name of region, where it has been recorded. In addition, the year of first record (called ‘event date’) of the alien population can be provided. Ideally, the identity of the taxa is be provided in two columns, one with the full scientific name including the authority and another one with the taxon name without the authority. The reason for this is that GBIF and OBIS require different input variables to match the respective taxonomies. If information about the authority is not available, taxon names can be provided without for both GBIF and OBIS although this decreases the chance to find a match of taxon names in GBIF. The geographic regions can be any combination of regions as long as these are represented in a shapefile (see below). The file format of the taxon-by-region file should be csv.
2. Shapefile: The workflow requires a spatial representation of the boundaries of all regions as a shapefile. The regions and the region names should match the selection of regions in the taxon-by-region file. A

default shapefile based on 517 marine and terrestrial regions worldwide is provided with the workflow in the folder Data/Input/Shapefiles.

3. AllLocations: This spreadsheet table contains a list of all regions considered in the workflow. In the file, provided together with the workflow, further details are provided for each region such as the continent or ISO codes, which are not required to run the workflow. Information in the column 'keywords' provides a way to match alternative names of the regions. If a name was found in this column, it will be changed to the name provided in the column 'Location'.
4. Neighbouring regions: This data set provides information about neighbouring terrestrial and marine regions. This information is required to assign marine species, which are listed in the regional checklist, to the respective marine ecoregion bordering this terrestrial region. For example, if the checklist of Turkey contains marine species and occurrence records of these species were obtained from OBIS or GBIF in the marine ecoregions bordering Turkey, the populations of these species were set as being alien in these marine ecoregions. If new regions are added to the workflow, this file should be updated accordingly.

Executing the DASCO workflow

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

Before running the workflow all scripts need to be loaded:

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

Step 0: Configuring the workflow

Global variables

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 <- "Taxon-by-region-file.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 folders 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 taxon-by-region data set, which contains the lists of taxon names and regions:

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 polygons 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 other input files (see ‘Data sets’ above).

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 ensure that previous output files are not overwritten.

GBIF accounts

Records from GBIF are obtained through the GBIF API, which requires having an account on GBIF (GBIF.org) and providing the account details here. As requested GBIF data can quickly reach millions of records and several GB of data, it is possible to split the request into chunks of manageable sizes. A single GBIF account can handle three requests simultaneously, and thus having `n` accounts allows splitting the requests into `n * 3` chunks. The number of accounts can be specified in the variable `n_accounts` (default set to 1). The variables `user`, `pwd`, and `email` provides the information of the GBIF accounts. If the user wants to chunk their request 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 <- "username1"`, `email <- "username1@address.com"`. The number will be automatically increased within the workflow. For instance, creating seven accounts will allow sending 21 requests in parallel. This requires having seven accounts, which should be named `username1`, `username2`, ... and `username1@address.com`, `username2@address.com`. While the username has to be unique, the email address could be universal to all accounts, which, however, requires an adjustment in the file `send_GBIF_request.R` by replacing `email <- gsub("1",x,email_base)` with `email <- email_base`. The variable `pwd` specifies the GBIF account password, which must be the same for every account as a default.

After providing the required information, the workflow can be executed.

Step 1: Check and create folder structure

As a first step of the workflow, the function `create_folders` checks if the necessary folder structure is provided and - if not - creates the correct folder structure. The input data sets have to be provided in the respective subfolders in `Data/Input/` (see section `Data sets` above)

The following function `prepare_dataset.R` loads the taxon-by-region data set, standardises the variable names and exports a standardised taxon-by-region file together with a taxon names.

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

Step 2: Obtaining occurrence data

In this step, species occurrence data are obtained from GBIF and OBIS. As both repositories require different input variables and provide different access, the process of obtaining data differs as well.

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

This function first standardises all taxon names through the `name_backbone` function from the `rgbif` package so that synonyms and misspellings can be determined. Only records for matched taxon names can be downloaded, which means that non-matching taxon names have to be removed from further processing. The taxon names provided by GBIF are further used within the workflow. These names are stored together with the GBIF keys and the original names in the output file `GBIF_SpeciesKeys_...` in the folder `Data/Output/`.

For each taxon, the number of occurrence records on GBIF.org is determined using the `occ_count` function from the `rgbif` package. This information allows splitting the data into n ($n = n_accounts * 3$) chunks of similar size as specified by the user. Each of the n chunks of the taxon list is then sent as a separate request to the GBIF API. Processing the requested data by GBIF may take some time depending on the amount of requested data. The R package `rgbif` provides a number of functions to handle and check the process of requests.

```
get_GBIF_download(path_to_GBIFdownloads,file_name_extension)
```

The `get_GBIF_download` function was created to download the files that contain occurrence data from GBIF.org automatically but three things need to be considered: 1 After running the `send_GBIF_request` function, users may need to wait up to several hours until GBIF has processed the request and prepared 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 directly from the website, but then the data needs to be stored in the correct folder as determined in the variable `path_to_GBIFdownloads`. 3 To prevent overwriting already stored files, the default option is set to `overwrite=FALSE`, which need to be changed if files are downloaded multiple times. This function utilizes the “`occ_download_get`” function from “`rgbif`” package to accomplish its task.

```
extract_GBIF_columns(path_to_GBIFdownloads,file_name_extension)
```

The function `extract_GBIF_columns` decompresses the zip files that were downloaded from GBIF and conducts an initial cleaning by removing fossil occurrence records, duplicated and missing records and records with coordinates not being longitudes or latitudes.

```
get_OBIS_records(path_to_OBISdownloads,file_name_extension)
```

The `get_OBIS_records` function is the only function that is required to download the necessary occurrence data from the OBIS website. This function uses the `occurrence` function from the `robis` package to perform the download. Three options are provided to specify the download. As the default, the following variables are obtained from OBIS for each record as specified in `fieldsobis`: `scientificName`, `scientificNameID`, `decimalLongitude`, `decimalLatitude`, `basisOfRecord`, `country`, `speciesid`, `marine`. Fossil records and records with missing entries in any of the entries for `species`, `decimalLongitude`, `decimalLatitude`, `scientificName` are removed.

The `get_habitats` function allows users to obtain the habitat information of list of species from WoRMS, Sealifebase and Fishbase databases. This function utilizes the species keys that are obtained from GBIF and OBIS to prevent false species names and synonyms to be processed. After the parsing, the function downloads the habitat information from WoRMS and Fishbase databases. Sealifebase database is already in the ‘/Data’ section, but if one decides to use the latest version sealifebase, they can change the ‘`update_sealifebase`’ object to ‘True’, thus updating the database. After acquiring the habitat information, the function processes this information into four columns, namely “terrestrial”, “marine”, “freshwater”, “brackish”; then this file is saved into the “Data/Output” folder for further use.

Step 3: Cleaning Occurrence Data

```
clean_GBIF_records(path_to_GBIFdownloads,file_name_extension,thin_records=TRUE)
clean_OBIS_records(path_to_OBISdownloads,file_name_extension,thin_records=FALSE)
```

Occurrence records obtained from GBIF and OBIS may require serious cleaning of records due to false or imprecise records. After initial cleaning within step 2 of the workflow, here coordinates with low precision of having less than two digits after the comma are removed. In addition, a series of tests as provided in the package `CoordinateCleaner` (Zizka et al. 2020) are performed. The selection of tests can be specified in the option `tests_for_cleaning` and include the following tests as a default: `capitals`, `centroids`, `equal`, `gbif`, `institutions`, `outliers`, `zeros`. These tests identify apparently wrong entries such as records found at the coordinates of the providing institute or country, or outlying records based on the geographic outlier test as described in Zizka et al. (2020).

Both of these functions for GBIF and OBIS provide the option to thin records by removing close coordinates. Particularly for GBIF, the number of records may be very high, which may cause memory issues although the data are already processed in chunks of data as specified by the user. In addition, depending on the resolution of the regions provided in the shapefile, a high resolution is not required and thinning enables accelerating the process without losing much information. Thinning is enabled for GBIF as a default (`'thin_records=TRUE'`). 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. ## Step 4: Determining alien populations

```
coords_to_regions_GBIF(name_of_shapefile,
                        path_to_GBIFdownloads,
                        realm_extension=TRUE,
                        file_name_extension)
coords_to_regions_OBIS(name_of_shapefile,
                       path_to_OBISdownloads,
                       realm_extension=TRUE,
                       file_name_extension)
```

In the fourth step of the workflow, the status of being native or alien is determined for each record obtained from GBIF or OBIS. This is done by identifying the region, where the respective coordinate is located, and extracting the status of the population from the original taxon-by-region file. A population is considered as being present in a particular region only if at least three records fall into the respective polygon. For instance, if a species has been recorded as being alien for Turkey and at least three records are listed for Turkey, all records found within Turkey for that species are taken as records of alien populations. These records can then be assigned to the regions as specified in shapefile `name_of_shapefile`. In this way, records can be assigned to e.g. sub-regions of Turkey or biogeographical areas extending beyond that country. Coordinates of population not identified as being alien populations are removed.

The DASCO workflow allows assigning taxa from regional checklists to marine regions if `realm_extension=TRUE`. As marine taxa are often not specified as such, information from the World Register of Marine Species (WoRMS) is used to identify marine species. This is done by using the function `get_WoRMS_habitats` from the package `worms`, which provides information about the habitat (marine, freshwater and terrestrial) of species listed on WoRMS. In addition, to avoid mis-classifications taxa of the following groups were considered as non-marine: classes of `insecta`, `arachnida`, `aves`, `amphibia` and `mammalia`, and `pyla` of `tracheophyta`, `anthocerotophyta` and `bryophyta`. In this way, also freshwater species were classified but as this classification depends on WoRMS, the list of freshwater species is certainly incomplete.

For marine taxa, the marine ecoregions bordering a certain terrestrial region are obtained from the file `RegionsMEOW_NeighbourList.csv`. Taxa identified as marine species and with available records in the respective marine ecoregions are then classified as being alien in the respective marine ecoregion.

Step 5: Final output

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

As a last step of the DASCO workflow, the function `final_DASCO_output` merges all files from GBIF and OBIS. If provided, first records were harmonised such that the earliest first record provided for a certain region is kept. This is done under the assumption that the first record of a taxon reported for a region also holds to the regions assigned by the workflow.

Finally, the output files are prepared and stored as `AlienRegions_FinalDB_(file_name_extension)` in the `Data/Output/` directory.