# Spatial R course - 4

*Tobias Andermann*

*5/25/2020*

## Accessing biodiveristy data through web services (IUCN)

In this tutorial we will work with another r-package that can be used to download biodiversity data: The IUCN redlist package `rredlist`.

Dependencies:

```
library(rredlist)
library(rworldmap)
library(taxize)
```

The IUCN RedList is a collective work of many researchers and conservationists, which compile information about species population trends, habitats, and evaluations of all applicable threats. This information is used to assign a protection status to each species. You can find more information about the IUCN RedList here.

### 1. Get your API token

In order to use this package you first need to apply for an API token. You can do that by filling out this form here.

Since it is hard to predict how long it will take for IUCN to approve your token request (could be minutes or hours), you can get a key from me, just approach me if you're getting to this part of the tutorial and haven't received your token yet.

Then store the API token as a string:

```
iucn_api = 'paste_your_iucn_token_here'
```

**Extra info:** If you're interested you can store this key in your R environment. It's not necessary for this exercise but if you want you can check out the documentation of the functions `rl_use_iucn()` and `edit_r_environ()` by typing `?` followed by the function name.

### 2. Pick a taxon and find synonyms

Now you can again pick a species of your choice for which you want to extract species data (such as conservation status, distribution, etc), in this example we're working with the tiger Panthera tigris.

```
taxon_name <- "Panthera tigris" # tiger
```

We can check for synonyms of our species name, using the rl_synonyms() function:

```
rl_synonyms(taxon_name,key=iucn_api)
```

```
## $name
## [1] "Panthera tigris"
##
## $count
## [1] 1
##
## $result
```

```
##   accepted_id   accepted_name        authority      synonym  syn_authority
## 1       15955 Panthera tigris (Linnaeus, 1758) Felis tigris Linnaeus, 1758
```

IUCN standardizes all its data to one main taxonomy, so you will not find any data associated with the synonyms outside of the accepted_name. Therefore make sure you continue working with the name listed in the `accepted_name` column.

Similarly as before in the GBIF tutorial, we can extract popular names (vernacular names) associated with our chosen taxon in the IUCN database:

```
rl_common_names(taxon_name,key=iucn_api)
```

```
## $name
## [1] "Panthera tigris"
##
## $result
##   taxonname primary language
## 1     Tiger    TRUE      eng
## 2     Tigre   FALSE      fre
```

### 3. Habitat information

The IUCN rredlist package offers several functions to extract information about the chosen taxon. You can check out the available functions and explanation in the official package documentation.

For example we can extract information about the habitats the species is found in using the `rl_habitats()` function.

```
rl_habitats(taxon_name,key=iucn_api)
```

```
## $name
## [1] "Panthera tigris"
##
## $result
##    code
## 1   1.1
## 2   1.4
## 3   1.5
## 4   1.6
## 5   1.7
## 6   1.9
## 7   3.5
## 8   3.6
## 9   4.5
## 10  4.6
##                                                                habitat
## 1                                                       Forest - Boreal
## 2                                                    Forest - Temperate
## 3                                         Forest - Subtropical/Tropical Dry
## 4                                  Forest - Subtropical/Tropical Moist Lowland
## 5  Forest - Subtropical/Tropical Mangrove Vegetation Above High Tide Level
## 6                                  Forest - Subtropical/Tropical Moist Montane
## 7                                        Shrubland - Subtropical/Tropical Dry
## 8                                     Shrubland - Subtropical/Tropical Moist
## 9                                        Grassland - Subtropical/Tropical Dry
## 10                 Grassland - Subtropical/Tropical Seasonally Wet/Flooded
```

```
##    suitability season majorimportance
## 1     Suitable     NA              Yes
## 2     Suitable     NA              Yes
## 3     Suitable     NA              Yes
## 4     Suitable     NA              Yes
## 5     Suitable     NA              Yes
## 6     Marginal     NA             <NA>
## 7     Suitable     NA              Yes
## 8     Suitable     NA              Yes
## 9     Marginal     NA             <NA>
## 10    Marginal     NA             <NA>
```

### 4. Threat status

One of the most interesting and unique information IUCN has to offer are the RedList assessments. Each species is classified into either one of the following categories:

- Least Concern (LC)
- Near threatened (NT)
- Vulnerable (VU)
- Endangered (EN)
- Critically endangered (CR)
- Extinct in the wild (EW)
- Extinct (EX)
- Data deficient (DD)

Let's see how our species is evaluated. We are using the rl_search() function which will return all sorts of information, but for now we're only interested in the category of the output:

```
threat_data = rl_search(taxon_name,key=iucn_api)
threat_data$result$category
```

```
## [1] "EN"
```

We can also view the history of IUCN assessments of our species and see if the trends have improved or worsened. Note that older evaluations may contain different categories, since IUCN has changed its' nomenclature several times in history:

```
rl_history(taxon_name,key=iucn_api)
```

```
## $name
## [1] "Panthera tigris"
##
## $result
##    year code   category
## 1  2015   EN Endangered
## 2  2011   EN Endangered
## 3  2010   EN Endangered
## 4  2008   EN Endangered
## 5  2002   EN Endangered
## 6  1996   EN Endangered
## 7  1994    E Endangered
## 8  1990    E Endangered
## 9  1988    E Endangered
## 10 1986    E Endangered
```

### 5. Extract geographic information

Unfortunately it is not possible to download range maps for your species via the package rredlist (at least not to my knowledge, IUCN is a bit particular about making their data available via programming interfaces). However, IUCN has range maps for most species in their database stored on their server. Precise range maps can instead be downloaded manually via the webpage, either for individual species, as we did in yesterdays tutorial, or in bulk for whole groups of taxa, e.g. all mammals, via this link.

Even though we can't get the actual range datavia the rredlist package, it at least offers a function `rl_occ_country()` which can be used to extract a list of countries where the species exists.

```
occurrence_countries = rl_occ_country(taxon_name,key=iucn_api)
occurrence_countries
```

```
## $name
## [1] "Panthera tigris"
##
## $count
## [1] 24
##
## $result
##    code                                country          presence origin
## 1    AF                            Afghanistan Extinct Post-1500 Native
## 2    BD                             Bangladesh            Extant Native
## 3    BT                                 Bhutan            Extant Native
## 4    CN                                  China            Extant Native
## 5    ID                              Indonesia            Extant Native
## 6    IN                                  India            Extant Native
## 7    IR              Iran, Islamic Republic of Extinct Post-1500 Native
## 8    KG                            Kyrgyzstan Extinct Post-1500 Native
## 9    KH                              Cambodia  Possibly Extinct Native
## 10   KP Korea, Democratic People's Republic of  Possibly Extinct Native
## 11   KZ                            Kazakhstan Extinct Post-1500 Native
## 12   LA       Lao People's Democratic Republic            Extant Native
## 13   MM                                Myanmar            Extant Native
## 14   MY                               Malaysia            Extant Native
## 15   NP                                  Nepal            Extant Native
## 16   PK                               Pakistan Extinct Post-1500 Native
## 17   RU                     Russian Federation            Extant Native
## 18   SG                              Singapore Extinct Post-1500 Native
## 19   TH                               Thailand            Extant Native
## 20   TJ                             Tajikistan Extinct Post-1500 Native
## 21   TM                           Turkmenistan Extinct Post-1500 Native
## 22   TR                                 Turkey Extinct Post-1500 Native
## 23   UZ                             Uzbekistan Extinct Post-1500 Native
## 24   VN                               Viet Nam  Possibly Extinct Native
##     distribution_code
## 1  Regionally Extinct
## 2              Native
## 3              Native
## 4              Native
## 5              Native
## 6              Native
## 7  Regionally Extinct
## 8  Regionally Extinct
```

```
## 9    Possibly Extinct
## 10   Possibly Extinct
## 11 Regionally Extinct
## 12             Native
## 13             Native
## 14             Native
## 15             Native
## 16 Regionally Extinct
## 17             Native
## 18 Regionally Extinct
## 19             Native
## 20 Regionally Extinct
## 21 Regionally Extinct
## 22 Regionally Extinct
## 23 Regionally Extinct
## 24   Possibly Extinct
```

As you can see, the output also contains the information if the species is extant or extinct in the country where it was once found. In the next step we use this information to plot a world map with countries highlighted where our species has been found, with different coloring depending on if the species is extinct or extant in the respective country.

### 6. Plotting geographic information

First let's turn the `presence` column, which contains the info if the taxon is extinct/extant in each country, into an array of 0 (extinct) and 1 (extant), in order to use this information for plotting:

```
extant_extinct = occurrence_countries$result$presence
extant_extinct[extant_extinct != "Extant"] <- 0
extant_extinct[extant_extinct == "Extant"] <- 1
extant_extinct
```

```
##  [1] "0" "1" "1" "1" "1" "1" "0" "0" "0" "0" "0" "1" "1" "1" "1" "0" "1"
## [18] "0" "1" "0" "0" "0" "0" "0"
```

Now let's extract the list of countries where our species has been found:

```
theCountries <- c(occurrence_countries$result$code) # ISO2 country codes
theCountries
```

```
##  [1] "AF" "BD" "BT" "CN" "ID" "IN" "IR" "KG" "KH" "KP" "KZ" "LA" "MM" "MY"
## [15] "NP" "PK" "RU" "SG" "TH" "TJ" "TM" "TR" "UZ" "VN"
```

Now we merge the list with our extinct/extant info with the list of the country names into one dataframe:

```
extDF <- data.frame(country = c(occurrence_countries$result$code), extant = c(extant_extinct))
extDF
```

```
##    country extant
## 1       AF      0
## 2       BD      1
## 3       BT      1
## 4       CN      1
## 5       ID      1
## 6       IN      1
## 7       IR      0
## 8       KG      0
```

```
## 9       KH      0
## 10      KP      0
## 11      KZ      0
## 12      LA      1
## 13      MM      1
## 14      MY      1
## 15      NP      1
## 16      PK      0
## 17      RU      1
## 18      SG      0
## 19      TH      1
## 20      TJ      0
## 21      TM      0
## 22      TR      0
## 23      UZ      0
## 24      VN      0
```

Now we will use the `rworldmap` package, which provides a function that allows us to find the countries on the world map based on their ISO2 codes (the 2-letter abbrevations in our country list):

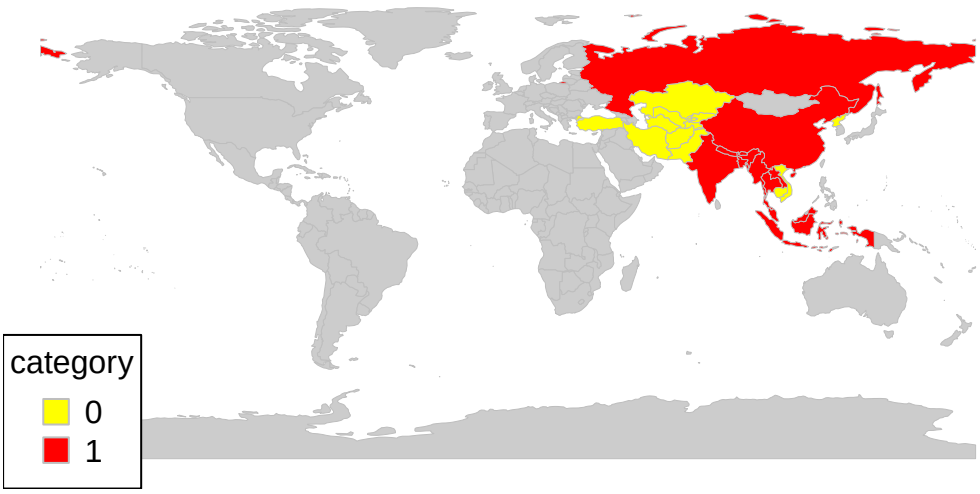
```
library(rworldmap)
extMap <- joinCountryData2Map(extDF, joinCode = "ISO2", nameJoinColumn = "country")
```

```
## 24 codes from your data successfully matched countries in the map
## 0 codes from your data failed to match with a country code in the map
## 219 codes from the map weren't represented in your data
```

This will join your data from extDF the dataframe with the country map polygons to produce a `SpatialPolygonsDataFrame` object.

Finally we can plot the map with the countries of presence of our species highlighted in red and the countries where the species went extinct in yellow. We'll use the `mapCountryData()` function from the rworldmap package to plot all countries and color them by the values in the target column of our dataframe, specficied with the `nameColumnToPlot=` setting.

```
mapCountryData(extMap, nameColumnToPlot="extant", catMethod = "categorical",
  missingCountryCol = gray(.8))
```



## Resolving taxon names (taxize)

Often in biology we are confronted with taxon names or lists of taxon names, and we need to somehow retrieve data for these taxa from public databases (as we did in the previous tutorial steps). The challenge here is usually to find the correct species name, since taxonomies can vary between databases and between different authors. Luckily there are great tools to standardize and resolve taxonomic issues in many cases, implemented in the `taxize` package.

The `taxize` package provides access to taxonomic information sourced from many databases, among them the Global Names Resolver (GNR) service provided by the Encyclopedia of Life. The advantage of `taxize` is that it is primarily designed for resolving taxonomic issues and thus has many useful functions for this purpose, compared to some similar but more simplified functionalities of the `rgbif` and `rredlist` packages we have seen in the previous tutorial steps.

Let's first load the package.

```
library(taxize)
```

### 7. Finding correct taxon name for database

Let's say we have a list of species names and we want to know if our species are spelled correctly.

Here, we are searching for two **misspelled species names**:

```
temp <- gnr_resolve(names = c("Pantera tigrsi", "Homo saapiens"))
temp
```

```
## # A tibble: 56 x 5
##    user_supplied_na~ submitted_name matched_name  data_source_title   score
##  * <chr>             <chr>          <chr>         <chr>               <dbl>
##  1 Pantera tigrsi    Pantera tigrsi Panthera tig~ Catalogue of Life    0.75
##  2 Pantera tigrsi    Pantera tigrsi Panthera tig~ ITIS                 0.75
##  3 Pantera tigrsi    Pantera tigrsi Panthera tig~ NCBI                 0.75
##  4 Pantera tigrsi    Pantera tigrsi Panthera tig~ Union 4              0.75
##  5 Pantera tigrsi    Pantera tigrsi Panthera tig~ The Interim Regist~  0.75
##  6 Pantera tigrsi    Pantera tigrsi Panthera tig~ Freebase             0.75
##  7 Pantera tigrsi    Pantera tigrsi Panthera tig~ GBIF Backbone Taxo~  0.75
##  8 Pantera tigrsi    Pantera tigrsi Panthera tig~ Encyclopedia of Li~  0.75
##  9 Pantera tigrsi    Pantera tigrsi Panthera tig~ TaxonConcept         0.75
## 10 Pantera tigrsi    Pantera tigrsi Panthera tig~ AnAge                0.75
## # ... with 46 more rows
```

In the output you can find the database you want to download data from and use the according correct species name (can vary between databases).

### 8. Getting species list for higher taxa

Let's say we have a taxonomic family name and want to find all species belonging to this family, for example all dogs of the family Canidae.

A number of data sources in taxize provide the capability to retrieve higher taxonomic names,for example the Catalogue of Life (COL) taxonomy (`db = "col"`). We can search the taxonomy for taxa belonging to our specified group using the `downstream()` function.

```
species_output = downstream("Canidae", downto = "Species", db = "col")
```

```
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
## Error : Too Many Requests (HTTP 429)
```

```
species_output
```

```
## $Canidae
##                        childtaxa_id      childtaxa_name childtaxa_rank
## 1  5f3ddd0e635c5fb11690722f002dbdfa Atelocynus microtis        species
## 2  36807bbc8e4efedbea119a1144c0dae3       Canis adustus        species
## 3  8b403291f26a2481f50229cbe2b66a9c        Canis anthus        species
## 4  2d360d349c645ad93fd960ae988101e9        Canis aureus        species
## 5  04afba115065ca57c2ea8f64fe876579       Canis latrans        species
## 6  519b057a22d3439f6fe5c9a1d8be84b8      Canis lupaster        species
## 7  bcd6035778291a7feaad52cb7ac167cb         Canis lupus        species
## 8  3f5e32547d4f78fbd3e3dd4755a27166     Canis mesomelas        species
## 9  e2c274e049e7a12fa066f96028092720      Canis simensis        species
## 10 9491363ed5ec89660e88360f5b26371d     Cerdocyon thous        species
##    childtaxa_extinct
## 1              FALSE
## 2              FALSE
## 3              FALSE
## 4              FALSE
## 5              FALSE
## 6              FALSE
## 7              FALSE
## 8              FALSE
## 9              FALSE
## 10             FALSE
##
## attr(,"class")
## [1] "downstream"
## attr(,"db")
## [1] "col"
```

You can now extract the list of all species belonging to your chosen family:

```
species_list = species_output$Canidae$childtaxa_name
species_list
```

```
##  [1] "Atelocynus microtis" "Canis adustus"       "Canis anthus"
##  [4] "Canis aureus"        "Canis latrans"       "Canis lupaster"
##  [7] "Canis lupus"         "Canis mesomelas"     "Canis simensis"
## [10] "Cerdocyon thous"
```

## Assignment 1

Now where you have a basic overview over the use of the `rgbif`, `rredlist`, and `taxize` packages, you are ready to approach a bioinformatic task based on biodiversity data:

Your supervisor asks you for help with a project about the cat family `Felidae`. Your task is to **create a map of global occurrences** of this family, **colored by species**. Further your supervisor asks you to retrieve a **list of IUCN threat statuses for all species of this family** (if you like a challenge, you can also create a second plot, colored by threat status instead of colored by species).

Since these data are supposed to be used in a publication, your supervisor expects you as a properly trained biodiversity data wizard to provide a DOI assigned dataset of `Felidae` occurrence records to be cited in the study.

**Tips for assignment:**

There are different ways of solving this task, one approach could be:

- create a download request containing all records assigned to the taxon `Felidae`, using the `occ_download()` function
- download the data and load into R (step 8 in the first tutorial)
- export the DOI reference of the data download (step 8 in the first tutorial)
- plot occurrences colored by name (step 8 in the first tutorial)
- retrieve a species list of the family `Felidae`, using the `taxize` package
- get the IUCN RedList status for each species, using the `rl_search()` function

## Assignment 2

For the very fast and motivated people among you:

You are being asked to plot the actual range maps of all Felidae species (according to IUCN) and on top plot all point occurrences (according to GBIF). For this purpose you need to download the actual range data, which can be downloaded for each species individually or as a big data-package, containing ranges of all mammal species (download here), and then you can extract the ranges of the species of interest. You can decide if you want to plot this all in one plot, or in separate plots, one for each species (doesn't have to be all Felidae species, but maybe a few selected ones).