**Code to calculate general and disaggregated Red List Indices (RLIs)**

Version 2.0. Instructions and recommendations

Maria Dias, [maria.dias@birdlife. org](mailto:maria.dias@birdlife.%20org)

Ash Simkins, [ashley.simkins@birdlife.org](mailto:ashley.simkins@birdlife.org)

February 2020

* The codes mentioned in this report are an update of the codes created in 2016 (project P90001: *Indicator-related work to a) code to calculate national Red List Indices (RLIs), and b) calculate and provide updated data on the coverage by protected areas of marine, terrestrial and freshwater Key Biodiversity Areas*).
* The current instructions relate only to the codes to calculate the Red List Index (hereafter RLI), the codes to estimate the proportion of each species’ distribution in each country are in Annex 4. You must either have the output of this code, or run the code, before running the codes to calculate the Red List Index.
* The codes were modified in order to simplify some of the steps, to add notes explaining the logic behind each step, and to incorporate the possibility of calculating “thematic” RLIs (i.e., RLIs for a certain group of species, drivers of genuine changes, or a combination of these). A description of the major modifications to the code in relation to the previous code is provided in section D.
* The codes calculate Global RLI, Subglobal RLIs and Thematic RLIs (see definitions below).
* The codes are written in R language and stored at [<https://github.com/BirdLifeInternational/rli-codes>].
* It is strongly recommended to read the description and scientific basis for the calculation of the Red List Index (Butchart et al. 2004[[1]](#footnote-1), 2005[[2]](#footnote-2), 2007[[3]](#footnote-3), 2010[[4]](#footnote-4); Rodrigues et al. 2014[[5]](#footnote-5)), and the metadata document for SDG Indicator 15.5.1. (available at <https://unstats.un.org/sdgs/metadata/files/Metadata-15-05-01.pdf>).
* Throughout the text below, references to specific commands in the codes are noted in **blue**; numbered sections of the code are noted with a “#”.

**Contents**

[A. Definitions 4](#_Toc36110425)

[B. Instructions to run the codes 6](#_Toc36110426)

[B.1. Instructions to run any type of RLIs 6](#_Toc36110427)

[B.2. Instructions to run General RLIs: 6](#_Toc36110428)

[B.3. Instructions to run Subglobal RLIs: 6](#_Toc36110429)

[B.4. Instructions to run Thematic RLIs (both global and subglobal): 7](#_Toc36110430)

[B.5. Instructions to run all RLIs at the same time (general, subglobal and thematic): 8](#_Toc36110431)

[B.5. Outputs 9](#_Toc36110432)

[C. Structure of the codes 10](#_Toc36110433)

[C.1. Working directory and input tables 10](#_Toc36110434)

[C.2. Basic parameters to set 12](#_Toc36110435)

[C.3. Define disaggregations to run 13](#_Toc36110436)

[C.3.1. Define iso groups for subglobal RLIs to run 13](#_Toc36110437)

[C.3.2. Define thematic drivers RLIs to run 13](#_Toc36110438)

[C.3.3. Define thematic Sp\_Groups RLIs to run 14](#_Toc36110439)

[C.3.4. Define thematic Drivers - Sp\_Groups RLIs to run 15](#_Toc36110440)

[C.4. Check validity of input data and prepare tables 15](#_Toc36110441)

[C.5. Custom functions 17](#_Toc36110442)

[C.6. Prepare “back casting” tables and run global analyses 18](#_Toc36110443)

[C.7. Prepare run subglobal analyses 19](#_Toc36110444)

[C.8. Combine and save results following UN template 20](#_Toc36110445)

[D. Major changes from the previous version 20](#_Toc36110446)

[E. Common errors and Recommendations 22](#_Toc36110447)

[Annex 1: Structure of the input tables 24](#_Toc36110448)

[Annex 2: Structure of the output tables 27](#_Toc36110449)

[Annex 3: General and Disaggregated RLIs calculator; Version 2.0 29](#_Toc36110450)

[Annex 4: R code to estimate the proportion of each species’ distribution in each country 56](#_Toc36110451)

# A. Definitions

* “**General RLI**”: RLI for all species and drivers of genuine changes at a global level (see table 1);
* “**Disaggregated RLIs**”: Any subset of RLIs, geographical or thematic. The current version of the code is written to automatically calculate two types of disaggregated RLIs: “**Subglobal RLIs**” and “**Thematic RLIs**”;
* **“Subglobal RLIs”**: Disaggregated RLIs that take into account the proportion of each species’ distribution in each country/region (based on iso groups – see definition below); can be “**National RLIs**” or “**Regional RLIs**” (see table 1);
* **“Global RLIs”**: Non subglobal RLIs, i.e., RLIS not disaggregated per country or region; can be **General – global** RLIs or **Thematic- global** RLIs;
* “**Thematic RLI**”: Disaggregated RLIs for specific impacts or groups of species (see table 1); Thematic RLIs can be calculated at a global level or at a subglobal level;
  + **Thematic global RLI**: RLI for all species of a certain group (e.g. marine, freshwater) or genuine changes driven by a certain stressor (e.g. utilisation, invasive alien species)[[6]](#footnote-6), or a combination of these;
  + **Thematic subglobal RLIs**: based on the same input data as previous, but take into account the proportion of each species’ distribution in each country/region (see above “Subglobal RLIs”);
  + **Thematic Drivers RLI**: thematic RLIs related to stressors; uses only genuine changes driven by a certain stressor (e.g. impacts of utilisation, impacts of invasive alien species)1; can be global or subglobal;
  + **Thematic Sp\_Groups RLI:** thematic RLIs for certain groups of species (e.g. internationally traded species, marine species); can be global or subglobal; can be global or subglobal;
  + **Thematic Drivers - Sp\_Groups RLI:** combination of the previous two; thematic RLIs related to stressors driving trends in a certain groups of species (e.g. impacts of utilisation on migratory species, impacts of invasive alien species on wetland species);
* **“Iso groups”**: Groups of countries or regions based on ISO3 codes (e.g. “ISO\_BL", "ISO\_SDG", "SDG\_Region", "LDC", "IPBES\_region"), used to run the subglobal RLIs;

Table 1: List of Red List Index (RLIs) (general and thematic, global and subglobal) that can be calculated with the version 2.0 of the code.

|  |  |  |  |
| --- | --- | --- | --- |
| **Main type** | **Sub types** | **Options** | **Description** |
| General | Global | **-** | RLI (individually for birds, mammals, amphibians, corals, cycads, and in aggregate)2 |
| (Disaggregated1)  Subglobal | National | ISO\_SDG | RLI - for each country, weighted by the proportion of each species’ range in each country matching ISOs used by the SDGs |
| ISO\_BL | RLI - for each country, weighted by the proportion of each species’ range in each country matching ISOs used in IBAT |
| Regional | SDG\_Region | RLI - for each SDG region, weighted by the proportion of each species’ range in each region |
| SDG\_Subregion | RLI - for each SDG subregion, weighted by the proportion of each species’ range in each subregion |
| LLDC\_SIDS | RLI - for LLDC and SIDS countries, weighted by the proportion of each species’ range in LLDC and SIDS countries |
| LDC | RLI - for LDC countries, weighted by the proportion of each species’ range in LDC countries |
| IPBES\_region | RLI - for each PBES region, weighted by the proportion of each species’ range in each region |
| CMS\_Region | RLI - for each CMS region, weighted by the proportion of each species’ range in each region |
| (Disaggregated1)  Thematic  all subtypes of thematic can be also Global or Subglobal | Drivers | utilisation | RLI - impacts of utilisation |
| IAS | RLI - impacts of invasive alien species |
| pollution | RLI - impacts of pollution |
| fisheries | RLI -impacts of fisheries |
| Sp\_Groups | internationallytraded | RLI - internationally traded species |
| medicine | RLI - species used for food and medicine |
| forest | RLI - forest-specialist species |
| wetland | RLI - wetland species |
| migratory | RLI - migratory species |
| marine | RLI - marine species |
| terrestrial | RLI - terrestrial species |
| freshwater | RLI - freshwater species |
| (user defined) | RLI - user defined group of species |
| Drivers - Sp\_Groups 3 | utilisation- migratory | RLI - impacts of utilisation on migratory species |
| fisheries – migratory | RLI - impacts of fisheries on migratory species |
| pollution - migratory | RLI - impacts of pollution on migratory species |
| IAS - migratory | RLI - impacts of invasive alien species on migratory species |
| IAS - wetland | RLI - impacts of invasive alien species on wetland species |

1 Disaggregated RLI can be both Subglobal or Thematic

2 All the RLI codes (General and Disaggregated) will run to all taxa and aggregate

3 Only the default examples are provided; the code is able to run any combination of group of species and drivers

Note that RLIs with trends based on small numbers of genuine changes may be easily misinterpreted and are probably not useful to end-users. As a general rule, RLIs based on fewer than 10 genuine changes over the time period should be treated with caution.

# B. Instructions to run the codes

* This chapter provides summarized instructions to run the codes for the different types of RLIs; It is strongly recommended the reading of the section C below (structure of the codes), in particular the subsections (C1 – C4);
* See also section E - Common errors and Recommendations;
* The code allows to run all RLIs at the same time (general and disaggregated; see table 1); however, it is possible to run just part of the codes by changing the settings (see table 2 and sections C2-C3).

## B.1. Instructions to run any type of RLIs

* 1. Define **working directory** and **location of output folders** (section C1);
  2. Provide names of **input tables** and **input folders** (section C1); it is fundamental to format the tables correctly, with the exact names of the columns as described in section C1 and Annex 1;
  3. Set basic parameters:
     + maximum year of the analyses (**maxyear**), e.g. 2020 (section C2);
     + number of iterations (**repetitions**); recommended value is ≥1000 (section C2);
     + if plots should be saved as pdf files (**pdfs=T**) (section B5 and C2);
     + if plots should be saved as png files (**pngs=T**) (section B5 and C2);
     + the standard deviation of the slope (**slopecv**) for the extrapolations in the linear models (see sections C2 and C5).

## B.2. Instructions to run General RLIs:

1. See B.1.;
2. set **saveglobal=T** (section C.2);
3. set **only\_thematic=F** (section C.2);
4. To run only General RLI:
   * set **skip\_subglobal=T** (section C.2);
   * set **skip\_thematic=T** (section C.2);

## B.3. Instructions to run Subglobal RLIs:

1. See B.1. ;
2. set **skip\_subglobal=F** (section C.2);
3. set **only\_thematic=F** (section C.2);
4. set **isos2use=** **c("ISO\_BL", "ISO\_SDG", "SDG\_Region", "SDG\_Subregion", "LLDC\_SIDS", "LDC", "IPBES\_region", "CMS\_Region")** or any combination of these (section C.3.1); important note: if to run both "ISO\_BL" and "ISO\_SDG", set “ISO\_BL” first in the list;
5. To save only Subglobal RLI:
   * set **saveglobal=F[[7]](#footnote-7)** (section C.2);
   * set **skip\_thematic=T** (section C.2);

## B.4. Instructions to run Thematic RLIs (both global and subglobal):

1. See B.1.;
2. Set **skip\_thematic=F** (section C.2);
3. Set **saveglobal=T** (section C.2);
4. To run Thematic Drivers, set **driv\_gc=list(utilisation=utilisation, IAS=IAS, pollution=pollution, fisheries=fisheries)** or a sub-list of these (section C.3.2);
5. To run Thematic Sp\_Groups based on default lists:
   * set **splists=c("internationallytraded", "medicine", "forest", "marine", "terrestrial", "freshwater")**  or a subset of these (section C.3.3);
   * set **other\_species\_lists=F**;
6. To run Thematic Sp\_Groups based on user-defined lists, set **other\_species\_lists=T** (section C.3.3), and provide correspondent input table (section C.1);
7. To run Thematic Drivers - Sp\_Groups, set **combs = data.frame (driv= c( "utilisation", "fisheries", "pollution", "IAS", "IAS"), spgroup = c( "migratory", "migratory", "migratory", "migratory","wetland"))** or another data frame with the required combination of species and drivers (section C.3.4); these species and drivers need also to be defined in **driv\_gc** and **splists**;
8. To run only Thematic Global:

* see previous settings B.4. 2-7;
* set **dis\_subglobal =F** (section C.2);

1. To run only Thematic Subglobal**[[8]](#footnote-8)**:

* see previous settings B.4. 3-7;
* set **saveglobal=F** (section C.2);
* set **skip\_subglobal=F** (section C.2);
* set **dis\_subglobal =T** (section C.2);
* set **isos2use=** **c("ISO\_BL", "ISO\_SDG", "SDG\_Region", "SDG\_Subregion", "LLDC\_SIDS", "LDC", "IPBES\_region", "CMS\_Region")** or any combination of these (section C.3.1);

1. To run both Thematic Global and Subglobal:

* see previous settings (B.4. 2-7);
* set **saveglobal=T** (section C.2);
* set **skip\_subglobal=F** (section C.2);
* set **dis\_subglobal =T** (section C.2);
* set **isos2use=** **c("ISO\_BL", "ISO\_SDG", "SDG\_Region", "SDG\_Subregion", "LLDC\_SIDS", "LDC", "IPBES\_region", "CMS\_Region")** or any combination of these (section C.3.1);

1. To run only Thematic RLIs (i.e., no General), set **only\_thematic =T** (section C.2).

## B.5. Instructions to run all RLIs at the same time (general, subglobal and thematic):

1. See B.1.;
2. set **saveglobal=T** (section C.2);
3. set **skip\_subglobal=F** (section C.2);
4. set **skip\_thematic=F** (section C.2);
5. set **only\_thematic=F** (section C.2);
6. set **dis\_subglobal=T** (section C.2);
7. set **isos2use=** **c("ISO\_BL", "ISO\_SDG", "SDG\_Region", "SDG\_Subregion", "LLDC\_SIDS", "LDC", "IPBES\_region", "CMS\_Region")** or any combination of these (section C.3.1);
8. set **driv\_gc=list(utilisation=utilisation, IAS=IAS, pollution=pollution, fisheries=fisheries)** or a sub-list of these (section C.3.2);
9. set **splists=c("internationallytraded", "medicine", "forest", "marine", "terrestrial", "freshwater")**  or a subset of these (section C.3.3); for user-defined lists, set **other\_species\_lists=T** (section C.3.3), and provide correspondent input table (section C.1);
10. set **combs = data.frame (driv= c( "utilisation", "fisheries", "pollution", "IAS", "IAS"), spgroup = c( "migratory", "migratory", "migratory", "migratory","wetland"))** or another data frame with the required combination of species and drivers (section C.3.4).

Table 2: Settings to run different RLIs; cells were left blank when irrelevant if T (True) or F (False) for that RLI. Relevant sections of the report are indicated

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Code** | **Command** | **General** | **Subglobal** | **Thematic** | | | | **All** |
|  |  |  |  | **Drivers** | **Sp\_Groups(1)** | **Drivers-Sp\_Groups(1)** | **(Any thematic)**  **Subglobal** |  |
| #1.2 | **saveglobal** | T | F\* | T | T | T | F\* | T |
| #1.2 | **skip\_subglobal** | T\* | F |  |  |  | F | F |
| #1.2 | **skip\_thematic** | T\* | T\* | F | F | F | F | F |
| #1.2 | **only\_thematic** | F | F | T\* | T\* | T\* | T\* | F |
| #1.2 | **dis\_subglobal** |  |  |  |  |  | T | T |
| #1.3.1 | **isos2use** |  | C.3.1. |  |  |  | C.3.1. | C.3.1. |
| #1.3.2 | **driv\_gc** |  |  | C.3.2. |  | C.3.2. | C.3.2. | C.3.2. |
| # 1.3.3 | **splists** |  |  |  | C.3.3. | C.3.3. | C.3.3. | C.3.3. |
| # 1.3.4 | **combs** |  |  |  |  | C.3.4. | C.3.4. | C.3.4. |
| Output folder |  | General / global | Subglobal / [iso group] | Thematic/ [theme] / global | Thematic/ [theme] / global | Thematic/ [theme] / global | Thematic/ [theme] / [iso group] |  |

\* to run only this type of RLI

(1) if to run user-specified groups of species, set **other\_species\_lists=T** (#1.2)

(2) see options of iso groups names and themes names in table 1 and section B.5.

## B.5. Outputs

* The analyses will generate a series of csv files with the results (see structure of the output csv files in annex 2);
* pdf and png files with the plots of the RLIs can also be generated by setting **pdfs=T** and **pngs=T**, respectively (section C.2);
* All the files are stored in subfolders within the “**output folder**” (see above and section C.1.) with the following structure:

**Subfolder level 1:** General [or] Subglobal [or] Thematic

**Subfolder level 2:** global [or] *name of the iso group*[[9]](#footnote-9) [or] *name of the theme*[[10]](#footnote-10)

**Subfolder level 2.5** (only for thematic RLIs): global [or] *name of the iso group*

**Subfolders level 3:**

csv tables

* + - * + [A]\_**aggregated**\_[B].csv (only for global RLIs) – results of the aggregated RLI;
        + [A]\_**all\_taxa**\_[B].csv (only for global RLIs) – results of each taxa and aggregated RLI;
        + [A]\_[B].csv (only for subglobal RLIs) – results of each taxa and aggregated RLI; outputs also number of species and of genuine changes in the country/region;

csv\_tables\_crlis (only for subglobal RLIs)

* + - * + [A]\_ **crli**\_[B].csv – results of the national contribution to the global RLI;

pdfs

* + - * + [A]\_**aggregated**\_[B].pdf – pdfd with the plot of the aggregated RLI;
        + [A]\_**all\_taxa**\_[B].pdf – pdf with the plot of the RLIs of all taxa and aggregated;
        + [A]\_ **all\_taxa\_without\_aggr\_general**\_[B].pdf – pdf with the plot of the RLIs of all taxa;
        + [A]\_ **birds**\_[B].pdf – pdf with the plot with RLI for birds;

pngs

* + - * + [A]\_**aggregated**\_[B].png – png with the plot of the aggregated RLI;
        + [A]\_**all\_taxa**\_[B].png – png with thw plot with the RLIs for all taxa and aggregated;
        + [A]\_ **all\_taxa\_without\_aggr\_general**\_[B].png – png with the plot of the RLIs of all taxa;
        + [A]\_ **birds**\_[B].png – png with the plot with RLI for birds;

[A] = Prefix of all the files = global [or] *iso group\_iso unit[[11]](#footnote-11)*;

[B] = Suffix of all the files = general [or] *theme*;

Examples: *global\_all\_taxa\_forest.pdf*, *ISO\_BL\_ZWE\_utilisation.csv*

* For subglobal RLIs, a single pdf will be also created with the results of all iso units of each iso group (i.e., countries or regions) combined in the same file;
* The results can be automatically stored following the UN templates (if **save\_UN\_template=T**; sections C.2. and C.8.) in the folder **finfolderP** and with the name set in **name\_UN\_template**.

# C. Structure of the codes

* The codes are divided in 6 main parts and several subparts:
  + Part 1: Set parameters and indicate file names and folder locations
    - 1.1. Set folder locations and names of files
    - 1.2. Set basic parameters for the analyses
    - 1.3. Set the disaggregations to run
  + Part 2: Check validity of input data and prepare tables
  + Part 3. Custom functions
  + Part 4. Run global analyses and prepare past tables ("back-casting")
    - 4.1. Reconstruct past tables
    - 4.2 global RLI
    - 4.3 exports csv, pdf and png with global RLIs
  + Part 5. Run analyses per region/country
    - 5.1 run analyses per region/country
    - 5.2 save the csv for each region/country
    - 5.3 creates single table with all regions/countries
    - 5.4 save final graphs per region/country in pdf and png
  + Part 6. Organize final file following the un template
* In order to run the codes, the **user only needs to set some parameters in part 1**, but it is strongly recommended to pay attention to the results of part 2 as well, where some checks are done to see the match between tables and if some default values are correct (see also section E. Common errors and Recommendations);
* The codes can be found in annex 3, and also in the file “RLI\_disaggreg\_calculations\_vs2.r”;

## C.1. Working directory and input tables

**→ Code part #1.1.** **SET FOLDER LOCATIONS AND NAMES OF FILES**

* The **working directory** (where all the input files should be stored) is defined in the beginning of the code (**wkfolder**), along with the **location of the output files** (**finfolderP**). If the folder for the outputs files does not exist, it will be created.
* The names of the files with the **input tables** should be provided in the following commands:

**commands \*1.1.1\***

* **1. Iso codes:** **isos=read.csv(*"Iso\_countries\_2019.csv"*)*[[12]](#footnote-12)*.** List of ISO codes, country names and respective regions/other classifications (important note: should have 0 (zero) – not NA – for missing values); see structure and field names in annex 1; see note below (section C.3.1.) regarding the match between the names of the columns and the countries/regions to run;
* **2. List of species**: **tabsp1=read.csv("*full\_species\_list.csv*")**. List of all species with the latest species SIS rec ID (TaxonID), Scientific name (Scientific), year of the latest category (year\_Cat), latest red list category (Cat\_latest) and taxonomic group (Group); see structure and field names in annex 1;
* **3. Genuine changes**: **gench1=read.csv("*genuine\_changes\_2019\_drivers.csv*")**. List of all genuine changes and respective drivers; NA/empty cells should be replaced by 0 (zero); see structure and field names in annex 1; see note below (section C.3.2.) regarding the match between the names of the columns and disaggregations to run;
* **4. Proportion of the species per country**: **sppp=read.csv("*Sp\_cnts\_pp\_FINAL\_2019.csv*")**. Table with the proportion of the distribution of each species in each country; see structure and field names in annex 1;
* **5**. **Codes genuine changes**: **gcc=read.csv("*RLI\_gc\_codes.csv*")**. Table with the codes of the assessments of genuine changes; see table in in annex 1;
* **6**. **Template for SDG reporting**: **codesreg=read.csv("*codings\_sdg\_template.csv*")**. Optional table, only if to build automatically the table for SDG reporting (see section C.8. below); see structure and field names in annex 1;
* **7. List of species to run other disaggregations based on lists of species**: **other\_groups=read.csv("*Other\_species\_lists.csv*")**. Optional table, only if to run thematic RLI for user-specified lists of species (see default lists in table 1); see structure and field names in annex 1.

**Important notes about updating input tables:**

* Table 1 needs to be updated if any ISO3 code changes (e.g. change in the jurisdiction of a territory);
* Tables 2-3 may need to be updated every year, if there are changes in the list of species, red list assessments or information regarding the drivers of genuine changes;
* Table 4 needs update when the list of species change, or the range maps change;
* Table 5 needs to be updated if any new complete assessment is made and added to the table with the genuine changes;
* Table 6 does not need updates, unless M49Codes (for SDG region/subregion definitions) change.

The codes also need **a folder per taxa** to be able to extract the information to run the thematic RLI.

These folders should be stored in the working directory. The names of these folders should be provided in the following commands:

**commands \*1.1.2\***

Mammals: **folder\_mammals="Mammalia\_2019-3"*[[13]](#footnote-13)***

Birds (table 1 of 2): **folder\_birds1="*Aves\_Part1\_Non\_Passeriformes\_2019-3*"**

Birds (table 2 of 2): **folder\_birds2="*Aves\_Part2\_Passeriformes\_2019-3*"**

Amphibians: **folder\_amphibia="*Amphibia\_2019-3*"**

Corals: **folder\_corals="*Reef\_forming\_Corals\_2019-3*"**

Cycads: **folder\_cycads="*Cycads\_2019-3*"**

Each folder should contain the following csv tables (automatically generated and provided by IUCN).

* "usetrade.csv" (to identify the lists of internationally traded species and species used for food and medicine);
* "habitats.csv" (to identify the lists of forest and wetland species);
* "all\_other\_fields.csv"" (to identify the list of migratory species);
* "assessments.csv" (to identify the lists of marine, terrestrial and freshwater species).

## C.2. Basic parameters to set

**→ Code part #1.2. SET BASIC PARAMETERS FOR THE ANALYSES**

This part of the code controls the parameters described in the following list, including the disaggregations to run (see also table 2):

**commands \*1.2\***

**maxyear=*2020[[14]](#footnote-14)*** (integer)[[15]](#footnote-15); sets the maximum year of the analyses; should change every year;

**repetitions=*1000*** (integer); number of iterations to run; ideally should be 1000 or more; usually set at 10 for testing;

**saveglobal=*T*** (T/F); set as F if no need to save Global RLIs (i.e., only to save the subglobal);

**skip\_subglobal=*T*** (T/F); set as T if no need to run subglobal RLIs (i.e., only to run the global);

**skip\_thematic=*F*** (T/F); set as T if only to run general RLIs (i.e., skip any thematic RLI);

**only\_thematic=*F*** (T/F); set as T if only to run thematic RLIs (i.e., skip the general RLI);

**dis\_subglobal=F** (T/F); set as T if thematic RLIs to be calculated also at a subglobal levels;

**save\_UN\_template=*F*** (T/F); set as T if to save the results in the UN template;

**other\_species\_lists=*F*** (T/F); set as T if to run RLIs for user-defined groups of species (i.e., not the standard derived from IUCN table);

**pdfs=*T*** (T/F); if plots with the results to be saved as pdf files;

**pngs=*T*** (T/F); if plots with the results to be saved as png files;

**slopecv=*0.6*** (number); sets the standard deviation of the slope; default value is 0.6;

**name\_UN\_template=** **"Table\_RLI\_2020\_Jan2020.csv"** (text); name for final file in UN template

## C.3. Define disaggregations to run

**→ Code part # 1.3. SET THE DISAGGREGATIONS TO RUN**

In this part of the code it is possible to define the disaggregations to run, both for subglobal RLIs (by setting iso groups) and thematic RLI (by setting the drivers, groups of species or combinations of these to use in the analyses).

### C.3.1. Define iso groups for subglobal RLIs to run

**→ Code part # 1.3.1. based on isocodes or combinations of isocodes**

The list of groups of isos to run should be defined in the command **isos2use**, as described below. This command creates a vector with the names of the columns in the **isos** file (see section with input tables above). Therefore, it is fundamental that the values in this vector match exactly with the names of the respective columns in the **isos** file (see also annex 1).

**command \*1.3.1.1\***

**isos2use=c("ISO\_BL","ISO\_SDG","SDG\_Region","SDG\_Subregion","LLDC\_SIDS","LDC","IPBES\_region","CMS\_Region")**

**Important notes:**

* Values can be added or removed from the list as needed;
* If to run only global RLI, set empty vector: **isos2use=NULL;**
* If to run both “ISO\_BL” and “ISO\_SDG”, “ISO\_BL” should be first in the list of values; this is because the codes are written to use the results of ISO\_BL in the ISO\_SDG analyses, when there is an exact match between both (to speed up processing time).

### C.3.2. Define thematic drivers RLIs to run

**→ Code part # 1.3.2. based on drivers of genuine changes**

The list of drivers of genuine changes to analyse should be set in the command \*1.3.2.2\* **driv\_gc**, as described below.

The codes use the current (Feb 2020) list of column names in the table of genuine changes provided by IUCN to indicate which columns should be used for each driver, as follows (**important note: these commands should not be changed**, unless the names of the columns in the input table change, to add a new driver or to redefine which columns should be used in a driver):

**command \*1.3.2.1\***

**utilisation=c("driverHuntingAll", "driverHuntingInternational", "driverLogging", "driverFisheries", "driverPlants")**

**IAS="driverInvasiveSpecies"**

**pollution="driverPollution"**

**fisheries="driverFisheries"**

**The setting of which of these drivers will run is done in the following command (**\*1.3.2.2\***):**

**command \*1.3.2.2\***

**driv\_gc=list(utilisation=utilisation,IAS=IAS,pollution=pollution,fisheries=fisheries)**

**Important notes:**

* The command \*1.3.2.2\* creates a list of vectors; it is fundamental to repeat the names of the values (**XXXX=XXX**) as shown (e.g. ***utilisation=utilisation***);
* Values can be added or removed from the list as needed; however, if new drivers to be added (that are not current in the command \*1.3.2.2\*), they should be also defined in the list of commands \*1.3.2.1\*, by indicating the relevant column names in the table of genuine changes corresponding to that driver
* If not to run any thematic drivers RLI, set empty vector: **driv\_gc=NULL;**

### C.3.3. Define thematic Sp\_Groups RLIs to run

**→ Code part # 1.3.3. based on groups of species**

The codes are written to identify automatically the following groups of species (based on tables provided by IUCN – see table 1 and section C.1., **commands \*1.1.2\***):

* internationally traded
* used for food and medicine
* forest
* wetland
* migratory
* marine
* terrestrial
* freshwater

However, the list of groups to be analysed can be only a subset of these, as defined in the command \*1.3.3\***splists** described below.

**command \*1.3.3\***

**splists=c("internationallytraded","medicine","forest","wetland","migratory","marine", "terrestrial","freshwater")**

It is also possible to run the RLI for other (user-defined) groups of species. In that case, a list of these groups and respective TaxonRecIDs should be provided (see format of this table in section C.1. and annex 1). In this case, set **other\_species\_lists=T**(see commands **\*1.2\***).

**Important notes:**

* If running user defined species (i.e., if **other\_species\_lists=T**), the code will not run the default groups of species listed in command \*1.3.3\*;
* If not to run any thematic Sp\_Groups RLI, set empty vector: **splists=NULL;**

### C.3.4. Define thematic Drivers - Sp\_Groups RLIs to run

**→ Code part # 1.3.4. based on combinations of drivers and groups of species**

The command below **\*1.3.4\*** creates a table with the combinations of drivers and groups of species to run. This table has two vectors (**driv** and **spgroup**), so the number of elements in these vectors should be the same, and the order of the elements in each vector will define the final combinations to run.

**command \*1.3.4\***

**combs=data.frame(driv=c("utilisation","fisheries", "pollution", "IAS", "IAS"), spgroup=c("migratory","migratory","migratory","migratory","wetland"))**

Each row of the resulting table is a combination of drivers and species to run. In the example provided in command **\*1.3.4\***, five “thematic drivers - sp\_groups” RLI will be set (the ones listed in table 1):

|  |  |  |
| --- | --- | --- |
|  | driv | spgroup |
| 1 | utilisation | migratory |
| 2 | fisheries | migratory |
| 3 | pollution | migratory |
| 4 | IAS | migratory |
| 5 | IAS | wetland |

**Important notes:**

* The names of vector driv should be also provided in command \***1.3.2.2**\* **driv\_gc**;
* The names of groups of species should be also provided in command **\*1.3.3\*** **splists**; if these are user-defined groups of species, set **other\_species\_lists=T** (see section C. 3.3.) and provide respective table;
* If not to run any thematic Drivers - Sp\_Groups RLI, set empty table: **combs=NULL**.

## C.4. Check validity of input data and prepare tables

**→ Code part #2. CHECK VALIDITY OF INPUT DATA AND PREPARE TABLES**

* This part of the code checks for possible errors and inconsistencies in the input data. The first check is to see the match between the names in the table with the iso codes and the iso groups to run provided in the command **isos2use**, for subglobal RLIs. All the values provided in the **isos2use** vector should correspond to columns in the table with the iso codes (see structure of the table in annex 1). Any misspelled value in **isos2use** vector (i.e., a name of iso group that is not in the table with the iso codes) will be ignored in the analyses.
* The code will also check for the match between the ISO3 values between the table with the iso codes and the table with the proportion of the species per country. It is expected that the only difference between the unique ISO3 values of both tables is “ABNJ” (only existing in table with iso codes). Any ISO3 value missing from the table with iso codes will be ignored.
* This part of the code will organize the lists of species for the Thematic - Sp\_Groups disaggregations (see table 1). This is done automatically based on the input folders with the data downloaded from the IUCN databases (see table 3).

Table 3: Settings used to extract the list of species from the IUCN folders for each taxa (see also section C.1. and annex 1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Disaggregation** | **csv file to use** | **Field name** | **Options to include** | **Qualifiers** |
| internationallytraded | usetrade | international | TRUE |  |
| medicine | usetrade | code | 1 or 3 |  |
| forest | habitats | code | Any value beginning with 1 | majorImportance'=’Yes’ OR no other habitats with 'Suitable' in ‘Suitability’ OR have no habitats with coding in ‘Suitability’ and only have forest habitats |
| wetland | habitats | code | Any value beginning with 5 or 15 | majorImportance'=’Yes’ OR no other habitats with 'Suitable' in ‘Suitability’ OR have no habitats with coding in ‘Suitability’ and only have wetland habitats |
| migratory | all\_other\_fields | MovementPatterns.pattern | Full Migrant |  |
| marine | assessments | systems | Marine |  |
| terrestrial | assessments | systems | Terrestrial |  |
| freshwater | assessments | systems | Freshwater (=Inland waters) |  |

* The genuine change table will be reorganized to add the codes of the genuine changes (see list of tables in section C.1.) and the scientific names, and to check for inconsistencies between the list of species in the table with the genuine changes and the list of species. It will also identify which entries of the table with the genuine changes don’t have a “primary” driver but have a “major secondary”.
* A vector with the weights of each red list category for the RLI calculations will be created, as follows:

**command \*4.1.\***

**wts=data.frame(iucn=c("LC","NT","VU","EN","CR","CR(PE)","CR(PEW)","EW","EX","DD"), wg=c(0,1,2,3,4,5,5,5,5,-1))**

* Possible errors in the latest red list categories in the table with the list of species will be corrected, based on the information in the genuine changes table;
* Some final checks are done, to see if the structure of the tables remained the same after the merges mentioned above, if all the red list categories codes are correct and if all the scientific names in the table with the genuine changes are also in the table with the proportion of the species per country

## C.5. Custom functions

**→ Code part #** **3. CUSTOM FUNCTIONS**

* A first set of very basic custom functions is created, to help with the calculations:

**first=function (x=x) x[1]** ## function to output the first value of a vector

**q95=function (x=x) quantile(x,.95)** ## function to estimate the quantile 95%

**q05=function (x=x) quantile(x,.05)** ## function to estimate the quantile 5%

**div0=function (x=x, y=y)** trunc((x-1)/y)-((x-1)/y) ## function that outputs 0 if y-1 is divisible by x (aux to open new devices in graphics outputs)

* The grey colour for the error bars of the plots is also defined;
* A general function to calculate the global RLI is created to apply in part #4 of the code; it follows the methodology described in detail in Butchart et al. 2007[[16]](#footnote-16); all the steps in the codes are annotated in the code; in summary, the function uses a table with the “back-casting” of all red list categories, based on the latest red list category and the genuine changes (see Butchart et al. 2007). The code first loops through all the years of assessment of each taxa. A random red list category will be assigned to any data-deficient (DD) species, based on the number of species in each non-DD category for each taxa. The code then does an interpolation to estimate the RLI for all the years between assessments of each taxon, and then extrapolates to the years before (left extrapolation) and after (right extrapolation) this interval. This extrapolation is done by first creating a linear model for the rate of the change in the first two years (for the left extrapolation) or the last two years (for the right extrapolation) with RLI assessments. The final slope considered for extrapolation is a random value based on the slope from the linear model and a potential standard deviation (set in **slopecv**; see C.2.). A moving average will then consider a random RLI from the interval of +- 2 years from each year. A bootstrapping loop will repeat this procedure the number of times set in **repetitions** (see C.2.), ideally 1000, and store all the results ((RLI per year, group and iteration).
* A general function to plot the final RLIs is created.

## C.6. Prepare “back casting” tables and run global analyses

**→ Code part #4** **RUN GLOBAL ANALYSES AND PREPARE PAST TABLES ("back-casting")**

* The first part of this code starts a loop that will run through all the RLIs to be calculated (see table 1 and section B). Depending on the initial settings, the codes will run through all RLIs or just some (table 2); this loop will only finish in the end of part #5 of the code.
* A subset of genuine changes to be used, or of species to be included, with be created depending on the RLI to run (General of Thematic).
  + If general, all species and all genuine changes will be considered;
  + If thematic, only those genuine changes and/or species corresponding to the disaggregations set in the beginning of the code will be considered;
    - if Thematic Drivers: all species will be included; only genuine changes driven by the drivers set in **driv\_gc**; only drivers indicated as “Primary” or “Major Secondary” are considered (see “structure of the table with the genuine changes and drivers” in annex 1); an “alternative genuine changes table” is also created, which considers also Secondary drivers, to be used in half of the iterations.
    - if Thematic Sp\_Groups: only the species set in **splists** (or user-defined lists if **other\_species\_lists=T**); all genuine changes of these species will be included;
    - if Thematic Drivers - Sp\_Groups: only the combinations of species and drivers defined in **combs** will be included;
* Based on the lists of species/genuine changes defined, the code will “back cast” the red list categories based on the latest red list categories and relevant genuine changes (part **# 4.1. reconstruct past tables**). The code will loop through all the years, starting by the latest, and subsequently change the previous categories when relevant, based on the table with the genuine changes; the output of this part is a table names **pastables**, which will used in the RLI calculation; an “alternative past table” is also created (**pastables\_alternative**), based on the “alternative genuine changes table”, to be used in half of the iterations of the calculations of Thematic Drivers RLIand Thematic Drivers - Sp\_Groups RLI. This way, the Secondary drivers are only considered in the calculations in half of the cases and their inclusion accounts also to the variability observed in the errors of these disaggregations;
* The global RLI is calculated to all taxa based on the code defined in part #3 (see C.5.) except for corals part (part **# 4.2. global RLI**); corals are calculated separately (and after) because the left extrapolation is not based on the first two years of assessments (cannot that current steep decline also occurred in the past), but rather on the average of “left slopes” of the other taxa; therefore, the RLIs for the other taxa are analysed first;
* The global RLI is then calculated for corals also based on the code defined in part #3 (see C.5.), using the left slope estimated for the other taxa;
* An aggregated RLI is calculated, as the mean value of the other taxa for each year;
* The RLIs for all taxa (including aggregated) for each year and repetition is then combined in a single table named **rliTotF**;
* The final RLI for each taxonomic group and year is calculated with the **rliTotF** table, by extracting the median values of all the repetitions, along with the quantiles 5% (**q05**) and 95% (**q95**);
* The RLI for all taxa is calculated for the entire period between the first global assessment of any taxa and the final year set in **maxyear** (see C.2.). This is necessary to have the same interval to calculate the aggregated RLI; however, the final output for each taxa is cropped to the period between assessments of that taxa; the final output for the aggregated RLI is cropped to the period between 10 years before the most recent first assessment of any taxa and the final year set in **maxyear**.
* Final outputs are saved as csv files (if **saveglobal=T**) in part **# 4.2. exports csv, pdf and png with global RLIs**; global RLIs are plotted and saved in pdf (if **pdfs=T**) and png files (if **pngs=T**) , if **saveglobal=T**; see details about these files in section B.5.

## C.7. Prepare run subglobal analyses

**→ Code part #** **5. RUN ANALYES PER REGION/COUNTRY**

* Part 5 is not independent of part 4, as it needs the outputs of part 4 to be able to run; this part is within the loop initiated in part 4, to go through all RLIs to calculate (general and disaggregated);
* The code starts with a loop to go through all the iso groups defined in the **isos2use** vector (see C.3.), and within this another loop to go through all the unique units (e.g. countries, regions) within each iso group (part **# 5.1 run analyses per region/country**);
* A modified version of the global RLI calculator (defined in C.5.) is applied; the only difference is that the RLI for each unit is weighted by the proportion of the range of the species in that unit (as detailed described in Rodrigues et al. 2014[[17]](#footnote-17)); this is done by multiplying the scores related with the red list category of each species (see definition of vector **wts** in **command \*4.1.\***, section C.4.) by the proportion of the range map of that species in that unit, based on the table with the proportion of the species per country; the number of species and genuine changes of each taxa in each country/region is also calculated and stored to be saved in the final csv per country (annex 2); the main changes between this code and the global RLI calculator are noted and explained in the code;
* As the loop goes through all the units (countries or regions, depending on the iso group), the csv files are automatically saved – part # **5.2 save the csv for each region/country** (see details about these files in section B.5).
* These files are then reloaded in the next part of the code, after finishing the loops to go through all countries/regions (part # **5.3 creates single table with all regions/countries**).
* The code allows to upload and combine all the csv files in a single table from previously stored data, if needed (option **cmbfiles**, set as FALSE by default);
* Subglobal RLIs are plotted and saved in the final part - part **# 5.4. save final graphs per region/country in pdf and png** - in pdf (if **pdfs=T**) and png files (if **pngs=T**); see details about these files in section B.5;

## C.8. Combine and save results following UN template

**→ Code part #** **6. ORGANIZE FINAL FILE FOLLOWING THE UN SDG TEMPLATE**

* This part of the code is independent from the other parts and can run separately, to compile results of previous analyses; it requires however that these results are stored in the output files automatically generated by the code (i.e., following the same structure as described in section B5, and saved in **finfolderP** (see also section C.1.);
* To be able to run this part of the code it is required to:
  + set **save\_UN\_template=T** (section C.2);
  + run the codes until end of part #1.2;
  + run (or have previously run) the following iso groups (see table 1 and section C.3.1.):
    - ISO\_SDG
    - SDG\_Region
    - LLDC\_SIDS
    - LDC
  + provide an input table with the template (see **codesreg**, section C.1.).
* The final csv file following the UN template will be stored in the folder defined in **finfolderP** (section C.1.); the name of this file is defined in **name\_UN\_template** (section C.2.).

# D. Major changes from the previous version

* The version 2.0 of the code is based on version 1.0 available in the report of the project P90001: *Indicator-related work to a) code to calculate national Red List Indices (RLIs), and b) calculate and provide updated data on the coverage by protected areas of marine, terrestrial and freshwater Key Biodiversity Areas*;
* The code was first simplified by creating some custom functions (i.e., to generate the global RLIs, to save the png and pdf plots), to be applied throughout the multiple repetitions of the RLI calculations (see section C.5.);
* Notes were added to the several commands of the code, to explain the reasoning behind each step; however, it is important to note that some basic knowledge on how the disaggregated RLI is calculated is required to understand the code (along with some knowledge in R code). The notes do not pretend to replace the information already published in Butchart et al. 2004[[18]](#footnote-18), 2005[[19]](#footnote-19), 2007[[20]](#footnote-20), 2010[[21]](#footnote-21); Rodrigues et al. 2014[[22]](#footnote-22). The reading of these publications is highly recommended to understand how the code works;
* The codes were organized in several parts and subparts, to facilitate their interpretation and application (see introduction section C);
* The codes were also updated to allow the possibility to run thematic disaggregations (table 1). This update implied some major changes to the codes, given that the previous version was designed to run only general and subglobal disaggregations. Some of the most important changes related with this update were:
  + Include a set of new commands to define which disaggregations to run (see C.2. and table 2), adding as much flexibility as possible; it is now possible to run only the general RLI, several combinations of disaggregated RLIs, or all at the same time (see C.3.);
  + Extract automatically information from IUCN databases to inform lists of species (for the Thematic Sp\_Groups RLI); see section C.4. and table 3;
  + Include the possibility of adding a user-defined list of species to calculate other (non-default) Thematic Sp\_Groups disaggregations; see section C.3.3.
  + Add a series of new commands related with the new format for the table with the genuine changes (which did not include the drivers in the previous versions, but used to include the scientific names of the species); see section C.4.;
  + Include an additional loop (starting in the beginning of part 4 and finishing in the end of part 5) to go through all the desired disaggregations, including a combination of subglobal and thematic; this required several new commands, mostly related with the creation of subsets of genuine changes and/or lists of species (section C.6.);
  + Add the possibility of considering the “Secondary” drivers of genuine changes in half of the iterations (section C.6.);
  + Calculate and store in the final csv files the number os species and genuine changes used in the calculations of subglobal RLIs (section C.7.);
  + Organize the output folders and files that are created automatically and stored within the set final folder, with self-explanatory names (section B.5.);
  + Include the possibility to save the outputs following a required template (section C.8.).

# E. Common errors and Recommendations

* In order to run smoothly, it is fundamental to provide the input tables in the correct format and with the required values;
* The most common source of problems when running the codes are related with incorrect formatting of the tables (see also C.1. and annex 1):
  + Mismatch between the lists of species provided in different tables (e.g. TaxonIDs in the table with the genuine changes that are not in the list of species; Scientific names in the lists of species that are not in the proportion of the species in each country, or vice versa);
  + Mismatch between ISO3 codes in different tables (e.g. ISO3 codes in the table with the proportion of the species in each country that are not listed in the table with the iso codes);
  + Errors in names of the regions in the table with the iso codes: avoid accents and non-common characters – can prevent code to run because of file names; ex: SDG\_Region “Western Asia<U+00A0>(M49) and Northern Africa (M49)”;
  + Misspelling of red list categories in table with the list of species: for example “CR(PE)” is often (but not always) written as “CR (PE)”; as this relates to CR species – CR (PE), CR (PEW), it can result in strong bias in the results; it should be written without spaces (see **command \*4.1.\***).
  + Missing values in the table with genuine changes, especially in the fields “periodStartCategory” and “periodEndCategory” (both essential to the code); note that all the empty cells of the drivers should be replaced by 0 (zero).
* All the issues mentioned above should be carefully checked in advance.
* Another potential source of error is related with wrong or contradictory settings; for example:
  + Cannot set **skip\_thematic=T** and **only\_thematic=T** at the same time;
  + If to run Thematic Drivers - Sp\_Groups RLI, the correspondent drivers and sp\_groups should also be set (see C.3.4.);
  + If to run Thematic Sp\_Groups RLI to user-defined groups of species **other\_species\_lists=T**, the correspondent input table should be provided
* The names of the folders of each taxonomic group (see section C.1., **commands \*1.1.2\***) should be carefully checked; pay particular attention to different names for the two folders of birds;
* Subglobal RLIs, particular the national ones (see table 1), take usually a long time to run (typically around 1 day in a machine with a processor Intel ® Core ™ i7 – 6700HQ CPU @ 2.60GHz with 16.0 GB of installed RAM). General and Thematic global RLIs are relatively fast and should take only a few minutes to run.

# Annex 1: Structure of the input tables

Structure of the table with the Iso codes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **countryname** | **ordem** | **country\_SIS** | **region** | **ISO3** | **ISO\_BL** | **ISO\_SDG** | **uname\_BL** | **uname\_SDG** | **Region(old\_MDG)** | **Developing** | **LDC** | **LLDC\_SIDS** | **CN\_SHAPEFILE** | **IPBES\_region** | **SDG\_Region** | **SDG\_Subregion** | **CMS\_Region** |
| Afghanistan | 66 | Afghanistan | West and Central Asia | AFG | AFG | AFG | Afghanistan | Afghanistan | Southern Asia | Developing | LDC | LLDC | 1 | Asia and the Pacific | Central Asia (M49) and Southern Asia (MDG=M49) | Southern Asia (MDG=M49) | Asia |
| Akrotiri and Dhekelia | 256 | Akrotiri and Dhekelia | Europe | XAD | CYP | CYP | Cyprus | Cyprus | Developed regions | 0 | 0 | 0 | 1 | Europe and Central Asia | Northern America (M49) and Europe (M49) | Western Asia (M49) | Europe |
| Aland | 248 | Aland Islands | Europe | ALA | ALA | FIN | Finland | Finland | Developed regions | 0 | 0 | 0 | 1 | Europe and Central Asia | Northern America (M49) and Europe (M49) | Europe (M49) | Europe |

Structure of the tables with the list of species

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TaxonID** | **Scientific** | **year\_Cat** | **Cat\_latest** | **Group** |
| 22687170 | Abeillia abeillei | 2016 | LC | Bird |
| 58647 | Abavorana luctuosa | 2014 | LC | Amphibian |
| 133411 | Acanthastrea bowerbanki | 2008 | VU | Coral |
| 42641 | Abditomys latidens | 2008 | DD | Mammal |
| 41979 | Bowenia serrulata | 2013 | LC | Cycad |

Structure of the table with the genuine changes and drivers

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RedListIndexLookup** | **periodStartCategory** | **periodEndCategory** | **driverAgriculture** | **driverLogging** | **driverFisheries** | **driverInvasiveSpecies** | **driverNativeSpecies** | **driverHuntingAll** | **driverHuntingInternational** | **driverResidential** | **driverEnergy** | **driverFire** | **driverDams** | **driverClimate** | **driverPollution** | **driverHumanDisturbance** | **driverTransportation** | **driverGeological** | **driverPlants** | **driverOtherDiseases** | **driverUnknown** | **assessmentid** | **taxonid** |
| 3.1 | EN | CR | Secondary | 0 | 0 | Primary | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Secondary | Secondary | 0 | 0 | 0 | 0 | 0 | 0 | 3015811 | 54519 |
| 3.1 | EN | CR | Secondary | Secondary | 0 | Primary | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Secondary | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3015954 | 54531 |

Note possible values for fields with the drivers (underlined): “Primary”, “Most Important Secondary”, “Secondary”

Structure of the table with the proportion of the species in each country

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scientific** | **group** | **TaxonID** | **ISO3** | **pp** |
| Abavorana luctuosa | Amphibian | 58647 | IDN | 0.214665 |
| Abavorana luctuosa | Amphibian | 58647 | MYS | 0.765143 |
| Abavorana luctuosa | Amphibian | 58647 | THA | 0.020191 |
| Acanthixalus sonjae | Amphibian | 100712 | CIV | 0.979034 |

Table with the codes of genuine changes

|  |  |  |  |
| --- | --- | --- | --- |
| **RedListIndexLookup** | **Group** | **start\_year** | **end\_year** |
| 1.1 | Bird | 1988 | 1994 |
| 1.2 | Bird | 1994 | 2000 |
| 1.3 | Bird | 2000 | 2004 |
| 1.4 | Bird | 2004 | 2008 |
| 1.5 | Bird | 2008 | 2012 |
| 1.6 | Bird | 2012 | 2016 |
| 1.7 | Bird | 2017 | 2020 |
| 2.1 | Mammal | 1996 | 2008 |
| 2.2 | Mammal | 2008 | 2020 |
| 3.1 | Amphibian | 1980 | 2004 |
| 3.2 | Amphibian | 2004 | 2020 |
| 4.1 | Coral | 1996 | 2008 |
| 4.2 | Coral | 2008 | 2020 |
| 6.1 | Cycad | 2003 | 2014 |

Table with template for SDG reporting

|  |  |  |  |
| --- | --- | --- | --- |
| **M49Code** | **ISO** | **CountryName** | **Reference** |
| 4 | AFG | Afghanistan | 1.0-Global |
| 248 | ALA | Åland Islands | 1.0-Global |
| 8 | ALB | Albania | 1.0-Global |

Structure of the table with lists of species to run other disaggregations

|  |  |
| --- | --- |
| **TaxonID** | **Group\_species** |
| 22678551 | Galliformes |
| 22678555 | Galliformes |
| 22678559 | Galliformes |
| 22678564 | Galliformes |
| 22678568 | Galliformes |
| 22678572 | Galliformes |
| 22678646 | Galliformes |
| 22678576 | Galliformes |
| 22691485 | Pigeons |
| 22691480 | Pigeons |
| 60760066 | Pigeons |
| 22691465 | Pigeons |

# Annex 2: Structure of the output tables

The output csv tables of the global RLIs have the following structure (rli=median RLI per year, qn05=quantile 5% of the confidence limits, qn95=quantile 95% of the confidence limits):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| group | year | rli | qn05 | qn95 |
| Amphibian | 1980 | 0.758396 | 0.755998 | 0.761161 |
| Amphibian | 1981 | 0.757381 | 0.755473 | 0.760328 |
| Amphibian | 1982 | 0.756988 | 0.754777 | 0.759472 |
| Amphibian | 1983 | 0.756338 | 0.75438 | 0.759021 |
| Amphibian | 1984 | 0.755518 | 0.753545 | 0.758344 |
| Amphibian | 1985 | 0.755102 | 0.752783 | 0.75755 |
| Amphibian | 1986 | 0.754582 | 0.752258 | 0.756753 |
| Amphibian | 1987 | 0.753877 | 0.751592 | 0.755992 |
| Amphibian | 1988 | 0.753315 | 0.750883 | 0.755088 |
| Amphibian | 1989 | 0.752418 | 0.750294 | 0.755089 |
| Amphibian | 1990 | 0.751916 | 0.749833 | 0.753827 |

The output csv tables of the subglobal RLIs have the following structure (rli= median RLI per year, qn05=quantile 5% of the confidence limits, qn95=quantile 95% of the confidence limits, n\_sp=number of species of the taxa in the country/region, n\_gen\_changes=number of genuine changes for the taxa in the country/region):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| country | group | year | rli | qn05 | qn95 | n\_sp | n\_gen\_changes |
| BRA | Bird | 1980 | 0.925321 | 0.924408 | 0.925965 | 1824 | 74 |
| BRA | Bird | 1981 | 0.925288 | 0.924424 | 0.925796 | 1824 | 74 |
| BRA | Bird | 1982 | 0.925236 | 0.924467 | 0.925743 | 1824 | 74 |
| BRA | Bird | 1983 | 0.925192 | 0.924484 | 0.925662 | 1824 | 74 |
| BRA | Bird | 1984 | 0.925154 | 0.924544 | 0.925526 | 1824 | 74 |
| BRA | Bird | 1985 | 0.925109 | 0.924544 | 0.925441 | 1824 | 74 |

The output csv tables of the subglobal CRLIs have the following structure (crli= median CRLI per year, crli05=quantile 5% of the confidence limits, crli95=quantile 95% of the confidence limits

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| country | group | year | crli | crli05 | crli95 |
| ABW | Amphibian | 1980 | 2.77E-08 | 2.77E-08 | 2.77E-08 |
| ABW | Amphibian | 1981 | 2.77E-08 | 2.77E-08 | 2.77E-08 |
| ABW | Amphibian | 1982 | 2.77E-08 | 2.77E-08 | 2.77E-08 |
| ABW | Amphibian | 1983 | 2.77E-08 | 2.77E-08 | 2.77E-08 |
| ABW | Amphibian | 1984 | 2.77E-08 | 2.77E-08 | 2.77E-08 |
| ABW | Amphibian | 1985 | 2.77E-08 | 2.77E-08 | 2.77E-08 |

# Annex 3: General and Disaggregated RLIs calculator; Version 2.0

# General and Disaggregated RLIs calculator

# Version 2.0 12/02/2020

# Maria Dias/BirdLife International; maria.dias@birdlife.org

# Script to calculate and plot the Global, Regional and Disaggregated RLIs

## See formats of input tables in word document "Code to calculate general and disaggregated Red List Indices (RLIs) Version 2.0. Instructions and recommendations"

# PART 1. SET PARAMETERS AND INDICATE FILE NAMES AND FOLDER LOCATIONS ###############################

########################################################################################################

# 1.1. SET FOLDER LOCATIONS AND NAMES OF FILES ############################################################

wkfolder="C:/Users/IUCN R codes/2020/input files" ## copy-paste the working folder, replacing "/" by "/" and ensuring no spaces before & after quotes

setwd(wkfolder)

finfolderP="C:/Users/IUCN R codes/2020/outputs\_RLI" ## copy-paste the location of the folder where the country files should be saved, replacing "/" by "/" and ensuring no spaces before & after quotes

isos=read.csv("Iso\_countries\_2019.csv") ## file with ISO codes; should be stored in the wkfolder specified above; should have the following columns (with these exact names): "ISO3","ISO\_BL","ISO\_SDG","LDC","LLDC\_SIDS","IPBES\_region","SDG\_Region","SDG\_Subregion","CMS\_Region"

tabsp1=read.csv("full\_species\_list.csv") ## file with list of species; should be stored in the wkfolder specified above; should have the follwoing columns (with these exact names): "TaxonID","Scientific","Group","Cat\_latest"; Scientific= Latin name; Group= taxonomic group (Amphibian, Bird, Coral, Cycad or Mammal); Cat\_latest= latest IUCN red list category - STANDARTIZED!! ONLY POSSIBLE OPTIONS ARE "LC","NT","VU","EN","CR","CR(PE)","CR(PEW)","EW","EX","DD"

gench1=read.csv("genuine\_changes\_2019\_drivers.csv") ### file with genuine changes; should be stored in the wkfolder specified above #make sure no files in the genuine changes go past the RL completion cycle (e.g. 2016-2020); should have the follwoing columns (with these exact names): "TaxonID","Scientific","Group","Cat\_latest"

sppp=read.csv("Sp\_cnts\_pp\_FINAL\_2019\_new\_corrected.csv") ## result of the spatial analyses (see Annex 1) to estimate the proportion of the range maps in each country; should be stored in the wkfolder specified above; should have the follwoing columns (with these exact names): "TaxonID","Scientific","ISO3","pp"

gcc=read.csv("RLI\_gc\_codes.csv") ### table with the groups and codes for the genuine changes table (with years of assessment)

codesreg=read.csv("codings\_sdg\_template.csv") # table with the codes for UN reporting; only needed if to save fowlloing their templates (see below "save\_UN\_template")

#other\_groups=read.csv("Other\_species\_lists.csv") # table with the taxonIDs of other groups of species to run; only needed if to run disaggregated RLIs for used-defined groups of species (see below option other\_species\_lists=T)

folder\_mammals="Mammalia\_2019-3"

folder\_birds1="Aves\_Part1\_Non\_Passeriformes\_2019-3"

folder\_birds2="Aves\_Part2\_Passeriformes\_2019-3"

folder\_amphibia="Amphibia\_2019-3"

folder\_corals="Reef\_forming\_Corals\_2019-3"

folder\_cycads="Cycads\_2019-3"

# 1.2. SET BASIC PARAMETERS FOR THE ANALYSES

maxyear=2020 ### max year of the analyses - change to new max year (was 2017)

repetitions=1000 ## n repetitions - change to new max rep (was 1000, reduced to reduce processing time)

saveglobal=T ### set as FALSE if no need to save global rlis (i.e., only to save the subglobal)

skip\_subglobal=F ### set as TRUE if no need to run subglobal rlis (i.e., only to run the global)

skip\_thematic=F # if only to run general RLIs (i.e., skip the thematic RLI)

only\_thematic=F # if only to run thematic RLIs (i.e., ignore the general RLI)

dis\_subglobal=T # if disaggregated RLIs to be calculated also at subglobal levels

save\_UN\_template=F # to save the results in the UN template

other\_species\_lists=F # if to run RLIs for user-defined groups of species (i.e., not the standard derived from IUCN table)

pdfs=T ### if plots to be saved as pdf files

pngs=T ### if plots to be saved as png files

slopecv=.6 ### standard deviation of the slope

name\_UN\_template="Table\_RLI\_2020\_Jan2020.csv"

# 1.3 SET THE DISAGGREGATIONS TO RUN

####### 1.3.1. based on isocodes or combinations of isocodes - check match between names and columns in isos table

isos2use=c("ISO\_BL","ISO\_SDG","SDG\_Region","SDG\_Subregion","LLDC\_SIDS","LDC","IPBES\_region","CMS\_Region")#,"SDG\_Subregion","IPBES\_region" ### IMPORTANT NOTE!!! IF TO RUN BOTH ISO\_BL AND ISO\_SDG, ISO\_BL SHOULD BE FIRST IN THE LIST

### add or remove as needed; global will always run by default; check perfect match with column names in isos table;

##if to run only global, set empty vector: isos2use=NULL

if (skip\_subglobal) isos2use=NULL

length(isos2use)

####### 1.3.2. based on drivers of genuine changes - check match between names in following vectors and columns in gench1 table

### options are: c(utilisation=utilisation,IAS=IAS,pollution=pollution,fisheries=fisheries) f

# IMPORTANT NOTE: leave empty vector if not to run any RLI based on the drivers of genuine changes: driv\_gc=NULL

# set here the fields in the table of the genuine changes corresponding to each driver

utilisation=c("driverHuntingAll", "driverHuntingInternational", "driverLogging", "driverFisheries", "driverPlants")

IAS="driverInvasiveSpecies"

pollution="driverPollution"

fisheries="driverFisheries"

# set here the drivers to run; repeat names as shown: XXX=XXXX (e.g. driv\_gc=list(utilisation=utilisation,IAS=IAS,pollution=pollution,fisheries=fisheries)

driv\_gc=list(utilisation=utilisation,IAS=IAS, fisheries=fisheries,pollution=pollution) #

#driv\_gc=NULL

if (skip\_thematic) driv\_gc=NULL

length(driv\_gc)

list\_drivers=c("driverAgriculture", "driverLogging","driverFisheries","driverInvasiveSpecies", "driverNativeSpecies",

"driverHuntingAll","driverHuntingInternational","driverResidential","driverEnergy","driverFire","driverDams","driverClimate",

"driverPollution","driverHumanDisturbance","driverTransportation","driverGeological","driverPlants","driverOtherDiseases","driverUnknown")

### 1.3.3. based on groups of species

## IMPORTANT NOTE: leave empty vector if not to run any RLI based on the groups of species: splists=NULL; if to run user-defined lists of species, set other\_species\_lists=T above and check that names below match the names in the table "other\_groups"

splists=c("internationallytraded","medicine","forest","wetland","migratory","marine","terrestrial","freshwater") # internationallytraded medicine forest wetland migratory marine terrestrial freshwater

if (other\_species\_lists) splists=unique(other\_groups$Group\_species)

splists

#splists=NULL

if (skip\_thematic) splists=NULL

length(splists)

### 1.3.4. based on combinations of drivers and groups of species

## n elements in each vector should match; names of vector driv should match the ones defined in 1.3.2

## IMPORTANT NOTE: leave empty vector if not to run any RLI based on the groups of species: combs=NULL

combs=data.frame(driv=c("utilisation","fisheries", "pollution", "IAS", "IAS"), spgroup=c("migratory","migratory","migratory","migratory","wetland"))

#combs=NULL

if (skip\_thematic) combs=NULL

if (length(combs)==0) combs=data.frame()

nrow(combs)

combs

n\_iter=1+length(driv\_gc)+length(splists)+nrow(combs) # number of iterations to run (number of disaggregated RLI + the general)

n\_iter

# PART 2. CHECK VALIDITY OF INPUT DATA AND PREPARE TABLES ###############################

########################################################################################################

#### VERY IMPORTANT - CHECK IF NAMES IN isos2use VECTOR MATCH COLUMN NAMES IN isos TABLE; FOLLOWING RESULT SHOULD BE 0

length(which(isos2use%in%names(isos)==F)) ### should be 0

isos2use=isos2use[which(isos2use%in%names(isos))] ## will delete the misspelled names (i.e., not matching with the columns in isos2use)

length(isos2use)

## verify if the number of iso codes in pps sp-countries is the same as in isos

lu=function (x=x) length(unique(x)) ## function to estimate the number of unique values in a vector

lu(sppp$ISO3) ##number of unique iso codes in table with the pps sp-countries (should be the same or one less - ABNJ - comparing with the isos)

isos$ISO3[which(!isos$isof%in%sppp$ISO3)] # confirm that the difference in isos between tables (isos and sp-countries-pp) is due to the ABNJ

sort(unique(sppp$ISO3[which(!sppp$ISO3%in%isos$ISO3)])) # confirm that the difference in isos between tables (isos and sp-countries-pp) is due to the ABNJ

## prepare table with species' lists

folders\_sp=list(folder\_birds1,folder\_birds2,folder\_mammals, folder\_amphibia, folder\_corals, folder\_cycads)

folders\_sp

## traded and medicine

trsp=NULL

medi=NULL

for (a in 1:length(folders\_sp))

{

t2read="usetrade.csv"

f2read=folders\_sp[[a]]

f2read

dat1=read.csv(paste(f2read,t2read, sep="/"))

head(dat1)

sp2extr1=dat1$internalTaxonId[dat1$international=="true"]

trsp=c(trsp,sp2extr1)

sp2extr2=dat1$internalTaxonId[dat1$code%in%c(1,3)]

medi=c(medi,sp2extr2)

}

tabsp1$internationallytraded=0

tabsp1$internationallytraded[tabsp1$TaxonID%in%trsp]=1

tabsp1$medicine=0

tabsp1$medicine[tabsp1$TaxonID%in%medi]=1

head(tabsp1[tabsp1$internationallytraded==1&tabsp1$Group=="Mammal",])

head(tabsp1[tabsp1$medicine==1&tabsp1$Group=="Mammal",])

# forest and wetlands

forest=NULL

wetland=NULL

for (a in 1:length(folders\_sp))

{

#a=3

t2read="habitats.csv"

f2read=folders\_sp[[a]]

f2read

dat1=read.csv(paste(f2read,t2read, sep="/"))

dat1$code2=floor(as.numeric(substr(as.character(dat1$code), 1,3)))

#with(dat1, aggregate(code2, list(code=code), max))

first=function (x=x) x[1]

# to identify species with no codes in suitability field

spnocode=with(dat1, aggregate(suitability, list(internalTaxonId=internalTaxonId), lu))

spnocode$first=with(dat1, aggregate(suitability, list(internalTaxonId=internalTaxonId), first))$x

head(spnocode)

unique(spnocode$first)

unique(dat1$suitability)

spnocode\_list=spnocode$internalTaxonId[spnocode$x==1&(!spnocode$first%in%c("Suitable","Marginal","Unknown"))]

spnocode\_list

if (length(spnocode\_list)>0) dat1$suitability[dat1$internalTaxonId%in%spnocode\_list]="Suitable" # to simplify, the suitability of all habitats of these non-coded species with be considered "suitable" - so they will be only included if they have no other habitats

# forest

codes2use=c(1,3) # codes for habitat=forest

dat2=dat1[dat1$code2%in%codes2use,] # table with habitat species

unique(dat2$code)

unique(dat2$code2)

habitat1=unique(dat2$internalTaxonId[dat2$majorImportance=="Yes"]) ## list of habitat species for which major importance is true

habitatall=unique(dat2$internalTaxonId) ## list of all habitat species

habitat2=habitatall[which(!habitatall%in%habitat1)] # habitat species for which major importance is not true

head(dat2[dat2$internalTaxonId%in%habitat2,])

unique(dat2[dat2$internalTaxonId%in%habitat2,]$majorImportance)

dat3=dat1[!dat1$code2%in%codes2use,]#table with non-habitat species

dat3=dat3[dat3$suitability=="Suitable",]

unique(dat3$code)

sort(unique(dat3$code2))

othersp=unique(dat3$internalTaxonId) # species with other suitable habitats

habitat3=habitat2[which(!habitat2%in%othersp)] # habitat species with no other suitable habitats

head(dat1[dat1$internalTaxonId%in%habitat3,])

habitatf=c(habitat1,habitat3)

forest=c(forest, habitatf)

## wetland

codes2use=c(5,15) # codes for habitat=forest

dat2=dat1[dat1$code2%in%codes2use,] # table with habitat species

unique(dat2$code)

unique(dat2$code2)

habitat1=unique(dat2$internalTaxonId[dat2$majorImportance=="Yes"]) ## list of habitat species for which major importance is true

habitatall=unique(dat2$internalTaxonId) ## list of all habitat species

habitat2=habitatall[which(!habitatall%in%habitat1)] # habitat species for which major importance is not true

head(dat2[dat2$internalTaxonId%in%habitat2,])

unique(dat2[dat2$internalTaxonId%in%habitat2,]$majorImportance)

dat3=dat1[!dat1$code2%in%codes2use,]#table with non-habitat species

dat3=dat3[dat3$suitability=="Suitable",]

head(dat3)

unique(dat3$code)

sort(unique(dat3$code2))

othersp=unique(dat3$internalTaxonId) # species with other suitable habitats

habitat3=habitat2[which(!habitat2%in%othersp)] # habitat species with no other suitable habitats

head(dat1[dat1$internalTaxonId%in%habitat3,])

habitatf=c(habitat1,habitat3)

wetland=c(wetland, habitatf)

} # ends loop for forest and wetland species

tabsp1$forest=0

tabsp1$forest[tabsp1$TaxonID%in%forest]=1

tabsp1$wetland=0

tabsp1$wetland[tabsp1$TaxonID%in%wetland]=1

head(tabsp1[tabsp1$forest==1&tabsp1$Group=="Mammal",])

head(tabsp1[tabsp1$wetland==1&tabsp1$Group=="Mammal",])

## migratory

migr=NULL

for (a in 1:length(folders\_sp))

{

#a=3

t2read="all\_other\_fields.csv"

f2read=folders\_sp[[a]]

f2read

dat1=read.csv(paste(f2read,t2read, sep="/"))

head(dat1)

sp2extr1=dat1$internalTaxonId[dat1$MovementPatterns.pattern=="Full Migrant"]

head(dat1[dat1$MovementPatterns.pattern=="Full Migrant",])

migr=c(migr,sp2extr1)

}

tabsp1$migratory=0

tabsp1$migratory[tabsp1$TaxonID%in%migr]=1

head(tabsp1[tabsp1$migratory==1&tabsp1$Group=="Mammal",])

### marine terrestrial freshwater

marine=NULL

terrest=NULL

freshw=NULL

for (a in 1:length(folders\_sp))

{

#a=3

t2read="assessments.csv"

f2read=folders\_sp[[a]]

f2read

dat1=read.csv(paste(f2read,t2read, sep="/"))

head(dat1)

unique(dat1$systems)

sp2extr1=unique(dat1$internalTaxonId[dat1$systems=="Marine"])

marine=c(marine,sp2extr1)

sp2extr2=unique(dat1$internalTaxonId[dat1$systems=="Terrestrial"])

terrest=c(terrest,sp2extr2)

sp2extr3=unique(dat1$internalTaxonId[dat1$systems=="Freshwater (=Inland waters)"])

freshw=c(freshw,sp2extr3)

} # ends loop marine, terrestr, fresh

tabsp1$marine=0

tabsp1$marine[tabsp1$TaxonID%in%marine]=1

tabsp1$terrestrial=0

tabsp1$terrestrial[tabsp1$TaxonID%in%terrest]=1

tabsp1$freshwater=0

tabsp1$freshwater[tabsp1$TaxonID%in%freshw]=1

head(tabsp1[tabsp1$marine==1&tabsp1$Group=="Mammal",])

head(tabsp1[tabsp1$terrestrial==1&tabsp1$Group=="Mammal",])

head(tabsp1[tabsp1$freshwater==1&tabsp1$Group=="Mammal",])

if (other\_species\_lists){

othspecies=unique(other\_groups$Group\_species)

for (a in 1:length(othspecies))

{

#a=1

extr1=othspecies[a]

extr1

sp2extr1=other\_groups$TaxonID[other\_groups$Group\_species==extr1]

tabsp1$newvar=0

tabsp1$newvar[tabsp1$TaxonID%in%sp2extr1]=1

names(tabsp1)[which(names(tabsp1)=="newvar")]=as.character(extr1)

head(tabsp1)

}}

head(tabsp1)

### prepare gen change table

nrow(gench1)

gench2=merge(gench1, gcc) # to add the group and years of assessment to the table of genuine changes

nrow(gench2) # should be the same as previous

gench2$cat\_start=gench2$periodStartCategory

gench2$cat\_end=gench2$periodEndCategory

if(!"TaxonID" %in% names(gench2)) gench2$TaxonID=gench2$taxonid

gench2=merge(gench2, tabsp1[,c("TaxonID","Scientific")]) ## to add the names of the species to the table

head(gench2)

nrow(gench2) ## if nrow is different from above - species are missing in the full list

unique(gench2$Group)

### identify which entries don't have a primary driver, but have a major secondary

gench3=data.frame(gench2[,which(names(gench2)%in%list\_drivers)])

head(gench3)

sp2check\_list1=NULL

sp2check\_list2=NULL

for (j in 1:length(gench2)) sp2check\_list1=c(sp2check\_list1,which(gench2[,j]=="Most Important Secondary",))

for (j in 1:length(gench2)) sp2check\_list2=c(sp2check\_list2,which(gench2[,j]=="Primary",))

sp2check\_list1=sort(unique(sp2check\_list1))

sp2check\_list2=sort(unique(sp2check\_list2))

sp2check=unique(sp2check\_list1[which(!sp2check\_list1%in%sp2check\_list2)]) # entries that don't have any primary driver but have a "Most Important Secondary"

length(sp2check)

gench3[sp2check,]

genchF=gench2

for (n in 1:length(sp2check)){

for (j in 1:length(gench3)) if (genchF[sp2check[n],which(names(genchF)%in%list\_drivers[j])] =="Most Important Secondary") genchF[sp2check[n],which(names(genchF)%in%list\_drivers[j])] ="Primary" # replace the "Most Important Secondary" to "Primary"

}

genchF[sp2check,]

nrow(genchF)

nrow(gench1)

wts=data.frame(iucn=c("LC","NT","VU","EN","CR","CR(PE)","CR(PEW)","EW","EX","DD"), wg=c(0,1,2,3,4,5,5,5,5,-1)) # table with the "weights" of each iucn class (for the rli calculations)

### correct possible errors in cat\_latest in tabsp1 table

correct\_cats=with(genchF, data.frame(TaxonID=TaxonID, cat\_end\_GC=cat\_end,year\_Cat=end\_year))

comptables=merge(tabsp1,correct\_cats)

comptables=merge(comptables, with(wts, data.frame(Cat\_latest=iucn, wg\_original=wg)))

comptables=merge(comptables, with(wts, data.frame(cat\_end\_GC=iucn, wg\_corrected=wg)))

head(comptables)

comptables$dif=comptables$wg\_original-comptables$wg\_corrected

comptables=comptables[comptables$dif!=0,]

nrow(comptables)

for (a in 1:nrow(comptables)) tabsp1$Cat\_latest[tabsp1$Scientific==comptables$Scientific[a]]=as.character(comptables$cat\_end\_GC[a])

#write.csv(tabsp1, "tabsp1\_corrected.csv", row.names=F)

####### preparation of tables for the analysis

nrow(sppp) ### check number of rows in table with the pps sp-countries (sppp)

head(sppp) ### check check structure of table with the pps sp-countries

sppp1=merge(sppp,tabsp1[,c("Scientific","Group","Cat\_latest")], sort=F)

nrow(sppp1) ### check if the number of rows of sppp1 remained the same after the merge

head(sppp1) ### check if the structure of sppp1 remained the same after the merge - but with some new columns

sppp1$iucn=sppp1$Cat\_latest

####### ends of table preparation

#################### verify if all IUCN codes are correct

cds1=unique(tbf$iucn)

length(cds1)

length(cds1[cds1%in%wts$iucn]) ## should be the same as previous

cds1=unique(genchF$cat\_start)

length(cds1)

length(cds1[cds1%in%wts$iucn]) ## should be the same as previous

cds1=unique(genchF$cat\_end)

length(cds1)

length(cds1[cds1%in%wts$iucn]) ## should be the same as previous

#verify if all the scientific names in genuine file are correct and are in in sp-cnts table

sp1=unique(genchF$Scientific) #tabf$RLC #gench$cat\_start #gench$cat\_end

length(sp1)

length(sp1[sp1%in%tbf$Scientific])

sp1[!sp1%in%tbf$Scientific] ### species that are missing in the sp-cnts table after excluding the DD (but are in the genuine file)

sprich=with(sppp1, aggregate(Scientific, list(group=Group), lu))

names(sprich)[length(sprich)]="rich"

sprich ### number of species per taxon

years=min(genchF$start\_year):maxyear # years to consider in the analyses: from the first assessment of all until current

years

# PART 3. CUSTOM FUNCTIONS ###########################################################################

########################################################################################################

first=function (x=x) x[1] ## function to output the first value of a vector

q95=function (x=x) quantile(x,.95) ## function to estimate the quantile 95%

q05=function (x=x) quantile(x,.05) ## function to estimate the quantile 5%

div0=function (x=x, y=y) trunc((x-1)/y)-((x-1)/y) ## function that outputs 0 if y-1 is divisible by x (aux open new devices in graphics outputs)

cinz=rgb(217/255,217/255,217/255,.4) ### grey transparent colour for the plots (in rgb)

################ function to calculate global RLI

# nreps=number of repetitions; groups2=list of taxonomic groups to analyse; leftextrapol=if slope should be extrapolated to the left; saveleftslope=if left slope should be saved; plotit=if each iteraction should be plotted

rlicalc=function (nreps=repetitions, groups2=groups2, years=years, leftextrapol=T, saveleftslope=T, plotit=F){

# create table with the desired structure

rliTot=data.frame(group=rep(rep(groups2, each=length(years)),length(nreps)), year=rep(rep(years, length(groups2)), length(nreps)), nrep=rep(1:nreps,each=length(years)\*length(groups2)), rli=NA)

outpts=list() # creates empty list to store outputs generated within each iteration of the following loop

if (saveleftslope) leftslopes=NULL

for (r in 1:nreps) ### loop to go through all repetitions defined above

{

#r=1

for (g in 1:length(groups2)) #loop to go through each taxonomic group

{

#g=1

group=as.character(groups2[g]) # selects the group g

#group="Bird"

rlig=data.frame()

group

tgroup=pastables[pastables$group==group,] ## past table (see below) for group g

if (altpasttable & ((r/2)-floor(r/2))==0) tgroup=pastables\_alternative[pastables\_alternative$group==group,] # alternative past tables, for disaggregated hald of the iterations for rli related to drivers of genuine changes

nrow(tgroup)

yearsa=sort(unique(tgroup$year)) # to create vector years of assessment of group g

yearsa # years of assessment of group g

### years of assessment

for (y in 1:length(yearsa)) ## loop to go through each year of assessment of group g

{

#y=1

year=yearsa[y] # to select year y

year

tb=tgroup[tgroup$year==year,] # assessment of group g in year y

tb=merge(tb,wts) # merge with table with the weight of each iucn class

head(tb) #check structure merged table

nrow(tb) #check number of rows of merged table

dds=tb$Scientific[tb$iucn=="DD"]

dds

head(tb[tb$iucn=="DD",])

allwgs=tb$wg[tb$wg>(-1)] ## pool of non-DD red list categories for this group

if (length(allwgs)==0) allwgs=0 ## if all species of this group are DD

if (length(dds)>0) ddr=sample(allwgs,length(dds))

#if (length(dds)>0) for (d in 1:length(dds)) tb$wg[tb$Scientific==dds[d]]=sample(allwgs,1) ## randomly selects a red list category for DD species, from the pool created above

sumwgs=sum(allwgs)

if (length(dds)>0) sumwgs=sum(allwgs)+sum(ddr)

rli1=1-(sumwgs/(wex\*nrow(tb))) ## calculation of red list index (rli) for year y

rli1 ##red list index (rli) for year of assessment y

rlig=rbind(rlig, data.frame(group=group,year=year, rli=rli1)) # saves the rli for group g and year y

} ## ends years of assessment

rlig # rli for all years of assessment of group g

### intrapolation and extrapolation (done within each group)

dat=rlig

dat

#interpolation

yearsti=min(dat$year): max(dat$year) # years between the first and last assessment - for interpolation

irli=approx(dat$year, dat$rli, xout=yearsti, method="linear", rule=2:2)$y ## linear interpolation of rli values based on existing years of assessment

irli # result of the interpolation

datf=data.frame(year=yearsti, rli=irli) # table with years of assessment and interpolated years and correspondent rlis for group g

datf

#with(datf, plot(year, rli, xlim=c(1975,2018), ylim=c(0.6,1.2))) # to plot the results

#extrapolation # y=a+bx

#left extrapolation

yearl=c(min(years)-2, min(years)-1, years[which(years<min(yearsti))]) ## vector with years for left extrapolation: since two years before the year of first assessment (of any group) and the year of first assessment of group g

yearl

x1=sort(dat$year)[1:2] ### first two years of assessment

x1

y1=c(dat$rli[dat$year==x1[1]],dat$rli[dat$year==x1[2]]) ### rli in first two years of assessment

y1 ###rlis

nv=lm(y1~x1) # linear model of the rate of change in rli (based on first two years of assessment) to estimate slope and intercept for extrapolation

mnslp <- coef(nv)[2] # slope of the model

useslp <- rnorm(1, mnslp, sd=slopecv\*abs(mnslp)) # random value of slope based on the slope from the linear model and a potential standard devidation (set in the begining of the code - "slopecv") - changes each repetition

if (leftextrapol==F) useslp <- meanleftslopes # for corals we don't estimate left slope the same way, but we use an average of the slopes of the other groups - so we set leftextrapol==F for corals (and run the code after the other groups)

irlel=y1[1] - (useslp\*(x1[1] -(yearl))) # estimates of rli in years before the first assessment, based on random values of slopes

if (leftextrapol) leftslopes=c(leftslopes,useslp) # to store the random value of slope generated in this iteration r - for future calculation to apply to corals

datf=rbind(datf, data.frame(year=yearl, rli=irlel)) ## adds left-extrapolated rlis to table datf

#with(datf, points(year, rli, col=2, pch=19, cex=.6))

datf

#right extrapolation - does exactly the same as previous, but to the years after the last assessment of group g until current year + 2

yearr=c(years[which(years>max(yearsti))],max(years)+1,max(years)+2) ## vector with years for right extrapolation: since year of latest assessment of group g until current year + 2

yearr

x1=sort(dat$year)[(nrow(dat)-1):nrow(dat)] #last two years of assessment

x1 ## years

y1=c(dat$rli[dat$year==x1[1]],dat$rli[dat$year==x1[2]]) ### rli in last two years of assessment

y1 ### rlis

nv=lm(y1~x1) # linear model of the rate of change in rli (based on last two years of assessment) to estimate slope and intercept for extrapolation

mnslp <- coef(nv)[2] # slope of the model

useslp <- rnorm(1, mnslp, sd=slopecv\*abs(mnslp)) # random value of slope based on the slope from the linear model and a potential standard devidation (set in the begining of the code - "slopecv") - changes each repetition

irler=y1[2] - (useslp\*(x1[2] -(yearr))) # estimates of rli in years after the last assessment, based on random values of slopes

datf=rbind(datf, data.frame(year=yearr, rli=irler)) ## adds right-extrapolated rlis to table datf

#with(datf, points(year, rli, col=2, pch=19, cex=.6))

datf=datf[order(datf$year),] # reorder the datf table by year

if (plotit) # plots the results if set to plot (i.e. if plotit=T, see lines above)

{

if (r<30) win.graph() # only plots for the first 30 iterations

with(datf, plot(year, rli, pch=19, col=2, main=paste(group,"s", sep="")))

with(datf[datf$year%in%yearsti,], points(year, rli, pch=19, col=4))

}

restmw=NULL

for (m in 3:(nrow(datf)-2)) restmw=c(restmw, sample(datf$rli[(m-2):(m+2)],1)) # moving average - considers a random rli from the interval of +- 2 years from each year - excludes the first and last two years in the final vector (but considers them for taking a random value)

finalmw=restmw

rliTot[rliTot$group==group&rliTot$nrep==r,]$rli=finalmw # stores the rli per year y for group g in repetition r as the result of the moving average

if (plotit) with(rliTot[rliTot$group==group&rliTot$nrep==r,], lines(year, rli, col=8)) # plots a line shwoing the values of the moving average if plotit=T

} #ends loop groups

} #ends loop nreps

outpts[[1]]=rliTot # stores the table with final rli value per year y, group g and repetition r in the list of outputs from this loop

if (saveleftslope) outpts[[2]]=mean(leftslopes) # stores the mean values of left slopes (for coral) in the list of outputs from this loop

return(outpts)

}

#### function to plot the rlis

plotrli=function(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=T, label=T){

difrgy=diff(range(basy, topy))

if (difrgy<miny) {

basy=topy-miny

difrgy=diff(range(basy, topy))}

with(ttuse,plot(year,rli, ylim=c(basy\*.99,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", yaxt="n", ylab="Red List Index of species survival"))

gs=unique(ttuse$group[ttuse$group!="aggregated"])

if(lu(gs)>0)

{

for (g in 1:length(gs))

{

#g=1

gs1=as.character(gs[g])

res2=ttuse[ttuse$group==gs1,]

with(res2,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

}

}

if (aggreg) with(ttuse[ttuse$group=="aggregated",], polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz)) #don't plot if don't want aggregated line

if(lu(gs)>0){

for (g in 1:length(gs))

{

gs1=as.character(gs[g])

res2=ttuse[ttuse$group==gs1,]

with(res2,lines(year,rli, col=2, lwd=2))

if (label) text(max(res2$year)+0.5, res2$rli[nrow(res2)],paste(gs1,"s", sep=""), cex=1, adj=c(0,0.5))

}}

if (aggreg) with(ttuse[ttuse$group=="aggregated",], lines(year,rli, col=4, lwd=2))

if (lu(gs)==1&gs[1]=="Bird"){

if (difrgy<0.3)

{

if (difrgy<0.1) {axis(2, seq(0,1, by=.002))} else axis(2, seq(0,1, by=.05))

}

if (difrgy>0.3) axis(2, seq(0,1, by=.1))

par(mgp=c(3.5, 1, 0))

title(xlab="Year")

}

else{

if (difrgy<0.3) axis(2, seq(0,1, by=.05))

if (difrgy>0.3) axis(2, seq(0,1, by=.1))

}

}

# PART 4. RUN GLOBAL ANALYSES AND PREPARE PAST TABLES ("back-casting") ##############################

######################################################################################################

mink=1

if (only\_thematic) mink=2

mink

for (k in mink:n\_iter) #### loop to go through all the disaggregations

{

#k=2

### k=1 runs the non-thematic RLIs (all species and genuine changes)

if (k ==1){

altpasttable=F # if to build and use an alternative pastable, to use in 50% of the iterations; only valid for dissagregations based on drivers of genuine changes (to deal with secondary drivers)

gench=genchF

tbf1=sppp1

print(k)

name2save="general"

finfolder=paste(finfolderP,"General", sep="/")

print(finfolder)

}

##iterations corresponding to the drivers of genuine changes

if (length(driv\_gc)>0){

if (k>1&k<(length(driv\_gc)+2))

{

altpasttable=T # if to build and use an alternative pastable, to use in 50% of the iterations; only valid for dissagregations based on drivers of genuine changes (to deal with secondary drivers)

driv=driv\_gc[[(k-1)]]

driv

driv\_name=names(driv\_gc)[[k-1]]

driv\_name

name2save=driv\_name

genchT=data.frame(genchF[,which(names(genchF)%in%driv)])

head(genchT)

length(genchT)

gc2use\_list=NULL

gc2use\_list\_altern=NULL

for (j in 1:length(genchT)) gc2use\_list=c(gc2use\_list,which(genchT[,j]%in%c("Primary","Most Important Secondary")))

for (j in 1:length(genchT)) gc2use\_list\_altern=c(gc2use\_list\_altern,which(genchT[,j]%in%c("Primary","Most Important Secondary","Secondary")))

length(gc2use\_list)

length(gc2use\_list\_altern)

gc2use\_list=sort(unique(gc2use\_list))

gc2use\_list\_altern=sort(unique(gc2use\_list\_altern))

gench=genchF[gc2use\_list,]

gench\_alt=genchF[gc2use\_list\_altern,]

head(gench)

nrow(gench)

#head(gench\_alt)

nrow(gench\_alt)

tbf1=sppp1

finfolder=paste(finfolderP,"Thematic",name2save, sep="/")

finfolder

print(k)

print(finfolder)

}}

##iterations corresponding to the groups of species

if (length(splists)>0){

if (k>(length(driv\_gc)+1)&k<(length(driv\_gc)+length(splists)+2))

{

altpasttable=F # if to build and use an alternative pastable, to use in 50% of the iterations; only valid for dissagregations based on drivers of genuine changes (to deal with secondary drivers)

splist=splists[(k-1-length(driv\_gc))]

splist

name2save=splist

gench=genchF

sp2use=tabsp1$Scientific[which(tabsp1[,which(names(tabsp1)==splist)]==1)]

tbf1=sppp1[sppp1$Scientific%in%sp2use,]

head(tbf1)

lu(tbf1$Scientific)

finfolder=paste(finfolderP,"Thematic",name2save, sep="/")

print(k)

print(finfolder)

}}

if (k>(length(driv\_gc)+length(splists)+1)) ##iterations corresponding to combinations of drivers and groups of species

{

altpasttable=T # if to build and use an alternative pastable, to use in 50% of the iterations; only valid for dissagregations based on drivers of genuine changes (to deal with secondary drivers)

comb=combs[(k-1-length(driv\_gc)-length(splists)),]

comb

driv\_name=comb[,1]

driv\_name

driv=driv\_gc[[which(names(driv\_gc)==driv\_name)]]

driv

genchT=data.frame(genchF[,which(names(genchF)%in%driv)])

head(genchT)

length(genchT)

gc2use\_list=NULL

gc2use\_list\_altern=NULL

for (j in 1:length(genchT)) gc2use\_list=c(gc2use\_list,which(genchT[,j]%in%c("Primary","Most Important Secondary")))

for (j in 1:length(genchT)) gc2use\_list\_altern=c(gc2use\_list\_altern,which(genchT[,j]%in%c("Primary","Most Important Secondary","Secondary")))

length(gc2use\_list)

gc2use\_list=sort(unique(gc2use\_list))

length(gc2use\_list)

gench=genchF[gc2use\_list,]

length(gc2use\_list\_altern)

gc2use\_list=sort(unique(gc2use\_list))

gc2use\_list\_altern=sort(unique(gc2use\_list\_altern))

gench\_alt=genchF[gc2use\_list\_altern,]

head(gench)

nrow(gench)

#head(gench\_alt)

nrow(gench\_alt)

splist=comb[,2]

splist

sp2use=tabsp1$Scientific[which(tabsp1[,which(names(tabsp1)==splist)]==1)]

tbf1=sppp1[sppp1$Scientific%in%sp2use,]

head(tbf1)

lu(tbf1$Scientific)

name2save=paste(driv\_name,splist, sep="\_")

finfolder=paste(finfolderP,"Thematic",name2save, sep="/")

finfolder

print(k)

print(finfolder)

}

groups=unique(tbf1$Group) # taxonomic groups to consider in the analyses

groups

# 4.1. reconstruct past tables - tables with all RL categories for each species in each year of assessment #################

pastables=data.frame()

for (g in 1:length(groups)) ### loop to run the code for each taxonomico group of speices

{

#g=1

group=as.character(groups[g]) ### to choose the group to analyse

#group="Bird"

group

tbgn=gench[gench$Group==group,] ## table with relevant genuine changes for this group and disaggregation

tbgn2=genchF[genchF$Group==group,] ## same as previous but all genuine changes for this group - to extract the years of assessment

tball=tbf1[tbf1$Group==group,] ### pp sp-cnts for this group

nrow(tbgn) # to check the number of genuine changes in this group

head(tbgn) # structure of table with genuine changes for this group

yearsa=sort(unique(c(tbgn2$start\_year,tbgn2$end\_year)))

yearsa # years of RL assessments for this group

yearsa2=yearsa[1:length(yearsa)-1]

yearsa2 # years of RL assessments for this group (excluding the last one)

tabsp=with(tabsp1[tabsp1$Scientific%in%tball$Scientific,], data.frame(Scientific=Scientific, iucn=Cat\_latest)) ## simplified table with the list of species of this group and latest iucn class

tabsp$year=max(yearsa) ## field with the max year of assessment of the group (equal to all species)

head(tabsp) ## check structure of table tabsp

sp1=unique(tball$Scientific) ## list of the species in this group

length(sp1) ## number of species in this group

tabsp2=tabsp # starting table with the latest iucn class - will be complemented with the status of the same species in the previous assessments in the next loop

tabspnew=tabsp # reference table with the list of species, iucn class and latest year of assessment

for (y in length(yearsa2):1) # loop to go through all the years of assessments "y" except the last (starting by the last)

{

#y=6

tabsp3=tabspnew # reference table with the list of species, iucn class and latest year of assessment (starting with the most recent one in the first loop)

tabsp3$year=yearsa2[y] # changes year to year of the assessment y

t2ch=tbgn[tbgn$start\_year==yearsa2[y],c("Scientific","cat\_start")] # simplified table of genuine changes in year y

head(t2ch) # check structure of table

nrow(t2ch) # number of genuine changes in year y

for (z in 1:nrow(t2ch)) # loop to go through all the genuine changes in year y

{

#z=1

tabsp3$iucn[tabsp3$Scientific==as.character(t2ch$Scientific[z])]=as.character(t2ch$cat\_start[z]) ## for species with genuine changes in year y, change table built above to the iucn class in that year y

}

tabspnew=tabsp3 # changes the reference table to account for the changes occured in year y - would feed the next iteration for previous year of assessment

tabsp2=rbind(tabsp2,tabsp3) ## adds iucn class of all sp in this group in year of assessment y to the starting table

}

lu(yearsa) ## number of years of comprehensive assessments

nrow(tabsp2)/lu(tabsp2$Scientific) ## should be the number of years of comprehensive assessments (i.e., same as previous)

tabsp2$group=group ## addign the name of the group to the table just created

pastables=rbind(pastables,tabsp2) ## merge all tables for different groups in a single table

}

if (altpasttable) {

pastables\_alternative=data.frame()

for (g in 1:length(groups)) ### loop to run the code for each taxonomico group of speices

{

#g=1

group=as.character(groups[g]) ### to choose the group to analyse

#group="Mammal"

group

tbgn=gench\_alt[gench\_alt$Group==group,] ## table with relevant genuine changes for this group and disaggregation

tbgn2=genchF[genchF$Group==group,] ## same as previous but all genuine changes for this group - to extract the years of assessment

tball=tbf1[tbf1$Group==group,] ### pp sp-cnts for this group

nrow(tbgn) # to check the number of genuine changes in this group

head(tbgn) # structure of table with genuine changes for this group

yearsa=sort(unique(c(tbgn2$start\_year,tbgn2$end\_year)))

yearsa # years of RL assessments for this group

yearsa2=yearsa[1:length(yearsa)-1]

yearsa2 # years of RL assessments for this group (excluding the last one)

tabsp=with(tabsp1[tabsp1$Scientific%in%tball$Scientific,], data.frame(Scientific=Scientific, iucn=Cat\_latest)) ## simplified table with the list of species of this group and latest iucn class

tabsp$year=max(yearsa) ## field with the max year of assessment of the group (equal to all species)

head(tabsp) ## check structure of table tabsp

sp1=unique(tball$Scientific) ## list of the species in this group

length(sp1) ## number of species in this group

tabsp2=tabsp # starting table with the latest iucn class - will be complemented with the status of the same species in the previous assessments in the next loop

tabspnew=tabsp # reference table with the list of species, iucn class and latest year of assessment

for (y in length(yearsa2):1) # loop to go through all the years of assessments "y" except the last (starting by the last)

{

#y=6

tabsp3=tabspnew # reference table with the list of species, iucn class and latest year of assessment (starting with the most recent one in the first loop)

tabsp3$year=yearsa2[y] # changes year to year of the assessment y

t2ch=tbgn[tbgn$start\_year==yearsa2[y],c("Scientific","cat\_start")] # simplified table of genuine changes in year y

head(t2ch) # check structure of table

nrow(t2ch) # number of genuine changes in year y

for (z in 1:nrow(t2ch)) # loop to go through all the genuine changes in year y

{

#z=1

tabsp3$iucn[tabsp3$Scientific==as.character(t2ch$Scientific[z])]=as.character(t2ch$cat\_start[z]) ## for species with genuine changes in year y, change table built above to the iucn class in that year y

}

tabspnew=tabsp3 # changes the reference table to account for the changes occured in year y - would feed the next iteration for previous year of assessment

tabsp2=rbind(tabsp2,tabsp3) ## adds iucn class of all sp in this group in year of assessment y to the starting table

}

lu(yearsa) ## number of years of comprehensive assessments

nrow(tabsp2)/lu(tabsp2$Scientific) ## should be the number of years of comprehensive assessments (i.e., same as previous)

tabsp2$group=group ## addign the name of the group to the table just created

pastables\_alternative=rbind(pastables\_alternative,tabsp2) ## merge all tables for different groups in a single table

}

} # ends alternative past tables

#write.csv(pastables\_alternative, "pastables\_alternative\_2020.csv", row.names=F) ## to save past tables if needed

#write.csv(pastables, "pastables\_2019.csv", row.names=F) ## to save past tables if needed

#pastables=read.csv("pastables.csv") ## to read existing past tables

wex=wts$wg[wts$iucn=="EX"] ## "weight" of extint species

head(pastables) # check the structure of past tables

tail(pastables) # check the structure of past tables

# 4.2 global RLI

# this step also estimates left slope for CORALS (left slope for corals is calculated separately - cannot use steep recent declines for past extrapolation)

if ("Coral"%in%groups) groupsf=c(as.character(groups[groups!="Coral"]),"Coral") # to ensure that Corals will be the last group to run (to use the mean slope for left extrapolation)

tt=proc.time() ## to calculate the time it takes to run

### non-corals

groups2=as.character(groups[groups!="Coral"]) ## to exclude corals from the next calculations (left slope is calculated in a different way)

rliTotGlobal\_noncorals\_list=rlicalc(nreps=repetitions, groups2=groups2, years=years, leftextrapol=T, saveleftslope=T, plotit=F) # calculates global rlis for all groups except corals

rliTotGlobal\_noncorals=rliTotGlobal\_noncorals\_list[[1]] # table resulting from global rli calculations for all groups except corals (see code above)

head(rliTotGlobal\_noncorals)

tail(rliTotGlobal\_noncorals)

meanleftslopes=rliTotGlobal\_noncorals\_list[[2]] # left slope estimate resulting from global rli calculations - to apply to corals (see code above)

meanleftslopes

### corals

if ("Coral"%in%groups){

rliTotGlobal\_corals\_list=rlicalc(nreps=repetitions, groups2="Coral", years=years, leftextrapol=F, saveleftslope=F, plotit=F) # calculates global rlis for corals (using left slope for extrapolation based on average of other groups)

rliTotGlobal\_corals=rliTotGlobal\_corals\_list[[1]] # table resulting from global rli calculations for corals (see code above)

head(rliTotGlobal\_corals)

tail(rliTotGlobal\_corals)

}

##### combines results for corals and other groups

rliTotGlobal=rliTotGlobal\_noncorals

if ("Coral"%in%groups) rliTotGlobal=rbind(rliTotGlobal\_noncorals,rliTotGlobal\_corals)

### aggregated rli

rlitotagrs=with(rliTotGlobal, aggregate(rli,list(year=year, nrep=nrep), mean)) # estimates aggregated rlis per year and per repetition as the average across groups

names(rlitotagrs)[length(rlitotagrs)]="rli"

rlitotagrs$group="aggregated"

head(rlitotagrs)

## table with rli for each group and aggregated

rliTotF=rbind(rliTotGlobal,rlitotagrs[,c("group","year","nrep","rli")]) ## combines rli for each group and aggregated in a single table

head(rliTotF)

tail(rliTotF)

max(rliTotF$nrep)# number of repetitions

max(rliTotF$rli) # maximum rli value

### final table with the median rli and 95% intervals (based on the repetitions) per year and group

resg=with(rliTotF, aggregate(rli, list(year=year, group=group), length)) # n repetitions

resg$rli=with(rliTotF, aggregate(rli, list(year=year, group=group), median))$x # median rli

resg$qn95=with(rliTotF, aggregate(rli, list(year=year, group=group), q95))$x # quantile 95%

resg$qn05=with(rliTotF, aggregate(rli, list(year=year, group=group), q05))$x # quantile 05%

resg

gs=unique(resg$group)

# to restructure the final table with the rli to use only the years between the first and last assessment of each group (for aggregated rli, starts 10 years before the first year of assessment of the latest group starting being assessed)

resgf=data.frame()

for (g in 1:lu(gs))

{

#g=1

gs1=as.character(gs[g])

gs1

if (gs1!="aggregated") yrs=min(genchF$start\_year[genchF$Group==gs1]):max(genchF$end\_year[genchF$Group==gs1]) else yrs=(max(with(genchF, aggregate(start\_year, list(Group), min))$x)-10):maxyear

rli3=resg[resg$group==gs1&resg$year%in%yrs,]

rli3=rli3[order(rli3$year),]

resgf=rbind(resgf,rli3)

}

head(resgf,20)

## to plot the global rlis

tab2plot=resgf

par(las=1)

with(tab2plot,plot(year,rli, ylim=c(min(qn05), max(qn95)), xlim=c(min(year), max(year)+4), type="n", xlab="Year", ylab="Red List Index of species survival"))

## plots first the error area

gs=unique(tab2plot$group[tab2plot$group!="aggregated"])

lu(gs)

for (g in 1:length(gs))

{

#g=1

gs1=as.character(gs[g])

res2=tab2plot[tab2plot$group==gs1,]

with(res2,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

}

with(tab2plot[tab2plot$group=="aggregated",], polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

### add the lines to the plot

for (g in 1:length(gs))

{

gs1=as.character(gs[g])

res2=tab2plot[tab2plot$group==gs1,]

with(res2,lines(year,rli, col=2, lwd=2))

text(max(res2$year)+0.5, res2$rli[nrow(res2)],paste(gs1,"s", sep=""), cex=.6, adj=c(0,0.5))

}

with(tab2plot[tab2plot$group=="aggregated",], lines(year,rli, col=4, lwd=2))

#abline(v=2015, lty=3, col=8)

#write.csv(resgf, "global\_RLIs.csv", row.names=F)

#pastables=read.csv("pastables.csv")

#meanleftslopes=(-0.0009302144)

#####

# 4.3 exports csv, pdf and png with global rlis

rlinew=resgf

final2save=rlinew[, c("group","year","rli","qn05","qn95")]

## 4.3.1 to save the global csvs

finfolderF=paste(finfolder, "/", "global/csv\_tables", sep="")

finfolderF

if (saveglobal){

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

finalname=paste(finfolderF, "/", "global\_all\_taxa\_", name2save,".csv", sep="")

finalname

write.csv(final2save,finalname, row.names=F)

final2save2=rlinew[rlinew$group=="aggregated", c("group","year","rli","qn05","qn95")]

finalname=paste(finfolderF, "/", "global\_aggregated\_", name2save,".csv", sep="") #edit to \_wo\_agg to plot without aggregated line

finalname

write.csv(final2save2,finalname, row.names=F)

}

## 4.3.2 to save the global pdfs

## all taxa and aggregated

if (saveglobal) {

if (pdfs) {

finfolderF=paste(finfolder, "/", "global/pdfs", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

#graph all taxa and aggregated

pdf1name=paste(finfolderF, "/","global","\_all\_taxa\_",name2save,".pdf", sep="")

pdf1name

pdf(file=pdf1name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=T)

dev.off()

#graph all taxa withouth aggregated

pdf1name=paste(finfolderF, "/","global","\_all\_taxa\_without\_aggr\_",name2save, ".pdf", sep="")

pdf1name

pdf(file=pdf1name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=F)

dev.off()

#graph aggregated

pdf2name=paste(finfolderF,"/", "global","\_aggregated\_",name2save, ".pdf", sep="")

pdf2name

pdf(file=pdf2name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="aggregated",], aggreg=T)

dev.off()

# graph birds

pdf3name=paste(finfolderF,"/", "global","\_birds\_",name2save, ".pdf", sep="")

pdf3name

pdf(file=pdf3name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,7,4,2), mgp=c(5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="Bird",], aggreg=T, label=F)

dev.off()

} ### ends condition pdfs=T

} # ends condition saveglobal=T

## 4.3.3 to save the global pngs

if (saveglobal) {

if (pngs) {

finfolderF=paste(finfolder, "/", "global/pngs", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF)

### png all taxa with aggregated

png1name=paste(finfolderF, "/","global","\_all\_taxa\_",name2save, ".png", sep="")

png1name

png(file=png1name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=T)

dev.off()

#graph all PNG without aggregated line (commented out)

png1name=paste(finfolderF, "/","global","\_all\_taxa\_without\_aggr\_",name2save, ".png", sep="")

png1name

png(file=png1name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=F, label=T) ### all without aggregated

dev.off()

#graph aggregated png

png2name=paste(finfolderF,"/", "global","\_aggregated\_",name2save, ".png", sep="") #edit to \_wo\_agg to plot without aggregated line

png2name

png(file=png2name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="aggregated",], aggreg=T) # only aggregated

dev.off()

# graph birds PNG

png3name=paste(finfolderF,"/", "global","\_birds\_",name2save, ".png", sep="")

png3name

png(file=png3name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,7,4,2), mgp=c(5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="Bird",], aggreg=T, label=F) # only birds

dev.off()

} ### ends condition pngs=T

} # ends condition saveglobal=T

# PART 5. RUN ANALYSES PER REGION/COUNTRY #############################################

#############################################################################################

# 5.1 run analyses per region/country (i.e., combinations of isos)

if (k==1 | dis\_subglobal){

if (length(isos2use)>0) { ### will only run if there's any isos2use set

for (f in 1:length(isos2use)) ## ### starts the loop to go through all the combinations if isos (countries or regions, etc - defined in isos2use vector at the start)

{

#f=1

iso2use=isos2use[f]

iso2use

isos$isof=isos[,which(names(isos)==iso2use)]

head(isos) ### check structure of isos table

# check if ISOs in sp-cnt table and isos tables are the same

lu(isos$isof) ### check number of unique iso codes (or groups, regions) in isos table

## verify if the number of iso codes in pps sp-countries is the same as in isos

lu(tbf1$ISO3) ##number of unique iso codes in table with the pps sp-countries (should be the same or one less - ABNJ - comparing with the isos)

isos$ISO3[which(!isos$isof%in%tbf1$ISO3)] # confirm that the difference in isos between tables (isos and sp-countries-pp) is due to the ABNJ; only valid for global RLIs

sort(unique(tbf1$ISO3[which(!tbf1$ISO3%in%isos$ISO3)])) # confirm that the difference in isos between tables (isos and sp-countries-pp) is due to the ABNJ; only valid for global RLIs

#merge tbf1 with isos

nrow(tbf1) ### check number of rows in table with the pps sp-countries (tbf1)

head(tbf1) ### check check structure of table with the pps sp-countries

tbf2=merge(tbf1, isos[,c("ISO3","isof")],by="ISO3",all.x=T)

nrow(tbf2) ### check number of rows in table with the pps sp-countries (tbf1) after the merge

head(tbf2) ### check check structure of table with the pps sp-countries after the merge

nreps=repetitions

plotit=F

plotit2=T

finalrlis=data.frame()

cts=unique(tbf2$isof) ### group of countries, regions or subregions (set in isof) - hereafter "cts"

lu(cts)

cts=as.character(cts)

cts[is.na(cts)]="0"

cts=cts[cts!="0"]

cts

torun=cts

others=NULL

##### to check differences between ISO\_BL and ISO\_SDG

if ("ISO\_BL"%in%isos2use & "ISO\_SDG"%in%isos2use & iso2use=="ISO\_SDG"){

difs=NULL

for (p in 1:nrow(isos))

{

#p=1

#p=3

dif1=grep(pattern=as.character(isos$ISO\_BL[p]), as.character(isos$ISO\_SDG[p])) # check the match between the iso\_BL code and iso\_SDG code for each iso3; 1 if there's a match; NULL if not

dif1

if(length(dif1)==0) dif1=2 ## attributes a value of 2 if there's no match

difs=c(difs, dif1)

}

difs=difs-1 # change vector of differences to: 1=different iso codes, 2=same iso codes

length(difs)

nrow(isos)

sum(difs) # number of isos that differ

head(isos,20)

difisos=sort(unique(isos$ISO\_SDG[difs==1]))

torun=unique(tbf2$isof)[unique(tbf2$isof)%in%difisos]

others=unique(tbf2$isof)[!unique(tbf2$isof)%in%difisos]

}

torun

others

tt=proc.time()

#SDG Diff to BL: ('FIN','GBR','FRA','NOR','AUS','CHN','USA')

if (length(cts)>0){ # if there's any cnt to run in this iso2use (for disaggregated RLIs can happen that there's no country to run, e.g. if all endemics of non-LLDC and non-SIDS countries)

for (x in 1:length(cts)) #length(cts) ### loop to go through all cts

#for (x in 1:4)

{

#x=1

#x=which(cts=="ARE")

cnt=cts[x] # to set the cts to be analysed in this iteraction

cnt

print(paste(x, cnt))

if (cnt %in%torun) {

tbct=tbf2[tbf2$isof==cnt,] ## table with the list of species of this cnt and respective pp of the distribution

# to calculate the number of species and genuine changes in this country/region - to save in the final csv file

nspecies=with(tbct, aggregate(Scientific, list(group=group),lu))

names(nspecies)[length(nspecies)]="n\_sp"

nspecies

genchtable=gench[gench$Scientific%in%tbct$Scientific,]

if (nrow(genchtable)>0){

ngenchanges=with(genchtable, aggregate(Scientific, list(group=Group),lu))

names(ngenchanges)[length(ngenchanges)]="n\_gen\_changes"

ngenchanges

nspecies=merge(nspecies, ngenchanges, all.x=T, all.y=T)

nspecies$n\_gen\_changes[is.na(nspecies$n\_gen\_changes)]=0

}

if (nrow(genchtable)==0) nspecies$n\_gen\_changes=0

nspecies

nspecies=rbind(nspecies, data.frame(group="aggregated", n\_sp=sum(nspecies$n\_sp), n\_gen\_changes=sum(nspecies$n\_gen\_changes)))

nspecies

nspecies$ordem=1:nrow(nspecies)

#unique groups

groupsf=as.character(unique(tbct$Group)) # unique groups existing in this cnt (not all taxonomic groups exist in all cnts)

groupsf<-na.omit(groupsf)

groupsf

## code to calculate the rli for this cnt - see similar code above for notes ("function to calculate global RLI"); main differences commented (related with the adaptation for taking into consideration the pp of each species in each cnt)

rliext=data.frame(country=cnt, group=rep(rep(groupsf, each=length(years)),nreps), year=rep(rep(years, length(groupsf)),nreps), nrep=rep(1:nreps,each=length(years)\*length(groupsf)), rli=NA)

head(rliext)

tail(rliext)

for (r in 1: nreps) #nreps

{

#r=1

print(paste(x, cnt, r))

for (g in 1:length(groupsf)) ### note that loop of groups has to be the first, to save the slopes to apply to corals

{

#g=1

group=as.character(groupsf[g])

group

spcnt=tbct[tbct$Group==group,]

lu(spcnt$Scientific)

nrow(spcnt)

spcnt2=with(spcnt, aggregate(pp, list(Scientific=Scientific), sum)) ## sum pp for species and country (could be split for marine/terrestrial or disjucnt polygons in country/eez layer with multiple entries)

nrow(spcnt2)

names(spcnt2)[length(spcnt2)]="pp"

head(spcnt2)

#max(spcnt2$pp)

spcnt2[spcnt2$pp==1,] ## endemic species

spcnt3=merge(spcnt2,pastables)

if (altpasttable & ((r/2)-floor(r/2))==0) spcnt3=merge(spcnt2,pastables\_alternative) # to alternative past tables in half of the iterations, for disaggregated rli related to drivers of genuine changes

head(spcnt3)

tail(spcnt3)

nrow(spcnt3)/nrow(spcnt2) ## should be the number of years of assessment for this group

yearsa=sort(unique(spcnt3$year))

yearsa

yearsa2=yearsa[1:length(yearsa)-1]

yearsa2

### years of assessment

rlig=data.frame()

for (y in 1:length(yearsa))

{

#y=1

year=yearsa[y]

year

tb=spcnt3[spcnt3$year==year,]

head(tb)

nrow(tb)

unique(tb$iucn)

tb[tb$iucn=="DD",]

tb[!tb$iucn%in%c("DD", "LC"),]

tb[tb$iucn=="LC",]

tb=merge(tb,wts)

head(tb)

nrow(tb)

dds=tb$Scientific[tb$iucn=="DD"]

dds

head(tb[tb$iucn=="DD",])

allwgs=tb$wg[tb$wg>(-1)]

if (length(allwgs)==0) allwgs=0 ## if all species of this group in this cnt are DD

if (length(dds)>0) {

tbndd=tb[tb$iucn!="DD",]

tbdd=tb[tb$iucn=="DD",]

tbdd$wg=sample(allwgs,nrow(tbdd), replace=T)

head(tbdd)

tb=rbind(tbndd, tbdd)

#for (d in 1:length(dds)) tb$wg[tb$Scientific==dds[d]]=sample(allwgs,1) # to replace the DD species by another category, randomly selected by the pool of other species

}

#tb[tb$iucn=="DD",]

tb$pp<-unlist(tb$pp)

tb$wxpp=tb$wg\*tb$pp ## weight of the change multiplied by the pp of the species in cnt

head(tb)

sum(tb$wxpp)

wex\*sum(tb$pp)

sum(tb$wxpp)/(wex\*sum(tb$pp))

1-round(sum(tb$wxpp)/(wex\*sum(tb$pp)),7)

rli1=1-round((sum(tb$wxpp)/(wex\*sum(tb$pp))),7)

rli1

crli=round((sum(tb$wxpp)/(wex\*sprich$rich[sprich$group==group])),7)

rlig=rbind(rlig, data.frame(group=group,year=year, rli=rli1))

if (y==1) tb1=tb

} ## ends loop year

rlig

dat=rlig

dat

#interpolation

yearsti=min(dat$year): max(dat$year)

irli=approx(dat$year, dat$rli, xout=yearsti, method="linear", rule=2:2)$y

irli

datf=data.frame(year=yearsti, rli=irli)

datf

#with(datf, plot(year, rli, xlim=c(1975,2018), ylim=c(0.6,1.2)))

#extrapolation # y=a+bx

#left

yearl=c(min(years)-2, min(years)-1, years[which(years<min(yearsti))])

yearl

x1=sort(dat$year)[1:2]

x1 ### years

y1=c(dat$rli[dat$year==x1[1]],dat$rli[dat$year==x1[2]])

y1 ###rlis

nv=lm(y1~x1)

mnslp <- coef(nv)[2]

useslp <- rnorm(1, mnslp, sd=slopecv\*abs(mnslp))

if (group=="Coral") useslp <- meanleftslopes

irlel=y1[1] - (useslp\*(x1[1] -(yearl)))

datf=rbind(datf, data.frame(year=yearl, rli=irlel))

datf

#right

yearr=c(years[which(years>max(yearsti))],max(years)+1,max(years)+2)

yearr

x1=sort(dat$year)[(nrow(dat)-1):nrow(dat)]

x1 ## years

y1=c(dat$rli[dat$year==x1[1]],dat$rli[dat$year==x1[2]])

y1 ### rlis

nv=lm(y1~x1)

mnslp <- coef(nv)[2]

useslp <- rnorm(1, mnslp, sd=slopecv\*abs(mnslp))

irler=y1[2] - (useslp\*(x1[2] -(yearr)))

datf=rbind(datf, data.frame(year=yearr, rli=irler))

datf=datf[order(datf$year),]

if (plotit)

{

win.graph()

with(datf, plot(year, rli, pch=19, col=2, main=paste(group,"s", sep="")))

with(datf[datf$year%in%yearsti,], points(year, rli, pch=19, col=4))

}

restmw=NULL

for (m in 3:(nrow(datf)-2)) restmw=c(restmw, sample(datf$rli[(m-2):(m+2)],1))

finalmw=restmw

rliext[rliext$group==group&rliext$nrep==r,]$rli=finalmw

#rliext[rliext$group==group&rliext$nrep==r&!is.na(rliext$group),]$rli=finalmw

if (plotit) with(rliext[rliext$group==group&rliext$nrep==r,], lines(year, rli, col=8))

} # ends loop group

} # ends loop rep

head(rliext)

tail(rliext)

max(rliext$nrep)

max(rliext$rli)

rlitotagrs=with(rliext, aggregate(rli,list(year=year, nrep=nrep), mean))

names(rlitotagrs)[length(rlitotagrs)]="rli"

rlitotagrs$group="aggregated"

rlitotagrs$country=cnt

head(rlitotagrs)

rliextF=rbind(rliext,rlitotagrs[,c("country","group","year","nrep","rli")])

head(rliextF)

tail(rliextF)

max(rliextF$nrep)

max(rliextF$rli)

resg=with(rliextF, aggregate(rli, list(year=year, group=group), length))

resg$rli=with(rliextF, aggregate(rli, list(year=year, group=group), median))$x

resg$qn95=with(rliextF, aggregate(rli, list(year=year, group=group), q95))$x

resg$qn05=with(rliextF, aggregate(rli, list(year=year, group=group), q05))$x

head(resg)

## to correct estimated rlis that went out of the bounds [0-1] due to extrapolation

resg$rli[resg$rli>1]=1

resg$rli[resg$rli<0]=0

resg$qn95[resg$qn95>1]=1

resg$qn95[resg$qn95<0]=0

resg$qn05[resg$qn05>1]=1

resg$qn05[resg$qn05<0]=0

gs=unique(resg$group)

resgf=data.frame()

for (g in 1:lu(gs))

{

#g=1

gs1=as.character(gs[g])

gs1

if (gs1!="aggregated") yrs=min(genchF$start\_year[genchF$Group==gs1]):max(genchF$end\_year[genchF$Group==gs1]) else yrs=(max(with(genchF, aggregate(start\_year, list(Group), min))$x)-10):maxyear

rli3=resg[resg$group==gs1&resg$year%in%yrs,]

rli3=rli3[order(rli3$year),]

resgf=rbind(resgf,rli3)

}

head(resgf,30)

final=resg

final$country=cnt

head(final)

final=merge(final, nspecies)

final=final[order(final$ordem, final$year),]

finalrlis=rbind(finalrlis,final)

# 5.2 save the csv for each region/country

final2save=final[, c("country","group","year","rli","qn05","qn95","n\_sp","n\_gen\_changes")]

if (k==1) finfolder=paste(finfolderP,"Subglobal", sep="/")

finfolderF=paste(finfolder, "/", iso2use, "/csv\_tables", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

#finalname=paste(finfolderF, "/", iso2use, "\_", cnt,".csv", sep="")

finalname=paste(finfolderF, "/", iso2use, "\_", cnt,"\_", name2save,".csv", sep="")

finalname

write.csv(final2save,finalname, row.names=F)

if (plotit2){

if (div0(x,6)==0)

{

win.graph(50,30)

par(mfrow=c(2,3), mar=c(2,2,2,1))

}

tab2plot=resgf

with(tab2plot,plot(year,rli, ylim=c(min(qn05), max(qn95)), xlim=c(min(year), max(year)+4), type="n", xlab="Year", ylab="Red List Index of species survival", main=cnt))

gs=unique(tab2plot$group[tab2plot$group!="aggregated"])

lu(gs)

for (g in 1:length(gs))

{

#g=1

gs1=as.character(gs[g])

res2=tab2plot[tab2plot$group==gs1,]

with(res2,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

}

with(tab2plot[tab2plot$group=="aggregated",], polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

for (g in 1:length(gs))

{

gs1=as.character(gs[g])

res2=tab2plot[tab2plot$group==gs1,]

with(res2,lines(year,rli, col=2, lwd=2))

text(max(res2$year)+0.5, res2$rli[nrow(res2)],paste(gs1,"s", sep=""), cex=.6, adj=c(0,0.5))

}

with(tab2plot[tab2plot$group=="aggregated",], lines(year,rli, col=4, lwd=2))

}

} #ends loop if (cnt %in%torun)

if (cnt %in%others) { # to simply read the csv already saved if iso is the same as used in previous loops (e.g. for same isos in ISO\_BL and ISO\_SDG)

finfolderF=paste(finfolder, "/", "ISO\_BL", "/csv\_tables", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

finalname=paste(finfolderF, "/", "ISO\_BL", "\_", cnt, "\_", name2save,".csv", sep="")

finalname

final2save=read.csv(finalname)

finfolderF=paste(finfolder, "/", iso2use, "/csv\_tables", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

finalname=paste(finfolderF, "/", iso2use, "\_", cnt, "\_", name2save,".csv", sep="")

finalname

write.csv(final2save,finalname, row.names=F)

} #ends loop to read files already saved in ISO\_BL folder

} # ends loop country

print((proc.time()-tt)[1]/60) ## time in minutes

# 5.3 creates single table with all regions/countries

# reads all csvs per cnt and merges into a single table "finalrlis"

fls=dir(finfolderF)

finalrlis=data.frame()

for (y in 1:lu(fls)) {

finalrlis=rbind(finalrlis, read.csv(paste(finfolderF, fls[y], sep= '/')))

}

str(finalrlis)

head(finalrlis)

lu(finalrlis$country)

max(finalrlis$rli)

min(finalrlis$rli)

#######################################################################################################

#combine all files in 1 (in case the country files are calculated in different machines, to save time ) - DON'T FORGET TO SET THE iso2use !!

cmbfiles=F

#cmbfiles=T

if(cmbfiles){

iso2use="ISO\_BL" ### set which iso2use to combine

finfolderF=paste(finfolder, "/", iso2use, "/csv\_tables", sep="")

finfolderF

isos$isof=isos[,which(names(isos)==iso2use)]

tbf2=merge(tbf1, isos[,c("ISO3","isof")],by="ISO3",all.x=T)

allfics=data.frame()

fics=dir(finfolderF, pattern = ".csv", full.names =T) ### copy paste here the folder with the countries' files

for (u in 1:length(fics)) allfics=rbind(allfics, read.csv(fics[u]))

lu(allfics$country)

finalrlis=allfics

}

str(finalrlis)

head(finalrlis)

lu(finalrlis$country)

max(finalrlis$rli)

###################################################

# 5.4 save final graphs per region/country in pdf and png

iso2use

sumspp=with(tbf2, aggregate(pp, list(country=isof, group=Group), sum))

names(sumspp)[length(sumspp)]="sumpp"

head(sumspp)

sum(sumspp$sumpp)

rliextf=finalrlis

min(rliextf$rli)

max(rliextf$rli)

head(rliextf)

tail(rliextf)

plotit=T

writefiles=T

if (pdfs) plotit=F

allcountries\_RLI=data.frame()

allcountries\_CRLI=data.frame()

cts=unique(rliextf$country)

length(cts)

graphics.off()

for (i in 1:length(cts)) #length(cts) ### loop to go through all cnts to create pngs, pdfs and calculate the national contribution to the global RLI (crli)

{

#i=1

#i=which(cts=="ABW")

i

ct1=as.character(cts[i])

ct1

rli2=rliextf[rliextf$country==ct1,]

namect1=as.character(ct1) ## name to appear in the graphic

if (iso2use=="ISO\_BL") namect1=paste(isos$uname\_BL[isos$isof==as.character(ct1)][1] ," (", ct1, ")", sep="")

if (iso2use=="ISO\_SDG") namect1=paste(isos$uname\_SDG[isos$isof==as.character(ct1)][1] ," (", ct1, ")", sep="")

namect1

gs=unique(rli2$group)

lu(gs)

rlinew=data.frame()

crlis=data.frame()

### loop to estimate the crli for each group g in cnt i

for (g in 1:lu(gs))

{

#g=1

gs1=as.character(gs[g])

gs1

if (gs1!="aggregated") yrs=min(genchF$start\_year[genchF$Group==gs1]):max(genchF$end\_year[genchF$Group==gs1]) else yrs=(max(with(genchF, aggregate(start\_year, list(Group), min))$x)-10):maxyear

rli3=rli2[rli2$group==gs1&rli2$year%in%yrs,]

rli3=rli3[order(rli3$year),]

if (gs1!="aggregated")

{

crli1=rli2[rli2$group==gs1,]

crli1$crli=crli1$rli\*sumspp$sumpp[sumspp$group==gs1&sumspp$country==ct1][1]/sprich$rich[sprich$group==gs1]

crli1$crli05=crli1$qn05\*sumspp$sumpp[sumspp$group==gs1&sumspp$country==ct1][1]/sprich$rich[sprich$group==gs1]

crli1$crli95=crli1$qn95\*sumspp$sumpp[sumspp$group==gs1&sumspp$country==ct1][1]/sprich$rich[sprich$group==gs1]

crlis=rbind(crlis,crli1[,c("country","group","year","crli","crli05","crli95")])

}

rlinew=rbind(rlinew,rli3)

}

rlinew

tail(crlis)

# aggregated crli

crlisag=with(crlis, aggregate(crli, list(country=country, year=year), mean)) #

names(crlisag)[length(crlisag)]="crli"

crlisag$group="aggregated"

crlisag$crli05=with(crlis, aggregate(crli05, list(country=country, year=year), mean))$x

crlisag$crli95=with(crlis, aggregate(crli95, list(country=country, year=year), mean))$x

yrs=(max(with(gench, aggregate(start\_year, list(Group), min))$x)-10):maxyear

crlisag=crlisag[crlisag$year%in%yrs,]

crlisag

crlisag[,c("country","group","year","crli","crli05","crli95")]

crlisf=data.frame()

gs2=unique(crlis$group)

for (g in 1:length(gs2))

{

gs1=as.character(gs2[g])

yrs=min(genchF$start\_year[genchF$Group==gs1]):max(genchF$end\_year[genchF$Group==gs1])

crlisf=rbind(crlisf, crlis[crlis$group==gs1&crlis$year%in%yrs,])

}

crlisf=rbind(crlisf,crlisag) ## final table with crli for each group and aggregated

finfolderF=paste(finfolder, "/", iso2use, "/csv\_tables\_crlis", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

rlinew2save=rlinew[, c("country","group","year","rli","qn05","qn95")]

#fnamect=paste(finfolderF, "/", iso2use, "\_", ct1,".csv", sep="")

#fnamect

#if (writefiles) write.csv(rlinew2save,fnamect, row.names=F)

fnamect2=paste(finfolderF,"/", iso2use, "\_", ct1,"\_crli\_",name2save,".csv", sep="")

fnamect2

if (writefiles) write.csv(crlisf,fnamect2, row.names=F)

allcountries\_RLI=rbind(allcountries\_RLI,rlinew2save)

allcountries\_CRLI=rbind(allcountries\_CRLI,crlisf)

if (plotit) {

if (div0(i,4)==0)

{

win.graph(30,20)

par(las=1, mfrow=c(4,4))

}}

##5.4.1 to save the pdfs for this cnt

if (pdfs){

finfolderF=paste(finfolder, "/", iso2use, "/pdfs", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

finfolderF2=paste(finfolderF, "/", iso2use, "\_", ct1, sep="")

finfolderF2

#graph all taxa

pdf1name=paste(finfolderF2,"\_all\_taxa\_",name2save,".pdf", sep="")

pdf1name

pdf(file=pdf1name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=T, label=T) ### all w aggregated

dev.off()

#graph aggregated

pdf2name=paste(finfolderF2,"\_aggregated\_",name2save,".pdf", sep="")

pdf2name

pdf(file=pdf2name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="aggregated",], aggreg=T) # only aggregated

dev.off()

# graph birds

pdf3name=paste(finfolderF2,"\_birds\_",name2save,".pdf", sep="")

pdf3name

#miny=.05

pdf(file=pdf3name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="Bird",], aggreg=T, label=F) # only birds

dev.off()

### crli

pdf3name=paste(finfolderF2,"\_crli\_",name2save,".pdf", sep="")

pdf3name

pdf(file=pdf3name, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,6,4,2), mgp=c(4.5, 1, 0))

with(crlisf,plot(year,crli, ylim=c(min(crli05),max(crli05)), xlim=c(min(year), max(year)+1), type="n", xlab="", ylab="National contribution to the global RLI"))

par(mgp=c(3.5, 1, 0))

title(xlab="Year")

gs=unique(crlisf$group[crlisf$group!="aggregated"])

lu(gs)

for (g in 1:length(gs))

{

#g=1

gs1=as.character(gs[g])

res2=crlisf[crlisf$group==gs1,]

with(res2,polygon(c(year,rev(year)),c(crli05,rev(crli95)), col=cinz, border=cinz))

}

with(crlisf[crlisf$group=="aggregated",], polygon(c(year,rev(year)),c(crli05,rev(crli95)), col=cinz, border=cinz))

for (g in 1:length(gs))

{

gs1=as.character(gs[g])

res2=crlisf[crlisf$group==gs1,]

with(res2,lines(year,crli, col=2, lwd=2))

if (plotit) text(max(res2$year)+0.5, res2$crli[nrow(res2)],paste(gs1,"s", sep=""), cex=.6, adj=c(0,0.5))

if (pdfs) text(max(res2$year)+0.5, res2$crli[nrow(res2)],paste(gs1,"s", sep=""), cex=1, adj=c(0,0.5))

}

with(crlisf[crlisf$group=="aggregated",], lines(year,crli, col=4, lwd=2))

dev.off()

} # ends loop if pdfs

#5.4.2 to save the pngs for this cnt

if (pngs){

finfolderF=paste(finfolder, "/", iso2use, "/pngs", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

finfolderF2=paste(finfolderF, "/", iso2use, "\_", ct1, sep="")

finfolderF2

##### png all groups and aggregated

png2name=paste(finfolderF2,"\_\_all\_taxa\_",name2save,".png", sep="")

png2name

png(file=png2name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew, aggreg=T, label=T) ### all w aggregated

dev.off()

####graph aggregated png

png2name=paste(finfolderF2,"\_aggregated\_",name2save,".png", sep="")

png2name

#if (pngs) png(file=png2name, width = 12, height = 10)

png(file=png2name, width = 12, height = 10, units="in", res=600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

plotrli(miny=.15, topy=min(c(1.02, max(rlinew$qn95)\*1.1)), basy=min(rlinew$qn05)\*.99, ttuse=rlinew[rlinew$group=="aggregated",], aggreg=T) # only aggregated

dev.off()

#graph birds png changing y-axis

png3name=paste(finfolderF2,"\_birds\_",name2save,".png", sep="")

png3name

png(file=png3name, width = 12, height = 10, units = "in", res = 600)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

ttuse=rlinew[rlinew$group=="Bird",]

if (nrow(ttuse)==0) plot(1:10, 1:10, type="n", main="no data for birds", xaxt="n", yaxt="n", xlab="", ylab="")

if (nrow(ttuse)>0){

topy=min(max(ttuse$qn95))

basy=min(ttuse$qn05)

topyR <- 0.05\*ceiling(topy/0.05) # round up to nearest 0.05 for top of y-axis

basyR <- 0.05\*floor(basy/0.05) # round down to nearest 0.05 for base of y-axis

if (topyR > 1.00){ # if top value of y axis is greater than 1, set to 1 (true value cannot be higher, only artificially made higher by adding arbitrary values onto the maximum RLI value)

topyR <- 1.00

}

difrgy=diff(range(basy, topy))

with(ttuse,plot(year,rli, yaxt="n", ylim=c(basyR,topyR), type="n", xlab="Year", ylab="Red List Index of species survival"))

with(ttuse, polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

with(ttuse, lines(year,rli, col=2, lwd=2))

if (difrgy<0.3) axis(2, seq(0,1, by=.05))

if (difrgy>0.3) axis(2, seq(0,1, by=.1))

}

dev.off()

} # ends loop if pngs

} # ends loop countries

lu(allcountries\_RLI$country)

lu(allcountries\_CRLI$country)

#5.5 to save csv and pdf with the results of all cnts together

## single csv with all cnts

finfolderF=paste(finfolder, "/", iso2use, "/csv\_tables", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

rliallcntsfile=paste(finfolderF,"/", iso2use, "\_", "allcountries\_RLI\_",name2save,".csv", sep="")

rliallcntsfile

#if (writefiles) write.csv(allcountries\_RLI,rliallcntsfile, row.names=F)

crliallcntsfile=paste(finfolderF,"/", iso2use, "\_", "allcountries\_CRLI\_",name2save,".csv", sep="")

crliallcntsfile

#if (writefiles) write.csv(allcountries\_CRLI,crliallcntsfile, row.names=F)

###### single pdf with all plots

finfolderF=paste(finfolder, "/", iso2use, "/pdfs", sep="")

finfolderF

if (dir.exists(finfolderF)==F) dir.create(finfolderF,recursive = T)

pdfallname=paste(finfolderF,"/", iso2use, "\_", "RLI\_all\_countries\_",name2save,".pdf", sep="")

pdfallname

pdf(file=pdfallname, width = 12, height = 10)

par(las=1,cex.axis=1.2, cex.lab=1.3, cex.main=1.5, mar=c(5,5,4,2), mgp=c(3.5, 1, 0))

cts=unique(allcountries\_RLI$country)

for (i in 1:length(cts)) #length(cts)

{

#i=1

#i=which(cts=="ABW")

i

ct1=as.character(cts[i])

ct1

rli2=rliextf[rliextf$country==ct1,]

namect1=as.character(ct1)

if (iso2use=="ISO\_BL") namect1=paste(isos$uname\_BL[isos$isof==as.character(ct1)][1] ," (", ct1, ")", sep="")

if (iso2use=="ISO\_SDG") namect1=paste(isos$uname\_SDG[isos$isof==as.character(ct1)][1] ," (", ct1, ")", sep="")

namect1

fspct=pastables[pastables$Scientific%in%unique(tbf2$Scientific[tbf2$isof==ct1]),]

head(fspct)

respc=with(fspct, aggregate(year, list(iucn=iucn, year=year, group=group), length))

rlinew=allcountries\_RLI[allcountries\_RLI$country==ct1,]

#graph all

miny=.15

topy=min(c(1.02, max(rlinew$qn95)\*1.1))

topy

basy=min(rlinew$qn05)\*.99

difrgy=diff(range(basy, topy))

if (difrgy<miny) {

basy=topy-miny

difrgy=diff(range(basy, topy))}

with(rlinew,plot(year,rli, ylim=c(basy\*.99,topy), xlim=c(min(year), max(year)+1), type="n", xlab="Year", yaxt="n", ylab="Red List Index of species survival", main=namect1))

if (difrgy<0.3) axis(2, seq(0,1, by=.05))

if (difrgy>0.3) axis(2, seq(0,1, by=.1))

gs=unique(rlinew$group[rlinew$group!="aggregated"])

lu(gs)

for (g in 1:length(gs))

{

#g=1

gs1=as.character(gs[g])

res2=rlinew[rlinew$group==gs1,]

with(res2,polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

}

with(rlinew[rlinew$group=="aggregated",], polygon(c(year,rev(year)),c(qn05,rev(qn95)), col=cinz, border=cinz))

for (g in 1:length(gs))

{

gs1=as.character(gs[g])

res2=rlinew[rlinew$group==gs1,]

with(res2,lines(year,rli, col=2, lwd=2))

text(max(res2$year)+0.5, res2$rli[nrow(res2)],paste(gs1,"s", sep=""), cex=1, adj=c(0,0.5))

}

with(rlinew[rlinew$group=="aggregated",], lines(year,rli, col=4, lwd=2))

}

dev.off()

} # ends loop if length(cts)>0

} ## ends the loop to go through all the ccombinations if isos

} ### ends loop of length(isos2use)>0

} ## ends loop of k=1 | dis\_subglobal

} ####### ends loop to go through all the disaggregations

#PART 6. ORGANIZE FINAL FILE FOLLOWING THE UN TEMPLATE ######################################################################

############################################################################################################################################

if (save\_UN\_template)

{

lu=function (x=x) length(unique(x)) ## function to estimate the number of unique values in a vector # repeated here to when this part is run independently

finfolder=paste(finfolderP,"General RLI", sep="/")

finfolder

tnames=data.frame(varis=paste("V",1:16, sep=""),

fnames=c("Indicator", "SeriesID", "SeriesDescription", "GeoAreaCode","GeoAreaName", "TimePeriod","Value","Time\_Detail",

"UpperBound","LowerBound","Source", "FootNote","Nature" ,"Units","Reporting Type","SeriesCode"))

tnames

table\_sdg=data.frame()

#global

finfolderF=paste(finfolder, "/", "global/csv\_tables", sep="")

finalname=paste(finfolderF, "/", "global\_aggregated.csv", sep="")

tab2r=read.csv(finalname)

tabint=data.frame(V6=tab2r$year, V7=tab2r$rli, V8=tab2r$year, V9=tab2r$qn95, V10=tab2r$qn05)

head(tabint,20)

tabint$V1="15.5.1"

tabint$V2="2840"

tabint$V3="Red List Index"

tabint$V4=1

tabint$V5="World"

tabint$V11="BirdLife International and IUCN (2020), based on global estimates of the extinction risk (IUCN Red List categories) of all mammals, birds, amphibians, corals and cycads, derived from local and national data, disaggregated to the national scale and weighted by the proportion of each species's distribution in the country or region."

tabint$V12=""

tabint$V13="E"

tabint$V14="INDEX"

tabint$V15="G"

tabint$V16="ER\_RSK\_LSTI"

tabintF=tabint[,c("V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16")]

head(tabintF)

#names(tabintF)=tnames$fnames

table\_sdg=rbind(table\_sdg,tabintF)

head(table\_sdg)

tail(table\_sdg)

#countries

finfolderF=paste(finfolder, "/", "ISO\_SDG/csv\_tables", sep="")

fls=dir(finfolderF)

tab2r=data.frame()

for (y in 1:lu(fls)) {

tab2r=rbind(tab2r, read.csv(paste(finfolderF, fls[y], sep= '/')))

}

tab2r=tab2r[tab2r$group=="aggregated",]

head(tab2r)

tab2r$ISO=tab2r$country

nrow(tab2r)

codes2=codesreg[codesreg$Reference=="1.0-Global",]

tab2r=merge(tab2r, codes2)

tab2r=tab2r[order(tab2r$year),]

tab2r=tab2r[order(tab2r$country),]

nrow(tab2r)

head(tab2r)

tabint=data.frame(V4=tab2r$M49Code, V5=tab2r$CountryName, V6=tab2r$year, V7=tab2r$rli, V8=tab2r$year, V9=tab2r$qn95, V10=tab2r$qn05, iso=tab2r$ISO)

head(tabint,20)

tabint$V1="15.5.1"

tabint$V2="2840"

tabint$V3="Red List Index"

tabint$V11="BirdLife International and IUCN (2020), based on global estimates of the extinction risk (IUCN Red List categories) of all mammals, birds, amphibians, corals and cycads, derived from local and national data, disaggregated to the national scale and weighted by the proportion of each species's distribution in the country or region."

tabint$V12=""

tabint$V13="E"

tabint$V14="INDEX"

tabint$V15="G"

tabint$V16="ER\_RSK\_LSTI"

tabintF=tabint[,c("V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16")]

head(tabintF)

#names(tabintF)=tnames$fnames

table\_sdg=rbind(table\_sdg,tabintF)

head(table\_sdg)

tail(table\_sdg)

#regional

finfolderF=paste(finfolder, "/", "SDG\_Region/csv\_tables", sep="")

fls=dir(finfolderF)

tab2r=data.frame()

for (y in 1:lu(fls)) {

tab2r=rbind(tab2r, read.csv(paste(finfolderF, fls[y], sep= '/')))

}

finfolderF=paste(finfolder, "/", "SDG\_Subregion/csv\_tables", sep="")

fls=dir(finfolderF)

for (y in 1:lu(fls)) {

tab2r=rbind(tab2r, read.csv(paste(finfolderF, fls[y], sep= '/')))

}

tab2r=tab2r[tab2r$group=="aggregated",]

head(tab2r)

tab2r$ISO=tab2r$country

nrow(tab2r)

codes2=codesreg[codesreg$Reference=="2.1-Regional (SDG)",]

tab2r=merge(tab2r, codes2)

nrow(tab2r)

head(tab2r)

tabint=data.frame(V4=tab2r$M49Code, V5=tab2r$CountryName, V6=tab2r$year, V7=tab2r$rli, V8=tab2r$year, V9=tab2r$qn95, V10=tab2r$qn05)

head(tabint,20)

tabint$V1="15.5.1"

tabint$V2="2840"

tabint$V3="Red List Index"

tabint$V11="BirdLife International and IUCN (2020), based on global estimates of the extinction risk (IUCN Red List categories) of all mammals, birds, amphibians, corals and cycads, derived from local and national data, disaggregated to the national scale and weighted by the proportion of each species's distribution in the country or region."

tabint$V12=""

tabint$V13="E"

tabint$V14="INDEX"

tabint$V15="G"

tabint$V16="ER\_RSK\_LSTI"

tabintF=tabint[,c("V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16")]

head(tabintF)

#names(tabintF)=tnames$fnames

table\_sdg=rbind(table\_sdg,tabintF)

head(table\_sdg)

tail(table\_sdg)

#other groups

finfolderF=paste(finfolder, "/", "LLDC\_SIDS/csv\_tables", sep="")

fls=dir(finfolderF)

tab2r=data.frame()

for (y in 1:lu(fls)) {

tab2r=rbind(tab2r, read.csv(paste(finfolderF, fls[y], sep= '/')))

}

finfolderF=paste(finfolder, "/", "LDC/csv\_tables", sep="")

fls=dir(finfolderF)

for (y in 1:lu(fls)) {

tab2r=rbind(tab2r, read.csv(paste(finfolderF, fls[y], sep= '/')))

}

tab2r=tab2r[tab2r$group=="aggregated",]

head(tab2r)

tab2r$ISO=tab2r$country

nrow(tab2r)

codes2=codesreg[codesreg$Reference=="2.3-Other groupings",]

tab2r=merge(tab2r, codes2)

nrow(tab2r)

head(tab2r)

tabint=data.frame(V4=tab2r$M49Code, V5=tab2r$CountryName, V6=tab2r$year, V7=tab2r$rli, V8=tab2r$year, V9=tab2r$qn95, V10=tab2r$qn05)

head(tabint,20)

tabint$V1="15.5.1"

tabint$V2="2840"

tabint$V3="Red List Index"

tabint$V11="BirdLife International and IUCN (2020), based on global estimates of the extinction risk (IUCN Red List categories) of all mammals, birds, amphibians, corals and cycads, derived from local and national data, disaggregated to the national scale and weighted by the proportion of each species's distribution in the country or region."

tabint$V12=""

tabint$V13="E"

tabint$V14="INDEX"

tabint$V15="G"

tabint$V16="ER\_RSK\_LSTI"

tabintF=tabint[,c("V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16")]

head(tabintF)

table\_sdg=rbind(table\_sdg,tabintF)

head(table\_sdg)

tail(table\_sdg)

table\_sdg=table\_sdg[table\_sdg$V8>1992,]

### change headers and save

names(table\_sdg)=tnames$fnames

head(table\_sdg)

write.csv(table\_sdg, paste(finfolderP, name\_UN\_template, sep="/"), row.names=F)

} # ends if save\_UN\_template

# Annex 4: R code to estimate the proportion of each species’ distribution in each country

R code (species\_country\_intersect\_2020\_cleaned.R) has been developed to estimate the proportion of each species’ distribution in each country. As with the RLI code, this code has been updated (from codes originally created in 2016) to improve the efficiency of the code, including running the majority of species-country overlaps using species maps as rasters (only using polygons for species with very small ranges) and simplifying some of the steps, in addition to undertaking species filtering outside of the code (i.e. prior to running the analysis).

**Things to do before running the code**

Species preparation:

* Download IUCN species range maps for the five taxonomic groups that are used in the RLI (and more as they are added); Birds, Amphibians, Mammals, Corals and Cycads – Bird range maps are available at request from BirdLife; all other taxa are available on request from the IUCN.
* Filter and dissolve species ranges to only include:
  + Presence = ‘Extant’, ‘Probably Extant’, ‘Possibly Extinct’ or ‘Extinct’ (Codes 1, 2, 4, 5)
  + Origin = ‘Native’, ‘Reintroduced’ or ‘Assisted Colonisation’ (Codes 1, 2 or 6)
  + Seasonality = any season, which are ‘Resident’, ‘Breeding Season’, ‘Non-breeding Season’, ‘Passage’ or ‘Season Occurrence Uncertain’ (Codes 1, 2, 3, 4 or 5)

Note when doing this dissolve, make sure to take the most recent year of the range revision if the species has more than one Presence-Origin-Season combination (if filtering species to run based on when the map was last edited)

* Create a list of marine species – using the marine column in the species shape files where marine species are coded as ‘true’. This is an important step as this allows you to identify which species need to be overlapped with the Exclusive Economic Zones (EEZs).
* Optional steps:
  + If you are only going to run this for a subset of species (to reduce the time to undertake the analyses), you need to filter to species that need to have the proportion of their range within each country updated. These species are: (i) newly recognised and newly defined species, (ii) species that have had changes to range maps, and (iii) species that occur in areas where country boundaries have been redefined (such as splitting of a group of islands into individual islands, where the proportion of the species range will need to be calculated separately for each).
  + Create a list of endemic species using tabular data from the IUCN and add these to the output file as an overlap of 1 (i.e. 100%) occurring within a country, as there is no need to run the analyses for these.

Country preparation:

* Country layer – download the General Administrative Areas Database (GADM) from <https://gadm.org/data.html>. Within the GADM layer, use gadm28\_adm0 for the analyses (iit may be easier to output this as a shapefile from the geodatabase for ease of use).
* Exclusive Economic Zone layer – download the EEZ layer from <http://www.marineregions.org/downloads.php>

**Instructions to run the code** (search for TODO to see areas you need to change):

* Change ‘folder’ to your working directory, which should be set to where you have saved the required input csv files (see list below)
* Set ‘shapesc’ to the folder where the species range maps and country and EEZ layers are.
* Set ‘shapesSp’ to there folder that your species range maps are in
* Change the year after file batches to the year of running (for example this year was 2020 and so the folder was chosen to be ‘files batches 2020’

**Input files:**

* CSV file inputs:
  + marine.csv – list of marine species

*need a table including species names (in a column called binomial) and their corresponding taxonID*

* + Iso\_countries\_2019.csv – list of countries, ISO codes, regions, etc. Note that you need to check if there are any changes each year (based on list of recognised countries provided from SDG) and if required add any regional aggregations to this file

*See Annex 1: Structure of the input tables, structure of the table with the Iso codes for the table structure*

* + *Optional:*
    - *endemic.csv – list of endemic species*

*need a table including species names (in a column called binomial) and their corresponding taxonID*

* + - *Sp\_cnts\_pp\_FINAL\_2018\_all.csv – need this if you are not running all species and so you need to filter last year’s dataset to remove any species you plan to run this year*
    - *unchanged\_output\_since\_last\_year\_updated.csv – need this if you are not running all species and so this is the filtered list of last years output (you need to produce this after filtering the above data)*

*table structure and column names should match format of Sp\_cnts\_pp\_FINAL\_2018\_all.csv*

* Shapefile inputs:
  + Country layer (shapefile)
  + EEZ layer (shapefile)
  + Species layer(s) (shapefile – note layers if running for all species it is best to run the code in separate batches).

Output files:

* Sp\_cnts\_pp\_run\_2019.csv - the proportion of each species’ distribution (pp) in each country (ISO3 coded) that were run in this analyses (this year)
* *Optional:* *Sp\_cnts\_pp\_FINAL\_2019\_all.csv*

*the proportion of each species’ distribution (pp) in each country (ISO3 coded) for all species (combined) – this file is only produced if the species-country overlap was not ran for all species i.e. you only ran a subset of species, and you therefore need to combine the results of this code (Sp\_cnts\_pp\_run\_2019.csv) with last years output (in which case you will need this output from last year called ‘Sp\_cnts\_pp\_FINAL\_2018\_all.csv’ to produce this). Make sure to filter last year’s output to remove any species that have been re-ran this year (and so only include species that were not run).*

**Code Structure:**

* Part 1 – load in packages, input files and set folder and working directories
* Part 2 – define functions that are needed for the overlap analyses
* Part 3 – loop through batches of species data (only one if not running batches)
* Part 4 – run species-country overlaps
  + 4.1 data set-up for species-country overlap (if species range is <1,000km2)
  + 4.2 species polygon country overlap (if species range is ≥1,000km2)
  + 4.3 species raster country overlap
  + 4.4 marine species raster EEZ overlap
* Part 5 – combine all output files (species-countries both polygons and raster and EEZs)
* Part 6 – check species values for missing species and if sum to 1
  + check if there are any species missing and if so why
  + *Optional: check if any endemic species have not run and if so add them from a list of endemic species (note to make sure you only include species that should have been run and have failed to produce an output)*
  + adjust up to 1 if over estimates due to overlapping range maps
  + adjustup to 1 if an underestimate due to species occurring in high seas where there is no jurisdiction)
* Part 7 – output file of species-country overlap information for use in the RLI
  + *Optional: combine with species that were not ran this year if not all species were run*

**R script to run species-country overlap (species\_country\_intersect\_2020\_cleaned.R):**

# pp species per country calculator v1. 26/06/2016

# Maria Dias(BirdLife International), updated by Ashley Simkins (BirdLife International) 16/03/2020

# Script to estimate the percentage of each species' distribution range overlapping with each country and EEZ

### IMPORTANT NOTES

# Given the complexity of the code, it is strongly recommended to copy-paste the code into an R editor (e.g. R-studio)

# The script was written assuming that the analyses will run in a conventional PC (e.g 16 GB RAM); analyses should be SPLIT PER BATCHES

# Some shapefiles (e.g. MAMMALS) can take several minutes to upload

### TODO Before running this code, you need to read the instructions in Appendix 1 of the RLI Report. This will explain what files you need to run the code, what format the files should be in, any steps you need to take prior to running it, and finally describing how to run the code and what to change in the below.

## Part 1 - load in packages, input files and set folder and working directories

### packages to install

kpacks <- c('sp','maptools', 'rgdal', 'geosphere','fields','spatstat','maps','rgeos','data.table', 'raster', 'tidyverse', 'sf', 'lwgeom') ### install automatically the relevant packages

new.packs <- kpacks[!(kpacks %in% installed.packages()[,"Package"])]

if(length(new.packs)) install.packages(new.packs)

lapply(kpacks, require, character.only=T)

remove(kpacks, new.packs)

####################################################

year\_run <- format(Sys.Date(), "%Y") #get the current year

### TODO set directories

folder <- "C:/Users/Ashley.Simkins/Documents/SDG/RLI/RLI\_revised\_2019/input files country intersection" ## copy paste here the working folder - where the csv input files are stored

setwd(folder)

shapesc <- 'C:/Users/Ashley.Simkins/Documents/SDG/RLI/RLI\_revised\_2019/shapefiles\_RLI' # folder with the shapefiles with species countries and EEZs (copy-paste the path if different from the one defined in "folder")

shapesSp <- 'C:/Users/Ashley.Simkins/Documents/SDG/RLI/RLI\_revised\_2019/shapefiles\_RLI/species\_to\_run\_RLI\_2019\_noPOS'

if (file.exists(paste('../files batches', year\_run))){ #create folder in which to store the output files

finfolder <- paste('../files batches', year\_run)

} else {

dir.create(file.path(paste('../files batches', year\_run)))

finfolder <- paste('../files batches', year\_run)

}

linkiso <- read\_csv("Iso\_countries\_2019.csv") #list of countries and iso codes

marinesp <- read\_csv("marine.csv") ## list of species that use marine ecosystems

## Optional depending on what you are running

# Use (uncomment - remove the #) if you have a list of endemic species

#endemics <- read\_csv("endemic.csv") ## list of endemic species

#load in country and eez layers

cntsw <- st\_read(dsn = paste(shapesc, '/Gadm\_28\_adm0', sep = ''), layer="gadm28\_adm0", stringsAsFactors = F, geometry\_column = T) #reads the Countries polygon layer

eezw <- st\_read(dsn = paste(shapesc, '/World\_EEZ\_v10\_20180221', sep = ''), layer="eez\_v10\_wo\_joint\_diss") #reads the EEZ polygon layer

##############################

## Part 2 - define functions needed for overlap analyses

## custom functions

getVertices <- function(x){

stopifnot("sf" %in% class(x))

n <- nrow(x[[1]])

vts <- data.frame()

for (z in 1:n){

a <- lapply(x@polygons, function(y) {y@Polygons[[z]]@coords})

vts <- rbind(vts,data.frame(x = a[[1]][,1], y = a[[1]][,2]))

}

return(vts)

}

shp2raster <- function(shp, mask.raster, label, value, transform = FALSE, proj.from = NA,

proj.to = NA, map = TRUE) {

require(raster, rgdal)

# use transform==TRUE if the polygon is not in the same coordinate system as

# the output raster, setting proj.from & proj.to to the appropriate

# projections

if (transform == TRUE) {

proj4string(shp) <- proj.from

shp <- spTransform(shp, proj.to)

}

# convert the shapefile to a raster based on a standardised background

# raster

r <- raster::rasterize(shp, mask.raster)

# set the cells associated with the shapfile to the specified value

r[!is.na(r)] <- value

# merge the new raster with the mask raster and export to the working

# directory as a tif file

#r <- mask(merge(r, mask.raster), mask.raster, filename = label, format = "GTiff",

# overwrite = T)

# plot map of new raster

if (map == TRUE) {

plot(r, main = label, axes = F, box = F)

}

names(r) <- label

return(r)

}

lu <- function (x=x) length(unique(x))

first <- function (x=x) x[1]

## to fix country/eez polygons without ISO - see first in a GIS software if input shape is different

cntsw[is.na(cntsw$ISO),] #lists all countries without isos in the country layer

eeziso <- na.omit(eezw[,c("Sovereign1","ISO\_Ter1")]) #only use first territory for disputed territories

eezw[is.na(eezw$ISO\_Ter1),] #lists all countryies without isos in the EEZ layer

nnulls <- nrow(eezw[is.na(eezw$ISO\_Ter1),]) #check if any countries missing ISO codes

eezw <- eezw[!is.na(eezw$ISO\_Ter1),] ### excludes EEZ polygons without ISO3 (disputed/jointed areas; 12 in 232 polygons)

#eezw <- eezw[eezw$Pol\_type!= 'Joint regime',] #filter out sites with unclear ownership

length(unique(eezw$ISO\_Ter1)) #check how many unique iso codes there are

cntab <- data.frame(ISO3 = cntsw$ISO, Areakm = as.numeric(st\_area(cntsw)))

cntab$id <- 1:nrow(cntab)

str(cntab)

nrow(cntsw)

eeztab <- data.frame(ISO3 = eezw$ISO\_Ter1, Areakm = as.numeric(st\_area(eezw)))

eeztab$id <- 1:nrow(eeztab)

str(eeztab)

nrow(eezw)

# Part 3 - DECIDE HERE WHICH BATCH TO RUN by what shapefiles are in the filepath defined at the start as shapesSp

#load in species layer of species to run in batches (via a loop)

batch\_test <- list.files(shapesSp, pattern = ".shp$")

for (r in 1:length(batch\_test)){

#r <- 4

tabsp <- as.character(batch\_test[r]) #call the file tabsp

tabsp <- tools::file\_path\_sans\_ext(tabsp) #drop the file extension (drop .shp)

print(tabsp)

spTr <- st\_read(dsn = shapesSp, layer = tabsp, stringsAsFactors = F) #read in species shapefiles

str(spTr)

nrow(spTr)

#############################################################

if (file.exists(paste(finfolder, tabsp, sep = "/"))){# set folder where the files of this batch will be saved - the folder should exist and be within the working folder, if it doesn't then this is created here

finfolder2 <- paste(finfolder, tabsp, sep="/")

} else {

dir.create(file.path(paste(finfolder, tabsp, sep = "/")))

finfolder2 <- paste(finfolder, tabsp, sep="/")

}

tab2use <- spTr[!duplicated('binomial', fromLast = T),] #if more than one year for a species, take the most recent year of assessment

nrow(spTr) - nrow(tab2use) #should be 0 if species only have one map in the spatial data

#spTr$area <- st\_area(spTr$geometry) #get the area of each species' range

tab2use$marine <- 0

tab2use$marine[tab2use$binomial %in% marinesp$binomial] <- 1 #sets any species in the marine table as marine in the species table

tab2use$marine[tab2use$Group=="Coral"] <- 1 #set all corals to marine

unique <- unique(tab2use) #remove any duplicate species records

nrow(unique(tab2use[tab2use$marine == 1,])) #number of marine species

head(tab2use)

species <- unique(tab2use$binomial)

length(species)

## Part 4 - Run species-country overlaps

#create dataframes for outputs

final1 <- data.frame() #create df for country overlaps with species polygon maps

taberrors <- data.frame() #create df for species-country overlap errors

marine1 <- data.frame() #create df for species-eez (marine) overlaps

finrasters <- data.frame() #create df for country overlaps with species raster maps

for (x in 1:length(species)){

#x <- 1

print(paste(x, "in", length(species)))

## Part 4.1 data set-up for species-country overlap

spr <- tab2use[x ,]

sp <- spr$binomial

sp

#marine=T

marine <- F

if (spr$marine == 1){

marine <- T

}

marine

spr$size <- (as.numeric(st\_area(spr))/1000000) #calculate area of species' range, divided by 1million to convert m2 to km2

shp <- spr

shp

str(shp)

#shp

#shp <- shp[shp$presence%in%c(1, 2, 4, 5),] #this should already have been filted and dissolved prior to input

#shp <- shp[shp$origin%in%c(1, 2, 6),]

#str(shp)

nrow(shp)

if (nrow(shp) > 0){ ### some shapes result in empty after subselecting by origin (e.g. mammal Solomys salamonis)

areaspdeg <- NULL

###overlap using the vector layer (for species with smaller ranges - less than 100,000km2)

#if (spr$size<1000)

#Size determines whether to use species polygons or rasters

if (spr$size < 1000){ #run polygon overlap if species range is less than 1,000km2

# if (! "sf" %in% class(cntsw)) {

# cntsw <- as(cntsw, Class = 'sf') #convert into sf objects for polygon overlap code to work

# }

## Part 4.2 Species polygon overlap with countries

if (nrow(shp) > 1){ #if more than one polygon in the shape file, combine into one, this loop should account for 2+ polygons for a species, i.e. presence, origin season combinations - note these should have been filtered prior to running the analyses

shp2 <- NULL

shp$id2 <- 1

shp2 <- tryCatch({st\_union(shp)}, error = function(e){})

if (nrow(shp2) > 0){

shp <- shp2

} #if shp2 exists, set shp to shp2 (i.e. the combined polygon)

}

if (nrow(shp) == 1){ #only one polygon per species (which should be the case as should have had all ranges dissolved prior to analyses)

map2 <- F

if (map2){

win.graph(6,6)

plot(shp, border = 6, lwd = 3)

map("world", add = T)

}

finalint <- data.frame()

a <- cntab

a$spp <- 0

a$endemic <- 0

a$marine <- spr$marine

a$overlpdegree <- 0

a$areaspdeg <- NA

areaspdeg <- suppressWarnings(tryCatch({as.numeric(st\_area(shp, byid = FALSE))}, error = function(e){}))

if (length(areaspdeg) > 0){

a$areaspdeg <- areaspdeg

}

inters <- tryCatch({st\_intersects(cntsw, shp, sparse = F)}, error = function(e){}) #which countries intersect with the species

if (length(inters) > 0){

a$inters <- inters

}

if (length(inters) == 0){

a$inters <- F

}

head(a)

length(which(a$inters == T)) #check how many countries a species overlaps with

if (length(which(a$inters == T)) == 1 & marine == F){ ## identifies endemic species - if only overlap with one country (hence length of overlap is 1)

aa <- a[a$inters == T,]

aa$spp <- 1

aa$endemic <- 1

finalint <- rbind(finalint, aa[,c("ISO3","Areakm","id","spp","endemic","marine","overlpdegree","areaspdeg")])

}

if (length(which(a$inters == T)) > 1 | (length(which(a$inters == T)) > 0 & marine == T)){ #if a species occurs in 2 or more countries, or is a marine species

#if (length(which(a$inters == T)) > 0 | (length(which(a$inters == T)) > 0 & marine == T)){ #to include endemic species

a$totover <- t(st\_contains(shp, cntsw, sparse = F)) #checks if species range within countries

#a$totover2 <- st\_covers(cntsw, shp, sparse = T)

a$spp[a$inters == T & a$totover == T] <- 1 ## species completely within a country

ctscov <- a$id[a$spp == 1]

if (length(ctscov) > 0){

for (w in 1:length(ctscov)){

id2 <- ctscov[w]

a$overlpdegree[a$id == id2] <- suppressWarnings(st\_area(cntsw[id2,]))

}

}

a[a$inters == T & a$totover == F,]

cts2cut <- a$id[a$inters == T & a$totover == F] ## countries that overlap with sp but not completely

length(cts2cut)

if (length(cts2cut) > 0){ #### to account for polygons not overlapping with any country (see coturnix, second polygons season 1)

for (y in 1:length(cts2cut)){ #looping through each year of overlap

#y=1

id1 <- cts2cut[y]

cnt1 <- cntsw[id1,]

cnt1

if (map2) plot(cnt1, add = T, col = "lightgrey")

overlp = NULL

overlp = tryCatch({st\_intersection(shp,cnt1)}, error = function(e){})

if (map2){

plot(overlp, add=T, col=2)

}

if (nrow(overlp) == 0){ #if no overlaps, set the overlap value to NA ##note this was throwing an error but I decided to just rasterise all values anyway and so commented this out

pp <- NA

}

if (nrow(overlp) > 0){ #if there is an overlap, calculate the proportional overlap

pp <- suppressWarnings(as.numeric(st\_area(overlp))/as.numeric(st\_area(cnt1)))

}

a$spp[a$id == id1] <- pp #set the proportional overlap to be either NA (if threw an error) or the calculated proportional overlap

if (nrow(overlp) > 0){

a$overlpdegree[a$id == id1] <- suppressWarnings(as.numeric(st\_area(overlp))) #add in area of overlap to dataframe

}

} #ends loop countries

} # ends loop when length(cts2cut) > 0

aa <- a[a$inters == T,]

finalint <- rbind(finalint, aa[,c("ISO3","Areakm","id","spp","endemic","marine","overlpdegree","areaspdeg")])

#finalint <- rbind(finalint, aa[,c("ISO3","id","spp","overlpdegree","areaspdeg")])

} # ends loop for non endemic species intersecting with a country i.e. length(which(a$inters==T))>1 and marine species

if (length(which(a$inters == T)) > 0){

finalint$Scientific <- sp

finalint$TaxonID <- spr$id\_no

#finalint$ordem <- spr$ordem

finalint$x <- x

finalint$batch <- tabsp

final1 <- rbind(final1, finalint)

savesp <- T

if (savesp){

tname4 <- paste(paste(folder, finfolder2, sep = "/"), "/", x, "\_", sp, ".csv", sep="") #shapesc ## CHANGE

tname4

#tname4=paste(paste(shapesc, "files batches",finfolder,"spbysp/",sep="/"),x, "\_",sp, ".csv", sep="") #

write.csv(finalint, tname4, row.names = F)

}

}

} # end loop for length(shp)==1 , i.e., no errors when dissolving seasons

# if (nrow(shp)>1)

# {

# }

if (length(which(a$inters == T)) == 0) #if not overlap with any countries (an error)

{

if (length(areaspdeg) > 0){

taberrors <- rbind(taberrors, data.frame(Scientific = sp, TaxonID = spr$id\_no, error="no\_ovelp\_with\_countries"))

}

tname3 <- paste(paste(folder, finfolder2, sep="/"), "/taberrors\_until\_", x, ".csv", sep="") #shapesc ## CHANGE

write.csv(taberrors, tname3, row.names=F)

}

} # end loop size < 1,000; does not run polygon overlay if run this

###### rasterize to overlap with countries - for large/complex geometries or when topologic errors occurred with vector calculations

## Part 4.3 species raster overlap with countries

if (spr$size >= 1000 | nrow(shp) > 1 | length(areaspdeg) == 0){

#if (nrow(shp) > 0){

#if (length(shp)>1|length(areaspdeg)==0)

d <- cntab

d$endemic <- spr$endemic

d$marine <- spr$marine

if (! "SpatialPolygonsDataFrame" %in% class(shp)) {

shp <- as(shp, Class='Spatial') #convert into sp objects for rasterisation to work

}

cntsw2 <- as(cntsw, Class='Spatial') #create an sp version of the country layer for raster code to work

# shp <- as(shp, Class='Spatial') #convert into sp objects for rasterisation to work

# cntsw2 <- as(cntsw2, Class='Spatial')

UDbbox <- bbox(shp) #creates bounding box for raster

#UDbbox <- as.matrix(data.frame(c(UDbbox[[1]], UDbbox[[2]]), c(UDbbox[[3]], UDbbox[[4]]))) #convert to matrix

maxrange <- max(diff(range(UDbbox[1,])),diff(range(UDbbox[2,]))) #take largest diff between min and max dimensions (in degrees)

maxrange

Res <- 0.5

maxrange

## define grid cell of raster depending on range

if (maxrange < 100){

Res <- 0.3

}

if (maxrange < 50){

Res <- 0.1

}

if (maxrange < 5){

Res <- 0.05

}

Res

BL <- floor(UDbbox[,1]) + (Res/2)

TR <- ceiling(UDbbox[,2])

maskr <- raster::raster(xmn = BL[1], ymn = BL[2], xmx = TR[1], ymx = TR[2], crs = CRS(proj4string(shp)), resolution=Res, vals=1)

#create raster of species

shpr <- shp2raster(shp, maskr, value = 1, label = "shpr", map = F) #map =F

map3 <- F

if(map3){

plot(shpr)

plot(cntsw2, add = T)

}

#do raster country overlaps

cntr1 <- raster::extract(shpr, cntsw2, cellnumbers = TRUE)

tpix <- sum(na.omit(getValues(shpr)))

tpix

cntr2 <- unlist(lapply(cntr1, function(x) if (!is.null(x)) sum(x[,2], na.rm = TRUE) else 0 ))

d$ppcntraster <- cntr2/tpix

dd <- d[d$ppcntraster > 0,]

if (nrow(dd) > 0){

dd$Scientific = spr$binomial

dd$TaxonID = spr$id\_no

#dd$ordem = spr$ordem

dd$x <- x

dd$batch <- finfolder

sum(dd$ppcntraster)

finrasters <- rbind(finrasters, dd)

}

dd

tname6 <- paste(paste(folder, finfolder2, sep = "/"), "/", x, "\_", sp, "\_country\_raster\_.csv", sep = "") #shapesc CHANGE

tname6

write.csv(dd, tname6, row.names=F)

}

######################

## Part 4.4. marine species raster overlap with EEZs

if (marine == T){

b = eeztab

b$endemic <- spr$endemic

b$marine <- spr$marine

if (! "SpatialPolygonsDataFrame" %in% class(shp)) {

shp <- as(shp, Class='Spatial') #convert into sp objects for rasterisation to work

}

eezw2 <- as(eezw, Class='Spatial') #convert eez layer into spatial for raster code to work

UDbbox <- bbox(shp)

#UDbbox <- as.matrix(data.frame(c(UDbbox[[1]], UDbbox[[2]]), c(UDbbox[[3]], UDbbox[[4]]))) #convert to matrix

maxrange=max(diff(range(UDbbox[1,])),diff(range(UDbbox[2,])))

maxrange

Res=0.5

maxrange

if (maxrange<100) Res=0.3

if (maxrange<50) Res=0.1

Res

BL <- floor(UDbbox[,1]) + (Res/2)

TR <- ceiling(UDbbox[,2])

maskr <- raster::raster(xmn=BL[1], ymn=BL[2], xmx=TR[1], ymx=TR[2], crs = CRS(proj4string(shp)), resolution=Res, vals=1)

shpr <- shp2raster(shp, maskr, value = 1, label = "shpr", map = F)

map4 <- F

if(map4){

plot(shpr)

plot(eezw2, add=T)

}

eez1 <- raster::extract(shpr, eezw2, cellnumbers=TRUE)

tpix <- sum(na.omit(getValues(shpr)))

tpix

eez2 <- unlist(lapply(eez1, function(x) if (!is.null(x)) sum(x[,2], na.rm = TRUE) else 0 ))

b$ppeez <- eez2/tpix

bb <- b[b$ppeez > 0,]

if (nrow(bb) > 0)

{

bb$Scientific = spr$binomial

bb$TaxonID = spr$id\_no

#bb$ordem=spr$ordem

bb$x <- x

bb$batch <- finfolder

marine1 <- rbind(marine1, bb)

}

tname5 <- paste(paste(folder, finfolder2, sep="/"),"/",x, "\_",sp, "\_eez\_.csv", sep="") #shapesc CHANGE

#tname5=paste(paste(shapesc, "files batches",finfolder,"spbysp/",sep="/"),x, "\_",sp, "\_eez\_.csv", sep="") #shapesc CHANGE

tname5

write.csv(bb, tname5, row.names=F)

}

} #ends loop for species with valid shapes after subseting for origin (length(shp)>0)

} #ends loop species

#write output files for each of species polygon-country overlaps, species raster-country overlaps and marine species raster-EEZ overlaps

fnamef = paste(paste(folder, finfolder,sep="/"),"/final\_",trunc(as.numeric(Sys.time())), ".csv", sep="")

fnamef

write.csv(final1, fnamef, row.names=F) #produces empty file as didn't overlay species polygons (only ran them as rasters to speed up analyses)

fnamef2 = paste(paste(folder, finfolder,sep="/"),"/final\_marine\_",trunc(as.numeric(Sys.time())), ".csv", sep="")

fnamef2

write.csv(marine1, fnamef2, row.names=F)

fnamef3 = paste(paste(folder, finfolder,sep="/"),"/final\_rast\_",trunc(as.numeric(Sys.time())), ".csv", sep="") #changed to \_rast as \_raster is then picked up in later steps so doublecounted

fnamef3

write.csv(finrasters, fnamef3, row.names=F)

} #end loop through the different batches

############################################################################

## Part 5 Merge all species batches for polgon and raster (and EEZ) overlaps

foldersp <- list.files(paste(folder, finfolder, sep="/")) #create a list of all batches run

foldersp <- foldersp[!grepl('.csv', foldersp)] #removing the final output csv files - i.e. only including folders with individual files for species (remove final, final\_marine and final\_rast)

tball <- data.frame()

for (g in 1:length(foldersp)){

#for (g in 1)

#g=1

#g=2

folderf <- paste(folder, finfolder, foldersp[g], sep="/")

folderf

print(foldersp[g])

fics <- dir(folderf, full.names = T)

fics2 <- dir(folderf, full.names = F)

length(fics)

#the below creates a table to create a record for each species which details the number of overlaps using polygons or rasters with countries and if they were overlapped with EEZ

tfile <- data.frame(ficc = fics, fic = fics2, resvect = 0, resraster = 0, rezeez = 0)

for (x in 1:nrow(tfile)){

tfile$resraster[x] <- length(strsplit(fics2[x], "\_raster")[[1]]) - 1 #number of countries the species raster map overlapped with

}

for (x in 1:nrow(tfile)){

tfile$rezeez[x] <- length(strsplit(fics2[x], "\_eez")[[1]]) - 1 #number of EEZs the species raster map overlapped with

}

tfile$resvect[tfile$resraster == 0 & tfile$rezeez == 0] <- 1 #number of countries the species polygon map overlapped with (either used raster or eez, not both, so mutually exclusive)

head(tfile[,2:5])

head(tfile)

nrow(tfile)

sum(tfile$resvect)

sum(tfile$resraster)

sum(tfile$rezeez)

########################################################################

#### results vector analyses

tabvect <- data.frame()

#if (sum(tfile$resvect)>0)

if (sum(tfile$resvect) > 1){

ficsf <- as.character(tfile$ficc[tfile$resvect==1])

ficsf2 <- as.character(tfile$fic[tfile$resvect==1])

length(ficsf)

for (z in 1:length(ficsf)){

#z=8

#z=which(ficsf2==paste(704,"Treron griveaudi.csv", sep="\_"))

#z=1

tabs <- read.csv(ficsf[z])

tabs

tabsf <- NULL

# tabs$pp <- tabs$overlpdegree/tabs$areaspdeg

# tabsf <- tabs[,c("Scientific","TaxonID","marine","ISO3","pp")]

# tabsf$ovl <- "terrestrial"

# tabsf$method <- "vector\_overlap"

# tabvect <- rbind(tabvect,tabsf)

if (tabs$marine[1] == 1){ #if species is marine

tabs$pp <- tabs$overlpdegree/tabs$areaspdeg

tabsf <- tabs[,c("Scientific","TaxonID","marine","ISO3","pp")]

tabsf$ovl <- "terrestrial"

tabsf$method <- "vector\_overlap"

tabvect <- rbind(tabvect,tabsf)

}

tabsf

#overlap of non-marine and endemic species

if (tabs$marine[1] == 0 & tabs$endemic[1] == 1){

tabs$pp <- 1 #overlap is 100% with one country for endemic species

tabsf <- tabs[,c("Scientific","TaxonID","marine","ISO3","pp")]

#tabsf <- tabs[,c("Scientific","id","ISO3","pp")] #replace TaxonID (lose this) with an ID column (not taxonID)

tabsf$ovl <- "terrestrial"

tabsf$method <- "vector\_overlap"

tabvect <- rbind(tabvect,tabsf)

}

tabsf

#overlap of non-marine and non-endemic species

if (tabs$marine[1] == 0 & tabs$endemic[1] == 0){

tabs$areaspkm <- tabs$Areakm\*tabs$spp

Sumareakm <- sum(tabs$areaspkm)

tabs$pp <- tabs$areaspkm/Sumareakm

tabsf <- tabs[,c("Scientific","TaxonID","marine","ISO3","pp")]

tabsf$ovl <- "terrestrial"

tabsf$method <- "vector\_overlap"

tabvect <- rbind(tabvect,tabsf)

}

tabsf

}

head(tabvect)

lu(tabvect$Scientific)

}

########################################################################

#### results raster analyses

tabrast <- data.frame()

if (sum(tfile$resraster) > 0){

ficsf <- as.character(tfile$ficc[tfile$resraster == 1])

ficsf2 <- as.character(tfile$fic[tfile$resraster == 1])

length(ficsf)

for (z in 1:length(ficsf)){

#z=1

#z=which(ficsf2==paste(704,"Treron griveaudi.csv", sep="\_"))

tabs <- read.csv(ficsf[z])

tabs

tabsf <- NULL

tabs$pp <- tabs$ppcntraster

if (nrow(tabs) > 0){

tabsf = tabs[,c("Scientific","TaxonID","marine","ISO3","pp")]

#tabsf <- tabs[,c("Scientific","id","ISO3","pp")]

tabsf$ovl <- "terrestrial"

tabsf$method <- "raster\_overlap"

tabrast <- rbind(tabrast,tabsf)

tabsf

}

}

head(tabrast)

}

########################################################################

#### results eez analyses

tabeez <- data.frame()

if (sum(tfile$rezeez) > 0){

ficsf <- as.character(tfile$ficc[tfile$rezeez == 1])

ficsf2 <- as.character(tfile$fic[tfile$rezeez == 1])

length(ficsf)

for (z in 1:length(ficsf)){

#z=1

#z=which(ficsf2==paste(704,"Treron griveaudi.csv", sep="\_"))

tabs <- read.csv(ficsf[z])

tabs

tabsf <- NULL

tabs$pp <- tabs$ppeez

if (nrow(tabs) > 0){

tabsf=tabs[,c("Scientific","TaxonID","marine","ISO3","pp")]

#tabsf <- tabs[ ,c("Scientific","id","ISO3","pp")]

tabsf$ovl <- "eez"

tabsf$method <- "raster\_overlap"

tabeez <- rbind(tabeez,tabsf)

tabsf

}

}

head(tabeez)

}

####################################################

### Part 5 - combine all files

nrow(tabvect)

nrow(tabrast)

nrow(tabeez)

tabf1 <- rbind(tabvect,tabrast)

tabf <- rbind(tabf1,tabeez)

print(lu(tabf$Scientific))

tabf$batch <- foldersp[g]

tabfinal <- tabf

tabfinal <- unique(tabfinal)

###### errors - marine species that do not overlap with any eez (missing values of pp; e.g. "Pterodroma caribbaea" )

#2 species that are missing are terrestrial endemics

# tabfinal$pp[tabfinal$Scientific == 'Mogera uchidai'] <- 1

# tabfinal$pp[tabfinal$Scientific == 'Peromyscus caniceps'] <- 1

# tabfinal$ISO3[tabfinal$Scientific == 'Mogera uchidai'] <- 'JPN' #island off coast of japan

# tabfinal$ISO3[tabfinal$Scientific == 'Peromyscus caniceps'] <- 'MSR' #montserrat

tabfinal <- tabfinal[!is.na(tabfinal$pp),] #step added so you don't try to sum species with NAs (sum of anything with NA equals NA)

res2 <- with(tabfinal, aggregate(pp, list(Scientific = Scientific), sum))

max(res2$x)

spnas <- unique(tabfinal[is.na(tabfinal$pp),]$Scientific)

spnas

lu(spnas) #use these spnas (species with NAs for overlap) to input back into the country-species overlay # end 2019 I did this and realised missing species were endemics so added (so not an issue they were missed as I have now re-run these via vector analyses) - the code is now updated to include endemic species

#tabfinal[tabfinal$Scientific==spnas[3],]

unique(tabfinal[is.na(tabfinal$pp),]$Scientific)

unique(tabfinal[is.na(tabfinal$pp),]$method)

unique(tabfinal[is.na(tabfinal$pp),]$ovl)

nrow(tabfinal)

nnas <- nrow(tabfinal[is.na(tabfinal$pp),])

tabfinal <- tabfinal[!is.na(tabfinal$pp),]

nrow(tabfinal)

nrow(tabfinal) + nnas

## Part 6 - check species values and adjust to 1 if above/below as appropriate (keeping same ratio between countries)

### sp with ovl > 1 (e.g. different geometries for eez and countries)

res1 <- with(tabfinal, aggregate(pp, list(Scientific = Scientific), sum))

max(res1$x)

sest <- res1$Scientific[res1$x > 1]

lu(sest)

if (lu(sest) > 0){

for (w in 1:length(sest)){

#w=1

dat2 <- tabfinal[tabfinal$Scientific == sest[w],]

dat2

ppt <- sum(dat2$pp)

tabfinal$pp[tabfinal$Scientific == sest[w]] = tabfinal$pp[tabfinal$Scientific == sest[w]]/sum(tabfinal$pp[tabfinal$Scientific == sest[w]])

sum(tabfinal$pp[tabfinal$Scientific == sest[w]])

}

#final check

res2 <- with(tabfinal, aggregate(pp, list(Scientific = Scientific), sum))

print(max(res2$x))

t1 <- with(tabf[tabf$Scientific%in%sest,], aggregate(pp, list(Scientific = Scientific, ISO3 = ISO3), sum))

nrow(t1)

t2 <- with(tabfinal[tabfinal$Scientific%in%sest,], aggregate(pp, list(Scientific = Scientific, ISO3 = ISO3), sum))

names(t2)[length(t2)] = "ppfixed"

nrow(t2)

tf <- merge(t1, t2, all.x=T, all.y=T)

tf$dif <- tf$x-tf$ppfixed

max(tf$dif)

print(tf[tf$dif > .1,])

}

tball <- rbind(tball, tabfinal)

}

lu(tball$Scientific)

fnamef2 <- paste("C:/Users/Ashley.Simkins/Documents/SDG/RLI/RLI\_revised\_2019/",trunc(as.numeric(Sys.time())), ".csv", sep = "") #shapesc CHANGE

fnamef2

# write.csv(tball, fnamef2, row.names=F)

######################################################### identify missing species

#tball=read.csv("//AZ-FILESERVER2/gis\_data/Tracking Ocean Wanderers/IUCN\_R\_scripts/RLI/final\_pp\_file\_V1.csv")

tabsp <- st\_set\_geometry(spTr, NULL) #create table of species

lu(tabsp$binomial)

unique(tball$batch)

tres = with(tball, aggregate(pp, list(Scientific=Scientific, marine=marine), sum))

tres$ncountries = with(tball, aggregate(pp, list(Scientific = Scientific, marine=marine), lu))$x

min(tres$x)

max(tres$x)

hist(tres$x)

tres$sumpp <- tres$x

head(tres[tres$sumpp > 2,])

### to see which species are missing in the table (for which we have map)

#tabm = merge(tabsp[,c("Scientific","Group","TaxonID","marine")], tres[,c("Scientific","ncountries","sumpp")], by="Scientific", all.x=T)

tabm <- merge(tabsp[,c( "binomial", "group", "id\_no")], tres[,c("Scientific","ncountries","sumpp")], by.x = "binomial", by.y = "Scientific", all.x=T)

nrow(tabsp)

nrow(tabm)

nrow(tabm[is.na(tabm$sumpp),])

misg <- tabm[is.na(tabm$sumpp),]

#unique(misg$marine)

unique(misg$group)

nrow(misg)

head(tball)

nrow(tball)

tail(tball)

# TODO if you have a list of endemic species and want to include this in the analyses, uncomment below:

# head(endemics)

# missing <- endemics[(endemics %in% misg),] #find the species that are missing from the species-country overlap - note if you are only running a sample make sure you only include species that should have been run

# tball2 <- rbind(tball, endemics) #add these missing endemics to the output - note make sure you have matching fields in the same order or else this won't work (i.e Scientific, TaxonID, marine, ISO3, pp (set to 1), ovl (terrestrial), method (endemic), batch (whatever folder it should have been in))

# tball <- tball2

### to check sum of overlap (should be 1 or near 1 for all non-marine species)

nrow(tres[tres$x < .8,])

#min(tres$marine[tres$x<.8]) ### should be 1 (i.e., only marine species) or extinct species

min(tres$x) ### should be 1 (i.e., only marine species) or extinct species

tres[tres$x<.8 & tres$marine == 0,] #should be 0

tres[tres$x<.5,]

tres$sumpp <- tres$x

#tball[tball$Scientific=="Arenaria melanocephala",] ### coastal species - rescale to sum 1

#tball[tball$Scientific=="Calidris virgata",] ### coastal species - rescale to sum 1

#tball[tball$Scientific=="Cycas sphaerica",] ### endemic species; step below to reset to 1

#### rescale all pps to sum up 1

sp2rescale <- tres$Scientific[tres$x < .999]

lu(sp2rescale)

for (y in 1:length(sp2rescale)){

#y=1

sp <- as.character(sp2rescale[y])

sp

tball[tball$Scientific == sp,]

sum(tball$pp[tball$Scientific == sp])

tball$pp[tball$Scientific == sp] <- tball$pp[tball$Scientific == sp]/sum(tball$pp[tball$Scientific == sp])

tball[tball$Scientific == sp,]

sum(tball$pp[tball$Scientific == sp])

}

#tres2 <- with(tball, aggregate(pp, list(Scientific = Scientific, marine = marine), sum))

tres2 <- with(tball, aggregate(pp, list(Scientific = Scientific), sum))

min(tres2$x)

max(tres2$x)

###########################

## Part 7 - write the output of the analyses to file

# save the final file

all <- merge(tball, tabm, by.x='Scientific' , by.y = 'binomial', all.x = T)

all <- select(all, c('Scientific', 'group', 'TaxonID', 'ISO3', 'pp', 'ovl', 'method', 'batch')) #subset and reorder columns to match usual output

write.csv(all, paste("../Sp\_cnts\_pp\_run\_", year\_run,".csv", sep=""), row.names = F) #write output of analyses to file

############# Optional - if didn't run all species

#Combine the new ran species with the unchanged species output to create a full list of species-country overlaps for the 5 taxonomic groups

new\_run <- read\_csv(paste("Sp\_cnts\_pp\_run\_", year\_run, ".csv", sep="")) #read in new species-country intersection output if needed

prev <- read\_csv('unchanged\_output\_since\_last\_year\_updated.csv') #read in last years species country overlap information for species that haven't had changes to their range maps and are still taxonomically recognised

prev <- select(prev, c('Scientific', 'group', 'TaxonID', 'ISO3', 'pp', 'ovl', 'method', 'batch')) #subset to useful columns

unchanged\_run <- prev[!(prev$Scientific %in% new\_run$Scientific),] #subset to species that were not re-ran from last years analyses to carry over

overlap\_final <- rbind(new\_run, unchanged\_run) #combine this years run overlaps with last years overlap information that hasn't changed

overlap\_final$group <- tolower(overlap\_final$group) #set all taxonomic group names to lower case as some upper and some lower case for first letter

write.csv(overlap\_final, paste("../Sp\_cnts\_pp\_FINAL\_", year\_run, "\_all.csv", sep=""), row.names = F) #write output for RLI analyses including last years

1. Butchart, S. H. M. et al. (2004). Measuring Global Trends in the Status of Biodiversity: Red List Indices for Birds. PLoS Biology 2(12): e383. doi: <https://doi.org/10.1371/journal.pbio.0020383> [↑](#footnote-ref-1)
2. Butchart, S. H. M. et al. (2005). Using Red List Indices to measure progress towards the 2010 target and beyond. Philos. Trans. R. Soc. B 360: 255–268.  [doi: 10.1098/rstb.2004.1583](https://royalsocietypublishing.org/doi/full/10.1098/rstb.2004.1583) [↑](#footnote-ref-2)
3. Butchart, S. H. M. et al. (2007). Improvements to the Red List Index. PLoS One 2(1): e140. doi: <https://doi.org/10.1371/journal.pone.0000140> [↑](#footnote-ref-3)
4. Butchart, S. H. M. et al. (2010). Global biodiversity: indicators of recent declines. Science 328: 1164–

   1168. doi: [10.1126/science.1187512](http://www.sciencemag.org/content/328/5982/1164.short) [↑](#footnote-ref-4)
5. Rodrigues, A. S. L. et al. (2014). Spatially explicit trends in the global conservation status of vertebrates.

   PLoS ONE 9(11): e113934. doi: <https://doi.org/10.1371/journal.pone.0113934> [↑](#footnote-ref-5)
6. Each genuine change in status (i.e. from one Red List category to another, in a specified time period, resulting from genuine improvement or deterioration in status rather than improved knowledge or revised taxonomcy etc) is coded according to which factors drove the change, either alone or in combination. Thematic Driver RLIs are calculated using only the subset of genuine changes in status driven by a particular driver. [↑](#footnote-ref-6)
7. The global RLI will always run by default, because the results are needed to calculate the subglobal RLIs; this command controls if the global results will be stored or not [↑](#footnote-ref-7)
8. The global RLI will always run by default, because the results are used to calculate the subglobal RLIs; this command controls if the global results will be stored or not [↑](#footnote-ref-8)
9. see “options” field in table 1, main type Subglobal [↑](#footnote-ref-9)
10. see “options” field in table 1, main type Thematic [↑](#footnote-ref-10)
11. “iso units” are the several countries or regions within an iso group [↑](#footnote-ref-11)
12. In *italic*, examples of names of the input tables [↑](#footnote-ref-12)
13. In *italic*, examples of names of the input folders [↑](#footnote-ref-13)
14. In *italic*, example of a possible value [↑](#footnote-ref-14)
15. in parenthesis, formats/values accepted [↑](#footnote-ref-15)
16. Butchart, S. H. M. et al. (2007). Improvements to the Red List Index. PLoS One 2(1): e140. doi: <https://doi.org/10.1371/journal.pone.0000140> [↑](#footnote-ref-16)
17. Rodrigues, A. S. L. et al. (2014). Spatially explicit trends in the global conservation status of vertebrates.

    PLoS ONE 9(11): e113934. doi: <https://doi.org/10.1371/journal.pone.0113934> [↑](#footnote-ref-17)
18. Butchart, S. H. M. et al. (2004). Measuring Global Trends in the Status of Biodiversity: Red List Indices for Birds. PLoS Biology 2(12): e383. doi: <https://doi.org/10.1371/journal.pbio.0020383> [↑](#footnote-ref-18)
19. Butchart, S. H. M. et al. (2005). Using Red List Indices to measure progress towards the 2010 target and beyond. Philos. Trans. R. Soc. B 360: 255–268.  [doi: 10.1098/rstb.2004.1583](https://royalsocietypublishing.org/doi/full/10.1098/rstb.2004.1583) [↑](#footnote-ref-19)
20. Butchart, S. H. M. et al. (2007). Improvements to the Red List Index. PLoS One 2(1): e140. doi: <https://doi.org/10.1371/journal.pone.0000140> [↑](#footnote-ref-20)
21. Butchart, S. H. M. et al. (2010). Global biodiversity: indicators of recent declines. Science 328: 1164–

    1168. doi: [10.1126/science.1187512](http://www.sciencemag.org/content/328/5982/1164.short) [↑](#footnote-ref-21)
22. Rodrigues, A. S. L. et al. (2014). Spatially explicit trends in the global conservation status of vertebrates.

    PLoS ONE 9(11): e113934. doi: <https://doi.org/10.1371/journal.pone.0113934> [↑](#footnote-ref-22)