**Code to calculate KBA-protected area (KBA-PA) overlap**

Version 2.0. Instructions and recommendations

Ashley Simkins, ashley.simkins@birdlife.org

March 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 to (a) calculating the KBA-PA overlap, (b) assigning random years to protected areas with no reported year of designation and (c) plotting graphs for the various regional groups. This code does not categorise KBAs into the four different systems (marine, terrestrial, freshwater, mountain) for the disaggregated SDG indicators; this is done prior to the analyses. For details, refer to the metadata for the respective system (marine: <https://unstats.un.org/sdgs/metadata/files/Metadata-14-05-01.pdf>, terrestrial and freshwater: <https://unstats.un.org/sdgs/metadata/files/Metadata-15-01-02.pdf>, and mountain: <https://unstats.un.org/sdgs/metadata/files/Metadata-15-04-01.pdf>).
* The codes were modified in order to simplify some of the steps, tidy up the code and update the code to use more efficient packages (such as using sf rather than sp for working with spatial data).
* The code randomisation was also updated, such that protected areas lacking a date of designation in the WDPA were assigned a year that is randomly generated from the years for other protected areas in the same country. This randomly generated year was also used to determine the order of overlaps of protected areas with each KBAs (where previously all randomly generated years were either overlaid at the start/end of all overlaps of protected areas with a given KBA). In addition, errors in overlap now show as NA rather than 9999.
* The codes are written in R language and stored at [<https://github.com/BirdLifeInternational/kba-overlap>].

Contents

[A. KBA-PA overlap code – KBA\_PA\_overlap\_2020\_clean.R 2](#_Toc37848864)

[B. Missing years randomisation code – randomiser\_2020\_updated\_clean.R 14](#_Toc37848865)

[C. Graph generation code – final\_graphs\_2020\_cleaned.R 20](#_Toc37848866)

[D. SDG Reporting – sdg\_reporting\_output\_cleaned.R 25](#_Toc37848867)

[Annex 1: Calculating protected area coverage of marine and terrestrial KBAs 32](#_Toc37848868)

# KBA-PA overlap code – KBA\_PA\_overlap\_2020\_clean.R

**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 ‘finfolder’ to the folder where you want the csv files for each countries’ KBA protected area coverage to be saved.
* Change line 506 to only include regional groups you want to run – remove or add regional groupings as required (these are based on the columns in ‘Iso\_countries\_year\_run.csv’ with a global grouping added (for global summary):

**Required input files:**

* Spatial files:
  + KBA shapefile – contains spatial layer for all confirmed KBAs (available at <http://www.keybiodiversityareas.org/site/requestgis>)
  + Cleaned protected area spatial layer – provided by UNEP-WCMC

*An explanation of how the WDPA layer is cleaned is provided in the metadata files for each of the 3 indicators (see urls above)*

* CSV files:

*Note in all of the following, ‘year\_run’ is the year the analyses is being run (i.e. for this year 2020); if you are running it for a different year to the current year (e.g. running 2020 in 2019 or vice versa), you will have to replace ‘year\_run’ with the year you want to run (so the code can find the right csv file)*

* + ‘classif\_KBAs\_FINAL\_year\_run.csv’ – this is the KBA classification file, which classifies sites to whether they are terrestrial, marine, freshwater and/or mountain KBAs

*1 means the site is classified as that system, 0 means the site isn’t classified as that system*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SitRecID** | **marine** | **terrestrial** | **Freshwater** | **mountain** | **ISO3** | **Country** |
| 11 | 1 | 1 | 0 | 0 | PRT | Portugal |
| 12 | 1 | 1 | 0 | 0 | PRT | Portugal |
| 17 | 0 | 1 | 0 | 1 | GIB | Gibraltar (to UK) |
| 18 | 1 | 1 | 0 | 0 | GIB | Gibraltar (to UK) |
| 19 | 0 | 1 | 0 | 1 | AND | Andorra |
| 21 | 0 | 1 | 0 | 1 | CYP | Cyprus |
| 24 | 0 | 1 | 0 | 0 | CYP | Cyprus |
| 25 | 0 | 1 | 0 | 1 | CYP | Cyprus |

* + ‘Iso\_countries\_year\_run.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 by the UN Statistics Division in their annual email requesting the latest data for SDG indicator 15.5.1 (currently directed to IUCN Chief Scientist Tom Brooks and UNEP-WCMC Programme officer Edward Lewis)) and if required add any regional aggregations to this file

*See below for the table structure – the table needs the country name and ISO3 code, all other regions are optional and can be added to or removed as required – note you must complete values for each country that is applicable. Region\_name is a generic example*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **countryname** | **ordem** | **country\_SIS** | **region** | **ISO3** | **ISO\_BL** | **ISO\_SDG** | **uname\_BL** | **uname\_SDG** | **Region\_name** | **etc** |
| Afghanistan | 66 | Afghanistan | West and Central Asia | AFG | AFG | AFG | Afghanistan | Afghanistan | Region 1 | ... |
| Akrotiri and Dhekelia | 256 | Akrotiri and Dhekelia | Europe | XAD | CYP | CYP | Cyprus | Cyprus | Region 1 | … |
| Aland | 248 | Aland Islands | Europe | ALA | ALA | FIN | Finland | Finland | Region 2 | … |

**Output files:**

* ‘Country\_name.csv’ – csv files for each country with each KBA and its protected area coverage for each new year of protected area designation that overlapped that KBA

*Each country output is stored in a folder called files\_country\_year\_run. Below is an example of the structure of the table produced for one country. SitRecID = KBA siteID, kba = kba area, ovl = area of overlap, year = year of protected area designation (0 if no protected areas overlap the kba), random = whether the designation year of the protected area was randomly assigned, nPAs = number of protected areas, percPA = percentage overlap of the kba with protected areas that year, ISO = ISO3 code, COUNTRY = country KBA is within. Note that when a kba has more than one year (e.g. site 31751), the overlap (ovl) and percentage overlap (percPA) refer to the increased overlap compared to the previous year (it is not total for that year); you would have to sum all overlaps across all years to determine the protected area overlap of that kba now.*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SitRecID** | **kba** | **ovl** | **year** | **random** | **nPAs** | **percPA** | **ISO** | **COUNTRY** |
| 31963 | 1.23E+09 | 4577777 | 1983 | FALSE | 1 | 0.003724 | DZA | Algeria |
| 46579 | 3.41E+09 | 0 | 0 | FALSE | 0 | 0 | DZA | Algeria |
| 6181 | 6.29E+09 | 0 | 0 | FALSE | 0 | 0 | DZA | Algeria |
| 31803 | 1.15E+09 | 0 | 0 | FALSE | 0 | 0 | DZA | Algeria |
| 6163 | 75337809 | 0 | 0 | FALSE | 0 | 0 | DZA | Algeria |
| 31751 | 2.53E+09 | 7.85E+08 | 1983 | FALSE | 3 | 0.309615 | DZA | Algeria |
| 31751 | 2.53E+09 | 1.84E+08 | 1999 | FALSE | 1 | 0.072588 | DZA | Algeria |
| 31963 | 1.23E+09 | 4577777 | 1983 | FALSE | 1 | 0.003724 | DZA | Algeria |
| 46579 | 3.41E+09 | 0 | 0 | FALSE | 0 | 0 | DZA | Algeria |
| 6181 | 6.29E+09 | 0 | 0 | FALSE | 0 | 0 | DZA | Algeria |

* ‘final\_tab\_year\_run.csv’ – csv file of all kba-pa overlaps for each new year of protected area designation

*The table is the same structure as above but includes all KBAs and their protected are overlaps (it is the combination of all country\_name.csv files*)

* ‘Input data for R regional\_grouping.csv’ – csv files for each regional grouping, including each kba that falls within that grouping and its change in protected area coverage and the year its coverage changed

*See example of an input table below (input data for R COUNTRY KBAs). siteid = kba siteID, region = regional grouping applied, country = country of site, perprotected = percentage of the site protected in that year (or additional percentage of site protected if multiple years), year = year of protected area designation, random\_year = whether the protected area year was randomly allocated or not*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **siteid** | **region** | **country** | **perprotected** | **year** | **random\_year** |
| 2904 | Albania | Albania | 45.52651 | 1995 | FALSE |
| 2907 | Albania | Albania | 2.717954 | 1996 | FALSE |
| 2908 | Albania | Albania | 1.535857 | 1966 | FALSE |
| 2908 | Albania | Albania | 0.067676 | 2004 | FALSE |
| 2908 | Albania | Albania | 0.345314 | 2016 | FALSE |
| 31748 | Albania | Albania | 9.035337 | 2007 | FALSE |
| 46562 | Albania | Albania | 0 | 0 | FALSE |
| 46567 | Albania | Albania | 85.87721 | 2005 | FALSE |
| 46623 | Albania | Albania | 0 | 0 | FALSE |

* ‘in\_out\_files\_year\_run.csv’ – list of input and output file names for regional groupings which will be used in the randomisation step

*Below shows an example of what this file may look like – note this will vary depending on which regional groupings you choose to produce (there is a row for each regional grouping). Code = regional grouping (columns in iso table that were selected and global summary), inputfile = file name written for that regional grouping, global = whether the regional grouping is a summary of all data or not, outputfile1 = file name that will be written for each regional grouping – these will be written after the randomisation code has run*

|  |  |  |  |
| --- | --- | --- | --- |
| **code** | **inputfile** | **global** | **outputfile1** |
| global | Input data for R global KBAs.csv | 1 | Output data for R global KBAs.csv |
| countryname | Input data for R countryname KBAs.csv | 0 | Output data for R countryname KBAs.csv |
| ordem | Input data for R ordem KBAs.csv | 0 | Output data for R ordem KBAs.csv |
| country\_SIS | Input data for R country\_SIS KBAs.csv | 0 | Output data for R country\_SIS KBAs.csv |
| region | Input data for R region KBAs.csv | 0 | Output data for R region KBAs.csv |
| ISO3 | Input data for R ISO3 KBAs.csv | 0 | Output data for R ISO3 KBAs.csv |
| ISO\_BL | Input data for R ISO\_BL KBAs.csv | 0 | Output data for R ISO\_BL KBAs.csv |
| ISO\_SDG | Input data for R ISO\_SDG KBAs.csv | 0 | Output data for R ISO\_SDG KBAs.csv |
| uname\_BL | Input data for R uname\_BL KBAs.csv | 0 | Output data for R uname\_BL KBAs.csv |
| uname\_SDG | Input data for R uname\_SDG KBAs.csv | 0 | Output data for R uname\_SDG KBAs.csv |
| Region.old\_MDG. | Input data for R Region.old\_MDG. KBAs.csv | 0 | Output data for R Region.old\_MDG. KBAs.csv |
| Developing | Input data for R Developing KBAs.csv | 0 | Output data for R Developing KBAs.csv |
| LDC | Input data for R LDC KBAs.csv | 0 | Output data for R LDC KBAs.csv |
| LLDC\_SIDS | Input data for R LLDC\_SIDS KBAs.csv | 0 | Output data for R LLDC\_SIDS KBAs.csv |
| CN\_SHAPEFILE | Input data for R CN\_SHAPEFILE KBAs.csv | 0 | Output data for R CN\_SHAPEFILE KBAs.csv |
| NOTES | Input data for R NOTES KBAs.csv | 0 | Output data for R NOTES KBAs.csv |
| IPBES\_region | Input data for R IPBES\_region KBAs.csv | 0 | Output data for R IPBES\_region KBAs.csv |
| SDG\_Region | Input data for R SDG\_Region KBAs.csv | 0 | Output data for R SDG\_Region KBAs.csv |
| SDG\_Subregion | Input data for R SDG\_Subregion KBAs.csv | 0 | Output data for R SDG\_Subregion KBAs.csv |
| CMS\_Region | Input data for R CMS\_Region KBAs.csv | 0 | Output data for R CMS\_Region KBAs.csv |

**Before running the analyses:**

* You need to check if all KBAs have been classified against all four systems – if not you will need to check which system(s) they should be classified as (for the method, see the metadata urls on page one). Add these updated records to ‘classif\_KBAs\_FINAL\_year\_run.csv

fields <- select(fields, c("global", "COUNTRY", "countryname","Developing","LDC","LLDC\_SIDS","IPBES\_region", "region", "SDG\_Region","ISO\_BL","ISO\_SDG","SDG\_Subregion"))

**Code Structure:**

* Part 1: Setup
  1. Load in packages and define functions
  2. Set working directories and load in input files
  3. Load in kba and protected area shapefiles
* Part 2: Data cleaning

2.1 Fix issues in ISO codes in protected area layer

2.2 KBAs missing ISO codes

2.3 Splitting transboundary protected areas

2.4 Create country list

* Part 3: Spatial Analysis – KBA-protected area overlap
  + Start to loop through each country, for the current country, create a list of kbas and protected areas in that country
  + If no protected areas in the country, write to output (overlap is 0)
  + For countries with at least one protected area, overlap each kba in a country with each protected area within the country to determine which (if any) overlap that kba
  + If a kba has no overlap with any protected areas, set the overlap to 0
  + For each KBA that overlaps with at least one protected area, order the years of protected area designation from oldest to most recent and overlap the kba with the protected area that was designated first (and if there are more than one protected area designated in this year, then these protected areas are combined).
  + Then compare the protected area(s) designated in the next year in the sequence to the previous protected area coverage and if there is a new area of overlap (area where protected areas did not previously overlap with the KBA but they now do), calculate this area. This continues through the list of designation years until all protected areas that overlap a KBA have been overlaid.
    - NB if a protected area had no reported year of designation, a random year was assigned from a list of years of designation of other protected areas in that country – where there were no protected areas in the country that had their year of designation reported, a year was randomly assigned based on all protected areas after 1986
  + A summary table of all kba-protected area overlap was written to csv file
* Part 4: Produce separate files per region
  + Any KBAs with a total protected area coverage over 1 (i.e. over 100%) were rescaled so that the total coverage was 1
  + Regional groups were filtered to include as required (this is where you need to edit the list to only include regions that you are interested in)
  + Csv files were written of the kba-pa overlap for each of the regional groupings of interest
  + Table of output file names for each regional grouping for the randomisation step were produced

**R Script:**

## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

## KBA-protected area overlap calculator v2.0

## Ash Simkins & Lizzie Pearmain, March 2020

## based on code by Maria Dias, 2016

## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Script to estimate the overlap of PAs and KBAs (giving earliest year of designation)

# In the 'marine' is any site with >=5% of its area as sea, as determined by intersection, likewise for 'mountain'. 'terrestrial' is any site with <=95% of its area as sea. 'freshwater' is any site with freshwater trigger species. See metadata for indicators for more details.

### IMPORTANT NOTES

# Given the complexity of the code, it is strongly recommended to copy-paste the code into an R editor (e.g. RStudio)

# The minimum requirement to run the script is a 16 GB RAM machine

# WDPA layer is extremely large, can take several minutes (up to half an hour) to load

# to facilitate the process, the code runs the script and saves a file country by country. This avoids reanalysing the countries already done in case of error

# it might occur an error preventing to calculate which kbas overlap with protected area. These situations are easily identifiable in the final csv file (filter by ovl=9999)

## ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

############# Part 1 - Setup #######################

#### Part 1.1 install and load packages ----

kpacks <- c('sf', 'dplyr','tidyverse', '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)

#### Define functions ----

lu <- function (x = x){

length(unique(x))

#nrow(unique(x))

}

#### 1.2 set file locations and working directories ----

## NB.in the KBA layer attribute table, the relevant fields should be "SitRecID" and "Country", not in capitals!

## TODO set folder to be your working directory, and finfolder to where you want to save the output of kba-pa overlap for each country

year\_run <- format(Sys.Date(), "%Y")

folder <- ("C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/KBA\_PA\_Overlap\_rewritten") ## set the working directory

finfolder <- paste("C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/KBA\_PA\_Overlap\_rewritten/files\_country", year\_run, sep="\_") #folder where the files per country will be saved

setwd(folder)

tabmf <- read.csv(paste("classif\_KBAs\_FINAL\_", year\_run, ".csv", sep = "")) ## file with types of kbas

isos <- read.csv(paste("Iso\_countries\_", year\_run, ".csv", sep = "")) ## file with ISO codes; should be stored in the wkfolder specified above; no changes in 2019, so 2018 file used

#### 1.3 Read in shapefiles ----

kbas <- st\_read(dsn = '../../../KBA/KBAsGlobal\_2019\_September\_02', layer = 'KbaGlobal\_2019\_September\_02\_POL', stringsAsFactors = F) #note field called Shape not geometry

#kbas<-st\_read(dsn = '.', layer = 'KBAs\_for\_2019\_SDG', stringsAsFactors = F, geometry\_column = T)

kbas <- kbas[!is.na(kbas$SitRecID),] #remove any NAs

#pas<- st\_read(dsn = 'WDPA\_Dec2018\_SDGs.gdb', layer = 'WDPA\_Dec\_2018\_Merge', stringsAsFactors=F)

pas <- st\_read(dsn = '../../../Protected Areas/wdpa\_nov2019/wdpa\_for\_birdlife/wdpa\_poly\_nov2019\_sdg\_prep.gdb', layer = 'wdpa\_poly\_nov2019\_sdg\_prep', stringsAsFactors = F)

#### TODO: CHECK GEOMETRY TYPES - continue from here: https://github.com/r-spatial/sf/issues/427

pas <- pas[!is.na(st\_dimension(pas)),]

as.character(unique(st\_geometry\_type(st\_geometry(pas)))) ## what geometries are in the dataset

#kbas <- st\_make\_valid(kbas) #repair any geometry issues

#pas <- st\_make\_valid(pas) #repair any geometry issues

## convert factors to characters in the dataframes

## PAs dataframe

pas$ISO3 <- as.character(pas$ISO3)

pas$PARENT\_ISO <- as.character(pas$PARENT\_ISO)

str(pas)

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

#### Part 2 - DATA CLEANING ----

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

## only need to run the following lines until the ISO3 in the kba layer is corrected -

## this changes the correct #ISO3 in the PA layer to match the wrong ISO in the kba layer.

## Otherwise these #bas are excluded because the ISO3 in the two layers don't match.

## When the ISO3 in the kba layer is corrected, these lines should be deleted.

#### 2.1 - fixing issues in ISO codes ----

pas$ISO3[(pas$ISO3)=='ALA'] <- 'FIN'

pas$ISO3[(pas$ISO3)=='ASC'] <- 'SHN'

pas$ISO3[(pas$ISO3)=='CPT'] <- 'FRA'

pas$ISO3[(pas$ISO3)=='GGY'] <- 'GBR'

pas$ISO3[(pas$ISO3)=='IMN'] <- 'GBR'

pas$ISO3[(pas$ISO3)=='JEY'] <- 'GBR'

pas$ISO3[(pas$ISO3)=='TAA'] <- 'SHN'

pas$ISO3[(pas$ISO3)=='WAK'] <- 'UMI'

pas$ISO3[(pas$ISO3)=='XAD'] <- 'CYP'

pas$ISO3[(pas$ISO3)=='XKO'] <- 'SRB'

pas$ISO3[(pas$ISO3)=='XNC'] <- 'CYP'

unassigned\_pas <- pas[pas$ISO3 == " " | is.na(pas$ISO3) | pas$ISO3 == '---',]

#### 2.2 - KBAs with no ISO code ----

unique(kbas$ISO3)

unique(kbas$Country[kbas$ISO3 == "---"])

kbas$ISO3[kbas$ISO3 == "---" & kbas$Country == "High Seas"] <- "ABNJ"

kbas$ISO3[kbas$ISO3 == "---" & kbas$Country == "Falkland Islands (Malvinas)"] <- "FLK"

#kbas$ISO3[kbas$ISO3 == "---" & kbas$Country != "High Seas"] <- "RUS"

unique(kbas$Country[kbas$ISO3 == " "])

unique(kbas$Country[is.na(kbas$ISO3)])

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Palau"] <- "PLW"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Aruba"] <- "ABW"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Aruba (to Netherlands)"] <- "ABW"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Guadeloupe"] <- "GLP"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Guadeloupe (to France)"] <- "GLP"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Norfolk Island"] <- "NFK"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Norfolk Island (to Australia)"] <- "NFK"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Lao People's Democratic Republic"] <- "LAO"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Laos"] <- "LAO"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "India"] <- "IND"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Cuba"] <- "CUB"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Libya"] <- "LBY"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Belarus"] <- "BLR"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Russian Federation"] <- "RUS"

kbas$ISO3[(kbas$ISO3 == " " | is.na(kbas$ISO3)) & kbas$Country == "Russia (Asian)"] <- "RUS"

kbas <- kbas[kbas$Country != 'Disputed',] #remove any sites that cannot be assigned a country as are disputed

unassigned\_kbas <- kbas[kbas$ISO3 == " " | is.na(kbas$ISO3) | kbas$ISO3 == '---',] #check if any sites don't have an ISO3 code, if any are missing, add in country name (if non are missing, will have 0 observations)

# site 27335 missing - in Belarus

#kbas$ISO3[kbas$SitRecID == 27335] <- 'BLR'

#kbas$Country[kbas$SitRecID == 27335] <- 'Belarus'

## Fill in the country field for these sites as well

#kbas$Country[kbas$ISO3 == "RUS"] <- "Russian Federation"

kbas\_without\_names <- kbas[kbas$Country == " ",] #checks if any KBAs are missing country names, should be 0, if not find out which sites are missing country names and add in country name

#### 2.3 Transboundary PAs ----

cnpa <- data.frame(ISO3 = unique(pas$ISO3))

cnpa$nchart <- nchar(as.character(cnpa$ISO3))

cnpa <- cnpa[cnpa$nchart>4, ] #where iso3 codes have more than 4 characters (more than one country per site)

cnpa

transb <- data.frame()

for (g in 1:nrow(cnpa)){ #this loop checks each transboundary pa and splits the iso code and only assigns the site to each country

#g=2

cnpa1 <- cnpa[g, ]

sp <- substr(cnpa1$ISO3, 4, 5)

if (sp == "; "){

cnpa2 <- data.frame(ISO3=strsplit(as.character(cnpa1$ISO3), split="; ")[[1]])

cnpa2$oISO3 <- as.character(cnpa1$ISO3)

}

if (sp != "; "){

cnpa2 <- data.frame(ISO3=strsplit(as.character(cnpa1$ISO3), split=";")[[1]])

cnpa2$oISO3 <- as.character(cnpa1$ISO3)

}

transb <- rbind(transb, cnpa2)

}

transb

#### 2.4 - create list of countries ----

# DgProjw <- CRS(proj4string(kbas)) #checks coordinate system - DEPRECATED in sf

kbas #check that in the console output: proj4string: +proj=longlat +datum=WGS84 +no\_defs

kbas <- kbas[!is.na(kbas$SitRecID),] #remove any NAs

listcnts <- as.character(unique(kbas$ISO3))

lu(listcnts)

#for reruns

#listcnts <- as.character(unique(kbas$ISO3[kbas$ISO3 %in% c('FIN', 'SHN', 'FRA', 'GBR', 'UMI', 'CYP', 'SRB')])) #missed countries

#lu(listcnts)

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

#### Part 3 - SPATIAL ANALYSIS ----

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

##### OVERLAP WITH PROTECTED AREAS

### per country

finaltab <- data.frame()

tt <- proc.time()

for (x in 1:length(listcnts)){ #starts loop for all countries

#for (x in 1:10){ #starts loop for all countries

#x=1

#x=8

#x <- which(listcnts==country) #find x based on country name

country <- listcnts[x]

## 1. Subset kbas and pas to this country

kba.c <- kbas[kbas$ISO3 == country, ]

pa.c <- pas[pas$ISO3 == country, ] ## protected areas within the country

#finds PA in country for transboundary sites and so includes in pa country list

if (country %in% transb$ISO3){

iso3 <- c(country, transb$oISO3[country == transb$ISO3])

iso3

pa.c <- pas[pas$ISO3 %in% iso3, ]

}

## 2. Print country name and ISO3 code to console

country.n <- kba.c$Country[1]

cat(x, '\t', country, '\t', country.n, '\n')

## 3. Plot map of KBAs and PAs to check ----

plotit <- F

if(plotit){

plot(kba.c$Shape, border=3)#kbas are in green

plot(pa.c$Shape, border=4, add=T) # pas are in blue

title(main=paste(country.n, country))

box()

axis(1)

axis(2)

}

### could refine by removing this bit when we've added in some preliminary analysis to REMOVE ALL COUNTRIES WITH NO PAs.

nrow(pa.c) ## number of PAs in the country

if (nrow(pa.c) == 0){ #finds all kbas with no protected area overlap - sets all output to 0 (0 overlap, no. of pas it overlaps with are 0, etc)

areasov <- data.frame(SitRecID = kba.c$SitRecID, kba = NA, ovl = 0, year = 0, random = F, nPAs = 0, percPA = 0, ISO = country, COUNTRY = country.n)

}

# areasov <- data.frame(SitRecID = kba.c$SitRecID, kba = st\_area(kba.c$Shape), ovl = 0, year = 0, nPAs = 0, percPA = 0, ISO = country, COUNTRY = country.n) #try to set KBA area even if no overlap

#}

## this may then no longer need an if statement.

if (nrow(pa.c) > 0){

ovkba <- NULL

ovkba <- st\_intersects(pa.c$Shape, kba.c$geometry, sparse = FALSE)

ovkba ## matrix where rows = PAs, and cols = KBAs

nrow(ovkba)

if (length(ovkba) == 0){ #if there is no matrix produced, this is an error so set all outputs to error i.e. 9999

areasov <- data.frame(SitRecID = NA, kba = NA, ovl = NA, year = NA, random = F, nPAs = NA, percPA = NA, ISO = country, COUNTRY = country.n)

}

if (length(ovkba) > 0){ #if there ARE overlaps between kbas and pas:

areasov <- data.frame()

#####next bit is new code which re-assigns missing years to a randomly selected year from PAs in the respective country # should be in data cleaning

pa.c$random <- F

if (min(pa.c$STATUS\_YR) == 0){

for (row in 1:nrow(pa.c)){

if (pa.c$STATUS\_YR[row] == 0){ #create a new column to identify any site that has had a year randomly allocated

pa.c$random[row] <- T

}

}

ryears <- pa.c$STATUS\_YR[pa.c$STATUS\_YR > 0] #select all years where the status year isn't 0

if (length(ryears) == 0){ #if all status years are 0

ryears <- pas$STATUS\_YR[pas$STATUS\_YR > 1986] #then use range of status years for all protected areas (not just in this country) later than 1986

}

if (length(ryears) == 1){ #if only one year that is not 0

ryears <- c(ryears, ryears)

}

pa.c$STATUS\_YR[pa.c$STATUS\_YR == 0] <- base::sample(ryears, nrow(pa.c[pa.c$STATUS\_YR == 0, ]), replace = T) ## selects a year randomly from the pool of possible years

}

########### end of pa year randomisation

for (z in 1:nrow(kba.c)){ ## starts loop for all kbas in the country (changed to nrow as was looking at columns rather than rows)

#for (z in 1:length(kba.c)){ ## starts loop for all kbas in the country

#z = 1

#z=which(kbac$SitRecID=="23937")

#z=which(kba.c$SitRecID=="26878")

#print(z)

kbaz <- kba.c[z, ]

head(kbaz)

akba <- NA #set to 9999 to incase next steps don't run

#akba <- suppressWarnings(tryCatch({gArea(kbaz, byid=FALSE)}, error=function(e){}))

akba <- as.numeric(suppressWarnings(tryCatch({st\_area(kbaz$geometry, byid = FALSE)}, error=function(e){})))

akba

#length(which(ovkba[z, ] == T)) # number of KBAs overlapping with protected area z

length(which(ovkba[ ,z] == T)) # find the number of pas that the 'zth' kba overlaps with (the particular kba the loop is currently processing)

if (length(which(ovkba[ ,z] == T)) > 0){ ### when at least 1 pa overlaps with the kba

#win.graph()

#pacz <- pa.c[which(ovkba[z, ]==T), ] #seems to be incorrect index

pacz <- pa.c[which(ovkba[ ,z] == T), ] #subset to pas that overlap this kba

nrow(pacz)

pacz

#pacz2 <- st\_make\_valid((pacz)) #repair geometry of pacz if necessary

if (plotit){

plot(kbaz$Shape)

plot(pacz$Shape, col=rgb(0,0,.8,0.2), border=0, add=T)

}

yearspacz <- pacz$STATUS\_YR #years of pas in kba z

ovf <- NULL

#ovf <- tryCatch({gIntersection(pacz, kbaz, byid=T)}, error=function(e){}) ## spatial intersection kba and all pas overlapping

ovf <- tryCatch({st\_intersection(pacz, kbaz)}, error = function(e){}) ## spatial intersection kba and all pas overlapping, results in polygon output for each overlap (in sf/dataframe)

#TODO this line doesn't always run if there is interesting geometry within the PA layer.

nrow(ovf)

class(ovf)

if ("sf" %in% class(ovf) & length(yearspacz) > 0){

ovfpol <- ovf #not needed but avoiding having to rename subsequent dataframes

years <- sort(unique(ovfpol$STATUS\_YR))

years

year1 <- min(years)

ovf1 <- ovfpol[ovfpol$STATUS\_YR == year1, ]

nrow(ovf1) #changed from length

ovf11 <- NULL

ovf11 <- tryCatch({st\_union(ovf1, by\_feature = F)}, error=function(e){})

#nrow(ovf11)

if(plotit) plot(ovf11, col = 2)

ovlz <- as.numeric(suppressWarnings(tryCatch({st\_area(ovf11, byid = FALSE)}, error=function(e){})))

if (length(ovlz) == 0){ #if there was an error, assign overlap to be 9999 (signifying an error)

ovlz <- NA

}

ovlz

random0 <- pacz$random[pacz$STATUS\_YR == year1] #creates vector of whether each protected area overlap within a given year was randomly sampled or if it was true designation year

random1 <- FALSE #sets random to false by default - i.e. PA designation has a given year

random1[TRUE %in% random0] <- TRUE #assign random as true if any protected areas in a given year were randomly allocated a year

areasov1 <- data.frame(SitRecID=kbaz$SitRecID, kba=akba, ovl=ovlz, year=year1, random = random1, nPAs=nrow(ovf1)) #creates row in output table with this site overlap area and associated information within it #sets numbers to numeric not units (removes m^2)

areasov1

if (length(years) > 1){

for (w in 2:length(years)){

#w=2

#w=4

#w=6

rema <- 1-(sum(areasov1$ovl[!is.na(areasov1$ovl)])/akba) ## to see if there is still any area left by the pas of year 1

rema

if (rema > 0.02){ #assuming 2% error in delineation of kbas compared to pas

year2 <- years[w]

year2

ovf2 <- ovfpol[ovfpol$STATUS\_YR == year2, ]

nrow(ovf2)

ovf22 <- NULL

ovf22 <- tryCatch({st\_union(ovf2, by\_feature = F)}, error=function(e){})

if(plotit){

plot(ovf22, add=T, col=w+1)

}

ovfprev <- ovfpol[ovfpol$STATUS\_YR < year2, ]

ovfprev3 <- tryCatch({st\_union(ovfprev, by\_feature = FALSE)}, error=function(e){}) #merge all polygons from previous years

if(plotit){

plot(ovfprev3, add=T, col=w+2)

}

ovf23 <- NULL

ovf23 <- tryCatch({st\_difference(ovf22, ovfprev3)}, error = function(e){}) #to determine if there is a difference in protected area coverage of kba the following year by making a new polygon of the area in the following year that wasn't in the previous year

if(plotit){

plot(ovf23, add=T, col="grey")

}

ovlz <- as.numeric(suppressWarnings(tryCatch({st\_area(ovf23, byid = FALSE)}, error = function(e){})))

if (length(ovlz)==0){

ovlz <- NA

}

ovlz

random2 <- pacz$random[pacz$STATUS\_YR == year2] #creates vector of whether each protected area overlap within a given year was randomly sampled or if it was true designation year

random3 <- FALSE #sets random to false by default - i.e. PA designation has a given year

random3[TRUE %in% random2] <- TRUE #assign random as true if any protected areas in a given year were randomly allocated a year

areasov1 <- rbind(areasov1,data.frame(SitRecID=kbaz$SitRecID, kba=akba, ovl=ovlz, year=year2, random = random3, nPAs=nrow(ovf2)))

areasov1

}

}

}

areasov1

} # ends loop for class(ovf)=="SpatialPolygons"

#if (nrow(ovf) == 0 | ovf == NULL | !"sf" %in% class(ovf)){

if (is.null(ovf) | !"sf" %in% class(ovf)){

areasov1 <- data.frame(SitRecID=kbaz$SitRecID, kba=akba, ovl=NA, year=0, random=F, nPAs=0) ## error in spatial overlap

}

} ## ends loop for PAs overlapping with the KBA

if (length(which(ovkba[ ,z] == T)) == 0){

areasov1 <- data.frame(SitRecID=kbaz$SitRecID, kba=akba, ovl=0, year=0, random=F, nPAs=0) ## if there are NO (zero/none) pas overlapping the kba

}

areasov <- rbind(areasov,areasov1)

} ## ends loop for all kbas in the country

areasov$percPA <- areasov$ovl/areasov$kba

areasov

max(areasov$percPA)

areasov$ISO <- country

areasov$COUNTRY <- country.n

} # ends loop for ovlkba>0

} ## ends loop for length(pac)>1

finaltab <- rbind(finaltab,areasov)

tname <- paste(finfolder,"/",country.n, ".csv", sep="")

tname

write.csv(areasov, tname, row.names=F)

}

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

head(finaltab)

str(finaltab)

lu(finaltab$x) #not sure what suppposed to do

finaltab <- unique(finaltab)

write.csv(finaltab, paste("finaltab\_", year\_run, ".csv", sep=""), row.names = F)

### end here

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

########## Part 4 - PRODUCE SEPARATE FILES FOR EACH REGION ###########################

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

isofiles = T ## if we need to bind all country files in a single table

if (isofiles){

fls <- dir(finfolder)

final1 <- data.frame()

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

final1 <- rbind(final1, read.csv(paste(finfolder, fls[y], sep = '/')))

}

}

write.csv(final1, paste("finaltab\_", year\_run, ".csv", sep=""), row.names = F) #only run if didn't run finaltab writing above (i.e. if compiling all countries later)

lu(final1$ISO)

if (isofiles == F){

final1 <- finaltab

}

##### CAN RESTART FROM HERE

final1 <- read\_csv(paste("finaltab\_", year\_run, ".csv", sep="")) # only run if not already loaded in (i.e. if final1 doesn't exist)

final <- final1[!is.na(final1$ovl),]

lu(final1$SitRecID) - lu(final$SitRecID) # number of KBAs with problems of geometry

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

str(final)

str(tabmf)

tabf <- merge(final, tabmf, by = "SitRecID")

str(tabf)

head(tabf)

max(tabf$percPA)

tabf$percPA[tabf$percPA > 1] <- 1

max(tabf$percPA)

resf <- with(tabf, aggregate(percPA, list(SitRecID = SitRecID), sum))

max(resf$x)

nrow(resf[resf$x > 1,]) # number of kbas for which the sum of the overlap of the multiple years is more than 1 (can be due to geometry oversimplification in gDifference step)

kbas2fix <- unique(resf$SitRecID[resf$x>1])

lu(kbas2fix)

#rescale kbas perc prot

if (length(kbas2fix) > 0){

for (k in 1:length(kbas2fix)){

tabf[tabf$SitRecID == kbas2fix[k], ]$percPA = tabf[tabf$SitRecID == kbas2fix[k], ]$percPA/sum(tabf[tabf$SitRecID == kbas2fix[k], ]$percPA)

}

}

#check if was fixed (result should be 1)

max(with(tabf, aggregate(percPA, list(SitRecID = SitRecID), sum))$x)

#tabf=read.csv("tabf\_propa.csv")

## subset for confirmed ibas

## link regions

isos$ISO <- isos$ISO3

str(tabf)

tabff <- merge(tabf, isos, by = "ISO")

str(tabff)

unique(tabff$year)

tabff$year[tabff$year == 0 & tabff$ovl != 0 & !is.na(tabff$ovl)] <- 3000 ## PAs with no year of designation; assume 3000 for posterior randomization (NOTE that changed in 2020; random process was added before the spatial overlap) - may no longer be needed but good to check if any sites get assigned 3000 and if so check why

unique(tabff$ovl[tabff$year == 3000]) ## should be NULL; only the kbas with 0 overlap with PAs have year=0, because random year was added to the codes before the overlapping step (modified in 2020)

#write.csv(tabff, "tabff.csv", row.names=F)

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

#tables to export

# define list or regions being used

# isos <- select(isos, c("global", "COUNTRY", "countryname","Developing","LDC","LLDC\_SIDS","IPBES\_region", "region", "SDG\_Region","ISO\_BL","ISO\_SDG","SDG\_Subregion"))

fields <- c('global', colnames(isos)) #this will create a list of all regional groups in the ISO table, with global added as a category for the global summary.

#TODO you will need to remove any of the regions from 'fields' that you don't want to run; to do this, write the column names of the files that you are interested in below and run the line to subset to only produce output for these regions; alternatively remove these from the isos table before running the line above to generate the list of fields

fields <- select(fields, c("global", "COUNTRY", "countryname","Developing","LDC","LLDC\_SIDS","IPBES\_region", "region", "SDG\_Region","ISO\_BL","ISO\_SDG","SDG\_Subregion"))

#create table required for randomisation process based on subset of regions being used for the run:

inout <- as.data.frame(fields) #create a table with the region codes

colnames(inout)[colnames(inout) == 'fields'] <- 'code' #rename fields as code

for (row in 1:nrow(inout)){ #loop through to generate the required input and output file names, and assign global as a distinct region (or not)

inout$inputfile[row] <- paste("Input data for R", inout$code[row], "KBAs.csv", sep=" ")

inout$global[row][inout$code[row] == 'global'] <- 1

inout$global[row][inout$code[row] != 'global'] <- 0

inout$outputfile1[row] <- paste("Output data for R", inout$code[row], "KBAs.csv", sep=" ")

}

write.csv(inout, paste("in\_out\_files", format(Sys.Date(), "\_%Y"), '.csv', sep=""), row.names = F) #write in\_out file to csv so that it can be used for the randomisation analyses

#create an input folder in which to store the kba-pa overlap for each regional grouping, to be used as an input in the randomisation code

if (!(file.exists(paste(folder, "/input tables ", year\_run, sep = "")))){ #create graphs folder

dir.create(paste(folder, "/input tables ", year\_run, sep = ""))

}

for (f in 1:length(fields)){

#f=7

ff <- as.character(fields[f])

ff

tfname <- paste("Input data for R", ff, "KBAs.csv", sep=" ")

tfname

if (ff == "global"){

tab2export <- with(tabff, data.frame(siteid = SitRecID, region = ff, country = COUNTRY, perprotected = percPA\*100, year = year, random\_year = random))

}

if (ff != "global"){

tab2export <- with(tabff, data.frame(siteid = SitRecID, region = tabff[,which(names(tabff) == ff)], country = COUNTRY, perprotected = percPA\*100, year = year, random\_year = random))

}

str(tab2export)

head(tab2export)

tail(tab2export)

write.csv(tab2export, paste("input tables", format(Sys.Date(), " %Y/"), tfname, sep=""), row.names = F)

}

# Missing years randomisation code – randomiser\_2020\_updated\_clean.R

**Instructions to run the code** (search for TODO to see areas you need to change):

* Change the setwd to your working directory, which should be set to where you have saved the required input csv files (see list below)

**Required input files:**

* CSV files:

*Note, as in the previous code, in all of the following, ‘year\_run’ is the year the analyses is being run (i.e. for this year 2020); if you are running it for a different year to the current year (e.g. running 2020 in 2019 or vice versa), you will have to replace ‘year\_run’ with the year you want to run (so the code can find the right csv file). The structure for each of these files are outlined in the previous section*

* + ‘classif\_KBAs\_FINAL\_year\_run.csv’ – this is the KBA classification file, which classifies sites to whether they are terrestrial, marine, freshwater and/or mountain KBAs
  + ‘in\_out\_files\_year\_run.csv’ – list of input and output files as written at the end of the kba-pa overlap code – so know what files to read in and to output
  + ‘Input data for R regional\_grouping.csv’ – csv files for each regional grouping, including each kba that falls within that grouping and its change in protected area coverage and they year its coverage changed

**Output files:**

* ‘Output data for R regional\_grouping.csv’ – csv files for each regional grouping with a kba-pa overlap value for each year from 1900 to the present year with confidence interverals, and how many kbas per regional grouping are fully protected. This is done for each system (all, terrestrial, marine, freshwater and mountain)

*Belows shows an example output table for country data for all KBAs (Output data for R country KBAs) – note when the system is all (i.e. includes all 4 systems), it doesn’t say all at the end, but if it were freshwater, the file name would say \_Freshwater at the end. year = year, 95CIlow/mid/high = 95% lower/middle/upper confidence interval, Count = the number of kbas completely covered by protected areas, Percentage = percentage of protected areas full covered, Percentage\_Area = mean percentage coverage of kbas by protected areas – all three of these are per regional grouping, region = regional grouping, code = regional grouping category applied (e.g. country, ISO\_BL, etc), sset = system*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| year | 95CIlow  Count | 95CImid  Count | 95CIhi  Count | 95CIlow  Percentage | 95CImid  Percentage | 95CIhi  Percentage | CI95low  Percentage  \_Area | CI95mid  Percentage  \_Area | CI95hi  Percentage  \_Area | region | code | sset |
| 2000 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2001 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2002 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2003 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2004 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2005 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2006 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2007 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2008 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |
| 2009 | 6 | 6 | 6 | 26.08696 | 26.08696 | 26.08696 | 28.05396 | 28.05396 | 28.05396 | Angola | country | all |

**Code Structure:**

* Part 1: Read in input files, set working directory and define functions
  + Functions include calculating random years, counting the number of kbas fully protected for each region and calculating the mean protected area coverage of kbas for each region
* Part 2: Randomisation code to create confidence intervals due to uncertainty in years of designation
  + loops through each regional grouping in turn to calculate the number of kbas with complete protected coverage per region, the mean protected area coverage of kbas per region and confidence intervals based on this (due to years with no protected area designation and so randomly allocated years of designation)
  + output csv files with the number of kbas fully protected per region and mean protected area coverage of kbas per region for each regional grouping and system

**R Script:**

# IBA randomizer 2.0

# JPW Scharlemann 14/04/2014 jorn.scharlemann@unep-wcmc.org, updated by Ash Simkins March 2020

# Script to randomly allocate year of protection and percentage protected to IBAs with missing data

# Missing years and percentage protection are allocated based on the PAs with known information from that country, or if the country has two or fewer IBAs that are protected information will be allocated at random from all IBAs

# data preparation:

#

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

# Prepare a datafile

# extract from Excel spreadsheet and save as .csv file

# ibaname (or IBA ID)

# country

# region

# perprotected = proportion protected of IBA (i.e. percentage protected / 100), 0 if missing information

# year = year of protection, random\_year coded as true if this overlap contains protected area overlap with unknown years (and so assigned randomly)

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

#### Part 1 - Read in input files and define functions #################

#TODO update wd to where you stored the Input data for R... files

year\_run <- format(Sys.Date(), "%Y")

setwd (paste("C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/KBA\_PA\_Overlap\_rewritten/input tables", year\_run))

#workd <- paste("C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/KBA\_PA\_Overlap\_rewritten/input tables", year\_run)

inout <- read.csv(paste("../in\_out\_files\_", year\_run, ".csv", sep = ""))

tabmf <- read.csv("../classif\_kbas\_FINAL\_2020.csv")

subsets <- c("all","marine","terrestrial","Freshwater", "mountain") #, "mountain"

ssets2run <- 1:5 #1:length(subsets)

#ssets2run=1 #for all only

yrs = seq(1900, as.numeric(format(Sys.Date(), "%Y")), by = 1) # years to be analysed, note the max year takes the year at the time the analyses are run

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

# define the custom functions

# correct sampling function, as standard R function not correct

resample <- function(x, size, ...){

if(length(x) <= 1){

if(!missing(size) && size == 0){

x[FALSE]

} else {

x

}

} else {

sample(x, size, ...)

}

}

# x=ibadat2$year

# y=ibadat2$country

# z=ibadat2$perprotected

# zx=ibadat2$random\_year

# random allocation of year protected for those with missing years, asign from all if none found in country where x = year, y = country and z = %protected, zx = random\_year

ranyear <- function(x, y, z, zx) {

out <- as.vector(c())

for (i in 1:length(y)) {

if(zx[i] == F) {

out <- append(out, x[i], after = length(out)) #if year designated was not random, send the table straight to the output of the function (don't need to assign a year)

} else {

if(length(as.numeric(x[y == y[i] & zx != TRUE & z > 0])) > 1){

out <- append(out, resample(as.numeric(x[y == y[i] & zx != TRUE & z > 0]), 1, replace = T), after = length(out)) #for kbas with random years, if there are more than 1 year reported, use one of these for the overlap year

} else {

if (glb){

out <- append(out, resample(as.numeric(x[y == y & zx != TRUE & z > 0]), 1, replace = T), after = length(out)) #use full protected area designation year list if <2 protected area years reported in the country

} else if (reg){

out = append(out, resample(poolryears, 1, replace = TRUE), after = length(out))

}

}

}

}

return(out)

}

# cummulative number of IBAs that are totally protected

# where x = protected and y = year and id = siteid

stattotp <-function(x, y, id) {

#yrs = seq(1900, 2016, by = 1)

out <- as.vector(c())

for (i in 1:length(yrs)){

temp = tapply(x[y <= yrs[i]], id[y <= yrs[i]], sum)

out = append(out, length(temp[temp >= 98]), after = length(out))

}

return(out)

}

# cummulative mean percentage of IBAs that are protected

# where x = protected and y = year and id = siteid

statavep <-function(x, y, id) {

#yrs = seq(1900, 2016, by = 1)

out <- as.vector(c())

for (i in 1:length(yrs))

{

temp = tapply(x[y <= yrs[i]], id[y <= yrs[i]], sum)

if (glb){

out = append(out, sum(temp/100), after = length(out))

}

if (reg){

if (length(temp)>0){

out = append(out, sum(temp/100), after = length(out))

}

if (length(temp)==0){

out = append(out, 0, after = length(out))

}

}

}

return(out)

}

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

#randomisation

mkdata <- function(x, y, z, xz, zz, j) {

# where X = %protected, y=country, z=year, xz = whether year was randomly generated, zz=siteID j=number of iterations

# results are written into an array with row, column, dimension representing samples, years from 1900-2016, and count of KBAs with 100% protection & mean area protected

# res[,,1] will give counts of IBAs with 100% protection, and res[,,2] will give the means

res <- matrix(data = NA, length(x), j)

rownames (res) <- zz # write the KBA ID number to rowname

for (i in 1:j) {

ry = ranyear(z, y, x, xz)

#st = stattotp (rp, ry)

#sm = statmeanp(rp, ry)

res[,i] <- ry

print(i)

}

return(res)

}

# select subsets

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

# calculate values

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

calcvals = function(x,y) {

res <- matrix(data = NA, j, length(yrs))

for (i in 1:j) {

res[i,] <- stattotp(x, y[,i], rownames(y)) #total coverage

print(i)

print(j)

}

return(res)

}

calcvals2 = function(x,y) {

res <- matrix(data = NA, j, length(yrs))

for (i in 1:j) {

res[i,] <- statavep(x, y[,i], rownames(y)) #mean coverage

print(i)

print(j)

}

return(res)

}

##################### end functions ########################################################

### Part 2 - run randomisation for each regional grouping in turn ###############

for (w in 1:nrow(inout)){ #loops through input files (using file names from inout csv file)

#w=13 #for CMS regions only

#w=12 #for new World\_bank only

#w=10

#w=3

#w=1

# Parameters that need to be set by user

j = 100 # number of randomisations needed

infile = as.character(inout$inputfile[w]) #"Input data for R global Region KBAs.csv"

glb = F

reg = T

if (inout$global[w] == 1){ #if global

glb <- T

reg <- F

}

glb

reg

#outfile = "Output data for R global KBAs.csv"

#outfile2 = "Output data for R Region KBAs.csv"

#if (glb) {

# NOTE: Outfiles will be overwritten, including those for regions.

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

ibadat1 <- read.csv(infile, header = TRUE, sep = ",", quote="\"", dec=".", comment.char="", as.is = TRUE) # read in data

outputfile1 <- as.character(inout$outputfile1[w])

# if (glb){

# outputfile1 <- "Output data for R Country KBAs"

# }

for (k in ssets2run){

#k=1

sset <- subsets[k]

if (k == 1) {

ibadat <- ibadat1

outfile <- paste(inout$outputfile1[w], ".csv", sep="")

}

if (k > 1) {

kbasi <- tabmf$SitRecID[tabmf[,which(names(tabmf)==sset)]==1]

ibadat <- ibadat1[ibadat1$siteid%in%kbasi,]

outfile <- paste(inout$outputfile1[w], "\_", sset, ".csv", sep="")

}

head(ibadat)

regs <- unique(ibadat$region)

# if (glb){

# regs <- "global"

# }

regs

length(regs)

regres <- data.frame()

for (z in 1:length(regs)){

#z=152

#z=which(regs=="Germany")

#if (reg){

z

ibadat <- ibadat1

ibadat2 <- ibadat[ibadat$region == regs[z],]

#ibadat <- ibadat2

#}

#attach (ibadat2)

#attach (ibadat)

n <- length(ibadat2$siteid)

ibaname <- unique(ibadat2$siteid)

m <- length(unique(ibadat2$siteid))

print(paste("Number of KBAs being analysed ", m," with ",n, " sites.", sep = ""))

poolryears <- ibadat2$year[ibadat2$random\_year == FALSE] #create a list of years that were not randomly assigned, to use if a country has no random years

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

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

# start data randomization

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

output <- mkdata(ibadat2$perprotected, ibadat2$country, ibadat2$year, ibadat2$random\_year, ibadat2$siteid, j) #creates a list of years to make the error bars from (j repetitions so in this case a list of 100 possible years)

# calculate the results ## % of PAs fully covered

alldata <- calcvals(ibadat2$perprotected, output)

res <- matrix(data = NA, ncol = 7, nrow = length(yrs), dimnames = list(c(seq(1,length(yrs), by =1)),c("year", "95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage", "95CImidPercentage", "95CIhiPercentage")))

res[,1] <- yrs

for (i in 1:length(yrs)){

res[i,2] <- quantile(alldata[,i], probs = c(0.025))

res[i,3] <- quantile(alldata[,i], probs = c(0.5))

res[i,4] <- quantile(alldata[,i], probs = c(0.975))

print(i)

}

res[,5] <- res[,2]/m \*100

res[,6] <- res[,3]/m \*100

res[,7] <- res[,4]/m \*100

res2 <- data.frame(res)

names(res2) = c("year", "95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage", "95CImidPercentage", "95CIhiPercentage")

#write.table(res, file = outfile, append = FALSE, quote = TRUE, sep = ",", eol = "\n", na = "NA", dec = ".", row.names = FALSE)

# calculate the results ## % ok KBAs covered

alldata <- calcvals2(ibadat2$perprotected, output)

res <- matrix(data = NA, ncol = 7, nrow = length(yrs), dimnames = list(c(seq(1,length(yrs), by =1)),c("year", "95CIlowCount","95CImidCount","95CIhiCount", "95CIlowPercentage\_Area", "95CImidPercentage\_Area", "95CIhiPercentage\_Area")))

res[,1] <- yrs

for (i in 1:length(yrs)){

#i=23

res[i,2] <- quantile(alldata[,i], probs = c(0.025))

res[i,3] <- quantile(alldata[,i], probs = c(0.5))

res[i,4] <- quantile(alldata[,i], probs = c(0.975))

print(i)

}

res[,5] <- res[,2]/m \*100

res[,6] <- res[,3]/m \*100

res[,7] <- res[,4]/m \*100

res2$CI95lowPercentage\_Area <- res[,5]

res2$CI95midPercentage\_Area <- res[,6]

res2$CI95hiPercentage\_Area <- res[,7]

res2$region <- regs[z]

res2$code <- inout$code[w]

res2$sset <- sset

if (glb){

write.csv(res2, file = outfile, row.names = F)

}

if (reg){

regres <- rbind(regres, res2)

}

#detach(ibadat)

} #ends loop of regs

if (reg){

str(regres)

unique(regres$region)

length(unique(regres$region))

tail(regres)

write.table(regres, file = outfile, append = FALSE, quote = TRUE, sep = ",", eol = "\n", na = "NA", dec = ".", row.names = FALSE)

}

} #ends loop of subsets (marine, Freshwater, etc)

# end script

} ## ends the loop of input files

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

# Graph generation code – final\_graphs\_2020\_cleaned.R

**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 output csv files (see list below)
* If you want to produce pdf graphs of each regional grouping kba-protected coverage, set pdf\_run = T (or to F if you don’t want to produce this). Do the same for png\_run depending on whether you want to produce the graphs as pngs or not.

**Required input files:**

* CSV files:

*Note, as in the previous code, in all of the following, ‘year\_run’ is the year the analyses is being run (i.e. for this year 2020); if you are running it for a different year to the current year (e.g. running 2020 in 2019 or vice versa), you will have to replace ‘year\_run’ with the year you want to run (so the code can find the right csv file)*

* ‘Output data for R regional\_grouping.csv’ – csv files for each regional grouping with a kba-pa overlap value for each year from 1900 to the present year with confidence interverals, and how many kbas per regional grouping are fully protected. This is done for each system (all, terrestrial, marine, freshwater and mountain)

*See structure of tables in the previous section*

**Output files:**

*Note for the below – each graph is written into a folder ‘final graphs year\_run’, and within that corresponding folders based on the system and regional\_grouping of the KBAs – note this is also captured in the file names (see below for the file name templates)*

* ‘Mean perc area protectedKBA\_regional\_grouping\_system’ – graph of mean KBA protected area coverage for a given region and system – saved as png and/or pdf depending on what you select at the start
* ‘PA coverage of KBAs\_regional\_grouping\_system’ – graph of the number of KBAs with complete protected area coverage for a given region and system – saved as png and/or pdf depending on what you select at the start
* ‘Mean perc area protectedKBAs all plots regional\_grouping\_system.pdf’ – compilation of all graphs per region (where there is more than one graph) as one PDF file
* ‘PA coverage system KBAs by country region year\_run.csv’ – there will be one of these for each system and for all files

*See the previous section’s output files for structure which is the same as: ‘Output data for R regional\_grouping.csv’*

**Code Structure:**

* Part 1: set working directory, produce necessary folders for output graphs and specific what to produce
* Part 2: produce graphs for each regional grouping, system combination for total number of KBAs that are fully covered by protected areas and the avearge KBA coverage by protected areas per regional grouping
* Part 3: combine all outputs into a single output file

**R Script:**

### Code to generate PDF and PNG outputs of the kba-protected area overlaps for each regional grouping

## Part 1 - set working directory, produce folders for output graphs, and specify what to produce

year\_run <- format(Sys.Date(), "%Y") #get the current year

#TODO set working directory

folder <- paste("C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/KBA\_PA\_Overlap\_rewritten/input tables ", year\_run, sep="")

setwd(folder)

#create main folder in which to store the output graphs

if (file.exists(paste(folder, "/../final graphs ", year\_run, sep = ""))){ #create graphs folder

finfolder <- paste(folder, "/../final graphs ", year\_run, sep = "")

} else {

dir.create(file.path(paste(folder, "/../final graphs ", year\_run, sep = "")))

finfolder <- paste(folder, "/../final graphs ", year\_run, sep = "")

}

#define function needed

lu <- function (x = x){

length(unique(x))

}

cname <- "KBAs" #define for use in filenames later

#TODO - set pdf\_run to true (T) if want to produce pdf outputs of the graphs, likewise for pngs, if you don't want to produce them, set them to false (F)

pdf\_run <- T #make PDF graphs

png\_run <- T #make PNG graphs

fls <- dir(pattern = "Output data for ") #only selects the randomisation output files in the folder

fls

length(fls)

## Part 2 - produce graphs for each regional grouping, system combination for total number of KBAs that are fully covered by protected areas and the avearge KBA coverage by protected areas per regional grouping

for (w in 1:length(fls)){ #for all data

#for (w in 56:length(fls)) #for just world bank data

#w=11

fl1 <- fls[w]

fl1

tab <- read.csv(fl1) ### or KBAs

# Data fixing (where errors in names)

if (!is.null(tab$region)){

tab$region <- as.character(tab$region) #set region column to be a character variable

#Encoding(tab$region) <- "UTF-8" #fix character encoding

tab$region[tab$region == "C<f4>te d'Ivoire"] <- "Cote d'Ivoire"

tab$region[tab$region == "Cura<e7>ao (to Netherlands)"] <- "Curacao (to Netherlands)"

tab$region[tab$region == "S<e3>o Tom<e9> e Pr<ed>ncipe"] <- "Sao Tome e Principe"

tab$region[tab$region == "R<e9>union (to France)"] <- "Reunion (to France)"

tab$region[tab$region == "Western Asia<U+00A0>(M49) and Northern Africa (M49)"] <- "Western Asia (M49) and Northern Africa (M49)"

}

sset <- tab$sset[1]

desc <- tab$code[1]

desc

#check and create folder for that regional grouping in which to store graphs

if (!(file.exists(paste(finfolder, sset, sep = "/")))){ #if folder does not already exist, create it

dir.create(file.path(paste(finfolder, sset, sep = "/")))

}

#check and create folder for that system in which to store graphs

if (!(file.exists(paste(finfolder, sset,desc, sep = "/")))){ #if subfolder does not already exist, create it

dir.create(file.path(paste(finfolder, sset, desc, sep = "/")))

}

#finfolder <- paste(folder, "final graphs/", sep="/") ### "graphs\_per\_country/" should be an existing subfolder in the folder mentioned above ; change if necessary

### Set up graphs

varis <- c("percKBAcov","percKBATot")

for (y in 1:length(varis)){

#y=1

vari <- varis[y]

if (vari == "percKBATot"){

tab$mid <- tab$X95CImidPercentage

tab$qn05 <- tab$X95CIlowPercentage

tab$qn95 <- tab$X95CIhiPercentage

nfile <- "PA coverage of "

lbgr <- "% sites completely covered by protected areas"

}

if (vari == "percKBAcov"){

tab$mid <- tab$CI95midPercentage\_Area

tab$qn05 <- tab$CI95lowPercentage\_Area

tab$qn95 <- tab$CI95hiPercentage\_Area

nfile <- "Mean perc area protected"

lbgr <- "mean % area covered by protected areas"

}

cinz <- grey(0.7)

min(tab$qn95 - tab$qn05)

max(tab$qn95 - tab$qn05)

head(tab)

str(tab)

cts <- unique(tab$region[tab$region != "0"])

lu(cts)

##### produce single PDF with graphs for all countries/regions

if ((length(cts) > 1) & pdf\_run){

pdfallname <- paste(finfolder, "/", sset,"/", desc, "/", nfile, cname, " all plots ", desc, "\_", sset, ".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))

for (i in 1:length(cts)){

#i=1

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

ct1 <- as.character(cts[i])

ct1

tabi <- tab[tab$region == ct1,]

head(tabi)

basy <- 0

topy <- 100

baseyr <- 1980

if (max(tabi$qn95) < 20){

topy <- 50

}

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))

with(tabi,plot(year, mid, ylim = c(basy,topy), xlim = c(baseyr, max(year) + 1), type = "n", xlab = "Year", ylab = lbgr, yaxt = "n", main = ct1))

axis(2, seq(basy, topy, by = (topy - basy)/5))

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

with(tabi, lines(year, mid, col = 2, lwd = 2))

}

dev.off()

} #end loop one pdf for all countries

#### one graph per country

if (length(cts) == 0){

ct1 <- "global"

tabi <- tab

head(tabi)

basy <- 0

topy <- 100

baseyr <- 1980

if (max(tabi$qn95) < 20){

topy <- 50

}

if (png\_run){

png1name <- paste(finfolder, "/", sset,"/", desc, "/", nfile, cname, "\_", ct1, "\_", sset, ".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))

with(tabi,plot(year,mid, ylim = c(basy,topy), xlim = c(baseyr, max(year) + 1), type = "n", xlab = "Year", ylab = lbgr, yaxt = "n"))

axis(2, seq(basy, topy, by = (topy - basy)/5))

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

with(tabi, lines(year, mid, col = 2, lwd = 2))

dev.off()

}

if (pdf\_run){

pdf1name <- paste(finfolder, "/", sset,"/", desc, "/", nfile, cname, "\_", ct1, "\_", sset, ".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))

with(tabi,plot(year,mid, ylim = c(basy,topy), xlim = c(baseyr, max(year) + 1), type = "n", xlab = "Year", ylab = lbgr, yaxt = "n"))

axis(2, seq(basy, topy, by = (topy - basy)/5))

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

with(tabi, lines(year, mid, col = 2, lwd = 2))

dev.off()

}

# if (!is.null(dev.list())){

# dev.off()

# }

}

if (length(cts) > 0){

for (i in 1:length(cts)){

#i=1

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

ct1 <- as.character(cts[i])

ct1

tabi <- tab[tab$region == ct1,]

head(tabi)

basy <- 0

topy <- 100

baseyr <- 1980

if (max(tabi$qn95) < 20){

topy <- 50

}

if (png\_run){

png1name <- paste(finfolder, "/", sset, "/", desc, "/", nfile, cname, "\_", ct1, "\_", sset, ".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))

with(tabi,plot(year, mid, ylim = c(basy, topy), xlim = c(baseyr, max(year) + 1), type = "n", xlab = "Year", ylab = lbgr, yaxt = "n"))

axis(2, seq(basy, topy, by = (topy - basy)/5))

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

with(tabi, lines(year, mid, col = 2, lwd = 2))

dev.off()

}

if (pdf\_run){

pdf1name <- paste(finfolder, "/", sset, "/", desc, "/", nfile, cname, "\_", ct1, "\_", sset, ".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))

with(tabi,plot(year, mid, ylim = c(basy, topy), xlim = c(baseyr, max(year) + 1), type = "n", xlab = "Year", ylab = lbgr, yaxt = "n"))

axis(2, seq(basy, topy, by = (topy - basy)/5))

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

with(tabi, lines(year, mid, col = 2, lwd = 2))

dev.off()

}

# if (!is.null(dev.list())){

# dev.off()

# }

}

} # ends loop one graph per country

} # ends loop varis

} # ends loop input files

## Part 3 - create single file for outputs

fls <- dir(pattern = "Output data for ")

fls

length(fls)

fin1 <- data.frame()

for (w in 1:length(fls)){ #length(fls)

#w=7

fl1 <- fls[w]

fl1

tab <- read.csv(fl1) ### or KBAs

head(tab)

nrow(tab)

unique(tab$region)

if (!"region" %in% names(tab)){

tab$region="Global"

}

head(tab)

tab <- tab[tab$region != "0",]

nrow(tab)

unique(tab$region)

tab2 <- tab[c("year","X95CIlowCount","X95CImidCount","X95CIhiCount","X95CIlowPercentage","X95CImidPercentage","X95CIhiPercentage","CI95lowPercentage\_Area","CI95midPercentage\_Area","CI95hiPercentage\_Area","region","sset")]

tab2$file1 <- fl1

fin1 <- rbind(fin1,tab2)

}

length(unique(fin1$file1)) # should be the number of output files to merge

subsets <- unique(fin1$sset)

for (g in 1:length(subsets)){

#g=1

sset <- subsets[g]

nffile <- paste("PA coverage ", sset, " KBAs by country region ", year\_run, ".csv", sep="")

nffile

fin2 <- fin1[fin1$sset == sset,]

nrow(fin2)

unique(fin2$sset)

fin2 <- fin2[,1:11]

names(fin2) <- c("year","Count of KBAs completely covered by PAs (lower CI)","Count of KBAs completely covered by PAs (estimate)","Count of KBAs completely covered by PAs (upper CI)","% KBAs completely covered by PAs (lower CI)","% KBAs completely covered by PAs (value)","% KBAs completely covered by PAs (upper CI)","Mean % area of each KBA covered by PAs (lower CI)","Mean % area of each KBA covered by PAs (value)","Mean % area of each KBA covered by PAs (upper CI)","Global/region/country")

write.csv(fin2, nffile, row.names = F)

}

# SDG Reporting – sdg\_reporting\_output\_cleaned.R

**Instructions to run the code:**

* Set folder to the working directory where the combined outputs were written for each of the four systems – these are saved at the end of the graph code
* Create the ‘regional\_groupings\_year\_run.csv’ table by updating last year’s table with any changes (reported in the files sent from SDG) – save this as regional\_groupings\_year where year is the year of the analyses (you will have to replace year\_run with the year if the year is not equal to actual year (i.e. if you are running 2019 data in 2020 or vice versa for example)

**Required input files:**

* ‘PA coverage system KBAs by country region year\_run.csv’ – there will be one of these for each system and for all files – note we will only use the system files but you will need to read in all 5, and the ‘all’ system (i.e. includes all four systems) file will be ignored

*See the randomisation section’s output files for structure which is the same as: ‘Output data for R regional\_grouping.csv’*

* ‘regional\_groupings\_year\_run.csv’ – this gives the regional groupings needed for SDG reporting and their code

*See below for how this table is structure: note it is in two parts, country level (left) and then region level (right)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M49 Code | ISO Code | Country Name |  | M49 Code(region) | Region Name | Reference Area Type [i.e. global, SDG groupings, MDG groupings, etc.] | 2019 Submission [if data submission is required] \* Note that some regions in "SDG regional groupings" share the same composition of countries as in "MDG regional groupings". | Sequence |
| 4 | AFG | Afghanistan | 1 | World | 1.0-Global | Required | 1 |
| 248 | ALA | Åland Islands | 1 | World | 1.0-Global | Required | 1 |
| 8 | ALB | Albania | 1 | World | 1.0-Global | Required | 1 |
| 12 | DZA | Algeria | 1 | World | 1.0-Global | Required | 1 |
| 16 | ASM | American Samoa | 1 | World | 1.0-Global | Required | 1 |
| 20 | AND | Andorra | 1 | World | 1.0-Global | Required | 1 |

* ‘Iso\_countries\_year\_run’ – list of iso codes, country names and all regional groupings as explain in the kba-pa overlap code (see there for the table structure)

*See kba-protected area overlap section to see how this table is structured*

**Output files:**

* ‘119-15.1.2-2837-ER\_PTD\_FRWRT-3116.csv’, ‘119-14.5.1-2999-ER\_MRN\_MPA-3781.csv’, ‘119-15.1.2-2839-ER\_PTD\_TERRS-4864.csv’, ‘119-15.4.1-2838-ER\_PTD\_MOTN-3857.csv’ – these are the output files for SDG reporting in the current specified format (as reported in February 2020 based on end 2019 data). There is a separate file produced for freshwater, marine, terrestrial and mountain ecosystems respectively.

*See an example of one of these below:*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indicator | SeriesID | Series  Description | GeoArea  Code | GeoArea  Name | Time  Period | Value | Time\_  Detail | Upper  Bound | Lower  Bound | Source | Foot  Note | Nature | Units | Reporting Type | Series  Code | RefAreaType\_  InternalUse  Only |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2000 | 28.70833 | 2000 | 28.84892 | 28.56812 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_  FRWRT | 1.0-Global |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2001 | 29.79595 | 2001 | 29.92979 | 29.65089 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_  FRWRT | 1.0-Global |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2002 | 30.60384 | 2002 | 30.74344 | 30.48327 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_  FRWRT | 1.0-Global |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2003 | 31.28956 | 2003 | 31.41253 | 31.16584 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_  FRWRT | 1.0-Global |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2004 | 34.15294 | 2004 | 34.2781 | 34.03645 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_  FRWRT | 1.0-Global |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2005 | 35.034 | 2005 | 35.14827 | 34.92646 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_  FRWRT | 1.0-Global |
| 15.1.2 | 2837 | Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%) | 1 | World | 2006 | 35.63584 | 2006 | 35.7534 | 35.55101 | BirdLife International,  IUCN and UNEP-WCMC (2019). … |  | C | PERCENT | G | ER\_PTD\_FRWRT | 1.0-Global |

**Code Structure:**

* Part 1 – Read in input files, folders and required packages, set the working directory and fix encoding issues
* Part 2 – Create output files for each system in the SDG reporting format (based on reporting in February 2020)

**R Script:**

#### Converts output of KBA-PA output into SDG reporting output

## Part 1 - read in input files, load libraries, set working directory and fix encoding issues

library(tidyverse)

library(xlsx)

year\_run <- format(Sys.Date(), "%Y") #get the current year

folder <- paste("C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/KBA\_PA\_Overlap\_rewritten/input tables", year\_run) #set working directory

#folder <- "C:/Users/Ashley.Simkins/Documents/SDG/KBA-PA overlap/input tables 2019\_someOverrodeDec19"

setwd(folder)

reg\_grp <- read\_csv(paste('../regional\_groupings\_', year\_run, '.csv', sep = "")) #load in regional groupings needed for reporting

# fix erroneous country names

#Encoding(tab3$GeoAreaName) <- "UTF-8" #fix character encoding

reg\_grp$`Country Name`[reg\_grp$`ISO Code` == "CIV"] <- "Cote d'Ivoire"

reg\_grp$`Country Name`[reg\_grp$`ISO Code` == "CUW"] <- "Curacao (to Netherlands)"

reg\_grp$`Country Name`[reg\_grp$`ISO Code` == "STP"] <- "Sao Tome e Principe"

reg\_grp$`Country Name`[reg\_grp$`ISO Code` == "REU"] <- "Reunion (to France)"

reg\_grp$`Country Name`[reg\_grp$`ISO Code` == "ALA"] <- "Aland Islands"

reg\_grp$`Region Name`[reg\_grp$`M49 Code(region)` == 747] <- "Western Asia (M49) and Northern Africa (M49)"

isos <- read.csv(paste('../Iso\_countries\_', year\_run, '.csv', sep = "")) #load iso codes

isos <- select(isos, c('countryname', 'ISO\_SDG')) #subset to country name and ISO\_SDG code

fls <- dir(pattern = "PA coverage")#delete unwanted files

fls

length(fls)

#ecosys <- c(mar, terr, fw, mnt) #create list of ecosystem outputs to run

## Part 2 - create output files for each system for SDG reporting

for (x in 2:length(fls)){ #don't go from 1 as 1 is all sites, and we only need disaggregations by ecosystem type

#mar <- select(mar, c('Global/region/country', 'year', 'Mean % area of each KBA covered by PAs (lower CI)', 'Mean % area of each KBA covered by PAs (value)', 'Mean % area of each KBA covered by PAs (upper CI)'))

fl1 <- fls[x]

fl1

tab <- read\_csv(fl1)

tab <- tab[tab$year >= 2000,] #subset to years greater than 2000

#tab <- gather(tab, Bounds, Value, `Mean % area of each KBA covered by PAs (lower CI)`:`Mean % area of each KBA covered by PAs (upper CI)`, factor\_key = T) #restructure data so fit reporting style (wide to long format)

#tab <- select(tab, c('Global/region/country', 'year', 'Value', 'Bounds')) #subset to required columns and reorder columns

tab <- select(tab, c('Global/region/country', 'year', 'Mean % area of each KBA covered by PAs (value)', 'Mean % area of each KBA covered by PAs (upper CI)', 'Mean % area of each KBA covered by PAs (lower CI)')) #select necessary columns

tab <- unique(tab)

#colnames(tab) <- c('GeoAreaName', 'TimePeriod', 'Value', 'Bounds') #change column names to match SDG reporting

colnames(tab) <- c('GeoAreaName', 'TimePeriod', 'Value', 'UpperBound', 'LowerBound') #change column names to match SDG reporting

tab$GeoAreaName[tab$GeoAreaName=='Global'] <- 'World' #replace name global with world to fit report style

#tab$Bounds <- as.character(tab$Bounds)

#tab$Bounds[tab$Bounds == 'Mean % area of each KBA covered by PAs (lower CI)'] <- 'LB'

#tab$Bounds[tab$Bounds == 'Mean % area of each KBA covered by PAs (value)'] <- 'MP'

#tab$Bounds[tab$Bounds == 'Mean % area of each KBA covered by PAs (upper CI)'] <- 'UB'

#add in necessary columns

tab$TimeDetail <- tab$TimePeriod

tab$Nature <- 'C'

tab$Units <- 'PERCENT'

tab$`Reporting Type` <- 'G'

tab$Source <- paste('BirdLife International, IUCN and UNEP-WCMC (', max(tab$TimePeriod), '). Based on spatial overlap between polygons for Key Biodiversity Areas from the World Database of Key Biodiveristy Areas (www.keybiodiversityareas.org) and polygons for protected areas from the World Database on Protected Areas (www.protectedplanet.net)', sep = "")

tab$FootNote <- ''

# tab$GeoAreaName <- as.character(tab$GeoAreaName) #set region column to be a character variable

# #Encoding(tab$GeoAreaName) <- "UTF-8" #fix character encoding

# #tab$GeoAreaName[tab$GeoAreaName == "C<f4>te d'Ivoire"] <- "Cote d'Ivoire"

# #tab$GeoAreaName[tab$GeoAreaName == "Cura<e7>ao (to Netherlands)"] <- "Curacao (to Netherlands)"

# #tab$GeoAreaName[tab$GeoAreaName == "S<e3>o Tom<e9> e Pr<ed>ncipe"] <- "Sao Tome e Principe"

# #tab$GeoAreaName[tab$GeoAreaName == "R<e9>union (to France)"] <- "Reunion (to France)"

# tab$GeoAreaName[tab$GeoAreaName == "Western Asia<U+00A0>(M49) and Northern Africa (M49)"] <- "Western Asia (M49) and Northern Africa (M49)"

reg\_grp <- select(reg\_grp, c('M49 Code', 'ISO Code', 'Country Name', 'M49 Code(region)', 'Region Name', 'Reference Area Type [i.e. global, SDG groupings, MDG groupings, etc.]')) #select columns

reg\_grpC <- select(reg\_grp, c('M49 Code', 'ISO Code', 'Country Name'))

colnames(reg\_grpC) <- c('GeoAreaCode', 'ISO Code', 'Country Name')

reg\_grpC$RefAreaType\_InternalUseOnly <- '3.0-Country' #add country ref

reg\_grpR <- select(reg\_grp, c('M49 Code(region)', 'Region Name', 'Reference Area Type [i.e. global, SDG groupings, MDG groupings, etc.]'))

colnames(reg\_grpR) <- c('GeoAreaCode2', 'Region Name', 'RefAreaType\_InternalUseOnly2')

reg\_grpR <- unique(reg\_grpR)

tab2 <- merge(tab, reg\_grpC, by.x='GeoAreaName', by.y='ISO Code', all.x=T)

Encoding(tab2$GeoAreaName) <- "UTF-8"

tab2$GeoAreaName[tab2$GeoAreaName == "Western Asia<U+00A0>(M49) and Northern Africa (M49)"] <- "Western Asia (M49) and Northern Africa (M49)"

tab2 <- unique(tab2)

#Encoding(tab2$`Country Name`) <- 'UTF-8'

for (row in 1:nrow(tab2)){

if(!is.na(tab2$`Country Name`[row])){

tab2$GeoAreaName[row] <- tab2$`Country Name`[row]

}

}

tab3 <- merge(tab2, reg\_grpR, by.x='GeoAreaName', by.y='Region Name', all.x=T)

for (row in 1:nrow(tab3)){

if (is.na(tab3$RefAreaType\_InternalUseOnly[row])){

tab3$GeoAreaCode[row] <- tab3$GeoAreaCode2[row]

tab3$RefAreaType\_InternalUseOnly[row] <- tab3$RefAreaType\_InternalUseOnly2[row]

}

}

#categories that didn't code up for some reason

tab3$RefAreaType\_InternalUseOnly[tab3$GeoAreaName == 'Western Asia (M49) and Northern Africa (M49)'] <- '2.1-Regional (SDG)'

tab3$GeoAreaCode[tab3$GeoAreaName == 'Western Asia (M49) and Northern Africa (M49)'] <- '747'

tab3$RefAreaType\_InternalUseOnly[tab3$GeoAreaName == 'LDC'] <- '2.3-Other groupings'

tab3$GeoAreaCode[tab3$GeoAreaName == 'LDC'] <- '199'

tab3$RefAreaType\_InternalUseOnly[tab3$GeoAreaName == 'LLDC'] <- '2.3-Other groupings'

tab3$GeoAreaCode[tab3$GeoAreaName == 'LLDC'] <- '432'

tab3$RefAreaType\_InternalUseOnly[tab3$GeoAreaName == 'SIDS'] <- '2.3-Other groupings'

tab3$GeoAreaCode[tab3$GeoAreaName == 'SIDS'] <- '722'

tab3$RefAreaType\_InternalUseOnly[tab3$GeoAreaName == 'Developing'] <- '2.5-Regional (MDG)'

tab3$GeoAreaCode[tab3$GeoAreaName == 'Developing'] <- '515'

tab3 <- tab3[!is.na(tab3$RefAreaType\_InternalUseOnly), ] #filter out any sites that didn't get ref codes and so shouldn't be in the final output

#tab2 <- merge(tab, isos, by.x = 'GeoAreaName', by.y = 'countryname', all.x = T) #combine with ISO list to help matching to reporting categories

#tab2 <- tab2[nchar(tab2$GeoAreaName) > 3,] #exclude iso codes

# unique(tab3$GeoAreaName[is.na(tab3$ISO\_SDG)]) #identify countries that didn't match iso codes - should only be regions (all countries should have an assigned iso code)

#

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Anguilla (to UK)'] <- 'AIA'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Cayman Islands (to UK)'] <- 'CYM'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'China (mainland)'] <- 'CHN'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Congo'] <- 'COG'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Congo, The Democratic Republic of the'] <- 'COD'

# tab2$ISO\_SDG[tab2$GeoAreaName == "Cote d'Ivoire"] <- 'CIV'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Hong Kong (China)'] <- 'HKG'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Iran, Islamic Republic of'] <- 'IRN'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Macedonia, the former Yugoslav Republic of'] <- 'MKD'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Martinique (to France)'] <- 'MTQ'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Mayotte (to France)'] <- 'MYT'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Palestinian Authority Territories'] <- 'PSE'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Puerto Rico (to USA)'] <- 'PRI'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Russia (Central Asian)'] <- 'RUS'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Sao Tome e Principe'] <- 'STP'

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Svalbard and Jan Mayen Islands (to Norway)'] <- 'SJM' #NOR

# tab2$ISO\_SDG[tab2$GeoAreaName == 'Taiwan, China'] <- 'CHN' #TWN

head(tab3)

tab <- tab3

if (grepl('Freshwater', fl1)){

tab$Indicator <- '15.1.2'

tab$SeriesID <- '2837'

tab$SeriesCode <- 'ER\_PTD\_FRWRT'

tab$SeriesDescription <- 'Mean proportion of freshwater Key Biodiversity Areas (KBAs) covered by protected areas (%)'

}

if (grepl('marine', fl1)){

tab$Indicator <- '14.5.1'

tab$SeriesID <- '2999'

tab$SeriesCode <- 'ER\_MRN\_MPA'

tab$SeriesDescription <- 'Mean proportion of marine Key Biodiversity Areas (KBAs) covered by protected areas (%)'

}

if (grepl('terrestrial', fl1)){

tab$Indicator <- '15.1.2'

tab$SeriesID <- '2839'

tab$SeriesCode <- 'ER\_PTD\_TERRS'

tab$SeriesDescription <- 'Mean proportion of terrestrial Key Biodiversity Areas (KBAs) covered by protected areas (%)'

}

if (grepl('mountain', fl1)){

tab$Indicator <- '15.4.1'

tab$SeriesID <- '2838'

tab$SeriesCode <- 'ER\_PTD\_MOTN'

tab$SeriesDescription <- 'Mean proportion of mountain Key Biodiversity Areas (KBAs) covered by protected areas (%)'

}

#tab <- select(tab, c('GeoAreaCode', 'GeoAreaName', 'RefAreaType\_InternalUseOnly', 'Indicator', 'SeriesID', 'SeriesCode', 'SeriesDescription', 'TimePeriod', 'Value', 'TimeDetail', 'Nature', 'Units', 'Bounds', 'Reporting Type', 'Source', 'FootNote'))

tab <- select(tab, c('Indicator', 'SeriesID', 'SeriesDescription', 'GeoAreaCode', 'GeoAreaName', 'TimePeriod', 'Value', 'TimeDetail', 'UpperBound', 'LowerBound', 'Source', 'FootNote', 'Nature', 'Units', 'Reporting Type', 'SeriesCode', 'RefAreaType\_InternalUseOnly'))

colnames(tab)[colnames(tab) == 'TimeDetail'] <- 'Time\_Detail'

head(tab)

colnames(tab)

tab <- tab[

order(tab[, "RefAreaType\_InternalUseOnly"], tab[, "GeoAreaName"], tab[, "TimePeriod"]),

]

# , "GeoAreaCode", "TimePeriod"

if (grepl('Freshwater', fl1)){

#write.xlsx(tab, '15.1.2\_BirdLife\_UNEP-WCMC\_IUCN\_03.12.2019', sheetName = "15.1.2\_FreshwaterKBAs%", row.names = F)

#write.csv(tab, '../15.1.2\_BirdLife\_UNEP-WCMC\_IUCN\_03.12.2019\_fw.csv', row.names = F)

write.csv(tab, '../119-15.1.2-2837-ER\_PTD\_FRWRT-3116.csv', row.names = F)

}

if (grepl('marine', fl1)){

#write.csv(tab, '../14.5.1\_BirdLife\_UNEP-WCMC\_IUCN\_03.12.2019\_.csv', row.names = F)

write.csv(tab, '../119-14.5.1-2999-ER\_MRN\_MPA-3781.csv', row.names = F)

}

if (grepl('terrestrial', fl1)){

#write.csv(tab, '15.1.2\_BirdLife\_UNEP-WCMC\_IUCN\_03.12.2019', sheetName = "15.1.2\_TerrestrialKBAs%", row.names = F)

#write.csv(tab, '../15.1.2\_BirdLife\_UNEP-WCMC\_IUCN\_03.12.2019\_terr.csv', row.names = F)

write.csv(tab, '../119-15.1.2-2839-ER\_PTD\_TERRS-4864.csv', row.names = F)

}

if (grepl('mountain', fl1)){

#write.csv(tab, '../15.4.1\_BirdLife\_UNEP-WCMC\_IUCN\_03.12.2019.csv', row.names = F)

write.csv(tab, '../119-15.4.1-2838-ER\_PTD\_MOTN-3857.csv', row.names = F)

}

}

# Annex 1: Calculating protected area coverage of marine and terrestrial KBAs

**Summary**

Currently, to calculate protected area coverage of marine and terrestrial KBAs, we overlap KBAs with a coastline layer and code any KBA with ≥5% of its area overlapping with the land as terrestrial, any KBA with ≥5% of its area overlapping with the sea as marine, and any KBA with ≥5% of its area overlapping with both land and sea as both terrestrial and marine. All sites coded as marine are included in the marine indicator, and all sites coded as terrestrial are included in the terrestrial indicator. Both indicators therefore include some area of KBAs in the ‘wrong’ realm. We had originally planned to modify this approach by splitting coastal KBAs into their marine and terrestrial components and including only the relevant ‘slice’ of each KBA in each indicator. However, after investigation, we concluded that there are two reasons why this would not be appropriate:

1. Many KBAs would have some of their area incorrectly assigned to terrestrial and/or marine from the spatial overlap with coastline layer due to errors in the position and boundary of many coastal KBAs.
2. Generally, marine and terrestrial areas in coastal locations have been identified as separate abutting IBAs/KBAs – i.e. one for the breeding colony (terrestrial component) and one for the sea (marine component), meaning that only a small proportion of sites are genuinely both marine and terrestrial.

**Analysis**

We compared the results of a spatial overlap between KBAs and a coastline layer with the marine IBA classifications maintained by the marine team at BirdLife. Of the 2,777 KBAs triggered by seabird species, 63% (1,742) should have been classified as marine or terrestrial only. For the remaining sites, it is unclear if they should be either one of the systems or both – so the number of sites that are both marine and terrestrial is likely lower than this. Of the 1,742 KBAs that should be either marine or terrestrial, **only 58.2%** (1,013) **were correctly coded** based on the spatial overlap, because of KBA boundaries not accurately matching to the coastline owing to errors in delineation and/or position. The 41.8% incorrectly assigned sites were either coded as both marine and terrestrial when they should be one or the other, or incorrectly classified as marine when should be terrestrial, or vice versa. We did not examine the accuracy of protected area spatial boundaries but assume they suffer from the same spatial and delineation issues.

**Conclusion**

Given these results, including the fact that the proportion of sites that are truly both terrestrial and marine is small, we conclude that splitting coastal KBAs into marine and terrestrial components using an intersection with a coastline layer would generate greater errors during the calculation of the indicator of protected area coverage of KBAs than are generated by the fact that KBAs coded as both marine and terrestrial are included in the calculation of both indicators.

The marine IBA classifications should continue to be reviewed and updated, in particular for the 37% of sites where it is unclear if they should be coded as marine, terrestrial or both. The KBA spatial boundaries are also being reviewed and revised so should become more accurate over time. It therefore makes sense to rerun the analyses described here in future to determine if and when to adopt an approach of splitting the sites that are both marine and terrestrial for the purposes of generating the indicators of protected area coverage.