Analysing the drivers of biodiversity change
An introduction to R for data analysis

Introduction

In the introductory lecture of this course, we saw that anthropogenic pressure is leading to severe a biodiversity decline. The five main anthropogenic pressures are: climate change, land use change, pollution, overexploitation, and biological invasions. Although these five threats are interlinked in many ways (e.g. deforestation impacts climate change, overexploitation can open some space for non-native species, etc.), it is interesting to examine their respective importance for biodiversity loss.

In this practical, we will download, extract, and interpret information from the IUCN Red List of Threatened Species to disentangle the relative importance of the different anthropogenic threats to biodiversity.

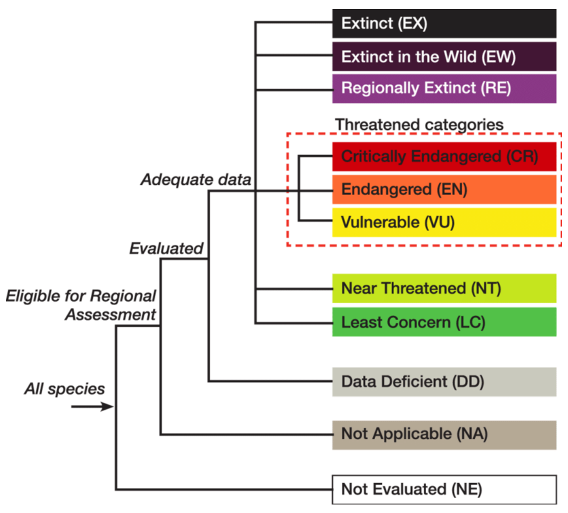
This basically what was done in Bellard et al. (2016) and Bellard et al. (2022).

Bellard, C., Cassey, P., & Blackburn, T. M. (2016). Alien species as a driver of recent extinctions. Biology letters, 12(2), 20150623.

Bellard, C., Marino, C. & Courchamp, F. Ranking threats to biodiversity and why it doesn't matter. Nature Communications 13, 2616 (2022). https://doi.org/10.1038/s41467-022-30339-y

The IUCN Red List of Threatened Species

The Red List, established in 1964, is the world's most comprehensive information source on the global extinction risk status of animal, fungus and plant species. It uses data on trends in population size, in geographic range, and analyses of probability of extinction within 20 years to assign each species to a threat category:



The methodology for assigning a species to a category is explicitly detailed in the IUCN Red List categories and criteria, version 3.1, second edition, available there: https://portals.iucn.org/library/node/10315.

For most assessed species, it also indicates the pressures that are threatening them. These pressures are classified following a Unified Classifications of Threats and Actions (see table in the other word document, taken from Salafsky et al. 2007: “Threats by level of classification.docx”).

Get the data from the IUCN Red List online portal

Go to https://www.iucnredlist.org/

Create a free account.

Click on “Advanced”.

Here we will assess the effect of the different threats on extinction events: select the categories “Extinct” & “Extinct in the wild” in the menu on the left hand side.

We now need to select the categories from the table above that correspond to the five general threats. Unfortunately, we need to make this selection for each threat separately: as you will see, the categories do not appear in the file you will download.

We would therefore need to select the following categories for each threat:

Threat	Category
Land use change	1, 2, 3, 4, 7.2
Overexploitation	5
Biological invasions	8.1
Pollution	9
Climate change	11

Make this selection for land use change.

Once you have finalized your selection, click on “Download” -> “Search results”. In the window that opens, indicate you need it for a practical, click on “Next”. Then answer “Yes” to the question “*Are the data requested for academic, research or educational use? See examples*.” You will then see a summary of your request, which should be pre-approved, and you will receive an automated email to confirm it. Refresh the page after a few seconds, you should be able to download a zip file.

Unzip the zip file, you will get a folder containing four files:

- “assessments.csv”
- “ReadMe.txt”
- “taxonomy.csv”
- “IUCN Red List_Terms and Conditions of Use_v3.pdf”

The “csv” extension stands for “comma-separated values”. These are text files containing the data we will work on. Each row contains a series of values (they can be numbers or text, we will see this in more details below), separated by a comma (sometimes other separators are used, such as semi-colons or tabulations).

Rename your folder “landuse”.

You could do this for each threat, which would generate 5 folders, one for each threat (you would also have to rename the four other folders as “overexploitation”, “invasion”, “pollution” and “climate”). To save some time, you can just download the zip file from Learn.

Pro tip: select what information you want to be included in the csv file

The IUCN Red List website is not very intuitive for this. By default, the csv file will only contain basic information, but you can add columns to it. To do so, go to your profile, and click on “Edit profile”. If you scroll down, you will see a list of elements called “DOWNLOAD OPTIONS”. You can select those you want to be included in your csv file.

Analyses

We will use R to analyse this data (and for all the practicals for this course actually).

First you need to define the working directory, i.e. the folder where all your folders, data files and scripts will be stored.

Import the data

Read the `read.csv()` help file (type `?read.csv` in the console) and import the “assessment.csv” file for each threat. read.csv() will create a data frame, so give a name to each data frame to keep track of the threat they correspond to.

Each data frame should have 23 columns. Each column represents a variable, whose name is contained in the header of the data frame. You can check the class of each variable with either of the following lines of code:

```
unlist(lapply(pollution, class))
str(pollution)
```

Don’t worry about understanding how `lapply` *works for now, but if you are interested, it is described in the “Introduction to R” document. Try excuting only* `lapply(pollution, class)`. *What is the difference? What does it mean?*

Often, you will work with variables of different types, and it will therefore be important to check that R has automatically assigned the correct type to each variable. Here we see a discrepancies: variable `populationTrend` was set to “logical” for the `pollution` data frame, whereas it was set to “character” for the others. This is because it contains no data, and R did that automatically to save some memory (a logical uses less memory than a character since it can only take 2 values, TRUE or FALSE). Here it does not matter because we will not work with this variable, but it could have been a problem otherwise.

In R, selecting a column in a data frame is performed with the symbol $. To get the vector of all species names for the “climate” data frame, one just write:

```
climate$scientificName
```

It is always a good idea to do some checks to your data. Here, we can check if no species is cited multiple times. To do so, we can compare the number of rows in the data frames with the number of unique species names:

```
nrow(landuse)
length(unique(landuse$scientificName))
```

What is the issue here? To remove duplicates, we can execute the following code:

```
if(nrow(landuse) != length(unique(landuse$scientificName))){
  landuse <- landuse[-
which(duplicated(landuse$scientificName)),]
}
```

`!=` means “not equal to”, i.e. “different from”. Do this for all data frames.

Instead of five different data frames, it could be more convenient to work with a single one containing all the data. For very large datasets especially, that will save some memory. However, as mentioned above, the csv files and therefore your data frames do not have a column indicating the threat category. We therefore need to add these. Here we have two options:

1. We can add a single column to each data frame indicating “land use”, or “overexploitation”, etc. However, some species are affected by multiple threats, and the five data frames have duplicate rows, that we will want to merge later.
2. We will therefore prefer to add five columns, one for each threat, and fill it with 0’s (if this is not the threat corresponding to the data frame) or 1’s (if this is the threat corresponding to the data frame).

To do so, you just need to assign the desired values to the new columns by writing the following code:

```
climate$landuse <- 0
climate$overexploitation <- 0
climate$invasion <- 0
climate$pollution<- 0
```

```
climate$climate <- 1
```

Because a value is assigned to it, the columns are automatically created and appended to the end of the data frame. Even though you assign a single value to a column with multiple ones, R automatically fills the whole column with this value.

Do this for each data frame (you need to change which variable gets filled with 1 of course!).

We will now combine all data frames together. All our data frames have exactly the same variables, i.e. the same column names with the same type (but of course each data frame has a different number of rows). If not, you messed up somewhere, go back to the previous instructions.

We can therefore (and we will) simply stack the data frames on top of each other to create a big data frame with the same number of columns as each of the smaller data frames, but the sum of their rows. We will do so using the function `rbind()` (check the documentation):

```
threats.dat <- rbind(landuse, overexploitation, invasion,
pollution, climate)
```

We still have a problem: some species were affected by multiple threats, and therefore appear multiple times in this data frame. We can indeed check the number of unique species names:

```
length(unique(threats.dat$scientificName))
```

which is lower than the number of rows, that we can get with:

```
dim(threats.dat)
```

We therefore need to merge these duplicated rows and delete redundancies. First, let's find which rows are duplicates of other rows with a smaller subscript (check the documentation for functions `which` and `duplicated`):

```
threats.dup <- which(duplicated(threats.dat$scientificName))
```

We can get this subset of the whole data by writing:
`threats.dat[threats.dup,1:23]`. Even this subset can have multiple identical rows. Why?

We need the unique rows from `threats.dat[threats.dup,1:23]`, so we can use `distinct()` from package `dplyr`:

```
threats.redundant <- distinct(threats.dat[threats.dup,1:23])
```

Now, for each of these unique rows, we need to merge all the rows that are duplicates. We will use a for loop to do so. A for loop is used to iterate over a sequence of values. These values can be numbers, in any order, but also a series of characters or words for example. Here we want to iterate over all unique rows, which are stored in `threats.redundant`. We will therefore write:

```
for(i in 1:nrow(threats.redundant)){

}
```

`i` will iterate from 1 to the number of rows of `threats.redundant`, with an increment of 1. The `{}` brackets are written around all the commands that will be executed iteratively within the for loop.

Within this for loop, we first need to select the rows that correspond to the row i in `threats.redundant`. To do this, we will use function `which()`, as previously:

```
row.number <- 
which(threats.dat$scientificName==threats.redundant$scientific
Name[i])
```

Explain in your own words what this line of code is doing and why. Can you figure out why we used "==" here?

We will then type:

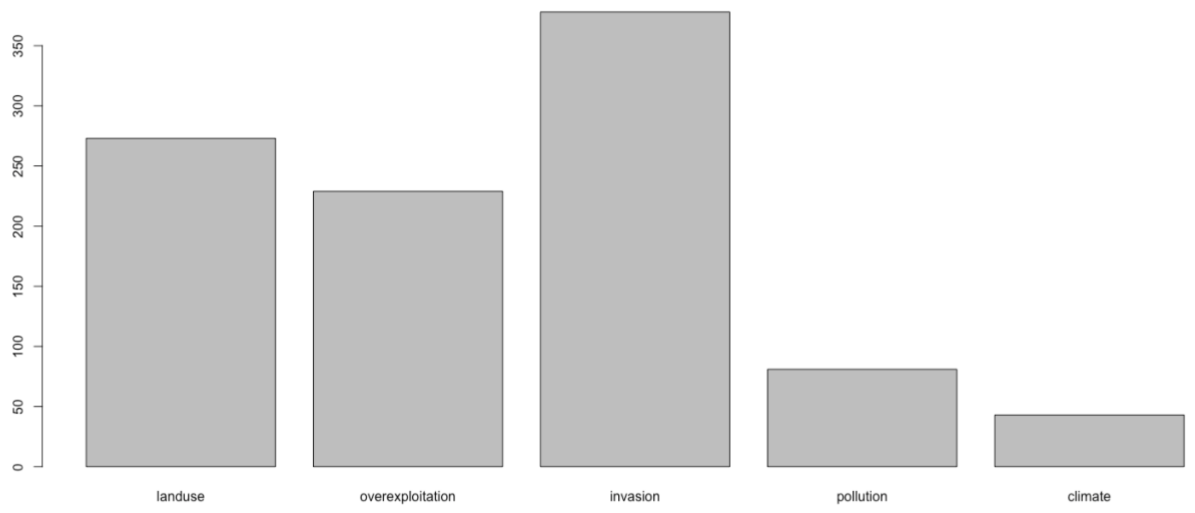
```
row.selected <- threats.dat[row.number,]
new.row <- 
data.frame(c(row.selected[1,1:23],colSums(row.selected[,24:28]
)))
threats.dat <- threats.dat[-row.number,]
threats.dat <- rbind(threats.dat,new.row)
```

Again, explain in plain words what each of these lines of code is doing. Tip: set i <- 1, execute each line one by one and check what is stored in `row.selected` and `new.row`, and how `threats.dat` changes. You can use `dim()` to check the dimension of `threats.dat`.

Let's look at general patterns

Which are the most important threats to biodiversity, based on the number of extinct species?

We can simply look at the size of the data frames of each threats: their number of rows indicates the number of species impacted by the threat. Or more conveniently, we can use our merged data frame to get these numbers one line of command. How would you do it? (Tip: the function to use is used some of the code above). Once you have these numbers, use `barplot()` to display them:



As we discussed earlier, some species are impacted by multiple threats. To visualize this, we could use a Venn diagram that will show the number of species threatened by all possible combinations of threats (https://en.wikipedia.org/wiki/Venn_diagram#:~:text=A%20Venn%20diagram%20is%20a,statistics%2C%20linguistics%20and%20computer%20science), using the package ggVennDiagram.

To install and load the package, use the code below:

```
if (!require(devtools)) install.packages("devtools")
devtools::install_github("gaospecial/ggVennDiagram")
library("ggVennDiagram")
```

The function `ggVennDiagram()` needs a list as an input. A list is a way to store multiple elements of potentially different types and sizes in a single object. Here we will create a list containing vectors of species names affected by each threat.

As before, we can create it from the original data frames:

```
x <- list(
  landuse=landuse$scientificName,
  overexploitation=overexploitation$scientificName,
  invasion=invasion$scientificName,
  pollution=pollution$scientificName,
  climate=climate$scientificName
)
```

Or we can write a more general and flexible code to generate the exact same thing:

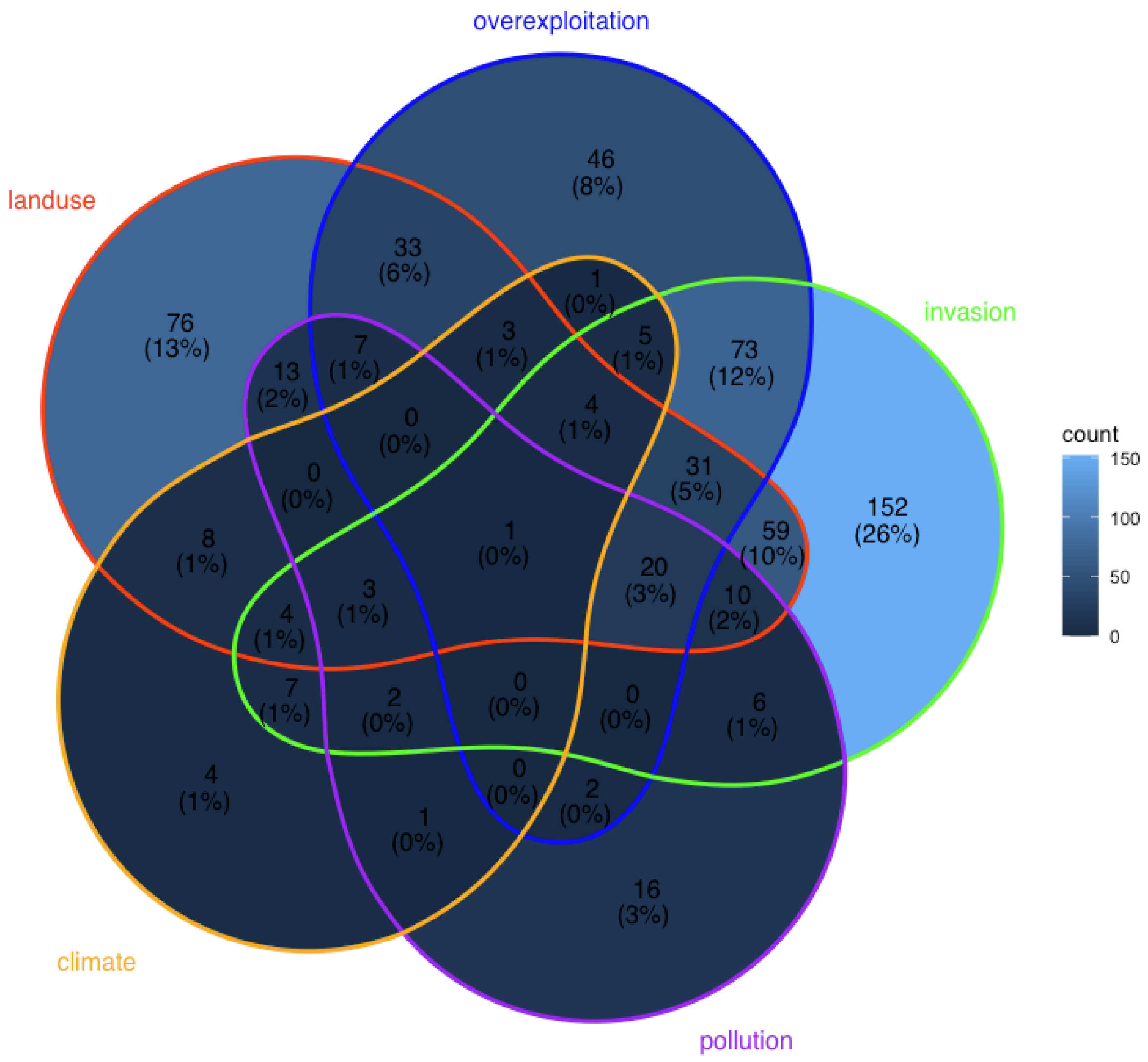
```
x <- list()
for(i in 1:5){
  x[[i]] <-
threats.dat$scientificName[which(threats.dat[23+i]==1)]
}
```

```
names(x) <- names(threats.dat[24:28])
```

Take some time to understand what this code is doing. If you wonder why we use a double bracket `[[i]]`, go to: https://cran.r-project.org/doc/manuals/R-lang.html#Indexing.

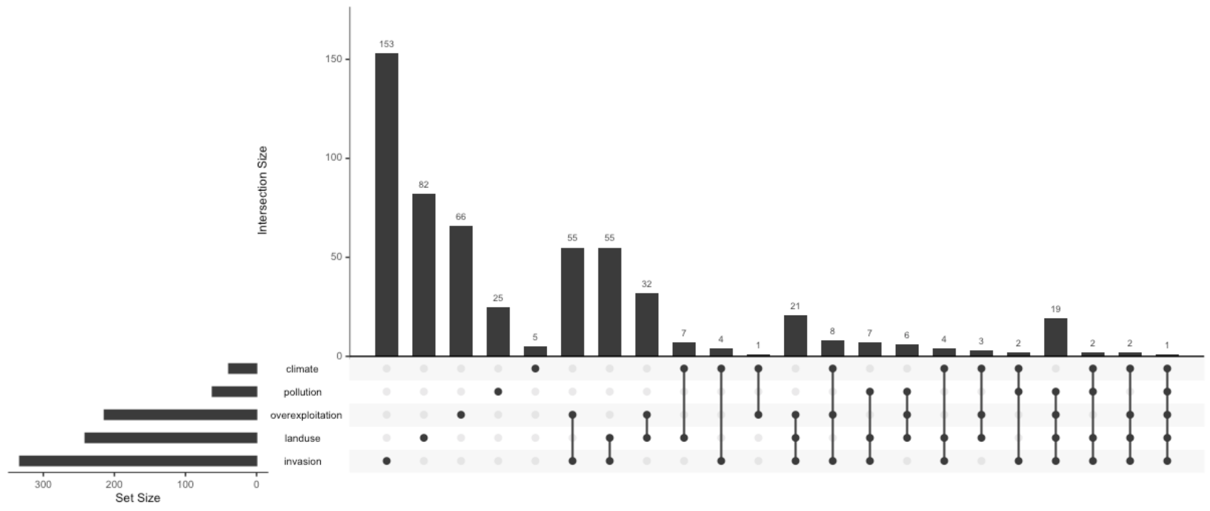
We can then plot the Venn diagram:

```
ggVennDiagram(x, label_alpha = 0 ,
set_color=c("red","blue","green","purple","orange"))
```



The issue with Venn diagrams is that they become very hard to read when we have more than 3 or 4 categories. Instead, we can use an UpSet plot using function `upset()` from package `UpSetR`:

```
if (!require(UpSetR)) install.packages("UpSetR")
library(UpSetR)
upset(data=threats.dat[c(3,24:28)],nsets=5)
```



Patterns for different taxonomic groups

In the previous analyses, we explored the importance of the five main anthropogenic threats for driving species extinction, using all species from the IUCN Red List. Now we will explore if these patterns are consistent across different taxonomic groups.

Does `threats.dat` have a column allowing to distinguish between taxonomic groups?

We need to get this information from the "taxonomy.csv" files downloaded previously. Our first task is to import these files to create data frames, and to combine these data frames into a single one where each species appears only once. Based on the code above, you should be able to do this (it only takes 2 lines of code once you have read the csv files and created the 5 new data frames, one using `rbind()`, the other using `distinct()`).

Let's name our combined data frame containing the taxonomic information "`all.tax`". `all.tax` should have the same number of rows as `threats.dat`.

We now need to relate it to `threats.dat`. To do so, we need the make sure that the species described by each row of `all.tax` is also described by the same row number in `threats.dat`. The easier way to do so is to sort the data frames by the species name:

```
all.tax <- all.tax[order(all.tax$scientificName),]
threats.dat <- threats.dat[order(threats.dat$scientificName),]
```

Check the documentation for function order(), and explain in your own words what this code does.

We can check how many classes are included in this dataset and how many species belong to each class by typing:

```
table(all.tax$className)
```

We will do these analyses for the best represented classes, i.e. fishes (`all.tax$className == "ACTINOPTERYGII"`), mammals (`all.tax$className == "MAMMALIA"`), birds (`all.tax$className == "AVES"`), amphibians(`all.tax$className == "AMPHIBIA"`), gastropods (`all.tax$className == "GASTROPODA"`), reptiles (`all.tax$className == "REPTILIA"`) and a class of flowering plants (`all.tax$className == "MAGNOLIOPSIDA"`).

We can now extract the data from threats.dat for each taxonomic group by selecting the rows in `threats.dat` that correspond to the desired class in `all.tax`. Write the code for this, using function `which()`.

Plot the upset plots for each taxonomic groups, and compare them visually to each other and to the global one. Are the different taxonomic groups affected by the same threats?

<u>Temporal patterns</u>

Now, we will look at temporal trends for all the data and for each taxonomic group, to see if we observe some change in the prevalence of a threat over another, or if it has been relatively consistent over the years. We will use the column `yearPublished` for convenience, because it is already in the appropriate format.

What other column(s) could we have used? What is the challenge, and how would you address it?

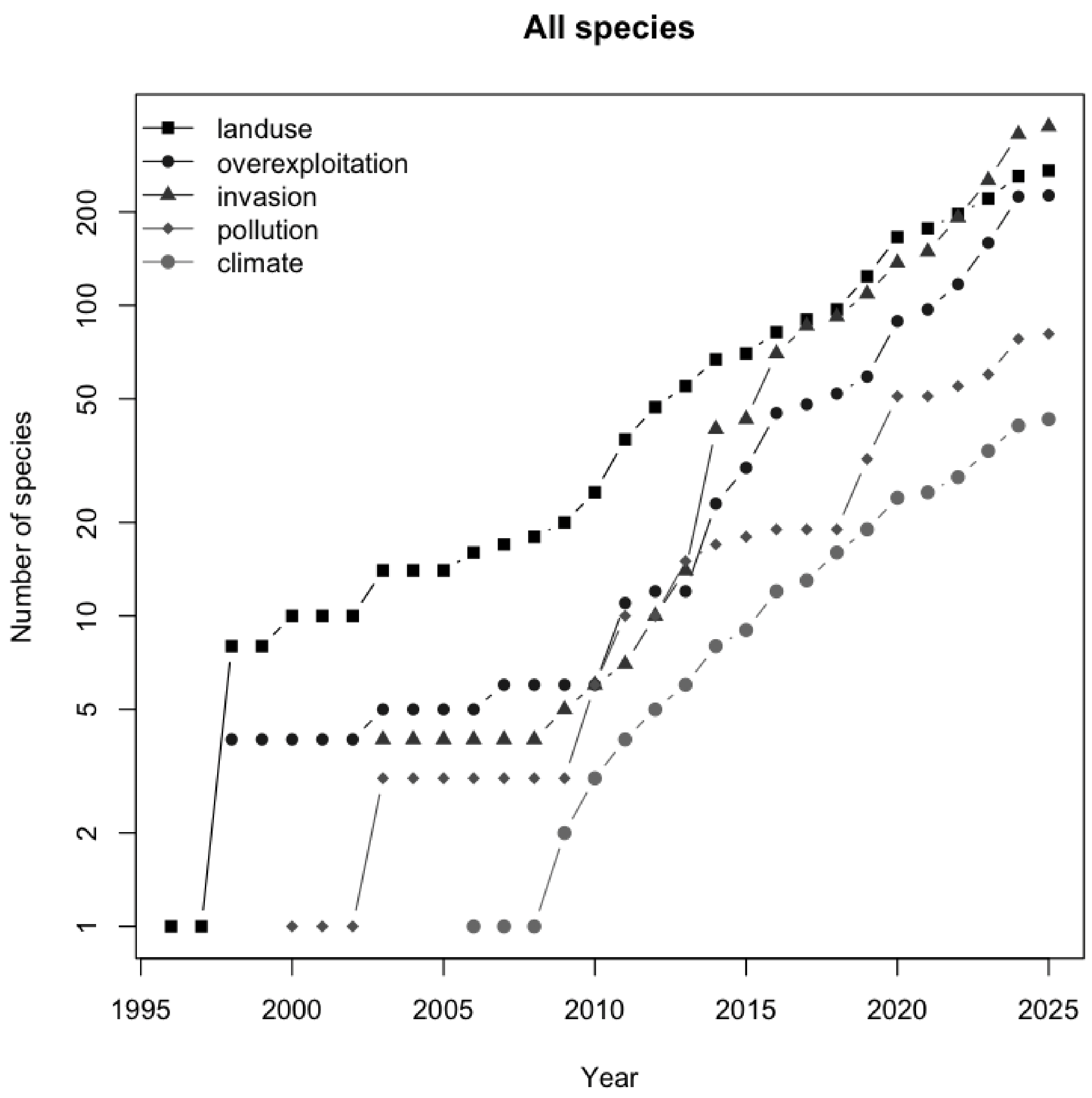
Let's first look at the period covered:

```
range(threats.dat$yearPublished)
```

That indicates us that we have data from 1996 to 2025. You will therefore use a for loop to select the data published before each year.

- First, select only the data before 1996
- Then, compute the sum of the columns 24 to 28 of the data frame and store it in a new variable `threats.time`
- Start a for loop for years 1997 to 2025
- Select only the data before the year in question
- Compute the sum of the columns 24 to 28 of the data frame and append it to `threats.time` using rbind

You will then plot changes in the number of species impacted by each threat through the years. It should give you something like this for the whole data (all taxonomic groups together – note the log scale on the y axis):



This was generated with this code:

```
plot(x=1996:2025,y=threats.time[,1],type =
"b",pch=15,ylim=c(1,max(threats.time)),xlab="Year",ylab="Numbe
r of species",main="All species",log="y")
lines(x=1996:2025,y=threats.time[,2],type =
"b",pch=16,col="gray10")
lines(x=1996:2025,y=threats.time[,3],type =
"b",pch=17,col="gray20")
lines(x=1996:2025,y=threats.time[,4],type =
"b",pch=18,col="gray30")
lines(x=1996:2025,y=threats.time[,5],type =
"b",pch=19,col="gray40")
legend(x="topleft",legend=c("landuse","overexploitation","inva
sion","pollution","climate"),bty="n",lty=1,pch=15:19,col =
c("black","gray10","gray20","gray30","gray40"))
```

Interpretation and discussion

After analysing data and describing your results, it is important to examine and discuss them critically, ideally in light of the literature. A discussion will always have some part of subjectivity, and it is hard to be fully exhaustive (as someone else may interpret the results under another light), but it is probably the hardest and most interesting part of a research article.

Let's as a few questions on the results to start thinking about a few things.

<u>Is there a clear worst threat?</u>

What can you say about these threats? Look at the temporal trends. Where should we invest if we want to prevent species extinction?

<u>Are extinct species from the Red List the best way to measure biodiversity loss?</u>

What other indices or metrics do you know that can or have been used to measure biodiversity loss? Do you think the conclusions could be different if these were used to assess the importance of the different anthropogenic threats?

<u>Can we infer causality from this data?</u>

When multiple threats are observed for the same species, are they independent from each other?

Reflect on the structure of the database

What do you think of the database (the csv files)? Compare the data from the sheet of specific species and the data in the csv file: is there anything major missing?

This would not be part of a discussion, but it is good to start looking at other databases to build yours for the Professional Skills course.

That was too easy and you got yourself extra time?

Do the same analyses using the `yearLastSeen` *variable for the temporal trends (i.e. implement the solution you proposed above).*

Do the same analyses for terrestrial vs marine species.

Do the same analyses for Oceania vs Europe vs South America vs Africa.

Do the same analyses for other threat categories.