Basic spatial R workshop - 2

Tobias Andermann

5/24/2020

Empirial data tutorial

Dependencies:

```
library(rgbif)
library(sp)
library(raster)
library(sf)
```

1. Occurrence count in raster

In this tutorial you will download species occurrence data of your species of choice from GBIF and plot the number of occurrences per cell as a measure of population density of the species for a country of choice.

> Task: Choose your own species and country when going through the following steps. Make sure you pick a species with sufficient available records (see command for checking this in the next code cell), otherwise the resulting raster will look quite boring. In the example I'm working with the species Aquila chrysaetos (golden eagle) in Sweden.

Getting the data

Here we we use the R package `rgbif` to download occurrence data directly from GBIF into R. GBIF is a large central database that contains all kinds of species occurrence records. Let's see how many occurrences we can find for our target species in the country of our choice. You can use the `publishingCountry=` option to only extract occurrences from a given country. Another useful argument is `hasCoordinate = T`, which only allows records with valid coordinates. You can choose to only extract summary data `return = "meta"` in order to first get an idea how many data points there are, before starting to download. This is recommendable because it can take a very long time to download tens of thousands of occurrences, so it's best to know beforehand what you are in for.

```
library(rgbif)
occ_search(scientificName = "Aquila chrysaetos", publishingCountry='SE', hasCo
ordinate = T, return =  "meta")$count
```

```
## [1] 79590
```

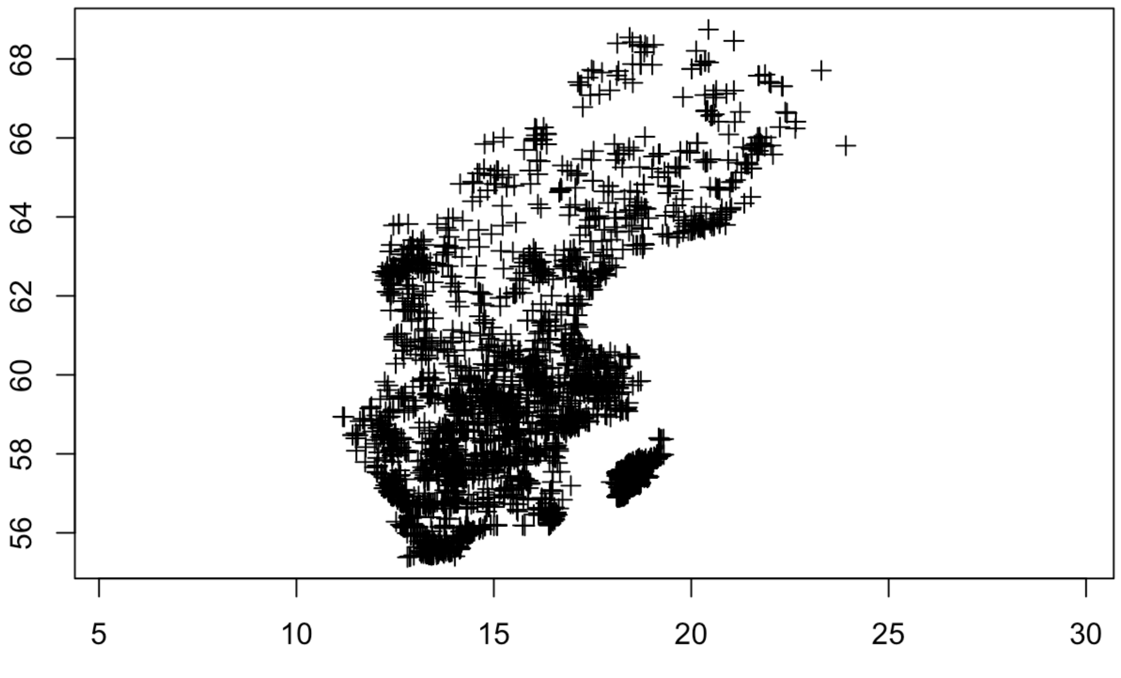
As you can see there are a lot of records in my case. What does it look like for your target species? Let's now download the actual data containing the coordinates for our target species by setting `return = "data"`. For this exercise you can keep the number of data points in a managable range by setting a limit, e.g. `limit=5000`:

```
golden_eagle <- occ_search(scientificName = "Aquila chrysaetos", publishingCou
ntry='SE',hasCoordinate = T,return =  "data", limit = 5000)
#library(tidyverse)
#write_csv(x = golden_eagle, path = "../data/golden_eagle.csv")
```

This is how you extract the longitude and latitude information from the `occ_search()` output.

```
lon = golden_eagle$decimalLongitude
lat = golden_eagle$decimalLatitude
```

Task: Turn the coordinates into a `SpatialPoints` object, as we did in the first tutorial. Store the resulting object under the name `species_occurrences` . You will need to have the `sp` package loaded for this `library(sp)` . Remember to first turn the coordinates into a 2D array and then transform them into a spatial object. Then plot the points. You can add the axes to your plot by adding `,axes=T` to your `plot()` command.



Loading a country map

I stored a file containing simple maps of all countries in the `data/ne_10m_admin_0_countries/` folder (available on the GitHub repo, which you probably already have downloaded during the first tutorial).

Task: Read the shape data of all countries, as we did for the shape files in the first tutorial and store it under the name `all_countries`. Remember you need to load the `sf` package for this, if it's not already loaded.

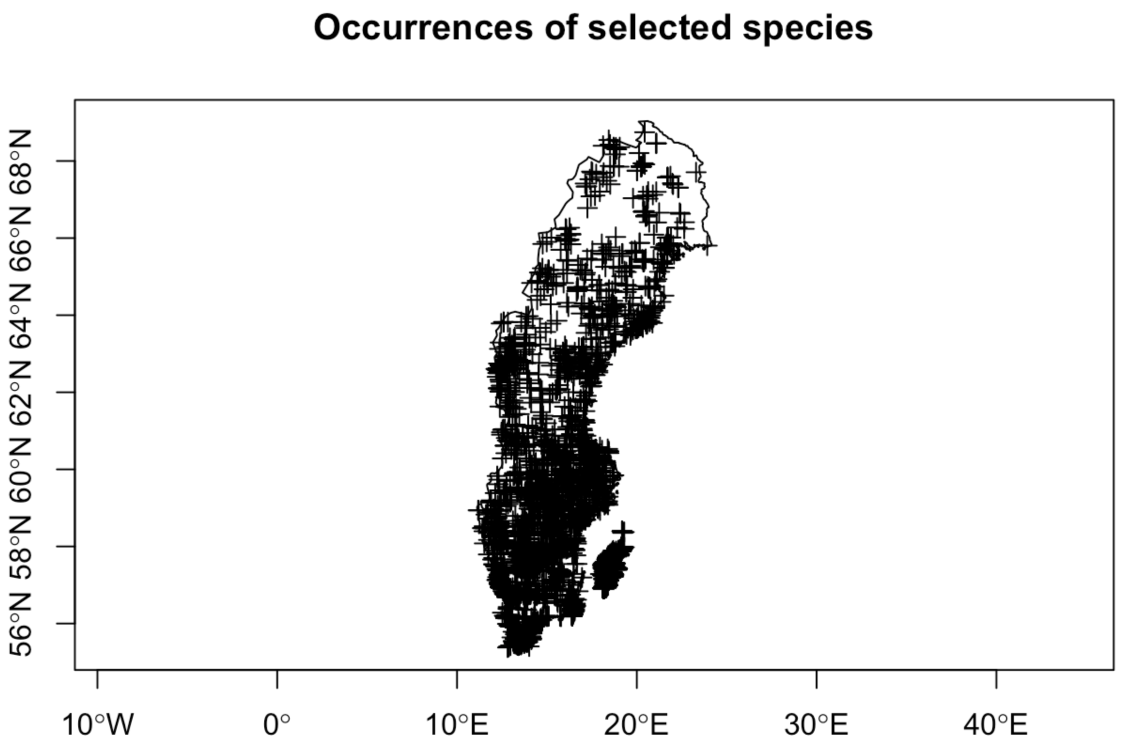
```
## Reading layer `ne_10m_admin_0_countries' from data source `/Users/tobias/Gi
tHub/spatial_R_course/data/ne_10m_admin_0_countries/ne_10m_admin_0_countries.s
hp' using driver `ESRI Shapefile'
## Simple feature collection with 255 features and 94 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -180 ymin: -90 xmax: 180 ymax: 83.6341
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

Extract the data for your country of choice from the `all_countries` object and convert the sf object into a spatial object using the `as(...,'Spatial')` function. The country names are stored in the `SOVEREIGNT` column of the dataframe. In the example below I first extract the index where the `SOVEREIGNT` column has the value `SWE` for Sweden. Look up here (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-3) what the 3-letter country code for your country of choice is.

```
selected_country_index = which(all_countries$SOVEREIGNT == 'Sweden')
selected_country_sf = all_countries[selected_country_index,1]
selected_country = as(selected_country_sf, 'Spatial')
```

Now plot the data points on the map of your country:

```
plot(selected_country,main='Occurrences of selected species',axes=T)
plot(species_occurrences,add=T)
```


Occurrences of selected species
56°N 58°N 60°N 62°N 64°N 66°N 68°N
10°W
0°
10°E
20°E
30°E
40°E

Count occurrences per grid cell

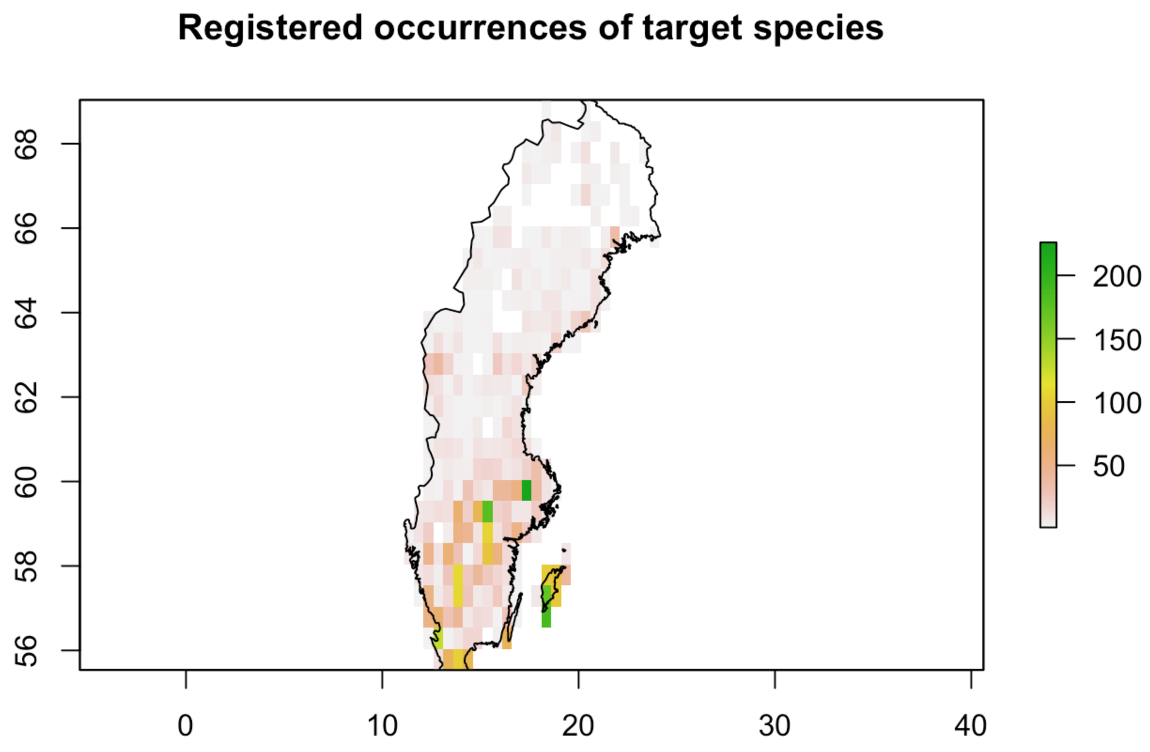
Now convert the country polygon into a raster and then count for each cell the number of occurrences of your species of choice. This will give you an idea where the species occurs in higher densities.

When using the `raster()` function you can set the resolution (in degrees in this case) to any value you like. The smaller the value, the smaller the cells and hence the more cells are being produced.

Then use the `rasterize()` function to rasterize the point data, which will count how many points fall into each cell. You can do this by calling the `rasterize()` function and providing the `SpatialPoints` object followed by the `raster` object. You also need to tell the function what needs to be done using the `fun=` settings, in our case you can use the keyword `'count'`, which will count the number of points that fall within each cell.

```
library(raster)
raw_raster <- raster(selected_country,resolution=0.5)
counts_per_cell = rasterize(species_occurrences,raw_raster,fun='count')
```

Task: Plot the resulting raster values of occurrence counts per cell and the polygon of your target country on top.



B) Plotting mammal diversity

In the exercise above we counted the number of occurrence points for each cell of a given raster. Now we are going to use the information of many individual presence/absence rasters (cells coded as 1 or 0) for all mammal species and add the values in each cell across all rasters in order to plot a map of global mammal diversity.

As input data we are going to use the "present natural" rasters, produced by a recent study by Faurby et al, 2015 ('Historic and prehistoric human-driven extinctions have reshaped global mammal diversity patterns'), who will also be teaching during this course. In this study the authors modeled ranges of each mammal species under a scenario of no human disturbance (i.e. where would species most likely occur if no humans were present).

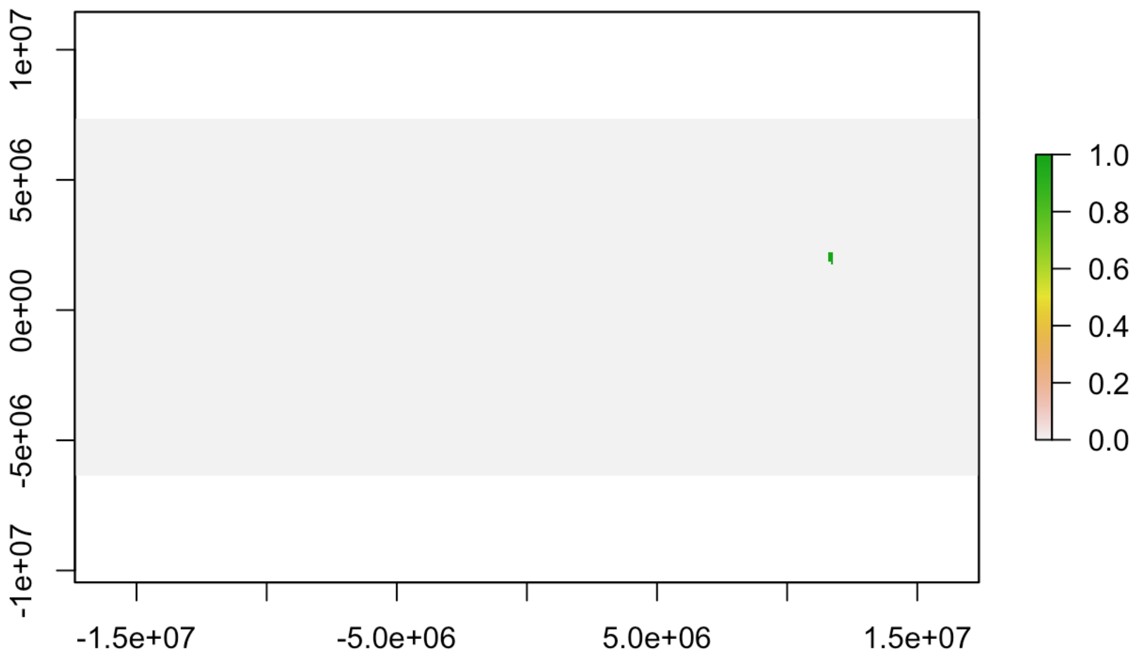
The data is available via the Phylacine database (https://megapast2future.github.io/PHYLACINE_1.2/). Click on the link in the 'Download the data' section to start the download. Unzip the downloaded file and copy it into your data folder that you downloaded from the course GutHub repo. The raster data we're after is stored in the `Data 2/Ranges/Present_natural/` folder (Windows users remmeber to replace all `/` with `\\`). You can see all files that are present in that folder using the `list.files()` command:

```
species_ranges=list.files('../data/Data 2/Ranges/Present_natural/')
```

The folder contains a separate raster file for each species. To read a raster file we can just use the `raster()` function and the file name. For example we can just select the first filename from our `species_ranges` list of files using `species_ranges[1]`. To actually find the file we need to join the filename with the path of the folder it's stored in, which we can do with the command `paste0()`

that allows you to join to strings:

```
filepath = paste0('../data/Data 2/Ranges/Present_natural/',species_ranges[1])
raster = raster(filepath)
plot(raster)
```



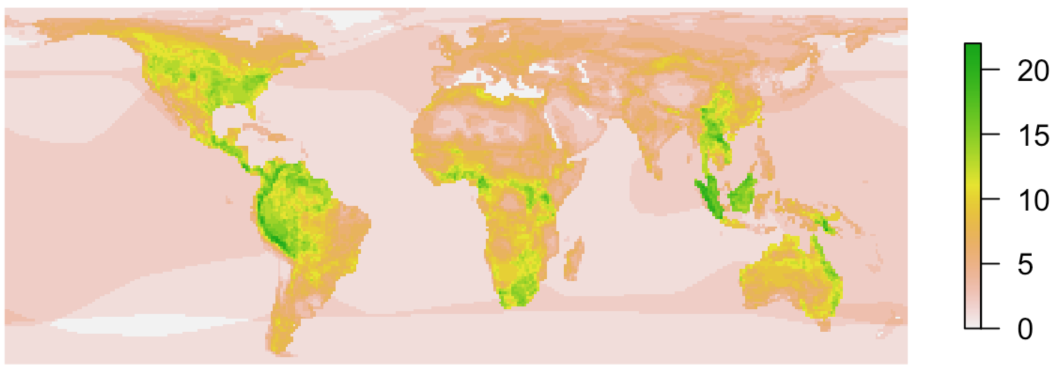
In the rasters each cell is coded as 0 (species absent) or 1 (species present). You can see that the first raster file belongs to a species with a rather small distribution. Let us now create a map with the global present natural mammal diversity (if humans had never effected mammals distributions). For this we can simply iterate through all raster files (using a `for(){}` loop, see here (https://rspatial.org/intr/10-flow.html#for-loops) for a quick tutorial on for-loops in R) and keep adding the rasters up, one after another, until we have a final raster with the total diversity for each cell. We first initiate the raster using the first file of the file list and then will go through the remaining files to add them one by one to the newly created raster object.

Reading all raster files will take quite a while (~ 30 min). If you don't have the time or patience to wait around you can just select 500 random species from the file list using the command `species_ranges = sample(species_ranges,500)` at the beginning of the code block below.

```
library(raster)
species_ranges = sample(species_ranges,500)
merged_rasters = raster(paste0('../data/Data 2/Ranges/Present_natural/',specie
s_ranges[1]))
for (i in species_ranges[2:length(species_ranges)]){
    species_raster=raster(paste0('../data/Data 2/Ranges/Present_natural/',i))
    merged_rasters = merged_rasters+species_raster
    }
```

Plot the resulting raster of potential mammal species diversity:

```
plot(merged_rasters,axes=F,box=FALSE)
```



You can already see that mammal diversity is distributed unevenly across the globe. However we have not yet delimited the geographic features, the only reason why you can see where the continents are placed is because the structure of the diversity data. Now we want to delimit land from water, by plotting a world-map polygon on top of this raster.

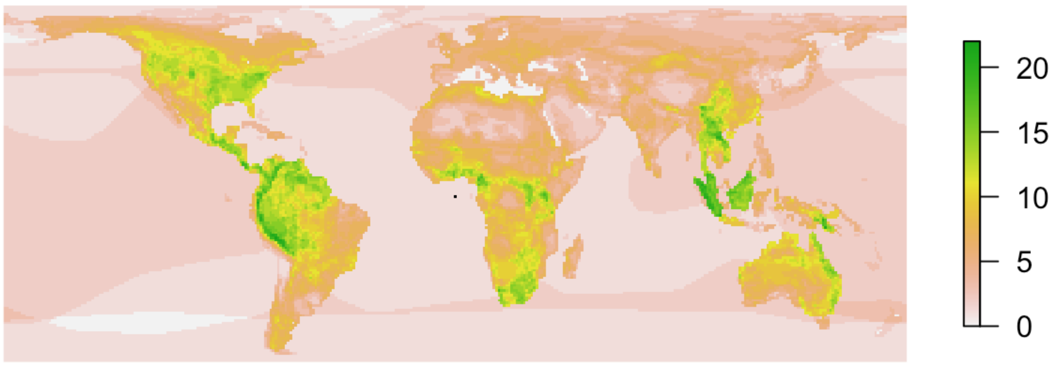
Load the world map stored at `global/ne_50m_land/ne_50m_land.shp` and turn it into a spatial object.

```
library(sf)
world = st_read('../data/global/ne_50m_land/ne_50m_land.shp')
```

```
## Reading layer `ne_50m_land' from data source `/Users/tobias/GitHub/spatial_
R_course/data/global/ne_50m_land/ne_50m_land.shp' using driver `ESRI Shapefile
'
## Simple feature collection with 1420 features and 3 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -180 ymin: -89.99893 xmax: 180 ymax: 83.59961
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

```
world_spatial = as(world, 'Spatial')
```

Task: Now plot the world polygon on top of the raster data to delimit land from water (hint: it's supposed to not work out properly, see next section).



Transforming coordinate system

What went wrong here? Do you see the little black dot in the center of the plot? That's our world polygon! It looks like the coordinate system of our raster data does not match with that of the world map polygon (different coordinate reference systems). Let's check the projections of the two objects:

```
projection(merged_rasters)
```

```
## [1] "+proj=cea +lon_0=0 +lat_ts=30 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_
defs +ellps=WGS84 +towgs84=0,0,0"
```

```
projection(world_spatial)
```

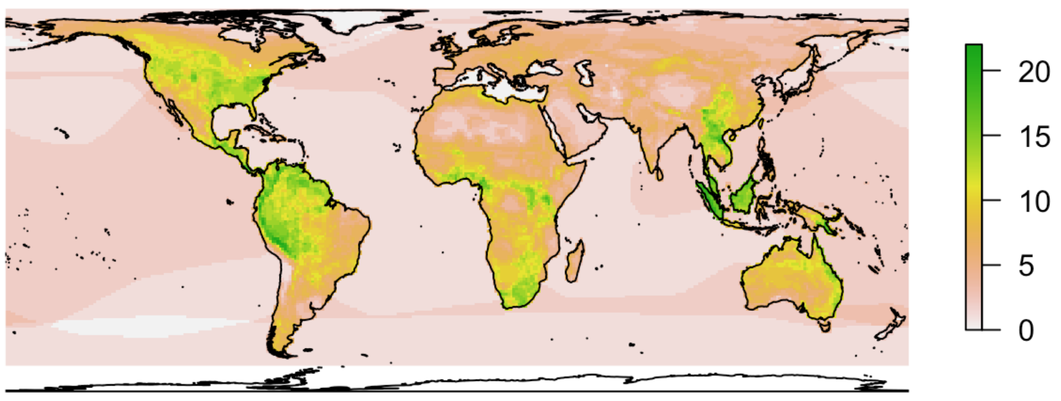
```
## [1] "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
```

They are indeed different! You don't need to understand the different projections in detail but all you need to know is how to convert one projection into another. There are different tools to do this, some of the commonly used ones are the `spTransform()` function of the `rgdal` package or the `ptransform()` function of the `proj4` package, but unfortunately those two packages can be complicated to install on some systems, so in this tutorial we stick to the `st_transform()` function which is part of the `sf` package, which we already loaded above. The `st_transform()` function can be used to transform an `sf` object (output of `st_read()` function). This is why we are using the `world` object and not the `SpatialPolygon` object (`world_spatial`) in the following command. All we need to provide is the target projection, so in this case `projection(merged_rasters)`:

```
transformed_world = st_transform(world,projection(merged_rasters))
transformed_world_spatial = as(transformed_world, 'Spatial')
```

Let's see what the plot looks like now:

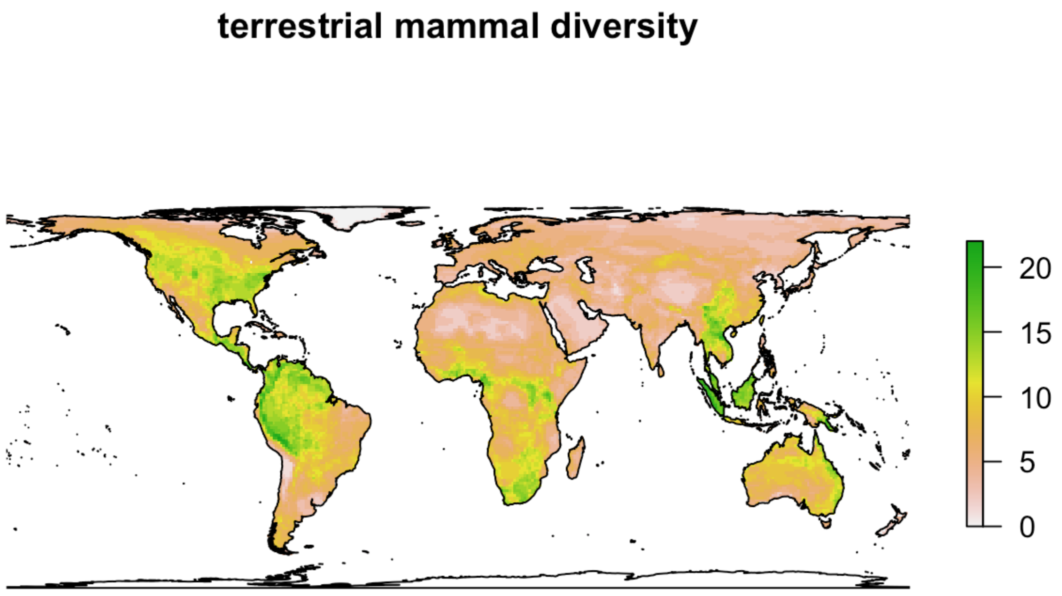
```
plot(merged_rasters,axes=F,box=FALSE)
plot(transformed_world_spatial,add=T)
```



Using mask function to make cell selection

This looks much better, but what if we only want to plot the terrestrial mammal diversity? The reason why also the ocean cells have values in the raster is because of marine mammals, such as whales and dolphins. In case we are only interested in terrestrial mammal diversity, we can use the `mask()` function, which only keeps those cells that are within a given polygon (in this case the world map):

```
land_cells = mask(merged_rasters,transformed_world_spatial)
plot(land_cells,axes=F,box=FALSE,main='terrestrial mammal diversity')
plot(transformed_world_spatial,add=T)
```



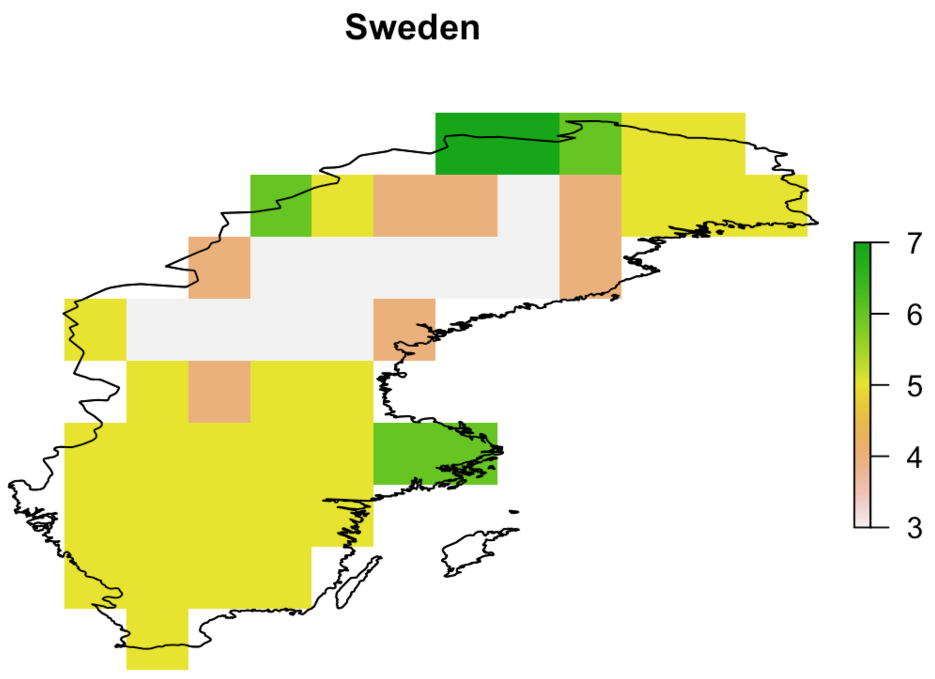
You can store the plot as a pdf file to look at it in some more detail:

```
pdf("../output_files/terrestrial_mammal_diversity.pdf")
plot(land_cells,axes=F,box=FALSE,main='terrestrial mammal diversity')
plot(transformed_world_spatial,add=T)
dev.off()
```

```
## quartz_off_screen
##                 2
```

Task: Now plot the mammal species diversity for a country of your choice. You know how to extract the polygon for a specific country and you have the raster of mammal diversity. Don't forget to transform the polygon to the right coordinate projection!

Tip: If you want to restrict the plot to only your region of interest you can do this the easiest by first plotting your (transformed) country polygon. This will make sure that the plot is centered on your region of interest. Then plot the raster on top of the polygon and then plot the polygon again, in order to make the polygon borders visible.



Transform matrix of coordinates into desired projection

What if we want to transform a matrix of coordinates that we have loaded into R into any given projection? For example let's try to plot the `species_occurrences` data from earlier on the transformed map from the previous section.

We first need to tell R what projection our coordinates are currently in. You might find this information stored in the metadata that came with your occurrence files or if you collected the data yourself you may be able to find the information in the settings of your GPS device. However, if you have no idea what the projection of your coordinates is, look at the values of your coordinates and try to make out waht reference system they might stem from. Often a good first guess is `'+proj=longlat +datum=WGS84'`.

In our case the data indeed is in `'+proj=longlat +datum=WGS84'`, so we can use the `CRS()` command from the `sp` package to turn this string into a coordinate reference system object. The `class()` command shows you what object class a given object belongs to.

```
library(sp)

crdref <- CRS('+proj=longlat +datum=WGS84')
class(crdref)
```

```
## [1] "CRS"
## attr(,"package")
## [1] "sp"
```

Now we use this information when transforming our coordinates into a `SpatialPoints` object, by setting the argument `proj4string=` to `crdref` , which we defined in the previous step. This will make sure that the informaiton about the coordinate projection is stored with our spatial data.

```
species_occurrences <- SpatialPoints(locations,proj4string=crdref)
```

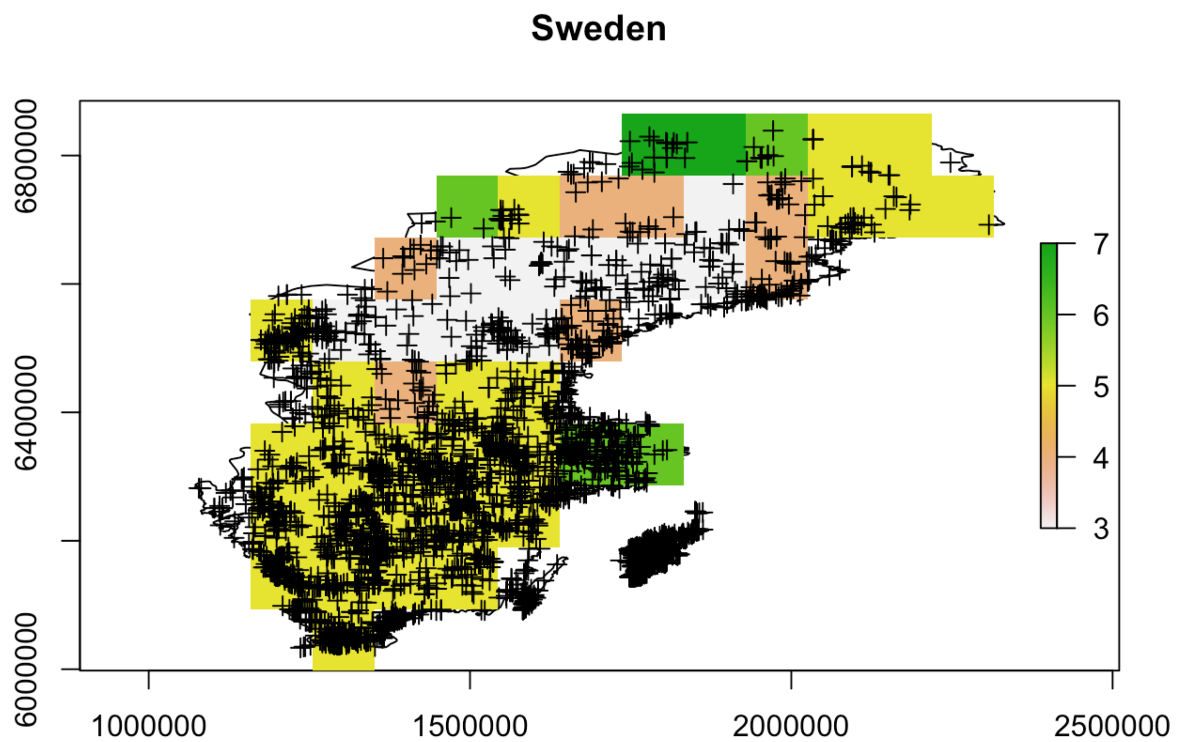
Now where your `SpatialPoints` object has CRS infomration, you can project it into any desired different projection. In this case this is the projection of our rasters from earlier (`projection(merged_rasters)`). We can again use the `st_transform()` function, just as we did earlier for the world map polygon. However before we can use that function we need to convert our `SpatialPoints` object into a `sf` object (the function `st_transform()` only works for `sf` objects). We can simply do that using the `st_as_sf()` function, as shown below:

```
target_projection = projection(merged_rasters)

species_occurrences_sf = st_as_sf(species_occurrences)
species_occurrences_transformed = st_transform(species_occurrences_sf,target_p
rojection)
```

Now let’s plot the data and see if it matches with our map.

```
plot(country_transformed_spatial,axes=T,main='Sweden')
plot(country_cells,add=T)
plot(species_occurrences_transformed,pch=3,add=T)
```



C) Plotting range of a species

In this part of the tutorial you will extract the current range of a species (data from IUCN) and plot it on top of the present natural range of the same species (where it would exists without human disturbance), in order to visualize the range contraction due to human impact. You can download range data of individual species from IUCN (https://www.iucnredlist.org/). You can search for your species of interest and then click on the blue Download button to the right. Seelct 'Range data'. You will probably be prompted to log in. You can just choose your google account to log in or make a new account at IUCN. Then you'll have to write what you want to use the data for (just write 'Needed for a course' or something short like that). You'll also have to agree to their terms and conditions and then a window will pop up that will suggest you to go to the accounts page. From there you can select your requested data under 'Saved downloads'. Click on 'download' (it may take a couple of minutes before your data is available and the button will appear) and store the data in the downlaoded github repo data folder. Then load the shape file into R.

```
library(sf)
species_data = st_read('../data/redlist_species_data_e5f54e49-d2e2-4f2e-ae85-1
afbfaa20cb1/data_0.shp')
```

```
## Reading layer `data_0' from data source `/Users/tobias/GitHub/spatial_R_cou
rse/data/redlist_species_data_e5f54e49-d2e2-4f2e-ae85-1afbfaa20cb1/data_0.shp'
using driver `ESRI Shapefile'
## Simple feature collection with 164 features and 15 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -14.89546 ymin: -34.08534 xmax: 42.54696 ymax: 16.634
32
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

Get species present natural raster

The present natural raster files are stored at `../data/Data 2/Ranges/Present_natural/` . To find the raster file of a specific species you can use a combination of the gsub command (which will replace the space with a '_' in the species name) and the paste0 command (which you can use to paste the .tif file ending to the file and to join the filename with the path to the folder):

```
species = 'Loxodonta africana'
filename = gsub(" ", "_", species)
file_path = paste0('../data/Data 2/Ranges/Present_natural/',filename,'.tif')
species_prenat_raster = raster(file_path)
```

Task: Use the commands you learned in this tutorial to plot the range of your species of choice on top of the present natural raster of the same species. The resulting plot should look something like this:

```
## Reading layer `data_0' from data source `/Users/tobias/GitHub/spatial_R_cou
rse/data/redlist_species_data_e5f54e49-d2e2-4f2e-ae85-1afbfaa20cb1/data_0.shp'
using driver `ESRI Shapefile'
## Simple feature collection with 164 features and 15 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -14.89546 ymin: -34.08534 xmax: 42.54696 ymax: 16.634
32
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

Loxodonta africana