GIS in R: Tutorial 1

Marie Auger-Méthé

1 Good R practices

If you want to be able to reproduce your analyses, it is important to keep track of your files, write down all coding steps, and preferably do version control. Keeping track of your steps is both important to make your research reproducible and easily shareable, and just because it is nice not to have to start from scratch every time you want to do an analysis.

1.1 Keep all our files in one place with R Studio projects

This is particularly important for spatial analyses because spatial data files can be challenging to handle and keeping all files in the same folder facilitate the process. Keep originals somewhere else so only changes stuff.

1.2 Make reproducible examples with R scripts

Share complete analysis by sending an R script. Put comments!

For those that are taking the class for credit, I want you to send me by e-mail for each tutorial a `R` script that has all of the code from the tutorial, including the code found in the document.

1.3 Keep track of your changes with version control

2 `Spatial` class

Vector data (i.e., Points, Lines, and Polygons) are handle in `R` using the foundation class `Spatial` associated with the package `sp`. To get a feel for how `Spatial` objects differ from other types of `R` objects, we will import a .csv table with the locations of bioprobes (this is a simplified version of the data available on the Ocean Tracking Network (OTN) website).

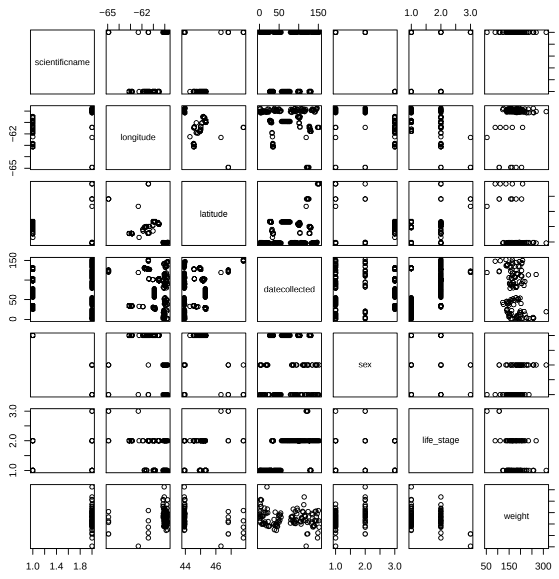
```r
# Read the .csv file with OTN Sable Island bioprobe data
bioPr <- read.csv("sableSealBioP.csv")
```

To get a sense for this object we will investigate the class, and some of the generic functions.

```r
# What's the class?
class(bioPr)

## [1] "data.frame"

# Can we summarise the information within it?
summary(bioPr)

##             scientificname   longitude         latitude
##  Gadus morhua      :609    Min.   :-64.93   Min.   :43.93
##  Halichoerus grypus:118    1st Qu.:-61.74   1st Qu.:44.56
##                            Median :-60.90   Median :45.01
##                            Mean   :-61.17   Mean   :44.89
##                            3rd Qu.:-60.51   3rd Qu.:45.18
##                            Max.   :-59.74   Max.   :47.80
##
##              datecollected sex         life_stage      weight
##  2011-06-09T00:00:00: 62   F: 81            :255   Min.   : 53.0
##  2012-11-19T00:00:00: 57   M: 37   ADULT    :469   1st Qu.:152.0
##  2012-11-20T00:00:00: 57   U:609   SUB-ADULT:  3   Median :169.0
##  2013-05-16T00:00:00: 49                           Mean   :174.7
##  2014-05-13T00:00:00: 45                           3rd Qu.:196.0
##  2013-11-16T00:00:00: 42                           Max.   :312.5
##  (Other)            :415                           NA's   :610
```

One of the generic function is the `plot` functions.

```r
plot(bioPr)
```



Since we know that the data includes the locations of bioprobes, it would be much better to be able to plot it spatially. To do this we will trasform this `data.frame` into a `Spatial` object. In particular, since our data is point data, we will transform the data.frame into a `SpatialPointsDataFrame`. The simplest way to do so is by assigning values to `coords` slot, which is the component that contains the coordinate values.

```
# Load package
library(sp)
# Create new data.frame with the exact same value as bioPr
bioPrSP <- bioPr
# Transform this new object into a SpatialPointsDataFrame using the columns
# that contain the latitude and longitude values
colnames(bioPrSP)


## [1] "scientificname" "longitude"      "latitude"       "datecollected"
## [5] "sex"            "life_stage"     "weight"


coordinates(bioPrSP) <- ~longitude + latitude
```

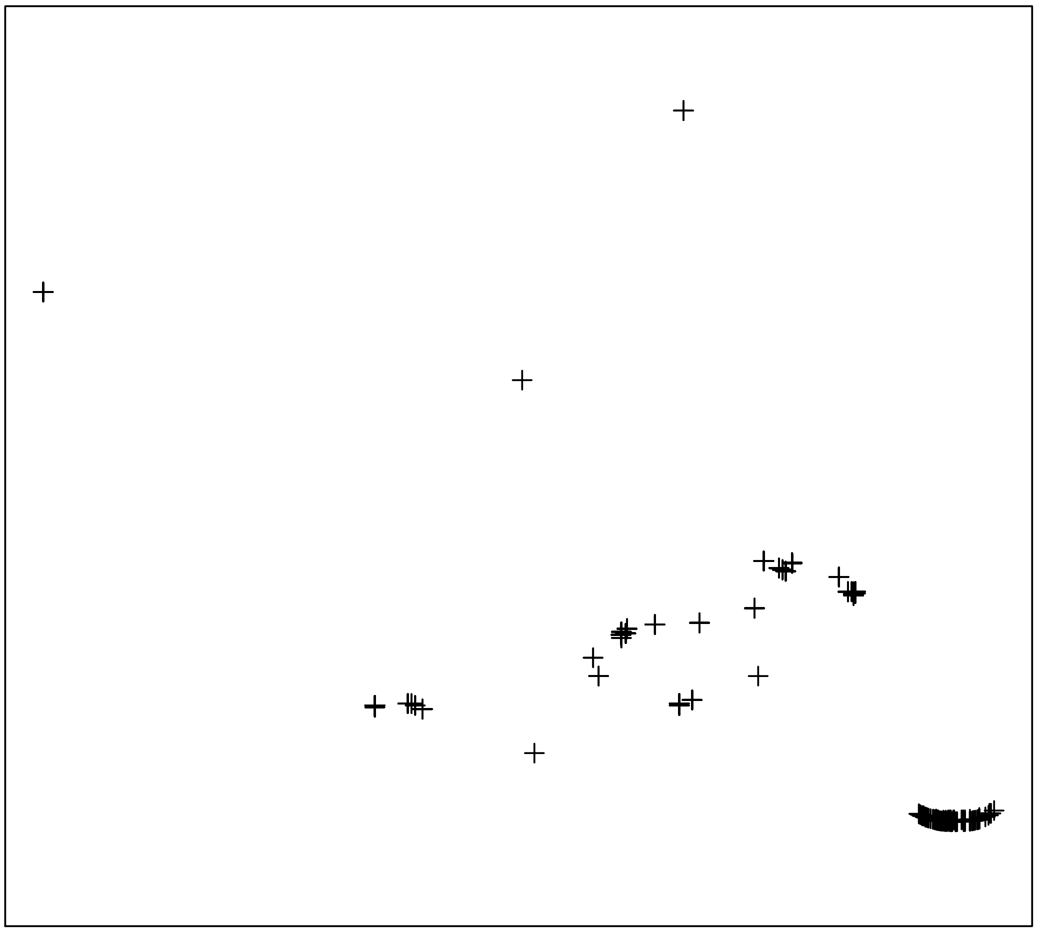
Now let's compare the results from the same generic functions.

```
class(bioPrSP)


## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"

summary(bioPrSP)

## Object of class SpatialPointsDataFrame
## Coordinates:
##                 min       max
## longitude -64.92774 -59.74377
## latitude   43.92560  47.80300
## Is projected: NA
## proj4string : [NA]
## Number of points: 727
## Data attributes:
##             scientificname             datecollected sex
##  Gadus morhua      :609    2011-06-09T00:00:00: 62   F: 81
##  Halichoerus grypus:118    2012-11-19T00:00:00: 57   M: 37
##                            2012-11-20T00:00:00: 57   U:609
##                            2013-05-16T00:00:00: 49
##                            2014-05-13T00:00:00: 45
##                            2013-11-16T00:00:00: 42
##                            (Other)            :415
##      life_stage      weight
##           :255   Min.   : 53.0
##  ADULT    :469   1st Qu.:152.0
##  SUB-ADULT:  3   Median :169.0
##                  Mean   :174.7
##                  3rd Qu.:196.0
##                  Max.   :312.5
##                  NA's   :610
```

We see now the bounding box (`bbox`) with the min and max values of the coordinates. We know that the data is not projected and that the `proj4string` slot has a missing value, more on this below. We also see that we have 727 points, which correspond to the 727 rows of the original `data.frame`. Finally we see a summary of the attributes of the points, which is the same summary as the one for the orginal data frame, when you exclude the columns with the coordinates.

But more interestingly, let's look at the `plot` function.

```
plot(bioPrSP)
# With an added box around the figure
box()
```



Here instead of plotting the data values from all the columns the plot shows the location of the row in space.

3 Projection

While this is great, this plot is not particularly informative. In part, because we don't have any reference points that tells us where the locations are. In Canada? In New Zealand?

To be able to put reference points we need to give to make sure the `proj4string` slot of the `Spatial` object is set to the correct projection. When we ran `summary(bioPrSP)` above, we could see both the Is projected is NA and the proj4string is NA, indicating that the data doesn't have a projection. This is not surprising since we have not specified the projection. We can confirm this with the `proj4string`.

```
proj4string(bioPrSP)

## [1] NA

is.projected(bioPrSP)

## [1] NA
```

I'm assuming that the latitude and longitude data from OTN is the conventional GPS data using WGS84 datum. To set it, you can use again the

```
proj4string(bioPrSP) <- CRS("+proj=longlat +datum=WGS84")
# Now bioPrSP has as datum
proj4string(bioPrSP)

## [1] "+proj=longlat +datum=WGS84"

# But it's still not projected
is.projected(bioPrSP)

## [1] FALSE
```

That's because lat lon is not a projection... bla bla bla, and we can set another value to proj4strig, need to projtranform... bla bla bla

Error when you have locations that are outside coordinates system, this can happen for example if there is a typo or for data that come from highly error-prone system (e.g. light curve location system). bla bla

```
# We will put a non-sensical coordinate in the data frame
bioPr$longitude[1] <- -300  # -300 degree longitude makese no sense
head(bioPr)
```

```
##   scientificname longitude latitude       datecollected sex life_stage
## 1   Gadus morhua  -300.000   44.953 2014-05-13T00:00:00   U
## 2   Gadus morhua   -61.752   44.953 2014-05-13T00:00:00   U
## 3   Gadus morhua   -61.778   44.945 2014-05-15T00:00:00   U
## 4   Gadus morhua   -61.745   44.978 2014-05-14T00:00:00   U
## 5   Gadus morhua   -61.745   44.978 2014-05-14T00:00:00   U
## 6   Gadus morhua   -61.745   44.978 2014-05-14T00:00:00   U
##   weight
## 1     NA
## 2     NA
## 3     NA
## 4     NA
## 5     NA
## 6     NA
```

```
# Now let's try to create the spatial object again, using the same steps
bioPrSP2 <- bioPr
coordinates(bioPrSP2) <- ~longitude + latitude
proj4string(bioPrSP2) <- CRS("+proj=longlat +datum=WGS84")
```

```
## Error in `proj4string<-`(`*tmp*`, value = <S4 object of class structure("CRS", package =
"sp")>):  Geographical CRS given to non-conformant data:  -300
```

We get an error because the longitude value -300 doesn't exist in the coordinate system we chose. As a result the object remains without a coordinate system.

```
proj4string(bioPrSP2)
```

```
## [1] NA
```

If you remove the row with the non-conformat data, you'll be able to assign the coordinate system.

```
# Get the index for the non-conformant coordinates
ncLonLat <- which(coordinates(bioPrSP2) == -300, arr.ind=TRUE)
# We know it should be in row 1, col 1, since we've put it there
ncLonLat
```

```
##      row col
## [1,]   1   1
```

```
# Remove that row
```

```r
bioPrSP2 <- bioPrSP2[-ncLonLat[,1],]
# Now assign the coordinate system
proj4string(bioPrSP2) <- CRS("+proj=longlat +datum=WGS84")
```

4 Attributes: subsetting

What if we wanted to create a object that only include a subset of the location. In this example we would like to only have the Atlantic cod (*Gadus morhua*) locations.

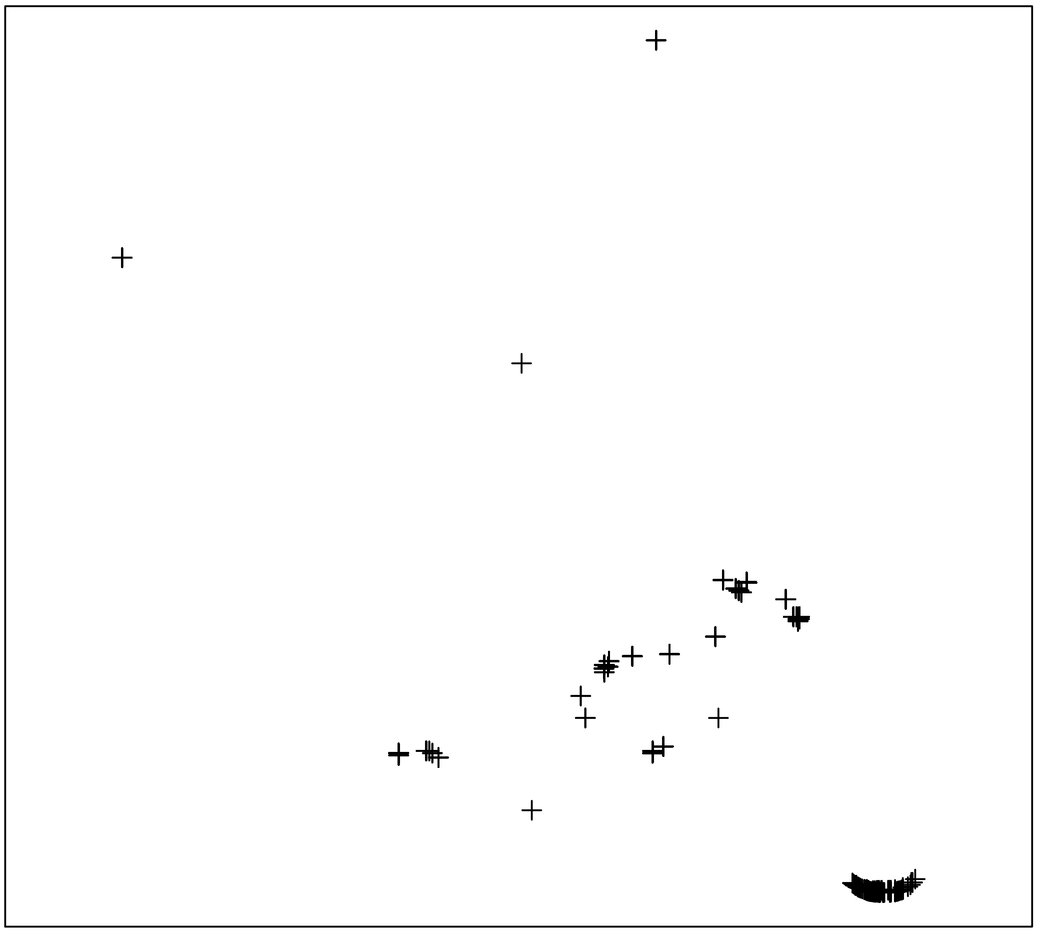
```r
# Only take the points that have 'Gadus morhua' in the column scientificname
codPrSP <- bioPrSP[bioPrSP$scientificname == "Gadus morhua", ]
# We now have 0 Halichoerus grypus, but we can see that the subset will keep
# the coordinate system, but will update the bounding box
summary(codPrSP)

## Object of class SpatialPointsDataFrame
## Coordinates:
##              min      max
## longitude -63.12 -60.5000
## latitude   44.30  45.3475
## Is projected: FALSE
## proj4string : [+proj=longlat +datum=WGS84]
## Number of points: 609
## Data attributes:
##             scientificname             datecollected sex
##  Gadus morhua      :609    2011-06-09T00:00:00: 62   F:  0
##  Halichoerus grypus:  0    2012-11-19T00:00:00: 57   M:  0
##                            2012-11-20T00:00:00: 57   U:609
##                            2013-05-16T00:00:00: 49
##                            2014-05-13T00:00:00: 45
##                            2013-11-16T00:00:00: 42
##                            (Other)            :297
##      life_stage      weight
##           :200   Min.   : NA
##  ADULT    :409   1st Qu.: NA
##  SUB-ADULT:  0   Median : NA
##                  Mean   :NaN
##                  3rd Qu.: NA
##                  Max.   : NA
##                  NA's   :609
```

```
# For example compare
bbox(bioPrSP)

##                  min       max
## longitude -64.92774 -59.74377
## latitude   43.92560  47.80300

bbox(codPrSP)

##               min      max
## longitude -63.12 -60.5000
## latitude   44.30  45.3475
```

The object class we have been using so far, `SpatialPointsDataFrame`, is actually a more complex level class for point data. I've presented first, because it's the class that I find the more useful because most of my data is poits with some kind of attributes. However, there may be cases where the data you have is only coordinates, with no further attributes associated with them. In this case you would use a `SpatialPoints` class. For example, let's say we only have the locations of the bioprobes, with no information on the species, sex, age, etc..

```
# We are making a bit of an artifical example here by first getting the
# coordinates of the bioPrSP
bioPrCoord <- coordinates(bioPrSP)
# Now let assume that this new object is the only information we have
summary(bioPrCoord)

##    longitude         latitude
##  Min.   :-64.93   Min.   :43.93
##  1st Qu.:-61.74   1st Qu.:44.56
##  Median :-60.90   Median :45.01
##  Mean   :-61.17   Mean   :44.89
##  3rd Qu.:-60.51   3rd Qu.:45.18
##  Max.   :-59.74   Max.   :47.80

# We can create a SpatialPoints object as follow
bioPrCoordSP <- SpatialPoints(bioPrCoord, proj4string = CRS("+proj=longlat +ellps=WGS84"))
# This now a SpatialPoints object, without any attribute
summary(bioPrCoordSP)

## Object of class SpatialPoints
## Coordinates:
```

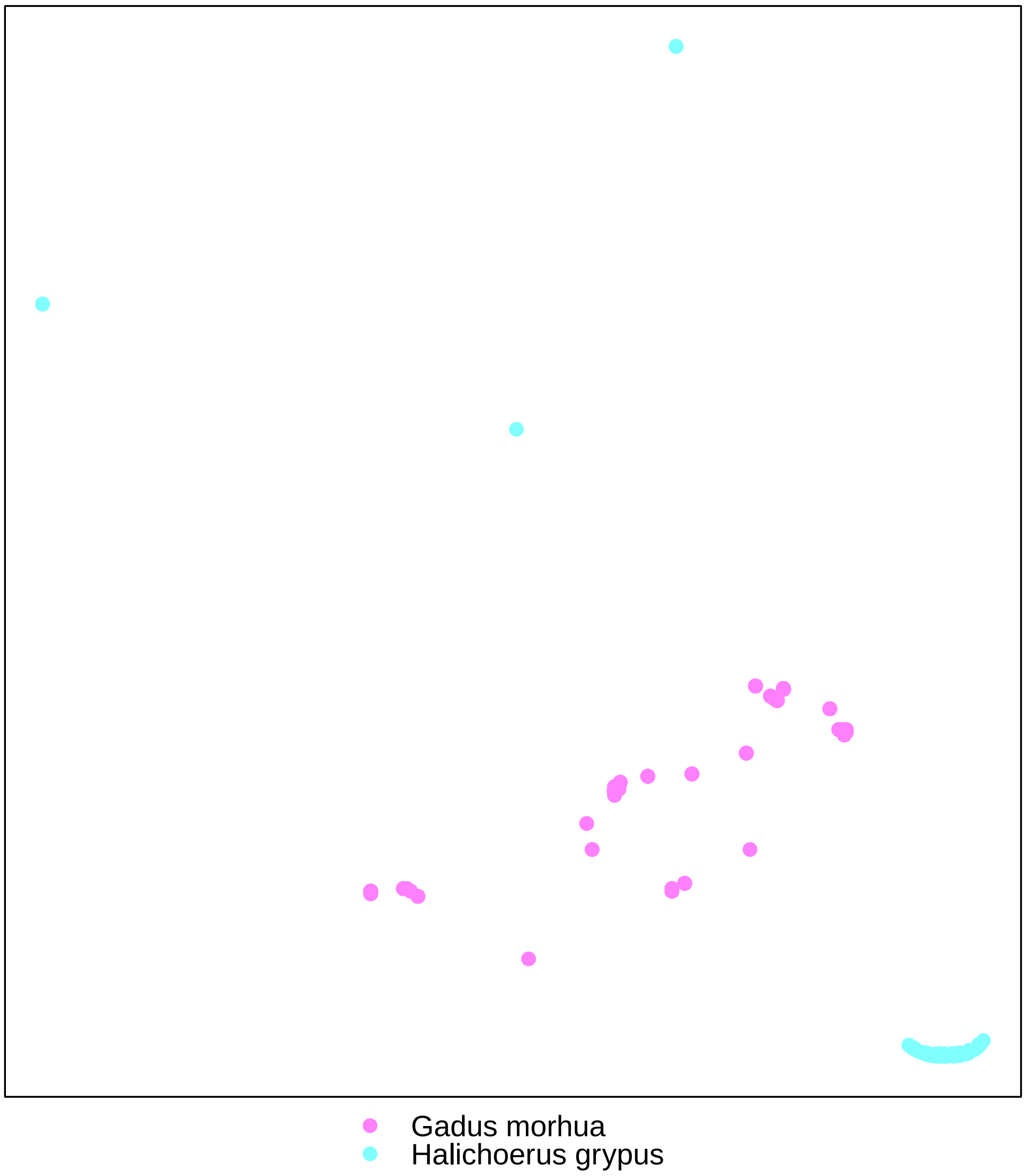
```
##                  min       max
## longitude -64.92774 -59.74377
## latitude   43.92560  47.80300
## Is projected: FALSE
## proj4string : [+proj=longlat +ellps=WGS84]
## Number of points: 727


# That can be plotted just like before
plot(bioPrCoordSP)
box()
```



It can happen that you may have the coordinated information coming from a different file than the file in which you have information. In which case you can pyt the two together using the `match.ID` option of `SpatialPointsDataFrame`... bla bla bla

The great thing about attributes is that they can be used to create symbols in plots.

```
spplot(bioPrSP,zcol="scientificname")
```



But more on this on the next class!

5 Exercise

Exercise 1

Create a spatial object with the data found in nbCod.csv, which is a slightly altered version from the Animal file available on the Shippagan, NB: Code tagging project from OTN, see project website.