Introduction to BIRDS

Alejandro Ruete and Debora Arlt

2019-07-29

Installing `BIRDS`

The package `BIRDS` is currently available on GitHub and you can freely download the source code here BIRDS on GitHub

The easiest option is to install the package directly from GitHub using the package `devtools`. If you do not already have `devtools` installed then need to install it.

```
install.packages('devtools')
library(devtools)
install_github('Greensway/BIRDS')
```

If you receive an error we would love to receive your bug report here

Basic example and data requirements

This package works with `data.frame` tables containing raw species observations as rows (i.e. primary biodiversity data, PBD), with minimal information required for each observation in the following columns:

1. x and y spatial coordinates in two columns (epsg:4326 assumed if not otherwise specified)
2. species idenfications
3. date of the observation (either in one formated column or three columns y-m-d)

The function `organiseBirds()` converts 'data.frame' into a `SpatialPointsDataFrame` adding to each observation a unique identifier for the assumed visit it belongs to, given some input parameters. The function `organizeBirds()` will interpret the DarwinCore standard for column names as default, but other names can be specified too. For more details on how a visit is defined and options on each function please refer to each function's help page and to the technical vignette.

Then, the function `summariseBirds()` will overlay the data with a given spatial grid and create a set of objects that summarize the data spatially, temporally, and spatiotemporally, and also provide other intermediate results useful for later analyses. Finally, the function `exportBirds()` helps the user to obtain the data ready to be plotted with widely known functions.

We use as an example a dataset consisting on 10k species observations of bumblebees (i.e. *Bombus* spp.) in Götaland (the southern part of Sweden) during 2000-2018. The dataset `bombusObs` is part of this package and its metadata can be found in the dataset help page `?bombusObs`.

You can easily create a grid over a sample area (i.e. `gotaland`), organize and summarize the data, and export the variables you want to plot:

```
library(BIRDS)
grid <- makeGrid(gotaland, gridSize = 10)
# The grid can be easily created in different ways.
```

```
PBD<-bombusObs
# alternatively, you could load a previously downloaded .CSV file
# PBD <- read.csv(file="path/to/your/file.csv)
OB <- organizeBirds(PBD, sppCol = "scientificName", simplifySppName = TRUE)
SB <- summariseBirds(OB, grid=grid)
#> Registered S3 method overwritten by 'xts':
#>   method     from
#>   as.zoo.xts zoo

# Number of observations
EBnObs <- exportBirds(SB, dimension = "temporal", timeRes = "yearly",
                      variable = "nObs", method = "sum")
# Number of visits
EBnVis <- exportBirds(SB, dimension = "temporal", timeRes = "yearly",
                      variable = "nVis", method = "sum")
# Average species list length (SLL) per year (i.e. the median over all visits
# per year and cell, and then the mean over cell values)
EBavgSll <- colMeans(SB$spatioTemporal[,,"Yearly","avgSll"], na.rm = TRUE)
# The ratio of number of observations over number of visits
relObs<-EBnObs/EBnVis
```

Then, with the functions you already know, plot this into a time series.

```
par(mar=c(4,4,1,6), las=1)
plot(names(EBnObs), EBnObs, type = "l", lwd = 3, xlab = "Year", ylab = "Number",
     ylim=c(0, max(EBnObs)), xaxp=c(2000, 2018, 18))
lines(names(EBnObs), EBnVis, lwd=3, lty=2)
lines(names(EBnObs), relObs*max(EBnObs)/max(relObs), lwd=3, lty=1, col="#78D2EB")
lines(names(EBnObs), EBavgSll*max(EBnObs)/max(EBavgSll), lwd=3, lty=1, col="#FFB3B5")
axis(4, at = seq(0, max(EBnObs), length.out = 5),
     labels = round(seq(0,max(relObs), length.out = 5), 1),
     lwd = 2, col = "#78D2EB", col.ticks = "#78D2EB")
axis(4, at = seq(0, max(EBnObs), length.out = 5) ,
     labels = round(seq(0,max(EBavgSll), length.out = 5), 1),
     lwd = 2, col = "#FFB3B5", col.ticks = "#FFB3B5", line = 3)
legend("topleft", legend=c("observations","visits"),
       lty = c(1,2), lwd = 3, bty = "n")
legend("bottomright", legend=c("n.observations / n.visits", "Mean SLL per cell"),
       lty = 1, lwd = 3, col = c("#78D2EB", "#FFB3B5"), bty = "n")
```

Altough we downloaded the data in April 2019, we know that the latest reports for 2018 were not yet uploaded and we see that in the figure.

Then, we can summarize spatial data in some maps, to answer e.g. where do we have visits at all?, or how often are visits performed in July vs December?

```
library(sp)
wNonEmpty<-unname( which( unlist(lapply(SB$overlaid, nrow)) != 0) )
EB<-exportBirds(SB, "Spatial", "Month", "nYears", "sum")
# because the dimension is "spatial", the result is a 'SpatialPolygonDataFrame'

palBW <- leaflet::colorNumeric(c("white", "navyblue"),
                               c(0, max(EB@data, na.rm = TRUE)),
```

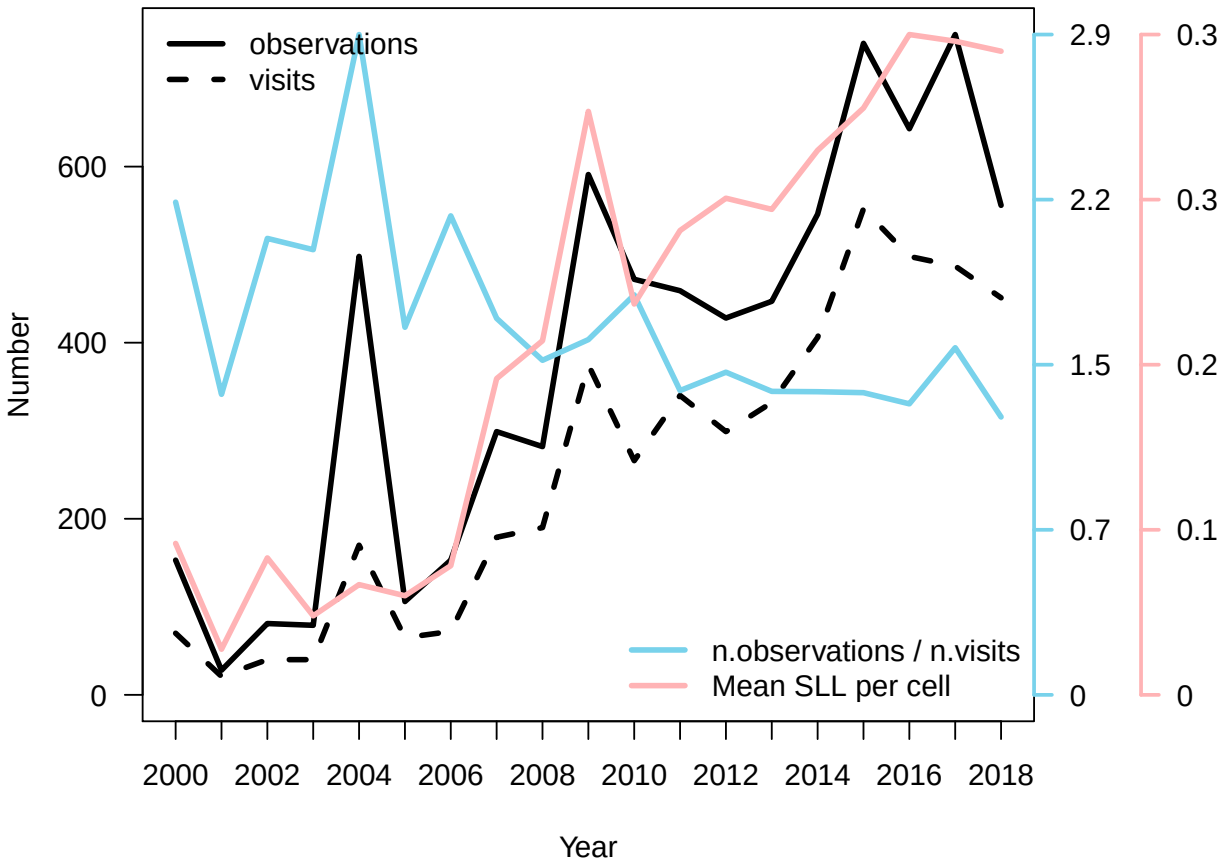
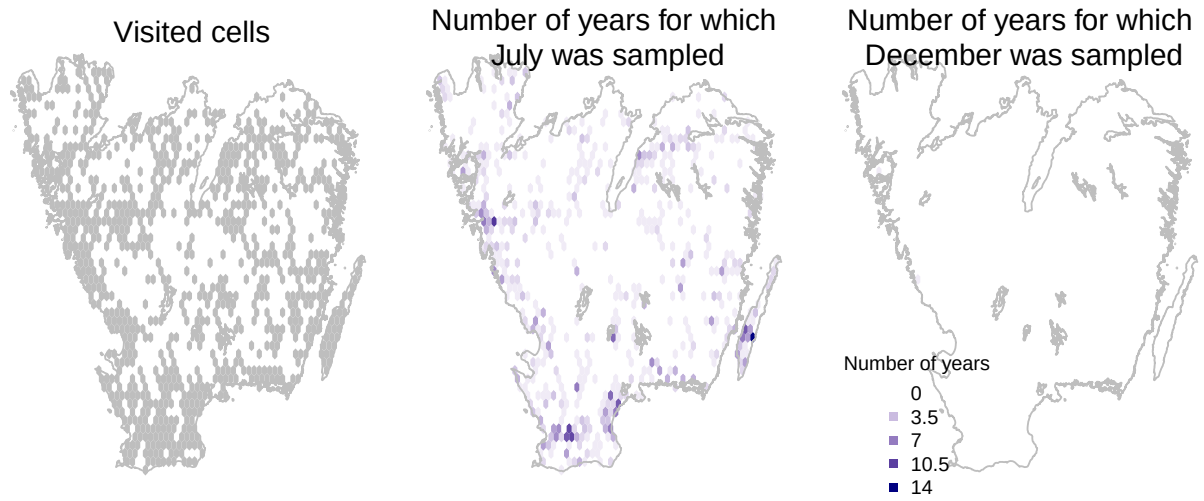


Figure 1: Time series for *Bombus* spp. dataset.

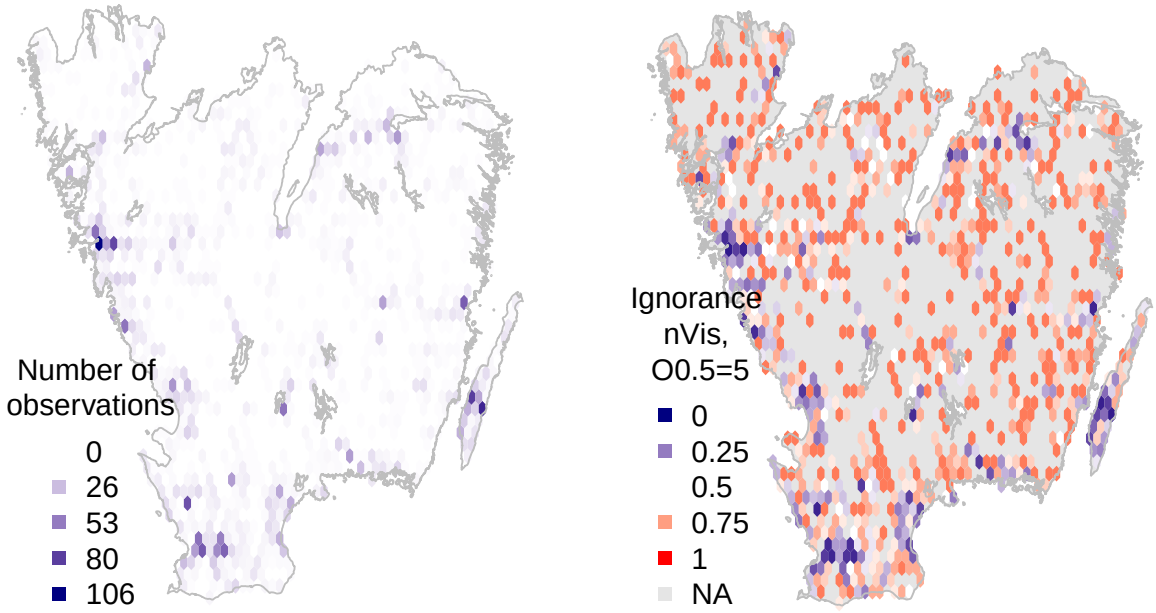
```
                                  na.color = "transparent")
par(mfrow=c(1,3), mar=c(1,1,1,1))
plot(SB$spatial[wNonEmpty,], col="grey", border = NA)
plot(gotaland, col=NA, border = "grey", lwd=1, add=TRUE)
mtext("Visited cells", 3, line=-1)

plot(EB, col=palBW(EB@data$Jul), border = NA)
plot(gotaland, col=NA, border = "grey", lwd=1, add=TRUE)
mtext("Number of years for which \nJuly was sampled", 3, line=-2)
plot(EB, col=palBW(EB@data$Dec), border = NA)
plot(gotaland, col=NA, border = "grey", lwd=1, add=TRUE)
mtext("Number of years for which \nDecember was sampled", 3, line=-2)
legend("bottomleft", legend=seq(0, max(EB@data, na.rm = TRUE), length.out = 5),
      col = palBW(seq(0, max(EB@data, na.rm = TRUE), length.out = 5)),
      title = "Number of years", pch = 15, bty="n")
```



Also, we could map the ignorance scores based on the number of visits using the function `exposeIgnorance()`

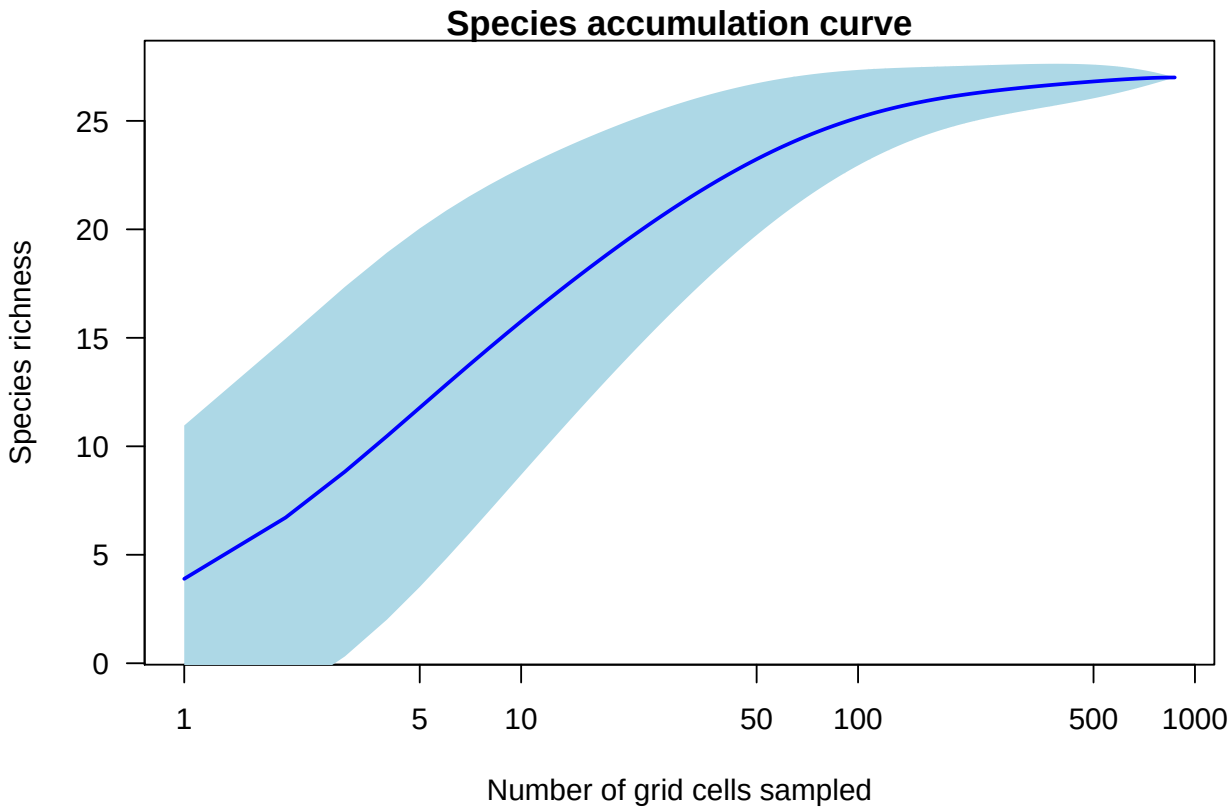
```
par(mfrow=c(1,2), mar=c(1,1,1,1))
palBW <- leaflet::colorNumeric(c("white", "navyblue"),
                                  c(0, max(SB$spatial@data$nVis, na.rm = TRUE)),
                                  na.color = "transparent")
seqNVis<-round(seq(0, max(SB$spatial@data$nVis, na.rm = TRUE), length.out = 5))
plot(SB$spatial, col=palBW(SB$spatial@data$nVis), border = NA)
plot(gotaland, col=NA, border = "grey", lwd=1, add=TRUE)
legend("bottomleft", legend=seqNVis, col = palBW(seqNVis),
      title = "Number of \nobservations", pch = 15, bty="n")

ign<-exposeIgnorance(SB$spatial@data$nVis, h = 5)
palBWR <- leaflet::colorNumeric(c("navyblue", "white","red"), c(0, 1),
                                  na.color = "transparent")
plot(gotaland, col="grey90", border = "grey90", lwd=1)
plot(SB$spatial, col=palBWR(ign), border = NA, add=TRUE)
plot(gotaland, col=NA, border = "grey", lwd=1, add=TRUE)
legend("bottomleft", legend=c(seq(0, 1, length.out = 5), "NA"),
      col = c(palBWR(seq(0, 1, length.out = 5)), "grey90"),
```

```
      title = "Ignorance \nnVis, \nO0.5=5", pch = 15, bty="n")
```



Or we could connect to other packages like `vegan` to continue with community analyses. In this case, we create a community matrix using the visits as sampling units, summarising for each grid cell the number of visits a species was observed (i.e. ignoring replicates within visits).

```
## Community analysis
CM <- communityMatrix(SB, sampleUnit="visit")
library(vegan)
sp1 <- specaccum(CM, method = "exact")
par(mar=c(4,4,1,1), xlog=TRUE, las=1)
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
     ci.col="lightblue", log="x",
     xlab="Number of grid cells sampled",
     ylab="Species richness",
     main="Species accumulation curve")
#> Warning in polygon(c(xaxvar, rev(xaxvar)), c(x$richness - ci * x$sd,
#> rev(x$richness + : "log" is not a graphical parameter
#> Warning in plot.xy(xy.coords(x, y), type = type, ...): "log" is not a
#> graphical parameter
```


Species accumulation curve
Species richness
0
5
10
15
20
25
1
5
10
50
100
500
1000
Number of grid cells sampled