Using R tools for analysis of primary biodiversity data provided by SBDI

Debora Arlt and Alejandro Ruete for the Swedish Biodiversity Data Infrastructure

2024-05-06

Contents

Introduction

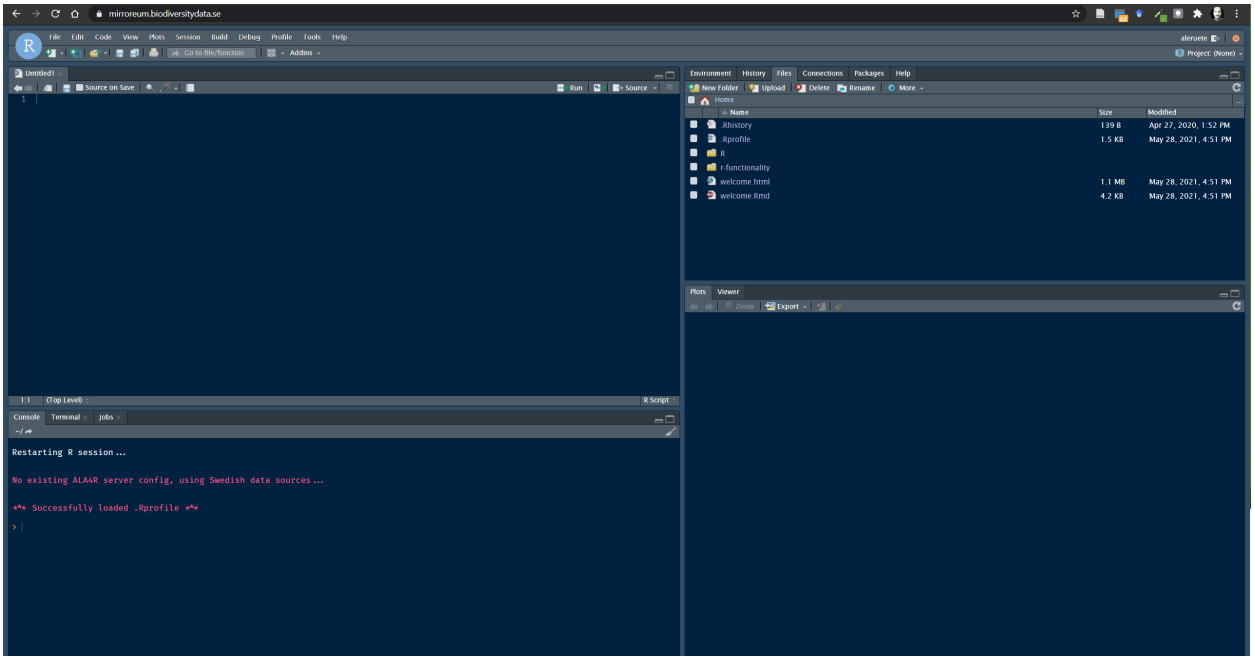
Biodiversity resources are increasingly international. The SBDI has made an effort to canalise biodiversity data and resources to help the research community access and analyse Swedish primary biodiversity data. Each research question draws its own challenges which are unique in themselves. Our aim here is to provide a few examples that prompt questions that may be asked at different stages of the process. The validity and appropriateness of a particular method depends on the individual researcher(s). For a comprehensive workflow on how to treat and analyse primary biodiversity data please refer to our tutorial on biodiversity analysis tools where we go through the complete workflow Data –> Cleaning –> Fitness evaluation –> Analysis

R and Mirroreum

The present tutorial is focused on the statistical programming language R. R is a free software environment for statistical computing and graphics that is widely used within the scientific community and where the complete analysis workflow can be documented in a fully reproducible way.

At SBDI we provide access for researchers and students to Mirroreum – an online web-based environment for Reproducible Open Research in the area of biodiversity analysis. Mirroreum is based on a Free and Open Source stack of software. Logging in, you immediately get access to a web-based version of R Studio with a large number of pre-installed packages such as all the packages offered from ROpenSci and more.

Compared to running R Studio on your own machine, Mirroreum offers more computational resources and a standardized environment where you can rely on all the relevant packages being installed and the configuration parameters being set appropriately. To know more about Mirroreum or to request an account please visit the SBDI documentation site



sbdi4r2 - a new R to search an access data

The sbdi4r2 package enables the R community to directly access data and resources hosted by SBDI. The goal is to enable observations of species to be queried and output in a range of standard formats. It includes some filter functions that allow you to filter prior to download. It also includes some simple summary functions, and some function for some simple data exploration. The examples included in this tutorial also show you how you can continue exploring and analyzing using other R package.

Please refer to the package documentation for details on how to install it. Once installed the sbdi4r2 package must be loaded for each new R session:

```
library(sbdi4r2)
```

Various aspects of the sbdi4r2 package can be customized.

E-mail address Each download request to SBDI servers is also accompanied by an "e-mail address" string that identifies the user making the request. You will need to provide an email address registered with the SBDI. You can create an account here. Once an email is registered with the SBDI, it should be stored in the config:

```
sbdi_config(email = "your.registered@emailaddress.com")
```

Else you can provide this e-mail address as a parameter directly to each call of the function occurrences().

Setting the download reason SBDI requires that you provide a reason when downloading occurrence data (via the sbdi4r2 `atlas_occurrences()` function). You can provide this as a parameter directly to each call of `atlas_occurrences()`, or you can set it once per session using:

```
sbdi_config(download_reason_id = "your_reason_id")
```

(See `sbdi_reasons()` for valid download reasons, e.g. * 3 for "education", * 7 for "ecological research", * 8 for "systematic research/taxonomy", * 10 for "testing")

Privacy ***NO*** other personal identification information is sent. You can see all configuration settings, including the the user-agent string that is being used, with the command:

```
sbdi_config()
```

Other packages needed

Some additional packages are needed for these examples. Install them if necessary with the following script:

```
to_install <- c("colorRamps", "cowplot","dplyr",
                "ggplot2", "leaflet", "maps", "mapdata",
                "remotes", "sf", "tidyr", "xts")
to_install <- to_install[!sapply(to_install,
                                 requireNamespace,
                                 quietly = TRUE)]
if (length(to_install) > 0)
    install.packages(to_install,
                     repos = "http://cran.us.r-project.org")

remotes::install_github("Greensway/BIRDS")
```

Your collaboration is appreciated

Open Source also means that you can contribute. You don't need to know how to program but every input is appreciated. Did you find something that is not working? Have suggestions for examples or text? you can always

1. Reach to us via the support center
2. Submit and issue to the GitHub code repository see how
3. Or contribute with your code or documents modifications by "forking" the code and submitting a "pull request"

The repositories you can contribute to are:

- Mirroreum https://github.com/mskyttner/mirroreum
- sbdi4r2 https://github.com/biodiversitydata-se/sbdi4r2
- the general analysis workflows https://github.com/biodiversitydata-se/biodiversity-analysis-tools
- this R-tools tutorial https://github.com/biodiversitydata-se/r-tools-tutorial

1 Example with fish data from SERS

In this example we are interested in exploring data from a specific data resource – the Swedish Electrofishing Registry - SERS (Department of Aquatic Resources, SLU Aqua). This database has 2.8 M observations starting in the 1950's.

```
library(sbdi4r2)
library(sf)
library(dplyr)
library(lubridate)
```

SBDI is a collection of many biodiversity databases. We start by searching for the data resource we are interested in by using the function `pick_filter()`. This is an interactive query guiding you through the many resources available to filtering your query (data resources, spatial layers, and curated species lists).

```
fq_str <- pick_filter("resource")
# follow the instructions
```

Follow the instructions. Your choices here would have been "in3" :arrow_right: "dr10" (data resource 10 = SERS). Your variable `fq_str` will now contain a string "data_resource_uid:dr10".

Note: the function pick_filter() is temporarily disabled until it could be adapted to the new galah framework.

But we are not interested in the complete database, we only want to look at the data from the last 10 years. For this we add another filter string. Both filter strings (for data resource and for time period) will be treated as AND factors.

Using the function `sbdi_call()` we can now query for the observations fulfilling our filter.

```
xf <- sbdi_call() |>
  filter(dataResourceUid == "dr10",
         year >= 2014, year <= 2024) |>
  atlas_occurrences()
```

```
## ---
```

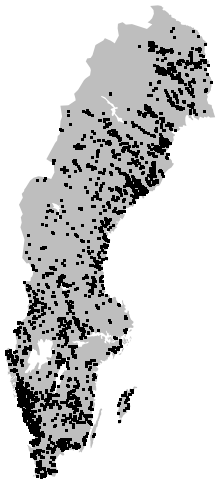
```
xf |>
  pull(dataResourceName) |>
  table() |>
  as.data.frame() |>
  rename("Source" = Var1)
```

```
##                                                                      Source
## 1 SLU Aqua  Institute of Freshwater Research Swedish Electrofishing Registry - SERS
##    Freq
## 1 26907
```

1.1 Plotting data on a map

There are many other ways of producing spatial plots in R, for example you can quickly plot all the observations with `plot()`:

```
data("swe_wgs84", package = "sbdi4r2", envir = environment())

plot(swe_wgs84[["Border"]]$geometry, col = "grey", border = NA)
  points(xf$decimalLongitude, xf$decimalLatitude, pch = ".", col = "black")
```



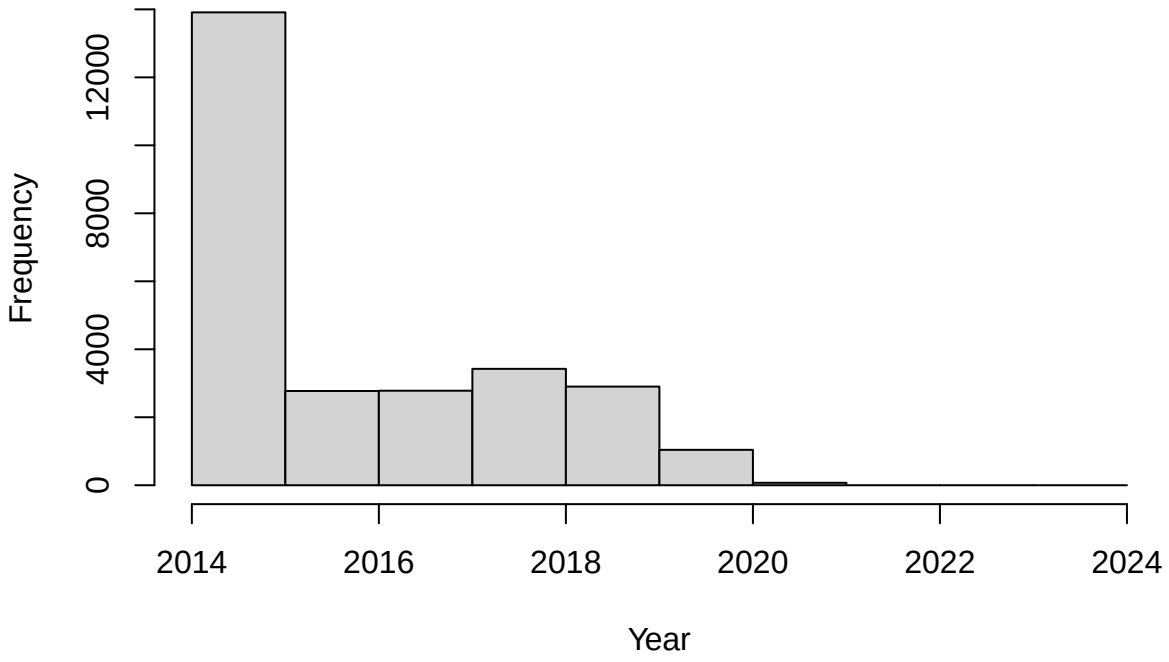
1.2 Temporal summary

A quick summary over the years reveals a drop in number of records over time.

```
xf$year <- year(xf$eventDate)
table(xf$year)
```

```
##
##  2014  2015  2016  2017  2018  2019  2020  2021
## 10299  3613  2773  2781  3424  2903  1042    72
```

```
hist(xf$year,
     breaks = seq(2014, 2024),
     xlab = "Year",
     main = "")
```



1.3 Species summary

In the same way we can summarise the number of observations for each species.

```
sppTab <- table(xf$scientificName)
sppDF <- as.data.frame(sppTab)
colnames(sppDF)[1] <- "species"
head(sppDF)
```

```
##                species Freq
## 1       Abramis brama    3
## 2   Alburnus alburnus  103
## 3   Anguilla anguilla  285
## 4           Astacidae   23
## 5     Astacus astacus   73
## 6 Barbatula barbatula   45
```

Perhaps, you want to send this table as a .CSV file to a colleague. Save the table:

```
write.csv(sppDF, "data/SERS_species_summary.csv")
# NOTE: again this will be saved on your working directory
```

1.4 Spatial biodiversity analysis

Let's now ask: How does the species richness vary across Sweden?

For this we want to summarise occurrences species-wise over a defined grid instead of plotting every observation point. First we need to overlay the observations with a grid. Here we are using the standard Swedish

grids with grid square size of 50, 25, 10 or 5 km provided as data in the sbdi4r2 package (with Coordinate Reference System = WGS84, EPSG:4326).

```
library(sf) # the function coordinates() and proj4string() are in sp
# load some shapes over Sweden's political borders
data("swe_wgs84", package = "sbdi4r2", envir = environment())
# a standard 50 km grid
data("Sweden_Grid_50km_Wgs84", package = "sbdi4r2", envir = environment())

grid <- Sweden_Grid_50km_Wgs84

# make the observations spatial
# NOTE: make sure there are no NAs in the columns defining the coordinates
# xf$data[!is.na(xf$data$longitude) | !is.na(xf$data$latitude),]

obs <- st_as_sf(as.data.frame(xf),
                coords = c("decimalLongitude","decimalLatitude"),
                crs = st_crs(4326))

# overlay the occurrence data with the grid
ObsInGridListID <- st_intersects(grid, obs)
ObsInGridList <- lapply(ObsInGridListID, function(x) st_drop_geometry(obs[x,]))
wNonEmpty <- unname( which( unlist(lapply(ObsInGridList, nrow)) != 0) )
```

The result `ObsInGridList` is a `list` object with a subset of the data for each grid cell. Now summarise occurrences within grid cells:

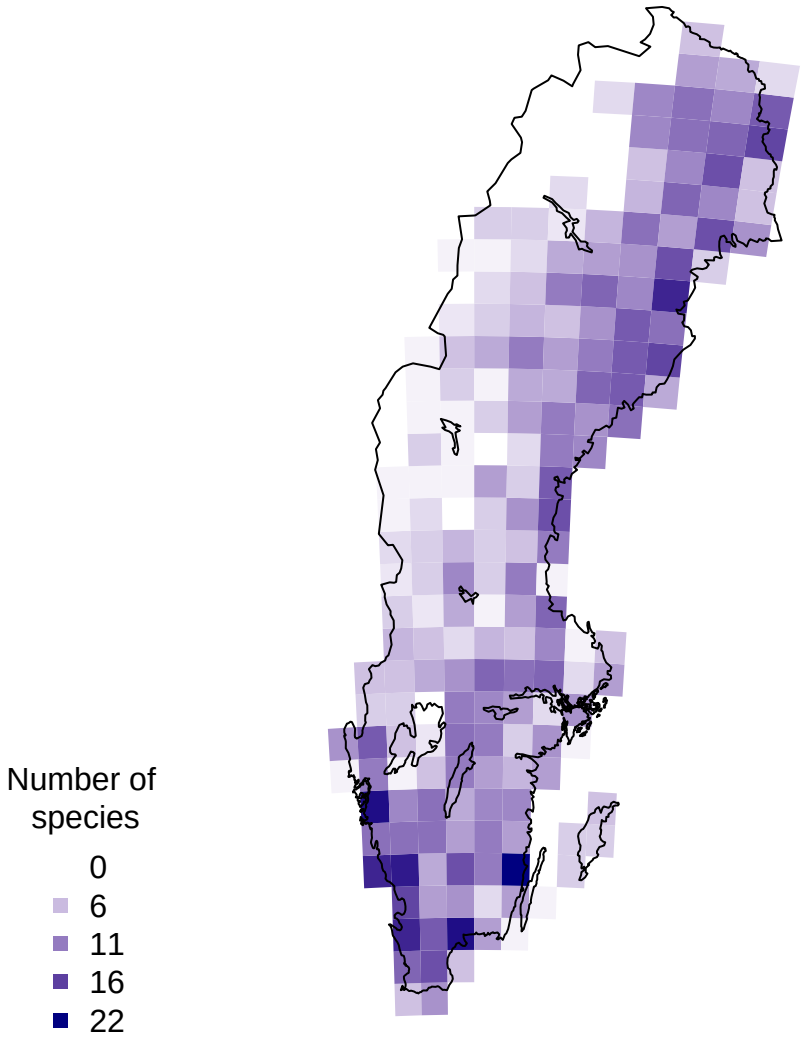
```
# check n the total number of observations
sum(unlist(lapply(ObsInGridList, nrow)))
```

```
## [1] 26907
```

```
# apply a summary over the grid cells
nCells <- length(ObsInGridList)
res <- data.frame("nObs" = as.numeric(rep(NA,nCells)),
                  "nYears" = as.numeric(rep(NA,nCells)),
                  "nSpp" = as.numeric(rep(NA,nCells)),
                  row.names = row.names(grid),
                  stringsAsFactors = FALSE)

cols2use <- c("scientificName", "year")
dataRes <- lapply(ObsInGridList[wNonEmpty],
                  function(x){
                    x <- x[,cols2use]
                    colnames(x) <- c("scientificName", "year")
                    return(c("nObs" = length(x[,"scientificName"]),
                             "nYears" = length(unique(x[,"year"])),
                             "nSpp" = length(unique(x[,"scientificName"]))
                             )
                           )
                    }
                  )
dataRes <- as.data.frame(dplyr::bind_rows(dataRes, .id = "gridID"))
res[wNonEmpty,] <- dataRes[,-1]
resSf <- st_as_sf(data.frame(res, st_geometry(grid)))
```

And finally plot the grid summary as a map:

```
palBW <- leaflet::colorNumeric(c("white", "navyblue"),
                               c(0, max(resSf$nSpp, na.rm = TRUE)),
                               na.color = "transparent")
oldpar <- par()
par(mar = c(1,1,0,0))
plot(resSf$geometry, col = palBW(resSf$nSpp), border = NA)
plot(swe_wgs84$Border$geometry, border = 1, lwd = 1, add = T)
legend("bottomleft",
       legend = round(seq(0, max(resSf$nSpp, na.rm = TRUE), length.out = 5)),
       col = palBW(seq(0, max(resSf$nSpp, na.rm = TRUE), length.out = 5)),
       title = "Number of \nspecies", pch = 15, bty="n")
par(oldpar)
```

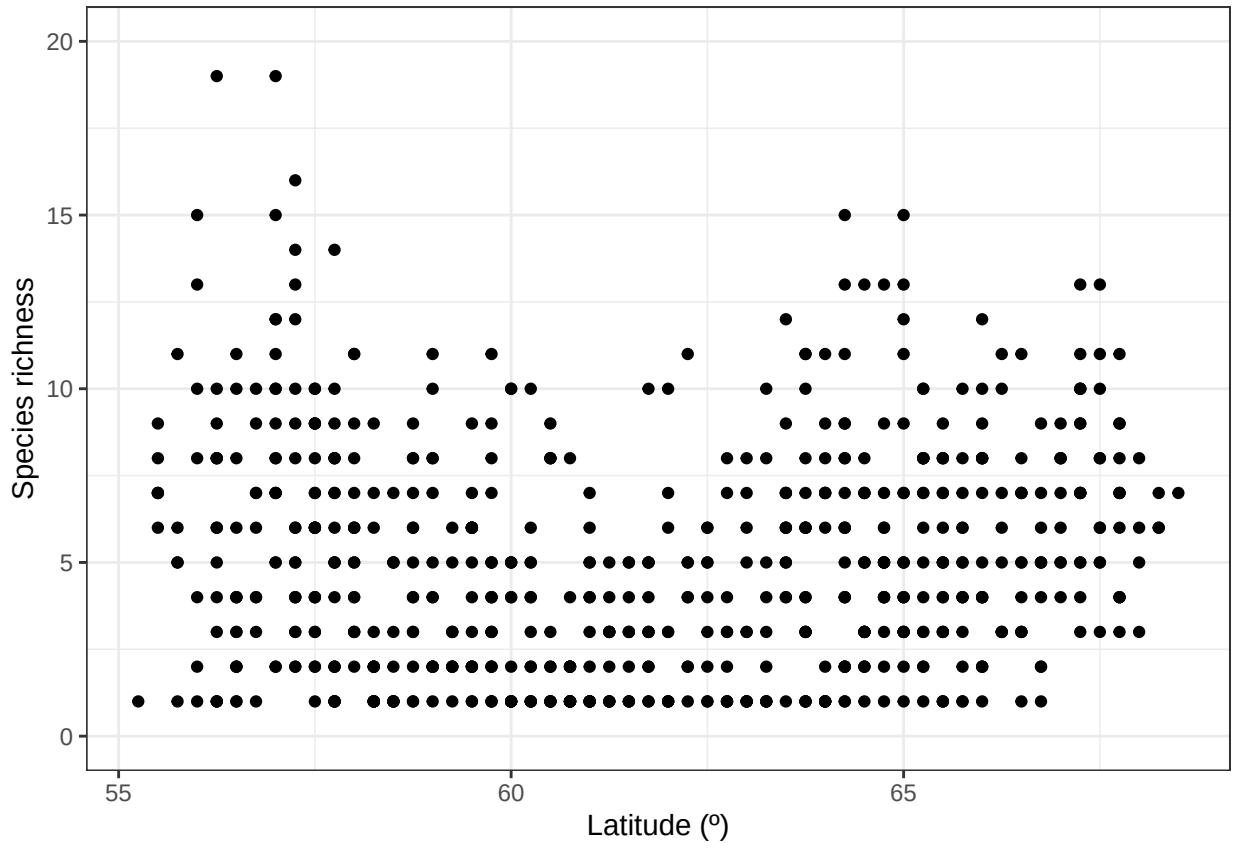


We may now ask whether species richness varies across latitude. So we go further by arranging the observations by latitude:

```r
library(dplyr)
library(tidyr)
xgridded <- xf |>
    mutate(longitude = round(decimalLongitude * 4)/4,
           latitude = round(decimalLatitude * 4)/4) |>
    group_by(longitude,latitude) |>
    ## subset to vars of interest
    select(longitude, latitude, scientificName) |>
    ## take one row per cell per species (presence)
    distinct() |>
    ## calculate species richness
    mutate(richness = n()) |>
    ## convert to wide format (sites by species)
    mutate(present = 1) |>
    do(tidyr::pivot_wider(data = .,
                          names_from = scientificName,
                          values_from = present,
                          values_fill = 0)) |>
    ungroup()
## where a species was not present, it will have NA: convert these to 0
sppcols <- setdiff(names(xgridded),
                   c("longitude", "latitude", "richness"))
xgridded <- xgridded |>
  mutate_at(sppcols, function(z) ifelse(is.na(z), 0, z))
```

And plot it accordingly:

```r
library(ggplot2)

ggplot(xgridded, aes(latitude, richness)) +
  labs(x = "Latitude (º)",
       y = "Species richness") +
  lims(y = c(0,20)) +
  geom_point() +
  theme_bw()
```



2 Example with opportunistic data on Dragonflies

In this example we are interested in exploring opportunistically collected data from the Swedish citizen science species observation portal - Artportalen.

2.1 Name searching

To begin with, we want be sure there is an unequivocal way to find the species within the order Odonata (dragonflies) and nothing else, so let's search for "odonata":

```
sx <- sbdi_call() |>
    sbdi_identify("Odonata") |>
    group_by(species) |>
    atlas_counts()
```

Now we can download the taxonomic data (note that the search is case-sensitive):

```
tx <- sbdi_call() |>
  sbdi_identify("Odonata") |>
  atlas_species() |>
  select("taxon_concept_id", "species_name", "taxon_rank","order", "family","genus", "vernacular_name")
```

You can save the `tx` object as the complete species list for later use.

2.2 Filter the search to get the observations

We start by searching for the data resource we are interested in using the function `pick_filter()`. This is an interactive query guiding you through the many resources available to filtering your query (data resources, spatial layers, and curated species lists).

```
# follow the instructions
fq_str <- pick_filter("resource")
```

Follow the instructions. Your choices here would have been "in3" –> "dr5". Your variable `fq_str` will now contain a string "data_resource_uid:dr5".

NOTE: the function pick_filter() is temporarily disabled until it could be adapted to the new galah framework.

We also want to filter spatially for Southern Sweden (Götaland).

Vector spatial layers (eg. polygons) can be imported in a number of different ways. SBDI APIs take as search input polygons in the so-called WKT Well Known Text format. So the first step is to load a vector layer and transform it into a WKT string. You could instead use the data we provided in the sbdi4r2 package `data("swe")`.

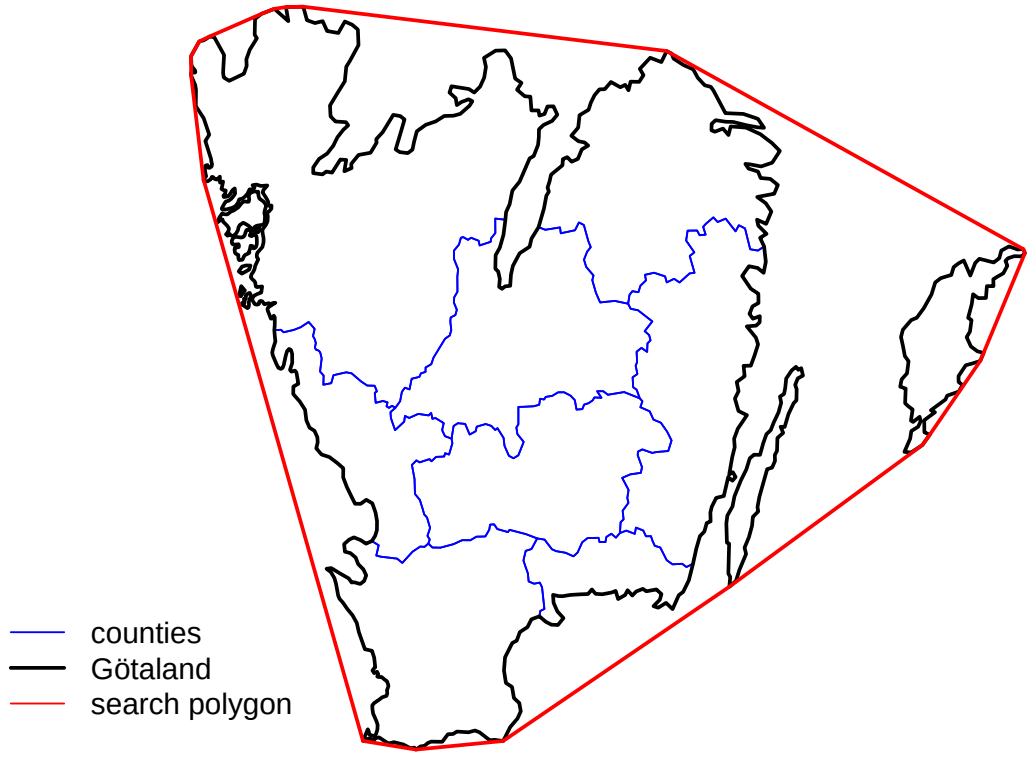
```
data("swe",package = "sbdi4r2")
wGotaland <- swe$Counties$LnNamn %in% c("Blekinge", "Gotlands", "Hallands",
                                        "Jönköpings", "Kalmar", "Kronobergs",
                                        "Östergötlands", "Skåne", "Västra Götalands")
gotaland_c <- swe$Counties[wGotaland,]
```

There are details about this polygon that we need to take care before. The WKT string should not be too long to be accepted by the API service. Also, the polygon we just got is projected in the coordinate system SWEREF99 TM, and the API service only accepts coordinates in a geodesic coordinate system WGS84. Let's construct the WKT string:

```
# transform the CRS
gotaland_c <- st_transform(gotaland_c,
                           crs = st_crs(4326))

# disolve the counties into one polygon
gotaland <- st_union(gotaland_c)

# create a convex hull of the polygon to simplify the geometry and
# reduce the length of the WKT string
gotaland_ch <- st_convex_hull(gotaland)

# create WKT string
wkt <- st_as_text(gotaland_ch)
```

The WKT string then looks like this:

```
## [1] "POLYGON ((11.13145 59.01184, 11.13161 58.90942, 11.25342 58.35786, 11.25893 58.34821, 12.81633 5
```



Next, we download the observations using the command `sbdi_call()`, but this time using `sbdi_geolocate()` to pass the search area.

```
xf <- sbdi_call() |>
  sbdi_identify("Odonata") |>
  sbdi_geolocate(wkt, type = "polygon") |>
  filter(dataResourceUid == "dr5",
         year >= 2010, year <= 2020) |>
  select(locality, recordedBy, taxonRank, group = "basic") |>
  atlas_occurrences()
```

```
## Request for 107107 occurrences placed in queue
## Current queue length: 1
```

```
## ----
```

```
## Downloading
```

```
save(xf, file = "an_appropriate_name.rdata")
load(file = "an_appropriate_name.rdata")
```

2.3 Quality and fit-for-use check

Before we can use the observation records we need to know if the observation effort (sampling effort) has varied over time and in space. We can approximate observation effort from the data by defining field visits i.e. occasions at which an observer has sampled observations. We reconstruct field visits (that is, assign each observation a visitUID) using using the package BIRDS. Additionally we want the data to be summarized over a grid of 25 km (provided through the sbdi4r2 package). The following functions will perform many different summaries at the same time. Please refer to the BIRDS package documentation for more detail.

```
remotes::install_github("Greensway/BIRDS")
library(BIRDS)
```

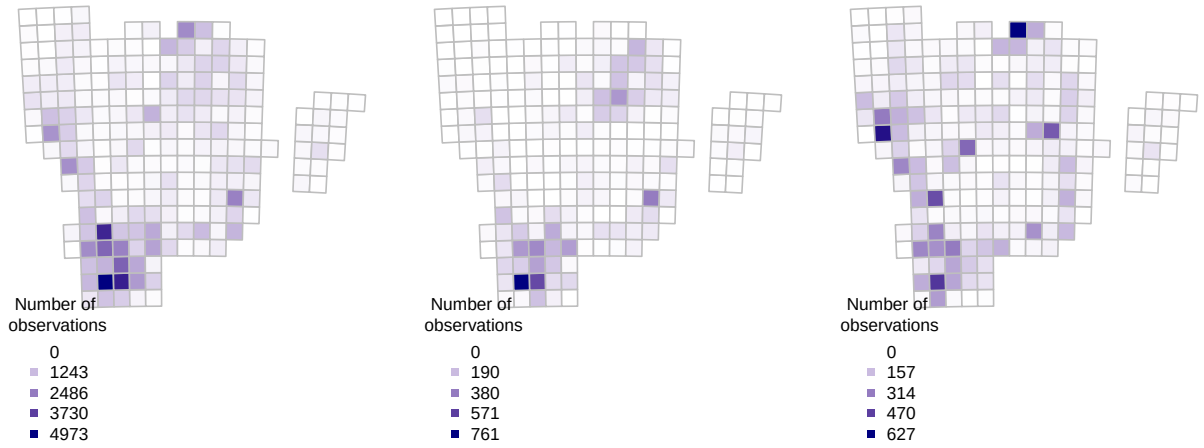
```
OB <- organiseBirds(xf, sppCol = "scientificName" ,
                    # We only want observations identified at the species level
                    taxonRankCol = "taxonRank", taxonRank = "species",
                    # the visits are defined by collector and named locality
                    idCols = c("locality", "recordedBy"),
                    timeCols = "eventDate",
                    xyCols = c("decimalLongitude","decimalLatitude") )

# We don't need the whole grid, just the piece that overlaps our searching polygon
wInt <- unlist(st_intersects(gotaland, Sweden_Grid_25km_Wgs84))
gotaland_grid25 <- Sweden_Grid_25km_Wgs84[wInt,]

SB <- summariseBirds(OB, grid = gotaland_grid25, spillOver = "unique")
```

Once summarised, we can see over space and for a few selected years how the number of observations is distributed:

```
maxC <- max(SB$spatial$nObs, na.rm = TRUE)
palBW <- leaflet::colorNumeric(c("white", "navyblue"),
                               c(0, maxC),
                               na.color = "transparent")
oldpar <- par()
par(mar = c(1,1,1,1), mfrow=c(1,3))
plot(SB$spatial$geometry, col=palBW(SB$spatial$nObs),
     border = "grey", main="All years") ## with palette
legend("bottomleft", inset = c(0,0.05),
       legend = round(seq(0, maxC, length.out = 5)),
       col = palBW(seq(0, maxC, length.out = 5)),
       title = "Number of \nobservations", pch = 15, bty="n")

## or export other combinations, e.g. one map per observed year
yearlySp <- exportBirds(SB,
                        dimension = "spatial",
                        timeRes = "yearly",
                        variable = "nObs",
                        method = "sum")

maxC <- max(yearlySp$'2010', na.rm = TRUE)
palBW <- leaflet::colorNumeric(c("white", "navyblue"),
                               c(0, maxC),
                               na.color = "transparent")

plot(yearlySp$geometry, col = palBW(yearlySp$'2010'),
     border = "grey",main = "2010")
legend("bottomleft", inset = c(0,0.05),
       legend = round(seq(0, maxC, length.out = 5)),
       col = palBW(seq(0, maxC, length.out = 5)),
       border = "grey",
       title = "Number of \nobservations",
       pch = 15, bty = "n")
```

```
maxC <- max(yearlySp$'2020', na.rm = TRUE)
palBW <- leaflet::colorNumeric(c("white", "navyblue"),
                               c(0, maxC),
                               na.color = "transparent")

plot(yearlySp$geometry, col = palBW(yearlySp$'2020'),
     border = "grey",main = "2020")
legend("bottomleft", inset = c(0,0.05),
       legend = round(seq(0, maxC, length.out = 5)),
       col = palBW(seq(0, maxC, length.out = 5)),
       border = "grey",
       title = "Number of \nobservations", pch = 15, bty="n")
par(oldpar)
```

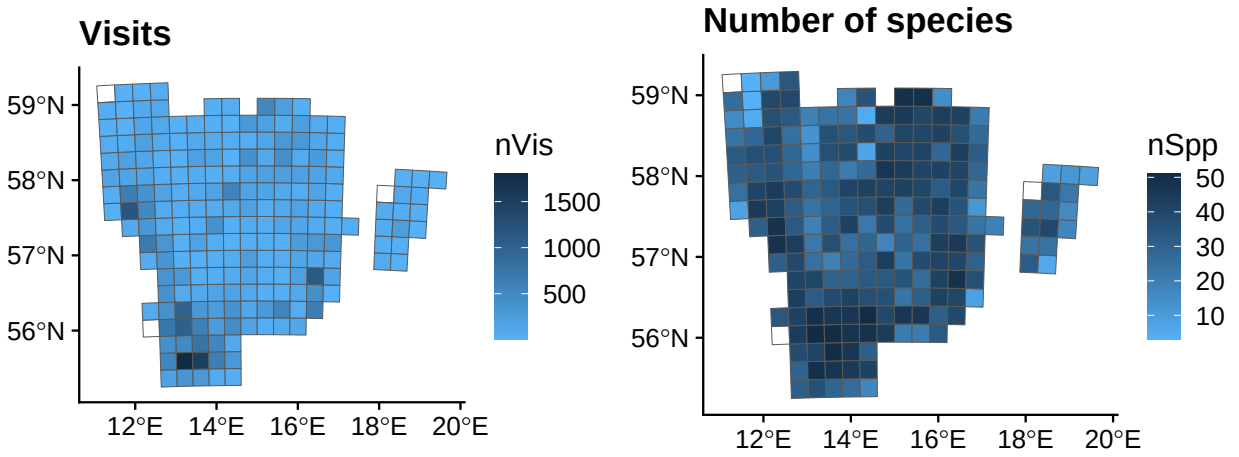
All years **2010** **2020**



We now want to use the number of field visits as the measure for sampling effort. :

```
library(cowplot)
library(ggplot2)
library(colorRamps)
library(gridExtra)

vis <- ggplot(data = SB$spatial, aes( fill = nVis)) +
  geom_sf() +
  ggtitle("Visits") +
  scale_fill_gradient(low = "#56B1F7",
                      high = "#132B43",
                      na.value = NA) +
  theme(plot.margin = margin(1, 1, 1, 1, "pt")) +
  theme_cowplot()

spp <- ggplot(data = SB$spatial, aes( fill = nSpp)) +
```

```
  geom_sf() +
  ggtitle("Number of species") +
  scale_fill_gradient(low = "#56B1F7",
                      high = "#132B43",
                      na.value = NA) +
  theme(plot.margin = margin(1, 1, 1, 1, "pt")) +
  theme_cowplot()

grid.arrange(vis, spp, ncol = 2)
```



2.3.0.1 Temporal check We see that `SB` contains an element called `SB$temporal` that contains a daily time series with time-specific rows when there is information. `xts` also supports day time, but dating below day resolution is not yet implemented in the `BIRDS` package.

```
sb.xts <- SB$temporal
head(sb.xts, 5)
```

```
##            nObs nVis nSpp
## 2010-03-22    3    3    2
## 2010-03-27    1    1    1
## 2010-04-02    2    1    2
## 2010-04-03    6    2    5
## 2010-04-12    7    6    2
```

Sub-setting is convenient in `xts` as you can do it with its dates and with a `/` for a range of dates.

```
sb.xts["2010-09-07"] #a specific day
```

```
##            nObs nVis nSpp
## 2010-09-07   19   10   12
```

```
sb.xts["2010-09-01/2010-09-15"] #for a period
```

```
##            nObs nVis nSpp
## 2010-09-01   46   19   14
## 2010-09-02   28   14   12
## 2010-09-03   23   10   10
## 2010-09-04   56   18   18
## 2010-09-05   73   26   11
## 2010-09-06   16    4    9
## 2010-09-07   19   10   12
## 2010-09-08   13    6    8
## 2010-09-09   32   12   14
## 2010-09-10    2    2    1
## 2010-09-11   16    9    8
## 2010-09-12   20   10    8
## 2010-09-13   14    5    9
## 2010-09-15    3    3    2
```

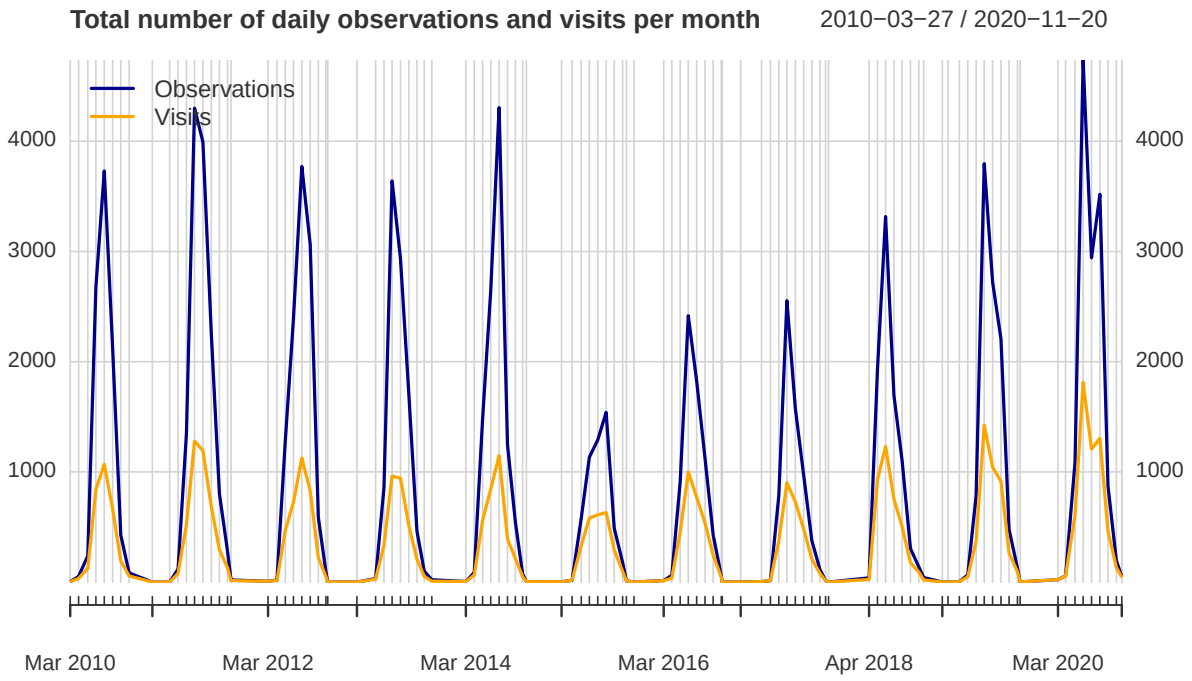
```
sb.xts["2010-09"] #a specific month
```

```
##            nObs nVis nSpp
## 2010-09-01   46   19   14
## 2010-09-02   28   14   12
## 2010-09-03   23   10   10
## 2010-09-04   56   18   18
## 2010-09-05   73   26   11
## 2010-09-06   16    4    9
## 2010-09-07   19   10   12
## 2010-09-08   13    6    8
## 2010-09-09   32   12   14
## 2010-09-10    2    2    1
## 2010-09-11   16    9    8
## 2010-09-12   20   10    8
## 2010-09-13   14    5    9
## 2010-09-15    3    3    2
## 2010-09-17    3    2    3
## 2010-09-18    9    5    5
## 2010-09-19   12    7    5
## 2010-09-21    3    2    3
## 2010-09-22    4    4    2
## 2010-09-23    3    3    2
## 2010-09-24   10    5    5
## 2010-09-25    7    4    6
## 2010-09-26    7    6    2
## 2010-09-28    2    2    2
## 2010-09-29    5    3    4
## 2010-09-30    2    2    2
```

The package `xts` has several tools for converting to different time periods. Here we use `apply.monthly` to obtain the total number of observations and visits per month. The plot command for an object of calss `xts` provides a many features. This makes it fairly easy to customize your plots. Read more in `?plot.xts`.

```
library(xts)
obs.m <- apply.monthly(sb.xts$nObs, "sum", na.rm = TRUE)
vis.m <- apply.monthly(sb.xts$nVis, "sum", na.rm = TRUE)

plot(obs.m,
     col = "darkblue",
```

```
     grid.ticks.on = "month",
     major.ticks = "year",
     grid.col = "lightgrey",
     main = "Total number of daily observations and visits per month")

lines(vis.m, col = "orange", lwd = 2, on = 1)
```



2.4 Species trends

We can now look at some particular species and ask whether those have changed in occurrence over time:

```
speciesSummary(SB)[,1:4]
```

```
##                      species nCells nObs nVis
## 1              Aeshna affinis     11  166  154
## 2             Aeshna caerulea     11   31   30
## 3               Aeshna cyanea    186 3930 3804
## 4              Aeshna grandis    197 6001 5896
## 5             Aeshna isoceles     42  475  469
## 6               Aeshna juncea    156 1302 1261
## 7                Aeshna mixta    145 2693 2618
## 8              Aeshna serrata     19  135  126
## 9           Aeshna subarctica     63  329  318
## 10             Aeshna viridis     59  225  194
## 11            Anax ephippiger      6   15   15
## 12             Anax imperator    103 2612 2470
## 13            Anax parthenope     31  276  272
## 14       Brachytron pratense    158 1668 1619
## 15      Calopteryx splendens    145 2419 2279
## 16          Calopteryx virgo    164 3742 3577
## 17        Coenagrion armatum     54  235  216
```

```
## 18      Coenagrion hastulatum    166 2532 2458
## 19      Coenagrion johanssoni     28  136  129
## 20       Coenagrion lunulatum    104  487  462
## 21          Coenagrion puella    179 3725 3567
## 22      Coenagrion pulchellum    177 3708 3586
## 23     Cordulegaster boltonii    113 1178 1164
## 24             Cordulia aenea    178 3579 3516
## 25      Crocothemis erythraea      2   13   13
## 26      Enallagma cyathigerum    193 5498 5190
## 27        Epitheca bimaculata     56  290  277
## 28           Erythromma najas    160 2438 2354
## 29       Erythromma viridulum     37  546  505
## 30      Gomphus vulgatissimus     99  770  752
## 31           Ischnura elegans    171 3586 3456
## 32           Ischnura pumilio     60  399  364
## 33            Lestes barbarus      3   43   39
## 34               Lestes dryas     90  614  584
## 35              Lestes sponsa    184 4699 4457
## 36              Lestes virens     90  683  651
## 37     Leucorrhinia albifrons    131  747  712
## 38      Leucorrhinia caudalis    116  464  444
## 39         Leucorrhinia dubia    129 1109 1078
## 40    Leucorrhinia pectoralis    140 1061 1035
## 41     Leucorrhinia rubicunda    153 1488 1442
## 42         Libellula depressa    174 2664 2574
## 43            Libellula fulva     24  293  268
## 44   Libellula quadrimaculata    198 6949 6837
## 45        Nehalennia speciosa     11  108  103
## 46   Onychogomphus forcipatus    100  937  919
## 47       Ophiogomphus cecilia     11   13   13
## 48      Orthetrum cancellatum    181 3806 3659
## 49     Orthetrum coerulescens    113 1069 1020
## 50       Platycnemis pennipes    128 1460 1404
## 51        Pyrrhosoma nymphula    169 3146 3074
## 52       Somatochlora arctica     61  192  180
## 53 Somatochlora flavomaculata    133 1065 1050
## 54     Somatochlora metallica    161 1718 1699
## 55              Sympecma fusca    106  860  839
## 56            Sympetrum danae    182 2998 2895
## 57        Sympetrum flaveolum    146 1012  980
## 58     Sympetrum fonscolombii     48  359  341
## 59     Sympetrum pedemontanum      2  143  127
## 60       Sympetrum sanguineum    189 4841 4644
## 61       Sympetrum striolatum    146 1468 1417
## 62         Sympetrum vulgatum    177 3431 3275
```

We pick two species and compare their trends in number of visits where the species where reported, relative to the total number of visits.

```
library(dplyr)
sppCount <- obsData(OB) |>
    group_by(year, visitUID) |>
    summarise("focalCountLq" = sum(scientificName == "Libellula quadrimaculata"),
              "focalCountSd" = sum(scientificName == "Sympetrum sanguineum"),
              "sppLength" = length(unique(scientificName)),
```

```
                .groups = "drop") |>
    ungroup() |>
    group_by(year) |>
    summarise("focalCountLq" = sum(focalCountLq),
              "focalCountSd" = sum(focalCountSd),
              "nVis" = length(unique(visitUID)),
              "relCountLq" = focalCountLq / nVis,
              "relCountSd" = focalCountSd / nVis,
              .groups = NULL)

oldpar <- par(no.readonly = TRUE)
plot(sppCount$year, sppCount$relCountLq,
     type = "l", lwd = 3, xlab = "Year",
     ylab = "Relative number of visits with observations",
     ylim = c(0, max(sppCount$relCountLq)),
     xaxp = c(2000, 2010, 10))
lines(sppCount$year, sppCount$relCountSd, lwd = 3, col = "#78D2EB")
legend("bottomright",
       legend = c("Libellula quadrimaculata","Sympetrum sanguineum"),
       text.font = 3, col = c("black", "#78D2EB"), lwd = 3, bty = "n")
```


```
par(oldpar)
```