In the below we determined what birds were observed in the county of Constance.

**Using rebird while waiting for the eBird’s full dataset**

In the following, we’ll use the rOpenSci’s package rebird to get and map all observations in the last 30 days near Radolfzell in Germany.

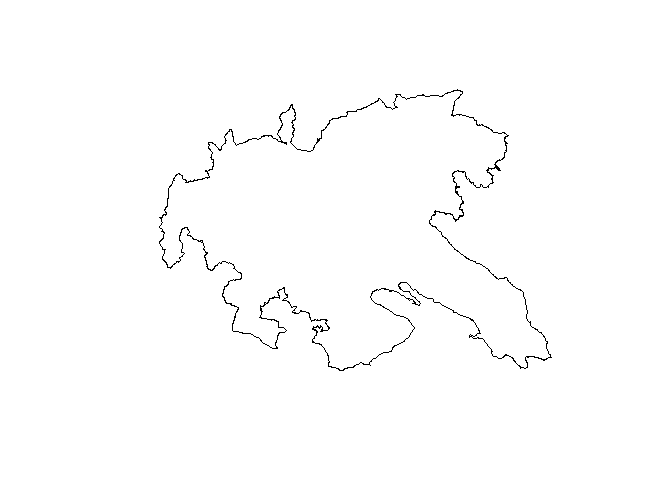
The *Radolfzell* part of that sentence is a bit different: I want all observations inside the polygon of the district of Constance (Landkreis Konstanz, including Radolfzell… and a protected natural area!) so I’ll first need to get it. For doing that I’ll use osmdata::getbb, that uses the free Nominatim API provided by Openstreetmap.

library("sf")

landkreis\_konstanz <- osmdata::getbb("Landkreis Konstanz",

format\_out = "sf\_polygon")

plot(landkreis\_konstanz)



Neither rebird nor spocc currently offer built-in trimming of occurrence data to a polygon (whereas osmdata does). A further difficulty created by eBird’s API is that it doesn’t allow for the use of a bounding box, but instead demands a lat, lng and a dist defining the radius of interest from given lat/lng in kilometers. Thanks to Marco Sciaini for providing me with an easy way to compute dist, using the sf package.

coord <- sf::st\_coordinates(landkreis\_konstanz)

bbox <- c(x1 = min(coord[, "X"]),

x2 = max(coord[, "X"]),

y1 = min(coord[, "Y"]),

y2 = max(coord[, "Y"]))

center <- c(x = (bbox["x1"] + bbox["x2"])/2,

y = (bbox["y1"] + bbox["y2"])/2)

dist <- landkreis\_konstanz %>%

sf::st\_cast("POINT") %>%

sf::st\_distance() %>%

max() \* 0.5

dist

## 24129.15 m

Now, we can make the query.

birds <- rebird::ebirdgeo(species = **NULL**,

lng = center["x.x1"],

lat = center["y.y1"],

back = 30,

dist = as.numeric(

units::set\_units(dist, "km")))

nrow(birds)

## [1] 55

str(birds)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 55 obs. of 12 variables:

## $ lng : num 8.94 8.94 8.94 8.94 8.94 ...

## $ locName : chr "Radolfzeller Aachmündung (Bodensee)" "Radolfzeller Aachmündung (Bodensee)" "Radolfzeller Aachmündung (Bodensee)" "Radolfzeller Aachmündung (Bodensee)" ...

## $ sciName : chr "Chroicocephalus ridibundus" "Motacilla alba" "Rallus aquaticus" "Aythya fuligula" ...

## $ obsValid : logi TRUE TRUE TRUE TRUE TRUE TRUE ...

## $ locationPrivate: logi FALSE FALSE FALSE FALSE FALSE FALSE ...

## $ obsDt : chr "2018-08-08 13:30" "2018-08-08 13:30" "2018-08-08 13:30" "2018-08-08 13:30" ...

## $ obsReviewed : logi FALSE FALSE FALSE FALSE FALSE FALSE ...

## $ comName : chr "Black-headed Gull" "White Wagtail" "Water Rail" "Tufted Duck" ...

## $ lat : num 47.7 47.7 47.7 47.7 47.7 ...

## $ locID : chr "L3314048" "L3314048" "L3314048" "L3314048" ...

## $ locId : chr "L3314048" "L3314048" "L3314048" "L3314048" ...

## $ howMany : int NA 2 1 1 NA 3 NA NA 8 20 ...

Now that we have the occurrence data, let’s plot it to see whether trimming is required.

crs <- sf::st\_crs(landkreis\_konstanz)

birds\_sf <- sf::st\_as\_sf(birds,

coords = c("lng", "lat"),

crs = crs)

library("ggplot2")

ggplot() +

geom\_sf(data = landkreis\_konstanz) +

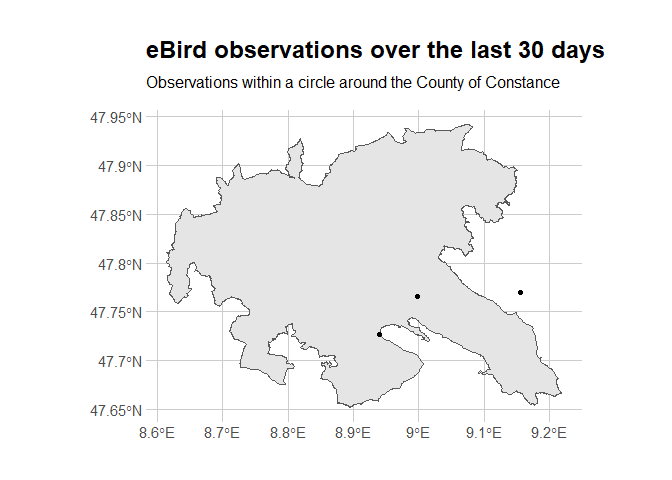
geom\_sf(data = birds\_sf) +

theme(legend.position = "bottom") +

hrbrthemes::theme\_ipsum() +

ggtitle("eBird observations over the last 30 days",

subtitle = "Observations within a circle around the County of Constance")



Yes, trimming is required! It’d have been too bad not to learn how to do it, anyway. We also add the MPI to the map.

*# which parts of the oject are in the county*

in\_indices <- sf::st\_within(birds\_sf, landkreis\_konstanz)

*# filter them*

trimmed\_birds <- dplyr::filter(birds\_sf,

lengths(in\_indices) > 0)

*# summarize to get no. of birds by location*

summarized\_birds <- trimmed\_birds %>%

dplyr::group\_by(locName) %>%

dplyr::summarise(n = n())

*# MPI*

mpi <- opencage::opencage\_forward("Am Obstberg 1 78315 Radolfzell",

limit = 1)$results

coords <- data.frame(lon = mpi$geometry.lng,

lat = mpi$geometry.lat)

crs <- sf::st\_crs(landkreis\_konstanz)

mpi\_sf <- sf::st\_as\_sf(coords,

coords = c("lon", "lat"),

crs = crs)

*# Map!*

ggplot() +

geom\_sf(data = landkreis\_konstanz) +

geom\_sf(data = summarized\_birds,

aes(size = n), show.legend = "point") +

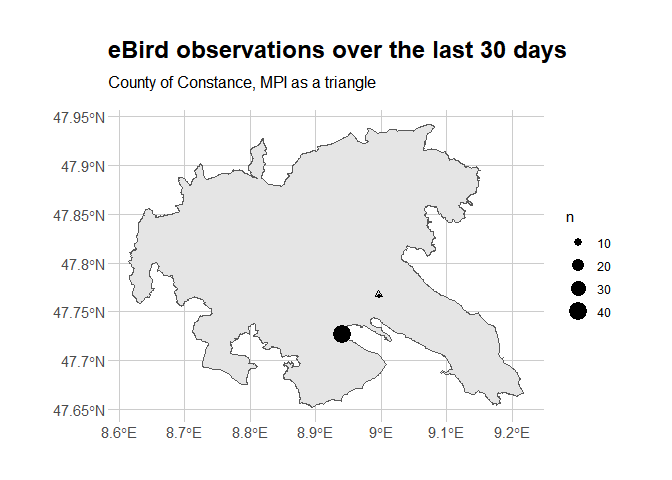
hrbrthemes::theme\_ipsum() +

ggtitle("eBird observations over the last 30 days",

subtitle = "County of Constance, MPI as a triangle") +

geom\_sf(data = mpi\_sf,

shape = 2)



We got 49 observations (nrow(trimmed\_birds)) of 49 species (length(unique(trimmed\_birds$comName))), over 2 places (length(unique(trimmed\_birds$locName))) during 5 observation sessions. Hopefully merely an appetizer to what we can get from using the full eBird dataset in the next section…

Note that the initial query could have been made with spocc which would have helped using the rOpenSci occurrence suite.

birds2 <- spocc::occ(from = "ebird",

ebirdopts = list(method = "ebirdgeo",

species = **NULL**,

lng = center["x.x1"],

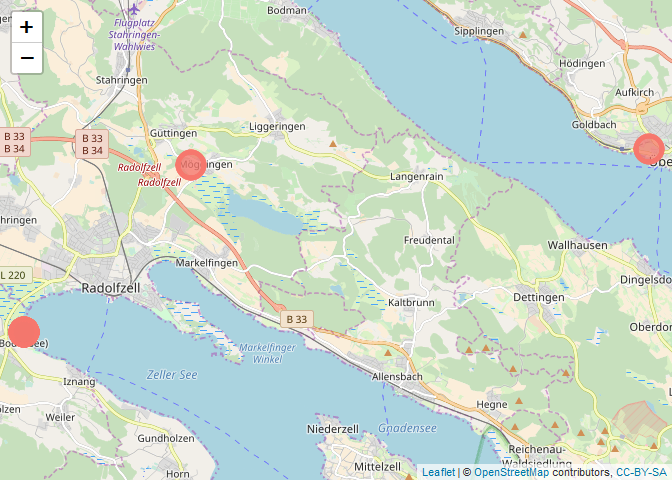
lat = center["y.y1"],

back = 30,

dist = as.numeric(

units::set\_units(dist, "km"))))

mapr::map\_leaflet(birds2)



Quite handy!

Now, let’s explore the whole eBird dataset for Germany.

🔗**Using auk to process EBD dataset for Germany**

After getting access to a custom dataset corresponding to the EBD for Germany only, I used auk’s documentation and this post to learn how to process it. Since I wasn’t planning on zero-filling the data to get presence/absence counts, I was able to ignore the sampling event data that contains the checklist-level information (e.g. time and date, location, and search effort information). For an example of a more advanced auk workflow involving the **full** EBD, and sampling data, refer to Matt Strimas-Mackey’s own blog post about his package.

🔗**Preparing the dataset**

Here, the workflow is to *clean* the data and to *filter* it using one of auk’s built-in filters and then polygon filtering as earlier in this post. All steps are quite fast, because the custom dataset for Germany isn’t too big (a few hundred megabytes).

Cleaning happens in the following:

ebd\_dir <- "C:/Users/Maelle/Documents/ropensci/ebird"

f <- file.path(ebd\_dir, "ebd\_DE\_relMay-2018.txt")

f\_clean <- file.path(ebd\_dir, "ebd\_DE\_relMay-2018\_clean.txt")

auk::auk\_clean(f, f\_out = f\_clean, remove\_text = **TRUE**)

Then one can filter the data. Note that the auk\_extent function that only retains observations within a bounding box has been renamed auk\_bbox in the dev version of auk, the old name will be deprecated soon.

ebd\_dir <- "C:/Users/Maelle/Documents/ropensci/ebird"

f\_in\_ebd <- file.path(ebd\_dir, "ebd\_DE\_relMay-2018\_clean.txt")

library("magrittr")

landkreis\_konstanz\_coords <- sf::st\_coordinates(landkreis\_konstanz)

ebd\_filter <- auk::auk\_ebd(f\_in\_ebd) %>%

auk::auk\_extent(c(min(landkreis\_konstanz\_coords[, "X"]),

min(landkreis\_konstanz\_coords[, "Y"]),

max(landkreis\_konstanz\_coords[, "X"]),

max(landkreis\_konstanz\_coords[, "Y"])))

ebd\_filter

## Input

## EBD: C:\Users\Maelle\Documents\ropensci\ebird\ebd\_DE\_relMay-2018\_clean.txt

##

## Output

## Filters not executed

##

## Filters

## Species: all

## Countries: all

## States: all

## BCRs: all

## Spatial extent: Lon 8.6 - 9.2; Lat 47.7 - 47.9

## Date: all

## Start time: all

## Last edited date: all

## Protocol: all

## Project code: all

## Duration: all

## Distance travelled: all

## Records with breeding codes only: no

## Complete checklists only: no

fs::dir\_create("ebird")

f\_out\_ebd <- "ebird/ebd\_lk\_konstanz.txt"

f\_out\_sampling <- "ebird/ebd\_lk\_konstanz\_sampling.txt"

ebd\_filtered <- auk::auk\_filter(ebd\_filter, file = f\_out\_ebd,

overwrite = **TRUE**)

On top of this filtering with auk, after loading the data we filter observations inside the polygon of the county.

crs <- sf::st\_crs(landkreis\_konstanz)

ebd <- auk::read\_ebd(f\_out\_ebd) %>%

sf::st\_as\_sf(coords = c("longitude", "latitude"),

crs = crs)

in\_indices <- sf::st\_within(ebd, landkreis\_konstanz)

ebd <- dplyr::filter(ebd, lengths(in\_indices) > 0)

ebd <- as.data.frame(ebd)

🔗**What are the observed birds?**

Before looking at species names, let’s have a brief look at the size and temporal extent of the data.

library("ggplot2")

dim(ebd)

## [1] 10156 41

ebd %>%

dplyr::mutate(year = lubridate::year(observation\_date)) %>%

ggplot() +

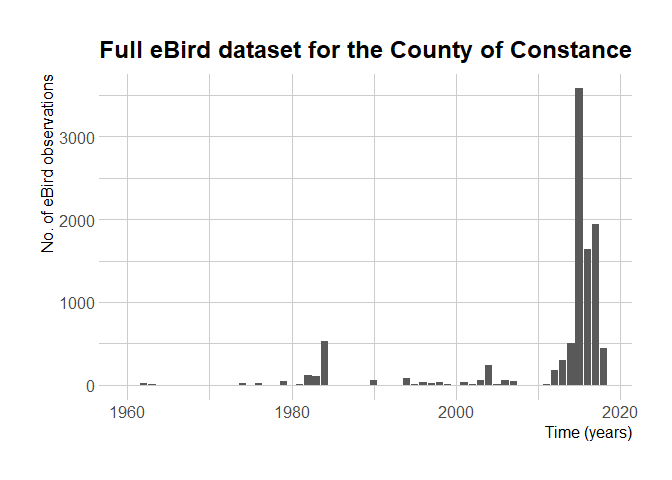
geom\_bar(aes(year)) +

hrbrthemes::theme\_ipsum(base\_size = 12, axis\_title\_size = 12, axis\_text\_size = 12) +

ylab("No. of eBird observations") +

xlab("Time (years)") +

ggtitle("Full eBird dataset for the County of Constance")



eBird started in 2002 but only became global in 2010. It allows people to enter older observations, though.

Now we can look at what birds have been reported the most.

ebd %>%

dplyr::filter(approved) %>%

dplyr::count(scientific\_name, common\_name) %>%

dplyr::arrange(- n) %>%

head(n = 10) %>%

knitr::kable()

| **scientific\_name** | **common\_name** | **n** |
| --- | --- | --- |
| Corvus corone | Carrion Crow | 288 |
| Turdus merula | Eurasian Blackbird | 285 |
| Anas platyrhynchos | Mallard | 273 |
| Fulica atra | Eurasian Coot | 268 |
| Parus major | Great Tit | 266 |
| Podiceps cristatus | Great Crested Grebe | 254 |
| Ardea cinerea | Gray Heron | 236 |
| Cygnus olor | Mute Swan | 234 |
| Cyanistes caeruleus | Eurasian Blue Tit | 233 |
| Chroicocephalus ridibundus | Black-headed Gull | 223 |

I had to google most of them, but only because I didn’t know the scientific and English names of these birds: they’re birds even I, not a birder, know, probably because they’re also common in Brittany where I grew up.

We can also look at birds whose observation was rejected. Out of 10156 observations only 64 were reviewed, and only 5 were not approved.

ebd %>%

dplyr::select(scientific\_name, common\_name,

approved, reviewed, reason) %>%

dplyr::filter(!approved) %>%

knitr::kable()

| **scientific\_name** | **common\_name** | **approved** | **reviewed** | **reason** |
| --- | --- | --- | --- | --- |
| Cygnus atratus | Black Swan | FALSE | TRUE | Species-Introduced/Exotic |
| Cygnus atratus | Black Swan | FALSE | TRUE | Species-Introduced/Exotic |
| Cygnus atratus | Black Swan | FALSE | TRUE | Species-Introduced/Exotic |
| Oxyura leucocephala | White-headed Duck | FALSE | TRUE | Species-Introduced/Exotic |
| Mareca sibilatrix | Chiloe Wigeon | FALSE | TRUE | Species-Introduced/Exotic |

Black Swans are mostly present in Australia, imported and escaped in a few other places but eBird mostly doesn’t accept the entry of exotic species although it’s debated. In any case, eBird’s curation of the data entered is quite admirable.

🔗**Who observed birds?**

Does the county of Constance have a super birder?

(first\_birder <- ebd %>%

dplyr::count(observer\_id) %>%

dplyr::arrange(- n) %>%

head(n = 1) )

## # A tibble: 1 x 2

## observer\_id n

## <chr> <int>

## 1 obsr457108 3551

(proportion <- round(first\_birder$n/nrow(ebd),

digits = 2))

## [1] 0.35

Wow, that person made 35% of eBird observations in the county! The EBD no longer provides names (consequence of the EU General Data Protection Regulation) but from the checklist ID one can get access to the checklist page e.g this one where the name of the observer is present. The super birder of the County of Constance is Antonio Anta Bink.

🔗**Conclusion**

🔗**R packages for occurrence data**

In this post I gave a rough view of what birds are present in the county around Radolfzell: Eurasian Blackbirds, Carrion Crows, Great Tits… but not Black Swans in eBird’s data. We mostly illustrated the use of two R packages accessing eBird’s data:

* auk for processing the gigantic whole eBird’s dataset.
* rebird for getting access to recent data via an API. rebird is part of a larger collection of packages for occurrence data within rOpenSci’s suite, with spocc being an umbrella package accessing several data sources; scrubr a helper for cleaning data obtained this way; and mapr a utility package for mapping such data.

We complemented this knowledge with some taxonomic and trait information below:

.

*# polygon for filtering*

landkreis\_konstanz <- osmdata::getbb("Landkreis Konstanz",

format\_out = "sf\_polygon")

crs <- sf::st\_crs(landkreis\_konstanz)

*# get and filter data*

f\_out\_ebd <- "ebird/ebd\_lk\_konstanz.txt"

library("magrittr")

ebd <- auk::read\_ebd(f\_out\_ebd) %>%

sf::st\_as\_sf(coords = c("longitude", "latitude"),

crs = crs)

in\_indices <- sf::st\_within(ebd, landkreis\_konstanz)

ebd <- dplyr::filter(ebd, lengths(in\_indices) > 0)

ebd <- as.data.frame(ebd)

ebd <- dplyr::filter(ebd, approved, lubridate::year(observation\_date) > 2010)

nrow(ebd)

## [1] 8599

We will also need these two data.frames later: abundance by species, and dictionary of names.

abundance <- dplyr::count(ebd, scientific\_name)

dico <- unique(dplyr::select(ebd, scientific\_name,

common\_name))

## 🔗Getting taxonomic information

In this section I would like to get an idea of how diverse the types of birds are in the County of Constance. I want to draw a phylogenetic tree of the local species, and for that, I’ll first retrieve the classification for each species from NCBI using the taxize package that “allows users to search over many taxonomic data sources for species names (scientific and common) and download up and downstream taxonomic hierarchical information - among other things.”.

We first query uid’s and then use the classification function, instead of passing the species name directly to classification, because the IDs are unique whereas results for species names aren’t. Rate limiting is thankfully managed by the package itself so we users do not need to worry about that.

ids <- taxize::get\_uid(unique(ebd$scientific\_name))

classif <- taxize::classification(ids)

fs::dir\_create("taxo")

save(classif, file = file.path("taxo", "classif.RData"))

There are 211 species, we get 211 elements in the classification (sum(lengths(classif)==3)), that is a list of data.frames:

load(file.path("taxo", "classif.RData"))

str(classif[[1]])

## 'data.frame': 31 obs. of 3 variables:

## $ name: chr "cellular organisms" "Eukaryota" "Opisthokonta" "Metazoa" ...

## $ rank: chr "no rank" "superkingdom" "no rank" "kingdom" ...

## $ id : chr "131567" "2759" "33154" "33208" ...

Now, we’ll represent the whole taxonomy as a tree, using the handy taxize::class2tree function and the great ggtree package by Guangchuang Yu.

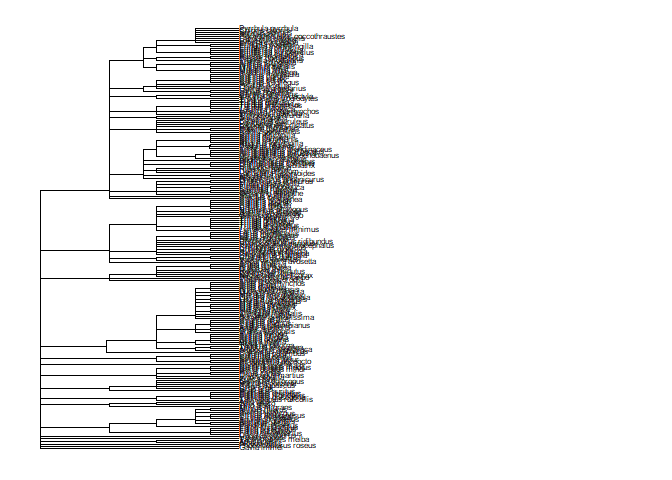
tree <- taxize::class2tree(classif)

library("ggplot2")

ggtree::ggtree(tree$phylo) +

ggtree::geom\_tiplab(aes(), size = 2, vjust=0.25) +

xlim(0, 150)



This tree is… unreadable. But at this point, it’s worth remembering that we got here using three taxize functions only: taxize::get\_uid, taxize::classification and taxize::class2tree. What a smooth workflow!

We’ll now concentrate on highlighting orders, inspired by this blog post.

There are 18 orders and I do not intend to add the highlighting command for each of them by hand! I’ll streamline the process, starting by automatically extracting the node ID of each order. The solution below might be a little over-complicated, so R taxonomy experts, please chime in! I transformed the tree phylo object to a phylo4 from the phylobase package maintained by François Michonneau, in order to easily retrieve all ancestor nodes for any group of species. Within an order, the order node ID is the highest common node ID.

p4 <- phylobase::phylo4(tree$phylo)

*# helper to translate labels*

translate <- function(scientific\_name){

if(scientific\_name %in% dico$scientific\_name){

dico$common\_name[dico$scientific\_name == scientific\_name]

}else{

scientific\_name

}

}

find\_order\_node\_id <- function(order, p4){

order\_members <- as.character(tree$classification$species[tree$classification$order == order])

nodes <- phylobase::ancestors(p4, order\_members, type = "ALL")

*# the ID is the higher common node IDs*

nodes <- purrr::map(nodes, as.numeric)

if(length(order\_members) > 1){

id <- max(Reduce(intersect, nodes))

}else{

id <- min(unlist(nodes))

}

common\_names <- purrr::map\_chr(order\_members, translate)

species <- stringr::str\_wrap(toString(common\_names),width = 50)

tibble::tibble(id = id,

order = order,

size = length(order\_members),

species = species)

}

orders <- purrr::map\_df(unique(tree$classification$order),

find\_order\_node\_id,

p4)

str(orders)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 18 obs. of 4 variables:

## $ id : num 237 238 235 239 231 10 228 226 224 222 ...

## $ order : Factor w/ 18 levels "Anseriformes",..: 3 12 13 1 5 4 15 11 16 17 ...

## $ size : int 32 86 9 35 5 1 7 6 5 3 ...

## $ species: chr "Black-headed Gull, Common Tern, Northern Lapwing,\nYellow-legged Gull, Common Sandpiper, Eurasian\nCurlew, Mew "| \_\_truncated\_\_ "Carrion Crow, Eurasian Magpie, House Sparrow,\nShort-toed Treecreeper, Eurasian Blackbird,\nEuropean Greenfinch"| \_\_truncated\_\_ "Gray Heron, Great Cormorant, Great Egret, Eurasian\nSpoonbill, Purple Heron, Cattle Egret, Little\nBittern, Lit"| \_\_truncated\_\_ "Mallard, Mute Swan, Common Goldeneye, Common\nMerganser, Common Pochard, Ferruginous Duck,\nGreen-winged Teal, "| \_\_truncated\_\_ ...

For each order, I’ll get a silhouette from Phylopic using Scott Chamberlain’s rphylopic package.

get\_results <- function(name){

id <- rphylopic::name\_search(name)

rphylopic::name\_images(id$canonicalName[1,1])

}

get\_pic <- function(order, classification){

message(order)

*# shortcurt for flamingos*

if(order == "Phoenicopteriformes"){

return(tibble::tibble(pic\_id = "28473411-c079-4654-bbb7-34a5615bb414",

order = "Phoenicopteriformes"))

}

classification <- classification[classification$order == order,]

results <- get\_results(order)

if (length(results$same) > 0){

*# best case*

pic\_id <- results$`same`[[1]]$`uid`

}else{

*# take the most common species*

*# and get any pic of it*

results <- get\_results(classification$species[

classification$n == max(classification$n, na.rm=**TRUE**)

][1])

results <- purrr::keep(results, function(x) length(x) > 0)

results <- unlist(results)

results <- results[length(results)]

pic\_id <- as.character(results)

}

tibble::tibble(pic\_id = pic\_id,

order = order)

}

library("magrittr")

classification <- tree$classification %>%

dplyr::mutate(species = as.character(species)) %>%

dplyr::left\_join(abundance,

by = c("species" = "scientific\_name"))

ids <- purrr::map\_df(orders$order, get\_pic, classification)

save(ids, file = file.path("taxo", "ids.RData"))

It is rather tricky to automatically get pics from Phylopic since you might not get one for the order itself, maybe one for the subtaxon instead, etc, so we made decisions blindly in the script above. In real life one might prefer selecting IDs by hand.

Now, we can highlight each order! One could add silhouettes to the tree itself with ggtree but I’ll add them on the side instead.

*# Get pics ids*

load(file.path("taxo", "ids.RData"))

*# Plot basic tree*

p <- ggtree::ggtree(tree$phylo)

*# Sort the orders by node id*

orders <- dplyr::arrange(orders, - id)

*# Helper to plot one order*

plot\_order <- function(order, orders,

ids, p){

*# Get index*

i <- which(orders$order == order)

*# From image ID get image itself*

*# and image metadata (copyright &co)*

img\_id <- ids$pic\_id[ids$order == order]

img <- rphylopic::image\_data(img\_id, 512)

img\_info <- rphylopic::image\_get(img\_id,

options = c("credit",

"licenseURL"))

if(is.null(img\_info$credit)){

img\_info$credit <- ""

}

*# Now, plot!*

p +

*# Highlight the order*

ggtree::geom\_hilight(node = orders$id[i],

fill = "salmon") +

*# Order name as title*

ggtitle(orders$order[i])+

xlim(0, 150) +

ylim(0, 250) +

*# Add species names on the side*

annotate("text", x = 110,

y = 200, label = orders$species[i],

size = 4) +

*# Credit at the bottom*

annotate("text", x = 110,

y = 0,

size = 2,

label = glue::glue("Silhouette: {img\_info$credit}\n{img\_info$licenseURL}"))

*# Save a first time*

filepath <- file.path("taxo", glue::glue("p{i}.png"))

ggsave(filepath, width = 7, height = 7)

*# Add silhouette via magick*

silhouette <- magick::image\_read(img[[1]])

magick::image\_read(filepath) %>%

magick::image\_composite(silhouette,

offset = "+1300+1400") %>%

magick::image\_write(filepath)

}

*# Create aaall plots*

purrr::walk(orders$order, plot\_order,

orders, ids, p)

Once we have created all these PNGs, we can join them into a gif using Jeroen Ooms’ gifski.

png\_files <- fs::dir\_ls("taxo", regexp = "[.]png$")

gifski::gifski(png\_files = png\_files,

gif\_file = file.path("2018-09-04-birds-taxo-traits\_files",

"figure-markdown\_github", "taxo.gif"),

delay = 3,

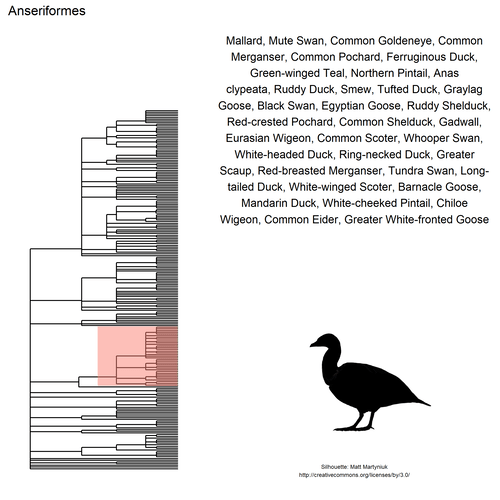
width = 500, height = 500,

progress = **FALSE**)

## [1] "/img/blog-images/2018-09-04-birds-taxo-traits/taxo.gif"

knitr::include\_graphics(file.path("2018-09-04-birds-taxo-traits\_files",

"figure-markdown\_github", "taxo.gif"))



This gif shows many species names and orders giving us a feeling for what we might encounter in the county of Constance, but it lacks quantitative information about the species. It’d be interesting to create trees such as the ones of the metacoder package to reflect abundance, possibly depending on the very local area (distance to watery area) or season, potentially using the taxize::downstream function to get all families in each order, even families not present in our occurrence dataset. This idea is beyond the scope of this post. What is in scope, now, is trying to get trait information for the species.

## 🔗Getting trait information

In ecology, traits are characteristics of organisms such as habitat, body size, threats, etc. It’s a whole bunch of data you can get for free based on species scientific names, from different data providers. The traits package, part of the rOpenSci’s suite, is an interface to various sources of traits data. In this section, we shall use data from BirdLife International: habitat and threats.

Now, we could be curious about the occurrence of these birds in *scientific  
work*. In this post, we will query the scientific literature and an open  
scientific data repository for species names: what have these birds been  
studied for? Read on if you want to learn how to use R packages allowing  
to do so!

**Getting a list of 50 species from occurrence data**

The single difference is our adding a step to keep only data for the  
most recent years.

# polygon for filtering

landkreis\_konstanz <- osmdata::getbb("Landkreis Konstanz",

format\_out = "sf\_polygon")

crs <- sf::st\_crs(landkreis\_konstanz)

# get and filter data

f\_out\_ebd <- "ebird/ebd\_lk\_konstanz.txt"

library("magrittr")

ebd <- auk::read\_ebd(f\_out\_ebd) %>%

sf::st\_as\_sf(coords = c("longitude", "latitude"),

crs = crs)

in\_indices <- sf::st\_within(ebd, landkreis\_konstanz)

ebd <- dplyr::filter(ebd, lengths(in\_indices) > 0)

ebd <- as.data.frame(ebd)

ebd <- dplyr::filter(ebd, approved, lubridate::year(observation\_date) > 2010)

For the sake of simplicity, we shall only use the 50 species observed  
the most often.

species <- ebd %>%

dplyr::count(common\_name, sort = TRUE) %>%

head(n = 50) %>%

dplyr::pull(common\_name)

The species are Carrion Crow, Eurasian Blackbird, Mallard, Eurasian  
Coot, Great Tit, Great Crested Grebe, Mute Swan, Great Cormorant,  
Eurasian Blue Tit, Gray Heron, Black-headed Gull, Common Chaffinch,  
Common Chiffchaff, Tufted Duck, European Starling, White Wagtail,  
European Robin, Little Grebe, Common Wood-Pigeon, Red-crested Pochard,  
Ruddy Shelduck, Graylag Goose, Red Kite, Common Buzzard, Eurasian  
Blackcap, Great Spotted Woodpecker, Eurasian Magpie, Gadwall, Common  
Pochard, Eurasian Nuthatch, Green-winged Teal, House Sparrow, Eurasian  
Jay, Yellow-legged Gull, Yellowhammer, Eurasian Green Woodpecker, Eared  
Grebe, Eurasian Reed Warbler, Barn Swallow, Northern Shoveler, Eurasian  
Moorhen, Black Redstart, Great Egret, White Stork, Eurasian Wren,  
Long-tailed Tit, Common House-Martin, Eurasian Kestrel, European  
Goldfinch and European Greenfinch  
(glue::glue\_collapse(species, sep = ", ", last = " and ")).

**Querying the scientific literature**

Just like rOpenSci has a taxonomic toolbelt  
(taxize) and a species  
occurrence data toolbelt (spocc),  
it has a super package for querying the scientific literature:  
fulltext! This package  
supports search for “PLOS via the rplos package, Crossref via the  
rcrossref package, Entrez via the rentrez package, arXiv via the aRxiv  
package, and BMC, Biorxiv, EuroPMC, and Scopus via internal helper  
functions”.

We shall use fulltext to retrieve the titles and abstracts of  
scientific articles mentioning each species, and will use tidytext to  
compute the most prevalent words in these works.

We first define a function retrieving the titles and abstracts of works  
obtained as result when querying one species name.

We use dplyr::bind\_rows because we want all results for one species at  
once, while fulltext returns a list of data.frames with one data.frame  
by data source.

.get\_papers <- function(species){

species %>%

tolower() %>%

fulltext::ft\_search() %>%

fulltext::ft\_get() %>%

fulltext::ft\_collect() %>%

fulltext::ft\_chunks(c("title", "abstract")) %>%

fulltext::ft\_tabularize() %>%

dplyr::bind\_rows()

}

.get\_papers(species[1]) %>%

dplyr::pull(title)

## [1] "Great spotted cuckoo nestlings have no antipredatory effect on magpie or carrion crow host nests in southern Spain"

## [2] "Donor-Control of Scavenging Food Webs at the Land-Ocean Interface"

## [3] "Formal comment to Soler et al.: Great spotted cuckoo nestlings have no antipredatory effect on magpie or carrion crow host nests in southern Spain"

## [4] "Socially Driven Consistent Behavioural Differences during Development in Common Ravens and Carrion Crows"

## [5] "Behavioral Responses to Inequity in Reward Distribution and Working Effort in Crows and Ravens"

## [6] "Early Duplication of a Single MHC IIB Locus Prior to the Passerine Radiations"

## [7] "Investigating the impact of media on demand for wildlife: A case study of Harry Potter and the UK trade in owls"

## [8] "New Caledonian Crows Rapidly Solve a Collaborative Problem without Cooperative Cognition"

## [9] "Nest Predation Deviates from Nest Predator Abundance in an Ecologically Trapped Bird"

## [10] "Dietary Compositions and Their Seasonal Shifts in Japanese Resident Birds, Estimated from the Analysis of Volunteer Monitoring Data"

If we were working on a scientific study, we’d add a few more filters,  
e.g. having the species mentioned in the abstract, and not only  
somewhere in the paper which is probably the way the different  
literature search providers define a match. But we’re not, so we can  
keep our query quite free! It is supported  
by open data (by the way you can access CITES trade data (international  
trade in endangered species) in R using  
cites and CITES  
Speciesplus database using  
rcites).

We then apply this function to all 50 species and keep each article only  
once.

get\_papers <- ratelimitr::limit\_rate(.get\_papers,

rate = ratelimitr::rate(1, 2))

all\_papers <- purrr::map\_df(species, get\_papers)

nrow(all\_papers)

## [1] 522

all\_papers <- unique(all\_papers)

nrow(all\_papers)

## [1] 378

Now, we get the most common words from titles and abstracts. For that we  
first append the title to the abstract which is a quick hack.

library("tidytext")

library("rcorpora")

stopwords <- corpora("words/stopwords/en")$stopWords

all\_papers %>%

dplyr::group\_by(title, abstract) %>%

dplyr::summarise(text = paste(title, abstract)) %>%

dplyr::ungroup() %>%

unnest\_tokens(word, text) %>%

dplyr::filter(!word %in% stopwords) %>%

dplyr::count(word, sort = TRUE) -> words

So, what are the most common words in these papers?

head(words, n = 10)

## word n

## 1 species 754

## 2 birds 514

## 3 virus 270

## 4 avian 268

## 5 bird 262

## 6 study 243

## 7 breeding 231

## 8 wild 227

## 9 populations 217

## 10 population 213

Not too surprising, and obviously less entertaining than looking at  
individual species’ results. Maybe a wordcloud can give us a better idea  
of the wide area of topics of studies involving our 50 most frequent  
bird species. We use the wordcloud  
package.

library("wordcloud")

with(words, wordcloud(word, n, max.words = 100))

wordcloud of titles and abstracts of scientific
papers

We see that topics include ecological words such as “foraging” but also  
epidemiological questions since “influenza” and “h5n1” come up. Now, how  
informative as this wordcloud can be, it’s a bit ugly, so we’ll prettify  
it using the wordcloud2  
package instead..

bird <- words %>%

head(n = 100) %>%

wordcloud2::wordcloud2(figPath = "bird.png",

color = "black", size = 1.5)

# https://www.r-graph-gallery.com/196-the-wordcloud2-library/

htmlwidgets::saveWidget(bird,

"tmp.html",

selfcontained = F)

I wasn’t able to webshot the resulting html despite increasing the  
delay parameter so I screenshot it by hand!

magick::image\_read("screenshot.png")



wordcloud shaped as a bird

The result is a bit kitsch, doesn’t include the word “species”, one  
needs to know it’s the silhouette of a bird to recognize it, and we’d  
need to work a bit on not reshaping the silhouette, but it’s fun as it  
is.

**Querying scientific open data**

There are quite a few scientific open data repositories out there, among  
which the giant [DataONE](https://www.dataone.org/) that has an API  
interfaced with an R package. We shall use it to perform a search  
similar to the previous section, but looking at the data indexed on  
DataONE. Since DataONE specializes in ecological and environmental data,  
we expect to find rather ecological data.

We first define a function to retrieve metadata of datasets for one  
species. It looks the species names in the abstract.

.get\_meta <- function(species){

cn <- dataone::CNode("PROD")

search <- list(q = glue::glue("abstract:{species}"),

fl = "id,title,abstract",

sort = "dateUploaded+desc")

result <- dataone::query(cn, solrQuery = search,

as="data.frame")

if(nrow(result) == 0){

NULL

}else{

# otherwise one line by version

result <- unique(result)

tibble::tibble(species = species,

title = result$title,

abstract = result$abstract)

}

}

Note that DataONE searching could be more precise: one can choose to  
search from a given data source only for instance..

get\_meta <- ratelimitr::limit\_rate(.get\_meta,

rate = ratelimitr::rate(1, 2))

all\_meta <- purrr::map\_df(species, get\_meta)

nrow(all\_meta)

## [1] 266

length(unique(all\_meta$species))

## [1] 35

35 species are represented.

all\_meta <- unique(all\_meta[,c("title", "abstract")])

nrow(all\_meta)

## [1] 104

We then extract the most common words.

all\_meta %>%

dplyr::group\_by(title, abstract) %>%

dplyr::summarise(text = paste(title, abstract)) %>%

dplyr::ungroup() %>%

unnest\_tokens(word, text) %>%

dplyr::filter(!word %in% stopwords) %>%

dplyr::count(word, sort = TRUE) -> data\_words

head(data\_words, n = 10)

## # A tibble: 10 x 2

## word n

##

## 1 data 153

## 2 species 120

## 3 birds 94

## 4 breeding 87

## 5 feeding 75

## 6 population 65

## 7 bird 60

## 8 genetic 58

## 9 study 56

## 10 effects 54

Data is the most common word which is quite logical for metadata of  
actual datasets. Let’s also have a look at a regular wordcloud.

with(data\_words, wordcloud(word, n, max.words = 100))

wordcloud of titles and abstracts of scientific
metadata

As expected, the words seem more focused on ecology than when looking at  
scientific papers. DataONE is a gigantic data catalogue, where one could

* study the results of such queries (e.g. meta studies of number of,  
  say, versions by datasets)
* or find data to integrate to a new study. If you want to *download*  
  data from DataONE,

**Conclusion**

In this post, we used the rOpenSci fulltext package, and the DataONE  
dataone package, to search for bird species names in scientific papers  
and scientific open datasets. We were able to draw wordclouds  
representing the diversity of topics of studies in which the birds had  
been mentioned or studied. Such a search could be fun to do for your  
favourite bird(s)! And in general, following the same approach you could  
answer your own specific research question.

**Scientific literature access**

As a reminder, the pipeline to retrieve abstracts and titles of works  
mentioning a bird species was quite smooth:

species %>%

tolower() %>%

fulltext::ft\_search() %>%

fulltext::ft\_get() %>%

fulltext::ft\_collect() %>%

fulltext::ft\_chunks(c("title", "abstract")) %>%

fulltext::ft\_tabularize() %>%

dplyr::bind\_rows()

fulltext gives you a lot of power!