Possum magic, a classic for Aussie kids. Hush the possum and her Grandma Poss encounter different Australian animals and travel across well eat their way through the country. It is an adorable story with great illustrations! Reading it will make you feel like travelling to Australia, for instance to useR! 2018, except you shouldn’t because it is a very scary country:

However, you can travel and learn geography without leaving the comfort of a snake-free home… by mapping Hush’s adventures! Which is what I decided to do.

We shall prepare data for the maps in two main steps: first, looking where one can find the animals mentioned in the book; second, attributing a longitude and latitude to the cities Hush goes to.

**Locating the animals, snake included**

In the book, we first learn about how the magic of Grandma Poss affects some animals: wombats, kookaburras, dingoes and emus. We then read how Hush’s invisibility (sorry, spoiler) makes her interact with koalas, kangaroos and snakes. We can imagine Hush and her grandmother live near these animals, in what is called “deep in the Australian bush”. We’ll map where all these animals live so that we can guess where Hush can live for the story to be credible (I admit that the magic part kind of throws this plan out the window, oh well). Moreover, given what Wikipedia says about the bush, Hush and her grandmother could live anywhere in Australia as long as it’s not in a city, and since Hush’s first city visit is in Adelaide, we can suppose she did not live too far away from there. Let’s see!

In order to map animals, we shall get occurrences from 2011 to 2016 via the rOpenSci spocc package, with GBIF as a data source, and using the scrubr package for cleaning, like in visualizing waxwings migration. When querying data via spocc one needs a bounding box, so I’ll first load an Australian map from the eechidna package,

Visualizing waxwings Migration

**Getting the occurrence data**

As mentioned above I got the data via spocc. I read the README of the Github repo and learnt that it’s called spocc like *sp*ecies *occ*urrence data. So now I should never forget how many “c” there are in the word *occurrence*. Now please send help for my remembering it has two “r”.

The spocc package interacts with so many data sources that I felt a bit overwhelmed. I guess ecologists are not often blessed with so much data. I even decided to get data for the two other species of waxwings, although it only worked for the Bohemian and Cedar waxwings in the end.

I decided to use only data from GBIF. You’ll find more information about GBIF for instance, and other data sources for species occurrences. Since I wanted to get data for different years and all 3 species of waxwings, which meant a lot of data so for getting it I sliced the Earth (I like how evil this sentence makes me sound, ah!). Note that I used overlapping slices because otherwise I didn’t get data at the limits between slices. I guess I could have made my slices slightly less overlapping though.

library("spocc")

library("dplyr")

library("purrr")

library("scrubr")

get\_slice <- function(longitude, year){

print(paste(year, longitude))

gbifopts <- list(limit = 200000,

year = year)

waxwings <- occ(query = "Bombycilla",

from = c('gbif'),

gbifopts = gbifopts,

geometry = c(longitude - 0.5, - 90,

longitude + 0.5, 90))

waxwings <- occ2df(waxwings)

}

longitudes <- seq( -180, 179, by = 0.5)

years <- rep(2011:2016, length(lo ngitudes))

longitudes <- rep(longitudes, 6)

waxwings <- map2(longitudes, years, get\_slice)

for (i in 1:length(waxwings)){

waxwings[[i]]$latitude <- as.numeric(waxwings[[i]]$latitude)

waxwings[[i]]$longitude <- as.numeric(waxwings[[i]]$longitude)

}

waxwings <- bind\_rows(waxwings)

waxwings <- unique(waxwings)

waxwings <- filter(waxwings, !is.na(name))

readr::write\_csv(waxwings, path = "uncleaned\_waxwings.csv")

**Cleaning the occurrence data**

Now because my MSc of ecology is far behind me (but still close to my heart!) I have no idea how to assess the quality of species occurrence data. And even if I did I would have been delighted by the discovery of another rOpenSci package, scrubr, whose aim is to clean species occurrence records. Because I had so much data, I cleaned each day separately, otherwise it was just too long and hard on my poor computer. Don’t start thinking scrubr is slow.

cleanup <- function(df){

print(df$date[1])

df <- df %>%

coord\_impossible() %>%

coord\_incomplete() %>%

coord\_unlikely()

if(nrow(df) > 1){

df <- dedup(df)

}

df <- df %>%

date\_standardize("%Y-%m-%d") %>%

date\_missing()

return(df)

}

waxwings <- readr::read\_csv("uncleaned\_waxwings.csv")

waxwings <- split(waxwings,

waxwings$date)

waxwings <- lapply(waxwings, cleanup)

waxwings <- bind\_rows(waxwings)

waxwings <- unique(waxwings)

waxwings <- dplyr::filter(waxwings, name != "Bombycilla japonica",

longitude < 50)

readr::write\_csv(waxwings, path = "waxwings.csv")

I removed the records with impossible, incomplete or unlikely coordinates (unlikely being e.g. a political centroid, impossible coordinates with a too high longitude), I also removed duplicate records and records without a data. This was so easy! I’d like the scrubr package to come clean my flat, as a team with the janitor package. Now, in real life, I’d probably be even stricter with data cleaning but for the scope of this blog post, using that package without any additionnal check was enough.

I also removed occurrences of the Japanese waxwing because they were too few of them, and occurrences with a longitude higher than 50 because it seemed weird to have non Japanese waxwings in Asia. After all this glorious data cleaning, I had 1015603 records.

**Exploring the data**

Let the fun begin!

**Number of occurrences by species over time**

library("ggplot2")

library("hrbrthemes")

library("dplyr")

library("lubridate")

waxwings <- mutate(waxwings,

week = update(date, wday = 1))

waxwings %>%

group\_by(week, name) %>%

summarize(n = n()) %>%

ggplot() +

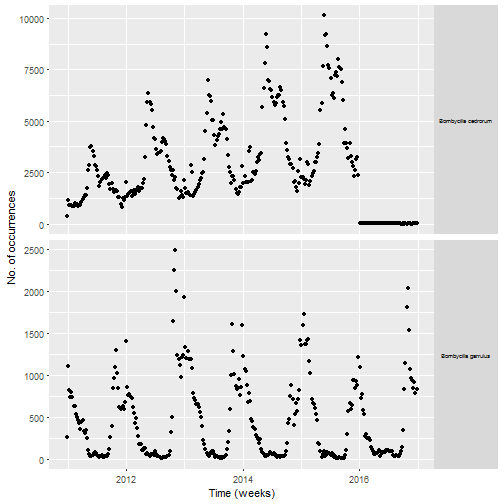
geom\_point(aes(week, n)) +

facet\_grid(name ~ ., scales = "free\_y") +

theme(strip.text.y = element\_text(angle = 0, size = 6)) +

xlab("Time (weeks)") +

ylab("No. of occurrences")



I have no idea why I have no data in 2016 for one of the two species. I decided to not investigate it further since I had enough birds for my primary goal which was mapping migration. The number of occurrences of Cedar waxwing increases over time before 2016, maybe because of more birders reporting sightings? For both species there is a clear seasonality, probably because these birds tend to breed in places where less people can observe them as we’ll see later in the post.

**Day-of-the-week effects**

waxwings <- mutate(waxwings,

wday = lubridate::wday(date, label = **TRUE**))

waxwings %>%

group\_by(update(date, wday = 1), wday) %>%

summarize(n = n()) %>%

ggplot() +

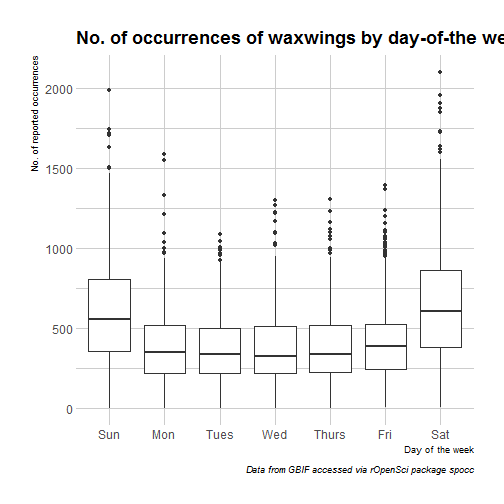
geom\_boxplot(aes(wday, n))+

labs(x="Day of the week", y="No. of reported occurrences",

title="No. of occurrences of waxwings by day-of-the week",

caption="Data from GBIF accessed via rOpenSci package spocc") +

theme\_ipsum()



So, more birds are reported on week-ends than on weekdays which I assume is due to a difference in human rather than bird behaviour (but who knows?). Note that for finer characterization of days where more people are birding, I could have used the bizdays package, but then I’d have limited my observations for one country only, because mixing holidays from different countries doesn’t sound like fun. Another thing that might influence sightings, beside people having the day off, might be the weather.

**Mapping the migrations!**

I first decided to plot the occurrences themselves on maps, by month, and to make a gif out of it. I then had to choose the colour and shape of the points used to represent the birds. Shape? Bird emojis of course! Regarding the colour, I was quite glad when someone For the Cedar waxwing I had to create a palette myself, which I’d never thought of doing if I hadn’t seen the birdcolourbot account, I’m not a colour artist and often stick to viridis. A bit of copy-paste work but much easier than having to mix paint colours for real.

bohemian\_palette <- c("#1A1A1A", "#878787",

"#B15929", "#E21A1C",

"#FEFF99")

cedar\_palette <- c("#050608", "#5D5C7A",

"#AF5F2A", "#8E2F49",

"#F3DD31")

To me both species look quite similar so I didn’t expect the palettes to be reallt different. I decided the colours would all have the same probability, instead of weighing them according them to their presence in the usual patterns of each species.

set.seed(1)

library("dplyr")

waxwings <- mutate(waxwings,

colour = factor(sample(1:5, size = nrow(waxwings), replace = **TRUE**)))

waxwings <- mutate(waxwings,

month = lubridate::month(date),

month\_name = lubridate::month(date, label = **TRUE**, abbr = **FALSE**))

bohemian <- filter(waxwings,

name == "Bombycilla garrulus")

cedar <- filter(waxwings,

name == "Bombycilla cedrorum")

After this preparation of the two data.frames I created a function for plotting a month of data for one species. A point I’d like to work on in the future for not being so ashamed of each of my maps are projections.

library("ggalt")

library("magick")

library("ggmap")

library("ggthemes")

library("emojifont")

load.emojifont('OpenSansEmoji.ttf')

wax\_map <- map\_data("world")

wax\_map <- wax\_map[wax\_map$region != "Antarctica",]

plot\_month\_species <- function(df, species,

name, palette){

p <- ggplot()

p <- p + geom\_map(data = wax\_map,

map = wax\_map,

aes(x = long, y = lat, map\_id = region),

color = "white", fill = "#7f7f7f",

size = 0.05, alpha = 1/4)

p <- p + theme\_map()

p <- p + geom\_text(aes(longitude, latitude,

col = colour),

label = emoji("bird"),

data = df,

family="OpenSansEmoji",

size = 5)

p <- p + scale\_colour\_manual(values = palette)

p <- p + ylim(min(species$latitude),

max(species$latitude))

p <- p + xlim(min(species$longitude),

max(species$longitude))

p <- p + theme(legend.position = "none")

outfil <- paste0("fig\_waxwings/", name, "\_", df$month[1], ".png")

ggsave(outfil, p, width=5, height=5)

image\_read(outfil) %>%

image\_annotate(text = as.character(df$month\_name[1]),

size = 100) %>%

image\_write(outfil)

outfil

}

I chose to annotate the graph after creating it in order not to have to use the emoji font for the month name.

I first applied the function to the Bohemian waxwings.

bohemian\_l <- split(bohemian, bohemian$month)

bohemian\_l %>%

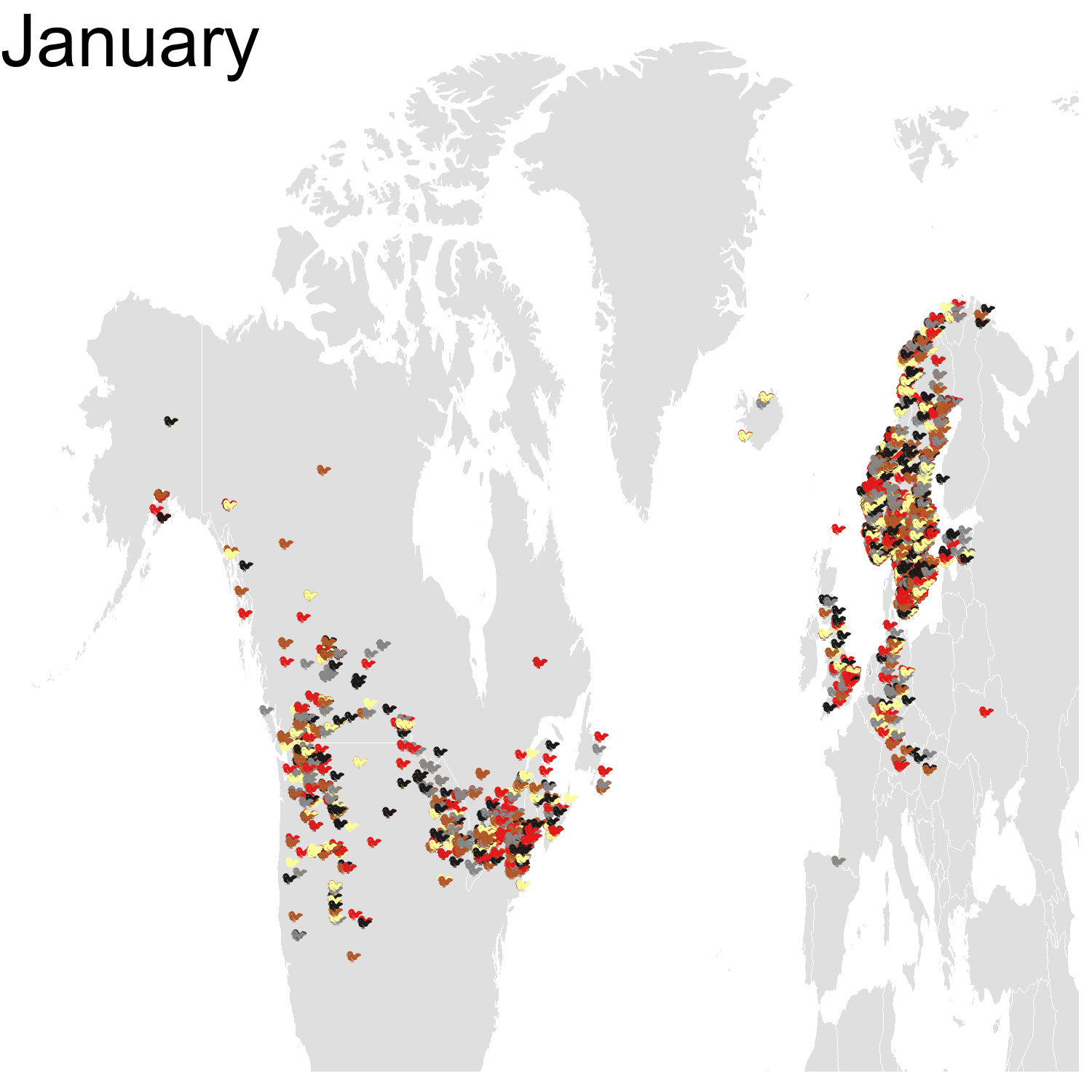
purrr::map(plot\_month\_species, species = bohemian, name = "bohemian", palette = bohemian\_palette) %>%

purrr::map(image\_read) %>%

image\_join() %>%

image\_animate(fps=1) %>%

image\_write("bohemian.gif")



First observation: I think the colours are pretty, but I also think I’ve just invented the concept of a confetti map and I’m not sure I’m proud of that. Then, regarding the underlying bird movements, Bohemian waxwings breed in the spring in regions farther up North than the regions where they winter and this can be seen on the map. There seem to be less sightings in the breeding season, probably because less people live up there, and apparently Santa’s helpers don’t even help.

Then it was the turn of the Cedar waxwing for which I had many more observations (871310 vs. 144293). Brace yourself for many confetti!

cedar\_l <- split(cedar, cedar$month)

cedar\_l %>%

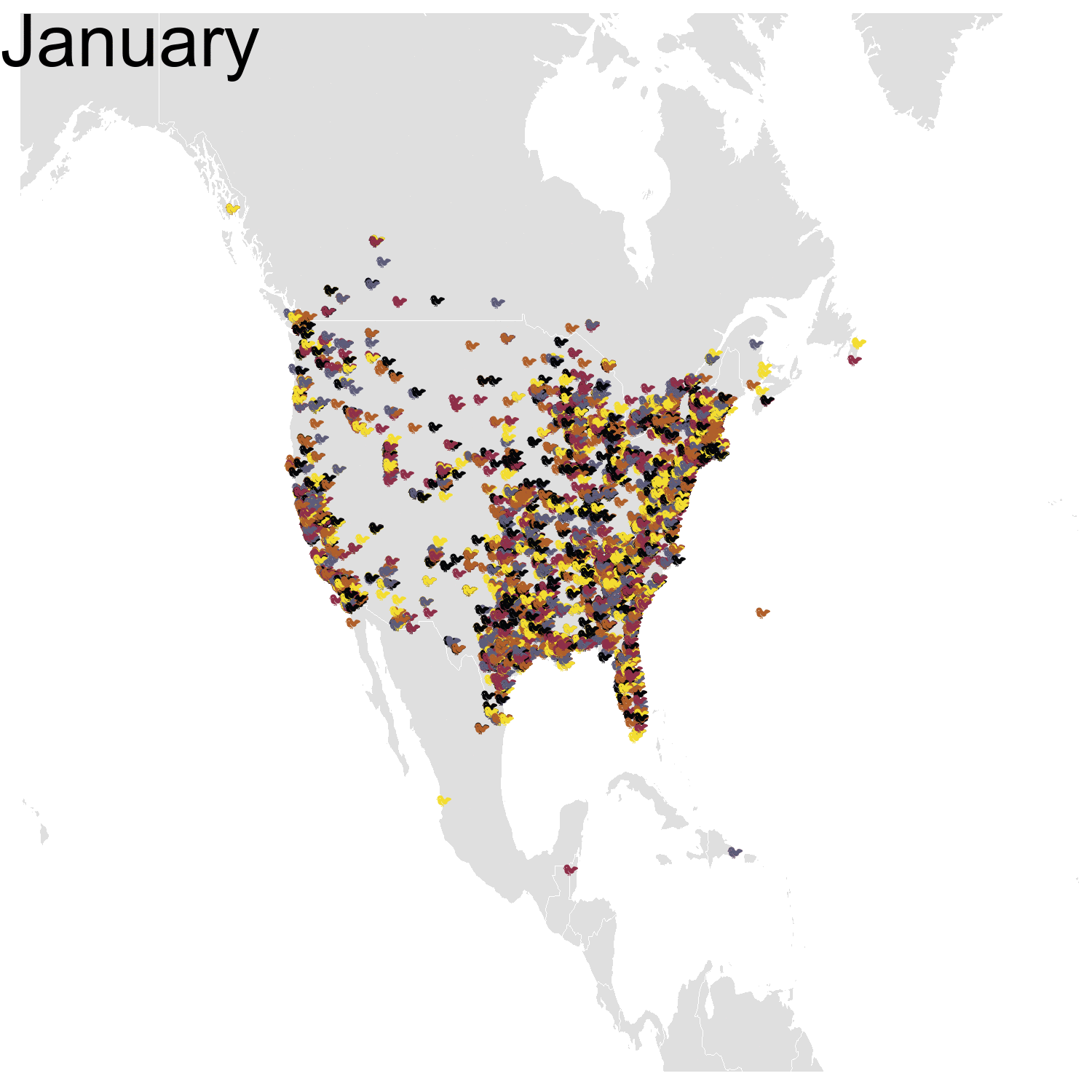
purrr::map(plot\_month\_species, species = cedar, name = "cedar", palette = cedar\_palette) %>%

purrr::map(image\_read) %>%

image\_join() %>%

image\_animate(fps=1) %>%

image\_write("cedar.gif")



I’ve had a lot of fun making my confetti gifs. I could extend the analysis by using spatial smoothing to draw the zone where waxwings are mostly seen each month. I think the general pattern is clear enough now, but there are some outliers like one Bohemian waxwing in Spain. If there really is a Bohemian waxwing in Spain I’d appreciate its visiting me because I’ve never seen a waxwing!

**Concluding**

I was impressed by rOpenSci tools for getting and cleaning occurrence data, and quite thankful for the species occurrence data provided by GBIF. I liked plotting migration thanks to species occurrence, although I guess the best method for knowing migration patterns is tracking birds. But then I’ll let professionals do this and keep my bird migration confetti map making as a nice hobby.

library("eechidna")

library("ggthemes")

library("ggplot2")

data(nat\_map\_2016)

Let’s just draw an empty map.

ozmap <- ggplot() +

geom\_map(aes(map\_id=id), data=nat\_map\_2016,

map=nat\_map\_2016) +

expand\_limits(x=nat\_map$long, y=nat\_map$lat) +

theme\_map()

ozmap



And now, let’s prepare the bounding box!

bbox <- c(min(nat\_map$long), min(nat\_map$lat),

max(nat\_map$long), max(nat\_map$lat))

I’ll make some assumptions about species and families since the book is not completely clear regarding which animals are meant exactly. Before pondering about this I had no idea there were three species of wombats! For wombats, I’m using the two possible families, for kookaburras, emus and kangaroos the whole family and for snake. For possums, I’m using a family of possums whose Wikipedia page did not indicate they were mostly urban.

animals <- tibble::tibble(name = c("wombat", "wombat", "kookaburra",

"dingo", "emu",

"koala", "kangaroo",

"snake", "possum"),

latin = c("Lasiorhinus", "Vombatus", "Dacelo",

"Canis lupus dingo", "Dromaius",

"Phascolarctos cinereus", "Macropus",

"Morelia spilota", "Phalangeridae"))

knitr::kable(animals)

| **name** | **latin** |
| --- | --- |
| wombat | Lasiorhinus |
| wombat | Vombatus |
| kookaburra | Dacelo |
| dingo | Canis lupus dingo |
| emu | Dromaius |
| koala | Phascolarctos cinereus |
| kangaroo | Macropus |
| snake | Morelia spilota |
| possum | Phalangeridae |

Here is the function to get occurrences for one species or family.

library("spocc")

library("scrubr")

library("magrittr")

get\_ozccurrences <- function(latin, year, bbox){

gbifopts <- list(year = year)

ozccurrences <- occ(query = latin,

from = c('gbif'),

gbifopts = gbifopts,

limit = 200000,

geometry = bbox)

ozccurrences <- occ2df(ozccurrences)

ozccurrences <- ozccurrences %>%

coord\_impossible() %>%

coord\_incomplete() %>%

coord\_unlikely() %>%

date\_standardize("%Y-%m-%d") %>%

date\_missing()

ozccurrences$latin <- latin

ozccurrences

}

Let’s test it on kookaburras in 2016, using the whole family. Note, the maps drawn in this section are really meant to just have a look. In a better analysis I’d try to define zones based on the occurrences for instance, and I’d be more perfectionist regarding the looks of the map.

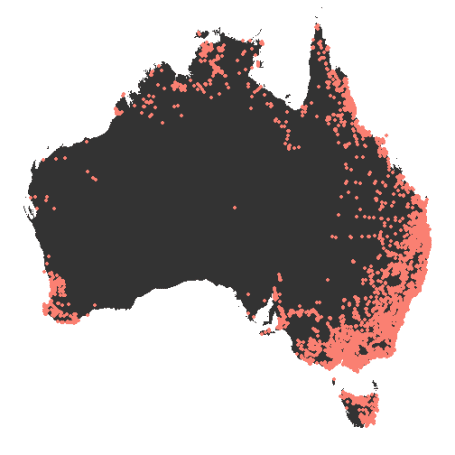
kookaburras <- get\_ozccurrences("Dacelo", bbox = bbox, year = 2016)

ozmap +

geom\_point(data = kookaburras,

aes(longitude, latitude),

col = "salmon")



Note that occurrences might be misleading, since to get one you need one observer, so we’ll probably get less occurrences when there are less humans without this meaning the density of the animal itself is lower. But since Hush’s story is told by a human, we expect Hush to live in an area with at least a few observations of the different animals! Otherwise, how could the author know about all this stuff?

all\_ozcurrences <- purrr::map2\_df(rep(animals$latin, 6),

rep(2011:2016, nrow(animals)),

get\_ozccurrences,

bbox = bbox)

all\_ozcurrences <- dplyr::rename(all\_ozcurrences, scientific\_name = name)

all\_ozcurrences <- dplyr::left\_join(all\_ozcurrences, animals, by = "latin")

readr::write\_csv(all\_ozcurrences, path = "data/2017-12-15-all\_ozcurrences.csv")

Now that we have a reasonable number of occurrences, we can check where the animals have been seen, hopefully all of them have at least one place in common, preferably not too far from Adelaide (whose location we’ll show in the next section). Although colours are not needed here, I’ll get a bit artistic and use the pretty “healthy reef” palette part of the ochRe package created during the rOpenSci Oz unconf this year. This palette has 9 colours, so it was handy.

library("ochRe")

ozmap +

geom\_point(data = all\_ozcurrences,

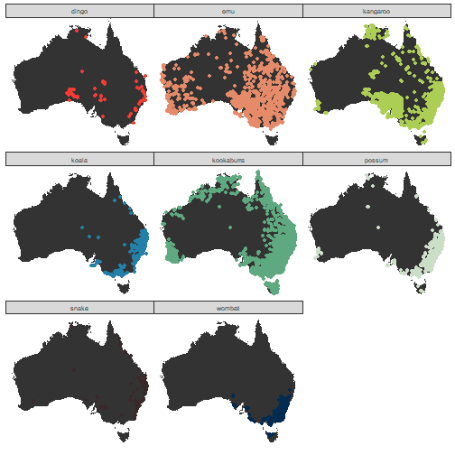
aes(longitude, latitude,

col = name)) +

facet\_wrap(~name)+

scale\_colour\_ochre(palette = "healthy\_reef") +

theme(legend.position = "none")



Ok, when you see where Adelaide is you’ll agree that it’s fine to assume Hush and Grandma Poss live in the bush a bit further North from Adelaide.

I also notice I only get no occurrence from the Northern hairy-nosed wombat but it seems to be rarer than the common wombat (ok, the name might have been a good clue for me before I got the data).

**Geocoding the cities**

The cities we need to geocode, and the food eaten by Hush and Grandma Poss while visiting them, are:

cities <- tibble::tibble(city = c("Adelaide", "Melbourne",

"Sydney", "Brisbane",

"Darwin", "Perth",

"Hobart"),

food = c("Anzac biscuit", "mornay and Minties",

"steak and salad", "pumpkin scones",

"vegemite", "pavlova",

"lamington"))

knitr::kable(cities)

| **city** | **food** |
| --- | --- |
| Adelaide | Anzac biscuit |
| Melbourne | mornay and Minties |
| Sydney | steak and salad |
| Brisbane | pumpkin scones |
| Darwin | vegemite |
| Perth | pavlova |
| Hobart | lamington |

We shall perfom geocoding using my own package opencage which is a wrapper for R of the OpenCage API.

library("opencage")

get\_lozcation <- function(city){

opencage\_results <- opencage::opencage\_forward(placename = city,

countrycode = "AU")

output <- opencage\_results$results

output$city <- city

output <- output[!is.na(output$components.city),]

output <- output[output$components.city == city,]

if(nrow(output) > 1){

output <- output[output$confidence == min(output$confidence),]

}

output

}

A small technical note, depending on the font you use in RMarkdown and that sort of encoding joy sources, you now get a flag from the OpenCage API! In my blogging font, it is a bit boring, e.g.:

adelaide <- get\_lozcation(city = "Adelaide")

adelaide$annotations.flag gives me . But still, a cool feature… among other tons of more useful features OpenCage has like the abbr parameter to get shorter names.

Now let’s geocode all cities!

geozcoded\_cities <- purrr::map\_df(cities$city, get\_lozcation)

geozcoded\_cities <- dplyr::left\_join(geozcoded\_cities, cities, by = "city")

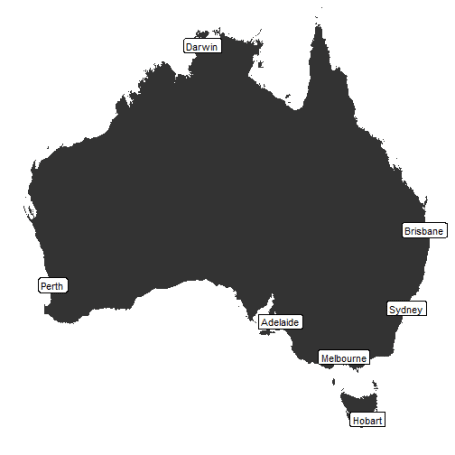
Cool, now we can plot them!

ozmap +

geom\_label(data = geozcoded\_cities,

aes(geometry.lng, geometry.lat,

label = city))



**Mapping the book!**

In this last section, I’ll invent a home address for Hush and Grandma Poss and produce an educative map containing both city names and city food specialties! Having 7 cities and one bush location, I’ll use an 8-colour ochRe palette, “namatjira\_qual” derived from the beautiful watercolour painting “Twin Ghosts”, by Aboriginal artist Albert Namatjira. Here again, no information is conveyed by the color.

library("ggrepel")

hush\_home <- tibble::tibble(geometry.lng = adelaide$geometry.lng,

geometry.lat = adelaide$geometry.lat + 0.01,

city = "the bush",

food = "Possum food")

hush\_places <- dplyr::bind\_rows(geozcoded\_cities, hush\_home)

ozmap +

geom\_point(data = hush\_places,

aes(geometry.lng, geometry.lat),

col = "grey20") +

geom\_label\_repel(data = hush\_places,

aes(geometry.lng + 0.3, geometry.lat + 0.3,

label = paste(food, "in", city),

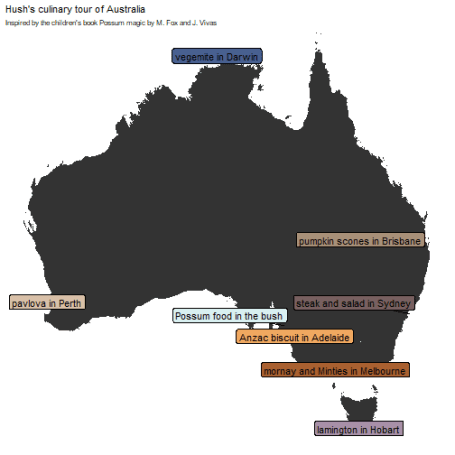
fill = city)) +

scale\_fill\_ochre(palette = "namatjira\_qual") +

theme(legend.position = "none") +

ggtitle("Hush's culinary tour of Australia",

subtitle = "Inspired by the children's book Possum magic by M. Fox and J. Vivas")



Now, I’ll show this map to our baby as soon as he’s old enough to get it, and then we’ll plan a culinary tour of Australia together! Probably not on a bike like Hush and Grandma Poss, unless we can whip up some magic!