eBird Data Processing

This document contains code to pre-process the eBird data.

Set-up chunk.

library(ggplot2)  
library(ggmap)

## Google Maps API Terms of Service: http://developers.google.com/maps/terms.

## Please cite ggmap if you use it: see citation("ggmap") for details.

library(knitr)  
library(sp)  
library(raster)  
library(ggcorrplot)  
knitr::opts\_knit$set(root.dir = "C:/Users/Ladd Irvine/Documents/Ladd Work/PhDStuff/Classes/Fall2018/SpeciesDistributionModeling/GroupProject-Lab1")

START HERE Load the eBird data subset.

load("C:/Users/Ladd Irvine/Documents/Ladd Work/PhDStuff/Classes/Fall2018/SpeciesDistributionModeling/GitStuff/FW599/janDuckDataNAmer.RData") # duckData

# Decision: PRIMARY\_CHECKLIST\_FLAG

See section 4.2.1 of the eBird documentation (pdf on Canvas).

table(duckData$PRIMARY\_CHECKLIST\_FLAG)

##   
## ? 1   
## 48953 261545

This takes the value 1 for unique checklists or for the primary checklist of a group count. Limit to PRIMARY\_CHECKLIST\_FLAG==1.

duckData\_v2 = duckData[which(duckData$PRIMARY\_CHECKLIST\_FLAG==1),]  
dim(duckData)

## [1] 310498 29

dim(duckData\_v2)

## [1] 261545 29

# Decision: COUNT\_TYPE

Check out Table 2 in the eBird documentation.

table(duckData\_v2$COUNT\_TYPE)

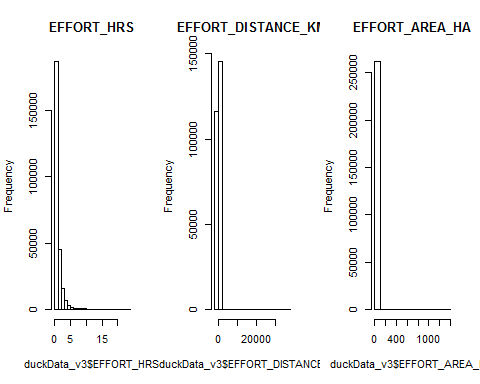
##   
## P21 P22 P23 P39 P41 P46 P47 P48 P50 P58   
## 113492 145167 1902 0 0 87 58 70 0 8   
## P60 P61 P62 P64 P65 P66 P69   
## 25 113 618 5 0 0 0

\*\* Are there any count types we want to categorically exclude? \*\* If so, create v3 of the data here:

duckData\_v3=duckData\_v2

# Decision: EFFORT variables

par(mfcol=c(1,3))  
hist(duckData\_v3$EFFORT\_HRS, main="EFFORT\_HRS") # applies to stationary counts  
hist(duckData\_v3$EFFORT\_DISTANCE\_KM, main="EFFORT\_DISTANCE\_KM") # applies to traveling counts  
hist(duckData\_v3$EFFORT\_AREA\_HA, main="EFFORT\_AREA\_HA") # applies to areal counts



par(mfcol=c(1,1))

Looks like there are some extreme values for distance and area! Closer look at the top 100 values:

sort(duckData\_v3$EFFORT\_DISTANCE\_KM,decreasing = TRUE)[1:100]

## [1] 37013.900 804.672 804.650 804.650 708.092 688.780 615.000  
## [8] 550.000 437.891 402.336 402.325 402.325 400.000 399.989  
## [15] 386.232 350.000 344.400 333.000 328.306 325.000 321.869  
## [22] 321.869 321.869 321.860 321.860 320.000 300.000 289.682  
## [29] 289.674 273.588 265.542 259.000 258.000 256.000 251.058  
## [36] 250.000 250.000 241.402 241.402 241.395 241.395 236.574  
## [43] 235.000 235.000 225.308 225.308 225.308 225.308 213.554  
## [50] 209.209 209.209 205.996 201.163 200.000 200.000 200.000  
## [57] 199.995 199.995 199.995 199.995 199.995 199.995 199.995  
## [64] 195.535 193.121 193.121 193.121 193.121 193.121 193.116  
## [71] 188.293 181.856 180.247 180.000 179.995 179.995 177.028  
## [78] 177.028 175.414 175.414 175.000 172.200 170.590 170.590  
## [85] 168.977 168.977 160.934 160.934 160.934 160.934 160.934  
## [92] 160.934 160.934 160.934 160.934 160.934 160.934 160.934  
## [99] 160.934 160.934

These values seem huge. For comparison, the WORLDCLIM data we’re working with has a spatial resolution of about 340 km^2 (we can get higher resolution if desired). This corresponds to a circle with a radius of about 10km. One could imaging limiting traveling counts to at most 10km. How many is this?

sum(duckData\_v3$EFFORT\_DISTANCE\_KM<=10) # checklists covering at most 10km

## [1] 252202

dim(duckData\_v3)[1] # total number of checklists

## [1] 261545

Not a huge reduction. Let’s go for it.

duckData\_v4 = duckData\_v3[which(duckData\_v3$EFFORT\_DISTANCE\_KM<=10),]

A similar process for area counts would make sense.

sort(duckData\_v4$EFFORT\_AREA\_HA,decreasing = TRUE)[1:100]

## [1] 1376.3359 1294.9940 1239.5521 1214.0569 1214.0569 1052.1827 687.0000  
## [8] 647.4970 643.4502 607.0284 600.0000 582.0000 526.0914 526.0913  
## [15] 520.0000 520.0000 517.9976 485.6228 485.6228 485.6228 485.6228  
## [22] 485.6228 461.3416 461.3416 445.1542 445.1542 430.0000 430.0000  
## [29] 404.6856 364.2171 360.0000 323.7485 313.0000 270.0002 258.9988  
## [36] 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988  
## [43] 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988  
## [50] 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988  
## [57] 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988  
## [64] 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988  
## [71] 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988 258.9988  
## [78] 242.8114 230.0000 202.3428 202.3428 202.3428 202.3428 202.3428  
## [85] 202.3428 200.0000 200.0000 192.0000 192.0000 184.0000 182.1085  
## [92] 182.1085 163.8977 161.8743 161.8743 161.8743 161.8743 141.6400  
## [99] 125.0479 121.4057

sum(duckData\_v4$EFFORT\_AREA\_HA<=10) # checklists covering at most 10km

## [1] 251577

dim(duckData\_v4)[1] # total number of checklists

## [1] 252202

duckData\_v5 = duckData\_v4[which(duckData\_v4$EFFORT\_AREA\_HA<=10),]  
dim(duckData\_v5)

## [1] 251577 29

\*\* Should we limit the amount of hours spent as well? \*\* If so, create v6 here.

# check what the longest effort records are

sort(duckData\_v5$EFFORT\_HRS,decreasing = TRUE)[1:100]#

## [1] 24.000 24.000 24.000 24.000 24.000 24.000 24.000 24.000 24.000 24.000  
## [11] 24.000 24.000 24.000 24.000 23.000 23.000 22.367 21.000 21.000 21.000  
## [21] 21.000 21.000 21.000 20.333 20.000 20.000 20.000 20.000 20.000 20.000  
## [31] 20.000 20.000 20.000 20.000 20.000 20.000 20.000 20.000 20.000 20.000  
## [41] 20.000 20.000 20.000 20.000 20.000 20.000 20.000 20.000 20.000 19.167  
## [51] 19.000 18.833 18.583 18.500 18.333 18.000 18.000 17.750 17.000 16.833  
## [61] 16.650 16.500 16.200 16.167 16.000 16.000 15.750 15.667 15.667 15.667  
## [71] 15.500 15.500 15.500 15.500 15.483 15.400 15.250 15.250 15.167 15.000  
## [81] 15.000 15.000 15.000 15.000 15.000 15.000 15.000 15.000 15.000 15.000  
## [91] 15.000 15.000 15.000 15.000 15.000 15.000 15.000 15.000 15.000 15.000

## Yes we should limit effort hours also. There are some records with 24h of effort which seems impossible for one person and unrealistic for a team effort. Additionally, that would include observations made at night which I’m not even sure how that would work. I guess you could do audio sampling but I think ducks are mostly sleeping at night?

## It would make the most sense to limit the effort hours to periods that could be completed during daylight hours. Given the latitudinal range of the data that can be a pretty long or short period of time but probably best to go with the longest daylight period at that time of year and assume most records in places with shorter windows of daylight will self-correct as most people will likely just be reporting daylight surveys.The lowest-latitude record is around 12.5 degrees lat (min(duckData\_v5$LATITUDE))so daylight would be about 12 hrs counting sunrise and sunset so lets use that for now…

# See how many records are left when you trim the effort hours to 12 or less

sum(duckData\_v5$EFFORT\_HRS<=12) # checklists covering 12 hrs or less

## [1] 251388

dim(duckData\_v5)[1] # total number of checklists

## [1] 251577

# doesn’t reduce the number that much so lets pull them out.

duckData\_v6 = duckData\_v5[which(duckData\_v5$EFFORT\_AREA\_HA<=12),]

# Map the new version

map = get\_map(location=c(min(duckData\_v6$LONGITUDE)-1,  
 min(duckData\_v6$LATITUDE)-1,  
 max(duckData\_v6$LONGITUDE)+1,  
 max(duckData\_v6$LATITUDE)+1),  
 maptype="satellite")

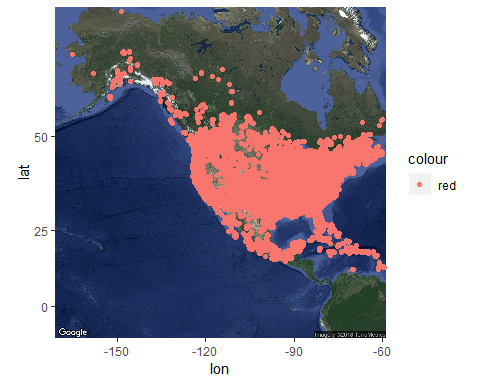
## Warning: bounding box given to google - spatial extent only approximate.

## converting bounding box to center/zoom specification. (experimental)

## Source : https://maps.googleapis.com/maps/api/staticmap?center=41.351921,-115.00167&zoom=3&size=640x640&scale=2&maptype=satellite&language=en-EN

p = ggmap(map)   
p + geom\_point(aes(x=LONGITUDE,y=LATITUDE,col="red"),data=duckData\_v6)

## Warning: Removed 583 rows containing missing values (geom\_point).



Zoom in on Oregon.

map = get\_map(location=c(min(duckData\_v6$LONGITUDE[duckData\_v6$STATE\_PROVINCE=="Oregon"])-1, min(duckData\_v6$LATITUDE[duckData\_v6$STATE\_PROVINCE=="Oregon"])-1, max(duckData\_v6$LONGITUDE[duckData\_v6$STATE\_PROVINCE=="Oregon"])+1, max(duckData\_v6$LATITUDE[duckData\_v6$STATE\_PROVINCE=="Oregon"])+1),  
 maptype="satellite")

## Warning: bounding box given to google - spatial extent only approximate.

## converting bounding box to center/zoom specification. (experimental)

## Source : https://maps.googleapis.com/maps/api/staticmap?center=44.124872,-120.736675&zoom=7&size=640x640&scale=2&maptype=satellite&language=en-EN

p2 = ggmap(map)   
p2 + geom\_point(aes(x=LONGITUDE,y=LATITUDE,col="red"),data=duckData\_v6)

## Warning: Removed 243095 rows containing missing values (geom\_point).

