# Final Project

*Jonathan Wasden*

*4/24/2019*

## Introduction

The goal of this project was to use data, given by Washington state, on local watersheds and salmon runs to understand use of the ggmaps package in R. Accompanying objectives for this project are:

- Data acquisition.
- Greater comprehension of data.
- Gaining more experience in cleaning and arranging data
- Better understanding of model creation.

## Data Collecting

Searching for this data was actually quite simple. I was able to acquire this data through the Washington State Department of Ecology. However, upon attempting to find the link for the data request page, I was disappointed to find the link I used was taken down. As of now I am unaware where that request page is now located.

## Data setup

The data set I obtained was in an excel format, which was thenonverted to a .csv file through excel's export function. From there very little cleanup was necessary to effectively use the data.

setup required a few packages and an addition to the data set to change any blank spaces with NAs.

```
library(plyr)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()   masks plyr::arrange()
## x purrr::compact()   masks plyr::compact()
## x dplyr::count()     masks plyr::count()
## x dplyr::failwith()  masks plyr::failwith()
## x dplyr::filter()    masks stats::filter()
## x dplyr::id()        masks plyr::id()
## x dplyr::lag()       masks stats::lag()
## x dplyr::mutate()    masks plyr::mutate()
## x dplyr::rename()    masks plyr::rename()
## x dplyr::summarise() masks plyr::summarise()
## x dplyr::summarize() masks plyr::summarize()
```

```
library(ggmap)
```

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(stringi)
library(ggplot2)
library(dplyr)

df1 = read.csv("Final_Project_files/Wasden.csv", na.strings = c("","NA"))
```

## Initial Cleanup

Some rows needed to be removed because that data was placed in completely different columns.

Other sections for cleanup required very little time, such as removing NAs, filtering out particularlly big outliers, and difficult to identify streams.

IT SHOULD BE NOTED: I made a mistake when doing cleanup during this step. The na.omit() command removes all NAs within the data set and their accompanying row. This ended up drastically reducing the entries in the data set. However, upon looking back during my coordinate search it was a bit of a relief to only locate 300 individual streams rather than over 700, which would have taken me much longer.

```
df = df1[c(2:4,7,11:15)]
df = na.omit(df)

dfINDX = df %>%
  filter(Species == "INDX")

dfFOOT = df %>%
  filter(Species == "FOOT")

dfSPOT = df %>%
  filter(Species == "SPOT")

dfSUPP = df %>%
  filter(Species == "SUPP")

dfREMOVE.1 = join(dfINDX, dfFOOT)
```

```
## Joining by: WRIA_Num, StreamName, StreamCatalogCode, Flow, Species, RunYear, Live_Count, Dead_Count,
```

```
dfREMOVE.2 = join(dfREMOVE.1, dfSPOT)
```

```
## Joining by: WRIA_Num, StreamName, StreamCatalogCode, Flow, Species, RunYear, Live_Count, Dead_Count,
```

```
dfREMOVE = join(dfREMOVE.2, dfSUPP)
```

```
## Joining by: WRIA_Num, StreamName, StreamCatalogCode, Flow, Species, RunYear, Live_Count, Dead_Count,
```

```
dfFINAL.1 = anti_join(df, dfINDX)
```

```
## Joining, by = c("WRIA_Num", "StreamName", "StreamCatalogCode", "Flow", "Species", "RunYear", "Live_C
```

```
dfFINAL.2 = anti_join(dfFINAL.1, dfFOOT)
```

```
## Joining, by = c("WRIA_Num", "StreamName", "StreamCatalogCode", "Flow", "Species", "RunYear", "Live_C
```

```
dfFINAL.3 = anti_join(dfFINAL.2, dfSPOT)
```

```
## Joining, by = c("WRIA_Num", "StreamName", "StreamCatalogCode", "Flow", "Species", "RunYear", "Live_C
```

```
dfFINAL = anti_join(dfFINAL.3, dfSUPP)
```

```
## Joining, by = c("WRIA_Num", "StreamName", "StreamCatalogCode", "Flow", "Species", "RunYear", "Live_C
```

```
dfCOORD = dfFINAL[-grep('^Unnamed',df$StreamName),]

dfCOORD.1 = dfCOORD %>%
  filter(PercentSeen > 100)

dfCOORD = anti_join(dfCOORD, dfCOORD.1)
```

```
## Joining, by = c("WRIA_Num", "StreamName", "StreamCatalogCode", "Flow", "Species", "RunYear", "Live_C
```

```
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "1", "Nooksack")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "2", "San Juan")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "3", "Lower Skagit-Samish")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "4", "Upper Skagit")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "5", "Stillaguamish")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "6", "Island")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "7", "Snohomish")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "8", "Cedar/Samish")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "9", "Duwamish/Green")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "10", "Puyallup/White")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "11", "Nisqually")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "12", "Chambers-Clover")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "13", "Deschutes")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "14", "Kennedy-Goldsborough")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "15", "Kitsap")
dfCOORD$WRIA_Num =  str_replace_all(dfCOORD$WRIA_Num, "16", "Skokomish-Dosewallips")
dfCOORD$WRIA_Num = str_replace_all(dfCOORD$WRIA_Num, "NooksackStillaguamish", "Stillaguamish")



dfCOORD$RunYear = as.numeric(as.character(dfCOORD$RunYear))
```

## Coordinate Search

This was, without a doubt, the most time intensive part of this project. For this I needed to manually search for the streams, rivers, lakes, and tributaries that these data points were located at.

IT SHOULD BE NOTED: A major obstacle with this step was finding the actual location. Upon looking at the data set you notice the watershed ID, RMupper, RMlower, and Stream Catalog Code.

All of these can be used when looking for the location the specific survey was taken. Although I did use the watershed ID to resolve disputes of where the locations were, I would be unable to go through any more of a detailed search given my time constraints. This will, however, be something I will attempt to do given the breadth of data I acquired.

```
dfFIRST = dfCOORD %>%
  filter(RunYear <= 2005)
```

```
dfSECOND = dfCOORD %>%
  filter(RunYear >=2006)

complete.cases(dfFIRST)

Streams<-as.data.frame(unique(dfFIRST$StreamName))

STREAMS <- as.data.frame(Streams[-c(3,5,8,12,16,17,20,26,66,78,79,81,83,91,94,98,101,106,109,
                      110,114,118,119,120,122,126,129,131,132,135,137,139,141,
                      142,149,152,155,156,159:165,167,168,170,171,173,176,178,
                      183,188,190,191,196,199,206:208,213,214,217,219,224,228,
                      230,232,235,238,244,246,249,253,256,258,261:263,304,305,
                      310,318,319,324,327,336,338,341,343,347,351,371,373:376,
                      379,380,382,383,385,393,399,400,402,406,409),])
STREAMS <- as.data.frame(STREAMS[-c(50),])
STREAMS <- as.data.frame(STREAMS[-c(24),])
colnames(STREAMS) <- c("StreamName")

Final.Coords = read.csv("Final_Project_files/Coords.csv")
Final.Coords = as.data.frame(Final.Coords[,c(1:3)])
Final.Coords = na.omit(Final.Coords)

Final.Template = right_join(dfCOORD,Final.Coords)
```

```
## Joining, by = "StreamName"
```

```
Final.Template$Live_Count = as.numeric(as.character(Final.Template$Live_Count))
Final.Template$Dead_Count = as.numeric(as.character(Final.Template$Dead_Count))

First.Final = Final.Template %>%
  filter(RunYear <= 2000)

Second.Final = Final.Template %>%
  filter(RunYear >2000 & RunYear <=2005)

Third.Final = Final.Template %>%
  filter(RunYear > 2005 & RunYear <= 2010)

Fourth.Final = Final.Template %>%
  filter(RunYear > 2010 & RunYear <= 2015)
```

## Playing with Models

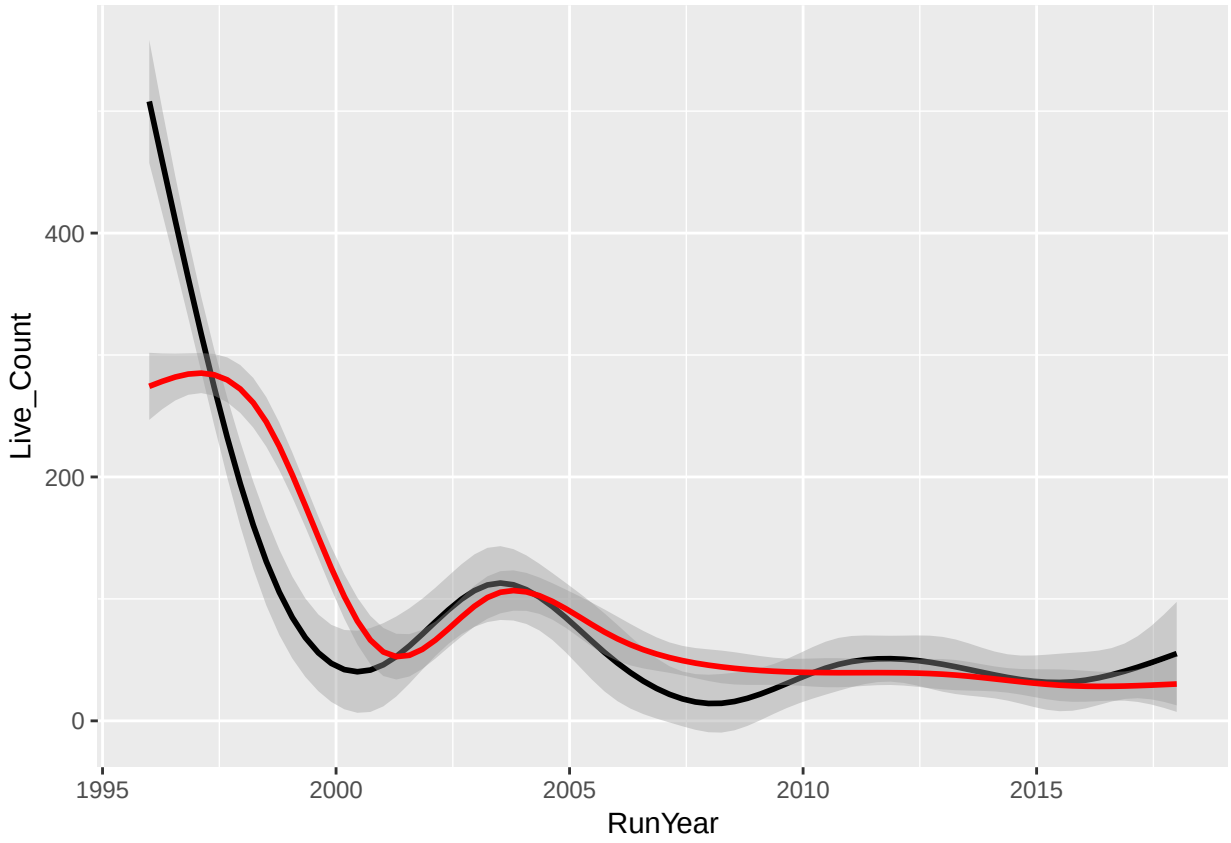
This step allowed me to play with models and see which variables could reflect the data the best. The primary variable I wanted to know about was the live count of salmon. What I found, by plotting different models and the data together by year, was that there are three variables that would be most reliable in predicting results in relation to live count. Those three are:Run Year, Species, and the flow of the area these surveys were taken.

```
library(modelr)

mod1.1 = lm(Live_Count ~ RunYear*Species, Final.Template)
mod1.2 = lm(Live_Count ~ RunYear*Species*Flow, Final.Template)
mod1.3 = lm(Live_Count ~ RunYear*Species*Dead_Count, Final.Template)
```

```
mod1.4 = lm(Live_Count ~ RunYear*Species*Flow*Dead_Count, Final.Template)


summary(mod1.1)


pred1 = add_predictions(Final.Template,mod1.2)
```

```
ggplot(pred1, aes(x=RunYear,color=Species)) +
  geom_smooth(aes(y=Live_Count),color="Black") +
  geom_smooth(aes(y=pred),color="Red")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 3 rows containing non-finite values (stat_smooth).
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## Warning: Removed 3 rows containing non-finite values (stat_smooth).
```




## Playing with ggpmap

Now comes the fun part. With this I was able to get help from a very particular source, https://www.littlemissdata.com/blog/maps. Along with knowing how to use ggplot as an introduction, I also, shamelessly, used their colour palette cause it has a very nice shade of blue.

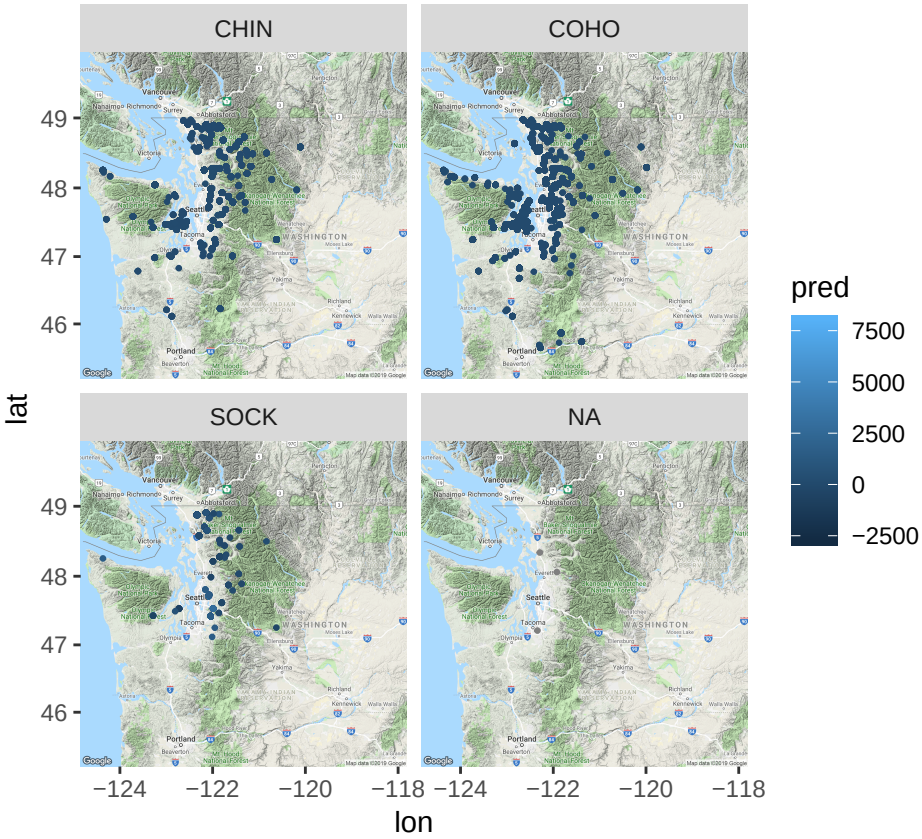
```
col1 = "#011f4b"
```

```
col2 = "#6497b1"

col3 = "#b3cde0"

col4 = "#CC0000"

colme = c(col1,col2,col3,col4)
ggmap::register_google(key = "AIzaSyDjTK7ZjYKGr1nE5PmaSncAg4g9L913C_o")
```

```
ggmap(get_googlemap(center = c(lon = -121.335167, lat = 47.608013),
                    zoom = 7, scale = 2,
                    maptype ='terrain')) +
  geom_point(aes(x = Longitude, y = Latitude,  colour = pred), data = pred1, size = 0.5) +
  theme(legend.position="right") +
  facet_wrap(facets = "Species")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.608013,-121.335167&zoom=7&size=640
```

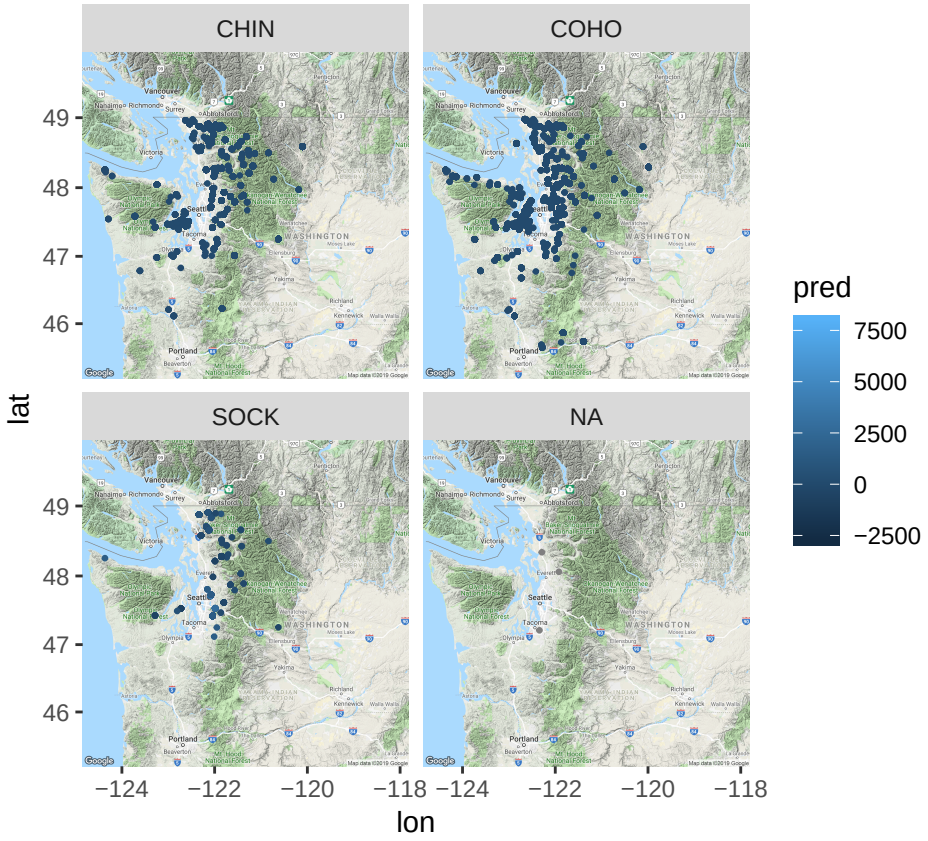
```
## Warning: Removed 11 rows containing missing values (geom_point).
```




```
ggmap(get_googlemap(center = c(lon = -121.335167, lat = 47.608013),
                    zoom = 7, scale = 2,
                    maptype ='terrain')) +
  geom_point(aes(x = Longitude, y = Latitude,  colour = pred), data = pred1, size = 0.5) +
  theme(legend.position="right") +
  facet_wrap(facets = "Species")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.608013,-121.335167&zoom=7&size=640
```

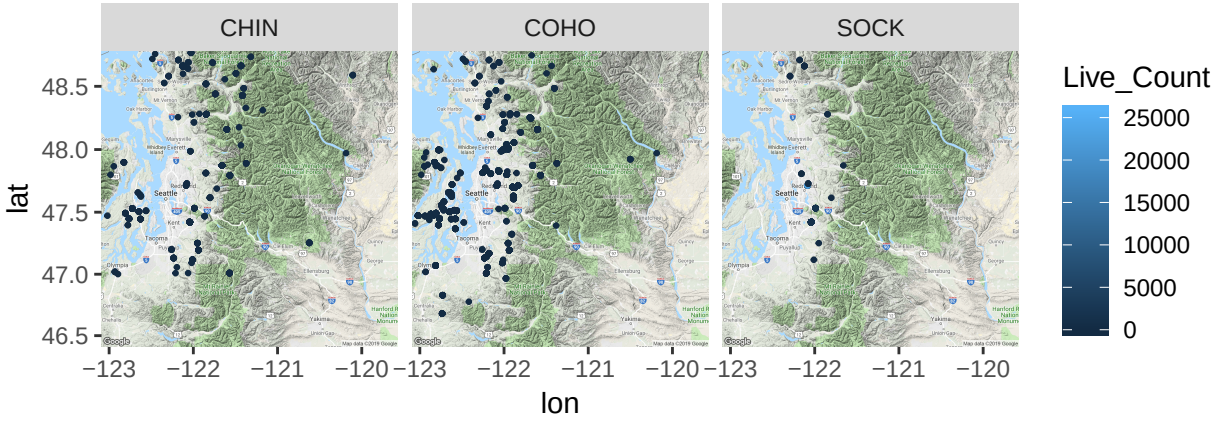
```
## Warning: Removed 11 rows containing missing values (geom_point).
```




```
ggmap(get_googlemap(center = c(lon = -121.335167, lat = 47.608013),
                    zoom = 8, scale = 2,
                    maptype ='terrain')) +
  geom_point(aes(x = Longitude, y = Latitude,  colour = Live_Count), data = First.Final, size = 0.5) +
  theme(legend.position="right") +
  facet_wrap(facets = "Species")
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.608013,-121.335167&zoom=8&size=640
```

```
## Warning: Removed 333 rows containing missing values (geom_point).
```

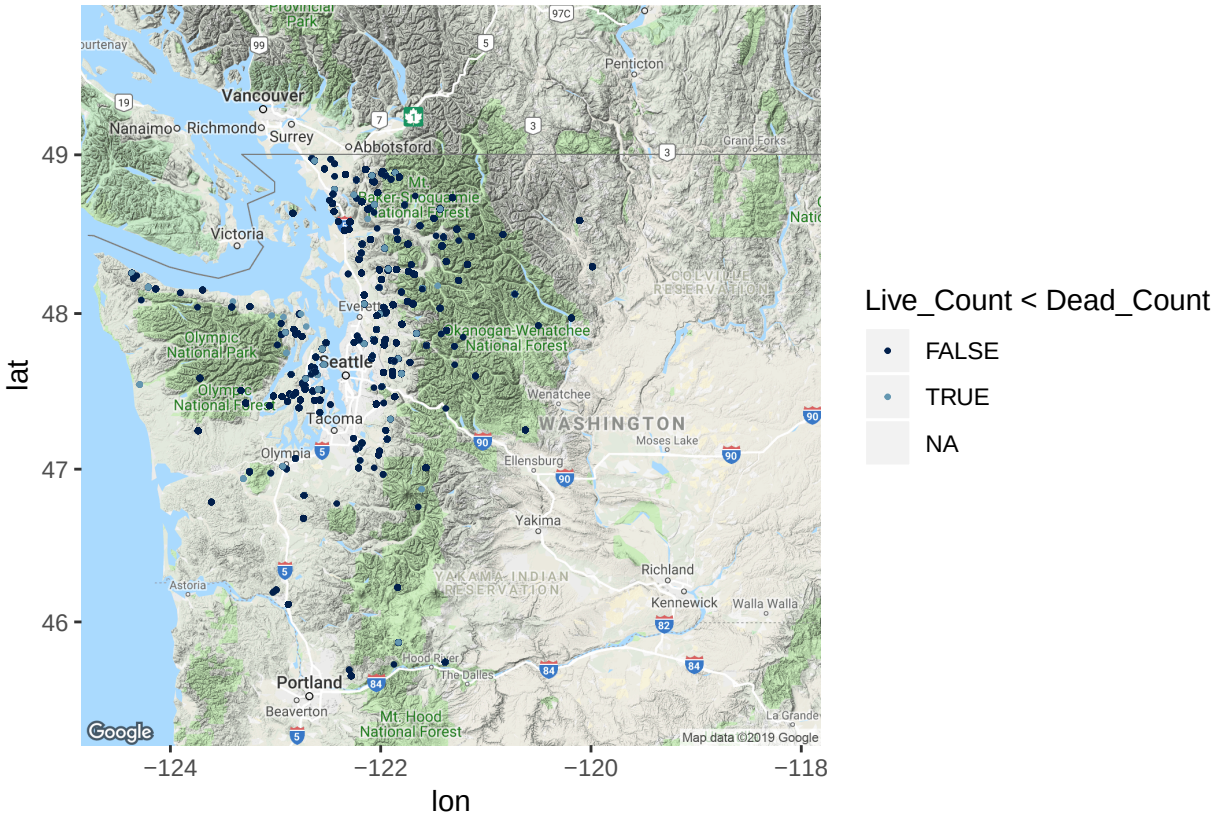



```
ggmap(get_googlemap(center = c(lon = -121.335167, lat = 47.608013),
                    zoom = 7, scale = 2,
```

```
                    maptype ='terrain',
                    color = 'color')) +
  geom_point(aes(x = Longitude, y = Latitude,  colour = Live_Count < Dead_Count), data = Final.Template
  theme(legend.position="right") +
  scale_color_manual(values = colme)
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.608013,-121.335167&zoom=7&size=640
```

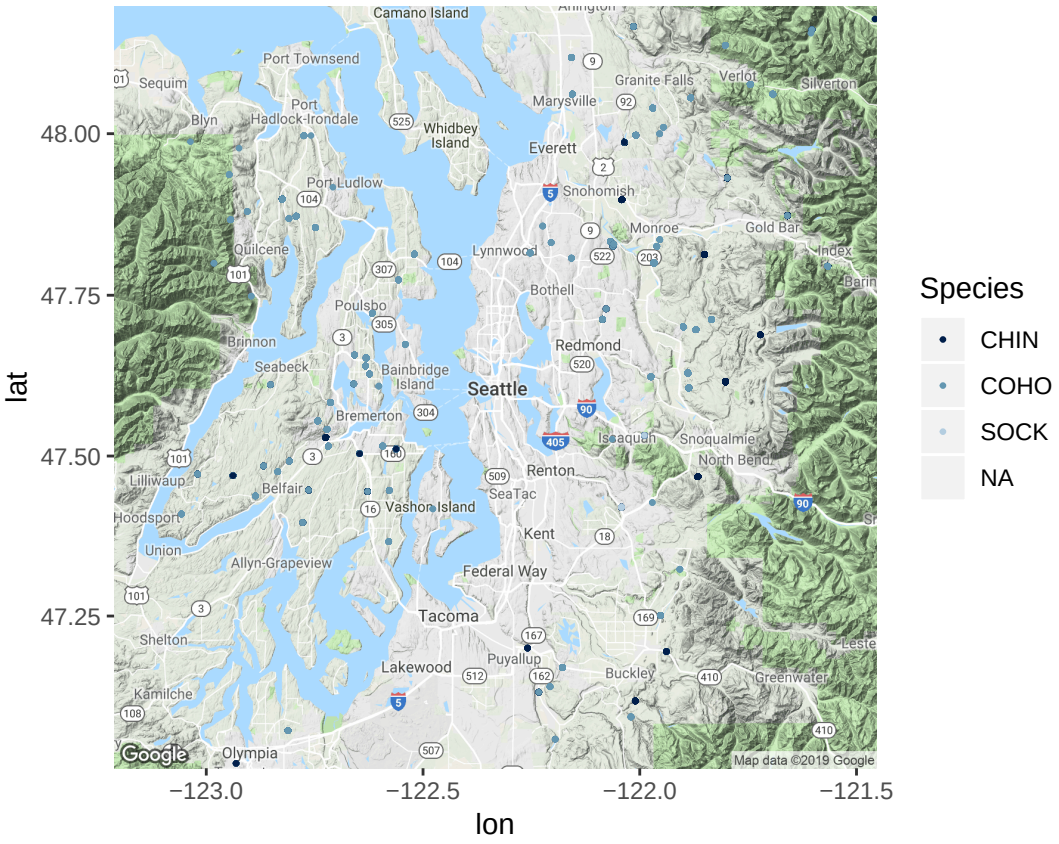
```
## Warning: Removed 14 rows containing missing values (geom_point).
```




```
ggmap(get_googlemap(center = c(lon = -122.335167, lat = 47.608013),
                    zoom = 9, scale = 2,
                    maptype ='terrain',
                    color = 'color')) +
  geom_point(aes(x = Longitude, y = Latitude,  colour = Species), data = Final.Template, size = 0.5) +
  theme(legend.position="right") +
  scale_color_manual(values = colme)
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=47.608013,-122.335167&zoom=9&size=640
```

```
## Warning: Removed 8472 rows containing missing values (geom_point).
```




## Conclusion

I would say this project has significantly helped me with understanding R, as well as testing my patience. All the objectives I had planned for me in this project were accomplished. However, the emphasis on models was not as significant as I probably would have hoped. I will end up leaving some more of that to work in the future, as my plans were to do already. But in the end, ggmap is a very fun package to use, and I imagine when used in tandem with other packages it would become even more enjoyable.