STA 841 Final Project
Predict'em All

Sarah Normoyle, Gonzalo Bustos

December 5, 2016

1 Introduction

PokemonGo is a augmented virtual reality game that came out in July 2016 where a player can catch, train, and fight creatures called Pokemon. These Pokemon appear randomly based on GPS location in the game as if they were in the same location as the player. Pokemon appear and disappear for varying time lengths. There are theoretically 151 species of Pokemon that can appear in the game, with some being much more common than others. These species can be classified into types, with some classified into more than one type. In addition to Pokemon, there are also Pokestops and Gyms based on GPS location in the game. Players can collect items such as Pokeballs at Pokestops and can train and fight Pokemon in Gyms. Because PokemonGo is a game, there is an underlying algorithm that produces Pokemon spawns. Without knowing the algorithm or the random distribution of Pokemon, we seek to find insight into how Pokemon are distributed.

2 Data

The dataset used for this project is from kaggle.com and can be found at [1]. The data set has about 300,000 observations of Pokemon sightings in the game PokemonGo from all over the world. Each observation in the dataset contains information about the sighting, including which Pokemon and location, time, and weather variables. We also collected data on which type each Pokemon is from the reference [2]. For the sake of simplicity and interpretation, we only considered a Pokemon's first type, which is traditionally used to sort Pokemon. We also have variables related to each Pokemon's stats and total stats, which we could also consider as a response variable. Variables that we will consider as covariates include: close to water (Boolean), distance to nearest Pokestop and nearest Gym, population density, terrain type (according to [Ref 3] about MODIS Land Cover), temperature, and time of day (in 24 hour period). Additionally, based on latitude and longitude, we added a couple variables, which are City and Country.

3 Research Question

The ultimate question of the game is being able to predict the next location of a Pokemon. This would require investigating the spatio-temporal relationship of Pokemon spawns, as well as other time and location specific variables. In addition to this question, another interesting question for players is what is the probability of observing a particular Pokemon or a particular type of Pokemon, especially a rare type. Given that our data set is non-uniform sample of exclusively observations of Pokemon, we will address the following question:

- Given that a Pokemon appears, what factors are related to the probability of observing a particular type of species of Pokemon?

4 Exploratory Data Analysis

The data set contains 296,021 observations from 9/2/2016 to 9/8/2016. About half of the observations are within the United States, and the other observations are scattered around the globe. The frequency of each type of Pokemon is different based on our non-representative sample and also based on how often they appear in the game. First we look at the counts per type of Pokemon.

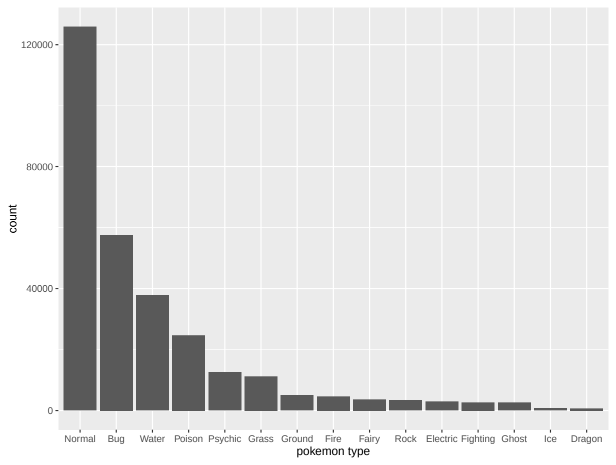

Figure 1: Frequency Counts of Pokemon

It is clear that the frequency of types are not uniform. However, our question is if the distribution of species changes based on other variables. We will consider location, time, and weather variables as potential covariates and explore the relationships among the variables.

4.1 Location

When looking at location as a potential covariate, we first look at where the observations in our dataset occur. Below is a heat map plot of just the United States.

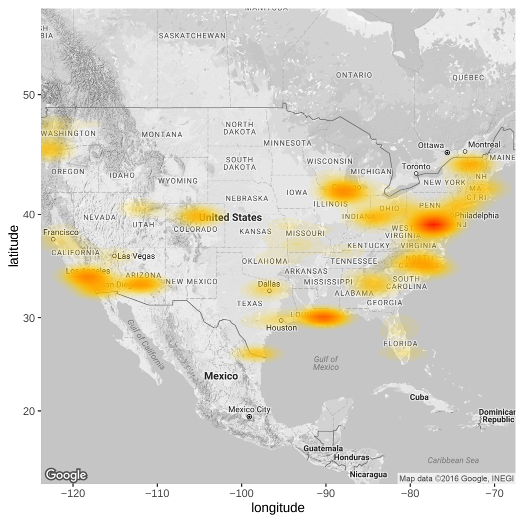

Figure 2: Heat Map of United States Pokemon Counts

After looking at this plot, we believe that our sample of data is neither a representative nor a random sample of Pokemon seen during the time period because certain parts of the country have no points and some parts of the country are over-sampled. Because we are focusing on comparing how species distribution changes among the sample, this is not concerning. Below is another plot showing the difference in distributions across species for terrain types and distance to water.

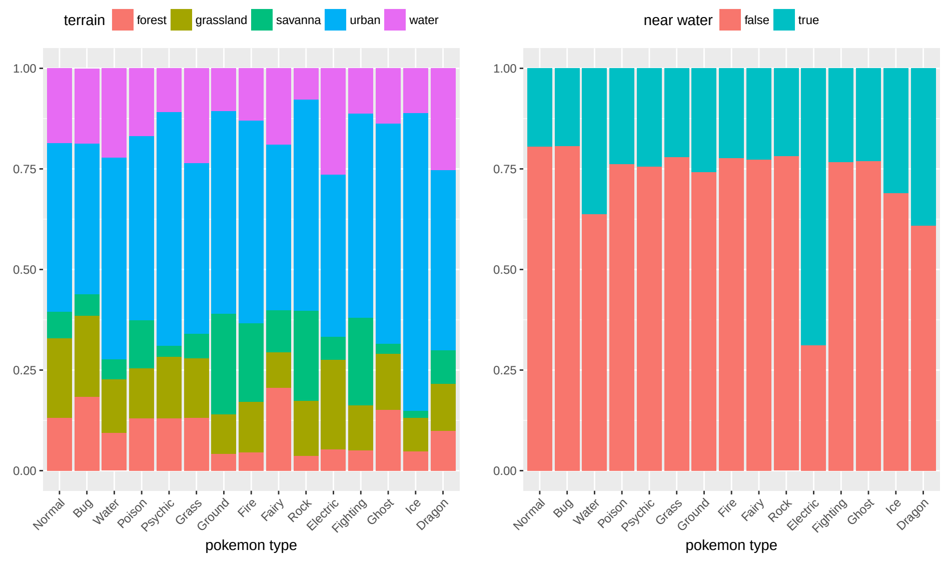

Figure 3: Terrain Types and Distance to Water

On the y-axis of these plots is the proportion of counts, and on the x-axis is type of Pokemon. There seems to be a relationship between terrain type and proportion of species as well as distance to water and proportion of species. For distance to water, of the Pokemon found close to water, there are more water and electric Pokemon for example. In addition, population density is also a variable that relates to the presence of a particular species. Perhaps certain types appear more where it is more/less dense.

4.2 Time and Weather

Because this sample of data is only over the course of 6 days, the main variable we can look into is time of day. From intuition, perhaps certain types of Pokemon are more or less likely to appear at night than other types of Pokemon. In addition, we investigated if weather had a relationship with type of Pokemon. Both time and weather did not seem to have a strong relationship with the distribution of Pokemon on their own, not when included in a final model.

5 Model Methods

5.1 Binary Logistic Regression

First, each type of species was treated as a binary regression problem, which means either that type of Pokemon or not. Of the 15 types of Pokemon, the ones that were most investigated were Normal, Bug, Water, and Poison because they have the most counts. This analysis addresses the question of to what extent do certain time and location variables relate to the probability of seeing a particular type of Pokemon compared to the seeing the others. Predictors were first fit individually and then were added in a step-wise manner to create a full model for each Pokemon type.

In addition to creating a model with fixed effects, we considered incorporating terms for random effects. These random effects capture additionally variation that is not captured by the current fixed effects. Random effects were added for the country of observation.

5.2 Multinomial Probit Regression

After binary logistic regression, a multinomial probit model was also fit to the data that allows for a multinomial response, which would be the different types of Pokemon.

6 Results

Individual predictors had a different relationship with type of Pokemon. For example, for predicting Water vs. not water closeToWater had the biggest change in deviance and the lowest AIC. For Bug Pokemon, type of terrain had biggest change in deviance compared to the null model. When running stepwise regression, each additional variable added significantly changed the AIC value. After looking at AIC and other values, the final model used for each binary regression of Pokemon:

$$\text{logit}(\pi) = \beta_0 + \beta_1 \text{ I(closeToWater)} + \beta_2 \text{ I(pop:rural)} + \beta_3 \text{ I(pop:suburban)} + \beta_4 \text{ I(pop:urban)} + \beta_5 \text{ I(gymin100m)} + \beta_6 \text{ I(pokestopIn100m)} + \beta_7 \text{ I(grassland)} + \beta_8 \text{ I(savanna)} + \beta_9 \text{ I(urban)} + \beta_{10} \text{ water} + (1|\text{Country})$$

The results after fitting 5 separate binary logistic regressions:

Fixed effect estimates:

Parameters	Normal Model	Water Model	Bug Model	Poison Model
Intercept	-0.218	-2.772	-1.055	-2.334
I(closeToWater)	-0.309	0.671	-0.199	0.086
I(rural)	-0.011	0.061	0.037	-0.240
I(pop:urban)	-0.132	0.199	-0.076	0.020
I(gymin100m)	0.017	0.069	-0.545	-0.072
I(pokestopIn100m)	-0.171	0.320	-0.137	0.116
I(grassland)	0.151	0.066	-0.224	-0.337
I(savanna)	-0.342	0.236	-0.757	0.512
I(terr:urban)	-0.071	0.448	-0.482	-0.059
I(terr:water)	0.034	0.574	-0.310	-0.192

Random effect estimates:

	Normal	Water	Bug	Poison
Country Variance	0.1597	0.3637	0.1576	0.2278

There are numerous differences regarding the log odds coefficients between the various binary models. Some variables make a big difference in log odds for some Pokemon types while they don't make much of a difference for other Pokemon types. For example, the odds of seeing a water Pokemon go up in population dense, urban areas near the water and near Pokestops while the odds of seeing a bug Pokemon go up in not population dense, rural areas with forest terrain not near the water and not near Pokestops. In addition, the random effect for Country ranges from about -1 to 1 on most of the models and helps to explain much of the variation. Perhaps there are other variables that differ between countries, or perhaps the distribution of species simply differs that much between countries.

4 data examples to show examples of probabilities from each of the 4 model estimates:

Covariates	1	2	3	4
closeToWater	true	false	false	true
pop density	urban	rural	mid	urban
gymin100m	true	false	false	true
pokestopIn100m	true	false	false	true
terrain	water	forest	grassland	savanna
country	Netherlands	Germany	Spain	UK

Probabilities for each of 4 models for each of 4 example (bolded is biggest of the examples for that model):

Model	1	2	3	4
Normal	0.3448	**0.4776**	0.4330	0.2903
Water	**0.5049**	0.0573	0.0887	0.2030
Bug	0.1189	**0.3578**	0.1738	0.1040
Poison	0.0337	0.0536	0.1093	**0.1197**

When the multinomial model was fit for the types, the fit is much worse because we are incorporating more types of Pokemon. The model always predicts a Normal Pokemon because this is the Pokemon with the highest mean probability. This type of model would need more work and more variables to be able to distinguish situations in which the other Pokemon were more probable. Overall, the models do not do well in predicting which Pokemon will appear given the data. However, we are able to see when the odds of seeing a Pokemon increases or decreases.

7 Limitations

A limitation with this data set is the non-uniform sampling of Pokemon. Because the occurrences of Pokemon are a random sample over time and location, we are not able to model the rate of Pokemon appearances. In addition, because the Pokemon are not sampled uniformly and because we only have presence and not absence data, it is more difficult to model the occurrence of all Pokemon.

However, despite this unequal sampling scheme, we are able to conduct an the analysis of differences between occurrences of types of species. The concern for sampling bias here would be if times or locations with a particular type of species were oversampled compared to the other types. However, if we control for the factors that introduce this sampling biases (such as population density) we are able to make observations about differences between types of species. Our models do not fit the data that well, but there a first step in exploring the detials of the game.

8 Conclusion and Extensions

Based on our analysis, there appears to be a small relationship between location and the distribution of types of species in PokemonGo, and these covariates relate to each of the species differently. We could continue this analysis to do further model comparisons, and explore additional models and covariates. There are also many potential further explorations of this dataset and PokemonGo that could be investigated. It is most likely that Pokemon spawns are not independent observations and are dependent on other Pokemon's spawn time and location. Therefore, building a spatio-temporal dependency structure could be beneficial for the model.

9 References

[1] https://www.kaggle.com/semioniy/predictemall.
[2] http://bulbapedia.bulbagarden.net/wiki/List_of_Pokemon_by_National_Pokedex_number.
[3] http://glcf.umd.edu/data/lc/.
[4] http://stackoverflow.com/questions/14334970/convert-latitude-and-longitude-coordinates-to-country-name-in-r.

10 Appendix: R Code

10.1 coords2country

From Reference [4].

```
library(sp)
library(rworldmap)

# The single argument to this function, points, is a data.frame in which:
#   - column 1 contains the longitude in degrees
#   - column 2 contains the latitude in degrees
coords2country = function(points)
{
        countriesSP <- getMap(resolution='low')


        #setting CRS directly to that from rworldmap
        pointsSP = SpatialPoints(points,
                proj4string=CRS(proj4string(countriesSP)))


        # use 'over' to get indices of the Polygons object
        # containing each point
        indices = over(pointsSP, countriesSP)

        # return the ADMIN names of each country
        indices$ADMIN

}
```

10.2 Cleaning

```
library(data.table)
library(dplyr)
## CLEANING
source("coords2country.R")
pokemon = fread("300k.csv")

filtered = dplyr::select(pokemon,
    pokemonId,
    latitude, longitude,
    appearedLocalTime,
    terrainType,
    closeToWater,
    temperature,
    weatherIcon,
    population_density,
    rural,
    urban,
    suburban,
    midurban,
    gymDistanceKm,
    gymIn100m,
    gymIn250m,
    pokestopIn100m,
    pokestopDistanceKm) %>% as.data.frame()

# pokemon classes
details = read.csv("pokemon_classes.csv", header = TRUE)

# join classes and stats with data
data = left_join(filtered, details, by = "pokemonId")

# water
data$water = 0
data$water[data$first_type == "Water"] = 1

# bug
data$bug = 0
data$bug[data$first_type == "Bug"] = 1

# pyschic
data$psychic = 0
data$psychic[data$first_type == "Psychic"] = 1

# poision
data$poison = 0
data$poison[data$first_type == "Poison"] = 1

# fighting
data$normal = 0
data$normal[data$first_type == "Normal"] = 1


data$high_stats = 0
data$high_stats[data$total_stats > 250] = 1


data$grouped_type = data$first_type
data$grouped_type = as.character(data$first_type)
data$grouped_type[!data$grouped_type %in%
    c("Poison", "Normal", "Water", "Bug") ] = "Other"

## DENSITY
```

```
data$type_density = "mid"
data$type_density[data$population_density > 800] = "urban"
data$type_density[data$population_density < 200] = "rural"

#get countries
data$country = coords2country(data.frame(data$longitude,data$latitude))
data$country = as.character(data$country)

# remove countries that didn't work
data = data[is.na(data$country) == FALSE,]

# grouped terrain types
data$terrain_grouped  = "water"
data$terrain_grouped[data$terrainType %in% c(1,2,4,5)] = "forest"
data$terrain_grouped[data$terrainType %in% c(7,8,9,16)] = "savanna"
data$terrain_grouped[data$terrainType %in% c(10,11,12)] = "grassland"
data$terrain_grouped[data$terrainType %in% c(13)] = "urban"
```

10.3 EDA

```
library(data.table)
library(tidyverse)
library(ggmap)
library(lubridate)
library(stringr)
library(gridExtra)

pokemon = fread('300k.csv')

#sara data
filtered_sara = select(pokemon, pokemonId, latitude, longitude,
                appearedLocalTime, terrainType,
                closeToWater, city, weather, temperature,
                windSpeed, windBearing, pressure, weatherIcon,
                population_density,
                population_density,
                gymDistanceKm,
                gymIn100m,
                gymIn250m,
                pokestopDistanceKm) %>% as.data.frame()

#pokemon classes
sara = read.csv('pokemon_classes.csv', header = TRUE)

#join classes and stats with data
sarita = left_join(filtered_sara, sara, by = 'pokemonId')

#date wrangling
sarita = sarita %>% mutate(pokedate = ymd_hms(str_replace(appearedLocalTime,'T',' ')))

sarita = sarita %>% mutate(poketime = hour(pokedate) + minute(pokedate)/60)
```

```
#add variable for hour intervals
sarita$hour_interval = cut(sarita$poketime,
                                breaks = seq(from = 0, to = 24, by = 2),
                                include.lowest = T, right = F)

#get countries
sarita$countries = coords2country(data.frame(sarita$longitude,sarita$latitude))

#grouped terrain type
sarita$terrain_grouped = "water"
sarita$terrain_grouped[sarita$terrainType %in% c(1,2,4,5)] = "forest"
sarita$terrain_grouped[sarita$terrainType %in% c(7,8,9,16)] = "savanna"
sarita$terrain_grouped[sarita$terrainType %in% c(10,11,12)] = "grassland"
sarita$terrain_grouped[sarita$terrainType %in% c(13)] = "urban"

#filter USA
usa = sarita %>% filter(countries == 'United States of America')

map_usa = get_map(location = 'USA', zoom = 4, color = 'bw', source = 'google')

ggmap(map_usa) +
        geom_point(data = usa, aes(x = longitude, y = latitude,
                                   color = closeToWater))

#filter Chile
chile = sarita %>% filter(countries == 'Chile')

map_chile = get_map(location = 'Chile', zoom = 6, color = 'bw',
                  source = 'google')

ggmap(map_chile) +
        geom_point(data = chile, aes(x = longitude, y = latitude,
                                    color = closeToWater))

#boxplot
ggplot(data = sarita, aes(x = first_type, y = total_stats)) +
        geom_boxplot()

#jitter
ggplot(data = sarita, aes(x = first_type, y = total_stats, color = closeToWater)) +
        geom_jitter(width = 0.5)

#hour interval
ggplot(data = sarita, aes(x = hour_interval, y = total_stats, color = first_type)) +
        geom_jitter(width = 0.5)


ggplot(data = sarita, aes(x = first_type, y = poketime)) +
        geom_jitter()
```

```
#plot terrainType vs first_type
ggplot(data = sarita, aes(x = first_type, y = terrainType, color = closeToWater)) +
        geom_jitter()

#bar plots
ggplot(data = sarita, aes(first_type)) + geom_bar()

#to change plot order of bars, change levels in underlying factor
reorder_size = function(x) {
        factor(x, levels = names(sort(table(x), decreasing = T)))
}

#eda plots
water_plot = ggplot(data = sarita, aes(reorder_size(first_type))) +
        geom_bar(aes(fill = closeToWater), position = 'fill') +
        xlab('pokemon type') +
        ylab('') +
        labs(fill = 'near water') +
        theme(legend.position = 'top') +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))

terrain_plot = ggplot(data = sarita, aes(reorder_size(first_type))) +
        geom_bar(aes(fill = terrain_grouped), position = 'fill') +
        xlab('pokemon type') +
        ylab('') +
        labs(fill = 'terrain') +
        theme(legend.position = 'top') +
        theme(axis.text.x = element_text(angle = 45, hjust = 1))

pdf('side_by_side_plot.pdf', width = 10, height = 6)
grid.arrange(terrain_plot, water_plot, ncol = 2)
dev.off()

#heat map
usa_map = ggmap(map_usa, base_layer = ggplot(aes(x = longitude, y = latitude),
                                             data = sarita))
pdf('heatmap.pdf', width = 8, height = 6)
usa_map +
        stat_density2d(aes(x = longitude, y = latitude, fill = ..level..,
                           alpha = ..level..), bins = 30, geom = 'polygon',
                       data = usa) +
        scale_fill_gradient(low = 'gold', high = 'red', guide = F) +
        guides(alpha = F)
dev.off()

#pokemon type frequency
pdf('count_by_type.pdf', width = 8, height = 6)
ggplot(data = sarita, aes(reorder_size(first_type))) +
        geom_bar() +
        xlab('pokemon type')
dev.off()
```

```
#playing with hour interval plots
ggplot(data = sarita, aes(hour_interval)) +
        geom_bar() +
        xlab('2-hour interval') +
        facet_wrap(~ first_type)

hour_counts = sarita %>%
        group_by(hour_interval, first_type) %>%
        summarize(count = n()) %>%
        mutate(freq = count / sum(count))

ggplot(hour_counts, aes(x = hour_interval, y = freq, group = first_type)) +
        geom_line(aes(color = first_type))
```

10.4 Models

```
library(MASS)
library(lme4)
library(nnet)

# Water: individual level models
fit1 = glm(water ~ closeToWater, data = data,
family = binomial(link = 'logit'))

fit2 = glm(water ~ temperature, data = data,
family = binomial(link = 'logit'))
fit3 = glm(water ~ type_density,
data = data,
    family = binomial(link = 'logit'))
fit4 = glm(water ~ terrainType,
data = data,
family = binomial(link = 'logit'))

fit5 = glm(water ~ type_density,
data = data,
family = binomial(link = 'logit'))

fit6 = glm(water ~ weatherIcon,
data = data,
family = binomial(link = 'logit'))

full_model = glm(water ~ closeToWater + temperature +
terrainType + type_density + gymIn100m,
data = data,
family = binomial(link = 'logit'))

null = glm(water ~ 1 ,
data = data,
family = binomial(link = 'logit'))

fit = stepAIC(null, scope=list(lower=null, upper=full_model),
 direction = "forward")



# bug: individual level models
fit1 = glm(bug ~ closeToWater, data = data,
family = binomial(link = 'logit'))

fit2 = glm(bug ~ temperature, data = data,
family = binomial(link = 'logit'))
fit3 = glm(bug ~ type_density,
data = data,
family = binomial(link = 'logit'))

fit4 = glm(bug ~ terrainType,
data = data,
family = binomial(link = 'logit'))



## BINARY RANDOM EFFECT MODELS
# fit model with random effects
fit_water = glmer(water ~
closeToWater +
type_density +
gymIn100m +
pokestopIn100m +
    terrain_grouped +
(1|country),
data = data,
family = binomial(link = 'logit'))

ref = ranef(fit_water)$country
REs = data.frame(country = row.names(ref), c(ref))
arrange(REs, X.Intercept.)
fixef(fit_water)


# BUG
fit_bug = glmer(bug ~
closeToWater +
type_density +
gymIn100m +
pokestopIn100m +
terrain_grouped +
(1|country),
data = data,
family = binomial(link = 'logit'))

ref = ranef(fit_bug)$country
REs = data.frame(country = row.names(ref), c(ref))
```

```
arrange(REs, X.Intercept.)
fixef(fit)


# NORMAL
fit_normal = glmer(normal ~
closeToWater +
type_density +
gymIn100m +
pokestopIn100m +
    terrain_grouped +
(1|country),
data = data,
family = binomial(link = 'logit'))
summary(fit_normal)
ref = ranef(fit_normal)$country
REs = data.frame(country = row.names(ref), c(ref))
arrange(REs, X.Intercept.)

# Poison
fit_poison = glmer(poison ~
closeToWater +
type_density +
gymIn100m +
pokestopIn100m +
    terrain_grouped +
(1|country),
data = data,
family = binomial(link = 'logit'))

summary(fit_poison)
ref = ranef(fit_poison)$country
REs = data.frame(country = row.names(ref), c(ref))
arrange(REs, X.Intercept.)

# new data predictions

newdata = data.frame(
closeToWater = c("true",  "false", "false", "true"),
type_density = c("urban",  "rural", "mid", "urban"),
gymIn100m = c("true",  "false", "false", "true"),
pokestopIn100m = c("true",  "false", "false", "true"),
terrain_grouped = c("water",
"forest", "grassland", "savanna"),
country = c("Netherlands",
"Germany", "Spain", "United Kingdom"))

predict(fit_water, newdata, type = 'response')
predict(fit_bug, newdata, type = 'response')
predict(fit_normal, newdata, type = 'response')
predict(fit_poison, newdata, type = 'response')
```

```
### MULTINOMIAL
multi_fit = multinom(grouped_type ~
closeToWater +
type_density +
gymIn100m +
pokestopIn100m +
terrain_grouped,
data = data)

predictions = predict(multi_fit, data)
```