West Nile Virus Project

Keith Hultman

12/4/2016

This project will attempt to predict the likelihood of a positive test for West Nile Virus (WNV) in mosquitos captured using traps around Chicago, IL. The data sets originate from the Kaggle competition, West Nile Virus Prediction.

The analysis will follow an iterative approach to finding a production model for West Nile forecasting. There are several data science iterative methods, and I will use the CRISP model for organizing this project. Additionally, this analysis conforms to reproducible research standards and can be reconstructed from source files available at my GitHub repository.

CRISP Iteration 1

Business Understanding

The goal for this project is to develop a predictive model that will forecast the probability of WNV presense in 138 mosquito traps around Chicago over the course of a season. WNV is a communicable disease that is spread through its most common vector, mosquitos. Most people infected with WNV develop no symtoms, but 1% will develop neurological symptoms including headache, high fever, neck stiffness, disorientation, coma, tremors, seizures, or paralysis, according to the CDC. There are no medical treatments or cures for WNV, so preventative measures are required to reduce infection to the human population. Local community methods of mosquito control include reduction of larval habitats and applying insecticides targetting larvae or adult mosquitoes.

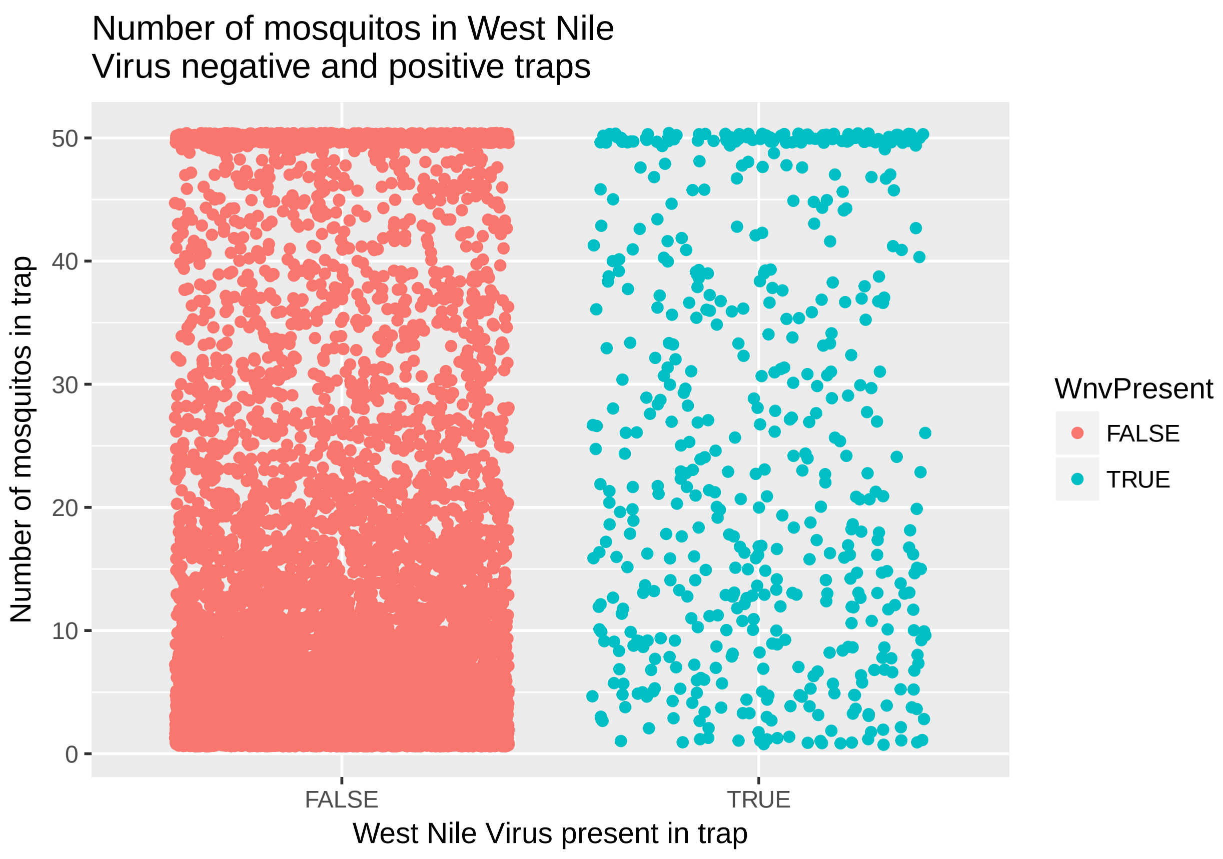
Being able to predict the most likely sites of WNV outbreak is highly valuable to reduce the financial and human cost of over-application of insecticides. This project will focus on developing a predictive model based on geographic information, weather conditions, presense of mosquito species, and historical trends of mosquito populations and WNV presense in the Chicago area.

Data Understanding

Kaggle supplies five data sets for the competition, three of which can be used in model training. The train.csv file includes several variables associated with each West Nile Virus test including the trap id, geo-coordinates, date, species of mosquito present, number of mosquitos present, and a binary variable indicating the presence or absence of West Nile Virus (WnvPresent), our target variable. The test.csv contains all of the same features as the train.csv file except the number of mosquitos present and the target WnvPresent variable. These data sets were obtained from West Nile Virus testing period between 2007 and 2014 with the training set containing the odd years and the test set containing the even years. The weather.csv file contains historic weather data collected at the O'Hare and Midway airports concurrent with the mosquito testing time period. The spray.csv file includes the date and location of chemical spraying conducted by the city during 2011 and 2013. Since this spray data set would only have a large impact during one of the training years, I did not include the spray data in any of my models. For the weather data, since the differences in variables between O'Hare and Midway were generally small, I chose to use only the O'Hare data for all traps, rather than compute the closest station for each trap.

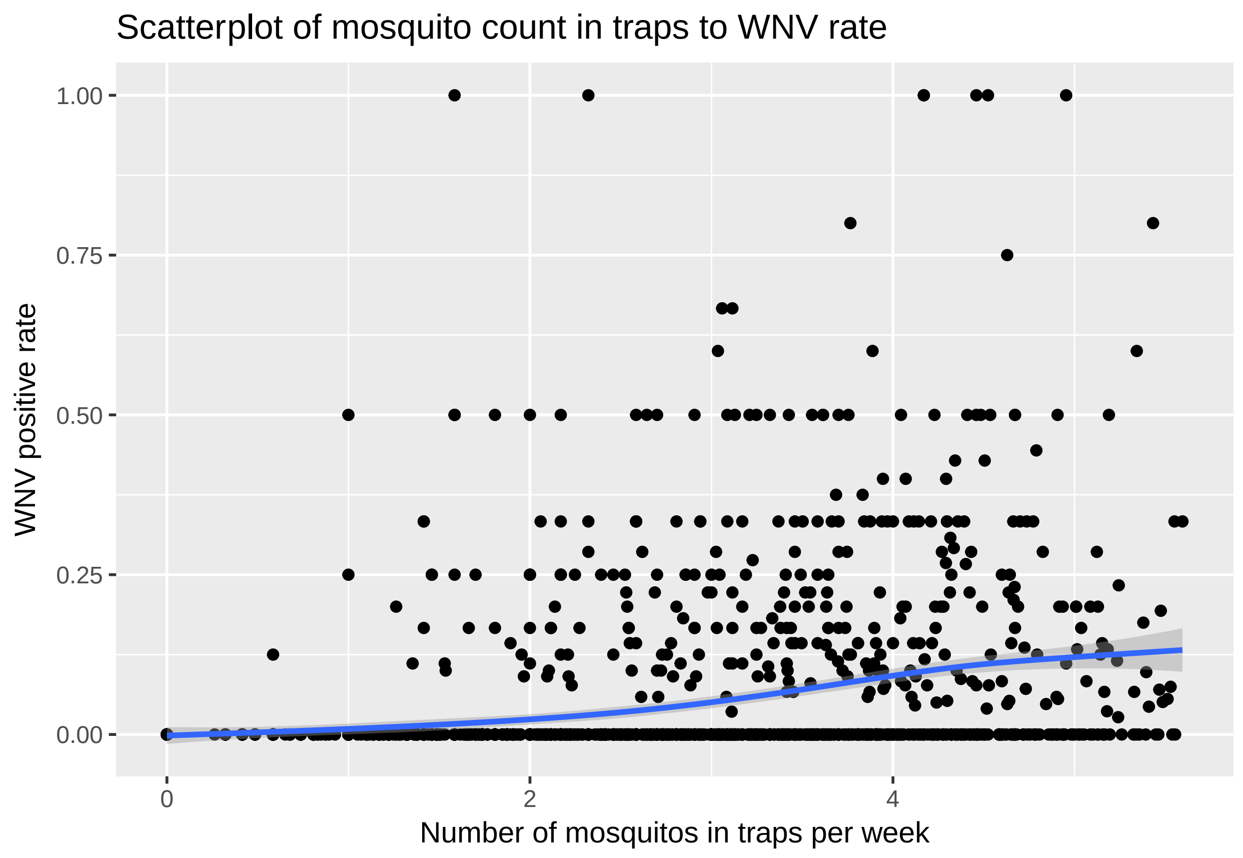
One of the largest factors influencing viral load within a population is the size of the population. A larger population means there are more mosquitos who can become infected and it also increases the number of interactions between hosts, allowing faster spread of virus from individual to individual. The size of the population can be estimated from the number of mosquitos found in each trap. However, *this variable is not present in the test set*, and will need to be imputed from other variables before a final model can be fit to the presence of West Nile using population as an input.

```
library(plyr)
suppressMessages(library(tidyverse))
load("./data/train.RData")
load("./data/test.RData")
source("./src/functions.R")

ggplot(train, aes(x=WnvPresent, y=NumMosquitos)) +
  geom_jitter(aes(color=WnvPresent)) +
  ggtitle("Number of mosquitos in West Nile \nVirus negative and positive traps") +
  xlab("West Nile Virus present in trap") + ylab("Number of mosquitos in trap")
```



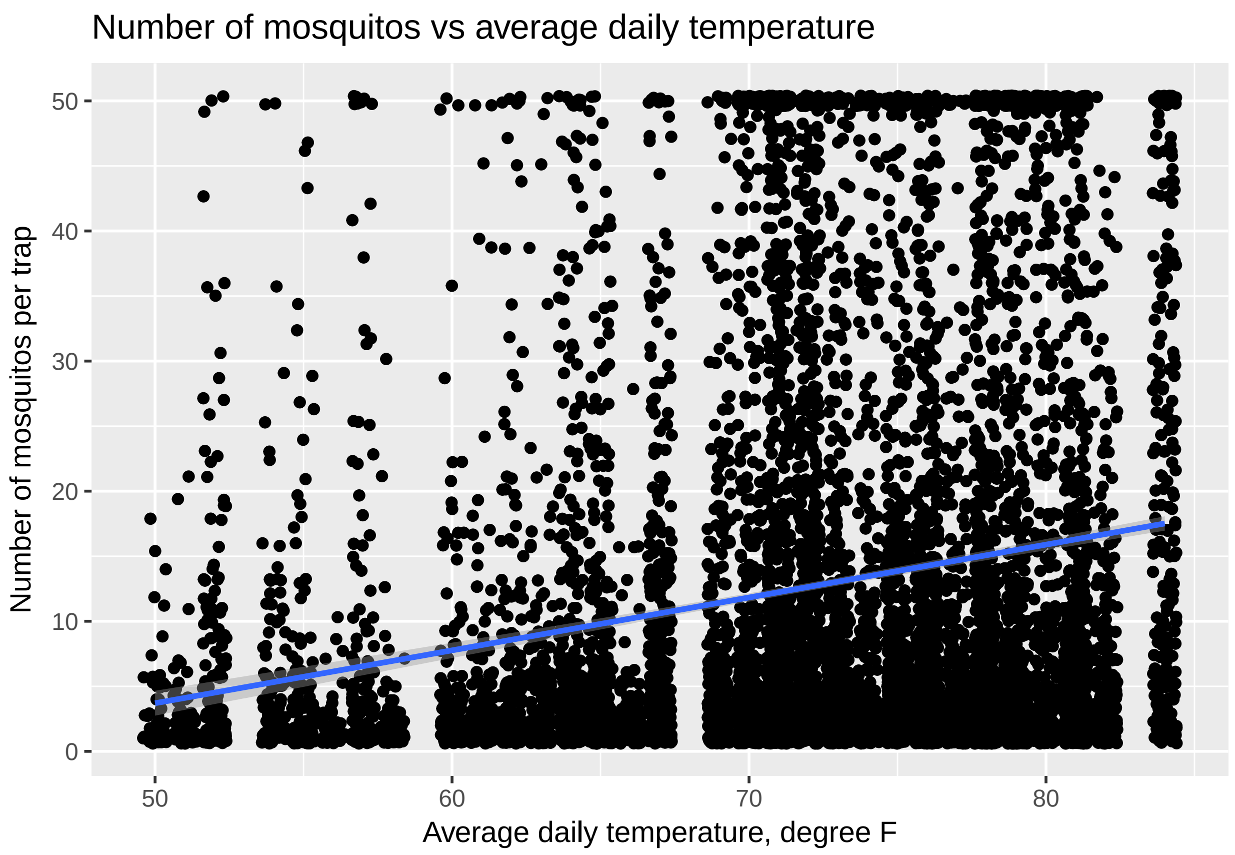
```
train %>% group_by(Trap, Week) %>%
  summarise(WNVrate = sum(WnvPresent)/n(), NumMosPer = sum(NumMosquitos)/n()) %>%
  ggplot(aes(log2(NumMosPer), WNVrate)) + geom_point() + stat_smooth() +
  ggtitle("Scatterplot of mosquito count in traps to WNV rate") +
  xlab("Number of mosquitos in traps per week") + ylab("WNV positive rate")
```

```
## `geom_smooth()` using method = 'gam'
```



Based on this initial data exploration, I will first try and develop a model that predicts the number of mosquitos in each trap and then use that predicted value as a variable in a model that predicts whether West Nile is present or absent. For predicting the number of mosquitos, I will first attempt using weather variables and weekly historic averages. Before modeling I examined scatterplots of various input variables agains mosquito counts, like the graph below comparing temperature to mosquito counts.

```r
ggplot(train, aes(x=Tavg, y=NumMosquitos)) +
  geom_jitter() +
  geom_smooth(method=lm) +
  ggtitle("Number of mosquitos vs average daily temperature") +
  xlab("Average daily temperature, degree F") + ylab("Number of mosquitos per trap")
```



Data Preparation

I cleaned up the weather and training data sets to impute missing values and delete non-informative columns. I next engineered several attributes from existing variables that I thought would be useful. Moving averages of temperature and precipitation, as well as weekly averages in overall mosquito counts were added to weather and training respectively. The code for variable engineering scripts (01_weather.R, 02_train.R) can be found in the src folder.

Modeling: Linear regression to predict mosquito population

For this first round of modeling, I will use linear regression to predict the number of mosquitos using weather data and historical averages per week. I expect that higher average temperatures and greater moisture should have a positive effect on the mosquito population.

Then I will use logistic regression to estimate the probability of the presence of West Nile Virus.

```
lm_variables <- c("Tmax",
                  "Tmin",
                  "Tavg",
                  "DewPoint",
                  "WetBulb",
                  "Sunrise",
                  "Sunset",
                  "PrecipTotal",
                  "ResultSpeed",
```

```r
                  "ma3",
                  "ma5",
                  "ma10",
                  "precip3d",
                  "precip5d",
                  "precip10d")

lm_variables <- paste(lm_variables, collapse = "+")
fmla <- paste("NumMosquitos", lm_variables, sep = "~")

lm1 <- lm(fmla, data = train)

train$lmNumMosq <- predict(lm1, newdata = train)
test$lmNumMosq <- predict(lm1, newdata = test)

summary(lm1)
```

```
##
## Call:
## lm(formula = fmla, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -42.612 -10.560  -5.217   4.735  48.433
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -232.57611   24.86694  -9.353  < 2e-16 ***
## Tmax           1.01581    0.34433   2.950 0.003184 **
## Tmin           1.45947    0.34312   4.254 2.12e-05 ***
## Tavg          -2.63073    0.67937  -3.872 0.000108 ***
## DewPoint      -0.75020    0.12237  -6.131 9.06e-10 ***
## WetBulb        1.04056    0.18660   5.576 2.52e-08 ***
## Sunrise        0.10456    0.01405   7.444 1.05e-13 ***
## Sunset         0.07951    0.01042   7.628 2.60e-14 ***
## PrecipTotal   -0.62699    0.40641  -1.543 0.122920
## ResultSpeed    0.32517    0.05713   5.692 1.29e-08 ***
## ma3            0.49141    0.10574   4.648 3.40e-06 ***
## ma5           -0.55230    0.12015  -4.597 4.34e-06 ***
## ma10           0.60029    0.07989   7.514 6.19e-14 ***
## precip3d     -34.52756    1.39091 -24.824  < 2e-16 ***
## precip5d      42.61891    1.95292  21.823  < 2e-16 ***
## precip10d      4.00813    1.31473   3.049 0.002305 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.2 on 10490 degrees of freedom
## Multiple R-squared:  0.1141, Adjusted R-squared:  0.1128
## F-statistic: 90.05 on 15 and 10490 DF,  p-value: < 2.2e-16
```

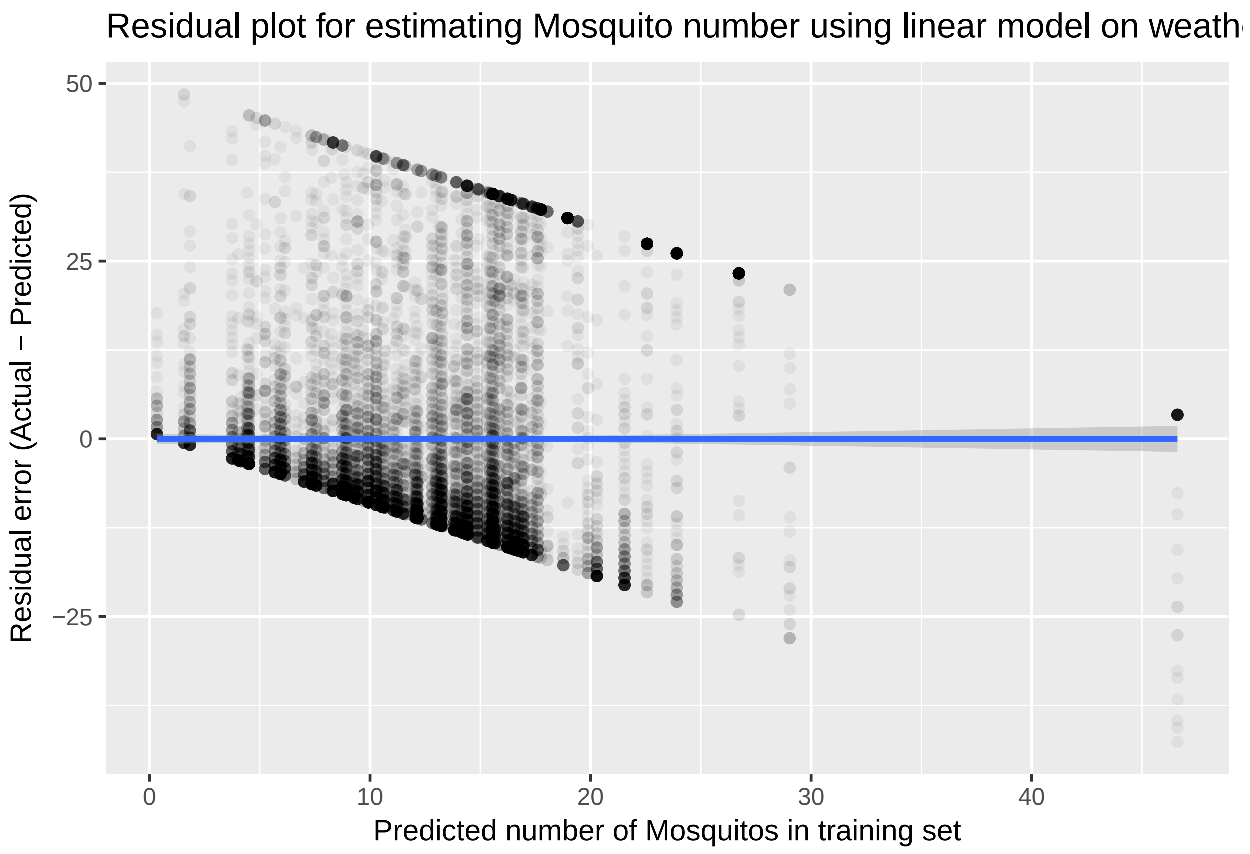
Evaluate linear model for predicting mosquito number

The R^2 of the first model is 0.11, meaning that only 11% of observed mosquito population variation is explained by our weather model.To further evaluate the linear regression model, we can examine the residual errors against predicted numbers in the training data.

```
RMSE <- function(x, y){
  sqrt(sum((x-y)^2))
}

print(paste("RMSE score of lm1: ", round(RMSE(train$lmNumMosq, train$NumMosquitos), 1)))
```

```
## [1] "RMSE score of lm1:  1556.4"
```

```
ggplot(data = train, aes(x=lmNumMosq, y=(NumMosquitos - lmNumMosq))) +
  geom_point(alpha = 0.05) +
  ggtitle("Residual plot for estimating Mosquito number using linear model on weather data") +
  xlab("Predicted number of Mosquitos in training set") +
  ylab("Residual error (Actual - Predicted)") +
  stat_smooth(method=lm)
```



The residual plot appears to show bias in overestimating mosquitos at the low end and underestimating at the high end. However, the trend shows that this is indeed zero. Let's now include historical averages for each week in the year. We can also remove some variables that were not significantly informative.

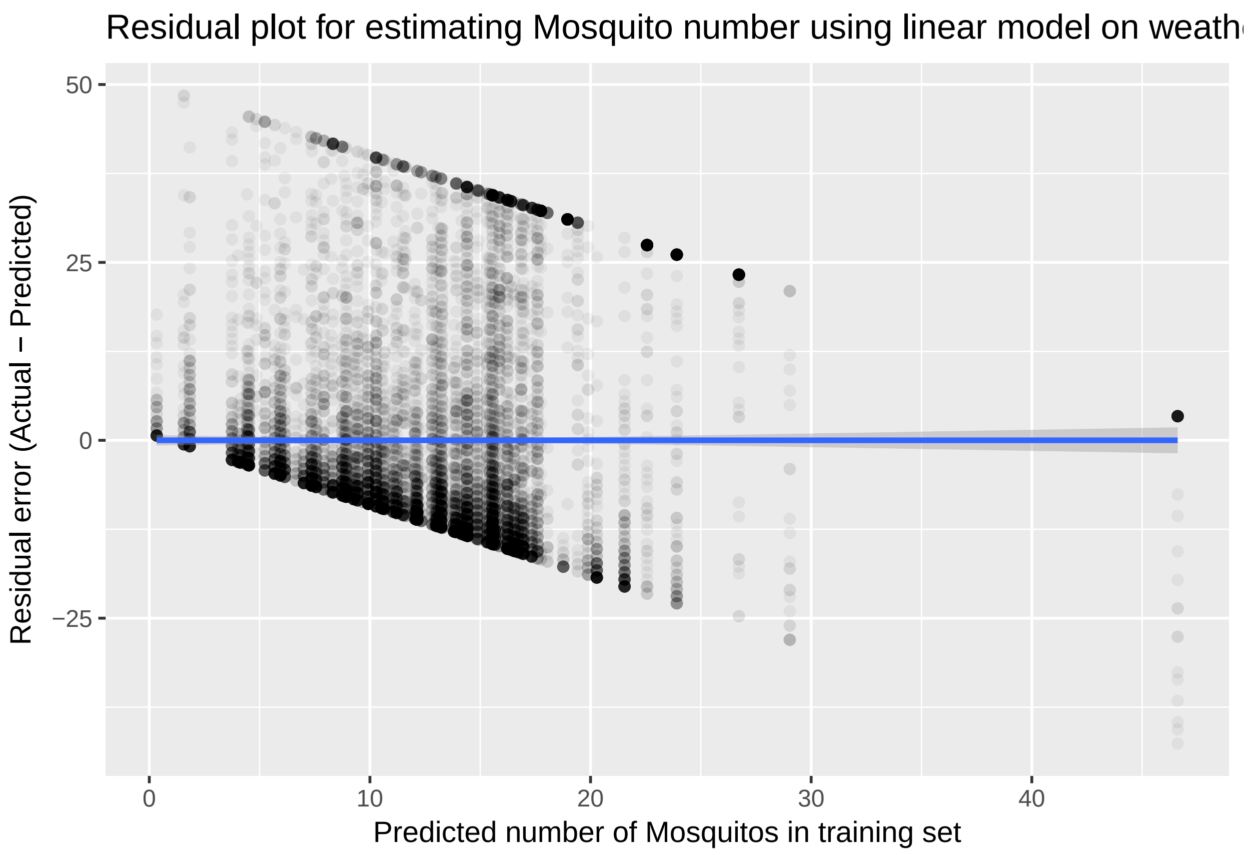
```r
lm_variables2 <- c("WeekAvgMos",
                   "Tavg",
                   "Tmin",
                   "WetBulb",
                   "ResultSpeed",
                   "ma5",
                   "ma10",
                   "precip5d")

lm_variables2 <- paste(lm_variables2, collapse = "+")
fmla2 <- paste("NumMosquitos", lm_variables2, sep = "~")

lm2 <- lm(fmla2, data = train)

train$lm2NumMosq <- predict(lm2, newdata = train)
test$lm2NumMosq <- predict(lm2, newdata = test)

summary(lm2)
```

```
##
## Call:
## lm(formula = fmla2, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.208 -11.510  -5.859   4.735  47.170
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -14.08445    2.70289  -5.211 1.92e-07 ***
## WeekAvgMos    0.81105    0.05907  13.730  < 2e-16 ***
## Tavg          0.20414    0.07286   2.802  0.00509 **
## Tmin          0.08354    0.06112   1.367  0.17171
## WetBulb      -0.08318    0.07265  -1.145  0.25229
## ResultSpeed   0.13716    0.05670   2.419  0.01558 *
## ma5           0.10689    0.07215   1.482  0.13849
## ma10         -0.09547    0.07869  -1.213  0.22510
## precip5d      1.03978    0.59204   1.756  0.07907 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.62 on 10497 degrees of freedom
## Multiple R-squared:  0.0633, Adjusted R-squared:  0.06258
## F-statistic: 88.67 on 8 and 10497 DF,  p-value: < 2.2e-16
```

```r
print(paste("RMSE score of lm1: ", round(RMSE(train$lm2NumMosq, train$NumMosquitos), 1)))
```

```
## [1] "RMSE score of lm1:  1600.4"
```

```r
ggplot(data = train, aes(x=lmNumMosq, y=(NumMosquitos - lmNumMosq))) +
  geom_point(alpha = 0.05) + stat_smooth(method=lm) +
  ggtitle("Residual plot for estimating Mosquito number using linear model on weather data") +
  xlab("Predicted number of Mosquitos in training set") + ylab("Residual error (Actual - Predicted)")
```



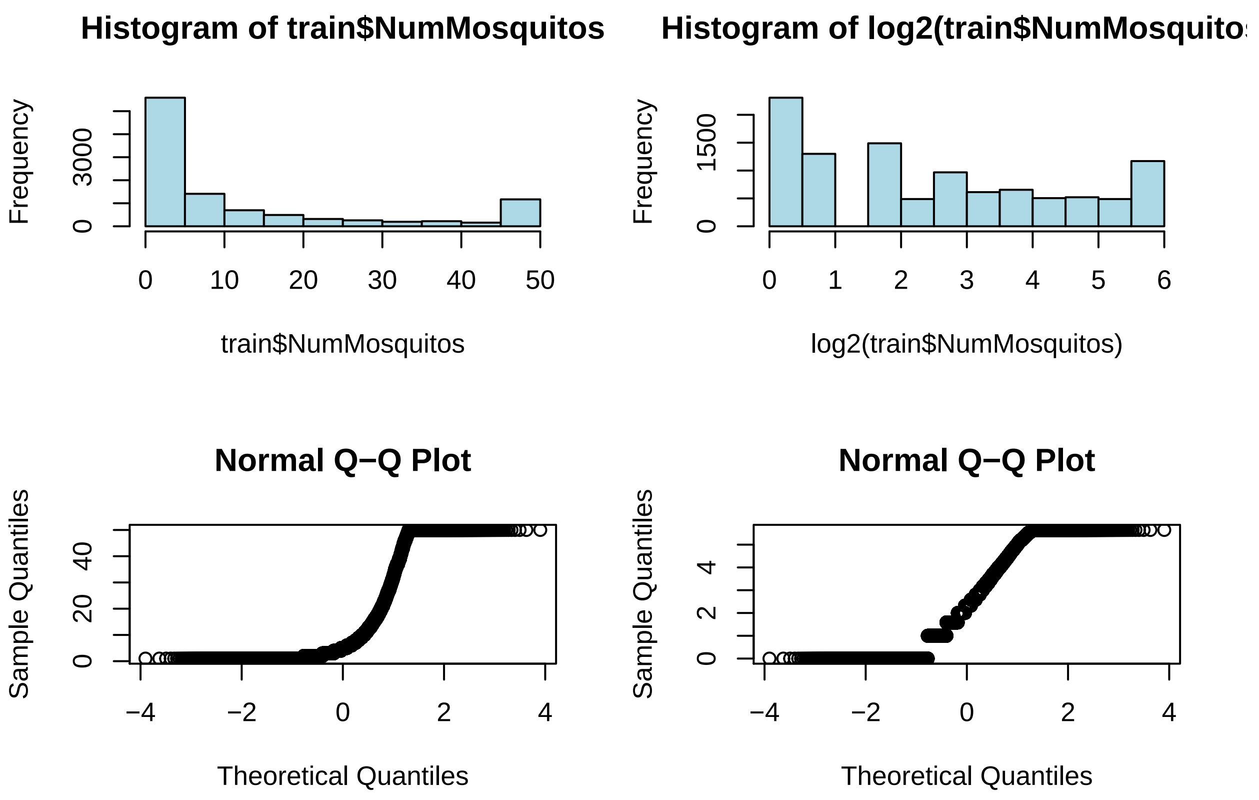
This results in an even lower R^2 value of 0.06 and a higher RMSE error score.

CRISP Iteration 2

Data Understanding: Mosquito number distribution

In order to get a better estimate of mosquito counts, I next wanted to go back and examine the distribution of the Mosquito counts a bit further.

```r
par(mfcol = c(2,2))
hist(train$NumMosquitos, col = "lightblue")
qqnorm(train$NumMosquitos)
hist(log2(train$NumMosquitos), col = "lightblue")
qqnorm(log2(train$NumMosquitos))
```



```r
par(mfcol = c(1,1))
```

The total number is capped at 50 mosquitos, and the distribution is centered close to zero with no negative values. Most observations have only 1 mosquito. I attempted to use log2 transformation on mosquito number but that did not improve the regression.

Modeling: Poisson Regression

The distribution of mosquitos looks closer to a capped Poisson distribution and this can be used to fit a general linear model.

```r
glm1 <- glm(fmla, data = train, family = "poisson")

train$glm1NumMosq <- predict(glm1, newdata = train)
test$glm1NumMosq <- predict(glm1, newdata = test)

summary(glm1)
```

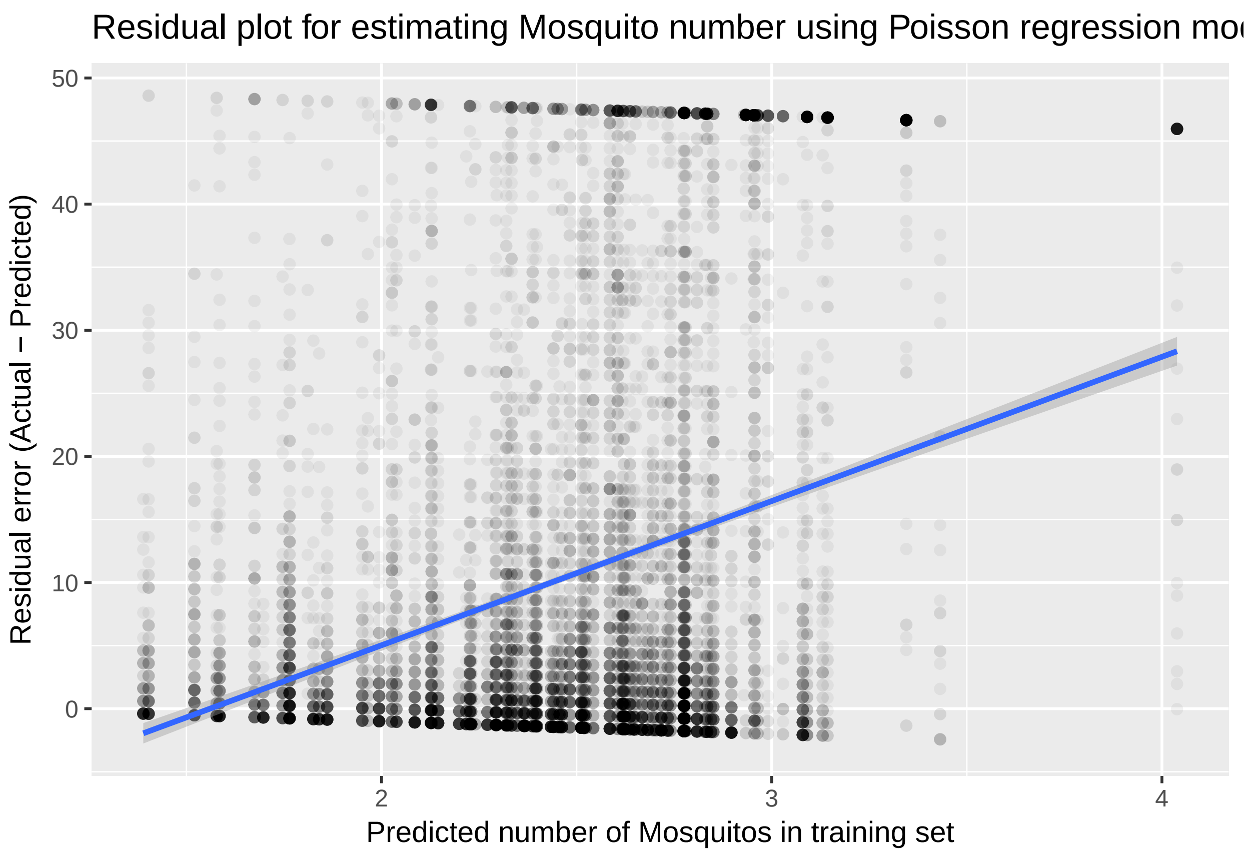
```
##
## Call:
## glm(formula = fmla, family = "poisson", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.181  -3.565  -2.020   1.351  12.611
##
## Coefficients:
```

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.867e+01  4.953e-01  -37.70   <2e-16 ***
## Tmax         6.956e-02  6.389e-03   10.89   <2e-16 ***
## Tmin         1.079e-01  6.420e-03   16.80   <2e-16 ***
## Tavg        -1.818e-01  1.257e-02  -14.46   <2e-16 ***
## DewPoint    -6.123e-02  2.425e-03  -25.25   <2e-16 ***
## WetBulb      8.017e-02  3.724e-03   21.53   <2e-16 ***
## Sunrise      8.952e-03  2.723e-04   32.88   <2e-16 ***
## Sunset       6.765e-03  2.089e-04   32.38   <2e-16 ***
## PrecipTotal -1.260e-01  1.008e-02  -12.50   <2e-16 ***
## ResultSpeed  3.413e-02  1.051e-03   32.48   <2e-16 ***
## ma3          3.961e-02  1.994e-03   19.87   <2e-16 ***
## ma5         -5.257e-02  2.267e-03  -23.19   <2e-16 ***
## ma10         5.876e-02  1.538e-03   38.20   <2e-16 ***
## precip3d    -1.777e+00  1.859e-02  -95.58   <2e-16 ***
## precip5d     2.105e+00  2.655e-02   79.28   <2e-16 ***
## precip10d    1.930e-01  2.184e-02    8.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 178683  on 10505  degrees of freedom
## Residual deviance: 155974  on 10490  degrees of freedom
## AIC: 193976
##
## Number of Fisher Scoring iterations: 5
```

```r
print(paste("RMSE score of lm1: ", round(RMSE(train$glm1NumMosq, train$NumMosquitos), 1)))
```

```
## [1] "RMSE score of lm1:  1955.4"
```

```r
ggplot(data = train, aes(x=glm1NumMosq, y=(NumMosquitos - glm1NumMosq))) +
  geom_point(alpha = 0.05) + stat_smooth(method = lm) +
  ggtitle("Residual plot for estimating Mosquito number using Poisson regression model on weather data"
  xlab("Predicted number of Mosquitos in training set") + ylab("Residual error (Actual - Predicted)")
```



This is much worse, as it only predicts up to 3 mosquitos per trap and appears to undercount mosquitos quite a bit. The RMSE is much higher at 1955. Thus far, the first linear regression model is the best for predicting Mosquito number on the training data.

Data understanding: WNV presence distribution

Moving on to predicting our target variable, the presence of West Nile Virus (WnvPresent in the data set), I will first examine the distribution of the variable.

```
table(train$WnvPresent)
```

```
##
## FALSE  TRUE
##  9955   551
```

```
WnvPresentOverall <- table(train$WnvPresent)[2] / sum(table(train$WnvPresent))
round(WnvPresentOverall, 3)
```

```
##  TRUE
## 0.052
```

Finding WNV is rare, in only ~ 5% of cases. This means our classifier is unbalanced and should be kept in mind when coming up with models that accurately predict such a rare event.

Modeling: Logistic regression to predict WNV infection

```
lgm1 <- glm(WnvPresent ~ NumMosquitos + ma10 + WetBulb + Tmin + Summer + Year, data = train, family = bi

train$Wnv.glm.Pred <- predict(lgm1, newdata = train, type = "response")

# Use linear regression for mosquito number to approximate mosquito number in test set
test$NumMosquitos <- test$lmNumMosq
test$Wnv.glm.Pred <- predict(lgm1, newdata = test, type = "response")
glm.submission1 <- select(test, Id, WnvPresent = Wnv.glm.Pred)
```

```
## Adding missing grouping variables: `Trap`, `Year`
```

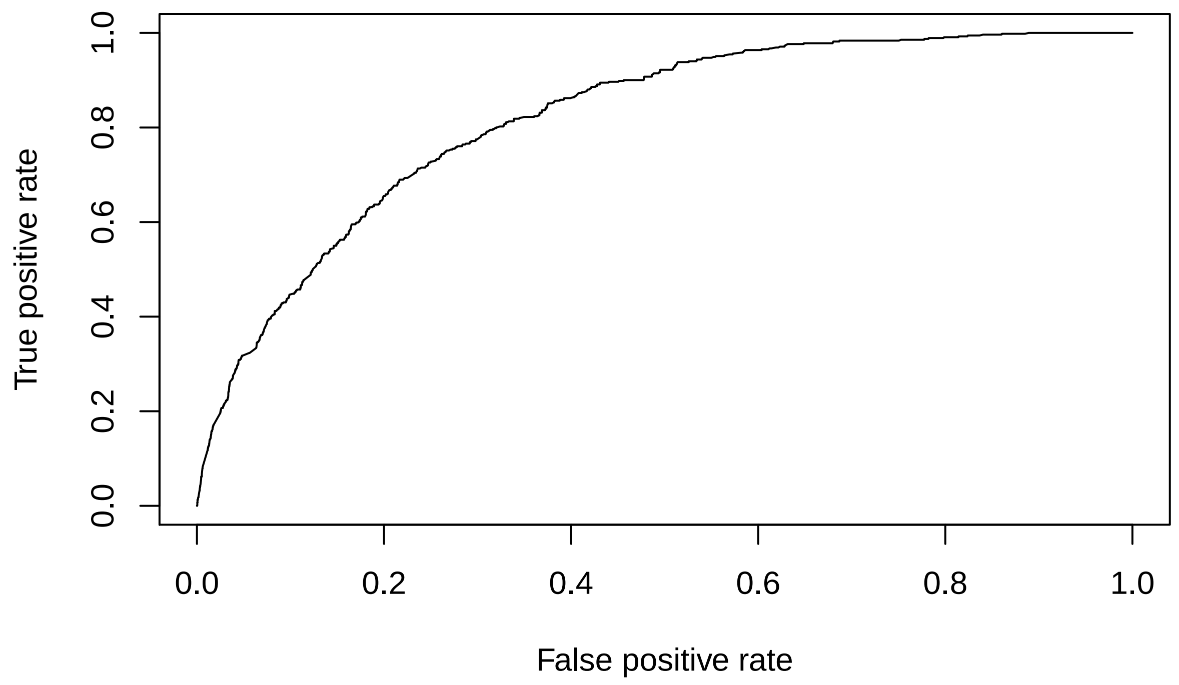
```
write_csv(glm.submission1, path = "./data/glm.submission1.csv")

summary(lgm1)
```

```
##
## Call:
## glm(formula = WnvPresent ~ NumMosquitos + ma10 + WetBulb + Tmin +
##     Summer + Year, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2425  -0.3443  -0.2067  -0.1136   3.3449
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.965e+02  3.962e+01 -10.009  < 2e-16 ***
## NumMosquitos  4.202e-02  2.439e-03  17.233  < 2e-16 ***
## ma10         -7.916e-02  1.419e-02  -5.579 2.42e-08 ***
## WetBulb       1.334e-01  2.131e-02   6.259 3.88e-10 ***
## Tmin         -7.576e-02  1.935e-02  -3.915 9.04e-05 ***
## Summer       -4.738e-01  2.862e-02 -16.556  < 2e-16 ***
## Year          1.970e-01  1.962e-02  10.041  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4321.2  on 10505  degrees of freedom
## Residual deviance: 3561.5  on 10499  degrees of freedom
## AIC: 3575.5
##
## Number of Fisher Scoring iterations: 7
```

Evaluation

Now we can take a look at how well our model will perform based on where the cutoff for probability should be for predicting WnvPresent.

```
model.evaluate(train$WnvPresent, train$Wnv.glm.Pred)
```



```
## Area under the curve: 0.8184
```

AUC is 0.8184 on training data.

Using bins for mosquito number

I will now try to group the number of mosquitos into bins, according to quartiles. These bins will be ordinal factors. Since we know that mosquito counts can never dip below zero, I will replace negative values for predicted number of mosquitos to zero before using this in the prediction for West Nile Virus.

```
train$NumMosQuartile <- with(train, cut(NumMosquitos, breaks=quantile(train$NumMosquitos, probs=seq(0,1

test$lmNumMosq[test$lmNumMosq <= 0] <- 1

test$NumMosQuartile <- with(test, cut(lmNumMosq, breaks=quantile(train$NumMosquitos, probs=seq(0,1, by=

train$NumMosQuartile <- as.factor(train$NumMosQuartile)
test$NumMosQuartile <- as.factor(test$NumMosQuartile)
```

Modeling: Logistic regression to predict WNV infection with binned mosquito numbers

```
lgm2 <- glm(WnvPresent ~ NumMosQuartile + ma10 + WetBulb + Tmin + Summer + Year, data = train, family =

train$Wnv.glm2.Pred <- predict(lgm2, newdata = train, type = "response")
```

```r
# Use linear regression for mosquito number to approximate mosquito number in test set
test$NumMosquitos <- test$NumMosQuartile
test$Wnv.glm2.Pred <- predict(lgm2, newdata = test, type = "response")
glm.submission2 <- select(test, Id, WnvPresent = Wnv.glm2.Pred)
```

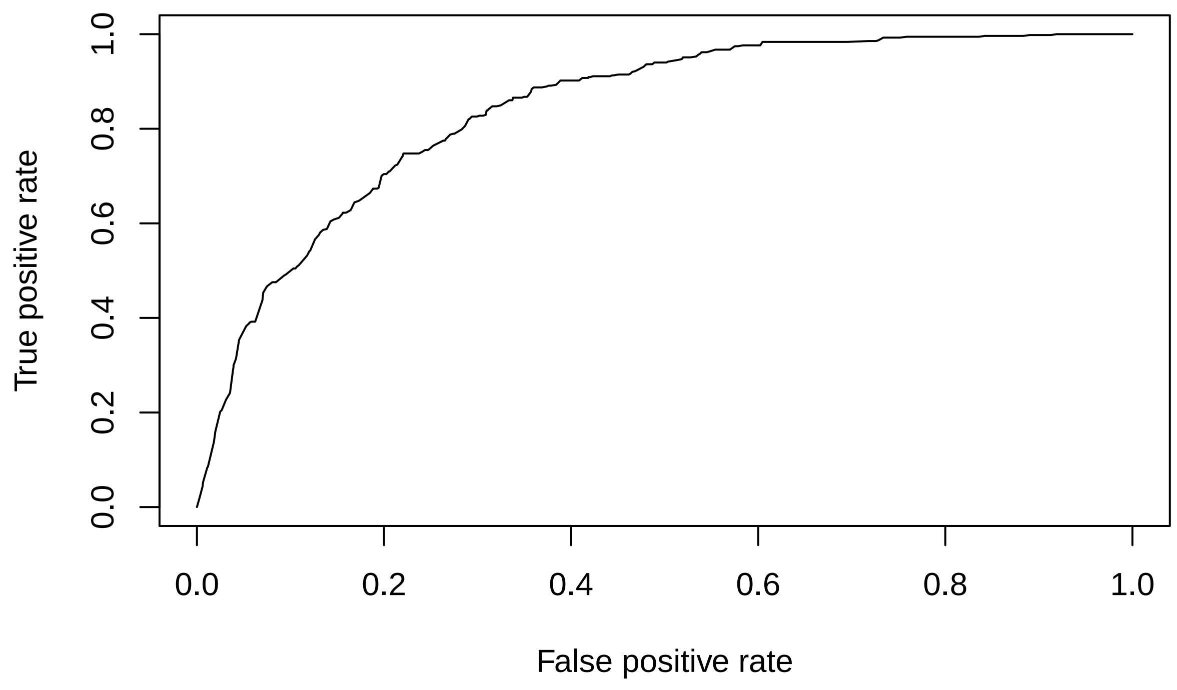
```
## Adding missing grouping variables: `Trap`, `Year`
```

```r
write_csv(glm.submission2, path = "./data/glm.submission2.csv")

summary(lgm2)
```

```
##
## Call:
## glm(formula = WnvPresent ~ NumMosQuartile + ma10 + WetBulb +
##     Tmin + Summer + Year, family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0248  -0.3429  -0.1763  -0.0858   3.5936
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -299.54122   39.02100  -7.676 1.64e-14 ***
## NumMosQuartile(2,5]     1.01709    0.24628   4.130 3.63e-05 ***
## NumMosQuartile(5,17]    1.96582    0.21362   9.202  < 2e-16 ***
## NumMosQuartile(17,50]   2.83117    0.20452  13.843  < 2e-16 ***
## ma10                   -0.07942    0.01420  -5.591 2.26e-08 ***
## WetBulb                 0.13264    0.02134   6.216 5.08e-10 ***
## Tmin                   -0.08044    0.01941  -4.145 3.40e-05 ***
## Summer                 -0.46714    0.02870 -16.274  < 2e-16 ***
## Year                    0.14838    0.01932   7.679 1.60e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4321.2  on 10505  degrees of freedom
## Residual deviance: 3440.0  on 10497  degrees of freedom
## AIC: 3458
##
## Number of Fisher Scoring iterations: 7
```

Evaluation

```r
model.evaluate(train$WnvPresent, train$Wnv.glm2.Pred)
```



```
## Area under the curve: 0.8406
```

The AUC for the test set, after submitting the predicted probabilities to Kaggle was 0.700. Since this is a bit lower than the 0.841 AUC from the training set, this model is likely overfit to the training set.

CRISP Iteration 3

Data understanding: Number of mosquitos, revisited

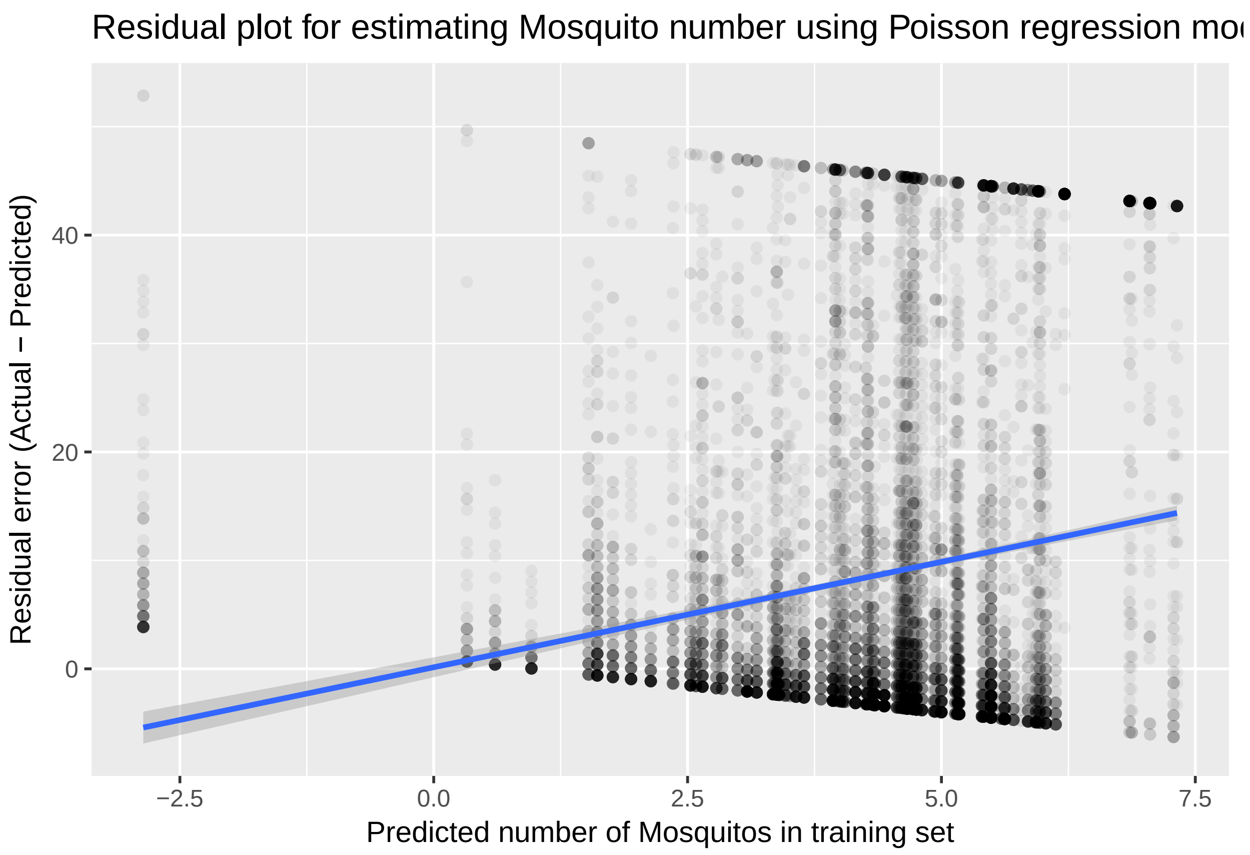
One important feature of how this data is collected and organized, is that the number of mosquitos in each trap is actually spread out over multiple rows. Therefore, we should collapse and combine multiple rows to find the number of mosquitos present in the environment. We also gain information from the separate rows mostly for the species information - some species are found in rows that are positive and not in rows that are negative from the same trap and this is useful information.

```r
fmla3 <- paste("Mosq_count", lm_variables, sep = "~")

glm2 <- glm(fmla3, data = train, family = "poisson")

train$glm2NumMosq <- predict(glm2, newdata = train)
test$glm2NumMosq <- predict(glm2, newdata = test)

summary(glm2)
```

```
##
## Call:
## glm(formula = fmla3, family = "poisson", data = train)
##
## Deviance Residuals:
```

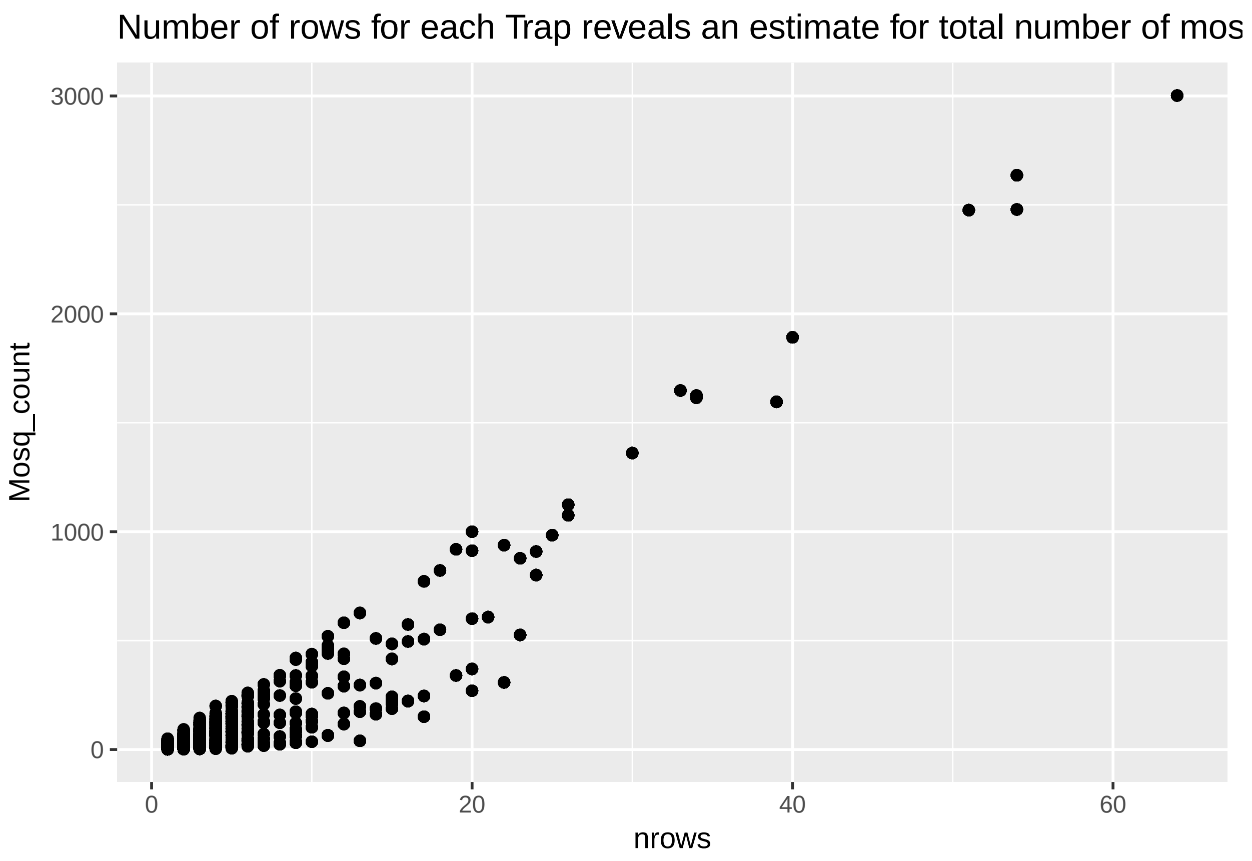
```
##     Min       1Q   Median       3Q      Max
## -53.745  -13.238   -6.646    1.230   84.041
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.418e+01  1.878e-01  -288.5   <2e-16 ***
## Tmax         5.604e-01  1.944e-03   288.3   <2e-16 ***
## Tmin         7.485e-01  1.912e-03   391.4   <2e-16 ***
## Tavg        -1.258e+00  3.766e-03  -334.1   <2e-16 ***
## DewPoint    -2.113e-01  8.359e-04  -252.8   <2e-16 ***
## WetBulb      2.280e-01  1.384e-03   164.8   <2e-16 ***
## Sunrise      2.557e-02  9.808e-05   260.8   <2e-16 ***
## Sunset       1.935e-02  7.986e-05   242.4   <2e-16 ***
## PrecipTotal -1.331e+00  5.961e-03  -223.3   <2e-16 ***
## ResultSpeed  6.789e-02  3.638e-04   186.6   <2e-16 ***
## ma3          2.239e-01  6.166e-04   363.1   <2e-16 ***
## ma5         -3.768e-01  7.061e-04  -533.6   <2e-16 ***
## ma10         2.412e-01  5.159e-04   467.5   <2e-16 ***
## precip3d    -4.021e+00  5.319e-03  -756.0   <2e-16 ***
## precip5d     3.930e+00  7.236e-03   543.1   <2e-16 ***
## precip10d    1.763e+00  5.443e-03   324.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 5712446  on 10505  degrees of freedom
## Residual deviance: 3472943  on 10490  degrees of freedom
## AIC: 3524708
##
## Number of Fisher Scoring iterations: 7
```

```r
ggplot(data = train, aes(x=glm2NumMosq, y=(NumMosquitos - glm2NumMosq))) +
  geom_point(alpha = 0.05) + stat_smooth(method = lm) +
  ggtitle("Residual plot for estimating Mosquito number using Poisson regression model on weather data"
  xlab("Predicted number of Mosquitos in training set") + ylab("Residual error (Actual - Predicted)")
```



The number of rows in the training set for each trap-date combination is correlated with total mosquitos in the trap. This feature, the number of rows for trap-date, is present in the test set as well.

```
ggplot(data = train, aes(nrows, Mosq_count)) +
  geom_point() +
  ggtitle("Number of rows for each Trap reveals an estimate for total number of mosquitos")
```



```r
cor(train$nrows, train$NumMosquitos)
```

```
## [1] 0.5686519
```

This feature was used by winning competitors to estimate the number of mosquitos. However, I feel this is a bit of cheating for the business understanding of the project which involves predicting the probability of WNV for an entire year. In actual practice, we would not know how many 'trapfulls' amounts of mosquitos will be tested in the future.

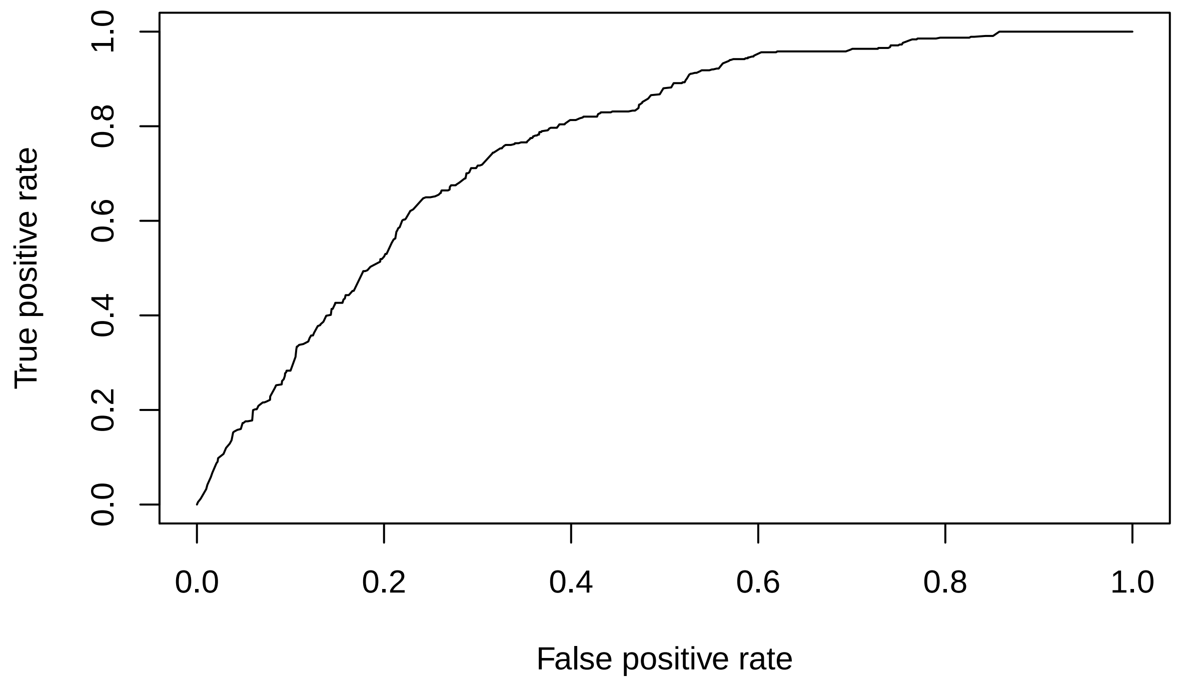
Modeling: Using number of rows as proxy for mosquito number in logistic regression

```r
lgm3 <- glm(WnvPresent ~ nrows + ma10 + WetBulb + Tmin + Summer + Year, data = train, family = binomial

train$Wnv.glm3.Pred <- predict(lgm3, newdata = train, type = "response")
test$Wnv.glm3.Pred <- predict(lgm3, newdata = test, type = "response")

summary(lgm3)
```

```
##
## Call:
## glm(formula = WnvPresent ~ nrows + ma10 + WetBulb + Tmin + Summer +
##     Year, family = binomial(link = "logit"), data = train)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7667  -0.3782  -0.2515  -0.1349   3.1624
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.706e+02  4.060e+01  -9.127  < 2e-16 ***
## nrows        2.093e-02  3.717e-03   5.630 1.80e-08 ***
## ma10        -5.963e-02  1.377e-02  -4.329 1.49e-05 ***
## WetBulb      1.224e-01  2.113e-02   5.795 6.83e-09 ***
## Tmin        -5.996e-02  1.899e-02  -3.157   0.0016 **
## Summer      -4.523e-01  2.774e-02 -16.306  < 2e-16 ***
## Year         1.835e-01  2.011e-02   9.124  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4321.2  on 10505  degrees of freedom
## Residual deviance: 3828.5  on 10499  degrees of freedom
## AIC: 3842.5
##
## Number of Fisher Scoring iterations: 7
```

```
model.evaluate(train$WnvPresent, train$Wnv.glm3.Pred)
```



```
## Area under the curve: 0.7672
```

```r
glm.submission3 <- select(test, Id, WnvPresent = Wnv.glm3.Pred)
```

```
## Adding missing grouping variables: `Trap`, `Year`
```

```r
write_csv(glm.submission3, path = "./data/glm.submission3.csv")
```

This gives an AUC of 0.711 on Kaggle.

Modeling: Boosted generalized linear classifier model

The logistic regression model might be improved with using a boosting algorithm

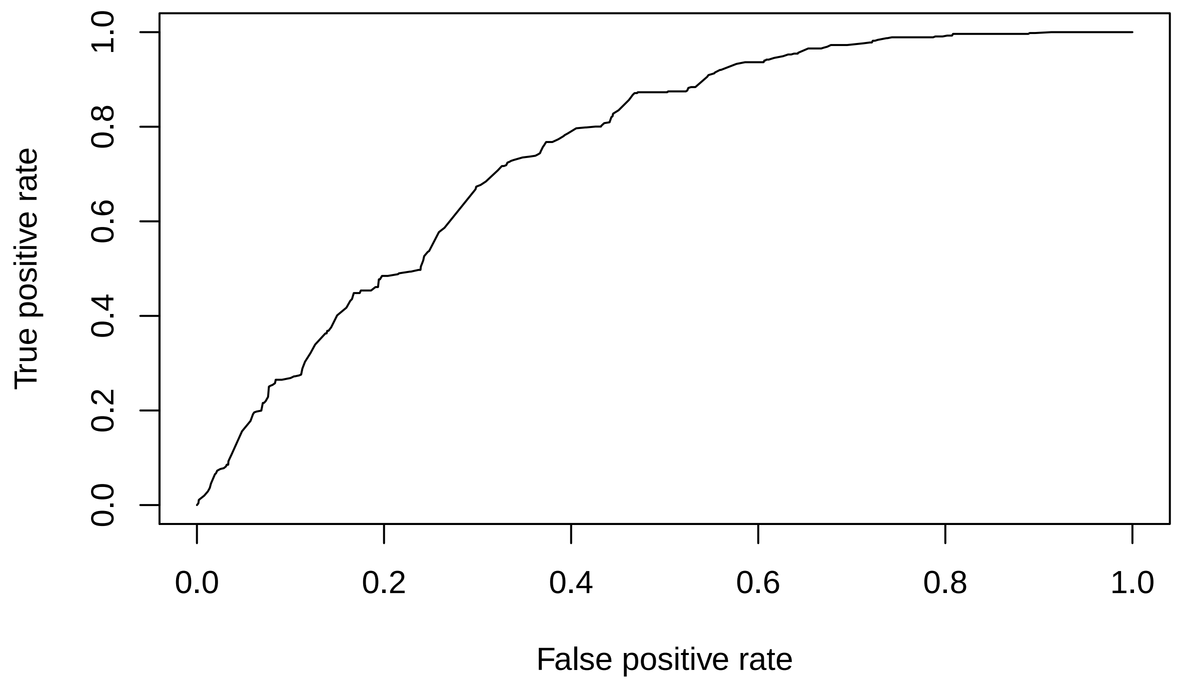
```r
suppressMessages(library(caret))

ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3,
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)

set.seed(300)
glmboost <- train(WnvPresent2 ~ nrows + precip10d + ma10 + Tmin + Summer + Year,
              data = train,
              method = "glmboost",
              metric = "ROC",
              trControl = ctrl,
              tuneLength = 5,
              center = TRUE,
              family = Binomial(link = c("logit")))
```

```
## Loading required package: mboost

## Loading required package: parallel

## Loading required package: stabs

## This is mboost 2.7-0. See 'package?mboost' and 'news(package  = "mboost")'
## for a complete list of changes.

##
## Attaching package: 'mboost'

## The following object is masked from 'package:tidyr':
##
##     extract

## The following object is masked from 'package:ggplot2':
##
##     %+%
```

```r
train$glmboost.Pred <- predict(glmboost, newdata = train, type = "prob")
test$glmboost.Pred <- predict(glmboost, newdata = test, type = "prob")
test$glmboost.Pred2 <- test$glmboost.Pred[,2]
glm.submission4 <- select(test, Id, WnvPresent = glmboost.Pred2)
```

```
## Adding missing grouping variables: `Trap`, `Year`
```

```r
write_csv(glm.submission4, path = "./data/glm.submission4.csv")
model.evaluate(train$WnvPresent, train$glmboost.Pred[,2])
```



```
## Area under the curve: 0.751
```

Glmboost delivers 0.720 AUC on Kaggle.

CRISP Iteration 4

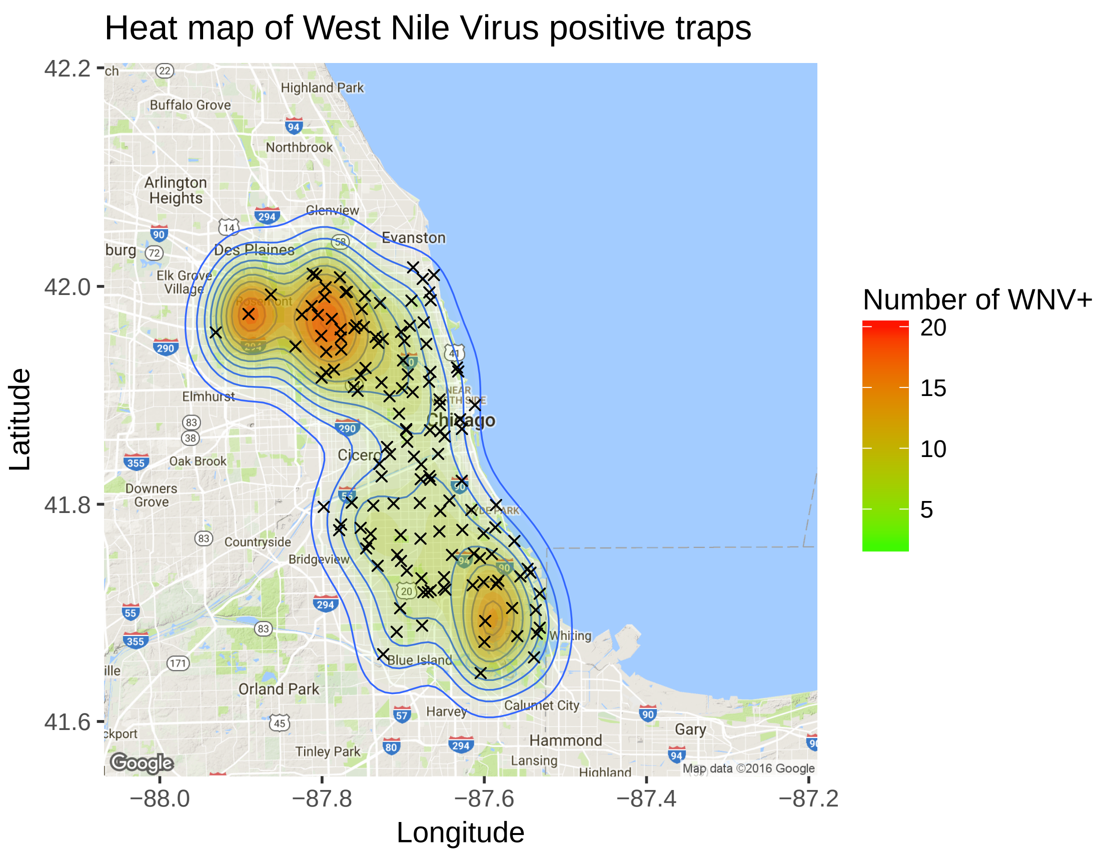
Business/data understanding: Effect of geography and location

The first set of models did not involve geographical information or any trap-specific information. There is likely to be an effect based on a trap's geographic location. Next, I wanted to visually examine the relationship between geography and West Nile presence. First I created a map of Wnv incidents as a density plot. Here, the density is based off of the overall positive count for each trap.

```r
suppressMessages(library(ggmap))
chicago <- get_map("Chicago")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Chicago&zoom=10&size=640x640&sca
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Chicago&sensor=false
```

```
wnpositive <- filter(train, WnvPresent == TRUE)
load("./data/traps.RData")

ggmap(chicago) + geom_density2d(data = wnpositive, aes(x = Longitude, y = Latitude), size = 0.3) +
  stat_density2d(data = wnpositive, aes(x = Longitude, y = Latitude, fill = ..level.., alpha = ..level.
  scale_fill_gradient(name="Number of WNV+", low = "green", high = "red") +
  scale_alpha(range = c(0, 0.3), guide = FALSE) +
  ggtitle("Heat map of West Nile Virus positive traps") +
  xlab("Longitude") + ylab("Latitude") +
  geom_point(data = traps, aes(x = Longitude, y = Latitude), shape = 4)
```



It’s clear from the density map that there are hotspot traps where West Nile is more common than others, and this appears to influence nearby traps.

Now that we have a good 2-dimensional view of the occurance of WNV, I next wanted to get a sense of how this pattern of West Nile Virus changed over time in relation to the geography. I did this by creating an animated version of this map using a new R package called gganimate. I chose to remove the background map of chicago to get a clearer visual of positive vs negative cases. Animating the map requires too much time to rebuild in the document, but the code can be found in the src folder called “07_animated_map.R”

After seeing how highly varied the density of WnvPresent traps were in the heat map and animated plots, my next set of models will try to incorporate this regional geographic information.

There are various ways to incorporate geography. I could incorporate the trap id in the model or I could incorporate the longitude and latitude for very high precision. The longitude/latitude method would require a decision tree or other model that allows for breaking up or clustering the locations in some non-linear way.

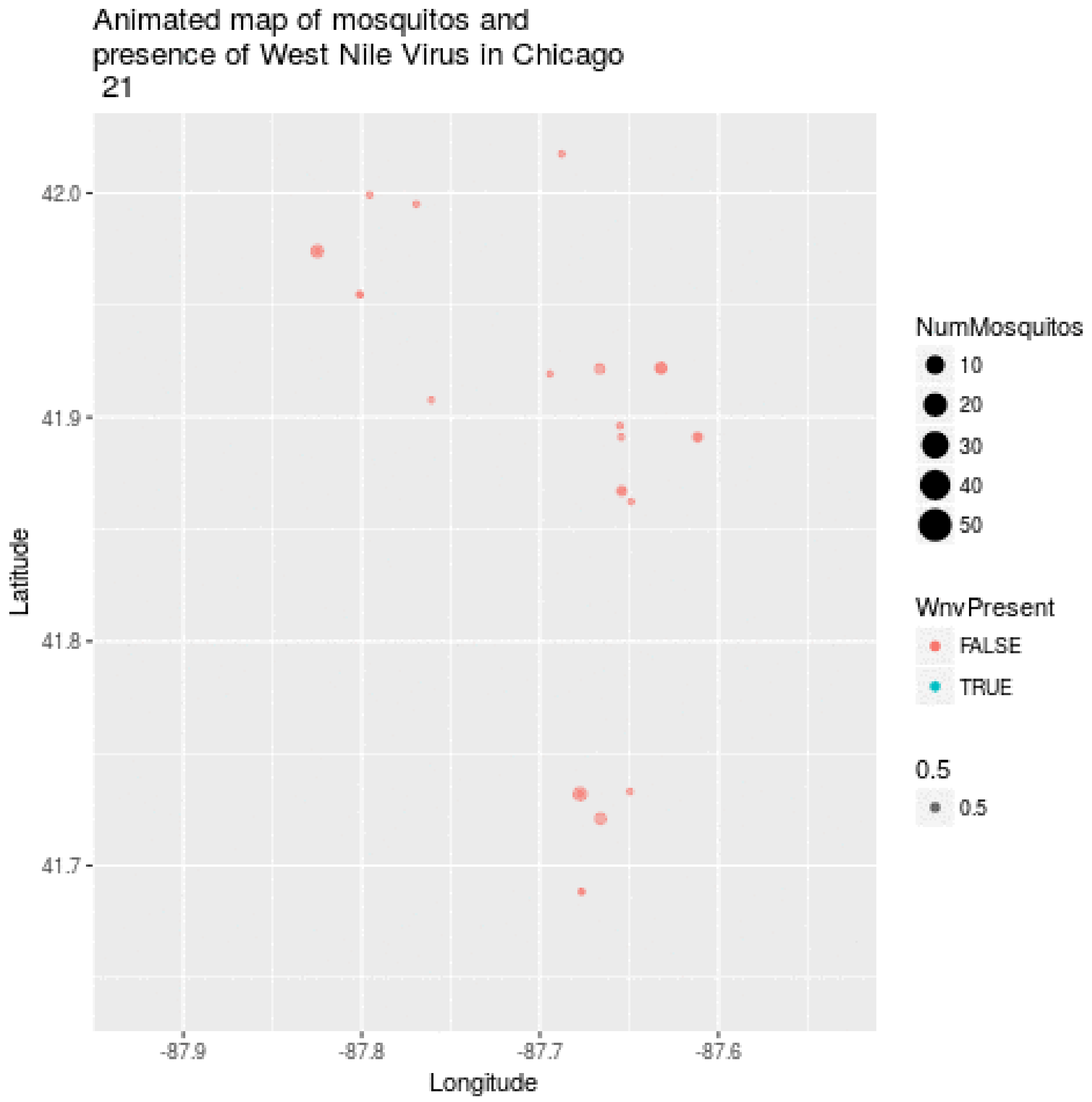


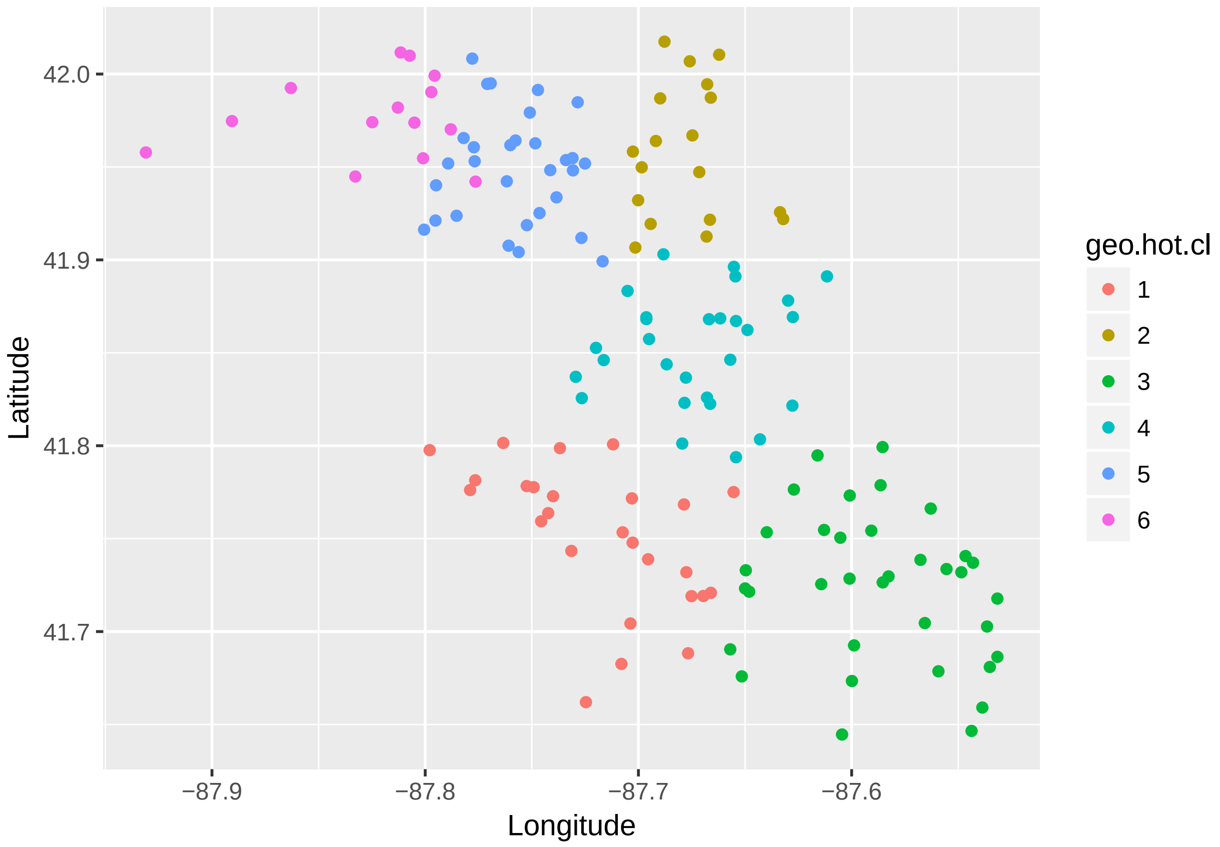
Figure 1: Animated map (not available in PDF document)

Instead, I am choosing to group the traps into clusters and include the cluster as a modeling parameter. The reasoning for this is that suspect that the true risk based on geography has a much lower actual precision than our ability to define latitude or longitude.

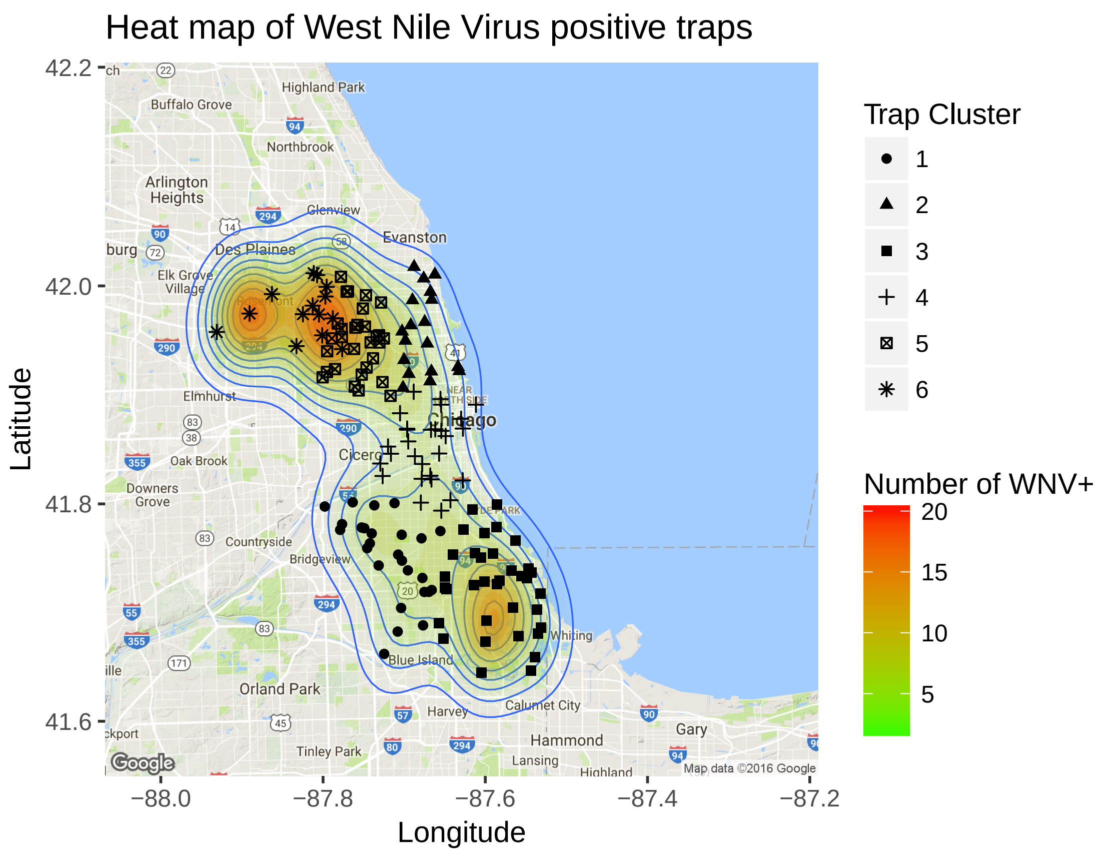
Data preparation: Clustering traps according to location and WNV presense

To create clusters of traps I want to incorporate not only geographical information, but WNV rate as well. This will help to create breaks in my clusters that are informative for predicting future WNV rate. The code for clustering is found in the "06_geo_cluster.R" script.

```r
load("./data/train_geo.RData")
load("./data/test_geo.RData")
load("./data/trap_geo_hot.RData")

ggplot(trap_geo_hot, aes(Longitude, Latitude, color = geo.hot.cl)) + geom_point()
```



```r
ggmap(chicago) + geom_density2d(data = wnpositive, aes(x = Longitude, y = Latitude), size = 0.3) +
  stat_density2d(data = wnpositive, aes(x = Longitude, y = Latitude, fill = ..level.., alpha = ..level.
    bins = 16, geom = "polygon") +
  scale_fill_gradient(name="Number of WNV+", low = "green", high = "red") +
  scale_alpha(range = c(0, 0.3), guide = FALSE) +
  ggtitle("Heat map of West Nile Virus positive traps") +
  xlab("Longitude") + ylab("Latitude") +
  geom_point(data = trap_geo_hot, aes(x = Longitude, y = Latitude, shape = geo.hot.cl)) + scale_shape_d
```



Modeling: Geo coordinates and running total of nrows

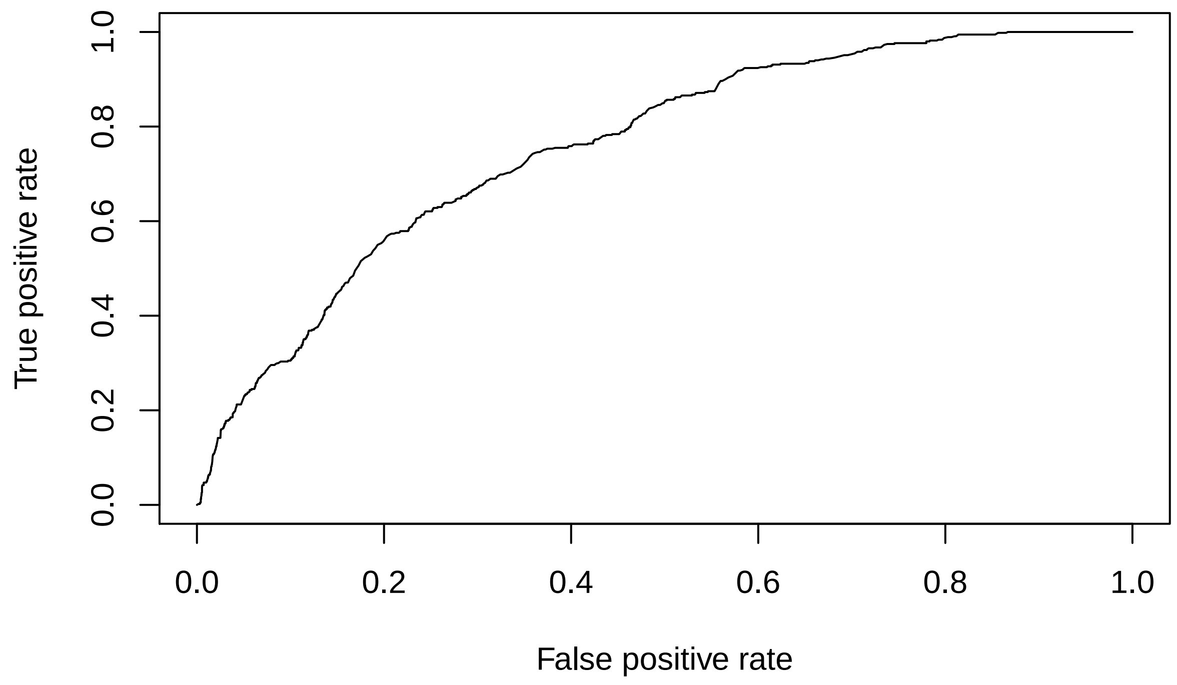
This final model will incorporate the geographic regions and the recent history of mosquito populations as estimated by the number of rows of observations in the training and test sets.

```r
suppressMessages(library(caret))

ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 3,
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)

set.seed(300)
glmboost2 <- train(WnvPresent2 ~ nrows + nrow.3d + nrow.5d + nrow.10d + ma10 + WetBulb + Summer + geo.he
              data = train_geo,
              method = "glmboost",
              metric = "ROC",
              trControl = ctrl,
              tuneLength = 5,
              center = TRUE,
              family = Binomial(link = c("logit")))

train_geo$glmboost2.Pred <- predict(glmboost2, newdata = train_geo, type = "prob")
test_geo$glmboost2.Pred <- predict(glmboost2, newdata = test_geo, type = "prob")
```

```r
test_geo$glmboost2.Pred2 <- test_geo$glmboost2.Pred[,2]
glm.submission5 <- select(test_geo, Id, WnvPresent = glmboost2.Pred2)
```

```
## Adding missing grouping variables: `Trap`, `Year`
```

```r
write_csv(glm.submission5, path = "./data/glm.submission5.csv")

train_geo$glmboost2.Pred <- train_geo$glmboost2.Pred[,2]
model.evaluate(train_geo$WnvPresent, train_geo$glmboost2.Pred)
```



```
## Area under the curve: 0.7573
```

This model achieves an AUC score of 0.715 on Kaggle.

Final ensembling and deployment

I extracted the probabilities obtained by each model and then averaged them using excel. This submission achieved a 0.715 AUC, which is not as good as the first glmboosted model. It's possible that these models are not varied enough in their type to benefit from ensembling. They are all generalized linear models. I attempted to use a C5.0 decision tree classifier and random forest models, but ran in to some difficulty as they kept giving me identical predictions for every observation in the test set. I think that part of the reason was due to the imbalance of the positive class. This can be mediated by oversampling positive cases or undersampling negative cases and building a model from such an artificially balanced data set.

```r
models <- test %>% select(Id, Wnv.glm.Pred, Wnv.glm2.Pred, Wnv.glm3.Pred, glmboost.Pred2)
```

```
## Adding missing grouping variables: `Trap`, `Year`
```

```
geo_model <- test_geo %>% select(Id, glmboost2.Pred2)
```

```
## Adding missing grouping variables: `Trap`, `Year`
```

```
models <- left_join(models, geo_model, by = "Id")

write_csv(models, path = "./data/final_submission.csv")
```

```
sessionInfo()
```

```
## R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] ggmap_2.6.1     mboost_2.7-0    stabs_0.5-1     caret_6.0-72
##  [5] lattice_0.20-34 pROC_1.8        ROCR_1.0-7      gplots_3.0.1
##  [9] dplyr_0.5.0     purrr_0.2.2     readr_1.0.0     tidyr_0.6.0
## [13] tibble_1.2      ggplot2_2.2.0   tidyverse_1.0.0 plyr_1.8.4
##
## loaded via a namespace (and not attached):
##  [1] maps_3.1.1         splines_3.3.1      foreach_1.4.3
##  [4] gtools_3.5.0       assertthat_0.1     sp_1.2-3
##  [7] stats4_3.3.1       coin_1.1-3         yaml_2.1.14
## [10] backports_1.0.4    quantreg_5.29      quadprog_1.5-5
## [13] digest_0.6.10      minqa_1.2.4        colorspace_1.2-6
## [16] sandwich_2.3-4     htmltools_0.3.5    Matrix_1.2-7.1
## [19] SparseM_1.72       mvtnorm_1.0-5      scales_0.4.1
## [22] gdata_2.17.0       jpeg_0.1-8         lme4_1.1-12
## [25] MatrixModels_0.4-1 mgcv_1.8-15        car_2.1-3
## [28] party_1.1-2        TH.data_1.0-7      nnet_7.3-12
## [31] lazyeval_0.2.0     proto_1.0.0        pbkrtest_0.4-6
## [34] survival_2.40-1    strucchange_1.5-1  magrittr_1.5
## [37] evaluate_0.10      nlme_3.1-128       MASS_7.3-45
## [40] tools_3.3.1        geosphere_1.5-5    RgoogleMaps_1.4.1
## [43] multcomp_1.4-6     stringr_1.1.0      munsell_0.4.3
## [46] compiler_3.3.1     caTools_1.17.1     grid_3.3.1
## [49] nloptr_1.0.4       iterators_1.0.8    rjson_0.2.15
## [52] bitops_1.0-6       labeling_0.3       rmarkdown_1.2
## [55] gtable_0.2.0       ModelMetrics_1.1.0 codetools_0.2-15
## [58] DBI_0.5-1          reshape2_1.4.2     R6_2.2.0
## [61] nnls_1.4           zoo_1.7-13         knitr_1.15.1
## [64] rprojroot_1.1      KernSmooth_2.23-15 modeltools_0.2-21
## [67] stringi_1.1.2      Rcpp_0.12.8        mapproj_1.2-4
## [70] png_0.1-7
```