DS4SI Final

Andrew Cooke

12/3/2021

R Markdown

```
library("broom")
library("ggplot2")
library("MASS")
library("gtable")
library("gridExtra")
library("stargazer")
library("dplyr")
library("corrplot")
library("RColorBrewer")
library("pscl")
library("smotefamily")
library("ROCR")
```

```
jlm_con <- read.csv("rwm_replic_data.csv")
```

These are the original models fitted in the paper. They will be used as a bench mark for the subsequent models created in this document.

```
# original models for individual and collective from paper
ind <- glm.nb(Individual ~ Mean_800 +
                Jewish_segment + JLR_station +
                Damascus_Gate_dis + With_settlements
              ,
              data=jlm_con)
summary(ind)
```

```
##
## Call:
## glm.nb(formula = Individual ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements, data = jlm_con, init.theta = 0.2119028013,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0268  -0.4003  -0.3108  -0.2604   4.5940
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.049e+00  6.506e-01  -3.149  0.00164 **
## Mean_800           2.669e-03  9.461e-04   2.821  0.00479 **
## Jewish_segment    -8.101e-01  4.050e-01  -2.000  0.04551 *
```

```
## JLR_station        1.602e+00  6.146e-01   2.606  0.00916 **
## Damascus_Gate_dis -1.337e-04  9.811e-05  -1.363  0.17300
## With_settlements   1.371e+00  5.953e-01   2.303  0.02130 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.2119) family taken to be 1)
##
##     Null deviance: 235.85  on 495  degrees of freedom
## Residual deviance: 136.76  on 490  degrees of freedom
##   (35 observations deleted due to missingness)
## AIC: 356.87
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.2119
##           Std. Err.:  0.0702
##
##  2 x log-likelihood:  -342.8710
```

```
col <- glm.nb(Collective ~ Mean_800 +
                Jewish_segment + JLR_station +
                Damascus_Gate_dis + With_settlements
              ,
              data=jlm_con)
summary(col)
```

```
##
## Call:
## glm.nb(formula = Collective ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements, data = jlm_con, init.theta = 0.05411806977,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9138  -0.6202  -0.5568  -0.4671   4.4100
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        4.1599387  0.7674618   5.420 5.95e-08 ***
## Mean_800          -0.0030227  0.0013386  -2.258 0.023942 *
## Jewish_segment    -1.4216306  0.4284490  -3.318 0.000906 ***
## JLR_station       -0.8330453  1.0751905  -0.775 0.438465
## Damascus_Gate_dis -0.0004761  0.0001059  -4.496 6.91e-06 ***
## With_settlements   1.8072408  0.9292998   1.945 0.051807 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.0541) family taken to be 1)
##
##     Null deviance: 233.52  on 495  degrees of freedom
## Residual deviance: 185.18  on 490  degrees of freedom
##   (35 observations deleted due to missingness)
## AIC: 1006.4
```

```
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.05412
##           Std. Err.:  0.00709
##
##  2 x log-likelihood:  -992.37300
```

Analysis 1: New Model Features

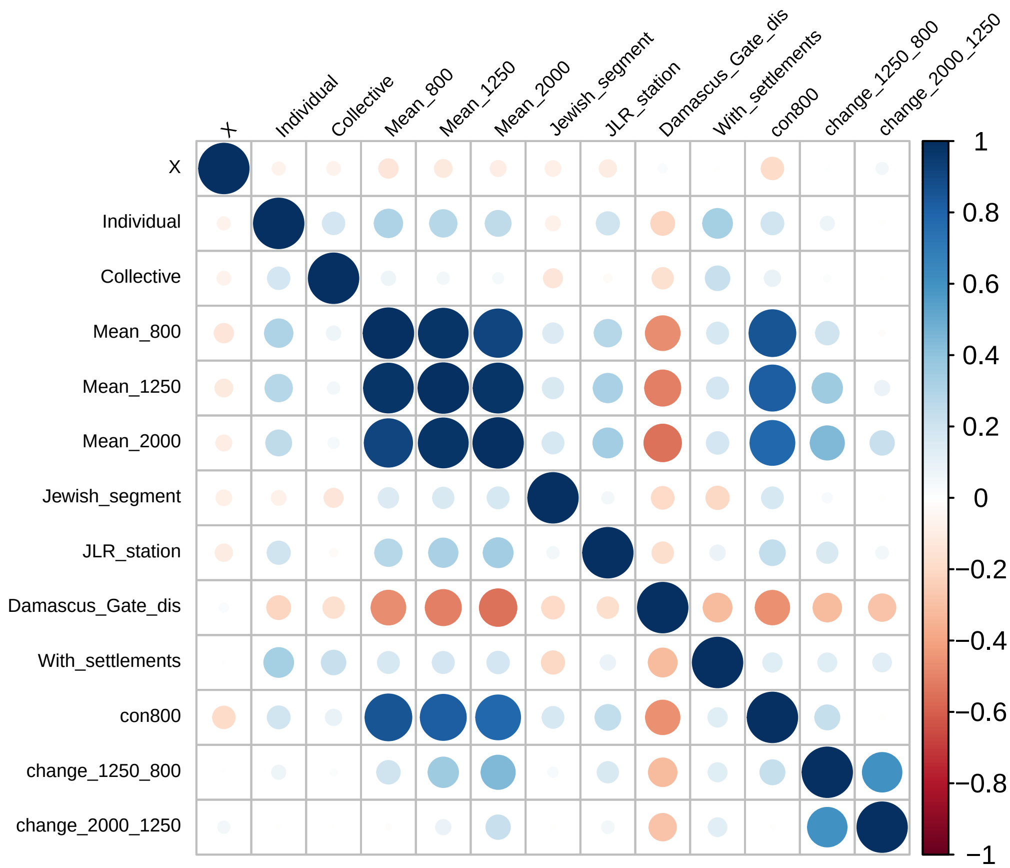
The original models use only one of Mean_800, Mean_1250, and Mean_2000 because they are highly correlated connectivity measurements only differing by the change of a hyperparameter. To capture the information from the difference in each of these while not introducing any more multicollinearity, we will add percentage change from Mean_800 to Mean_1250 and Mean_1250 to Mean_2000.

```
# For each column, take each null value and replace with the mean
# avoid future errors in modeling and improve usability
for (i in colnames(jlm_con)) {
  jlm_con[[i]][is.na(jlm_con[[i]])]<-mean(jlm_con[[i]], na.rm = TRUE)
  print(i)
}
```

```
## [1] "X"
## [1] "Individual"
## [1] "Collective"
## [1] "Mean_800"
## [1] "Mean_1250"
## [1] "Mean_2000"
## [1] "Jewish_segment"
## [1] "JLR_station"
## [1] "Damascus_Gate_dis"
## [1] "With_settlements"
## [1] "con800"
```

```
# percentage change
jlm_con <- jlm_con %>%
  mutate(change_1250_800 = (Mean_1250 - Mean_800)/Mean_800,
         change_2000_1250 = (Mean_2000 - Mean_1250)/Mean_1250)
```

```
M <-cor(jlm_con) # correlation table
corrplot(M, tl.col = "black",
         tl.cex = 0.6, tl.srt = 45) # create plot and format
```




```r
# Individual with new features
change_ind <- glm.nb(Individual ~ Mean_800 +
                       Jewish_segment + JLR_station +
                       Damascus_Gate_dis + With_settlements +
                       change_1250_800 + change_2000_1250,
                     data=jlm_con)
summary(change_ind)
```

```
##
## Call:
## glm.nb(formula = Individual ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 +
##     change_2000_1250, data = jlm_con, init.theta = 0.2297031878,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1745  -0.3791  -0.2781  -0.2068   3.9709
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.5244646  0.9456932  -2.669 0.007598 **
## Mean_800           0.0022099  0.0009872   2.238 0.025189 *
## Jewish_segment    -0.7249856  0.4153377  -1.746 0.080892 .
## JLR_station        1.2859580  0.6189097   2.078 0.037730 *
## Damascus_Gate_dis -0.0001769  0.0001028  -1.721 0.085261 .
```

```
## With_settlements   1.5757928  0.5950643   2.648 0.008094 **
## change_1250_800    1.8918201  0.5461205   3.464 0.000532 ***
## change_2000_1250  -1.2225095  0.5535674  -2.208 0.027215 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.2297) family taken to be 1)
##
##     Null deviance: 254.58  on 530  degrees of freedom
## Residual deviance: 134.76  on 523  degrees of freedom
## AIC: 354.77
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.2297
##           Std. Err.:  0.0744
##
##  2 x log-likelihood:  -336.7710
```

```r
# Collective with new features
change_col <- glm.nb(Collective ~ Mean_800 +
                       Jewish_segment + JLR_station +
                       Damascus_Gate_dis + With_settlements +
                       change_1250_800 + change_2000_1250,
                     data=jlm_con)
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: algorithm did not converge

## Warning in glm.nb(Collective ~ Mean_800 + Jewish_segment + JLR_station + :
## alternation limit reached
```

```r
summary(change_col)
```

```
##
## Call:
## glm.nb(formula = Collective ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 +
##     change_2000_1250, data = jlm_con, init.theta = 0.05178554773,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9176  -0.5978  -0.5250  -0.4545   4.3498
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        5.3329241  1.0178363   5.239 1.61e-07 ***
## Mean_800          -0.0040215  0.0014031  -2.866 0.004155 **
## Jewish_segment    -1.4131564  0.4238599  -3.334 0.000856 ***
## JLR_station       -0.2225585  1.0996204  -0.202 0.839607
## Damascus_Gate_dis -0.0005173  0.0001078  -4.801 1.58e-06 ***
## With_settlements   2.0977664  0.9473417   2.214 0.026803 *
## change_1250_800   -0.1231852  0.6034244  -0.204 0.838241
```

```
## change_2000_1250  -0.7220507  0.4447644  -1.623 0.104494
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.0518) family taken to be 1)
##
##     Null deviance: 240.98  on 530  degrees of freedom
## Residual deviance: 187.28  on 523  degrees of freedom
## AIC: 1018.7
##
## Number of Fisher Scoring iterations: 1
##
##
##                Theta:  0.05179
##            Std. Err.:  0.00681
## Warning while fitting theta: alternation limit reached
##
##  2 x log-likelihood:  -1000.75000
```

```
AIC(ind, change_ind)
```

```
## Warning in AIC.default(ind, change_ind): models are not all fitted to the same
## number of observations
```

```
##            df      AIC
## ind         7 356.8709
## change_ind  9 354.7711
```

```
AIC(col, change_col)
```

```
## Warning in AIC.default(col, change_col): models are not all fitted to the same
## number of observations
```

```
##            df      AIC
## col         7 1006.373
## change_col  9 1018.750
```

The new features slightly improved the individual model, but the collective model had a higher AIC without the new features. Now we will remove insignificant features from the models.

```
# include significant variables only
change_ind_s <- glm.nb(Individual ~ Mean_800 +
                       JLR_station +
                       With_settlements +
                       change_1250_800 + change_2000_1250,
                     data=jlm_con)
summary(change_ind_s)
```

```
##
## Call:
## glm.nb(formula = Individual ~ Mean_800 + JLR_station + With_settlements +
##     change_1250_800 + change_2000_1250, data = jlm_con, init.theta = 0.1882846099,
##     link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0697  -0.3727  -0.2967  -0.2224   3.3827
##
```

```
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)    
## (Intercept)      -4.0724018  0.5890602  -6.913 4.73e-12 ***
## Mean_800          0.0030341  0.0008494   3.572 0.000354 ***
## JLR_station       1.3759807  0.6492114   2.119 0.034051 *  
## With_settlements  2.1074658  0.5593996   3.767 0.000165 ***
## change_1250_800   1.9749739  0.5594448   3.530 0.000415 ***
## change_2000_1250 -1.1556702  0.5448410  -2.121 0.033912 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for Negative Binomial(0.1883) family taken to be 1)
## 
##     Null deviance: 232.96  on 530  degrees of freedom
## Residual deviance: 131.11  on 525  degrees of freedom
## AIC: 357.38
## 
## Number of Fisher Scoring iterations: 1
## 
## 
##               Theta:  0.1883 
##           Std. Err.:  0.0575 
## 
##  2 x log-likelihood:  -343.3810
```

```r
change_col_s <- glm.nb(Collective ~ Mean_800 +
                       Jewish_segment +
                       Damascus_Gate_dis + With_settlements
                       ,
                     data=jlm_con)
```

```
## Warning: glm.fit: algorithm did not converge
```

```r
summary(change_col_s)
```

```
## 
## Call:
## glm.nb(formula = Collective ~ Mean_800 + Jewish_segment + Damascus_Gate_dis + 
##     With_settlements, data = jlm_con, init.theta = 0.05081959334, 
##     link = log)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -0.9112  -0.5975  -0.5346  -0.4434   4.3408  
## 
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)    
## (Intercept)        4.4813722  0.7748360   5.784 7.31e-09 ***
## Mean_800          -0.0038581  0.0013333  -2.894 0.003807 ** 
## Jewish_segment    -1.4119594  0.4286968  -3.294 0.000989 ***
## Damascus_Gate_dis -0.0005312  0.0001040  -5.110 3.22e-07 ***
## With_settlements   2.0688365  0.9549365   2.166 0.030276 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```
## (Dispersion parameter for Negative Binomial(0.0508) family taken to be 1)
##
##     Null deviance: 237.34  on 530  degrees of freedom
## Residual deviance: 186.57  on 526  degrees of freedom
## AIC: 1014.7
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  0.05082
##           Std. Err.:  0.00665
##
##  2 x log-likelihood:  -1002.73200
```

```
AIC(ind, change_ind, change_ind_s)
```

```
## Warning in AIC.default(ind, change_ind, change_ind_s): models are not all fitted
## to the same number of observations
```

```
##              df      AIC
## ind           7 356.8709
## change_ind    9 354.7711
## change_ind_s  7 357.3811
```

```
AIC(col, change_col, change_col_s)
```

```
## Warning in AIC.default(col, change_col, change_col_s): models are not all fitted
## to the same number of observations
```

```
##              df      AIC
## col           7 1006.373
## change_col    9 1018.750
## change_col_s  6 1014.732
```

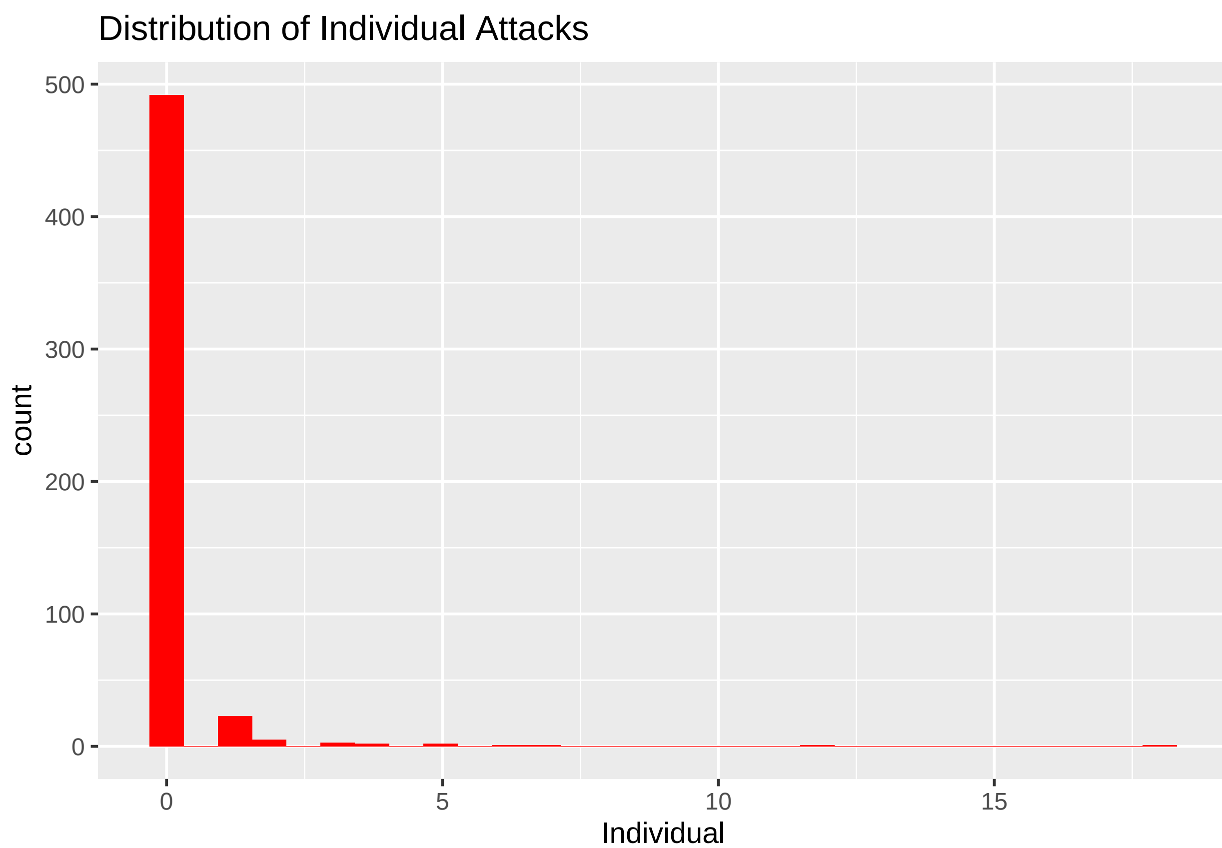
AIC increased for both models despite removing insignificant predictors.

Analysis 2: Zero inflation

Some count datasets have a higher than expected proportion of 0 values. This muddies the assumptions of traditional negative binomial and poisson regression. When this occurs, we need to use a zero inflated model. This is essentially an aggregation of two separate models: one is a logistic model that predicts whether or not there will be a zero count and then a poisson or negative binomial regression when there is a non-zero count. Let's see if a zero inflated model is needed.

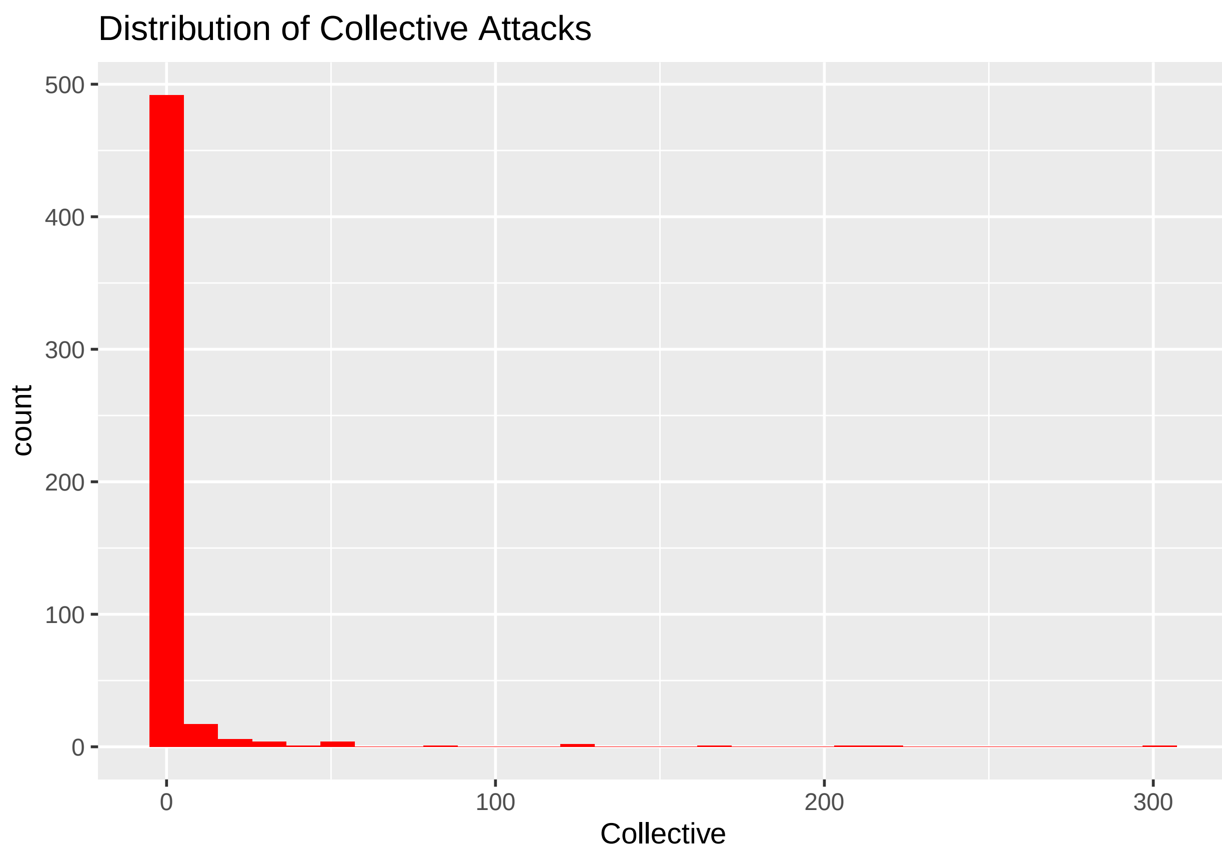
```
# create histogram with number of individual attacks
p <- ggplot(jlm_con, aes(Individual)) +
  geom_histogram(fill = "red") +
  labs(title = "Distribution of Individual Attacks")
p
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# create histogram with number of collective attacks
p <- ggplot(jlm_con, aes(Collective)) +
  geom_histogram(fill = "red") +
  labs(title = "Distribution of Collective Attacks")
p
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Both target variables show a glaring need for a zero inflated model. We will also add a zero inflated poisson model for comparison.

```
# Zero inflated NB
m1 <- zeroinfl(Individual ~ Mean_800 +
                 Jewish_segment + JLR_station +
                 Damascus_Gate_dis + With_settlements +
                 change_1250_800 + change_2000_1250,
               data = jlm_con, dist = "negbin")
```

```
## Warning in sqrt(diag(vc)[np]): NaNs produced
```

```
summary(m1)
```

```
## Warning in sqrt(diag(object$vcov)): NaNs produced

##
## Call:
## zeroinfl(formula = Individual ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 + change_2000_1250,
##     data = jlm_con, dist = "negbin")
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -0.81956 -0.20773 -0.14063 -0.07984 27.06199
##
## Count model coefficients (negbin with log link):
##                       Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)       -1.1247299        NaN     NaN      NaN
## Mean_800           0.0024542  0.0004620   5.312 1.08e-07 ***
## Jewish_segment    -0.9090768  0.4045777  -2.247 0.024641 *
## JLR_station       -0.0420230  0.5152271  -0.082 0.934995
## Damascus_Gate_dis  0.0001217        NaN     NaN      NaN
## With_settlements   0.9121265  0.3822321   2.386 0.017018 *
## change_1250_800    2.8369914  0.7923206   3.581 0.000343 ***
## change_2000_1250  -2.6933362  0.7146039  -3.769 0.000164 ***
## Log(theta)         0.3648201        NaN     NaN      NaN
##
## Zero-inflation model coefficients (binomial with logit link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.7050966  1.0478004  -0.673   0.5010
## Mean_800           0.0005144  0.0013589   0.379   0.7050
## Jewish_segment    -0.7905150  0.6778581  -1.166   0.2435
## JLR_station       -3.0616312  1.3424286  -2.281   0.0226 *
## Damascus_Gate_dis  0.0005931        NaN     NaN      NaN
## With_settlements  -1.6607999  0.9855475  -1.685   0.0920 .
## change_1250_800    2.4691178  1.1094564   2.226   0.0260 *
## change_2000_1250  -1.9952472  1.1719382  -1.703   0.0887 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 1.4403
## Number of iterations in BFGS optimization: 28
## Log-likelihood: -151.8 on 17 Df
```

```r
m2 <- zeroinfl(Collective ~ Mean_800 + #Why does collective have NA's?
                 Jewish_segment + JLR_station +
                 Damascus_Gate_dis + With_settlements +
                 change_1250_800 + change_2000_1250,
               data = jlm_con, dist = "negbin")
```

```
## Warning in value[[3L]](cond): system is computationally singular: reciprocal
## condition number = 4.21449e-43FALSE
```

```r
summary(m2)
```

```
##
## Call:
## zeroinfl(formula = Collective ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 + change_2000_1250,
##     data = jlm_con, dist = "negbin")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -0.2275 -0.2239 -0.2195 -0.2118 42.7753
##
## Count model coefficients (negbin with log link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        5.3330095         NA      NA       NA
## Mean_800          -0.0040211         NA      NA       NA
## Jewish_segment    -1.4132460         NA      NA       NA
## JLR_station       -0.2225553         NA      NA       NA
## Damascus_Gate_dis -0.0005173         NA      NA       NA
```

```
## With_settlements   2.0977372         NA      NA       NA
## change_1250_800   -0.1232510         NA      NA       NA
## change_2000_1250  -0.7220523         NA      NA       NA
## Log(theta)        -2.9606425         NA      NA       NA
##
## Zero-inflation model coefficients (binomial with logit link):
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.186677         NA      NA       NA
## Mean_800          -0.006662         NA      NA       NA
## Jewish_segment     0.604700         NA      NA       NA
## JLR_station       -0.463005         NA      NA       NA
## Damascus_Gate_dis -0.118919         NA      NA       NA
## With_settlements  -2.084917         NA      NA       NA
## change_1250_800   -0.121940         NA      NA       NA
## change_2000_1250   0.213955         NA      NA       NA
##
## Theta = 0.0518
## Number of iterations in BFGS optimization: 28
## Log-likelihood: -500.4 on 17 Df
```

```
# Zero inflated poisson
m3 <- zeroinfl(Individual ~ Mean_800 +
                 Jewish_segment + JLR_station +
                 Damascus_Gate_dis + With_settlements +
                 change_1250_800 + change_2000_1250,
               data = jlm_con, dist = "poisson")
summary(m3)
```

```
## Warning in sqrt(diag(object$vcov)): NaNs produced
```

```
##
## Call:
## zeroinfl(formula = Individual ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 + change_2000_1250,
##     data = jlm_con, dist = "poisson")
##
## Pearson residuals:
##      Min       1Q   Median       3Q      Max
## -1.17089 -0.21094 -0.14042 -0.08624 23.70977
##
## Count model coefficients (poisson with log link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.4528200  0.2180741  -2.076  0.03785 *
## Mean_800           0.0023077  0.0003035   7.605 2.86e-14 ***
## Jewish_segment    -0.8566942  0.3027506  -2.830  0.00466 **
## JLR_station       -0.6397092  0.2673343  -2.393  0.01671 *
## Damascus_Gate_dis  0.0001203        NaN     NaN      NaN
## With_settlements   0.5976990  0.2742446   2.179  0.02930 *
## change_1250_800    2.3233378  0.4507267   5.155 2.54e-07 ***
## change_2000_1250  -2.2560454  0.7565157  -2.982  0.00286 **
##
## Zero-inflation model coefficients (binomial with logit link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.3112868  0.8487250   0.367   0.7138
## Mean_800          -0.0002960  0.0010672  -0.277   0.7815
```

```
## Jewish_segment    -0.6502491  0.5551071  -1.171   0.2414
## JLR_station       -3.2235829  1.0898188  -2.958   0.0031 **
## Damascus_Gate_dis  0.0005133        NaN     NaN      NaN
## With_settlements  -1.6910422  0.7424192  -2.278   0.0227 *
## change_1250_800    1.6193549  0.9307211   1.740   0.0819 .
## change_2000_1250  -1.1212935  1.0874855  -1.031   0.3025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 20
## Log-likelihood: -160.8 on 16 Df
```

```r
m4 <- zeroinfl(Collective ~ Mean_800 +
                 Jewish_segment + JLR_station +
                 Damascus_Gate_dis + With_settlements +
                 change_1250_800 + change_2000_1250,
               data = jlm_con, dist = "poisson")
summary(m4)
```

```
## Warning in sqrt(diag(object$vcov)): NaNs produced

##
## Call:
## zeroinfl(formula = Collective ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 + change_2000_1250,
##     data = jlm_con, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.9511 -0.3972 -0.2802 -0.2053 77.7781
##
## Count model coefficients (poisson with log link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        5.5268676  0.0782230  70.655  < 2e-16 ***
## Mean_800          -0.0016326  0.0001340 -12.183  < 2e-16 ***
## Jewish_segment    -1.1637368  0.0602026 -19.330  < 2e-16 ***
## JLR_station       -1.7560651  0.2190078  -8.018 1.07e-15 ***
## Damascus_Gate_dis -0.0002187        NaN     NaN      NaN
## With_settlements   0.2701551  0.0545328   4.954 7.27e-07 ***
## change_1250_800    0.1519943  0.0772209   1.968    0.049 *
## change_2000_1250  -0.9518285  0.0751366 -12.668  < 2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.327e-01  6.383e-01  -0.208   0.8353
## Mean_800          -1.738e-04  8.779e-04  -0.198   0.8431
## Jewish_segment     5.829e-01  3.005e-01   1.940   0.0524 .
## JLR_station       -7.862e-01  6.521e-01  -1.206   0.2279
## Damascus_Gate_dis  3.701e-04  7.137e-05   5.186 2.15e-07 ***
## With_settlements  -2.061e+00  5.298e-01  -3.889   0.0001 ***
## change_1250_800   -9.009e-02  4.401e-01  -0.205   0.8378
## change_2000_1250   1.720e-01  3.281e-01   0.524   0.6002
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Number of iterations in BFGS optimization: 21
## Log-likelihood: -2116 on 16 Df
```

```
AIC(ind, change_ind, change_ind_s, m1, m3)
```

```
## Warning in AIC.default(ind, change_ind, change_ind_s, m1, m3): models are not
## all fitted to the same number of observations

##              df      AIC
## ind           7 356.8709
## change_ind    9 354.7711
## change_ind_s  7 357.3811
## m1           17 337.5441
## m3           16 353.6886
```

```
AIC(col, change_col, change_col_s, m2, m4)
```

```
## Warning in AIC.default(col, change_col, change_col_s, m2, m4): models are not
## all fitted to the same number of observations

##              df      AIC
## col           7 1006.373
## change_col    9 1018.750
## change_col_s  6 1014.732
## m2           17 1034.750
## m4           16 4263.348
```

The individual model was further improved over the original model as it had more zero inflation, but zero inflation did not help with the collective model. The zero inflated negative binomial collective model did not even have any significant variables. Let's compare the performance after we remove insignificant variables.

```
# Zero Inflated Significant Only

m1_s <- zeroinfl(Individual ~ Mean_800 +
                 Jewish_segment +
                 With_settlements +
                 change_1250_800 + change_2000_1250 |
                 JLR_station + change_1250_800,
               data = jlm_con, dist = "negbin")
summary(m1_s)
```

```
##
## Call:
## zeroinfl(formula = Individual ~ Mean_800 + Jewish_segment + With_settlements +
##     change_1250_800 + change_2000_1250 | JLR_station + change_1250_800,
##     data = jlm_con, dist = "negbin")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -0.6903 -0.2474 -0.1974 -0.1415 11.4091
##
## Count model coefficients (negbin with log link):
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -4.1669427  0.8376119  -4.975 6.53e-07 ***
## Mean_800          0.0032545  0.0007825   4.159 3.19e-05 ***
## Jewish_segment   -0.7134611  0.3662011  -1.948  0.05138 .
## With_settlements  1.7231614  0.5258080   3.277  0.00105 **
## change_1250_800   3.1222308  0.7241900   4.311 1.62e-05 ***
```

```
## change_2000_1250 -1.1117871  0.6303159  -1.764  0.07776 .
## Log(theta)       -0.5101692  0.4384006  -1.164  0.24454
##
## Zero-inflation model coefficients (binomial with logit link):
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.205      1.730  -1.275   0.2024
## JLR_station       -8.291     16.480  -0.503   0.6149
## change_1250_800    2.180      1.153   1.891   0.0586 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Theta = 0.6004
## Number of iterations in BFGS optimization: 41
## Log-likelihood: -165.1 on 10 Df
```

```r
# m2 had no significant variables

m3_s <- zeroinfl(Individual ~ Mean_800 +
                 Jewish_segment + JLR_station +
                 change_1250_800 + change_2000_1250 |
                 With_settlements,
               data = jlm_con, dist = "poisson")
summary(m3_s)
```

```
##
## Call:
## zeroinfl(formula = Individual ~ Mean_800 + Jewish_segment + JLR_station +
##     change_1250_800 + change_2000_1250 | With_settlements, data = jlm_con,
##     dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.0982 -0.2513 -0.2086 -0.1562  7.6804
##
## Count model coefficients (poisson with log link):
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.2922147  0.6588723  -0.444  0.65740
## Mean_800          0.0024594  0.0004681   5.254 1.49e-07 ***
## Jewish_segment   -1.0145835  0.2426252  -4.182 2.89e-05 ***
## JLR_station      -0.5120722  0.3181991  -1.609  0.10755
## change_1250_800   1.9968645  0.5048621   3.955 7.64e-05 ***
## change_2000_1250 -1.6407714  0.5473876  -2.997  0.00272 **
##
## Zero-inflation model coefficients (binomial with logit link):
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.1779     0.2590   8.409  < 2e-16 ***
## With_settlements  -2.6540     0.5846  -4.540 5.63e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 15
## Log-likelihood: -186.6 on 8 Df
```

```r
m4_s <- zeroinfl(Collective ~ Mean_800 +
                 Jewish_segment + JLR_station +
```

```r
                 With_settlements +
                 change_1250_800 + change_2000_1250 |
                   With_settlements + Damascus_Gate_dis,
               data = jlm_con, dist = "poisson")
summary(m4_s)
```

```
##
## Call:
## zeroinfl(formula = Collective ~ Mean_800 + Jewish_segment + JLR_station +
##     With_settlements + change_1250_800 + change_2000_1250 | With_settlements +
##     Damascus_Gate_dis, data = jlm_con, dist = "poisson")
##
## Pearson residuals:
##     Min      1Q  Median      3Q     Max
## -1.9019 -0.4048 -0.2992 -0.2045 49.8414
##
## Count model coefficients (poisson with log link):
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       4.5190438  0.0838302  53.907  < 2e-16 ***
## Mean_800         -0.0009771  0.0001374  -7.110 1.16e-12 ***
## Jewish_segment   -1.2361845  0.0615439 -20.086  < 2e-16 ***
## JLR_station      -1.7589454  0.2137700  -8.228  < 2e-16 ***
## With_settlements  0.5070585  0.0617457   8.212  < 2e-16 ***
## change_1250_800   0.2005601  0.0783305   2.560   0.0105 *
## change_2000_1250 -0.8936405  0.0729621 -12.248  < 2e-16 ***
##
## Zero-inflation model coefficients (binomial with logit link):
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        2.703e-01  2.732e-01   0.989    0.322
## With_settlements  -2.328e+00  5.086e-01  -4.577 4.71e-06 ***
## Damascus_Gate_dis  3.661e-04  5.514e-05   6.638 3.17e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Number of iterations in BFGS optimization: 13
## Log-likelihood: -2228 on 10 Df
```

```r
AIC(ind, change_ind, change_ind_s, m1, m3, m1_s, m3_s)
```

```
## Warning in AIC.default(ind, change_ind, change_ind_s, m1, m3, m1_s, m3_s):
## models are not all fitted to the same number of observations

##              df      AIC
## ind           7 356.8709
## change_ind    9 354.7711
## change_ind_s  7 357.3811
## m1           17 337.5441
## m3           16 353.6886
## m1_s         10 350.1444
## m3_s          8 389.2941
```

```r
AIC(col, change_col, change_col_s, m2, m4, m4_s)
```

```
## Warning in AIC.default(col, change_col, change_col_s, m2, m4, m4_s): models are
## not all fitted to the same number of observations
```

```
##              df      AIC
## col           7 1006.373
## change_col    9 1018.750
## change_col_s  6 1014.732
## m2           17 1034.750
## m4           16 4263.348
## m4_s         10 4476.999
```

All models decreased in performance.

Analysis 3: Logistic models

With count data problems, there is often a logistic "hurdle" model that predicts whether the count will be above 0 or some other threshold. This fits nicely with the research question of this paper since a robust model that can predict whether or not there will be an attack would be more useful than a less accurate count model that loses performance discerning between 1, 2, or 3 attacks.

```
# Create indicator, if 0 attacks then 0. If more, then 1
jlm_con <- jlm_con %>%
  mutate(Individual_ind = ifelse(Individual == 0, 0, 1),
         Collective_ind = ifelse(Collective == 0, 0, 1))

# logistic regression models
ind_log <- glm(Individual_ind ~ Mean_800 +
                 Jewish_segment + JLR_station +
                 Damascus_Gate_dis + With_settlements +
                 change_1250_800 + change_2000_1250,
               data=jlm_con, family = "binomial")
summary(ind_log)
```

```
##
## Call:
## glm(formula = Individual_ind ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 +
##     change_2000_1250, family = "binomial", data = jlm_con)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4504  -0.3178  -0.2120  -0.1413   3.2829
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.0816321  1.1146060  -0.970 0.331839
## Mean_800           0.0015781  0.0010952   1.441 0.149598
## Jewish_segment     0.1089836  0.4809711   0.227 0.820742
## JLR_station        2.1939317  0.5965413   3.678 0.000235 ***
## Damascus_Gate_dis -0.0004320  0.0001553  -2.782 0.005401 **
## With_settlements   1.7442062  0.6161562   2.831 0.004643 **
## change_1250_800    0.2165371  0.6975106   0.310 0.756224
## change_2000_1250  -0.7878197  0.6105191  -1.290 0.196908
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 278.74  on 530  degrees of freedom
```

```
## Residual deviance: 193.51  on 523  degrees of freedom
## AIC: 209.51
##
## Number of Fisher Scoring iterations: 7
```

```
col_log <- glm(Collective_ind ~ Mean_800 +
                 Jewish_segment + JLR_station +
                 Damascus_Gate_dis + With_settlements +
                 change_1250_800 + change_2000_1250,
               data=jlm_con, family = "binomial")
summary(col_log)
```

```
##
## Call:
## glm(formula = Collective_ind ~ Mean_800 + Jewish_segment + JLR_station +
##     Damascus_Gate_dis + With_settlements + change_1250_800 +
##     change_2000_1250, family = "binomial", data = jlm_con)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8204  -0.5409  -0.3863  -0.2731   2.5354
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.866e-01  6.724e-01   0.278   0.7813
## Mean_800           1.082e-04  8.531e-04   0.127   0.8991
## Jewish_segment    -6.047e-01  2.969e-01  -2.037   0.0417 *
## JLR_station        4.630e-01  5.925e-01   0.782   0.4345
## Damascus_Gate_dis -3.742e-04  8.412e-05  -4.449 8.64e-06 ***
## With_settlements   2.085e+00  5.300e-01   3.933 8.37e-05 ***
## change_1250_800    1.219e-01  4.343e-01   0.281   0.7790
## change_2000_1250  -2.140e-01  3.172e-01  -0.675   0.4999
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 460.37  on 530  degrees of freedom
## Residual deviance: 365.65  on 523  degrees of freedom
## AIC: 381.65
##
## Number of Fisher Scoring iterations: 5
```

```
#Logistic Significant Only
ind_log_s <- glm(Individual_ind ~ JLR_station +
                 Damascus_Gate_dis + With_settlements,
               data=jlm_con, family = "binomial")
summary(ind_log_s)
```

```
##
## Call:
## glm(formula = Individual_ind ~ JLR_station + Damascus_Gate_dis +
##     With_settlements, family = "binomial", data = jlm_con)
##
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.2870  -0.3388  -0.2184  -0.1310   3.3281
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.9239754  0.4859934  -1.901  0.05727 .
## JLR_station        2.3759799  0.5435540   4.371 1.24e-05 ***
## Damascus_Gate_dis -0.0005241  0.0001244  -4.213 2.52e-05 ***
## With_settlements   1.3687809  0.5203486   2.631  0.00853 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 278.74  on 530  degrees of freedom
## Residual deviance: 200.52  on 527  degrees of freedom
## AIC: 208.52
##
## Number of Fisher Scoring iterations: 7
```

```
col_log_s <- glm(Collective_ind ~ Jewish_segment +
                 Damascus_Gate_dis + With_settlements,
               data=jlm_con, family = "binomial")
summary(col_log_s)
```

```
##
## Call:
## glm(formula = Collective_ind ~ Jewish_segment + Damascus_Gate_dis +
##     With_settlements, family = "binomial", data = jlm_con)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8558  -0.5456  -0.3919  -0.2710   2.5545
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.199e-01  3.812e-01   0.314   0.7532
## Jewish_segment    -5.771e-01  2.819e-01  -2.047   0.0407 *
## Damascus_Gate_dis -3.801e-04  7.158e-05  -5.310 1.10e-07 ***
## With_settlements   2.067e+00  5.270e-01   3.923 8.76e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 460.37  on 530  degrees of freedom
## Residual deviance: 367.01  on 527  degrees of freedom
## AIC: 375.01
##
## Number of Fisher Scoring iterations: 5
```

```
AIC(ind, change_ind, change_ind_s, m1, m3, m1_s, m3_s, ind_log, ind_log_s)
```

```
## Warning in AIC.default(ind, change_ind, change_ind_s, m1, m3, m1_s, m3_s, :
## models are not all fitted to the same number of observations
```

```
##                df      AIC
## ind             7 356.8709
## change_ind      9 354.7711
## change_ind_s    7 357.3811
## m1             17 337.5441
## m3             16 353.6886
## m1_s           10 350.1444
## m3_s            8 389.2941
## ind_log         8 209.5080
## ind_log_s       4 208.5230
```

```
AIC(col, change_col, change_col_s, m2, m4, m4_s, col_log, col_log_s)
```

```
## Warning in AIC.default(col, change_col, change_col_s, m2, m4, m4_s, col_log, :
## models are not all fitted to the same number of observations
```

```
##                df       AIC
## col             7 1006.3734
## change_col      9 1018.7500
## change_col_s    6 1014.7320
## m2             17 1034.7500
## m4             16 4263.3479
## m4_s           10 4476.9994
## col_log         8  381.6481
## col_log_s       4  375.0099
```

As expected, the logistic models are much more informative than the count models.

Analysis 4: Interaction Terms

Perhaps some interaction terms could be having an effect on the target variable. these following models will see if there is any multiplicative relationship.

```
# Logistic Interaction terms
ind_log_t <- glm(Individual_ind ~ JLR_station +
                   Damascus_Gate_dis + With_settlements
                   + JLR_station:Damascus_Gate_dis
                   + JLR_station:With_settlements
                   + Damascus_Gate_dis:With_settlements
                 ,
                 data=jlm_con, family = "binomial")
summary(ind_log_t)
```

```
##
## Call:
## glm(formula = Individual_ind ~ JLR_station + Damascus_Gate_dis +
##     With_settlements + JLR_station:Damascus_Gate_dis + JLR_station:With_settlements +
##     Damascus_Gate_dis:With_settlements, family = "binomial",
##     data = jlm_con)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4275  -0.3421  -0.2320  -0.1471   3.2327
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                         -1.0970580  0.5263660  -2.084 0.037141 *
```

```
## JLR_station                         3.8920636  1.7972080   2.166 0.030340 *
## Damascus_Gate_dis                  -0.0004687  0.0001321  -3.549 0.000387 ***
## With_settlements                    1.7596450  0.9986584   1.762 0.078068 .
## JLR_station:Damascus_Gate_dis      -0.0005042  0.0005180  -0.973 0.330406
## JLR_station:With_settlements        7.5723915 70.3945963   0.108 0.914336
## Damascus_Gate_dis:With_settlements -0.0002332  0.0004757  -0.490 0.623952
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 278.74  on 530  degrees of freedom
## Residual deviance: 198.35  on 524  degrees of freedom
## AIC: 212.35
##
## Number of Fisher Scoring iterations: 11
```

```r
col_log_t <- glm(Collective_ind ~ Jewish_segment +
                   Damascus_Gate_dis + With_settlements
                   + Jewish_segment:Damascus_Gate_dis
                   + Jewish_segment:With_settlements
                   + Damascus_Gate_dis:With_settlements
                 ,
                 data=jlm_con, family = "binomial")
summary(col_log_t)
```

```
##
## Call:
## glm(formula = Collective_ind ~ Jewish_segment + Damascus_Gate_dis +
##     With_settlements + Jewish_segment:Damascus_Gate_dis + Jewish_segment:With_settlements +
##     Damascus_Gate_dis:With_settlements, family = "binomial",
##     data = jlm_con)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7211  -0.5424  -0.3899  -0.2703   2.5712
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                         2.771e-01  4.814e-01   0.576   0.5648
## Jewish_segment                     -8.630e-01  6.730e-01  -1.282   0.1997
## Damascus_Gate_dis                  -4.030e-04  9.823e-05  -4.103 4.08e-05 ***
## With_settlements                    1.668e+00  1.004e+00   1.661   0.0968 .
## Jewish_segment:Damascus_Gate_dis    4.455e-05  1.488e-04   0.300   0.7646
## Jewish_segment:With_settlements     5.573e+00  9.026e+00   0.617   0.5370
## Damascus_Gate_dis:With_settlements -1.169e-05  3.523e-04  -0.033   0.9735
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 460.37  on 530  degrees of freedom
## Residual deviance: 363.46  on 524  degrees of freedom
## AIC: 377.46
##
```

```
## Number of Fisher Scoring iterations: 7
```

```
AIC(ind, change_ind, change_ind_s, m1, m3, m1_s, m3_s,
    ind_log, ind_log_s, ind_log_t)
```

```
## Warning in AIC.default(ind, change_ind, change_ind_s, m1, m3, m1_s, m3_s, :
## models are not all fitted to the same number of observations
```

```
##              df      AIC
## ind           7 356.8709
## change_ind    9 354.7711
## change_ind_s  7 357.3811
## m1           17 337.5441
## m3           16 353.6886
## m1_s         10 350.1444
## m3_s          8 389.2941
## ind_log       8 209.5080
## ind_log_s     4 208.5230
## ind_log_t     7 212.3509
```

```
AIC(col, change_col, change_col_s, m2, m4, m4_s, col_log, col_log_s, col_log_t)
```

```
## Warning in AIC.default(col, change_col, change_col_s, m2, m4, m4_s, col_log, :
## models are not all fitted to the same number of observations
```

```
##              df       AIC
## col           7 1006.3734
## change_col    9 1018.7500
## change_col_s  6 1014.7320
## m2           17 1034.7500
## m4           16 4263.3479
## m4_s         10 4476.9994
## col_log       8  381.6481
## col_log_s     4  375.0099
## col_log_t     7  377.4576
```

No interaction terms were significantly significant. Their presence in the model increased AIC for both individual and collective. The logistic models with only significant variables have been the most robust models.

Analysis 5: SMOTE

As discussed, a very high proportion of the observations had zero attacks. This introduces the problem of a class imbalance into the logistic regression approach. We will need to make the models more sensitive to instances where there is an attack. This can be accomplished by upsampling the data. Rather than randomly upsampling the minority class with replacement, we will use Synthetic Minority Oversampling Technique (SMOTE) to generate new synthetic samples of the minority class based on points between nearest neighbors.

Additionally, we will need to alter our model comparison strategy. In this case we are mostly interested in not incorrectly classifying attacks as non-attacks while maintaining high accuracy in preparation for future attack instances rather than observing the pure statistical robustness of the regression model. Instead of AIC, we will split the data into training and testing, SMOTE the training set, and use a combination of accuracy, precision, recall, and F1 score to evaluate models.

```
# SMOTE
set.seed(42)

split <- runif(nrow(jlm_con)) # random number0 to 1 for each obs
```

```r
jlm_con <- cbind.data.frame(jlm_con, data.frame(split = split)) # add to df

jlm_con <- jlm_con %>%
  mutate(split_ind = ifelse(jlm_con$split > 0.2, 1, 0)) # 80% train size

train <- jlm_con[jlm_con$split_ind == 1, ]
test <- jlm_con[jlm_con$split_ind == 0, ]

# Only want to smote based on relevant significant variables. Inclusion of
# others will throw off the SMOTE nearest neighbors algorithm

ind_train <- train[ , c('Individual_ind', 'JLR_station', 'Damascus_Gate_dis',
                        'With_settlements')]
col_train <- train[ , c('Collective_ind', 'Jewish_segment', 'Damascus_Gate_dis',
                        'With_settlements')]
```

```r
# Train base models on training sets with significant features from prev model
ind_base <- glm(Individual_ind ~ JLR_station +
                  Damascus_Gate_dis + With_settlements,
                data=ind_train, family = "binomial")
summary(ind_base)
```

```
##
## Call:
## glm(formula = Individual_ind ~ JLR_station + Damascus_Gate_dis +
##     With_settlements, family = "binomial", data = ind_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3245  -0.3641  -0.2572  -0.1641   3.1459
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -1.0366691  0.5261995  -1.970 0.048826 *
## JLR_station        2.3907955  0.5705496   4.190 2.79e-05 ***
## Damascus_Gate_dis -0.0004439  0.0001273  -3.487 0.000488 ***
## With_settlements   0.9958460  0.5949293   1.674 0.094152 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.50  on 408  degrees of freedom
## Residual deviance: 172.62  on 405  degrees of freedom
## AIC: 180.62
##
## Number of Fisher Scoring iterations: 6
```

```r
col_base <- glm(Collective_ind ~ Jewish_segment +
                  Damascus_Gate_dis + With_settlements,
                data=col_train, family = "binomial")
summary(col_base)
```

```
##
## Call:
```

```
## glm(formula = Collective_ind ~ Jewish_segment + Damascus_Gate_dis +
##     With_settlements, family = "binomial", data = col_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7064  -0.5727  -0.4236  -0.2813   2.4872
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.406e-01  4.202e-01   0.335   0.7379
## Jewish_segment     -5.434e-01  3.069e-01  -1.770   0.0767 .
## Damascus_Gate_dis -3.623e-04  7.742e-05  -4.680 2.87e-06 ***
## With_settlements   1.681e+00  5.537e-01   3.035   0.0024 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 364.78  on 408  degrees of freedom
## Residual deviance: 304.98  on 405  degrees of freedom
## AIC: 312.98
##
## Number of Fisher Scoring iterations: 5
```

```
# dup_size in smote function is desired minority size / actual minority size
# Want it to be 40% of total

IND_DUP_SIZE = 0.4 / (table(train$Individual_ind)[2] /
                        (table(train$Individual_ind)[2]
                         + table(train$Individual_ind)[1]))
COL_DUP_SIZE = 0.4 / (table(train$Collective_ind)[2] /
                        (table(train$Collective_ind)[2]
                         + table(train$Collective_ind)[1]))

# create smoted training sets using smote function
ind_smote_df <- SMOTE(ind_train, ind_train$Individual_ind, K = 3,
                      dup_size = IND_DUP_SIZE)
ind_smote_df <- ind_smote_df$data

col_smote_df <- SMOTE(col_train, col_train$Collective_ind, K = 3,
                      dup_size = COL_DUP_SIZE)
col_smote_df <- col_smote_df$data
```

```
ind_smote <- glm(Individual_ind ~ JLR_station +
                  Damascus_Gate_dis + With_settlements,
                data=ind_smote_df, family = "binomial")
summary(ind_smote)
```

```
##
## Call:
## glm(formula = Individual_ind ~ JLR_station + Damascus_Gate_dis +
##     With_settlements, family = "binomial", data = ind_smote_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -2.4951  -0.7033  -0.4334   0.5884   2.5345
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         5.797e-01  2.747e-01   2.110   0.0349 *
## JLR_station         3.462e+00  5.476e-01   6.322 2.58e-10 ***
## Damascus_Gate_dis -4.263e-04  6.189e-05  -6.889 5.62e-12 ***
## With_settlements   1.060e+00  4.141e-01   2.560   0.0105 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 727.54  on 568  degrees of freedom
## Residual deviance: 514.84  on 565  degrees of freedom
## AIC: 522.84
##
## Number of Fisher Scoring iterations: 5
```

```r
col_smote <- glm(Collective_ind ~ Jewish_segment +
                   Damascus_Gate_dis + With_settlements,
                 data=col_smote_df, family = "binomial")
summary(col_smote)
```

```
##
## Call:
## glm(formula = Collective_ind ~ Jewish_segment + Damascus_Gate_dis +
##     With_settlements, family = "binomial", data = col_smote_df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3010  -0.8472  -0.5694   1.0052   2.0299
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         1.3095468  0.3079253   4.253 2.11e-05 ***
## Jewish_segment    -0.6472729  0.2217350  -2.919 0.003510 **
## Damascus_Gate_dis -0.0003676  0.0000524  -7.015 2.31e-12 ***
## With_settlements   1.9043904  0.5323322   3.577 0.000347 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 715.72  on 542  degrees of freedom
## Residual deviance: 587.33  on 539  degrees of freedom
## AIC: 595.33
##
## Number of Fisher Scoring iterations: 4
```

```r
# for each model, add predict column to the test set and probability column
test <- test %>%
  mutate(ind_base_prob = predict(ind_base, newdata = test, type = "response"),
         col_base_prob = predict(col_base, newdata = test, type = "response"),
         ind_smote_prob = predict(ind_smote, newdata = test, type = "response"),
```

```
         col_smote_prob = predict(ind_smote, newdata = test, type = "response"),
         ind_base_pred = if_else(ind_base_prob >= 0.5, 1, 0),
         col_base_pred = if_else(col_base_prob >= 0.5, 1, 0),
         ind_smote_pred = if_else(ind_smote_prob >= 0.5, 1, 0),
         col_smote_pred = if_else(col_smote_prob >= 0.5, 1, 0))
```

```
# Based on confusion matrix find accuracy, precision, recall, f1 for all models
ind_base_confusion = as.matrix(table(Actual_Values = test$Individual_ind,
                                     Predicted_Values = test$ind_base_pred))
print(ind_base_confusion)
```

```
##              Predicted_Values
## Actual_Values   0   1
##             0 115   0
##             1   6   1
```

```
IND_BASE_ACC = (ind_base_confusion[1,1] + ind_base_confusion[2,2]) /
  (nrow(test))
IND_BASE_PREC = (ind_base_confusion[2,2])/
  (ind_base_confusion[2,2]+ind_base_confusion[1,2])
IND_BASE_RECALL = (ind_base_confusion[2,2])/
  (ind_base_confusion[2,2]+ind_base_confusion[2,1])
IND_BASE_F1 = 2*((IND_BASE_PREC * IND_BASE_RECALL) /
                   (IND_BASE_PREC+IND_BASE_RECALL))

col_base_confusion = as.matrix(table(Actual_Values = test$Collective_ind,
                                     Predicted_Values = test$col_base_pred))
print(col_base_confusion)
```

```
##              Predicted_Values
## Actual_Values   0   1
##             0 106   0
##             1   9   7
```

```
COL_BASE_ACC = (col_base_confusion[1,1] + col_base_confusion[2,2]) /
  (nrow(test))
COL_BASE_PREC = (col_base_confusion[2,2])/(col_base_confusion[2,2]+
                                             col_base_confusion[1,2])
COL_BASE_RECALL = (col_base_confusion[2,2])/(col_base_confusion[2,2]+
                                               col_base_confusion[2,1])
COL_BASE_F1 = 2*((COL_BASE_PREC * COL_BASE_RECALL) / (COL_BASE_PREC+
                                                        COL_BASE_RECALL))

ind_smote_confusion = as.matrix(table(Actual_Values = test$Individual_ind,
                                      Predicted_Values = test$ind_smote_pred))
print(ind_smote_confusion)
```

```
##              Predicted_Values
## Actual_Values   0   1
##             0 111   4
##             1   2   5
```

```
IND_SMOTE_ACC = (ind_smote_confusion[1,1] + ind_smote_confusion[2,2]) /
  (nrow(test))
IND_SMOTE_PREC = (ind_smote_confusion[2,2])/(ind_smote_confusion[2,2]+
                                               ind_smote_confusion[1,2])
```

```r
IND_SMOTE_RECALL = (ind_smote_confusion[2,2])/(ind_smote_confusion[2,2]+
                                                 ind_smote_confusion[2,1])
IND_SMOTE_F1 = 2*((IND_SMOTE_PREC * IND_SMOTE_RECALL) / (IND_SMOTE_PREC+
                                                           IND_SMOTE_RECALL))

col_smote_confusion = as.matrix(table(Actual_Values = test$Collective_ind,
                                      Predicted_Values = test$col_smote_pred))
print(col_smote_confusion)
```

```
##              Predicted_Values
## Actual_Values   0   1
##             0 104   2
##             1   9   7
```

```r
COL_SMOTE_ACC = (col_smote_confusion[1,1] + col_smote_confusion[2,2]) /
  (nrow(test))
COL_SMOTE_PREC = (col_smote_confusion[2,2])/(col_smote_confusion[2,2]+
                                               col_smote_confusion[1,2])
COL_SMOTE_RECALL = (col_smote_confusion[2,2])/(col_smote_confusion[2,2]+
                                                 col_smote_confusion[2,1])
COL_SMOTE_F1 = 2*((COL_SMOTE_PREC * COL_SMOTE_RECALL) / (COL_SMOTE_PREC+
                                                           COL_SMOTE_RECALL))
```

For the non-smoted individual model, you would be nearly as effective just predicting no attacks as the model only predicted one attack. However for collective, the smoted model predicted more positives, but they were both false positives indicating the SMOTE training did not imporve the model

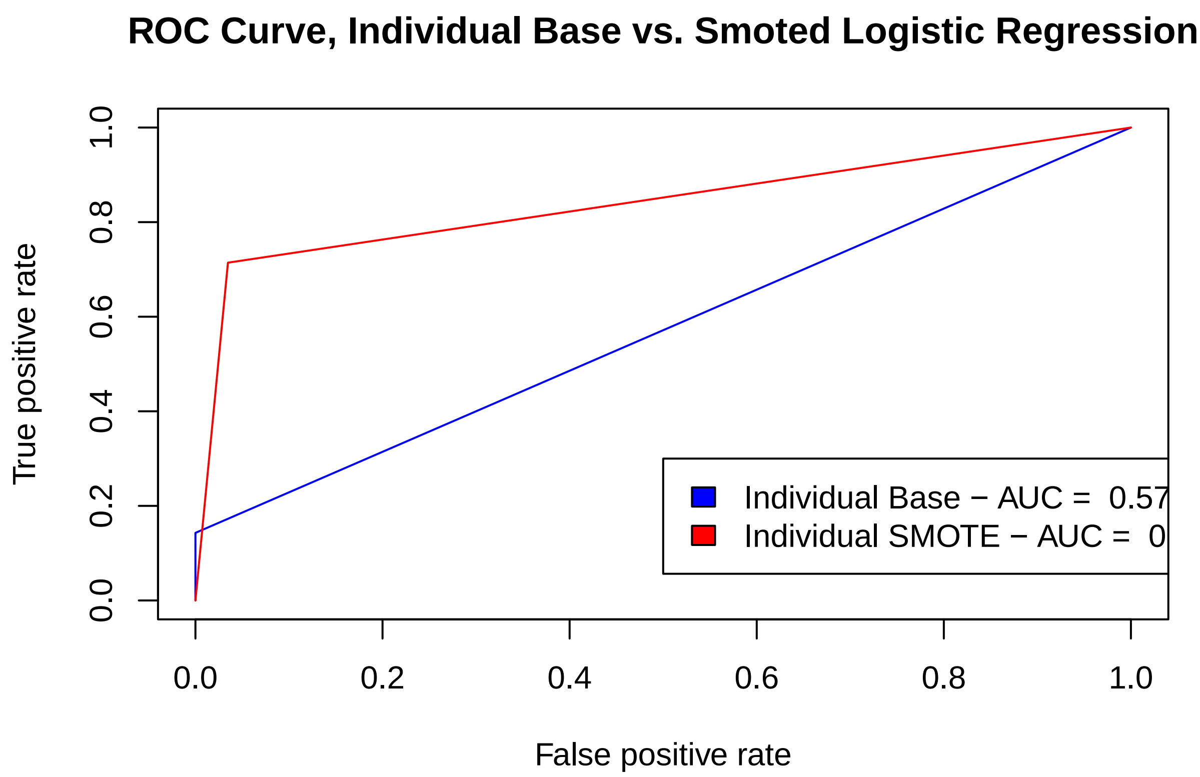
```r
# Nicely formatted table with model performance metrics
performance <- data.frame(model = c('ind_base', 'col_base', 'ind_smote',
                                    'col_smote'),
                          accuracy = c(IND_BASE_ACC, COL_BASE_ACC,
                                       IND_SMOTE_ACC, COL_SMOTE_ACC),
                          precision = c(IND_BASE_PREC, COL_BASE_PREC,
                                        IND_SMOTE_PREC, COL_SMOTE_PREC),
                          recall = c(IND_BASE_RECALL, COL_BASE_RECALL,
                                     IND_SMOTE_RECALL, COL_SMOTE_RECALL),
                          f1 = c(IND_BASE_F1, COL_BASE_F1, IND_SMOTE_F1,
                                 COL_SMOTE_F1))
print(performance)
```

```
##       model  accuracy precision    recall        f1
## 1  ind_base 0.9508197 1.0000000 0.1428571 0.2500000
## 2  col_base 0.9262295 1.0000000 0.4375000 0.6086957
## 3 ind_smote 0.9508197 0.5555556 0.7142857 0.6250000
## 4 col_smote 0.9098361 0.7777778 0.4375000 0.5600000
```

For individual, the smoted model had the same accuracy, but higher recall and F1 indicating it is more useful. For collective, the most effective model remains the logistic regression with only the significant attributes including no interaction terms. Sometimes simple is best.

```r
# arrays necessary for roc plot function
predi <- prediction(test$ind_base_pred, test$Individual_ind)
perfi <- performance(predi,"tpr","fpr")
predis <- prediction(test$ind_smote_pred, test$Individual_ind)
perfis <- performance(predis,"tpr","fpr")
# plot curve
```

```
plot(perfi, col = "blue")
plot(perfis, add = TRUE, col = "red")
# add auc levels
ibase_label <- round(performance(predi, measure = "auc")@y.values[[1]], 2)
ibase_label <- paste("Individual Base - AUC = ", ibase_label)
ismote_label <- round(performance(predis, measure = "auc")@y.values[[1]], 2)
ismote_label <- paste("Individual SMOTE - AUC = ", ismote_label)
# add legends
legend(0.5,0.3,legend=c(ibase_label, ismote_label),
       fill = c("blue","red")
)
title("ROC Curve, Individual Base vs. Smoted Logistic Regression")
```




```
predc <- prediction(test$col_base_pred, test$Collective_ind)
perfc <- performance(predc,"tpr","fpr")
predcs <- prediction(test$col_smote_pred, test$Collective_ind)
perfcs <- performance(predcs,"tpr","fpr")
plot(perfc, col = "blue")
plot(perfcs, add = TRUE, col = "red")
cbase_label <- round(performance(predc, measure = "auc")@y.values[[1]], 2)
cbase_label <- paste("Collective Base - AUC = ", cbase_label)
csmote_label <- round(performance(predcs, measure = "auc")@y.values[[1]], 2)
csmote_label <- paste("Collective SMOTE - AUC = ", csmote_label)
legend(0.5,0.3,legend=c(cbase_label, csmote_label),
       fill = c("blue","red")
)
```

```r
title("ROC Curve, Collective Base vs. Smoted Logistic Regression")
```