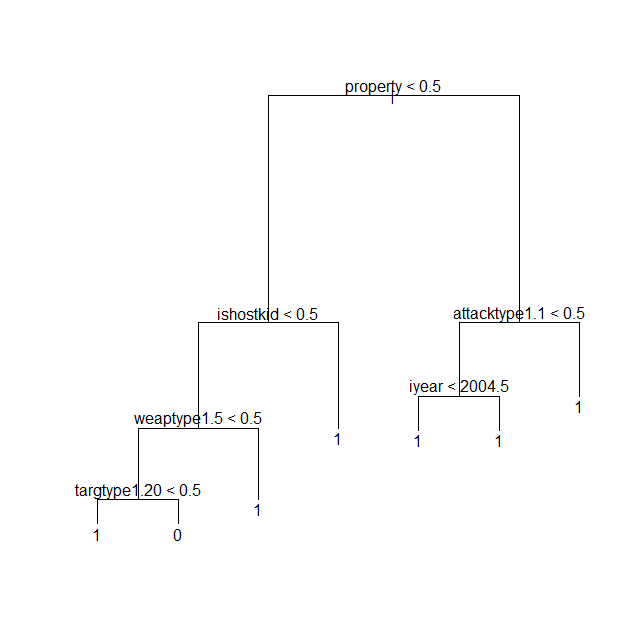
# Project Report Stat Learning

## Question 1: Predicting Attack Success

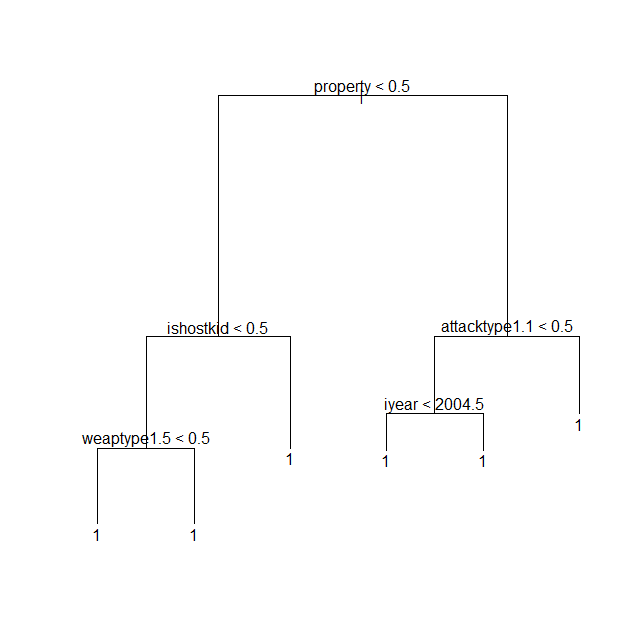
After loading in the data, I looked at the features. Simply using backwards selection to remove variables was not going to work as there were no rows with all values included; models would not find the data usable in its current state. Just deleting one or two columns was not enough to make the data usable, so I looked at the number of NAs per column; I removed anything with over 10000 NAs as most of these were either secondary or tertiary variables measuring the same thing (e.g. “targtype2” and “targtype3”) or variables that were only relevant to a very small subset of the data (e.g. “ransomamt” or “ransompaid”, which are only relevant if there was a ransom) or finally, variables that didn’t align with the goals of the model (e.g. “nkill” or “nwound” which wouldn’t be helpful for predicting future attacks, as they aren’t relevant until after the attack). After removing the variables, I hot encoded the remaining ones so that an incorrect idea of the category’s relationship was not derived from the fact that many of the categorical variables were recorded as numbers. I also noted that the target variable “success” was unbalanced (only 11% failures); however, I decided to see what I could be done leaving it in its original state.

Since most of the remaining variables were categorical (everything but “latitude” and “longitude”), I decided to use decision trees. This seemed like the most natural decision, as making left/right decisions with binary variables seemed appropriate. Originally, I used the “tree” package however the results were less the stellar (see *Figure 1*).

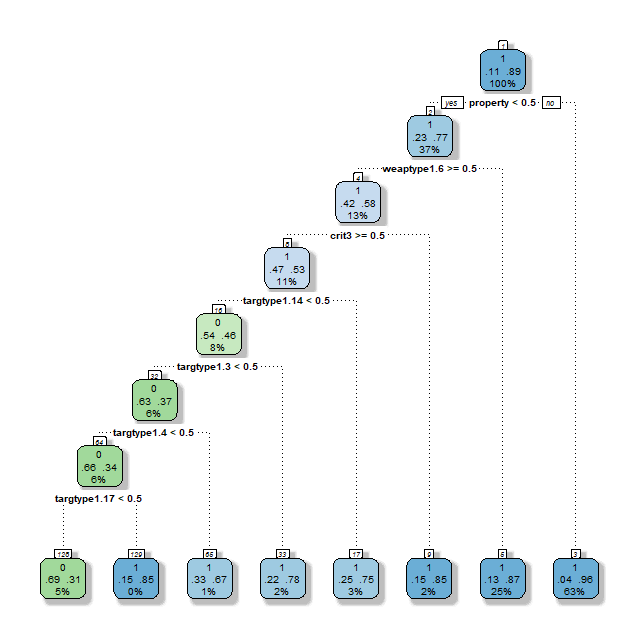


One problem is that this tree is obviously not well balanced; it will only predict failure in one case, that of “targtype1.20” being true (after other variable queries of course). It turns out that “targtype1.20” stands for the target type being unknown, which makes it useless (you get the same information by noting that all the other targtype1 columns are false). After removing the column, I got a new tree as seen in *Figure 2*.

Figure 1 – Initial tree model



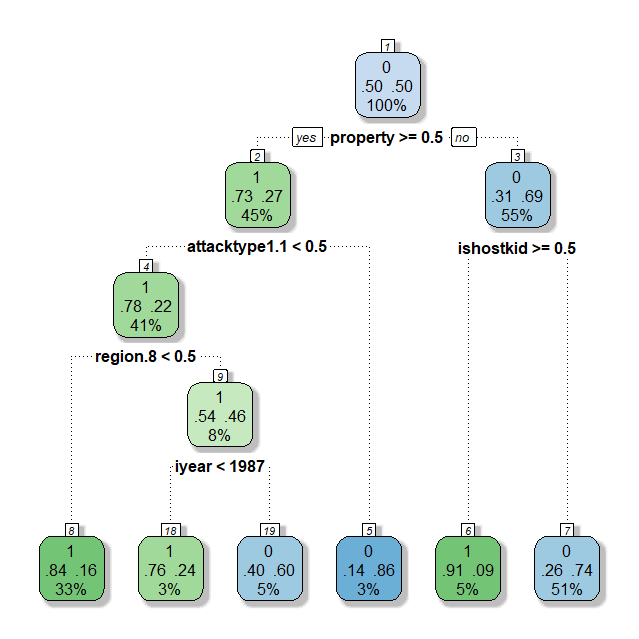
*Figure 2 –* *Tree model after deleting “targtype1.20”*

Now the tree only predicted success, never failure. While this gives reasonably high accuracy, it is not a very informative model. In an attempt to find the probability of success at each leaf, I found a package called “rpart.” This package was an equivalent package to “tree” however, it had additional information on some of its plots. There must have been a subtle difference in how it calculated the tree, as when I ran it on the same data a got a different tree (*Figure 3*).

*Figure 3 –* *Tree created with the “rpart” package*

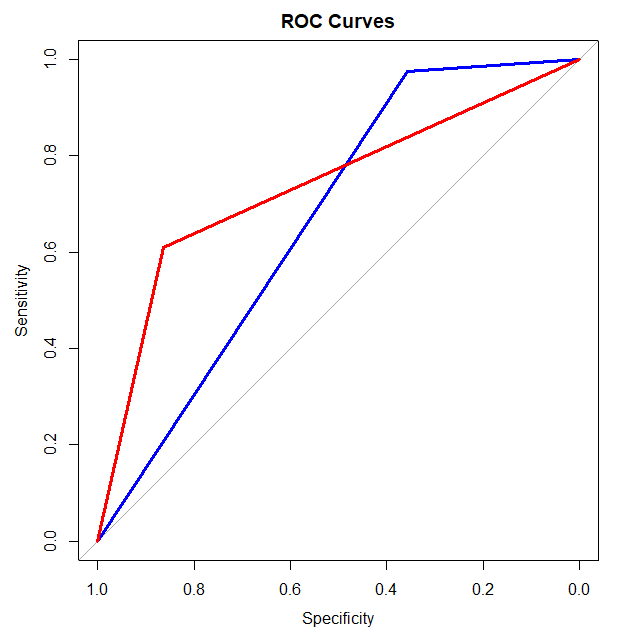
This model still has issues. Its precision and recall on testing data are 0.685 and 0.337 respectively (with 0 as the positive class), which is not the best. On top of that, the accuracy it does have seems derive from a quirk in how success is defined then anything truly meaningful. Notice the two highest nodes: “property” is a binary variable indicating whether there was any property damage as a result of the attack, “weaptype1.6” corresponds to attacks with explosive weapons. If you look in the documentation for the dataset, it states that “A bombing is successful if the bomb or explosive device detonates. Bombings are considered unsuccessful if they do not detonate. The success or failure of the bombing is not based on whether it hit the intended target.” Clearly in many cases, if there is no property damage and the weapon is an explosive, then it will have not been successful. This gives us very little information about success/failure in general.

The problem that seemed the easiest to fix was the issue of unbalanced data. I tried a technique called oversampling, where you duplicate instances of the data from the underrepresented class. The result was the following tree (*Figure 4*).



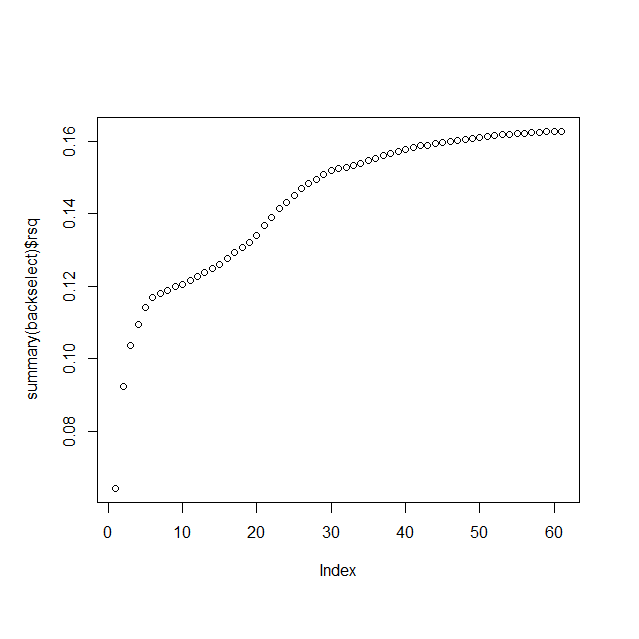
*Figure 4 – New* *tree created with balanced data*

This model had a precision and recall of 0.213 and 0.864 respectively. It seemed to have overcorrected in the opposite direction. Neither of the two tree models can be improved significantly by altering the thresholds. This new model is a bit better than the previous one as can be seen by looking at the ROC curves (*Figure 5*).

  
*Figure 5 – ROC curves; Blue: unbalanced tree model, Red: balanced tree model.*

The AUC for the first and second respectively is 0.667 and 0.7367.

Next, I decided to try out some different types of models, first Logistic Regression. I started with the unbalanced data and used backward selection to select the best variables. I picked 30 variables based off the following plot (*Figure 6*).

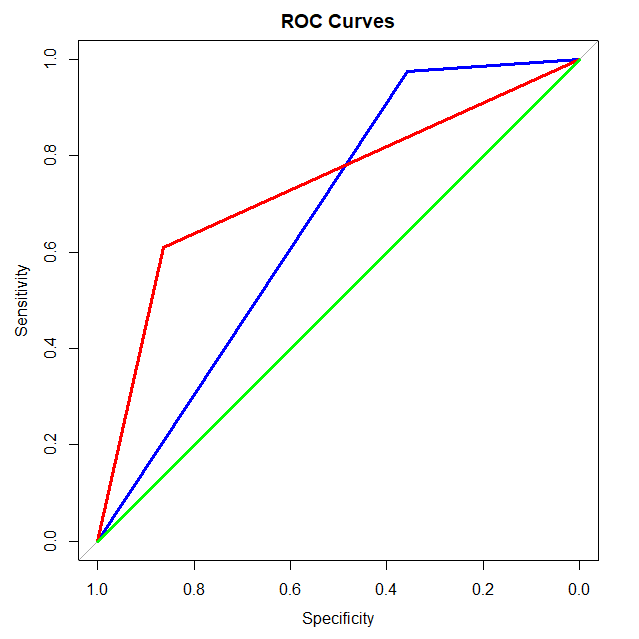


*Figure 6 – Number of variables vs. R-squared*

The results were:

Call:  
glm(formula = success ~ ., family = "binomial", data = data.selected)  
  
Deviance Residuals:   
 Min 1Q Median 3Q Max   
-4.1486 0.1687 0.2604 0.4120 2.0687   
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) -1.09071 0.07308 -14.925 < 2e-16 \*\*\*  
region.6 0.40873 0.03452 11.840 < 2e-16 \*\*\*  
region.11 0.60679 0.04979 12.187 < 2e-16 \*\*\*  
attacktype1.1 1.09595 0.05195 21.098 < 2e-16 \*\*\*  
attacktype1.2 1.35816 0.05009 27.116 < 2e-16 \*\*\*  
attacktype1.3 2.22194 0.07321 30.349 < 2e-16 \*\*\*  
attacktype1.6 -0.32326 0.19378 -1.668 0.095282 .   
attacktype1.7 0.34666 0.08019 4.323 1.54e-05 \*\*\*  
targtype1.1 1.34196 0.07408 18.116 < 2e-16 \*\*\*  
targtype1.2 1.06465 0.07127 14.938 < 2e-16 \*\*\*  
targtype1.3 1.62197 0.06929 23.409 < 2e-16 \*\*\*  
targtype1.4 1.56869 0.06578 23.846 < 2e-16 \*\*\*  
targtype1.6 0.87450 0.14295 6.118 9.50e-10 \*\*\*  
targtype1.7 0.77704 0.10222 7.602 2.92e-14 \*\*\*  
targtype1.8 1.52167 0.11964 12.719 < 2e-16 \*\*\*  
targtype1.9 0.76934 0.27037 2.845 0.004434 \*\*   
targtype1.10 1.69781 0.14349 11.832 < 2e-16 \*\*\*  
targtype1.11 1.02639 0.30233 3.395 0.000687 \*\*\*  
targtype1.12 1.60504 0.25804 6.220 4.97e-10 \*\*\*  
targtype1.14 1.90767 0.06749 28.265 < 2e-16 \*\*\*  
targtype1.15 1.86101 0.13835 13.452 < 2e-16 \*\*\*  
targtype1.16 1.28116 0.20019 6.400 1.56e-10 \*\*\*  
targtype1.17 2.01486 0.11742 17.160 < 2e-16 \*\*\*  
targtype1.18 0.94274 0.25077 3.759 0.000170 \*\*\*  
targtype1.19 1.02183 0.09257 11.039 < 2e-16 \*\*\*  
targtype1.21 1.75442 0.10605 16.543 < 2e-16 \*\*\*  
targtype1.22 1.39467 0.15435 9.036 < 2e-16 \*\*\*  
weaptype1.6 -1.87432 0.05872 -31.920 < 2e-16 \*\*\*  
property 2.35592 0.03481 67.680 < 2e-16 \*\*\*  
ishostkid 4.28417 0.20892 20.506 < 2e-16 \*\*\*  
INT\_LOG -0.53078 0.04417 -12.018 < 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: 50455 on 79650 degrees of freedom  
Residual deviance: 39050 on 79620 degrees of freedom  
(102040 observations deleted due to missingness)  
AIC: 39112  
  
Number of Fisher Scoring iterations: 7

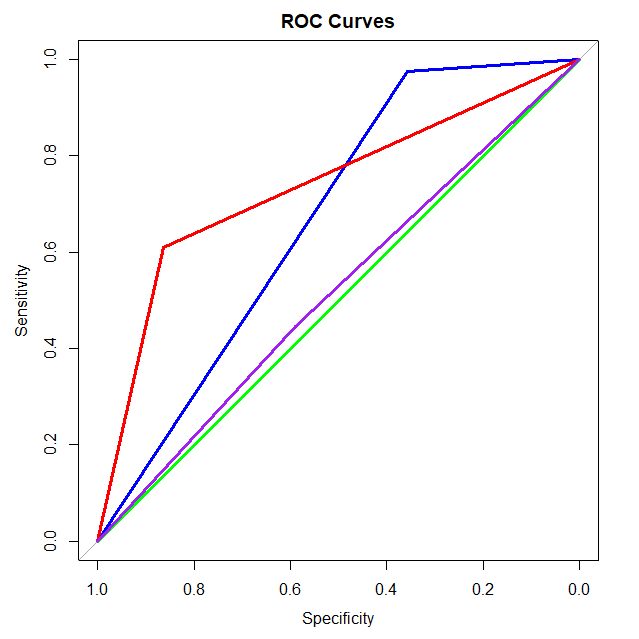
A quick look at the ROC curve indicates that this model was not particularly effective (*Figure 7*).

  
*Figure 7 – ROC curves; Blue: unbalanced tree model, Red: balanced tree model, Green: logistic regression model*

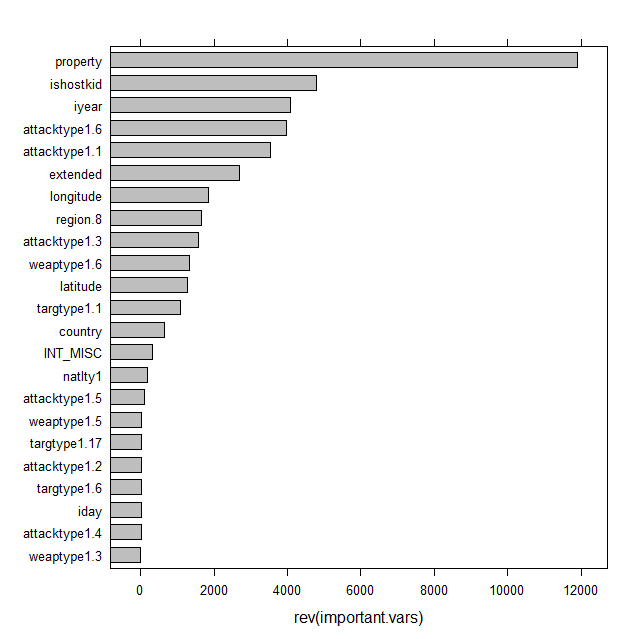
I then tried logistic regression on the balanced data, using similar methods as before.

Call:  
glm(formula = success ~ ., family = "binomial", data = data.selected.2)  
  
Deviance Residuals:   
 Min 1Q Median 3Q Max   
-2.9358 -0.7775 0.1648 0.7868 3.1412   
  
Coefficients:  
 Estimate Std. Error z value Pr(>|z|)   
(Intercept) -1.827e+01 1.360e+00 -13.441 <2e-16 \*\*\*  
iyear 1.116e-02 6.785e-04 16.454 <2e-16 \*\*\*  
extended -2.190e+00 9.809e-02 -22.330 <2e-16 \*\*\*  
country 1.429e-03 6.667e-05 21.436 <2e-16 \*\*\*  
region.1 7.187e-01 4.502e-02 15.965 <2e-16 \*\*\*  
region.2 -1.077e+00 4.449e-02 -24.214 <2e-16 \*\*\*  
region.3 -3.865e-01 2.552e-02 -15.149 <2e-16 \*\*\*  
region.6 -6.709e-01 2.390e-02 -28.074 <2e-16 \*\*\*  
region.10 -3.667e-01 2.226e-02 -16.473 <2e-16 \*\*\*  
region.11 -7.792e-01 2.838e-02 -27.451 <2e-16 \*\*\*  
crit3 -1.388e+00 6.375e-02 -21.781 <2e-16 \*\*\*  
doubtterr -8.777e-01 5.201e-02 -16.875 <2e-16 \*\*\*  
attacktype1.1 -6.278e-01 4.054e-02 -15.486 <2e-16 \*\*\*  
attacktype1.2 -9.684e-01 3.782e-02 -25.604 <2e-16 \*\*\*  
attacktype1.3 -1.957e+00 4.867e-02 -40.202 <2e-16 \*\*\*  
attacktype1.6 2.047e-01 8.594e-02 2.381 0.0173 \*   
attacktype1.7 -7.858e-01 3.992e-02 -19.685 <2e-16 \*\*\*  
attacktype1.8 -1.291e+00 9.132e-02 -14.133 <2e-16 \*\*\*  
targtype1.1 -4.436e-01 2.429e-02 -18.263 <2e-16 \*\*\*  
targtype1.3 -6.598e-01 2.237e-02 -29.490 <2e-16 \*\*\*  
targtype1.4 -1.018e+00 4.039e-02 -25.194 <2e-16 \*\*\*  
targtype1.8 -6.692e-01 5.252e-02 -12.743 <2e-16 \*\*\*  
targtype1.14 -1.023e+00 2.137e-02 -47.893 <2e-16 \*\*\*  
targtype1.15 -1.032e+00 5.717e-02 -18.055 <2e-16 \*\*\*  
targtype1.21 -7.897e-01 4.084e-02 -19.334 <2e-16 \*\*\*  
natlty1 -1.656e-03 1.016e-04 -16.297 <2e-16 \*\*\*  
weaptype1.5 -4.694e-01 3.084e-02 -15.223 <2e-16 \*\*\*  
weaptype1.6 1.437e+00 4.240e-02 33.879 <2e-16 \*\*\*  
property -2.223e+00 1.670e-02 -133.112 <2e-16 \*\*\*  
ishostkid -3.219e+00 8.591e-02 -37.469 <2e-16 \*\*\*  
INT\_LOG 4.312e-01 2.417e-02 17.839 <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: 187339 on 135136 degrees of freedom  
Residual deviance: 134985 on 135106 degrees of freedom  
AIC: 135047  
  
Number of Fisher Scoring iterations: 6

This gave a slightly better result, however nowhere near as good as the decision tree models (*Figure 8*).

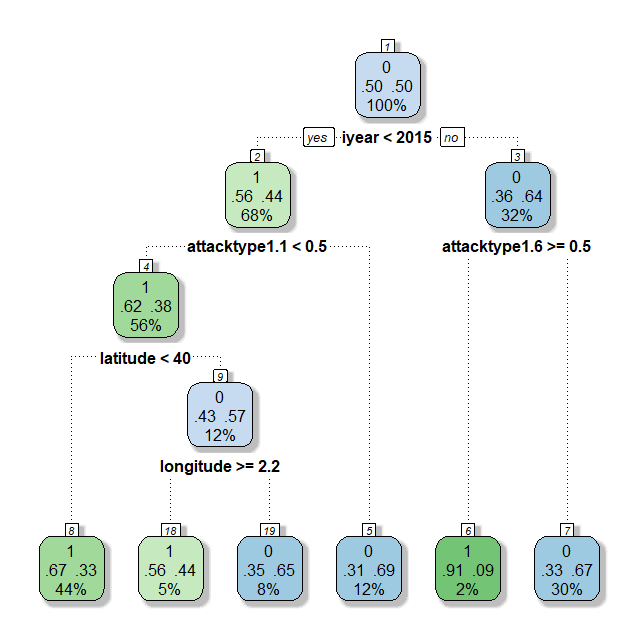
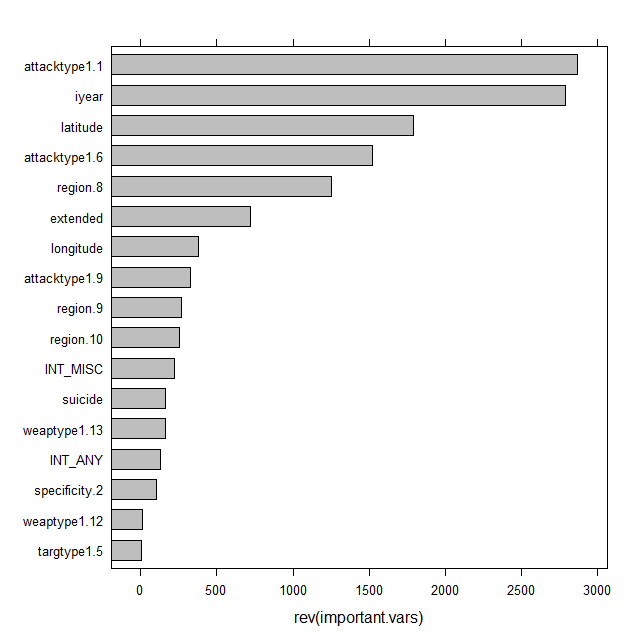
  
*Figure 8 – ROC curves; Blue: unbalanced tree model, Red: balanced tree model, Green: logistic regression model, Purple: balanced logistic regression model*

At this point I decided that logistic regression did not seem to be going anywhere. The second tree model (the one with the balanced data) seemed to be the best, so I decided to see what could be done to refine it. First, I looked at the variable importance of the balanced tree model (*Figure 9*).



*Figure 9 – Variable importance of variables in balanced tree model*

First off, the variable “property” seems to be vastly more important in predictions than any of the other variables. This might be because weapon types and attacks that cause property damage are more likely to result in success. However, another possibility is this derives from the fact that, for many definitions of success, property damage might be sufficient to designate the attack as successful. Additionally, many variables overlap and are somewhat redundant. For instance, “ishostkid” is a boolean feature measuring whether hostages were taken, “attacktype1.6” are kidnappings.

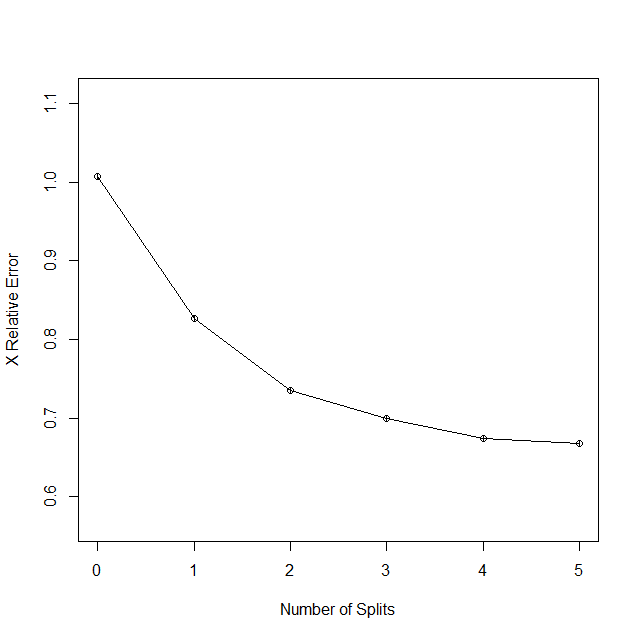
Considering all this, I decided to remove a few of the variables and see how the model performed. I

*Figure 10 and 11 – New tree with removed variables and new variable importance plot*

removed, “country” and “natlty1” (both redundant with region, a subtle difference between nationality and region exists, however the difference is slight enough that I didn’t see any reason to keep it for this question in particular). I removed “property” and “ishostkid” for the reason mentioned above; I also removed “weaptype1.6” (Explosives) as there already existed an attack type (“attacktype1.3”) that corresponded with explosive attacks. Running “rpart” with the changes gets us the following two figures (*Figure 10* and *11*).

The two attack types in the tree (*Figure* 10) are assassinations to the left and kidnappings to the right. As for the new model’s accuracy, its precision and recall on testing data were 0.2031986 and 0.6045894 respectively with an accuracy score of 0.6888043. “iyear” is a curious variable to be on top; the main question that comes to mind is whether its importance is due a major shift in how attacks are carried out, or just a shift in how and which attacks are recorded.

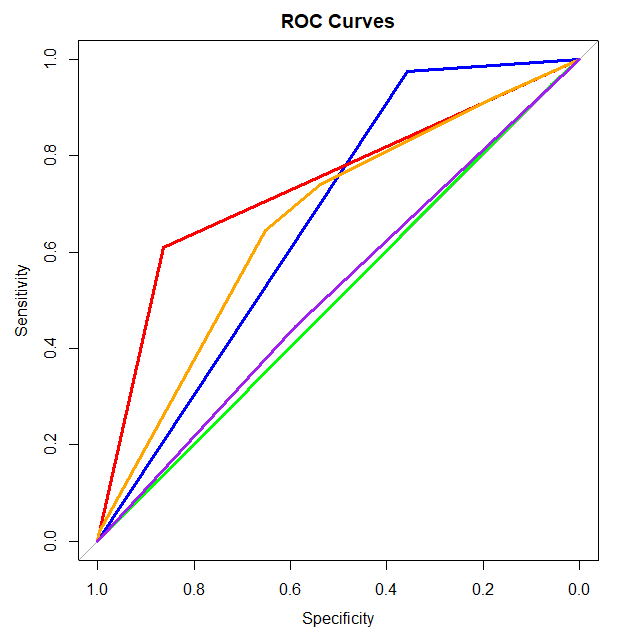
Both the longitude and latitude are interesting features of this tree model; however, I was not convinced that they were necessary and not the result of overfitting. I used rsq.rpart() to measure the error against the number of splits obtaining the following plot (*Figure 12*).



*Figure 12 – Error vs. Number of Splits*

This plot seems to indicate that between two and four splits are needed. Two splits leave only “iyear” and the assassination attack type; four leaves everything but longitude. The two-split model has the following results: *precision:* 0.1970123, *recall:* 0.5415459, *accuracy:* 0.7001087 and *AUC:* 0.6344, while the the four-split model results are: *precision:* 0.1888562, *recall:* 0.6516908, *accuracy:* 0.6459239, and *AUC:* 0.6634. Due to the AUC, the four-split model seems to be the best of the two.

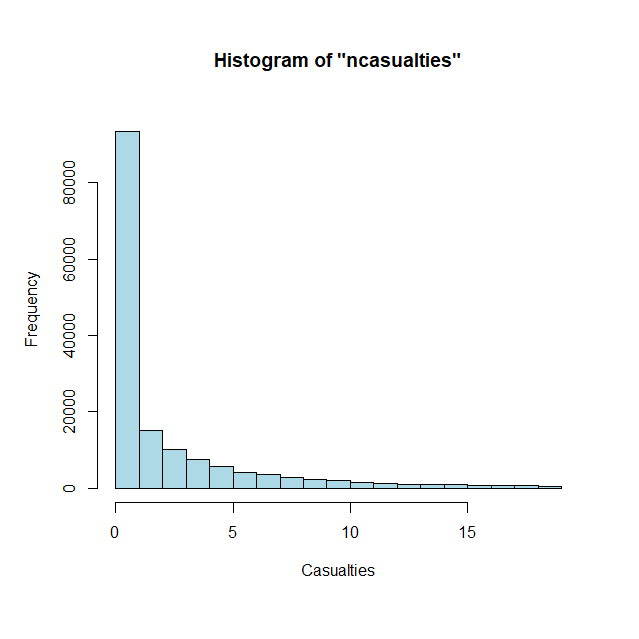
If we compare all the models, we get the following ROC curves (*Figure 13*):

*Figure 13 – Blue Model (unbalanced tree) AUC: 0.667; Red Model (balanced tree) AUC: 0.7367; Orange Model (final tweaked tree) AUC: 0.6634; Green Model (unbalanced logistic regression) AUC: 0.5027; Purple Model (balanced logistic regression) AUC: 0.5183.*

Judging purely by the AUC, the balanced tree model is by far the best. It’s also interesting to note that the tweaked model (orange) and the balanced model (red) are actually very similar if you look at their trees. The primary difference is the root node, however the rest of the nodes are very similar; in place of “ishostkid” (boolean true for hostages/kidnapping) we have “attacktype1.6” (attack type for kidnapping) and in place of latitude < 40 we have region.8 (region 8 is Western Europe; above the 40th parallel is dominated mostly by Europe though it has the north half of North America as well).

In summery, the main results were that attacks involving techniques that might cause property damage (especially explosives) and attacks involving hostages tended to have higher success. On the other hand, assassination (especially in first world countries such as most of Europe and North America) were much less likely to be successful. Surprisingly variables such as the ones relating to weapon type were not very relevant. It is important to note that the accuracy, even on the best model, was not particularly high. This might be due to a variety of reasons. Terrorist attacks are complicated and rely on many different factors; some of these factors might have been outside the scope of the data we had to work with.

## Question 2: Predicting the Number of Casualties

Upon loading in the dataset, I created a new column “ncasualties” by adding together the values in the “nkill” and “nwounded” columns. I then removed all values from the dataset where “ncasualties” was NA or “success” equaled 0 (as we were only looking at casualties in successful attacks). I then looked at the distribution of the casualty values (*Figure* 13). We can see that the data is very skewed towards 0; since we know that all these attacks are success, that means that the majority of attacks are fairly small scale.

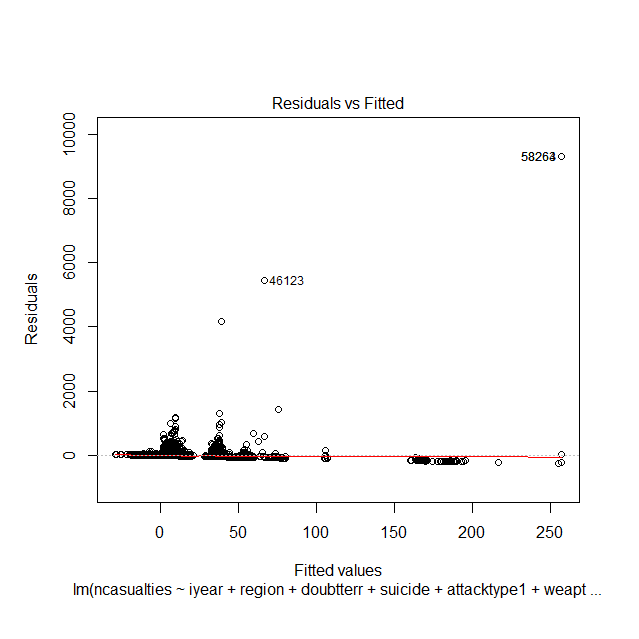
Using similar methods as before I eliminated variables that had too many NAs etc. until I was left with the following columns: “iyear”, “imonth”, “iday”, “extended”, “region”, “latitude”, “longitude”, “specificity”, “vicinity”, “crit1”, “crit2”, “crit3”, “doubtterr”, “multiple”, “suicide”, “attacktype1”, “targtype1”, “targsubtype1”, “guncertain1”, “individual”, “weaptype1”, “property”, “ishostkid”, “INT\_LOG”, “INT\_IDEO”, “INT\_MISC”, “INT\_ANY”, and “ncasualties”.

*Figure 14 – Histogram of “ncasualties”*

My first idea was to use linear regression. After some simple backwards selection, this was the result:

Call:  
lm(formula = ncasualties ~ iyear + region + doubtterr + suicide +   
 attacktype1 + weaptype1 + property, data = data.reduced)  
  
Residuals:  
Min 1Q Median 3Q Max   
-253.8 -4.7 -2.1 1.1 9317.2   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 213.10736 28.51301 7.474 7.82e-14 \*\*\*  
iyear -0.06943 0.01310 -5.298 1.17e-07 \*\*\*  
region2 -4.46732 0.99273 -4.500 6.80e-06 \*\*\*  
region3 -5.71651 0.92051 -6.210 5.31e-10 \*\*\*  
region4 5.62777 1.93936 2.902 0.003710 \*\*   
region5 -3.42710 0.97944 -3.499 0.000467 \*\*\*  
region6 -2.23579 0.91833 -2.435 0.014908 \*   
region7 -2.49325 2.15763 -1.156 0.247868   
region8 -6.10576 0.93419 -6.536 6.35e-11 \*\*\*  
region9 -3.40691 1.11726 -3.049 0.002294 \*\*   
region10 -1.54492 0.91766 -1.684 0.092272 .   
region11 -0.74989 0.96563 -0.777 0.437409   
region12 -6.10587 3.01056 -2.028 0.042547 \*   
doubtterr0 4.13478 0.46674 8.859 < 2e-16 \*\*\*  
doubtterr1 3.57055 0.53652 6.655 2.84e-11 \*\*\*  
suicide1 30.72977 0.64205 47.862 < 2e-16 \*\*\*  
attacktype12 2.11076 0.48842 4.322 1.55e-05 \*\*\*  
attacktype13 -0.33209 0.76624 -0.433 0.664720   
attacktype14 36.33662 2.00327 18.139 < 2e-16 \*\*\*  
attacktype15 2.87128 1.58366 1.813 0.069823 .   
attacktype16 1.01797 0.67555 1.507 0.131847   
attacktype17 -3.37531 0.89968 -3.752 0.000176 \*\*\*  
attacktype18 -17.68287 1.92286 -9.196 < 2e-16 \*\*\*  
attacktype19 4.86736 0.98620 4.935 8.00e-07 \*\*\*  
weaptype12 -1.44303 12.12562 -0.119 0.905271   
weaptype13 -79.44702 33.18771 -2.394 0.016673 \*   
weaptype15 -71.22404 11.90049 -5.985 2.17e-09 \*\*\*  
weaptype16 -70.40969 11.91669 -5.908 3.46e-09 \*\*\*  
weaptype17 -98.22634 15.82346 -6.208 5.39e-10 \*\*\*  
weaptype18 -72.42853 11.91641 -6.078 1.22e-09 \*\*\*  
weaptype19 -67.19917 11.87838 -5.657 1.54e-08 \*\*\*  
weaptype110 110.06501 12.45258 8.839 < 2e-16 \*\*\*  
weaptype111 -70.65288 12.52367 -5.642 1.69e-08 \*\*\*  
weaptype112 -65.69406 12.74178 -5.156 2.53e-07 \*\*\*  
weaptype113 -72.25010 11.91555 -6.064 1.34e-09 \*\*\*  
property0 -2.58906 0.41002 -6.314 2.72e-10 \*\*\*  
property1 1.38415 0.38757 3.571 0.000355 \*\*\*  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
Residual standard error: 43.87 on 145380 degrees of freedom  
(1 observation deleted due to missingness)  
Multiple R-squared: 0.03811, Adjusted R-squared: 0.03788   
F-statistic: 160 on 36 and 145380 DF, p-value: < 2.2e-16

The R-squared very low. Clearly this model will need some work.

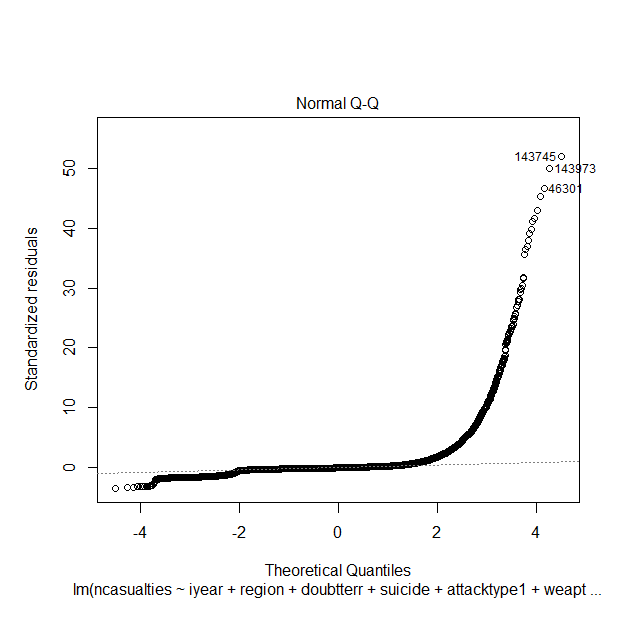


*Figure 15 – Residual Plot*

Looking at residuals plot (*Figure 15*), we notice some extreme outliers. It turns out that point 58264 (and 58263) correspond to 9/11, and the other outlying points are other major attacks. In total there are only 12 attacks with over 1000 casualties out of the over 100,000 attacks remaining in the dataset. Since it does not seem reasonable that we could ever predict attacks that fall outside the normal casualty range by so much, I removed the 12 points from the dataset. Rerunning the model, I got the following results:

Call:  
lm(formula = ncasualties ~ iyear + region + doubtterr + suicide +   
 attacktype1 + weaptype1, data = data.reduced.2)  
  
Residuals:  
Min 1Q Median 3Q Max   
-60.24 -4.64 -2.55 0.42 909.19   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 262.77283 10.94803 24.002 < 2e-16 \*\*\*  
iyear -0.10688 0.00499 -21.420 < 2e-16 \*\*\*  
region2 2.05599 0.39423 5.215 1.84e-07 \*\*\*  
region3 0.77755 0.36561 2.127 0.03345 \*   
region4 5.48763 0.77185 7.110 1.17e-12 \*\*\*  
region5 2.91011 0.38937 7.474 7.83e-14 \*\*\*  
region6 4.20847 0.36508 11.527 < 2e-16 \*\*\*  
region7 4.39938 0.85798 5.128 2.94e-07 \*\*\*  
region8 0.24785 0.37144 0.667 0.50460   
region9 2.66134 0.44413 5.992 2.07e-09 \*\*\*  
region10 5.16334 0.36498 14.147 < 2e-16 \*\*\*  
region11 5.69176 0.38362 14.837 < 2e-16 \*\*\*  
region12 1.31177 1.19733 1.096 0.27326   
doubtterr0 3.47393 0.18561 18.717 < 2e-16 \*\*\*  
doubtterr1 3.12688 0.21334 14.657 < 2e-16 \*\*\*  
suicide1 25.23968 0.25544 98.809 < 2e-16 \*\*\*  
attacktype12 3.95146 0.18289 21.605 < 2e-16 \*\*\*  
attacktype13 1.58591 0.29736 5.333 9.66e-08 \*\*\*  
attacktype14 1.15050 0.79741 1.443 0.14908   
attacktype15 4.58751 0.62573 7.331 2.29e-13 \*\*\*  
attacktype16 1.32022 0.26812 4.924 8.49e-07 \*\*\*  
attacktype17 1.11295 0.34404 3.235 0.00122 \*\*   
attacktype18 4.59734 0.76252 6.029 1.65e-09 \*\*\*  
attacktype19 6.25976 0.38504 16.257 < 2e-16 \*\*\*  
weaptype12 -26.72508 4.82397 -5.540 3.03e-08 \*\*\*  
weaptype13 -68.16785 13.20023 -5.164 2.42e-07 \*\*\*  
weaptype15 -53.80673 4.73306 -11.368 < 2e-16 \*\*\*  
weaptype16 -51.92758 4.73898 -10.958 < 2e-16 \*\*\*  
weaptype17 -54.93696 6.29412 -8.728 < 2e-16 \*\*\*  
weaptype18 -54.99424 4.73892 -11.605 < 2e-16 \*\*\*  
weaptype19 -53.44174 4.72445 -11.312 < 2e-16 \*\*\*  
weaptype110 -49.41042 4.95702 -9.968 < 2e-16 \*\*\*  
weaptype111 -53.15076 4.98055 -10.672 < 2e-16 \*\*\*  
weaptype112 -54.77565 5.06758 -10.809 < 2e-16 \*\*\*  
weaptype113 -54.60392 4.73920 -11.522 < 2e-16 \*\*\*  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
Residual standard error: 17.45 on 145370 degrees of freedom  
(1 observation deleted due to missingness)  
Multiple R-squared: 0.09148, Adjusted R-squared: 0.09127   
F-statistic: 430.5 on 34 and 145370 DF, p-value: < 2.2e-16

While the R-squared is more than double the previous model, it is still far from good.

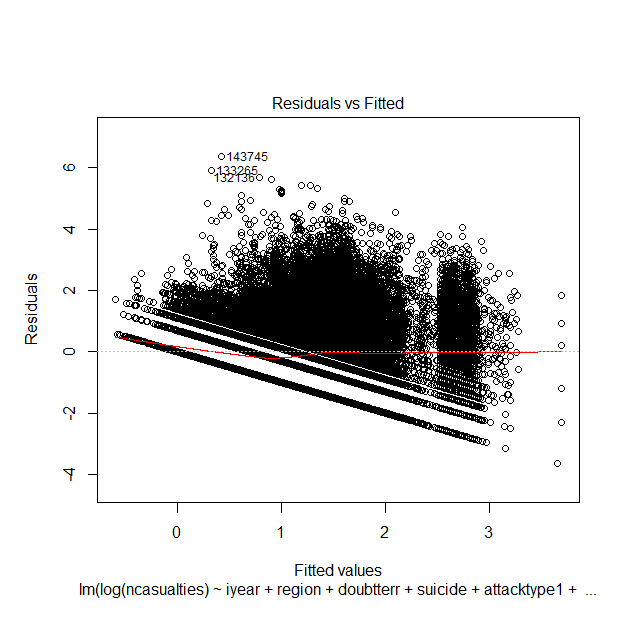


*Figure 16 – Normal Q-Q plot for linear regression model*

We can clearly see from the above plot that the error is not normally distributed. Since we know from before that the casualty values are severely right skewed, I attempted a log transformation on y.

Call:  
lm(formula = log(ncasualties) ~ iyear + region + doubtterr +   
 suicide + attacktype1 + weaptype1, data = data.reduced.2)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-3.6547 -0.7535 -0.1097 0.6325 6.3843   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 38.5660985 0.8014083 48.123 < 2e-16 \*\*\*  
iyear -0.0191543 0.0003804 -50.347 < 2e-16 \*\*\*  
region2 0.5161453 0.0398330 12.958 < 2e-16 \*\*\*  
region3 0.3752907 0.0383190 9.794 < 2e-16 \*\*\*  
region4 0.7545299 0.0762236 9.899 < 2e-16 \*\*\*  
region5 0.3284338 0.0386982 8.487 < 2e-16 \*\*\*  
region6 0.5115517 0.0373911 13.681 < 2e-16 \*\*\*  
region7 0.4368656 0.0667876 6.541 6.14e-11 \*\*\*  
region8 -0.1139399 0.0400401 -2.846 0.00443 \*\*   
region9 0.2562456 0.0424318 6.039 1.56e-09 \*\*\*  
region10 0.5644484 0.0374099 15.088 < 2e-16 \*\*\*  
region11 0.7222902 0.0382417 18.888 < 2e-16 \*\*\*  
region12 0.3141168 0.1177996 2.667 0.00767 \*\*   
doubtterr0 0.3580619 0.0153206 23.371 < 2e-16 \*\*\*  
doubtterr1 0.3194174 0.0165441 19.307 < 2e-16 \*\*\*  
suicide1 1.1566424 0.0151576 76.308 < 2e-16 \*\*\*  
attacktype12 0.8505029 0.0112768 75.420 < 2e-16 \*\*\*  
attacktype13 0.7238033 0.0207265 34.922 < 2e-16 \*\*\*  
attacktype14 0.6755933 0.0853396 7.917 2.47e-15 \*\*\*  
attacktype15 1.2180299 0.0634637 19.193 < 2e-16 \*\*\*  
attacktype16 0.5429563 0.0204749 26.518 < 2e-16 \*\*\*  
attacktype17 0.7525432 0.0434218 17.331 < 2e-16 \*\*\*  
attacktype18 0.6898279 0.0518974 13.292 < 2e-16 \*\*\*  
attacktype19 1.2648407 0.0277521 45.576 < 2e-16 \*\*\*  
weaptype12 0.7668915 0.2834906 2.705 0.00683 \*\*   
weaptype13 -1.0817681 1.0515435 -1.029 0.30360   
weaptype15 -0.7113235 0.2775516 -2.563 0.01038 \*   
weaptype16 -0.1504465 0.2781002 -0.541 0.58852   
weaptype18 -0.6217755 0.2795682 -2.224 0.02615 \*   
weaptype19 -0.7853500 0.2769776 -2.835 0.00458 \*\*   
weaptype110 0.0423404 0.2930375 0.144 0.88512   
weaptype111 0.6673424 0.3964585 1.683 0.09233 .   
weaptype112 -0.7349479 0.3089684 -2.379 0.01738 \*   
weaptype113 -0.8011802 0.2783216 -2.879 0.00400 \*\*   
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  
  
Residual standard error: 1.014 on 95812 degrees of freedom  
(1 observation deleted due to missingness)  
Multiple R-squared: 0.2297, Adjusted R-squared: 0.2295   
F-statistic: 866 on 33 and 95812 DF, p-value: < 2.2e-16

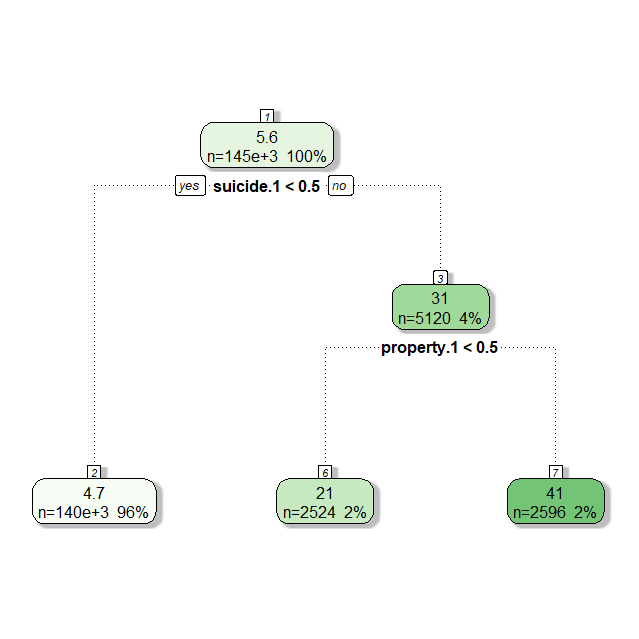
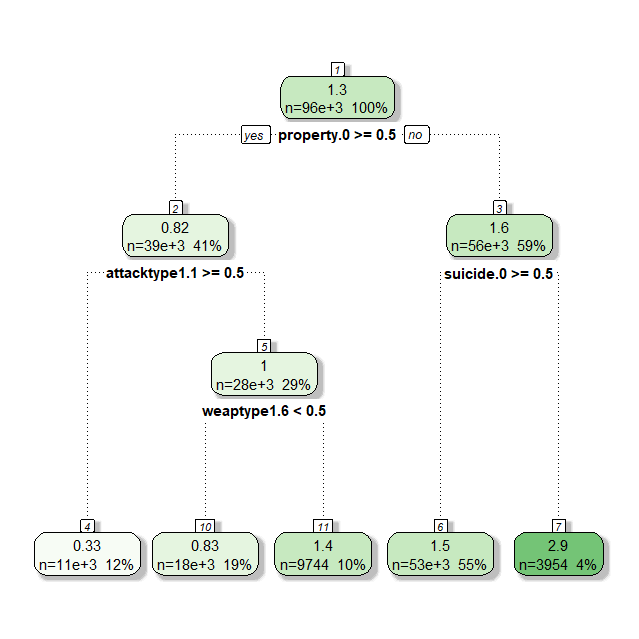
While the R-squared is substantially better, that cannot be accurately compared as we transformed y.



*Figure 17 – Residuals plot for log transformed linear regression model*

Looking at the residuals plot (*Figure 17*), it was clear that there was some sort of pattern in the error, implying that the model was not accurately representing the data. Interaction terms were tried to very little effect, so it seemed clear at this point that linear regression was probably not the best model to answer this question.

Next, I tried tree models. Using both y and log(y) as the response, I ended up with the following two trees (*Figure 18* and *19*).



*Figure 18 and 19 – Left: unbalanced tree model; right: log transformed tree model.*

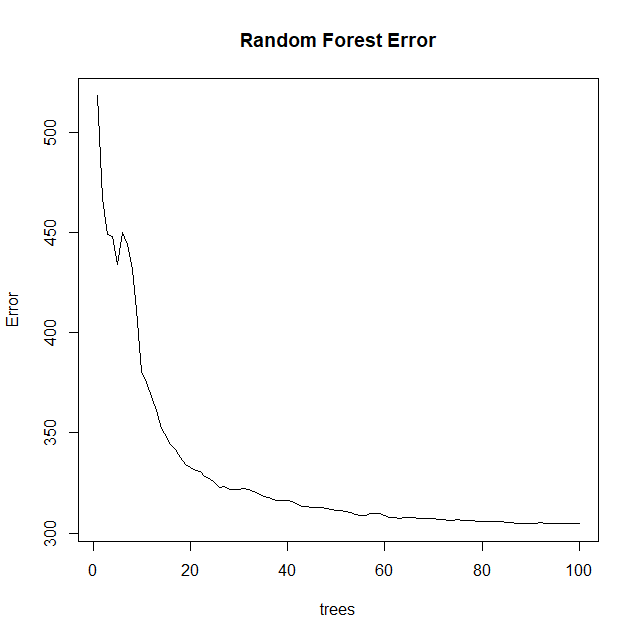
Their R-squared values are: 0.08451011 and 0.06648582 respectively, so neither are very good on that front. However, the variables the model finds important are interesting. Suicide attacks seem to result in higher casualties, as well as property damage. Explosive weapons (“weaptype1.6”) increases casualties, while assassinations (“attacktype1.1”) decreases casualties. All these observations do make intuitive sense, however there is not much more we can get out of a single tree.

Before moving on to random forest, I decided to give SVMs a try. Using a radial kernel, I got the following results:



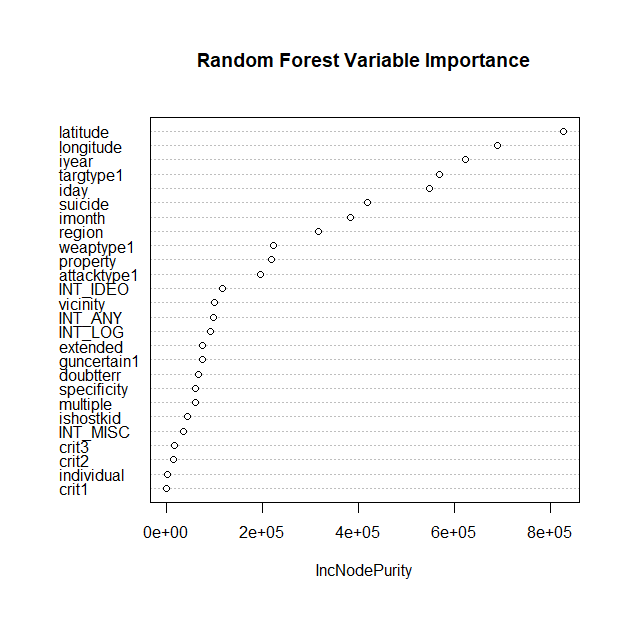
Cost factor of 100 seems to be the best, as that is where the R-squared on the testing data maxes out. Not a huge amount was learned here, however it was an interesting experiment that resulted in the best model so far.

Next, I moved on to random forest.



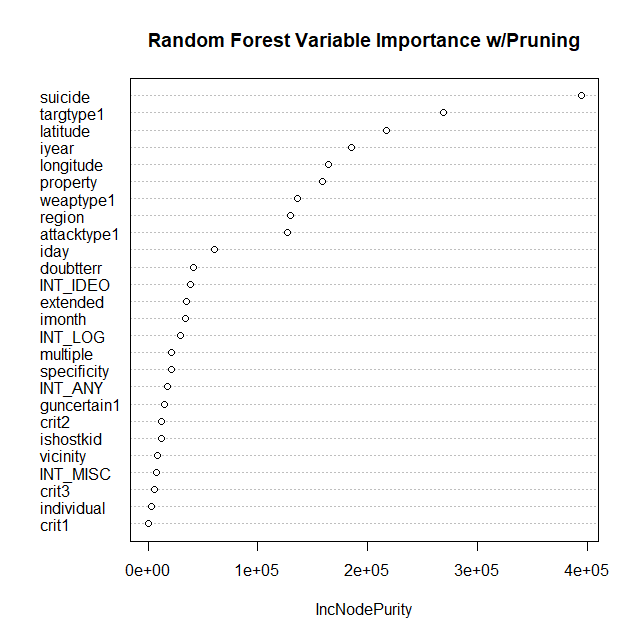
*Figure 20 – Plot that shows error vs. different number of trees for random forest*

Looking at the amount of error vs. the number of trees (*Figure 20*), around 60 trees seemed like the right number. With 60 trees I got an R-squared of 0.2019902 (on testing data as always), the best model so far.



*Figure 21 – Variable importance plot for random forest model*

Very few of the variables that were important in the basic tree models or in linear regression show up here, an exception being “suicide” and “property” though both seem drastically less important then they were in previous models. Most of the important variables in this model are time/location based, the target type also seems to be an important factor. This started to make me a bit suspicious that perhaps the random forest was overfitting the data by using the time and place to pinpoint the specific attack; an R-squared of 0.8545023 on training data seemed to confirm this. Setting the “nodesize” parameter to 100, gave much different results. The R-squared on testing data was a bit lower (0.1606361) however, as you can see below, the variable importance levels were much closer to what would be expected (notice suicide has about the same level of importance as the previous model, however longitude and latitude don’t have overinflated importance).



*Figure 21 – Variable importance plot for pruned random forest model*

In summary, it seems that predicting the exact value for casualties is difficult problem. Both random forest and SVMs seem to have the best results; suicide attacks seemed to be the most significant indicator, followed by target and location (temporal and special) variables, finally followed by factors such as weapon and attack type.