shredCVL using Logistic regression in R

1 March, 2018

Contents

Introduction

*****For the second project we'll explore user data from shredCVL to identify accounts likely belonging to bots. The data set has variables about profile configuration (`default_profile`, `default_profile_image`), connectivity (`friendsCount`, `followersCount`), and some information about the nature of their tweets (`diversity`, `mean_mins_between_tweets`). Additionally, there's an outcome variable called `bot` that denotes whether the account belongs to a bot (`bot == 1`) or to a human (`bot == 0`).*********

Exploratory data analysis

We've got a brand new data set, so let's familiarize ourselves by conducting an exploratory data analysis. Let's start by summarizing the whole data set to see what the variable values are.

```
shredCVL <- readRDS("capstone dataset/shredCVL")

summary(shredCVL)
```

```
##   screenName             bot         statusesCount      friendsCount
##  Length:2618        Min.   :0.000   Min.   :      0   Min.   :    0
##  Class :character   1st Qu.:0.000   1st Qu.:   2524   1st Qu.:  258
##  Mode  :character   Median :0.000   Median :  13042   Median :  661
##                     Mean   :0.195   Mean   :  45605   Mean   : 1683
##                     3rd Qu.:0.000   3rd Qu.:  41709   3rd Qu.: 1678
##                     Max.   :1.000   Max.   :1742257   Max.   :68232
##                                     NA's   :29        NA's   :29
##  followersCount     listedCount      acct_created
##  Min.   :      0   Min.   :    0   Min.   :2007-02-08 04:24:56
##  1st Qu.:    277   1st Qu.:    4   1st Qu.:2010-04-16 14:56:21
##  Median :    915   Median :   22   Median :2012-09-06 16:14:22
##  Mean   :  10366   Mean   :  143   Mean   :2012-11-08 23:14:03
##  3rd Qu.:   2296   3rd Qu.:  104   3rd Qu.:2015-07-21 08:54:27
##  Max.   :6604309   Max.   :39981   Max.   :2017-12-11 13:58:41
##  NA's   :29        NA's   :29
##  julianCreated        acct_age          langDiv
##  Length:2618        Min.   :   1361   Min.   : 0.39
```

```
##  Class :difftime   1st Qu.: 766986   1st Qu.: 2.95
##  Mode  :numeric    Median :2275666   Median : 5.57
##                    Mean   :2221613   Mean   : 5.59
##                    3rd Qu.:3534304   3rd Qu.: 7.50
##                    Max.   :5209655   Max.   :29.07
##                                      NA's   :193
##  mean_time_betwn_tweets     App                Count         App.BoN
##  Min.   :      0        Length:2618        Min.   :   1   Min.   :0.00
##  1st Qu.:     40        Class :character   1st Qu.:  62   1st Qu.:0.00
##  Median :    127        Mode  :character   Median : 412   Median :0.00
##  Mean   :  12156                           Mean   : 772   Mean   :0.03
##  3rd Qu.:    549                           3rd Qu.: 657   3rd Qu.:0.00
##  Max.   :3598898                           Max.   :6238   Max.   :1.00
##  NA's   :38                                NA's   :164    NA's   :164
##      mCount
##  Min.   :1.00
##  1st Qu.:1.00
##  Median :1.00
##  Mean   :1.29
##  3rd Qu.:1.00
##  Max.   :4.00
##
```

From the summary, we can see that there are a couple `factor` variables in the data set, `bot`, `default_profile`, `default_profile_image` and `geo_enabled`. Before exploring further, let's first tell R that those columns represent categorical variables.

```
names(shredCVL)
```

```
##  [1] "screenName"             "bot"
##  [3] "statusesCount"          "friendsCount"
##  [5] "followersCount"         "listedCount"
##  [7] "acct_created"           "julianCreated"
##  [9] "acct_age"               "langDiv"
## [11] "mean_time_betwn_tweets" "App"
## [13] "Count"                  "App.BoN"
## [15] "mCount"
```

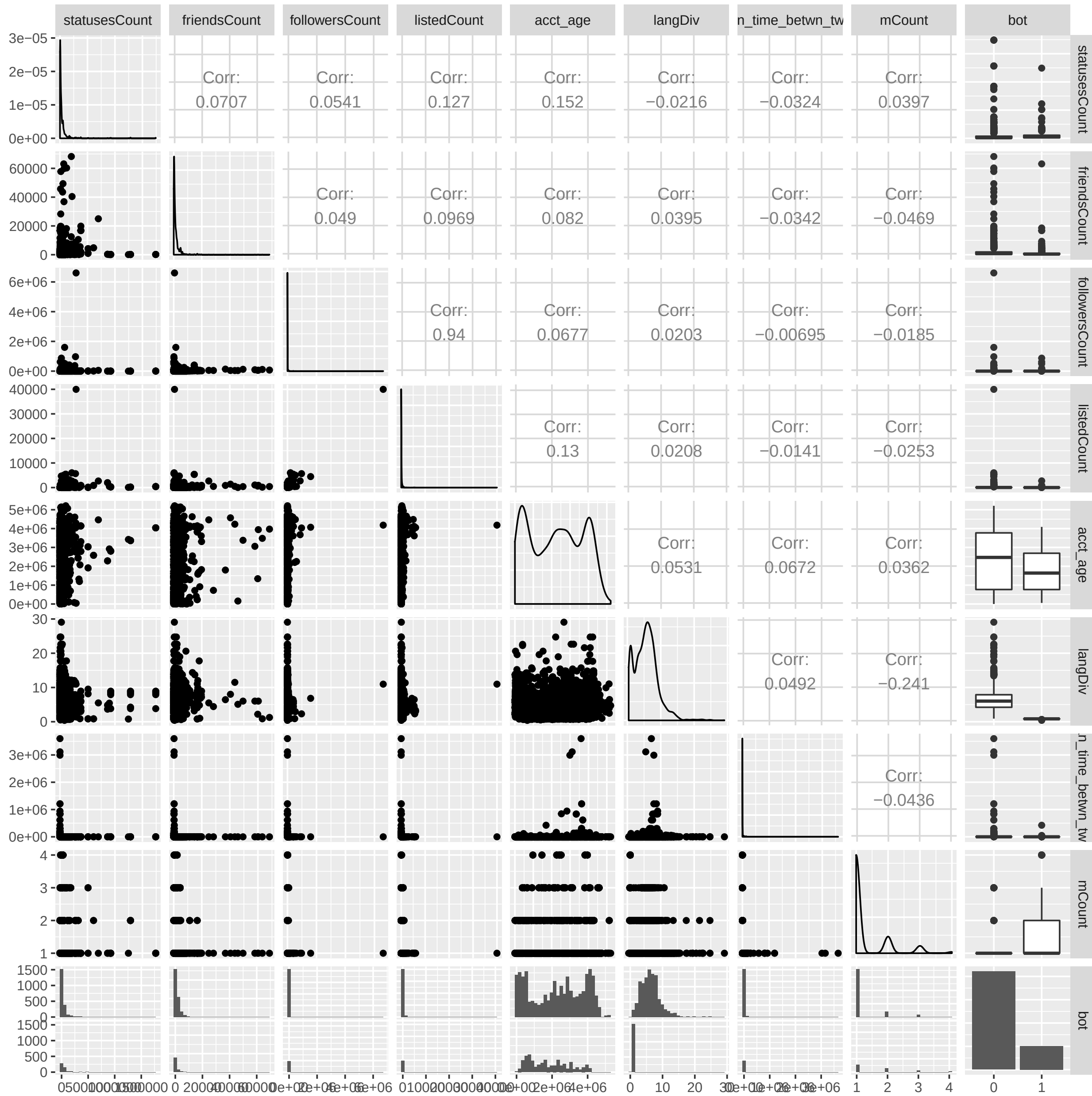
```
shredCVL$bot = factor(shredCVL$bot)
shredCVL$App.BoN = factor(shredCVL$App.BoN)

shredCVL <- shredCVL %>%
    select(statusesCount,friendsCount,followersCount,listedCount,acct_age,langDiv,mean_time_betwn_tweet

summary(shredCVL)
```

```
##  statusesCount      friendsCount   followersCount     listedCount
##  Min.   :      0   Min.   :    0   Min.   :      0   Min.   :    0
##  1st Qu.:   2524   1st Qu.:  258   1st Qu.:    277   1st Qu.:    4
##  Median :  13042   Median :  661   Median :    915   Median :   22
##  Mean   :  45605   Mean   : 1683   Mean   :  10366   Mean   :  143
##  3rd Qu.:  41709   3rd Qu.: 1678   3rd Qu.:   2296   3rd Qu.:  104
##  Max.   :1742257   Max.   :68232   Max.   :6604309   Max.   :39981
##  NA's   :29        NA's   :29      NA's   :29        NA's   :29
##     acct_age          langDiv      mean_time_betwn_tweets     mCount
##  Min.   :   1361   Min.   : 0.39   Min.   :      0        Min.   :1.00
```

```
##  1st Qu.: 766986   1st Qu.: 2.95   1st Qu.:     40        1st Qu.:1.00  
##  Median :2275666   Median : 5.57   Median :    127        Median :1.00  
##  Mean   :2221613   Mean   : 5.59   Mean   :  12156        Mean   :1.29  
##  3rd Qu.:3534304   3rd Qu.: 7.50   3rd Qu.:    549        3rd Qu.:1.00  
##  Max.   :5209655   Max.   :29.07   Max.   :3598898        Max.   :4.00  
##                    NA's   :193     NA's   :38                          
##  bot     
##  0:2107  
##  1: 511  
##          
##          
##          
##          
##          
```

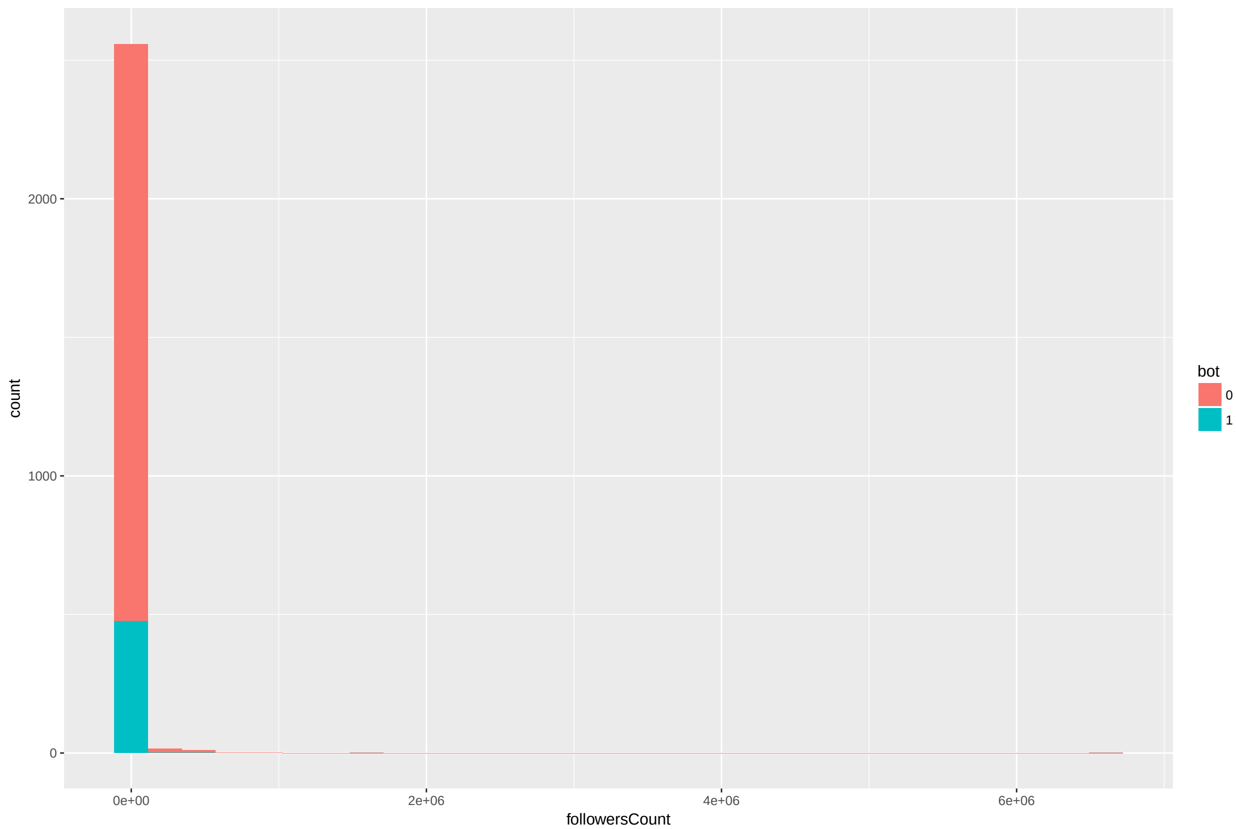
Like before, we can evaluate many relationships simultaneously with `ggpairs`.

```r
# inspect many trends with ggpairs
ggpairs(shredCVL)
```

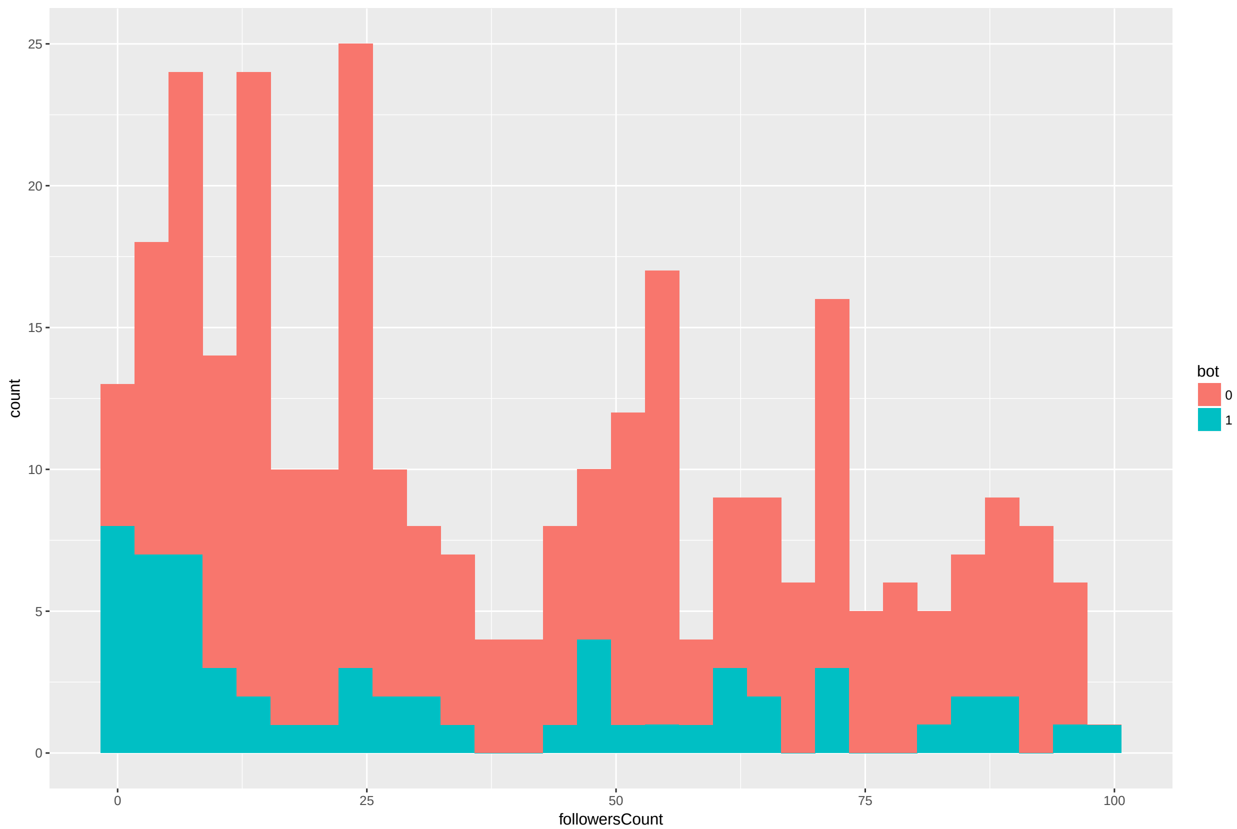


Once we have some initial hypotheses we can make more specific plots.

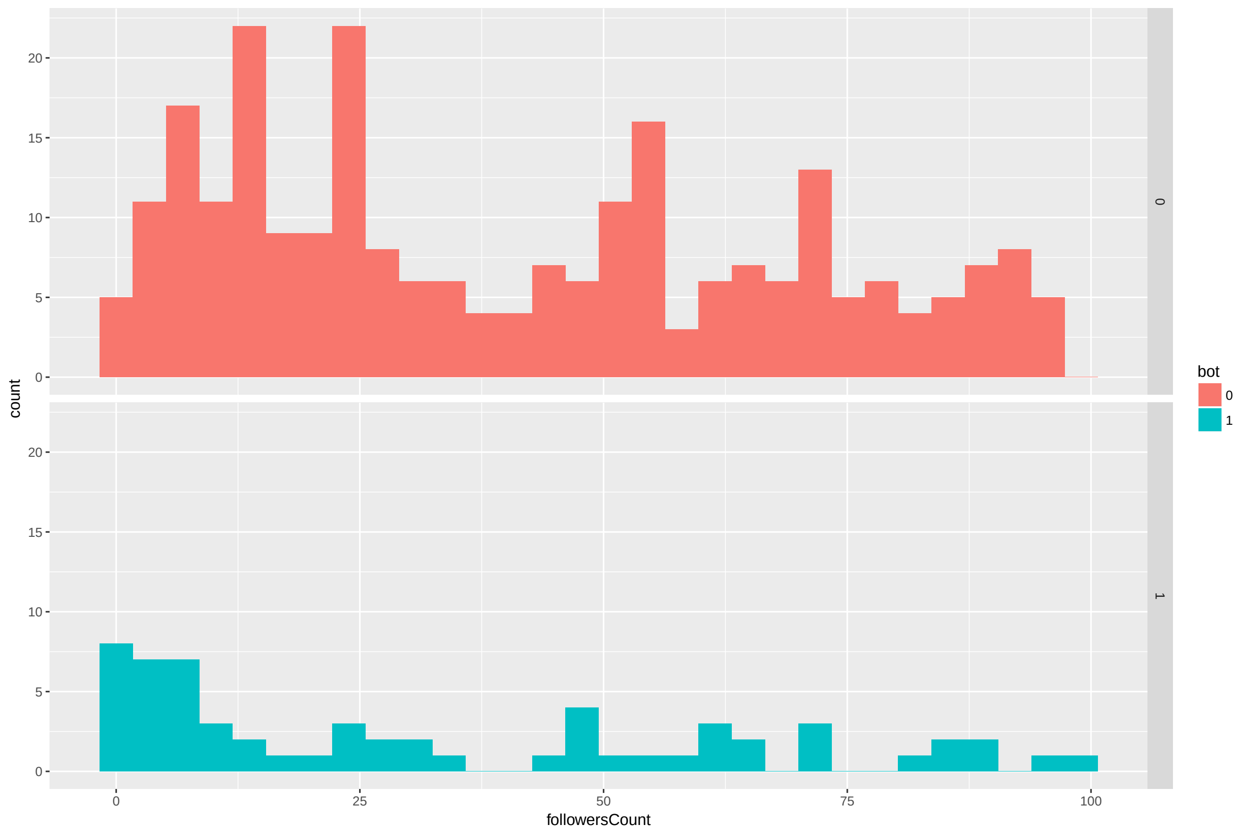
```
ggplot(shredCVL, aes(x = followersCount, fill = bot)) +
  geom_histogram()
```



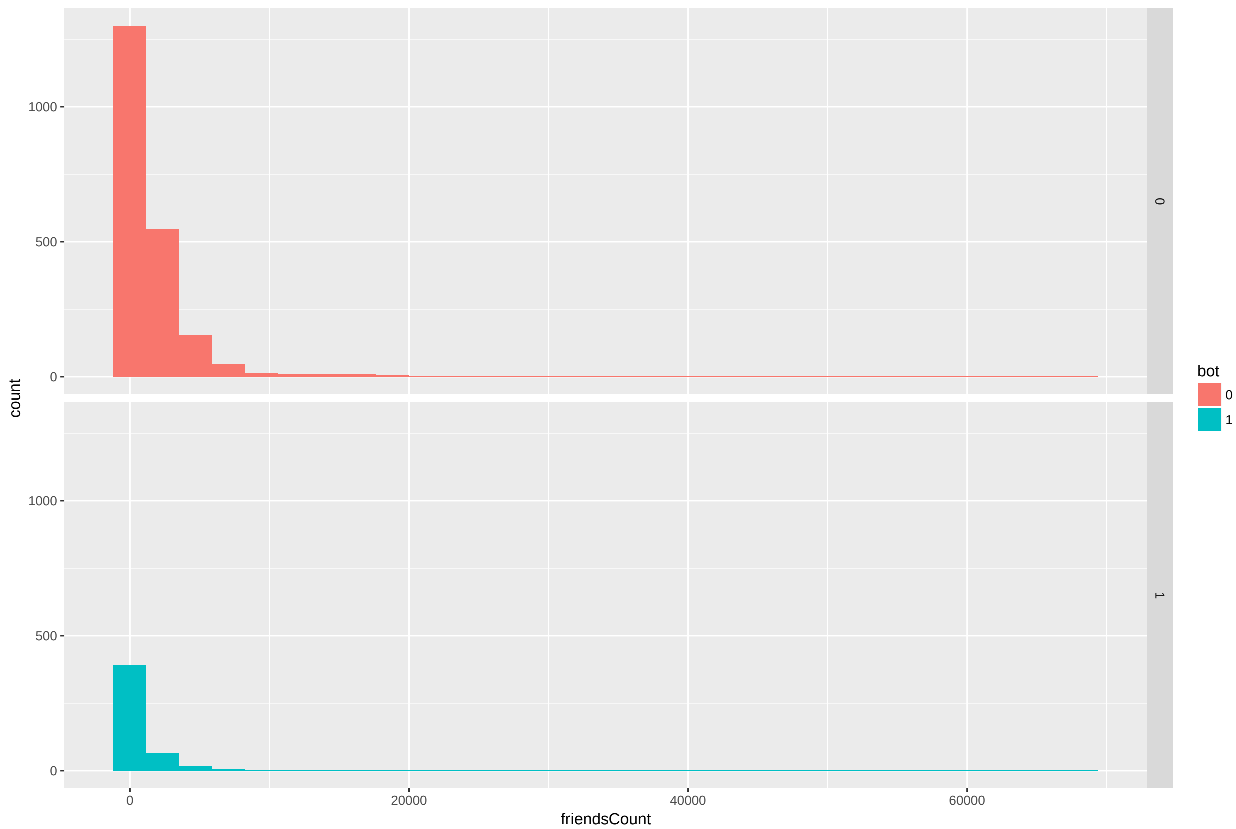

```r
# Some people have a lot of followers, but most don't. we need to lob off
# the long tail so we can see the distribution better
ggplot(filter(shredCVL, followersCount < 100),
       aes(x = followersCount, fill = bot)) +
  geom_histogram()
```



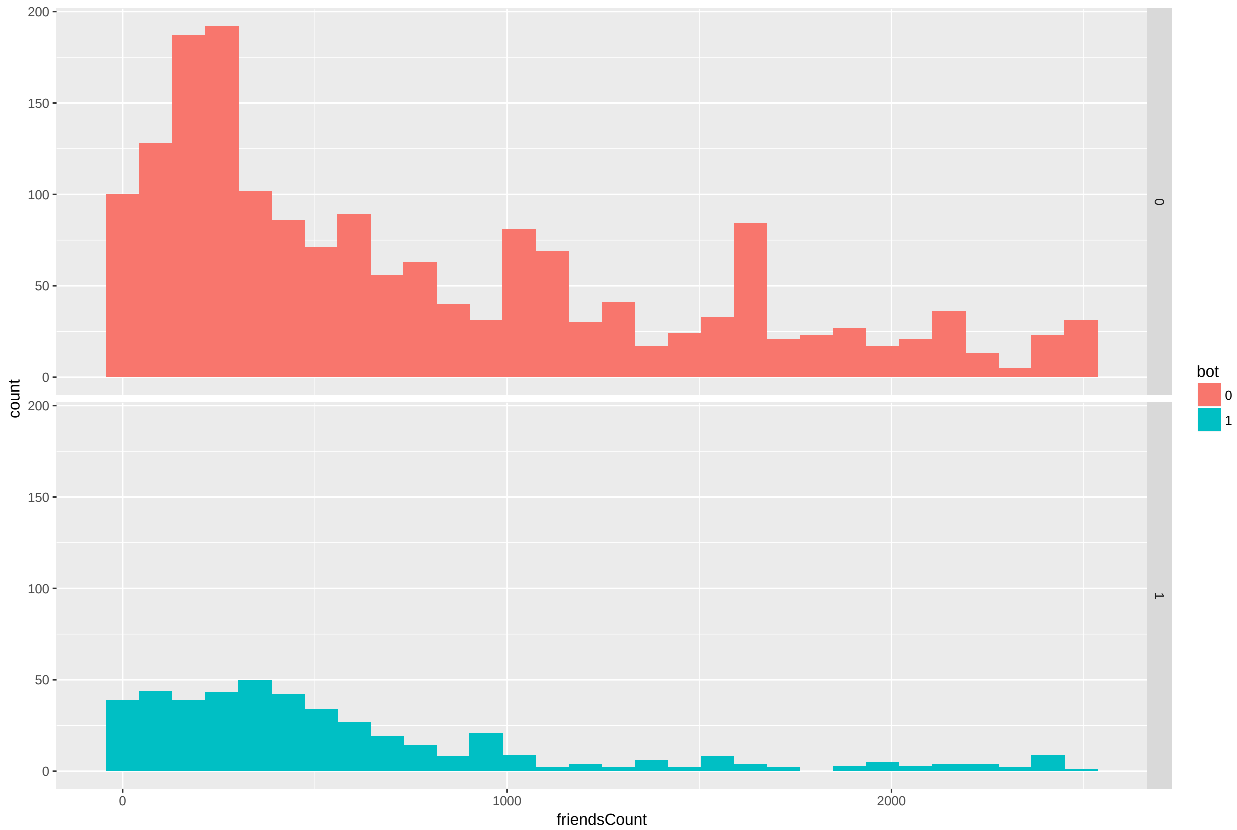

```
ggplot(filter(shredCVL, followersCount < 100),
       aes(x = followersCount, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



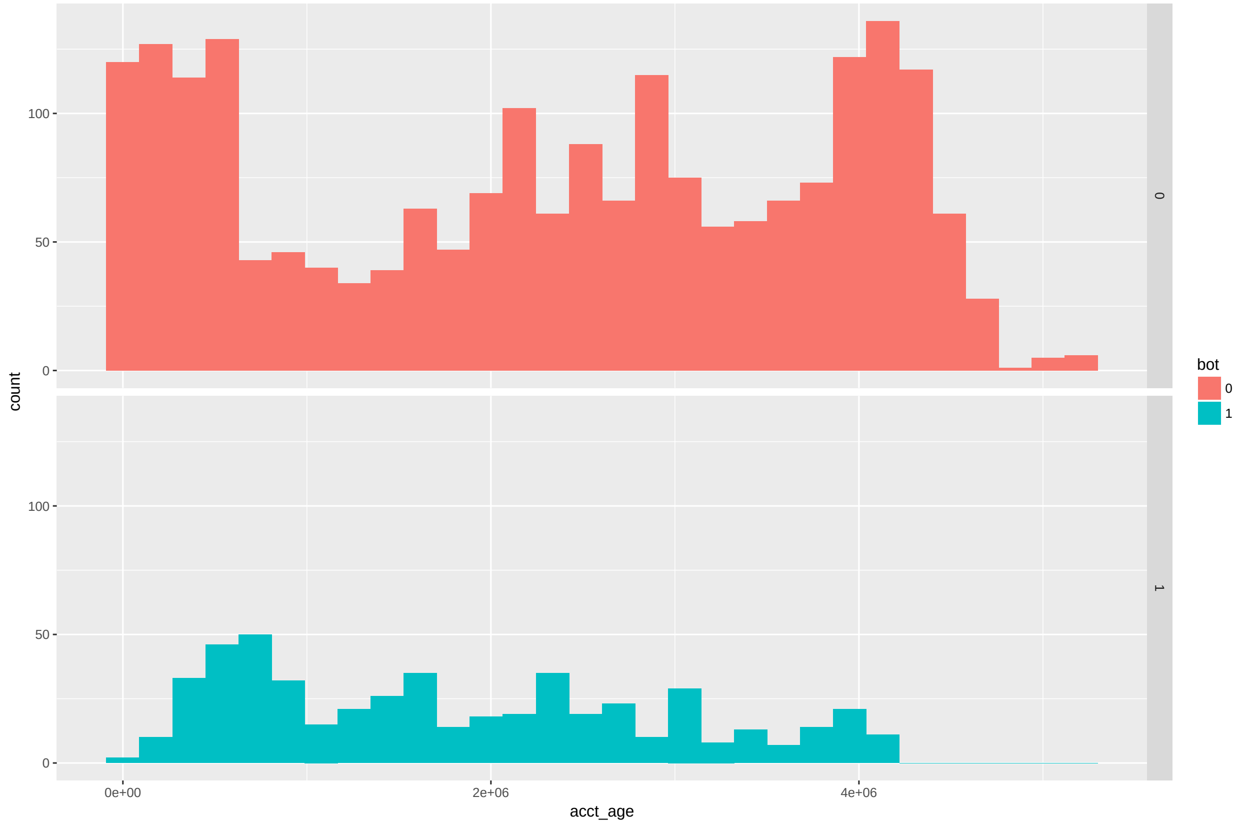

```r
# how about the number of people they follow?
ggplot(shredCVL, aes(x = friendsCount, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



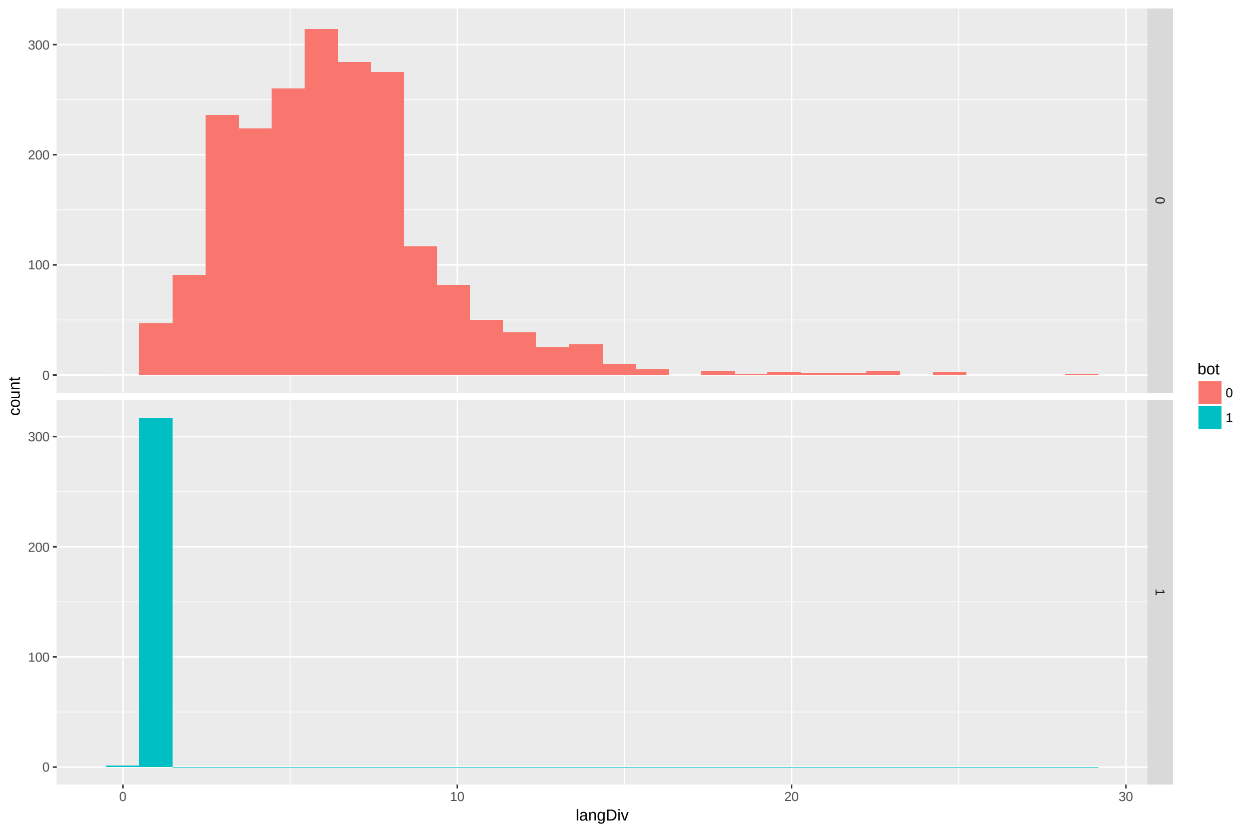

```r
# it's a little hard to see
ggplot(filter(shredCVL, friendsCount < 2500),
       aes(x = friendsCount, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```




```r
# what about account age?
ggplot(shredCVL, aes(x = acct_age, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```




```r
# lexical diversity
ggplot(shredCVL, aes(x = langDiv, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```

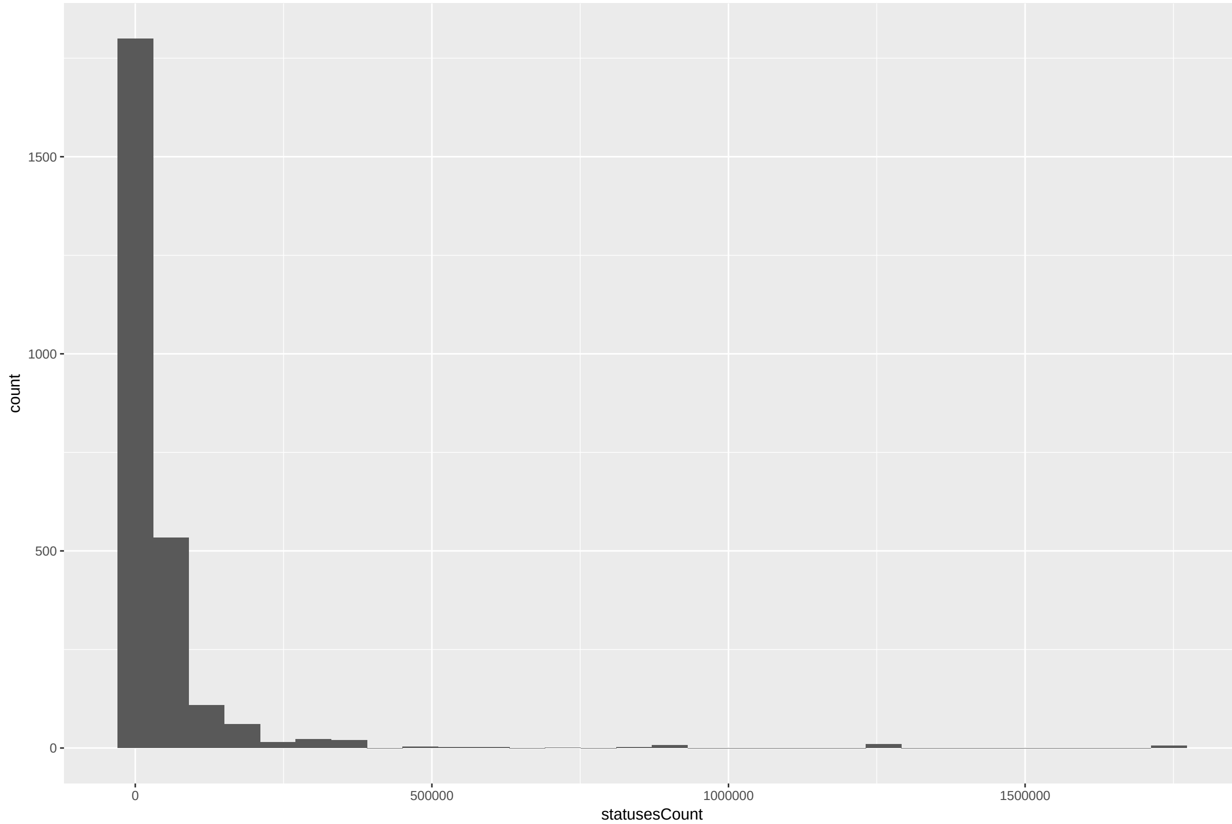



```r
# what are the average values?
avg_diversity =
  shredCVL %>%
    group_by(bot) %>%
    summarize(avg_diversity = mean(langDiv, na.rm = TRUE))

# add it to the plot
ggplot(shredCVL, aes(x = langDiv, fill = bot)) +
  geom_histogram() +
  geom_vline(data = avg_diversity, aes(xintercept = avg_diversity)) +
  facet_grid(bot ~.)
```

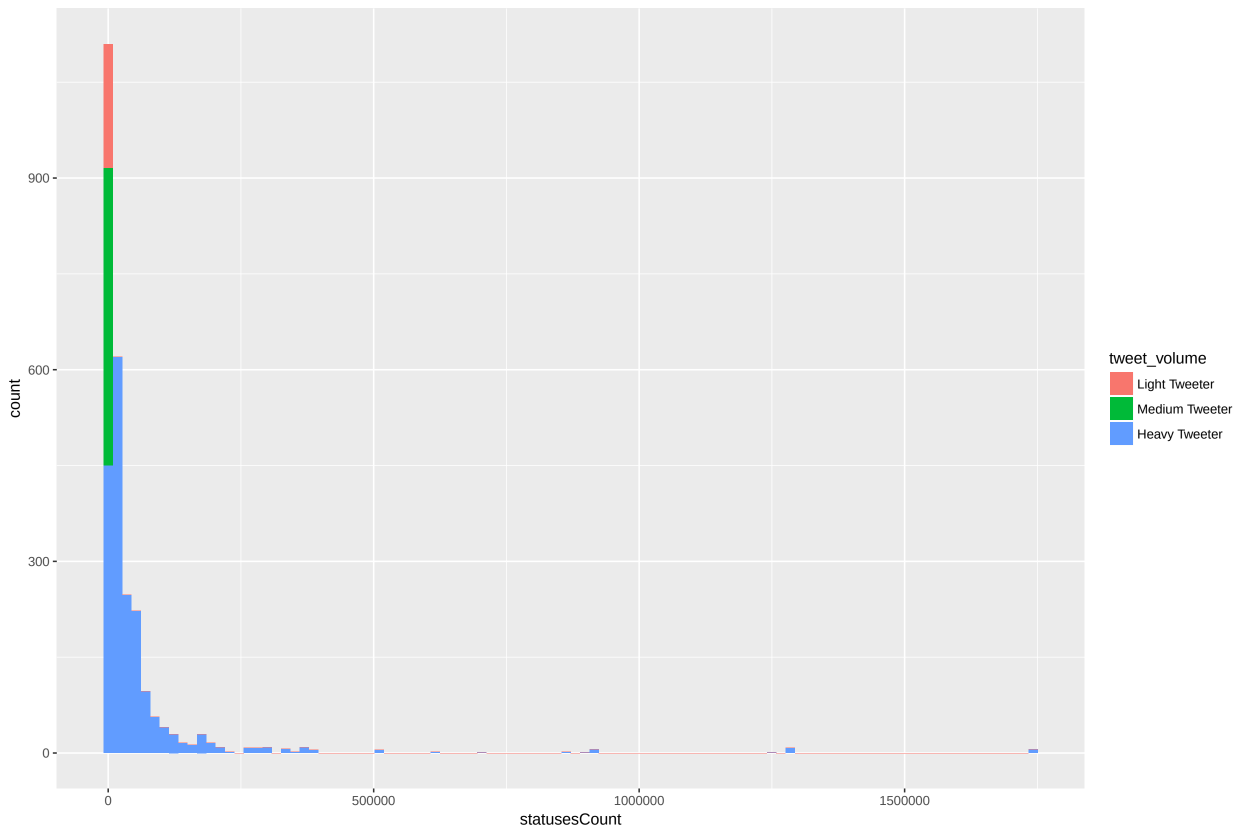
Feature engineering

Feature engineering is the process of creating predictor variables using domain knowledge. We can test hypotheses about the importance of various relationships by creating new predictors that help interrogate those relationships. For example, you might hypothesize a relationship between the number of tweets made and the lexical diversity that is relevant to model. To test that, make a new categorical variable indicating whether an account holder is a 'heavy tweeter', 'medium tweeter' or 'light tweeter':

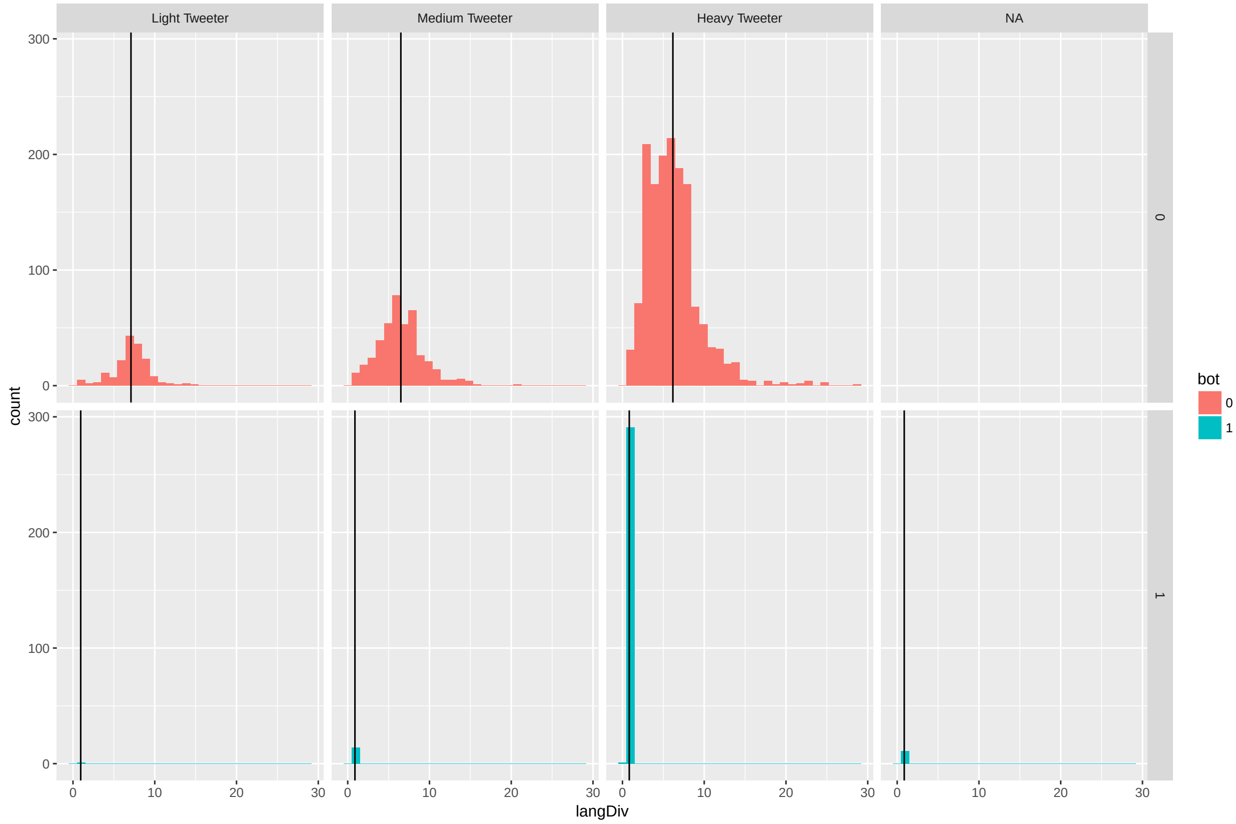
```r
# the number of tweets per account has a long tail
ggplot(shredCVL, aes(x = statusesCount)) +
  geom_histogram()
```




```r
# break into three categories by quantile
#quantile(shredCVL$statusesCount)

# low tweeters will be the bottom 25%,
shredCVL$tweet_volume = NA
shredCVL$tweet_volume = ifelse(shredCVL$statusesCount <= 188,
                               'Light Tweeter',
                               shredCVL$tweet_volume)

shredCVL$tweet_volume = ifelse((shredCVL$statusesCount > 188 & shredCVL$statusesCount <= 2646),
                               'Medium Tweeter',
                               shredCVL$tweet_volume)

shredCVL$tweet_volume = ifelse(shredCVL$statusesCount > 2646,
                               'Heavy Tweeter',
                               shredCVL$tweet_volume)

shredCVL$tweet_volume = factor(shredCVL$tweet_volume, levels = c('Light Tweeter', 'Medium Tweeter', 'Hea

# plot it!
ggplot(shredCVL, aes(x = statusesCount)) +
  geom_histogram(aes(fill = tweet_volume), bins = 100)
```




```
# update the figure
avg_diversity =
  shredCVL %>%
    group_by(bot, tweet_volume) %>%
    summarize(avg_diversity = mean(langDiv, na.rm = TRUE))

ggplot(shredCVL, aes(x = langDiv, fill = bot)) +
  geom_histogram() +
  geom_vline(data = avg_diversity, aes(xintercept = avg_diversity)) +
  facet_grid(bot ~ tweet_volume)
```



Logisitic Regression

Training and testing sets

```
set.seed(243)
shredCVL = na.omit(shredCVL)

# select the training observations
in_train = createDataPartition(y = shredCVL$bot,
                               p = 0.75, # 75% in train, 25% in test
                               list = FALSE)

train = shredCVL[in_train, ]
test = shredCVL[-in_train, ]
```

Training logisitic regressions

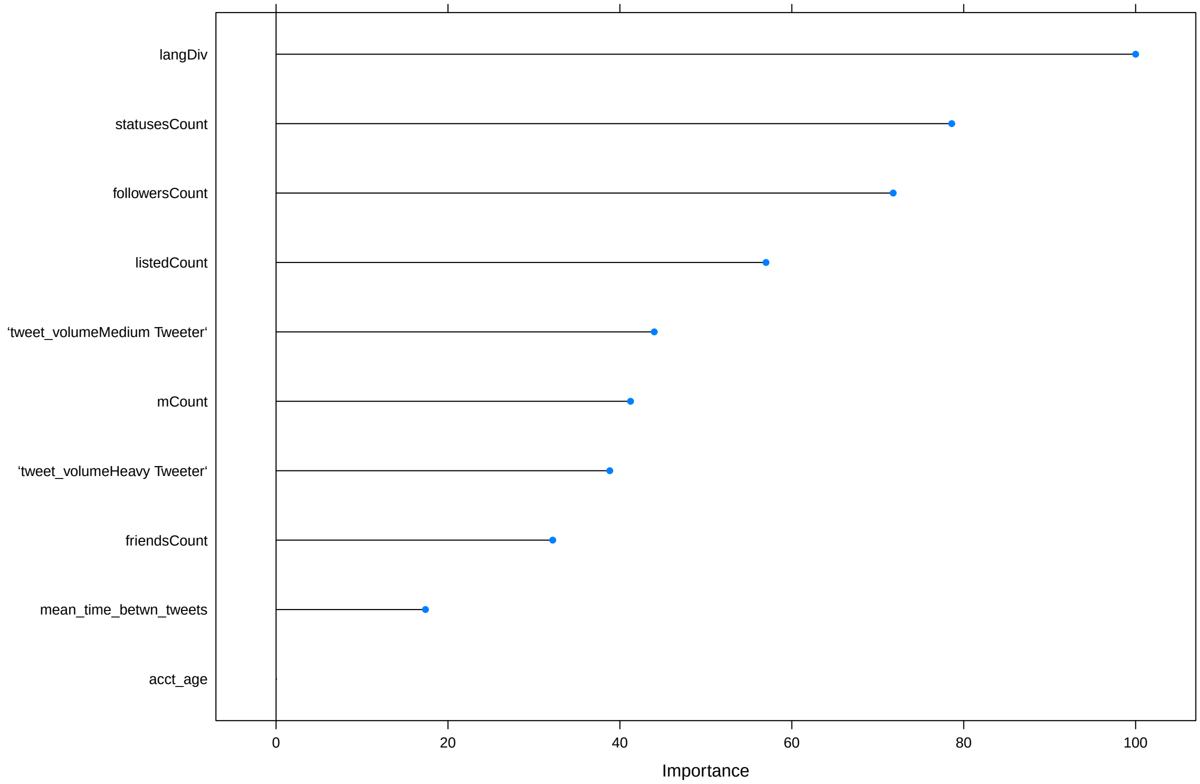
Check out this page for more types of logistic regression to try out.

```
logistic_model = train(bot ~ .,
                       data = train,
                       method = 'glm',
                       family = binomial,
                       preProcess = c('center', 'scale'))
```

```r
summary(logistic_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -2.30    0.00    0.00    0.00    1.45
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -1.49e+02   5.18e+01   -2.87   0.0041
## statusesCount                 8.40e+00   3.70e+00    2.27   0.0230
## friendsCount                 -5.47e-01   5.87e-01   -0.93   0.3508
## followersCount                3.41e+01   1.64e+01    2.08   0.0378
## listedCount                  -3.61e+01   2.19e+01   -1.65   0.0990
## acct_age                      3.64e-03   1.07e+00    0.00   0.9973
## langDiv                      -1.13e+02   3.92e+01   -2.89   0.0038
## mean_time_betwn_tweets        2.91e+00   5.76e+00    0.51   0.6133
## mCount                        4.45e+00   3.73e+00    1.19   0.2323
## `tweet_volumeMedium Tweeter`  2.86e+00   2.24e+00    1.27   0.2026
## `tweet_volumeHeavy Tweeter`   9.31e-01   8.28e-01    1.12   0.2607
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1382.827  on 1811  degrees of freedom
## Residual deviance:   16.257  on 1801  degrees of freedom
## AIC: 38.26
##
## Number of Fisher Scoring iterations: 20
```

```r
plot(varImp(logistic_model))
```




```r
# test predictions
logistic_predictions = predict(logistic_model, newdata = test)
confusionMatrix(logistic_predictions, test$bot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 523   0
##          1   3  76
##
##                Accuracy : 0.995
##                  95% CI : (0.986, 0.999)
##     No Information Rate : 0.874
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.978
##  Mcnemar's Test P-Value : 0.248
##
##             Sensitivity : 0.994
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 0.962
##              Prevalence : 0.874
##          Detection Rate : 0.869
##    Detection Prevalence : 0.869
##       Balanced Accuracy : 0.997
```

```
##
##        'Positive' Class : 0
##
```

There are subset selection methods for logistic regression as well. Try out `method = 'glmStepAIC'`:

```r
# stepwise logisitic regression
step_model = train(bot ~ .,
                   data = train,
                   method = 'glmStepAIC',
                   family = binomial,
                   preProcess = c('center', 'scale'))
```

```r
summary(step_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
##  -2.16    0.00    0.00    0.00    1.44
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -149.93      46.73   -3.21   0.0013
## statusesCount                      8.55       3.08    2.77   0.0056
## followersCount                    17.12      13.25    1.29   0.1962
## listedCount                      -37.52      16.91   -2.22   0.0265
## langDiv                         -113.12      34.95   -3.24   0.0012
## mCount                             4.54       3.49    1.30   0.1925
## `tweet_volumeMedium Tweeter`       2.19       1.98    1.10   0.2696
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1382.827  on 1811  degrees of freedom
## Residual deviance:   18.014  on 1805  degrees of freedom
## AIC: 32.01
##
## Number of Fisher Scoring iterations: 16
```

```r
step_predictions = predict(step_model, newdata = test)
confusionMatrix(step_predictions, test$bot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 523   0
##          1   3  76
##
##                Accuracy : 0.995
##                  95% CI : (0.986, 0.999)
##     No Information Rate : 0.874
##     P-Value [Acc > NIR] : <2e-16
##
```

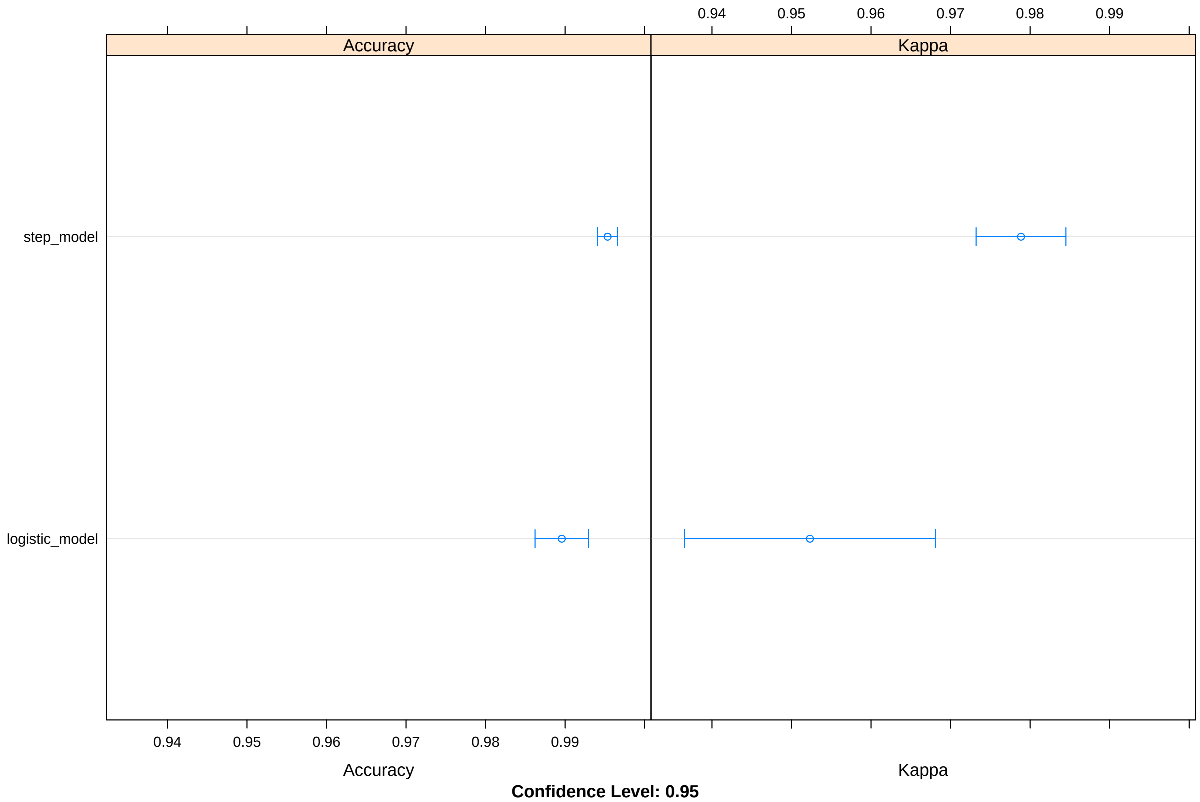
```
##                   Kappa : 0.978
##  Mcnemar's Test P-Value : 0.248
##
##             Sensitivity : 0.994
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 0.962
##              Prevalence : 0.874
##          Detection Rate : 0.869
##    Detection Prevalence : 0.869
##       Balanced Accuracy : 0.997
##
##        'Positive' Class : 0
##
```

How do the models compare?

```
# compare
results = resamples(list(logistic_model = logistic_model,
                         step_model = step_model))

# compare accuracy and kappa
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: logistic_model, step_model
## Number of resamples: 25
##
## Accuracy
##                  Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## logistic_model 0.9719  0.9869 0.9911 0.9896  0.9955 0.9985    0
## step_model     0.9881  0.9940 0.9956 0.9953  0.9970 1.0000    0
##
## Kappa
##                  Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## logistic_model 0.8631  0.9458 0.9604 0.9523  0.9802 0.9934    0
## step_model     0.9496  0.9719 0.9803 0.9789  0.9872 1.0000    0
```

```
# plot results
dotplot(results)
```


Accuracy
Kappa
step_model
logistic_model
0.94
0.95
0.96
0.97
0.98
0.99
Confidence Level: 0.95