Logistic regression in R

May 09, 2016

Contents

Introduction

For the second project we'll explore user data from Twitter to identify accounts likely belonging to bots. The data set has variables about profile configuration (`default_profile`, `default_profile_image`), connectivity (`friends_count`, `followers_count`), 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`).

```
library(dplyr)
library(ggplot2)
library(GGally)
library(caret)

twitter = read.delim('bot_or_not.tsv',
                     sep = '\t',
                     header = TRUE)
```

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.

```
summary(twitter)
```

```
##       bot          statuses_count   default_profile  default_profile_image
##  Min.   :0.0000   Min.   :     0   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:   188   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.0000   Median :   723   Median :0.0000   Median :0.00000
##  Mean   :0.1587   Mean   :  3277   Mean   :0.2897   Mean   :0.03117
##  3rd Qu.:0.0000   3rd Qu.:  2646   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :137264   Max.   :1.0000   Max.   :1.00000
```

```
##  friends_count     followers_count     favourites_count  geo_enabled
##  Min.   :     11   Min.   :      0.0   Min.   :     0   Min.   :0.0000
##  1st Qu.:    300   1st Qu.:     95.0   1st Qu.:    14   1st Qu.:0.0000
##  Median :    615   Median :    288.0   Median :   122   Median :0.0000
##  Mean   :   2358   Mean   :   3709.3   Mean   :  1100   Mean   :0.4418
##  3rd Qu.:   1229   3rd Qu.:    830.5   3rd Qu.:   593   3rd Qu.:1.0000
##  Max.   :1175187   Max.   :1396699.0   Max.   :176219   Max.   :1.0000
##   listed_count     account_age_hours   diversity
##  Min.   :   0.00   Min.   : 2072     Min.   :0.0050
##  1st Qu.:   4.00   1st Qu.:30285     1st Qu.:0.6254
##  Median :  16.00   Median :47484     Median :0.6963
##  Mean   :  84.77   Mean   :43664     Mean   :0.6791
##  3rd Qu.:  51.00   3rd Qu.:56718     3rd Qu.:0.7626
##  Max.   :9491.00   Max.   :78841     Max.   :1.0000
##  mean_mins_between_tweets mean_tweet_length mean_retweets
##  Min.   :    -15.7        Min.   :  8.50    Min.   :   1.000
##  1st Qu.:   1152.8        1st Qu.: 80.79    1st Qu.:   1.167
##  Median :   3851.7        Median : 91.74    Median :   1.636
##  Mean   :  14715.4        Mean   : 91.41    Mean   :   3.873
##  3rd Qu.:  10823.8        3rd Qu.:103.28    3rd Qu.:   2.424
##  Max.   :1139015.0        Max.   :287.88    Max.   :1961.300
##    reply_rate
##  Min.   :0.0000
##  1st Qu.:0.1232
##  Median :0.3137
##  Mean   :0.3411
##  3rd Qu.:0.5279
##  Max.   :1.0000
```

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.

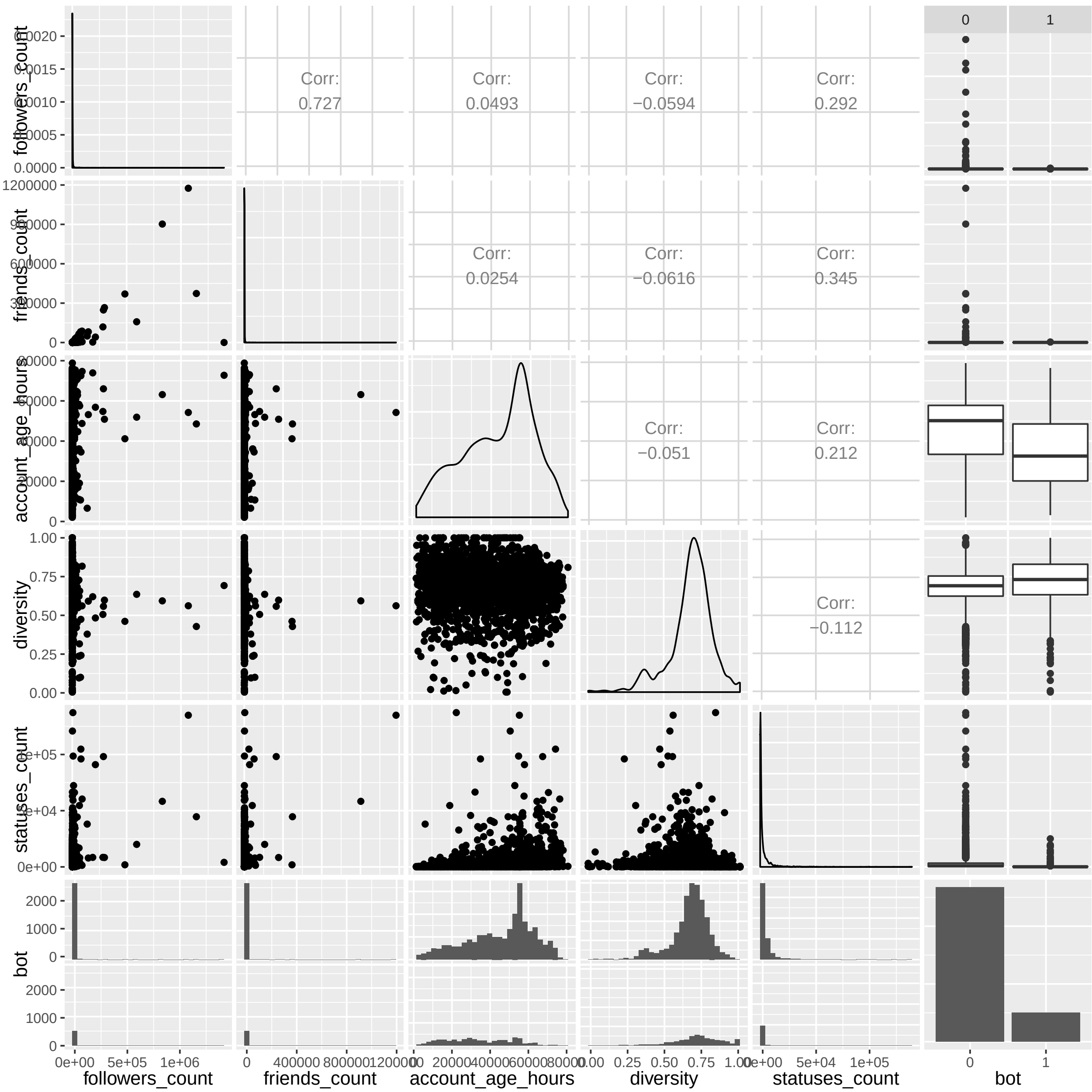
```r
twitter$bot = factor(twitter$bot)
twitter$default_profile = factor(twitter$default_profile)
twitter$default_profile_image = factor(twitter$default_profile_image)
twitter$geo_enabled = factor(twitter$geo_enabled)

summary(twitter)
```

```
##  bot      statuses_count   default_profile default_profile_image
##  0:2672   Min.   :     0   0:2256          0:3077
##  1: 504   1st Qu.:   188   1: 920          1:  99
##           Median :   723
##           Mean   :  3277
##           3rd Qu.:  2646
##           Max.   :137264
##  friends_count     followers_count     favourites_count geo_enabled
##  Min.   :     11   Min.   :      0.0   Min.   :     0   0:1773
##  1st Qu.:    300   1st Qu.:     95.0   1st Qu.:    14   1:1403
##  Median :    615   Median :    288.0   Median :   122
##  Mean   :   2358   Mean   :   3709.3   Mean   :  1100
##  3rd Qu.:   1229   3rd Qu.:    830.5   3rd Qu.:   593
```

```
##  Max.   :1175187   Max.   :1396699.0   Max.   :176219
##   listed_count     account_age_hours   diversity
##  Min.   :   0.00   Min.   : 2072     Min.   :0.0050
##  1st Qu.:   4.00   1st Qu.:30285     1st Qu.:0.6254
##  Median :  16.00   Median :47484     Median :0.6963
##  Mean   :  84.77   Mean   :43664     Mean   :0.6791
##  3rd Qu.:  51.00   3rd Qu.:56718     3rd Qu.:0.7626
##  Max.   :9491.00   Max.   :78841     Max.   :1.0000
##  mean_mins_between_tweets mean_tweet_length mean_retweets
##  Min.   :    -15.7        Min.   :  8.50    Min.   :   1.000
##  1st Qu.:   1152.8        1st Qu.: 80.79    1st Qu.:   1.167
##  Median :   3851.7        Median : 91.74    Median :   1.636
##  Mean   :  14715.4        Mean   : 91.41    Mean   :   3.873
##  3rd Qu.:  10823.8        3rd Qu.:103.28    3rd Qu.:   2.424
##  Max.   :1139015.0        Max.   :287.88    Max.   :1961.300
##    reply_rate
##  Min.   :0.0000
##  1st Qu.:0.1232
##  Median :0.3137
##  Mean   :0.3411
##  3rd Qu.:0.5279
##  Max.   :1.0000
```

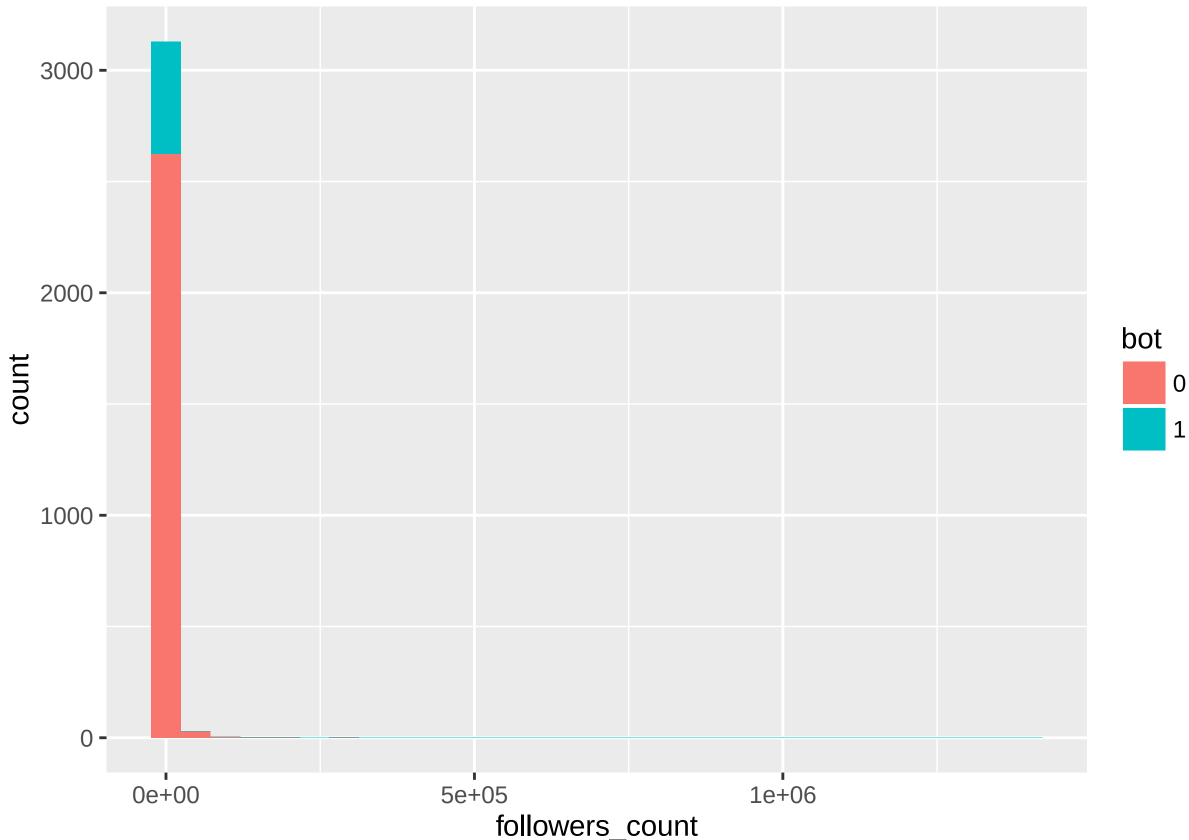
Like before, we can evaluate many relationships simultaneously with `ggpairs`.

```r
# inspect many trends with ggpairs
ggpairs(twitter[ , c('followers_count', 'friends_count', 'account_age_hours',
                  'diversity', 'statuses_count', 'bot')])
```

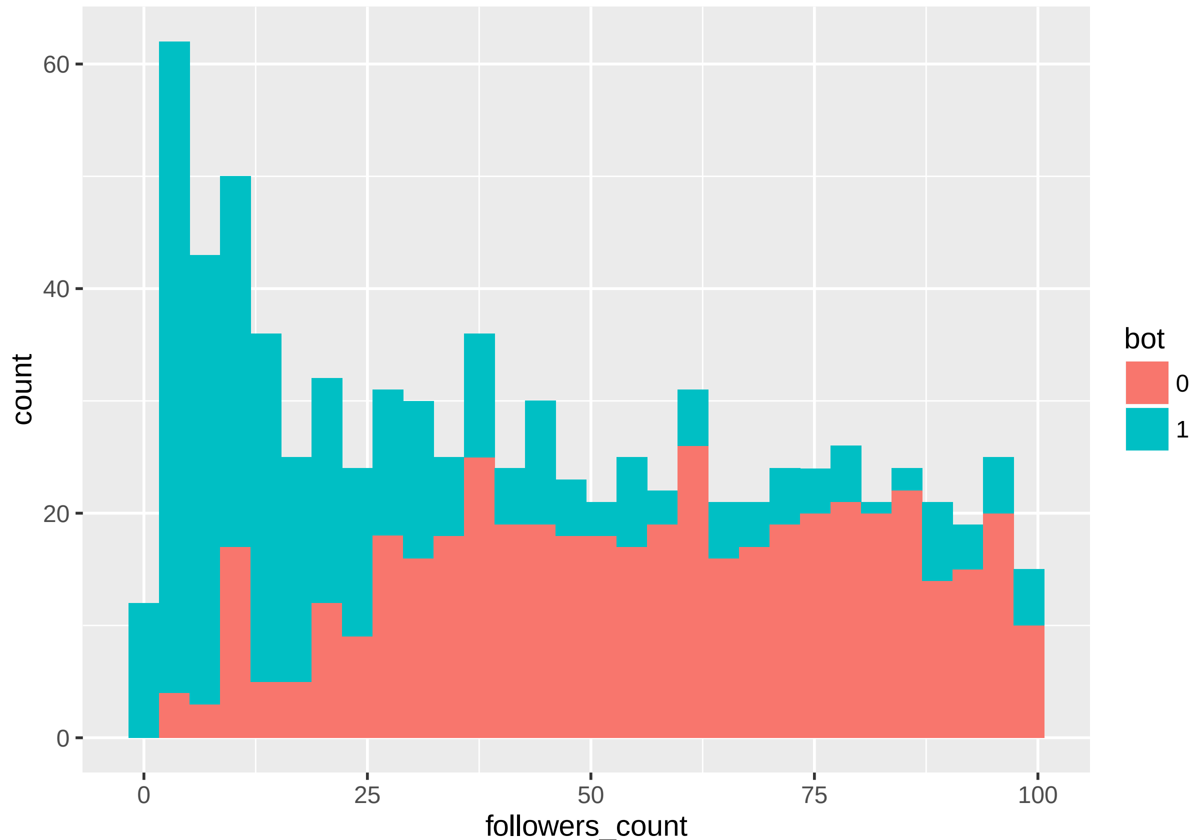


Once we have some initial hypotheses we can make more specific plots.

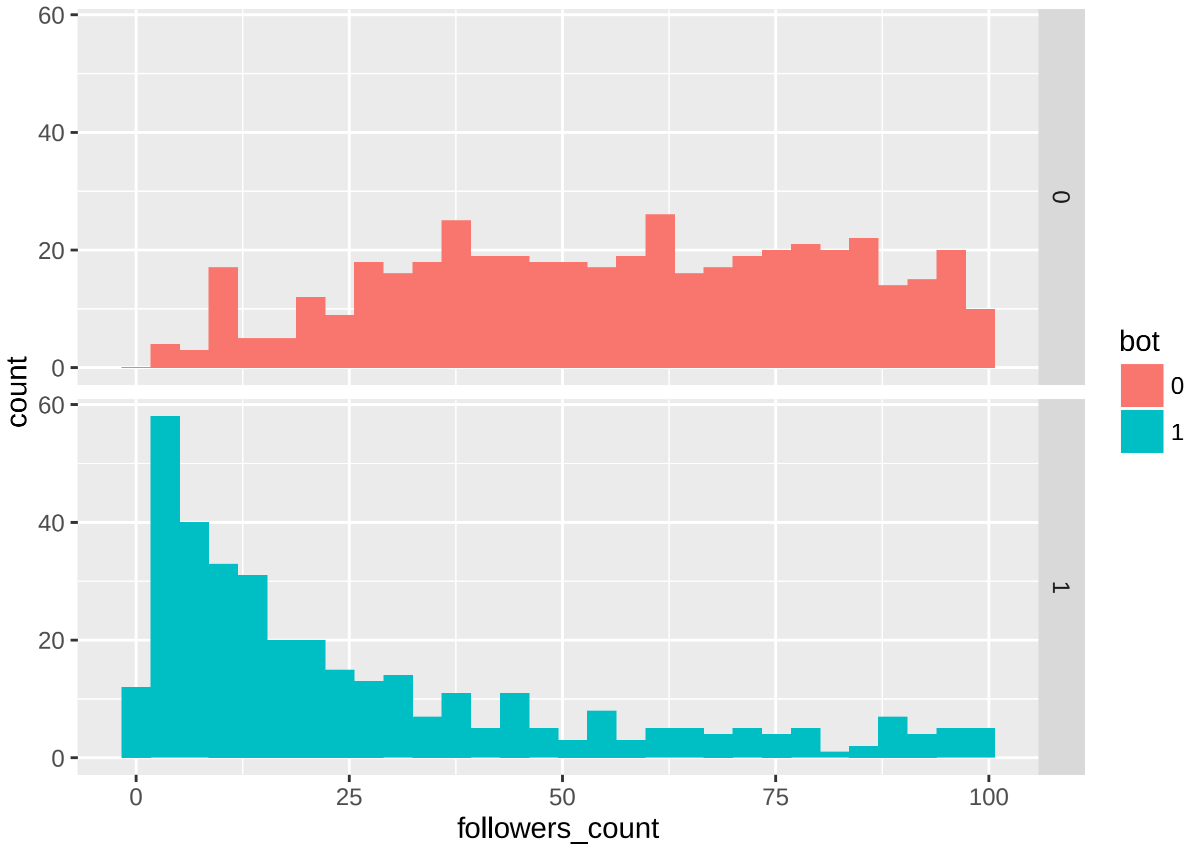
```
ggplot(twitter, aes(x = followers_count, fill = bot)) +
  geom_histogram()
```



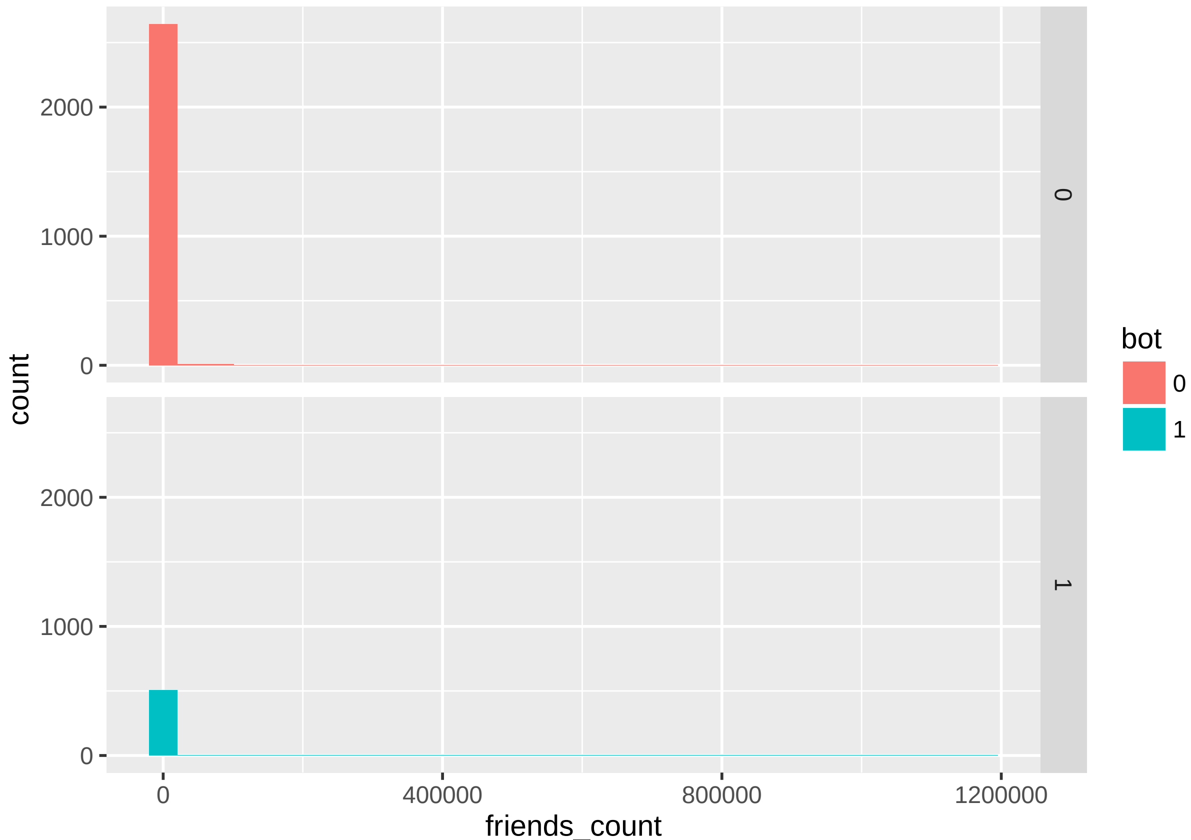

```r
# Some people have a lot of followers, but most don't. we need to lob off
# the long tail so we can see the distribution better
ggplot(filter(twitter, followers_count < 100),
       aes(x = followers_count, fill = bot)) +
  geom_histogram()
```



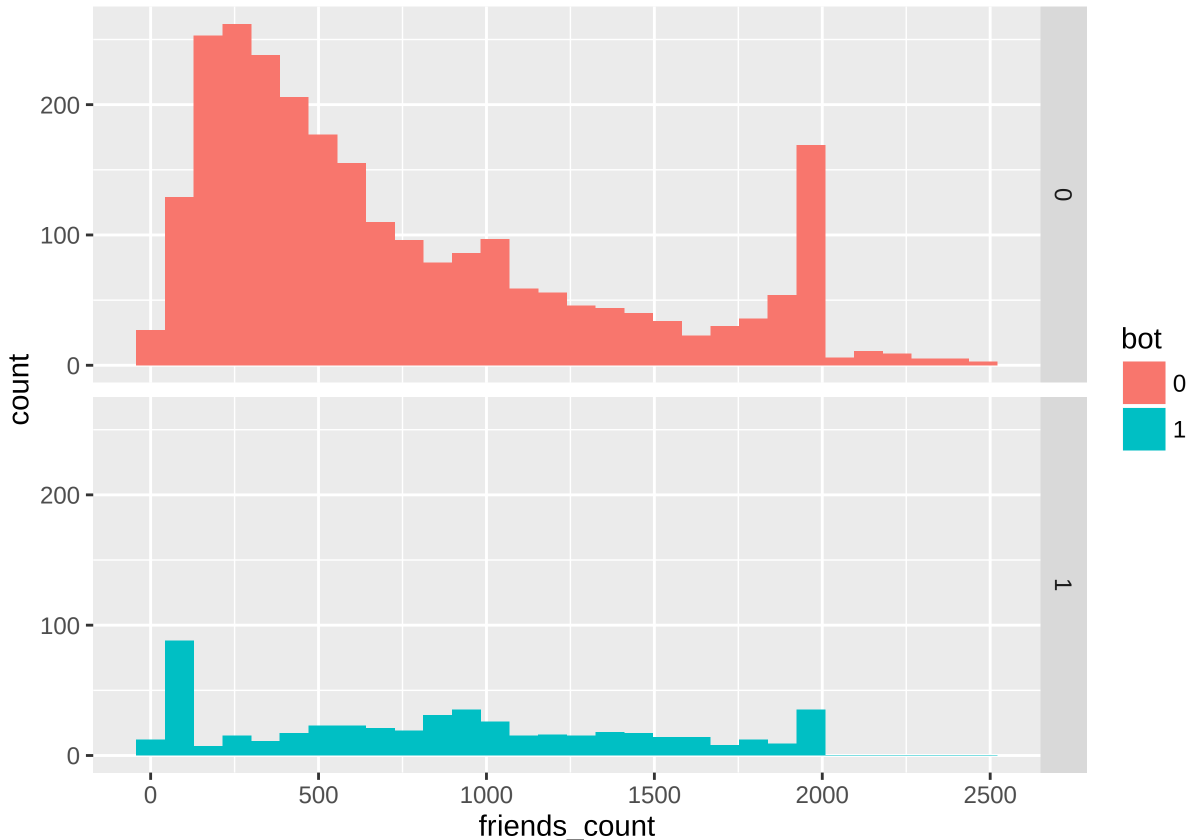

```
ggplot(filter(twitter, followers_count < 100),
       aes(x = followers_count, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



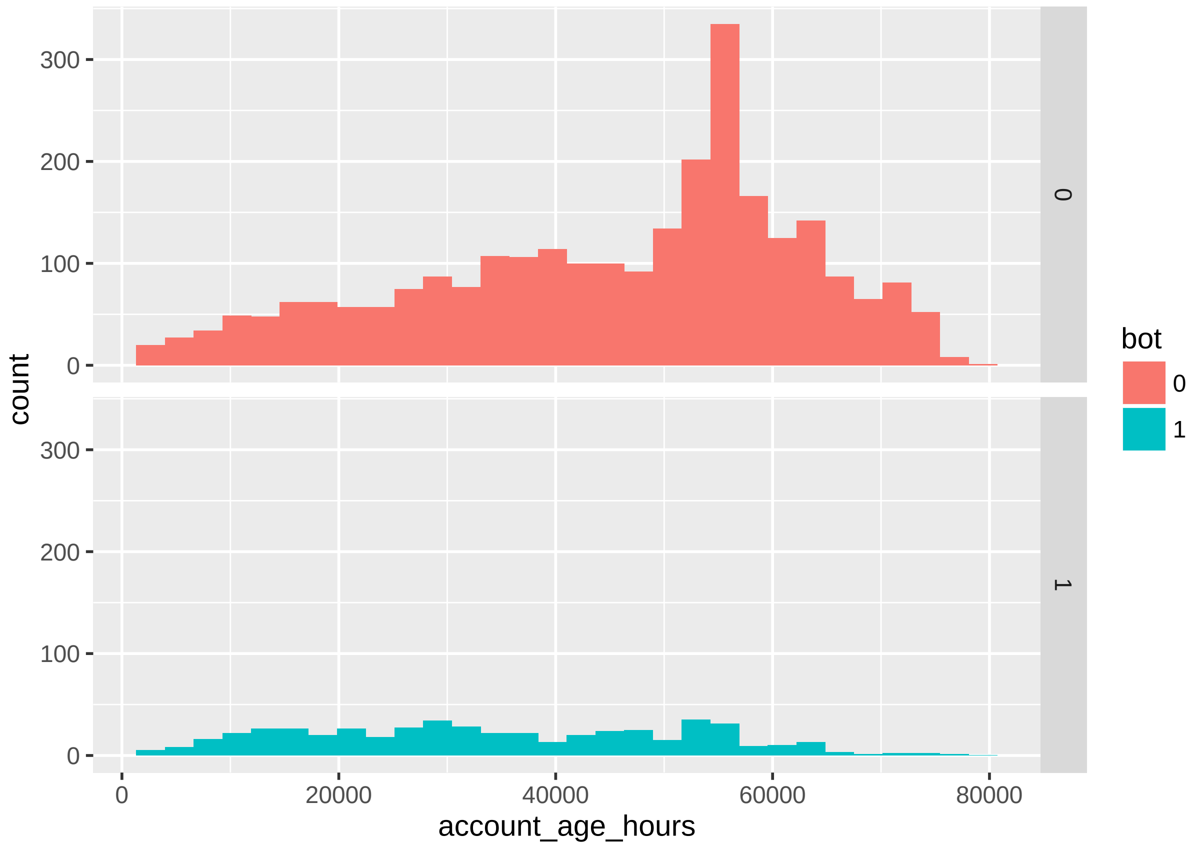

```r
# how about the number of people they follow?
ggplot(twitter, aes(x = friends_count, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



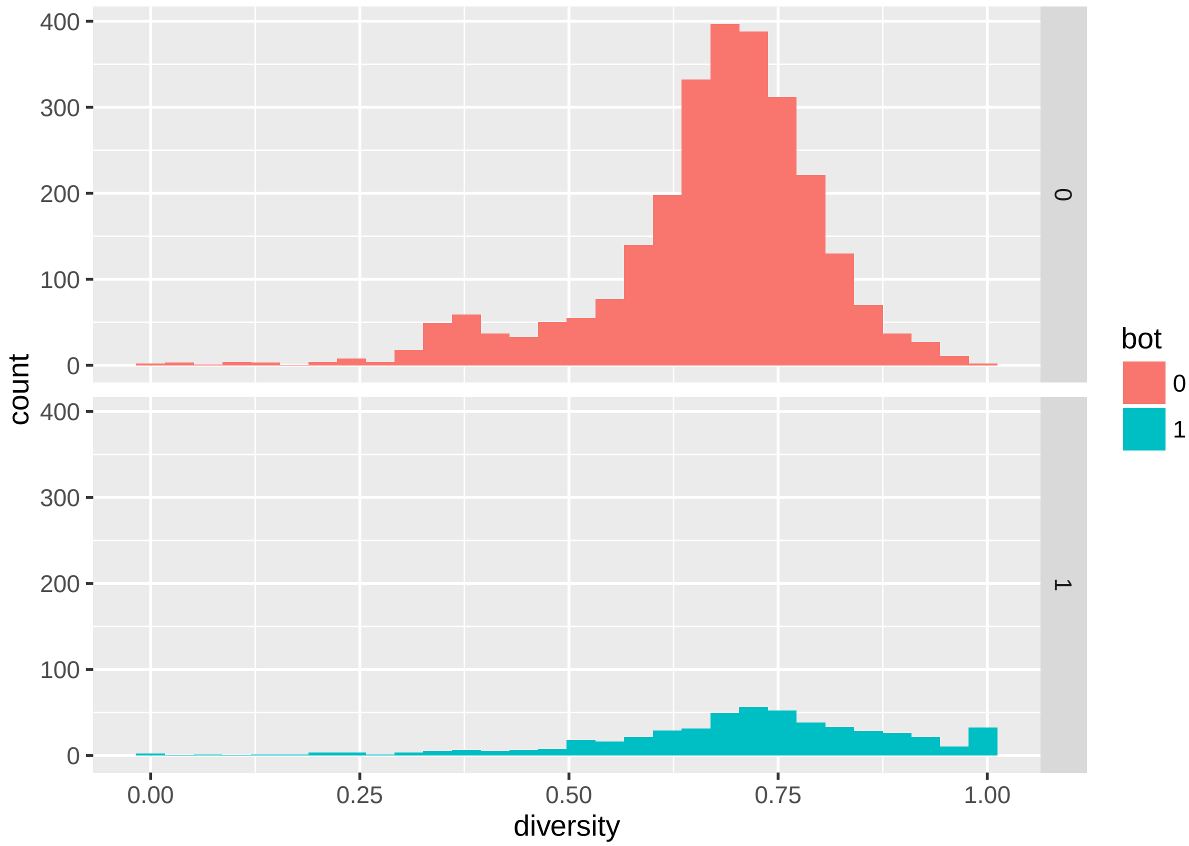

```r
# it's a little hard to see
ggplot(filter(twitter, friends_count < 2500),
       aes(x = friends_count, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```



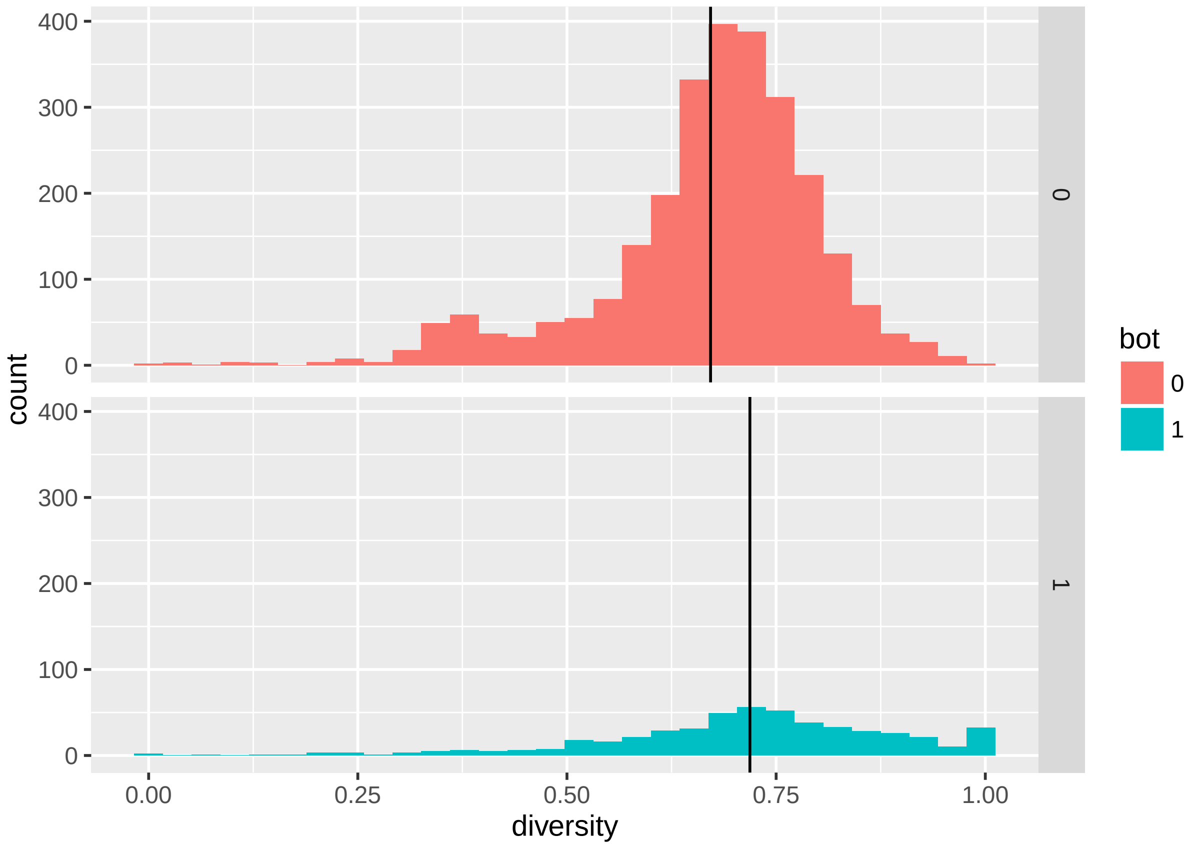

```r
# what about account age?
ggplot(twitter, aes(x = account_age_hours, fill = bot)) +
  geom_histogram() +
  facet_grid(bot ~.)
```




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



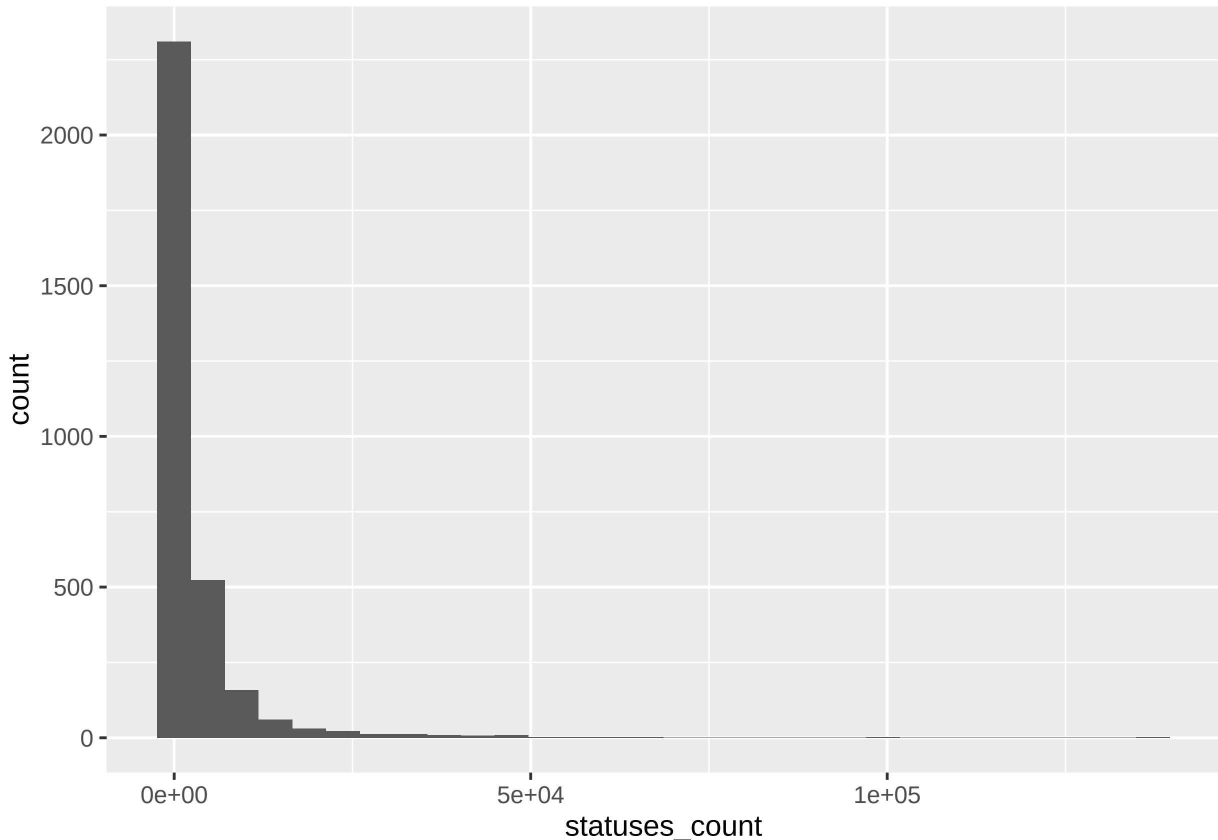

```r
# what are the average values?
avg_diversity =
  twitter %>%
    group_by(bot) %>%
    summarize(avg_diversity = mean(diversity, na.rm = TRUE))

# add it to the plot
ggplot(twitter, aes(x = diversity, 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':

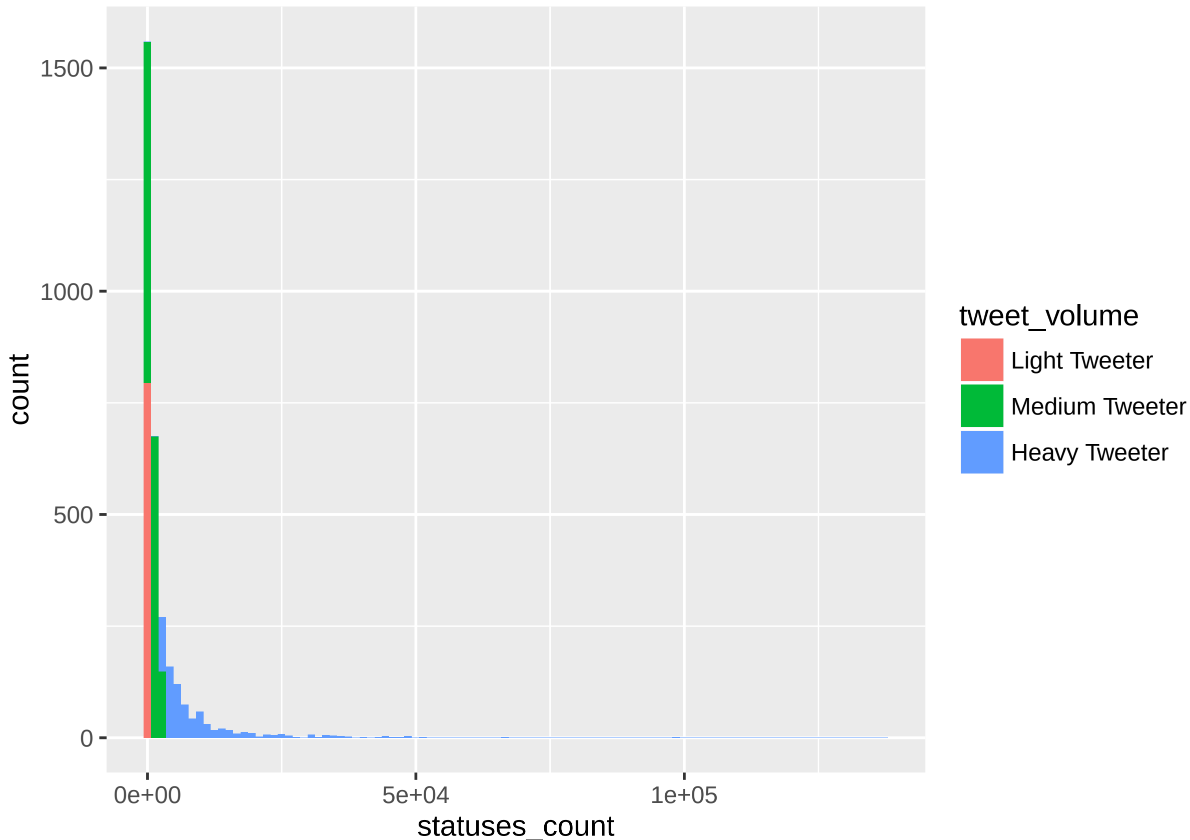
```
# the number of tweets per account has a long tail
ggplot(twitter, aes(x = statuses_count)) +
  geom_histogram()
```



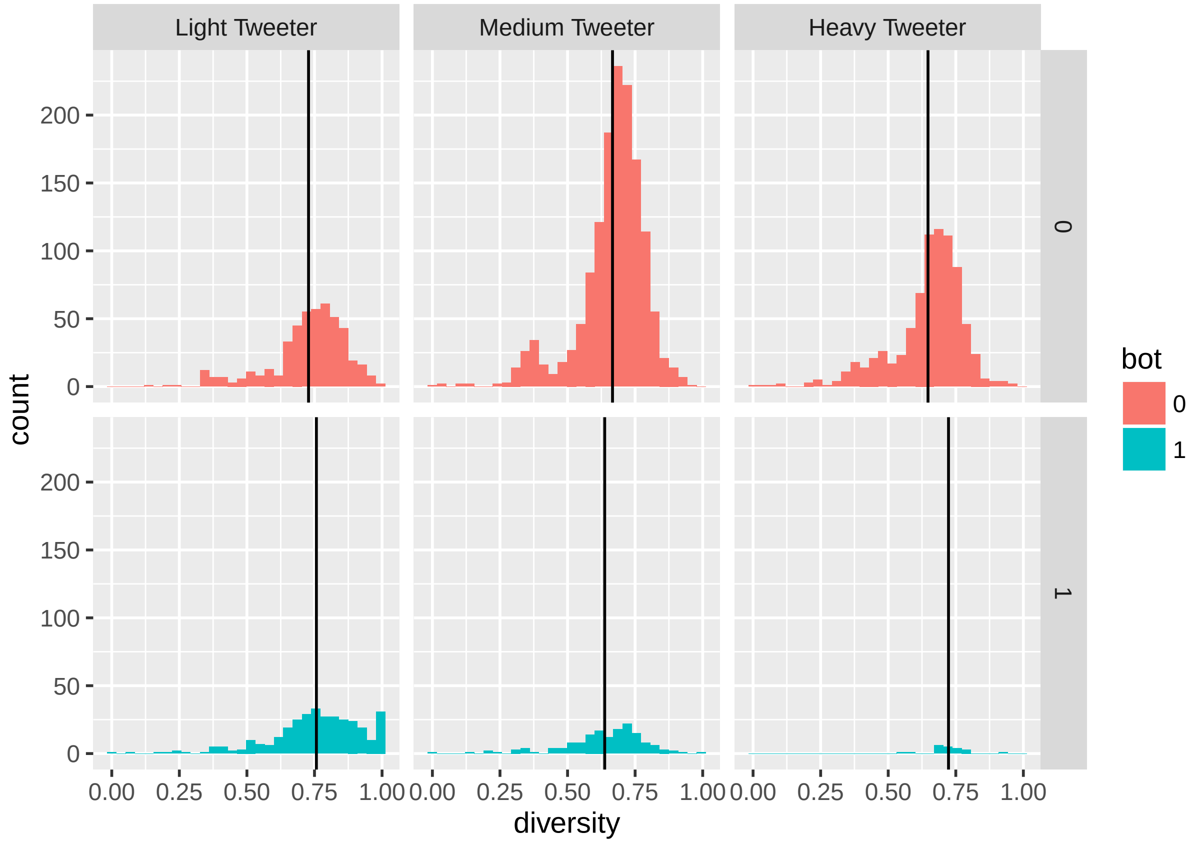

```r
# break into three categories by quantile
quantile(twitter$statuses_count)
```

```
##       0%      25%      50%      75%     100%
##      0.0    188.0    723.0   2646.5 137264.0
```

```r
# low tweeters will be the bottom 25%,
twitter$tweet_volume = NA
twitter$tweet_volume = ifelse(twitter$statuses_count <= 188,
                              'Light Tweeter',
                              twitter$tweet_volume)

twitter$tweet_volume = ifelse((twitter$statuses_count > 188 & twitter$statuses_count <= 2646),
                              'Medium Tweeter',
                              twitter$tweet_volume)

twitter$tweet_volume = ifelse(twitter$statuses_count > 2646,
                              'Heavy Tweeter',
                              twitter$tweet_volume)

twitter$tweet_volume = factor(twitter$tweet_volume, levels = c('Light Tweeter', 'Medium Tweeter', 'Heavy

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




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

ggplot(twitter, aes(x = diversity, 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)
twitter = na.omit(twitter)

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

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

Training logisitic regressions

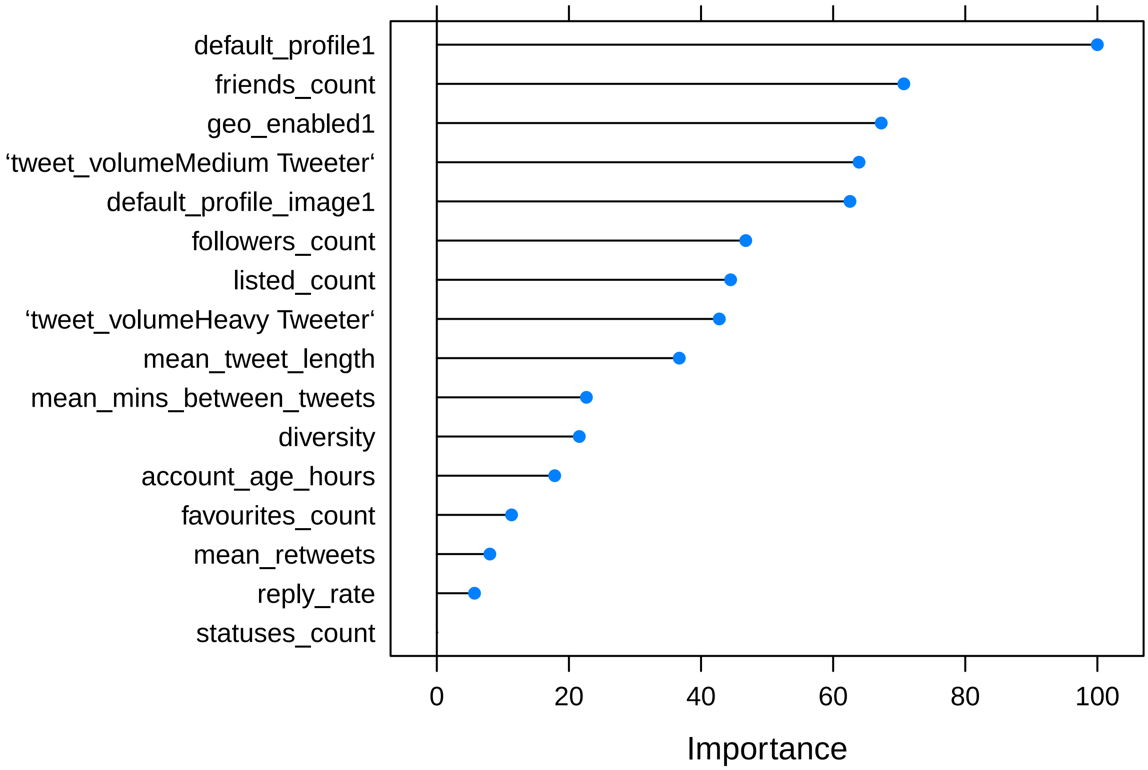
Check out this page for more types of logistic regression to try out.

```
logistic_model = train(bot ~ .,
                       data = train,
                       method = 'glm',
```

```r
                        family = binomial,
                        preProcess = c('center', 'scale'))

summary(logistic_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8057  -0.4543  -0.2644  -0.1400   4.2037
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -3.66773    0.51971  -7.057 1.70e-12 ***
## statuses_count                -0.05628    0.29941  -0.188 0.850892
## default_profile1               0.58481    0.06572   8.898  < 2e-16 ***
## default_profile_image1         0.52817    0.09371   5.636 1.74e-08 ***
## friends_count                 23.55816    3.71218   6.346 2.21e-10 ***
## followers_count              -40.02490    9.39093  -4.262 2.03e-05 ***
## favourites_count              -0.45419    0.38694  -1.174 0.240474
## geo_enabled1                  -0.51964    0.08592  -6.048 1.47e-09 ***
## listed_count                   3.25166    0.80043   4.062 4.86e-05 ***
## account_age_hours             -0.13779    0.07907  -1.743 0.081399 .
## diversity                      0.14801    0.07161   2.067 0.038742 *
## mean_mins_between_tweets       0.18157    0.08404   2.161 0.030729 *
## mean_tweet_length             -0.23152    0.06839  -3.385 0.000711 ***
## mean_retweets                 -1.07502    1.21159  -0.887 0.374925
## reply_rate                    -0.05054    0.07376  -0.685 0.493177
## `tweet_volumeMedium Tweeter`  -0.47440    0.08243  -5.756 8.64e-09 ***
## `tweet_volumeHeavy Tweeter`   -0.66668    0.17038  -3.913 9.12e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2084.2  on 2381  degrees of freedom
## Residual deviance: 1358.7  on 2365  degrees of freedom
## AIC: 1392.7
##
## Number of Fisher Scoring iterations: 15
```

```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
##          0 660  81
##          1   8  45
##
##                Accuracy : 0.8879
##                  95% CI : (0.8639, 0.909)
##     No Information Rate : 0.8413
##     P-Value [Acc > NIR] : 0.0001104
##
##                   Kappa : 0.4512
##  Mcnemar's Test P-Value : 2.312e-14
##
##             Sensitivity : 0.9880
##             Specificity : 0.3571
##          Pos Pred Value : 0.8907
##          Neg Pred Value : 0.8491
##              Prevalence : 0.8413
##          Detection Rate : 0.8312
##    Detection Prevalence : 0.9332
##       Balanced Accuracy : 0.6726
##
```

```
##        '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
## -2.8225  -0.4520  -0.2657  -0.1406   4.1964
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -3.67541    0.51856  -7.088 1.36e-12 ***
## default_profile1              0.58261    0.06562   8.879  < 2e-16 ***
## default_profile_image1        0.52775    0.09347   5.646 1.64e-08 ***
## friends_count                23.78836    3.70813   6.415 1.41e-10 ***
## followers_count             -40.30223    9.35289  -4.309 1.64e-05 ***
## favourites_count             -0.50388    0.37668  -1.338 0.180997
## geo_enabled1                 -0.51986    0.08588  -6.053 1.42e-09 ***
## listed_count                  3.25599    0.80530   4.043 5.27e-05 ***
## account_age_hours            -0.14007    0.07901  -1.773 0.076256 .
## diversity                     0.14543    0.07162   2.031 0.042300 *
## mean_mins_between_tweets      0.18105    0.08431   2.147 0.031762 *
## mean_tweet_length            -0.22615    0.06810  -3.321 0.000897 ***
## mean_retweets                -1.14154    1.21830  -0.937 0.348761
## `tweet_volumeMedium Tweeter`  -0.48019    0.08206  -5.851 4.87e-09 ***
## `tweet_volumeHeavy Tweeter`   -0.69051    0.15135  -4.562 5.06e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2084.2  on 2381  degrees of freedom
## Residual deviance: 1359.2  on 2367  degrees of freedom
## AIC: 1389.2
##
## Number of Fisher Scoring iterations: 15
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 660  79
##          1   8  47
##
##                Accuracy : 0.8904
##                  95% CI : (0.8666, 0.9113)
##     No Information Rate : 0.8413
##     P-Value [Acc > NIR] : 4.658e-05
##
##                   Kappa : 0.468
##  Mcnemar's Test P-Value : 6.153e-14
##
##             Sensitivity : 0.9880
##             Specificity : 0.3730
##          Pos Pred Value : 0.8931
##          Neg Pred Value : 0.8545
##              Prevalence : 0.8413
##          Detection Rate : 0.8312
##    Detection Prevalence : 0.9307
##       Balanced Accuracy : 0.6805
##
##        'Positive' Class : 0
##
```

How do the models compare?

```r
# 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 0.8002  0.8777 0.8832 0.8780  0.8897 0.8976    0
## step_model     0.8654  0.8753 0.8815 0.8842  0.8939 0.9036    0
##
## Kappa
##                  Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## logistic_model 0.2075  0.4444 0.4747 0.4513  0.4939 0.5407    0
## step_model     0.3869  0.4416 0.4706 0.4699  0.5001 0.5573    0
```

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