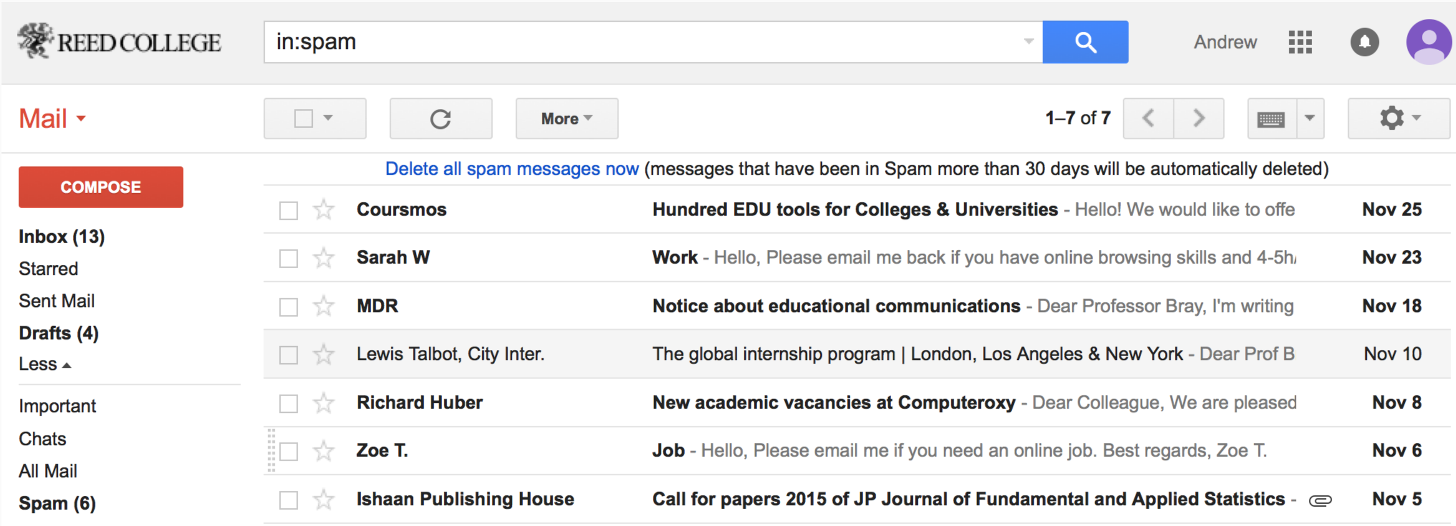
Logistic Regression

Building a spam filter

```
head(email)
```

```
##   spam to_multiple from cc sent_email                time image attach
## 1    0           0    1  0          0 2011-12-31 22:16:41     0      0
## 2    0           0    1  0          0 2011-12-31 23:03:59     0      0
## 3    0           0    1  0          0 2012-01-01 08:00:32     0      0
## 4    0           0    1  0          0 2012-01-01 01:09:49     0      0
## 5    0           0    1  0          0 2012-01-01 02:00:01     0      0
## 6    0           0    1  0          0 2012-01-01 02:04:46     0      0
##   dollar winner inherit viagra password num_char line_breaks format
## 1      0     no       0      0        0    11.37         202      1
## 2      0     no       0      0        0    10.50         202      1
## 3      4     no       1      0        0     7.77         192      1
## 4      0     no       0      0        0    13.26         255      1
## 5      0     no       0      0        2     1.23          29      0
## 6      0     no       0      0        2     1.09          25      0
##   re_subj exclaim_subj urgent_subj exclaim_mess number log_num_char
## 1       0            0           0            0    big       2.4310
## 2       0            0           0            1  small       2.3518
## 3       0            0           0            6  small       2.0507
## 4       0            0           0           48  small       2.5845
## 5       0            0           0            1   none       0.2078
## 6       0            0           0            1   none       0.0871
```

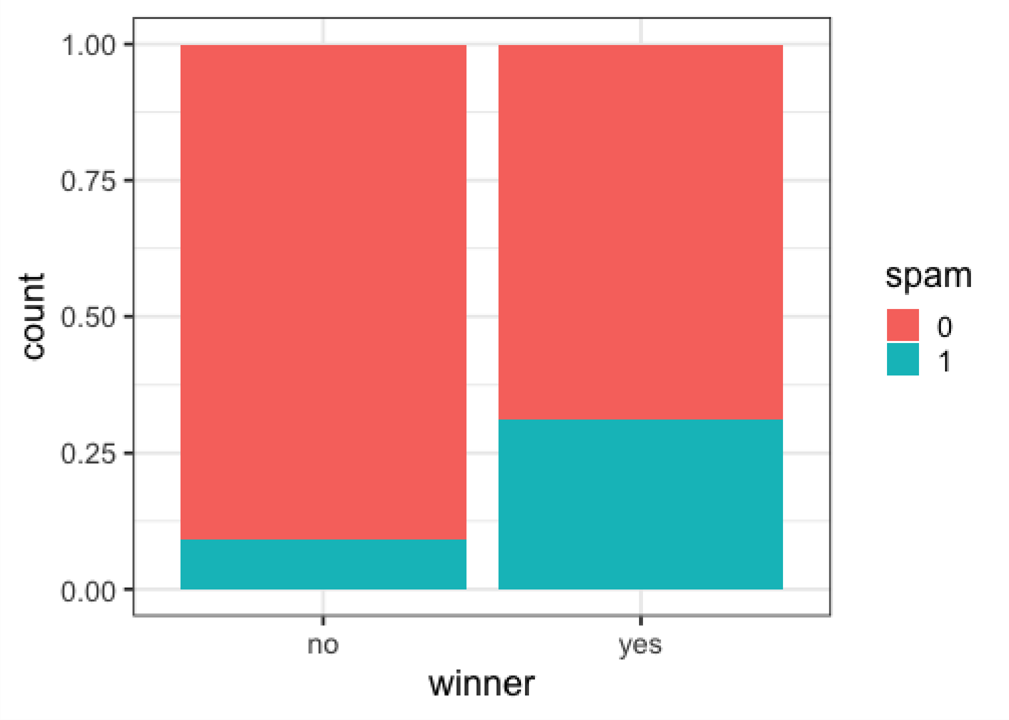
How was the data collected?



1. Choose a single email account
2. Save each email that comes in during a given time frame
3. Create dummy variables for each text component of interest
4. Visually classify each as spam or not

Simple Filter A

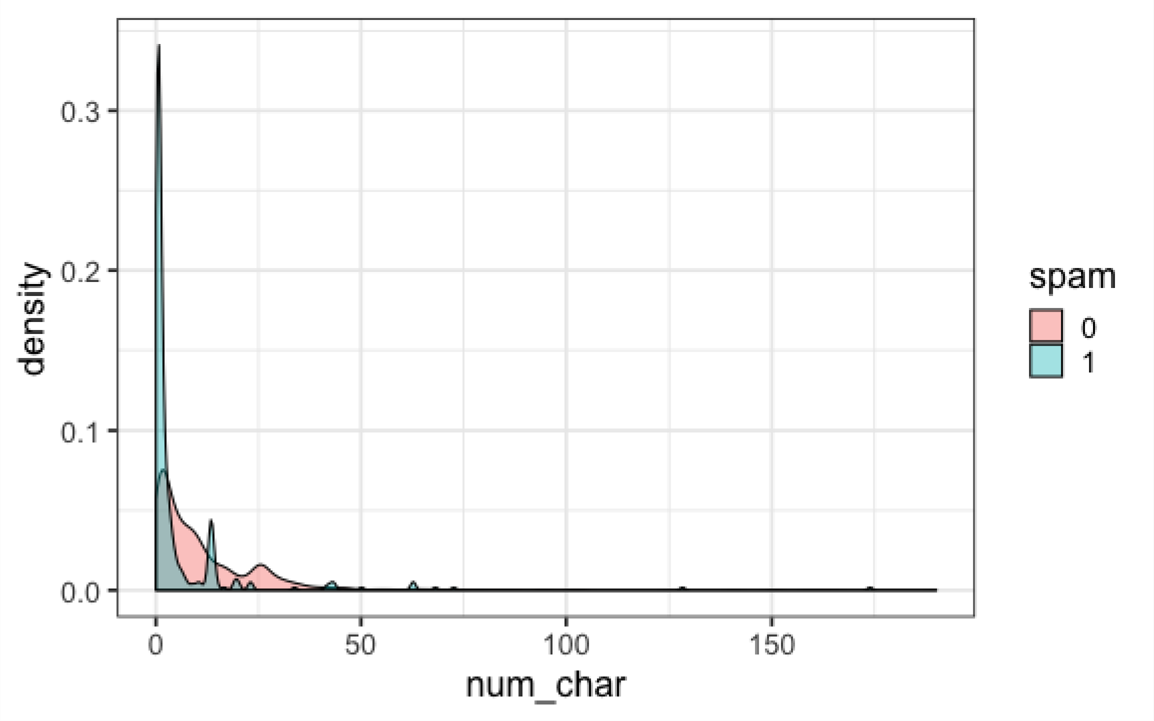
Predicting spam or not using the presence of "winner".



If "winner" then "spam"?

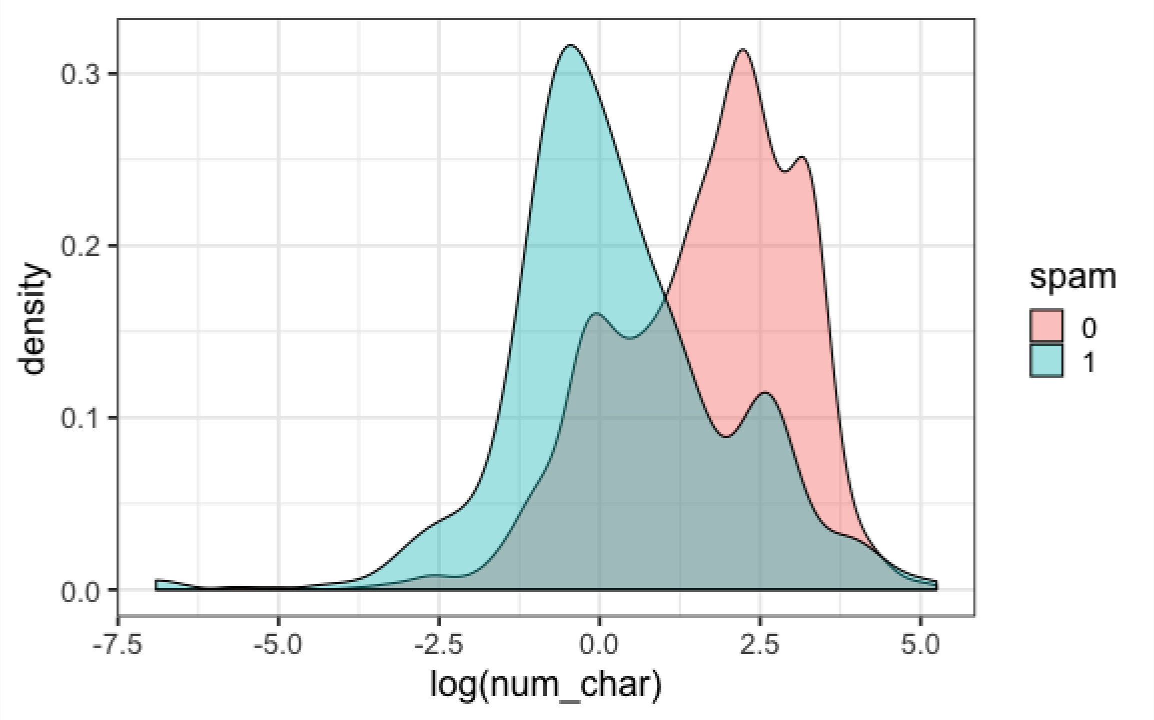
Simple Filter B

Predicting spam or not using number of characters (in K)



Simple Filter B, cont.

Predicting spam or not using log number of characters (in K)



If `log(num_char)` < 1, then "spam"?

Challenges

Each simple filter can be thought of as a regression model.

Filter A

$spam \sim winner; \quad X_1 \sim X_2$

Filter B

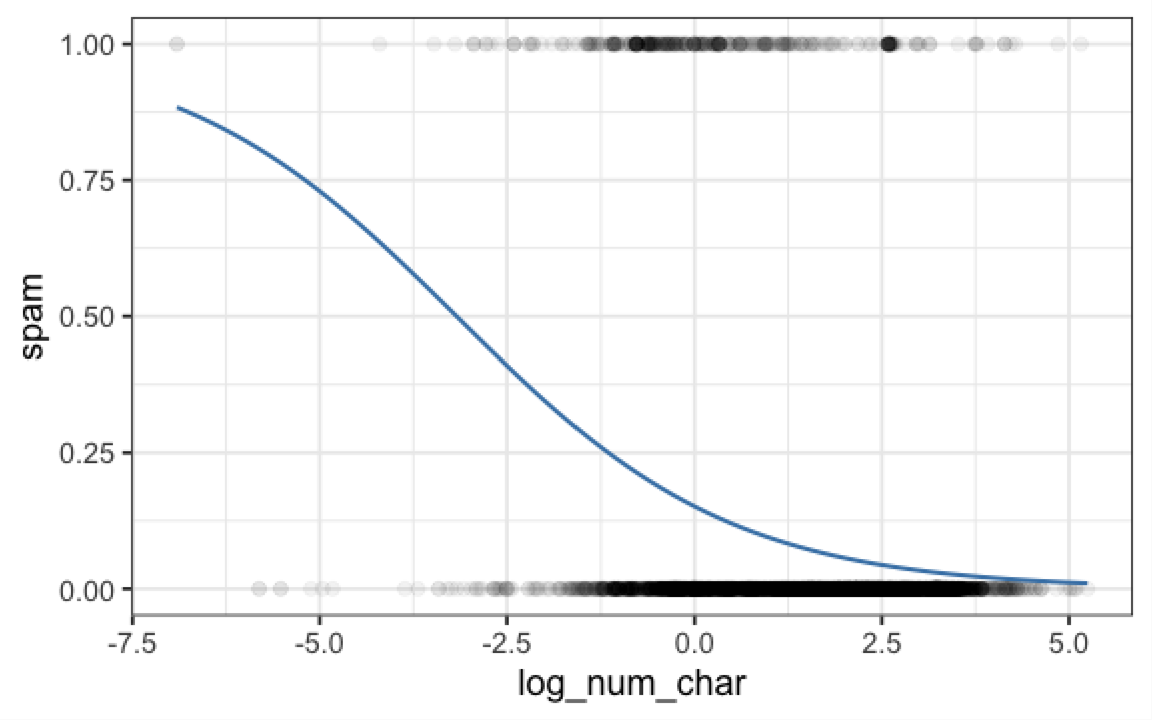
$spam \sim log(num\_char); \quad X_1 \sim W_1$

Each one by itself has poor predictive power, so how can we combine them into a single stronger model?

cartoons + boardwork

Logistic Regression for B

$$spam \sim log(num\_char)$$



Model fitting

```r
m1 <- glm(spam ~ log_num_char, data = email, family = "binomial")
summary(m1)
```

```
## 
## Call:
## glm(formula = spam ~ log_num_char, family = "binomial", data = email)
## 
## Deviance Residuals: 
##    Min       1Q   Median       3Q      Max  
## -1.815  -0.467  -0.335  -0.255   3.013  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept)   -1.7244     0.0606   -28.4   <2e-16 ***
## log_num_char  -0.5435     0.0365   -14.9   <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: 2437.2  on 3920  degrees of freedom
## Residual deviance: 2190.3  on 3919  degrees of freedom
## AIC: 2194
## 
## Number of Fisher Scoring iterations: 5
```

Interpreting Log. Reg.

1. Each row of the summary output is still a H-test on that parameter being 0.
2. A positive slope estimate indicates that there is a positive association.
3. Each estimate is still conditional on the other variables held constant.

A more sophisticated model

```r
m2 <- glm(spam ~ log_num_char + to_multiple +
            attach + dollar + inherit + viagra,
          data = email,
          family = "binomial")
summary(m2)
```

```
##
## Call:
## glm(formula = spam ~ log_num_char + to_multiple + attach + dollar +
##     inherit + viagra, family = "binomial", data = email)
##
## Deviance Residuals:
##    Min       1Q   Median       3Q      Max
## -1.931  -0.455  -0.328  -0.236   3.034
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.59642    0.06443  -24.78  < 2e-16 ***
## log_num_char -0.54869    0.03831  -14.32  < 2e-16 ***
## to_multiple -1.92889    0.30493   -6.33  2.5e-10 ***
## attach       0.19970    0.06552    3.05   0.0023 **
## dollar      -0.00456    0.01898   -0.24   0.8102
## inherit      0.40003    0.15166    2.64   0.0083 **
## viagra       1.73607   40.59296    0.04   0.9659
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comparing models: confusion matrix

```
make_conf_mat(m1, email)
```

```
## # A tibble: 4 x 3
## # Groups:   spam [2]
##     spam pred      n
##    <dbl> <lgl> <int>
## 1      0 FALSE  3541
## 2      0 TRUE     13
## 3      1 FALSE   362
## 4      1 TRUE      5
```

```
make_conf_mat(m2, email)
```

```
## # A tibble: 4 x 3
## # Groups:   spam [2]
##     spam pred      n
##    <dbl> <lgl> <int>
## 1      0 FALSE  3537
## 2      0 TRUE     17
## 3      1 FALSE   357
## 4      1 TRUE     10
```

Test-train

In the test-train paradigm, you balance descriptive power with predictive accuracy by separating your data set into:

1. **Training set**: used to fit your model
2. **Testing set**: used to evaluate predictive accuracy

Related to cross-validation...

Dividing the data

```
set.seed(501)
train_indices <- sample(1:nrow(email),
                        size = floor(nrow(email)/2))
head(train_indices)
```

```
## [1] 2008 3046 2886 3032 3236 3808
```

```
train_data <- email %>%
  slice(train_indices)
test_data <- email %>%
  slice(-train_indices)
```

```
dim(train_data)
```

```
## [1] 1960   22
```

```
dim(test_data)
```

```
## [1] 1961   22
```

Training

```r
m1 <- glm(spam ~ log_num_char,
          data = train_data,
          family = "binomial")
m2 <- glm(spam ~ log_num_char + to_multiple +
            attach + dollar + inherit + viagra,
          data = train_data,
          family = "binomial")
```

Testing

```
make_conf_mat(m1, test_data)
```

```
## # A tibble: 4 x 3
## # Groups:   spam [2]
##    spam pred      n
##   <dbl> <lgl> <int>
## 1     0 FALSE  1783
## 2     0 TRUE      4
## 3     1 FALSE   171
## 4     1 TRUE      3
```

```
make_conf_mat(m2, test_data)
```

```
## # A tibble: 4 x 3
## # Groups:   spam [2]
##    spam pred      n
##   <dbl> <lgl> <int>
## 1     0 FALSE  1782
## 2     0 TRUE      5
## 3     1 FALSE   171
## 4     1 TRUE      3
```

Extending the model

A GLM consists of three things:

1. A linear predictor
2. A distribution of the response
3. A link function between the two

MLR: Normal distribution, identity link function

Logisitic Regression: Binomial distribution, logit link function

Poisson Regression: Poisson distribution, logarithm