Assignment 3: Predictive Modeling in Binary Classification

Christina Macholan | Predict 454, Section 55

INTRODUCTION

In this exercise, I explored whether it is possible to reliably distinguish unsolicited bulk email (also known as spam) from the rest of an user's legitimate incoming emails. Building a model to filter spam can greatly benefit the recipient, who would otherwise waste time sorting and disposing of the messages manually. Such filters can also save money for Internet Service Providers (ISPs), whose systems can otherwise become bogged down with delivering large quantities of spam when they are not able to re-route our refuse the messages when they reach their servers.

While filtering out spam from legitimate email can be useful, false positives -- filtering out an email as spam when it is not spam -- are considered highly undesirable since potentially critical emails could go unseen and unread. As a result, the modeling problem for this exercise is unique in that the aim is to achieve the highest accuracy possible while minimizing the false positive rate, or “fall-out” rate, of the model.

DATA QUALITY CHECK

The data set I will use to build the spam filtration models for this exercise is provided by University of California Irvine at https://archive.ics.uci.edu/ml/datasets/Spambase.[1] The 4,600 emails recorded in the dataset came from various sources within the Hewlett-Packard (HP) internal email network. Spam emails were collected by the research team from various accounts where spam was flagged by the user or the postmaster. Non-spam emails were collected from work and personal emails from one of the data authors, George Forman.

Because the two classes of emails were collected via different approaches, some of the word frequency variables that are specific to Mr. Forman’s email (e.g. “george” and “650”) may need to be omitted to make the spam filtration model generalizable instead of personalized. Alternatively, these variables could be stand-ins for user-specific variables that would be dynamic when the model is deployed (e.g. “george” would be replaced by the user’s name in the spam filter model for that user).

Table 1 provides an overview of the 58 variables in the dataset. See Table A.1 in the appendix for a more detailed description of each variable.

TABLE 1: Description of variable types

	Variable Type	Count	Description
Predictor Variables	Word Frequency Variables	48	Continuous variables ranging from 0 to 100 that measure the frequency of specific words in the email body (e.g. 100 * count of WORD / total words)
	Character Frequency Variables	6	Continuous variables ranging from 0 to 100 that measure the frequency of specific characters in the email body (e.g. 100 * count of WORD / total words)
	Capital Letter Sequence Lengths	3	Continuous variables that record the average, longest, and total sequence of consecutive capital letters in the email body
Response Variable	Binary Spam Variable	1	A binary classification variable that distinguishes emails that are spam (1) from those that are not (0)

[1] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Response Variable

As shown in Figure 1, of the 4,600 emails in the data set, 1,812 (39%) are marked as spam, and 2,788 (61%) are legitimate emails. With this as our status quo, the baseline prediction accuracy that I want to surpass is 61%, which is the expected accuracy if I simply let all emails through to the user with no spam filter in place. This approach has the benefit of having a 0% fall-out rate, so no legitimate emails are masked to the user, but our aim is to do better on accuracy while maintaining a low fall-out rate.

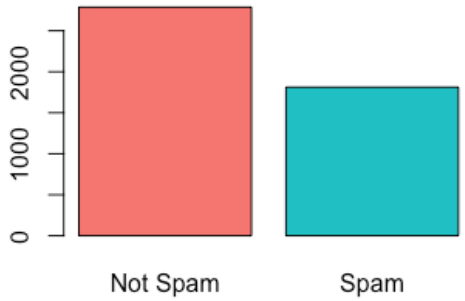


FIGURE 1: Frequency of spam emails in the University of California Irvine dataset

Predictor Variables

A review of the data set shows no missing values for the candidate predictor variables. There are, however, several notable traits about the predictors that should be accounted for during the modeling process.

First, for 51 out of the 54 frequency variables, over half of the values are zero. For example, the variable for the frequency of the word “credit” zero values for 91% of records. This zero-inflated distribution suggests that there is something distinctive about zero values. It may be appropriate to set up separate binary flag variables to distinguish the missing (0 value) frequencies from the other records.

Second, for most of the frequency variables, the distribution for the frequencies is highly skewed to the right, even when the zero-value records are removed. This is true of the capital letter sequence variables as well. For all of these variables, using a log10 transformation, as shown in the violin plot in Figure 2, helps highlight actual outliers and whether the distribution for the variable differs for spam emails and legitimate emails. In the case of the word “credit”, the log10 transformation shows that emails with the highest frequency of the word credit are more likely to be spam, and the ones with the lowest frequency are likely not to be spam. A complete set of violin plots for all predictor variables can be found in Figure A.1 in the Appendix.

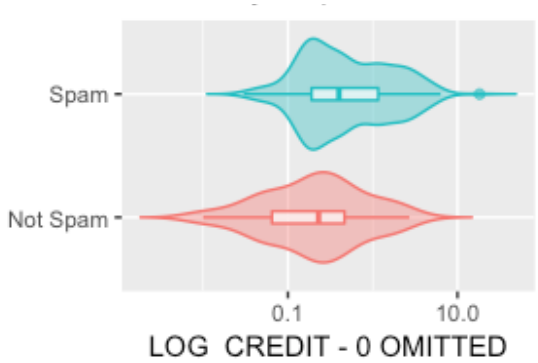


FIGURE 2: Violin plot of Log10 of the frequency of the word “credit”
Broken out by the response variable with any zero-value predictor variable records removed.

In order to capture the information in the extremes of each of the log10 transformed variables, I will create “outlier” flag variables to input into any models that I build for the top and bottom 1% of values.

EXPLORATORY DATA ANALYSIS

Figure A.1 in the Appendix includes a complete set of visual plots used to understand the bivariate relationships between the predictor variables and the response variable. Of these plots, the ones for the eight flag variables shown in Figure 3 (“remove”, “free”, “font”, “credit”, “000”, “money”, “$”, “!”) show that these words and characters appear much more frequently in spam emails.

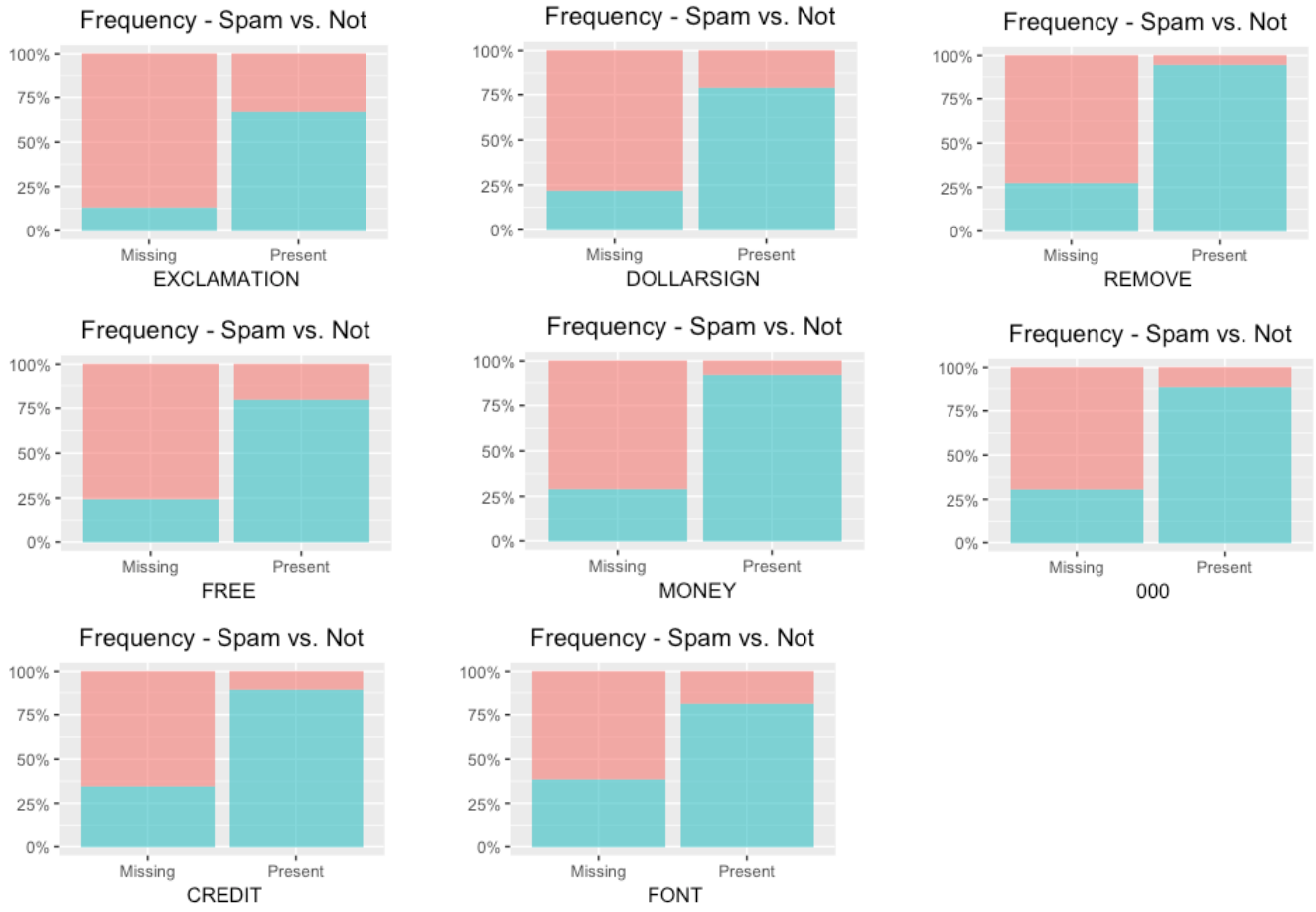


FIGURE 3: Highly differentiated variables for Spam vs. Not Spam
Consistent with the previous plots, the red indicates “not spam” and green indicates “spam”

On the other hand, the terms and characters are much more likely to not appear in the spam within this sample are: “415”, “lab”, “labs”, “cs”, “telnet”, “857”, “george”, “hp” and “hpl.” Of these nine variables, the first seven rarely show in the spam sample data. It is likely that these are all terms specific to Mr. Forman whose emails were sampled for the non-spam subset of data. Because of this issue, we should be wary of any models that rely too heavily on any of these variables for predicting out-of-sample accurately. Figure 4 shows three sample plots of these sparse variables.

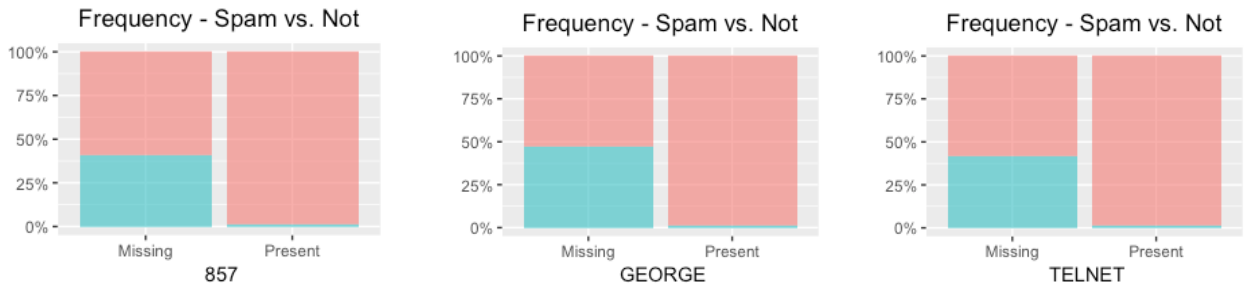


FIGURE 4: Variables with low representation in the "Spam" dataset

Of the three capital letter sequence variables -- the average of the capital letter sequences, the longest capital letter sequence, and the total letters in capitals -- each shows a different quality of separation between the response variable A plot of the log of each of these variables is shown in Figure 5. For the log average length, the spam category shows many more and more extreme outliers. For the log longest length, the maximum lengths are more highly separated, those the general distributions are similar. For the log total length, the mean and medians appear to be well separated, but with a closer overlap for the overall range of the data.

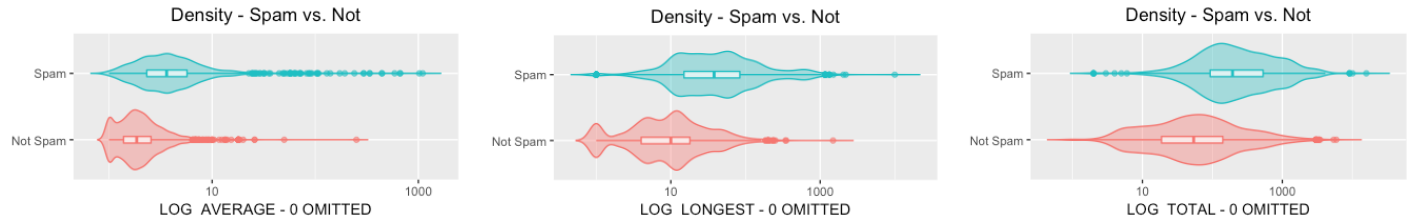


FIGURE 5: Violin plots of the capital letter sequence length variables (average, longest, and total)

As a final step of the exploratory data analysis, we move on to our first simple decision tree model using the `rpart` function in R. A plot of the relative variable importance in Figure 6 shows that the top nine variables are from the list of top variables that we identified in Figure 3. A few additional variables that we had not highlighted in the previous analysis are also included in their various flag, high/low outlier flag, or log10 forms, including "edu", "your", "business", and "make."

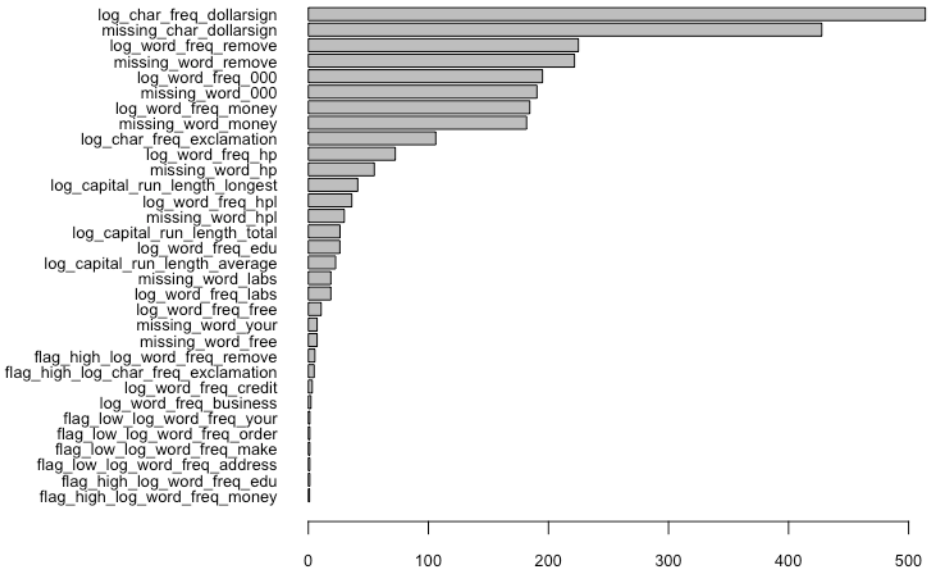


FIGURE 6: Relative variable importance for rpart classification tree

A plot of the first five levels of the decision tree in Figure 7 shows the relationship and decision rules for the proposed tree.

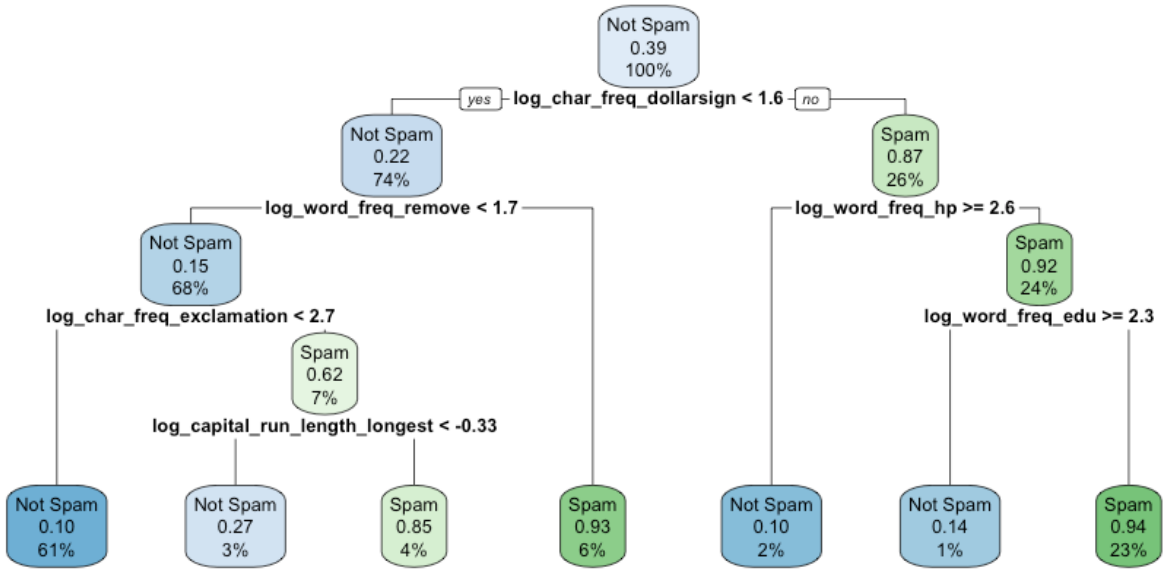


FIGURE 7: Decision tree plot displaying decision rule nodes

MODEL BUILDING

I chose to use 70% (3,220 observations) of the original dataset to train a series of classification models to predict whether or not an email is spam. These models were each validated against the remaining 30% (1,380) out-of-sample observations to determine which model achieved the best accuracy with a minimal fall-out (false positive) rate.

For all models created, the continuous variables were standardized to have a mean of value and standard deviation of one *for all non-zero values* before modeling and predicting. For the frequency variables, I first multiplied all values by 1000 before taking the log with base 10 in order to prevent the log transformation from creating negative values. The zero values were left as zeros. Additionally, variables were added to flag zero-values for all continuous variables to account for the case where the binary differentiating is better for predicting than the continuous variable. We also added high and low outlier indicator variables for data points that fell outside of the 99% and 1% quantiles for the full dataset.

In order to simplify model comparison, I created my.function.accuracy and my.function.fallout (see Appendix), which both calculate a list of model performance metrics including the accuracy rate, fall-out rate (false positive rate), and miss rate (false negative rate). my.function.accuracy scans through 99 potential cut-off values (from 0.01 to 0.99) to find the model with the highest accuracy. my.function.fallout scans through 99 potential cut-off values (from 0.01 to 0.99) to find the model with the lowest fall-out rate.

TREE MODEL

For the first set of models, I used the `rpart` and `tree` functions in R to build single decision trees. Model 1 and Model 3 included all variables, while Models 2 and 4 included only the flag variables for words / characters missing and for outliers. The four models are compared in Table 2.

The model that minimized the out-of-sample fall-out rate was the `rpart` model including all variables in the modeling process. This model misses 19.89% of the spam email on out-of-sample data. The plot in Figure 6 above shows which of these variables held importance in the model.

The model that performed best according to out-of-sample predictive accuracy used the `ctree` algorithm and also included all variables. While it did increase out-of-sample predictive accuracy by about 1%, it also increased the fall-out rate by over 1%. Of the four tree models, Model 1 appears to be the best for the spam filtering use case.

Table 2: Tree model comparison

	Accuracy	Fall-Out FPR	Miss Rate FNR	True Positive	False Positive	True Negative	False Negative
Model 1 Rpart all vars.	89.57%	3.92%	19.89%	451	32	785	112
Model 2 Rpart binary vars. only	86.88%	5.26%	24.51%	425	43	774	138
Model 3 Tree all vars.	90.65%	5.26%	15.28%	477	43	774	86
Model 4 Tree binary vars. only	87.61%	9.3%	16.87%	468	76	741	95

LOGISTIC REGRESSION MODELS WITH LASSO SUBSET SELECTION

Next, I used logistic regression to create a set of models including various combinations of the predictor variables. Models with relatively high fall-out rates were discarded. The results for the best models are shown in Table 3.

The best performing logistic model according to fall-out rate was created by initially using the 30 variables selected by the Lasso method and reducing down to only those that showed as significant when the model was first run. The resulting model had an out-of-sample accuracy of only 75.36%, but it is better than the other 0% fall-out rate model -- the full Lasso model. Table 4 shows the model coefficient estimates for the selected logistic regression model.

The one concern with this model is that the absence of the name “george” has the largest influence on the model prediction. Unless we are building a personalized prediction model that dynamically feeds in the email account owner’s name, this variable could be irrelevant in a production environment.

TABLE 3: Linear model comparison with and without variable selection

	Accuracy	Fall-Out FPR	Miss Rate FNR	True Positive	False Positive	True Negative	False Negative
Model 5 GLM with all vars. Cutoff at 0.48	93.99%	4.65%	7.99%	518	38	779	45
Model 6 GLM with all vars. Cutoff at 0.99	81.96%	1.47%	42.1%	326	12	805	237
Model 7 GLM with Lasso, Lambda Min, Alpha = 1, Cutoff at 0.40	93.91%	4.90%	7.82%	519	40	777	44
Model 8 GLM with Lasso, Lambda Min, Alpha = 1, Cutoff at 0.98	71.30%	0.00%	70.34%	167	0	817	396

Model 9 GLM with Lasso selected vars. (insig. Vars. removed) Cutoff at 0.42	92.32%	5.88%	10.30%	505	48	769	58
Model 10 GLM with Lasso selected vars. Cutoff at 0.99	75.36%	0.00%	60.39%	340	0	817	223

TABLE 4: Coefficients for best Logistic Regression Model

	Estimate	Std.	Error	z-value	Pr(>|z|)
(Intercept)	-10.26266	0.69702	-14.724	<2e-16	***
missing_word_george	4.00521	0.5382	7.442	9.93E-14	***
missing_word_1999	2.32899	0.26413	8.817	<2e-16	***
missing_word_edu	1.86965	0.34373	5.439	5.35E-08	***
log_word_freq_conference	-1.03171	0.2581	-3.997	6.41E-05	***
log_capital_run_length_average	0.90193	0.11732	7.688	1.50E-14	***
log_word_freq_remove	0.84512	0.10235	8.257	<2e-16	***
log_word_freq_meeting	-0.78231	0.16608	-4.711	2.47E-06	***
log_char_freq_dollarsign	0.77527	0.09218	8.41	<2e-16	***
log_char_freq_exclamation	0.66515	0.05884	11.305	<2e-16	***
log_word_freq_free	0.52594	0.06406	8.21	<2e-16	***
log_word_freq_000	0.37098	0.11198	3.313	0.000923	***
log_word_freq_our	0.36235	0.06103	5.937	2.90E-09	***
log_word_freq_re	-0.32627	0.0703	-4.641	3.47E-06	***
log_word_freq_internet	0.31838	0.08612	3.697	2.18E-04	***
log_capital_run_length_total	0.19642	0.13007	1.51	0.131022	
log_word_freq_business	0.18309	0.08356	2.191	2.84E-02	*

SUPPORT VECTOR MODELS (SVM)

The next set of models I created used Support Vector Machines (SVMs), which have more flexibility to contour around extremes and outliers than logistic regression or decision trees. For both models, I used the default radial kernel. For the first model a kept the gamma and cost variables set to the default levels (gamma = 1 / 214; cost = 1). For the second model, I had the SVM select the best combination of gamma (between 10^{-6} and 10^{-1}) and cost (10 or 100) using 10-fold cross validation. The selected model had a gamma of 0.01 and cost of 10. Despite tuning to the optimal parameters, neither of the SVMs performed well on fall-out rate (both >4%), though the accuracy for both models is quite high compared to previous models (>94%). The comparison of the models is shown in Table 5.

TABLE 5: SVM model comparison

	Accuracy	Fall-Out FPR	Miss Rate FNR	True Positive	False Positive	True Negative	False Negative
Model 11 SVM with all variables and default parameters	93.99%	4.04%	8.88%	784	33	513	50
Model 12 SVM with best gamma and cost (gamma = 0.01, cost = 10)	94.35%	4.41%	7.46%	781	36	521	42

RANDOM FOREST MODELS WITH BAGGING & BOOSTING

I created the last set of models using the aggregated decision tree methods of Bagging, Boosting, and Random Forests.

Between bagging and random forests, the bagged model with only 18 variables tried at each split performed better than the random forest models with 36 variables tried and the one with only 6 variables tried at each split. The results are compared in Table 6. All three models had accuracies over 95% and fall-out rates around 3%.

On the other hand, the boosted decision tree models by far outperformed every other classification model that I explored. The first boosted model was able to achieve an expected accuracy of 95.29% on the out-of-sample test data set. This model was created using the 0.01 shrinkage parameter (i.e. learning rate) and 5,000 trees with a probability cutoff at 0.47. Of the 214 continuous and flag variables put into the model, 93 had non-zero influence with the values of log_char_freq_dollarsign, log_char_freq_dollarsign, log_char_freq_exclamation, log_word_freq_remove, log_capital_run_length_average, and log_word_freq_hp variables having the highest importance. The second model was created using the same algorithm, but with a cutoff rate of 0.99. The accuracy of the model falls to 91.81% with this new cutoff, but the fall-out rate falls down to 1.22%. This is the most balanced of any of the models between accuracy and fall-out.

TABLE 6: Random Forests, Bagging, and Boosting model comparison

	Accuracy	Fall-Out FPR	Miss Rate FNR	True Positive	False Positive	True Negative	False Negative
Model 13 RandomForest with mtry=36	94.93%	3.3%	7.64%	790	27	520	43
Model 14 RandomForest with mtry=6	94.28%	2.94%	9.77%	793	24	508	55
Model 15 RandomForest with mtry=18	95.07%	2.69%	8.17%	795	22	517	46
Model 16 5000 trees, shrinkage = 0.1, Cutoff at 0.47	95.29%	3.92%	5.86%	785	32	530	33
Model 17 5000 trees, shrinkage = 0.1, Cutoff at 0.99	91.81%	1.22%	18.29%	807	10	460	103

MODEL COMPARISON & CONCLUSIONS

Of all the models created during this exercise, the highest accuracy model that predicts a 0% fall-out rate is Model 10 -- the logistic regression model that used a subset of the variables selected by Lasso selection. The model has an out-of-sample prediction accuracy of 75%. The coefficients for this model are shown in Table 4.

If there is any tolerance for an occasional non-spam email to be caught in the spam filter, the next best model is Model 17, which has a slight increase in fall-out rate to 1.22%, but a big jump in accuracy to 92%. This model is from the bagged random forest model with 0.1 shrinkage.

Other models performed better according to overall out-of-sample accuracy, but the aim here was to find a model with relatively high accuracy (and minimally above the baseline of 61%) that could maintain a zero or very low false positive rate for spam.

	Accuracy	Fall-Out FPR	Miss Rate FNR	True Positive	False Positive	True Negative	False Negative
Model 10 GLM with Lasso selected vars. Cutoff at 0.99	75.36%	0.00%	60.39%	340	0	817	223
Model 17 5000 trees, shrinkage = 0.1, Cutoff at 0.99	91.81%	1.22%	18.29%	807	10	460	103

As a next step for this exercise, I would look at including interaction variables in the model building process to see if the associations between any variables added new predictive value to the models. In addition, there could be significant value in combining two or more of these models into one ensemble model that performs better in combination than any one of the individual models alone.

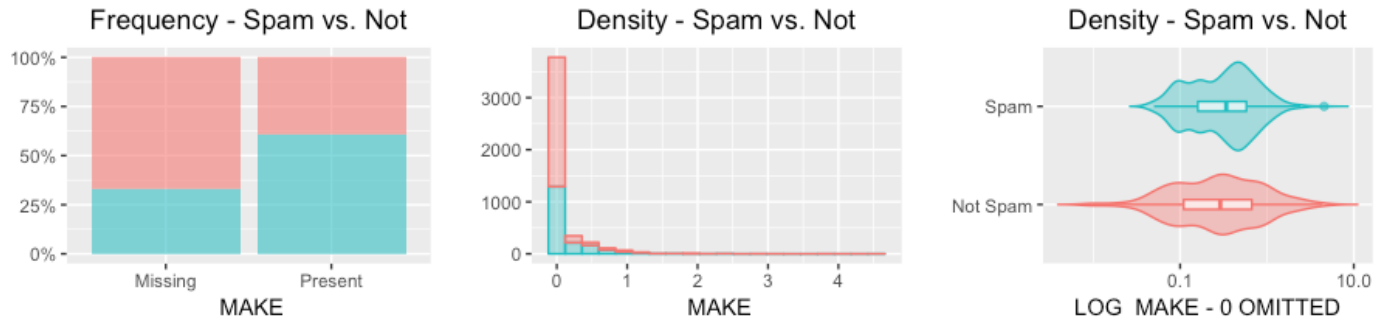
APPENDIX A

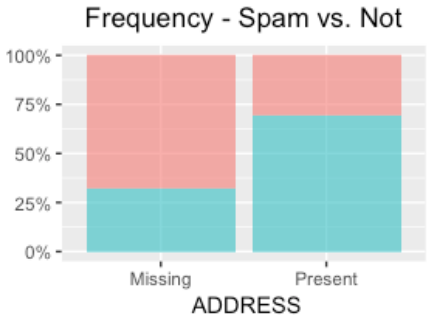
TABLE A.1: Detailed description of 57 original variables

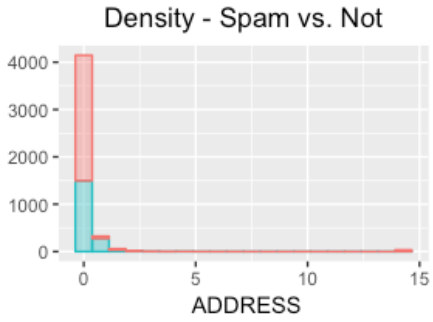
Variable Name	Data Type	Description
word_freq_make	continuous [0,100]	% of words in email that match "make"
word_freq_address	continuous [0,100]	% of words in email that match "address"
word_freq_all	continuous [0,100]	% of words in email that match "all"
word_freq_3d	continuous [0,100]	% of words in email that match "3d"
word_freq_our	continuous [0,100]	% of words in email that match "our"
word_freq_over	continuous [0,100]	% of words in email that match "over"
word_freq_remove	continuous [0,100]	% of words in email that match "remove"
word_freq_internet	continuous [0,100]	% of words in email that match "internet"
word_freq_order	continuous [0,100]	% of words in email that match "order"
word_freq_mail	continuous [0,100]	% of words in email that match "mail"
word_freq_receive	continuous [0,100]	% of words in email that match "receive"
word_freq_will	continuous [0,100]	% of words in email that match "will"
word_freq_people	continuous [0,100]	% of words in email that match "people"
word_freq_report	continuous [0,100]	% of words in email that match "report"
word_freq_addresses	continuous [0,100]	% of words in email that match "addresses"
word_freq_free	continuous [0,100]	% of words in email that match "free"
word_freq_business	continuous [0,100]	% of words in email that match "business"
word_freq_email	continuous [0,100]	% of words in email that match "email"
word_freq_you	continuous [0,100]	% of words in email that match "you"
word_freq_credit	continuous [0,100]	% of words in email that match "credit"
word_freq_your	continuous [0,100]	% of words in email that match "your"
word_freq_font	continuous [0,100]	% of words in email that match "font"
word_freq_000	continuous [0,100]	% of words in email that match "000"
word_freq_money	continuous [0,100]	% of words in email that match "money"
word_freq_hp	continuous [0,100]	% of words in email that match "hp"
word_freq_hpl	continuous [0,100]	% of words in email that match "hpl"
word_freq_george	continuous [0,100]	% of words in email that match "george"
word_freq_650	continuous [0,100]	% of words in email that match "650"
word_freq_lab	continuous [0,100]	% of words in email that match "lab"
word_freq_labs	continuous [0,100]	% of words in email that match "labs"
word_freq_telnet	continuous [0,100]	% of words in email that match "telnet"
word_freq_857	continuous [0,100]	% of words in email that match "857"
word_freq_data	continuous [0,100]	% of words in email that match "data"
word_freq_415	continuous [0,100]	% of words in email that match "415"

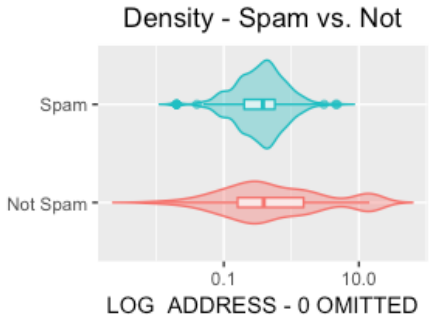
word_freq_85	continuous [0,100]	% of words in email that match "85"
word_freq_technology	continuous [0,100]	% of words in email that match "technology"
word_freq_1999	continuous [0,100]	% of words in email that match "1999"
word_freq_parts	continuous [0,100]	% of words in email that match "parts"
word_freq_pm	continuous [0,100]	% of words in email that match "pm"
word_freq_direct	continuous [0,100]	% of words in email that match "direct"
word_freq_cs	continuous [0,100]	% of words in email that match "cs"
word_freq_meeting	continuous [0,100]	% of words in email that match "meeting"
word_freq_original	continuous [0,100]	% of words in email that match "original"
word_freq_project	continuous [0,100]	% of words in email that match "project"
word_freq_re	continuous [0,100]	% of words in email that match "re"
word_freq_edu	continuous [0,100]	% of words in email that match "edu"
word_freq_table	continuous [0,100]	% of words in email that match "table"
word_freq_conference	continuous [0,100]	% of words in email that match "conference"
char_freq_semicolon	continuous [0,100]	% of characters in email that match ";"
char_freq_parantheses	continuous [0,100]	% of characters in email that match "("
char_freq_brackets	continuous [0,100]	% of characters in email that match "["
char_freq_exclamation	continuous [0,100]	% of characters in email that match "!"
char_freq_dollarsign	continuous [0,100]	% of characters in email that match "$"
char_freq_hashtag	continuous [0,100]	% of characters in email that match "#"
capital_run_length_average	continuous [1,...]	Avg. length of uninterrupted seq. of capital letters.
capital_run_length_longest	continuous [1,...]	Longest length of uninterrupted seq. of capital letters.
capital_run_length_total	continuous [1,...]	Total length of uninterrupted seq. of capital letters.
is_spam	nominal {0,1}	**Response Variable.** Whether the message is spam (1) or not (0).

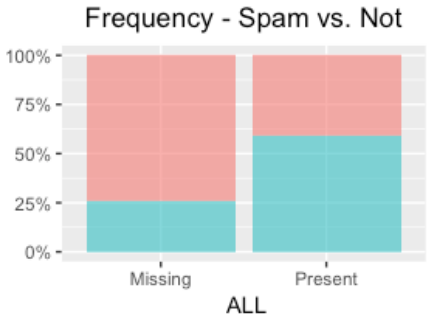
FIGURE A.1: Frequency, histogram, and violin plots for all predictor variables

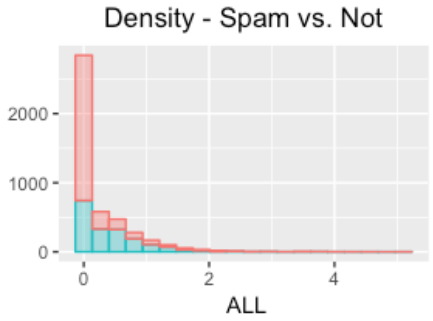


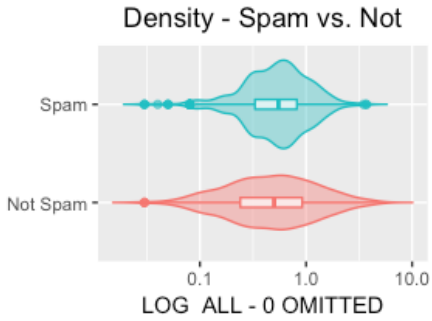

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
ADDRESS

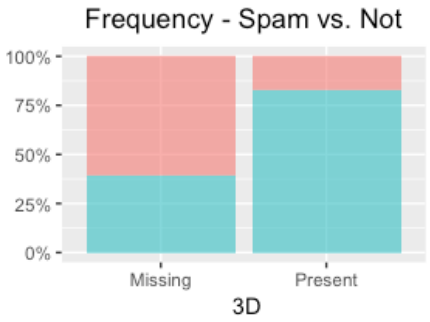

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
ADDRESS

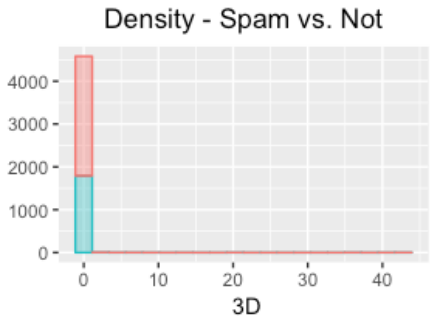

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG ADDRESS - 0 OMITTED

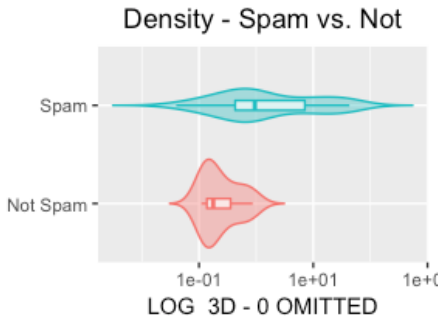

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
ALL

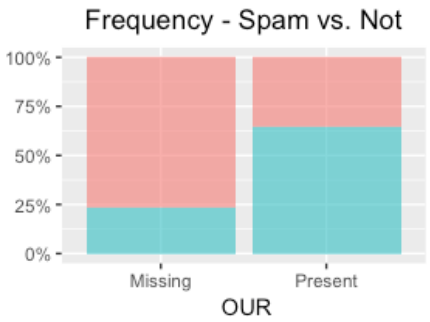

Density - Spam vs. Not
2000
1000
0
0
2
4
ALL

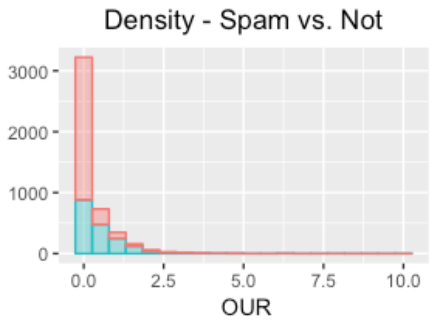

Density - Spam vs. Not
Spam
Not Spam
0.1
1.0
10.0
LOG ALL - 0 OMITTED

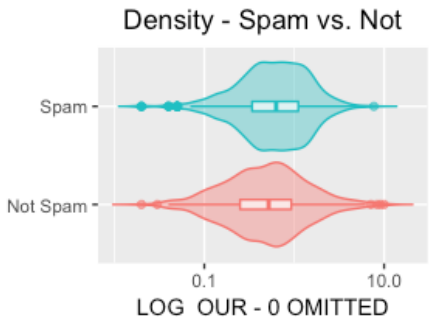

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
3D

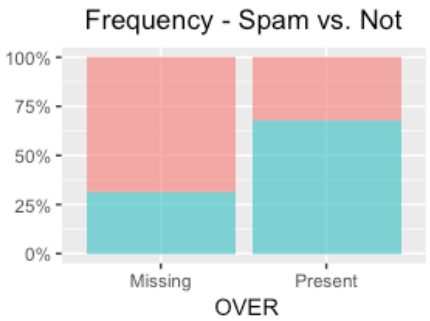

Density - Spam vs. Not
4000
3000
2000
1000
0
0
10
20
30
40
3D

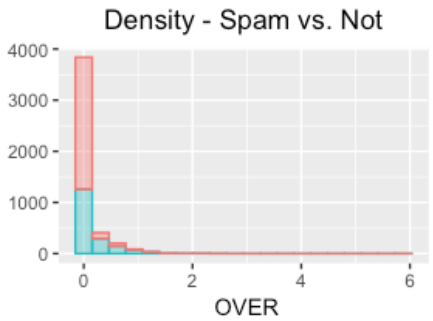

Density - Spam vs. Not
Spam
Not Spam
1e-01
1e+01
LOG 3D - 0 OMITTED

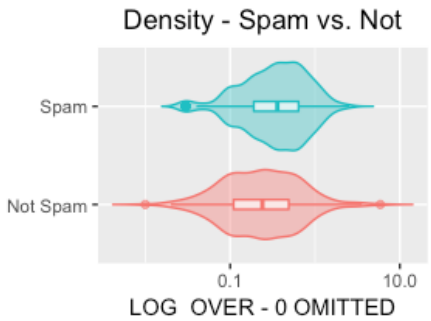

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
OUR

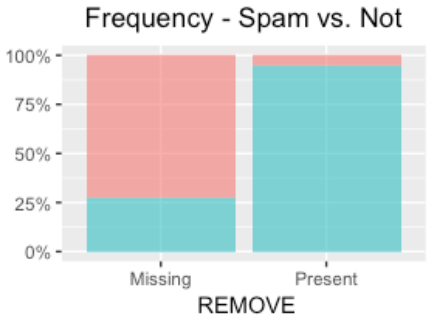

Density - Spam vs. Not
3000
2000
1000
0
0.0
2.5
5.0
7.5
10.0
OUR

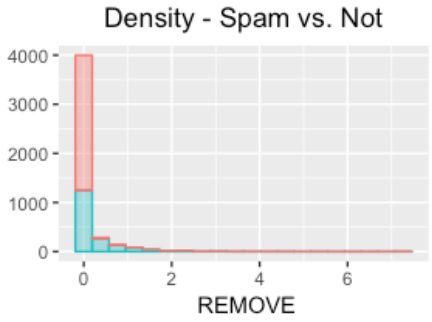

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG OUR - 0 OMITTED

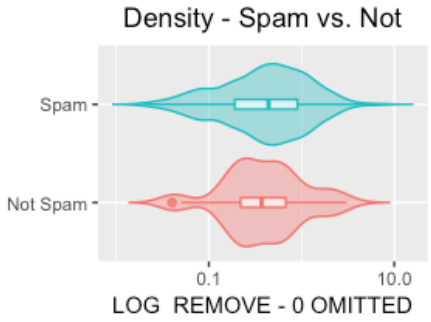

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
OVER

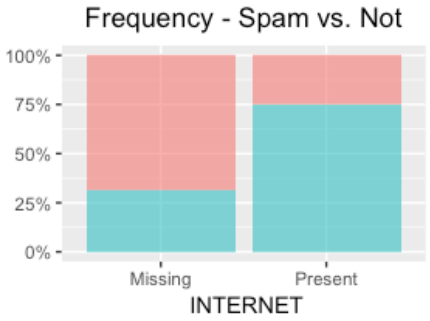

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
OVER

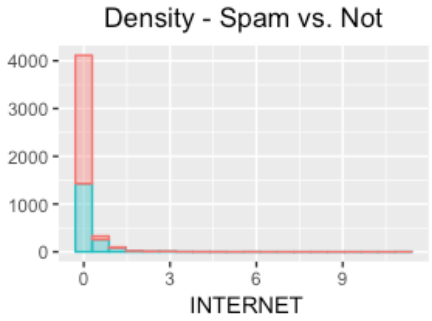

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG OVER - 0 OMITTED

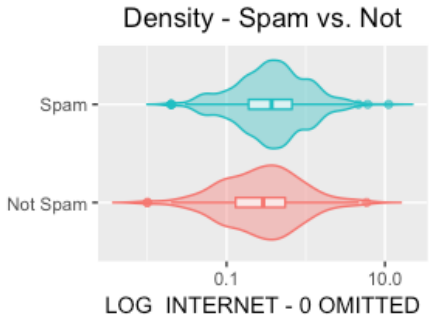

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
REMOVE

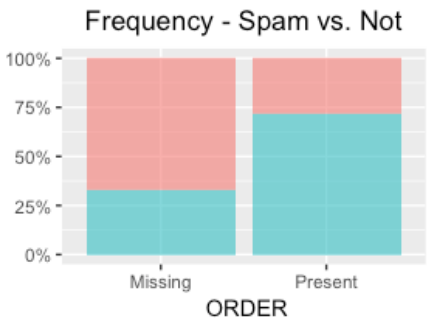

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
REMOVE

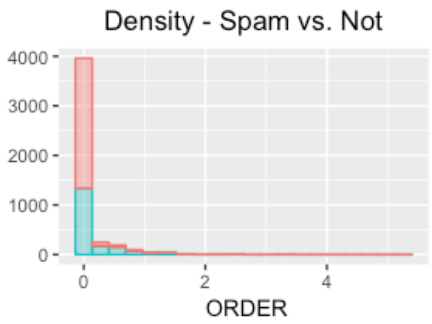

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG REMOVE - 0 OMITTED

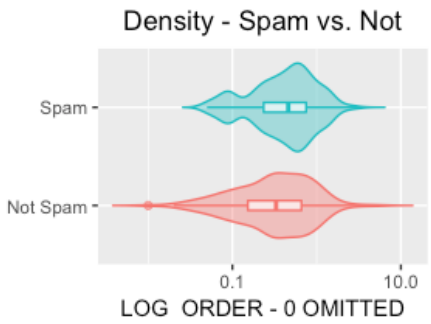

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
INTERNET

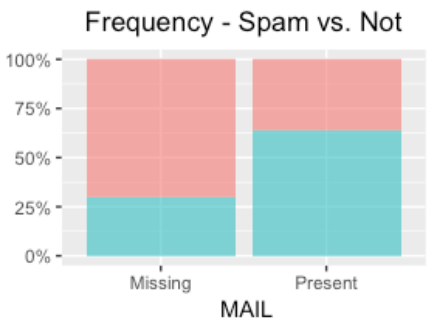

Density - Spam vs. Not
4000
3000
2000
1000
0
0
3
6
9
INTERNET

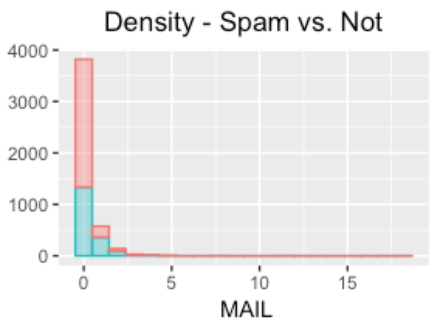

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG INTERNET - 0 OMITTED

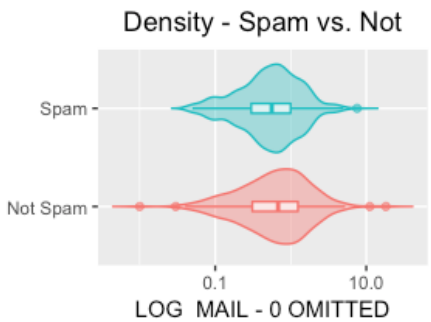

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
ORDER

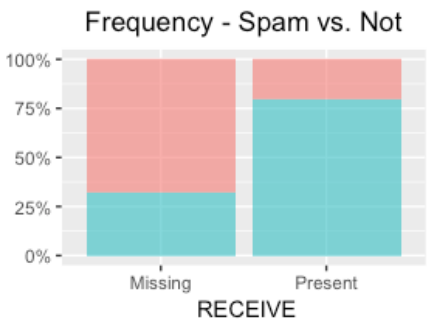

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
ORDER

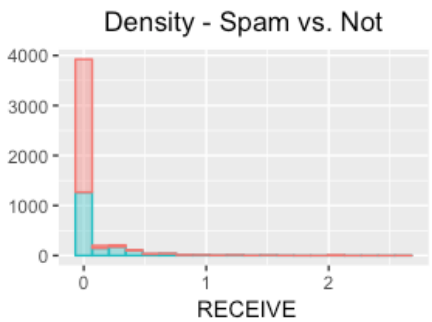

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG ORDER - 0 OMITTED

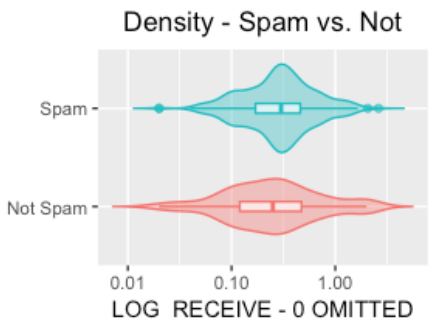

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
MAIL

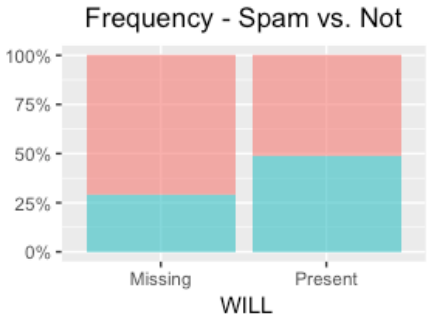

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
MAIL

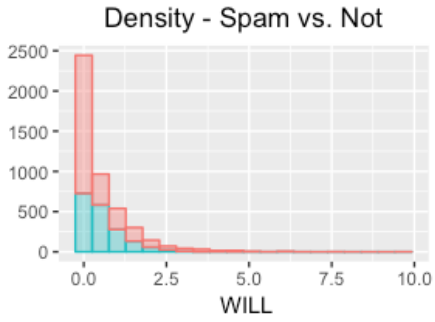

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG MAIL - 0 OMITTED

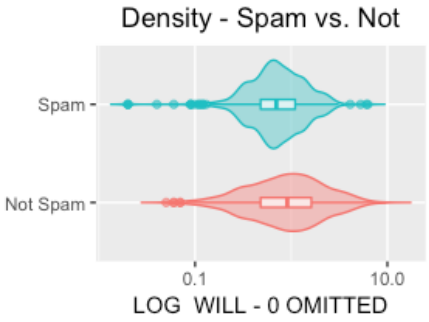

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
RECEIVE

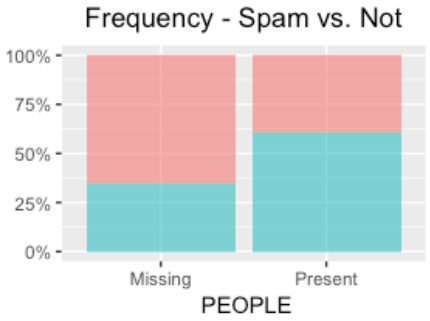

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
RECEIVE

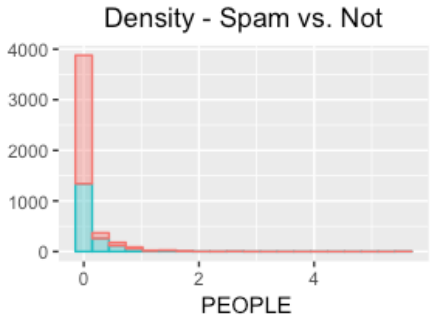

Density - Spam vs. Not
Spam
Not Spam
0.01
0.10
1.00
LOG RECEIVE - 0 OMITTED

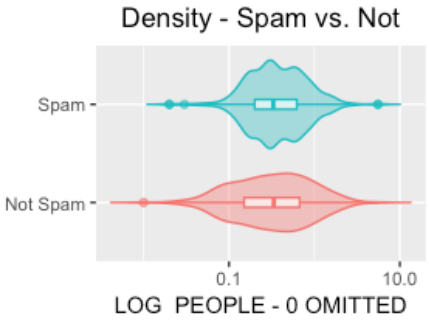

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
WILL

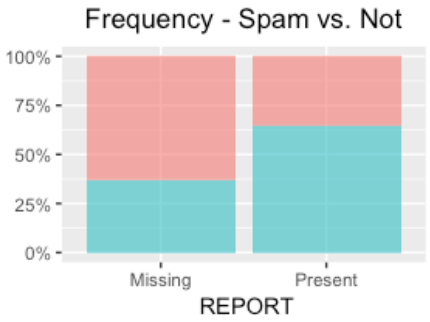

Density - Spam vs. Not
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
WILL

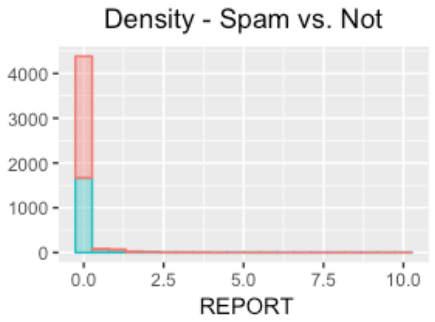

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG WILL - 0 OMITTED

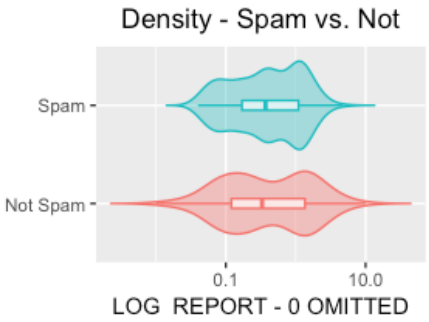

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
PEOPLE

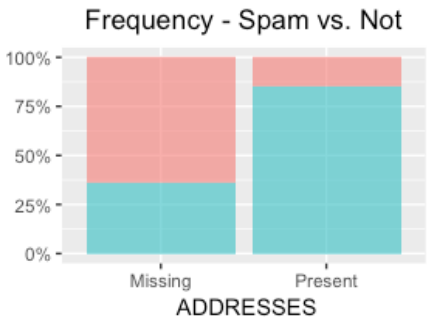

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
PEOPLE

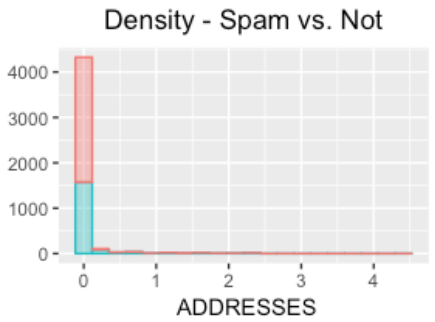

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG PEOPLE - 0 OMITTED

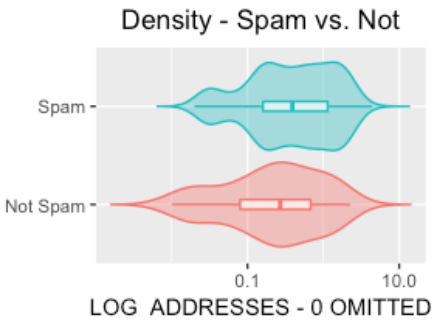

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
REPORT

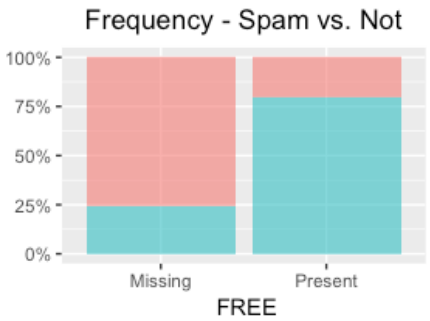

Density - Spam vs. Not
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
10.0
REPORT

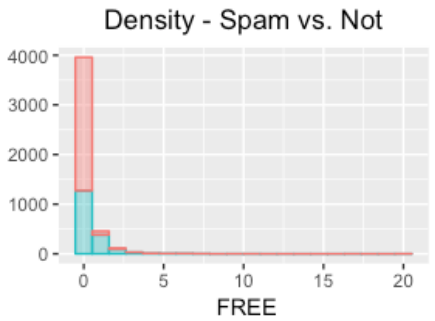

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG REPORT - 0 OMITTED

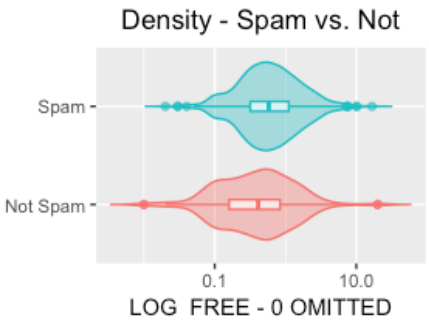

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
ADDRESSES

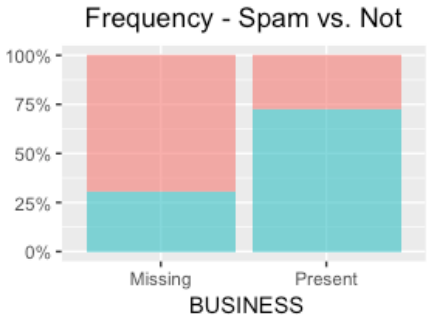

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
4
ADDRESSES

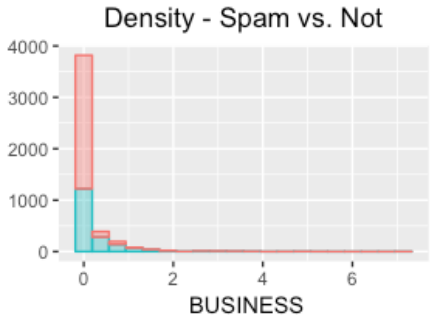

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG ADDRESSES - 0 OMITTED

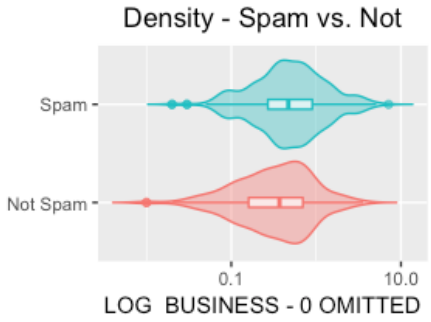

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
FREE

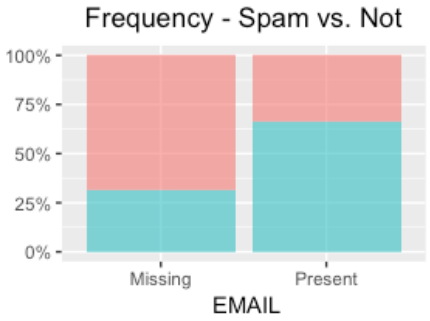

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
20
FREE

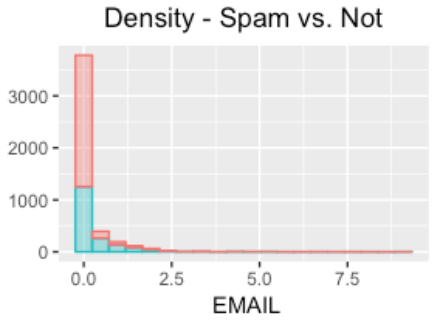

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG FREE - 0 OMITTED

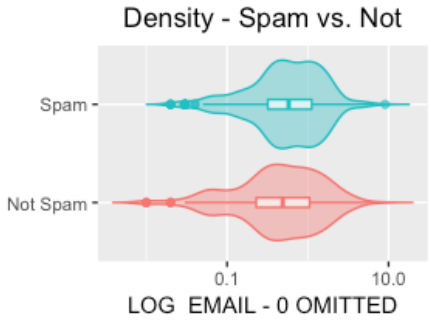

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
BUSINESS

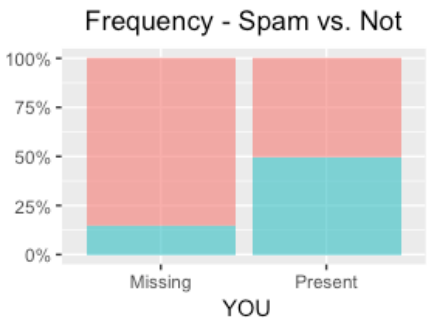

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
BUSINESS

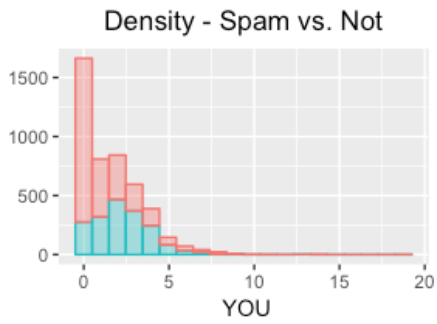

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG BUSINESS - 0 OMITTED

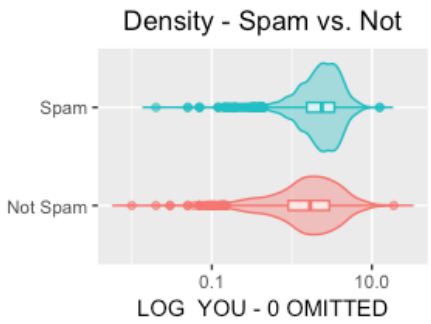

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
EMAIL

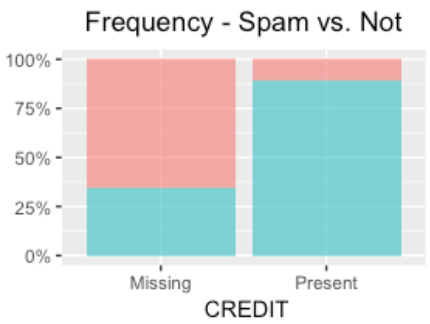

Density - Spam vs. Not
3000
2000
1000
0
0.0
2.5
5.0
7.5
EMAIL

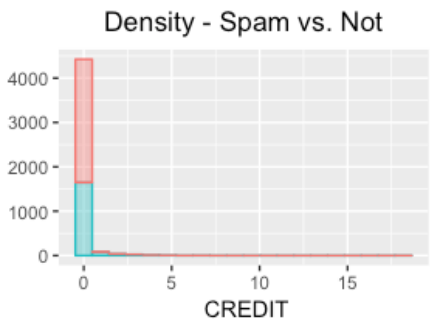

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG EMAIL - 0 OMITTED

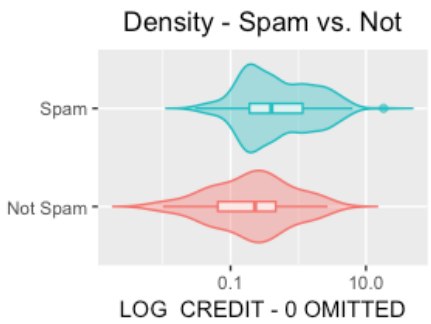

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
YOU

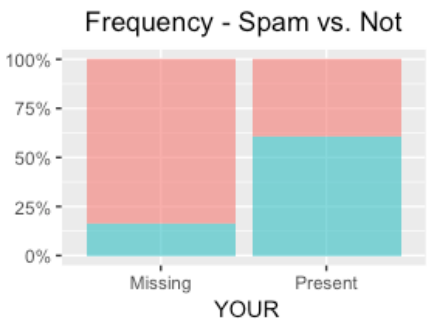

Density - Spam vs. Not
1500
1000
500
0
0
5
10
15
20
YOU

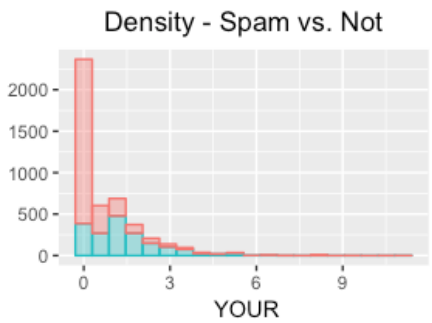

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG YOU - 0 OMITTED

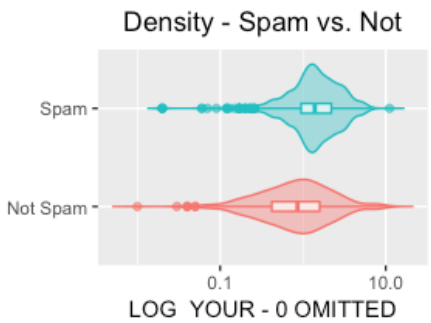

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
CREDIT

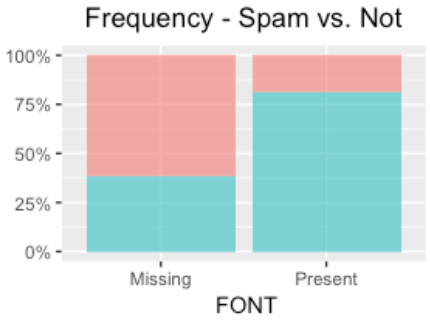

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
CREDIT

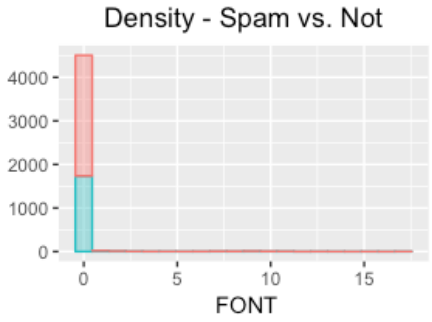

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG CREDIT - 0 OMITTED

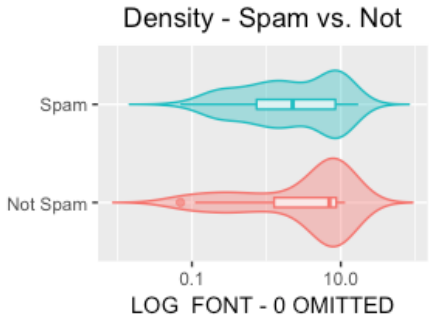

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
YOUR

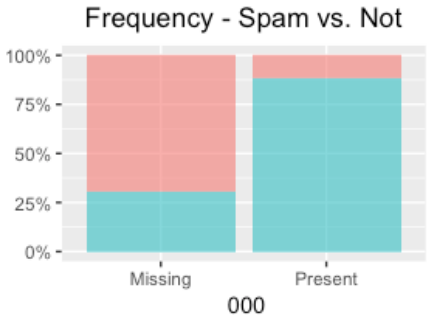

Density - Spam vs. Not
2000
1500
1000
500
0
0
3
6
9
YOUR

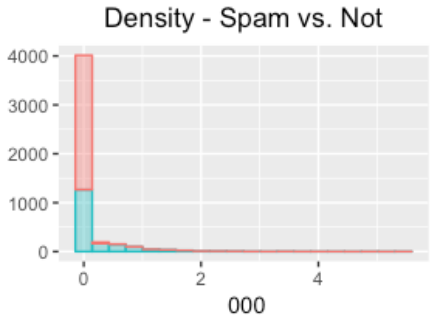

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG YOUR - 0 OMITTED

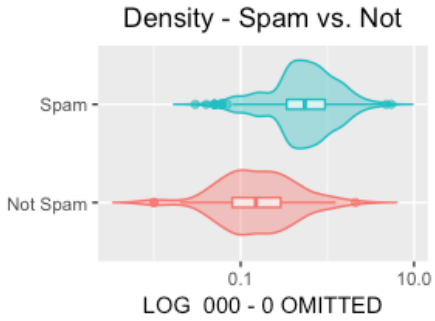

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
FONT

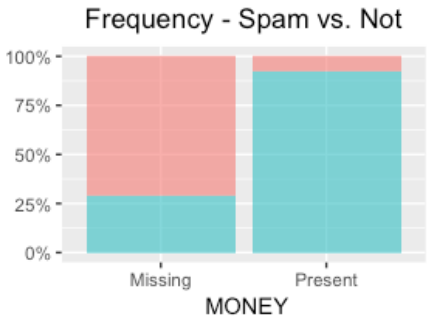

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
FONT

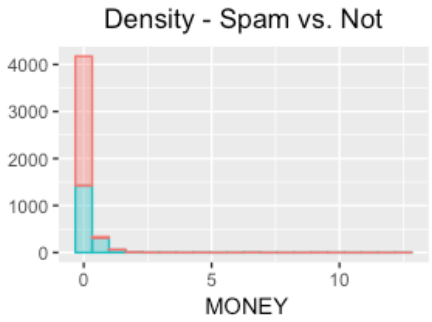

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG FONT - 0 OMITTED

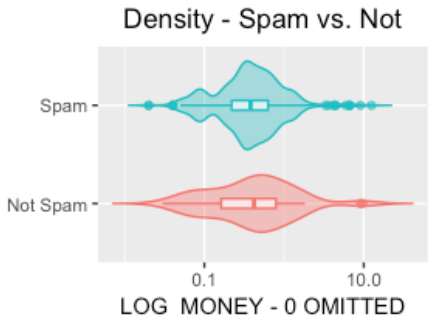

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
000

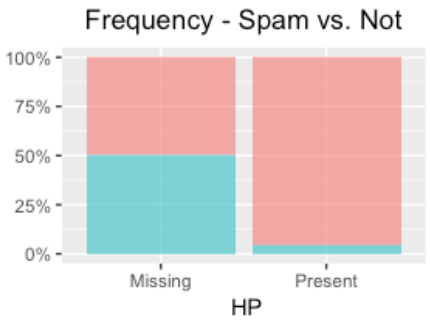

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
000

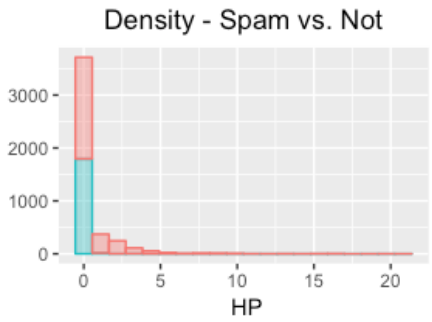

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG 000 - 0 OMITTED

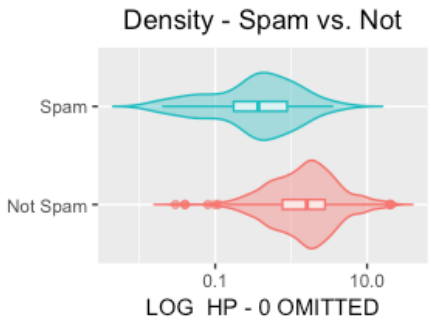

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
MONEY

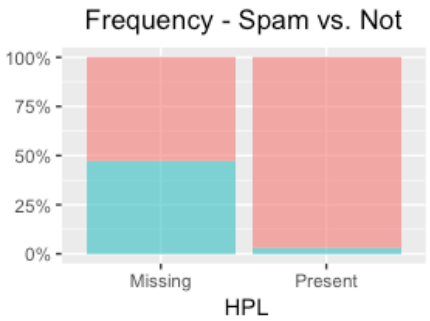

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
MONEY

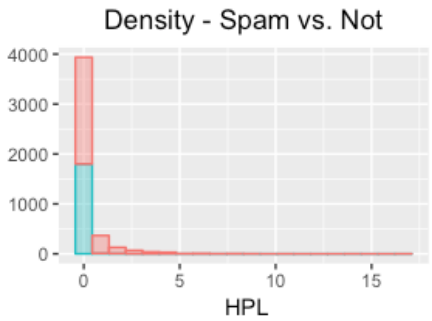

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG MONEY - 0 OMITTED

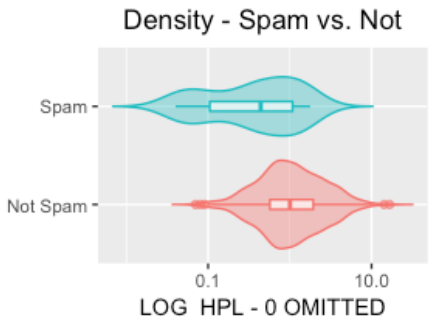

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
HP

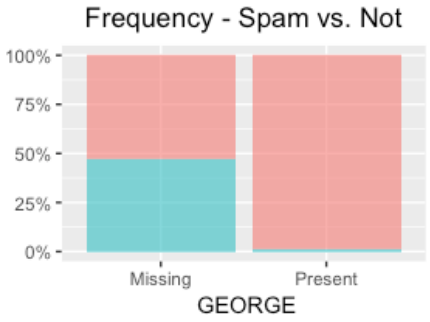

Density - Spam vs. Not
3000
2000
1000
0
0
5
10
15
20
HP

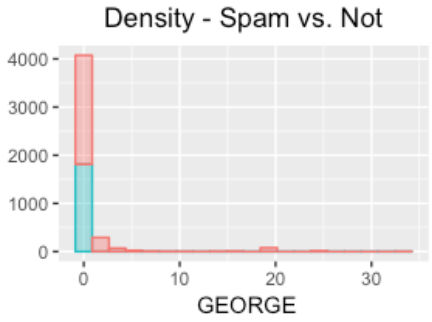

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG HP - 0 OMITTED

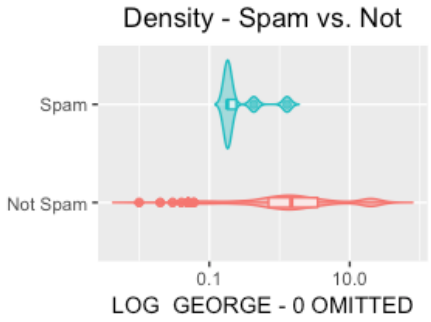

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
HPL

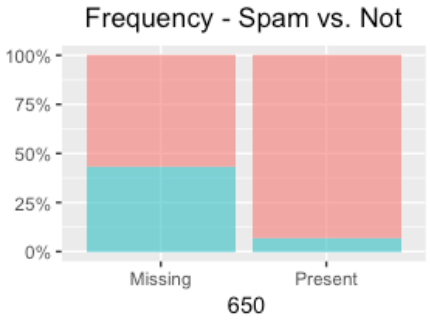

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
HPL

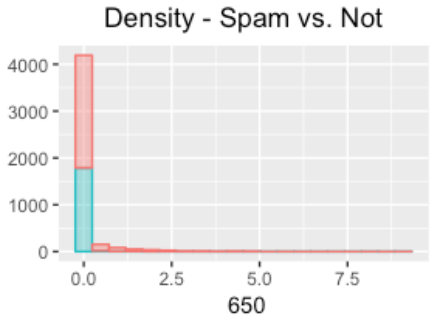

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG HPL - 0 OMITTED

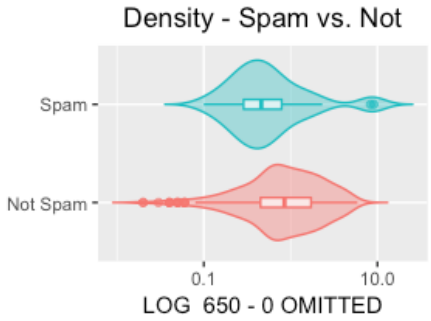

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
GEORGE

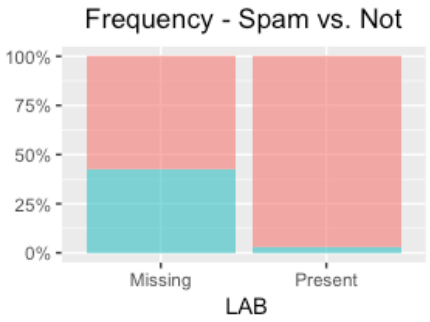

Density - Spam vs. Not
4000
3000
2000
1000
0
0
10
20
30
GEORGE

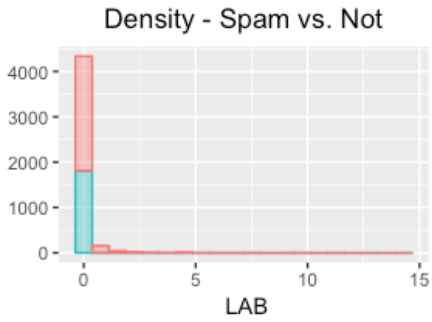

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG GEORGE - 0 OMITTED

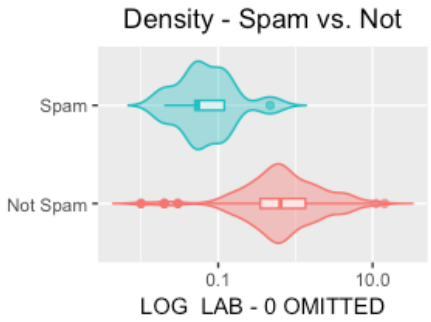

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
650

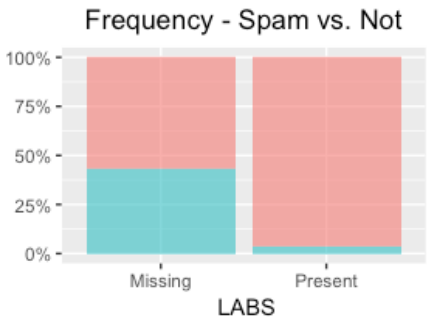

Density - Spam vs. Not
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
650

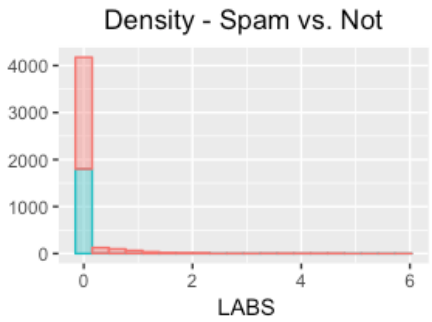

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG 650 - 0 OMITTED

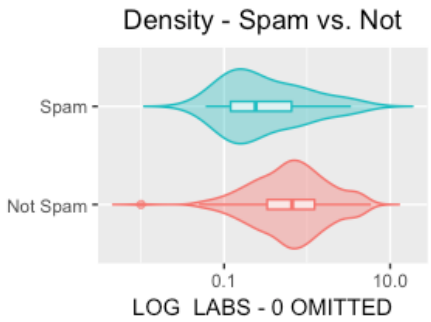

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
LAB

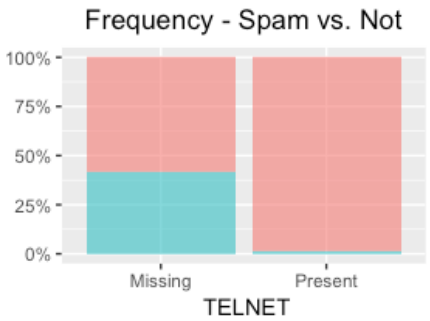

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
LAB

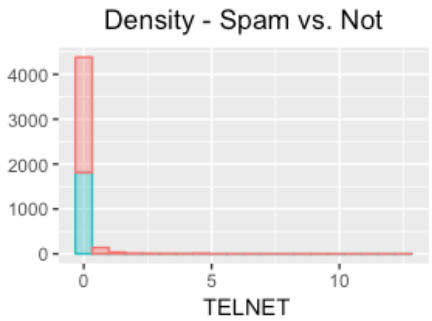

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG LAB - 0 OMITTED

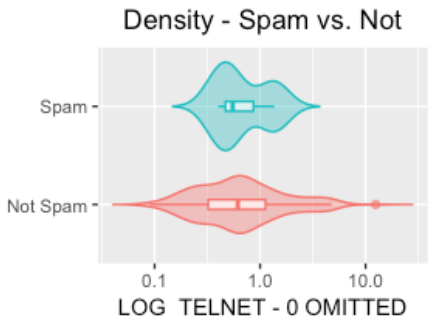

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
LABS

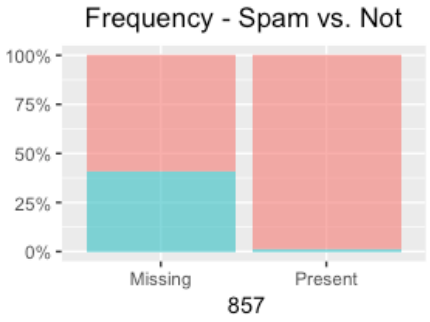

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
LABS

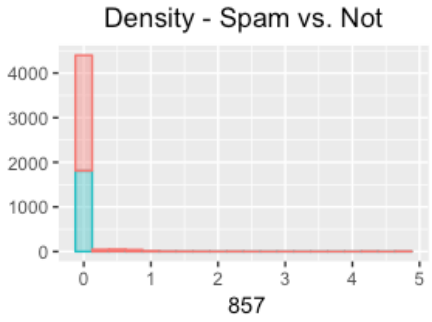

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG LABS - 0 OMITTED

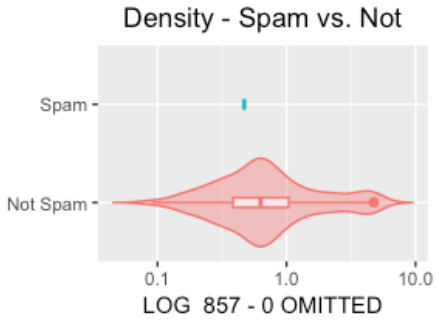

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
TELNET

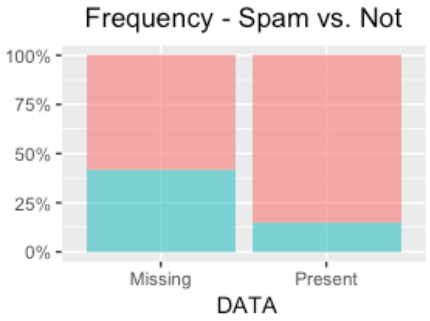

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
TELNET

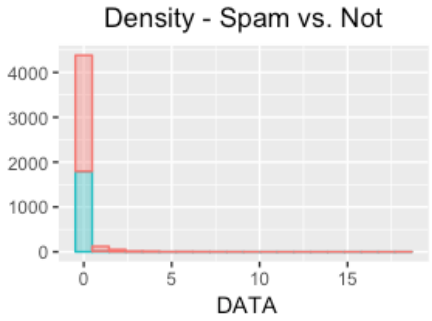

Density - Spam vs. Not
Spam
Not Spam
0.1
1.0
10.0
LOG TELNET - 0 OMITTED

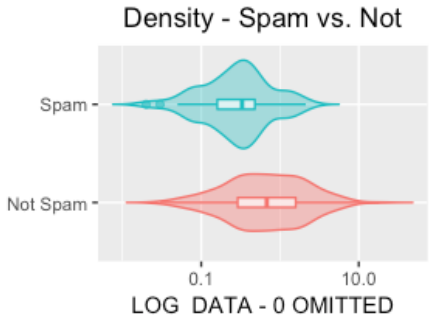

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
857

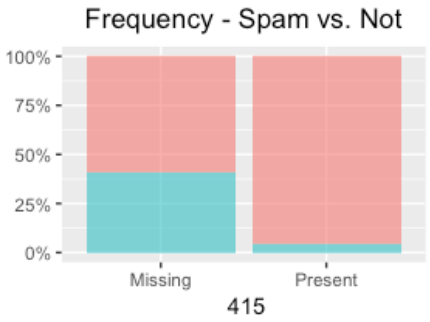

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
4
5
857

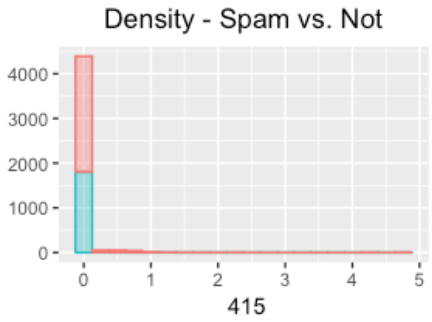

Density - Spam vs. Not
Spam
Not Spam
0.1
1.0
10.0
LOG 857 - 0 OMITTED

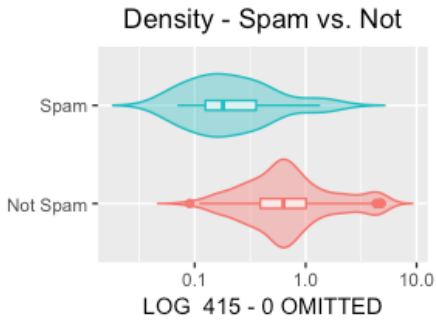

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
DATA

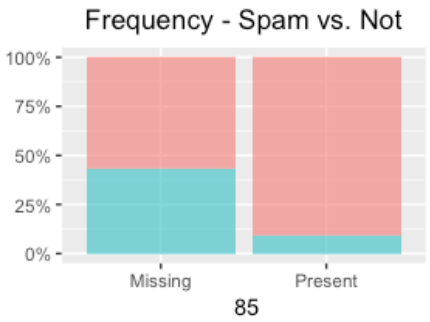

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
DATA

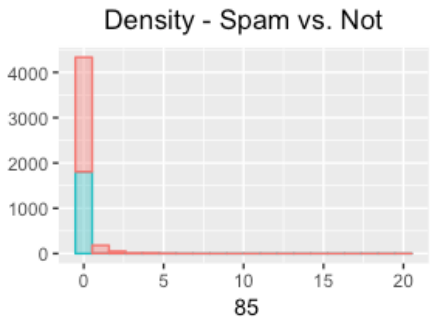

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG DATA - 0 OMITTED

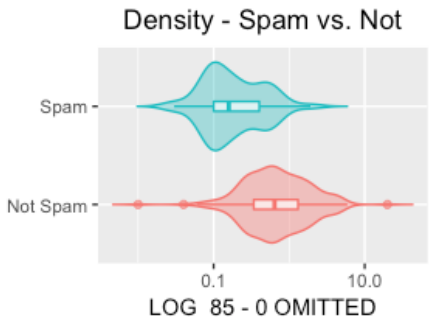

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
415

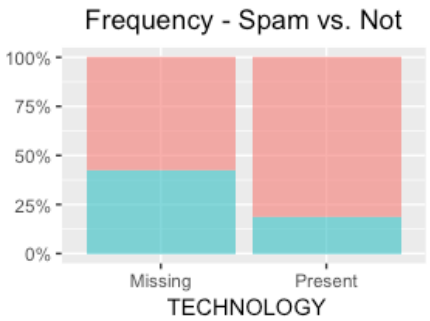

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
4
5
415

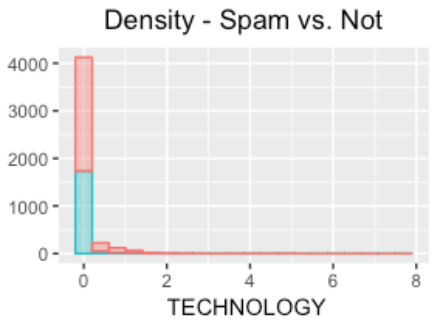

Density - Spam vs. Not
Spam
Not Spam
0.1
1.0
10.0
LOG 415 - 0 OMITTED

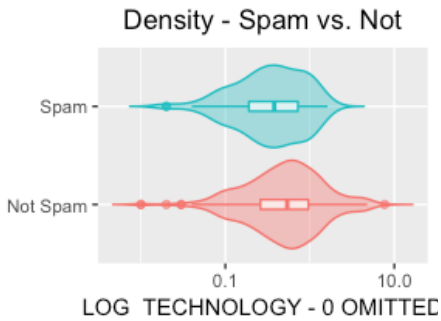

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
85

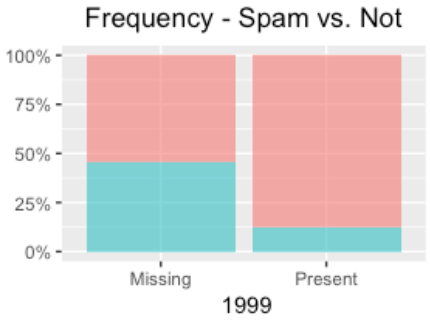

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
20
85

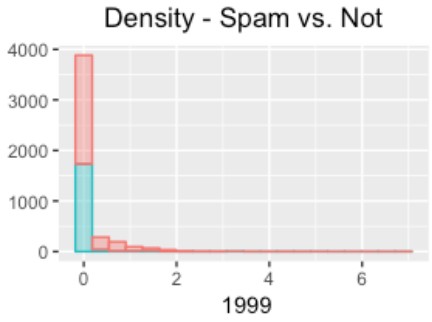

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG 85 - 0 OMITTED

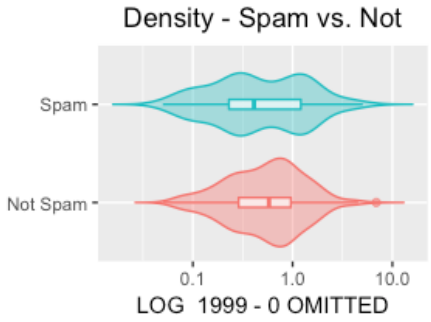

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
TECHNOLOGY

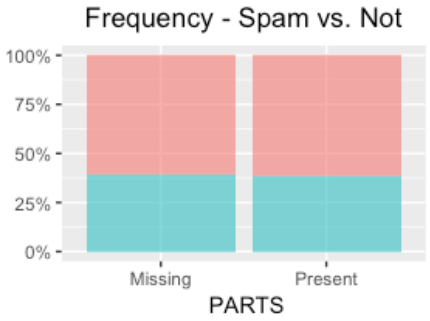

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
8
TECHNOLOGY

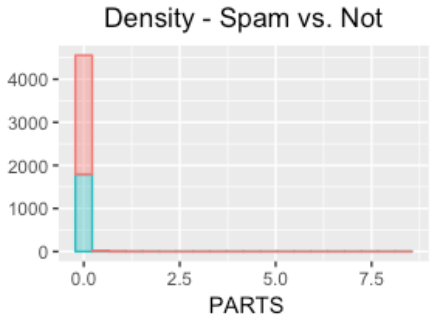

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG TECHNOLOGY - 0 OMITTED

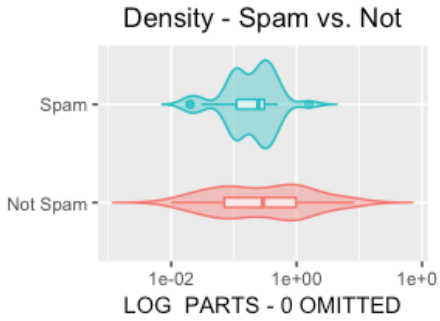

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
1999

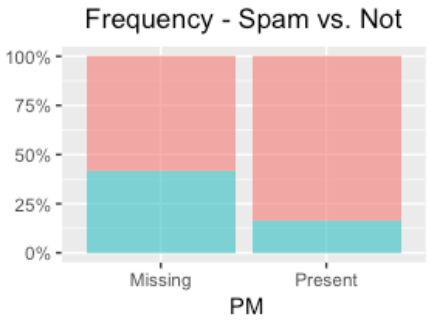

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
1999

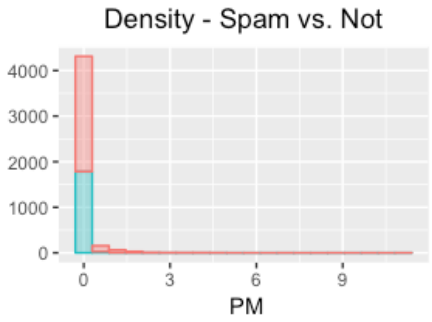

Density - Spam vs. Not
Spam
Not Spam
0.1
1.0
10.0
LOG 1999 - 0 OMITTED

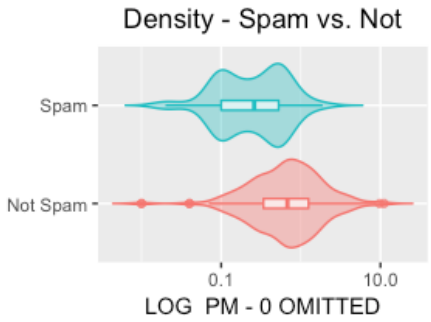

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
PARTS

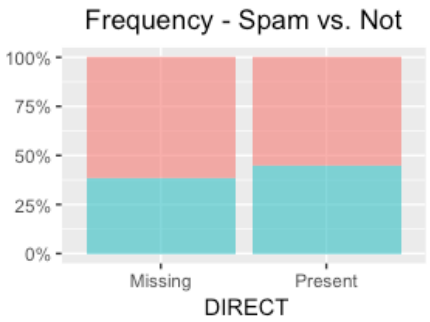

Density - Spam vs. Not
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
PARTS

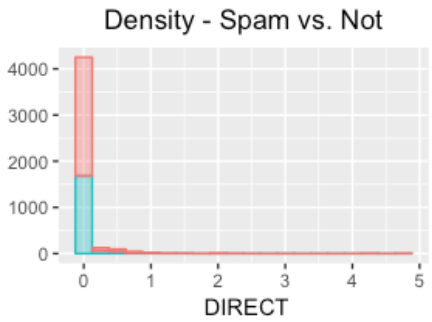

Density - Spam vs. Not
Spam
Not Spam
1e-02
1e+00
1e+0
LOG PARTS - 0 OMITTED

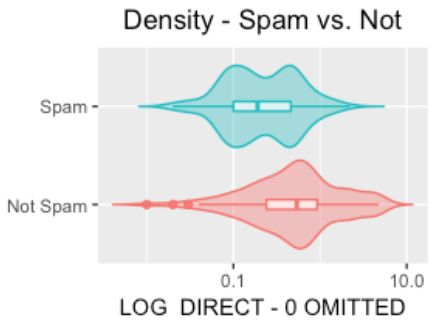

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
PM

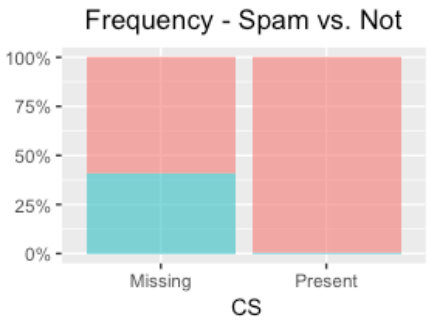

Density - Spam vs. Not
4000
3000
2000
1000
0
0
3
6
9
PM

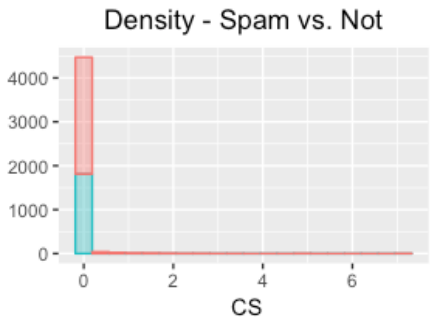

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG PM - 0 OMITTED

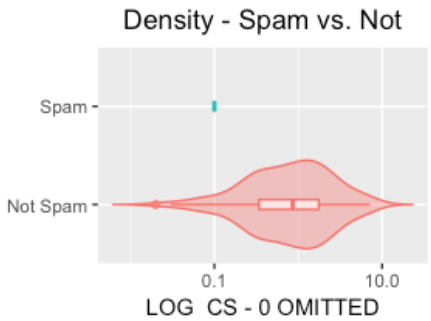

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
DIRECT

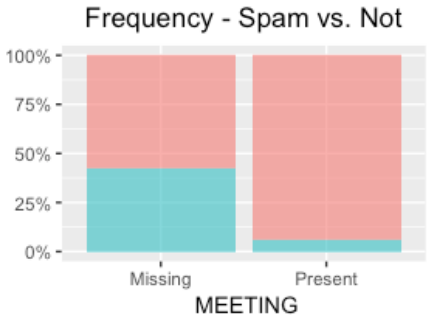

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
4
5
DIRECT

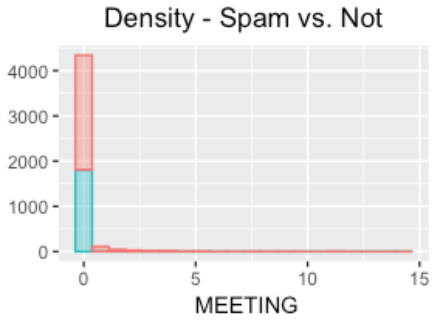

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG DIRECT - 0 OMITTED

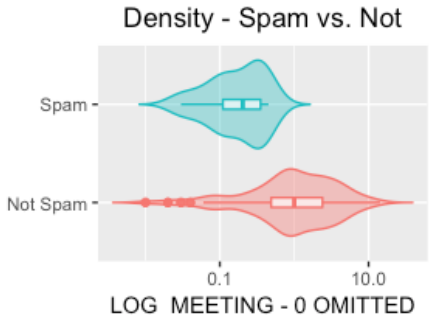

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
CS

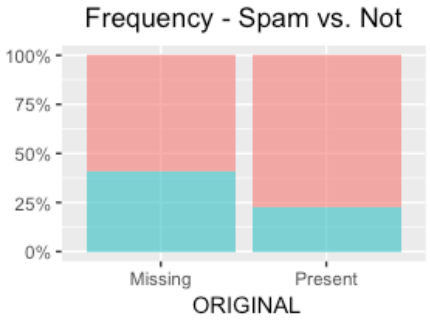

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
CS

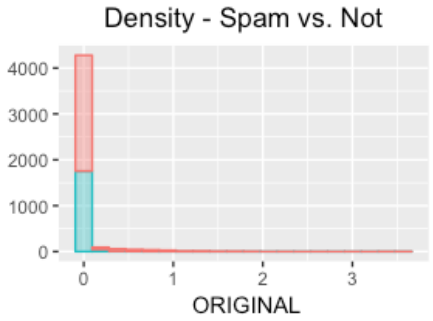

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG CS - 0 OMITTED

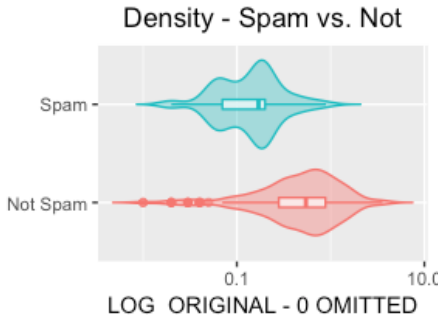

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
MEETING

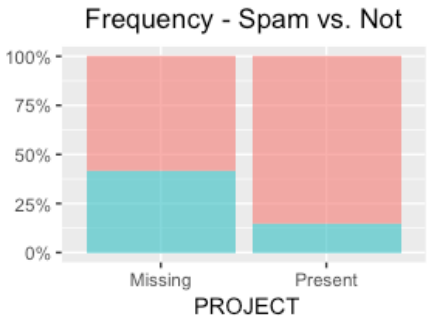

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
MEETING

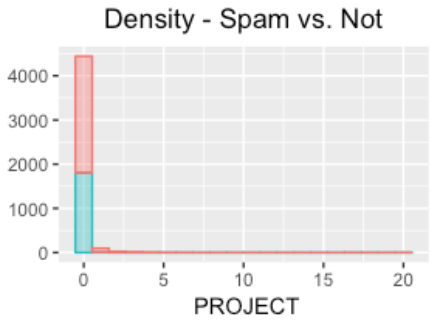

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG MEETING - 0 OMITTED

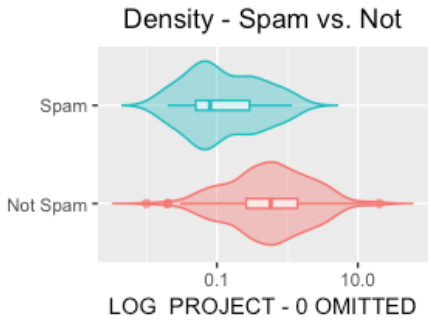

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
ORIGINAL

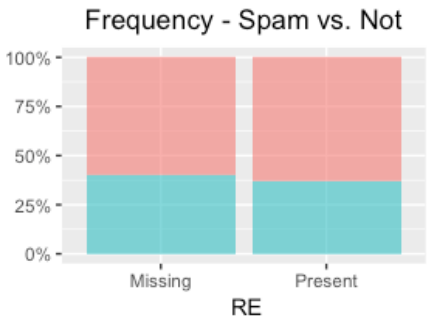

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
ORIGINAL

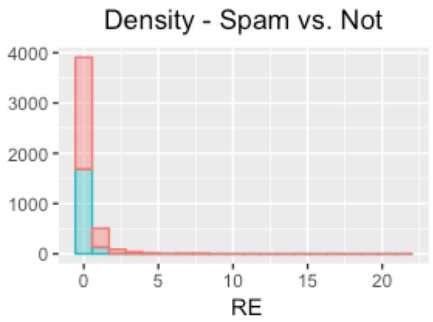

Density - Spam vs. Not
Spam
Not Spam
0.1
LOG ORIGINAL - 0 OMITTED

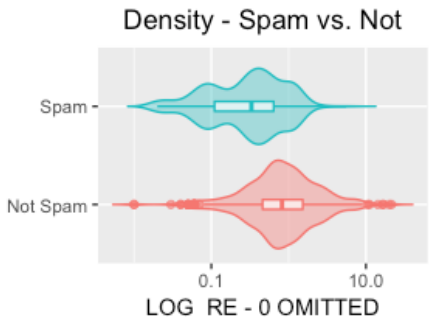

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
PROJECT

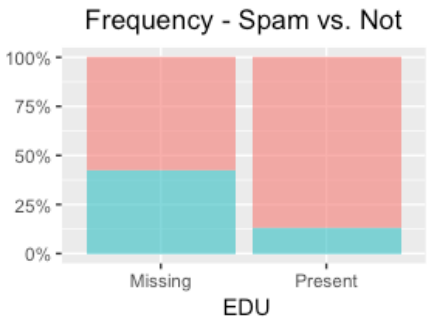

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
20
PROJECT

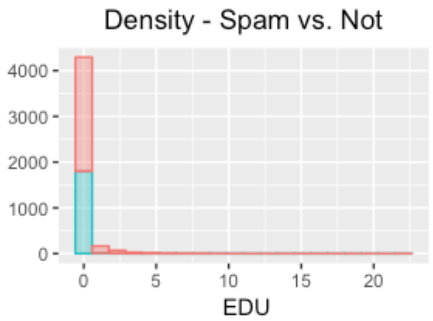

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG PROJECT - 0 OMITTED

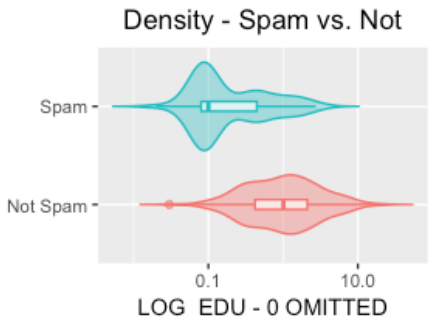

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
RE

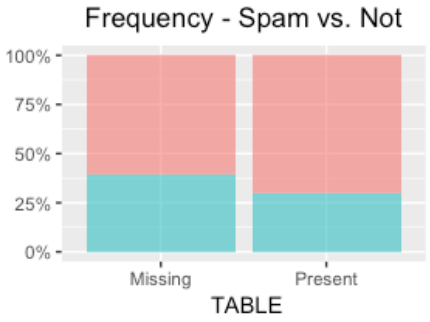

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
20
RE

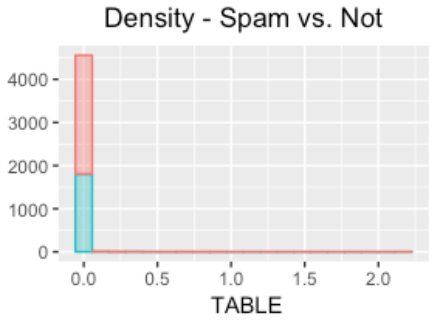

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG RE - 0 OMITTED

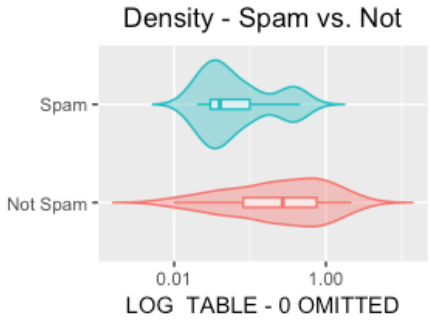

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
EDU

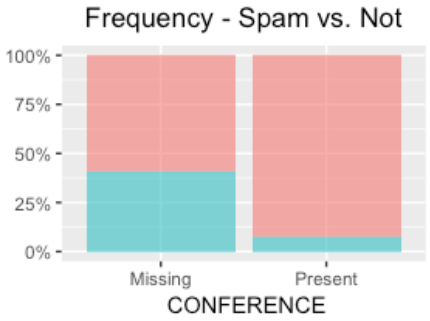

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
20
EDU

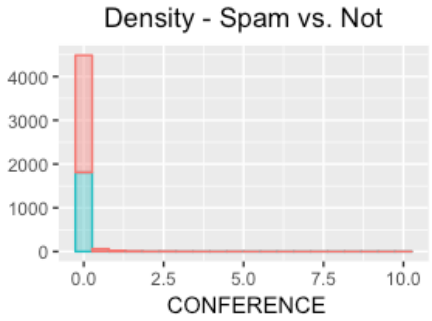

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG EDU - 0 OMITTED

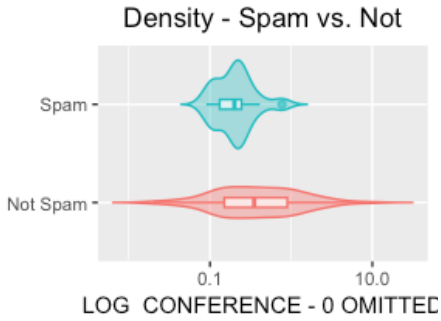

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
TABLE

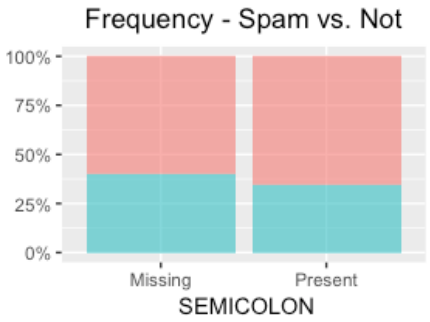

Density - Spam vs. Not
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0
TABLE

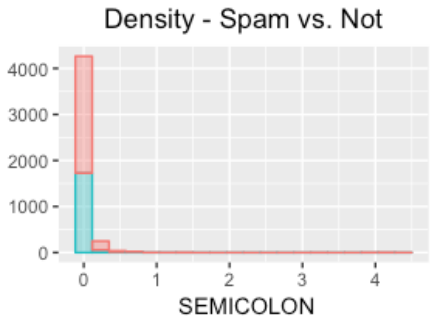

Density - Spam vs. Not
Spam
Not Spam
0.01
1.00
LOG TABLE - 0 OMITTED

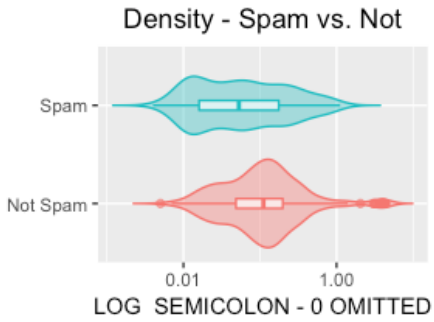

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
CONFERENCE

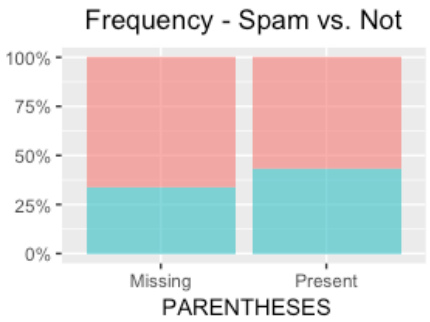

Density - Spam vs. Not
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
10.0
CONFERENCE

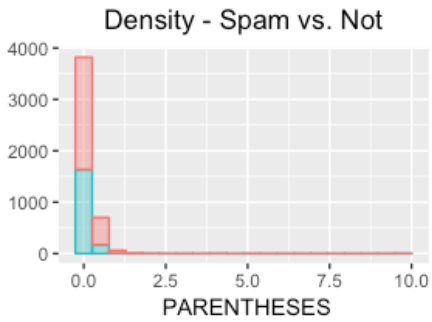

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG CONFERENCE - 0 OMITTED

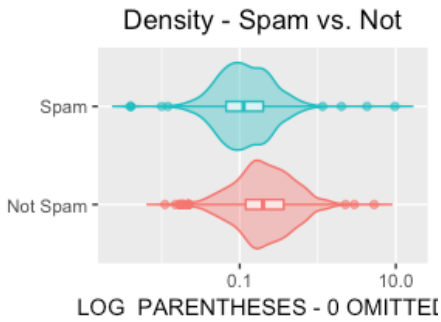

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
SEMICOLON

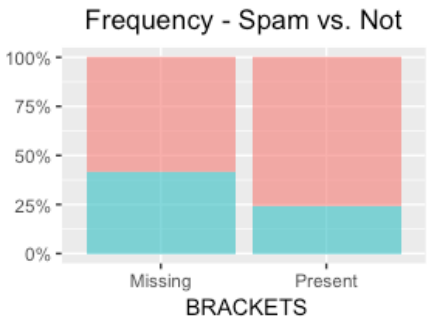

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
4
SEMICOLON

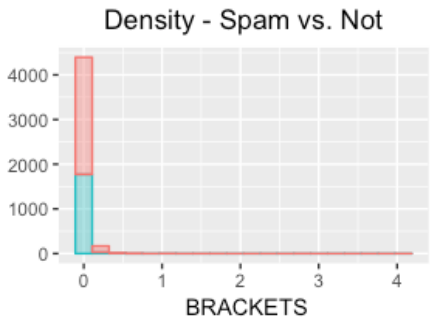

Density - Spam vs. Not
Spam
Not Spam
0.01
1.00
LOG SEMICOLON - 0 OMITTED

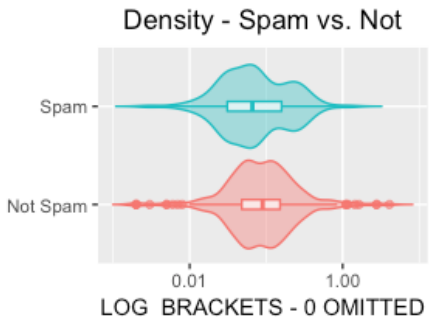

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
PARENTHESES

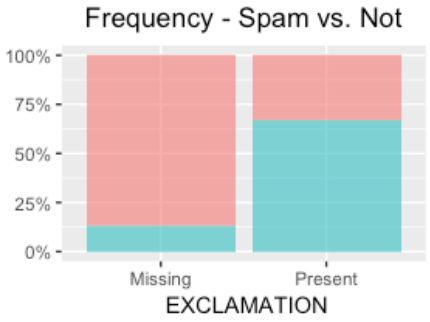

Density - Spam vs. Not
4000
3000
2000
1000
0
0.0
2.5
5.0
7.5
10.0
PARENTHESES

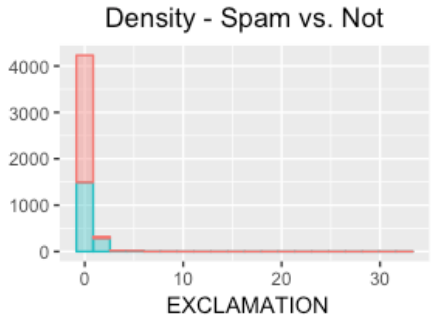

Density - Spam vs. Not
Spam
Not Spam
0.1
10.0
LOG PARENTHESES - 0 OMITTED

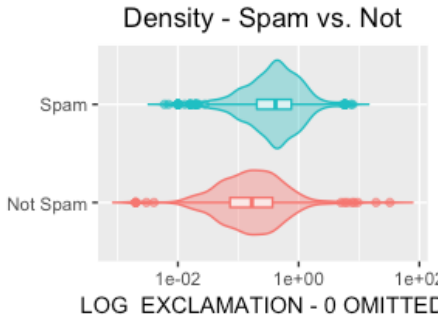

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
BRACKETS

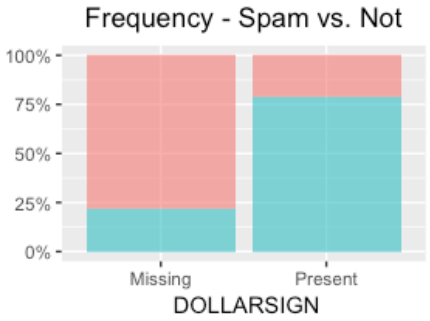

Density - Spam vs. Not
4000
3000
2000
1000
0
0
1
2
3
4
BRACKETS

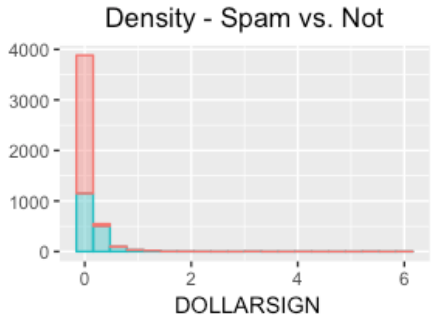

Density - Spam vs. Not
Spam
Not Spam
0.01
1.00
LOG BRACKETS - 0 OMITTED

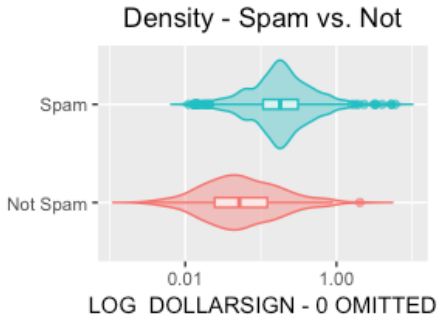

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
EXCLAMATION

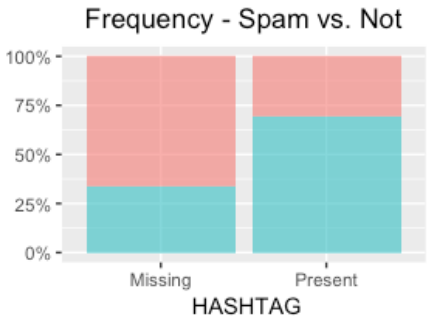

Density - Spam vs. Not
4000
3000
2000
1000
0
0
10
20
30
EXCLAMATION

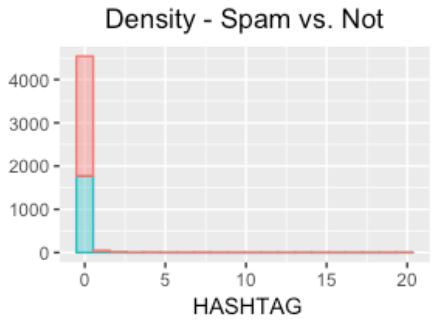

Density - Spam vs. Not
Spam
Not Spam
1e-02
1e+00
LOG EXCLAMATION - 0 OMITTED

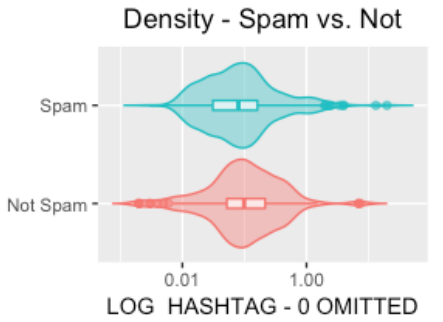

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
DOLLARSIGN

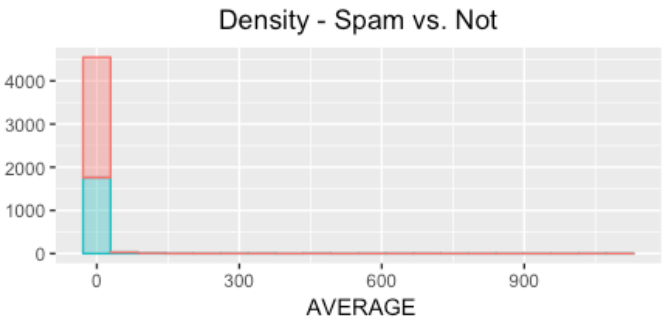

Density - Spam vs. Not
4000
3000
2000
1000
0
0
2
4
6
DOLLARSIGN

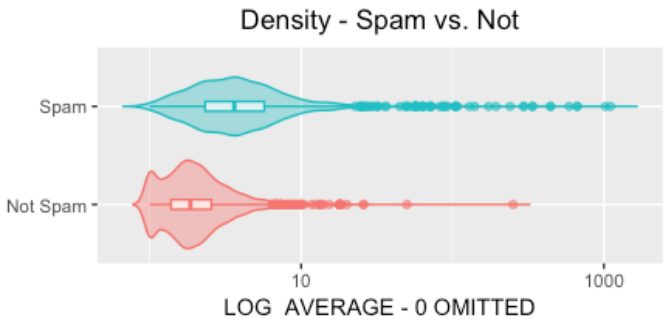

Density - Spam vs. Not
Spam
Not Spam
0.01
1.00
LOG DOLLARSIGN - 0 OMITTED

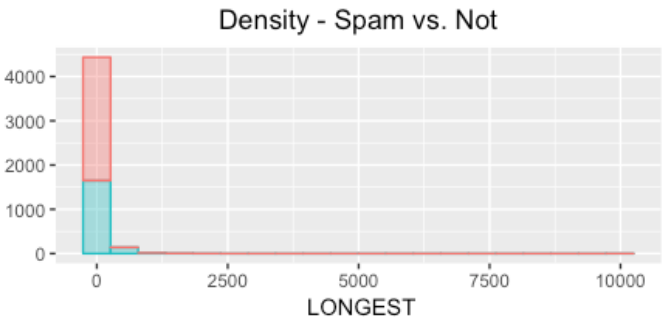

Frequency - Spam vs. Not
100%
75%
50%
25%
0%
Missing
Present
HASHTAG

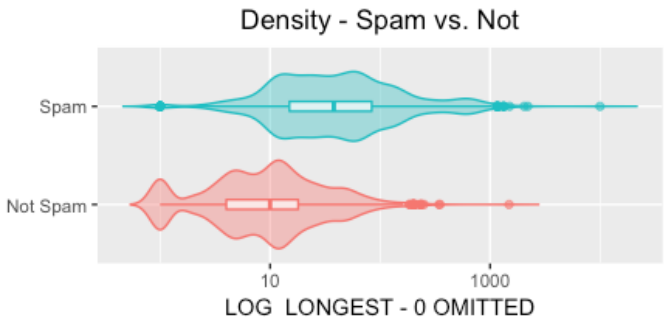

Density - Spam vs. Not
4000
3000
2000
1000
0
0
5
10
15
20
HASHTAG

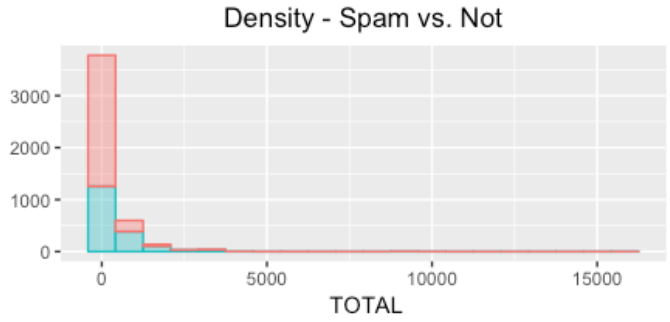

Density - Spam vs. Not
Spam
Not Spam
0.01
1.00
LOG HASHTAG - 0 OMITTED

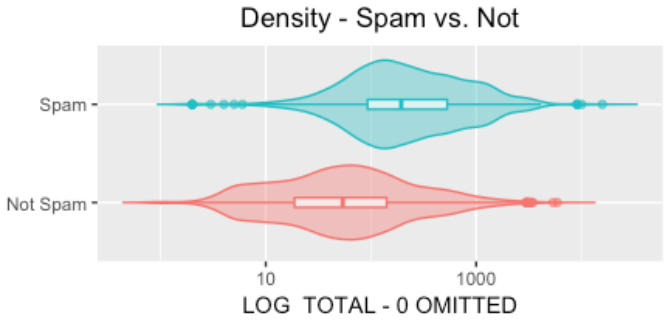

Density - Spam vs. Not
4000
3000
2000
1000
0
0
300
600
900
AVERAGE


Density - Spam vs. Not
Spam
Not Spam
10
1000
LOG AVERAGE - 0 OMITTED


Density - Spam vs. Not
4000
3000
2000
1000
0
0
2500
5000
7500
10000
LONGEST


Density - Spam vs. Not
Spam
Not Spam
10
1000
LOG LONGEST - 0 OMITTED





APPENDIX B

R Code for Bivariate Plots

```
library(ggplot2)
library(scales)
library(gridExtra)

var.list = names(spam)[c(1:57,169)]
density.plot.list<- list()
for (i in 1:57) {
    d <- ggplot(spam, aes_string(x = var.list[[i]], color = var.list[[58]], fill = var.list[[58]])) +
        geom_histogram(alpha = 0.4, bins = 20) +
        #scale_x_log10() +
        #scale_y_continuous(labels = percent_format(),) +
        xlab(toupper(gsub(".*_", " ", var.list[[i]]))) +
        ylab("") +
        ggtitle("Density - Spam vs. Not") +
        theme(legend.position="none", plot.title = element_text(hjust = 0.5))
    density.plot.list[[i]] <- d
}
rm(i,d,var.list)

var.list = names(spam)[c(1:57,169)]
violin.plot.list<- list()
for (i in 1:57) {
    v <- ggplot(spam, aes_string(y = var.list[[i]], x = var.list[[58]], color = var.list[[58]], fill =
var.list[[58]])) +
        geom_violin(alpha = 0.4, trim = FALSE) +
        geom_boxplot(width = 0.1, fill = "white", alpha = 0.6) +
        scale_y_log10() +
        #scale_y_continuous(labels = percent_format(), limits = c(0,1.25)) +
        xlab("") +
        ylab(toupper(paste0("LOG ", gsub(".*_", " ", var.list[[i]])," - 0 OMITTED"))) +
        coord_flip() +
        ggtitle("Density - Spam vs. Not") +
        theme(legend.position="none", plot.title = element_text(hjust = 0.5))
    violin.plot.list[[i]] <- v
}
rm(i,v,var.list)

var.list = names(spam)[c(58:111,169)]
frequency.plot.list<- list()
for (i in 1:54) {
    f <- ggplot(spam, aes_string(x=var.list[[i]], fill = var.list[[55]]), color = var.list[[55]]) +
        geom_bar(position = "fill", alpha = 0.6) +
        scale_y_continuous(labels = percent_format()) +
        xlab(toupper(gsub(".*_", " ", var.list[[i]]))) +
        ylab("") +
        ggtitle("Frequency - Spam vs. Not") +
```

```
        theme(legend.position="none", plot.title = element_text(hjust = 0.5))
    frequency.plot.list[[i]] <- f
}
rm(i,f,var.list)
```

R Code for Accuracy Maximizing Cutoff Function

```
my.function.accuracy <- function(pred, valid, ref) {
    require(caret)
    pred.grid <- seq(0.01, 0.99, by = 0.01)
    accuracy <- c()
    fallout <- c()
    miss <- c()

    for (i in 1:length(pred.grid)){
        c.hat.valid <- ifelse(pred>=pred.grid[i], "Spam", "Not Spam")
        cf <- confusionMatrix(c.hat.valid, valid, positive = ref)
        accuracy[i] <-
round(100*((cf$table[1]+cf$table[4])/(cf$table[1]+cf$table[2]+cf$table[3]+cf$table[4])),2)
        fallout[i] <- round(100*(cf$table[2]/(cf$table[1]+cf$table[2])),2)
        miss[i] <- round(100*(cf$table[3]/(cf$table[3]+cf$table[4])),2)
    }

    cutoff.level.accuracy <- which.max(accuracy)
    c.hat.valid <- ifelse(pred>=cutoff.level.accuracy/100, "Spam", "Not Spam")
    cf <- confusionMatrix(c.hat.valid, valid, positive = ref)

    return(list(CutOff = cutoff.level.accuracy / 100,
                AccuracyRate = accuracy[cutoff.level.accuracy],
                FalloutRate = round(100*(cf$table[2]/(cf$table[1]+cf$table[2])),2),
                MissRate = round(100*(cf$table[3]/(cf$table[3]+cf$table[4])),2),
                ConfusionMatrix = table(c.hat.valid,valid),
                FallOutVsAccuracy = cbind(Cutoff = pred.grid, FalloutRate = fallout, AccuracyRate = accuracy,
                                          MissRate = miss),
                AccuracyPlot = plot(pred.grid,accuracy)))
}
```

R Code for Fallout Minimizing Cutoff Function

```
my.function.fallout <- function(pred, valid, ref) {
    require(caret)
    pred.grid <- seq(0.01, 0.99, by = 0.01)
    accuracy <- c()
    fallout <- c()
    miss <- c()

    for (i in 1:length(pred.grid)){
        c.hat.valid <- ifelse(pred>=pred.grid[i], "Spam", "Not Spam")
        cf <- confusionMatrix(c.hat.valid, valid, positive = ref)
        accuracy[i] <-
round(100*((cf$table[1]+cf$table[4])/(cf$table[1]+cf$table[2]+cf$table[3]+cf$table[4])),2)
        fallout[i] <- round(100*(cf$table[2]/(cf$table[1]+cf$table[2])),2)
        miss[i] <- round(100*(cf$table[3]/(cf$table[3]+cf$table[4])),2)
    }

    cutoff.level.fallout <- which.min(fallout)
    c.hat.valid <- ifelse(pred>=cutoff.level.fallout/100, "Spam", "Not Spam")
    cf <- confusionMatrix(c.hat.valid, valid, positive = ref)

    return(list(CutOff = cutoff.level.fallout / 100,
                AccuracyRate = accuracy[cutoff.level.fallout],
                FalloutRate = round(100*(cf$table[2]/(cf$table[1]+cf$table[2])),2),
                MissRate = round(100*(cf$table[3]/(cf$table[3]+cf$table[4])),2),
```

```
                ConfusionMatrix = table(c.hat.valid,valid),
                FallOutVsAccuracy = cbind(Cutoff = pred.grid, FalloutRate = fallout, AccuracyRate = accuracy,
                                          MissRate = miss),
                AccuracyPlot = plot(pred.grid,fallout)))
}
```

R Code for Logistic Regression

```
####################################
### Logistic Regression - Model 1 ###
####################################

# create the model using the training data set
model.log1 <- glm(is_spam ~ . , data.train.std.c, family=binomial("logit"))

# review model summary statistics
coef(model.log1)
summary(model.log1)

# predict outcomes for the validation  set
pred.valid.log1 <- predict(model.log1, data.valid.std.c, type="response") # n.valid post probs

# calculate confusion matrix
my.data.log1.accuracy <- my.function.accuracy(pred.valid.log1,c.valid,ref="Spam")
my.data.log1.fallout <- my.function.fallout(pred.valid.log1,c.valid,ref="Spam")

my.data.log1.accuracy
my.data.log1.fallout
```

R Code for Classification Tree & Corresponding Plots

```
library(tree)
library(ISLR)
library(caret)
library(rpart)
library(rpart.plot)

####################################
### Classification Tree - Model 1 ###
####################################

set.seed(1234)
model.rpartfit <- rpart(is_spam ~ ., data = data.train.std.c)

printcp(model.rpartfit) # display the results
plotcp(model.rpartfit) # visualize cross-validation results
summary(model.rpartfit) # detailed summary of splits

par(mar=c(5,15,4,2)+0.1)
barplot(model.rpartfit$variable.importance[order(model.rpartfit$variable.importance)],
        cex.names = 0.8, horiz = TRUE, cex.axis = 0.8, las=1)

rpart.plot(model.rpartfit, uniform=TRUE, main="", cex = 0.8)

pred.valid.rpart1 <- predict(model.rpartfit, data.valid.std.c, type="class")

# calculate confusion matrix
my.data.rpart1 <- my.function.1(pred.valid.rpart1,c.valid, ref = "Spam")
my.data.rpart1
```

R Code for Lasso Subset Selection

```
library(glmnet)

########################
### Lasso - Model 1 ###
########################

x=model.matrix(is_spam~.,data.train.std.c)[,-215]
y=data.train.std.c$is_spam
grid=10^seq(10,-2,length=100)

model.lasso1 <- glmnet(x, y, alpha=1, lambda=grid, family = "binomial")
plot(model.lasso1)

set.seed(1)
cv.out=cv.glmnet(x, y,alpha=1, family = "binomial")
plot(cv.out)
bestlam=cv.out$lambda.min
x.valid.std.2 <- model.matrix(is_spam~.,data.valid.std.c)[,-215]
lasso.pred=predict(model.lasso1,s=bestlam,newx=x.valid.std.2,type="response")
mean((lasso.pred-c.valid)^2)
out=glmnet(x,y,alpha=1,lambda=grid, family = "binomial")
lasso.coef1=predict(out,type="coefficients",s=bestlam)[1:215,]
lasso.coef1
sort(lasso.coef1[lasso.coef!=0],decreasing = TRUE)
names(lasso.coef1[lasso.coef!=0])

# predict outcomes for the validation  set
post.valid.lasso1 <- predict(model.lasso1,s=bestlam,newx=x.valid.std.2,type="response")

# calculate confusion matrix
my.data.lasso1.accuracy <- my.function.accuracy(post.valid.lasso1, c.valid, ref = "Spam")
my.data.lasso1.fallout <- my.function.fallout(post.valid.lasso1, c.valid, ref = "Spam")

my.data.lasso1.accuracy
my.data.lasso1.fallout
```

R Code for Support Vector Machine with Tuning Parameters

```
#####################
### SVM - Model 2 ###
#####################

model.svm2 <- tune.svm(is_spam ~ ., data = data.train.std.c, gamma = 10^(-6:-1), cost = 10^(1:2))

# review model summary statistics
summary.model.svm2 <- summary(model.svm2)
summary.model.svm2
best.gamma <- summary.model.svm2$best.parameters$gamma
best.cost <- summary.model.svm2$best.parameters$cost

# set the best model (gamme = 0.01; cost = 10)
model.svm2.best <- summary.model.svm2$best.model

# predict outcomes for the validation  set
pred.valid.svm2 <- predict(model.svm2.best, data.valid.std.c, type = "response") # n.valid post probs

# calculate confusion matrix
my.data.svm2 <- my.function.1(pred.valid.svm2,c.valid,ref="Spam")
my.data.svm2
```

R Code for Random Forest and Corresponding Plots

```
##############################
### Random Forest - Model 2 ###
##############################

# create the model using the training data set
set.seed(1)
model.reg.rf2 <- randomForest(is_spam~. , data=data.train.std.c, mtry=6, importance=TRUE)

# review model summary statistics
model.reg.rf2
importance(model.reg.rf2)
varImpPlot(model.reg.rf2)

# predict outcomes for the validation  set
pred.reg.rf2 <- predict(model.reg.rf2,data.valid.std.c, type="response")
my.data.reg.rf2 <- my.function.1(pred.reg.rf2, c.valid, ref = "Spam")
my.data.reg.rf2
```

R Code for Boosted Regression Trees

```
library(gbm)

########################################
### Boosted Regression Tree - Model 1 ###
########################################

data.train.std.c.2 <- data.train.std.c
data.train.std.c.2$is_spam <- ifelse(data.train.std.c.2$is_spam == "Spam", 1, 0)

# create the model using the training data set
set.seed(1)
model.reg.boost1 <- gbm(is_spam~. , data=data.train.std.c.2, n.trees=5000, interaction.depth=4)

# review model summary statistics
model.reg.boost1
summary(model.reg.boost1)
```

```
#plot(model.reg.boost1, i = "rgif")
#plot(model.reg.boost1, i = "lgif")
#plot(model.reg.boost1, i = "agif")

# predict outcomes for the validation  set
pred.reg.boost1 <- predict(model.reg.boost1, newdata=data.valid.std.c, n.trees=5000, type="response")

# calculate confusion matrix
my.data.reg.boost1.accuracy <- my.function.accuracy(pred.reg.boost1, c.valid, ref = "Spam")
my.data.reg.boost1.fallout <- my.function.fallout(pred.reg.boost1, c.valid, ref = "Spam")
```