Data Analysis Workflow

Data Analysis

Data analysis workflow ...

- Define the question
- Define the ideal data set
- Determine what data you can access and obtain the data
- Clean the data
- Exploratory data analysis
- Statistical analysis
- Interpret the results
- Challenge the results
- Synthesize/write up results
- Create reproducible code

Detecting emails that are SPAM from the ones that are not

Image: Freepik.com

Define the question

- Can I use quantitative characteristics of the emails to classify them as SPAM?

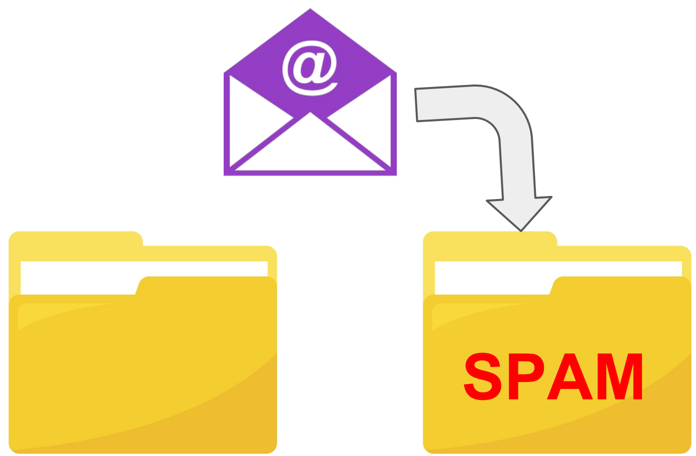


Image: Freepik.com

Define the ideal dataset

spam

From kernlab v0.9-26
by Alexandros Karatzoglou

99th
Percentile

Spam E-Mail Database

A data set collected at Hewlett-Packard Labs, that classifies 4601 e-mails as spam or non-spam. In addition to this class label there are 57 variables indicating the frequency of certain words and characters in the e-mail.

Keywords datasets

Usage

```
data(spam)
```

Details

The data set contains 2788 e-mails classified as `"nonspam"` and 1813 classified as `"spam"`.

The ``spam'' concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography... This collection of spam e-mails came from the collectors' postmaster and individuals who had filed spam. The collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

```
library(kernlab)
data(spam)
```

Clean the data

```
> set.seed(3435)
> trainIndicator = rbinom(4601, size = 1, prob = 0.5)
> table(trainIndicator)
trainIndicator
   0    1
2314 2287
> trainSpam = spam[trainIndicator==1, ]
> testSpam = spam[trainIndicator==0, ]
```

Exploratory data analysis

- Look at summaries of the data
- Check for missing data
- Create exploratory plots
- Perform exploratory analyses

Column names

```
> names(trainSpam)
 [1] "make"             "address"          "all"
 [4] "num3d"            "our"              "over"
 [7] "remove"           "internet"         "order"
[10] "mail"             "receive"          "will"
[13] "people"           "report"           "addresses"
[16] "free"             "business"         "email"
[19] "you"              "credit"           "your"
[22] "font"             "num000"           "money"
[25] "hp"               "hpl"              "george"
[28] "num650"           "lab"              "labs"
[31] "telnet"           "num857"           "data"
[34] "num415"           "num85"            "technology"
[37] "num1999"          "parts"            "pm"
[40] "direct"           "cs"               "meeting"
[43] "original"         "project"          "re"
[46] "edu"              "table"            "conference"
[49] "charSemicolon"    "charRoundbracket" "charSquarebracket"
[52] "charExclamation"  "charDollar"       "charHash"
[55] "capitalAve"       "capitalLong"      "capitalTotal"
[58] "type"
```

First few rows of the training data

```
> head(trainSpam)
   make address  all num3d  our over remove internet order mail receive will
1  0.00    0.64 0.64     0 0.32 0.00   0.00        0  0.00 0.00    0.00 0.64
7  0.00    0.00 0.00     0 1.92 0.00   0.00        0  0.00 0.64    0.96 1.28
9  0.15    0.00 0.46     0 0.61 0.00   0.30        0  0.92 0.76    0.76 0.92
12 0.00    0.00 0.25     0 0.38 0.25   0.25        0  0.00 0.00    0.12 0.12
14 0.00    0.00 0.00     0 0.90 0.00   0.90        0  0.00 0.90    0.90 0.00
16 0.00    0.42 0.42     0 1.27 0.00   0.42        0  0.00 1.27    0.00 0.00
   people report addresses free business email  you credit your font num000
1    0.00      0         0 0.32        0  1.29 1.93   0.00 0.96    0      0
7    0.00      0         0 0.96        0  0.32 3.85   0.00 0.64    0      0
```

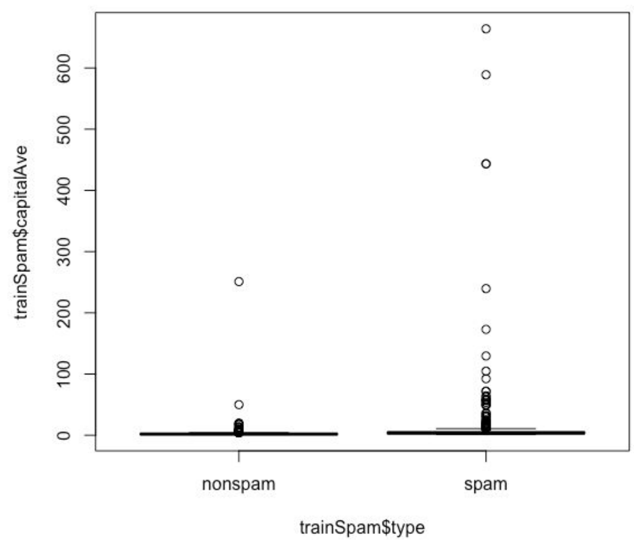
How many of the emails are flagged as SPAM

```
> table(trainSpam$type)

nonspam     spam
   1381      906
```

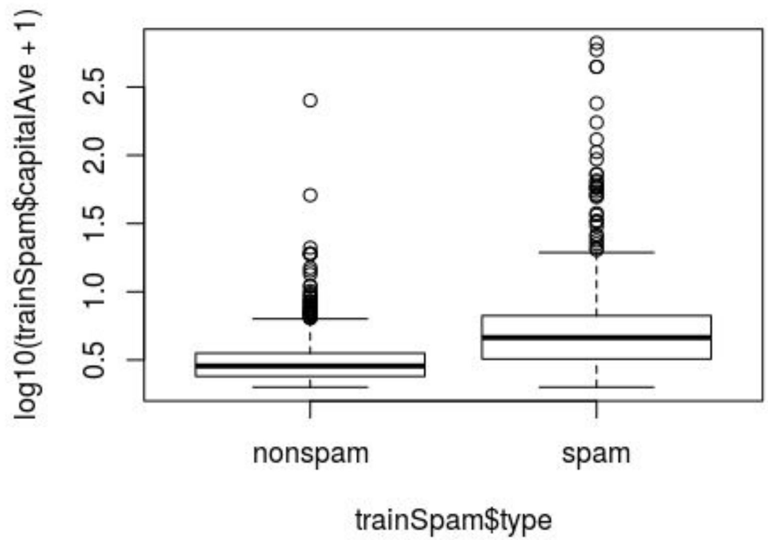
Plot the average length of capital letters in the text of the email for SPAM and non-SPAM emails

```
> plot(trainSpam$capitalAve ~ trainSpam$type)
```



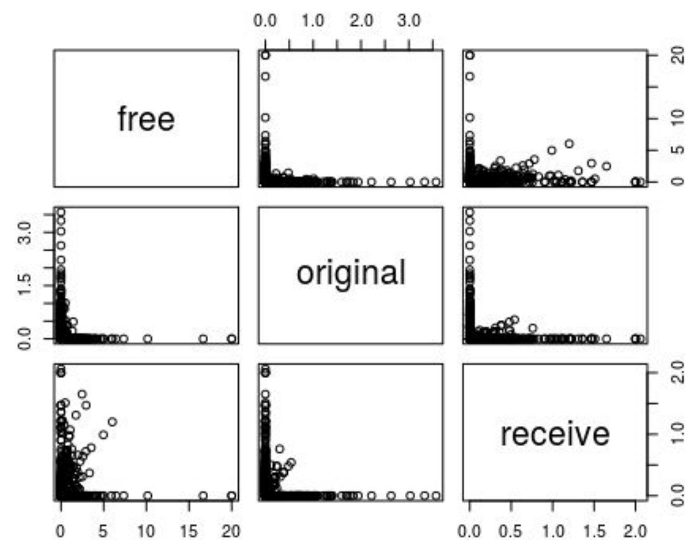
Log transformation

```
> plot(log10(trainSpam$capitalAve + 1) ~ trainSpam$type)
```



Relationship between some of the predictors

```
> library(dplyr)
> trainSpam %>%
+   select(free, original, receive) %>%
+   plot()
```



Statistical analysis

- Exact methods depend on the question of interest
- Transformations/processing should be accounted for when necessary
- Measures of uncertainty should be reported

Prediction

```
> trainSpam$numType = as.numeric(trainSpam$type)-1
> costFunction = function(x,y){sum(x!=(y > 0.5))}
> cvError = rep(NA,55)
> library(boot)
> for(i in 1:55){
+   lmFormula = as.formula(paste("numType~",names(trainSpam)[i],sep=""))
+   glmFit = glm(lmFormula,family="binomial",data=trainSpam)
+   cvError[i] = cv.glm(trainSpam,glmFit,costFunction,2)$delta[2]
+ }
There were 50 or more warnings (use warnings() to see the first 50)
> which.min(cvError)
[1] 53
> names(trainSpam)[which.min(cvError)]
[1] "charDollar"
```

calculates the estimated K-fold cross-validation prediction error

Prediction

```
> predictionModel = glm(numType ~ charDollar,family="binomial",data=trainSpam)
Warning message:
glm.fit: fitted probabilities numerically 0 or 1 occurred
> predictionTest = predict(predictionModel,testSpam)
> predictedSpam = rep("nonspam",dim(testSpam)[1])
> predictedSpam[predictionModel$fitted > 0.5] = "spam"
> table(predictedSpam,testSpam$type)

predictedSpam nonspam spam
      nonspam    1346  458
      spam         61  449
> (61+458)/(1346+458 + 61 + 449)
[1] 0.2242869
```

Number of SPAM emails that were flagged as non-SPAM.

Number of non-SPAM emails that were flagged as SPAM.

Prediction error

Interpret results

- Describes (only if you observe a phenomenon without doing any inferential or predictive analysis)
- Correlates with/associated with (only if you look at the association between variables without any causal interpretation)
- Leads to/causes (only if you have performed causal inference analysis)
- Predicts (only if you have performed predictive analysis)

Make sure you give enough explanation to your analysis!

- Give an explanation as to what your numbers are telling (and not telling)
- If you do regression analysis, interpret the coefficients
- Interpret measures of uncertainty

In our example ...

- The fraction of characters that are dollar signs can be used to predict if an email is Spam
- Anything with more than 6.6% dollar signs is classified as Spam
- More dollar signs always means more Spam under our prediction
- Our test set error rate was 22.4%

Challenge results

- Challenge question
- Challenge data source
- Challenge processing
- Challenge analysis
- Challenge conclusions
- Challenge measures of uncertainty
- Challenge choices of terms to include in models
- Think of potential alternative analyses

Synthesize/write-up results

- Lead with the question
- Summarize the analyses into the story
- Don't include every analysis, include it
    - If it is needed for the story
    - If it is needed to address a challenge
- Order analyses according to the story, rather than chronologically
- Include "pretty" figures that contribute to the story

In our example ...

- Lead with the question
  - Can you use quantitative characteristics of the emails to classify them as SPAM/HAM?
- Describe the approach
  - The source of our SPAM data and how we created training/test sets
  - Explored relationships
  - Choose logistic model on training set by cross validation
  - Applied to test, 78% test set accuracy

In our example … (cont'd)

- Interpret results
  - Number of dollar signs seems reasonable, e.g. "**Make CASH from home $$$$!**"
- Challenge results
  - 78% isn't that great
  - you could use more variables
  - Why logistic regression?

Create reproducible code

Make sure:

- Files are properly named.
- There is some explanation of the data.
- Each code file has some description as to what it does.
- Wherever you should add comments for important code chunks within your code files.