Ideal Steps for Data Analytics

Mr. Sachin B.

Steps in a data analysis

1. Define the question
2. Define the ideal data set
3. Determine what data you can access
4. Obtain the data
5. Clean the data
6. Exploratory data analysis
7. Statistical prediction/modeling
8. Interpret results
9. Challenge results
10. Synthesize/write up results
11. Create reproducible code

1. Define the question

- This is important part of data analysis that is to, **define a question**.
- Not every data analysis starts with the very specific or coherent question.
  - But the, the more effort we can put into coming up with a reasonable question,
  - the less effort we will need to spend having to filter through a lot of stuff.
- The important reason for defining a question is **Dimension reduction**.
  - because if you're interested in, a specific variable, like height or weight, then you can remove, a lot of other variables that don't really pertain to that at all.
- The idea is, if we can narrow down our question as specifically as possible.
  - Then that will serve to reduce the kind of noise that we have to deal with when you're going through a potentially very large data set.
- It is important to think about what type of question you're interested in answering
  - before, delving into all the details of your data set.
- If we randomly apply statistical methods to data sets to find an interesting answer.
  - We will find something interesting almost certainly, but it may not be reproducible and it may not be really meaningful.
- A proper data analysis has a scientific context, it has at least some general question that we're trying to investigate.
  - It will narrow down the dimensionality of the problem.
  - And then we'll apply the appropriate statistical methods to the appropriate data.
- **Start with a general question**

  - Can I automatically detect emails that are SPAM or not?

- **Make it concrete**
  - Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?

2. Define the ideal data set

The data set may depend on your goal or a Question

- Descriptive - a whole population
- Exploratory - a random sample with many variables measured
- Inferential - the right population, randomly sampled
- Predictive - a training and test data set from the same population
- Causal - data from a randomized study
- Mechanistic - data about all components of the system

2.1 Descriptive

- If we are interested in a descriptive problem, you might think of a whole population.
- Here we don't need to sample anything.
- We might want to get the entire census or population that we are looking for.
- **Example: All the emails in the universe**

2.2 Exploratory

- If we want to explore our question. we might just take a random sample with a bunch of variables measured.

2.3 Inferential

- If we want to make inference about a problem then we have to be very careful about
  1. the sampling mechanism and
  2. the definition of the population that you're sampling from
- Because typically when we make an inferential statement,
  - we are drawing from a sample to make a conclusion about a larger population.
  - So there the sampling mechanism is very important.

2.4 Prediction

- If we want to make a prediction, then you're going to need a **training set and a test data set** from a population that you're interested in, so that you can build a **model and a classifier**.

2.5 Causal

- If we want to make a causal statement, we want to see, if we modify certain component, then something else happens.
- So basically, we're going to need experimental data, and one type of experimental data like a randomized trial or a randomized study.

2.6 Mechanistic

- If we want to make mechanistic types of statements, we need data about all the different components of the system that you're trying to describe.

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

1. If we use Gmail that holds all the emails in the Gmail system stored on Google's data centers.
    - because that would be a whole population of emails, and
    - then we can just kind of build our classifier based on all this data, and
    - then we wouldn't have to worry about sampling, because we'd have all the data.
    - so **that would be an ideal data set**.

3. Determine what data you can access

1. Sometimes you can find data free on the web
2. Other times you may need to buy the data (Be sure to respect the terms of use)
3. If the data don't exist, you may need to generate it yourself

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

- In the real world, we have to think about, what are the data that you can actually access.
    - Maybe someone at Google actually, can access all the emails that go through Gmail. But, but even in that extreme case, it may be difficult.
    - And furthermore, most people are not going to be able to access that.
- So, sometimes we have to go for something that's not quite the ideal data set.
    1. So, We might be able to find free data on the web.
    2. We might need to buy some data from a provider - In these kinds of cases, we should be sure to respect the terms of use for the data.
    3. If the data simply do not exist out there, We may need to generate the data ourself in some way.

- Getting all the data from Google will probably not be possible
- So one possible solution is, comes from the UCI machine on your repository, which is the spam based data set.
    - And this is a collection of spam that was created by people at Hewlett Packard who collected some, a couple thousand spam messages. Spam and regular messages, and classified them appropriately.
- So we can use this database to explore your problem of how to classify emails into spam or ham.

4. Obtain the data

- Try to obtain the raw data
- Be sure to reference the source
- Polite emails to investigator
- Data from an internet source, record the url and time accessed

4.1 Try to obtain the raw data

- When we obtain the data, the first goal is to try to obtain the raw data.
- For example, from the UCI machine on your repository.

4.2 Be sure to reference the source

- Always reference the source
  - so wherever we get the data from, we should always reference the source and keep track of where it came from.

4.3 Polite emails to investigator

- If we need to get data from a person or an investigator that we're not familiar with often a very polite email will go a long way.
- They may be willing to share that data with us.

4.4 Data from an internet source

- If we get data from an internet source,
  - Record the URL which is the website indicator of where we got the data and the time and date that you access that.
  - So people have a reference, when that data was available.
- In the future, the website might go down or the URL may change or may not be available,
  - At that time we got that data we documented how we got it.

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

- Since we don't have access to Google's data centers, It is the `Spam E-mail Database` which we can get from the `Kern Lab package` in R.
- So it comes with the Kern Lab package and so if we install the Kern Lab package we can load the data set right away.

http://search.r-project.org/library/kernlab/html/spam.html

```
# If it isn't installed, install the kernlab package with install.packages()

 install.packages("kernlab")
```

```
library(kernlab)
data(spam)

str(spam[,1:5])
```

```
## 'data.frame':    4601 obs. of  5 variables:
##  $ make   : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ address: num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all    : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ num3d  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our    : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
```

5. Clean the data

- Raw data often needs to be processed
- If it is pre-processed, make sure you understand how
- Understand the source of the data (census, sample, convenience sample, etc.)
- May need reformating, subsampling - record these steps
- Determine if the data are good enough - if not, quit or change data

5.1 Raw data often needs to be processed

- Any data set needs a little bit of Cleaning.
- Often raw data we typically need to be processed in some way to get it into a form where you can model it or feed it into a modeling program

5.2 If it is pre-processed, make sure you understand how

- If it's already pre-processed, it's important that you understand how it was.
- Try to get some documentation about what the pre-processing was and how it was done.

5.3 Understand the source of the data (census, sample, convenience sample, etc.)

- We have to understand from where the data come,
- for example, if it came from a survey, you need to know how the sampling was done.
  - was it a convenient sample,
  - did the data come from an observational study,
  - did it come from experiments the source of the data is very important.

5.4 May need reformating, subsampling - record these steps

- We may need to reformat the data in a certain way to get it to work in a certain type of analysis.
- If the data set is extremely large you may want to sub-sample the data set to make it more manageable.
- And anything we do to clean the data, it is very important that we record these steps and write them down in scripts or whatever is most convenient.
- Because we or someone else is going to reproduce these steps if they want to reproduce your findings.
  - And if we don't document all these pre-processing steps,
  - then no one will ever be able to do it again.

5.5 Determine if the data are good enough - if not, quit or change data

- Once we've cleaned the data and we've gotten a basic look at it, it's important to determine of the data are good enough to solve your problems because in some cases they may not be good enough.
  - we may not have enough data,
  - We may not have enough variables or enough characteristics,
  - the sampling of the data may be inappropriate for your question.
- If you determined the data are not good enough for your question, then
  - you've got to quit or
  - try again, or

  - change the data, or
  - try a different question.

- **It's important to not to just push on with the data you have just because that's all you've got, because that can lead inappropriate inferences or conclusions**.

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

I. Our cleaned data set

- Our data set is from UCI machine Learning Repository which had already been cleaned up and it was available in the current lab package as a data set.
- So this data set has 4,600 observations or emails that had been kind of characterized along 58 different variables.

http://search.r-project.org/library/kernlab/html/spam.html

```
library(kernlab)
data(spam)

str(spam[,1:5])
```

```
## 'data.frame':    4601 obs. of  5 variables:
##  $ make   : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
##  $ address: num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ all    : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
##  $ num3d  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our    : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
```

```
dim(spam)
```

```
## [1] 4601   58
```

II. Subsampling our data set

- First thing that we need to do with this data set if we want to build a model to classify emails into spam or not is that we need to **split the data set into test set and a training set**.
  - So the idea is that we're going to use part of the test of the data set to build our model, and then
  - we're going to use another part of the data set which is independent of the first part to actually determine how good our model is making a prediction.

- So here we're taking a random half of the data set, we're flipping a coin with the rbinom function, to generate a random coin flip with probability of half so that will separate the the data set into two pieces.

```
# Perform the subsampling
set.seed(3435)

trainIndicator = rbinom(n = 4601,size = 1, prob = 0.5)
table(trainIndicator)
```

```
## trainIndicator
##    0    1
## 2314 2287
```

- So we can see that 2314 are going to be one half and 2287 will be in the other half.
- And so the training set will be one set and the test set will be another set of data.

```
trainSpam <- spam[trainIndicator == 1, ]
testSpam <- spam[trainIndicator == 0, ]

# Checking first 5 Variables
str(trainSpam[,1:5])
```

```
## 'data.frame':    2287 obs. of  5 variables:
##  $ make   : num  0 0 0.15 0 0 0 0 0 0 0 ...
##  $ address: num  0.64 0 0 0 0 0.42 0 0 0 0 ...
##  $ all    : num  0.64 0 0.46 0.25 0 0.42 0 0.55 0 0 ...
##  $ num3d  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our    : num  0.32 1.92 0.61 0.38 0.9 1.27 0 1.11 2.94 1.16 ...
```

```
str(testSpam[,1:5])
```

```
## 'data.frame':    2314 obs. of  5 variables:
##  $ make   : num  0.21 0.06 0 0 0 0 0.06 0 0 0 ...
##  $ address: num  0.28 0 0 0 0 0 0.12 0 0.69 0 ...
##  $ all    : num  0.5 0.71 0 0 0 0 0.77 0 0.34 1.42 ...
##  $ num3d  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ our    : num  0.14 1.23 0.63 0.63 1.85 1.88 0.19 0 0.34 0.71 ...
```

6. Exploratory data analysis

- Look at summaries of the data
- Check for missing data
- Create exploratory plots
- Perform exploratory analyses (e.g. clustering)

6.1 Look at summaries of the data

- It would be useful to look at kind of
  - what did the data look like,
  - what's the distribution of the data,
  - what are the relationships between the variables
- So we want to look at basic summaries
  - one dimensional,
  - two dimensional summaries of the data.

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

```
names(trainSpam)
```

If we look at the column names of the training data sets,

```
##  [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"
```

- you can see that they're all words, technical parameters or statistics of words which frequently occurs in email content.

```
head(trainSpam)
```

If you look at the first few rows,

```
##    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
## 9    0.00      0         0 0.00        0  0.15 1.23   3.53 2.00    0      0
## 12   0.12      0         0 0.00        0  0.00 1.16   0.00 0.77    0      0
## 14   0.90      0         0 0.00        0  0.00 2.72   0.00 0.90    0      0
## 16   0.00      0         0 1.27        0  0.00 1.70   0.42 1.27    0      0
##    money hp hpl george num650 lab labs telnet num857 data num415 num85
## 1   0.00  0   0      0       0   0    0      0       0 0.00      0     0
```

```
## 7   0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 9   0.15  0   0      0      0   0    0      0      0 0.15      0     0
## 12  0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 14  0.00  0   0      0      0   0    0      0      0 0.00      0     0
## 16  0.42  0   0      0      0   0    0      0      0 0.00      0     0
##    technology num1999 parts pm direct cs meeting original project re edu table
## 1           0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 7           0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 9           0    0.00     0  0   0.00  0       0      0.3       0  0   0     0
## 12          0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 14          0    0.00     0  0   0.00  0       0      0.0       0  0   0     0
## 16          0    1.27     0  0   0.42  0       0      0.0       0  0   0     0
##    conference charSemicolon charRoundbracket charSquarebracket charExclamation
## 1           0         0.000            0.000                 0           0.778
## 7           0         0.000            0.054                 0           0.164
## 9           0         0.000            0.271                 0           0.181
## 12          0         0.022            0.044                 0           0.663
## 14          0         0.000            0.000                 0           0.000
## 16          0         0.000            0.063                 0           0.572
##    charDollar charHash capitalAve capitalLong capitalTotal type
## 1       0.000    0.000      3.756          61          278 spam
## 7       0.054    0.000      1.671           4          112 spam
## 9       0.203    0.022      9.744         445         1257 spam
## 12      0.000    0.000      1.243          11          184 spam
## 14      0.000    0.000      2.083           7           25 spam
## 16      0.063    0.000      5.659          55          249 spam
```

- we can see that basically that these are the frequencies at which they occur in a given email.
- So we can see the word make does not appear in that first email and the word mail does not appear, so things like that.
- So these are all basically frequency counts, or frequencies of words within each of the emails.

```
table(trainSpam$type)
```

Summaries: If we look at the outcome of training data set

```
##
## nonspam    spam
##    1381     906
```

- we see that 906 of the emails are spam, are classified as spam.
- And the other 1381 are classified as non-spam.
- So these, this is what we're going to use to kind of build our model for predicting the spam emails.

6.2 Check for missing data

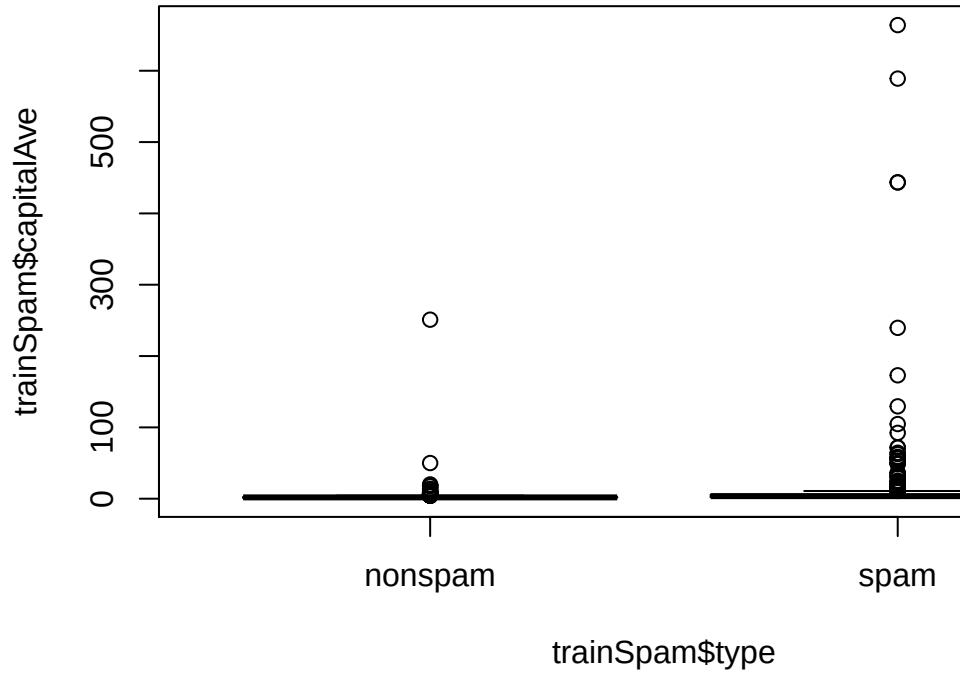
- we need to check for, is there are any missing data,

6.3 Create exploratory plots

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

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

We can make some plots to compare what are the frequencies of certain characteristics between

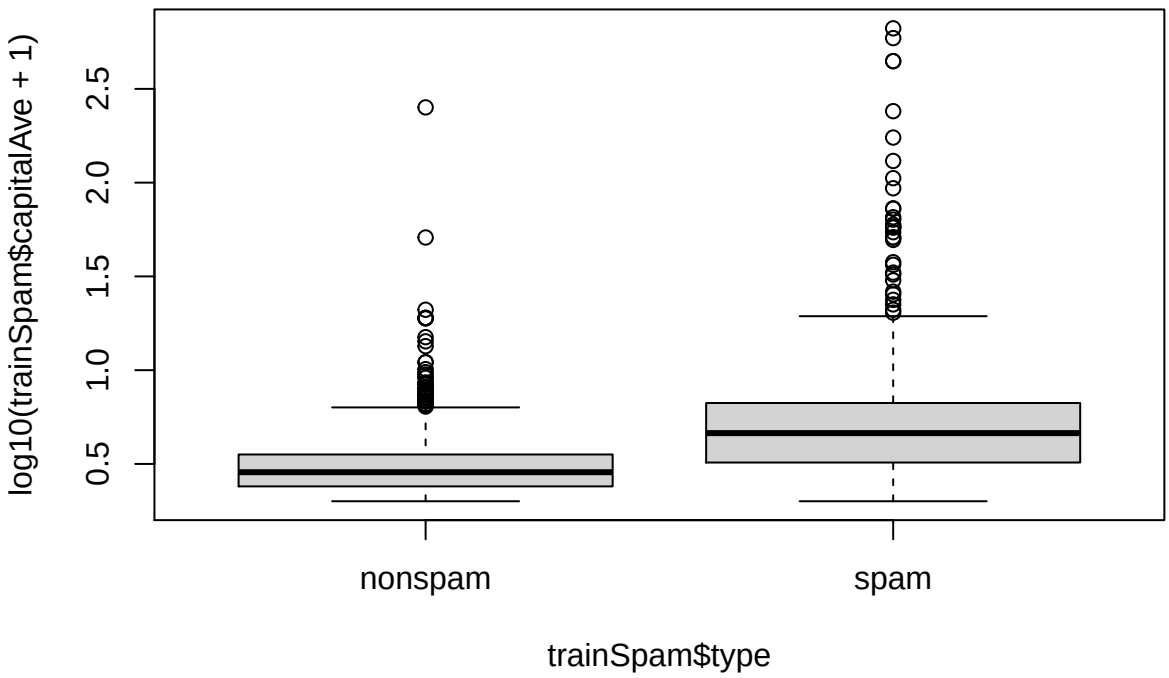


the spam and the non spam emails.

- Here we're looking at a variable called 'capital ave' i.e. the average number of capital letters.
- We can see that its difficult to look at this picture, because the data are highly skewed.
- In these kinds of situations it's often useful to just kind of look at the log transformation of the variable.

- We are going to to take the base ten log of the variable, and compare them to spam and nonspam.
- Since there are a lot of zeros in this particular variable, taking the log of zero doesn't really make sense.
- So we'll just add 1 to that variable so we can take the log and kind of get a rough sense of what the data look like.
- Typically, we wouldn't want to just add 1 to a variable but since we're just exploring the data by making exploratory plots, it's okay to do that in this case.

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

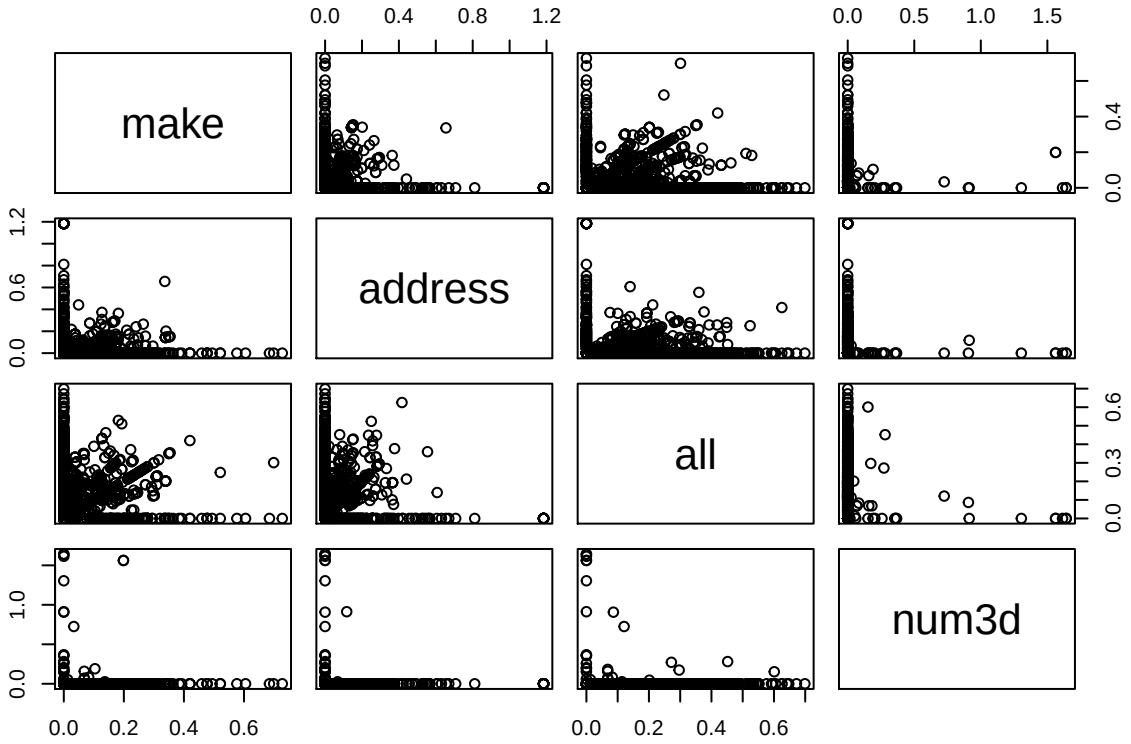


- So here you can see that, **the spam emails have a much higher rate of capital letters than the non spam emails**.
- If you've ever seen spam emails, you're probably familiar with that phenomenon.
- And so that's one useful relationship to see there.

Relationship between Predictors

- We can look at pairwise relationships between the different variables in the plots.

```
plot(log10(trainSpam[ ,1:4] +1))
```



- We've got a pairs plot of a few of the variables
- As this is the log transformation of each of the variables we can see that
  - some of them are correlated,
  - some of them are not particularly correlated, and so that's useful to know.

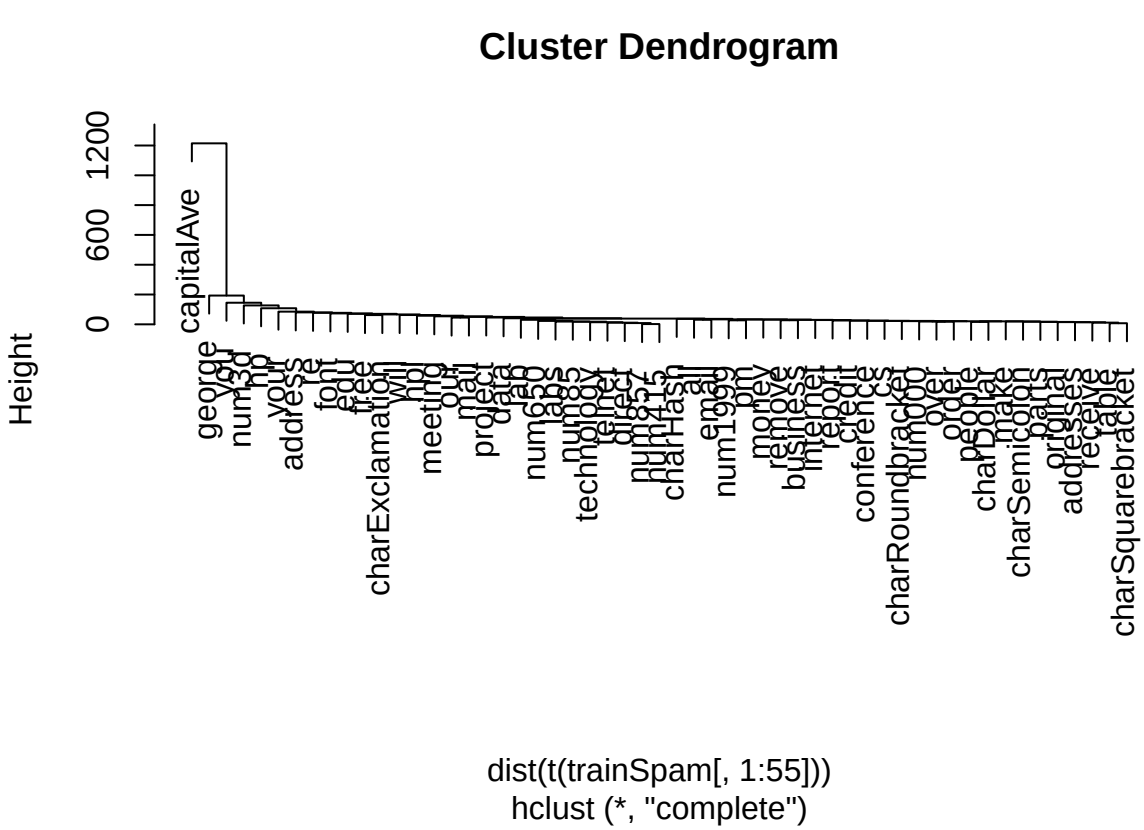
6.4 Perform exploratory analysis (e.g. clustering)

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

- We can explore the predictors space a little bit more by doing a hierarchical cluster analysis
- So this is a first cut at trying to do that with the hclust function in R.

```
hCluster <- hclust(dist(t(trainSpam[,1:55])))

plot(hCluster)
```

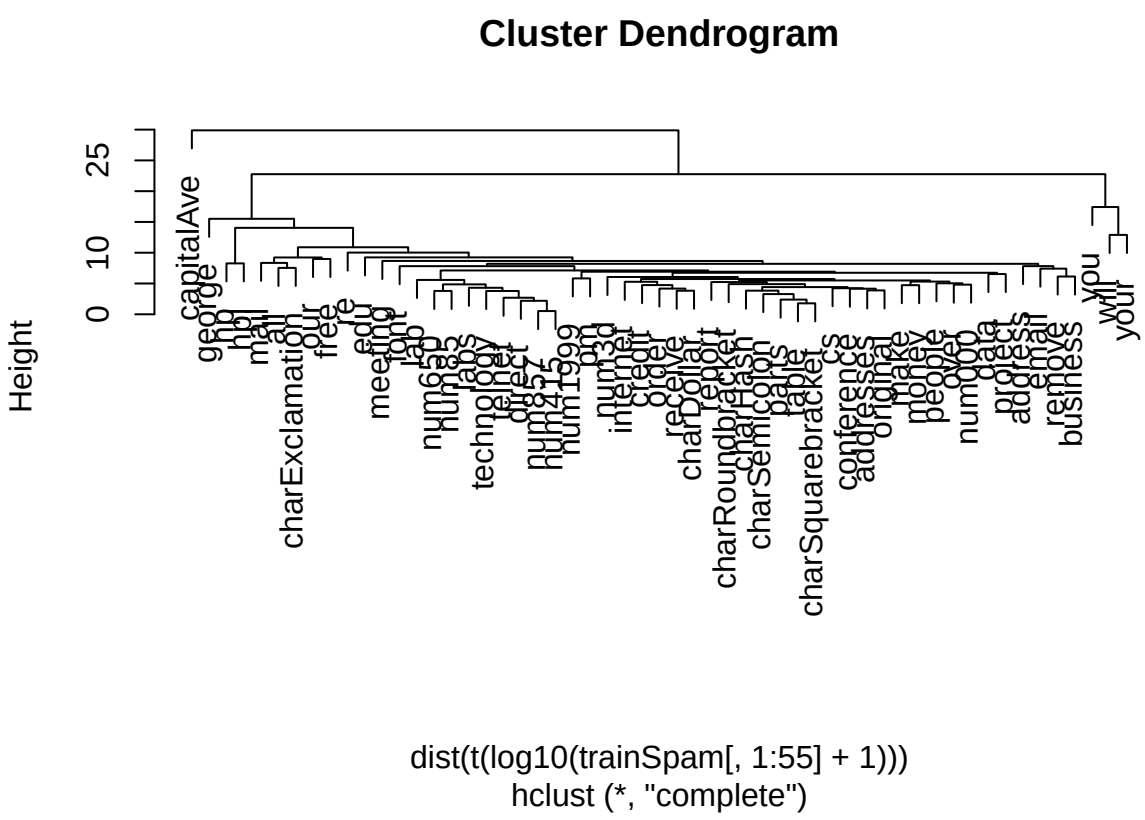


- We can see the Dendrogram just to see, what predictors or what words or characteristics tend to cluster together.
- And it's not particularly helpful at this point although it does separate out this one variable capital total.
- But if we recall that the clustering algorithms can be sensitive to any skewness in the distribution of the individual variables.
- So it may be useful to redo the clustering analysis after a transformation of the predictor space.

- So here we will take a 'log to the base 10' log transformation of the predictors in the training data set, and
  - again, we are going to add one to each one, just to avoid taking the log of zero.

```
hCluster <- hclust(dist(t(log10(trainSpam[,1:55] + 1))))

plot(hCluster)
```



- We can see it's a little bit more interesting, the dendrogram is separated out a few clusters
  - 'Capital average' is one kind of cluster all by itself.
  - There's another cluster that includes 'you will' or 'your'.
  - And then there are a bunch of other words that kind of lump more ambiguously together.
- This may be something worth exploring a little bit further if we see some particular kind of characteristics that are interesting.

7. Statistical prediction/modeling

- Statistical modeling Should be informed by the results of exploratory analysis
- Exact methods depend on the question of interest
- Transformations/processing should be accounted for when necessary
- Measures of uncertainty should be reported

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

- So here we're going to just do a very basic statistical model.
- We're going to go through each of the variables in the data set and try to fit a generalizing model,
  - in this case a **logistic regression**,
  - to see if we can predict an email is spam or not by using just a single variable.

1. So here using the reformulate function to create a formula that includes

i. *the response*, which is just the 'type of email' and
ii. *one of the variables* of the data set, and

2. We're just going to cycle through all the variables in this data set using this for-loop to build a logistic regression model.
3. Then subsequently calculate the cross validated error rate of predicting spam emails from a single variable.
4. Once we've done this, we're going to try to figure out, which of the individual variables has the minimum cross validated error rate.
5. So we can take this vector of CV error, and just figure out which one is the minimum.

```
trainSpam$numType <- as.numeric(trainSpam$type) - 1

trainSpam$type[901:910]
```

```
##  [1] spam    spam    spam    spam    spam    spam    nonspam nonspam nonspam
## [10] nonspam
## Levels: nonspam spam
```

```
trainSpam$numType[901:910]
```

```
##  [1] 1 1 1 1 1 1 0 0 0 0
```

```
############################################################

costFunction <- function(x,y)
                {
                  sum(x != (y > 0.5))

                }

cvError <- rep(NA, 55)

############################################################

library(boot)

for(i in 1:55)
{
  lmFormula <- reformulate(names(trainSpam)[i], response = "numType")
  glmFit <- glm(lmFormula, family = "binomial", data = trainSpam)
  cvError[i] <- cv.glm(trainSpam, glmFit, costFunction, 2)$delta[2]
}

############################################################

## Which predictor has minimum cross-validated error?

names(trainSpam)[which.min(cvError)]
```

```
## [1] "charDollar"
```

- It turns out that the predictor that has the minimum cross validated error rate is the variable called `charDollar`.
- This is an indicator of the number of dollar signs in the email.

Get measure of uncertainty

- If we take this best model (charDollar variable) from this set of 55 predictors and just re-fit the model. This is a logistic regression model.
- We can actually make predictions now from the model on the test data - recall that we split the data set into two parts and built the training model on the training data set.

- In a logistic regression we don't get specific predictions out of 0 1 classifications of each of the messages
- We get a probability that a message is going to be spam or not.
- Then we have to take this continuous probability, which ranges between 0 and 1, and determine at what point, at what cutoff, do we think that the email is spam.
- And so we're, going to draw the cut off here at 0.5,
  - so if the probability is above 50%, we're just going to call it a spam email.

```
### Use the best model from the group
predictionModel = glm(numType ~ charDollar, family = "binomial", data = trainSpam)

### Get predictions on the test set

predictionTest = predict(predictionModel, testSpam)

predictedSpam = rep("nonspam", dim(testSpam)[1])

## Classify as 'spam' for those with prob > 0.5

predictedSpam[predictionModel$fitted > 0.5] = "spam"
```

- So once we've created our classification, we can take a look at the predicted values from our model and then compare them with the actual values from the test data set, because we know which was spam, and which was not.

```
## Classification Table

table(predictedSpam,testSpam$type)
```

```
##
## predictedSpam nonspam spam
##       nonspam    1346  458
##       spam         61  449
```

- we can calculate the error rate.
- The mistakes that we made are on the off diagonal elements of this table, 61 and 458.
  - 61 were classified as spam, that were not actually spam, and
  - 458 were classify as non spam but actually were spam.
- we can calculate the error rate.

```
###  Error Rate
(61 + 458)/(1346 + 458 + 61 + 449)
```

```
## [1] 0.2242869
```

- Error rate is about 22%.

8. Interpret results

- Use the appropriate language
  - describes
  - correlates with/associated with
  - leads to/causes
  - predicts
- Give an explanation
- Interpret coefficients
- Interpret measures of uncertainty

8.1 Use the appropriate language

- It's important when we interpret our findings to use appropriate language.
- And to not be to not use language that goes beyond the analysis that you actually did.
- If you're in this type of application where we're just looking at some data, we're building a predictive model.
- You want to use words like
  - prediction or
  - it correlates with or
  - certain variables may be associated with the outcome or
  - the analysis is descriptive,
- Think carefully what kind of language you use to interpret your results.

8.2 Give an explanation

- It's good to give an explanation,
- If you can think of, why certain models predict better than others, it would be useful to kind of give an explanation of what you think that is.

8.3 Interpret coefficients

- If there are coefficients in the model that you need to interpret it's useful, you can do that here.

8.4 Interpret measures of uncertainty

- It's useful to bring in measures of uncertainty, to kind of calibrate your interpretation of the final results.

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

- 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%

9. Challenge results

- Challenge all steps
  - Question
  - Data source
  - Processing
  - Analysis
  - Conclusions
- Challenge measures of uncertainty
- Challenge choices of terms to include in models
- Think of potential alternative analyses

- It's important that you challenge all the results that you've found.
  - Because if you don't do it, someone else is going to do it once they see your analysis, and
  - Be one step ahead of everyone by doing it yourself first.

9.1 Challenge all steps

- It's good to challenge everything, the whole process by which you gone through this problem.
- The question itself is that, is the question even a valid question to ask
  - where the data came from,
  - how you got the data,
  - how you processed the data,
  - how you did the analysis and
  - any conclusions that you drew.

9.2 Challenge measures of uncertainty

- If you have measures of uncertainty, are those the appropriate measure of uncertainty.

9.3 Challenge choices of terms to include in models

- If you built models,
  - Why is your model the best model?
  - Why is it an appropriate model for this problem?
  - How do you choose the things to include in your model?

9.4 Think of potential alternative analysis

- It's useful to think potential alternative analysis that might be useful
  - It doesn't mean that you have to do those alternative analysis, in the sense that you might stick to your original.
  - But it may be useful to try alternative analysis just in case they may be useful in different ways or may produce better predictions.

10. 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

- Synthesis is very important because typically in any data analysis, there are going to be many things that you did.
- When you present them to a another person or to a group, you're going to have to winnow it down to the kind of most important aspects and to tell a coherent story.

10.1 Lead with the question

- Typically you want to lead with the question that you were trying to address.
- If people understand the question then they can draw up a context in their mind and have a better understanding of the framework in which you're operating.
- That will lead to what kinds of data are appropriate for this question or what kinds of analyses would be appropriate.

10.2 Summarize the analyses into the story

- you can summarize the analyses as you're telling the story.
- It's important that you don't include every analysis that you ever did but
  - only if its needed for kind of telling a coherent story.
  - It's useful to sometimes keep these analyses in your back pocket though, even if you don't talk about it, because someone may challenge what you've done and it's useful to say, "we did do that analysis but, it was problematic because of whatever the reason"

10.3 Order analyses according to the story, rather than chronologically

- It's important to order the analysis that you did according to the story that you're telling and often that order is not the same as the order in which you actually did the analysis.
- It's usually not that useful to talk about the analysis that you did kind of chronologically, or the order in which you did them, because the order in which you did them is often very scattered and doesn't make sense in retrospect.

10.4 Include "pretty" figures that contribute to the story

- When you are telling the story or you're presenting to someone or to group, it's useful to include very well done figures so that people can understand what you're trying to say in one picture or two.

Example: For, Defining the ideal data set for classifying SPAM/HAM Email

- Lead with the question
  - Can I use quantitative characteristics of the emails to classify them as SPAM/HAM?
- Describe the approach
  - Collected data from UCI -> created training/test sets
  - Explored relationships
  - Choose logistic model on training set by cross validation
  - Applied to test, 78% test set accuracy
- Interpret results
  - Number of dollar signs seems reasonable, e.g. "Make easy Money `$$$$` !"
- Challenge results
  - 78% isn't that great
  - I could use more variables
  - Why logistic regression?

11. Create reproducible code

- Make sure that you document your analysis as you go.
- You can use tools like `R Markdown` and `knitr` and `R Studio` to document your analyses as you do them.
- You can preserve the R code as well as any kind of a written summary of your analysis in a single document using Knitr.
- Make sure that all of what you do is reproducible by either yourself or by other people because ultimately that's the standard by which most of, big data analysis will be judged.
- If someone can not reproduce it then the conclusions that draw will be considered as not worthy as one analysis where the results are reproducible will consider.
- So try to stay organized.
- Try to use the tools reproducible research to keep things organized and reproducible that will make your evidence for your conclusions much more powerful.