MET CS688 C1

Web Analytics and Mining

Zlatko Vasilkoski

Statistical Methods With R

Data Samples and Precision

Online polls vs standard polls

Which is better? Large number of people or a smaller subset?

- In 1936, Literary Digest sent ballots to 10 million (¼) of the electorate regarding the U.S. presidency. 2.4 million people replied. Their conclusion was that Alfred Landon will win against Franklin Roosevelt with a large margin of 55% vs 41%. Roosevelt won by 67% vs 31%.
- George Gallup, founder of Gallup, Inc. (1935), on a much smaller sample correctly predicted that Franklin Roosevelt will win, making Gallup, Inc. a leader in American polling.
- Neglecting sampling bias (names from a phone book and car registration) in favor of size. Are they representative sample? Gallup carefully selected a representative sample.
- Very, very relevant to online sampling, larger data not always better.
- 1998 US university poll - the most influential person in the last millennium – Jamie Pollock - played for Manchester City, a team which then fell into what was then Division Two. Pollock scored a bizarre own goal over his own goalkeeper. The own goal condemned Manchester City to relegation to the third tier for the first time, whilst keeping the other team in the division. As a result, a group of QPR fans thanked him by voting him the "most influential man of the past 2,000 years" in an internet poll, where "Jesus and Marx came second, and third.“

Sources: http://www.bbc.co.uk/programmes/p049k2dj

Statistical Methods with R

Classifying SMS messages

- This is a document classification project where you are trying to classify spam vs ham SMS messages.
- Dataset Origin: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
- Explore the dataset.
- Classify SMS messages:
  1. Use your Newsgroups classification algorithm
     - Calculate effectiveness of your classification (CM & F Score)
  2. Use different algorithms from machine learning package "caret“ to classify the SMS messages
     - Use knn algorithm
     - Use SVM (Support Vector Machine) algorithm
     - Decision Tree algorithm
     - Compare effectiveness of these 3 classifications (CM & F Score)
  3. Use your own classification algorithm
     - Calculate effectiveness of your classification (CM & F Score)

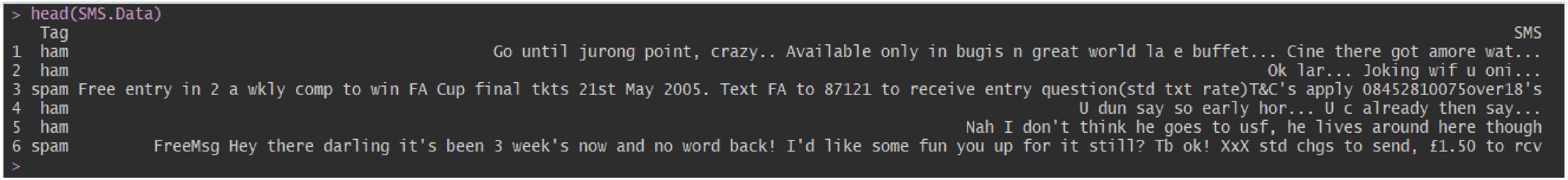
Classifying SMS messages – Lab Project

This is a document classification project where you are trying to classify spam vs ham SMS messages.

- Dataset Origin: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
    - Note: The data is included in the attached project.
- Explore the dataset.
- Classify SMS messages using KNN Classification package "class" :
    1. Use your Newsgroups classification algorithm
        - Use provided code on Blackboard
            - Note: The code loads saved dtm
            - Sub sets only the first 1000 SMS messages
        - Use your own code to calculate effectiveness of your classification (CM & F Score)
            - Note: It may take up to 10 min for the knn function to perform the classification.
- Submit your code as zipped project to Blackboard.

Explore the dataset.

- The data is already tagged (classified) as “Spam” or “Ham”. Note the 2-Column Structure. The first “Tab” separated column is the Tag, and the second is the SMS content.
  - Tag: with values “Ham” & “Spam”.
  - SMS: Containing the SMS text.
- Notice the difference in the text content of “Ham” vs. “Spam”.
  - Some of the SMS content contains words from other languages, phone numbers, URL’s, characters and symbols.
  - The English words and sentences used in these SMS often contain slang.
- Use your observations to design your Classification Algorithm.



```
> head(SMS.Data)
  Tag                                                                                                                                                          SMS
1  ham                                             Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2  ham                                                                                                                               Ok lar... Joking wif u oni...
3 spam Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4  ham                                                                                                           U dun say so early hor... U c already then say...
5  ham                                                                                               Nah I don't think he goes to usf, he lives around here though
6 spam         FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
>
```

Algorithm Outline

To perform the Classification, use R's "tm" package for Text Mining, and perform the following steps:

1. Load SMS data, create corpus, implement pre-processing and create the document term matrix
    - Create Document-Term Matrix (dtm) with
        - `dtm <- DocumentTermMatrix(SMS.Corpus.PP, control=list(wordLengths=c(1,Inf), bounds=list(global=c(1,Inf))) )`
    - Create tf-idf weighted version of term document matrix that takes into account how often a term is used in the entire corpus as well as in a single document. The logic is that if a term is used in the entire corpus frequently, it is probably not as important when differentiating documents are compared.
        - `dtm.weight <- DocumentTermMatrix(SMS.Corpus.PP, control = list( wordLengths=c(1,Inf), bounds=list(global=c(1,Inf)), weighting = weightTfIdf))`
2. Split the dtm into 60% for Training and 40% for Testing
3. Implement different classification algorithms.
4. Calculate effectiveness and compare them for each of these classification algorithms:
    1. knn algorithm package "class"
    2. Use package "caret" with 3 times repeated 10-fold cross-validation
        - knn algorithm
        - SVM algorithm
        - Decision Tree algorithm
    3. Create your own classification algorithm based on Zipf's sentiment and Bayesian Classification.

Methods Used to Improve Classification & Regression

- Boosting (No replacement) Sequential
  - Draw random subset d1 from training set D and train C1.
  - Draw second subset d2 from training set D and add 50% of previously falsely classified to train C2.
  - Find d3 in D on which C1 & C2 disagree to train C3.
  - Combine all C’s by majority voting.

- Bagging & Bootstrapping (replacement) Parallel
  - Choose random sample with replacement
  - Generate n different bootstrap training samples
  - Train algorithms on each separately.
  - Average predictions at the end.

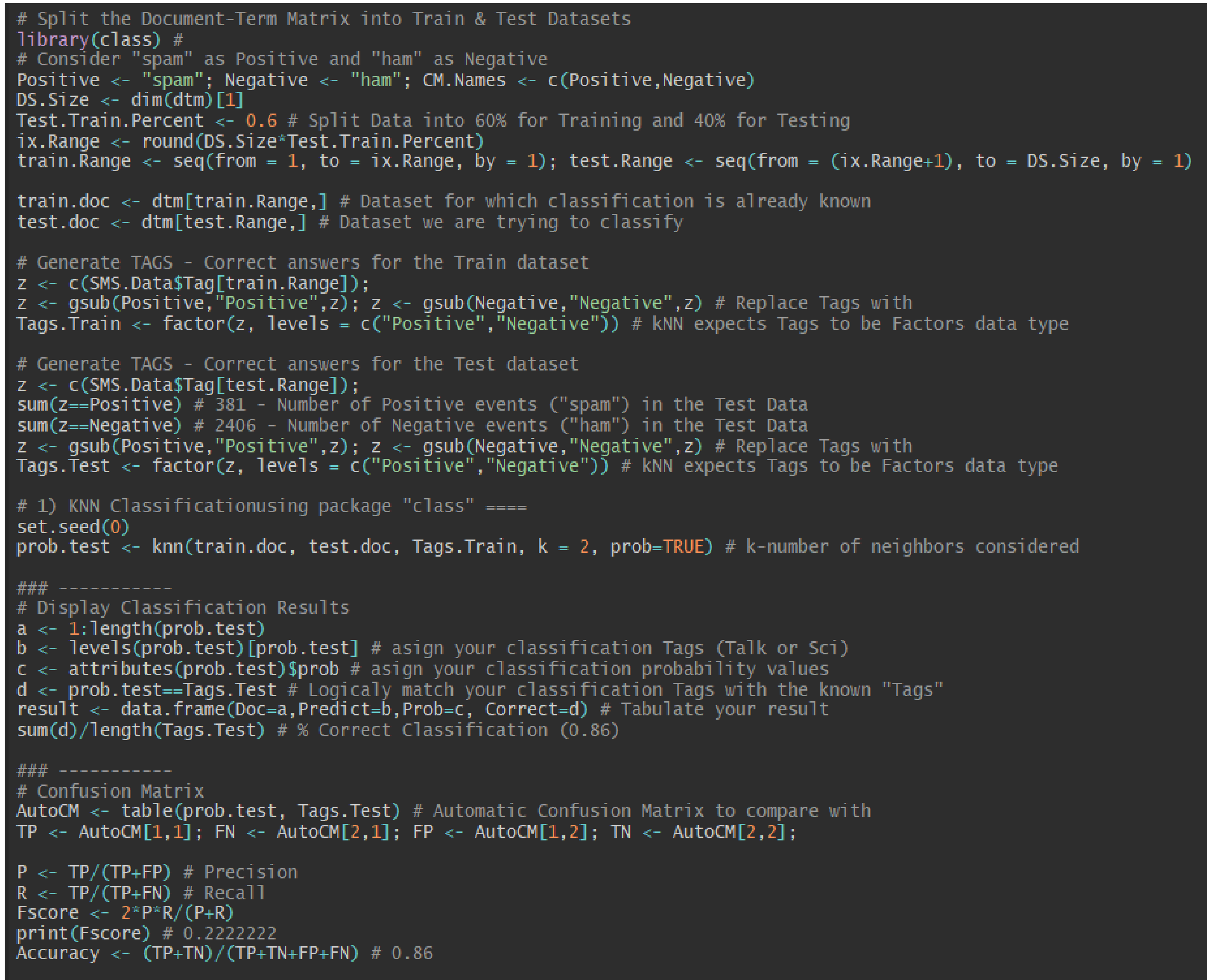
- Stacking is a similar to boosting but uses another model/approach

Split DTM into Train & Test Datasets

Split the dtm into

- 60% for Training
- 40% for Testing

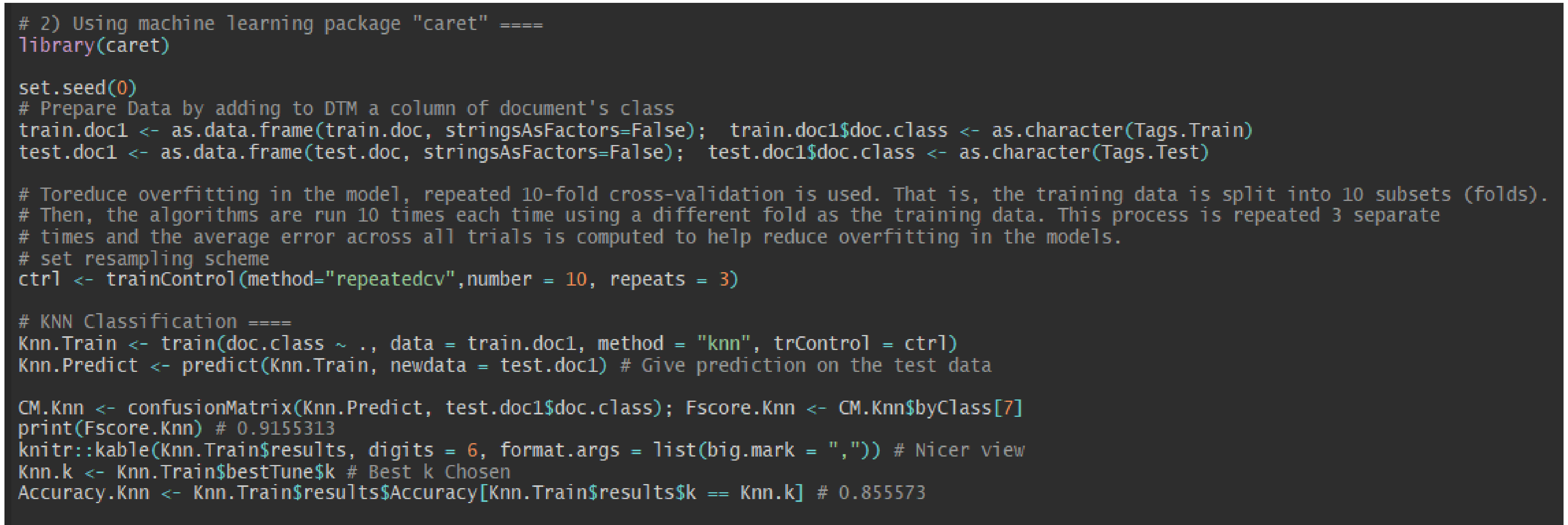
Implement knn algorithm from package "class"



```r
# Split the Document-Term Matrix into Train & Test Datasets
library(class) #
# Consider "spam" as Positive and "ham" as Negative
Positive <- "spam"; Negative <- "ham"; CM.Names <- c(Positive,Negative)
DS.Size <- dim(dtm)[1]
Test.Train.Percent <- 0.6 # Split Data into 60% for Training and 40% for Testing
ix.Range <- round(DS.Size*Test.Train.Percent)
train.Range <- seq(from = 1, to = ix.Range, by = 1); test.Range <- seq(from = (ix.Range+1), to = DS.Size, by = 1)

train.doc <- dtm[train.Range,] # Dataset for which classification is already known
test.doc <- dtm[test.Range,] # Dataset we are trying to classify

# Generate TAGS - Correct answers for the Train dataset
z <- c(SMS.Data$Tag[train.Range]);
z <- gsub(Positive,"Positive",z); z <- gsub(Negative,"Negative",z) # Replace Tags with
Tags.Train <- factor(z, levels = c("Positive","Negative")) # kNN expects Tags to be Factors data type

# Generate TAGS - Correct answers for the Test dataset
z <- c(SMS.Data$Tag[test.Range]);
sum(z==Positive) # 381 - Number of Positive events ("spam") in the Test Data
sum(z==Negative) # 2406 - Number of Negative events ("ham") in the Test Data
z <- gsub(Positive,"Positive",z); z <- gsub(Negative,"Negative",z) # Replace Tags with
Tags.Test <- factor(z, levels = c("Positive","Negative")) # kNN expects Tags to be Factors data type

# 1) KNN Classificationusing package "class" ====
set.seed(0)
prob.test <- knn(train.doc, test.doc, Tags.Train, k = 2, prob=TRUE) # k-number of neighbors considered

### -----------
# Display Classification Results
a <- 1:length(prob.test)
b <- levels(prob.test)[prob.test] # asign your classification Tags (Talk or Sci)
c <- attributes(prob.test)$prob # asign your classification probability values
d <- prob.test==Tags.Test # Logicaly match your classification Tags with the known "Tags"
result <- data.frame(Doc=a,Predict=b,Prob=c, Correct=d) # Tabulate your result
sum(d)/length(Tags.Test) # % Correct Classification (0.86)

### -----------
# Confusion Matrix
AutoCM <- table(prob.test, Tags.Test) # Automatic Confusion Matrix to compare with
TP <- AutoCM[1,1]; FN <- AutoCM[2,1]; FP <- AutoCM[1,2]; TN <- AutoCM[2,2];

P <- TP/(TP+FP) # Precision
R <- TP/(TP+FN) # Recall
Fscore <- 2*P*R/(P+R)
print(Fscore) # 0.2222222
Accuracy <- (TP+TN)/(TP+TN+FP+FN) # 0.86
```

K-Nearest Neighbors (kNN) - Classification ML Algorithm

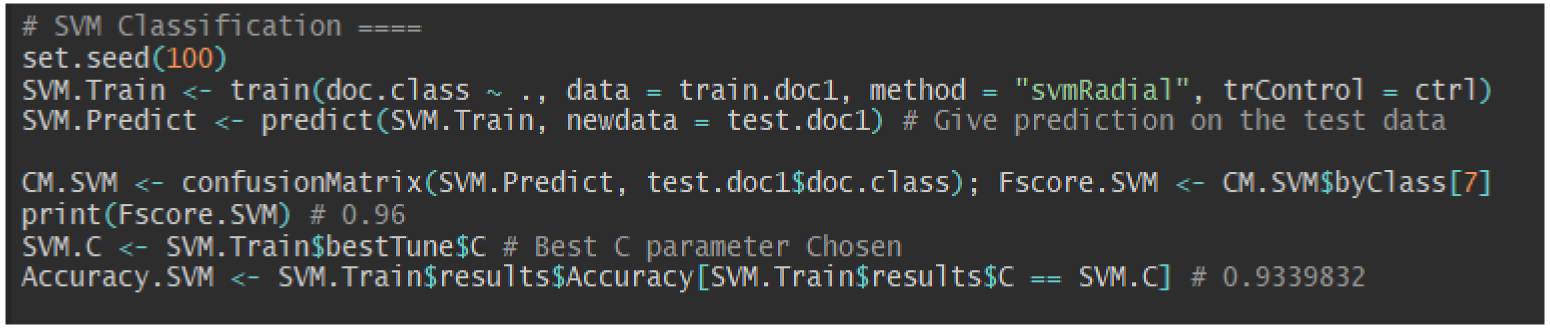
- kNN assumes that the classification of a new document is most similar to the classification of other documents that are “nearby” in some vector space.
- It predicts the class of a new document by finding the k most similar documents (neighbors) in the training data and using the class among those k neighbors that’s the most common. Alternatively, the classes of the neighbors are weighted using the similarity of each neighbor to the new document, where similarity is measured by Euclidean distance or the cosine value between two document vectors.
- kNN doesn’t rely on prior probabilities and is computationally efficient.
- Note the function “trainControl()” hat does Bagging & Bootstrapping.



```
# 2) Using machine learning package "caret" ====
library(caret)

set.seed(0)
# Prepare Data by adding to DTM a column of document's class
train.doc1 <- as.data.frame(train.doc, stringsAsFactors=False);  train.doc1$doc.class <- as.character(Tags.Train)
test.doc1 <- as.data.frame(test.doc, stringsAsFactors=False);  test.doc1$doc.class <- as.character(Tags.Test)

# Toreduce overfitting in the model, repeated 10-fold cross-validation is used. That is, the training data is split into 10 subsets (folds).
# Then, the algorithms are run 10 times each time using a different fold as the training data. This process is repeated 3 separate
# times and the average error across all trials is computed to help reduce overfitting in the models.
# set resampling scheme
ctrl <- trainControl(method="repeatedcv",number = 10, repeats = 3)

# KNN Classification ====
Knn.Train <- train(doc.class ~ ., data = train.doc1, method = "knn", trControl = ctrl)
Knn.Predict <- predict(Knn.Train, newdata = test.doc1) # Give prediction on the test data

CM.Knn <- confusionMatrix(Knn.Predict, test.doc1$doc.class); Fscore.Knn <- CM.Knn$byClass[7]
print(Fscore.Knn) # 0.9155313
knitr::kable(Knn.Train$results, digits = 6, format.args = list(big.mark = ",")) # Nicer view
Knn.k <- Knn.Train$bestTune$k # Best k Chosen
Accuracy.Knn <- Knn.Train$results$Accuracy[Knn.Train$results$k == Knn.k] # 0.855573
```

Support Vector Machine (SVM) - Classification ML Algorithms

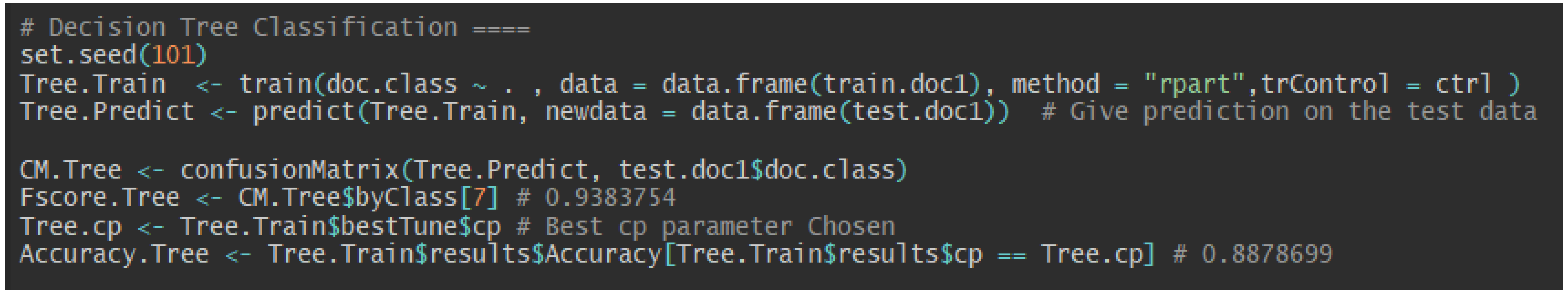
- SVM is a non-probabilistic **binary** linear classifier. Its training algorithm builds a model that assigns new examples into one of the 2 categories.
- The SVM model created by the training algorithm is a representation of the training data as points in space, mapped so that the examples of the 2 separate categories are divided by a clear gap that is as wide as possible.
- New train data are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.
- SVM finds the optimal plane with maximum distance (support vectors) from the nearest training data points.
- Non-linear classification is also possible using kernels for mapping into high-dimensional feature spaces.
- If the dataset has more than two classes, a multiclass SVM implementation must be performed. This is done by reducing the single multiclass problem into multiple binary classification problems.



```
# SVM Classification ====
set.seed(100)
SVM.Train <- train(doc.class ~ ., data = train.doc1, method = "svmRadial", trControl = ctrl)
SVM.Predict <- predict(SVM.Train, newdata = test.doc1) # Give prediction on the test data

CM.SVM <- confusionMatrix(SVM.Predict, test.doc1$doc.class); Fscore.SVM <- CM.SVM$byClass[7]
print(Fscore.SVM) # 0.96
SVM.C <- SVM.Train$bestTune$C # Best C parameter Chosen
Accuracy.SVM <- SVM.Train$results$Accuracy[SVM.Train$results$C == SVM.C] # 0.9339832
```

Decision Tree - Classification ML Algorithms

- Decision tree is a tree-like structure in which each internal node represents a ”test" on an attribute.
  - Each branch represents the outcome of the test
  - Each leaf node represents a class label
  - The paths from root to leaf represents classification rules.
- Decision trees work by recursively (iteratively) finding the variable (in a group of variables) that best splits the outcome into two classes.
- The recursion stops when all groups are sufficiently small/homogeneous/pure. It produces a non-linear model and the class/leaf will depend on many variables.
- Decision tree can be viewed as predictive models which map observations about an item to conclusions about the items target value.
- They are easy to interpret, can handle non-linear relationships well, no need to transform the input features, and they can be used for regression as well.
- Prone to overfitting, especially with a large number of variables, so proper cross-validation is necessary.

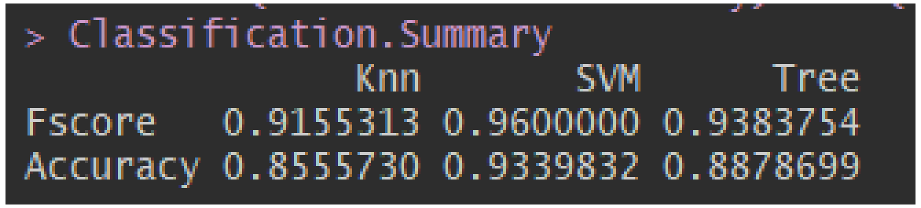


```
# Decision Tree Classification ====
set.seed(101)
Tree.Train   <- train(doc.class ~ . , data = data.frame(train.doc1), method = "rpart",trControl = ctrl )
Tree.Predict <- predict(Tree.Train, newdata = data.frame(test.doc1))  # Give prediction on the test data

CM.Tree <- confusionMatrix(Tree.Predict, test.doc1$doc.class)
Fscore.Tree <- CM.Tree$byClass[7] # 0.9383754
Tree.cp <- Tree.Train$bestTune$cp # Best cp parameter Chosen
Accuracy.Tree <- Tree.Train$results$Accuracy[Tree.Train$results$cp == Tree.cp] # 0.8878699
```

Compare Effectiveness of the Classification ML Algorithms

Compare effectiveness of different ML algorithms from the machine learning package "caret“ to classify the SMS messages

- knn algorithm
- SVM (Support Vector Machine) algorithm
- Decision Tree algorithm



```
> Classification.Summary
               Knn       SVM      Tree
Fscore   0.9155313 0.9600000 0.9383754
Accuracy 0.8555730 0.9339832 0.8878699
```

Zipf’s sentiment and Bayesian Classification Algorithm Outline

Use Custom Sentiment Analysis to perform the Classification, using R’s “tm” package for Text Mining, and performing the following steps:

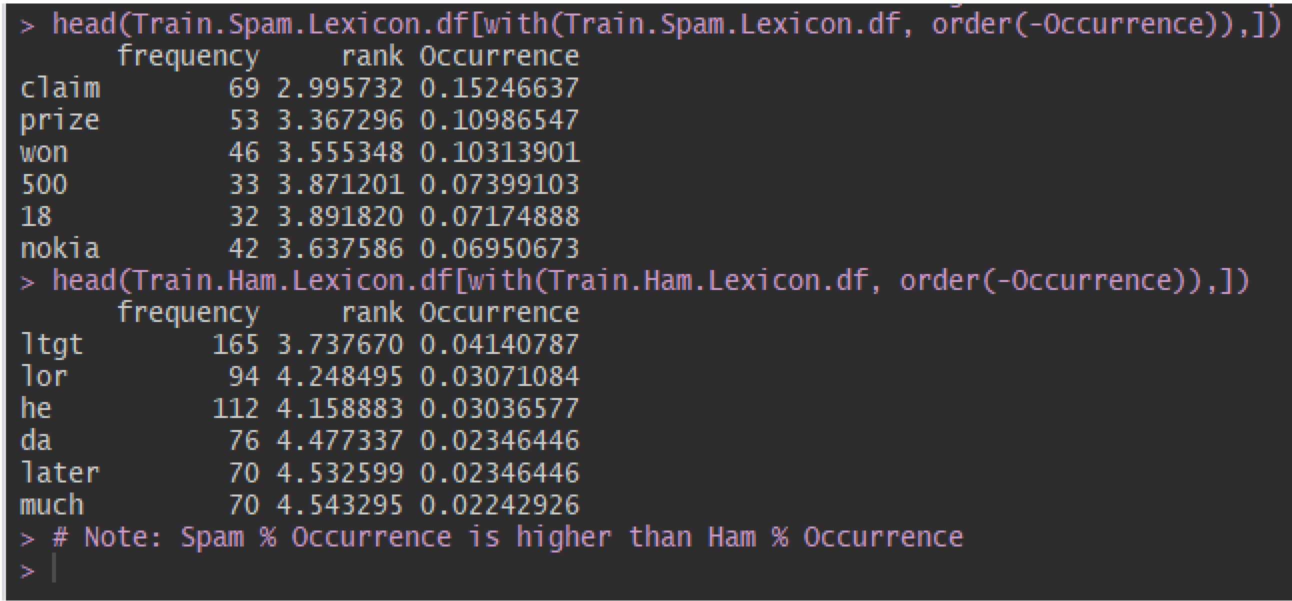
1. Split Data into 60% for Training and 40% for Testing
2. Train Data
   a) Split Train Data into Spam and Ham
   b) Create Train Corpora
   c) Train Corpora Pre-Processing
3. Test Data
   a) Split Test Data into Spam and Ham
   b) Create Test Corpora
   c) Test Corpora Pre-Processing
4. Create Train Data Spam & Ham Lexicons (words appearing only in Spam/Ham)
   - Find: Frequency, Rank and Occurrence (% of Docs (rows) in which given term (Col) occurs i.e. `length(which(dtm[,x] > 0))/nrow(dtm)`)
5. Calculate Test Data Score Based on Train Data Spam & Ham Lexicons
6. Create Classification Metrics based on Test Data Score

Term Occurrence as a %

- Occurrence is % of Docs (dtm Rows) in which given term (dtm Columns) occurs.
- Can be calculated simply:
    - For each term in the dtm
    - count the non-zero elements (rows) in each column and divide by the total number of rows to obtain the occurrence as %.
    - Term x Occurrence = { length(which(dtm[,x] > 0))/nrow(dtm) }
- Use "sapply(1:ncol(dtm), function(x) {…})"
- Check the data and see which terms are the strongest indicators of Spam. Sort by the occurrence column.

- Note: Spam % Occurrence is higher than

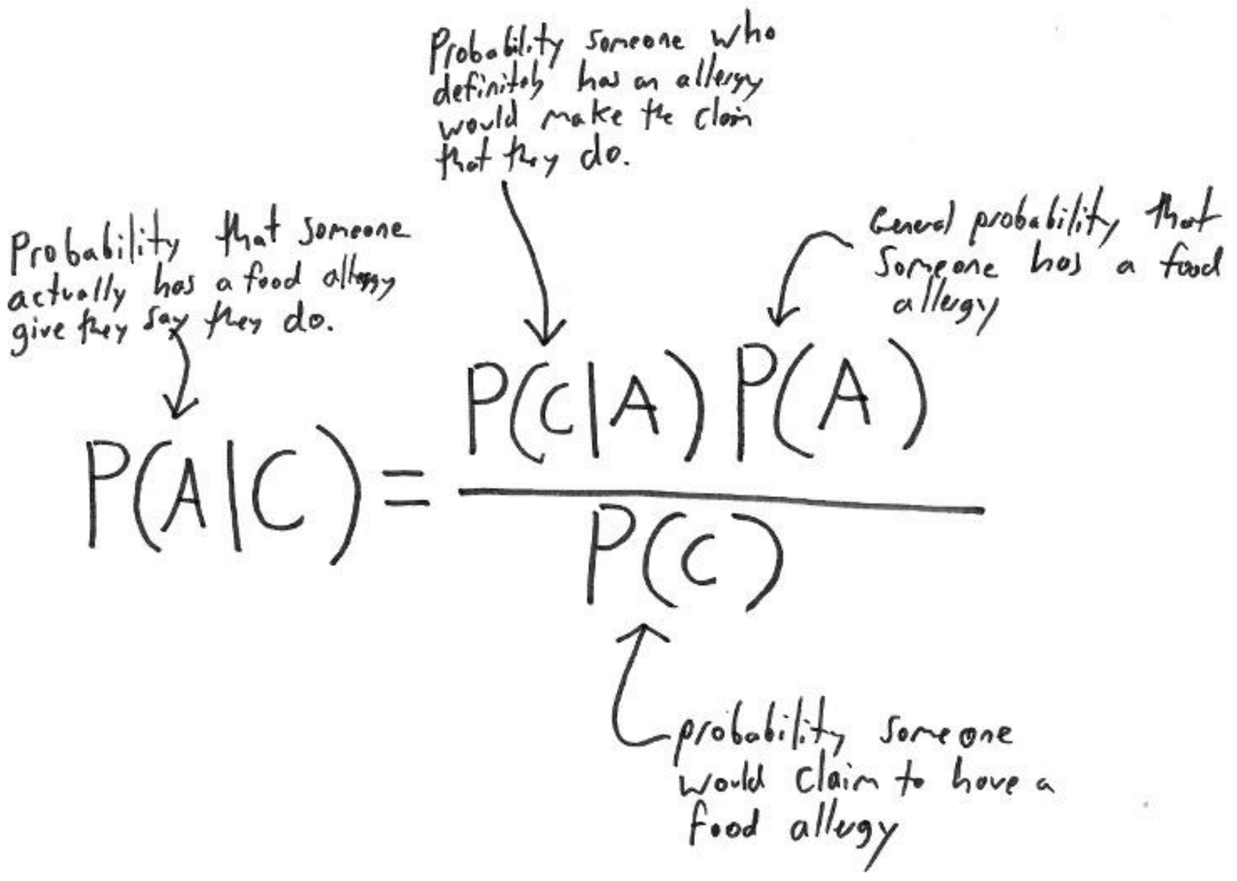
  Ham % Occurrence!
- Note: Higher Ham frequency is due to more Ham SMS.

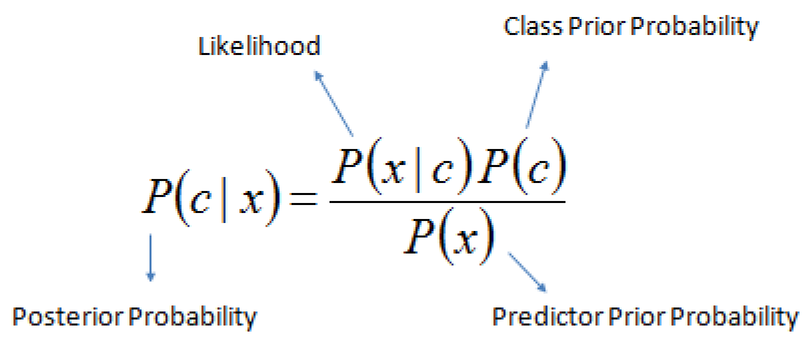


```
> head(Train.Spam.Lexicon.df[with(Train.Spam.Lexicon.df, order(-Occurrence)),])
      frequency     rank Occurrence
claim        69 2.995732 0.15246637
prize        53 3.367296 0.10986547
won          46 3.555348 0.10313901
500          33 3.871201 0.07399103
18           32 3.891820 0.07174888
nokia        42 3.637586 0.06950673
> head(Train.Ham.Lexicon.df[with(Train.Ham.Lexicon.df, order(-Occurrence)),])
      frequency     rank Occurrence
ltgt        165 3.737670 0.04140787
lor          94 4.248495 0.03071084
he          112 4.158883 0.03036577
da           76 4.477337 0.02346446
later        70 4.532599 0.02346446
much         70 4.543295 0.02242926
> # Note: Spam % Occurrence is higher than Ham % Occurrence
> 
```

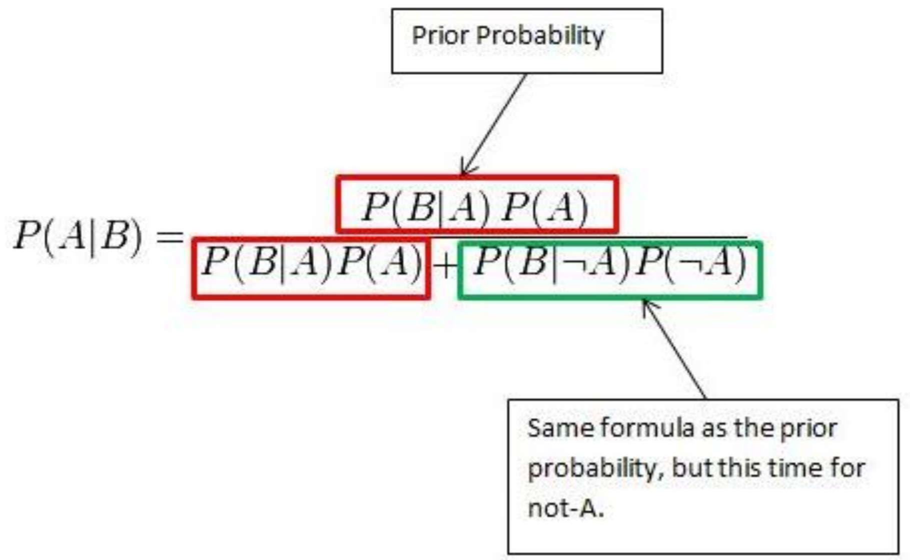
Bayesian Theorem

- A


Probability someone who definitely has an allergy would make the claim that they do.
Probability that someone actually has a food allergy give they say they do.
General probability that someone has a food allergy
$P(A|C) = \frac{P(C|A)\,P(A)}{P(C)}$
probability someone would claim to have a food allergy


Likelihood
Class Prior Probability
$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

$$P(c\,|\,\mathrm{X}) = P(x_1\,|\,c) \times P(x_2\,|\,c) \times \cdots \times P(x_n\,|\,c) \times P(c)$$


Prior Probability
$P(A|B) = \frac{P(B|A)\,P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}$
Same formula as the prior probability, but this time for not-A.

Bayesian Theorem

Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available (dynamic analysis of a sequence of data). Bayesian inference, where H is hypothesis and E is an observed event is:

- $P(H|E) = \frac{P(E|H)}{P(E)} P(H)$
- Posterior probability $P(H|E)$ - Estimate of the probability for the hypothesis H given the observed evidence E.
- A prior probability $P(H)$ - estimate of the probability of the hypothesis H before the current evidence E observed.
- The "likelihood function" $P(E|H)$ is derived from a statistical model for the observed data.
- The evidence probability (marginal likelihood or "model evidence") $P(E)$ - Prob of observing event E.
- For 2 events $P(H_1|E) = \frac{P(E|H_1)}{P_{tot}(E)} P(H_1) = \frac{P(E|H_1)}{P(E|H_1)P(H_1)+P(E|H_2)P(H_2)} P(H_1)$

Bayesian Theorem – Example 1

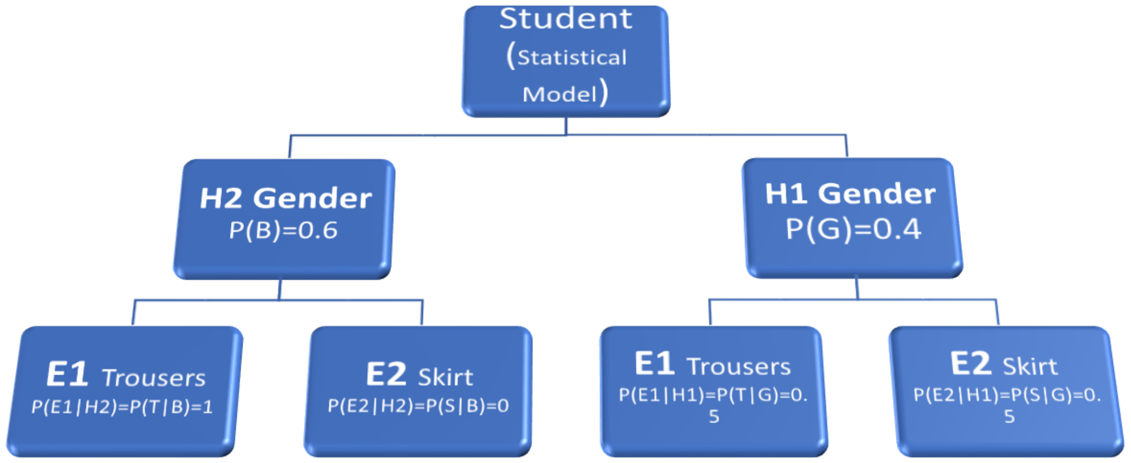
Suppose there is a mixed school having 60% boys and 40% girls as students. The girls wear trousers or skirts in equal numbers; the boys all wear trousers. An observer sees a (random) student from a distance; all the observer can see is that this student is wearing trousers. What is the probability this student is a girl? The correct answer can be computed using Bayes' theorem.

$H_1 = G\ ; H_2 = B\ ;$ $\quad$ $E_1 = T\ ;\ E_2 = S$

$P(H_1) = P(G) = 0.4$ $\quad$ $P(H_2) = P(B) = 0.6$

$P(E_1|H_1) = P(T|G) = 0.5$ $\quad$ $P(E_1|H_2) = P(T|B) = 1$

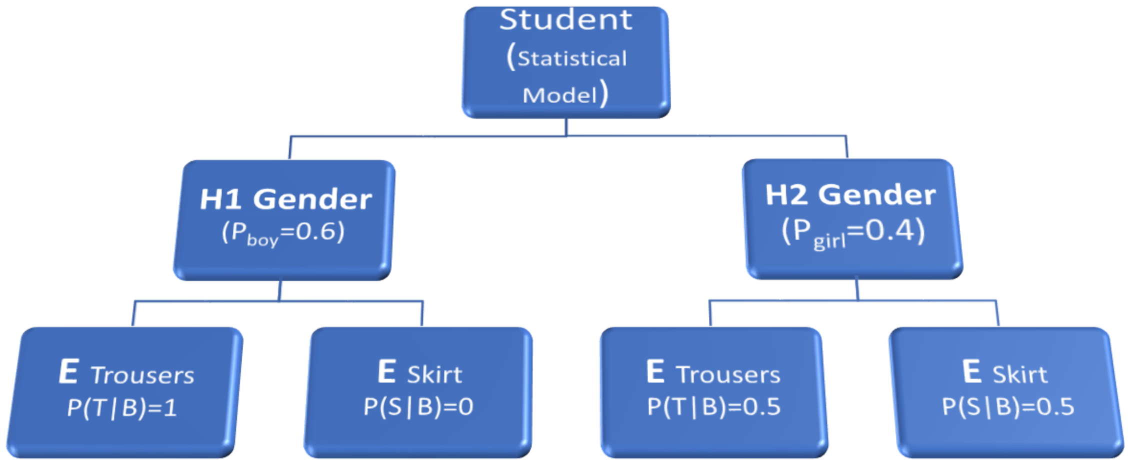
$P(E_2|H_1) = P(S|G) = 0.5$ $\quad$ $P(E_2|H_2) = P(S|B) = 0$



- $P(H_1|E_1) = \frac{P(E_1|H_1)}{P_{tot}(E)} P(H_1) = \frac{P(E_1|H_1)}{P(E_1|H_1)P(H_1)+P(E_1|H_2)P(H_2)} P(H_1)$
- $P(H_1|E) = \frac{P(E|H_1)}{P(E|H_1)P(H_1)+P(E|H_2)P(H_2)} P(H_1) = \frac{0.5*0.4}{0.4*0.5 + 1*0.6} = 0.25$
- $P(G|T) = 0.25$

Bayesian Theorem – Example 1

A


Student (Statistical Model)
H1 Gender (P_{boy}=0.6)
H2 Gender (P_{girl}=0.4)
E Trousers P(T|B)=1
E Skirt P(S|B)=0
E Trousers P(T|B)=0.5
E Skirt P(S|B)=0.5

Bayesian Theorem – Example 2

You're about to get on a plane to London. You want to know if you should bring an umbrella. You call 3 random friends of yours who live there and ask each independently if it's raining. Each of your friends has a 2/3 chance of telling you the truth and a 1/3 chance of messing with you by lying. All 3 friends tell you that "Yes" it is raining. What is the probability that it's actually raining in London?

H_1-rain, H_2 - no rain, E = yyy. **Assume** $P(H_1) = 1/4$, $P(H_2) = 3/4$

- $P(E|H_1) = P(yyy|R) = \left(\frac{2}{3}\right)^3$ and $P(E|H_2) = P(yyy|!R) = \left(\frac{1}{3}\right)^3$
- $P(H_1|E) = \frac{P(E|H_1)}{P(E|H_1)P(H_1)+P(E|H_2)P(H_2)}P(H_1) = \frac{\left(\frac{2}{3}\right)^3\frac{1}{4}}{\left(\frac{2}{3}\right)^3\frac{1}{4}+\left(\frac{1}{3}\right)^3\frac{3}{4}} = \frac{8}{11}$

Bayesian Classification

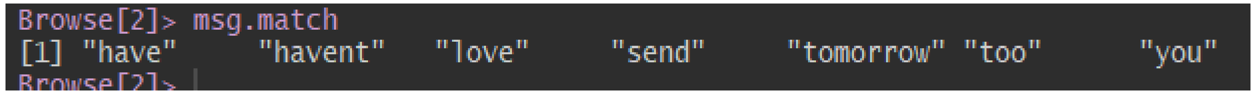
- Create Train ham/spam frequency table from Zipfs dtm containing:
  - | Term | Spam Freq | Ham Freq | Occurrence |
- Problem: What to do about the terms that are not in the training set? Simple rule:
  - assign a very small probability to terms that are not in the training set.
- Estimate the “spamminess” of a term given its context.

Automatic Document Classification.

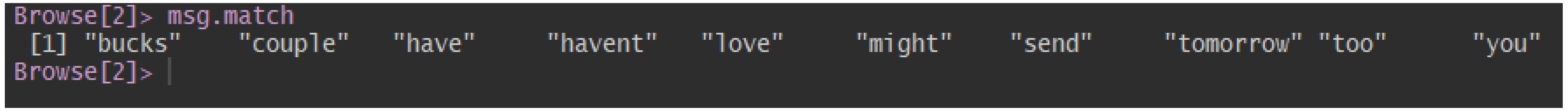
- Look at the words used in the documents and treat the presence or absence of each word as a feature. This would give you as many features as there are words in your vocabulary.
- Naïve Bayes—an extension of the Bayesian classifier—is a popular algorithm for the document-classification problems.
- $P(H_i|E) = \frac{P(E|H_i)}{P(E)} P(H_i)$ where H_i is the class i (Spam or Ham).
- Classification Score based on $P(H_i|E) < P(H_j|E)$ or $P(H_i|E) > P(H_j|E)$
- Use the right side of the formula to get the value on the left. Do this for each class and compare the two probabilities.
  1. Calculate $P(H_i)$ - frequency of class i by adding up how many times that class (spam/ham) appears, normalized (dividing) by the total number of SMS.
  2. Calculate $P(E|H_i)$-rewrite it as $P(E_0|H_i) \cdot P(E_1|H_i) \cdot P(E_2|H_i) \cdots$, we can calculate probability like this if we assume that all the words are independently likely (conditional independence).
  3. Note: No need to calculate $P(E)$ since they are all the same, and we only need to compare $P(H_i|E) < P(H_j|E)$ or $P(H_i|E) > P(H_j|E)$!!!

Automatic Document Classification.

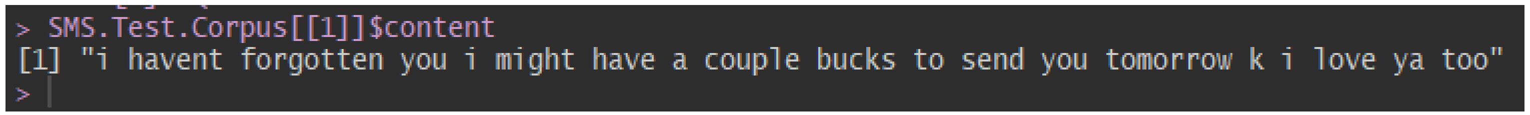
- Classification Score based on $S = P(H_i|E) = P(E|H_i) \cdot P(H_i)$
  - $P(H_i)$ – Prior probability, equals the ratio of the number of Spam/Ham SMS.
  - $P(E|H_i) = P_{new} \cdot P_{occurrence}$ – Calculate as product of 2 terms
    1. $P_{new} = P_0^n$ – If new words encountered, assign to them a very small occurrence (prior) probability $P_0 = 10^{-6}$.
       - Count the number n of new Test SMS words (not present in Train data) and take $P_0 = 10^{-6}$.
    2. $P_{occurrence}$– Product of word occurrences (feature) in SMS.
       - Multiply the occurrence of the existing Test SMS words present in Train data.
  - A



```
Browse[2]> msg.match
[1] "have"     "havent"   "love"     "send"     "tomorrow" "too"      "you"
Browse[2]>
```



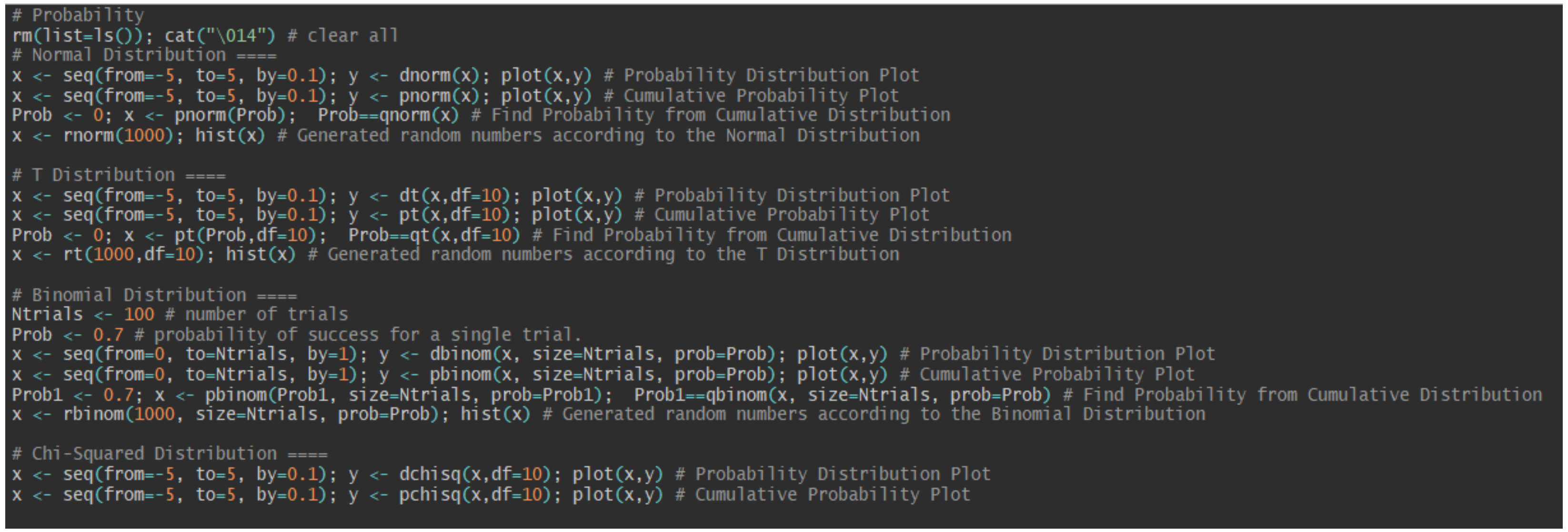
```
Browse[2]> msg.match
 [1] "bucks"    "couple"   "have"     "havent"   "love"     "might"    "send"     "tomorrow" "too"      "you"
Browse[2]>
```



```
> SMS.Test.Corpus[[1]]$content
[1] "i havent forgotten you i might have a couple bucks to send you tomorrow k i love ya too"
>
```

Probability in R

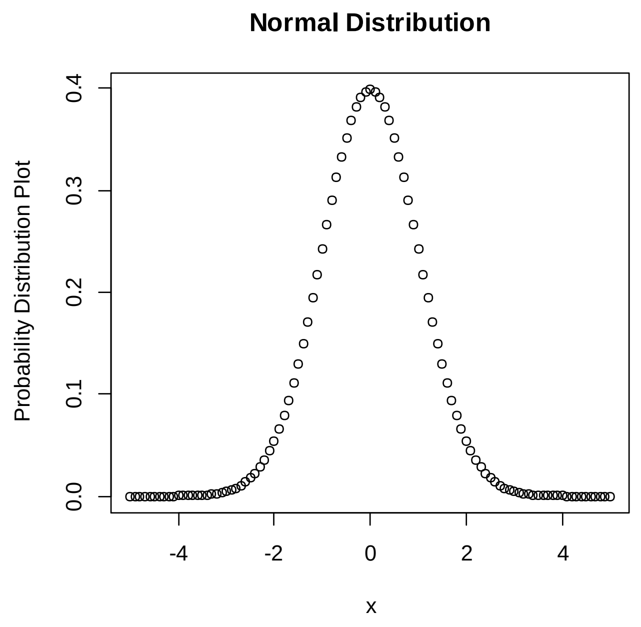
- Several probability distributions
  - Normal Distribution
    - dnorm(x) – Generates Probability Distribution for sequence x.
    - pnorm(x) – Generates Cumulative Probability for sequence x.
    - qnorm(x) – Find Probability from given Cumulative Distribution x (gives 0 for x=0.5).
    - rnorm(n) – Generates n random numbers according to the Normal Distribution.
  - Binomial Distribution
    - dbinom – Generates Probability Distribution
    - pbinom – Generates Cumulative Probability
    - qbinom – Find Probability from given Cumulative Distribution
  - Chi-Squared Distribution
    - dchisq – Generates Probability Distribution
    - pchisq – Generates Cumulative Probability
  - T Distribution
    - dt – Generates Probability Distribution
    - pt – Generates Cumulative Probability
    - qt – Find Probability from given Cumulative Distribution

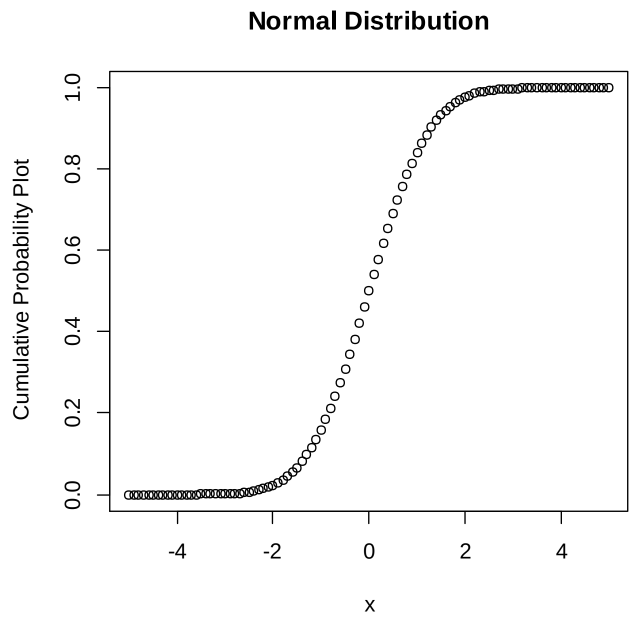


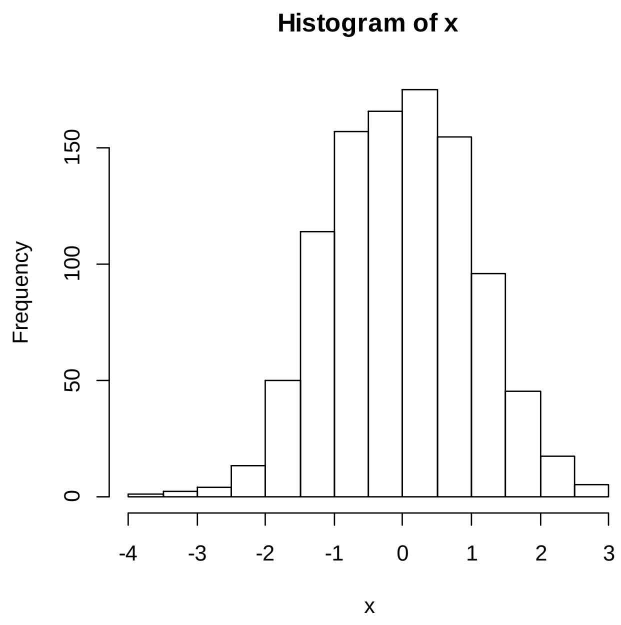
```r
# Probability
rm(list=ls()); cat("\014") # clear all
# Normal Distribution ====
x <- seq(from=-5, to=5, by=0.1); y <- dnorm(x); plot(x,y) # Probability Distribution Plot
x <- seq(from=-5, to=5, by=0.1); y <- pnorm(x); plot(x,y) # Cumulative Probability Plot
Prob <- 0; x <- pnorm(Prob);  Prob==qnorm(x) # Find Probability from Cumulative Distribution
x <- rnorm(1000); hist(x) # Generated random numbers according to the Normal Distribution

# T Distribution ====
x <- seq(from=-5, to=5, by=0.1); y <- dt(x,df=10); plot(x,y) # Probability Distribution Plot
x <- seq(from=-5, to=5, by=0.1); y <- pt(x,df=10); plot(x,y) # Cumulative Probability Plot
Prob <- 0; x <- pt(Prob,df=10);  Prob==qt(x,df=10) # Find Probability from Cumulative Distribution
x <- rt(1000,df=10); hist(x) # Generated random numbers according to the T Distribution

# Binomial Distribution ====
Ntrials <- 100 # number of trials
Prob <- 0.7 # probability of success for a single trial.
x <- seq(from=0, to=Ntrials, by=1); y <- dbinom(x, size=Ntrials, prob=Prob); plot(x,y) # Probability Distribution Plot
x <- seq(from=0, to=Ntrials, by=1); y <- pbinom(x, size=Ntrials, prob=Prob); plot(x,y) # Cumulative Probability Plot
Prob1 <- 0.7; x <- pbinom(Prob1, size=Ntrials, prob=Prob1);  Prob1==qbinom(x, size=Ntrials, prob=Prob) # Find Probability from Cumulative Distribution
x <- rbinom(1000, size=Ntrials, prob=Prob); hist(x) # Generated random numbers according to the Binomial Distribution

# Chi-Squared Distribution ====
x <- seq(from=-5, to=5, by=0.1); y <- dchisq(x,df=10); plot(x,y) # Probability Distribution Plot
x <- seq(from=-5, to=5, by=0.1); y <- pchisq(x,df=10); plot(x,y) # Cumulative Probability Plot
```

Probability and R - Normal Distribution

- Normal Distribution
  - dnorm(x) – Generates Probability Distribution for sequence x.
  - pnorm(x) – Generates Cumulative Probability for sequence x.
  - qnorm(x) – Find Probability from given Cumulative Distribution x (gives 0 for x=0.5).
  - rnorm(n) – Generates n random numbers according to the Normal Distribution.

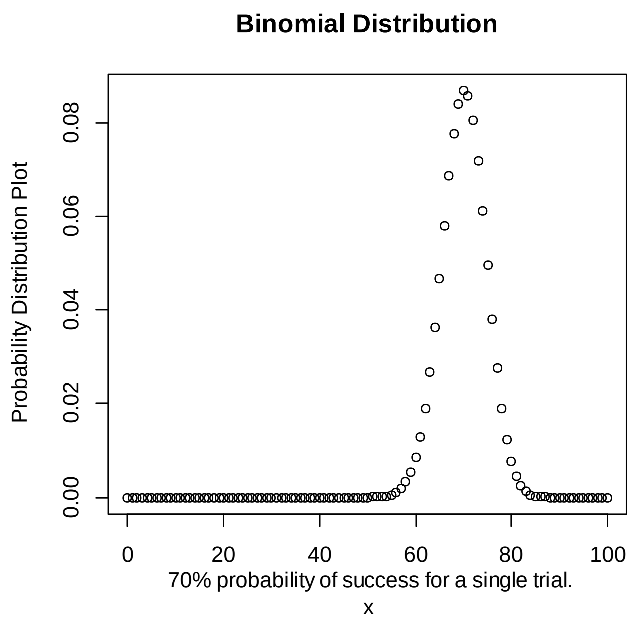


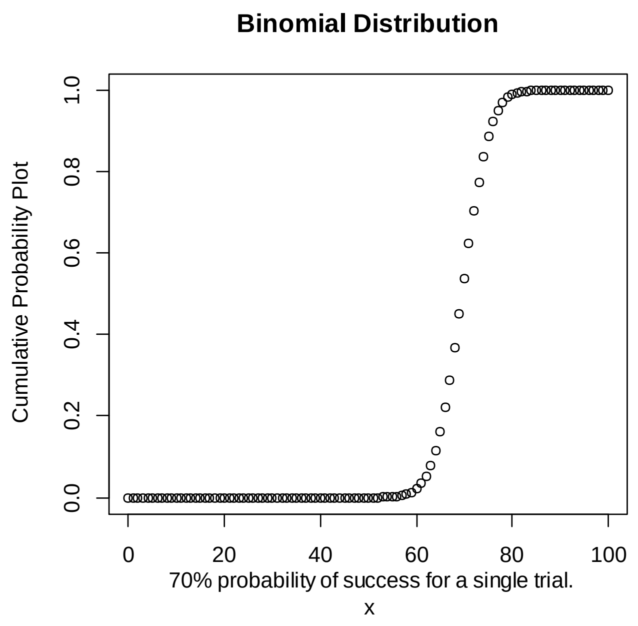


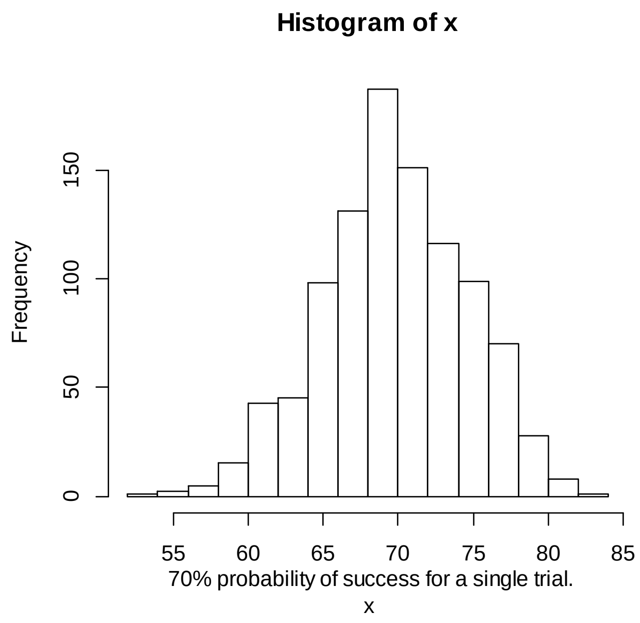


Probability and R - Binomial Distribution

- Binomial Distribution
  - dbinom – Generates Probability Distribution.
  - pbinom – Generates Cumulative Probability.
  - qbinom – Find Probability from given Cumulative Distribution.
  - rbinom (n) – Generates n random numbers according to the Binomial Distribution.







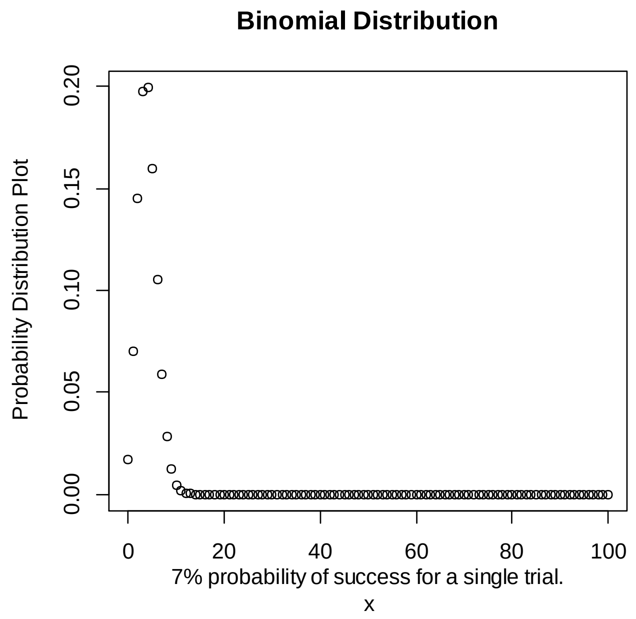
Binomial Distribution Example with R

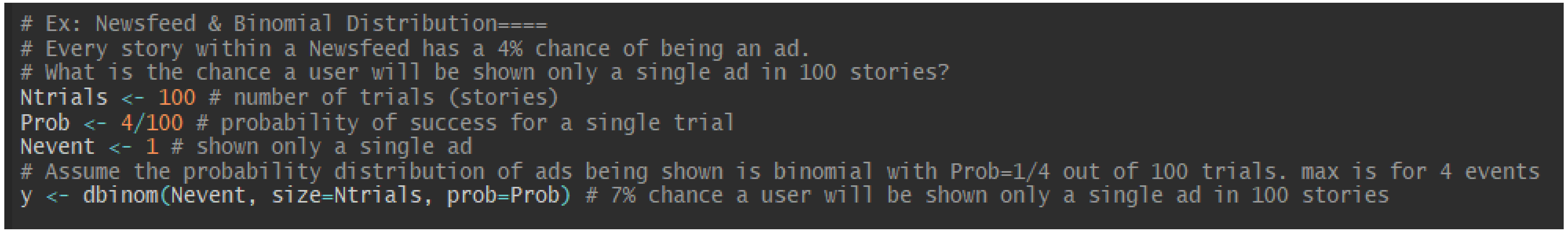
Newsfeed and Binomial Distribution Example

- Newsfeed and ads stories are being shown to a customer.
- Every story within a Newsfeed has a 4% chance of being an ad.
- Assume the probability distribution of ads being shown to a customer is binomial.
- What is the chance a user will be shown only a single ad in 100 stories?

Answer: 7%

- To find the answer use Binomial Distribution (dbinom) with
  - Ntrials = 100 # number of trials (stories)
  - Prob = 4/100 # probability of success for a single trial
  - Nevent = 1 # shown only a single ad
- dbinom(Nevent, size=Ntrials, prob=Prob) = 7%

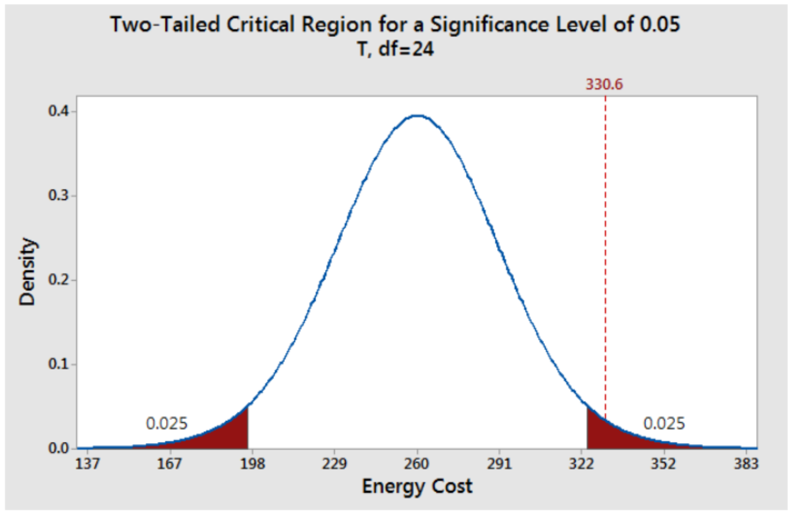


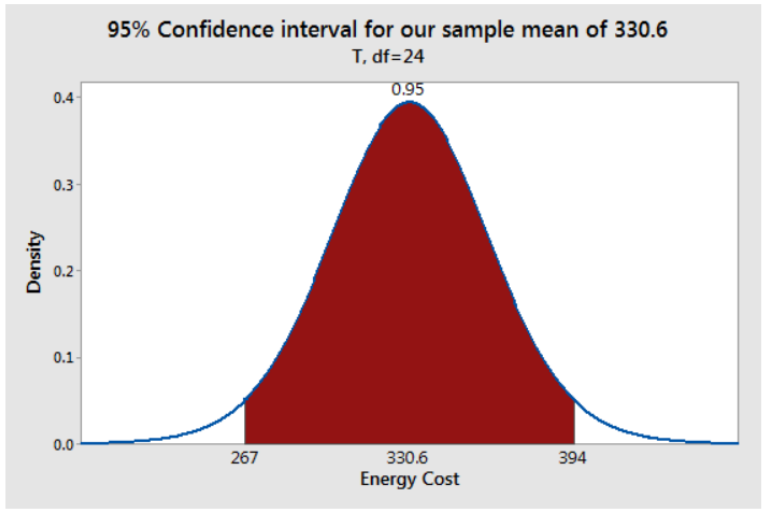


```
# Ex: Newsfeed & Binomial Distribution====
# Every story within a Newsfeed has a 4% chance of being an ad.
# What is the chance a user will be shown only a single ad in 100 stories?
Ntrials <- 100 # number of trials (stories)
Prob <- 4/100 # probability of success for a single trial
Nevent <- 1 # shown only a single ad
# Assume the probability distribution of ads being shown is binomial with Prob=1/4 out of 100 trials. max is for 4 events
y <- dbinom(Nevent, size=Ntrials, prob=Prob) # 7% chance a user will be shown only a single ad in 100 stories
```

Statistical Methods - Hypothesis testing

1. **Hypothesis testing**. Retain or reject hypothesis based on measurements of observed samples. The decision is often based on a statistical mechanism called hypothesis testing.
    - Determine whether the means of two groups are equal to each other.
    - The null hypothesis is that the two means are equal.
    - Type 1 error (false positives) - falsely rejecting a null hypothesis when the null hypothesis is true.
    - α - significance level of hypothesis testing - the probability of committing a type 1 error.
    - Type 2 error (false negatives)
    - P-value to retain or reject hypothesis.
        - If p.value < 0.05 => Reject the Null Hypothesis. The analyzed sample is statistically significant (sample statistic is unusual enough relative to the null hypothesis). Conclusion: Means are not the same.

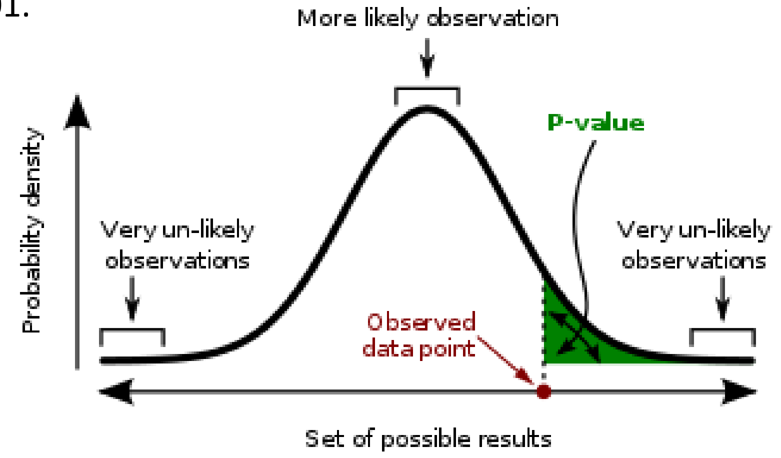




Hypothesis testing – p Value

p Value - The probability of observing a more extreme result assuming the null hypothesis is true.

- Used in the context of null hypothesis testing in order to quantify the idea of statistical significance of evidence (reductio ad absurdum argument).
- The smaller the p-value, the higher the significance because it tells the investigator that the null hypothesis may not adequately explain the observation.
- The null hypothesis is **rejected** if the p-Value (probability) is less than or equal to a small, but arbitrarily pre-defined threshold value (level of significance).
  - If $p \leq \alpha$ reject the Null Hypothesis.
  - The level of significance α is arbitrary but commonly set to 0.05, 0.01, 0.005, or 0.001.

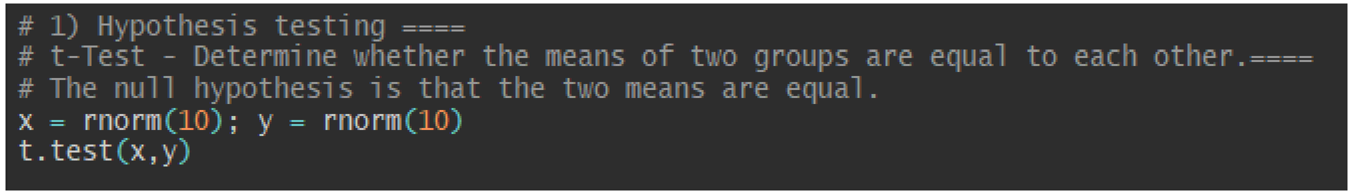


A p-value (shaded green area) is the probability of an observed (or more extreme) result assuming that the null hypothesis is true.

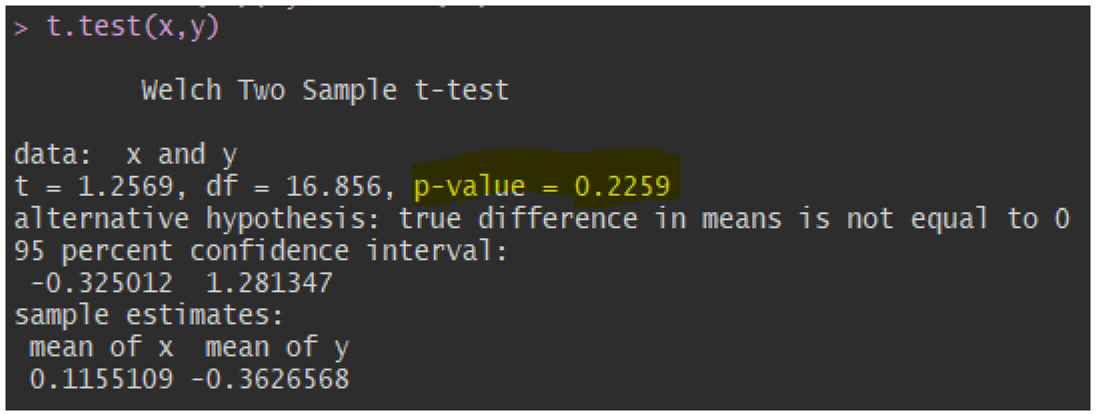
Hypothesis testing Example

Use t-Test to determine whether the means of two sequences x and y are equal to each other.

- Generate 2 random numbers sequences x and y, according to the Normal Distribution.
    - rnorm(n)
- Take the null hypothesis to be:
    - The two means of x and y are equal.
- Find for which n you can retain the null hypothesis.



```
# 1) Hypothesis testing ====
# t-Test - Determine whether the means of two groups are equal to each other.====
# The null hypothesis is that the two means are equal.
x = rnorm(10); y = rnorm(10)
t.test(x,y)
```



```
> t.test(x,y)

        Welch Two Sample t-test

data:  x and y
t = 1.2569, df = 16.856, p-value = 0.2259
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.325012  1.281347
sample estimates:
 mean of x  mean of y
 0.1155109 -0.3626568
```

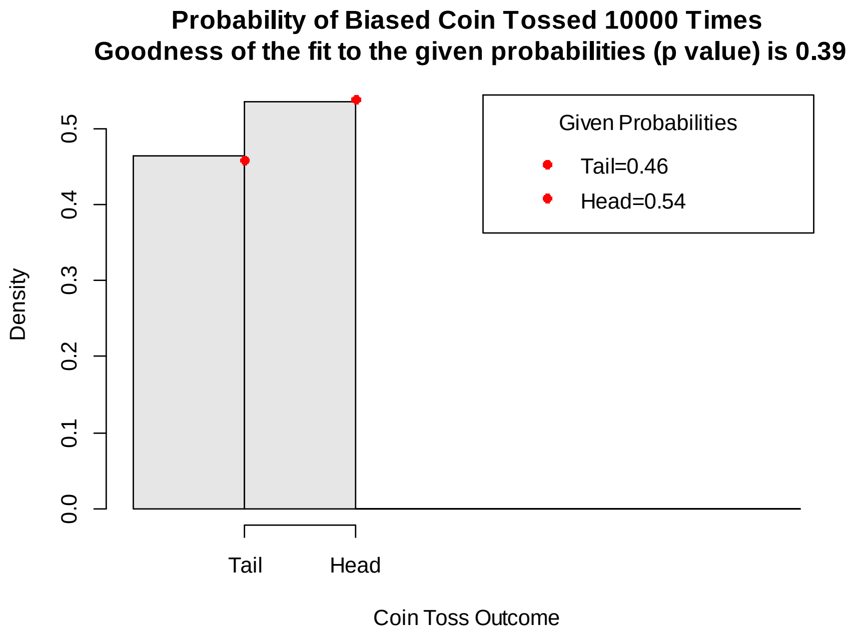
Statistical Methods - Statistical inference

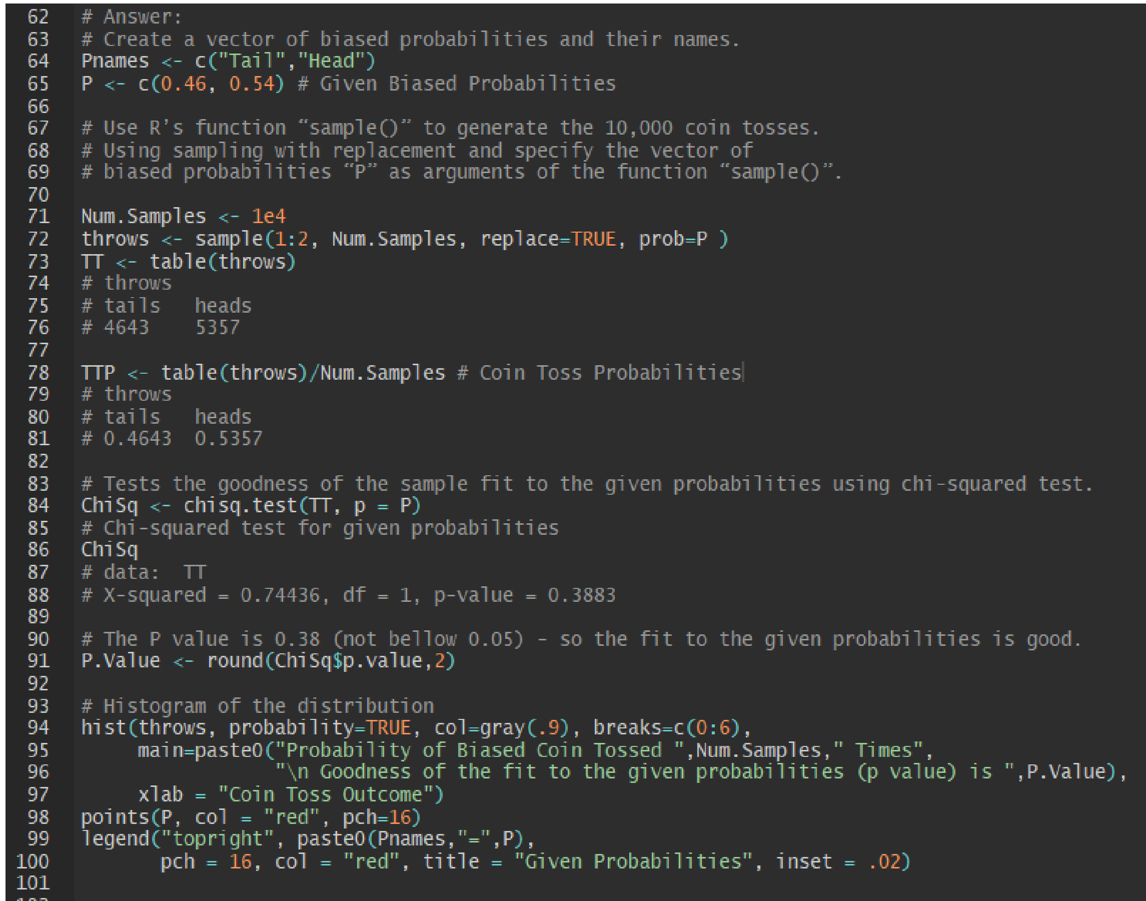
- ANOVA - a statistical tool used in several ways to develop and confirm an explanation for the observed data. Exploratory data analysis, employs an additive data decomposition.
- ANOVA Null Hypothesis (assumption): Y has normal distribution for each categorical group in X.
- ANOVA is useful for comparing (testing) three or more means (groups or variables) for statistical significance.

- ANOVA Example
  - Study effects of tea (X) on weight loss (Y).
    - individuals randomly split into smaller groups and drinking different kinds of tea (X).
    - individuals (X) split into groups based on an attribute they possess.
  - Blood pressure Y among 3 age groups X.
    - ANOVA Null Hypothesis is: Y has normal distribution for each cathegory (age group).
    - Check for normality
    - Check for equal variance
    - ANOVA uses F (Fisher) – statistic, which is simply a ratio of two variances (how far is the data are scattered from the mean).
    - For one-way ANOVA, the ratio of the between-group variability to the within-group variability follows an F-distribution when the null hypothesis is true.

Hypothesis Testing Examples: Biased Coin

- A coin is sold for cheaters which is weighted unevenly so that the probability of a head is 0.54 versus the probability of a tail which is 0.46.
- Create a code that can prove/disprove the hypothesis if a coin is biased.
- Generate 10,000 coin tosses and create a plot of the probability.





```r
# Answer:
# Create a vector of biased probabilities and their names.
Pnames <- c("Tail","Head")
P <- c(0.46, 0.54) # Given Biased Probabilities

# Use R's function "sample()" to generate the 10,000 coin tosses.
# Using sampling with replacement and specify the vector of
# biased probabilities "P" as arguments of the function "sample()".

Num.Samples <- 1e4
throws <- sample(1:2, Num.Samples, replace=TRUE, prob=P )
TT <- table(throws)
# throws
# tails   heads
# 4643    5357

TTP <- table(throws)/Num.Samples # Coin Toss Probabilities
# throws
# tails   heads
# 0.4643  0.5357

# Tests the goodness of the sample fit to the given probabilities using chi-squared test.
ChiSq <- chisq.test(TT, p = P)
# Chi-squared test for given probabilities
ChiSq
# data:  TT
# X-squared = 0.74436, df = 1, p-value = 0.3883

# The P value is 0.38 (not bellow 0.05) - so the fit to the given probabilities is good.
P.Value <- round(ChiSq$p.value,2)

# Histogram of the distribution
hist(throws, probability=TRUE, col=gray(.9), breaks=c(0:6),
     main=paste0("Probability of Biased Coin Tossed ",Num.Samples," Times",
                 "\n Goodness of the fit to the given probabilities (p value) is ",P.Value),
     xlab = "Coin Toss Outcome")
points(P, col = "red", pch=16)
legend("topright", paste0(Pnames,"=",P),
       pch = 16, col = "red", title = "Given Probabilities", inset = .02)

```

A

- A