Intermediate RStudio Training

January 23, 2018

GSA

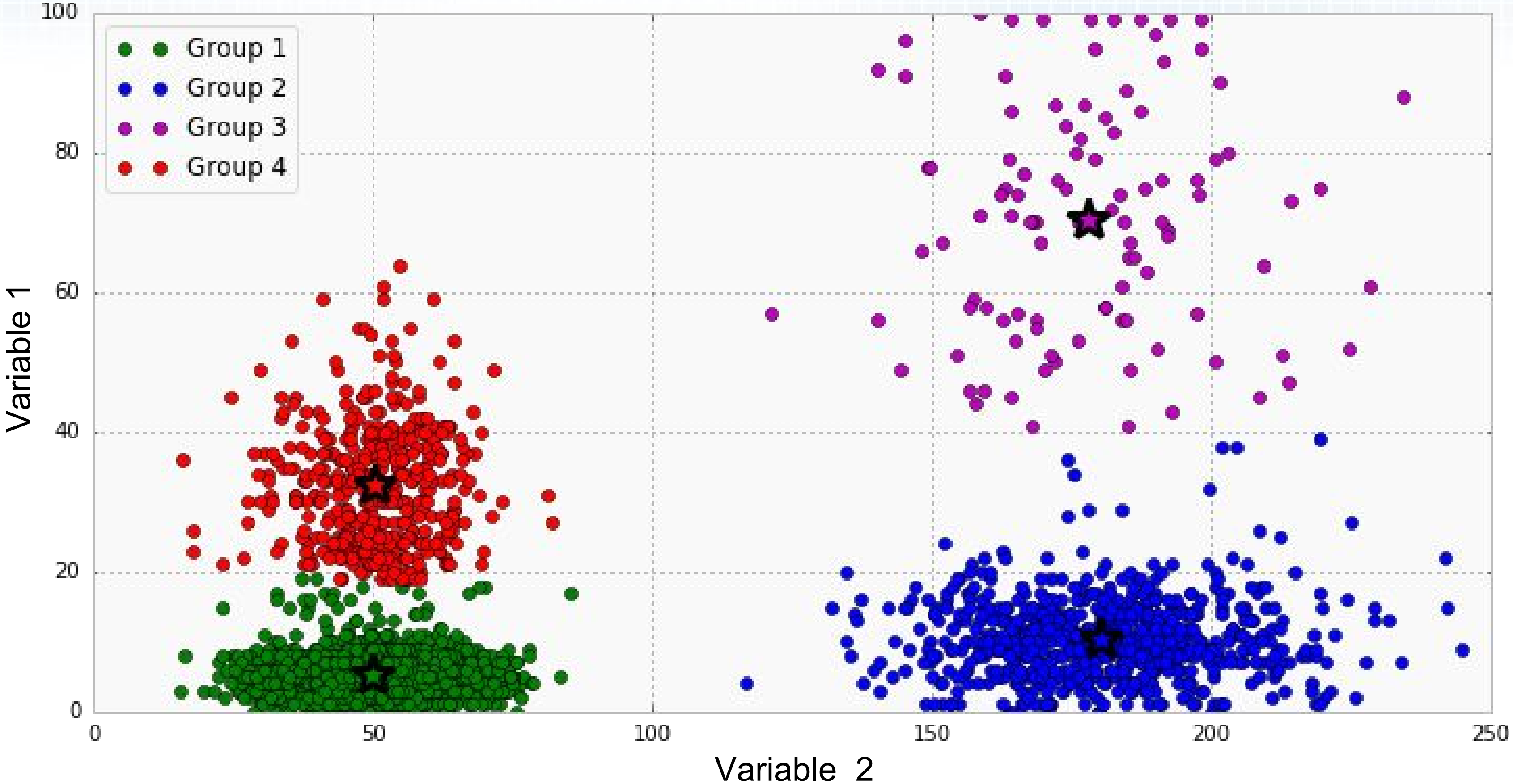
D2D
DATA TO DECISIONS

Course Outline

- Clusters
- Outliers
- Regression
- Text Analytics
- Q & A

Clustering

Grouping of similar objects in multivariate data set
Example: grouping by distance from a center



Clustering with k-means()

- **kmeans()** clustering is the most commonly used algorithm for partitioning a given data set into a set of k groups/clusters
  - ➢ k represents the number of groups pre-specified by the analyst. It classifies objects in multiple groups (i.e., clusters), such that objects within the same cluster are as similar as possible
  - ➢ In k-means clustering, each cluster is represented by its center (i.e, centroid) which corresponds to the mean of points assigned to the cluster

```
kmeans(x, centers, iter.max = 10, nstart = 1,
    algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
    "MacQueen"), trace=FALSE)
```

- **nstart()** is optional parameter in kmeans() setting initial number of configurations, e.g. nstart=25 will generate 25 initial random centroids and choose the best one for the algorithm
- **set.seed(seed)** is random number generator, which is useful for creating simulations or random objects that can be reproduced, required by nstart()

GSA

D2D
DATA TO DECISIONS

R Script for k-means

```
library(ggplot2)
ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()
set.seed(20)
irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)
irisCluster
table(irisCluster$cluster, iris$Species)
irisCluster$cluster <- as.factor(irisCluster$cluster)
ggplot(iris, aes(Petal.Length, Petal.Width, color = irisCluster$cluster)) + geom_point()
```

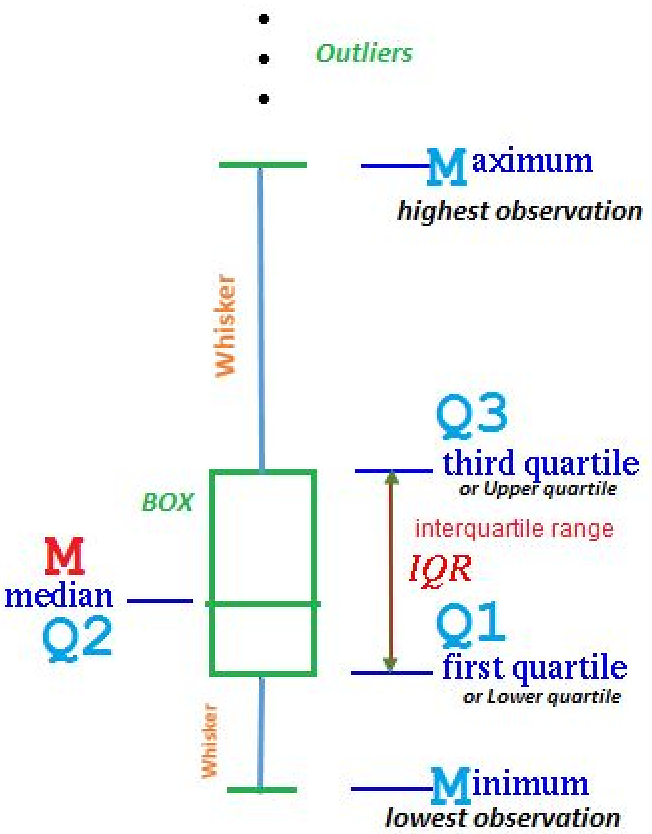
Or:

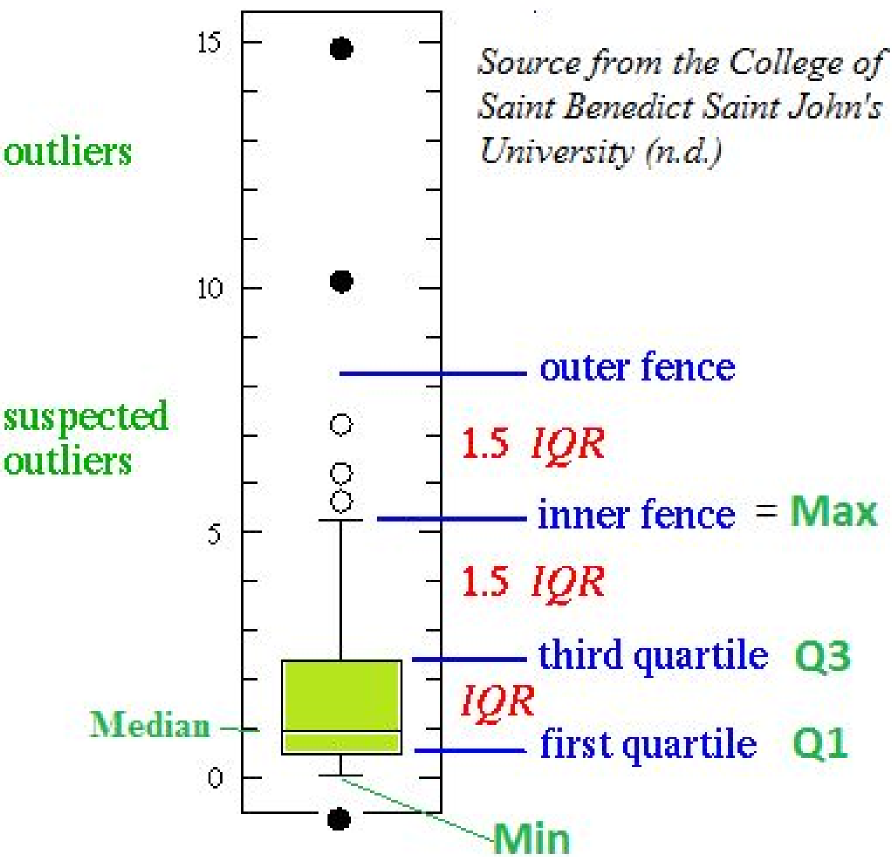
```
ggplot(iris, aes(y = Petal.Length, x = seq(1, length(iris$Sepal.Length)), color = irisCluster$cluster)) + geom_point()
irisCluster1 = kmeans(iris[,3], 3, nstart = 20)
irisCluster1
table(irisCluster1$cluster, iris$Species)
```

GSA

D2D
DATA TO DECISIONS

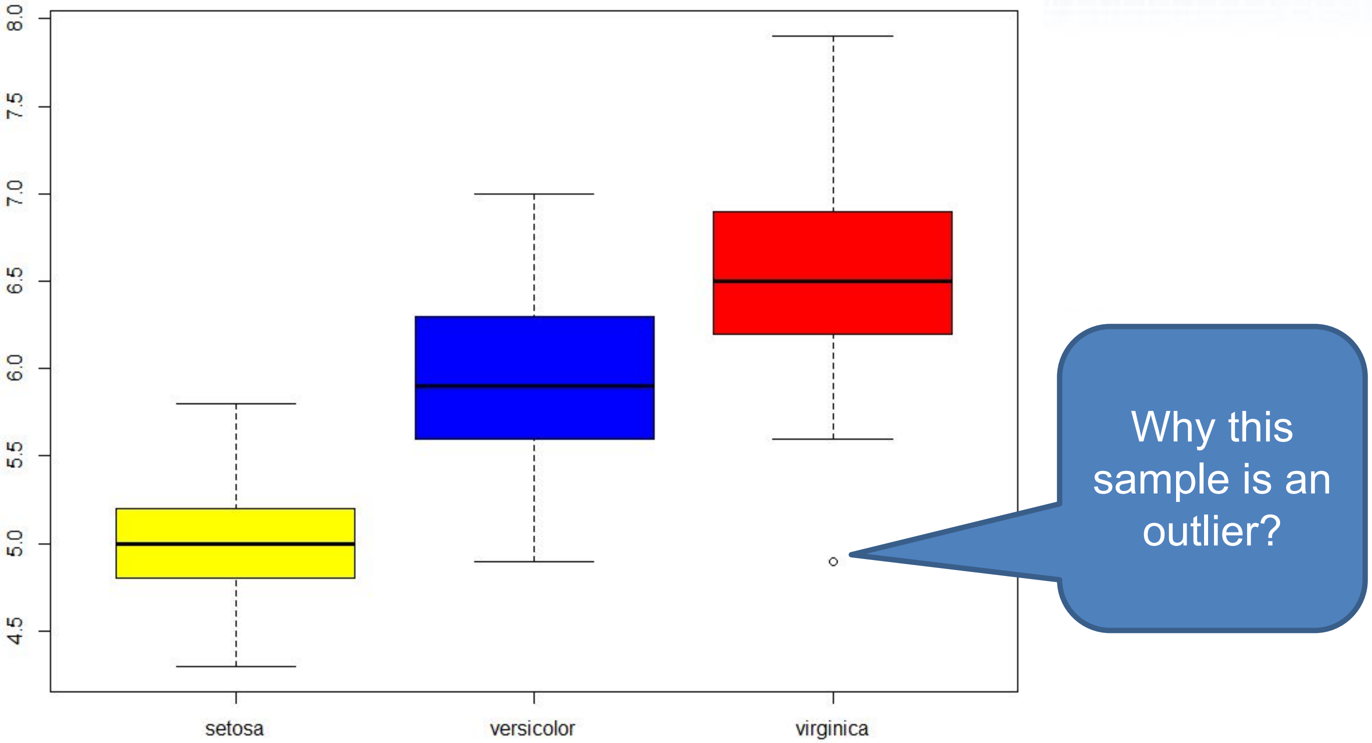
Outliers with Boxplots





Outliers in Iris Boxplot

boxplot(Sepal.Length ~ Species, data=iris, col= c("yellow", "blue", "red"), ylab="Sepal Length")



Lets explore boxplot defaults:

boxplot.default(x, ..., **range = 1.5**, width = NULL, varwidth = FALSE, notch = FALSE, names, data = sys.frame(sys.parent()), plot = TRUE, border = par("fg"), col = NULL, log = "", pars = NULL)

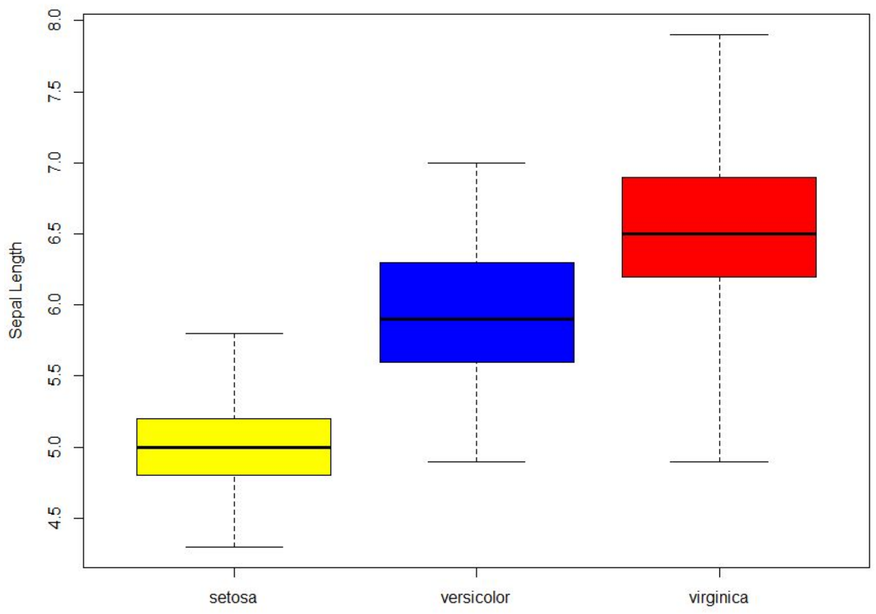
GSA

D2D
DATA TO DECISIONS

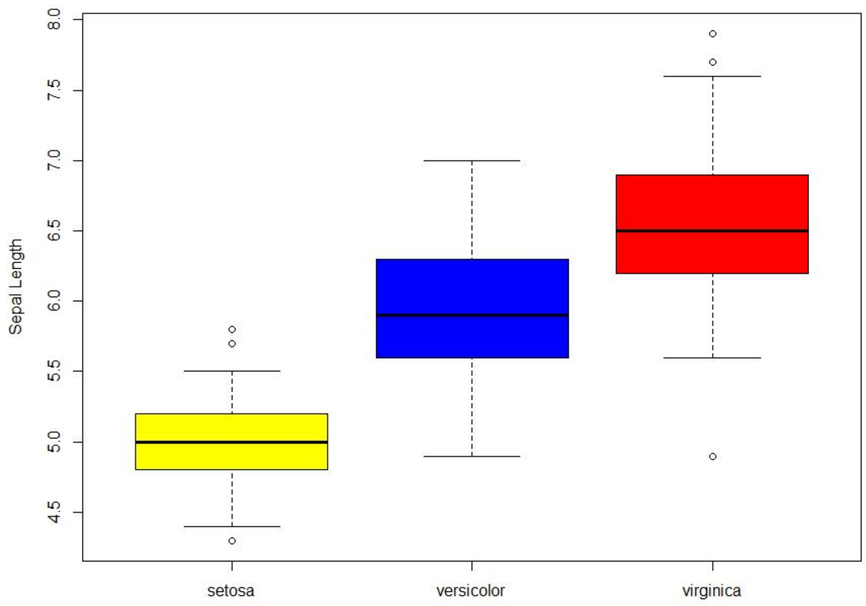
Outliers in Boxplot

by Changing Range

boxplot(Sepal.Length ~ Species, data=iris, **range = 0**, col= c("yellow", "blue", "red"), ylab="Sepal Length")



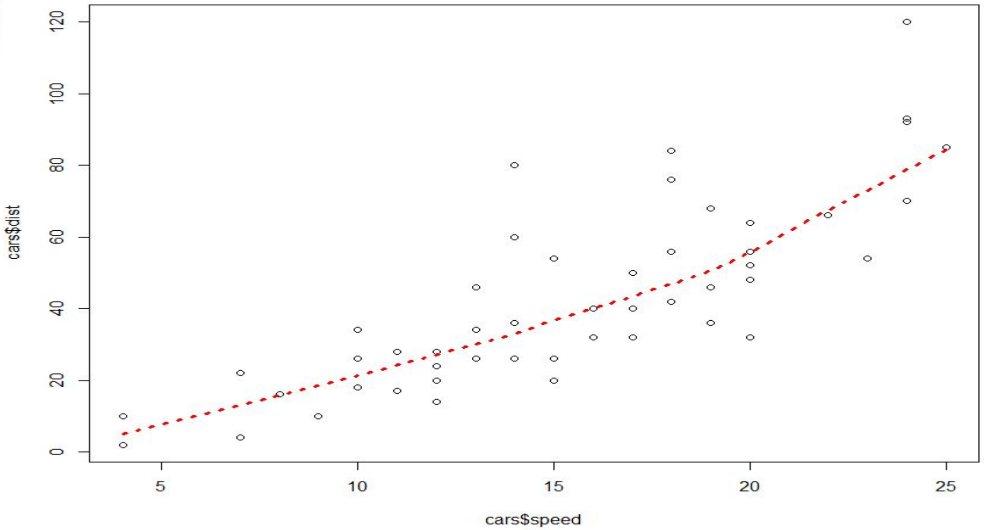
boxplot(Sepal.Length ~ Species, data=iris, **range = 1**, col= c("yellow", "blue", "red"), ylab="Sepal Length")



Range = X determines how far the plot whiskers extend out from the box. X is a multiplier for the IQR value. X = zero causes the whiskers to extend to the data extremes (min/max).

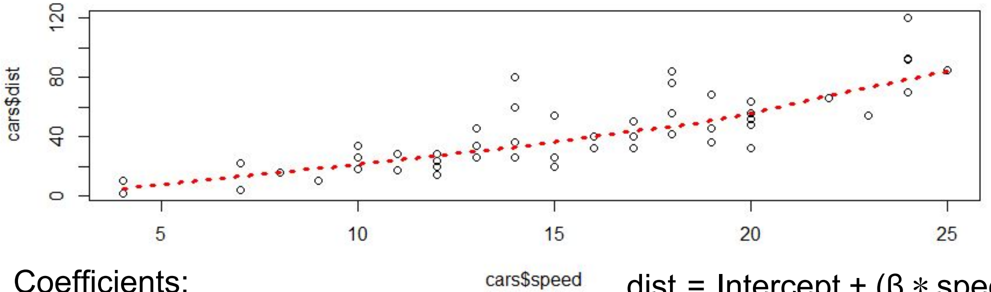
Regression with Scatter Plot

```
scatter.smooth(x = cars$speed, y = cars$dist, lpars = list(col = "red", lwd = 3, lty = 3))
```



OR with linear model:

```
linearMod = lm(dist ~ speed, data=cars)
print(linearMod)
```



Coefficients:

(Intercept)	speed
-17.579	3.932

dist = Intercept + (β ∗ speed)

=> dist = −17.579 + 3.932∗speed

Text Mining

- Natural languages are different from programming languages
- The semantic or the meaning of a statement depends on the context, tone and other factors
- Unlike programming languages, natural languages are ambiguous

- Text mining deals with helping computers understand the “meaning” of the text.
- Some of the common text mining applications include sentiment analysis e.g if a tweet about a movie says something positive or not, text classification e.g classifying the mails you get as spam or ham etc.

Text Preprocessing

- Text data contains white spaces, punctuations, stopwords etc.
- These characters do not convey much information and are hard to process.
- For example, English stopwords like “the”, “is” etc. do not tell you much information about the sentiment of the text, entities mentioned in the text, or relationships between those entities.
- Depending upon the task at hand, we deal with such characters differently.
- Preprocessing includes
  - Convert the text to lower case, so that words like “write” and “Write” are considered the same word for analysis
  - Remove numbers
  - Remove English stopwords e.g “the”, “is”, “of”, etc
  - Remove punctuation e.g “,”, “?”, etc
  - Eliminate extra white spaces
  - Stemming our text

GSA

D2D
DATA TO DECISIONS

Text Processing Vocabulary

- NLP – natural language processing
- Corpus (pl. corpora) – large structured set of texts
- Stopwords – words (such as “the”, “is”, “etc”) that are ignored by search engines
- Stemming - reduce “infected”/derived word to their stem, e.g. “cars” to “car”
- Lemmatization – distinguish “saw” – hand tool – from “saw” past form of “see”, remove “inflectional” endings to return the base of a word known as lemma
- N-grams – continuous sequence of n items from text, e.g.: phonems, syllables, letters, words, base pairs
- Tokenization – breaking of texts (strings) into pieces (words, sentences, phases, symbols - tokens) for further analysis.

GSA

D2D
DATA TO DECISIONS

Document Term Matrix

- DTM is a matrix that lists all occurrences of words in the corpus, by document
- The documents are represented by rows and the terms (or words) by columns
- If a word occurs in a particular document, then the matrix entry for corresponding to that row and column is 1 (2 if twice, and so on), else it is 0
- Example: Assume we have a simple corpus consisting of two documents, Doc1 and Doc2, with the following content:
  - *Doc1*: bananas are yellow
  - *Doc2*: bananas are good

  The DTM for this corpus would look like:

	bananas	*are*	*yellow*	*good*
Doc1	1	1	1	0
Doc2	1	1	0	1

- If we transpose rows and columns we will get TDM

Text Analytics Steps

- Create Corpus
- Preprocess
- Optional: Lemmatization
- Create TDM (or DTM) data frame
  - Rows - document number
  - Columns – words
  - Cells – frequency of a word in a doc
- Generate 1-gram, bigram and trigram matrices
- Analyze
  - Create summaries
  - Compute Statistics
  - Visualize
    - Histograms
    - Word clouds

Sentiment Analysis Steps

- Create Corpus
- Preprocess
- Stem text
- Optional: Lemmatization
- Create a subset (training data)
  - ➢Manually assign sentiment (e.g. positive Vs. negative (or use sentiment vocabulary)
  - ➢Create sentiment score table:
    - Rows - document number
    - Columns – words
    - Cells – score(s) for each word: how often it shows in negative/positive documents
  - ➢Test the scores on the test data to obtain confidence level
- Apply sentiment scores to all documents for overall results

GSA

D2D

DATA TO DECISIONS

Text Analytics: Wordcloud

```
# i have a dream wordcloud

# Load
library("tm")
library("SnowballC")
library("wordcloud")
library("RColorBrewer")

# Read the text file from internet
filePath <- "http://www.sthda.com/sthda/RDoc/example-files/martin-luther-king-i-have-a-dream-speech.txt"
text <- readLines(filePath)
# Load the data as a corpus
docs <- Corpus(VectorSource(text))
inspect(docs)

# text transformation replacing "/", "@" and "|" with space:
toSpace <- content_transformer(function (x , pattern ) gsub(pattern, " ", x))
docs <- tm_map(docs, toSpace, "/")
docs <- tm_map(docs, toSpace, "@")
docs <- tm_map(docs, toSpace, "\\|")

# text cleaning
# Convert the text to lower case
docs <- tm_map(docs, content_transformer(tolower))
# Remove numbers
docs <- tm_map(docs, removeNumbers)
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("english"))
# Remove your own stop word
# specify your stopwords as a character vector
docs <- tm_map(docs, removeWords, c("blabla1", "blabla2"))
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
# Text stemming
# docs <- tm_map(docs, stemDocument)

# build tdm
dtm <- TermDocumentMatrix(docs)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d, 10)

# generate wordcloud
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
        max.words=200, random.order=FALSE, rot.per=0.35,
        colors=brewer.pal(8, "Dark2"))
```

GSA

D2D
DATA TO DECISIONS

R References

- https://www.tidytextmining.com/tidytext.html
- http://garrettgman.github.io/tidying/
- https://www.r-bloggers.com/intro-to-text-analysis-with-r/
- https://eight2late.wordpress.com/2015/05/27/a-gentle-introduction-to-text-mining-using-r/
  - ➢Required packages: tm, SnowballC, ggplot2, wordcloud
- Tutorial: https://www.springboard.com/blog/text-mining-in-r/ w/ Hillary's emails
  - ➢Packages for tutorial
    RSQLite, 'SQLite' Interface for R
    tm, framework for text mining applications
    SnowballC, text stemming library
    Wordcloud, for making wordcloud visualizations
    Syuzhet, text sentiment analysis
    ggplot2, one of the best data visualization libraries
    quanteda, N-grams
  - ➢Datasource: https://www.kaggle.com/kaggle/hillary-clinton-emails
- Quanteda
  - ➢https://cran.r-project.org/web/packages/quanteda/vignettes/quickstart.html

GSA

D2D
DATA TO DECISIONS

Q & A