Ve572 Lecture 6

Manuel and Jing

UM-SJTU Joint Institute

June 5, 2018

Q: Anyone uses twitter? Anyone follow Donald Trump's twitter account?

- There was a claim about his tweets circulating in 2016:

Todd Vaziri

> *Every non-hyperbolic tweet is from iPhone.*
>
> *i.e. his staff handled*
>
> *Every hyperbolic tweet is from Android.*
>
> *i.e. really from him*


```r
> library(twitteR)
> # access to the twitter API.
> consumer_key = "your_consumer_key"
> consumer_secret = "your_consumer_secret"
> access_token = "your_access_token"
> access_secret = "your_access_secret"
> setup_twitter_oauth(consumer_key, consumer_secret,
+                     access_token, access_secret)
```

```
> library(dplyr)
>
> trump_tweets =
+   userTimeline("realDonaldTrump", n = 5)
> class(trump_tweets)
```

```
[1] "list"
```

```
> (trump_tweets_tb =
+   as_tibble(
+     purrr::map_dfr(trump_tweets, as.data.frame)))
```

```
# A tibble: 5 x 16
  text          favorited favoriteCount replyToSN created             truncated
  <chr>         <lgl>             <dbl> <lgl>     <dttm>              <lgl>
1 Secretary Po? FALSE            53953. NA        2018-05-09 12:35:51 TRUE
2 I am pleased? FALSE            77505. NA        2018-05-09 12:30:56 TRUE
3 Congratulati? FALSE            32838. NA        2018-05-09 12:00:30 TRUE
4 Candace Owen? FALSE            62859. NA        2018-05-09 11:48:17 TRUE
5 The Fake New? FALSE            56652. NA        2018-05-09 11:38:45 TRUE
# ... with 10 more variables: replyToSID <lgl>, id <chr>, replyToUID <lgl>,
#   statusSource <chr>, screenName <chr>, retweetCount <dbl>, isRetweet <lgl>,
#   retweeted <lgl>, longitude <lgl>, latitude <lgl>
```

```
> rm(list = ls())
> # Tweets in 2016
> load("~/Desktop/trump_tweets.rda")
> # load R object that was saved
> trump_tweets_tb
```

```
# A tibble: 1,512 x 16
  text           favorited favoriteCount replyToSN created             truncated
  <chr>          <lgl>             <dbl> <chr>     <dttm>              <lgl>
1 My economic ? FALSE             9214. NA        2016-08-08 15:20:44 FALSE
2 Join me in F? FALSE             6981. NA        2016-08-08 13:28:20 FALSE
# ... with 1,510 more rows, and 10 more variables: replyToSID <lgl>, id <chr>,
#   replyToUID <chr>, statusSource <chr>, screenName <chr>, retweetCount <dbl>,
#   isRetweet <lgl>, retweeted <lgl>, longitude <chr>, latitude <chr>
```

```
> (sel_tb =
+      select(trump_tweets_tb,
+             id, statusSource, text, created))
```

```
# A tibble: 1,512 x 4
  id                 statusSource          text              created
  <chr>              <chr>                 <chr>             <dttm>
1 762669882571980801 "<a href=\"http://tw? My economic pol? 2016-08-08 15:20:44
2 762641595439190016 "<a href=\"http://tw? Join me in Faye? 2016-08-08 13:28:20
# ... with 1,510 more rows
```

```
> head(sel_tb$statusSource)
```

```
[1] "<a href=\".../android\" rel=\"nofollow\">Twitter for Android</a>"
[2] "<a href=\".../iphone\" rel=\"nofollow\">Twitter for iPhone</a>"
[3] "<a href=\".../iphone\" rel=\"nofollow\">Twitter for iPhone</a>"
[4] "<a href=\".../android\" rel=\"nofollow\">Twitter for Android</a>"
[5] "<a href=\".../android\" rel=\"nofollow\">Twitter for Android</a>"
[6] "<a href=\".../android\" rel=\"nofollow\">Twitter for Android</a>"
```

```
> (ext_tb =
+      extract(sel_tb,
+              col = statusSource,
+              into = "source",
+              regex = "Twitter for (.*?)<"))
```

```
# A tibble: 1,512 x 4
  id                 source  text                          created
  <chr>              <chr>   <chr>                         <dttm>
1 762669882571980801 Android My economic policy speech wil? 2016-08-08 15:20:44
2 762641595439190016 iPhone  Join me in Fayetteville, Nort? 2016-08-08 13:28:20
# ... with 1,510 more rows
```

```
> unique(ext_tb$source)
```

```
[1] "Android" "iPhone"  NA         "iPad"
```

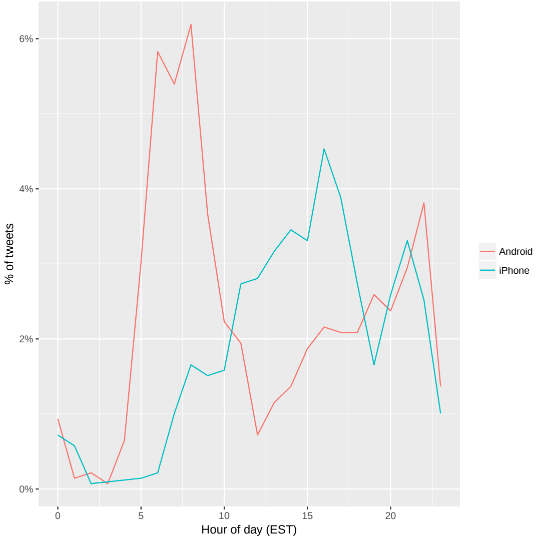
```
> trump_tidy_tb =
+   filter(ext_tb,
+           source %in% c("iPhone", "Android"))
>
> by_source = group_by(trump_tidy_tb, source)
> summarise(by_source, freq = n())
```

```
# A tibble: 2 x 2
  source    freq
  <chr>    <int>
1 Android    762
2 iPhone     628
```

- In practice, you would do those steps in one chunk using a compact syntax

```
> trump_tidy_tb = trump_tweets_tb %>%
+   select(id, statusSource, text, created) %>%
+   extract(statusSource, "source",
+           "Twitter for (.*?)<") %>%
+   filter(source %in% c("iPhone", "Android"))
```

- Investigating the relation between % of tweets by source and time, we have



```r
> trump_tidy_tb %>%
+   count(source, hour =
+           lubridate::hour(
+             lubridate::with_tz(
+               created, "EST"))) %>%
+   mutate(percent = n / sum(n)) %>%
+   ggplot(
+     aes(hour,
+         percent, color = source)
+     ) +
+   geom_line() +
+   scale_y_continuous(
+     labels = scales::percent_format()
+     ) +
+   labs(x = "Hour of day (EST)",
+        y = "% of tweets",
+        color = "")
```

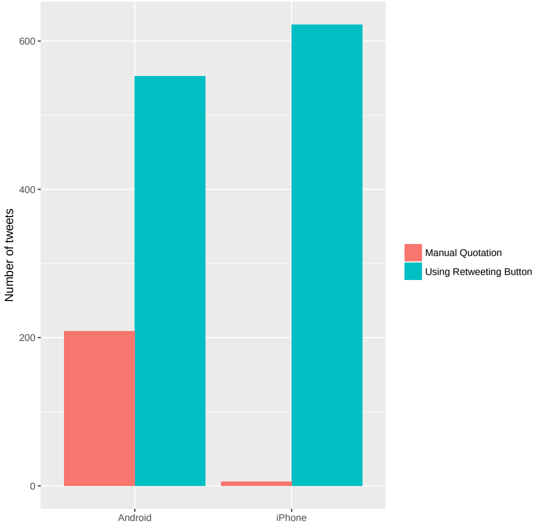
- Another place we can spot a difference is in Trump's tendency of

  "manually retweeting"

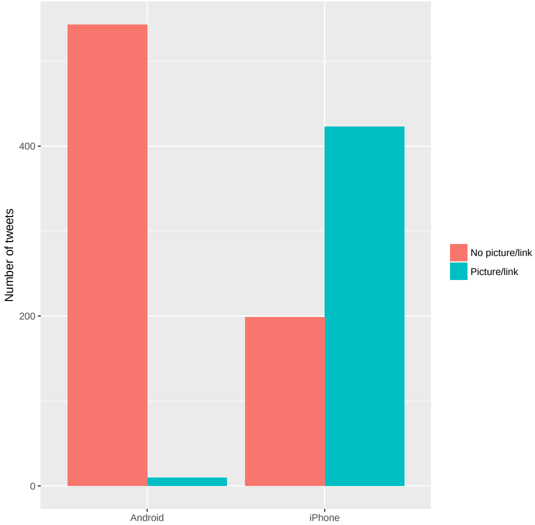
  others by copying-pasting, then putting them under quotation marks.

```
> library(stringr) # Better handling on strings
> tweet_quotation_counts = trump_tidy_tb %>%
+   count(source, quotation =
+           ifelse(str_detect(text, '^"'),
+             "Manual Quotation",
+             "Using Retweeting Button"))
>
> ggplot(tweet_quotation_counts,
+        aes(source, n, fill = quotation)) +
+   geom_bar(stat = "identity",
+            position = "dodge") +
+   labs(x = "", y = "Number of tweets", fill = "")
```

- Almost all of those quoted tweets are posted from the android device.



- Another difference involves sharing links or pictures in tweets.



```
> tweet_picture_counts = trump_tidy_tb %>%
+   filter(!str_detect(text, '^"')) %>%
+ # we have to remove retweeting cases
+ # that were done manually
+   count(source, picture =
+           ifelse(str_detect(text, "t.co"),
+             "Picture/link", "No picture/link")
+         )
> # twitter uses the domain https://t.co/
> # for all pictures and links, e.g.
> trump_tidy_tb$text[2]
```

```
[1] "Join me in Fayetteville, North Carolina tomorrow evening at 6pm. Tickets now
available at: https://t.co/Z80d4MYIg8"
```

```
> ggplot(tweet_picture_counts,
+        aes(source, n, fill = picture)) +
+   geom_bar(stat = "identity",
+            position = "dodge") +
+   labs(x = "", y = "Number of tweets", fill = "")
```

Q: What were the most common words in Trump's tweets overall?

```
> library(tidytext) # Good for tidy up strings
> reg = "([^A-Za-z\\d#@']|'(?![A-Za-z\\d#@]))"
> # Separator between words in his tweets

> trump_words = trump_tidy_tb %>%
+   filter(!str_detect(text, '^"')) %>%
+   mutate(text = str_replace_all(
+     text, "https://t.co/[A-Za-z\\d]+|&amp;",
+     "")) %>%
+   # remove all pictures and links
+   unnest_tokens(word, text,
+                 token = "regex",
+                 pattern = reg) %>%
+   # split sentences into words
+   filter(!word %in% stop_words$word,
+          str_detect(word, "[a-z]"))
>   # keep only relevant words
```

```
> trump_words
```

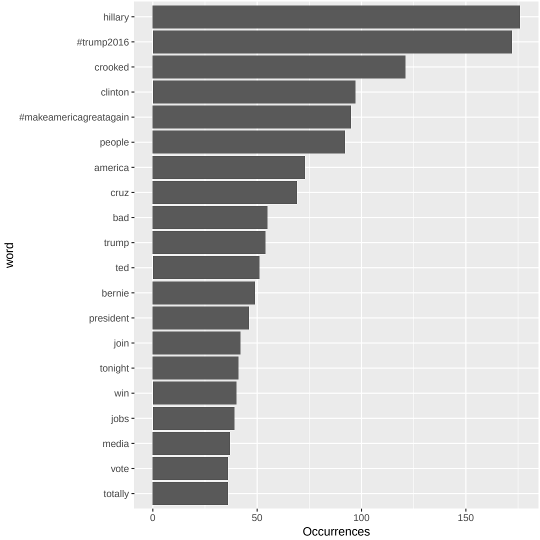
```
# A tibble: 8,753 x 4
  id                 source created             word
  <chr>              <chr>  <dttm>              <chr>
1 676494179216805888 iPhone 2015-12-14 20:09:15 record
2 676494179216805888 iPhone 2015-12-14 20:09:15 health
# ... with 8,751 more rows
```

```
> trump_tidy_tb
```

```
# A tibble: 1,390 x 4
  id                 source  text                           created
  <chr>              <chr>   <chr>                          <dttm>
1 762669882571980801 Android My economic policy speech wil? 2016-08-08 15:20:44
2 762641595439190016 iPhone  Join me in Fayetteville, Nort? 2016-08-08 13:28:20
# ... with 1,388 more rows
```

```
> trump_words %>%
+    count(word, sort = TRUE) %>%
+    head(20) %>%
+    mutate(word = reorder(word, n)) %>%
+    ggplot(aes(word, n)) +
+    geom_bar(stat = "identity") +
+    ylab("Occurrences") +
+    coord_flip()
```

- Recall the data is for 2016, so no surprise there!



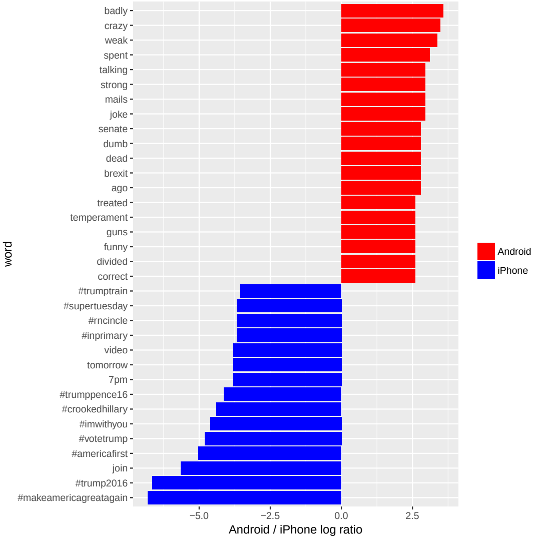
- We sort words according whether it is more likely to coming from an android

$$\log\left(\frac{\frac{\#\text{ Android}+1}{\text{Total Android}+1}}{\frac{\#\text{ iPhone}+1}{\text{Total iPhone}+1}}\right)$$

```
> android_iphone_ratios = trump_words %>%
+   count(word, source) %>%
+   # count the occurence of a word by source
+   spread(source, n, fill = 0) %>%
+   # convert source into two columns
+   mutate_at(c("Android", "iPhone"),
+             funs((. + 1) / sum(. + 1))) %>%
+   # apply a function two both columns
+   mutate(logratio = log2(Android / iPhone)) %>%
+   # create a new column
+   arrange(desc(logratio))
```

```
> android_iphone_ratios %>%
+   group_by(logratio > 0) %>%
+   top_n(15, abs(logratio)) %>%
+   # 15 posistive and 15 negative
+   ungroup() %>%
+   mutate(word = reorder(word, logratio)) %>%
+   # sort word according to logratio
+   ggplot(aes(word,
+              logratio,
+              fill = logratio < 0)) +
+   geom_bar(stat = "identity") +
+   coord_flip() +
+   ylab("Android / iPhone log ratio") +
+   scale_fill_manual(
+     name = "", labels = c("Android", "iPhone"),
+     values = c("red", "blue"))
```

Q: What can we say based on the following plot?



- The NRC emotion lexicon, which comes with `library(tidytext)`,

```
> (nrc = sentiments %>%
+    filter(lexicon == "nrc") %>%
+    select(word, sentiment))
```

```
# A tibble: 13,901 x 2
  word      sentiment
  <chr>     <chr>
1 abacus    trust
2 abandon   fear
3 abandon   negative
4 abandon   sadness
# ... with 1.39e+04 more rows
```

is a way to associate common English words to 2 sentiments:

negative or positive

and 8 emotions:

anger, fear, anticipation, trust, surprise, sadness, joy, and disgust

- To measure the sentiment of the Android and iPhone tweets,

```
> trump_words
```

```
# A tibble: 8,753 x 4
  id                 source created             word
  <chr>              <chr>  <dttm>              <chr>
1 676494179216805888 iPhone 2015-12-14 20:09:15 record
2 676494179216805888 iPhone 2015-12-14 20:09:15 health
3 676494179216805888 iPhone 2015-12-14 20:09:15 #makeamericagreatagain
# ... with 8,750 more rows
```

we divide the number of Trump's words into each of the following categories

```
> unique(nrc$sentiment)
```

```
 [1] "trust"     "fear"      "negative"     "sadness"     "anger"
 [6] "surprise"  "positive"  "disgust"      "joy"         "anticipation"
```

```
> (join_tb = inner_join( # Join the two data sets
+    trump_words, nrc, by = "word"))
```

```
# A tibble: 5,109 x 5
  id                 source created             word     sentiment
  <chr>              <chr>  <dttm>              <chr>    <chr>
1 676509769562251264 iPhone 2015-12-14 21:11:12 accolade anticipation
2 676509769562251264 iPhone 2015-12-14 21:11:12 accolade joy
3 676509769562251264 iPhone 2015-12-14 21:11:12 accolade positive
# ... with 5,106 more rows
```

```r
> # Count the number of sentiment grouped by tweet
> (count_tb = count(join_tb, sentiment, id))
```

```
# A tibble: 3,722 x 3
  sentiment id                     n
  <chr>     <chr>              <int>
1 anger     680503951440121856     1
2 anger     680734915718176768     1
3 anger     685490467329425408     1
# ... with 3,719 more rows
```

```r
> # Add all possible combination of id and sentiment
> (complete_tb =
+      complete(count_tb,
+               sentiment, id, fill = list(n = 0)))
```

```
# A tibble: 8,790 x 3
  sentiment id                     n
  <chr>     <chr>              <dbl>
1 anger     676509769562251264    0.
2 anger     680496083072593920    0.
3 anger     680503951440121856    1.
# ... with 8,787 more rows
```

- This dataset gives the counts of the 10 categories for each tweet.

```r
> # Create a data set on tweets by source
> # One row for each of his tweets
> (sources = trump_words %>%
+      group_by(source) %>%
+      mutate(total = n()) %>%
+      # create a new variable
+      # total number of iPhone/Android
+      ungroup() %>%
+      distinct(id, source, total))
```

```
# A tibble: 1,172 x 3
  id                 source  total
  <chr>              <chr>   <int>
1 676494179216805888 iPhone   3852
2 676509769562251264 iPhone   3852
3 680496083072593920 Android  4901
# ... with 1,169 more rows
```

```r
> length(unique(trump_words$id))
```

```
[1] 1172
```

```
> # Put source back into the dataset
> (complete_with_source_tb =
+   inner_join(complete_tb, sources))
```

```
Joining, by = "id"
# A tibble: 8,790 x 5
  sentiment id                     n source  total
  <chr>     <chr>              <dbl> <chr>   <int>
1 anger     676509769562251264    0. iPhone   3852
2 anger     680496083072593920    0. Android  4901
3 anger     680503951440121856    1. Android  4901
# ... with 8,787 more rows
```

```
> # counts by source and sentiment
> words_by_source_sentiment =
+   complete_with_source_tb %>%
+   group_by(source, sentiment, total) %>%
+   summarize(counts = sum(n)) %>%
+   ungroup()
>
> words_tidy_source_sentiment
```

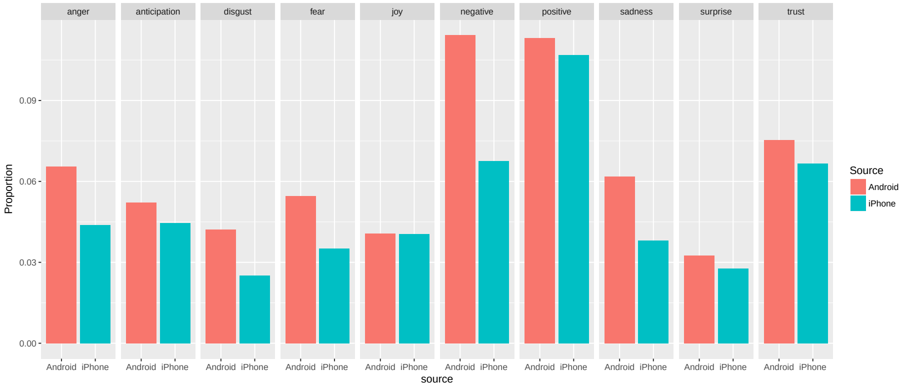
- Everything together in a single chunk

```
> sources = trump_words %>%
+   group_by(source) %>%
+   mutate(total = n()) %>%
+   ungroup() %>%
+   distinct(id, source, total)
>
> words_by_source_sentiment = trump_words %>%
+   inner_join(nrc, by = "word") %>%
+   count(sentiment, id) %>%
+   ungroup() %>%
+   complete(sentiment, id, fill = list(n = 0)) %>%
+   inner_join(sources) %>%
+   group_by(source, sentiment, total) %>%
+   summarize(counts = sum(n)) %>%
+   ungroup()
>
> words_tidy_source_sentiment
```

```
# A tibble: 20 x 4
   source  sentiment    total counts
   <chr>   <chr>        <int>  <dbl>
 1 Android anger         4901   321.
 2 Android anticipation  4901   256.
 3 Android disgust       4901   207.
 4 Android fear          4901   268.
 5 Android joy           4901   199.
 6 Android negative      4901   560.
 7 Android positive      4901   555.
 8 Android sadness       4901   303.
 9 Android surprise      4901   159.
10 Android trust         4901   369.
11 iPhone  anger         3852   169.
12 iPhone  anticipation  3852   172.
13 iPhone  disgust       3852    97.
14 iPhone  fear          3852   135.
15 iPhone  joy           3852   156.
16 iPhone  negative      3852   260.
17 iPhone  positive      3852   412.
18 iPhone  sadness       3852   147.
19 iPhone  surprise      3852   107.
20 iPhone  trust         3852   257.
```

```
> ggplot(words_by_source_sentiment, aes(
+   source, counts/total, fill = source)) +
+   geom_bar(stat = "identity", position = "dodge") +
+   labs(y = "Proportion", fill = "Source") +
+   facet_grid(~sentiment)
```

- It seems there is a clear difference in the category “negative.

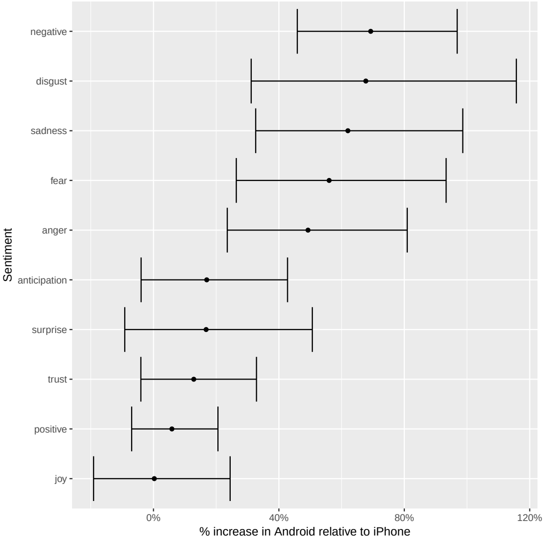


- However, graphical analysis alone is not enough for other categories.
- This is a count data, so let us test by assuming Poisson assumptions.

```
> sentiment_differences
+   words_by_source_sentiment %>%
+   group_by(sentiment) %>%
+   do(broom::tidy(poisson.test(.$counts, .$total)))
>
> sentiment_differences
```

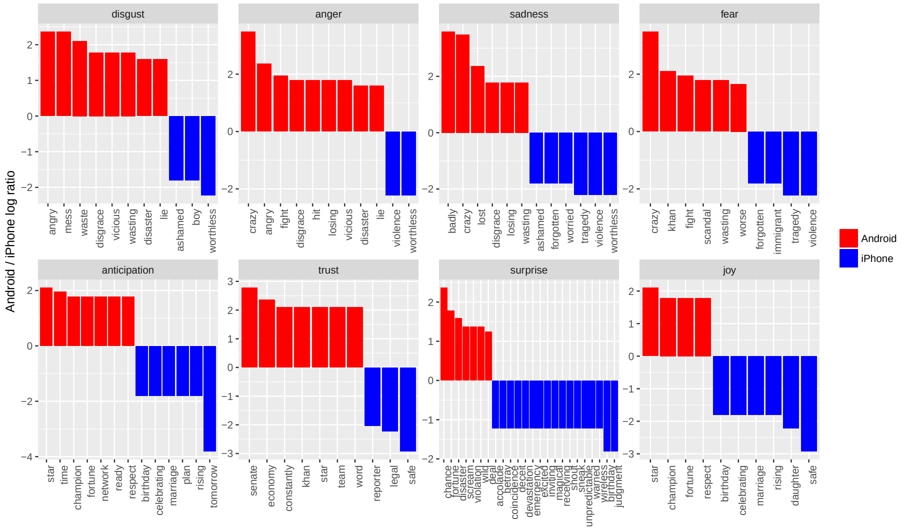
```
# A tibble: 10 x 9
# Groups:   sentiment [10]
   sentiment  estimate statistic  p.value parameter conf.low conf.high method
   <chr>         <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl> <fct>
 1 anger          1.49      321. 2.19e- 5      274.    1.24       1.81 Compari?
 2 anticipat?     1.17      256. 1.19e- 1      240.    0.960      1.43 Compari?
 3 disgust        1.68      207. 1.78e- 5      170.    1.31       2.16 Compari?
 4 fear           1.56      268. 1.89e- 5      226.    1.26       1.93 Compari?
 5 joy            1.00      199. 1.00e+ 0      199.    0.809      1.24 Compari?
 6 negative       1.69      560. 7.09e-13      459.    1.46       1.97 Compari?
 7 positive       1.06      555. 3.82e- 1      541.    0.930      1.21 Compari?
 8 sadness        1.62      303. 1.15e- 6      252.    1.33       1.99 Compari?
 9 surprise       1.17      159. 2.17e- 1      149.    0.908      1.51 Compari?
10 trust          1.13      369. 1.47e- 1      351.    0.960      1.33 Compari?
# ... with 1 more variable: alternative <fct>
```

- And we can visualise the difference with a 95% confidence interval:



```r
> sentiment_differences %>%
+   ungroup() %>%
+   mutate(sentiment =
+            reorder(sentiment, estimate)) %>%
+   mutate_at(c("estimate",
+               "conf.low",
+               "conf.high"),
+             funs(. - 1)) %>%
+   ggplot(aes(estimate, sentiment)) +
+   geom_point() +
+   geom_errorbarh(aes(
+     xmin = conf.low, xmax = conf.high)) +
+   scale_x_continuous(
+     labels = scales::percent_format()) +
+   labs(
+     x = "% increase in Android relative to iPhone",
+     y = "Sentiment")
```

Q: Which words in each category are driving those differences?



```r
> android_iphone_ratios %>%
+   inner_join(nrc, by = "word") %>%
+   filter(!sentiment %in%
+            c("positive", "negative")) %>%
+   mutate(sentiment = reorder(sentiment, -logratio),
+          word = reorder(word, -logratio)) %>%
+   group_by(sentiment) %>%
+   top_n(10, abs(logratio)) %>% ungroup() %>%
+   ggplot(aes(
+     word, logratio, fill = logratio < 0)) +
+   facet_wrap(~ sentiment,
+              scales = "free", nrow = 2) +
+   geom_bar(stat = "identity") +
+   theme(axis.text.x =
+           element_text(angle = 90, hjust = 1)) +
+   labs( x = "",
+         y = "Android / iPhone log ratio") +
+   scale_fill_manual(
+     name = "", values = c("red", "blue"),
+     labels = c("Android", "iPhone"))
```

- We have only dealt with small datasets, for which efficiency is not an issue.

```
> system.time({
+   n = 1e3       # number of data points to load
+   dota2items.df =
+     read.table("~/Desktop/purchase_log.csv",
+     sep = ",", header = TRUE,
+     nrows = n) # Max rows
+
+   item.name.df =
+     read.table("~/Desktop/item_ids.csv",
+                sep = ",", header = TRUE,
+                stringsAsFactors = FALSE)
+ })
```

```
  user  system elapsed
 0.004   0.001   0.003
```

```r
> str(dota2items.df, vec.len = 1)
```

```
'data.frame':   1000 obs. of  4 variables:
 $ item_id    : int  44 29 ...
 $ time       : int  -81 -63 ...
 $ player_slot: int  0 0 ...
 $ match_id   : int  0 0 ...
```

```r
> nrow(item.name.df); head(item.name.df)
```

```
[1] 189
  item_id         item_name
1       1             blink
2       2  blades_of_attack
3       3        broadsword
4       4         chainmail
5       5          claymore
6       6 helm_of_iron_will
```

- We need to be careful when the dataset becomes even moderately big!

```
> system.time({
+   dota2items.df = read.table(
+     "~/Desktop/purchase_log.csv",
+     sep = ",", header = TRUE,
+     nrows = 1e7)
+ })
```

```
  user  system elapsed
20.874   0.283  21.791
```

```
> format(object.size(dota2items.df),
+          units = "auto")
```

```
[1] "152.6 Mb"
```

- R was originally designed to be extremely dynamic and flexible.

Static VS Dynamic Software

Dynamic	Static
Flexible	Inflexible
Slow	Fast

which means there are various ways of doing the same job in R.

- Currently, there are three main approaches of manipulating data in R.
  - base
  - dplyr
  - data.table
- We will cover data.table next, which is designed to be efficient for big data.

```
> library(data.table)
```

- Before data.table, here is a textbook example of trading flexibility for speed

```
> m = 1e5; rbenchmark::benchmark(
+   "1.Slow for loop" = {
+     x = NULL
+     for (i in 1:m){x[i] = sqrt(i)}
+   },
+   "2.Preallocation" = {
+     x = double(m);
+     for (i in 1:m){x[i] = sqrt(i)}
+   },
+   "3.Vectorisation" = {
+     x = sqrt(1:m)
+   }, replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
             test replications elapsed relative
3 3.Vectorisation            5   0.003        1
2 2.Preallocation            5   0.414      138
1 1.Slow for loop            5 102.528    34176
```

```r
> rbenchmark::benchmark( # Recall this has 189 rows
+   "Base" = {
+     item.name.df =
+       read.table("~/Desktop/item_ids.csv",
+                  sep = ",", header = TRUE,
+                  stringsAsFactors = FALSE)
+   },
+   "data.table" = {
+     item_name_dt =
+       fread("~/Desktop/item_ids.csv",
+             sep = ",", header = TRUE,
+             stringsAsFactors = FALSE)
+   }, replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
        test replications elapsed relative
2 data.table            5   0.001        1
1       Base            5   0.004        4
```

```r
> rbenchmark::benchmark( # This has 18,193,745 rows
+   "Base" = {              # Significantly bigger!
+     dota2items.df =
+       read.table("~/Desktop/purchase_log.csv",
+                  sep = ",", header = TRUE)
+   },
+   "data.table" = {
+     dota2items_dt =
+       fread("~/Desktop/purchase_log.csv",
+             sep = ",", header = TRUE)
+   },
+   replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
        test replications elapsed relative
2 data.table            5  10.427    1.000
1       Base            5 205.571   19.715
```

```
> rbenchmark::benchmark( # sort the data frame
+   "Base" = {
+     order.base.df =
+       dota2items.df[order(dota2items.df$match_id,
+                           dota2items.df$time,
+                           decreasing = TRUE), ]
+   },

+   "data.table" = {
+     order_dt =
+       dota2items_dt[order(-match_id,-time)]
+   },
+   replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
        test replications elapsed relative
2 data.table            5   5.237    1.000
1       Base            5  11.140    2.127
```

```
> head(order.base.df)
```

```
         item_id time player_slot match_id
18193441     147 2849           1    49999
18193549     158 2768           4    49999
18193482      96 2760           2    49999
18193480      58 2742           2    49999
18193481      24 2742           2    49999
18193548      55 2742           4    49999
```

```
> head(order_dt)
```

```
   item_id time player_slot match_id
1:     147 2849           1    49999
2:     158 2768           4    49999
3:      96 2760           2    49999
4:      58 2742           2    49999
5:      24 2742           2    49999
6:      55 2742           4    49999
```

```r
> rbenchmark::benchmark( # Subset columns
+   "Base" = {
+     col.base.df =
+       order.base.df[, !names(order.base.df)
+                     %in% c("player_slot")]
+   },

+   "data.table" = {
+     col_dt = order_dt[, !"player_slot"]
+   },
+   replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
        test replications elapsed relative
1       Base            5   0.001        1
2 data.table            5   0.229      229
```

```r
> rbenchmark::benchmark( # Add a column
+   "Base" = {
+     col.base.df$time_r =
+       rank(order.base.df$time)
+   },

+   "data.table" = {
+     col_dt[, time_r := rank(time)]
+   },
+   replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
        test replications elapsed relative
1       Base            5  46.444    1.000
2 data.table            5  48.019    1.034
```

```r
> rbenchmark::benchmark( # Subset rows
+   "Base" = {
+     row.base.df =
+       dota2items.df[dota2items.df$player_slot == 0
+                     & dota2items.df$time>0, ]
+   },

+   "data.table" = {
+     row_dt =
+       dota2items_dt[player_slot == 0 & time>0 ]
+   },
+   replications = 5, order = "relative",
+   columns = c("test", "replications",
+               "elapsed", "relative"))
```

```
        test replications elapsed relative
2 data.table            5   1.578    1.000
1       Base            5   2.457    1.557
```

```
> rbenchmark::benchmark(
+   "Base" = {
+     item.counts.df =
+       aggregate(time~item_id,
+                 data = row.base.df,
+                 FUN = length)
+     item.median.time.df =
+       aggregate(time~item_id,
+                 data = row.base.df,
+                 FUN = median)
+     item.summary.base.df =
+       merge(item.name.df,
+             item.counts.df,
+             by = "item_id")
+     item.summary.base.df =
+       merge(item.summary.base.df,
+             item.median.time.df,
+             by = "item_id")
```

```
+       colnames(item.summary.base.df)[-1:-2] =
+         c("counts", "median_time")
+     },
+     "data.table" = {
+       item_counts_dt =
+         row_dt[, length(time), by = item_id ]
+       colnames(item_counts_dt)[2] = "counts"
+       item_median_time_dt =
+         row_dt[, .(median_time = median(time)),
+                 by = item_id ]
+       item_summary_dt =
+         merge(item_name_dt, item_counts_dt,
+                 by = "item_id")
+       item_summary_dt =
+         merge(item_summary_dt, item_median_time_dt,
+                 by = "item_id")
+     },
+     replications = 5, order = "relative",
+     columns = c("test", "replications",
+                 "elapsed", "relative"))
```

```
        test replications elapsed relative
2 data.table            5   0.256    1.000
1       Base            5  16.817   65.691
```

```r
> head(item_summary_dt)
```

```
   item_id         item_name counts median_time
1:       1             blink  14740      1173.0
2:       2   blades_of_attack  26309       518.0
3:       3        broadsword  10834      1445.0
4:       4         chainmail  14214      1310.5
5:       5          claymore   8365      1056.0
6:       6  helm_of_iron_will   6008      1056.0
```

- Another problem with large datasets in R:

R objects live in memory entirely

which means R reads Data into RAM all at once!

- This feature improves speed until there is NOT enough memory.
- Solution:
  1. Subset data before loading into R
  2. Workarounds within R (< 10GB)
  3. Connect and interact with database within R (> 10GB)
  4. Hadoop and Spark (> 10GB)