The general idea is to calculate a *sentiment score* for each tweet so we can know how positive or negative is the posted message.
There are different ways to calculate such scores,
and you can even create your own formula.
We'll use a very simple yet useful approach to define our score formula

$$\text{Score} = \text{Number of positive words} - \text{Number of negative words}$$

If Score > 0, this means that the sentence has an overall 'positive opinion'
If Score < 0, this means that the sentence has an overall 'negative opinion'
If Score = 0, then the sentence is considered to be a 'neutral opinion'

In order to count the number of positive and negative words, we need a very important ingredient: an opinion lexicon in english, which fortunately it is provided by Hu and Liu and it can be accessed from: http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA,

Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th International World Wide Web conference (WWW-2005), May 10-14, 2005, Chiba, Japan.

You can download the text files containing the positive and negative words:
positive_words.txt
negative_words.txt

Another important ingredient, shared by Jeff Breen, is the very handy function to calculate score sentiments: check Breen's github repo on sentiment analysis for more details.

Example: Mood and Drinking

Let me show you a simple example of some of the things we can do with sentiment analysis.
Research Question: What's the mood associated with tweets containing some kind of drink? More specifically, what's the mood associated to drinks such as wine, beer, coffee and soda?

Step 1: Load necessary packages

```
library(twitteR)
library(plyr)
library(stringr)
```

Step 2: Define function score.sentiment

```
# function score.sentiment
score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
   # Parameters
   # sentences: vector of text to score
   # pos.words: vector of words of postive sentiment
   # neg.words: vector of words of negative sentiment
   # .progress: passed to laply() to control of progress bar

   # create simple array of scores with laply
   scores = laply(sentences,
   function(sentence, pos.words, neg.words)
   {
      # remove punctuation
      sentence = gsub("[[:punct:]]", "", sentence)
      # remove control characters
      sentence = gsub("[[:cntrl:]]", "", sentence)
      # remove digits?
```

```
      sentence = gsub('\\d+', '', sentence)

      # define error handling function when trying tolower
      tryTolower = function(x)
      {
         # create missing value
         y = NA
         # tryCatch error
         try_error = tryCatch(tolower(x), error=function(e) e)
         # if not an error
         if (!inherits(try_error, "error"))
         y = tolower(x)
         # result
         return(y)
      }
      # use tryTolower with sapply
      sentence = sapply(sentence, tryTolower)

      # split sentence into words with str_split (stringr package)
      word.list = str_split(sentence, "\\s+")
      words = unlist(word.list)

      # compare words to the dictionaries of positive & negative terms
      pos.matches = match(words, pos.words)
      neg.matches = match(words, neg.words)

      # get the position of the matched term or NA
      # we just want a TRUE/FALSE
      pos.matches = !is.na(pos.matches)
      neg.matches = !is.na(neg.matches)

      # final score
      score = sum(pos.matches) - sum(neg.matches)
      return(score)
      }, pos.words, neg.words, .progress=.progress )

   # data frame with scores for each sentence
   scores.df = data.frame(text=sentences, score=scores)
   return(scores.df)
}
```

Step 3: We need to import the files containing the positive and negative words

```
# import positive and negative words
pos = readLines("positive_words.txt")
neg = readLines("negative_words.txt")
```

Step 4: Let's harvest tweets talking about wine, beer, coffee, and soda

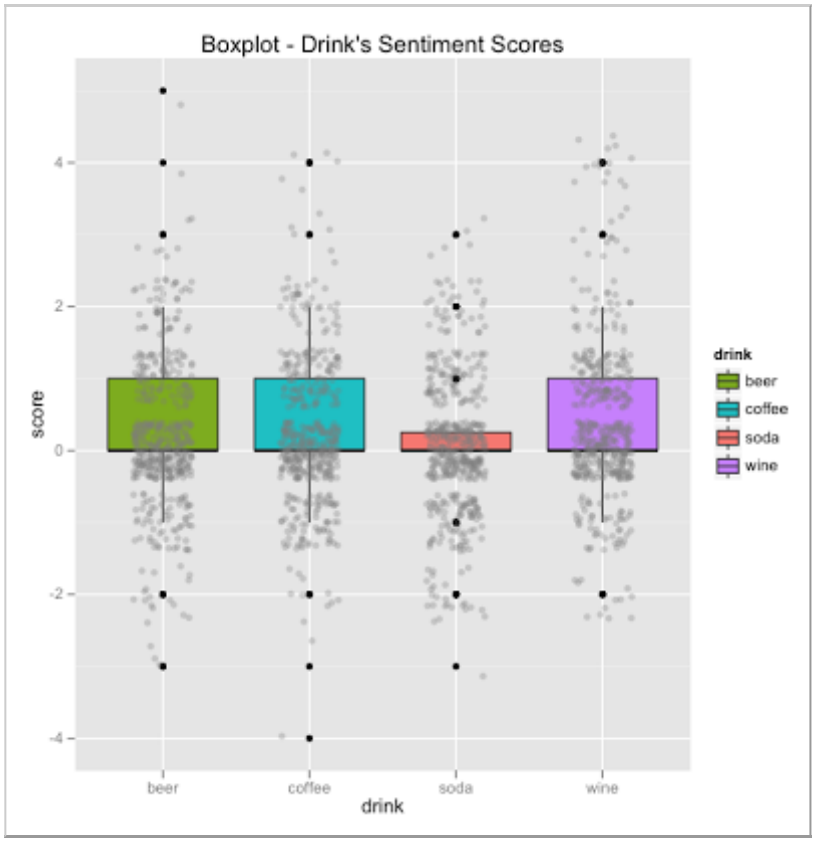
```
# tweets with drinks
wine_tweets = searchTwitter("wine", n=500, lang="en")
beer_tweets = searchTwitter("beer", n=500, lang="en")
cofe_tweets = searchTwitter("coffee", n=500, lang="en")
soda_tweets = searchTwitter("soda", n=500, lang="en")

# get text
wine_txt = sapply(wine_tweets, function(x) x$getText())
beer_txt = sapply(beer_tweets, function(x) x$getText())
cofe_txt = sapply(cofe_tweets, function(x) x$getText())
soda_txt = sapply(soda_tweets, function(x) x$getText())

# how many tweets of each drink
nd = c(length(wine_txt), length(beer_txt), length(cofe_txt), length(soda_txt))

# join texts
drinks = c(wine_txt, beer_txt, cofe_txt, soda_txt)
```

Step 5: Apply score.sentiment and calculate more results

```
# apply function score.sentiment
scores = score.sentiment(ari_txt, pos, neg, .progress='text')

# add variables to data frame
scores$drink = factor(rep(c("wine", "beer", "coffee", "soda"), nd))
scores$very.pos = as.numeric(scores$score >= 2)
scores$very.neg = as.numeric(scores$score <= -2)

# how many very positives and very negatives
numpos = sum(scores$very.pos)
numneg = sum(scores$very.neg)

# global score
global_score = round( 100 * numpos / (numpos + numneg) )
```

Step 6: Get a boxplot

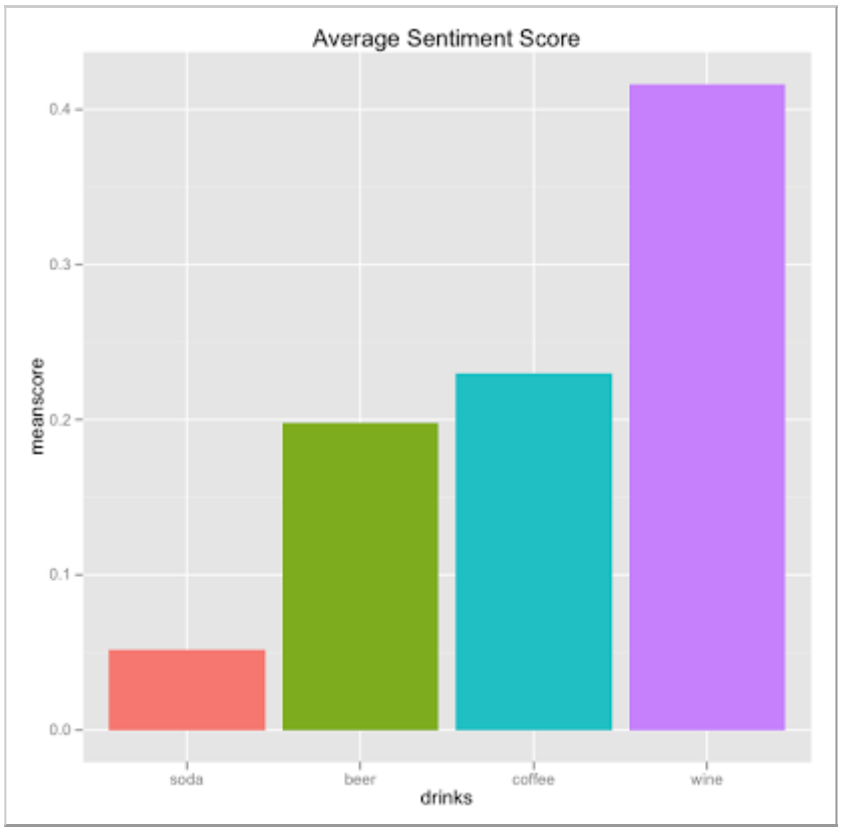
```
# colors
cols = c("#7CAE00", "#00BFC4", "#F8766D", "#C77CFF")
names(cols) = c("beer", "coffee", "soda", "wine")

# boxplot
ggplot(scores, aes(x=drink, y=score, group=drink)) +
geom_boxplot(aes(fill=drink)) +
scale_fill_manual(values=cols) +
geom_jitter(colour="gray40",
    position=position_jitter(width=0.2), alpha=0.3) +
opts(title = "Boxplot - Drink's Sentiment Scores")
```



Step 7: Make some barplots
As you can tell, wine gets the highest sentiment score, while soda the lowest one

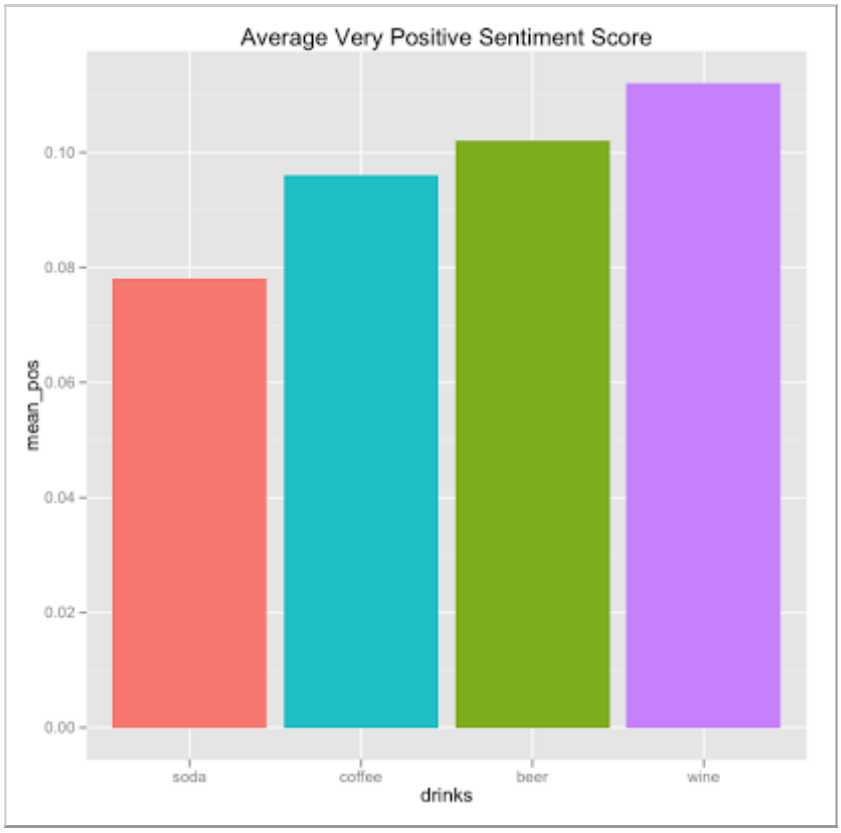
```
# barplot of average score
meanscore = tapply(scores$score, scores$drink, mean)
df = data.frame(drink=names(meanscore), meanscore=meanscore)
df$drinks <- reorder(df$drink, df$meanscore)
```

```
ggplot(df, aes(y=meanscore)) +
geom_bar(data=df, aes(x=drinks, fill=drinks)) +
scale_fill_manual(values=cols[order(df$meanscore)]) +
opts(title = "Average Sentiment Score",
    legend.position = "none")
```



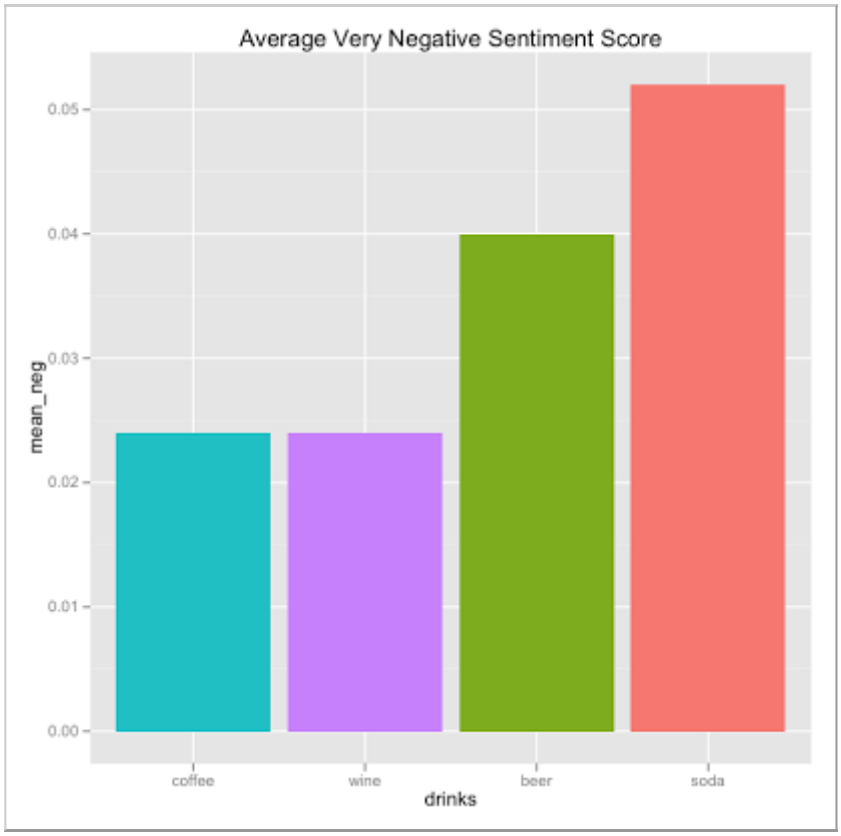
If we examine the very positive scores, we'll see that wine receives the highest values

```
# barplot of average very positive
drink_pos = ddply(scores, .(drink), summarise, mean_pos=mean(very.pos))
drink_pos$drinks <- reorder(drink_pos$drink, drink_pos$mean_pos)

ggplot(drink_pos, aes(y=mean_pos)) +
geom_bar(data=drink_pos, aes(x=drinks, fill=drinks)) +
scale_fill_manual(values=cols[order(drink_pos$mean_pos)]) +
opts(title = "Average Very Positive Sentiment Score",
    legend.position = "none")
```



Conversely, if we check the very negative scores, soda is the one that has the worst score

```
# barplot of average very negative
drink_neg = ddply(scores, .(drink), summarise, mean_neg=mean(very.neg))
drink_neg$drinks <- reorder(drink_neg$drink, drink_neg$mean_neg)

ggplot(drink_neg, aes(y=mean_neg)) +
geom_bar(data=drink_neg, aes(x=drinks, fill=drinks)) +
scale_fill_manual(values=cols[order(drink_neg$mean_neg)]) +
opts(title = "Average Very Negative Sentiment Score",
legend.position = "none")
```


Average Very Negative Sentiment Score
mean_neg
0.05
0.04
0.03
0.02
0.01
0.00
coffee
wine
beer
soda
drinks