# Summarizing a Corpus

Text data is messy and to make sense of it you often have to clean it a bit first. For example, do you want “Tuesday” and “Tuesdays” to count as separate words or the same word? Most of the time we would want to count this as the same word. Similarly with “run” and “running”. Furthermore, we often are not interested in including punctuation in the analysis - we just want to treat the text as a “bag of words”. There are libraries in R that will help you do this.

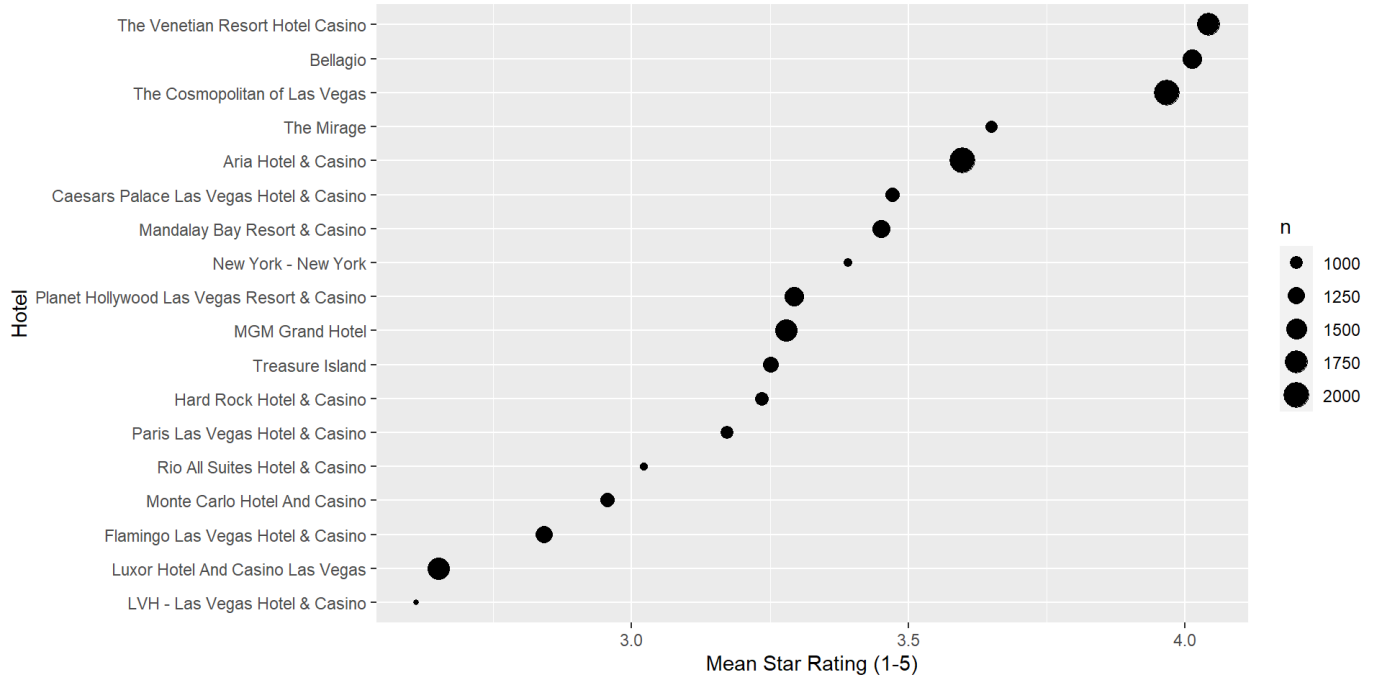
All code and data for this and the other two modules on text mining can be fond in here (https://www.dropbox.com/sh/smgrrechpwlbodg/AAA8nkEzYsJ8T2Pufnv1Hbnra?dl=1).

In the following let’s look at Yelp reviews of Las Vegas hotels:

```
reviews <- read_rds('data/vegas_hotel_reviews.rds')
business <- read_rds('data/vegas_hotels_info.rds')
```

This data contains customer reviews of 18 hotels in Las Vegas. In addition to text, each review also contain a star rating from 1 to 5. Let’s see how the 18 hotels fare on average in terms of star ratings:

```
library(tidyverse)
library(scales)
library(lubridate)

reviews %>%
  left_join(select(business,business_id,name),
            by='business_id') %>%
  group_by(name) %>%
  summarize(n = n(),
            mean.star = mean(as.numeric(stars))) %>%
  arrange(desc(mean.star)) %>%
  ggplot() +
  geom_point(aes(x=reorder(name,mean.star),y=mean.star,size=n))+
  coord_flip() +
  ylab('Mean Star Rating (1-5)') +
  xlab('Hotel')
```




So The Venetian, Bellagio and The Cosmopolitan are clearly the highest rated hotels, while Luxor and LVH are the lowest rated. Ok, but what is behind these ratings? What are customers actually saying about these hotels? This is what we can hope to find through a text analysis.

# Constructing a Document Term Matrix

In the following we will make use of two libraries for text mining:

```
## install packages
install.packages(c("wordcloud","tidytext"))  ## only run once
library(tidytext)
library(wordcloud)
```

Simple text summaries is often based on the **document term matrix**. This is an array where each row corresponds to a document and each column corresponds to a word. The entries of the array are simply counts of how many times a certain word occurs in a certain document. Let's look at a simple example:

```
example <- data.frame(doc_id = c(1:4),
                      text=c("I have a brown dog. My dog loves walks.",
                       "My dog likes food.",
                       "I like food.",
                       "Some dogs are black."),
                      stringsAsFactors = F)
```

This data contains four documents. The first document contains 8 unique words (or "terms"). The word "brown" occurs once while "dog" occurs twice. We can create document-term counts using the tidytext's library *unnest_tokens* and then a simple count:

```
example %>%
  unnest_tokens(word,text) %>%
  count(doc_id,word)
```

```
##    doc_id  word n
## 1       1     a 1
## 2       1 brown 1
## 3       1   dog 2
## 4       1  have 1
## 5       1     i 1
## 6       1 loves 1
## 7       1    my 1
## 8       1 walks 1
## 9       2   dog 1
## 10      2  food 1
## 11      2 likes 1
## 12      2    my 1
## 13      3  food 1
## 14      3     i 1
## 15      3  like 1
## 16      4   are 1
## 17      4 black 1
## 18      4  dogs 1
## 19      4  some 1
```

This data contains four documents. The first document contains 8 unique words (or “terms”). The word “brown” occurs once while “dog” occurs twice. Note that punctuation is removed and lower-casing is done automatically.

We are usually performing text summaries with the goal of understanding what a document or set of documents are about. To this end many of the often used words in the English language are useless - words such as the, and, it and so on. These words are called “stop words” and we can remove them automatically by using an *anti_join* :

```
example %>%
  unnest_tokens(word,text) %>%
  count(doc_id,word) %>%
  anti_join(stop_words)
```

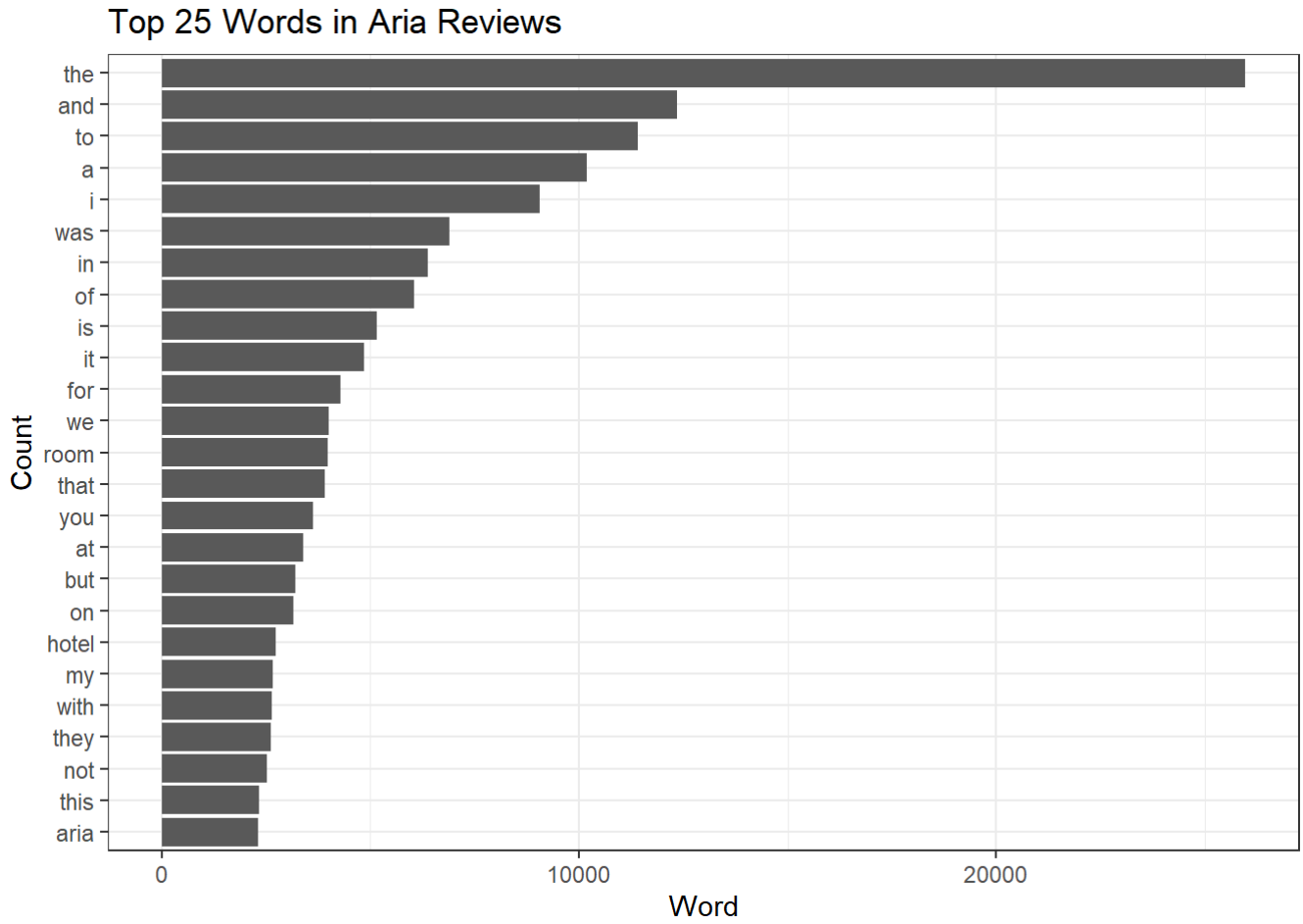
```
##    doc_id  word n
## 1       1 brown 1
## 2       1   dog 2
## 3       1 loves 1
## 4       1 walks 1
## 5       2   dog 1
## 6       2  food 1
## 7       2 likes 1
## 8       3  food 1
## 9       4 black 1
## 10      4  dogs 1
```

We now have a more distilled and interesting representation of each document. Let’s try this out on some real data - the hotel reviews above. Suppose we are interersted in summarizing the reviews for the Aria hotel:

```
aria.id <-  filter(business,
        name=='Aria Hotel & Casino')$business_id
aria.reviews <- filter(reviews,
                       business_id==aria.id)


AriaTidy <- aria.reviews %>%
  select(review_id,text,stars) %>%
  unnest_tokens(word,text)

AriaFreqWords <- AriaTidy %>%
  count(word)
```

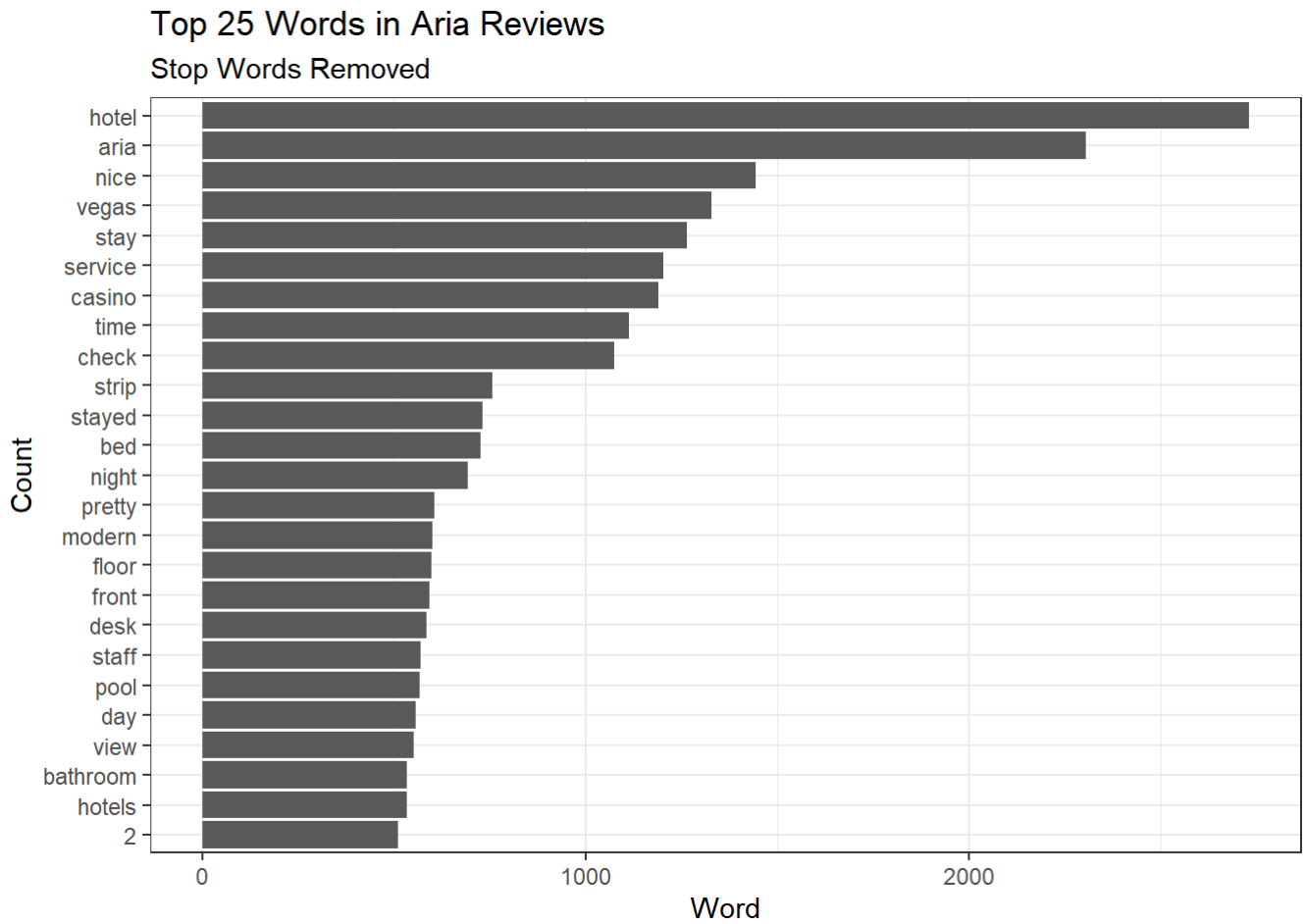
Let's plot the top words:

```
AriaFreqWords %>%
  top_n(25) %>%
  ggplot(aes(x=fct_reorder(word,n),y=n)) + geom_bar(stat='identity') +
  coord_flip() + theme_bw()+
  labs(title='Top 25 Words in Aria Reviews',
       x='Count',
       y= 'Word')
```




ok - so here we see the problem with stop words. Let's remove them:

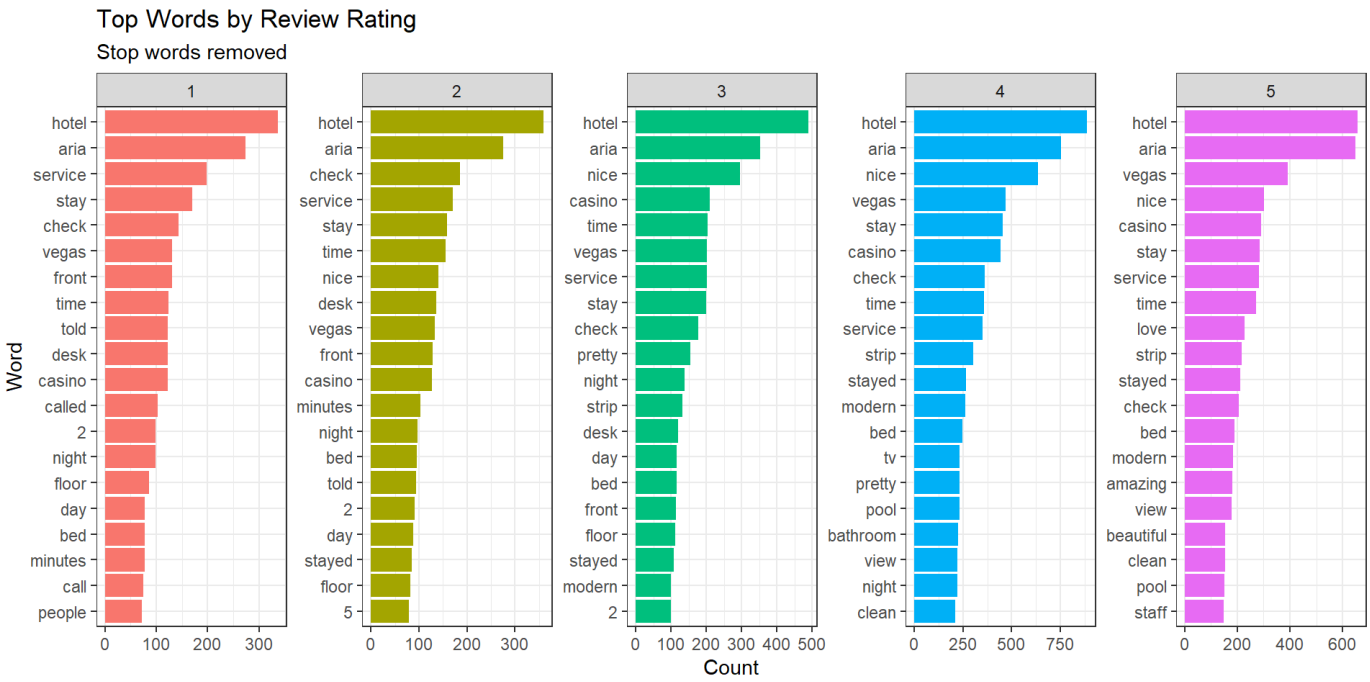
```
AriaFreqWords %>%
  anti_join(stop_words) %>%
  top_n(25) %>%
  ggplot(aes(x=fct_reorder(word,n),y=n)) + geom_bar(stat='identity') +
  coord_flip() + theme_bw()+
  labs(title='Top 25 Words in Aria Reviews',
       subtitle = 'Stop Words Removed',
       x='Count',
       y= 'Word')
```




Much better! But how do these top words vary with the rating of the underlying reviews? This is an easy extension:

```
AriaFreqWordsByRating <- AriaTidy %>%
  count(stars,word)
```

```
AriaFreqWordsByRating %>%
  anti_join(stop_words) %>%
  group_by(stars) %>%
  top_n(20) %>%
  ggplot(aes(x=reorder_within(word,n,stars),
             y=n,
             fill=stars)) +
  geom_bar(stat='identity') +
  coord_flip() +
  scale_x_reordered() +
  facet_wrap(~stars,scales = 'free',nrow=1) +
  theme_bw() +
  theme(legend.position = "none")+
  labs(title = 'Top Words by Review Rating',
       subtitle = 'Stop words removed',
       x = 'Word',
       y = 'Count')
```



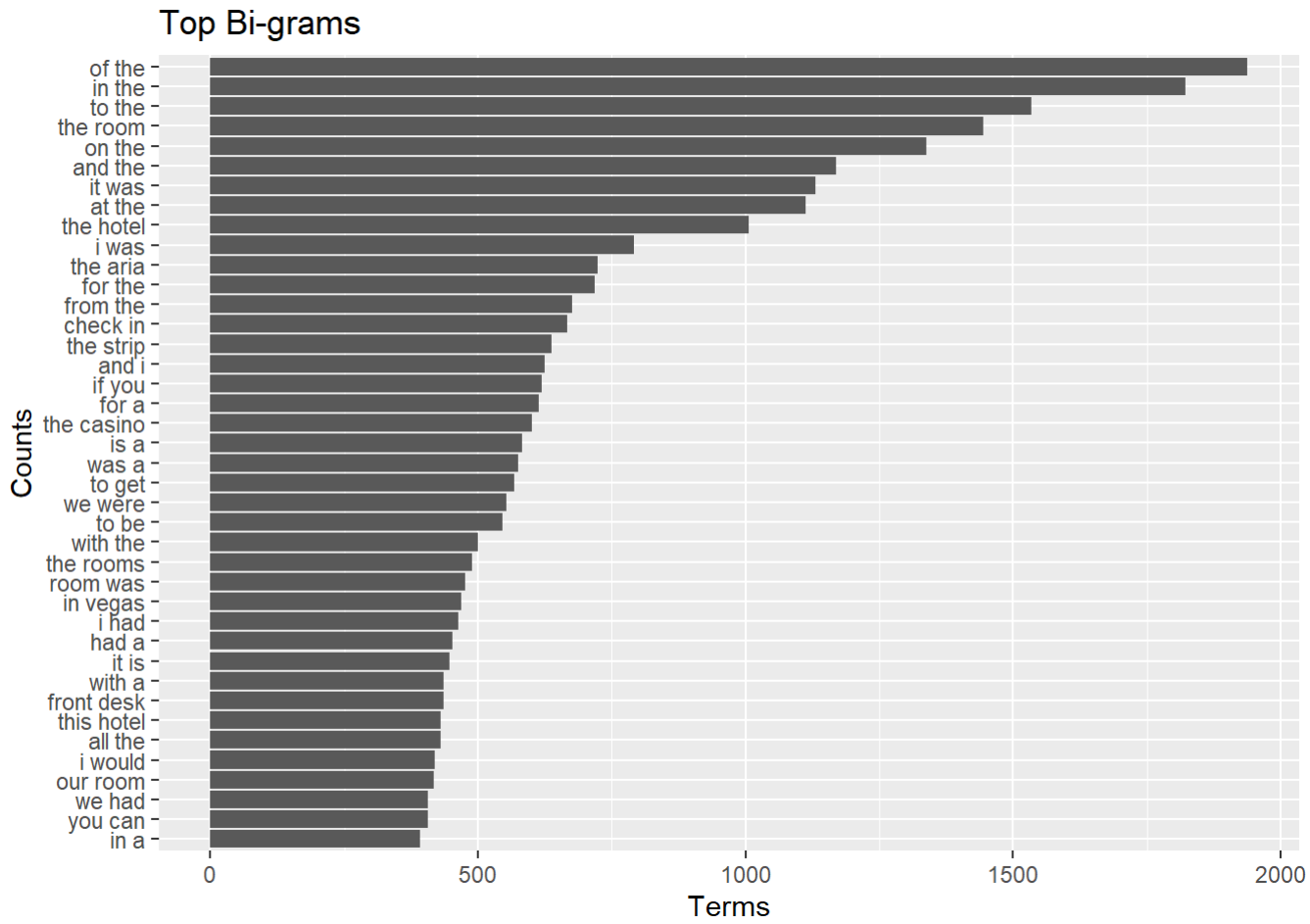

Here we can begin to see what is behind low rated experiences and high rated experiences.

A typical visualization of document term arrays are word clouds. These can easily be done in R:

```
topWords <- AriaFreqWords %>%
  anti_join(stop_words) %>%
  top_n(100)

wordcloud(topWords$word,
          topWords$n,
          scale=c(5,0.5),
          colors=brewer.pal(8,"Dark2"))
```


Above we looked at terms in isolation. Sometimes we may want to summarize terms two at a time. These are called “bi-grams”:

```
aria.reviews %>%
  select(review_id,text) %>%
  unnest_tokens(bigram,text,token="ngrams",n=2) %>%
  count(bigram) %>%
  top_n(40) %>%
  ggplot(aes(x=fct_reorder(bigram,n),y=n)) + geom_bar(stat='identity') +
  coord_flip() +
  labs(title="Top Bi-grams",
       x = "Counts",
       y = "Terms")
```




All of the above analyses looked at overall term frequencies. Sometimes we may have a different objective: What is one given document about? The default metric for this question is a document's Term-Frequency-Inverse Document-Frequency (TF-IDF). A word's TF-IDF is a measure of how important a word is to a document. It is calculated as

$$\text{TF(word)} = \frac{\text{number of times word appears in document}}{\text{total number of words in document}}$$

$$\text{IDF(word)} = \log_e \frac{\text{total number of documents}}{\text{Number of documents with word in it}}$$

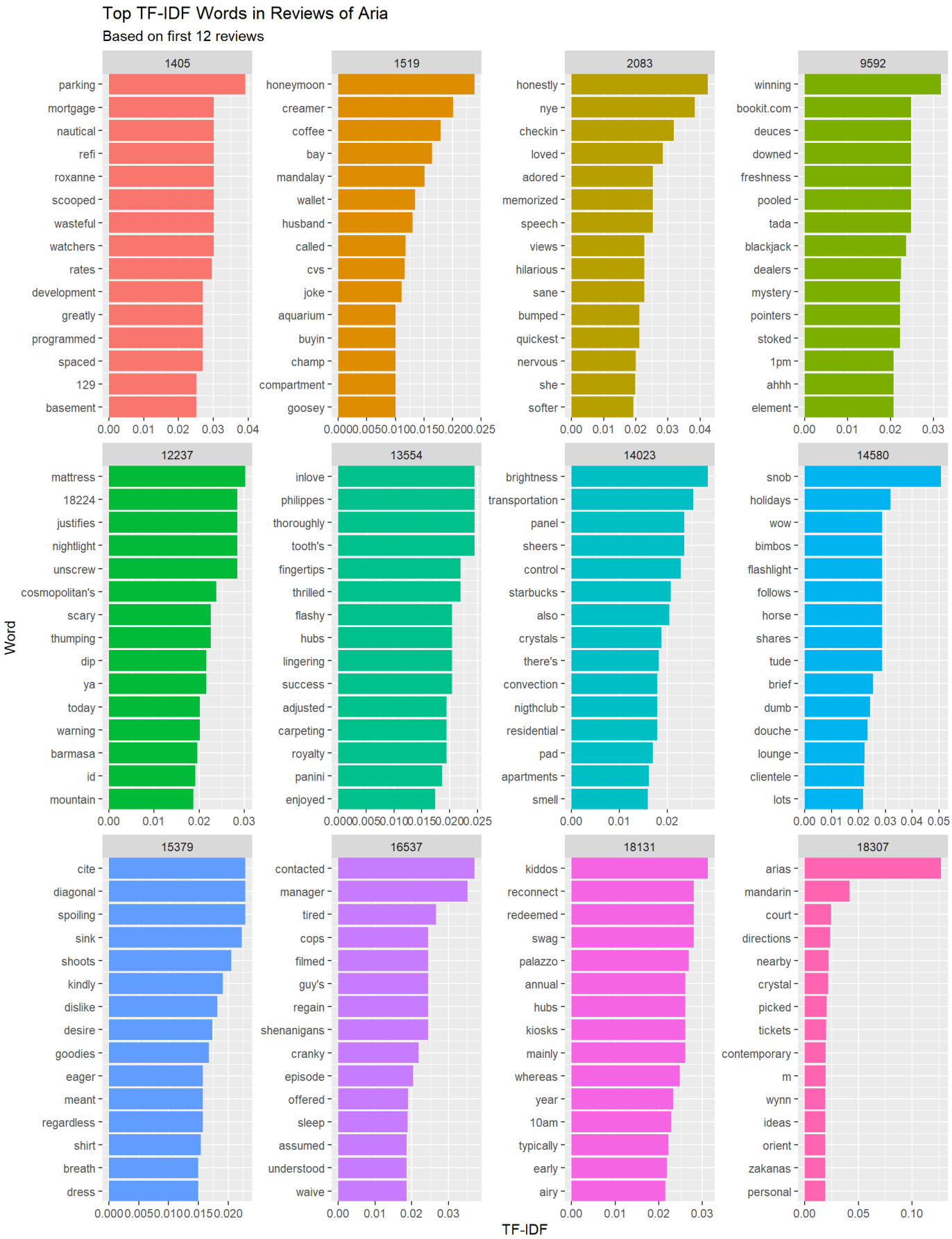
$$\text{TF-IDF(word)} = \text{TF(word)} \times \text{IDF(word)}$$

Words with high TF-IDF in a document will tend to be words that are rare (when compared to other documents) but not too rare. In this way they are informative about what the document is about! We can easily calculate TF-IDF using **bind_tf_idf**:

```
tidyReviews <- aria.reviews %>%
  select(review_id,text) %>%
  unnest_tokens(word, text) %>%
  count(review_id,word)

minLength <- 200  # focus on long reviews
tidyReviewsLong <- tidyReviews %>%
  group_by(review_id) %>%
  summarize(length = sum(n)) %>%
  filter(length >= minLength)

tidyReviewsTFIDF <- tidyReviews %>%
  filter(review_id %in% tidyReviewsLong$review_id) %>%
  bind_tf_idf(word,review_id,n) %>%
  group_by(review_id) %>%
  arrange(desc(tf_idf)) %>%
  slice(1:15) %>% # get top 15 words in terms of tf-idf
  ungroup() %>%
  mutate(xOrder=n():1) %>%  # for plotting
  inner_join(select(aria.reviews,review_id,stars),by='review_id')  # get star ratings
```

Here is a plot of the top TF-IDF words for 12 reviews:

```
nReviewPlot <- 12
plot.df <- tidyReviewsTFIDF %>%
  filter(review_id %in% tidyReviewsLong$review_id[1:nReviewPlot])

plot.df %>%
  mutate(review_id_n = as.integer(review_id)) %>%
  ggplot(aes(x=xOrder,y=tf_idf,fill=factor(review_id_n))) +
  geom_bar(stat = "identity", show.legend = FALSE) +
  facet_wrap(~ review_id_n,scales='free') +
  scale_x_continuous(breaks = plot.df$xOrder,
                     labels = plot.df$word,
                     expand = c(0,0)) +
  coord_flip()+
  labs(x='Word',
       y='TF-IDF',
       title = 'Top TF-IDF Words in Reviews of Aria',
       subtitle = paste0('Based on first ',
                         nReviewPlot,
                         ' reviews'))+
  theme(legend.position = "none")
```




One can think of these as “key-words” for each review.

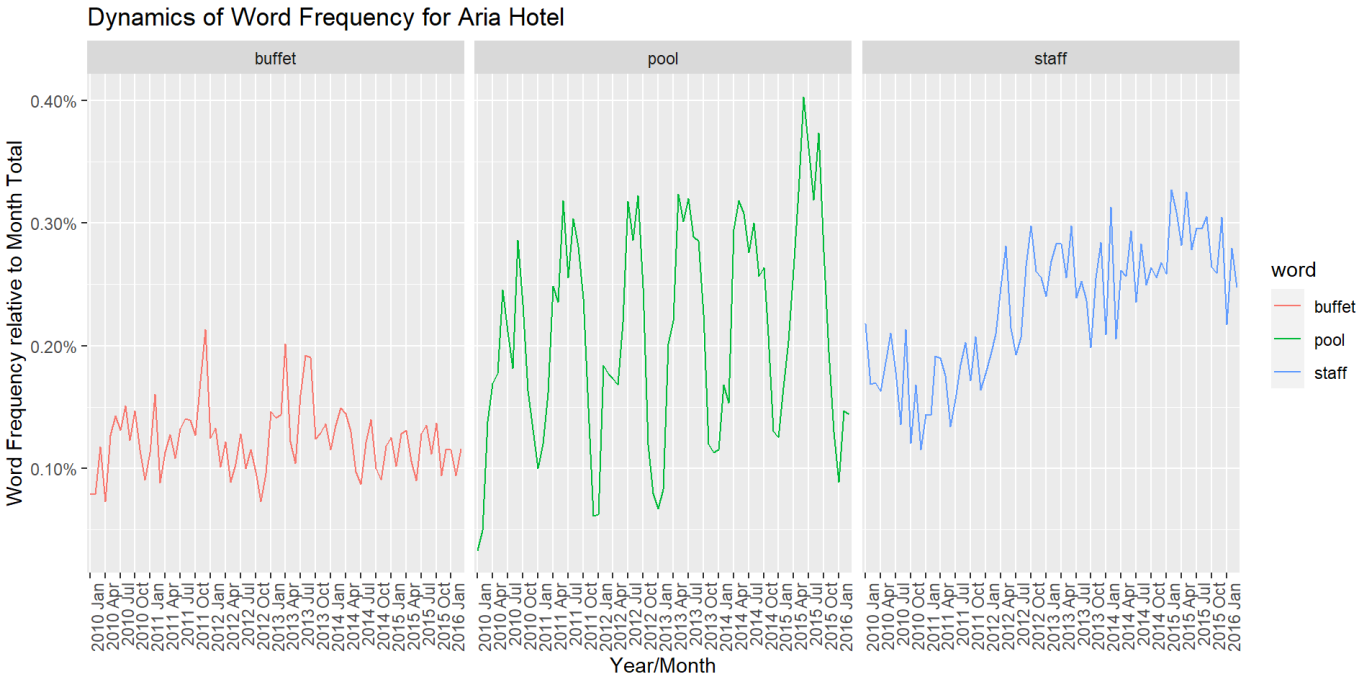
# Aria v2.0

If you want to try out a much bigger database of Aria reviews, you can use the data in the file **AriaReviewsTrip.rds**. This contains 10,905 reviews of the Aria hotel scraped from Tripadvisor.

```
aria <- read_rds('data/AriaReviewsTrip.rds') %>%
  rename(text = reviewText)

meta.data <- aria %>%
  select(reviewID,reviewRating,date,year.month.group)
```

With this sample size you can do more in-depth analyses. For example, how do word frequencies change over time? In the following we focus on the terms “buffet”,“pool” and “staff”. We calculate the relative frequency of these three terms for each month (relative to the total number of terms used that month).

```
ariaTidy <- aria %>%
  select(reviewID,text) %>%
  unnest_tokens(word,text) %>%
  count(reviewID,word) %>%
  inner_join(meta.data,by="reviewID")

total.terms.time <- ariaTidy %>%
  group_by(year.month.group) %>%
  summarize(n.total=sum(n))

## for the legend
a <- 1:nrow(total.terms.time)
b <- a[seq(1, length(a), 3)]

ariaTidy %>%
  filter(word %in% c("pool","staff","buffet")) %>%
  group_by(word,year.month.group) %>%
  summarize(n = sum(n)) %>%
  left_join(total.terms.time, by='year.month.group') %>%
  ggplot(aes(x=year.month.group,y=n/n.total,color=word,group=word)) +
  geom_line() +
  facet_wrap(~word)+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  scale_x_discrete(breaks=as.character(total.terms.time$year.month.group[b]))+
  scale_y_continuous(labels=percent)+xlab('Year/Month')+
  ylab('Word Frequency relative to Month Total')+
  ggtitle('Dynamics of Word Frequency for Aria Hotel')
```




We see three different patterns for the relative frequencies: “buffet” is used in a fairly stable manner over this time period, while “pool” displays clear seasonality, rising in popularity in the summer months. Finally, we see an upward trend in the use of “staff”.

We can try a similar analysis where consider word frequency dynamics for different satisfaction segments:

```
aria.tidy2 <- ariaTidy %>%
  mutate(year = year(date),
         satisfaction = fct_recode(factor(reviewRating),
                                   "Not Satisfied"="1",
                                   "Not Satisfied"="2",
                                   "Neutral"="3",
                                   "Neutral"="4",
                                   "Satisfied"="5"))

total.terms.rating.year <- aria.tidy2 %>%
  group_by(satisfaction,year) %>%
  summarize(n.total = sum(n))


aria.tidy2 %>%
  filter(word %in% c("pool","staff","buffet","food","wait","casino","line","check","c
         lean")) %>%
  group_by(satisfaction,year,word) %>%
  summarize(n = sum(n)) %>%
  left_join(total.terms.rating.year, by=c('year','satisfaction')) %>%
  ggplot(aes(x=year,y=n/n.total,color=satisfaction,group=satisfaction)) +
  geom_line(size=1,alpha=0.25) + geom_point() +
  facet_wrap(~word,scales='free')+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  scale_y_continuous(labels=percent)+xlab('Year')+
  ylab('Word Frequency relative to Month Total')+
  labs(title='Dynamics of Word Frequency for Aria Hotel',
       subtitle='Three Satisfaction Segments')
```