Introduction to Business Analytics

Lecture 8: Text Analytics with R

Iegor Vyshnevskyi

Woosong University

April 17/18, 2023

Agenda

1. Intro to Text Mining
2. Text Mining: Process
3. Text Mining: Techniques
4. Text Mining: Methods
5. Text Mining: Applications
6. Text Mining: Practical use
7. In-class Assignment

1. Intro to Text Mining

Overview

- According to some reports in 2018, around ***2.5 quintillion bytes*** of data are created every day and it's going to increase every year.
- The data we create includes videos, audios, images, ***text*** and many more.
- Data is key to businesses and proper utilization of those data adds value to the organization.
- Data is collected from the customer through social media, emails, text messages and many more on a day to day basis.
- As most of the data such as email, customer feedback, text messages consists of text data -> ***text mining*** is becoming extremely important for *identifying important patterns and trends within text*.

Text as Data

- Textual data provides a means of understanding all human behavior through a data-driven, analytical approach.

Text as Data (cont.)

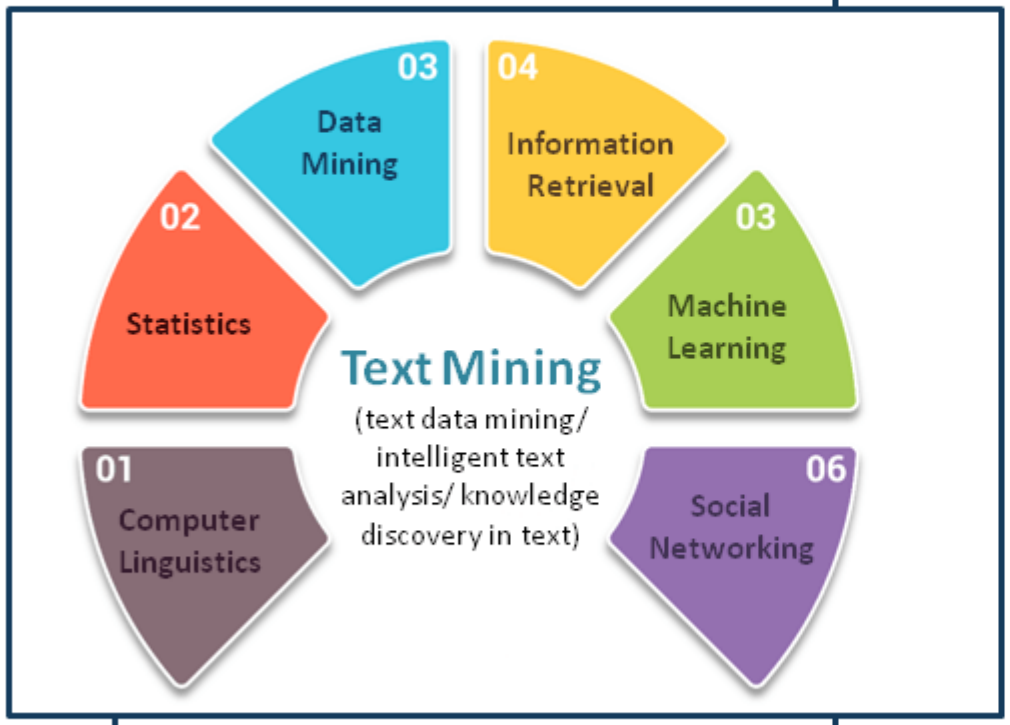
- There are *many reasons* why text has business value:
  - *Big Text*: there is more textual data than numerical data.
  - *Text is versatile*. Nuances and behavioral expressions are not conveyed with numbers, so analyzing text allows us to explore these aspects of human interaction.
  - *Text contains emotive content*. This has led to the ubiquity of "Sentiment analysis".
  - Text contains *opinions and connections*.
  - Numbers aggregate; *text disaggregates*. Text allows us to drill down into underlying behavior when understanding human interaction.

Definition: Text-Mining

- ***Text mining*** is the large-scale, automated processing of plain text language in digital form to extract data that is converted into useful quantitative or qualitative information.
- Text mining is automated on big data that is not amenable to human processing within reasonable time frames. It entails extracting data that is converted into information of many types.
  - Simple: Text mining may be simple as key word searches and counts.
  - Complicated: It may require language parsing and complex rules for information extraction.
- Involves structured text, such as the information in forms and some kinds of web pages.
- May be applied to unstructured text is a much harder endeavor.
- Text mining is also aimed at unearthing unseen relationships in unstructured text as in meta analyses of research papers, see Van Noorden 2012.

Definition: Text-Mining

Importance of text mining.



"Text Mining is a valuable resource in social networking and blogging, customer relations management, tracking public opinion and text filtering."

2. Text Mining: Process

TM Process

The *process of text mining* combines several techniques that enable us to deduce the information from the unstructured data. The general process in text mining are:

1. The first step involved in text mining is *collection of data* i.e. text. The data can be collected from different sources such as websites, emails, social media, blogs and others. All the available data are gathered together in the initial step.
2. After the collection of data, *text pre-processing* is carried out. This is the main step in text mining and requires lots of time and effort. The collected data may be structured, semi-structured and unstructured. In text pre-processing all the available data is cleansed to create structured data. These steps consist of methods such as text cleanup, tokenization, filtering, stemming, lemmatization, linguistic processing, part of speech recognition and word sense disambiguation.

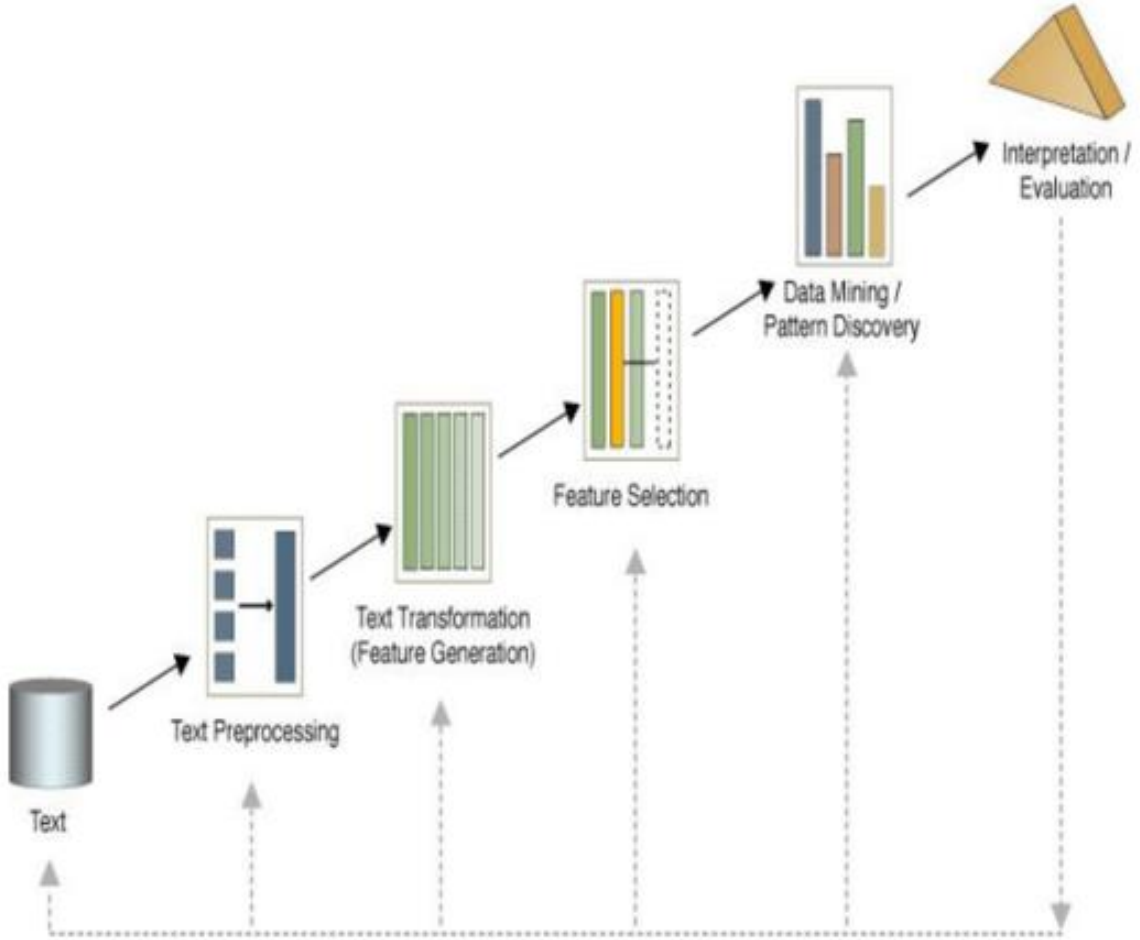
TM Process (cont.)

3. After the pre-processing of data, various techniques are used to *analyse the data*. The analysis must be carried out on structured data as it gives efficient results. The common methods used in these steps are information extraction, information retrieval, categorization, clustering, visualization and summarization.

4. Finally the obtained *results are evaluated and stored* for future reference. In this way the data obtained from different sources are used to get the meaningful patterns using text mining.

TM Process (cont.)

- Text preprocessing
  - Syntactic/Semantic text analysis
- Features Generation
  - Bag of words
- Features Selection
  - Simple counting
  - Statistics
- Text/Data Mining
  - Classification-Supervised learning
  - Clustering-Unsupervised learning
- Analyzing results



3. Text Mining: Techniques

TM Techniques

Text mining techniques are used to discover the insights from structured text/data. These text mining techniques use different tools, methods and applications for their execution. The various text mining techniques are:

1. ***Information Extraction:*** It is one of the most famous text mining techniques. This technique focuses on identifying the extraction of entities, attributes and their relationships from the available textual data. Whatever information is extracted from the data is then stored in a database for future access and retrieval. The efficiency and relevancy of the results are evaluated using precision and recall processes.
2. ***Information Retrieval***: Information Retrieval is the process of extracting relevant and associated patterns based on a specific set of words or phrases. In this text mining technique, information retrieval systems make use of different algorithms to track and monitor user behaviors and discover relevant data accordingly.

TM Techniques (cont.)

3. Categorization: Categorization is one of the most popular supervised learning methods. In this technique, normal language texts are assigned to a predefined set of classes or topics depending upon their content. In this technique, the text documents are gathered, processed and analysed to find the right topics or indexes for each document. Naive Bayesian classifier, Decision tree, Nearest Neighbour classifier and Support Vendor Machines are commonly used to categorize the texts.

4. Clustering: Clustering is one of the most popular unsupervised learning methods in which the data points that are neither classified nor labeled. In this technique the group of text documents which have similar contents are divided into a cluster. The K-means clustering algorithm is one of the most used clustering techniques in which the available data are divided into different clusters by using mean values.

TM Techniques (cont.)

5. Visualization: Visualization is used to simplify and enhance the discovery of useful information with visual cues. It uses visual cues such as text flags to indicate individual documents or document categories and colours to indicate the density of a category, entity, phrase, etc. It is used in placing the large sources of textual data in visual hierarchy.

4. Summarization: Summarization is used to reduce the length of the document and summarize the document's details in brief. Summarization determines the most important points in a lengthy document and replaces the entire set of documents with new important points quickly and efficiently. Summarization involves three steps i.e. pre-processing, processing, and development. Pre-processing step involves building a structured representation of the text whereas different algorithms are used to get a summary of text in processing and finally the development step is where the final text summary is obtained.

4. Text Mining: Methods

TM Methods

There are lots of methods developed to solve the text mining problem which is relevant information retrieval according to the user's requirements. According to information retrieval there are four main methods used in text mining:

1. ***Term Based Method (TBM):*** Term refers to a word with semantic meaning. In term based methods a document is analyzed on the basis of the term and has the advantage of efficient computational performance as well as mature theories from their weighting. Term based method faces enormous challenges in case of polysemy and synonymy. Polysemy means a word hasmultiple meanings and synonymy is multiple words with the same meaning. The semantic meaning of many extracted terms is uncertain and does not provide full information for answering what the user wants.
2. ***Phrase Based Method (PBM):*** Phrase based method may have advantages over term based method as it carries more semantics like information and is less ambiguous. In phrase based methods the document is analyzed on a phrase basis as it is more discriminative than individual terms.

TM Methods (cont.)

3. Concept Based Method (CBM): In this method terms are analyzed on sentences and document level. Concept based methods can effectively discriminate between non important terms and meaningful terms which describe the meaning of the sentences. This model normally relies upon natural language processing techniques. Feature selection is used to optimize the representation and to remove the noise and ambiguity in the document.

4. Pattern Taxonomy Method (PTM): In this method documents are analyzed in terms of pattern basis. Taxonomy refers to the process of finding the root words. Patterns can be structured into a taxonomy by using is-a relationship and can be discovered using techniques like rule mining, frequent item set mining, sequential pattern mining and closed pattern mining. This method refines discovered patterns in text documents and has efficient performance than that of concept based and term based methods.

5. Text Mining: Applications

TM Applications

Text mining has improved user experiences and business decisions. Most of the companies are using text mining tools to add value to their organization and products. Some application areas of text mining includes:

1. ***Risk Management:*** One of the main causes of business failure is due to lack of proper or insufficient risk analysis. Integrating risk management software powered by text mining technologies canhelp businesses to stay updated with all current trends in the market and boost their abilities to cover up the potential risks.
2. ***Customer Care Service:*** When the business system is integrated with text analytical tools, feedback systems, chatbots, online reviews, support tickets and social media profiles, it enables us to improve the customer experience with speed. Text mining and sentiment analysis can provide mechanisms for us to prioritize key points for our customer, allowing us to respond to urgent issues in real-time and helps to increase the customer satisfaction.

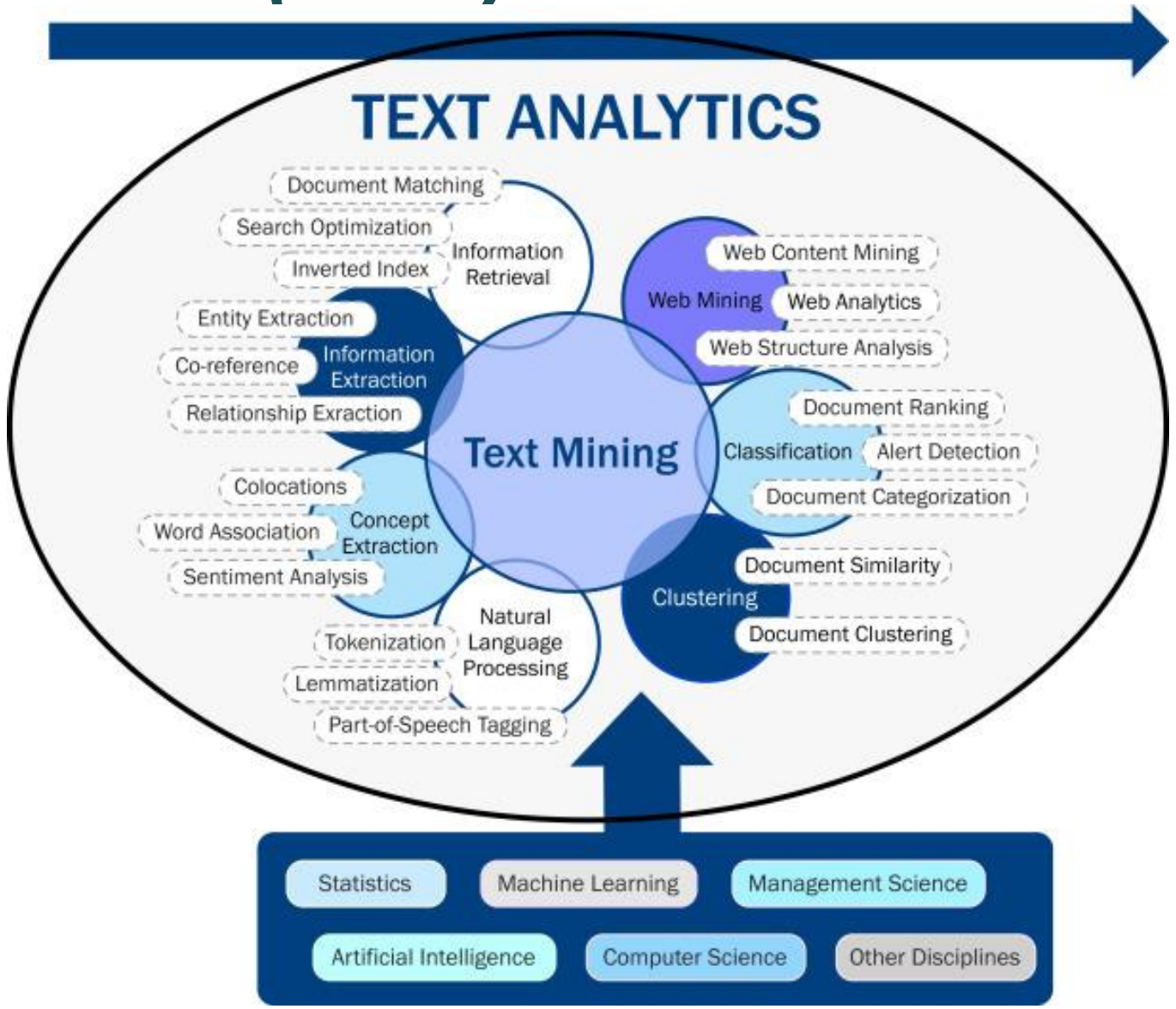
TM Applications (cont.)

3. Healthcare: One of the major applications of text mining is the healthcare sector as it provides valuable information to the researchers. Manual investigation of medical research can be very costly and time consuming. As text mining provides an automation method for extracting the valuable information from the medical literature, it is becoming extremely popular in the medical field as well.

4. Spam Filtering: Text mining is used to filter and exclude the emails from inboxes and thus improving the overall user experiences. With the help of this application it reduces the risk of cyber attacks to the end users.

5. Fraud Detection: By combining the outcomes of the text analysis with relevant structured data we are able to process the user profile and claims efficiently as well as to detect and prevent frauds.

TM Applications (cont.)



6. Text Mining: Practical use

Loading packages

```r
# Load the packages
install.packages("tidyverse")
library(tidyverse)
```

Result:

```
-- Attaching packages -----------------
v ggplot2 3.0.0     v purrr   0.2.5
v tibble  2.0.0     v dplyr   0.7.8
v tidyr   0.8.2     v stringr 1.3.1
v readr   1.1.1     v forcats 0.3.0
-- Conflicts --------------------------
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Importing review data

Let's open the file "Roomba Reviews.csv" and quickly look at it.

```
# Importing review data
review_data <- read_csv("Roomba Reviews.csv")
review_data
```

Result:

```
> review_data
# A tibble: 1,833 × 5
   Date     Product                     Stars Title                                  Review
   <chr>    <chr>                       <dbl> <chr>                                  <chr>
 1 2/28/15  iRobot Roomba 650 for Pets      5 Five Stars                             "You would not believe how well th…
 2 1/12/15  iRobot Roomba 650 for Pets      4 Four Stars                             "You just walk away and it does th…
 3 12/26/13 iRobot Roomba 650 for Pets      5 Awesome love it.                       "You have to Roomba proof your hou…
 4 8/4/13   iRobot Roomba 650 for Pets      3 Love-hate this vaccuum                 "Yes, it's a fascinating, albeit e…
 5 12/22/15 iRobot Roomba 650 for Pets      5 This vacuum is fantastic!!             "Years ago I bought one of the ori…
 6 12/27/15 iRobot Roomba 650 for Pets      5 Wow!                                   "Wow.Wow.  I never knew my floors …
 7 8/17/15  iRobot Roomba 650 for Pets      1 Terrible Product - Ruined our Hardwood Floors! "Wow.. I don't know what to say. T…
 8 12/28/15 iRobot Roomba 650 for Pets      5 Super-impressed by how well it works!  "Wow, wow, WOW!  I wanted to get o…
 9 1/19/14  iRobot Roomba 650 for Pets      5 LOVE THIS                              "Wow, the Roomba is the best chris…
10 7/2/15   iRobot Roomba 650 for Pets      5 Stress is bad; Roomba is great.        "Wow, it changes your life. First …
# … with 1,823 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Using filter() and summarize()

Let’s check the average Stars of iRobot Roomba 650 for Pets.

```r
# Using filter() and summarize()
review_data %>%
  filter(Product == "iRobot Roomba 650 for Pets") %>%
  summarize(Stars_mean = mean(Stars))
```

Result:

```
# A tibble: 1 x 1
  Stars_mean
       <dbl>
1       4.49
```

Using group_by() and summarize()

Let's check the average Stars of Products.

```
# Using group_by() and summarize()
review_data %>%
  group_by(Product) %>%
  summarize(Stars_mean = mean(Stars))
```

Result:

```
# A tibble: 2 × 2
  Product                                  Stars_mean
  <chr>                                         <dbl>
1 iRobot Roomba 650 for Pets                     4.49
2 iRobot Roomba 880 for Pets and Allergies       4.42
```

Unstructured data

Let’s check the average Review of Products.

```
# Unstructured data
review_data %>%
  group_by(Product) %>%
  summarize(Review_mean = mean(Review))
```

Unstructured data *doesn't have a predefined data model or structure*.

Common unstructured data examples include customer information, product catalogs, and financial records.

Result:

```
# A tibble: 2 × 2
  Product                                  Review_mean
  <chr>                                         <dbl>
1 iRobot Roomba 650 for Pets                       NA
2 iRobot Roomba 880 for Pets and Allergies         NA
Warning messages:
1: In mean.default(Review) :
  argument is not numeric or logical: returning NA
2: In mean.default(Review) :
  argument is not numeric or logical: returning NA
```

Since this type of data is not organized in a predefined manner, it's more difficult to process and analyze using traditional methods.

Summarizing with n()

If we cant do any calculations with Reviews, then let's count rows.

```
# Summarizing with n()
review_data %>%
  summarize(Number_rows = n())
```

Result:

```
# A tibble: 1 x 1
  Number_rows
        <int>
1        1833
```

Summarizing with n() (cont.)

If we cant do any calculations with Reviews, then let's count rows for Products.

```
# Summarizing with n()
review_data %>%
  group_by(Product) %>%
  summarize(Number_rows = n())
```

Result:

```
# A tibble: 2 x 2
  Product                                  Number_rows
  <chr>                                         <int>
1 iRobot Roomba 650 for Pets                      633
2 iRobot Roomba 880 for Pets and Allergies        1200
```

Summarizing with count()

If we cant do any calculations with Reviews, then let's count rows for Products.

```
# Summarizing with count()
review_data %>%
  count(Product)
```

Result:

```
# A tibble: 2 x 2
  Product                                  n
  <chr>                                <int>
1 iRobot Roomba 650 for Pets             633
2 iRobot Roomba 880 for Pets and Allergies  1200
```

Summarizing with count() (cont.)

If we cant do any calculations with Reviews, then let's count rows for Products and change the order.

```
# Summarizing with count() & change order
review_data %>%
  count(Product) %>%
  arrange(desc(n))
```

Result:

```
# A tibble: 2 × 2
  Product                                          n
  <chr>                                        <int>
1 iRobot Roomba 880 for Pets and Allergies      1200
2 iRobot Roomba 650 for Pets                     633
```

Tokenizing text

Some natural language processing (NLP) vocabulary:

- Bag of words: Words in a document are independent
- Every separate body of text is a document
- Every unique word is a term
- Every occurrence of a term is a token
- Creating a bag of words is called tokenizing

Using unnest_tokens()

We want to get each word of the text

```
# Using unnest_tokens()
install.packages("tidytext")
library(tidytext)

tidy_review <- review_data %>%
  unnest_tokens(Word, Review)
tidy_review
```

Result:

```
# A tibble: 229,481 × 5
   Date     Product                      Stars Title       Word
   <chr>    <chr>                        <dbl> <chr>       <chr>
 1 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  you
 2 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  would
 3 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  not
 4 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  believe
 5 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  how
 6 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  well
 7 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  this
 8 2/28/15  iRobot Roomba 650 for Pets       5 Five Stars  works
 9 1/12/15  iRobot Roomba 650 for Pets       4 Four Stars  you
10 1/12/15  iRobot Roomba 650 for Pets       4 Four Stars  just
# … with 229,471 more rows
```

Counting words

We want to count each word of the text

```
# Counting words
tidy_review %>%
  count(Word) %>%
  arrange(desc(n))
```

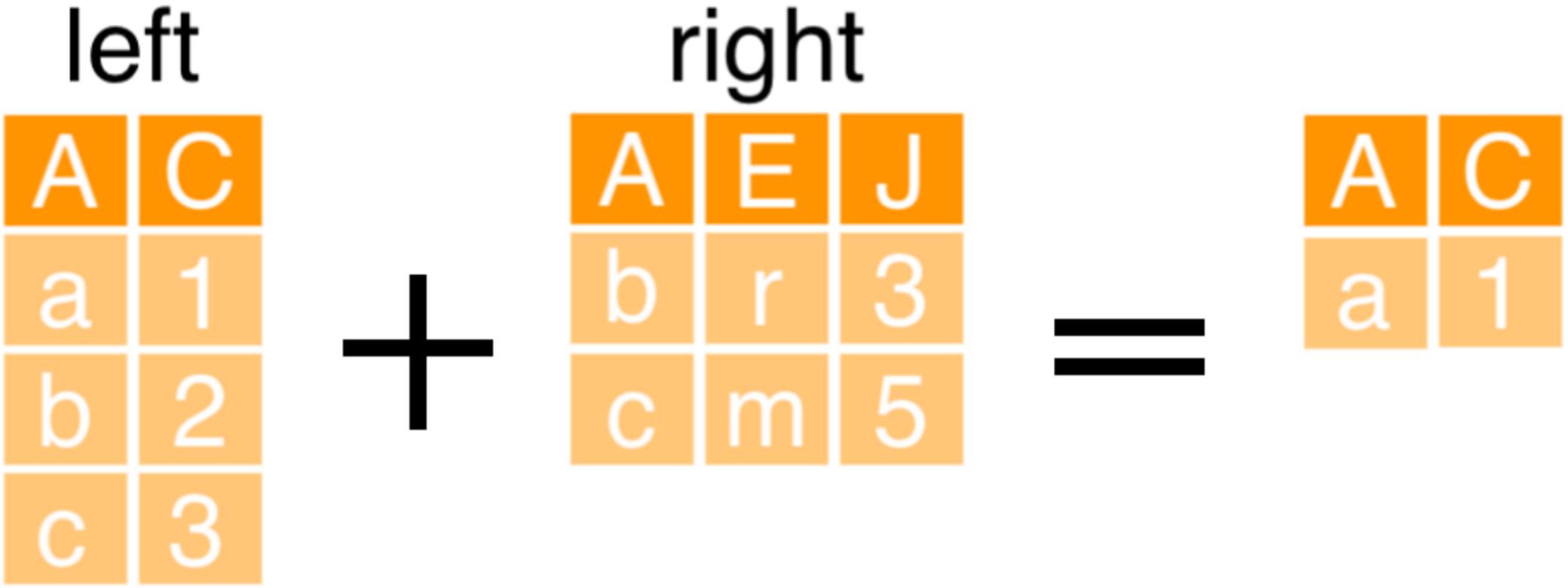
Result:

```
# A tibble: 10,310 × 2
   Word      n
   <chr> <int>
 1 the   11785
 2 it     7905
 3 and    6794
 4 to     6440
 5 i      6034
 6 a      5884
 7 is     3347
 8 of     3229
 9 have   2470
10 that   2410
# … with 10,300 more rows
```

But notice that the words include common words like *the* and *this*.

Using anti_join()

anti_join() return all rows from *left* without a match in *right*



Using anti_join() (cont.)

Stopwords are the words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For example, the words like the, he, have etc.

We want to keep meaningful word to have an accurate analysis.

```
# See stopwords
View(stop_words)

# Using anti_join()
tidy_review2 <- review_data %>%
  unnest_tokens(Word, Review) %>%
  anti_join(stop_words, by = c("Word" = "word"))
tidy_review2
```

Result:

```
# A tibble: 78,868 × 5
   Date     Product                    Stars Title                 Word
   <chr>    <chr>                      <dbl> <chr>                 <chr>
 1 1/12/15  iRobot Roomba 650 for Pets     4 Four Stars            walk
 2 1/12/15  iRobot Roomba 650 for Pets     4 Four Stars            rest
 3 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.      roomba
 4 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.      proof
 5 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.      house
 6 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.      awesome
 7 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.      pet
 8 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.      cleans
 9 8/4/13   iRobot Roomba 650 for Pets     3 Love-hate this vaccuum fascinating
10 8/4/13   iRobot Roomba 650 for Pets     3 Love-hate this vaccuum albeit
# … with 78,858 more rows
```

Counting words again

We want to count each meaningful word of the text

```
# Counting words again
tidy_review2 %>%
  count(word) %>%
  arrange(desc(n))
```

Result:

Notice that there are no common words like *the* and *this*.

```
# A tibble: 9,672 × 2
   word          n
   <chr>     <int>
 1 roomba     2286
 2 clean      1204
 3 vacuum      989
 4 hair        900
 5 cleaning    809
 6 time        795
 7 house       745
 8 floors      657
 9 day         578
10 floor       561
# … with 9,662 more rows
```

Plotting word counts

Recall our text

```
# Our data
tidy_review <- review_data %>%
  mutate(id = row_number()) %>%
  unnest_tokens(Word, Review) %>%
  anti_join(stop_words, by = c("Word" = "word"))
tidy_review
```

Result:

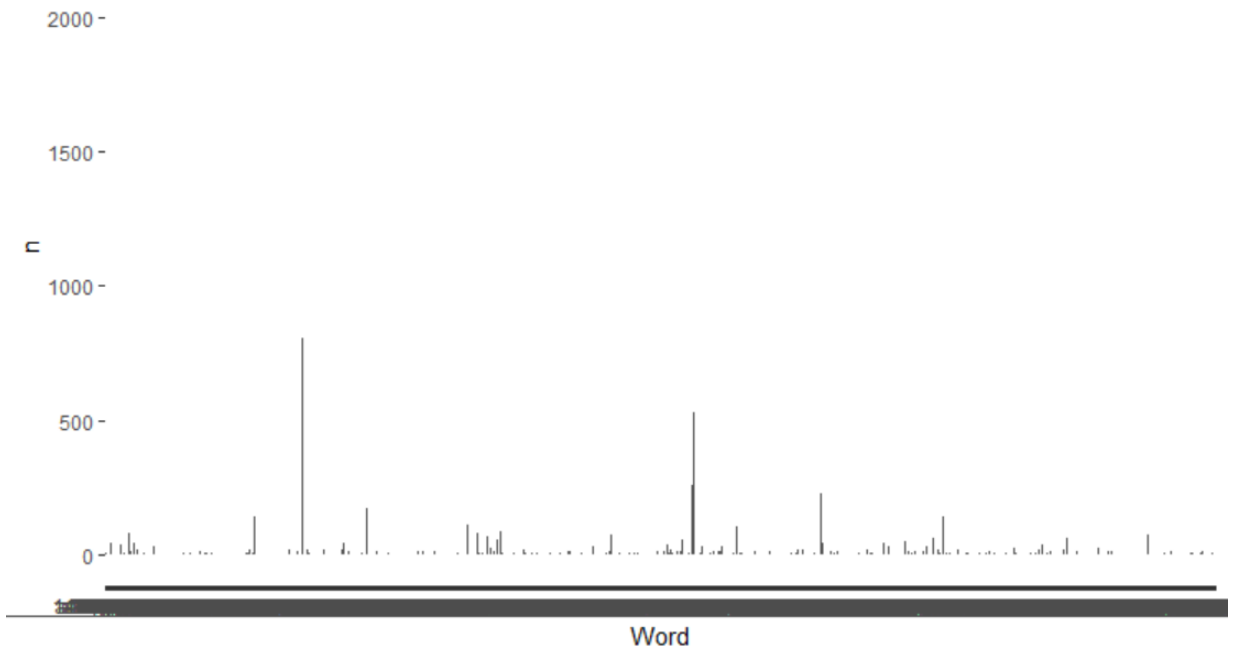
```
# A tibble: 78,868 × 6
   Date     Product                    Stars Title                    id Word
   <chr>    <chr>                      <dbl> <chr>                 <int> <chr>
 1 1/12/15  iRobot Roomba 650 for Pets     4 Four Stars                2 walk
 2 1/12/15  iRobot Roomba 650 for Pets     4 Four Stars                2 rest
 3 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.          3 roomba
 4 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.          3 proof
 5 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.          3 house
 6 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.          3 awesome
 7 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.          3 pet
 8 12/26/13 iRobot Roomba 650 for Pets     5 Awesome love it.          3 cleans
 9 8/4/13   iRobot Roomba 650 for Pets     3 Love-hate this vaccuum    4 fascinating
10 8/4/13   iRobot Roomba 650 for Pets     3 Love-hate this vaccuum    4 albeit
# … with 78,858 more rows
```

Visualizing counts with geom_col()

We create a table with words' counts and visualize that.

```
# Visualizing counts with geom_col()
word_counts <- tidy_review %>%
  count(Word) %>%
  arrange(desc(n))
ggplot( word_counts, aes(x = Word, y = n)) +
  geom_col()
```

Result:



filter() before visualizing

We want to see words that are used more than 300 times.

```
# filter() before visualizing
word_counts2 <- tidy_review %>%
  count(Word) %>%
  filter(n > 300) %>%
  arrange(desc(n))
```

Result:

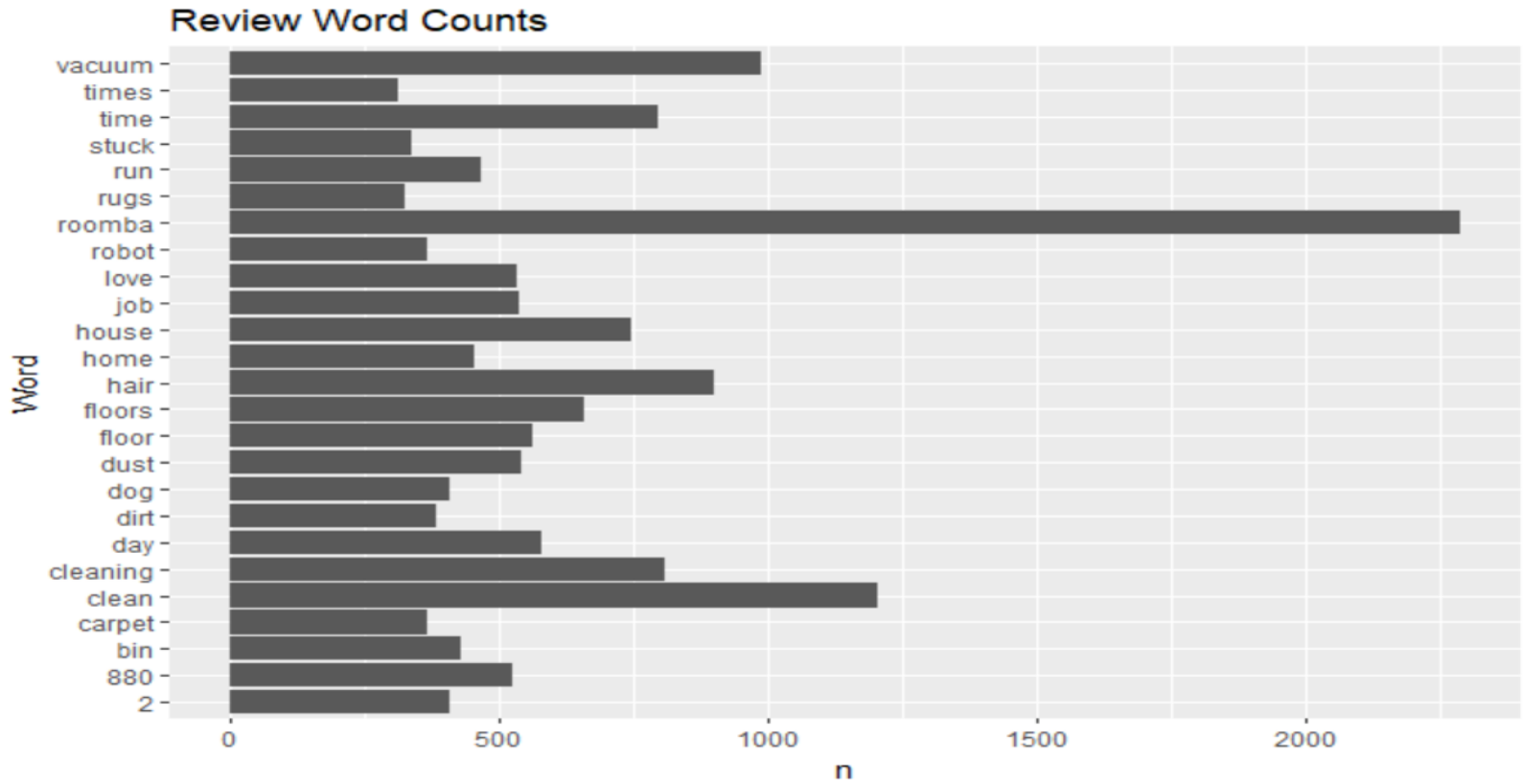
```
# A tibble: 25 × 2
   Word          n
   <chr>     <int>
 1 roomba     2286
 2 clean      1204
 3 vacuum      989
 4 hair        900
 5 cleaning    809
 6 time        795
 7 house       745
 8 floors      657
 9 day         578
10 floor       561
# … with 15 more rows
```

Using coord_flip()

We want to have a smooth visualization with words on the Y axes.

```
# Using coord_flip()
ggplot( word_counts2, aes(x = Word, y = n)) +
  geom_col() +
  coord_flip() +
  ggtitle("Review Word Counts")
```

Result:



Custom stop words

We want to remove some custom stopwords and update illustration.

```
# Custom stop words
# Using tribble() and bind_rows()
custom_stop_words <- tribble( ~word, ~lexicon,
  "roomba", "CUSTOM", "2","CUSTOM")
stop_words2 <- stop_words %>%
  bind_rows(custom_stop_words)
```

```
# Removing stop words again
tidy_review <- review_data %>%
  mutate(id = row_number()) %>%
  select(id, Date, Product, Stars, Review) %>%
  unnest_tokens(Word, Review) %>%
  anti_join(stop_words2, by = c("Word" = "word"))
tidy_review
tidy_review %>%
  filter(Word == "roomba")
```

Result:

```
# A tibble: 76,175 × 5
      id Date     Product                    Stars Word
   <int> <chr>    <chr>                      <dbl> <chr>
 1     2 1/12/15  iRobot Roomba 650 for Pets     4 walk
 2     2 1/12/15  iRobot Roomba 650 for Pets     4 rest
 3     3 12/26/13 iRobot Roomba 650 for Pets     5 proof
 4     3 12/26/13 iRobot Roomba 650 for Pets     5 house
 5     3 12/26/13 iRobot Roomba 650 for Pets     5 awesome
 6     3 12/26/13 iRobot Roomba 650 for Pets     5 pet
 7     3 12/26/13 iRobot Roomba 650 for Pets     5 cleans
 8     4 8/4/13   iRobot Roomba 650 for Pets     3 fascinating
 9     4 8/4/13   iRobot Roomba 650 for Pets     3 albeit
10     4 8/4/13   iRobot Roomba 650 for Pets     3 expensive
# … with 76,165 more rows
# ℹ Use `print(n = ...)` to see more rows
> tidy_review %>%
+   filter(Word == "roomba")
# A tibble: 0 × 5
```

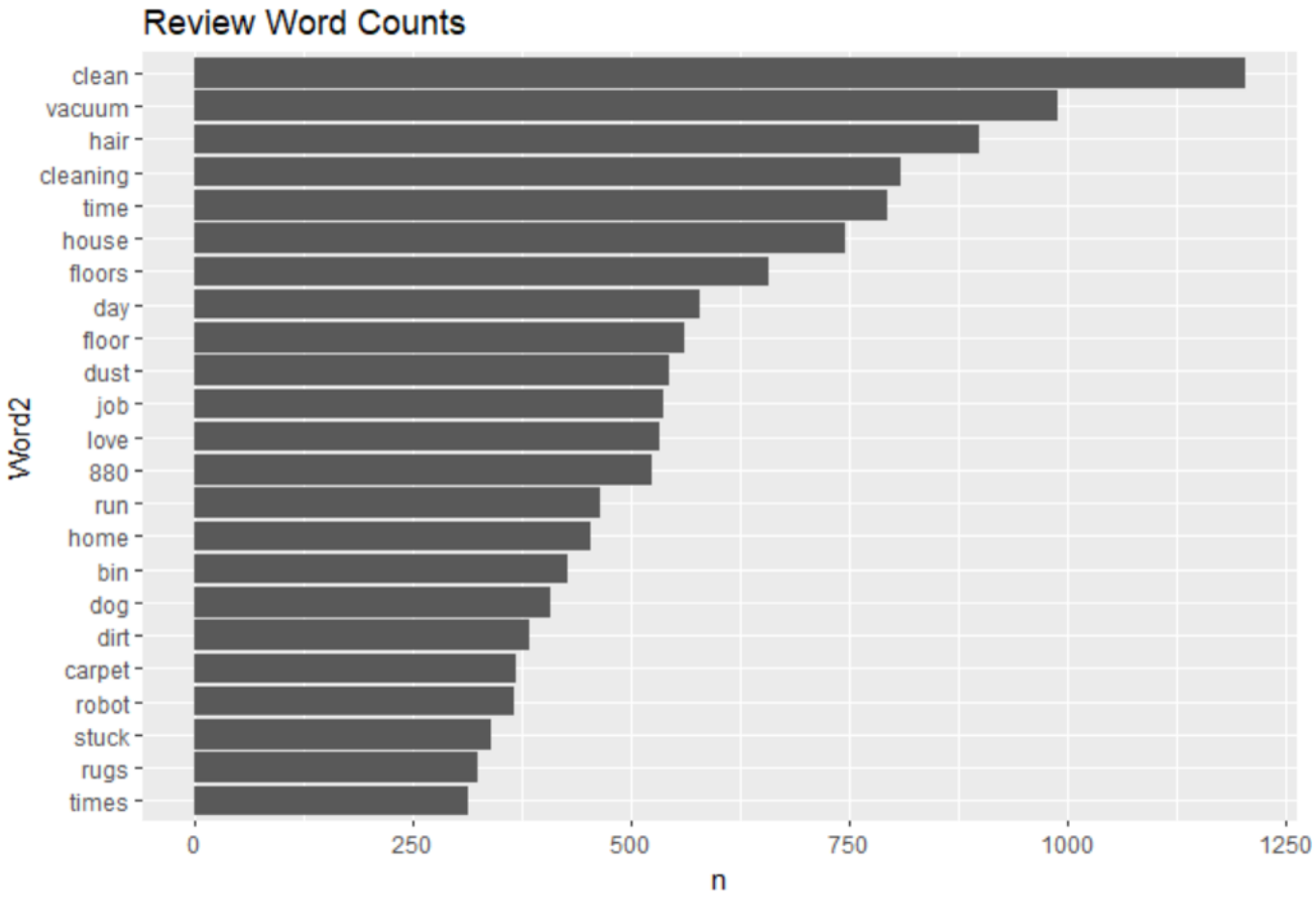
Using fct_reorder()

We want to have a bit smoother visualization.

```
# Using fct_reorder()
word_counts <- tidy_review %>%
  count(Word) %>%
  filter(n > 300) %>%
  mutate(Word2 = fct_reorder(Word, n))

ggplot( word_counts, aes(x = Word2, y = n)) +
  geom_col() +
  coord_flip() +
  ggtitle("Review Word Counts")
```

Result:



Faceting word count plots

Counting by product

```
# Counting by product
tidy_review %>%
  count(Word, Product) %>%
  arrange(desc(n))
```

Result:

```
# A tibble: 12,719 × 3
   Word     Product                                                  n
   <chr>    <chr>                                                <int>
 1 clean    iRobot Roomba 880 for Pets and Allergies               815
 2 vacuum   iRobot Roomba 880 for Pets and Allergies               678
 3 hair     iRobot Roomba 880 for Pets and Allergies               595
 4 cleaning iRobot Roomba 880 for Pets and Allergies               560
 5 880      iRobot Roomba 880 for Pets and Allergies               518
 6 house    iRobot Roomba 880 for Pets and Allergies               494
 7 time     iRobot Roomba 880 for Pets and Allergies               494
 8 floors   iRobot Roomba 880 for Pets and Allergies               405
 9 love     iRobot Roomba 880 for Pets and Allergies               403
10 dust     iRobot Roomba 880 for Pets and Allergies               399
# … with 12,709 more rows
```

Using top_n()

We select and return only top-10 rows

```
# Using top_n()
tidy_review %>%
  count(Word, Product) %>%
  group_by(Product) %>%
  top_n(10, n)
```

Result:

```
# A tibble: 20 x 3
# Groups:   Product [2]
   Word     Product                                                n
   <chr>    <chr>                                              <int>
 1 880      iRobot Roomba 880 for Pets and Allergies             518
 2 clean    iRobot Roomba 650 for Pets                           389
```

Using ungroup()

We select and return only top-10 rows and ungroup

```
# Using top_n()
tidy_review %>%
  count(Word, Product) %>%
  group_by(Product) %>%
  top_n(10, n)
```

Result:

```
# A tibble: 20 × 3
   Word     Product                                                    n
   <chr>    <chr>                                                  <int>
 1 880      iRobot Roomba 880 for Pets and Allergies                 518
 2 clean    iRobot Roomba 650 for Pets                               389
 3 clean    iRobot Roomba 880 for Pets and Allergies                 815
 4 cleaning iRobot Roomba 650 for Pets                               249
 5 cleaning iRobot Roomba 880 for Pets and Allergies                 560
 6 day      iRobot Roomba 650 for Pets                               209
```

Using fct_reorder() and facet_wrap()

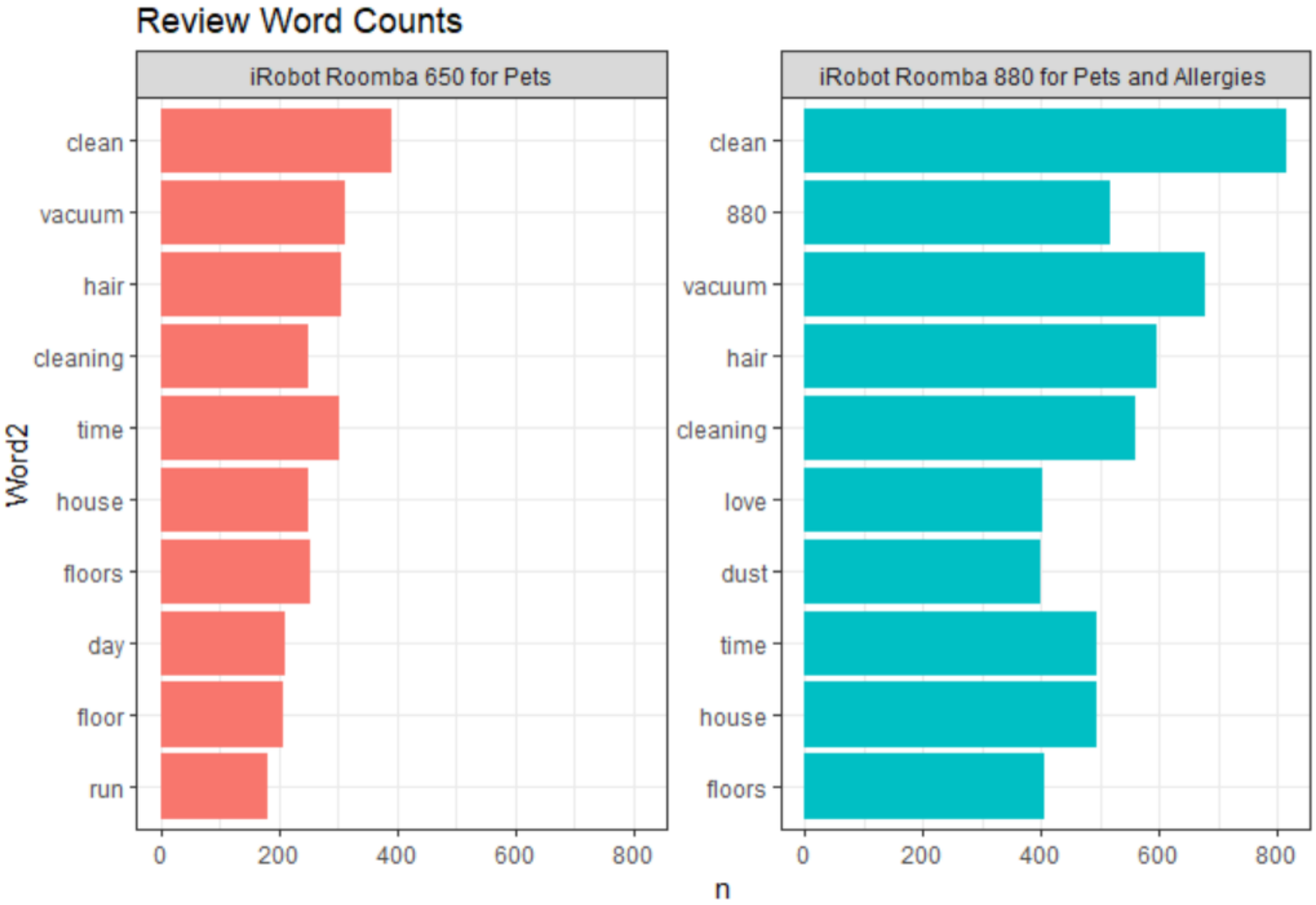
We select and return only top-10 rows, and ungroup, and reorder, and have plots for each product.

```
# Using fct_reorder() and facet_wrap()
word_counts <- tidy_review %>%
  count(Word, Product) %>%
  group_by(Product) %>%
  top_n(10, n) %>%
  ungroup() %>%
  mutate(Word2 = fct_reorder(Word, n))

ggplot(word_counts, aes(x = Word2, y = n, fill = Product)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ Product, scales = "free_y") +
  coord_flip() +
  ggtitle("Review Word Counts") +
  theme_bw()
```

Using fct_reorder() and facet_wrap()

We select and return only top-10 rows, and ungroup, and reorder to get:



Plotting word clouds

We want to see a cloud of words

```
# Using wordcloud()
install.packages("wordcloud")
library(wordcloud)

word_counts <- tidy_review %>%
  count(Word)
wordcloud( words = word_counts$Word,
  freq = word_counts$n,
  max.words = 30,
  colors = "green")
```

Result:

7. In-class Assignment

You will be working with ‘Twitter Data.csv’ file. Please load the file into R Studio:

```
> head(Twitter.Data)
# A tibble: 6 × 15
  tweet_id airline_sen…¹ airli…² negat…³ negat…⁴ airline airli…⁵ name  negat…⁶ retwe…⁷ text  tweet…⁸ tweet…⁹ tweet…ˣ
     <dbl> <chr>           <dbl> <chr>     <dbl> <chr>   <chr>   <chr> <chr>   <dbl> <chr> <chr>   <chr>    <chr>
1  5.70e17 neutral         1     NA           NA Virgin… NA      cair… NA          0 "@Vi… NA      2/24/2… NA
2  5.70e17 positive        0.349 NA            0 Virgin… NA      jnar… NA          0 "@Vi… NA      2/24/2… NA
3  5.70e17 neutral         0.684 NA           NA Virgin… NA      yvon… NA          0 "@Vi… NA      2/24/2… Lets P…
4  5.70e17 negative        1     Bad Fl…   0.703 Virgin… NA      jnar… NA          0 "@Vi… NA      2/24/2… NA
5  5.70e17 negative        1     Can't …   1     Virgin… NA      jnar… NA          0 "@Vi… NA      2/24/2… NA
6  5.70e17 negative        1     Can't …   0.684 Virgin… NA      jnar… NA          0 "@Vi… NA      2/24/2… NA
# … with 1 more variable: user_timezone <chr>, and abbreviated variable names ¹airline_sentiment,
#   ²airline_sentiment_confidence, ³negativereason, ⁴negativereason_confidence, ⁵airline_sentiment_gold,
#   ⁶negativereason_gold, ⁷retweet_count, ⁸tweet_coord, ⁹tweet_created, ˣtweet_location
```

```
> glimpse(Twitter.Data)
Rows: 14,640
Columns: 15
$ tweet_id                     <dbl> 5.70306e+17, 5.70301e+17, 5.70301e+17, 5.70301e+17, 5.70301e+17, 5.70301e+17,…
$ airline_sentiment            <chr> "neutral", "positive", "neutral", "negative", "negative", "negative", "positi…
$ airline_sentiment_confidence <dbl> 1.0000, 0.3486, 0.6837, 1.0000, 1.0000, 1.0000, 0.6745, 0.6340, 0.6559, 1.000…
$ negativereason               <chr> NA, NA, NA, "Bad Flight", "Can't Tell", "Can't Tell", NA, NA, NA, NA, NA, NA,…
$ negativereason_confidence    <dbl> NA, 0.0000, NA, 0.7033, 1.0000, 0.6842, 0.0000, NA, NA, NA, 0.0000, NA, NA, N…
$ airline                      <chr> "Virgin America", "Virgin America", "Virgin America", "Virgin America", "Virg…
$ airline_sentiment_gold       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ name                         <chr> "cairdin", "jnardino", "yvonnalynn", "jnardino", "jnardino", "jnardino", "cjm…
$ negativereason_gold          <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ retweet_count                <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
$ text                         <chr> "@VirginAmerica What @dhepburn said.", "@VirginAmerica plus you've added comm…
$ tweet_coord                  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ tweet_created                <chr> "2/24/2015 11:35", "2/24/2015 11:15", "2/24/2015 11:15", "2/24/2015 11:15", "…
$ tweet_location               <chr> NA, NA, "Lets Play", NA, NA, NA, "San Francisco CA", "Los Angeles", "San Dieg…
$ user_timezone                <chr> "Eastern Time (US & Canada)", "Pacific Time (US & Canada)", "Central Time (US…
```