Text Data Visualization by tidytext :: Cheat Sheet

Why we use tidytext?

Text mining and natural language processing is an entire field with a lot of tools and ways to analyze. **tidytext** is one of those packages that intends to help with text mining in R, and provides supplemental functions for the mining process with easy implementation.

When we meet text data, using **tidytext** package can make many text analysis tasks easier and more effective. Much of the infrastructure needed for text mining with tidy data frames already exists in other widely used packages like dplyr, tidyr and ggplot2. In this cheatsheet, we provide functions with examples to allow the use of tidytext package combined with existing infrastructure to do the text analysis work.

Creator
Xiaolin Sima (xs2483) Yanni Chen (yc4179)

Citation
Silge,Julia and Robinson, David. Introduction to Tidytext. August 19th 2022. Retrieved from https://cran.r-project.org/web/packages/tidytext/vignettes/tidytext.html

We will use the data from the Wine Tasting Dataset:
https://www.kaggle.com/datasets/mysarahmadbhat/wine-tasting

province	variety	description	title	winery
Oregon	Pinot Noir	Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rustic, earthy, herbal characteristics. Nonetheless, if you think of it as a pleasantly unfussy country wine, it's a good companion to a hearty winter stew.	Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)	Sweet Cheeks

To **convert to tidy data**, restructure it as one-token-per-row using the **unnest_tokens** function

```
wine_tasting %>% unnest_tokens(word, description)
```

province	variety	title	winery	word
Oregon	Pinot Noir	Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)	Sweet Cheeks	much
Oregon	Pinot Noir	Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)	Sweet Cheeks	like
Oregon	Pinot Noir	Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)	Sweet Cheeks	the

Now the data is in one-word-per-row format, we can manipulate it with tidy tools like dplyr.

```
wine_tasting %>% unnest_tokens(word, description) %>% count(word, sort = TRUE)
```

word	n
and	85755
the	56821
a	46048
of	41492
with	28442

We can use above code to **count frequency of each word**.

From the left we can see that a lot of the large frequency words are some unimportant words, which need to be further preprocessed.

We can remove **stop words** (words filtered out before procesing text since insignificant)

```
wine_tasting %>%
unnest_tokens(word, description) %>%
anti_join(stop_words) %>%
count(word, sort = TRUE)
```

→

word	n
wine	19195
flavors	17075
fruit	12242
cherry	8531
acidity	8337

anti_join applied here is a filtering join: drops all observations in the initial data frame that have a match in the second data frame

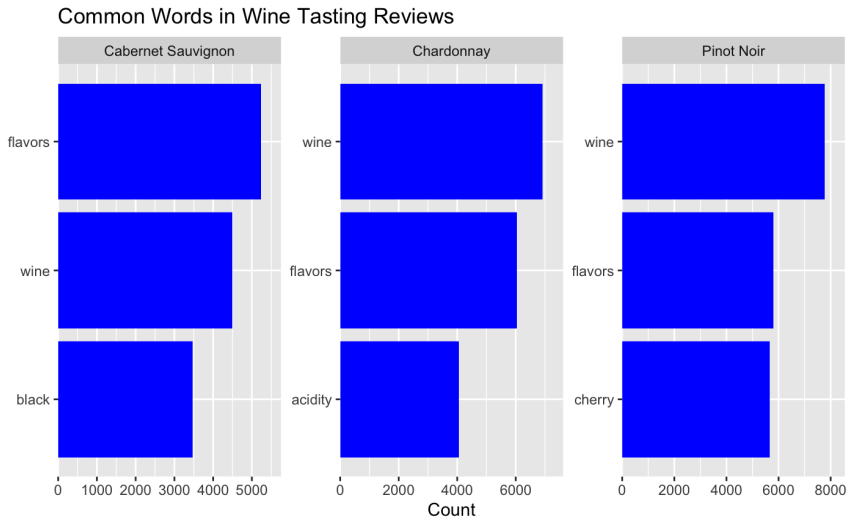
By below code, we can visualize the top 3 **word frenquency** in each wine variety.

slice_max helps to choose the number of words we want, by changing that, we can get the top n word frequency.

reorder_within is a nifty function within tidytext: it reorders a column before plotting with faceting, such that the values are ordered within each facet. This requires two functions: **reorder_within** applied to the column, then either **scale_x_reordered** or **scale_y_reordered** added to the plot.

```
top_variety ← wine_tasting %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  group_by(variety) %>%
  count(word) %>%
  slice_max(n, n = 3)

top_variety %>%
  ggplot(aes(x = n, y = reorder_within(word, n, variety))) +
  geom_bar(stat = "identity", fill="blue") +
  scale_x_continuous(expand = expansion(mult = c(0, .1)),
                     name = "Count") +
  scale_y_reordered(sep = "___") +
  facet_wrap(~variety, scales = "free") +
  labs(y = NULL,
       title = "Common Words in Wine Tasting Reviews")
```



We can also do word **sentiment analysis** using the built in sentiment list to get sentiment of each word and its frequency.

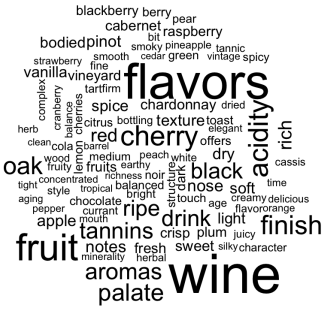
inner_join drops the unmatched observations

word	sentiment	n
rich	positive	4783
soft	positive	3552
sweet	positive	3228

```
sentiment ← get_sentiments("bing")
word_sentiment ← wine_tasting %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  mutate(word = str_extract(word, "[a-z']+")) %>%
  inner_join(sentiment) %>%
  count(word, sentiment, sort = TRUE)
```

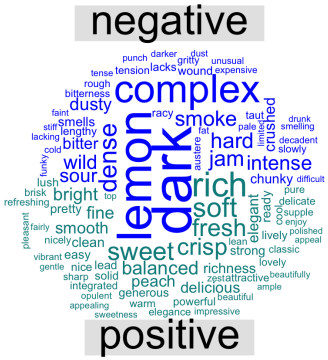
We can also combine tidytext with **Wordcloud** package.

Larger words means higher frequency.




```
wine_tasting %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

We can also do the sentiment analysis to tag positive and negative words using an **inner join,** then find the most common positive and negative words, and finally indicate the result by Wordcloud.




```
wine_tasting %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  inner_join(sentiment) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("blue", "darkcyan"),
                   max.words = 100)
```