Lab 1

Zhen Lin

2023-01-23

Data

We'll work with the #tidytuesday data for 2019, specifically the #rstats dataset, containing nearly 500,000 tweets over a little more than a decade using that hashtag.

The data is in under Dataset tab of Week 3 module on Canvas.

You can import the dataset using the code below.

```
library(rio)
library(here)
d <- rio::import(here::here("rstats_tweets.rds"),
                 setclass = "tbl_df")
```

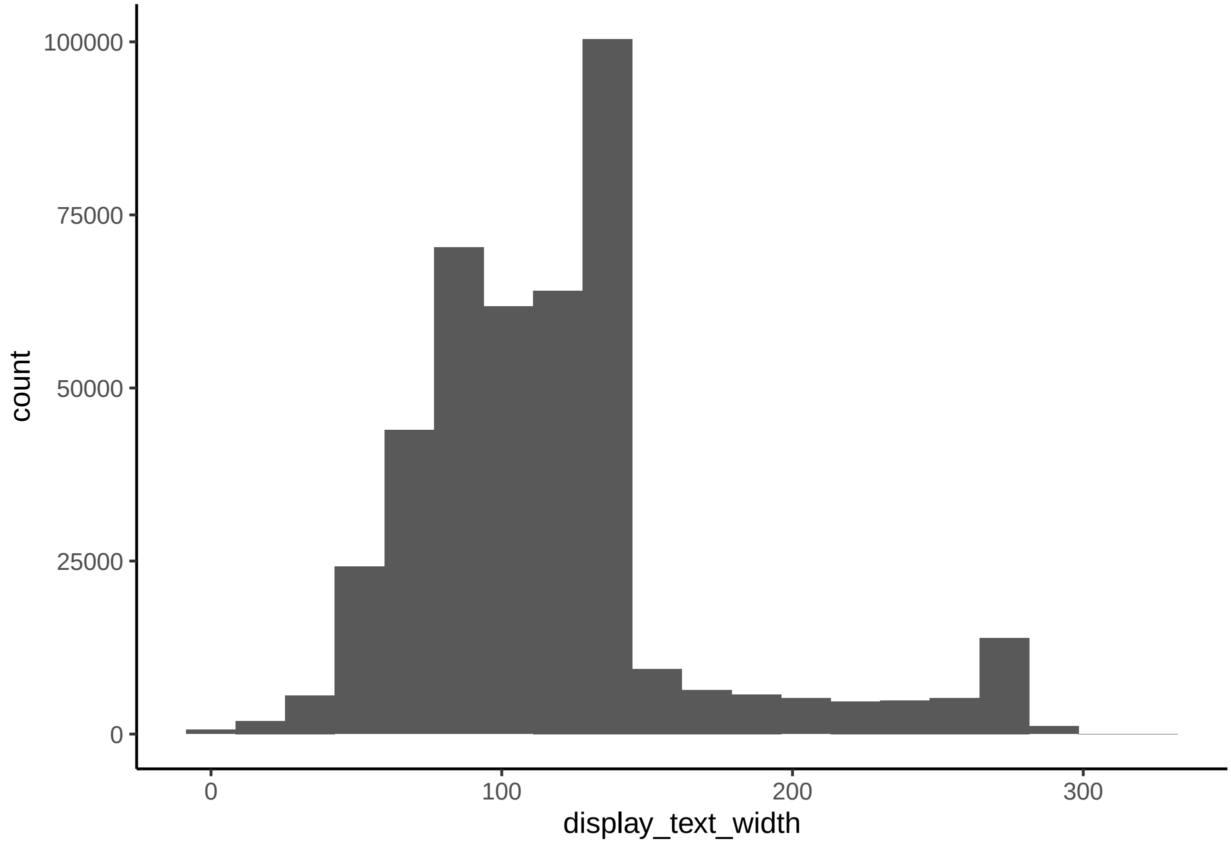
If you need help with processing text data, please revisit the notebook introduced in Week 1.

https://www.kaggle.com/code/uocoeeds/introduction-to-textual-data

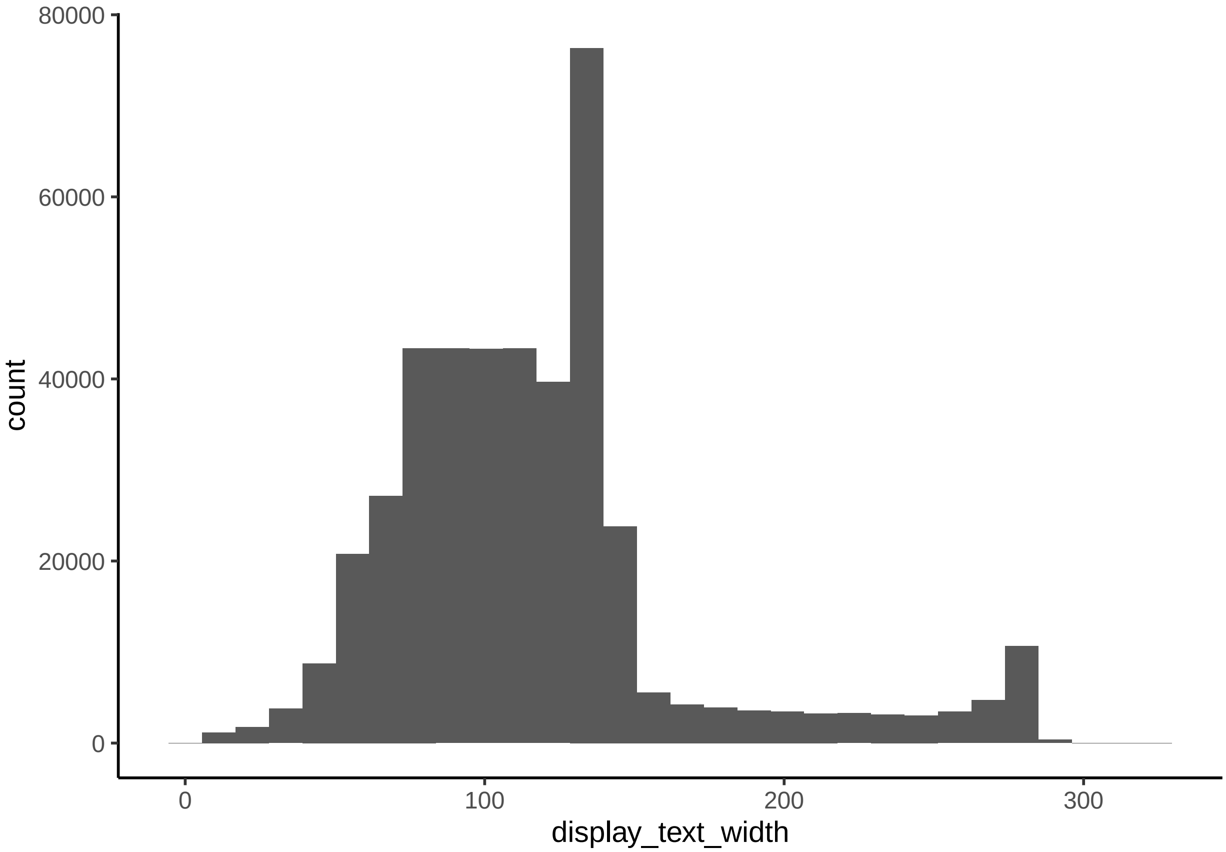
Histogram and Density plots

1. Create a histogram the column `display_text_width` using the `ggplot2` package and `geom_histogram()` function. Try at least four different numbers of bins (e.g., 20, 30, 40, 50) by manipulating the `bins=` argument. Select what you think best represents the data for each. Provide a brief justification for your decision. For all plots you created, change the default background color from grayish to white.

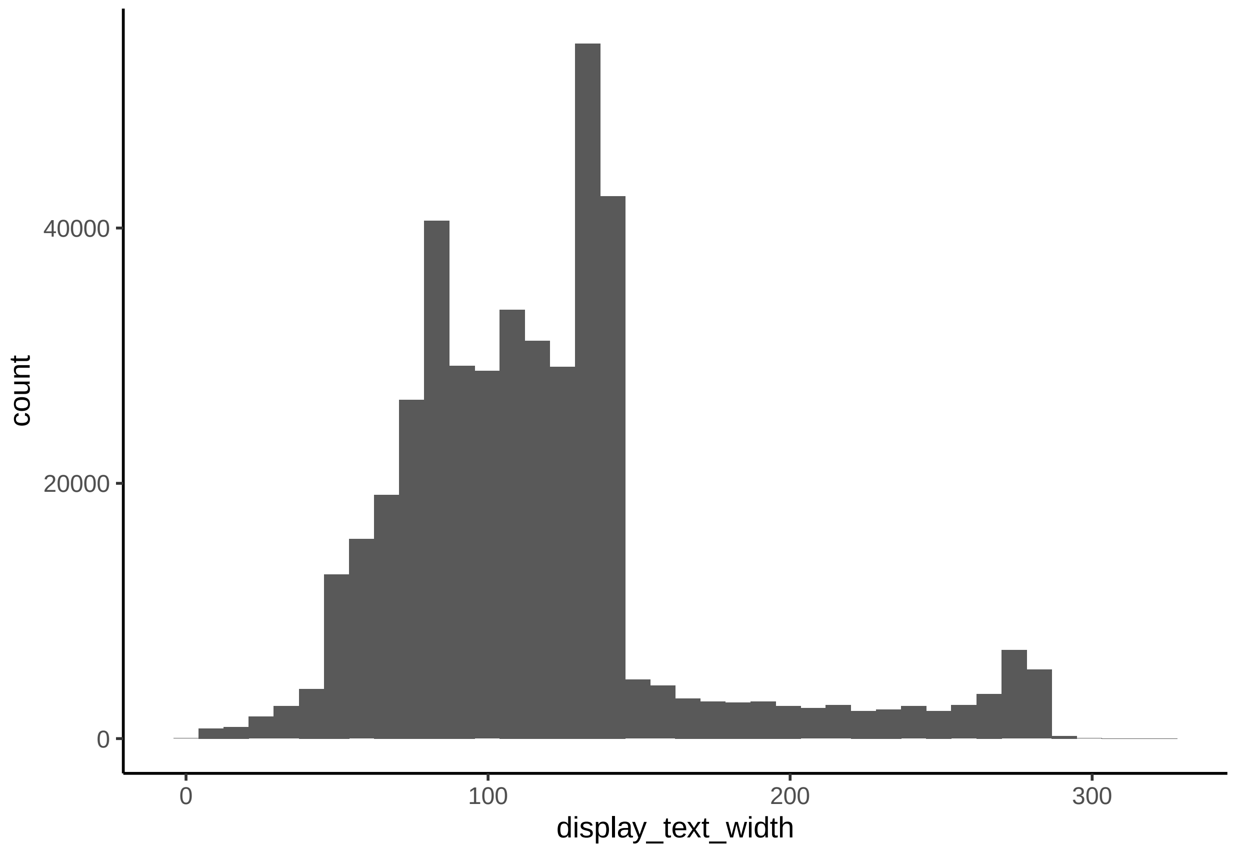
```
library(ggplot2)

ggplot(d, aes(display_text_width))+
  geom_histogram(bins = 20) +
  theme_classic()
```



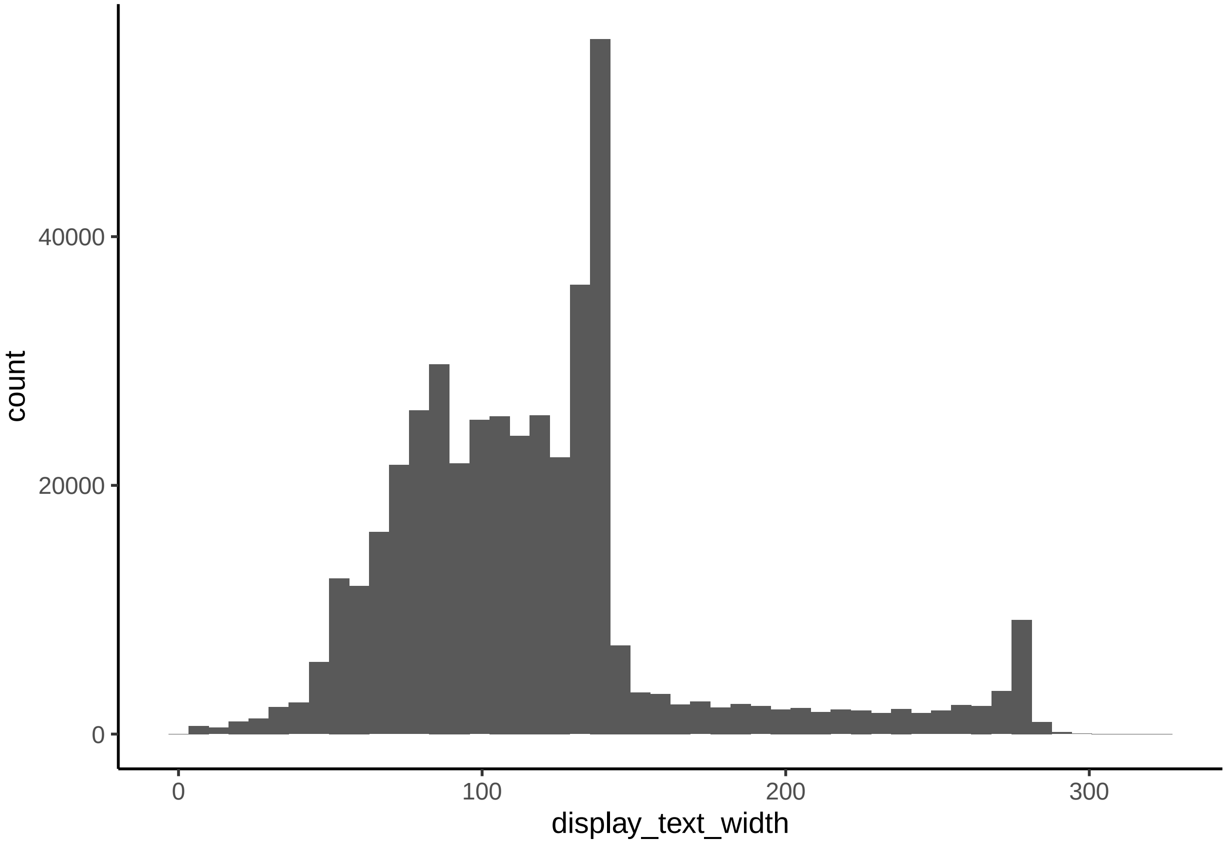
```r
ggplot(d, aes(display_text_width))+
  geom_histogram(bins = 30) +
  theme_classic()
```



```
ggplot(d, aes(display_text_width))+
  geom_histogram(bins = 40) +
  theme_classic()
```



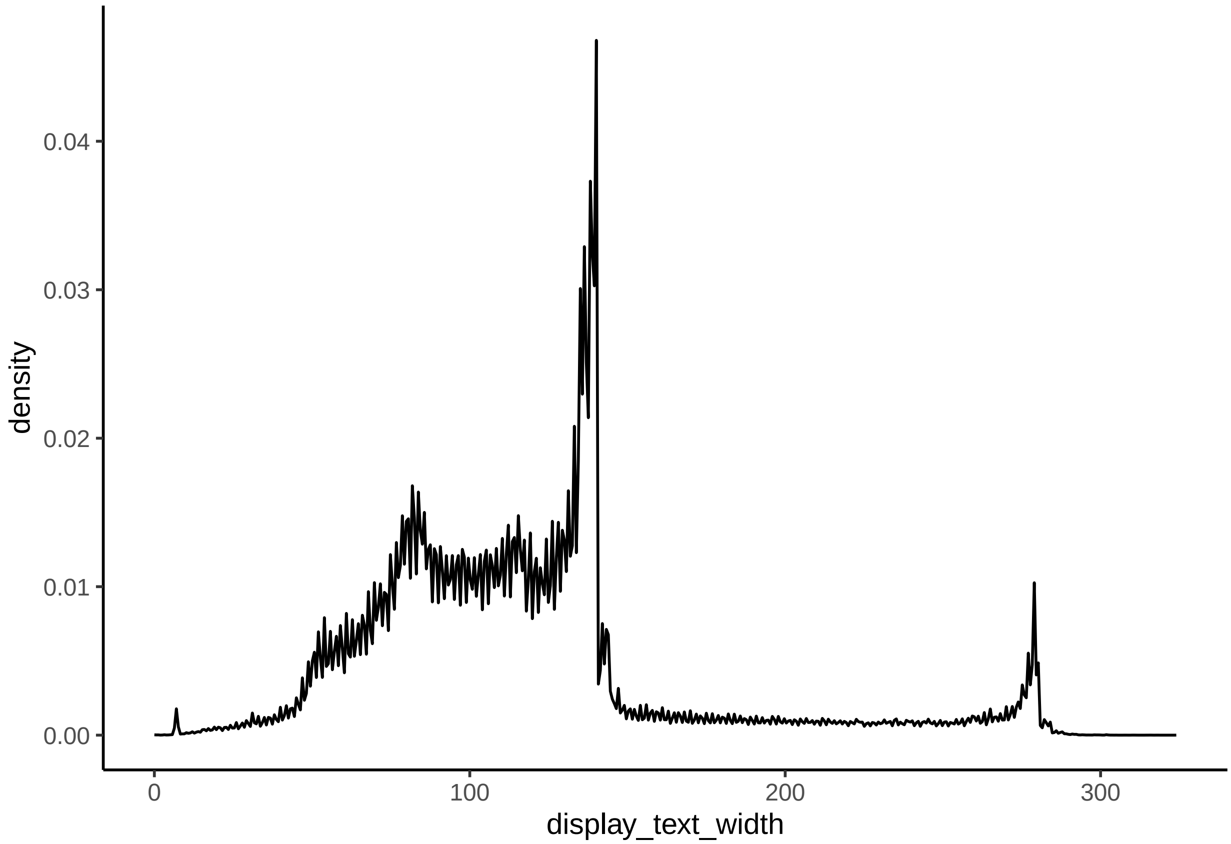
```
ggplot(d, aes(display_text_width))+
  geom_histogram(bins = 50) +
  theme_classic()
```



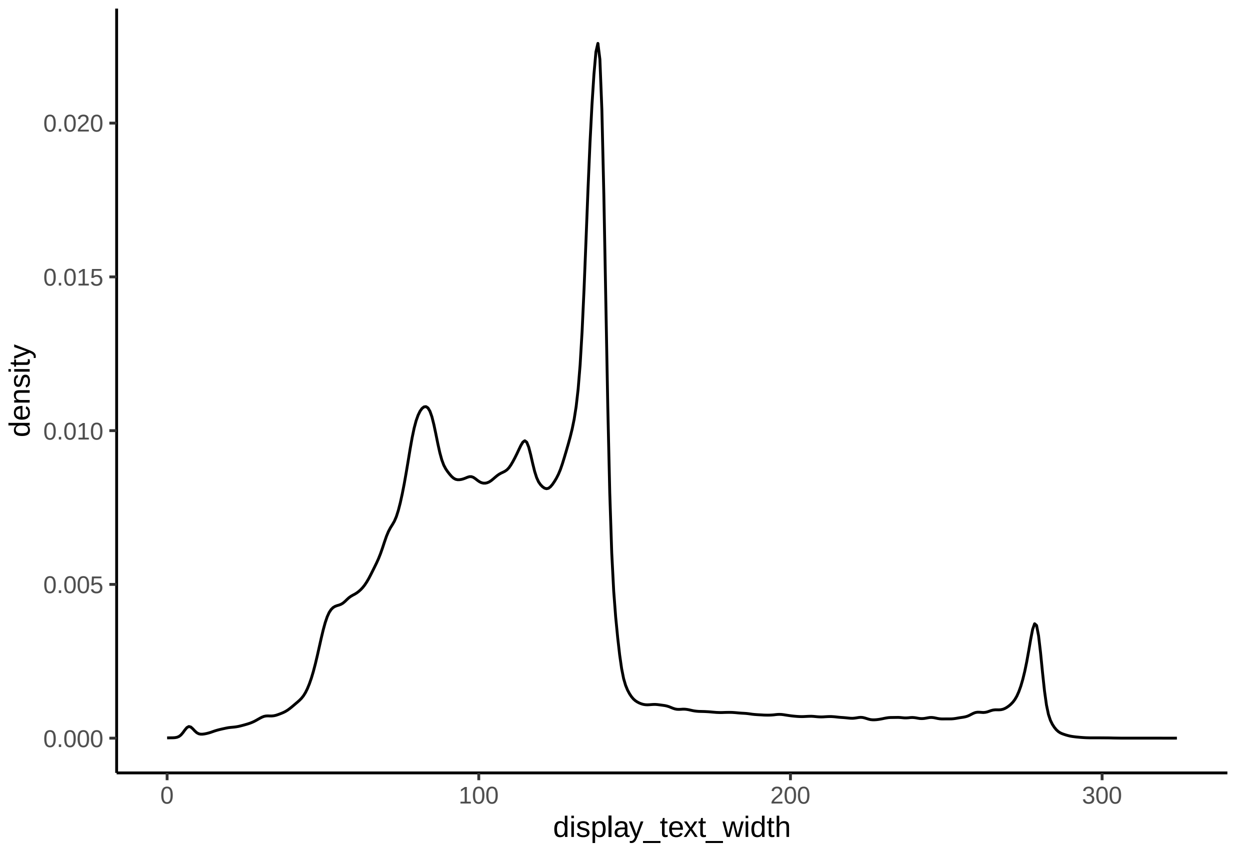
I think when "bins = 20" can better represent the data because it clearly displays the small peak in near 100, a high peak in 150, and tiny peak in 280, just like the other graphs like to show. With 20, it is with enough bins to represent the details. The tendency and flow in the chart looks smooth and clear. The y-axis also display more information than the other bins.

2. Create a density plot for the column `display_text_width` using the `ggplot2` package and `geom_density()` function. Fill the inside of density plot with a color using the `fill=` argument. Try at least four different numbers of smoothing badwith (e.g., 0.2, 1.5, 3, 5) by manipulating the `bw=` argument. Select what you think best represents the data for each. Provide a brief justification for your decision.

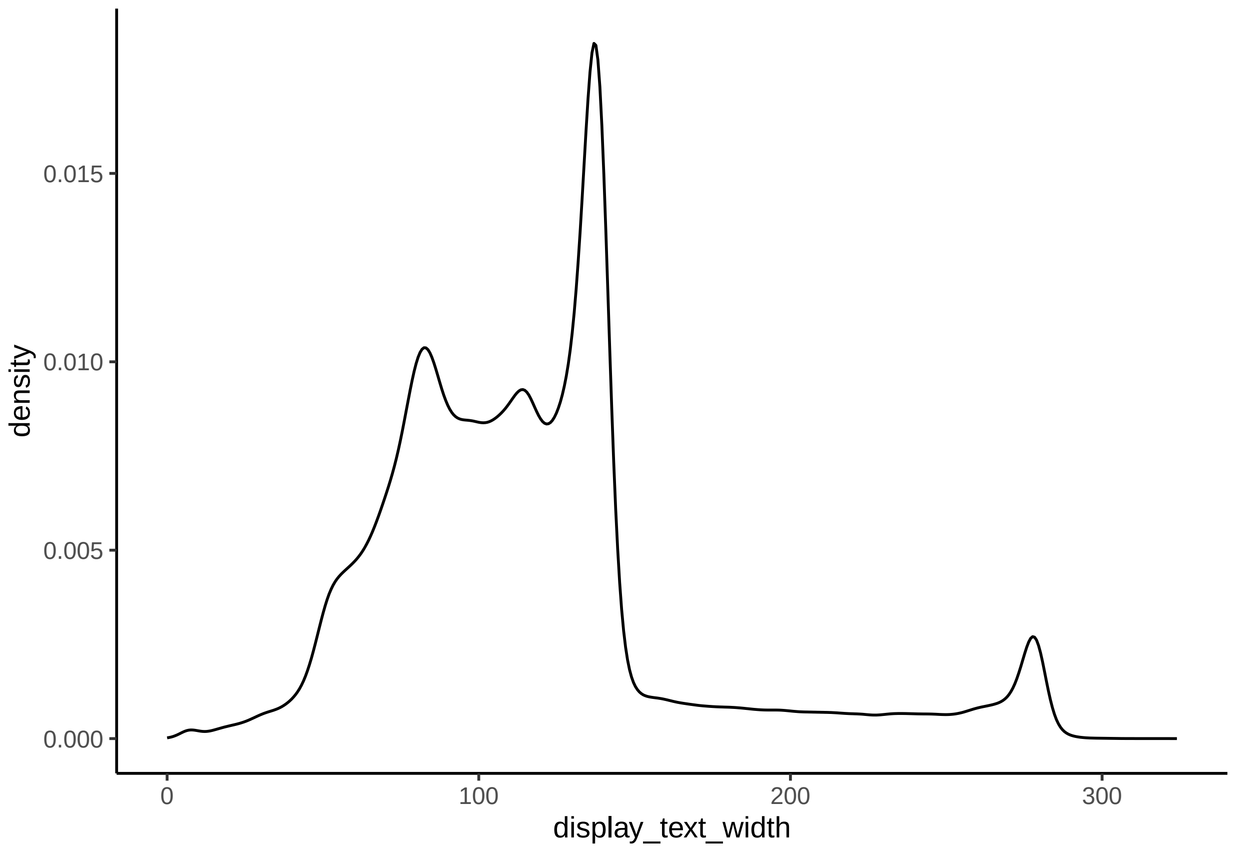
```
library(ggplot2)

ggplot(d, aes(display_text_width))+
  geom_density(bw = 0.2) +
      theme_classic()
```



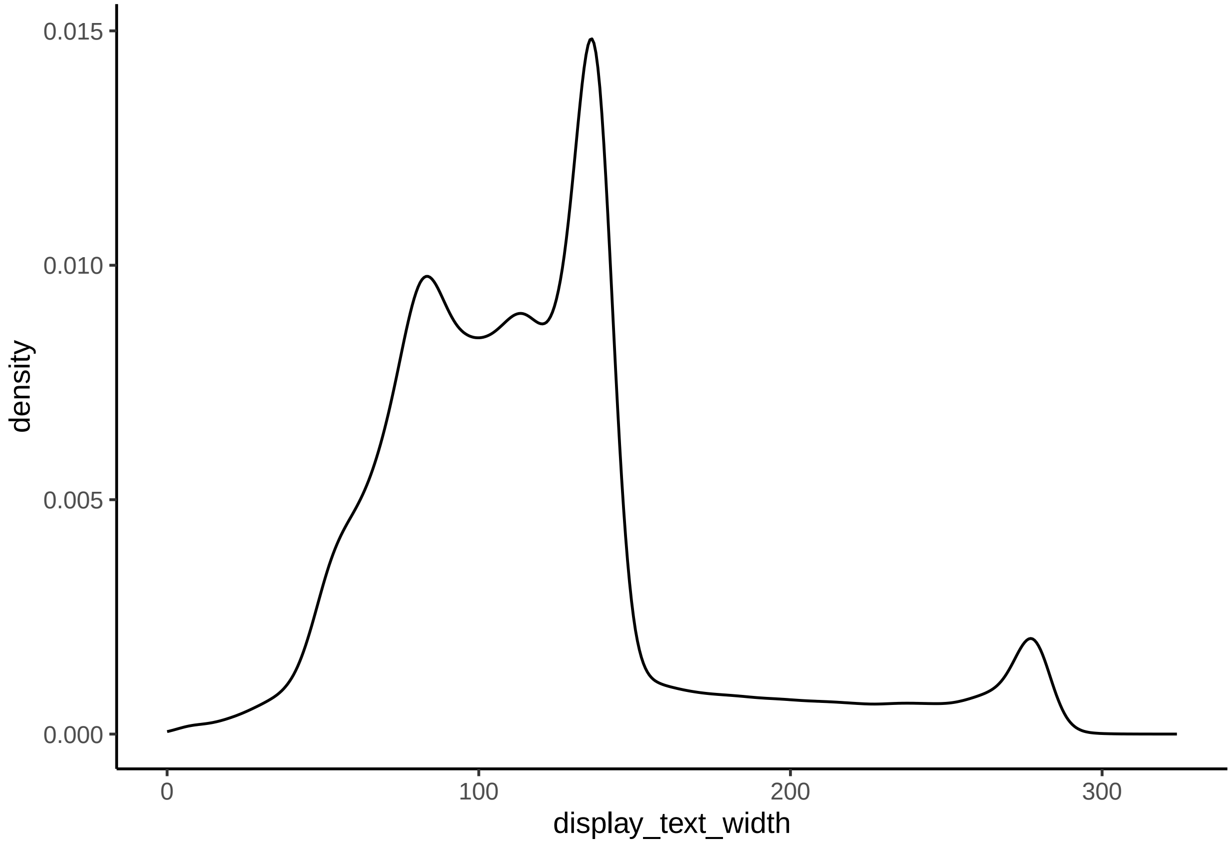
```
ggplot(d, aes(display_text_width))+
  geom_density(bw = 1.5) +
      theme_classic()
```



```
ggplot(d, aes(display_text_width))+
  geom_density(bw = 3) +
      theme_classic()
```



```
ggplot(d, aes(display_text_width))+
  geom_density(bw = 5) +
      theme_classic()
```

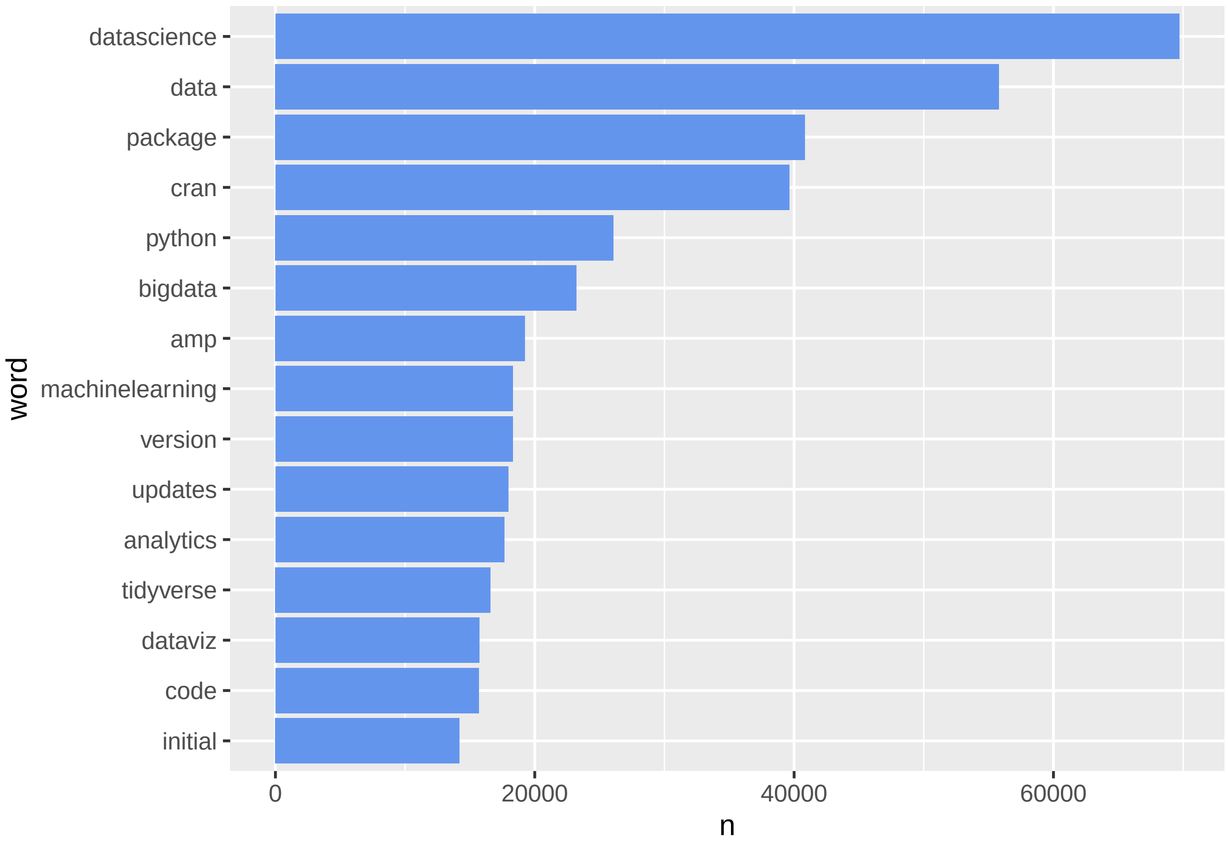


I think when "bw == 3", it can better represent the graph because it clearly represents the details of previous histogram (i.e., the peaks and flow). The "bw == 0.2 and 1.5" are undersmooth the density curve, while the "bw ==5" is oversmooth the density curve.

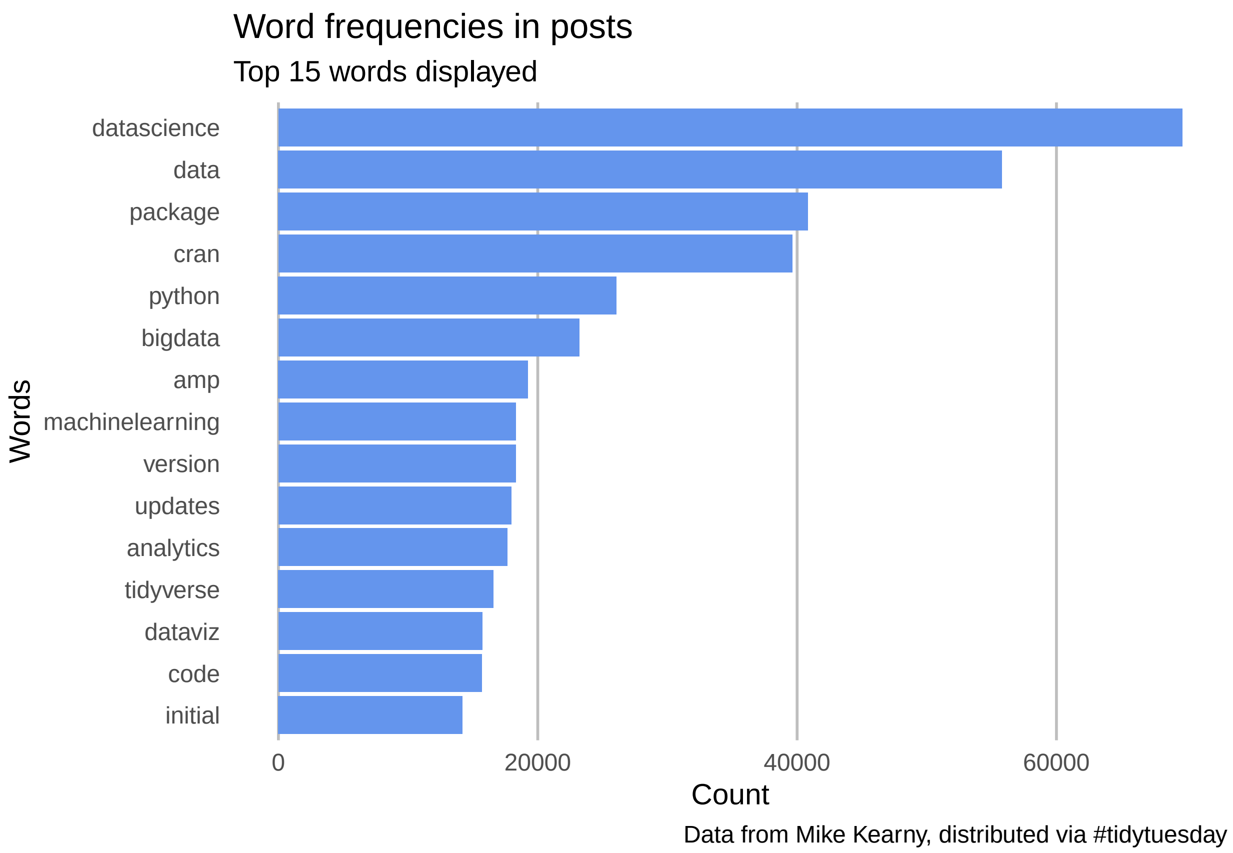
Barplot

3. Using the information `text` column, create the following figure of the 15 most common words represented in these posts by using the `ggplot2()` package and `geom_col()` function. Remove the stop words, and also exclude the words such as 't.co','https','http','rt','rstats'.

```
library(tidytext)
library(tidyverse)
library(dplyr)

#remove words
remove_words    <- c("t.co","https","http","rt","rstats")


word <- d %>%
  select(user_id, text) %>%
  unnest_tokens(word, text)

words_15  <- word %>%
      anti_join(stop_words) %>%
      filter(!word %in% remove_words) %>%
      count(word, sort = TRUE) %>%
      mutate(word = fct_reorder(word, n)) %>%
      slice(1:15)

ggplot(words_15, aes(n, word)) +
      geom_col(fill = "cornflowerblue")
```



4. Style the plot so it (mostly) matches the below. It does not need to be exact, but it should be close.


Word frequencies in posts
Top 15 words displayed
Words
datascience
data
package
cran
python
bigdata
amp
machinelearning
version
updates
analytics
tidyverse
dataviz
code
initial
0
20000
40000
60000
Count
Data from Mike Kearny, distributed via #tidytuesday