Lab 1

Seulbi Lee

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.

```
d <- rio::import("~/OneDrive - University Of Oregon/Year 2 (22-23)/Winter 2023/EDUC 652 Zopluoglu Datavi
                 setclass = "tbl_df")
```

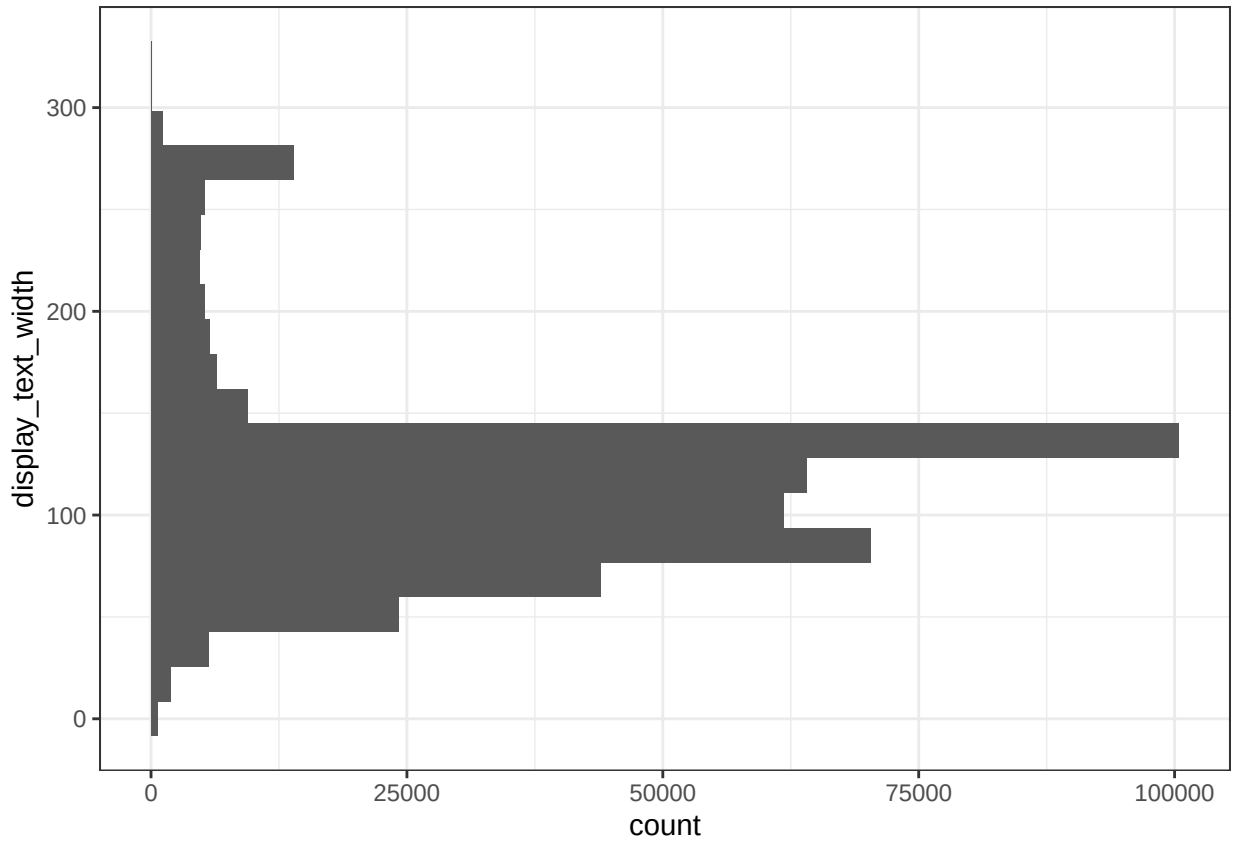
If you need help with processing text data, please revisit the notebook introduced in Week 1.

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

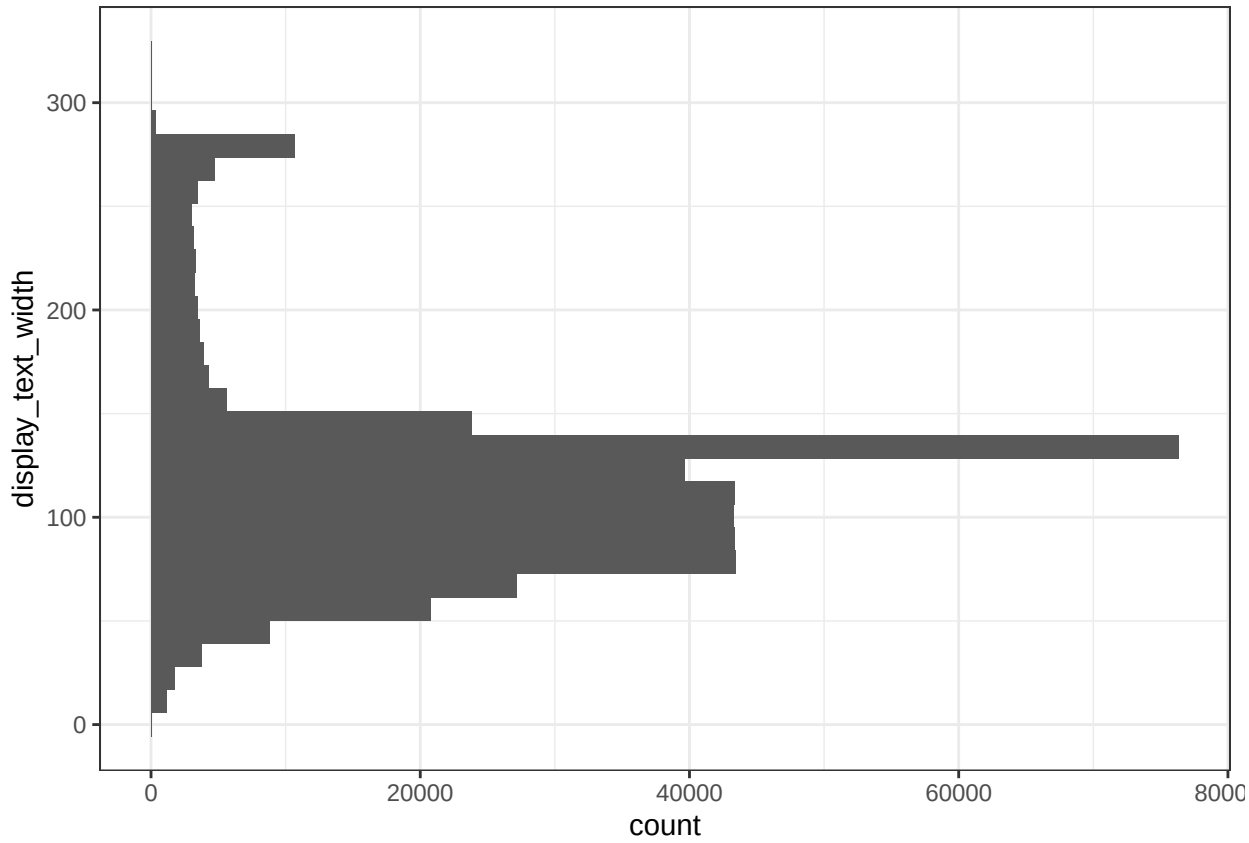
Histogram and Density plots

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.

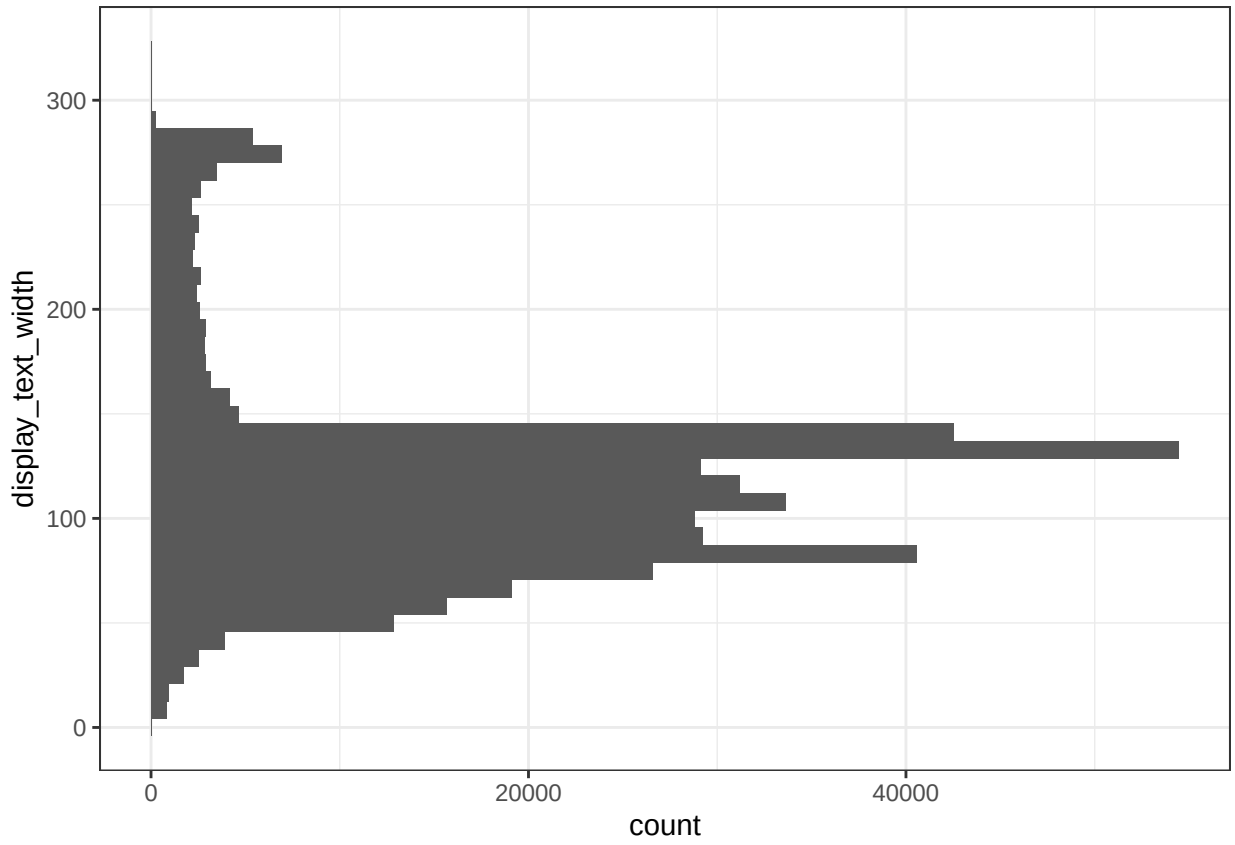
```
# knitted document wouldn't have shown the histograms without importing the rds in this chunk
d <- rio::import("~/OneDrive - University Of Oregon/Year 2 (22-23)/Winter 2023/EDUC 652 Zopluoglu Datavi
                 setclass = "tbl_df")

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



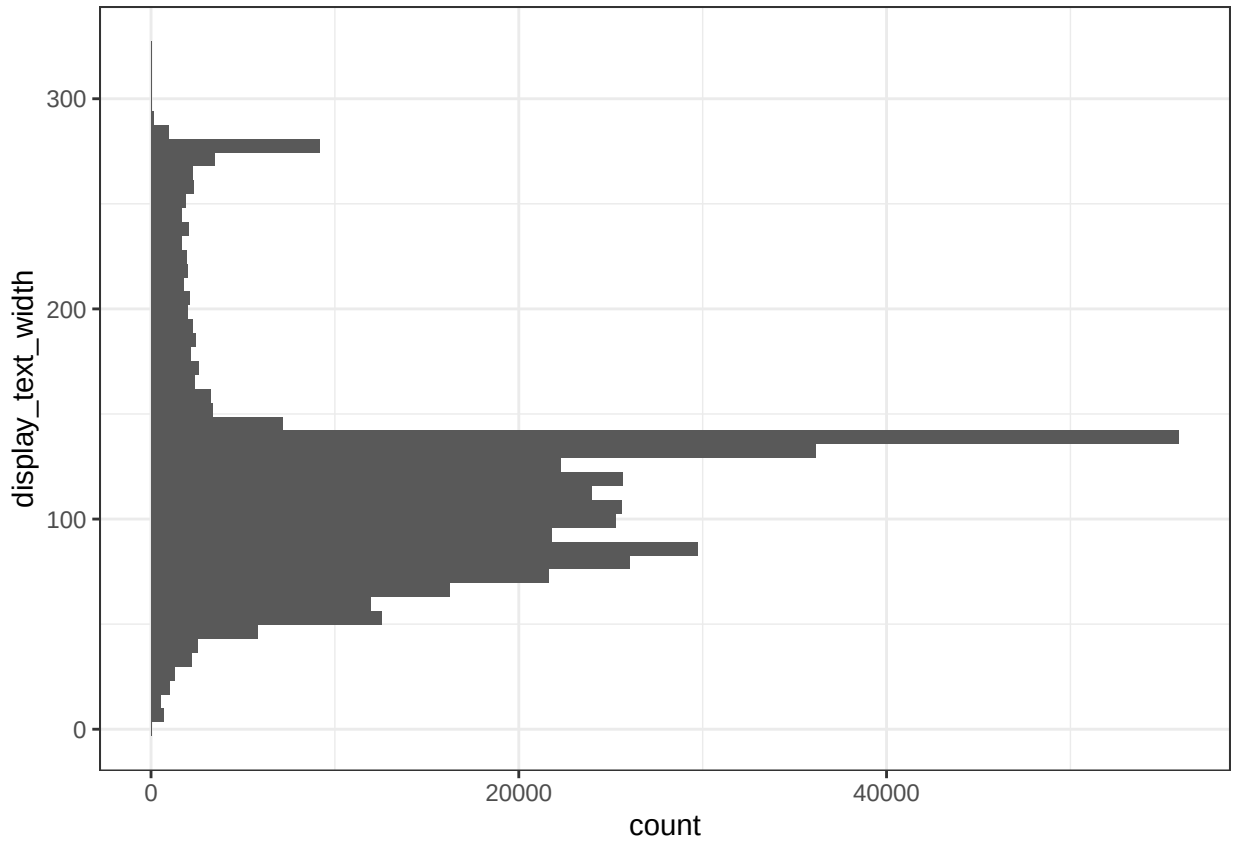

```
ggplot(d, aes(y=display_text_width)) +
  geom_histogram(bins=30) +
  theme_bw()
```




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



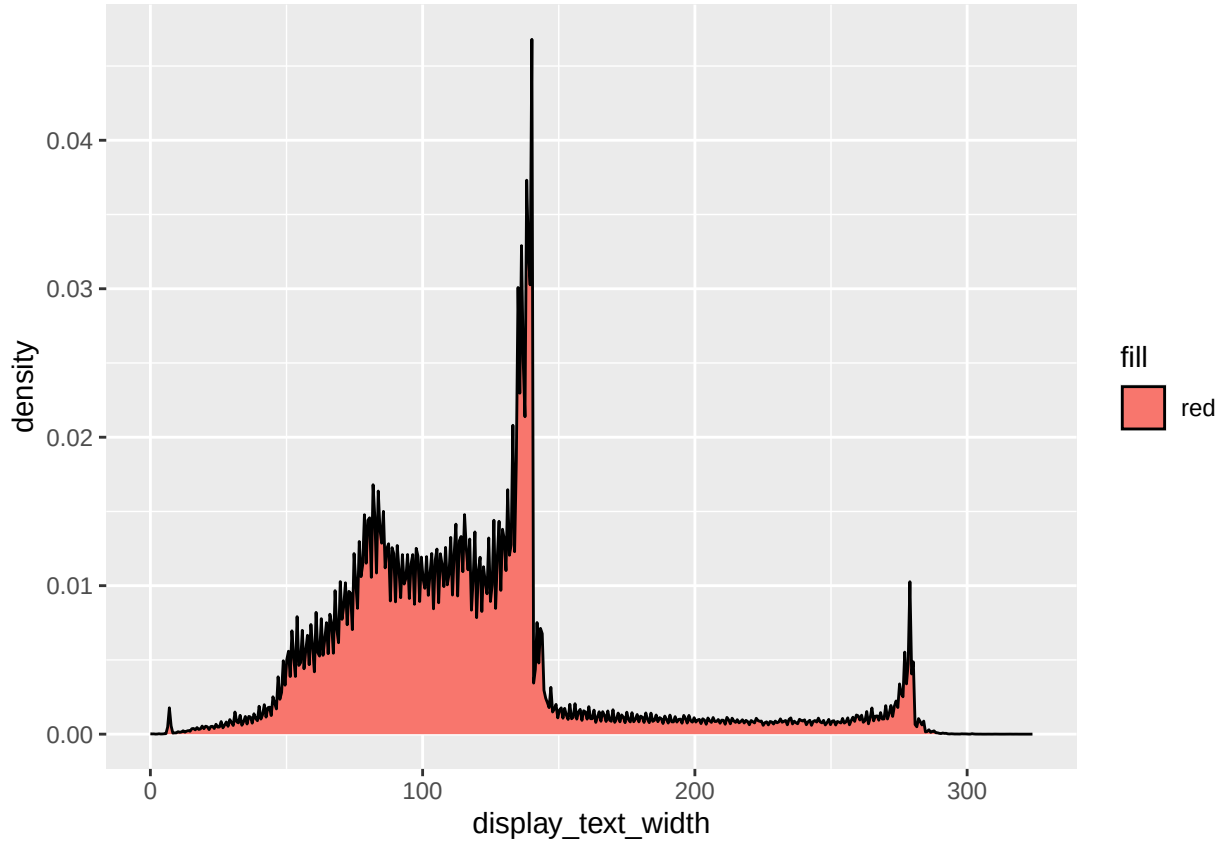

```
ggplot(d, aes(y=display_text_width)) +
  geom_histogram(bins=50) +
  theme_bw()
```



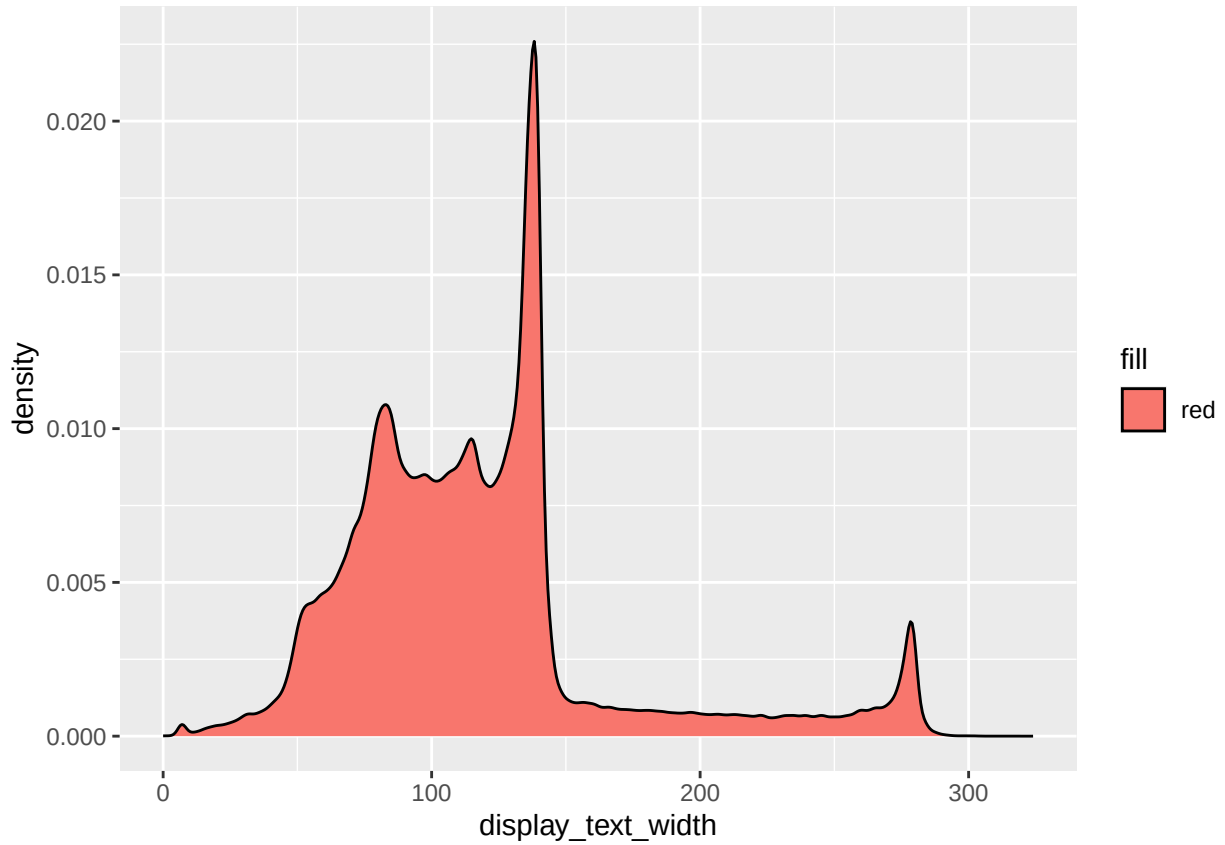
Answer: 40 bins seem to deliver enough information without making too many (and thin) bars because graphs with 40 and 50 bins have similar trend while those with 20 and 30 bins do not demonstrate all variance that can be observed in those with 40 and 50 bins.

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 bandwidth (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.

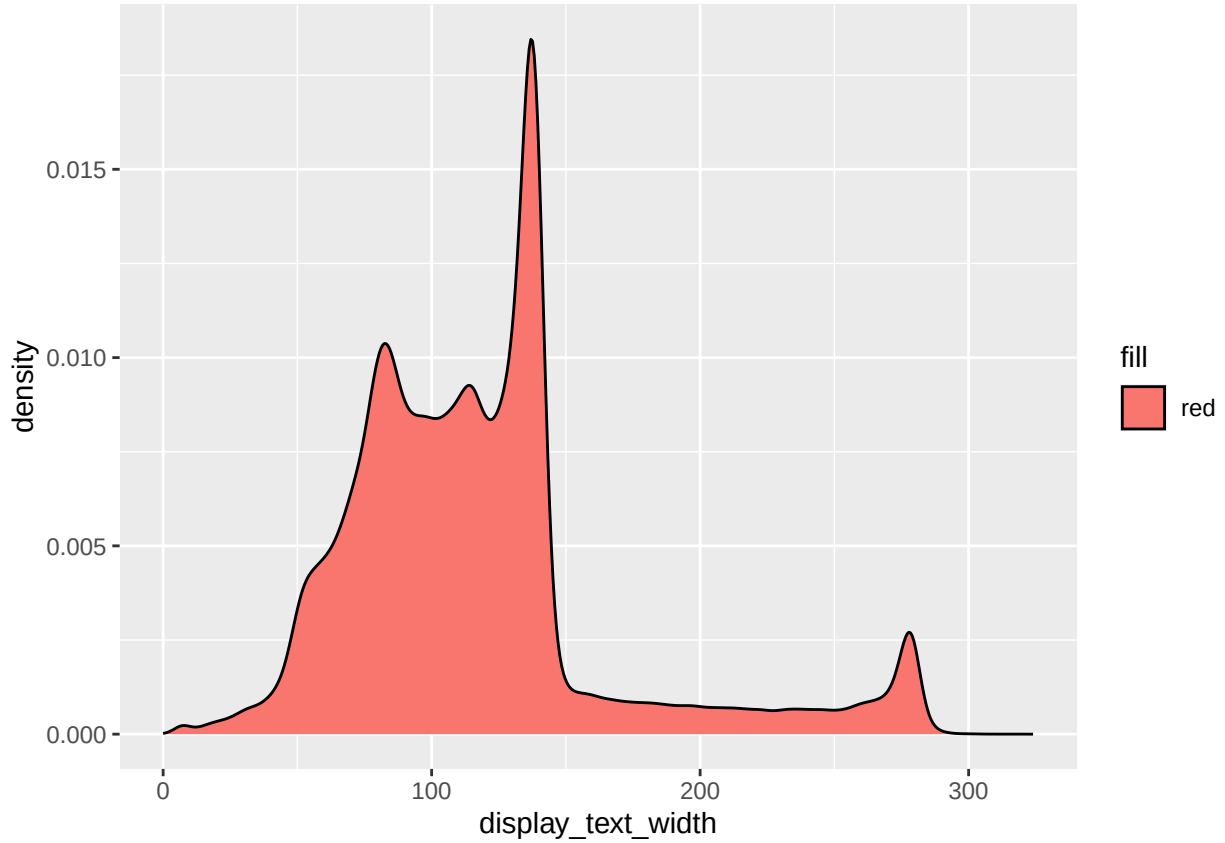
```
# kept having an error if it wasn't imported again
d <- rio::import("~/OneDrive - University Of Oregon/Year 2 (22-23)/Winter 2023/EDUC 652 Zopluoglu Datavi
                 setclass = "tbl_df")

ggplot(d, aes(display_text_width, fill="red")) +
  geom_density(bw=0.2)
```



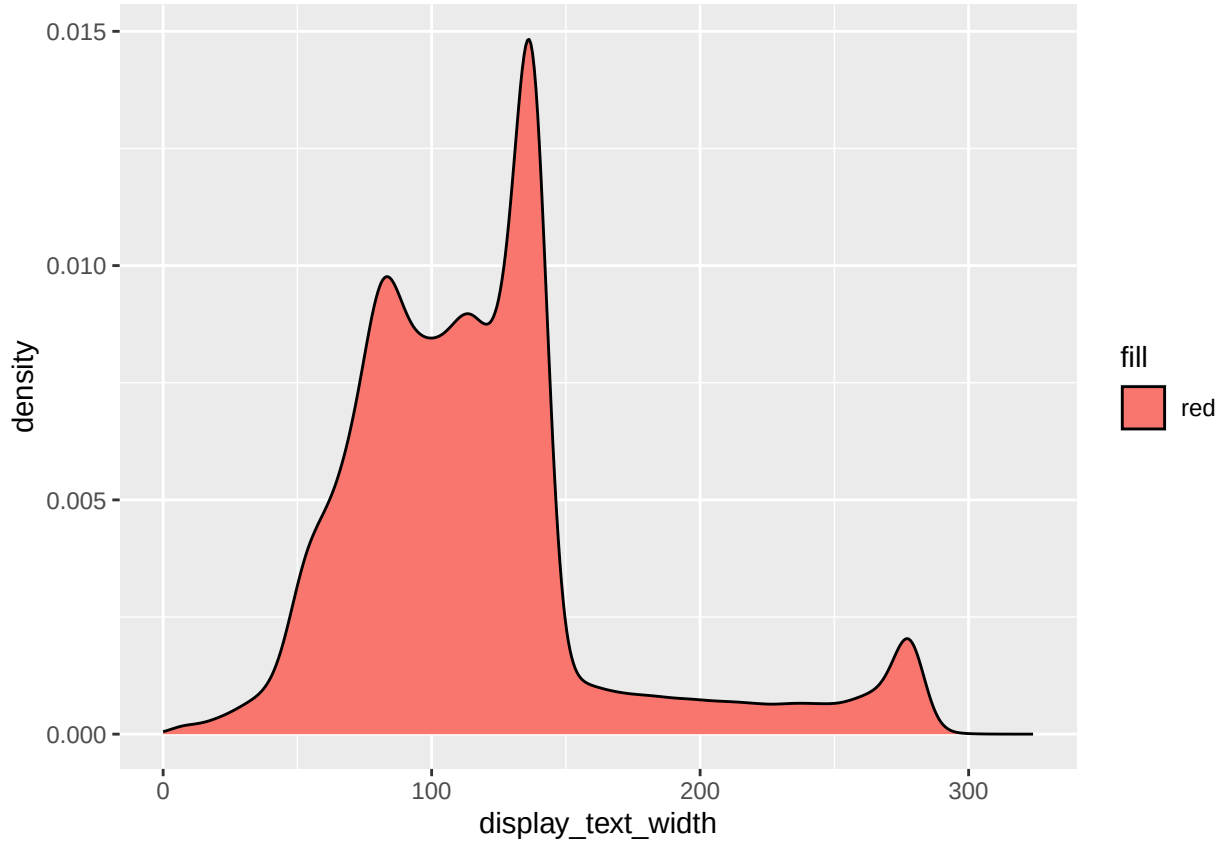

```
ggplot(d, aes(display_text_width, fill="red")) +
  geom_density(bw=1.5)
```




```
ggplot(d, aes(display_text_width, fill="red")) +
  geom_density(bw=3)
```




```
ggplot(d, aes(display_text_width, fill="red")) +
  geom_density(bw=5)
```



Answer: I would prefer bw=3 due to its adequate smoothness and inclusion of variances compared to those with bw=0.2 or 1.5.

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'.

```
df <- tibble(
  paragraph = seq_along(d$text),
  text = d$text
)

tidy_words <- df %>%
  unnest_tokens(word, text)

tidy_words %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 529,704 x 2
##    word        n
##    <chr>   <int>
##  1 rstats 430687
##  2 t.co   404960
```

```
##  3 https  310647
##  4 to     139471
##  5 r      128458
##  6 the    126901
##  7 in     111910
##  8 http   105427
##  9 a      101742
## 10 and     96096
## # ... with 529,694 more rows
```

```
stop_words
```

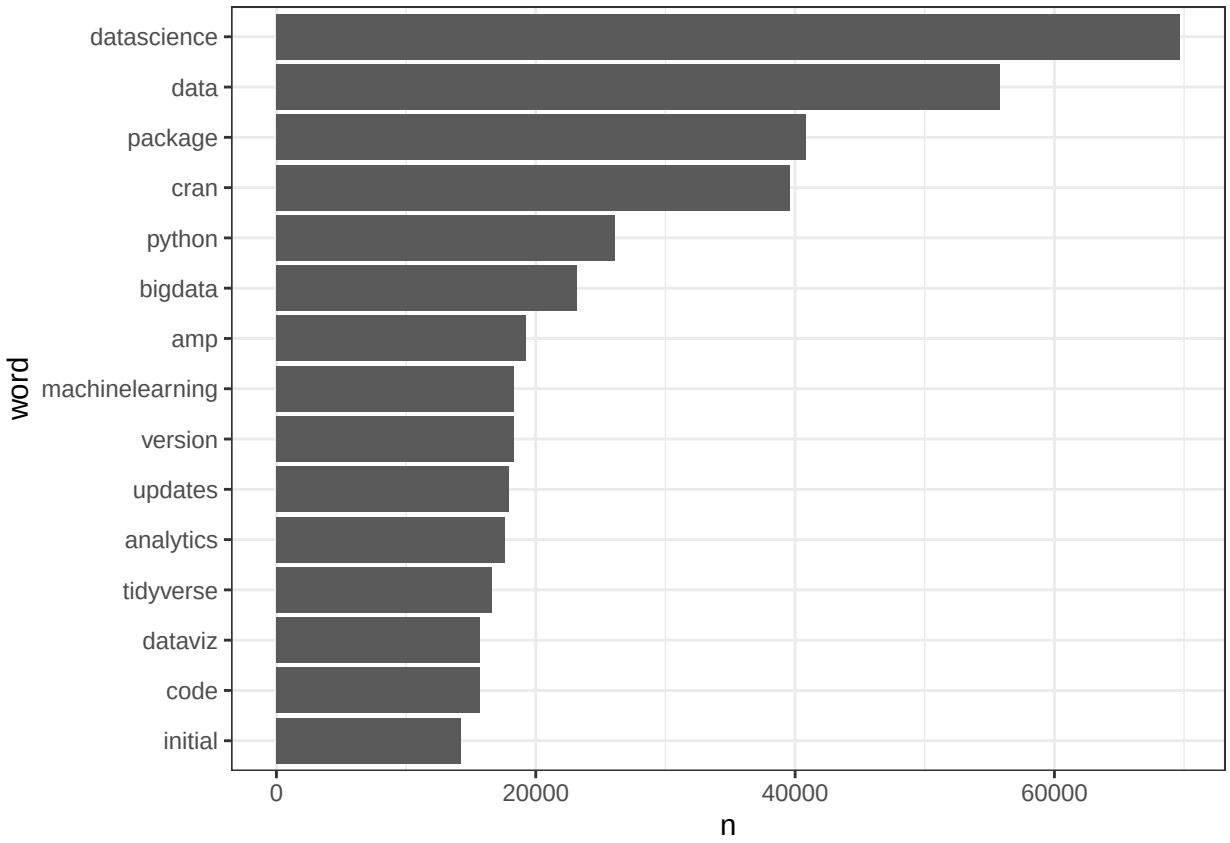
```
## # A tibble: 1,149 x 2
##    word        lexicon
##    <chr>       <chr>
##  1 a           SMART
##  2 a's         SMART
##  3 able        SMART
##  4 about       SMART
##  5 above       SMART
##  6 according   SMART
##  7 accordingly SMART
##  8 across      SMART
##  9 actually    SMART
## 10 after       SMART
## # ... with 1,139 more rows
```

```
remove_words <- c('t.co','https','http','rt','rstats')

cool_graph <-
tidy_words %>%
  anti_join(stop_words) %>%
  count(word, sort = TRUE) %>%
  filter(!word %in% c('t.co','https','http','rt','rstats')) %>%
  mutate(word = reorder(word, n)) %>%
  slice(1:15) %>%
  ggplot(aes(n, word)) +
    geom_col(fill = "#5A5A5A")+
    theme_bw()
```

```
## Joining, by = "word"
```

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

```
cool_graph
```


datascience
data
package
cran
python
bigdata
amp
machinelearning
version
updates
analytics
tidyverse
dataviz
code
initial
word
0
20000
40000
60000
n