Project3

Zeyu Li

2023-10-14

Notes

Dear TA:

Sometimes when I open my html file, the name of Beyoncé can not be presented correctly. And because I also use “Beyoncé” as conditions for filter in my code, if it’s not correctly presented, some code can’t work well. Please use “File”-“Reopen with Encoding”-“UTF-8” for the rmd file, it may help. I also submit a pdf file from dropbox in case the code can not present correctly. Thank you for your consideration!

Prepare packages

```
library("tidyverse")
```

```
## ── Attaching core tidyverse packages ──────────────────────────────────────── tidyverse 2.
0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────────────────────────────── t
idyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library("here")
```

```
## here() starts at D:/JHU/Term 1/Statistical Computing/Sta_com_proj3
```

```
library("lubridate")
library("ggplot2")
library("forcats")
library("stringr")
library("tidytext")
library("wordcloud")
```

```
## 载入需要的程辑包：RColorBrewer
```

```
library("textdata")
```

```
rds_files <- c("b_lyrics.RDS", "ts_lyrics.RDS", "sales.RDS")
if (!dir.exists(here("data"))) {
    dir.create(here("data"))
}

## Check whether we have all 3 files
if (any(!file.exists(here("data", rds_files)))) {
    ## If we don't, then download the data
    b_lyrics <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-09-29/beyonce_lyrics.csv")
    ts_lyrics <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-09-29/taylor_swift_lyrics.csv")
    sales <- readr::read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-09-29/sales.csv")

    ## Then save the data objects to RDS files
    saveRDS(b_lyrics, file = here("data", "b_lyrics.RDS"))
    saveRDS(ts_lyrics, file = here("data", "ts_lyrics.RDS"))
    saveRDS(sales, file = here("data", "sales.RDS"))
}
```

```
b_lyrics <- readRDS(here("data", "b_lyrics.RDS"))
ts_lyrics <- readRDS(here("data", "ts_lyrics.RDS"))
sales <- readRDS(here("data", "sales.RDS"))
```

Part 1: Explore album sales

In this section, the goal is to explore the sales of studio albums from Beyoncé and Taylor Swift.

Notes

- In each of the subsections below that ask you to create a plot, you must create a title, subtitle, x-axis label, and y-axis label with units where applicable. For example, if your axis says “sales” as an axis label, change it to “sales (in millions)”.

Part 1A

In this section, we will do some data wrangling.

1. Use `lubridate` to create a column called `released` that is a `Date` class. However, to be able to do this, you first need to use `stringr` to search for pattern that matches things like this “(US)[51]” in a string like this “September 1, 2006 (US)[51]” and removes them. (**Note**: to get full credit, you must create the regular expression).
2. Use `forcats` to create a factor called `country` (**Note**: you may need to collapse some factor levels).
3. Transform the `sales` into a unit that is album sales in millions of dollars.
4. Keep only album sales from the UK, the US or the World.
5. Auto print your final wrangled tibble data frame.

```
#1
partla <- sales %>%
  mutate(
    released = str_remove_all(released, "\\([A-Z]{2}\\)\\[[0-9]{2}\\]")
    )
partla$released = mdy(partla$released)
#2
country_levels <- c("AUS","CAN","FR","FRA","JPN","UK","US","World","WW")
partla$country = factor(partla$country, levels = country_levels)
partla$country <- fct_collapse(partla$country, "World" = c("World","WW"))
#3
partla$sales = partla$sales/1000000
#4
partla <- filter(partla, country == "UK" | country == "US" | country == "World")
#5
partla
```

```
## # A tibble: 36 × 8
##    artist       title        country  sales released   re_release  label formats
##    <chr>        <chr>        <fct>    <dbl> <date>     <chr>       <chr> <chr>
##  1 Taylor Swift Taylor Swift US       5.72  2006-10-24 March 18, … Big … CD, CD…
##  2 Taylor Swift Fearless     World   12     2008-11-11 October 27… Big … CD, CD…
##  3 Taylor Swift Fearless     US       7.18  2008-11-11 October 27… Big … CD, CD…
##  4 Taylor Swift Fearless     UK       0.609 2008-11-11 October 27… Big … CD, CD…
##  5 Taylor Swift Speak Now    World    5     2010-10-25 <NA>        Big … CD, CD…
##  6 Taylor Swift Speak Now    US       4.69  2010-10-25 <NA>        Big … CD, CD…
##  7 Taylor Swift Speak Now    UK       0.169 2010-10-25 <NA>        Big … CD, CD…
##  8 Taylor Swift Red          World    6     2012-10-22 <NA>        Big … CD, CD…
##  9 Taylor Swift Red          US       4.46  2012-10-22 <NA>        Big … CD, CD…
## 10 Taylor Swift Red          UK       0.693 2012-10-22 <NA>        Big … CD, CD…
## # ℹ 26 more rows
```

Part 1B

In this section, we will do some more data wrangling followed by summarization using wrangled data from Part 1A.

1. Keep only album sales from the US.
2. Create a new column called `years_since_release` corresponding to the number of years since the release of each album from Beyoncé and Taylor Swift. This should be a whole number and you should round down to “14” if you get a non-whole number like “14.12” years. (**Hint**: you may find the `interval()` function from `lubridate` helpful here, but this not the only way to do this.)
3. Calculate the most recent, oldest, and the median years since albums were released for both Beyoncé and Taylor Swift.

```
#1
partlb <- filter(partla, country == "US")
#2
partlb <- partlb %>%
  mutate(
    years_since_release = round(time_length(interval(released, today()), unit = "year"))
  )
#3
partlb %>%
  group_by(artist) %>%
  summarise(
    most_recent = min(years_since_release),
    oldest = max(years_since_release),
    median = median(years_since_release)
  ) -> partlb_table
partlb_table
```

```
## # A tibble: 2 × 4
##   artist       most_recent oldest median
##   <chr>              <dbl>  <dbl>  <dbl>
## 1 Beyoncé                7     20   13.5
## 2 Taylor Swift           4     17   11
```

Part 1C

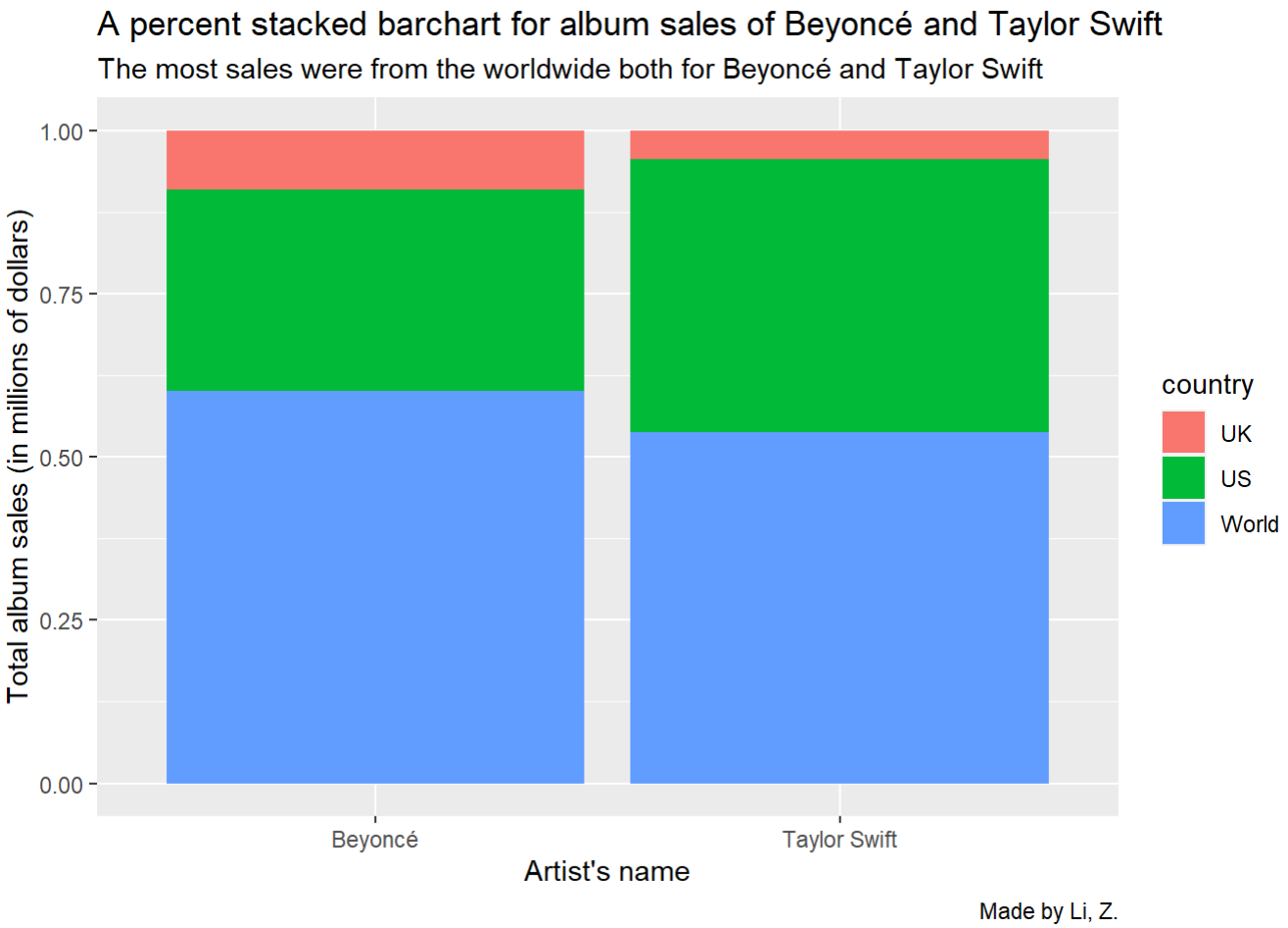
Using the wrangled data from Part 1A:

1. Calculate the total album sales for each artist and for each `country` (only sales from the UK, US, and World).
2. Using the total album sales, create a percent stacked barchart (https://r-graph-gallery.com/48-grouped-barplot-with-ggplot2) using `ggplot2` of the percentage of sales of studio albums (in millions) along the y-axis for the two artists along the x-axis colored by the `country`.

```
#1
partlc <- partla %>%
  group_by(artist, country) %>%
  summarise(
    total_sales = sum(sales)
  )
```

```
## `summarise()` has grouped output by 'artist'. You can override using the
## `.groups` argument.
```

```
#2
ggplot(partlc, aes(fill=country, y=total_sales, x=artist)) +
  geom_bar(position="fill", stat="identity") +
  labs(title = "A percent stacked barchart for album sales of Beyoncé and Taylor Swift", subtitle = "T
most sales were from the worldwide both for Beyoncé and Taylor Swift", x = "Artist's name", y = "Total a
lbum sales (in millions of dollars)", caption = "Made by Li, Z.")
```



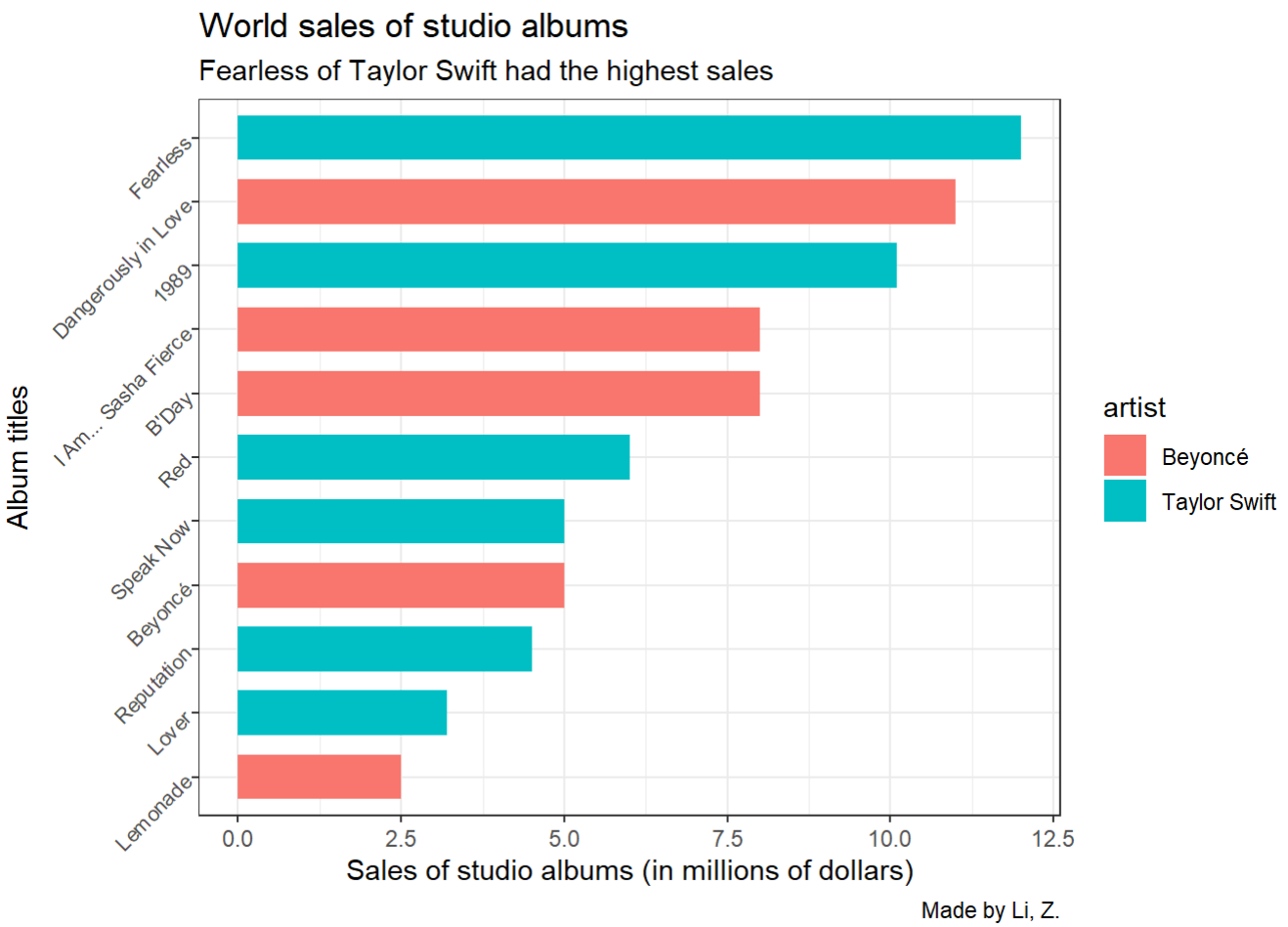
Part 1D

Using the wrangled data from Part 1A, use `ggplot2` to create a bar plot for the sales of studio albums (in millions) along the x-axis for each of the album titles along the y-axis.

Note:

- You only need to consider the global World sales (you can ignore US and UK sales for this part).
- The title of the album must be clearly readable along the y-axis.
- Each bar should be colored by which artist made that album.
- The bars should be ordered from albums with the most sales (top) to the least sales (bottom) (**Note**: you must use functions from `forcats` for this step).

```
partld <- subset(partla, partla$country == "World")
partld %>%
  ggplot(aes(sales, fct_reorder(title, sales), fill = artist)) +
  geom_bar(width = 0.7, stat="identity") +
  theme_bw() +
  theme(axis.text.y.left = element_text(size = 8, angle = 45)) +
  labs(title = "World sales of studio albums", subtitle = "Fearless of Taylor Swift had the highest sale
s", x = "Sales of studio albums (in millions of dollars)", y = "Album titles", caption = "Made by Li,
Z.")
```



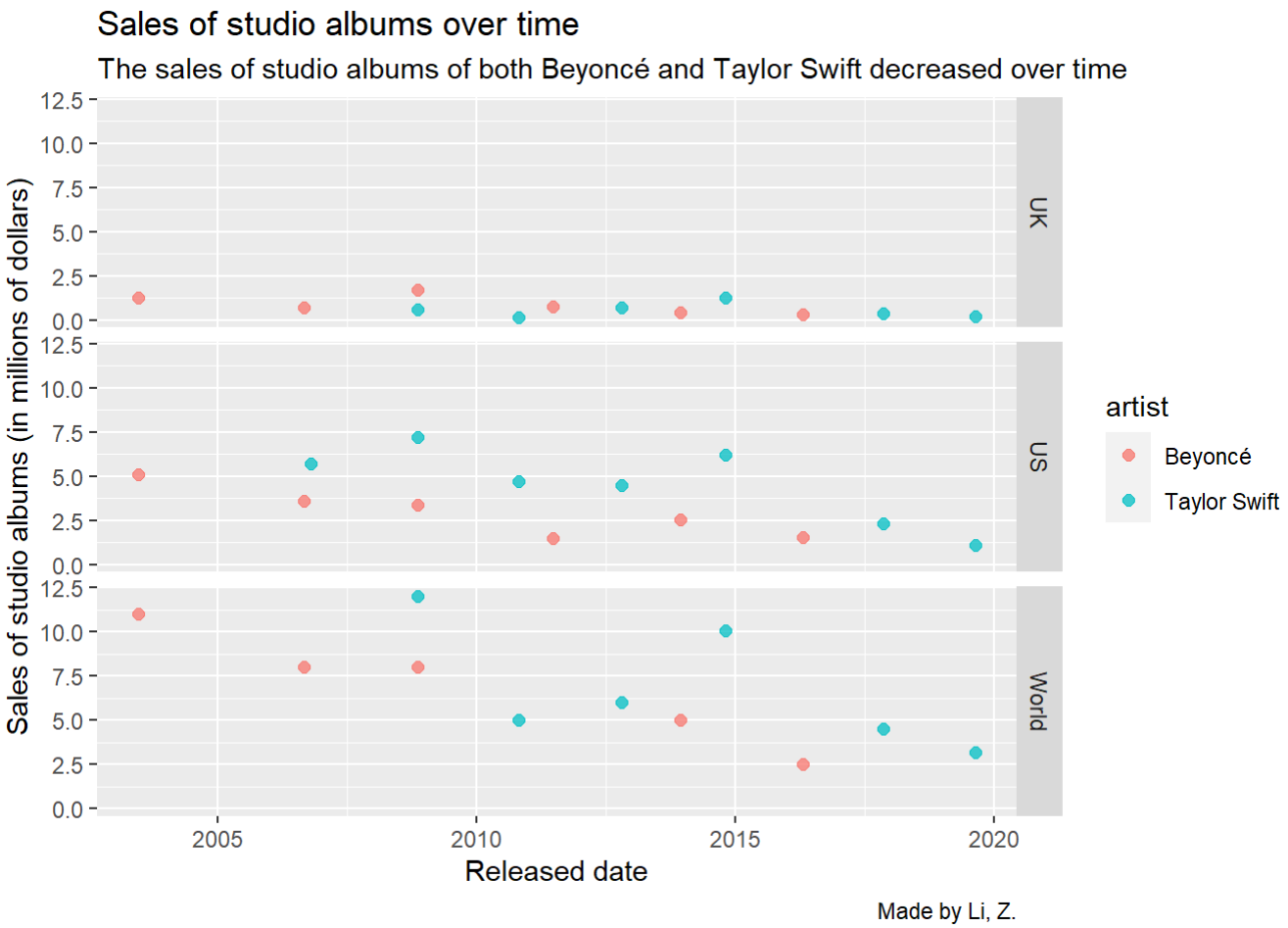
Part 1E

Using the wrangled data from Part 1A, use `ggplot2` to create a scatter plot of sales of studio albums (in millions) along the y-axis by the released date for each album along the x-axis.

Note:

- The points should be colored by the artist.
- There should be three scatter plots (one for UK, US and world sales) faceted by rows.

```
part1a %>%
  ggplot(aes(released, sales, color = artist)) +
  geom_point(size = 2, alpha = 0.75) +
  facet_grid(country~.) +
  labs(title = "Sales of studio albums over time", subtitle = "The sales of studio albums of both Beyonc
é and Taylor Swift decreased over time", x = "Released date", y = "Sales of studio albums (in millions o
f dollars)", caption = "Made by Li, Z.")
```



Part 2: Exploring sentiment of lyrics

In Part 2, we will explore the lyrics in the `b_lyrics` and `ts_lyrics` datasets.

Part 2A

Using `ts_lyrics`, create a new column called `line` with one line containing the character string for each line of Taylor Swift's songs.

- How many lines in Taylor Swift's lyrics contain the word "hello"? For full credit, show all the rows in `ts_lyrics` that have "hello" in the `line` column and report how many rows there are in total.
- How many lines in Taylor Swift's lyrics contain the word "goodbye"? For full credit, show all the rows in `ts_lyrics` that have "goodbye" in the `line` column and report how many rows there are in total.

```
ts_lines <-
    ts_lyrics %>%
    unnest_tokens(
        output = line,
        input = Lyrics,
        token = "lines"
    )

ts_lines_hello <- subset(ts_lines, grepl("hello", tolower(ts_lines$line)) == T)
ts_lines_hello
```

```
## # A tibble: 6 × 4
##   Artist       Album    Title                 line
##   <chr>        <chr>    <chr>                 <chr>
## 1 Taylor Swift Fearless Love Story            "and say, \"hello\""
## 2 Taylor Swift Red      I Almost Do           "that i can't say \"hello\" to y…
## 3 Taylor Swift Red      Everything Has Changed "'cause all i know is we said he…
## 4 Taylor Swift Red      Everything Has Changed "'cause all i know is we said he…
## 5 Taylor Swift Red      Everything Has Changed "all i know is we said hello"
## 6 Taylor Swift Red      Everything Has Changed "all i know is we said hello"
```

```
nrow(ts_lines_hello)
```

```
## [1] 6
```

```
#another solution
detect_line <- function(data, word) {
  a <- subset(data, grepl(word, tolower(data$line)) == T)
  print(a)
  nrow(a)
}
detect_line(ts_lines, "hello")
```

```
## # A tibble: 6 × 4
##   Artist       Album    Title                 line
##   <chr>        <chr>    <chr>                 <chr>
## 1 Taylor Swift Fearless Love Story            "and say, \"hello\""
## 2 Taylor Swift Red      I Almost Do           "that i can't say \"hello\" to y…
## 3 Taylor Swift Red      Everything Has Changed "'cause all i know is we said he…
## 4 Taylor Swift Red      Everything Has Changed "'cause all i know is we said he…
## 5 Taylor Swift Red      Everything Has Changed "all i know is we said hello"
## 6 Taylor Swift Red      Everything Has Changed "all i know is we said hello"
```

```
## [1] 6
```

```
detect_line(ts_lines, "goodbye")
```

```
## # A tibble: 12 × 4
##    Artist       Album        Title                     line
##    <chr>        <chr>        <chr>                     <chr>
##  1 Taylor Swift Taylor Swift Tied Together With A Smile "goodbye, baby"
##  2 Taylor Swift Speak Now    Mine                      "braced myself for the …
##  3 Taylor Swift Speak Now    Back to December          "you gave me all your l…
##  4 Taylor Swift Speak Now    Long Live                 "and force us into a go…
##  5 Taylor Swift Red          I Almost Do               "and risk another goodb…
##  6 Taylor Swift Red          Come Back Be Here         "stumbled through the l…
##  7 Taylor Swift 1989         All You Had to Do Was Stay "but people like me are…
##  8 Taylor Swift reputation   Getaway Car               "said goodbye in "
##  9 Taylor Swift reputation   Getaway Car               "said goodbye in "
## 10 Taylor Swift Lover        Death By A Thousand Cuts  "saying goodbye is deat…
## 11 Taylor Swift Lover        Death By A Thousand Cuts  "'cause saying goodbye …
## 12 Taylor Swift Lover        Daylight                  "i'll tell you truth, b…
```

```
## [1] 12
```

Part 2B

Repeat the same analysis for `b_lyrics` as described in Part 2A.

```
detect_line(b_lyrics, "hello")
```

```
## # A tibble: 91 × 6
##    line                        song_id song_name artist_id artist_name song_line
##    <chr>                         <dbl> <chr>         <dbl> <chr>           <dbl>
##  1 Hello world, well I just b… 2220711 "Dreamgi…       498 Beyoncé             5
##  2 Hello Stevie, How you feel… 1981227 "Fingert…       498 Beyoncé             6
##  3 Fellow great Americans, he… 2715227 "FREEDOM…       498 Beyoncé            52
##  4 You had me at hello (Hello)   80249 "Hello"         498 Beyoncé            15
##  5 Hello (Hello)                 80249 "Hello"         498 Beyoncé            16
##  6 Hello (Hello)                 80249 "Hello"         498 Beyoncé            17
##  7 You had me at hello (Hello)   80249 "Hello"         498 Beyoncé            18
##  8 Hello (Hello)                 80249 "Hello"         498 Beyoncé            19
##  9 Hello (Hello)                 80249 "Hello"         498 Beyoncé            20
## 10 'Cause you had me at hello…   80249 "Hello"         498 Beyoncé            24
## # ℹ 81 more rows
```

```
## [1] 91
```

```
detect_line(b_lyrics, "goodbye")
```

```
## # A tibble: 12 × 6
##    line                          song_id song_name artist_id artist_name song_line
##    <chr>                           <dbl> <chr>         <dbl> <chr>           <dbl>
##  1 We only said goodbye with …  139043 Back to …       498 Beyoncé            12
##  2 We only said goodbye with …  139043 Back to …       498 Beyoncé            21
##  3 We only said goodbye with …  139043 Back to …       498 Beyoncé            24
##  4 Thank God, I found the goo…   51492 Best Thi…       498 Beyoncé            38
##  5 Thank God, I found the goo… 1946060 Best Thi…       498 Beyoncé            42
##  6 Thank God, I found the goo… 4241137 Best Thi…       498 Beyoncé            38
##  7 It's so hard to say goodbye  435491 Gift fro…       498 Beyoncé            23
##  8 I never want to say goodbye  435491 Gift fro…       498 Beyoncé            24
##  9 I never, ever want to say …  435491 Gift fro…       498 Beyoncé            25
## 10 We've got to say goodbye    1844620 Hard To …       498 Beyoncé            29
## 11 Don't have to say goodbye   1224115 Slow Love       498 Beyoncé            42
## 12 Somewhere between hi and g…  141848 Yes             498 Beyoncé            27
```

```
## [1] 12
```

Part 2C

Using the `b_lyrics` dataset,

1. Tokenize each lyrical line by words.
2. Remove the "stopwords".
3. Calculate the total number for each word in the lyrics.
4. Using the "bing" sentiment lexicon, add a column to the summarized data frame adding the "bing" sentiment lexicon.
5. Sort the rows from most frequent to least frequent words.
6. Only keep the top 25 most frequent words.
7. Auto print the wrangled tibble data frame.
8. Use `ggplot2` to create a bar plot with the top words on the y-axis and the frequency of each word on the x-axis. Color each bar by the sentiment of each word from the "bing" sentiment lexicon. Bars should be ordered from most frequent on the top to least frequent on the bottom of the plot.
9. Create a word cloud of the top 25 most frequent words.

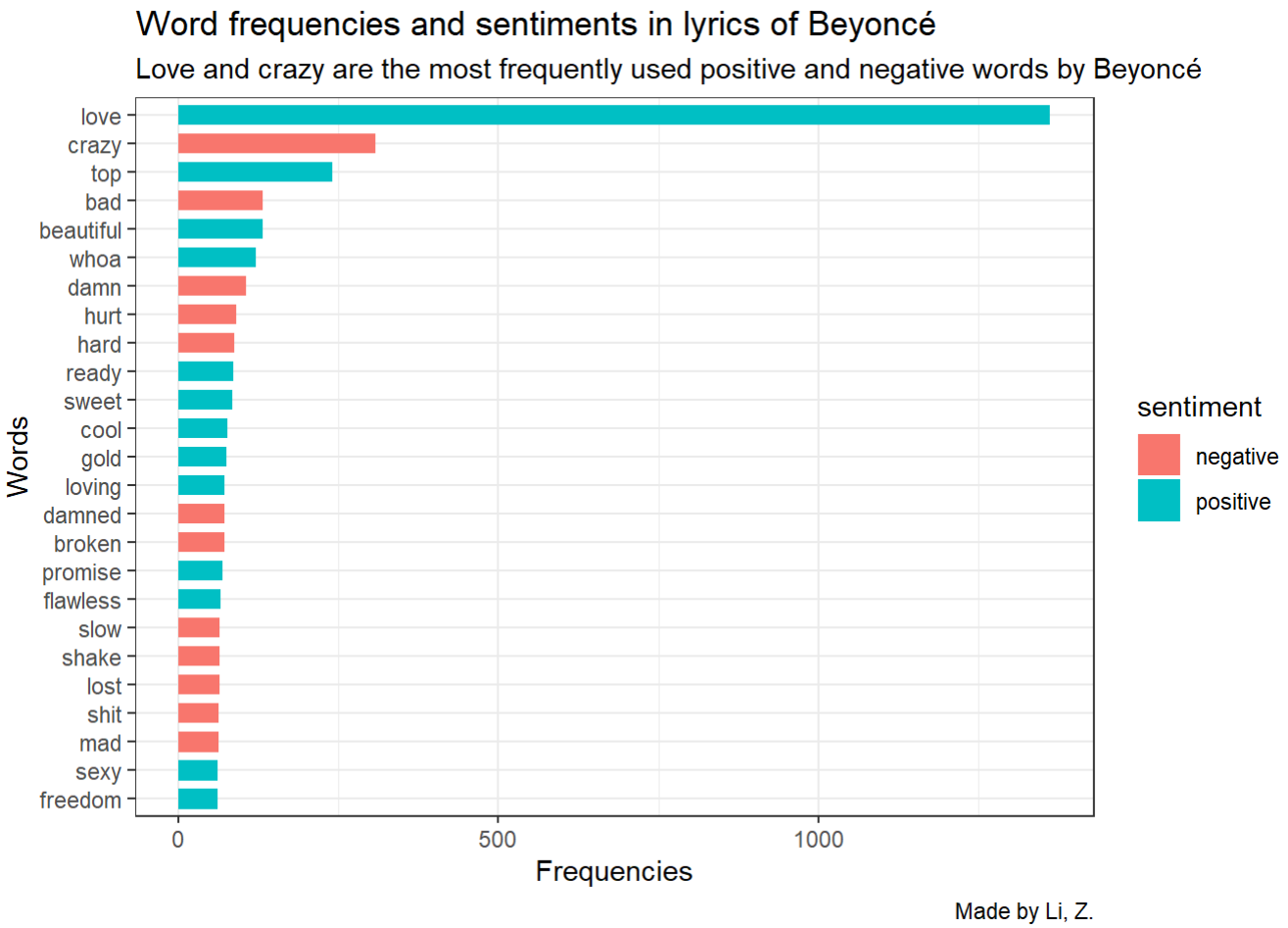
```
#1,2
b_words <- b_lyrics %>%
  unnest_tokens(
    output = word,
    input = line,
    token = "words"
  ) %>%
  anti_join(stop_words)
```

```
## Joining with `by = join_by(word)`
```

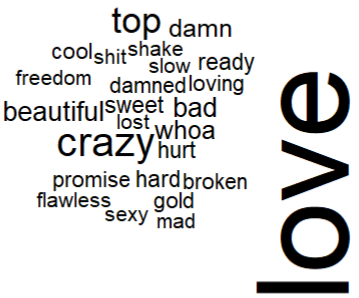
```
#3,5
b_wordnew <- b_words %>%
  count(word, sort = TRUE)
#4
b_wordnew <- inner_join(b_wordnew, get_sentiments("bing"), by = "word")
#6
b_wordnew25 <- b_wordnew[1:25,]
#7
print(b_wordnew25)
```

```
## # A tibble: 25 × 3
##    word          n sentiment
##    <chr>     <int> <chr>
##  1 love       1362 positive
##  2 crazy       308 negative
##  3 top         241 positive
##  4 bad         132 negative
##  5 beautiful   131 positive
##  6 whoa        121 positive
##  7 damn        106 negative
##  8 hurt         90 negative
##  9 hard         87 negative
## 10 ready        85 positive
## # ℹ 15 more rows
```

```
#8
b_wordnew25 %>%
  ggplot(aes(n, fct_reorder(word, n), fill = sentiment)) +
  geom_bar(width = 0.7, stat="identity") +
  theme_bw() +
  labs(title = "Word frequencies and sentiments in lyrics of Beyoncé", subtitle = "Love and crazy are th
e most frequently used positive and negative words by Beyoncé", x = "Frequencies", y = "Words", caption
= "Made by Li, Z.")
```




```
#9
b_wordnew25 %>%
  with(wordcloud(word, n, max.words = 25))
```



Part 2D

Repeat the same analysis as above in Part 2C, but for `ts_lyrics`.

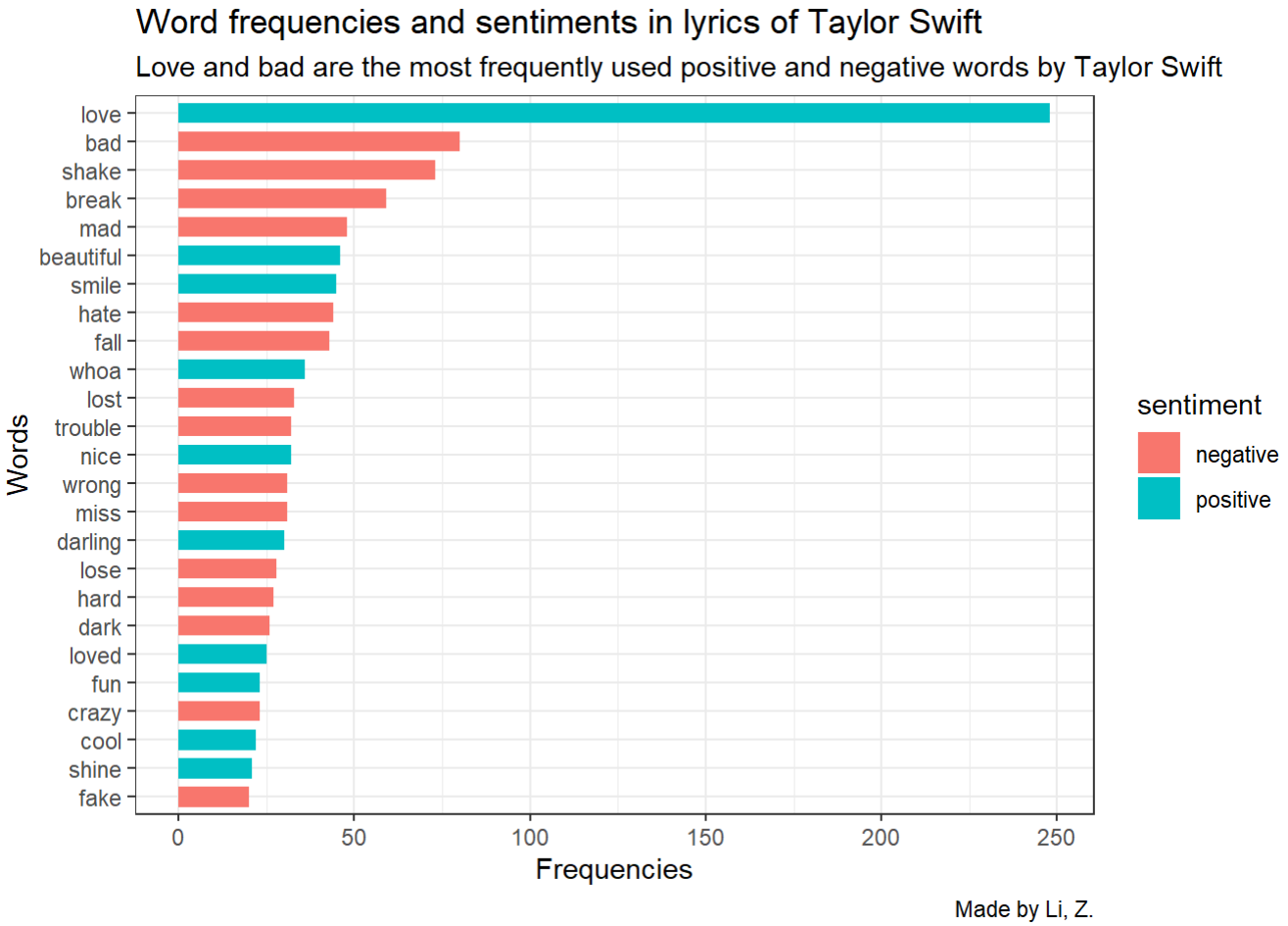
```
#1,2
ts_words <- ts_lyrics %>%
  unnest_tokens(
    output = word,
    input = Lyrics,
    token = "words"
  ) %>%
  anti_join(stop_words)
```

```
## Joining with `by = join_by(word)`
```

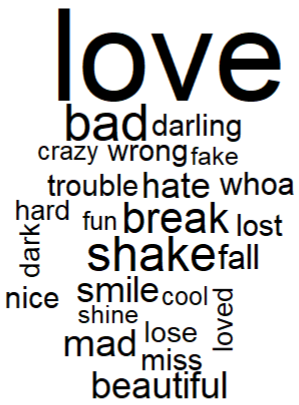
```
#3,5
ts_wordnew <- ts_words %>%
  count(word, sort = TRUE)
#4
ts_wordnew <- inner_join(ts_wordnew, get_sentiments("bing"), by = "word")
#6
ts_wordnew25 <- ts_wordnew[1:25,]
#7
print(ts_wordnew25)
```

```
## # A tibble: 25 × 3
##    word          n sentiment
##    <chr>     <int> <chr>
##  1 love        248 positive
##  2 bad          80 negative
##  3 shake        73 negative
##  4 break        59 negative
##  5 mad          48 negative
##  6 beautiful    46 positive
##  7 smile        45 positive
##  8 hate         44 negative
##  9 fall         43 negative
## 10 whoa         36 positive
## # ℹ 15 more rows
```

```
#8
ts_wordnew25 %>%
  ggplot(aes(n, fct_reorder(word, n), fill = sentiment)) +
  geom_bar(width = 0.7, stat="identity") +
  theme_bw() +
  labs(title = "Word frequencies and sentiments in lyrics of Taylor Swift", subtitle = "Love and bad a
the most frequently used positive and negative words by Taylor Swift", x = "Frequencies", y = "Words",
aption = "Made by Li, Z.")
```




```
#9
ts_wordnew25 %>%
  with(wordcloud(word, n, max.words = 25))
```



Part 2E

Using the `ts_lyrics` dataset,

1. Tokenize each lyrical line by words.
2. Remove the “stopwords”.
3. Calculate the total number for each word in the lyrics **for each Album**.
4. Using the “afinn” sentiment lexicon, add a column to the summarized data frame adding the “afinn” sentiment lexicon.
5. Calculate the average sentiment score **for each Album**.
6. Auto print the wrangled tibble data frame.
7. Join the wrangled data frame from Part 1A (album sales in millions) with the wrangled data frame from #6 above (average sentiment score for each album).
8. Using `ggplot2`, create a scatter plot of the average sentiment score for each album (y-axis) and the album release data along the x-axis. Make the size of each point the album sales in millions.
9. Add a horizontal line at y-intercept=0.
10. Write 2-3 sentences interpreting the plot answering the question “How has the sentiment of Taylor Swift’s albums have changed over time?”. Add a title, subtitle, and useful axis labels.

```
#1,2
ts_words <- ts_lyrics %>%
  unnest_tokens(
    output = word,
    input = Lyrics,
    token = "words"
  ) %>%
  anti_join(stop_words)
```

```
## Joining with `by = join_by(word)`
```

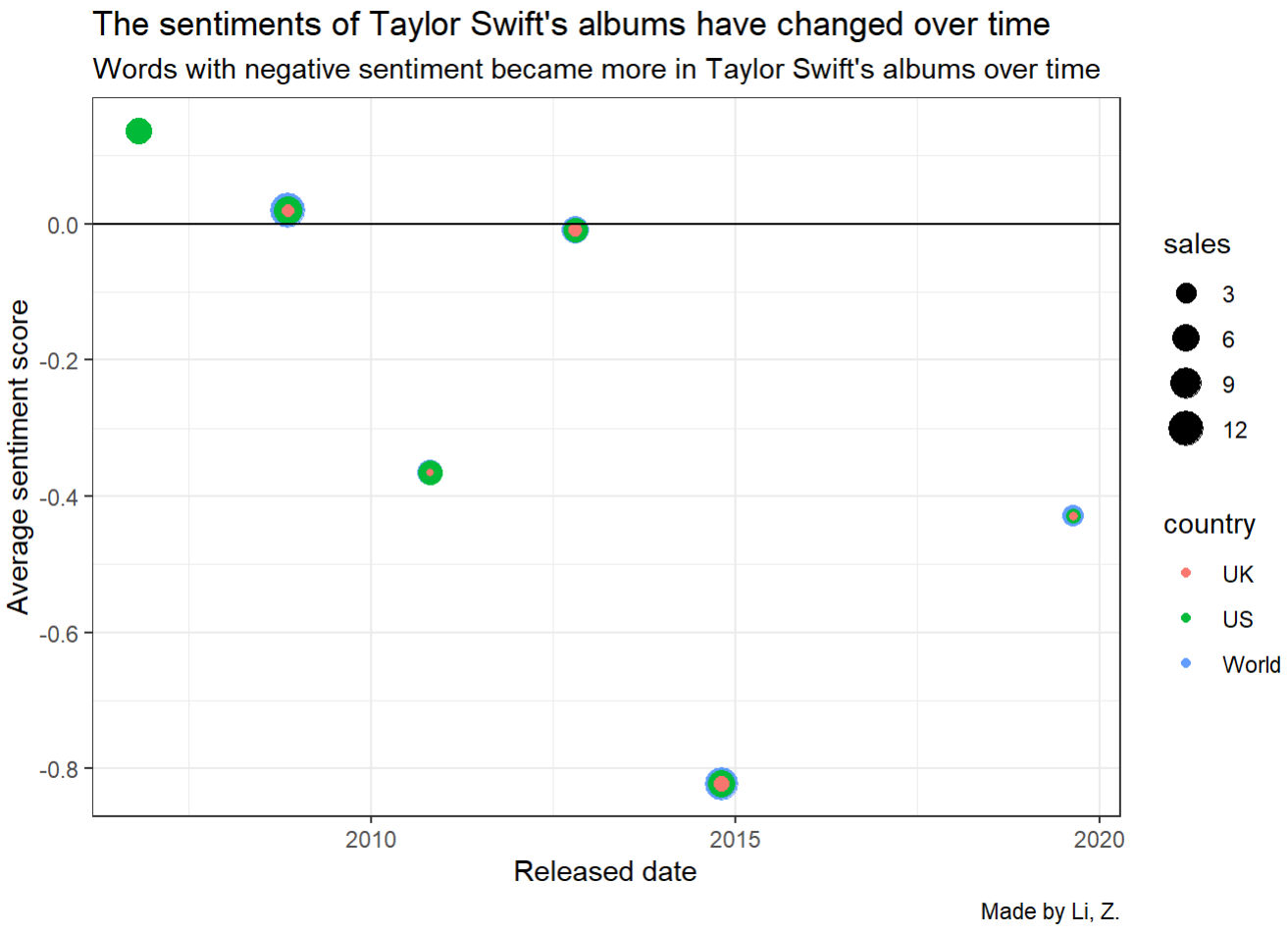
```
#3
ts_words_num <- ts_words %>%
  group_by(Album, word) %>%
  summarise(
    n_eachalbum = n()
  )
```

```
## `summarise()` has grouped output by 'Album'. You can override using the
## `.groups` argument.
```

```
#4
ts_words_num <- inner_join(ts_words_num, get_sentiments("afinn"), by = "word")
#5
ts_words_senti <- ts_words_num %>%
  group_by(Album) %>%
  summarise(
    mean_senti = mean(value, na.rm = T)
  )
#6
print(ts_words_senti)
```

```
## # A tibble: 8 × 2
##   Album        mean_senti
##   <chr>             <dbl>
## 1 1989           -0.822
## 2 Fearless        0.0192
## 3 Lover          -0.429
## 4 Red            -0.00870
## 5 Speak Now      -0.365
## 6 Taylor Swift    0.137
## 7 folklore       -0.613
## 8 reputation     -0.495
```

```
#7-9
colnames(ts_words_senti) <- c("title", "mean_senti")
ts_join <- inner_join(partla, ts_words_senti, by = "title")
ts_join %>%
  ggplot(aes(released, mean_senti)) +
  geom_point(aes(size = sales, color = country)) +
  geom_hline(aes(yintercept = 0)) +
  theme_bw() +
  labs(title = "The sentiments of Taylor Swift's albums have changed over time", subtitle = "Words with negative sentiment became more in Taylor Swift's albums over time", x = "Released date", y = "Average sentiment score", caption = "Made by Li, Z.")
```



The average sentiment scores of Taylor Swift's albums were greater than 0 before 2010, indicating more positive words than negative words. The average sentiment scores were always lower than 0 after 2010 with fluctuations. The album "1989", which was released in 2014, presented the lowest sentiment score of -0.82.

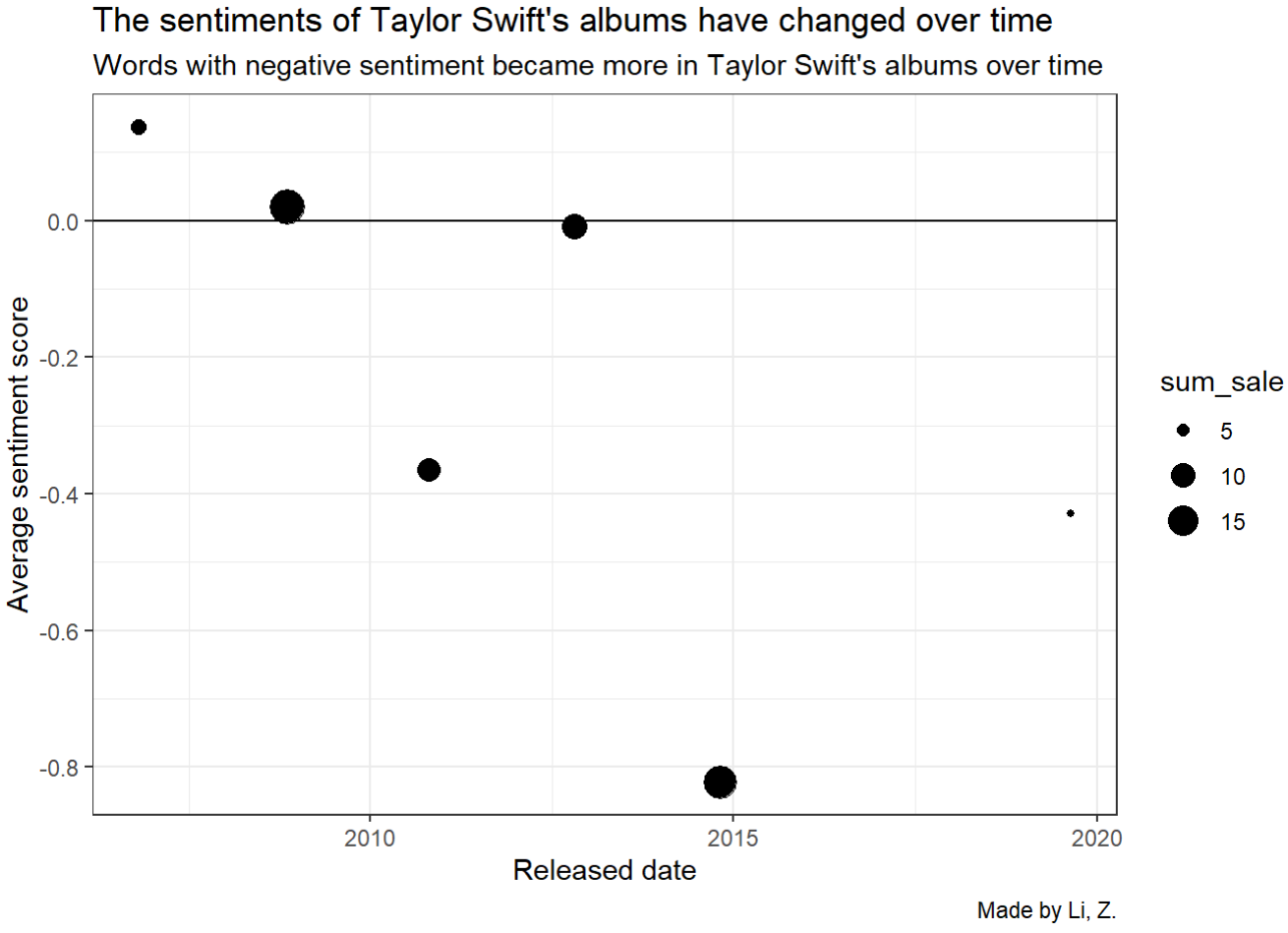
Part 2E: Another solution

The wrangled data frame from Part 1A has three countries for sales, so the scatter plot didn't combine them to a whole sale(namely, the plot use three points that overlapped with each other instead of a large point to represent the total sale). The step was not mentioned in the instructions. But I think it's more reasonable to combine them, so I do as follows:

```
partla_new <- partla %>%
  group_by(title, released) %>%
  summarise(
    sum_sale = sum(sales)
  )
```

```
## `summarise()` has grouped output by 'title'. You can override using the
## `.groups` argument.
```

```
ts_join_new <- inner_join(partla_new, ts_words_senti, by = "title")
ts_join_new %>%
  ggplot(aes(released, mean_senti)) +
  geom_point(aes(size = sum_sale)) +
  geom_hline(aes(yintercept = 0)) +
  theme_bw() +
  labs(title = "The sentiments of Taylor Swift's albums have changed over time", subtitle = "Words with
negative sentiment became more in Taylor Swift's albums over time", x = "Released date", y = "Average se
ntiment score", caption = "Made by Li, Z.")
```



R session information

```
options(width = 120)
sessioninfo::session_info()
```

```
## ─ Session info ─────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────
──────────────
##  setting  value
##  version  R version 4.3.0 (2023-04-21 ucrt)
##  os       Windows 10 x64 (build 19045)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  Chinese (Simplified)_China.utf8
##  ctype    Chinese (Simplified)_China.utf8
##  tz       America/New_York
##  date     2023-10-22
##  pandoc   3.1.1 @ D:/安装/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown)
##
## ─ Packages ─────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────
────────────────
##  package      * version date (UTC) lib source
##  bslib          0.5.0   2023-06-09 [1] CRAN (R 4.3.1)
##  cachem         1.0.8   2023-05-01 [1] CRAN (R 4.3.1)
##  cli            3.6.1   2023-03-23 [1] CRAN (R 4.3.0)
##  colorspace     2.1-0   2023-01-23 [1] CRAN (R 4.3.0)
##  digest         0.6.31  2022-12-11 [1] CRAN (R 4.3.1)
##  dplyr        * 1.1.2   2023-04-20 [1] CRAN (R 4.3.0)
##  evaluate       0.21    2023-05-05 [1] CRAN (R 4.3.1)
##  fansi          1.0.4   2023-01-22 [1] CRAN (R 4.3.0)
##  farver         2.1.1   2022-07-06 [1] CRAN (R 4.3.0)
##  fastmap        1.1.1   2023-02-24 [1] CRAN (R 4.3.1)
##  forcats      * 1.0.0   2023-01-29 [1] CRAN (R 4.3.1)
##  fs             1.6.2   2023-04-25 [1] CRAN (R 4.3.1)
##  generics       0.1.3   2022-07-05 [1] CRAN (R 4.3.0)
##  ggplot2      * 3.4.2   2023-04-03 [1] CRAN (R 4.3.0)
##  glue           1.6.2   2022-02-24 [1] CRAN (R 4.3.0)
##  gtable         0.3.3   2023-03-21 [1] CRAN (R 4.3.0)
##  here         * 1.0.1   2020-12-13 [1] CRAN (R 4.3.1)
##  highr          0.10    2022-12-22 [1] CRAN (R 4.3.1)
##  hms            1.1.3   2023-03-21 [1] CRAN (R 4.3.1)
##  htmltools      0.5.5   2023-03-23 [1] CRAN (R 4.3.1)
##  janeaustenr    1.0.0   2022-08-26 [1] CRAN (R 4.3.1)
##  jquerylib      0.1.4   2021-04-26 [1] CRAN (R 4.3.1)
##  jsonlite       1.8.5   2023-06-05 [1] CRAN (R 4.3.1)
##  knitr          1.43    2023-05-25 [1] CRAN (R 4.3.1)
##  labeling       0.4.2   2020-10-20 [1] CRAN (R 4.3.0)
##  lattice        0.21-8  2023-04-05 [2] CRAN (R 4.3.0)
##  lifecycle      1.0.3   2022-10-07 [1] CRAN (R 4.3.0)
##  lubridate    * 1.9.2   2023-02-10 [1] CRAN (R 4.3.1)
##  magrittr       2.0.3   2022-03-30 [1] CRAN (R 4.3.0)
##  Matrix         1.6-1   2023-08-14 [1] CRAN (R 4.3.1)
##  munsell        0.5.0   2018-06-12 [1] CRAN (R 4.3.0)
##  pillar         1.9.0   2023-03-22 [1] CRAN (R 4.3.0)
##  pkgconfig      2.0.3   2019-09-22 [1] CRAN (R 4.3.0)
##  purrr        * 1.0.1   2023-01-10 [1] CRAN (R 4.3.0)
##  R6             2.5.1   2021-08-19 [1] CRAN (R 4.3.0)
##  rappdirs       0.3.3   2021-01-31 [1] CRAN (R 4.3.1)
##  RColorBrewer * 1.1-3   2022-04-03 [1] CRAN (R 4.3.0)
##  Rcpp           1.0.10  2023-01-22 [1] CRAN (R 4.3.1)
```

```
##  readr         * 2.1.4   2023-02-10 [1] CRAN (R 4.3.1)
##  rlang           1.1.1   2023-04-28 [1] CRAN (R 4.3.0)
##  rmarkdown       2.22    2023-06-01 [1] CRAN (R 4.3.1)
##  rprojroot       2.0.3   2022-04-02 [1] CRAN (R 4.3.1)
##  rstudioapi      0.14    2022-08-22 [1] CRAN (R 4.3.1)
##  sass            0.4.6   2023-05-03 [1] CRAN (R 4.3.1)
##  scales          1.2.1   2022-08-20 [1] CRAN (R 4.3.0)
##  sessioninfo     1.2.2   2021-12-06 [1] CRAN (R 4.3.1)
##  SnowballC       0.7.1   2023-04-25 [1] CRAN (R 4.3.0)
##  stringi         1.7.12  2023-01-11 [1] CRAN (R 4.3.0)
##  stringr       * 1.5.0   2022-12-02 [1] CRAN (R 4.3.1)
##  textdata      * 0.4.4   2022-09-02 [1] CRAN (R 4.3.1)
##  tibble        * 3.2.1   2023-03-20 [1] CRAN (R 4.3.0)
##  tidyr         * 1.3.0   2023-01-24 [1] CRAN (R 4.3.0)
##  tidyselect      1.2.0   2022-10-10 [1] CRAN (R 4.3.0)
##  tidytext      * 0.4.1   2023-01-07 [1] CRAN (R 4.3.1)
##  tidyverse     * 2.0.0   2023-02-22 [1] CRAN (R 4.3.1)
##  timechange      0.2.0   2023-01-11 [1] CRAN (R 4.3.1)
##  tokenizers      0.3.0   2022-12-22 [1] CRAN (R 4.3.1)
##  tzdb            0.4.0   2023-05-12 [1] CRAN (R 4.3.1)
##  utf8            1.2.3   2023-01-31 [1] CRAN (R 4.3.0)
##  vctrs           0.6.2   2023-04-19 [1] CRAN (R 4.3.0)
##  withr           2.5.0   2022-03-03 [1] CRAN (R 4.3.0)
##  wordcloud     * 2.6     2018-08-24 [1] CRAN (R 4.3.1)
##  xfun            0.39    2023-04-20 [1] CRAN (R 4.3.1)
##  yaml            2.3.7   2023-01-23 [1] CRAN (R 4.3.0)
##
##  [1] C:/Users/13392/AppData/Local/R/win-library/4.3
##  [2] D:/安装/R-4.3.0/library
##
## ──────────────────────────────────────────────────────────────────────────────
──────────────────────────────────────────────────────────────────────────────
──────────────────────
```