STAT 413/613 HW 2: stringr and lubridate

Yunting Chiu

2020-09-29

Instructions

Rename the starter file under the analysis directory as `hw_01_yourname.Rmd` and use it for your solutions.
1. Modify the "author" field in the YAML header.
2. Stage and Commit R Markdown and HTML files (no PDF files).
3. **Push both .Rmd and HTML files to GitHub**.
- Make sure you have knitted to HTML prior to staging, committing, and pushing your final submission.
4. **Commit each time you answer a part of question, e.g. 1.1**
5. **Push to GitHub after each major question**, e.g., Scrabble and Civil War Battles
- **Committing and Pushing are graded elements for this homework.**
6. When complete, submit a response in Canvas

- Only include necessary code to answer the questions.
- Most of the functions you use should be from the tidyverse. Too much base R will result in point deductions.
- Use Pull requests and or email to ask me any questions. If you email, please ensure your most recent code is pushed to GitHub.
- Learning objectives:
    - Manipulate dates and times with lubridate.

Loading Libraries

```
library(readr)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.2     ✓ dplyr   1.0.2
## ✓ tibble  3.0.3     ✓ stringr 1.4.0
## ✓ tidyr   1.1.2     ✓ forcats 0.5.0
## ✓ purrr   0.3.4
```

```
## ── Conflicts ────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(tidyr)
library(ggthemes)
library(forcats)
```

1 Scrabble Words

For this exercise, we are using the Collins Scrabble Words (https://en.wikipedia.org/wiki/Collins_Scrabble_Words), which is most commonly used outside of the United States. The dictionary most often used in the United States is the Tournament Word List (https://en.wikipedia.org/wiki/Official_Tournament_and_Club_Word_List).

WARNING: Do not try `str_view()` or `str_view_all()` on these data.It will stall your computer.

1. Use a readr function to load the 2015 list of Collins Scrabble Words into R from your data folder or from https://data-science-master.github.io/lectures/data/words.txt (https://data-science-master.github.io/lectures/data/words.txt)
    - (note: “NA” is an official Scrabble word).

```
scrabble <- read_tsv(file = "../data/words.txt")
```

```
## Parsed with column specification:
## cols(
##   word = col_character()
## )
```

```
scrabble %>%
  # removed NA
  filter(!is.na(word)) -> scrabble_01
head(scrabble_01)
```

```
## # A tibble: 6 x 1
##   word
##   <chr>
## 1 AA
## 2 AAH
## 3 AAHED
## 4 AAHING
## 5 AAHS
## 6 AAL
```

2. What are the six longest words that have the most “X”’s in them?

```
scrabble_01 %>%
  # count the length of word, and count "X" in each word
  mutate(Xcount = str_count(word, "X"),
         count = str_length(word)) -> scrabble_02

scrabble_02 %>%
  arrange(desc(Xcount)) %>%
  filter(Xcount == max(scrabble_02$Xcount)) %>%
  arrange(desc(count)) -> six_longest

head(six_longest)
```

```
## # A tibble: 6 x 3
##   word          Xcount count
##   <chr>          <int> <int>
## 1 COEXECUTRIXES      2    13
## 2 EXTRATEXTUAL       2    12
## 3 COEXECUTRIX        2    11
## 4 EXECUTRIXES        2    11
## 5 SAXITOXINS         2    10
## 6 XANTHOXYLS         2    10
```

3. How many words have an identical first and second half of the word? If a word has an odd number of letters, exclude the middle character.

- MURMUR counts because MUR is both the first and second half.
- JIGAJIG counts because the middle A is excluded so JIG is both the first and second half.
- Save the results to a variable.

```
scrabble_02 %>%
  # distinguish whether the length of word is odd or even
  mutate(odd_even = if_else(str_length(word) %% 2 == 0, "even", "odd")) %>%

  # filter head half, if the length is odd, we remain the middle one of this word.
  mutate(head_h = if_else(odd_even == "even",
                          str_sub(word, 1, count/2), str_sub(word, 1, floor(count/2))))
 %>%

  # filter tail half
  mutate(tail_h = if_else(odd_even == "even",
                          str_sub(word, (count/2)+1, count), str_sub(word, ceiling(coun
t/2)+1, count))) %>%
  # Equal is TRUE, otherwise FALSE, and adding in df
  mutate(Halves = str_detect(head_h, tail_h)) -> scrabble_03

  # count number if Halves is TRUE
scrabble_03 %>%
  filter(Halves == TRUE) %>%
  nrow()
```

```
## [1] 254
```

- Quick-Solution

```
  # scrabble_02 %>% arrange(desc(count)) -> make sure the longest string is 15 units,
  # so the half of string is 7 units at most

  # () = group
  #.? = 0 or 1
  # period = any word
  # \\1 for the match in the first parentheses

scrabble_03 %>%
  filter(str_detect(word, "^(.|..|...|....|.....|......|.......).?\\1$")) -> Two_Halves

nrow(Two_Halves)
```

```
## [1] 254
```

4. Use the results from 3 to find the longest word with an identical first and second half of the word?

```
Two_Halves %>%
  arrange(desc(count)) %>%
  head(1)
```

```
## # A tibble: 1 x 7
##   word         Xcount count odd_even head_h tail_h Halves
##   <chr>         <int> <int> <chr>    <chr>  <chr>  <lgl>
## 1 CHIQUICHIQUI      0    12 even     CHIQUI CHIQUI TRUE
```

2 Civil War Battles

The data in “civil_war_theater.csv” contains a information on American Civil War battles, taken from Wikipedia (https://en.wikipedia.org/wiki/List_of_American_Civil_War_battles).

Variables include:

- `Battle` : The name of the battle.
- `Date` : The date(s) of the battle in different formats depending upon the length of the battle.
  - If it took place on one day, the format is “month day, year”.
  - If it took place over multiple days, the format is “month day_start-day_end, year”.
  - If it took place over multiple days and months, the format is “month_start day_start - month_end day_end, year”.
  - If it took place over multiple days,months, and years, the format is “month_start day_start, year_start - month_end day_end, year_end”.
- `State` : The state where the battle took place. Annotations (e.g. describing that the state was a territory at the time) are in parentheses.
- `CWSAC` : A rating of the military significance of the battle by the Civil War Sites Advisory Commission. `A` = Decisive, `B` = Major, `C` = Formative, `D` = Limited.
- `Outcome` : Usually `"Confederate victory"` , `"Union victory"` , or `"Inconclusive"` , followed by notes.
- `Theater` : An attempt to to identify which theater of war is most associated with the battle

1. Use a readr function and relative paths to load the data into R.

```
CivilWar <- read_csv(file = "../data/civil_war_theater.csv")
```

```
## Parsed with column specification:
## cols(
##   Battle = col_character(),
##   Date = col_character(),
##   State = col_character(),
##   CWSAC = col_character(),
##   Theater = col_character(),
##   Outcome = col_character()
## )
```

```
head(CivilWar)
```

```
## # A tibble: 6 x 6
##   Battle          Date      State      CWSAC Theater Outcome
##   <chr>           <chr>     <chr>      <chr> <chr>   <chr>
## 1 Battle of Fort… July 11-… District … B     Eastern Union victory: Failed Conf…
## 2 Battle of Hanc… January … Maryland   D     Eastern Inconclusive: Unsuccessful…
## 3 Battle of Sout… Septembe… Maryland   B     Eastern Union victory: McClellan d…
## 4 Battle of Anti… Septembe… Maryland   A     Eastern Tactically inconclusive; s…
## 5 Battle of Will… July 6-1… Maryland   C     Eastern Inconclusive: Meade and Le…
## 6 Battle of Boon… July 8, … Maryland   D     Eastern Inconclusive: Indecisive a…
```

The next several questions will help you take the dates from all the different formats and create a consistent set of start date and end date variables in the data frame. We will start by calculating how many years, and months are in each battle.

2. Add a variable to the data frame with the number of years for each battle.

- Create a character variable as follows. This can be used as a pattern in a regular expression.

```
year_regex <- stringr::str_c(1861:1865, collapse = "|")
year_regex
```

```
## [1] "1861|1862|1863|1864|1865"
```

- Use `year_regex` to now count the number of years in each battle, add this to the data frame, and save the data frame.

```
CivilWar %>%
  mutate(count_Year = str_count(Date, year_regex)) %>%
  select(Battle,count_Year, everything()) -> CivilWar_02

head(CivilWar_02)
```

```
## # A tibble: 6 x 7
##   Battle        count_Year Date     State    CWSAC Theater Outcome
##   <chr>              <int> <chr>    <chr>    <chr> <chr>   <chr>
## 1 Battle of Fo…          1 July 11… Distric… B     Eastern Union victory: Faile…
## 2 Battle of Ha…          1 January… Maryland D     Eastern Inconclusive: Unsucc…
## 3 Battle of So…          1 Septemb… Maryland B     Eastern Union victory: McCle…
## 4 Battle of An…          1 Septemb… Maryland A     Eastern Tactically inconclus…
## 5 Battle of Wi…          1 July 6-… Maryland C     Eastern Inconclusive: Meade …
## 6 Battle of Bo…          1 July 8,… Maryland D     Eastern Inconclusive: Indeci…
```

3. Add a variable to the data frame with the number of months for each battle.

- Consider R's built-in vector of month names: `month.name` .

```
month.name
```

```
##  [1] "January"   "February"  "March"     "April"     "May"       "June"
##  [7] "July"      "August"    "September" "October"   "November"  "December"
```

```
  # create each month name with regex first
month_name <- str_c(month.name, collapse = "|")
print(month_name)
```

```
## [1] "January|February|March|April|May|June|July|August|September|October|Novembe
r|December"
```

- Use `month.name` to count the number of month names in the `Date` variable in each battle.
- Add this to the data frame. (You might need to do something similar to what we did in part 2).

```
CivilWar_02 %>%
  mutate(count_Month = str_count(Date, month_name)) %>%
  select(Battle,count_Year,count_Month, everything()) -> CivilWar_03

head(CivilWar_03)
```

```
## # A tibble: 6 x 8
##   Battle     count_Year count_Month Date    State  CWSAC Theater Outcome
##   <chr>           <int>       <int> <chr>   <chr>  <chr> <chr>   <chr>
## 1 Battle of…          1           1 July 1… Distr… B     Eastern Union victory:…
## 2 Battle of…          1           1 Januar… Maryl… D     Eastern Inconclusive: …
## 3 Battle of…          1           1 Septem… Maryl… B     Eastern Union victory:…
## 4 Battle of…          1           1 Septem… Maryl… A     Eastern Tactically inc…
## 5 Battle of…          1           1 July 6… Maryl… C     Eastern Inconclusive: …
## 6 Battle of…          1           1 July 8… Maryl… D     Eastern Inconclusive: …
```

4. Add a variable to the data frame that is `TRUE` if `Date` spans multiple days and is `FALSE` otherwise. Spanning multiple months and/or years also counts as `TRUE` .

```
CivilWar_03 %>%
  # str_detect: Match a fixed string, and return TRUE or FALSE
  mutate(Multiple_days = str_detect(Date, "-")) %>%
  select(Battle:count_Month, Multiple_days, everything()) -> CivilWar_04

head(CivilWar_04)
```

```
## # A tibble: 6 x 9
##   Battle  count_Year count_Month Multiple_days Date  State CWSAC Theater Outcome
##   <chr>        <int>       <int> <lgl>         <chr> <chr> <chr> <chr>   <chr>
## 1 Battle…          1           1 TRUE          July… Dist… B     Eastern Union …
## 2 Battle…          1           1 TRUE          Janu… Mary… D     Eastern Inconc…
## 3 Battle…          1           1 FALSE         Sept… Mary… B     Eastern Union …
## 4 Battle…          1           1 FALSE         Sept… Mary… A     Eastern Tactic…
## 5 Battle…          1           1 TRUE          July… Mary… C     Eastern Inconc…
## 6 Battle…          1           1 FALSE         July… Mary… D     Eastern Inconc…
```

5. Make four new data frames by filtering the data based on the length of the battles:
    - a data frame with the data for only those battles spanning just one day,
    - a data frame with the data for only those battles spanning multiple days in just one month,
    - a data frame with the data for only those battles spanning multiple months but not multiple years, and,
    - a data frame with the data for only those battles spanning multiple years.

```
# df of battles spanning just one day
CivilWar_04 %>%
  filter(Multiple_days == FALSE) -> Battles_one_Day

head(Battles_one_Day)
```

```
## # A tibble: 6 x 9
##   Battle  count_Year count_Month Multiple_days Date  State CWSAC Theater Outcome
##   <chr>        <int>       <int> <lgl>         <chr> <chr> <chr> <chr>   <chr>
## 1 Battle…          1           1 FALSE         Sept… Mary… B     Eastern Union …
## 2 Battle…          1           1 FALSE         Sept… Mary… A     Eastern Tactic…
## 3 Battle…          1           1 FALSE         July… Mary… D     Eastern Inconc…
## 4 Battle…          1           1 FALSE         July… Mary… B     Eastern Confed…
## 5 Battle…          1           1 FALSE         Augu… Mary… D     Eastern Inconc…
## 6 Battle…          1           1 FALSE         June… Penn… C     Eastern Inconc…
```

```
# df of battles spanning multiple days in just one month
CivilWar_04 %>%
   filter(Multiple_days == TRUE & count_Month == 1) -> Battles_one_Month

head(Battles_one_Month)
```

```
## # A tibble: 6 x 9
##   Battle  count_Year count_Month Multiple_days Date  State CWSAC Theater Outcome
##   <chr>        <int>       <int> <lgl>         <chr> <chr> <chr> <chr>   <chr>
## 1 Battle…          1           1 TRUE          July… Dist… B     Eastern Union …
## 2 Battle…          1           1 TRUE          Janu… Mary… D     Eastern Inconc…
## 3 Battle…          1           1 TRUE          July… Mary… C     Eastern Inconc…
## 4 Battle…          1           1 TRUE          July… Penn… A     Eastern Union …
## 5 Battle…          1           1 TRUE          May … Virg… D     Eastern Inconc…
## 6 Battle…          1           1 TRUE          Marc… Virg… B     Eastern Inconc…
```

```
# df of battles spanning multiple months but not multiple years
CivilWar_04 %>%
  filter(Multiple_days == TRUE & count_Year == 1 & count_Month != 1) -> Battles_multiple
_Months

head(Battles_multiple_Months)
```

```
## # A tibble: 6 x 9
##   Battle  count_Year count_Month Multiple_days Date  State CWSAC Theater Outcome
##   <chr>        <int>       <int> <lgl>         <chr> <chr> <chr> <chr>   <chr>
## 1 Battle…          1           2 TRUE          May … Virg… D     Eastern Inconc…
## 2 Siege …          1           2 TRUE          Apri… Virg… B     Eastern Inconc…
## 3 Battle…          1           2 TRUE          May … Virg… B     Eastern Inconc…
## 4 Battle…          1           2 TRUE          Apri… Virg… C     Eastern Inconc…
## 5 Battle…          1           2 TRUE          Apri… Virg… C     Eastern Inconc…
## 6 Battle…          1           2 TRUE          Apri… Virg… A     Eastern Confed…
```

```
# df of battles spanning multiple years
CivilWar_04 %>%
  filter(count_Year != 1) -> Battles_multiple_Years

head(Battles_multiple_Years)
```

```
## # A tibble: 1 x 9
##   Battle  count_Year count_Month Multiple_days Date  State CWSAC Theater Outcome
##   <chr>        <int>       <int> <lgl>         <chr> <chr> <chr> <chr>   <chr>
## 1 Battle…          2           2 TRUE          Dece… Tenn… A     Western Union …
```

6. For each of the four new data frames,

- Add two new variables:
  - `Start` should contain the start-date of each battle.
  - `End` should contain the end-date of each battle.
  - Hint: look at help for `separate()`
  - Make sure these are `Date` class objects.
- Remove the `Date` variable from each data frame.
- Save the data frames with the new variables

```
Battles_one_Day %>%
  mutate(Start = mdy(Date), End = mdy(Date)) %>%
  select(-Date) -> Battles_one_Day_01

# cancel useless variables
head(Battles_one_Day_01)
```

```
## # A tibble: 6 x 10
##   Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##   <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
## 1 Battl…          1           1 FALSE         Mary… B     Eastern Union …
## 2 Battl…          1           1 FALSE         Mary… A     Eastern Tactic…
## 3 Battl…          1           1 FALSE         Mary… D     Eastern Inconc…
## 4 Battl…          1           1 FALSE         Mary… B     Eastern Confed…
## 5 Battl…          1           1 FALSE         Mary… D     Eastern Inconc…
## 6 Battl…          1           1 FALSE         Penn… C     Eastern Inconc…
## # … with 2 more variables: Start <date>, End <date>
```

```
Battles_one_Month %>%
  separate(Date, c("tmp_1", "tmp_2", "tmp_3"), sep = " ") %>%
  separate(tmp_2, c("startdate", "enddate"), sep = "-" ) %>%
# creating Start
  mutate(Starttmp = paste(tmp_1, startdate, tmp_3),
         Start = mdy(Starttmp)) %>%
# creating End
  mutate(Endtmp = paste(tmp_1, enddate, tmp_3),
         End = mdy(Endtmp)) %>%
# cancel useless variables
  select(-tmp_1, -startdate, -enddate, -tmp_3, -Starttmp, -Endtmp) -> Battles_one_Month_
01

head(Battles_one_Month_01)
```

```
## # A tibble: 6 x 10
##   Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##   <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
## 1 Battl…          1           1 TRUE          Dist… B     Eastern Union …
## 2 Battl…          1           1 TRUE          Mary… D     Eastern Inconc…
## 3 Battl…          1           1 TRUE          Mary… C     Eastern Inconc…
## 4 Battl…          1           1 TRUE          Penn… A     Eastern Union …
## 5 Battl…          1           1 TRUE          Virg… D     Eastern Inconc…
## 6 Battl…          1           1 TRUE          Virg… B     Eastern Inconc…
## # … with 2 more variables: Start <date>, End <date>
```

```
Battles_multiple_Months %>%
  separate(Date, c("tmp_1", "tmp_2"), sep = "-") %>%
  separate(tmp_2, c("tmp_3", "tmp_4"), sep = ",") %>%
# creating Start
  mutate(Starttmp = paste(tmp_1, tmp_4),
         Start = mdy(Starttmp)) %>%
# creating End
  mutate(Endtmp = paste(tmp_3, tmp_4),
         End = mdy(Endtmp)) %>%
# cancel useless variables
  select(-tmp_1, -tmp_3, -tmp_4, -Starttmp, -Endtmp) -> Battles_multiple_Months_01

head(Battles_multiple_Months_01)
```

```
## # A tibble: 6 x 10
##   Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##   <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
## 1 Battl…          1           2 TRUE          Virg… D     Eastern Inconc…
## 2 Siege…          1           2 TRUE          Virg… B     Eastern Inconc…
## 3 Battl…          1           2 TRUE          Virg… B     Eastern Inconc…
## 4 Battl…          1           2 TRUE          Virg… C     Eastern Inconc…
## 5 Battl…          1           2 TRUE          Virg… C     Eastern Inconc…
## 6 Battl…          1           2 TRUE          Virg… A     Eastern Confed…
## # … with 2 more variables: Start <date>, End <date>
```

```
Battles_multiple_Years %>%
  separate(Date, c("tmp_1", "tmp_2"), sep = "-") %>%
# creating Start and End
  mutate(Start = mdy(tmp_1),
         End = mdy(tmp_2)) %>%
# cancel useless variables
  select(-tmp_1, -tmp_2) -> Battles_multiple_Years_01

head(Battles_multiple_Years_01)
```

```
## # A tibble: 1 x 10
##   Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##   <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
## 1 Battl…          2           2 TRUE          Tenn… A     Western Union …
## # … with 2 more variables: Start <date>, End <date>
```

7. Create a new data frame with all the battles and the Start and End dates by binding the rows of the four data frames as updated in part 6

```
Battles_one_Day_01 %>%
  # use full_join to combine all df
  full_join(Battles_one_Month_01,
            by = c("Battle", "count_Year", "count_Month", "Multiple_days", "State", "CWS
AC", "Theater", "Outcome", "Start", "End")) %>%
  full_join(Battles_multiple_Months_01,
            by = c("Battle", "count_Year", "count_Month", "Multiple_days", "State", "CWS
AC", "Theater", "Outcome", "Start", "End")) %>%
  full_join(Battles_multiple_Years_01,
            by = c("Battle", "count_Year", "count_Month", "Multiple_days", "State", "CWS
AC", "Theater", "Outcome", "Start", "End")) -> CivilWar_05 # tided all of the battles da
tes.

head(CivilWar_05)
```

```
## # A tibble: 6 x 10
##   Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##   <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
## 1 Battl…          1           1 FALSE         Mary… B     Eastern Union …
## 2 Battl…          1           1 FALSE         Mary… A     Eastern Tactic…
## 3 Battl…          1           1 FALSE         Mary… D     Eastern Inconc…
## 4 Battl…          1           1 FALSE         Mary… B     Eastern Confed…
## 5 Battl…          1           1 FALSE         Mary… D     Eastern Inconc…
## 6 Battl…          1           1 FALSE         Penn… C     Eastern Inconc…
## # … with 2 more variables: Start <date>, End <date>
```

- use bind_rows
- If df all have the same columns (same names and types) it is better to use bind_rows. If you need to add columns then the join that fits the need, usually left join or inner join and rarely do you want a full join.

```
CivilWar_test_01 <- bind_rows(Battles_one_Day_01, Battles_one_Month_01,
                     Battles_multiple_Months_01, Battles_multiple_Years_01)

CivilWar_test_01
```

```
## # A tibble: 384 x 10
##    Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##    <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
##  1 Battl…          1           1 FALSE         Mary… B     Eastern "Union…
##  2 Battl…          1           1 FALSE         Mary… A     Eastern "Tacti…
##  3 Battl…          1           1 FALSE         Mary… D     Eastern "Incon…
##  4 Battl…          1           1 FALSE         Mary… B     Eastern "Confe…
##  5 Battl…          1           1 FALSE         Mary… D     Eastern "Incon…
##  6 Battl…          1           1 FALSE         Penn… C     Eastern "Incon…
##  7 Battl…          1           1 FALSE         Virg… C     Eastern "Confe…
##  8 Battl…          1           1 FALSE         Virg… C     Eastern "Confe…
##  9 First…          1           1 FALSE         Virg… A     Eastern "Confe…
## 10 Battl…          1           1 FALSE         Virg… B     Eastern "Confe…
## # … with 374 more rows, and 2 more variables: Start <date>, End <date>
```

8. Calculate the number of days each battle spanned.

- What's the longest battle of the war?
- How long did it last?

Siege of Port Hudson is the longest battle with 50 days.

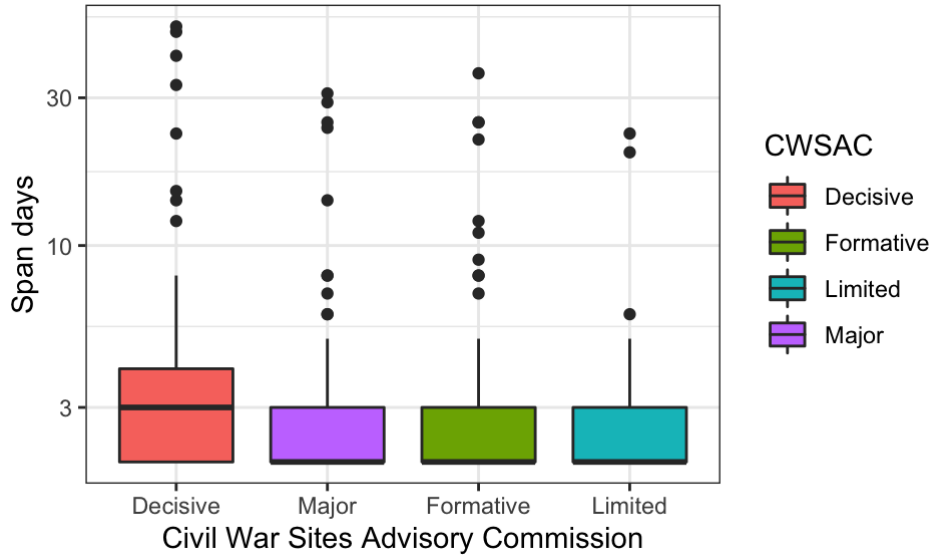
```
CivilWar_05 %>%
  mutate(Span = (End - Start)+1) %>%
  arrange(desc(Span)) -> CivilWar_06

head(CivilWar_06, 1)
```

```
## # A tibble: 1 x 11
##   Battle count_Year count_Month Multiple_days State CWSAC Theater Outcome
##   <chr>       <int>       <int> <lgl>         <chr> <chr> <chr>   <chr>
## 1 Siege…          1           2 TRUE          Loui… A     Western Union …
## # … with 3 more variables: Start <date>, End <date>, Span <drtn>
```

9. Is there an association between the CWSAC significance of a battle and its duration?

- Create an appropriate plot.
- Interpret the plot in one sentence to answer the question.
- Extra Credit: Test for a linear relationship using `lm()` and interpret the results in one sentence based on the p-value and adjusted R-squared.
- Interpretation: If this battle was a "Decisive boxplot", it's mean it mostly took the longest duration. By contrast, such as the "Limited boxplot", it has not taken longer times during this period, also the result of this kind of battle was limited.

```
CivilWar_06 %>%
 # recode in "CWSAC" variable
 # the variable type of Period should be numeric
  mutate(CWSAC = recode(CWSAC, "A" = "Decisive",
                               "B" = "Major",
                               "C" = "Formative",
                               "D" =  "Limited"),
 # the variable type of Period should be numeric type so that scale_y_log10() would be w
orked
         Span = as.numeric(Span)) -> CivilWar_07

# starting to plot
CivilWar_07 %>%
  mutate(CWSAC = as.factor(CWSAC)) %>%
 # log10 not define 0 (0 day), so we need plus 1 after "Span"

  ggplot(mapping = aes(x = fct_relevel(CWSAC,"Decisive","Major", # re-level the x-aris
                                       "Formative", "Limited" ),
                       y = Span+1, fill = CWSAC))+
  geom_boxplot()+
  theme_bw()+
  scale_y_log10()+
  xlab("Civil War Sites Advisory Commission")+
  ylab("Span days")
```



- Extra Credit: Test for a linear relationship using `lm()` and interpret the results in one sentence based on the p-value and adjusted R-squared.

- After we used Simple linear regression, we can know adjusted R-squared is 0.0622s and p-value is 0.00002026 (***p-value: 2.026e-05***).
- Interpretation: Because of the p-value of this data are very small which is less than the significance level(typically ≤ 0.05), which means the test hypothesis is false or should be rejected.

```
CivilWar07_SLR <- lm(Span ~ CWSAC, CivilWar_07)
summary(CivilWar07_SLR)
```

```
##
## Call:
## lm(formula = Span ~ CWSAC, data = CivilWar_07)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.778 -1.808 -0.808 -0.680 43.222
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      6.7778     0.8625   7.858 4.06e-14 ***
## CWSACFormative  -4.3920     1.0038  -4.375 1.57e-05 ***
## CWSACLimited    -4.9907     1.0266  -4.861 1.71e-06 ***
## CWSACMajor      -3.9701     1.0324  -3.845 0.000141 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.786 on 380 degrees of freedom
## Multiple R-squared:  0.0622, Adjusted R-squared:  0.0548
## F-statistic: 8.401 on 3 and 380 DF,  p-value: 2.026e-05
```

10. Extra Credit: Did the theaters of war (https://en.wikipedia.org/wiki/Theater_(warfare)) shift during the American Civil War?

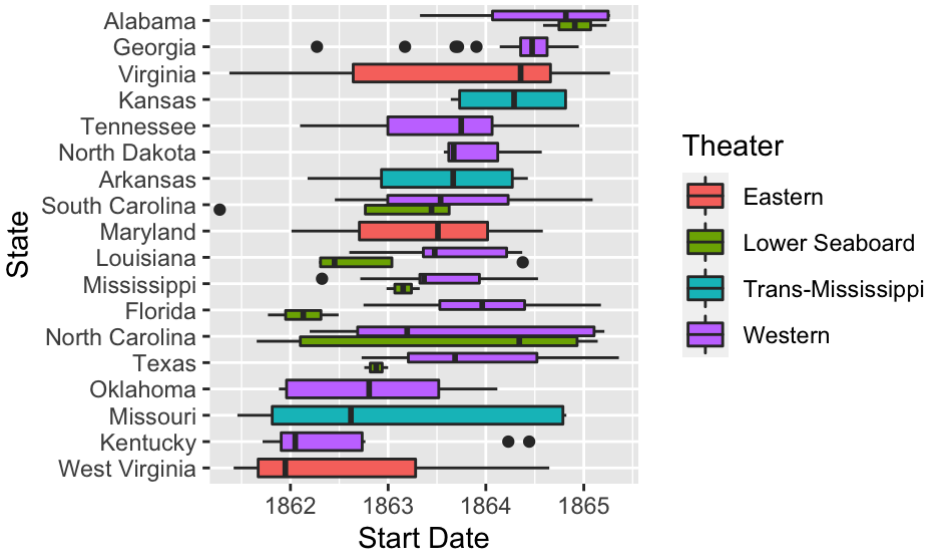
- Reproduce this plot in R:
- Interpret the plot in one sentence.
- Hints:
    - Filter out states with two or fewer battles.
    - Use regex to clean up the state names and then convert to factors
    - Use forcats to reorder the states by the start date.
    - Use `coord_flip()` to make horizontal boxplots.

```
CivilWar_07 %>%
  # count Battles of each State and add to new variable
  group_by(State) %>%
  mutate(num_Battles_States = n()) %>%
  # filter out states with two or fewer battles.
  filter(!num_Battles_States  <= 2)  -> CivilWar_08

CivilWar_08 %>%
  # changing each State's name with regex
  # reference: https://www.itread01.com/content/1548936390.html
  mutate(State = str_replace_all(State,
                                 "^North Dakota \\(Dakota Territory\\s\\sat the time\\)
$",
                                 "North Dakota"),
         State = str_replace_all(State,
                                 "^West Virginia \\(Virginia at the time\\)$",
                                 "West Virginia"),
         State = str_replace_all(State,
                                 "^Oklahoma \\(Indian Territory at the time\\)$",
                                 "Oklahoma")) %>%
  # make sure State has factor type that we can use "forcats" package later
  mutate(State = as.factor(State)) -> CivilWar_09

# check each State have been renamed successfully
unique(CivilWar_09$State)
```

```
##  [1] Louisiana      Mississippi    Missouri       North Carolina Virginia
##  [6] Georgia        South Carolina Alabama        Tennessee      Maryland
## [11] West Virginia  Arkansas       Kentucky       Florida        North Dakota
## [16] Oklahoma       Texas          Kansas
## 18 Levels: Alabama Arkansas Florida Georgia Kansas Kentucky ... West Virginia
```

- Starting to plot

```
CivilWar_09 %>%
  # use fct_reorder to reorder the states by the start date.
  ggplot(mapping = aes(x = fct_reorder(State, Start), y = Start, fill = Theater))+
  # use `coord_flip()` to make horizontal boxplots.
  coord_flip()+
  geom_boxplot()+
  labs(x = "State", y = "Start Date")
```



Note for extra credit 2.2.10EC: - 0.25 No interpretation. Consider str_extract(State,"((\w+\s\w+))|(\w+)") and parse_factor()