DATA-413/613 HW 2: Tidyverse Review

Michael Lewis

2022-09-13

Instructions

Rename the starter.Rmd file under the analysis directory as `hw02_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., Enable and Civil War Battles. 6. When complete, submit a response in Canvas

Analyze the Enable Word List

The ENABLE word list is used in many online or app-based games such as Words with Friends. It is an acronym for Enhanced North American Benchmark Lexicon. Unlike many Scrabble word lists, it is unconstrained by word length but also has fewer words. It was developed in 1997 so does not have many "modern" words, e.g., blog or cellphone.

1. Use a {readr} function and relative path to load the enable1_words.txt into R from your data folder using arguments so there are no warnings or messages. There should be 172,820 rows. Do not suppress warnings and messages.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- 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
```

```
enable_list <- read_csv("../data/enable1_words.txt",
    col_names = FALSE)
```

```
## Rows: 172820 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): X1
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#editing col name
enable_list %>%
  mutate(words = X1) %>%
  select(-X1) -> enable_list

nrow(enable_list)
```

```
## [1] 172820
```

2. What word(s) have the most "m"'s in them? There should be 7. Of the *words with the next-greatest number of "m"'s*, use a slice function to find the six longest words from longest to shortest? Why do you get 11 words and not 6?

```
#Words with the most "m"'s
enable_list %>%
  mutate(m_count = str_count(words, "m")) %>%
  arrange(desc(m_count)) %>%
  head(n=7)
```

```
## # A tibble: 7 x 2
##   words             m_count
##   <chr>               <int>
## 1 immunocompromised       4
## 2 mammogram               4
## 3 mammograms              4
## 4 mammonism               4
## 5 mammonisms              4
## 6 mesembryanthemum        4
## 7 mesembryanthemums       4
```

```
  #Six longest w/ next greatest number of "m"'s
enable_list %>%
  mutate(m_count = str_count(words, "m")) %>%
  mutate(nletters = str_length(words)) %>%
```

```
  filter(m_count == 3) %>%
  select(words, nletters) %>%
  slice_max(nletters, n=6)
```

```
## # A tibble: 11 x 2
##    words                  nletters
##    <chr>                     <int>
##  1 immunohistochemistries       22
##  2 immunocytochemistries        21
##  3 immunocytochemically         20
##  4 immunohistochemistry         20
##  5 agammaglobulinemias          19
##  6 hemidemisemiquavers          19
##  7 immunocytochemistry          19
##  8 immunohematological          19
##  9 immunohematologists          19
## 10 immunohistochemical          19
## 11 parasympathomimetic          19
```

You get 11 results because there are multiple ties and the slice_max function (as written here) does not specify what should be done with ties. This can be resolved by using "with_ties" within the slice_max function.

3. How many words have an identical first and second half of the word? DATA 613-students must solve using a regex pattern.

    - If a word has an odd number of letters, exclude the middle character.

```
enable_list %>%
mutate(same_halves = str_starts(words, str_sub(words, ceiling((str_length(words)/2)+1))))
```

```
## # A tibble: 172,820 x 2
##    words    same_halves
##    <chr>    <lgl>
##  1 aa       TRUE
##  2 aah      FALSE
##  3 aahed    FALSE
##  4 aahing   FALSE
##  5 aahs     FALSE
##  6 aal      FALSE
##  7 aalii    FALSE
##  8 aaliis   FALSE
##  9 aals     FALSE
## 10 aardvark FALSE
## # ... with 172,810 more rows
```

```
#Testing on below words
#murmur
enable_list %>%
mutate(same_halves = str_starts(words, str_sub(words, ceiling((str_length(words)/2)+1)))) %>%
filter(words == "murmur")
```

```
## # A tibble: 1 x 2
##   words  same_halves
##   <chr>  <lgl>
## 1 murmur TRUE
```

```
#derider
enable_list %>%
mutate(same_halves = str_starts(words, str_sub(words, ceiling((str_length(words)/2)+1)))) %>%
filter(words == "derider")
```

```
## # A tibble: 1 x 2
##   words   same_halves
##   <chr>   <lgl>
## 1 derider TRUE
```

```
#saving
enable_list %>%
mutate(same_halves = str_starts(words, str_sub(words, ceiling((str_length(words)/2)+1)))) -> enable_lis
```

- "murmur" counts because "mur" is both the first and second half.
- "derider" counts because the middle "i" is excluded so "der" is both the first and second half.
- Save the results to a variable in a data frame that includes the original variables.

4. Use the results from 3 to find the longest word(s) with an identical first and second half of the word? There should be four words.

```
enable_list %>%
  mutate(word_length = str_length(words)) %>%
  filter(same_halves == TRUE) %>%
  arrange(desc(word_length))
```

```
## # A tibble: 134 x 3
##    words     same_halves word_length
##    <chr>     <lgl>             <int>
##  1 einsteins TRUE                  9
##  2 muckamuck TRUE                  9
##  3 okeydokey TRUE                  9
##  4 outshouts TRUE                  9
##  5 beriberi  TRUE                  8
##  6 caracara  TRUE                  8
##  7 chowchow  TRUE                  8
##  8 couscous  TRUE                  8
##  9 froufrou  TRUE                  8
## 10 greegree  TRUE                  8
## # ... with 124 more rows
```

```
#Four longest words with identical first and second half
```

Country Names

The goal is to create an updated country code data frame with the original and world bank names where they exist along with a set of new names without punctuation.

1. Load the data `country_codes` from the {gapminder} package and use a {readr} function and relative path to read in the World Bank data in `country.csv`. These two data sets are not consistent on all of the country names.

```
library(gapminder)
data("country_codes")
read_csv("../data/country.csv") -> wbank
```

```
## Rows: 263 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): Country Code, Region, IncomeGroup, TableName
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#cleaning col names
wbank %>%
mutate(Region = as.factor(Region), IncomeGroup = as.factor(IncomeGroup), CountryCode = `Country Code`) %
select(-`Country Code`) -> wbank
```

2. Use a {dplyr} join function to show only the country **names** from the gapminder country codes that are **not in** the World Bank data. There should be 21.

```
anti_join(country_codes, wbank, by = c("country"="TableName"))
```

```
## # A tibble: 21 x 3
##    country          iso_alpha iso_num
##    <chr>            <chr>       <int>
##  1 Bahamas          BHS            44
##  2 Brunei           BRN            96
##  3 Cape Verde       CPV           132
##  4 Cote d'Ivoire    CIV           384
##  5 Egypt            EGY           818
##  6 French Guiana    GUF           254
##  7 Gambia           GMB           270
##  8 Guadeloupe       GLP           312
##  9 Hong Kong, China HKG           344
## 10 Iran             IRN           364
## # ... with 11 more rows
```

3. Use a {dplyr} join function to add the country names from the World Bank data to the `country_codes` data frame to a new variable called `wb_name` **for only those countries that are in the {gapminder} `country_codes` data frame** and save to a data frame called `country_codes_wb`.

```
left_join(country_codes, wbank, by=c("iso_alpha"="CountryCode")) -> country_codes_wb
country_codes_wb
```

```
## # A tibble: 187 x 6
##    country     iso_alpha iso_num Region                    IncomeGroup  Table~1
##    <chr>       <chr>       <int> <fct>                     <fct>        <chr>
```

```
##  1 Afghanistan AFG             4 South Asia                  Low income   Afghan~
##  2 Albania     ALB             8 Europe & Central Asia       Upper middl~ Albania
##  3 Algeria     DZA            12 Middle East & North Africa  Upper middl~ Algeria
##  4 Angola      AGO            24 Sub-Saharan Africa          Lower middl~ Angola
##  5 Argentina   ARG            32 Latin America & Caribbean   High income  Argent~
##  6 Armenia     ARM            51 Europe & Central Asia       Upper middl~ Armenia
##  7 Aruba       ABW           533 Latin America & Caribbean   High income  Aruba
##  8 Australia   AUS            36 East Asia & Pacific         High income  Austra~
##  9 Austria     AUT            40 Europe & Central Asia       High income  Austria
## 10 Azerbaijan  AZE            31 Europe & Central Asia       Upper middl~ Azerba~
## # ... with 177 more rows, and abbreviated variable name 1: TableName
```

```
country_codes_wb %>%
  mutate(wb_name = TableName) %>%
  select(-TableName) -> country_codes_wb
```

4. Use code count how many world bank names use some form of punctuation. There should be 16.

- Note: the accent circumflex "^" does not count as punctuation but as part of a letter.

```
country_codes_wb %>%
  mutate(punctuation = str_detect(wb_name,"[:punct:]", negate = FALSE)) %>%
  select(wb_name, punctuation) %>%
  filter(punctuation == "TRUE")
```

```
## # A tibble: 16 x 2
##    wb_name               punctuation
##    <chr>                 <lgl>
##  1 Bahamas, The          TRUE
##  2 Congo, Dem. Rep.      TRUE
##  3 Congo, Rep.           TRUE
##  4 Côte d'Ivoire         TRUE
##  5 Egypt, Arab Rep.      TRUE
##  6 Gambia, The           TRUE
##  7 Guinea-Bissau         TRUE
##  8 Hong Kong SAR, China  TRUE
##  9 Iran, Islamic Rep.    TRUE
## 10 Korea, Rep.           TRUE
## 11 Korea, Rep.           TRUE
## 12 Macao SAR, China      TRUE
## 13 Micronesia, Fed. Sts. TRUE
## 14 Timor-Leste           TRUE
## 15 Venezuela, RB         TRUE
## 16 Yemen, Rep.           TRUE
```

5. Create a new column in the data frame **right after `country`** where you use {stringr} functions to:

- a. Replace all of the punctuation or white spaces in the world bank names with an _, and then,
- b. Remove any trailing _, and then,
- c. Replace any double __ with a single _.

- d. Now filter to show only the 16 rows with the new names. One of them should look like `Congo_Dem_Rep`.

```
country_codes_wb %>%
  mutate(punctuation = str_detect(wb_name,"[:punct:]", negate = FALSE)) %>%
  mutate(name_clean = str_replace_all(wb_name,"[:punct:]|[\\s]", "_")) %>%
  #A- replace all punctuation with '_'
  mutate(name_clean = str_remove(name_clean,"_$")) %>%
  #B- remove trailing '_'
  mutate(name_clean = str_replace_all(name_clean, "__", "_")) %>%
  #C- replace double '__' with '_'
  filter(punctuation == TRUE) %>%
  select(wb_name, name_clean, everything())
```

```
## # A tibble: 16 x 8
##    wb_name               name_c~1 country iso_a~2 iso_num Region Incom~3 punct~4
##    <chr>                 <chr>    <chr>   <chr>     <int> <fct>  <fct>   <lgl>
##  1 Bahamas, The          Bahamas~ Bahamas BHS          44 Latin~ High i~ TRUE
##  2 Congo, Dem. Rep.      Congo_D~ Congo,~ COD         180 Sub-S~ Low in~ TRUE
##  3 Congo, Rep.           Congo_R~ Congo,~ COG         178 Sub-S~ Lower ~ TRUE
##  4 Côte d'Ivoire         Côte_d_~ Cote d~ CIV         384 Sub-S~ Lower ~ TRUE
##  5 Egypt, Arab Rep.      Egypt_A~ Egypt   EGY         818 Middl~ Lower ~ TRUE
##  6 Gambia, The           Gambia_~ Gambia  GMB         270 Sub-S~ Low in~ TRUE
##  7 Guinea-Bissau         Guinea_~ Guinea~ GNB         624 Sub-S~ Low in~ TRUE
##  8 Hong Kong SAR, China  Hong_Ko~ Hong K~ HKG         344 East ~ High i~ TRUE
##  9 Iran, Islamic Rep.    Iran_Is~ Iran    IRN         364 Middl~ Upper ~ TRUE
## 10 Korea, Rep.           Korea_R~ Korea,~ KOR         410 East ~ High i~ TRUE
## 11 Korea, Rep.           Korea_R~ Korea,~ KOR         410 East ~ High i~ TRUE
## 12 Macao SAR, China      Macao_S~ Macao,~ MAC         446 East ~ High i~ TRUE
## 13 Micronesia, Fed. Sts. Microne~ Micron~ FSM         583 East ~ Lower ~ TRUE
## 14 Timor-Leste           Timor_L~ Timor-~ TLS         626 East ~ Lower ~ TRUE
## 15 Venezuela, RB         Venezue~ Venezu~ VEN         862 Latin~ Upper ~ TRUE
## 16 Yemen, Rep.           Yemen_R~ Yemen,~ YEM         887 Middl~ Low in~ TRUE
## # ... with abbreviated variable names 1: name_clean, 2: iso_alpha,
## #   3: IncomeGroup, 4: punctuation
```

- There is no need to do the replacements in a single step - three steps is fine.

Civil War Battles

The file "civil_war_theater.csv" contains data on American Civil War battles, taken from Wikipedia.

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 path to load the data into R while using an argument of the {readr} function to specify the column types to be character. Visually inspect the data.

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

```
## Rows: 384 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): Battle, Date, State, CWSAC, Theater, Outcome
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
civil_war_theater %>%
  mutate(State = parse_character(State),
         Theater = parse_character(Theater),
         CWSAC = parse_character(CWSAC))
```

```
## # A tibble: 384 x 6
##    Battle                                      Date  State CWSAC Theater Outcome
##    <chr>                                       <chr> <chr> <chr> <chr>   <chr>
##  1 Battle of Fort Stevens                      July~ Dist~ B     Eastern Union ~
##  2 Battle of Hancock                           Janu~ Mary~ D     Eastern Inconc~
##  3 Battle of South Mountainor Boonsboro        Sept~ Mary~ B     Eastern Union ~
##  4 Battle of Antietam or Sharpsburg            Sept~ Mary~ A     Eastern Tactic~
##  5 Battle of Williamsport                      July~ Mary~ C     Eastern Inconc~
##  6 Battle of Boonsboro                         July~ Mary~ D     Eastern Inconc~
##  7 Battle of Monocacy (Battle of Monocacy Jun~ July~ Mary~ B     Eastern Confed~
##  8 Battle of Folck's Mill                      Augu~ Mary~ D     Eastern Inconc~
##  9 Battle of Hanover                           June~ Penn~ C     Eastern Inconc~
## 10 Battle of Gettysburg                        July~ Penn~ A     Eastern Union ~
## # ... with 374 more rows
```

The next several questions will help you take the dates from all the different formats and add variables for start date and end date with a consistent format.

Suggest documenting in the text the steps of your plan to solve each problem so your approach and rationale are clear. Then implement your plan in code.

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.

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

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

```
civil_war_theater %>%
  mutate(year_count = str_count(Date, year_regex)) %>%
  select(Battle, Date, year_count,everything()) -> civil_war_theater

civil_war_theater
```

```
## # A tibble: 384 x 7
##    Battle                              Date  year_~1 State CWSAC Theater Outcome
##    <chr>                               <chr>   <int> <chr> <chr> <chr>   <chr>
##  1 Battle of Fort Stevens              July~       1 Dist~ B     Eastern Union ~
##  2 Battle of Hancock                   Janu~       1 Mary~ D     Eastern Inconc~
##  3 Battle of South Mountainor Boonsbo~ Sept~       1 Mary~ B     Eastern Union ~
##  4 Battle of Antietam or Sharpsburg    Sept~       1 Mary~ A     Eastern Tactic~
##  5 Battle of Williamsport              July~       1 Mary~ C     Eastern Inconc~
##  6 Battle of Boonsboro                 July~       1 Mary~ D     Eastern Inconc~
##  7 Battle of Monocacy (Battle of Mono~ July~       1 Mary~ B     Eastern Confed~
##  8 Battle of Folck's Mill              Augu~       1 Mary~ D     Eastern Inconc~
##  9 Battle of Hanover                   June~       1 Penn~ C     Eastern Inconc~
## 10 Battle of Gettysburg                July~       1 Penn~ A     Eastern Union ~
## # ... with 374 more rows, and abbreviated variable name 1: year_count
```

- Hint: 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 directly after `Date`, and save the data frame.

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

```
month_regex <- stringr::str_c(month.name, collapse = "|")
month_regex
```

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

```
civil_war_theater %>%
  mutate(month_count = str_count(Date, month_regex)) %>%
  select(Battle, Date, month_count, year_count,everything()) -> civil_war_theater
```

- 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"
  ```

- Use `month.name` to count the number of month names in the `Date` variable in each battle.
- Add this to the data frame directly after `Date` and save it. (Do something similar to part 2).

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

```
civil_war_theater %>%
  mutate(day_count = str_detect(Date, "-", negate = FALSE)) %>%
  select(Battle, Date, day_count, month_count, year_count,everything()) -> civil_war_theater
```

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,

```
civil_war_theater %>%
  filter(day_count == FALSE) -> df_one_day
```

    - a data frame with the data for only those battles spanning multiple days in just one month,

```
civil_war_theater %>%
  filter(day_count == TRUE & month_count == 1) -> df_multiple_days_one_month
```

    - a data frame with the data for only those battles spanning multiple months but not multiple years, and,

```
civil_war_theater %>%
  filter(month_count != 1 & year_count == 1) -> df_mult_months_one_year
```

- a data frame with the data for only those battles spanning multiple years.

```
civil_war_theater %>%
  filter(year_count != 1) -> df_multiple_years
```

- How many rows are in each data frame?

```
nrow(df_one_day)
```

```
## [1] 255
```

```
nrow(df_multiple_days_one_month)
```

```
## [1] 103
```

```
nrow(df_mult_months_one_year)
```

```
## [1] 25
```

```
nrow(df_multiple_years)
```

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

- Check your results for completeness or duplication/missing by using code to show (`TRUE`or `FALSE`) if the total of the rows in the four data frames equals the total number of rows in the original data frame. If the result is `FALSE`, suggest checking your work,

```
sum(nrow(df_one_day),
nrow(df_multiple_days_one_month),
nrow(df_mult_months_one_year),
nrow(df_multiple_years)) == nrow(civil_war_theater)
```

```
## [1] TRUE
```

6. Manipulate each of the four data individually as follows: by adding two new variables to the data frame. How you add the new variables will be different for each of the four data frames.

- Add two new variables to the data frame.
  - The new variable `Start` should contain the first date of each battle.
  - The new variable `End` should contain the last date of each battle.
  - `Start` and `End` **must be `Date` class objects.**
    - Hint: look at help for `separate()` and use {lubridate} functions.
- Remove the `Date` variable from each data frame.
- Save the data frame.

```
library(lubridate)
df_one_day %>%
  mutate(Start = mdy(Date),
         End = mdy(Date)) %>%
select(-Date) %>%
select(Battle, Start, End, everything()) -> df_one_day

df_one_day
```

```
## # A tibble: 255 x 10
##    Battle      Start      End        day_c~1 month~2 year_~3 State CWSAC Theater
##    <chr>       <date>     <date>     <lgl>     <int>   <int> <chr> <chr> <chr>
##  1 Battle of ~ 1862-09-14 1862-09-14 FALSE         1       1 Mary~ B     Eastern
##  2 Battle of ~ 1862-09-17 1862-09-17 FALSE         1       1 Mary~ A     Eastern
##  3 Battle of ~ 1863-07-08 1863-07-08 FALSE         1       1 Mary~ D     Eastern
##  4 Battle of ~ 1864-07-09 1864-07-09 FALSE         1       1 Mary~ B     Eastern
##  5 Battle of ~ 1864-08-01 1864-08-01 FALSE         1       1 Mary~ D     Eastern
##  6 Battle of ~ 1863-06-30 1863-06-30 FALSE         1       1 Penn~ C     Eastern
##  7 Battle of ~ 1861-06-10 1861-06-10 FALSE         1       1 Virg~ C     Eastern
##  8 Battle of ~ 1861-07-18 1861-07-18 FALSE         1       1 Virg~ C     Eastern
##  9 First Batt~ 1861-07-21 1861-07-21 FALSE         1       1 Virg~ A     Eastern
## 10 Battle of ~ 1861-10-21 1861-10-21 FALSE         1       1 Virg~ B     Eastern
## # ... with 245 more rows, 1 more variable: Outcome <chr>, and abbreviated
## #   variable names 1: day_count, 2: month_count, 3: year_count
```

```
df_multiple_days_one_month %>%
  separate(col = Date, into = c("Month", "Start", "End", "Year")) %>%
  mutate(Start = str_c(Month, Start, Year, sep = ","),
         Start = mdy(Start),

         End = str_c(Month, End, Year, sep = ","),
         End = mdy(End)) %>%
select(-Month, -Year) -> df_multiple_days_one_month

df_multiple_days_one_month
```

```
## # A tibble: 103 x 10
##    Battle      Start      End        day_c~1 month~2 year_~3 State CWSAC Theater
##    <chr>       <date>     <date>     <lgl>     <int>   <int> <chr> <chr> <chr>
##  1 Battle of ~ 1864-07-11 1864-07-12 TRUE          1       1 Dist~ B     Eastern
##  2 Battle of ~ 1862-01-05 1862-01-06 TRUE          1       1 Mary~ D     Eastern
##  3 Battle of ~ 1863-07-06 1863-07-16 TRUE          1       1 Mary~ C     Eastern
##  4 Battle of ~ 1863-07-01 1863-07-03 TRUE          1       1 Penn~ A     Eastern
##  5 Battle of ~ 1861-05-18 1861-05-19 TRUE          1       1 Virg~ D     Eastern
##  6 Battle of ~ 1862-03-08 1862-03-09 TRUE          1       1 Virg~ B     Eastern
##  7 Battle of ~ 1862-06-27 1862-06-28 TRUE          1       1 Virg~ D     Eastern
##  8 First Batt~ 1862-08-22 1862-08-25 TRUE          1       1 Virg~ D     Eastern
##  9 Battle of ~ 1862-08-25 1862-08-27 TRUE          1       1 Virg~ B     Eastern
## 10 Second Bat~ 1862-08-28 1862-08-30 TRUE          1       1 Virg~ A     Eastern
## # ... with 93 more rows, 1 more variable: Outcome <chr>, and abbreviated
## #   variable names 1: day_count, 2: month_count, 3: year_count
```

```
df_mult_months_one_year %>%
  separate(col = Date, into = c("Start", "End"), sep = "-") %>%
  separate(col = End, into = c("End", "Year"), sep = ",") %>%
  mutate(Start = str_c(Start, Year),
         Start = mdy(Start),

         End = str_c(End, Year),
         End = mdy(End)
         ) %>%
  select(-Year) -> df_mult_months_one_year

df_mult_months_one_year
```

```
## # A tibble: 25 x 10
##    Battle      Start      End        day_c~1 month~2 year_~3 State CWSAC Theater
##    <chr>       <date>     <date>     <lgl>     <int>   <int> <chr> <chr> <chr>
##  1 Battle of ~ 1861-05-29 1861-06-01 TRUE          2       1 Virg~ D     Eastern
##  2 Siege of Y~ 1862-04-05 1862-05-04 TRUE          2       1 Virg~ B     Eastern
##  3 Battle of ~ 1862-05-31 1862-06-01 TRUE          2       1 Virg~ B     Eastern
##  4 Battle of ~ 1863-04-11 1863-05-04 TRUE          2       1 Virg~ C     Eastern
##  5 Battle of ~ 1863-04-11 1863-05-04 TRUE          2       1 Virg~ C     Eastern
##  6 Battle of ~ 1863-04-30 1863-05-06 TRUE          2       1 Virg~ A     Eastern
##  7 Battle of ~ 1863-11-27 1863-12-02 TRUE          2       1 Virg~ B     Eastern
##  8 Battle of ~ 1864-05-31 1864-06-12 TRUE          2       1 Virg~ A     Eastern
##  9 Battle of ~ 1864-09-30 1864-10-02 TRUE          2       1 Virg~ B     Eastern
```

```
## 10 Battle of ~ 1865-03-27 1865-04-08 TRUE          2       1 Alab~ B     Lower ~
## # ... with 15 more rows, 1 more variable: Outcome <chr>, and abbreviated
## #   variable names 1: day_count, 2: month_count, 3: year_count
```

```
df_multiple_years %>%
  separate(col = Date, into = c("Start", "End"), sep = "-") %>%
  mutate(Start = mdy(Start),
         End = mdy(End)) -> df_multiple_years

df_multiple_years
```

```
## # A tibble: 1 x 10
##   Battle       Start      End        day_c~1 month~2 year_~3 State CWSAC Theater
##   <chr>        <date>     <date>     <lgl>     <int>   <int> <chr> <chr> <chr>
## 1 Battle of S~ 1862-12-31 1863-01-02 TRUE          2       2 Tenn~ A     Western
## # ... with 1 more variable: Outcome <chr>, and abbreviated variable names
## #   1: day_count, 2: month_count, 3: year_count
```

You may use the following snippets of code for one or more of the data frames but you have to match to the correct data frame and put **in the correct order**.

- These snippets are not enough for any single data frame.
  separate(col = Date, into = c("Start", "End"), sep = "-")
  mutate(Start = mdy(Date), End = mdy(Date))
  separate(col = Date, into = c("Month", "Start", "End", "Year"))
  mutate(End = mdy(End), year = year(End), Start = str_c(Start, year, sep = ","),Start = mdy(Start))
  select(-Month, -Year)
- You should have four updated data frames when complete.

7. Use a single call to a {dplyr} function to bind the rows of the four updated data frames into a single new data frame with all the battles.

```
bind_rows(df_mult_months_one_year, df_multiple_days_one_month, df_one_day, df_multiple_years) -> updated

updated_cw
```

```
## # A tibble: 384 x 10
##    Battle      Start      End        day_c~1 month~2 year_~3 State CWSAC Theater
##    <chr>       <date>     <date>     <lgl>     <int>   <int> <chr> <chr> <chr>
##  1 Battle of ~ 1861-05-29 1861-06-01 TRUE          2       1 Virg~ D     Eastern
##  2 Siege of Y~ 1862-04-05 1862-05-04 TRUE          2       1 Virg~ B     Eastern
##  3 Battle of ~ 1862-05-31 1862-06-01 TRUE          2       1 Virg~ B     Eastern
##  4 Battle of ~ 1863-04-11 1863-05-04 TRUE          2       1 Virg~ C     Eastern
##  5 Battle of ~ 1863-04-11 1863-05-04 TRUE          2       1 Virg~ C     Eastern
##  6 Battle of ~ 1863-04-30 1863-05-06 TRUE          2       1 Virg~ A     Eastern
##  7 Battle of ~ 1863-11-27 1863-12-02 TRUE          2       1 Virg~ B     Eastern
##  8 Battle of ~ 1864-05-31 1864-06-12 TRUE          2       1 Virg~ A     Eastern
##  9 Battle of ~ 1864-09-30 1864-10-02 TRUE          2       1 Virg~ B     Eastern
## 10 Battle of ~ 1865-03-27 1865-04-08 TRUE          2       1 Alab~ B     Lower ~
## # ... with 374 more rows, 1 more variable: Outcome <chr>, and abbreviated
## #   variable names 1: day_count, 2: month_count, 3: year_count
```

8. Add a variable for the number of days for each battle and save the data frame.

- **After looking at the shortest number of days**, what were the median and mean number of days of battles?

```
updated_cw %>%
  mutate(duration = End - Start) -> updated_cw


mean(updated_cw$duration)
```

```
## Time difference of 1.846354 days
```

```
median(updated_cw$duration)
```

```
## Time difference of 0 days
```

The mean number of days of battle is 1.85 days. The median is 0 days.

- What percentage of battles were longer than average length? What does this suggest about the distribution of battle length

```
updated_cw %>%
  filter(duration > 1.846354) %>%
  nrow()/nrow(updated_cw) -> percent_cw
percent_cw*100
```

```
## [1] 19.79167
```

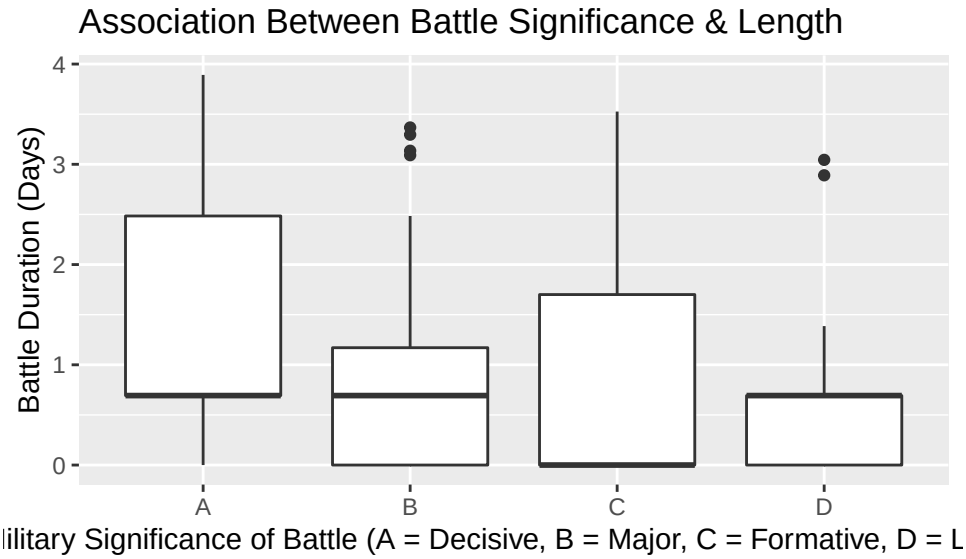
19.8% of the battles were longer than the average length of 1.85 days. This suggests that most battles were relatively short in nature – lasting less than 1.8 days. It also suggests that the data may be skewed to the right with the vast majority of battles (about 80 percent) being less than the mean (of 1.85 days).

9. Is there an association between the factor of CWSAC significance of a battle and the log of its length in days?

```
updated_cw %>%
  mutate(CWSAC = parse_factor(CWSAC)) %>%
  mutate(CWSAC = fct_relevel(CWSAC, c("A", "B", "C", "D"))) %>%
#reordering based on military significance, most significant = A [farthest left]; least significant = D
#`CWSAC`: A rating of the military significance of the battle by the Civil War Sites Advisory Commissio
  mutate(duration = as.numeric(duration)) %>%

  ggplot(mapping = aes(x = CWSAC, y= log(duration))) +
  geom_boxplot() +
  labs(y = "Battle Duration (Days)", x = "Military Significance of Battle (A = Decisive, B = Major, C =
  ggtitle("Association Between Battle Significance & Length")
```

```
## Warning: Removed 255 rows containing non-finite values (stat_boxplot).
```



Plot Interpretation: There appears to be a moderate association between military significance and battle length, generally the variability in battle duration decreases with military significance, however, it is unclear whether these differences are statistically significant.

- Create an appropriate plot.
- Interpret the plot in one sentence to answer the question.
- Use `aov()` to test whether the mean length of a battle is the same for each level of CWSAC significance and interpret the `summary()` results in one sentence using on the p-value.

```
aov(updated_cw$duration ~ updated_cw$CWSAC) -> aov_cw
summary(aov_cw)
```

```
##                    Df Sum Sq Mean Sq F value   Pr(>F)
## updated_cw$CWSAC    3    844  281.27   8.401 2.03e-05 ***
## Residuals         380  12722   33.48
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

AOV Interpretation: Because the p-value is sufficiently small ($p < .001$), we reject the null hypothesis and assume that there is a statistically significant difference in the mean values between the four groups (of military significance).

10. Review the Wikipedia page for this data. In just a few sentences, discuss whether you believe there are any ethical issues and/or potential for bias associated with the production of this data or the use of this data.

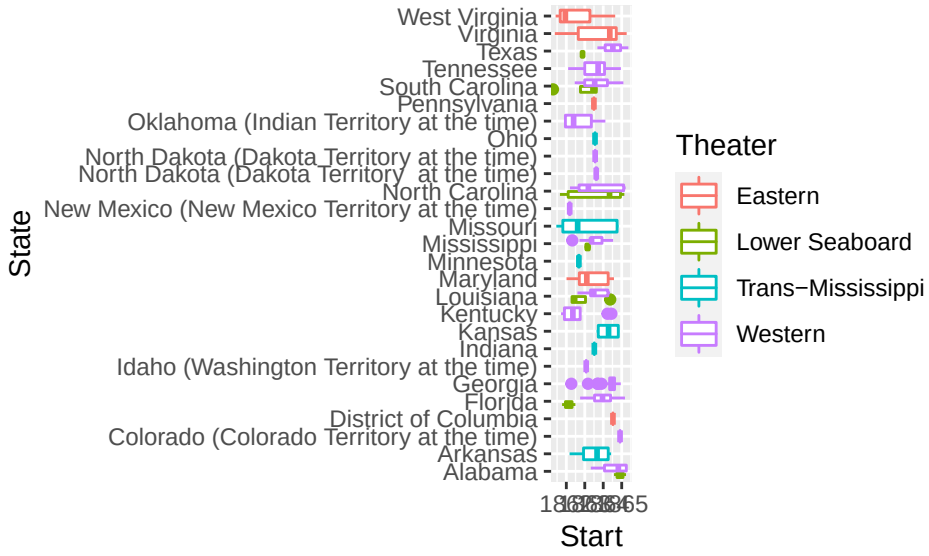
According to the Wikipedia page this data was collected by the US government in 1993 with the intention of "classifying the preservation status of historic battlefield land". With this in mind, it is reasonable to assume that if the government didn't want to spend resources preserving various battlefields that they would be biased towards minimizing the significance of various battles in this dataset. This could be a concern given the government is both the one *collecting* and *leveraging* this data.

11. Extra Credit: Did the theaters of war shift during the American Civil War?

- Reproduce this plot in R
  - Filter out states with two or fewer battles.
  - If a state has battles in different theaters, split the battles from the state into the correct theater.
- Interpret the plot in one sentence.

```
updated_cw %>%
  mutate(State = parse_factor(State)) %>%
  group_by(State) %>%
  summarise(count_battles = n()) %>%
  filter(count_battles >= 2) %>%
  pull(State) -> drop_states #states to drop


updated_cw %>%
  filter(State != drop_states) %>%
  select(State, Theater, Start) %>%
  group_by(State) %>%
  mutate(State = recode(State, "West Virginia (Virginia at the time)" = "West Virginia",
                        "North Dakota (Dakota Territory  at the time)" = "North Dakota (Dakota Territory
  ggplot(mapping = aes(x = Start, y = State, color = Theater)) +
  geom_boxplot()
```



Extra Credit (1 Pt)

1. Listen to the first 30 minutes or so (6:12 - 36:10) of the following podcast and provide a short answer describing the most interesting idea you heard.

SDS 337: Hadley Wickham Talks Integration and Future of R and Python