Tidyverse Examples

Evan Bowman

2023-03-24

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.

```
read_delim("data/enable1_words.txt", delim = " ", col_names = "words", show_col_types = F)
```

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

```
enable <- read_delim("data/enable1_words.txt", delim = " ", col_names = "words", show_col_types = F)
```

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?

```
most_ms <- enable |>
  mutate(count_m = str_count(words, "M|m"))|>
  arrange(desc(count_m))
tibble(most_ms)
```

```
## # A tibble: 172,820 x 2
##    words               count_m
##    <chr>                 <int>
##  1 immunocompromised         4
##  2 mammogram                 4
##  3 mammograms                4
##  4 mammonism                 4
##  5 mammonisms                4
##  6 mesembryanthemum          4
##  7 mesembryanthemums         4
##  8 agammaglobulinemia        3
##  9 agammaglobulinemias       3
## 10 agammaglobulinemic        3
## # ... with 172,810 more rows
```

```
most_ms|>
  slice(8:172820)|>
  mutate(length = str_length(words))|>
  filter(count_m == 3)|>
  slice_max(n = 6, order_by = length)
```

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

The code returned 11 words instead of 6 because the code returns the six longest words but will include any other words of the same length. In this case, the slice_max command returned an additional five words since they also had 19 letters in them.

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.

- "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.

```
enable_identical <- enable |>
  mutate(identical_halves = str_detect(words, "^(.+).?\\1$"))
```

```
enable_identical|>
  group_by(identical_halves) |>
  count()
```

```
## # A tibble: 2 x 2
## # Groups:   identical_halves [2]
##   identical_halves      n
##   <lgl>             <int>
## 1 FALSE            172686
## 2 TRUE                134
```

There are 134 letters that have an identical first and second half of the word.

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_identical|>
  filter(identical_halves == TRUE)|>
  mutate(length = str_count(words))|>
  arrange(desc(length))
```

```
## # A tibble: 134 x 3
##    words     identical_halves 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
```

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.

```
country_codes <- gapminder::country_codes
country <- read_csv("data/country.csv", show_col_types = F)
```

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, country, 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`.

```
country_codes_wb <-inner_join(country_codes, country, by = c("iso_alpha" = "Country Code"))|>
  rename("wb_name" = "TableName")
head(country_codes_wb)
```

```
## # A tibble: 6 x 6
##   country     iso_alpha iso_num Region                    IncomeGroup   wb_name
##   <chr>       <chr>       <int> <chr>                     <chr>         <chr>
## 1 Afghanistan AFG             4 South Asia                Low income    Afghan~
## 2 Albania     ALB             8 Europe & Central Asia     Upper middle~ Albania
## 3 Algeria     DZA            12 Middle East & North Africa Upper middle~ Algeria
## 4 Angola      AGO            24 Sub-Saharan Africa        Lower middle~ Angola
## 5 Argentina   ARG            32 Latin America & Caribbean High income   Argent~
## 6 Armenia     ARM            51 Europe & Central Asia     Upper middle~ Armenia
```

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

```
str_subset(country_codes_wb$wb_name, "[:punct:]")
```

```
##  [1] "Bahamas, The"          "Congo, Dem. Rep."      "Congo, Rep."
##  [4] "Côte d'Ivoire"         "Egypt, Arab Rep."      "Gambia, The"
##  [7] "Guinea-Bissau"         "Hong Kong SAR, China"  "Iran, Islamic Rep."
## [10] "Korea, Rep."           "Korea, Rep."           "Macao SAR, China"
## [13] "Micronesia, Fed. Sts." "Timor-Leste"           "Venezuela, RB"
## [16] "Yemen, Rep."
```

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

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 _.

```
# a)
country_codes_wb<- country_codes_wb|>
  mutate(format_name = str_replace_all(wb_name, "[[:punct:] [:blank:]]" , "_"), .after = country)

# b)
country_codes_wb <- country_codes_wb|>
  mutate(format_name = str_remove_all(format_name, "_$"), .after = country)

# c)
country_codes_wb <- country_codes_wb|>
  mutate(format_name = str_replace_all(format_name, "__", "_"), .after = country)

tibble(country_codes_wb)
```

```
## # A tibble: 181 x 7
##    country     format_name iso_alpha iso_num Region              Incom~1 wb_name
##    <chr>       <chr>       <chr>       <int> <chr>               <chr>   <chr>
##  1 Afghanistan Afghanistan AFG             4 South Asia          Low in~ Afghan~
##  2 Albania     Albania     ALB             8 Europe & Central A~ Upper ~ Albania
##  3 Algeria     Algeria     DZA            12 Middle East & Nort~ Upper ~ Algeria
##  4 Angola      Angola      AGO            24 Sub-Saharan Africa  Lower ~ Angola
##  5 Argentina   Argentina   ARG            32 Latin America & Ca~ High i~ Argent~
##  6 Armenia     Armenia     ARM            51 Europe & Central A~ Upper ~ Armenia
##  7 Aruba       Aruba       ABW           533 Latin America & Ca~ High i~ Aruba
##  8 Australia   Australia   AUS            36 East Asia & Pacific High i~ Austra~
##  9 Austria     Austria     AUT            40 Europe & Central A~ High i~ Austria
## 10 Azerbaijan  Azerbaijan  AZE            31 Europe & Central A~ Upper ~ Azerba~
## # ... with 171 more rows, and abbreviated variable name 1: IncomeGroup
```

- 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|>
  filter(str_detect(wb_name, "[:punct:]"))
```

```
## # A tibble: 16 x 7
##    country              format_name      iso_a~1 iso_num Region Incom~2 wb_name
##    <chr>                <chr>            <chr>     <int> <chr>  <chr>   <chr>
##  1 Bahamas              Bahamas_The      BHS          44 Latin~ High i~ Bahama~
##  2 Congo, Dem. Rep.     Congo_Dem_Rep    COD         180 Sub-S~ Low in~ Congo,~
##  3 Congo, Rep.          Congo_Rep        COG         178 Sub-S~ Lower ~ Congo,~
##  4 Cote d'Ivoire        Côte_d_Ivoire    CIV         384 Sub-S~ Lower ~ Côte d~
##  5 Egypt                Egypt_Arab_Rep   EGY         818 Middl~ Lower ~ Egypt,~
##  6 Gambia               Gambia_The       GMB         270 Sub-S~ Low in~ Gambia~
##  7 Guinea-Bissau        Guinea_Bissau    GNB         624 Sub-S~ Low in~ Guinea~
##  8 Hong Kong, China     Hong_Kong_SAR_C~ HKG         344 East ~ High i~ Hong K~
##  9 Iran                 Iran_Islamic_Rep IRN         364 Middl~ Upper ~ Iran, ~
## 10 Korea, Dem. Rep.     Korea_Rep        KOR         410 East ~ High i~ Korea,~
## 11 Korea, Rep.          Korea_Rep        KOR         410 East ~ High i~ Korea,~
```

```
## 12 Macao, China          Macao_SAR_China  MAC         446 East ~ High i~ Macao ~
## 13 Micronesia, Fed. Sts. Micronesia_Fed_~ FSM         583 East ~ Lower ~ Micron~
## 14 Timor-Leste           Timor_Leste      TLS         626 East ~ Lower ~ Timor-~
## 15 Venezuela             Venezuela_RB     VEN         862 Latin~ Upper ~ Venezu~
## 16 Yemen, Rep.           Yemen_Rep        YEM         887 Middl~ Low in~ Yemen,~
## # ... with abbreviated variable names 1: iso_alpha, 2: IncomeGroup
```

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.

```
cw_battles <- read_csv("data/civil_war_theater.csv", show_col_types = F)
head(cw_battles)
```

```
## # A tibble: 6 x 6
##   Battle                                Date          State CWSAC Theater Outcome
##   <chr>                                 <chr>         <chr> <chr> <chr>   <chr>
## 1 Battle of Fort Stevens                July 11-12, ~ Dist~ B     Eastern Union ~
## 2 Battle of Hancock                     January 5-6,~ Mary~ D     Eastern Inconc~
## 3 Battle of South Mountainor Boonsboro  September 14~ Mary~ B     Eastern Union ~
## 4 Battle of Antietam or Sharpsburg      September 17~ Mary~ A     Eastern Tactic~
## 5 Battle of Williamsport                July 6-16, 1~ Mary~ C     Eastern Inconc~
## 6 Battle of Boonsboro                   July 8, 1863  Mary~ D     Eastern Inconc~
```

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"
```

```
# Using year_regex variable we can use str_count to identify if any of the year_regex values are in the

cw_battles <- cw_battles |>
  mutate(years = str_count(Date, year_regex), .after = Date)
head(cw_battles)
```

```
## # A tibble: 6 x 7
##   Battle                                Date    years State CWSAC Theater Outcome
##   <chr>                                 <chr>   <int> <chr> <chr> <chr>   <chr>
## 1 Battle of Fort Stevens                July 1~     1 Dist~ B     Eastern Union ~
## 2 Battle of Hancock                     Januar~     1 Mary~ D     Eastern Inconc~
## 3 Battle of South Mountainor Boonsboro Septem~     1 Mary~ B     Eastern Union ~
## 4 Battle of Antietam or Sharpsburg      Septem~     1 Mary~ A     Eastern Tactic~
## 5 Battle of Williamsport                July 6~     1 Mary~ C     Eastern Inconc~
## 6 Battle of Boonsboro                   July 8~     1 Mary~ D     Eastern Inconc~
```

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

- 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).

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

# Using the same process as the year calculation, we can identify how many months were in each battle.
cw_battles <- cw_battles |>
  mutate(months_dur = str_count(Date, month_regex), .after = Date)
head(cw_battles)
```

```
## # A tibble: 6 x 8
##   Battle                          Date  month~1 years State CWSAC Theater Outcome
##   <chr>                           <chr>   <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battle of Fort Stevens          July~       1     1 Dist~ B     Eastern Union ~
## 2 Battle of Hancock               Janu~       1     1 Mary~ D     Eastern Inconc~
## 3 Battle of South Mountainor Bo~ Sept~       1     1 Mary~ B     Eastern Union ~
## 4 Battle of Antietam or Sharpsb~ Sept~       1     1 Mary~ A     Eastern Tactic~
## 5 Battle of Williamsport          July~       1     1 Mary~ C     Eastern Inconc~
## 6 Battle of Boonsboro             July~       1     1 Mary~ D     Eastern Inconc~
## # ... with abbreviated variable name 1: months_dur
```

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`.

```
# The '-' represents a battle spanning multiple days. Str_detect would allow for the required True/Fals
cw_battles <- cw_battles |>
  mutate(multi_days = str_detect(Date, "-"), .after = Date)
head(cw_battles)
```

```
## # A tibble: 6 x 9
##   Battle                 Date  multi~1 month~2 years State CWSAC Theater Outcome
##   <chr>                  <chr> <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battle of Fort Stevens July~ TRUE          1     1 Dist~ B     Eastern Union ~
## 2 Battle of Hancock      Janu~ TRUE          1     1 Mary~ D     Eastern Inconc~
## 3 Battle of South Mount~ Sept~ FALSE         1     1 Mary~ B     Eastern Union ~
## 4 Battle of Antietam or~ Sept~ FALSE         1     1 Mary~ A     Eastern Tactic~
## 5 Battle of Williamsport July~ TRUE          1     1 Mary~ C     Eastern Inconc~
## 6 Battle of Boonsboro    July~ FALSE         1     1 Mary~ D     Eastern Inconc~
## # ... with abbreviated variable names 1: multi_days, 2: months_dur
```

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,

```
day <- cw_battles |>
  filter(multi_days == FALSE)
tibble(day)
```

```
## # A tibble: 255 x 9
##    Battle                Date  multi~1 month~2 years State CWSAC Theater Outcome
##    <chr>                 <chr> <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
##  1 Battle of South Moun~ Sept~ FALSE         1     1 Mary~ B     Eastern "Union~
##  2 Battle of Antietam o~ Sept~ FALSE         1     1 Mary~ A     Eastern "Tacti~
##  3 Battle of Boonsboro   July~ FALSE         1     1 Mary~ D     Eastern "Incon~
##  4 Battle of Monocacy (~ July~ FALSE         1     1 Mary~ B     Eastern "Confe~
##  5 Battle of Folck's Mi~ Augu~ FALSE         1     1 Mary~ D     Eastern "Incon~
##  6 Battle of Hanover     June~ FALSE         1     1 Penn~ C     Eastern "Incon~
##  7 Battle of Big Bethel  June~ FALSE         1     1 Virg~ C     Eastern "Confe~
##  8 Battle of Blackburn'~ July~ FALSE         1     1 Virg~ C     Eastern "Confe~
##  9 First Battle of Bull~ July~ FALSE         1     1 Virg~ A     Eastern "Confe~
## 10 Battle of Ball's Blu~ Octo~ FALSE         1     1 Virg~ B     Eastern "Confe~
## # ... with 245 more rows, and abbreviated variable names 1: multi_days,
## #   2: months_dur
```

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

```
days_month <- cw_battles |>
  filter(multi_days == TRUE & months_dur == 1)
tibble(days_month)
```

```
## # A tibble: 103 x 9
##    Battle                Date  multi~1 month~2 years State CWSAC Theater Outcome
##    <chr>                 <chr> <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
##  1 Battle of Fort Steve~ July~ TRUE          1     1 Dist~ B     Eastern Union ~
##  2 Battle of Hancock     Janu~ TRUE          1     1 Mary~ D     Eastern Inconc~
```

```
##  3 Battle of Williamspo~ July~ TRUE          1     1 Mary~ C     Eastern Inconc~
##  4 Battle of Gettysburg  July~ TRUE          1     1 Penn~ A     Eastern Union ~
##  5 Battle of Sewell's P~ May ~ TRUE          1     1 Virg~ D     Eastern Inconc~
##  6 Battle of Hampton Ro~ Marc~ TRUE          1     1 Virg~ B     Eastern Inconc~
##  7 Battle of Garnett's ~ June~ TRUE          1     1 Virg~ D     Eastern Inconc~
##  8 First Battle of Rapp~ Augu~ TRUE          1     1 Virg~ D     Eastern Inconc~
##  9 Battle of Manassas S~ Augu~ TRUE          1     1 Virg~ B     Eastern Confed~
## 10 Second Battle of Bul~ Augu~ TRUE          1     1 Virg~ A     Eastern Confed~
## # ... with 93 more rows, and abbreviated variable names 1: multi_days,
## #   2: months_dur
```

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

```
months <- cw_battles |>
  filter(months_dur != 1 & years == 1)
tibble(months)
```

```
## # A tibble: 25 x 9
##    Battle                Date  multi~1 month~2 years State CWSAC Theater Outcome
##    <chr>                 <chr> <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
##  1 Battle of Aquia Creek May ~ TRUE          2     1 Virg~ D     Eastern "Incon~
##  2 Siege of Yorktown (1~ Apri~ TRUE          2     1 Virg~ B     Eastern "Incon~
##  3 Battle of Seven Pines May ~ TRUE          2     1 Virg~ B     Eastern "Incon~
##  4 Battle of Suffolk (H~ Apri~ TRUE          2     1 Virg~ C     Eastern "Incon~
##  5 Battle of Suffolk (N~ Apri~ TRUE          2     1 Virg~ C     Eastern "Incon~
##  6 Battle of Chancellor~ Apri~ TRUE          2     1 Virg~ A     Eastern "Confe~
##  7 Battle of Mine Run    Nove~ TRUE          2     1 Virg~ B     Eastern "Incon~
##  8 Battle of Cold Harbor May ~ TRUE          2     1 Virg~ A     Eastern "Confe~
##  9 Battle of Peebles' F~ Sept~ TRUE          2     1 Virg~ B     Eastern "Union~
## 10 Battle of Spanish Fo~ Marc~ TRUE          2     1 Alab~ B     Lower ~ "Union~
## # ... with 15 more rows, and abbreviated variable names 1: multi_days,
## #   2: months_dur
```

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

```
years <- cw_battles |>
  filter(years != 1)
tibble(years)
```

```
## # A tibble: 1 x 9
##   Battle                Date  multi~1 month~2 years State CWSAC Theater Outcome
##   <chr>                 <chr> <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battle of Stones Rive~ Dece~ TRUE          2     2 Tenn~ A     Western Union ~
## # ... with abbreviated variable names 1: multi_days, 2: months_dur
```

- How many rows are in each data frame?

```
nrow(day)
```

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

```
nrow(days_month)
```

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

```
nrow(months)
```

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

```
nrow(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,

```
diff_rows <- nrow(day) + nrow(days_month) + nrow(months) + nrow(years)
cw_rows <- nrow(cw_battles)

identical(diff_rows, cw_rows)
```

```
## [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.**
- Remove the `Date` variable from each data frame.
- Save the data frame.

```
# Day
day1 <- day |>
  mutate(start = mdy(Date),
         end = mdy(Date), .after = Date) |>
  select(-Date)
head(day1)
```

```
## # A tibble: 6 x 10
##   Battle start      end        multi~1 month~2 years State CWSAC Theater Outcome
##   <chr>  <date>     <date>     <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battl~ 1862-09-14 1862-09-14 FALSE         1     1 Mary~ B     Eastern Union ~
## 2 Battl~ 1862-09-17 1862-09-17 FALSE         1     1 Mary~ A     Eastern Tactic~
## 3 Battl~ 1863-07-08 1863-07-08 FALSE         1     1 Mary~ D     Eastern Inconc~
## 4 Battl~ 1864-07-09 1864-07-09 FALSE         1     1 Mary~ B     Eastern Confed~
## 5 Battl~ 1864-08-01 1864-08-01 FALSE         1     1 Mary~ D     Eastern Inconc~
## 6 Battl~ 1863-06-30 1863-06-30 FALSE         1     1 Penn~ C     Eastern Inconc~
## # ... with abbreviated variable names 1: multi_days, 2: months_dur
```

```
# Days and Month
days_month1 <- days_month |>
  separate(col = Date, into = c("Month", "Start", "End", "Year")) |>
  mutate(month = Month, year = Year) |>
  unite("start", c(Month, Start, Year), sep = " ") |>
  unite("end", c(month, End, year), sep = " ") |>
  mutate(start = mdy(start),
         end = mdy(end))
head(days_month1)
```

```
## # A tibble: 6 x 10
##   Battle start      end        multi~1 month~2 years State CWSAC Theater Outcome
##   <chr>  <date>     <date>     <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battl~ 1864-07-11 1864-07-12 TRUE          1     1 Dist~ B     Eastern Union ~
## 2 Battl~ 1862-01-05 1862-01-06 TRUE          1     1 Mary~ D     Eastern Inconc~
## 3 Battl~ 1863-07-06 1863-07-16 TRUE          1     1 Mary~ C     Eastern Inconc~
## 4 Battl~ 1863-07-01 1863-07-03 TRUE          1     1 Penn~ A     Eastern Union ~
## 5 Battl~ 1861-05-18 1861-05-19 TRUE          1     1 Virg~ D     Eastern Inconc~
## 6 Battl~ 1862-03-08 1862-03-09 TRUE          1     1 Virg~ B     Eastern Inconc~
## # ... with abbreviated variable names 1: multi_days, 2: months_dur
```

```
#  Multiple Months
months1 <- months |>
  separate(Date, into = c("start", "end"), sep = "-") |>
  separate(end, into = c("end", "year"), sep = ",") |>
  mutate(Year = year) |>
  unite("start", c(start, year), sep = " ") |>
  unite("end", c(end, Year), sep = " ") |>
  mutate(start = mdy(start),
         end = mdy(end))
head(months1)
```

```
## # A tibble: 6 x 10
##   Battle start      end        multi~1 month~2 years State CWSAC Theater Outcome
##   <chr>  <date>     <date>     <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battl~ 1861-05-29 1861-06-01 TRUE          2     1 Virg~ D     Eastern Inconc~
## 2 Siege~ 1862-04-05 1862-05-04 TRUE          2     1 Virg~ B     Eastern Inconc~
## 3 Battl~ 1862-05-31 1862-06-01 TRUE          2     1 Virg~ B     Eastern Inconc~
## 4 Battl~ 1863-04-11 1863-05-04 TRUE          2     1 Virg~ C     Eastern Inconc~
## 5 Battl~ 1863-04-11 1863-05-04 TRUE          2     1 Virg~ C     Eastern Inconc~
## 6 Battl~ 1863-04-30 1863-05-06 TRUE          2     1 Virg~ A     Eastern Confed~
## # ... with abbreviated variable names 1: multi_days, 2: months_dur
```

```
# Multiple Years
years1 <- years |>
  separate(Date, into = c("start", "end"), sep = "-") |>
  mutate(start = mdy(start),
         end = mdy(end))
head(years1)
```

```
## # A tibble: 1 x 10
##   Battle start      end        multi~1 month~2 years State CWSAC Theater Outcome
```

```
##   <chr>  <date>     <date>     <lgl>     <int> <int> <chr> <chr> <chr>   <chr>
## 1 Battl~ 1862-12-31 1863-01-02 TRUE          2     2 Tenn~ A     Western Union ~
## # ... with abbreviated variable names 1: multi_days, 2: months_dur
```

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.

```
cw_updated <- bind_rows(day1, days_month1, months1, years1)
identical(cw_rows, nrow(cw_updated))
```

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

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

```
cw_updated_days <- cw_updated |>
  mutate(days = (end - start) +1, .after = end)
head(cw_updated_days)
```

```
## # A tibble: 6 x 11
##   Battle   start      end        days  multi~1 month~2 years State CWSAC Theater
##   <chr>    <date>     <date>     <drt> <lgl>     <int> <int> <chr> <chr> <chr>
## 1 Battle ~ 1862-09-14 1862-09-14 1 da~ FALSE         1     1 Mary~ B     Eastern
## 2 Battle ~ 1862-09-17 1862-09-17 1 da~ FALSE         1     1 Mary~ A     Eastern
## 3 Battle ~ 1863-07-08 1863-07-08 1 da~ FALSE         1     1 Mary~ D     Eastern
## 4 Battle ~ 1864-07-09 1864-07-09 1 da~ FALSE         1     1 Mary~ B     Eastern
## 5 Battle ~ 1864-08-01 1864-08-01 1 da~ FALSE         1     1 Mary~ D     Eastern
## 6 Battle ~ 1863-06-30 1863-06-30 1 da~ FALSE         1     1 Penn~ C     Eastern
## # ... with 1 more variable: Outcome <chr>, and abbreviated variable names
## #   1: multi_days, 2: months_dur
```

```
# Median/Mean number of days
median(cw_updated_days$days)
```

```
## Time difference of 1 days
```

```
mean(cw_updated_days$days)
```

```
## Time difference of 2.846354 days
```

```
# Percentage of Battles
percentage <- (sum(cw_updated_days$days > 2.846354) / nrow(cw_updated_days)) * 100
percentage
```

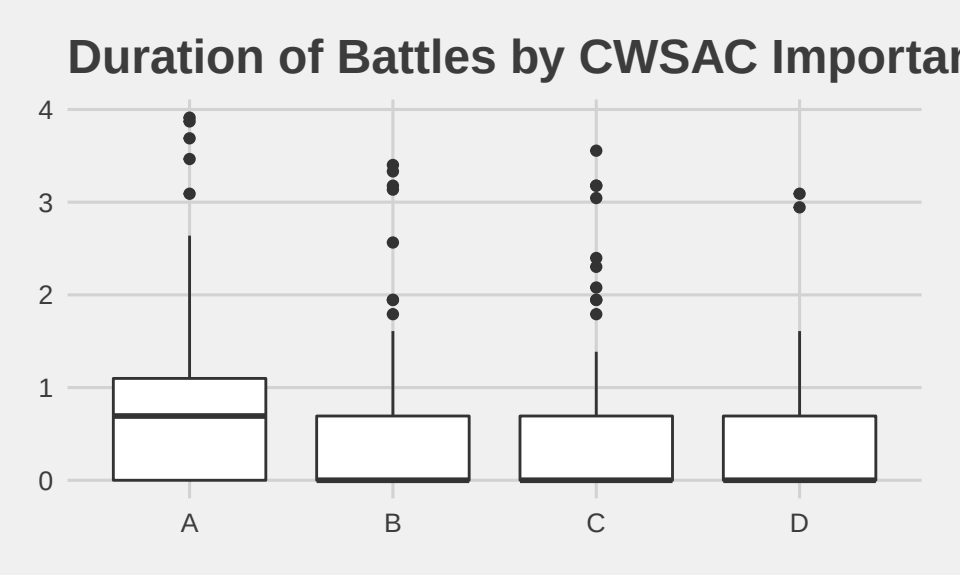
```
## [1] 19.79167
```

Only about 20 percent of battles had a longer duration than the mean days for all battles. This tells us that the distribution of battle duration is heavily right skewed.

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

- 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.

```
# Plot
cw_updated_days |>
  ggplot(aes(CWSAC, log(as.numeric(days))))+
  geom_boxplot()+
  ggtitle("Duration of Battles by CWSAC Importance")+
  xlab("CWSAC Level")+
  ylab("Days")+
  ggthemes::theme_fivethirtyeight()
```



From the plot, it appears that there is a slight difference between the mean duration for level A and the other three CWSAC levels.

```
#aov
cw_aov <-aov(days ~ CWSAC, data = cw_updated_days)
broom::tidy(cw_aov)
```

```
## # A tibble: 2 x 6
##   term         df  sumsq meansq statistic    p.value
##   <chr>     <dbl>  <dbl>  <dbl>     <dbl>      <dbl>
## 1 CWSAC         3   844.  281.       8.40  0.0000203
## 2 Residuals   380 12722.   33.5     NA    NA
```

Analysis of variance confirms the hypothesis from the plot. With a p-value close to zero we reject the null hypothesis. There is evidence at least one CWSAC level's mean duration is different from the other levels.