Homework 4 Solution

Disclaimer

This homework solution is for the sole benefit of students taking Stat 220 from Prof. Bastola during Spring term 2024. Dissemination of this solution to people who are not registered for this course is not permitted and will be considered grounds for Academic Dishonesty for the all individuals involved in the giving and receiving of the solution.

Assignment prompt

Problem 1: flights

Use the `flights` data frame from the `nycflights13` package to answer the questions below. (see help `?nycflights13::flights` for more details) Use `dplyr` to answer the questions below.

a.

What plane (by `tailnum`) traveled the most times from NYC airports in 2013?

answer:

Flight tail number N725MQ has the most flights from NYC airports in 2013 with 575 flights.

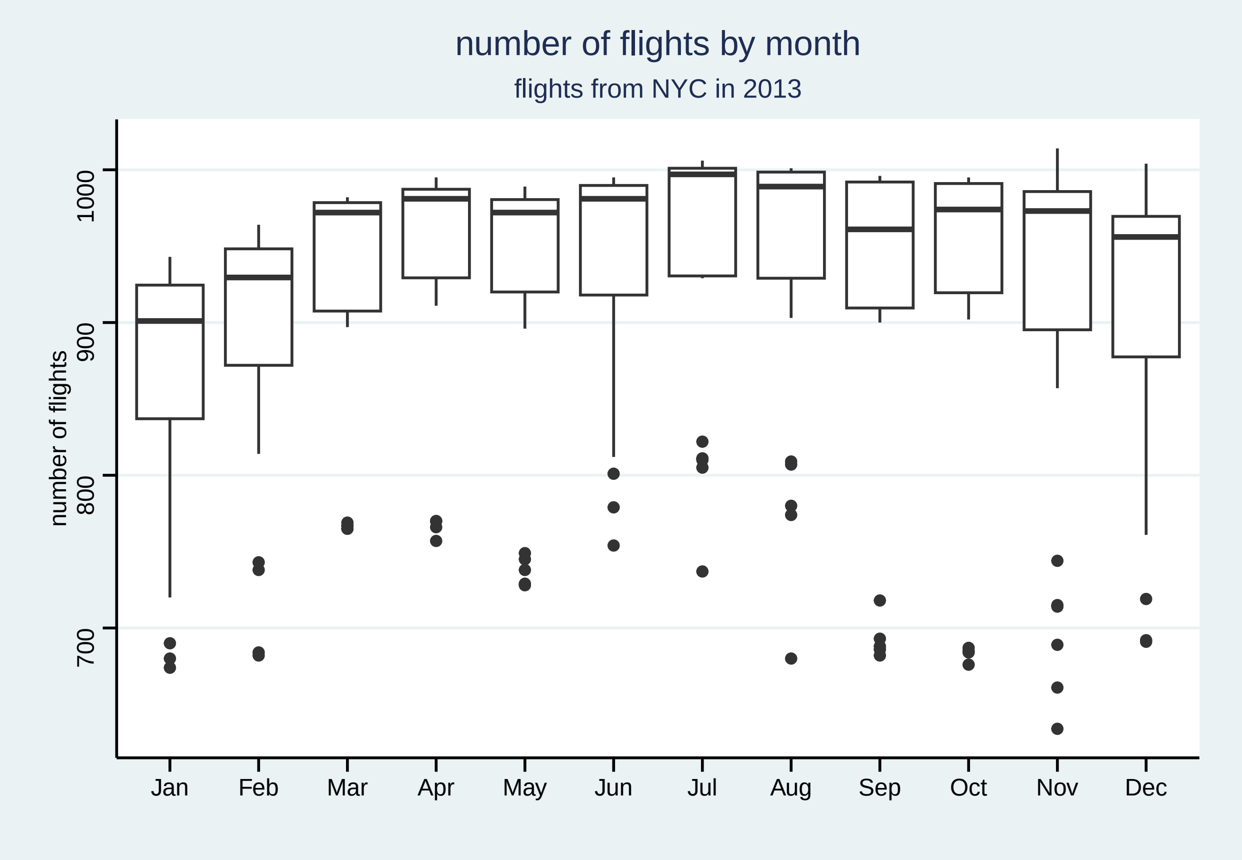
```
> flights %>%
+   count(tailnum, sort = TRUE)
# A tibble: 4,044 x 2
   tailnum     n
   <chr>   <int>
 1 <NA>     2512
 2 N725MQ    575
 3 N722MQ    513
 4 N723MQ    507
 5 N711MQ    486
 6 N713MQ    483
 7 N258JB    427
 8 N298JB    407
 9 N353JB    404
10 N351JB    402
# i 4,034 more rows
```

b.

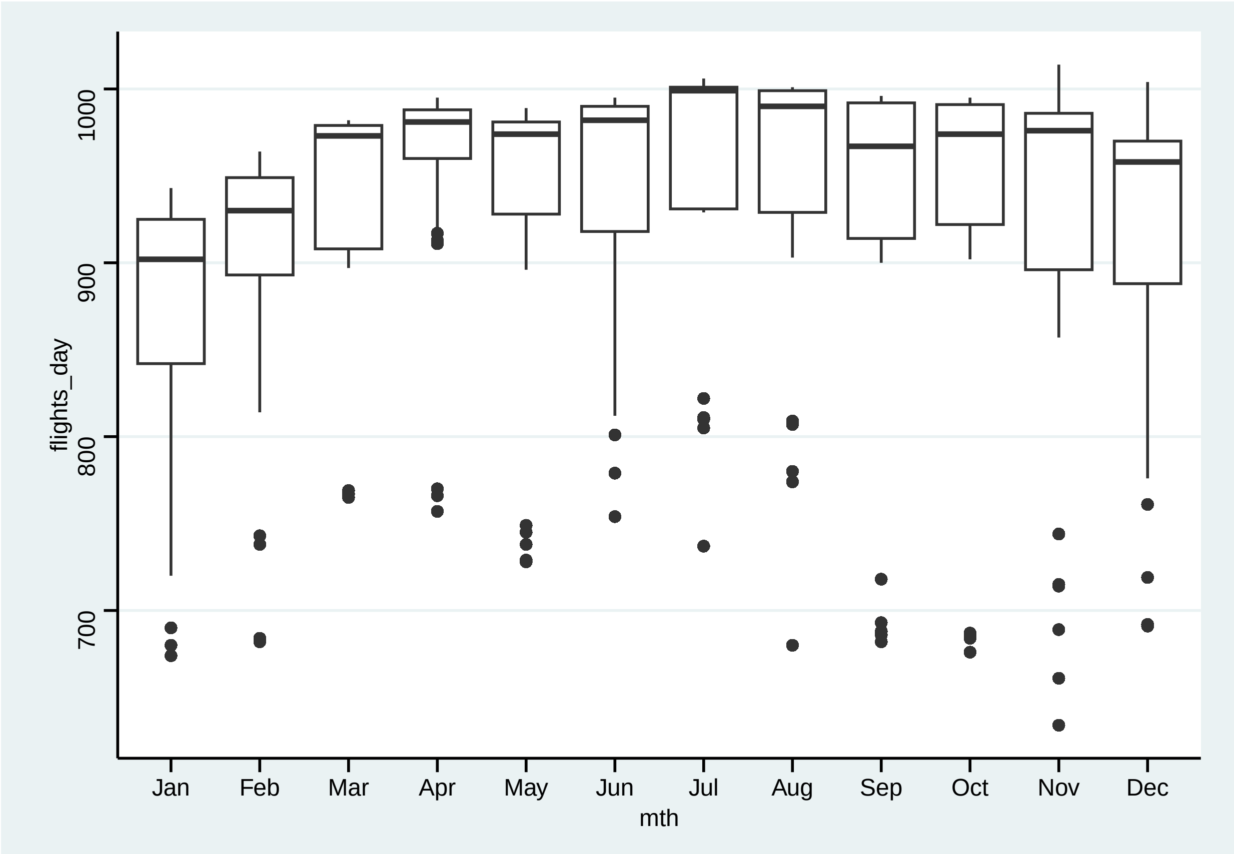
Load the `lubridate` package, then add a labeled month variable called `mth` using the command `mth = month(time_hour, label = TRUE)`. Then compute the number of flights from NYC airports for each day in 2013 and create a boxplot (in `ggplot2`) of number of flights per day by month (using `mth`). Which time of the year tends to see the most flights from NYC?

answer: Departures seem highest in the summer and lowest at the start of the year (Jan/Feb).

```r
> flights %>%
+   mutate(mth = month(time_hour, label = TRUE)) %>% # month
+   count(mth, day) %>%    # number of flights for each day
+   ggplot(aes(x = mth, y = n)) +
+     geom_boxplot() +
+     labs(
+       x = "",
+       y = "number of flights",
+       title = "number of flights by month",
+       subtitle = "flights from NYC in 2013")
```




```r
> flights %>% mutate(mth = month(time_hour, label = TRUE)) %>%
+  group_by(mth, day) %>%
+  mutate(flights_day = n()) %>%
+  ggplot(aes(x = mth, y = flights_day)) +
+   geom_boxplot()
```

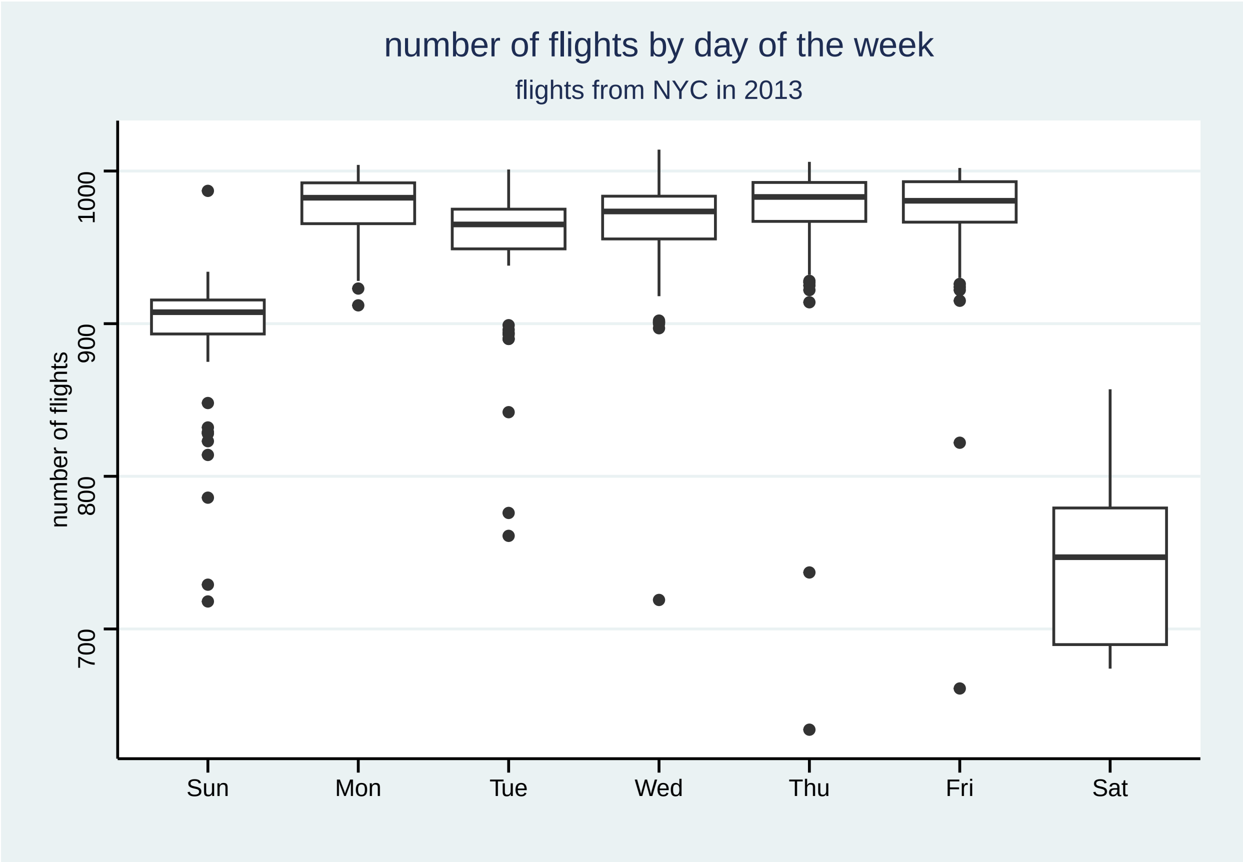


c.

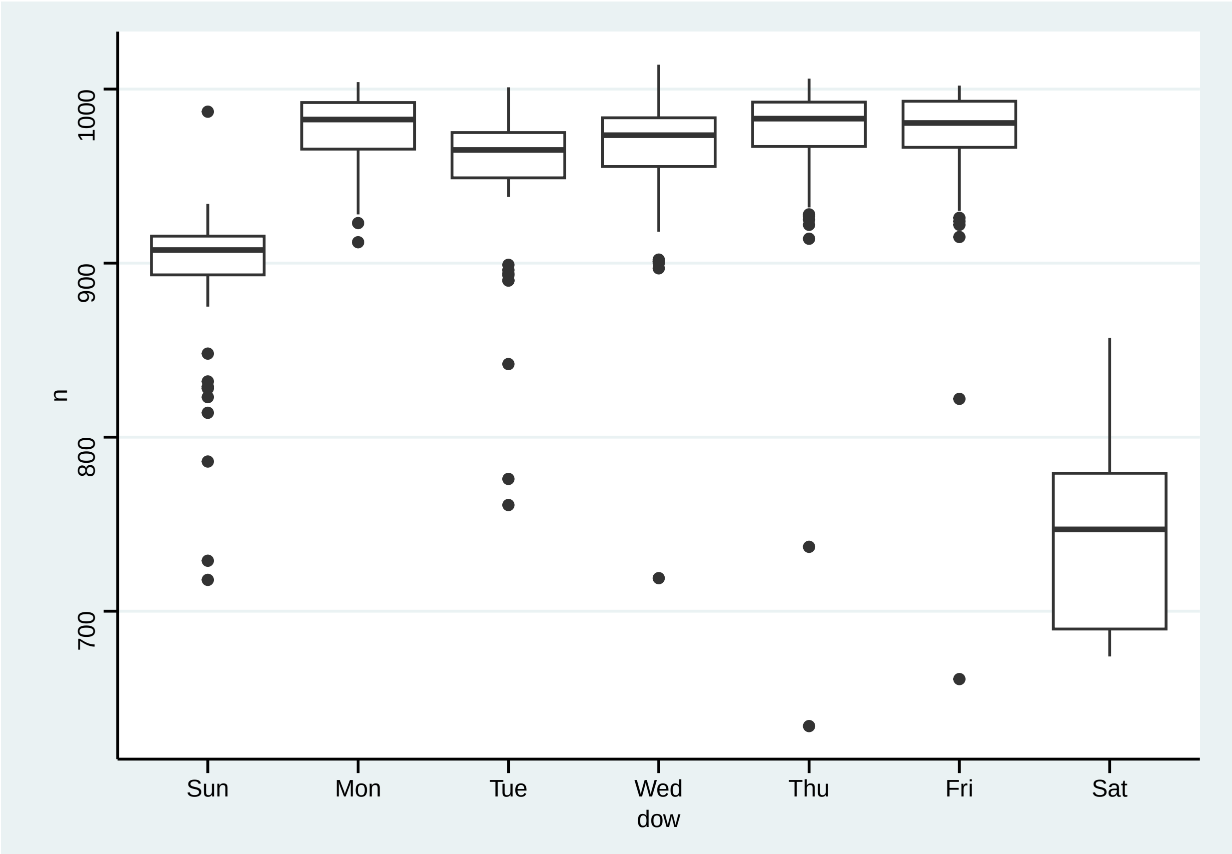
Use the `lubridate` package to add a labeled day of the week variable called `dow` using the command `dow = wday(time_hour, label = TRUE)`. Then compute the number of flights from NYC airports for each day in 2013 and create a boxplot (in `ggplot2`) of number of flights per day by day of the week. Which day of the week sees the fewest flights from NYC?

answer: Saturday has by far the fewest departures (but largest IQR of departures). Note the different approach used below to both count the number of flights in a day (using month/day of the month combos) and record the day of the week by recording the `first` value of `dow` in each month/day of the month grouping. Using `first` is one way to keep the value of a variable that is constant over all grouping entries.

```
> flights %>%
+   mutate(dow = wday(time_hour, label = TRUE)) %>% # add weekday
+   group_by(month, day) %>% # group by day of year
+   summarize(
+     n = n(),     # number of flights that day
+     dow = first(dow)    # need to keep day of week
+     ) %>%
+   ggplot(aes(x = dow, y = n)) +
+     geom_boxplot() +
+     labs(
+       x = "",
+       y = "number of flights",
+       title = "number of flights by day of the week",
+       subtitle = "flights from NYC in 2013")
```




```
> flights %>% mutate(dow = wday(time_hour, label = TRUE)) %>%
+  group_by(month, day) %>%
+  summarize(
+     n = n(),      # number of flights that day
+     dow = first(dow)    # need to keep day of week
+     ) %>%
+  ggplot(aes(x = dow, y = n)) +
+   geom_boxplot()
```



Problem 2: top destinations

More with the `nycflights13` data. Consider `top_dest`, the top 10 destinations out of NYC area in 2013:

```
> top_dest <- flights %>%
+   count(dest) %>%
+   slice_max(n, n = 10)
```

a.

Use a `dplyr` filtering join command to create a `flights` data subset that only contains destinations in the `top_dest` top 10 destinations. What is the dimension of this data set?

answer:

```
> flights10 <- semi_join(flights, top_dest)
> dim(flights10)
[1] 141145     19
> flights10 %>% slice(1:2)
# A tibble: 2 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1  2013     1     1      542            540         2      923            850
2  2013     1     1      554            600        -6      812            837
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
```

```
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

The number of rows is 141145 and columns is 19

A filtering join is done with the `semi_join` command. This selects the rows in `flights` that have a destination in `top_dest` but it does *not* add the columns of `top_dest` into the resulting data frame.

Joins like `left_join`, `inner_join`, etc, are not filtering joins, but rather, they will merge all columns of both data sets into one data frame. The number of rows may be correct with `left` or `inner` joins, but the column counts will be off.

b.

Use your joined data from (a) to compute the median number of minutes between flights to each destination. Hint: Use the `make_datetime` function to convert the scheduled departure date/time (year,month,day,hour,minute) to a date/time variable, then use either the `difftime` or `interval` and `int_length` function to compute the number of minutes between scheduled departures for each destination. (Note that we can't use the `time_hour` variable because it doesn't have a minute measure in it.)

Answer:

Here we create a date/time variable from individual components. (But, as an alternative way to get scheduled time stamp, we could also use the `update(time_hour, minute=minute)` command to update this variable with the scheduled `minute` variable in the data.)

```
> flights10 <- flights10 %>%
+   mutate(sch_datetime = make_datetime(  # one way to get scheduled time/date variable
+     year = year,
+     month = month,
+     day = day,
+     hour = hour,
+     min = minute)
+     )
```

After grouping by destination, you need to verify that the rows are arranged by time (otherwise time differences between successive rows is meaningless).

One way to get median time is to find the `interval` length between one data/time and the date/time above it (`lag`). Here we need to divide by 60 since `int_length` are measured in seconds:

```
> flights10 %>%
+   group_by(dest) %>%      # group by destination
+   arrange(sch_datetime) %>%    # make sure ordered earliest to latest
+   mutate(diff = int_length(interval(lag(sch_datetime), sch_datetime))/60) %>%   #get time between fli
+   summarize(medianMins = median(diff, na.rm=TRUE))
# A tibble: 10 x 2
   dest  medianMins
   <chr>      <dbl>
 1 ATL           15
 2 BOS           17
 3 CLT           18
 4 DCA           34
 5 FLL           24
 6 LAX           19
 7 MCO           20
 8 MIA           25
```

```
 9 ORD              15
10 SFO              20
```

Problem 3: Energy

The energy dataset `EnergyData1516.csv` gives energy use (kiloWatt hour) every 15 minutes for the 15-16 academic year for all buildings on campus that have an energy meter installed.

Read the energy data in again using the `read_csv` command below that specifies column type for `Timestamp` and `dayWeek` and defaults to double types for all other. The order of the factor levels of `dayWeek` are also given so we get days ordered correctly in plots. Note that you will need to wrap a variable name in backticks if it starts with a non-letter character. See the `glimpse` command below for an example.

```
> energy <- readr::read_csv("https://raw.githubusercontent.com/deepbas/statdatasets/main/energy.csv",
+                       col_type = cols(
+                        .default = col_double(),
+                         Timestamp = col_datetime(format = ""),
+                         dayWeek = col_factor(levels=c("Mon","Tues","Wed","Thurs","Fri","Sat","Sun"))
+                        ))
> dim(energy)
[1] 35129    90
```

a.

Create a tidy version of the `energy` data called `energy_narrow` that puts building names in a column called `building` and energy values into a column called `energyKWH`. The chunk below gives you a start to this task. Check that your `energy_narrow` data frame contains 2,880,578 rows and 10 columns.

```
> names(energy)  # check variable names for use in pivot
 [1] "Timestamp"                          "year"
 [3] "month"                              "weekOfYear"
 [5] "dayOfMonth"                         "dayWeek"
 [7] "timeHour"                           "timeMinute"
 [9] "100_Nevada_Street"                  "104_Maple_St."
[11] "106_Winona_St."                     "Allen_House"
[13] "Alumni_Guest_House/Johnson_House"   "Arboretum_Office"
[15] "Art_Studios"                        "Benton_House"
[17] "Berg_House"                         "Bird_House"
[19] "Boliou_Memorial_Art_Bldg."          "Burton_Hall"
[21] "Cassat_Hall_/_James_Hall"           "Center_for_Mathematics_&_Computing"
[23] "Chaney_House"                       "Clader_House"
[25] "College_Warehouse"                  "Cowling_Gym"
[27] "Dacie_Moses_House"                  "Davis_Hall"
[29] "Douglas_House"                      "Evans_Hall"
[31] "Faculty_Club_/_Annex"               "Farm_House"
[33] "Geffert_House"                      "Generator_Building"
[35] "Goodhue_Hall"                       "Goodsell_Observatory"
[37] "Gould_Memorial_Library"             "Grounds_Building"
[39] "Headley_Cottage"                    "Headley_House"
[41] "Henrickson_House"                   "Henry_House"
[43] "Hill_House"                         "Hilton_House"
[45] "Hoppin_House_(Alumni)"              "Hulings_Hall"
[47] "Hunt_Cottage"                       "Huntington_House"
[49] "James_Hall"                         "Jewett_House"
```

```
[51] "Jones_House"                        "Laird_Hall"
[53] "Laird_Stadium"                      "Language_&_Dining_Center"
[55] "Leighton_Hall"                      "Main_Campus"
[57] "Mudd_Hall_of_Science"               "Music_Hall"
[59] "Musser_Hall"                        "Myers_Hall"
[61] "Nourse_Hall"                        "Nutting_House"
[63] "Olin_Hall_of_Science"               "Page_House_West"
[65] "Parish_House_"                      "Parr_House"
[67] "Pollock_House"                      "Prairie_Warehouse"
[69] "Prentice_House"                     "Rayment_House"
[71] "Recreation_Center"                  "Rice_House"
[73] "Rogers_House"                       "Ryberg_House"
[75] "Sayles-Hill"                        "Scoville_Hall"
[77] "Seccombe_House"                     "Severance_Hall"
[79] "Skinner_Memorial_Chapel"            "Sperry_House"
[81] "Stimson_House"                      "Strong_House"
[83] "Student_Townhouses"                 "Water_Tower"
[85] "Watson_Hall"                        "Weitz_Center_for_Creativity"
[87] "West_Gym"                           "Whittier_House"
[89] "Willis_Memorial_Hall"               "Wilson_House"
> energy_narrow <- energy %>%
+   pivot_longer(
+     cols = `100_Nevada_Street`:Wilson_House,
+     names_to = "building",
+     values_to = "energyKWH"
+   )
```

b.

Use `lubridate` to convert the `Timestamp` to a `date` variable (just date, no time). Then use this new date variable to create a new data set that contains the daily mean and standard deviation of energy use for Laird Hall.

answer:

```
> laird <- energy_narrow %>%
+   filter(building == "Laird_Hall" ) %>%
+   mutate(date_1516 = date(Timestamp)) %>%
+   group_by(date_1516) %>%
+   summarize(
+             mean = mean(energyKWH, na.rm = TRUE),
+             sd =  sd(energyKWH, na.rm = TRUE)
+            ) %>%
+   mutate(month = month(date_1516, label=TRUE))
```

c.

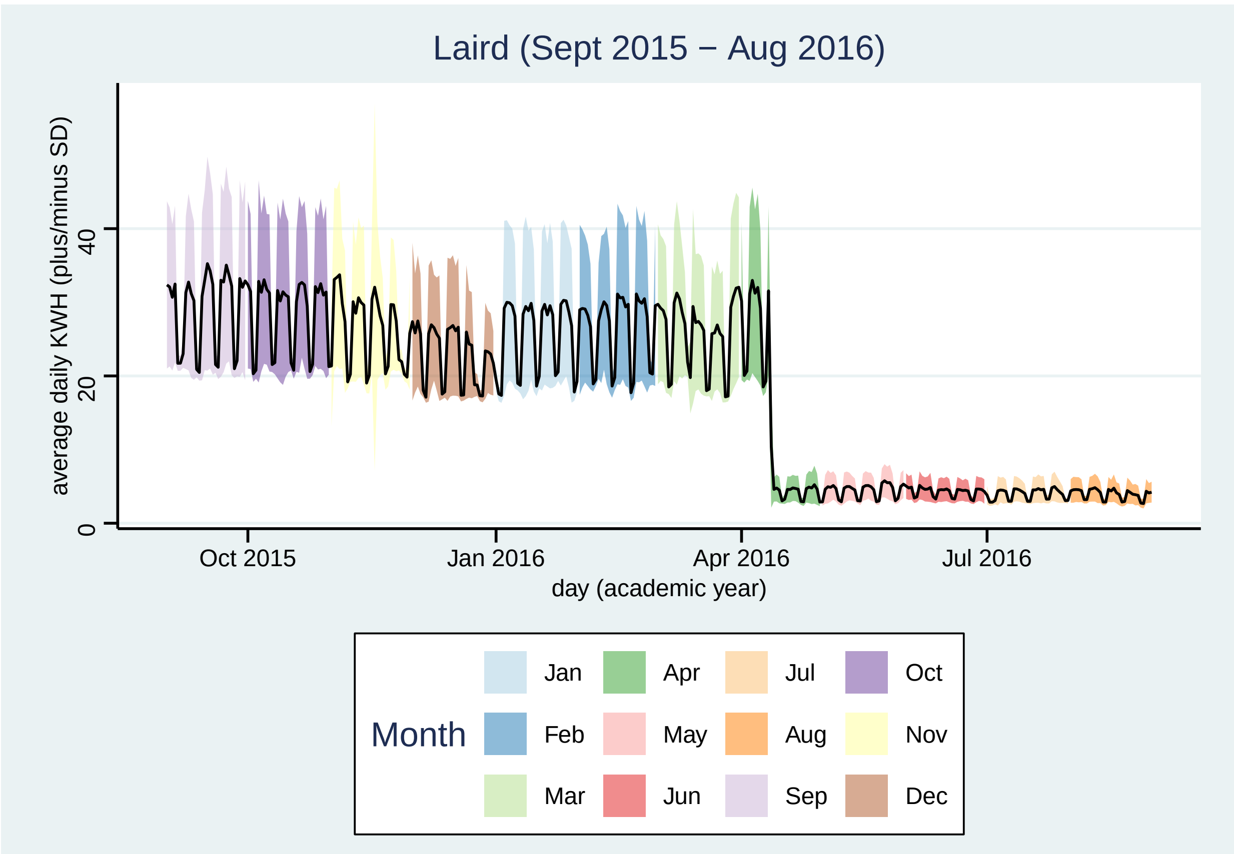
Create a time series (line) graph of mean daily energy use for Laird Hall with the following features:

- a black line geometry to show mean daily energy use by day
- has ribbon geometry layer with limits that are one standard deviation above and below the mean energy use
- in the ribbon geom, use a fill color to denote month with a transparency of 0.5

Describe the trends observed for both mean usage and SD of usage.

answer:

```
> laird %>%
+   ggplot(aes(x = date_1516)) +
+   geom_ribbon(aes(ymin = mean - sd, ymax = mean + sd,
+                   fill = month), alpha = .5) +
+   geom_line(aes(y = mean)) +
+     scale_fill_brewer(palette="Paired", name ="Month") +
+   labs(title ="Laird (Sept 2015 - Aug 2016)",
+        x = "day (academic year)",
+        y = "average daily KWH (plus/minus SD)")
```



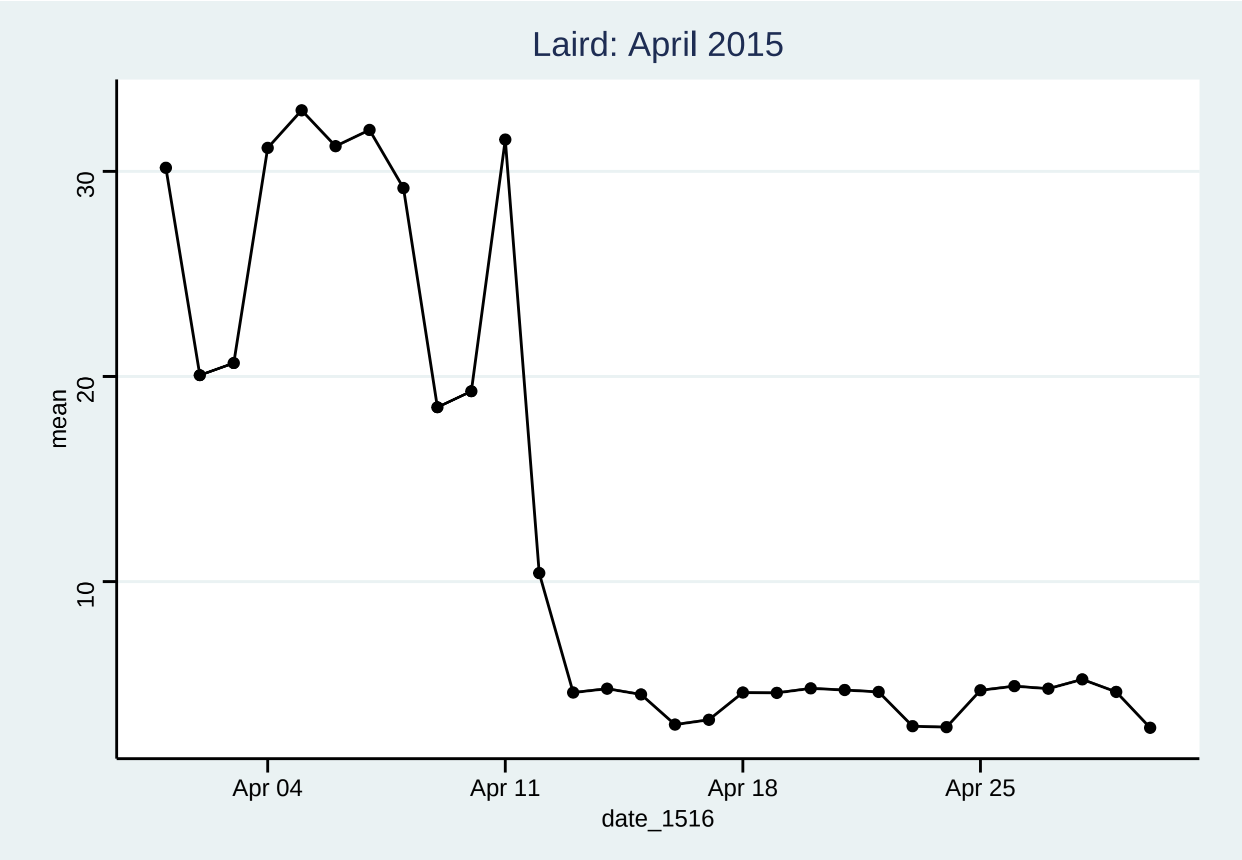
d.

Martha Larson says that the Laird energy meter was adjusted in April 2016 because it was reading too high. Use the drop in daily usage to determine what day in April this adjustment occurred.

answer:

```
> # arrange, ungroup and slice first row, another way to get max
> laird_april <- laird %>%
+   filter(month == "Apr") %>%
+   arrange(date_1516) %>%
+   mutate(gap = abs(mean - lag(mean)))
> laird_april %>%
+   ggplot(aes(x = date_1516, y = mean)) +
+   geom_point() +
+   geom_line() +
```

```
+   ggtitle("Laird: April 2015")
```




```
> mgap <- laird_april %>%
+   arrange(desc(gap)) %>%
+   ungroup() %>%
+   slice(1)
> mgap
# A tibble: 1 x 5
  date_1516   mean    sd month   gap
  <date>     <dbl> <dbl> <ord> <dbl>
1 2016-04-12  10.4  8.30 Apr    21.1
```

The mean daily usage dropped by 21.1 on 2016-04-12.

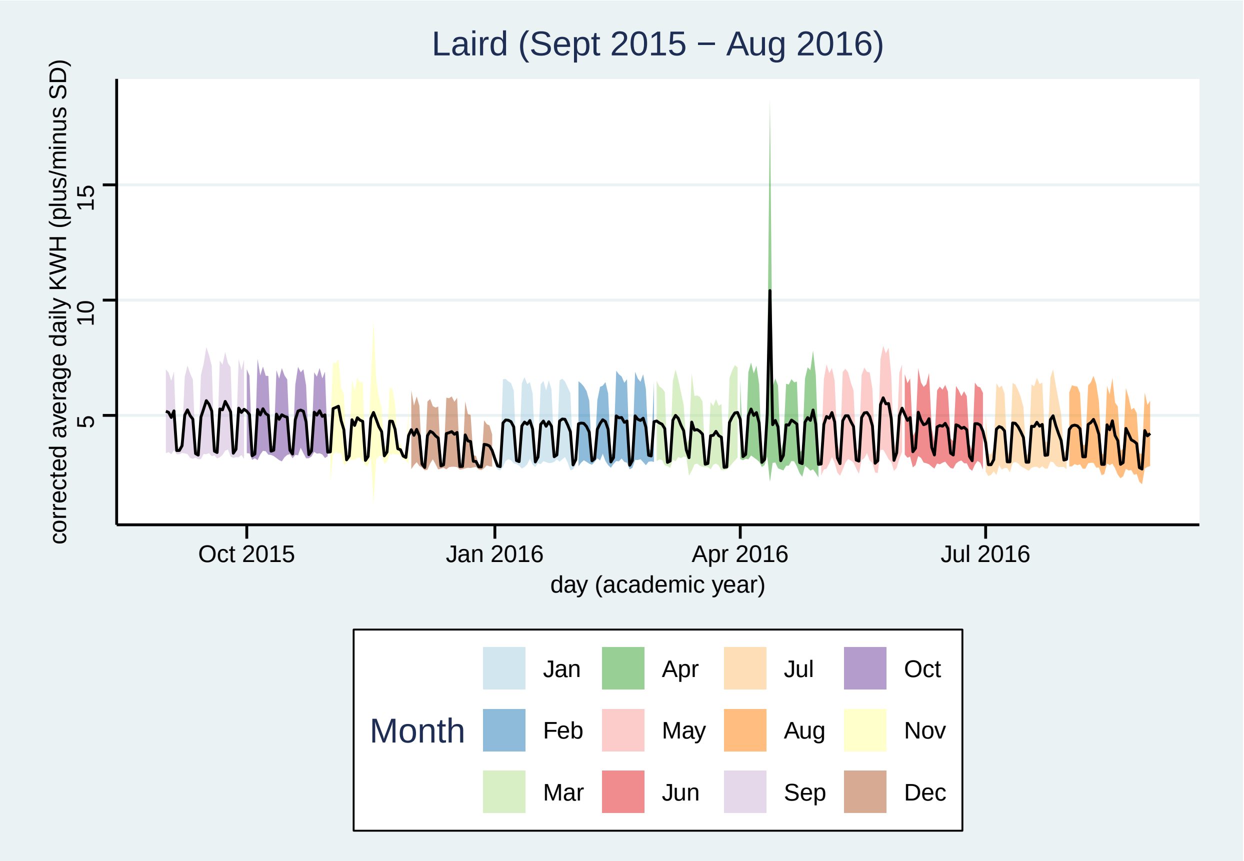
e.

Martha says the higher readings in Laird are due to an incorrect meter that was reading too high. To correct these "too high" readings, we need to multiply them by a factor of 0.16. Do this to get "corrected" average and SD daily readings (for this time period), then replot the graph from part (b). Does this correction to the pre-April correction readings seem to bring the "too high" readings back in line with the post-April correction readings?

answer: We could apply this correction to the raw 15-min KWH, or the mean and SD summaries since the mean/sd of $0.16x$ equals the mean/sd of x multiplied by 0.16. I first pull the timestamp data for the day of the drop from the un-grouped data. We can compare `POSIXct` values just like any number so we can use a logic statement to ID the cases occurring before and after the drop.

```
> laird <- laird %>%
+   mutate(
+     correct = date_1516 < mgap$date_1516,
+     mean_cor = ifelse(correct, .16*mean, mean),
+     sd_cor = ifelse(correct, .16*sd, sd) )
> laird
# A tibble: 366 x 7
   date_1516   mean    sd month correct mean_cor sd_cor
   <date>     <dbl> <dbl> <ord> <lgl>      <dbl>  <dbl>
 1 2015-09-01  32.4 11.4  Sep   TRUE        5.18  1.82
 2 2015-09-02  32.1 10.8  Sep   TRUE        5.13  1.73
 3 2015-09-03  30.7  9.94 Sep   TRUE        4.91  1.59
 4 2015-09-04  32.5 10.7  Sep   TRUE        5.20  1.70
 5 2015-09-05  21.7  1.02 Sep   TRUE        3.48  0.164
 6 2015-09-06  21.7  1.02 Sep   TRUE        3.48  0.163
 7 2015-09-07  23.0  1.92 Sep   TRUE        3.68  0.307
 8 2015-09-08  31.3 10.4  Sep   TRUE        5.01  1.66
 9 2015-09-09  32.8 12    Sep   TRUE        5.24  1.92
10 2015-09-10  31.2 11.5  Sep   TRUE        4.99  1.83
# i 356 more rows
```

```
> laird %>%
+   ggplot(aes(x = date_1516)) +
+   geom_ribbon(aes(ymin = mean_cor - sd_cor, ymax = mean_cor + sd_cor,
+                   fill = month), alpha = .5) +
+   geom_line(aes(y = mean_cor)) +
+     scale_fill_brewer(palette="Paired", name ="Month") +
+   labs(title ="Laird (Sept 2015 - Aug 2016)",
+        x = "day (academic year)",
+        y = "corrected average daily KWH (plus/minus SD)")
```



In our corrected plot we see much more consistent readings across the year wrt both mean and SD. The day of the drop was still above the trend of the correct (and corrected) readings because its average was a mix of correct and uncorrected readings. Since the adjustment occured on that day I didn't make a correction to this mean (and if I had, then it will be unusually low). If I had corrected the raw 15-min. values we would not see this high reading, but to do this I would also have needed the *time* of the correction rather than just the day. (This is doable, just not something you needed to do!)

Problem 4: UN Vote

```
> unvotes <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/
> roll_calls <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/da
> issues <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2
```

```
> # Merge data frames
> merged_data <- unvotes %>%
+   left_join(roll_calls, by = "rcid", multiple = "all") %>%
+   left_join(issues, by = "rcid", multiple = "all") %>%
+   tidyr::drop_na(country, country_code, vote, issue, date) %>%
+   mutate(vote = factor(vote))
Warning in left_join(., issues, by = "rcid", multiple = "all"): Detected an unexpected many-to-many rela
i Row 382 of `x` matches multiple rows in `y`.
i Row 3009 of `y` matches multiple rows in `x`.
i If a many-to-many relationship is expected, set `relationship =
  "many-to-many"` to silence this warning.
```

a. The ‘vote’ column has been converted to a factor. Change the levels of the ‘vote’ column to follow the order: “yes”, “no”, “abstain”.

Answer:

```
> library(forcats)
> merged_data <- merged_data %>%
+   mutate(vote = fct_relevel(vote, c("yes", "no", "abstain")))
>
> levels(merged_data$vote)
[1] "yes"     "no"      "abstain"
```

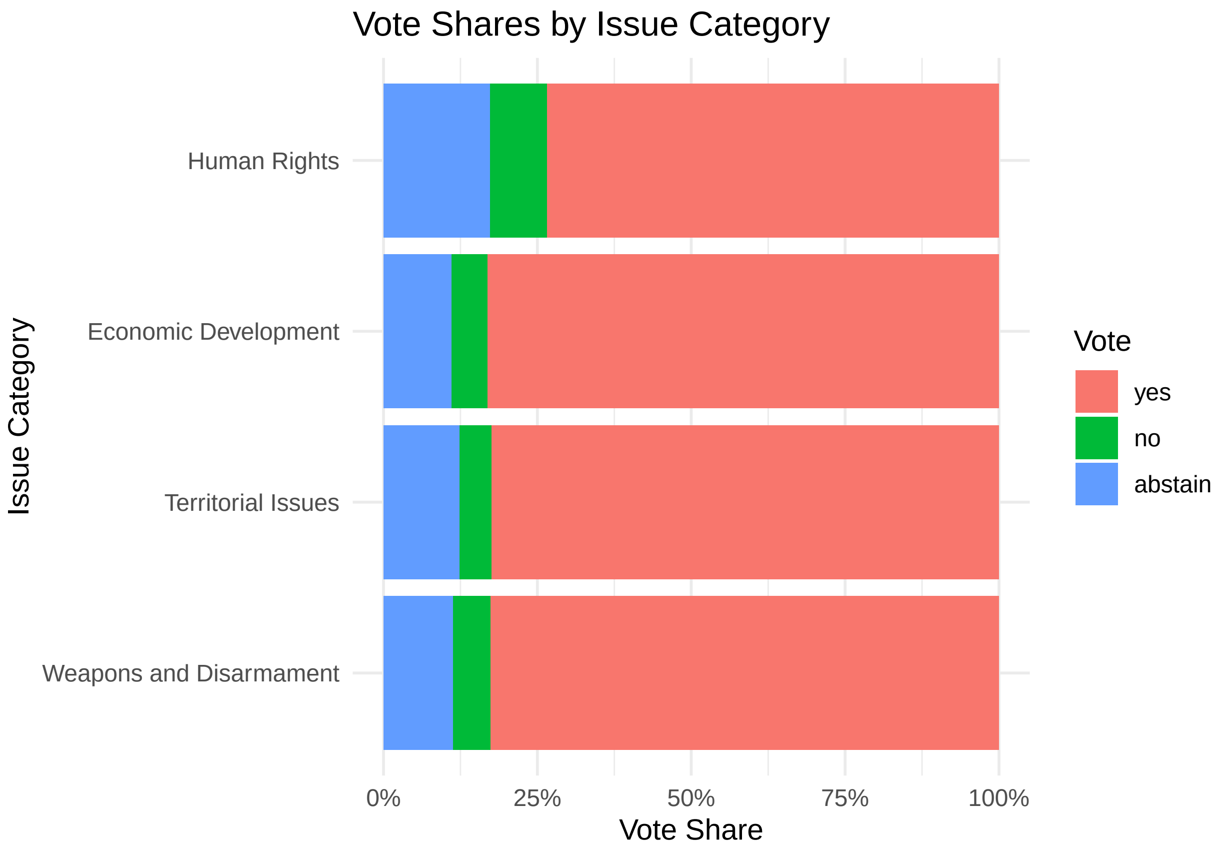
b. Recode the ‘issue’ column into a new ‘issue_category’ column with the following categories: “Territorial Issues”, “Weapons and Disarmament”, “Human Rights”, and “Economic Development”. Create a stacked bar plot showing the vote shares by issue category.

Answer:

```
> merged_data <- merged_data %>%
+   mutate(issue_category = fct_recode(issue,
+                                     "Territorial Issues" = "Palestinian conflict",
+                                     "Territorial Issues" = "Colonialism",
+                                     "Weapons and Disarmament" = "Nuclear weapons and nuclear materia
+                                     "Weapons and Disarmament" = "Arms control and disarmament",
+                                     "Human Rights" = "Human rights",
+                                     "Economic Development" = "Economic development"))
>
> vote_shares_by_issue_category <- merged_data %>%
+   group_by(issue_category, vote) %>%
+   summarise(vote_count = n()) %>%
+   mutate(total_votes = sum(vote_count), vote_share = vote_count / total_votes)
>
> vote_shares_by_issue_category
# A tibble: 12 x 5
# Groups:   issue_category [4]
   issue_category          vote    vote_count total_votes vote_share
   <fct>                   <fct>        <int>       <int>      <dbl>
 1 Weapons and Disarmament yes         249088      301561     0.826
 2 Weapons and Disarmament no           18604      301561     0.0617
 3 Weapons and Disarmament abstain      33869      301561     0.112
 4 Territorial Issues      yes         235642      285745     0.825
 5 Territorial Issues      no           14981      285745     0.0524
 6 Territorial Issues      abstain      35122      285745     0.123
 7 Economic Development    yes          89516      107684     0.831
 8 Economic Development    no            6326      107684     0.0587
 9 Economic Development    abstain      11842      107684     0.110
10 Human Rights            yes         114041      155351     0.734
11 Human Rights            no           14480      155351     0.0932
12 Human Rights            abstain      26830      155351     0.173
```

```
> library(ggplot2)
>
> ggplot(vote_shares_by_issue_category, aes(x = issue_category, y = vote_share, fill = vote)) +
+   geom_bar(stat = "identity", position = "stack") +
+   scale_y_continuous(labels = scales::percent) +
+   labs(title = "Vote Shares by Issue Category",
```

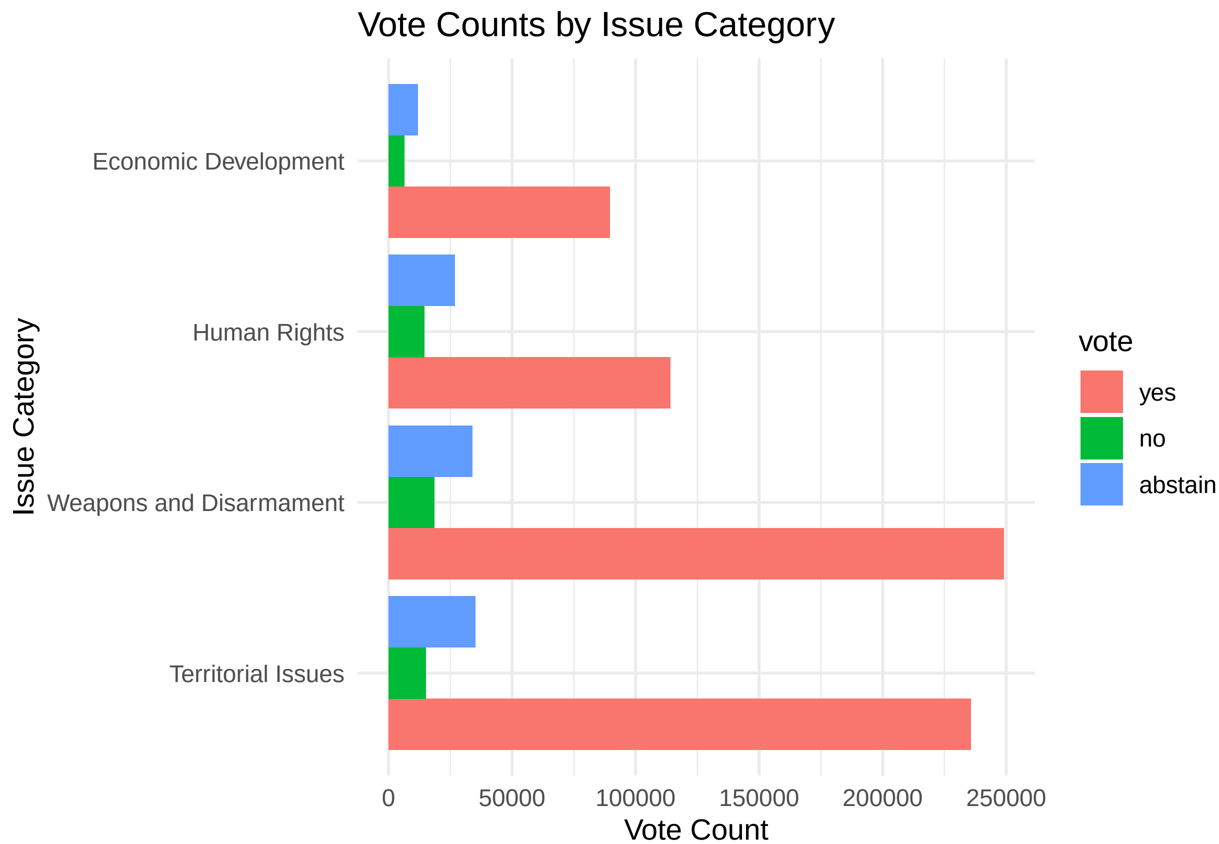
```
+         x = "Issue Category",
+         y = "Vote Share",
+         fill = "Vote") +
+     theme_minimal() + coord_flip()
```



c. Relevel the 'issue_category' variable to follow the order: "Territorial Issues", "Weapons and Disarmament", "Human Rights", "Economic Development". Create a bar plot showing the vote counts by issue category using this new order.

Answer:

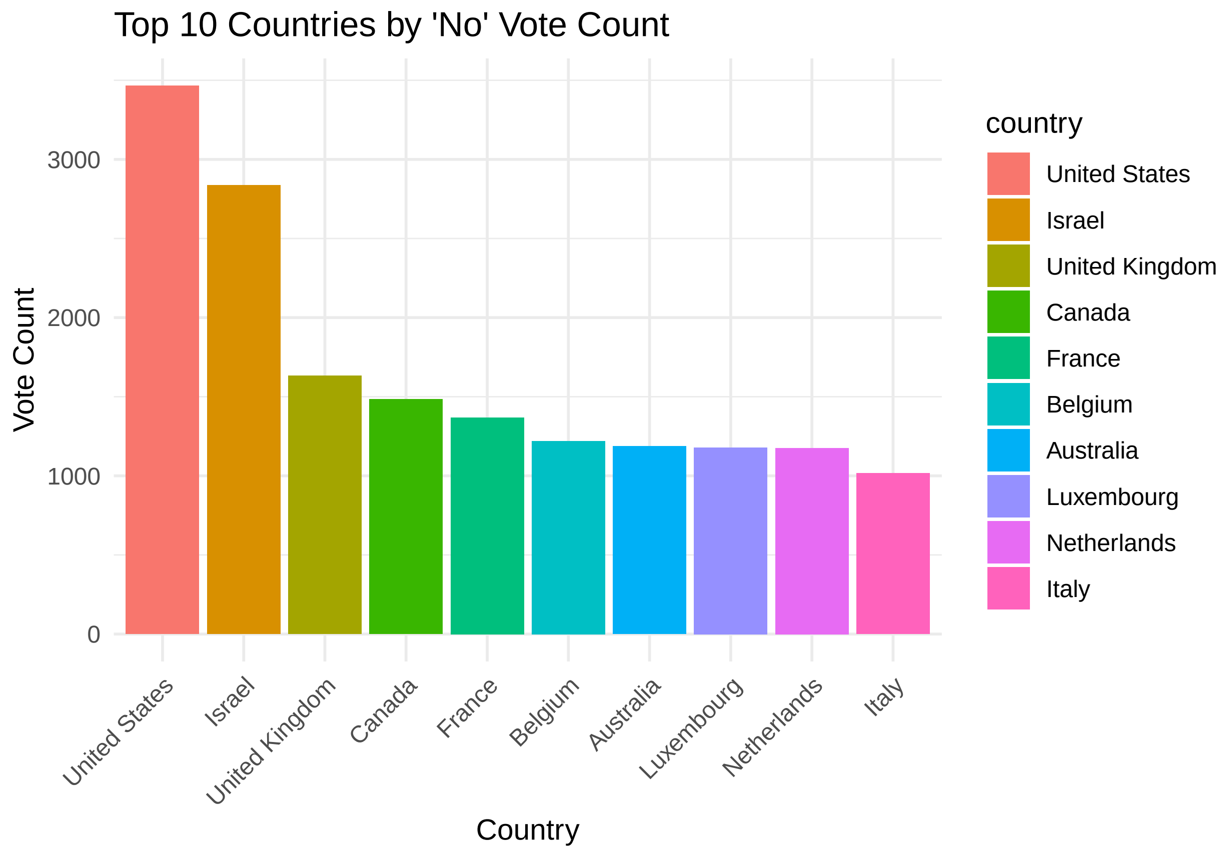
```
> releveled_data <- merged_data %>%
+   mutate(issue_category = fct_relevel(issue_category, "Territorial Issues", "Weapons and Disarmament"
>
> vote_counts_by_issue_category <- releveled_data %>%
+   count(issue_category, vote)
>
> vote_counts_by_issue_category %>%
+   ggplot(aes(x = issue_category, y = n, fill = vote)) +
+   geom_bar(stat = "identity", position = "dodge") +
+   theme_minimal() +
+   labs(title = "Vote Counts by Issue Category", x = "Issue Category", y = "Vote Count") + coord_flip(
```



d. Reorder countries based on the frequency of 'no' votes. Create a bar plot showing the top 10 countries with the highest number of 'no' votes.

Answer:

```
> library(ggplot2)
>
> merged_data %>%
+   filter(vote == "no") %>%
+   count(country) %>%
+   mutate(country = fct_reorder(country, n, .desc = TRUE)) %>%
+   arrange(desc(n)) %>%
+   slice(1:10) %>%
+   ggplot(aes(x = country, y = n, fill = country)) +
+   geom_bar(stat = "identity") +
+   theme_minimal() +
+   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
+   labs(title = "Top 10 Countries by 'No' Vote Count", x = "Country", y = "Vote Count")
```



e. Use `fct_collapse()` to create a new variable called 'region' by grouping countries into the following regions: "Americas", "Europe", "Asia", and "Middle East".

- Americas: "United States", "Canada", "Brazil", "Argentina", "Mexico"
- Europe: "United Kingdom", "France", "Germany", "Italy", "Spain"
- Asia: "China", "Japan", "India", "South Korea", "Russia"
- Middle East: "Iran", "Israel", "Saudi Arabia", "Turkey", "United Arab Emirates"

Answer:

```
> merged_data <- merged_data %>%
+   mutate(region = fct_collapse(country,
+                               Americas = c("United States", "Canada", "Brazil", "Argentina", "Mexic
+                               Europe = c("United Kingdom", "France", "Germany", "Italy", "Spain"),
+                               Asia = c("China", "Japan", "India", "South Korea", "Russia"),
+                               Middle_East = c("Iran", "Israel", "Saudi Arabia", "Turkey", "United A
```