

# STAT 436 Midterm

- This exam lasts from 2:30 - 3:45 on March 23, 2023. There are 7 questions.
- This exam is closed notes and closed computer.
- You may use a 1-page cheat sheet (8.5 x 11in or A4 size). You may use both sides, but the cheat sheet must be handwritten.
- If you need extra space, you may write on the back of the page. Please indicate somewhere that your answer continues.
- The instructors can only answer clarifying questions during the exam.

Question	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total
Score								
Possible	3	3	5	5	5	4	5	30

## Q1

Circle whether the following statements about faceting and compound figures are TRUE or FALSE.

- TRUE FALSE To arrange facets across separate rows, we can use `facet_grid(feature ~ .)`.
- TRUE FALSE To improve readability in a compound figure, we should always align figure baselines.
- TRUE FALSE If `p1`, `p2`, and `p3` are `ggplot2` figures, then we can combine them horizontally (onto a single row) using the code,  

```
library(patchwork)
p1 / p2 / p3
```
- TRUE FALSE Each facet in a figure created with `facet_grid` shows a different subset of rows of the dataset appearing in the initial `ggplot()` call.

## Q2

Circle whether the following statements about data tidying are TRUE or FALSE.

- TRUE FALSE The output of the `mutate` function never changes the number of rows in the input `data.frame`.
- TRUE FALSE If we want to tally the total number of gold medals won by each country in the olympics dataset,

```
olympics <- read_csv("https://uwmadison.box.com/shared/static/rzw8h2x6dp5693gdbpgxaf2koqijo12l.csv")
select(Name, Country, Gold)
head(olympics, 4)
```

```
## # A tibble: 4 x 3
##   Name                Country      Gold
##   <chr>              <chr>      <dbl>
## 1 Lamusi A          People's Republic of China    0
## 2 A G Kruger        United States of America    0
## 3 Jamale Aarrass    France                0
## 4 Abdelhak Aatakni  Morocco              0
```

then it will be sufficient to run,

```
olympics %>%
  count(Country, Gold)
```

- c. TRUE FALSE The dataset below gives country-level population and tuberculosis prevalence,

```
head(table2, 4)
```

```
## # A tibble: 4 x 4
##   country    year type      count
##   <chr>      <dbl> <chr>      <dbl>
## 1 Afghanistan 1999 cases        745
## 2 Afghanistan 1999 population 19987071
## 3 Afghanistan 2000 cases        2666
## 4 Afghanistan 2000 population 20595360
```

If we are interested in computing the number of cases per 10000 people, then it will be sufficient to run,

```
table2 %>%
  pivot_longer(c("type", "count")) %>%
  mutate(10000 * cases / population)
```

- d. TRUE FALSE A tidy dataset must store variables explicitly within a column rather than implicitly within column names.

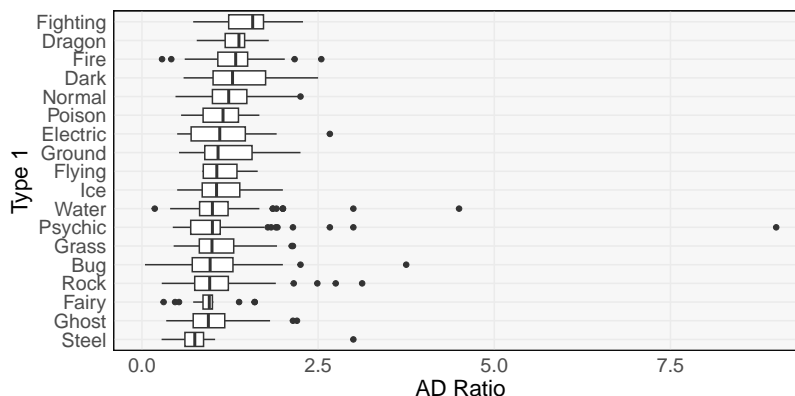
### Q3

We will revisit the Pokemon dataset. For each part below, describe how the data would have to be transformed to support the associated visualization.

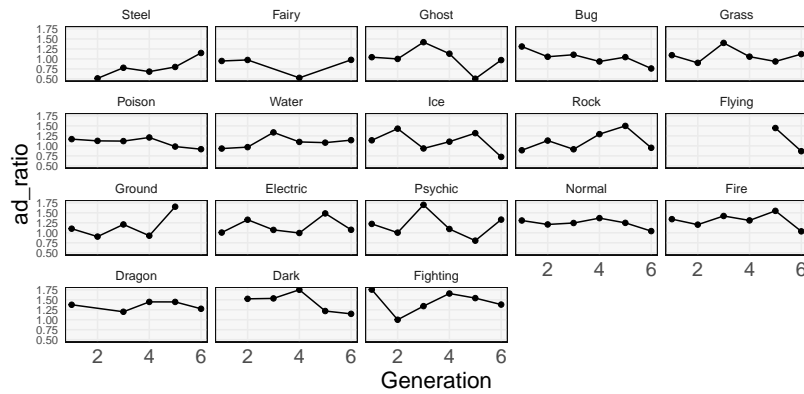
```
pokemon <- read_csv("https://uwmadison.box.com/shared/static/hf5cmx3ew3ch0v6t0c2x56838er1lt2c.csv") %>%
  select(Name, type_1, Generation, Attack, Defense)
head(pokemon, 4)
```

```
## # A tibble: 4 x 5
##   Name                type_1 Generation Attack Defense
##   <chr>              <chr>      <dbl>   <dbl>   <dbl>
## 1 Bulbasaur          Grass           1     49     49
## 2 Ivysaur             Grass           1     62     63
## 3 Venusaur           Grass           1     82     83
## 4 VenusaurMega Venusaur Grass           1    100    123
```

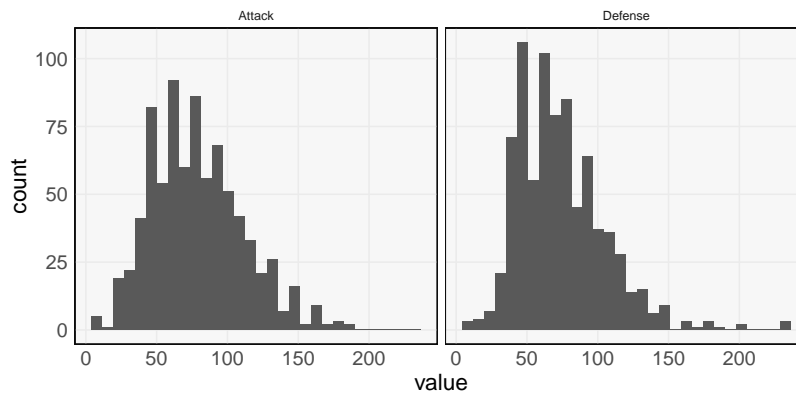
- a. [1.25 points] Derive a new column containing the attack-to-defense ratio for each Pokemon, defined as  $\frac{\text{Attack}}{\text{Defense}}$ , as needed to generate the boxplot below.



- b. [1.25 points] Derive a dataset with the average attack-to-defense ratio for each `type_1` and `Generation` combination, as needed for the line plot below.



- c. [1.25 points] Derive a dataset with the `Attack` and `Defense` variables within a single column which could be used as a faceting variable in the histogram below.



- d. [1.25 points] Provide code for *one* of the three above visualizations. You may assume that the data have been appropriately reshaped.

## Q4

This problem asks you to study the reactive graph of a Shiny app built on the GCFN carbon emissions dataset. The app is built from two datasets, one of which includes time series across countries:

```
carbon_ts <- read_csv("https://raw.githubusercontent.com/krisrs1128/stat679_code/main/activities/week8/carbon_ts.csv")
head(carbon_ts, 4)
```

```
## # A tibble: 4 x 4
##   country   `Country Code`  year emissions
##   <chr>      <chr>        <dbl>    <dbl>
## 1 Afghanistan AFG          1992         0
## 2 Afghanistan AFG          1993         0
## 3 Afghanistan AFG          1994         0
## 4 Afghanistan AFG          1995         0
```

and another which includes features derived from these series:

```
carbon_features <- read_csv("https://raw.githubusercontent.com/krisrs1128/stat679_code/main/activities/week8/carbon_features.csv")
select(country, trend_strength, curvature)
head(carbon_features, 4)
```

```
## # A tibble: 4 x 3
##   country   trend_strength curvature
##   <chr>        <dbl>    <dbl>
## 1 Afghanistan 0.972    0.218
## 2 Albania     0.945   -0.356
## 3 Algeria     0.164    0.232
## 4 Angola      0.931    0.502
```

The app includes two visualizations – a heatmap of the emissions time series and a scatterplot of the derived features. It also has a table showing the original data. The user can select countries to highlight by specifying a range of trend strengths and curvatures in the sliders. See the screenshot below.

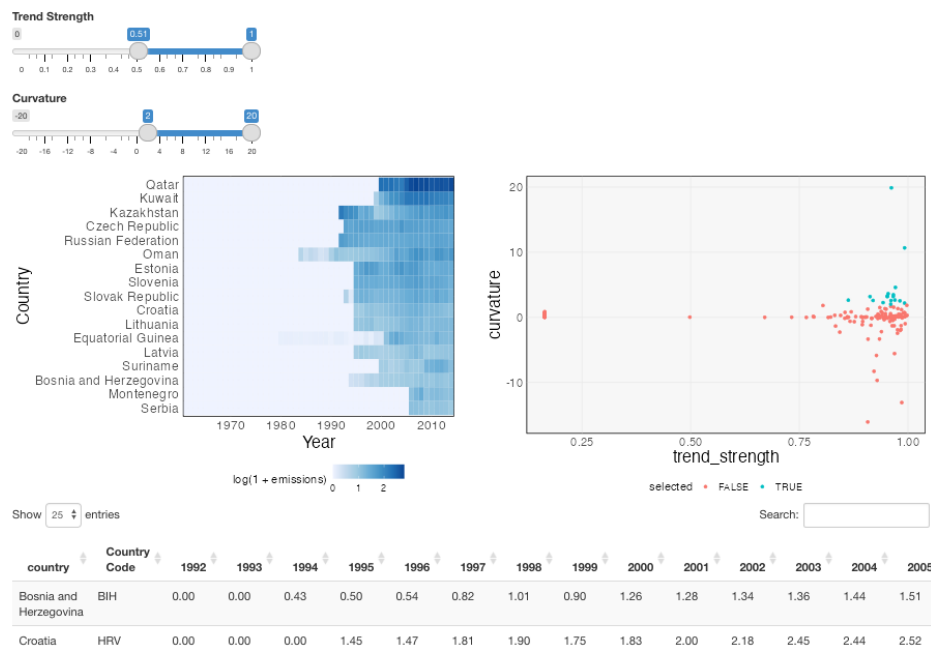


Figure 1: The shiny app to analyze in Problem Q4.

Here is the complete code for the app:

```

ui <- fluidPage(
  sliderInput("trend_strength", "Trend Strength", 0, 1, c(0, 1)),
  sliderInput("curvature", "Curvature", -20, 20, c(-20, 20)),
  fluidRow(
    column(6, plotOutput("heatmap")),
    column(6, plotOutput("scatterplot"))
  ),
  dataTableOutput("table")
)

server <- function(input, output) {
  output$heatmap <- renderPlot({
    cur_countries <- carbon_features %>%
      filter(
        trend_strength >= input$trend_strength[1],
        trend_strength <= input$trend_strength[2],
        curvature >= input$curvature[1],
        curvature <= input$curvature[2]
      ) %>%
      pull(country)

    carbon_ts %>%
      filter(country %in% cur_countries) %>%
      ggplot() +
      geom_tile(aes(year, reorder(country, emissions), fill = log(1 + emissions))) +
      labs(x = "Year", y = "Country") +
      scale_fill_distiller(direction = 1)
  })

  output$scatterplot <- renderPlot({
    cur_countries <- carbon_features %>%
      filter(
        trend_strength >= input$trend_strength[1],
        trend_strength <= input$trend_strength[2],
        curvature >= input$curvature[1],
        curvature <= input$curvature[2]
      ) %>%
      pull(country)

    carbon_features %>%
      mutate(selected = country %in% cur_countries) %>%
      ggplot() +
      geom_point(aes(trend_strength, curvature, col = selected))
  })

  output$table <- renderDataTable({
    cur_countries <- carbon_features %>%
      filter(
        trend_strength >= input$trend_strength[1],
        trend_strength <= input$trend_strength[2],
        curvature >= input$curvature[1],
        curvature <= input$curvature[2]
      ) %>%
      pull(country)

    carbon_ts %>%

```

```

    filter(country %in% cur_countries) %>%
    pivot_wider(names_from = year, values_from = emissions)
  })
}

```

- a. [2 points] Draw the reactive graph associated with this implementation. Be sure to distinguish input, output, and reactive nodes.
  
- b. [3 points] Briefly describe how you would use `reactive({})` to simplify the implementation above. Sketch the updated reactive graph.

## Q5

The `trees` data below includes a subset of data from the New York City Tree Census. Each row corresponds to one tree. For each tree, we have both its location and its health status.

```

trees <- read_sf("https://raw.githubusercontent.com/krisrs1128/stat679_code/main/activities/week7/trees.ge
mutate(health = factor(health, levels = c("Good", "Fair", "Poor")))

head(trees, 4)

```

```

## Simple feature collection with 4 features and 14 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: -74.00477 ymin: 40.71486 xmax: -73.97391 ymax: 40.72647
## Geodetic CRS:   WGS 84
## # A tibble: 4 x 15
##   tree_id block_id species_group health tree_dbh stump_diam curb_loc steward sidewalk address
##   <dbl>   <dbl> <chr>          <fct>   <dbl>   <dbl> <chr>   <chr>   <chr>   <chr>
## 1  365757   102878 Other          Good     12      0 OnCurb  None    NoDamage 229 EAST 2 STREET
## 2  101366   100497 Callery pear  Good      6      0 OnCurb  1or2    NoDamage 290 BROADWAY
## 3  417933   103333 ginkgo       Fair      2      0 OnCurb  1or2    NoDamage 10-17 F D R DRIVE
## 4  103244   103112 pin oak      Good     20      0 OnCurb  None    Damage   121 AVENUE A

```

- a. [1.5 points] Provide code for a scatterplot of the locations of all trees, coloring each in according to its `species_group`. See the figure below for an example result<sup>1</sup>.

---

<sup>1</sup>Though printed in grayscale, the circles would appear in different colors on a computer screen.

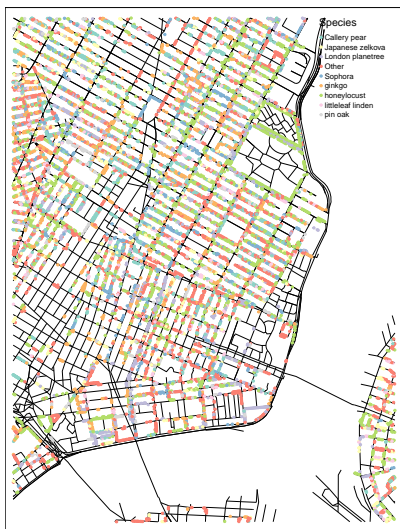


- b. [1.5 points] The `roads` data below includes the trajectories of roads within the same geographic region. Each row corresponds to one street name.

```
roads <- read_sf("https://uwmadison.box.com/shared/static/28y5003s1d0w9nqjnk9xme2n86xazuuj.geojson")
head(roads, 4)
```

```
## Simple feature collection with 4 features and 6 fields
## Geometry type: MULTILINESTRING
## Dimension: XY
## Bounding box: xmin: -73.99366 ymin: 40.7032 xmax: -73.9648 ymax: 40.70745
## Geodetic CRS: WGS 84
## # A tibble: 4 x 7
## STATEFP COUNTYFP LINEARID FULLNAME RTTYP MTFCC
## <chr> <chr> <chr> <chr> <chr> <chr>
## 1 36 047 110422541625 New Dock St M S1400 ((-73.99309 40.70333, -73.99357 40.70446,
## 2 36 047 110422543106 Washington St M S1400 ((-73.98954 40.70386,
## 3 36 047 110422538238 Taylor St M S1400 ((-73.96654 40.7037,
## 4 36 047 110422539438 Bedford Ave M S1400 ((-73.96485 40.70745,
```

Create a version of your visualization from (a) that includes the street map in the background. See the figure below for an example result.



- c. [1 point] What additional layer would you add to your answer from (c) so that trees with different health statuses are placed into different facets? See the figure below for an example result.



- d. [1 point] For both the **trees** and **roads** datasets, state whether they are **raster** format, **vector** format, or **neither**. Briefly justify your choices.



## Q6

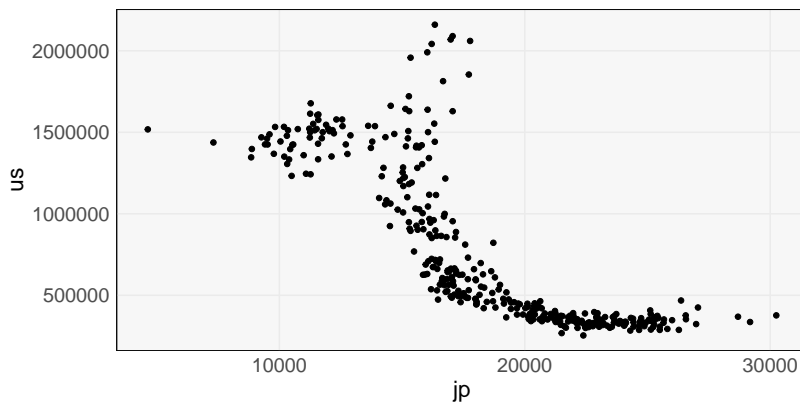
The dataset below includes the number of Spotify streams of the most streamed song of 2017, “Shape of You.” We have created a `tsibble` object from this dataset, filtering to plays from Japan and the US.

```
spotify <- read_csv("https://uwmadison.box.com/shared/static/hvplyr3jy6vbt7s80lqgfx81ai4hdl0q.csv") %>%
  filter(region %in% c("jp", "us")) %>%
  as_tsibble(region, index = date)
```

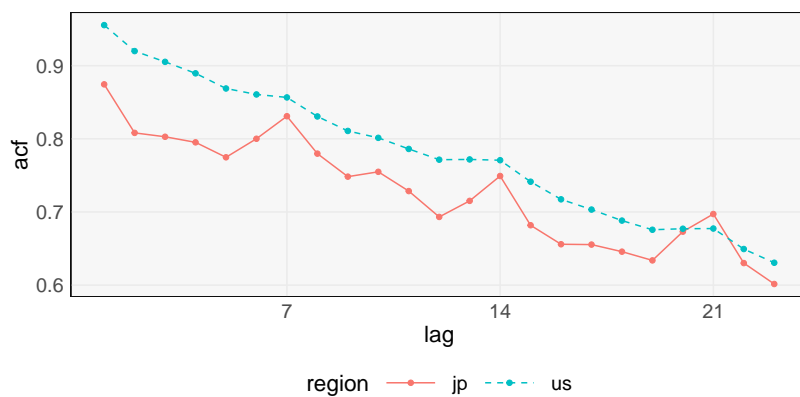
```
head(spotify, 4)
```

```
## # A tsibble: 4 x 5 [1D]
## # Key:      region [1]
##   artist    track_name    region date      streams
##   <chr>      <chr>      <chr> <date>      <dbl>
## 1 Ed Sheeran Shape of You jp      2017-01-06    4639
## 2 Ed Sheeran Shape of You jp      2017-01-07    7313
## 3 Ed Sheeran Shape of You jp      2017-01-08    8851
## 4 Ed Sheeran Shape of You jp      2017-01-09    9827
```

- a. [2 points] Provide code to create the scatterplot below, reshaping the data as necessary.



- b. [2 points] The code below visualizes the associated autocorrelation functions for this dataset. Based on its output comment on both countries' time series trend strength and seasonality.



## Q7

The dataset below describes hourly demand in a bikesharing service. Each row corresponds to a different hour (**hr**), and the variable **count** gives the number of bikes that have been checked out. The dataset additionally includes variables describing characteristics of the day during which the bike was checked out (e.g., **weekday** gives the day of the week) and the weather at the current hour (e.g., **temp** and **hum** are temperature and humidity).

```
bike <- read_csv("https://uwmadison.box.com/shared/static/f16jmkkskylf11hnd5rpslzdja929g2.csv")
head(bike, 4)
```

```
## # A tibble: 4 x 11
##   dteday      season    yr  mnth    hr holiday weekday  temp    hum windspeed count
##   <date>      <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl> <dbl> <dbl>   <dbl> <dbl>
## 1 2011-01-01      1     0     1     0     0       6 0.24 0.81         0    16
## 2 2011-01-01      1     0     1     1     0       6 0.22 0.8         0    40
## 3 2011-01-01      1     0     1     2     0       6 0.22 0.8         0    32
## 4 2011-01-01      1     0     1     3     0       6 0.24 0.75         0    13
```

a. [2.5 points] Propose an interactive visualization to answer the questions:

- How does bike demand vary over any given day?
- In what temperature and humidity ranges is bike demand highest?

Sketch and briefly annotate the layout of your proposed visualization. Describe what types of interactivity would be supported – how would the user provide input, and how would the visualization respond? Ensure that at least one graphical query is included.

b. [2.5 points] Provide code for a Shiny app's **server** component implementing your design from part (a). You may use shorthand for **ggplot2** figures (e.g., write **ggplot** code for “histogram of X” instead of the entire **ggplot** call). However, be as detailed as possible about how your graphical query would be implemented – include comments to ensure this part is clear.

