STAT 442: Assignment 1

DUE: Wednesday January 29 by 11:59pm EST

Part 1 - Reading: Evaluating Effects of Background Stories on Graph Perception [12 marks, 2 marks each]

Read the first 7 pages of the article "Evaluating Effects of Background Stories on Graph Perception" by Ying Zhao et al. and answer the following questions. Each answer should be at MOST one or two sentences, and you may copy directly from the source article without citing.

a) What is the difference between a graph (any type of visualization) and a graph (visualization of a network)?

A graph is an abstract model representing relations among entities, while a graph visualization (e.g., node-link diagram) is a method to visually represent connections among entities.

b) In the context of this article, what is a UBSer and an FBSer?

UBSers: Viewers unaware of the background story.

FBSers: Viewers aware of the background story.

c) What is one of the three hypotheses that the article explores about background stories? (Quoting any one of the three is fine)

H2. Background stories can affect the performance of identifying graph structures.

d) What is a metric proposed for graph readability?

One metric is the number of edge crossings and edge bends, proposed by Purchase et al., to measure the aesthetic readability of graph layouts.

e) What is the name that the article uses for the context of the graph that is not shown in the graph. In other words, the "missing datapoints that are in our head", or the "exformation"?

The article refers to it as "non-displayed contexts," which include invisible knowledge and experience that influence how viewers perceive graphs.

f) What are the three types of graph structures that the participants were asked to identify in one of the experiments?

The three types are:

- high degree nodes
- bridges
- communities

Part 2 - Make a tier list [16 marks]

(16 marks) A client wants a function to make a tier list in R with text instead of images. (Credit to Tierzoo)

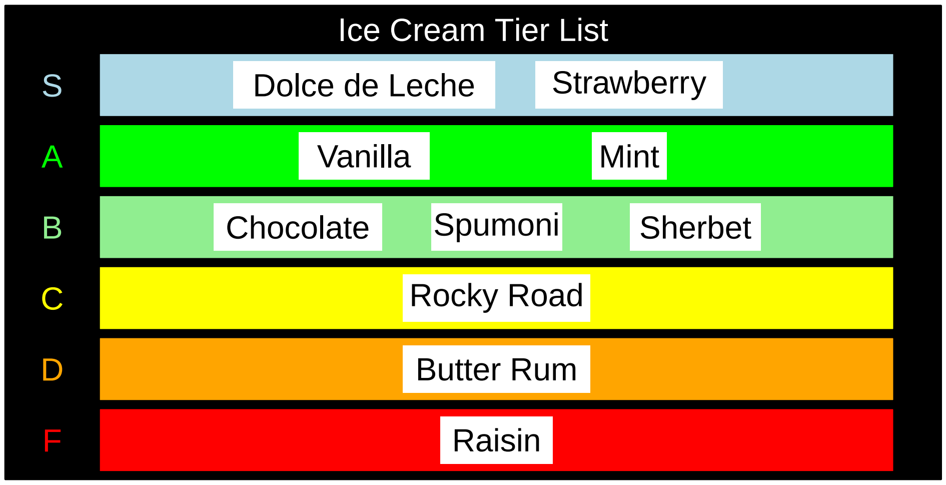
Make a function called `tierlist` that does the following:

- Draws a black background with rectangles for each of the tiers described by `tiernames` and `tiercols`.
- Draws a main title in white text described in `main` above the rectangles.
- Takes in a dataframe `df` that has a `tier` variable and `text` variable.
- For each row of `df`, makes a white box in the appropriate tier and fills it with the black text in `text`.
- Each bar should have room for four boxes, but only filled boxes should be draw

```r
tierlist = function(df, tiernames=c("S","A","B","C","D","F"),
                    tiercols = c("lightblue","green", "lightgreen", "yellow", "orange", "red"),
                    main="My Tier List") {

  Ntiers = length(tiernames)
  # Counts number of tiers to be created

  tier_start_y = y_top * 0.9
  # Starts tiers 10% from the top of the window

  tier_start_x = ((X_right - x_left) * 0.1 + x_left)
  # Starts tiers 10% from the left of the window

  space = (tier_start_y - y_bottom) / Ntiers
  # Space the tiers have by the number to tiers to ensure they are equal

  tier_height = space * 0.9
  # Tier height so that there is a gap between tiers

  rect(x_left, y_bottom, X_right, y_top, col = "black")
  text((X_right + x_left) / 2, y_top * 0.95, labels = main, col = "white")
  # Draws the background and created the title

  for (i in 1:Ntiers) {
    this_df = subset(df, tier == tiernames[i])
    Nboxes = nrow(this_df)

    if (i > 1) {
      tier_start_y = tier_start_y - space
      #changes the y for the second and so on tiers
    }

    rect(tier_start_x, tier_start_y - tier_height, X_right * 0.95, tier_start_y, col = tiercols[i])
    # Draws the tier rectangle


    text((x_left + tier_start_x) / 2, ((tier_start_y - tier_height) + tier_start_y) / 2, tiernames[i],
    # Creates the tier label centered with the tier bar

    for (j in 1:Nboxes) {
      cent = (((X_right * 0.95) - tier_start_x) / (Nboxes + 1))
      x_pos = tier_start_x + cent * j
      y_pos = ((tier_start_y - tier_height) + tier_start_y) / 2
      rect(x_pos - (nchar(this_df$text[j]) * 0.01) * (X_right - x_left),
           y_pos - ((y_top - y_bottom) * 0.05),
           x_pos + (nchar(this_df$text[j]) * 0.01) * (X_right - x_left),
           y_pos + ((y_top - y_bottom) * 0.05),
           col = "white", border = NA)
      # Draws the white box behind the text

      text(x_pos, y_pos, this_df$text[j])
      # Adds the text on top of the white box
    }
  }
}
```

Test the function with this code

```
title = "Ice Cream Tier List"
tier = c("S","S","A","A","B","B","B","C","D","F")
text = c("Dolce de Leche", "Strawberry",
         "Vanilla", "Mint",
         "Chocolate", "Spumoni", "Sherbet",
         "Rocky Road", "Butter Rum", "Raisin")

df = data.frame(tier, text)

  plot.new()
  x_left = 0
  X_right = 10
  y_bottom = 0
  y_top = 10
  plot.window(xlim = c(x_left, X_right), ylim = c(y_bottom, y_top))
  tierlist(df, tiernames=c("S","A","B","C","D","F"),
                    tiercols = c("lightblue","green", "lightgreen", "yellow", "orange", "red"),
                    main=title)
```



Part 3 - Write a Four-way Venn Diagram function [16 marks]

A client wants a function to make a standard four-way Venn diagram, like one of the following:

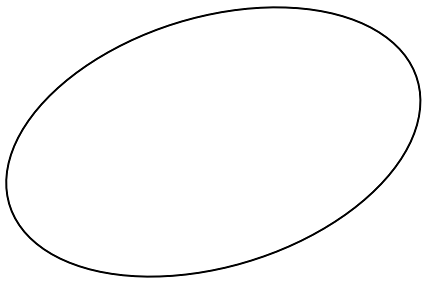
a) (8 marks) Make a function called `venn.oval` using the following code as a starting point. This function should draw an oval with the following parameters:

- Take in a set of center xy coordinates, a long axis xy coordinates, a short axis xy coordinates, and a fill colour.
- Create a convex `polygon()` of 8 sides that resembles an oval, that is symmetric along the long axis (from centerxy to longxy) and the short axis (from centerxy to shortxy)
- The oval should not simply be a diamond, that would be four sides.

```r
venn.oval = function(centerxy = c(0,0), longxy = c(2,-2),
                 shortxy = c(1,1), col="#FFFFFF")
{
# We can use the mathematical function for an ellipse to use in Venn diagram

h <- centerxy[1]  # Center x
k <- centerxy[2]  # Center y
a <- 3  # Semi-major axis (long)
b <- 2  # Semi-minor axis (short)
angle <- 65 *(pi/180) # Angle of the ellipse

# Generate ellipse points using mathematic function of a ellipse
t <- seq(0, 2*pi, length.out = 100)
x <- h + a * cos(t) * cos(angle) - b * sin(t) * sin(angle)
y <- k + a * cos(t) * sin(angle) + b * sin(t) * cos(angle)
polygon(x, y)
  }
```

Test the function with the following code

```r
plot.new()
plot.window(xlim=c(-5,5), ylim=c(-5,5))
venn.oval(centerxy = c(0,0), longxy = c(2,-2),
                 shortxy = c(1,1), col="#FFFFFF")
```



b) (4 marks) Make a four-way venn diagram function called `venn4`. The function should take in the following parameters:

- Call `venn.oval` four times, using the colours Cyan, Magenta, Yellow, and Black, all at 80/255 opacity.
- Use the hex codes for the colours
- Be arranged in such a way that all 16 possible overlaps (include 'none of the four ovals' and 'all of the four ovals') are on the screen.

```r
venn4 = function(centerx = c(-1,0,2,3),
                 centery = c(0,1,1,0),
                  cols = rep("#FFFFFF", 4),
                  angle = c(0,0,0,0 )) {

  angle <- angle * pi / 180  # Convert degrees to radians

  for (i in 1:length(centerx)) {
    h <- centerx[i]  # Center x
    k <- centery[i]  # Center y
    a <- 2  # Semi-major axis (long)
    b <- 1  # Semi-minor axis (short)
    theta <- angle[i]  # Correct angle for this ellipse

    # Generate ellipse points without rotation
    t <- seq(0, 2*pi, length.out = 100)
    x_ellipse <- a * cos(t)  # Standard ellipse x-coordinates
    y_ellipse <- b * sin(t)  # Standard ellipse y-coordinates

    # Apply rotation transformation
    x <- h + x_ellipse * cos(theta) - y_ellipse * sin(theta)
    y <- k + x_ellipse * sin(theta) + y_ellipse * cos(theta)

    # Plot each ellipse
    polygon(x, y, col = adjustcolor(cols[i], alpha = 0.5), border = cols[i])
  }
}
```

Test your function with the following code

```
plot.new()
plot.window(xlim=c(-5,5), ylim=c(-5,5))

venn4(centerx = c(-1,-0.25,0.25,1),
      centery = c(0,1,1,0),
      cols = c("#00FFFF", "#FF00FF", "#FFFF00", "#000000"),
     angle = c(-60,-60,60,60))
```