STAT 479: MIDTERM 1

- This exam lasts from 1:20 - 2:10pm on February 28, 2022. 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 will only be able to answer clarifying questions during the exam. They will be sitting at the back of the room.

Question	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total
Score Possible	2	2	4	4	6	6	6	30

Q1 [2 points]

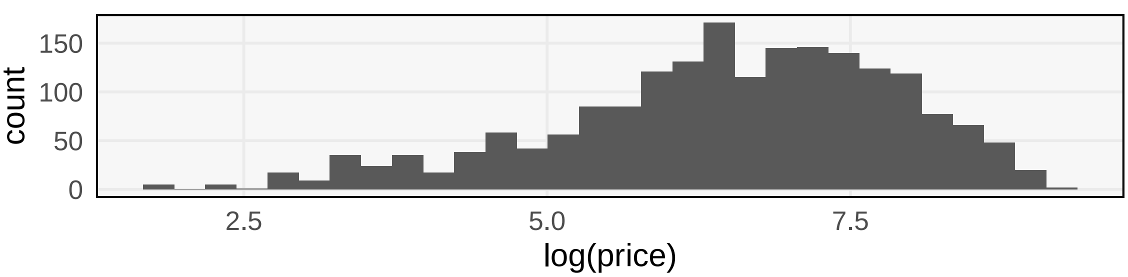
Circle all the true statements for compound and faceted figures.

a. Whenever possible, each panel in a faceted plot should be made to have its own y-axis scale.
b. When using the `patchwork` package, two plots `p1` and `p2` can be placed side-by-side using `p1 / p2`.
c. When using the `patchwork` package, the `plot_layout` function can be used to collect legends.
d. Relative to interactively filtered displays, faceted plots are less taxing on the reader's memory.

Q2 [2 points]

The code below generates a histogram of prices for a subset of items sold at Ikea, a furniture outlet.

```
ikea <- read_csv("https://uwmadison.box.com/shared/static/iat31h1wjg7abhd2889cput7k264bdzd.csv") %>%
  filter(category %in% c("Tables", "Bookcases", "Beds", "Cabinets", "Sofas", "Wardrobes"))
ggplot(ikea) +
  geom_histogram(aes(log(price)))
```



The four plots (i - iv) are generated by faceting this base plot by furniture category (the variable `category`). On the blank lines below, put the number of the plot that matches the corresponding faceting command, or leave the line empty if the corresponding plot does not appear.

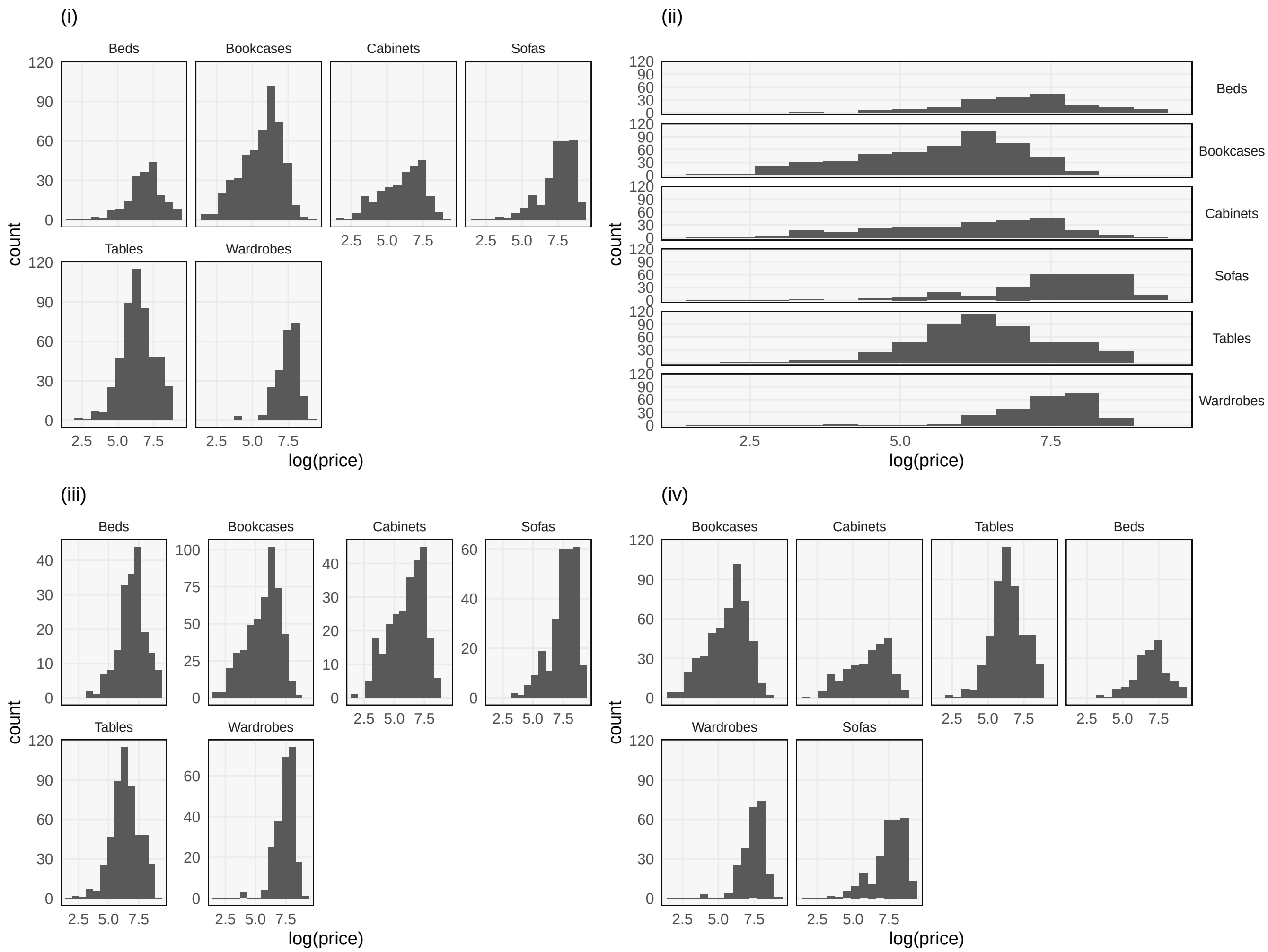
___ a. `facet_grid(category ~ .)`

___ b. `facet_wrap(~ category)`

___ c. `facet_wrap(~ category, scales = "free_y")`

___ d. `facet_wrap(~ category, axes = "separate")`

___ e. `facet_wrap(~ reorder(category, price))`



Q3 [4 points]

Consider the sales data below.

```
##   region quarter sales
## 1      A      Q1     6
## 2      A      Q2     5
## 3      A      Q3     3
## 4      A      Q4     2
## 5      B      Q1     4
## 6      B      Q2     8
## 7      B      Q3     2
## 8      B      Q4     6
```

a. [2 points] Provide code to compute the total sales for each quarter, across both regions. The result should look like the table below.

```
## # A tibble: 4 x 2
##   quarter total
##   <chr>   <dbl>
## 1 Q1         10
## 2 Q2         13
## 3 Q3          5
## 4 Q4          8
```

b. [2 points] Provide code to compute the proportion of each quarter's sales that came from each region. The result should look like the table below.

```
## # A tibble: 8 x 4
##   region quarter sales  prop
##   <chr>  <chr>   <dbl> <dbl>
## 1 A      Q1          6 0.6
## 2 A      Q2          5 0.385
## 3 A      Q3          3 0.6
## 4 A      Q4          2 0.25
## 5 B      Q1          4 0.4
## 6 B      Q2          8 0.615
## 7 B      Q3          2 0.4
## 8 B      Q4          6 0.75
```

Q4 [4 points]

Consider the `reactive()` and `observeEvent()` functions implemented by the `shiny` package. What are common use cases for the two functions? In what ways are the functions different?

Q5 [6 points]

Below, we provide three approaches to visualizing species abundance over time in an antibiotics dataset.

```
antibiotic <- read_csv("https://uwmadison.box.com/shared/static/5jmd9pku62291ek20lioevsw1c588ahx.csv")
antibiotic
```

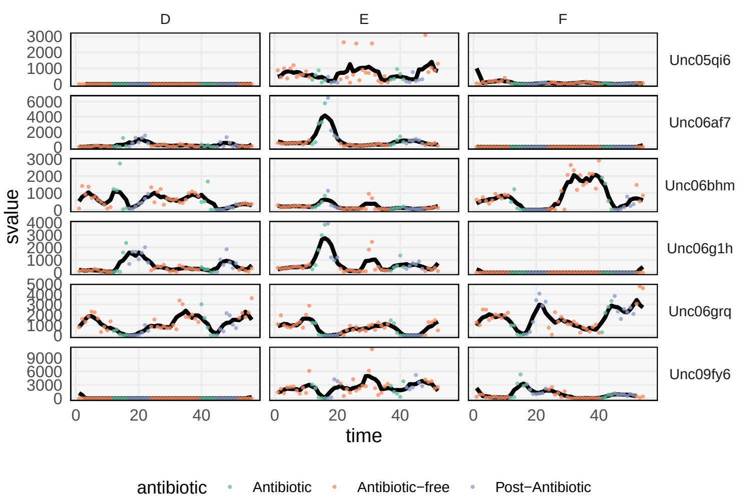
```
## # A tibble: 972 x 7
##    species  sample value ind    time svalue antibiotic
##    <chr>    <chr>  <dbl> <chr> <dbl>  <dbl> <chr>
##  1 Unc05qi6 D1         0 D         1   NA   Antibiotic-free
##  2 Unc05qi6 D2         0 D         2   NA   Antibiotic-free
##  3 Unc05qi6 D3         0 D         3    0   Antibiotic-free
##  4 Unc05qi6 D4         0 D         4    0   Antibiotic-free
##  5 Unc05qi6 D5         0 D         5    0   Antibiotic-free
##  6 Unc05qi6 D6         0 D         6    0.2 Antibiotic-free
##  7 Unc05qi6 D7         0 D         7    0.2 Antibiotic-free
##  8 Unc05qi6 D8         1 D         8    0.2 Antibiotic-free
##  9 Unc05qi6 D9         0 D         9    0.2 Antibiotic-free
## 10 Unc05qi6 D10        0 D        10    0.2 Antibiotic-free
## # ... with 962 more rows
```

For each approach, describe,

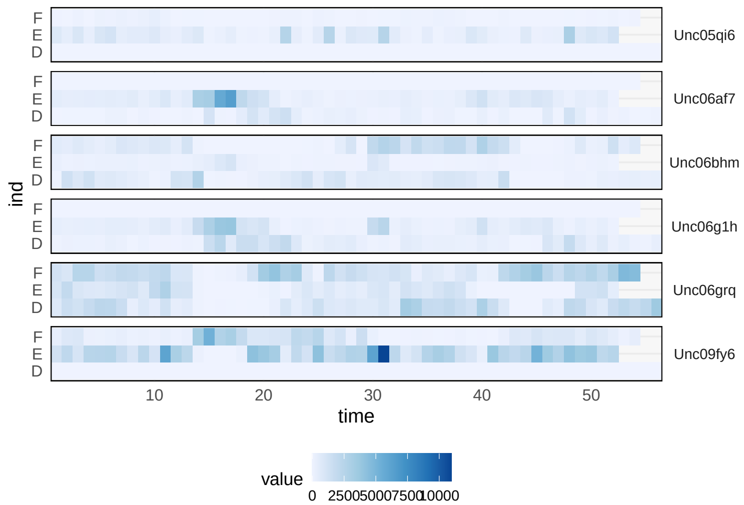
- One type of visual comparison for which the visualization is well-suited.
- One type of visual comparison for which the visualization is poorly-suited.

Make sure to explain your reasoning.

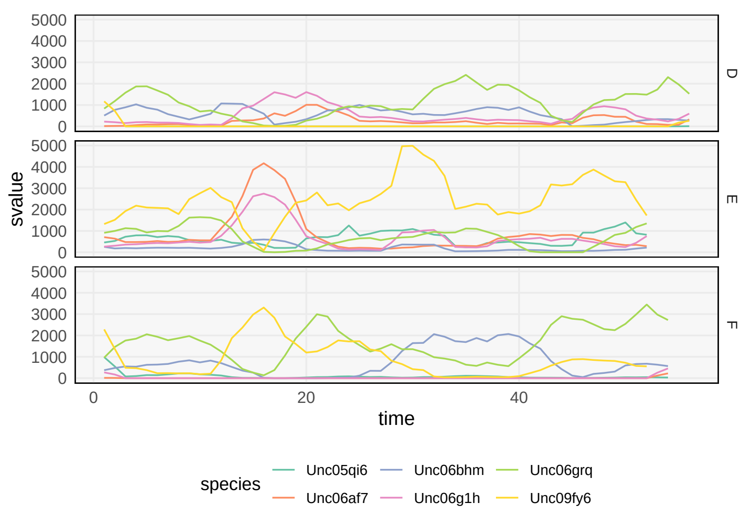
a. [1.5 points] Approach 1



b. [1.5 points] Approach 2



c. [1.5 points] Approach 3



d. [1.5 points] Sketch code that could be used to make one of the three visualizations above.

Q6 [6 points]

The following questions refer to the NYC flights dataset. The first few lines are printed below.

```
library(nycflights13)
flights %>% select(carrier, air_time, distance)
```

```
## # A tibble: 336,776 x 3
##    carrier air_time distance
##    <chr>      <dbl>    <dbl>
##  1 UA           227     1400
##  2 UA           227     1416
##  3 AA           160     1089
##  4 B6           183     1576
##  5 DL           116      762
##  6 UA           150      719
##  7 B6           158     1065
##  8 EV            53      229
##  9 B6           140      944
## 10 AA           138      733
## # ... with 336,766 more rows
```

a. [3 points] Provide code to create a new column giving the average speed of the flight: $\texttt{speed} := \frac{\texttt{distance}}{\texttt{air\_time}}$.

b. [3 points] Is there a large variation in flight speed across carriers? Design and sketch code for a visualization that could be used to answer this question (you may assume the output of (a)).

Q7 [6 points]

The code below sets up a Shiny app for interactively visualizing athlete weight and heights in the 2012 London Olympics. We would like to have an interactive scatterplot of `Weight` vs. `Height, cm` that updates which points (athletes) are highlighted depending on sports have been selected by a dropdown menu. Code for generating the scatterplot is provided in the function `scatterplot`.

```r
library(shiny)
library(tidyverse)
olympics <- read_csv("https://uwmadison.box.com/shared/static/rzw8h2x6dp5693gdbpgxaf2koqijo12l.csv")

#' Scatterplot with highlighted points
#'
#' Assumes a column in df called "selected" saying whether points should be
#' larger / darker
scatterplot <- function(df) {
  ggplot(df) +
    geom_point(
      aes(Weight, `Height, cm`,
          alpha = as.numeric(selected),
          size = as.numeric(selected))
      ) +
    scale_alpha(range = c(0.05, .8)) +
    scale_size(range = c(0.1, 1))
}

ui <- fluidPage(
  selectInput("dropdown", "Select a Sport", choices = unique(olympics$Sport), multiple = TRUE),
  plotOutput("scatterplot")
)

server <- function(input, output) {
  ### fill this in...
}
```

a. [3 points] Provide server code which would allow the scatterplot to update the highlighted athletes depending on the currently selected sports.

b. [3 points] We have been asked to also print a table of the selected athletes. Assume the UI has the form,

```
ui <- fluidPage(
  selectInput("dropdown", "Select a Sport", choices = unique(olympics$Sport), multiple = TRUE),
  plotOutput("scatterplot"),
  dataTableOutput("table")
)
```

Describe changes to your solution to (a) to meet the new requirements. How would you minimize code duplication? Be as specific as possible.