Homework 3

Due 9/17/2024

Classmates/other resources consulted: N.A.

```
library(tidyverse)
library(nycflights13)
# To make the figures show up smaller in the knitted file:
knitr::opts_chunk$set(fig.width=5, fig.height=3)
```

Throughout this assignment, you will use the following data sets. Run the following code chunk to import these data sets:

```
artists <- read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2023/2023-01-17/artists.csv")
```

Question 1 (16 points)

a. **Import the artists data set by running the code chunk above. Output this data set and explain what is in it (you can find more information about the data set at https://github.com/rfordatascience/tidytuesday/tree/master/data/2023/2023-01-17 (https://github.com/rfordatascience/tidytuesday/tree/master/data/2023/2023-01-17))**

```
artists
```

```
## # A tibble: 3,162 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…[1]
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Adélaïde Labi…             14  2013 French             French
## 10 Adélaïde Labi…             15  2016 French             French
## # i 3,152 more rows
## # i abbreviated name: [1]artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

This data set lists several artists and information about their representation in Art textbooks and other venues.

b. **Transform the artists data set to only include rows where the book is Janson.**

```
filter(artists, book == "Janson")
```

```
## # A tibble: 1,219 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 A. R. Penck                 5  1995 German             German
##  2 A. R. Penck                 6  2001 German             German
##  3 Aaron Siskind               3  1986 American           American
##  4 Aaron Siskind               4  1991 American           American
##  5 Aaron Siskind               5  1995 American           American
##  6 Aaron Siskind               6  2001 American           American
##  7 Adolph Gottli…              5  1995 American           American
##  8 Adolph Gottli…              6  2001 American           American
##  9 Adolphe Willi…              7  2007 French             French
## 10 Adolphe Willi…              8  2011 French             French
## # i 1,209 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

c. **Transform the artists data set to only include rows where the edition_number is larger than 10.**

```
filter(artists, edition_number > 10)
```

```
## # A tibble: 1,040 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas              11  2001 American           American
##  2 Aaron Douglas              12  2005 American           American
##  3 Aaron Douglas              13  2009 American           American
##  4 Aaron Douglas              14  2013 American           American
##  5 Aaron Douglas              15  2016 American           American
##  6 Aaron Douglas              16  2020 American           American
##  7 Adélaïde Labi…             14  2013 French             French
##  8 Adélaïde Labi…             15  2016 French             French
##  9 Adélaïde Labi…             16  2020 French             French
## 10 Adolphe Willi…             11  2001 French             French
## # i 1,030 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

d. **Transform the artists data set to only include rows where the artist's nationality is Chinese, Indian, Japanese, Korean, Iranian, or Thai.**

```
filter(artists, artist_nationality %in% c("Chinese", "Indian", "Japanese", "Korean", "Iranian", "T
hai"))
```

```
## # A tibble: 81 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…[1]
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Ando Hiroshige             14  2013 Japanese           Other
##  2 Ando Hiroshige             15  2016 Japanese           Other
##  3 Ando Hiroshige             16  2020 Japanese           Other
##  4 Hokusai                     4  1959 Japanese           Other
##  5 Hokusai                     6  1975 Japanese           Other
##  6 Huang Binhong              11  2001 Chinese            Other
##  7 Kamol Tassana…             11  2001 Thai               Other
##  8 Kano Hogai                 14  2013 Japanese           Other
##  9 Kano Hogai                 15  2016 Japanese           Other
## 10 Kano Hogai                 16  2020 Japanese           Other
## # i 71 more rows
## # i abbreviated name: [1]artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

e. **Transform the artists data set to only include rows where the artist's nationality is *not* Chinese, Indian, Japanese, Korean, Iranian, or Thai. Hint: just make one small change to your code from the previous part.**

```
filter(artists, !artist_nationality %in% c("Chinese", "Indian", "Japanese", "Korean", "Iranian",
"Thai"))
```

```
## # A tibble: 3,081 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Adélaïde Labi…             14  2013 French             French
## 10 Adélaïde Labi…             15  2016 French             French
## # i 3,071 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

f. **Transform the artists data set to only include rows where the artist's nationality is French and the year is between between 2000 and 2010.**

```
filter(artists, artist_nationality == "French", year > 2000, year < 2010)
```

```
## # A tibble: 203 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Adolphe Willi…             11  2001 French             French
##  2 Adolphe Willi…             12  2005 French             French
##  3 Adolphe Willi…             13  2009 French             French
##  4 André Derain               11  2001 French             French
##  5 André Derain               12  2005 French             French
##  6 André Derain               13  2009 French             French
##  7 Anne Louis Gi…             11  2001 French             French
##  8 Anne Louis Gi…             12  2005 French             French
##  9 Anne Louis Gi…             13  2009 French             French
## 10 Antoine Jean …             11  2001 French             French
## # i 193 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

g. **Transform the artists data set to only include rows where the artist's nationality includes "American". For example, you should artists that are American, German-American, Cuban-American, etc. You should not need to type out all of these nationalities in your solution.**

```
filter(artists, grepl("American", artist_nationality))
```

```
## # A tibble: 980 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Albert Bierst…             11  2001 German-American    Other
## 10 Albert Bierst…             12  2005 German-American    Other
## # i 970 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

h. **Transform the artists data set to only show rows where the artist's gender is female or their race is not white**

```
filter(artists, artist_gender == "Female"| !artist_race == "White")
```

```
## # A tibble: 540 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Adélaïde Labi…             14  2013 French             French
## 10 Adélaïde Labi…             15  2016 French             French
## # i 530 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

Question 2 (10 points)

a. **Sort the artists data set by edition_number, from earliest (first edition) to latest.**

```
artists %>% arrange(edition_number)
```

```
## # A tibble: 3,162 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Arthur B. Dav…              1  1926 American           American
##  2 Auguste Renoir              1  1926 French             French
##  3 Chaucer, Prin…              1  1926 British            British
##  4 Claude Monet                1  1926 French             French
##  5 Eugène Delacr…              1  1926 French             French
##  6 George Inness               1  1926 American           American
##  7 Honoré Daumier              1  1926 French             French
##  8 James Abbott …              1  1926 American           American
##  9 Jean Auguste …              1  1926 French             French
## 10 Jean François…              1  1926 French             French
## # i 3,152 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

b. **Sort the artists data set by year, from the most recent year to the oldest year.**

```
artists %>% arrange(desc(edition_number))
```

```
## # A tibble: 3,162 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas              16  2020 American           American
##  2 Adélaïde Labi…             16  2020 French             French
##  3 Albert Bierst…             16  2020 German-American    Other
##  4 Alfred Stiegl…             16  2020 American           American
##  5 Ana Mendieta               16  2020 Cuban-American     Other
##  6 Ando Hiroshige             16  2020 Japanese           Other
##  7 André Derain               16  2020 French             French
##  8 Andreas Gursky             16  2020 German             German
##  9 Andy Warhol                16  2020 American           American
## 10 Angelica Kauf…             16  2020 Swiss              Other
## # i 3,152 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

c. **Write a command to output the columns of artists from artist_name to artist_ethnicity, in order. Hint: You should not need to write them all out.**

```
select(artists, artist_name:artist_ethnicity)
```

```
## # A tibble: 3,162 × 8
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Adélaïde Labi…             14  2013 French             French
## 10 Adélaïde Labi…             15  2016 French             French
## # i 3,152 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 3 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>
```

d. **Write a command to output all variables of artists except for moma_count_to_year and whitney_count_to_year. Hint: You should not need to write them all out.**

```
select(artists, -moma_count_to_year, -whitney_count_to_year)
```

```
## # A tibble: 3,162 × 12
##    artist_name    edition_number  year artist_nationality artist_nationality_o…¹
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Adélaïde Labi…             14  2013 French             French
## 10 Adélaïde Labi…             15  2016 French             French
## # i 3,152 more rows
## # i abbreviated name: ¹artist_nationality_other
## # i 7 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, artist_race_nwi <chr>
```

e. **Write a command to output year and all columns of the artists data set that reference artists, that is, that include the string “artist” in the column name. Hint: you should not need to write out all columns that include the string “artist”.**

```
select(artists, year, matches("artist"))
```

```
## # A tibble: 3,162 × 9
##     year artist_name     artist_nationality artist_nationality_o…[1] artist_gender
##    <dbl> <chr>           <chr>              <chr>                    <chr>
##  1  1991 Aaron Douglas   American           American                 Male
##  2  1996 Aaron Douglas   American           American                 Male
##  3  2001 Aaron Douglas   American           American                 Male
##  4  2005 Aaron Douglas   American           American                 Male
##  5  2009 Aaron Douglas   American           American                 Male
##  6  2013 Aaron Douglas   American           American                 Male
##  7  2016 Aaron Douglas   American           American                 Male
##  8  2020 Aaron Douglas   American           American                 Male
##  9  2013 Adélaïde Labil… French             French                   Female
## 10  2016 Adélaïde Labil… French             French                   Female
## # i 3,152 more rows
## # i abbreviated name: [1]artist_nationality_other
## # i 4 more variables: artist_race <chr>, artist_ethnicity <chr>,
## #   artist_unique_id <dbl>, artist_race_nwi <chr>
```

Question 3 (8 points)

a. (3 points) **Explain why the following three commands all produce the same tibble**

```
artists %>% filter(artist_name == "Lorna Simpson") %>% select(artist_name, year, whitney_count_to_
year)
```

```
## # A tibble: 3 × 3
##   artist_name    year whitney_count_to_year
##   <chr>         <dbl>                 <dbl>
## 1 Lorna Simpson  2001                     3
## 2 Lorna Simpson  2005                     6
## 3 Lorna Simpson  2009                     7
```

```
Lorna_Simpson <- filter(artists, artist_name == "Lorna Simpson")
select(Lorna_Simpson, artist_name, year, whitney_count_to_year)
```

```
## # A tibble: 3 × 3
##   artist_name    year whitney_count_to_year
##   <chr>         <dbl>                 <dbl>
## 1 Lorna Simpson  2001                     3
## 2 Lorna Simpson  2005                     6
## 3 Lorna Simpson  2009                     7
```

```
select(filter(artists, artist_name == "Lorna Simpson"), artist_name, year, whitney_count_to_year)
```

```
## # A tibble: 3 × 3
##   artist_name    year whitney_count_to_year
##   <chr>         <dbl>                 <dbl>
## 1 Lorna Simpson  2001                     3
## 2 Lorna Simpson  2005                     6
## 3 Lorna Simpson  2009                     7
```

The same transformations are being performed in each case, just written in different ways and in a different order.

b. (5 points) **Write a command or series of commands that (1) transforms the artists data set to only keep rows where the year is 1990 or later; (2) adds a new column for the total museum exhibition count, which is the moma_count_to_year plus the whitney_count_to_year; (3) Sorts the data by total number of museum exhibitions, from largest to smallest; and (4) moves the artist, year, total museum exhibitions, moma_count_to_year, and whitney_count_to_year to the left of the data set, displaying all the other columns after them. (Hint: use pipes)**

```
artists %>%
  filter(year >= 1990) %>%
  mutate(total_count_to_year = moma_count_to_year + whitney_count_to_year) %>%
  arrange(desc(total_count_to_year)) %>%
  select(artist_name, year, total_count_to_year, moma_count_to_year, whitney_count_to_year, everyt
hing())
```

```
## # A tibble: 2,194 × 15
##    artist_name      year total_count_to_year moma_count_to_year
##    <chr>           <dbl>               <dbl>              <dbl>
##  1 Jean (Hans) Arp  2020                  64                 64
##  2 Jean (Hans) Arp  2016                  63                 63
##  3 Jean (Hans) Arp  2013                  60                 60
##  4 Jean (Hans) Arp  2009                  59                 59
##  5 Jean (Hans) Arp  2011                  59                 59
##  6 Jean (Hans) Arp  2007                  58                 58
##  7 Jean (Hans) Arp  2005                  57                 57
##  8 Jean (Hans) Arp  2001                  56                 56
##  9 Max Beckmann     2013                  50                 48
## 10 Max Beckmann     2016                  50                 48
## # i 2,184 more rows
## # i 11 more variables: whitney_count_to_year <dbl>, edition_number <dbl>,
## #   artist_nationality <chr>, artist_nationality_other <chr>,
## #   artist_gender <chr>, artist_race <chr>, artist_ethnicity <chr>, book <chr>,
## #   space_ratio_per_page_total <dbl>, artist_unique_id <dbl>,
## #   artist_race_nwi <chr>
```

Question 4 (10 points)

a. **Explain why the comparison x == y in the following code doesn't produce FALSE, even though x and y are different vectors.**

```
x <- c(5,2,9,4)
y <- c(5,2,11,6)
x == y
```

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

Piece wise comparison.

b. **Explain why you get the answer that you do in the following code.**

```
x <- c(TRUE, TRUE, FALSE, TRUE)

y <- c(TRUE, FALSE, TRUE, FALSE)

x | y
```

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

Piecewise OR comparison.

c. **Explain why you get the answer that you do in the following code.**

```
x <- c(TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, FALSE)
sum(x)
```

```
## [1] 5
```

Counts trues.

d. **Explain why the following results return FALSE, and how you should compare these values instead. You can give one explanation for both, you do not need to give a separate explanation for each. (note: every computer is different, and while they both return FALSE on my computer, they may not both return FALSE on your computer. Regardless, discuss why FALSE might show up as an output and what you should do instead).**

```
0.58 - 0.08 == 0.5
```

```
## [1] FALSE
```

```
sqrt(7)^2 == 7
```

```
## [1] FALSE
```

Floating point operations are non-deterministic. Instead check if difference is within a small tolerance level.

e. **This code makes a small tibbles with information about three pets (you don't need to know how this code chunk works):**

```
pets <- tibble(name = c("Sparky", "Lassie", "Patches"),
               weight_last_month = c(30, 55, 12),
               weight_this_month = c(32, 53, 11))
pets
```

```
## # A tibble: 3 × 3
##   name    weight_last_month weight_this_month
##   <chr>               <dbl>             <dbl>
## 1 Sparky                 30                32
## 2 Lassie                 55                53
## 3 Patches                12                11
```

You want to add a new column to this tibble that says which of the two weight values for each pet is smaller, and write the following code:

```
pets %>% mutate(smaller_weight = min(weight_last_month, weight_this_month))
```

```
## # A tibble: 3 × 4
##   name    weight_last_month weight_this_month smaller_weight
##   <chr>               <dbl>             <dbl>          <dbl>
## 1 Sparky                 30                32             11
## 2 Lassie                 55                53             11
## 3 Patches                12                11             11
```

Explain how you should change your code so that the new column correctly has values 30, 53, and 11.
Need to do piecewise min.

```
pets %>% mutate(smaller_weight = pmin(weight_last_month, weight_this_month))
```

```
## # A tibble: 3 × 4
##   name    weight_last_month weight_this_month smaller_weight
##   <chr>               <dbl>             <dbl>          <dbl>
## 1 Sparky                 30                32             30
## 2 Lassie                 55                53             53
## 3 Patches                12                11             11
```

Question 5 (4 points)

a. **Why does NA | TRUE not result in NA, but NA | FALSE results in NA? (Hint: What is the definition of | ? When is A|B TRUE?)**

```
NA | TRUE
```

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

```
NA | FALSE
```

```
## [1] NA
```

is the OR operator which returns TRUE if at least one of the operands is TRUE. Since NA is not known to be true or false, the output is NA.

b. **Why does NA & FALSE not result in NA?**

```
NA & FALSE
```

```
## [1] FALSE
```

& is the AND operator which returns false if either A or B is FALSE.

Question 6 (14 points)

a. (3 points) **Summarize the values in the artists data set: Make a new tibble with a single row, and values for the earliest year, the most recent year, the mean space_ratio_per_page_total, the median space_ratio_per_page_total, and the total number of rows.**

```
artists %>% summarize(
  earliest_year = min(year),
  most_recent_year = max(year),
  mean_space_ratio_per_page_total = mean(space_ratio_per_page_total),
  median_space_ratio_per_page_total = median(space_ratio_per_page_total),
  total_rows = n()
  )
```

```
## # A tibble: 1 × 5
##   earliest_year most_recent_year mean_space_ratio_per_p…¹ median_space_ratio_p…²
##           <dbl>            <dbl>                    <dbl>                  <dbl>
## 1          1926             2020                    0.530                  0.409
## # i abbreviated names: ¹mean_space_ratio_per_page_total,
## #   ²median_space_ratio_per_page_total
## # i 1 more variable: total_rows <int>
```

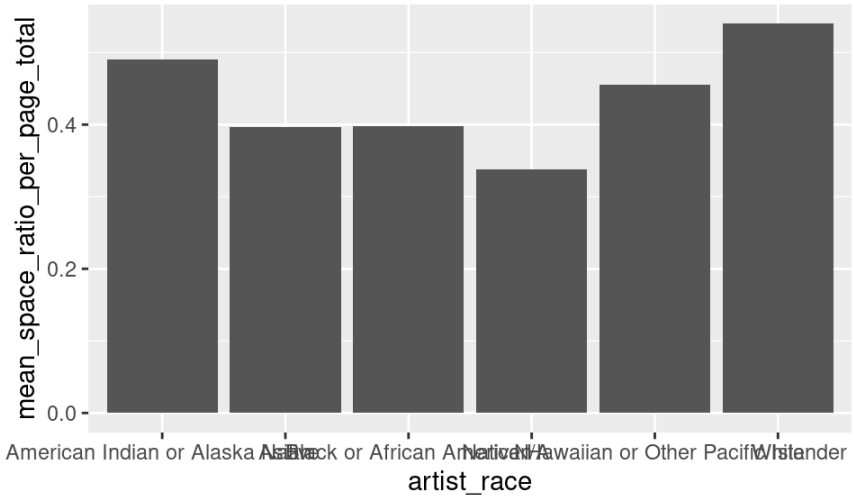
b. (2 points) **Provide the same information as in the previous part, but with a row for each artist_race instead of one row for the whole data set.**

```
race_summary <- artists %>% group_by(artist_race) %>%
  summarize(
  earliest_year = min(year),
  most_recent_year = max(year),
  mean_space_ratio_per_page_total = mean(space_ratio_per_page_total),
  median_space_ratio_per_page_total = median(space_ratio_per_page_total),
  total_rows = n()
  )

race_summary
```

```
## # A tibble: 6 × 6
##   artist_race              earliest_year most_recent_year mean_space_ratio_per…¹
##   <chr>                            <dbl>            <dbl>                  <dbl>
## 1 American Indian or Alas…          1936             2020                  0.491
## 2 Asian                             1926             2020                  0.396
## 3 Black or African Americ…          1986             2020                  0.398
## 4 N/A                               1926             2020                  0.337
## 5 Native Hawaiian or Othe…          2001             2020                  0.456
## 6 White                             1926             2020                  0.540
## # i abbreviated name: ¹mean_space_ratio_per_page_total
## # i 2 more variables: median_space_ratio_per_page_total <dbl>, total_rows <int>
```

c. (3 points) **Using the table you made in part (b), make a bar chart that has a bar for each artist_race, where the height (or length) of the bar is the average space_ratio_per_page_total.**

```
ggplot(race_summary, aes(x = artist_race, y = mean_space_ratio_per_page_total)) + geom_bar(stat =
"identity")
```



d. (3 points) **We see in the previous two parts that one of the artist_race values is N/A. Why does the following command not work to find the artists in the data set where the race is N/A, and how would you change it so it does?**

```
filter(artists, is.na(artist_race))
```

```
## # A tibble: 0 × 14
## # i 14 variables: artist_name <chr>, edition_number <dbl>, year <dbl>,
## #   artist_nationality <chr>, artist_nationality_other <chr>,
## #   artist_gender <chr>, artist_race <chr>, artist_ethnicity <chr>, book <chr>,
## #   space_ratio_per_page_total <dbl>, artist_unique_id <dbl>,
## #   moma_count_to_year <dbl>, whitney_count_to_year <dbl>,
## #   artist_race_nwi <chr>
```

The command doesn't work because N/A is encoded as a string in this example. Filtering by the string "N/A" works.

```
filter(artists, artist_race == "N/A")
```

```
## # A tibble: 29 × 14
##    artist_name    edition_number  year artist_nationality artist_nationality_o…[1]
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Guerrilla Gir…             11  2001 American           American
##  2 Guerrilla Gir…             12  2005 American           American
##  3 Guerrilla Gir…             13  2009 American           American
##  4 N/A1                       13  2009 N/A                Other
##  5 N/A1                       14  2013 N/A                Other
##  6 N/A1                       15  2016 N/A                Other
##  7 N/A1                       16  2020 N/A                Other
##  8 N/A10                       2  1936 N/A                Other
##  9 N/A13                       1  1926 N/A                Other
## 10 N/A13                       2  1936 N/A                Other
## # i 19 more rows
## # i abbreviated name: [1]artist_nationality_other
## # i 9 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>
```

e. (3 points) **Instead of changing your command, change the data set: Add a new column for race: this column should be identical to the artist_race column for the races that are not N/A, but the races that are N/A should instead be something that can be found using is.na().**

```
mutate(artists, artist_race_na = ifelse(grepl("N/A", artist_race), NA, artist_race))
```

```
## # A tibble: 3,162 × 15
##    artist_name    edition_number  year artist_nationality artist_nationality_o…[1]
##    <chr>                   <dbl> <dbl> <chr>              <chr>
##  1 Aaron Douglas               9  1991 American           American
##  2 Aaron Douglas              10  1996 American           American
##  3 Aaron Douglas              11  2001 American           American
##  4 Aaron Douglas              12  2005 American           American
##  5 Aaron Douglas              13  2009 American           American
##  6 Aaron Douglas              14  2013 American           American
##  7 Aaron Douglas              15  2016 American           American
##  8 Aaron Douglas              16  2020 American           American
##  9 Adélaïde Labi…             14  2013 French             French
## 10 Adélaïde Labi…             15  2016 French             French
## # i 3,152 more rows
## # i abbreviated name: [1]artist_nationality_other
## # i 10 more variables: artist_gender <chr>, artist_race <chr>,
## #   artist_ethnicity <chr>, book <chr>, space_ratio_per_page_total <dbl>,
## #   artist_unique_id <dbl>, moma_count_to_year <dbl>,
## #   whitney_count_to_year <dbl>, artist_race_nwi <chr>, artist_race_na <chr>
```

Question 7 (6 points)

a. (2 points) **For the flights data set, make a new column that ranks the flights according to arrival delay, from largest to smallest.**

```
mutate(flights, arr_delay_rank = dense_rank(desc(arr_delay)))
```

```
## # A tibble: 336,776 × 20
##     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      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 12 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>, arr_delay_rank <int>
```

b. (2 points) **Modify your code from the previous part so that your new column ranks the flights according to arrival delay from largest to smallest *within each day of the year*. That is, all flights on January 1st should be ranked from largest to smallest, all flights on January 2nd should be ranked from largest to smallest, etc.**

```
flights %>% group_by(year, month, day) %>%
  mutate(arr_delay_rank = dense_rank(desc(arr_delay)))
```

```
## # A tibble: 336,776 × 20
## # Groups:   year, month, day [365]
##     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      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 12 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>, arr_delay_rank <int>
```

c. (2 points) **Filter the data set you made in the previous part to only contain the flight(s) with the longest arrival delay on each day, that is, flights where the rank of the arrival delay is less than 2 (Your resulting table will have 366 rows because one day had a tie for the most-delayed flight, where both flights received rank 1.5).**

```
flights %>% group_by(year, month, day) %>%
  mutate(arr_delay_rank = dense_rank(desc(arr_delay))) %>%
  filter(arr_delay_rank < 2)
```

```
## # A tibble: 366 × 20
## # Groups:   year, month, day [365]
##     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      848           1835       853     1001           1950
##  2  2013     1     2     1607           1030       337     2003           1355
##  3  2013     1     3     2056           1605       291     2239           1754
##  4  2013     1     4     2123           1635       288     2332           1856
##  5  2013     1     5     1344            817       327     1635           1127
##  6  2013     1     6      943            700       163     1227            932
##  7  2013     1     7     2021           1415       366     2332           1724
##  8  2013     1     8     1307            959       188     1426           1122
##  9  2013     1     9      641            900      1301     1242           1530
## 10  2013     1    10     1121           1635      1126     1239           1810
## # i 356 more rows
## # i 12 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>, arr_delay_rank <int>
```

Question 8 (10 points)

a. (3 points) **In the flights data set, the hour column identifies which hour during the day each flight was scheduled to take off. Make a summary tibble that shows, for each hour, how many flights were scheduled to take off and, of the flights that did take off, what the median and mean departure delays are.**

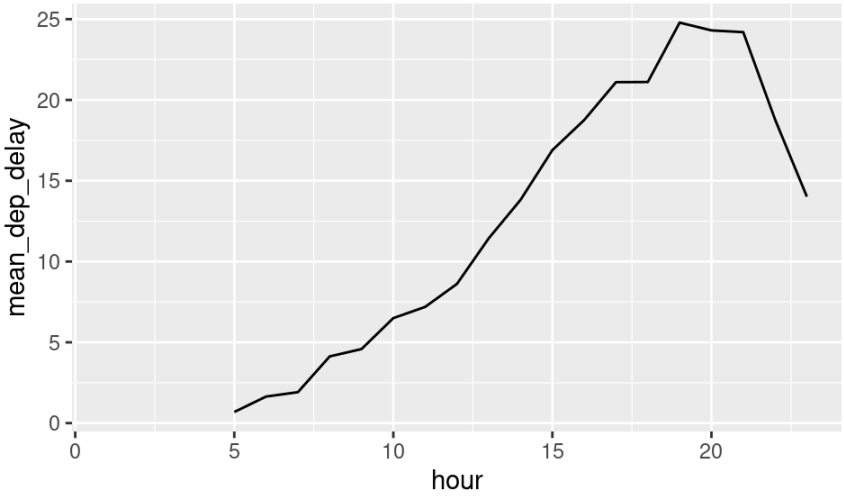
```
hourly_flights <- flights %>%
  group_by(hour) %>%
  summarize(
          count_flights = n(),
          median_dep_delay = median(dep_delay, na.rm = TRUE),
          mean_dep_delay = mean(dep_delay, na.rm = TRUE)
  )

hourly_flights
```

```
## # A tibble: 20 × 4
##     hour count_flights median_dep_delay mean_dep_delay
##    <dbl>         <int>             <dbl>          <dbl>
##  1     1             1                NA        NaN
##  2     5          1953                -3          0.688
##  3     6         25951                -3          1.64
##  4     7         22821                -3          1.91
##  5     8         27242                -3          4.13
##  6     9         20312                -3          4.58
##  7    10         16708                -3          6.50
##  8    11         16033                -3          7.19
##  9    12         18181                -2          8.61
## 10    13         19956                -1         11.4
## 11    14         21706                -1         13.8
## 12    15         23888                 0         16.9
## 13    16         23002                 0         18.8
## 14    17         24426                 1         21.1
## 15    18         21783                 1         21.1
## 16    19         21441                 2         24.8
## 17    20         16739                 2         24.3
## 18    21         10933                 3         24.2
## 19    22          2639                 0         18.8
## 20    23          1061                -1         14.0
```

b. (2 points) **Make a line graph (using a new geom we used this week) that shows, for each hour during the day, what the average departure delay is for that hour.**

```
ggplot(hourly_flights, aes(x = hour, y = mean_dep_delay)) + geom_line()
```

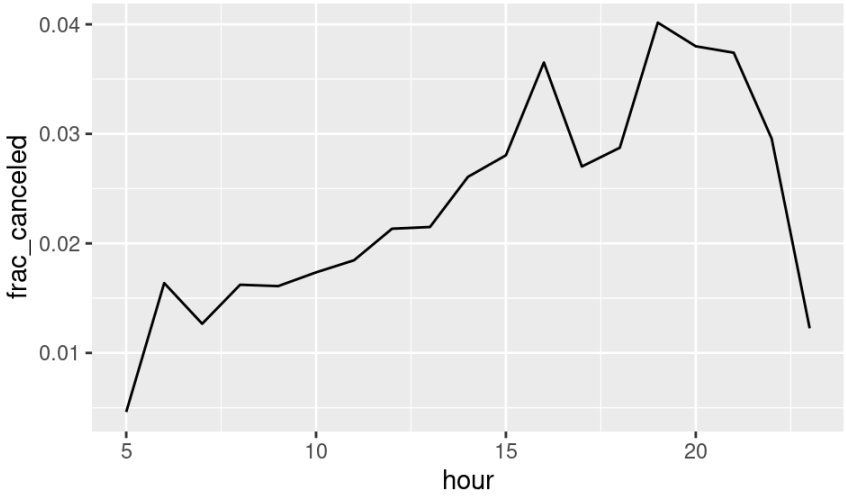
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```



c. (5 points) **Write a command or series of commands that (1) Removes from the flights data set all flights where the hour is 1 (there is only one such flight, and removing it will improve our visualization), (2) Groups the flights data set by hour, (3) Makes a summary tibble showing both the total number of flights and the number of canceled flights each hour, (4) Makes a new column for frac_canceled, which is the number of canceled flights divided by the total number of flights, and (5) Makes a line graph that shows the frac_canceled values for each hour.**

Based on this line graph, what hour of the day would you prefer to fly?

```
summary_flights <- flights %>%
  filter(hour != 1) %>%
  group_by(hour) %>%
  summarize(
    total_flights = n(),
    canceled_flights = sum(is.na(dep_time))
  ) %>%
  mutate(frac_canceled = canceled_flights / total_flights)

ggplot(summary_flights, aes(x = hour, y = frac_canceled)) +
  geom_line()
```



I want to fly at max. 19 o clock.

Question 9 (6 points)

a. **In the U.S., mailing addresses have zipcodes consisting of five digits, then a dash, then four digits. An example might be 91711-4285. Suppose you have a tibble, like the following example, where the first five digits are in a different column than the last four digits.**

```
## # A tibble: 7 × 2
##   Zip   PlusFour
##   <chr> <chr>
## 1 91711 3452
## 2 20322 3009
## 3 93782 8473
## 4 78392 8762
## 5 87639 2563
## 6 47628 5416
## 7 20874 5726
```

Add a new column in this data set consisting of the entire zip code, in the correct format.

```
mutate(zip_codes, zip_combined = paste(Zip, PlusFour, sep = "-"))
```

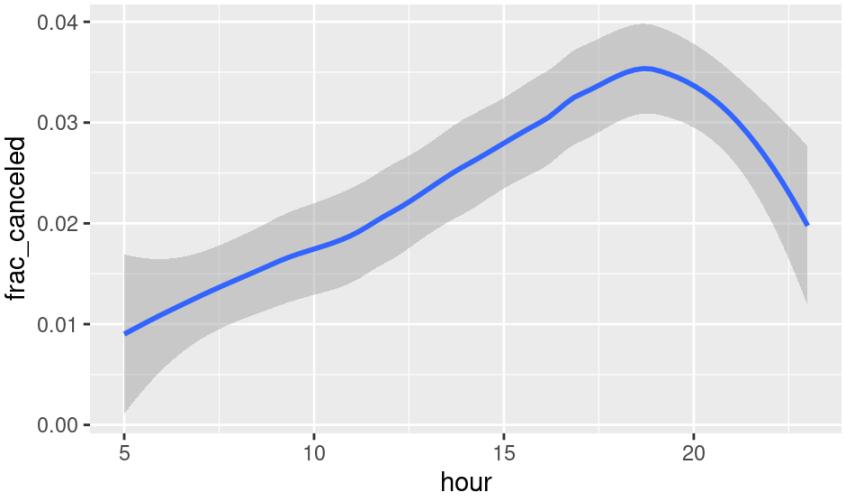
```
## # A tibble: 7 × 3
##   Zip   PlusFour zip_combined
##   <chr> <chr>    <chr>
## 1 91711 3452     91711-3452
## 2 20322 3009     20322-3009
## 3 93782 8473     93782-8473
## 4 78392 8762     78392-8762
## 5 87639 2563     87639-2563
## 6 47628 5416     47628-5416
## 7 20874 5726     20874-5726
```

b. **Explain, in your own words, what geom_smooth() does, and make a ggplot (using any data set you'd like) that uses geom_smooth() and at least one other appropriately-chosen geom.**

geom_smooth is a line plot that also shows the confidence interval for the series. Here are some examples.

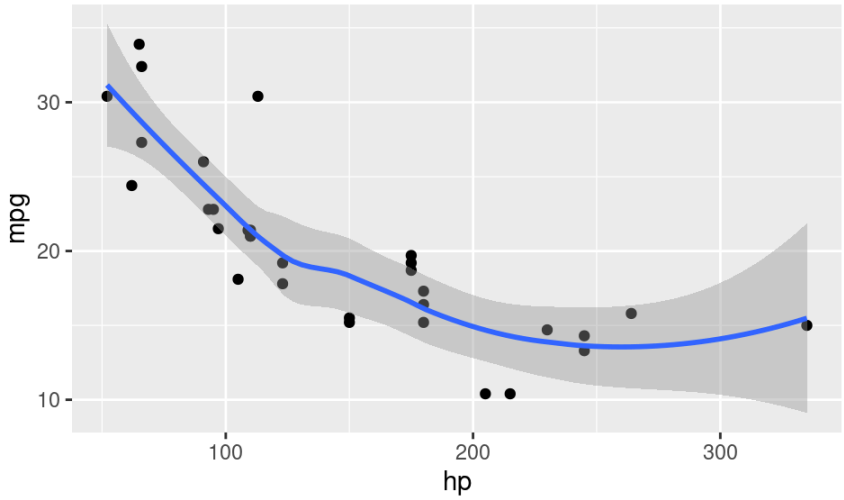
```
ggplot(summary_flights, aes(x = hour, y = frac_canceled)) +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
ggplot(mtcars, aes(x = hp, y = mpg)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Question 10 (3 points)

> **As we discussed in class, removing NA values from a calculation without investigating why those values are NA is a bad idea. Come up with your own example (different from the ones we discussed in class) where removing NA values when performing a calculation could introduce bias in your results.**

Survey respondents of a specific class not answering a specific question that made them uncomfortable. This would bais the results and remove important information.

Question 11 (13 points)

> **Consider the following data set, where each row is an individual who contracted a Delta Variant Covid-19 case in the UK.**

```
covid <- read_csv("https://www.openintro.org/data/csv/simpsons_paradox_covid.csv")
```

```
## Rows: 268166 Columns: 3
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): age_group, vaccine_status, outcome
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
covid
```

```
## # A tibble: 268,166 × 3
##    age_group vaccine_status outcome
##    <chr>     <chr>          <chr>
##  1 under 50  vaccinated     death
##  2 under 50  vaccinated     death
##  3 under 50  vaccinated     death
##  4 under 50  vaccinated     death
##  5 under 50  vaccinated     death
##  6 under 50  vaccinated     death
##  7 under 50  vaccinated     death
##  8 under 50  vaccinated     death
##  9 under 50  vaccinated     death
## 10 under 50  vaccinated     death
## # i 268,156 more rows
```

a. (3 points) **Group the data set by vaccine status and outcome, and make a summary table with four rows (for each combination of vaccinated/unvaccinated and death/survive) and a column showing the number of individuals in each of these four categories. With a calculator (because we haven't learned how to do this yet with code), what fraction of vaccinated individuals died and what fraction of unvaccinated individuals died? Which is higher?**

```
covid %>%
  group_by(vaccine_status, outcome) %>%
  summarize(count = n())
```

```
## `summarise()` has grouped output by 'vaccine_status'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 4 × 3
## # Groups:   vaccine_status [2]
##   vaccine_status outcome   count
##   <chr>          <chr>     <int>
## 1 unvaccinated   death       253
## 2 unvaccinated   survived 150799
## 3 vaccinated     death       481
## 4 vaccinated     survived 116633
```

Fraction of vaccinated individuals who died: 0.0041 (or about 0.41%) Fraction of unvaccinated individuals who died: 0.0017 (or about 0.17%)

b. (2 points) **Now filter the data set to only include individuals under 50, and perform the same steps as part (a). Which group had higher death rates, vaccinated or unvaccinated individuals?**

```
covid %>%
  filter(age_group == "under 50") %>%
  group_by(vaccine_status, outcome) %>%
  summarize(count = n())
```

```
## `summarise()` has grouped output by 'vaccine_status'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 4 × 3
## # Groups:   vaccine_status [2]
##   vaccine_status outcome   count
##   <chr>          <chr>     <int>
## 1 unvaccinated   death        48
## 2 unvaccinated   survived 147564
## 3 vaccinated     death        21
## 4 vaccinated     survived  89786
```

Fraction of vaccinated individuals who died: 0.0002 (or about 0.02%) Fraction of unvaccinated individuals who died: 0.0003 (or about 0.03%)

c. (2 points) **Now filter the data set to only include individuals who are in the age group 50 +, and perform the same steps as parts (a) and (b). Which group had higher death rates, vaccinated or unvaccinated individuals?**

```
covid %>%
  filter(age_group != "under 50") %>%
  group_by(vaccine_status, outcome) %>%
  summarize(count = n())
```

```
## `summarise()` has grouped output by 'vaccine_status'. You can override using
## the `.groups` argument.
```

```
## # A tibble: 4 × 3
## # Groups:   vaccine_status [2]
##   vaccine_status outcome  count
##   <chr>          <chr>    <int>
## 1 unvaccinated   death      205
## 2 unvaccinated   survived  3235
## 3 vaccinated     death      460
## 4 vaccinated     survived 26847
```

Fraction of vaccinated individuals who died: 0.017 (or about 1.7%) Fraction of unvaccinated individuals who died: 0.06 (or about 6%)

d. (3 points) **Explain, in your own words, how your answers to (a), (b), and (c) don't contradict each other. You may find it helpful to look at https://en.wikipedia.org/wiki/Simpson%27s_paradox (https://en.wikipedia.org/wiki/Simpson%27s_paradox)**

The results show Simpson's paradox, which means that a trend looks one way in separate groups but changes when the groups are combined. For example, vaccinated people under 50 had lower death rates, but when we look at everyone together, vaccinated people had higher death rates because more older people, who are more at risk, were vaccinated.

e. (3 points) **What ethical harms can result from failing to group data in relevant ways, or otherwise grouping (or failing to group) data inappropriately?**

If data isn't grouped the right way, it can lead to wrong conclusions, like thinking a treatment is harmful when it's actually helping. This can cause people to make bad decisions, like avoiding vaccines. This is why ethics are always important when working with data.