Problem Set #2

INSERT YOUR NAME HERE

Invalid Date

The aim of this problem set is to give you practice completing data management tasks associated with filtering/isolating observations, sorting observations, and selecting variables. This can be done using the `filter()`, `arrange()`, and `select()` functions from the `tidyverse` package. Filtering/sorting the data can also be done using `base R`'s subsetting operators and `subset()`/`order()` functions (not covered in class but examples provided below).

For the following questions, you'll be asked to complete the same task multiple ways based on the `tidyverse` and `base R` approaches. We want you to understand that there are several ways to complete the same task and we want you to practice completing the same task in different ways.

Question 1: Load and inspect `df_event` dataset

1. In the code chunk below, complete the following:
    - Load the `tidyverse` library
    - Use the `load()` and `url()` functions to download the `df_event` dataframe from the url: `https://github.com/emoriebeck/psc290-data-FQ23/raw/main/05-assignments/02-ps2/ps`
        - Each row in `df_event` represents a recruiting visit

```
rm(list = ls())

library(tidyverse)
#> -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
#> v dplyr     1.1.3     v readr     2.1.4
#> v forcats   1.0.0     v stringr   1.5.0
#> v ggplot2   3.4.3     v tibble    3.2.1
#> v lubridate 1.9.3     v tidyr     1.3.0
#> v purrr     1.0.2
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
```

```
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
load(url("https://github.com/emoriebeck/psc290-data-FQ23/raw/main/05-assignments/02-ps2/ps
```

2. Inspect the `df_event` dataframe:

   - Use `names()` to identify the column names in the dataframe
   - Use `typeof()` to show the data type of the `event_state` column
   - Use `str()` to show the structure of the `med_inc` column
   - Use `table()` to show the categorical values of the `event_type` column

```
names(df_event)
#>  [1] "instnm"              "univ_id"             "instst"
#>  [4] "pid"                 "event_date"          "event_type"
#>  [7] "zip"                 "school_id"           "ipeds_id"
#> [10] "event_state"         "med_inc"             "pop_total"
#> [13] "pct_white_zip"       "pct_black_zip"       "pct_asian_zip"
#> [16] "pct_hispanic_zip"    "pct_amerindian_zip"  "pct_nativehawaii_zip"
#> [19] "pct_tworaces_zip"    "pct_otherrace_zip"   "fr_lunch"
#> [22] "titlei_status_pub"   "total_12"            "school_type_pri"
#> [25] "school_type_pub"     "g12offered"          "g12"
#> [28] "total_students_pub"  "total_students_pri"  "event_name"
#> [31] "event_location_name" "event_datetime_start"
typeof(df_event$event_state)
#> [1] "character"
str(df_event$med_inc)
#>  num [1:18680] 71714 89122 70137 70137 71024 ...
table(df_event$event_type)
#>
#> 2yr college 4yr college       other  private hs   public hs
#>         951         531        2001        3774       11423
```

Question 2: Filtering/isolating observations

Filtering can be done using multiple approaches: `tidyverse`'s `filter()` function, `base R`'s subsetting operators, and `base R`'s `subset()` function. Here is an example of using each method to obtain the total number of recruiting visits to California from the `df_event` dataframe:

```r
# tidyverse using filter()
nrow(filter(df_event, event_state == 'CA'))

# base R using subsetting operators
nrow(df_event[df_event$event_state == 'CA', ])

# base R using subset()
nrow(subset(df_event, event_state == 'CA'))
```

1. Your turn! Count the number of recruiting events that satisfy all the following criteria:

   - By the University of Massachusetts-Amherst (`univ_id`: `166629`)
   - An out-of-state public high school (use `event_type`, `event_state`, and `instst`, which is the visiting university's home state)
   - Average median household income is greater than or equal to $100,000 (`med_inc`)
   - Make sure to drop any `NA` values

   Use `nrow()` to obtain the count. Do the filtering in the 3 ways below. You should get the same answer.

tidyverse using `filter()`:

```r
df_event |>
  dplyr::filter(
    univ_id == 166629,
    event_type == "public hs",
    event_state != instst,
    med_inc >= 100000) |>
  nrow()
#> [1] 264
```

base R using subsetting operators (hint: use `which()` to drop `NA`s):

```r
nrow(df_event[which(df_event$univ_id == 166629 &
           df_event$event_type == "public hs" &
           df_event$event_state != df_event$instst &
           df_event$med_inc >= 100000), ])
#> [1] 264
```

base R using `subset()`:

```r
nrow(subset(df_event,
            univ_id == 166629 &
              event_type == "public hs" &
              event_state != df_event$instst &
              med_inc >= 100000))
#> [1] 264
```

2. Count the number of recruiting events that satisfy all the following criteria:

   - By the University of South Carolina-Columbia (`univ_id`: `218663`) or by the University of Alabama (`univ_id`: `100751`)
   - And either:
     - An in-state 2-year college visit (use `event_type`, `event_state`, and `instst`, which is the visiting university's home state) OR
     - A zip code with population under 10,000 (use `pop_total`)
   - Make sure to drop any `NA` values
   - Note the order of precedence: `&` is higher in priority than `|`

tidyverse using `filter()`:

base R using subsetting operators (hint: use `which()` to drop `NA`s):

base R using `subset()`:

Question 3: Sorting observations

1. Create a new dataframe that contains the events in `df_events` sorted by:

   - Ascending `univ_id`
   - Ascending `event_date`
   - Ascending `event_state`
   - Descending `pct_white_zip`
   - Descending `med_inc`

   Then preview the first 10 rows using `head()`. Do this in 2 ways: using `tidyverse`'s `arrange()` and `base R`'s `order()`.

tidyverse using `arrange()`:

base R using `order()`:

Question 4: Selecting variables

1. Create a new dataframe by selecting the columns `univ_id`, `event_date`, `event_type`, `zip`, and `med_inc` from `df_event`. Use the `names()` function to show what columns (variables) are in the newly created dataframe.

   Do this in 3 ways: using `tidyverse`'s `select()`, `base R`'s subsetting operators, and `base R`'s `subset()`.

tidyverse using `select()`:

base R using subsetting operators:

base R using `subset()`:

Question 5: Additional practice with `df_school_all` dataframe

1. In the code chunk below, complete the following:

   - Use the `load()` and `url()` functions to download the `df_school_all` dataframe from the url: `https://github.com/emoriebeck/psc290-data-FQ23/raw/main/05-assignments/02`
     - Each row in `df_school_all` represents a high school (includes both public and private)
     - There are columns (e.g., `visit_by_100751`) indicating the number of times a university visited that high school
     - The variable `total_visits` identifies the number of visits the high school received from all (16) public research universities in this data collection sample
   - Use `names()` to identify the column names in the dataframe
   - Use `table()` to show the categorical values of the `school_type` column

2. Use the tidyverse functions `arrange()` and `select()` to do the following:

   - Sort `df_school_all` descending by `total_visits`
   - Select the following variables: `name`, `state_code`, `city`, `school_type`,`total_visits`, `med_inc`, `pct_white`, `pct_black`, `pct_hispanic`, `pct_asian`, `pct_amerindian`
     - Note: You can do this in one step by wrapping the `select()` function around `arrange()`, or you can do this in two steps by creating an intermediate dataframe.

   Print the first 10 rows of the final dataframe using `head()`, which represents the top 10 most visited schools by the 16 universities.

3. Building upon the previous question, print the following (select same variables as above):

(A) Top 10 most visited public high schools in California
(B) Top 10 most visited private high schoools in California

Render to pdf and submit problem set

Render to pdf by clicking the "Knit" button near the top of your RStudio window (icon with blue yarn ball) or drop down and select "Knit to PDF"

- Go to the Canvas –> Assignments –> Problem Set 2
- Submit both .qmd and pdf files

- Use this naming convention "lastname_firstname_ps#" for your .Rmd and pdf files (e.g. beck_emorie_ps2.qmd & beck_emorie_ps2.pdf)