Stats_506_PS4

Problem 1

```r
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 becom
```

```r
library(nycflights13)
```

```r
data("airlines")
data("airports")
data("flights")
data("planes")
data("weather")
```

Task (a)

```
departure_delays <- flights %>%
  left_join(airports, by = c("origin" = "faa")) %>%
  group_by(name) %>%
  filter(n() >= 10) %>%
  summarise(mean_dep_delay = mean(dep_delay, na.rm = TRUE),
            median_dep_delay = median(dep_delay, na.rm = TRUE)) %>%
  arrange(desc(mean_dep_delay))

departure_delays
```

```
# A tibble: 3 x 3
  name                 mean_dep_delay median_dep_delay
  <chr>                         <dbl>            <dbl>
1 Newark Liberty Intl            15.1               -1
2 John F Kennedy Intl            12.1               -1
3 La Guardia                     10.3               -3
```

```
arrival_delays <- flights %>%
  left_join(airports, by = c("dest" = "faa")) %>%
  group_by(name) %>%
  filter(n() >= 10) %>%
  summarise(mean_arr_delay = mean(arr_delay, na.rm = TRUE),
            median_arr_delay = median(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(mean_arr_delay))

arrival_delays
```

```
# A tibble: 99 x 3
   name                           mean_arr_delay median_arr_delay
   <chr>                                   <dbl>            <dbl>
 1 Columbia Metropolitan                    41.8               28
 2 Tulsa Intl                               33.7               14
 3 Will Rogers World                        30.6               16
 4 Jackson Hole Airport                     28.1               15
 5 Mc Ghee Tyson                            24.1                2
 6 Dane Co Rgnl Truax Fld                   20.2                1
 7 Richmond Intl                            20.1                1
 8 Akron Canton Regional Airport            19.7                3
```

```
 9 Des Moines Intl                            19.0                0
10 Gerald R Ford Intl                         18.2                1
# i 89 more rows
```

Task (b)

```r
flight_speeds <- flights %>%
  right_join(planes, by = "tailnum") %>%
  group_by(model) %>%
  summarise(avg_speed_mph = mean(speed, na.rm = TRUE),
            num_flights = n()) %>%
  filter(avg_speed_mph == max(avg_speed_mph, na.rm = TRUE))
flight_speeds
```

```
# A tibble: 1 x 3
  model   avg_speed_mph num_flights
  <chr>           <dbl>       <int>
1 DC-9-51           432          91
```

Problem 2

```r
nnmaps <- readr::read_delim("./chicago-nmmaps.csv")
```

```
Rows: 1461 Columns: 11
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): city, season, month
dbl  (7): temp, o3, dewpoint, pm10, yday, month_numeric, year
date (1): date

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.
```

```r
#' Function to find the average temperature of given time
#'
#' @param month a numeric value
#' @param year a numeric value
#' @param data input data, which should be in tibble format
```

```r
#' @celsius a boolean value
#' @average_fn a function object, mean by default
#' @return The z-score for `x`
get_temp<- function(month, year, data, celsius = FALSE, average_fn = mean){

  # Check for invalid year
  if (!is.numeric(year)){
    return("Invalid year. Please provide numeric year")
  }

  # Process for month of type "character"
  if (is.character(month)){

    # Convert month into abbreviation if possible
    month_names <- c("January", "February", "March", "April",
                     "May", "June", "July", "August", "September",
                     "October", "November", "December")
    month_abbrs <- c("Jan", "Feb", "Mar", "Apr",
                     "May", "Jun", "Jul", "Aug",
                     "Sep", "Oct", "Nov", "Dec")
    if (month %in% month_names){
      month <- month_abbrs[match(month, month_names)]
    }

    # Filter the data by month abbreviations
    if (month %in% month_abbrs){
      filtered_data <- data %>%
        filter(month == !!month,
               year == !!year)

    # Case for the input is characters but not a valid month name
    } else {
      return("Invalid month. Please provide a valid month name.")
    }

  # Process for month of type "numeric"
  } else if (is.numeric(month)){

    # Valid numeric month input
    if (month %in% (1:12)){
      filtered_data <- data %>%
```

```r
        filter(month_numeric == !!month,
               year == !!year)

    # Invalid numeric month input. Example: 13
    } else {
      return("Invalid month. Please provide a numeric month (1-12).")
    }

  # Non-numeric or non-character input
  } else {
    return("Invalid month. Please provide a name or a number")
  }

  # No data matching input month and year
  if (nrow(filtered_data) == 0){
    return("No data matched")
  }

  # Calculate the mean using given mean function
  avg_temperature <- filtered_data %>%
    summarize(
      average = average_fn(temp)
    ) %>%
    pull(average)

  # Convert Celsius into Fahrenheit if needed
  if (celsius) {
    avg_temperature <- (avg_temperature-32)*5/9
  }

  return(avg_temperature)
}
```

We can prove our code works by evaluating the following.

Test 1

```r
get_temp("Apr", 1999, data = nnmaps)
```

```
[1] 49.8
```

Test 2

```r
get_temp("Apr", 1999, data = nnmaps, celsius = TRUE)
```

```
[1] 9.888889
```

Test 3

```r
get_temp(10, 1998, data = nnmaps, average_fn = median)
```

```
[1] 55
```

Test 4

```r
get_temp(13, 1998, data = nnmaps)
```

```
[1] "Invalid month. Please provide a numeric month (1-12)."
```

Test 5

```r
get_temp(2, 2005, data = nnmaps)
```

```
[1] "No data matched"
```

Test 6

```r
get_temp("November", 1999, data =nnmaps, celsius = TRUE,
         average_fn = function(x) {
           x %>% sort -> x
           x[2:(length(x) - 1)] %>% mean %>% return
         })
```

```
[1] 7.301587
```

Problem 3

The outputs of this problem are in the folder `SAS_outputs` of the GitHub repository.

Task (a)

```
%let in_path = ~/HW/PS4_p3/dataset;
%let out_path = ~/HW/PS4_p3/output_data;
libname in_lib "&in_path.";
libname out_lib "&out_path.";

/* Load data*/
data recs;
 set in_lib.recs2020_public_v5;
 keep state_name nweight;
run;

/* Aggregate by group */
proc summary data=recs;
  class state_name;
  var nweight;
  output out=out_lib.AllStatesSummary
    sum=nweight_sum;
run;

/* Create a new table to save the macro */
/* Actually, no need to sort here */
proc sort
  data=out_lib.AllStatesSummary
  out=out_lib.AllStatesSummary_sorted;
  by descending nweight_sum;
run;

/* Obtain the total weight*/
data out_lib.AllStatesSummary_sorted;
   set out_lib.AllStatesSummary_sorted;
   if _type_ = 0; /* Row recording the total weight */
   call symputx('my_sum', nweight_sum); /* save it to a macro */
run;

/* Calculate the weight percentage for all states */
```

```
data out_lib.AllStatesPercentage;
   set out_lib.AllStatesSummary;
   percentage = nweight_sum * 100 / &my_sum;
   drop _freq_;
   drop _type_;
   output;
run;

/* Find the state with highest weight */
proc sort
  data=out_lib.AllStatesPercentage
  out=out_lib.AllStatesPercentage;
  by descending nweight_sum;
run;

proc print data=out_lib.AllStatesPercentage(obs=5);
run;

/* Presenting result for michigan */
data out_lib.michigan_percentage;
    set out_lib.AllStatesPercentage;
    where state_name = "Michigan";
    output;
run;

proc print data=out_lib.michigan_percentage;
run;
```

From the output of `SAS`, we can see California has the highest percentage (10.670%) of records.

And of all records, 3.17247% observations correspond to Michigan.

Task (b)

```
data recs;
  set in_lib.recs2020_public_v5;
  keep state_name nweight DOLLAREL;
run;

/* Create a histogram of the total electricity cost for strictly positive costs */
```

```
proc univariate data=recs noprint;
  where DOLLAREL > 0; /* Filter for strictly positive costs */
  var DOLLAREL;
  histogram DOLLAREL;
  ods select Histogram;
run;
```

This generates the histogram we want.

Task (c)

```
data recs;
  set in_lib.recs2020_public_v5;
  keep state_name nweight DOLLAREL;
run;

data recs_log;
  set recs;
  where DOLLAREL > 0; /* Filter for strictly positive costs */
  log_cost = log(DOLLAREL);
run;

/* Create a histogram of the natural log of the total electricity cost */
proc univariate data=recs_log noprint;
  var log_cost;
  histogram log_cost; /* Use the transformed variable */
  ods select Histogram;
run;
```

In this chunk of code, we first carry out the log transformation on the variable `DOLLAREL`, after which we save it to another variable. And finally, we obtain the histogram of transformed data.

Task (d)

```
data out_lib.recs_filtered;
    set in_lib.recs2020_public_v5;
    keep state_name nweight DOLLAREL TOTROOMS PRKGPLC1;
    where PRKGPLC1 ne -2; /* remove N.A. data */
run;
```

```
data out_lib.recs_log;
    set out_lib.recs_filtered;
    where DOLLAREL > 0; /* Filter for strictly positive costs */
    log_cost = log(DOLLAREL);
run;

proc glm data=out_lib.recs_log;
    class PRKGPLC1;
    model log_cost = PRKGPLC1 TOTROOMS;
    weight nweight;
    output out=out_lib.ModelParameters p=Predicted_log;
run;
```

First, we remove the observations with invalid `PRKGPLC1` and `DOLLAREL = 0`.Them, by `proc glm`, we can obtain the weighted linear model.

Task (e)

```
data out_lib.ModelParameters;
    set out_lib.ModelParameters;
    predicted = exp(Predicted_log);
run;

proc sgplot data=out_lib.ModelParameters;
    scatter x=DOLLAREL y=predicted;
    title "Scatterplot of predicted vs DOLLAREL";
run;
```

In this task, we create a new variable to contain the value of predicted values and the the value after `exp` transformation. Then, we call `proc sgplot` to obtain the scatter plot.

Problem 4

Task (b)

First, we load the data into SAS.

```
%let in_path = ~/HW/PS4_p4/dataset;
%let out_path = ~/HW/PS4_p4/output_files;
```

```
libname in_lib "&in_path.";
libname out_lib "&out_path.";

/* Create the "public_data" dataset using PROC SQL */
proc sql;
  create table out_lib.public_data as
  select
    B3 as fin_cf,
    ND2 as thi_dis,
    B7_b as rate_econ,
    GH1 as have_house,
    ppeducat as edu,
    race_5cat as race,
    weight_pop as weight,
    caseID
  from in_lib.public2022;
quit;
```

Task (c)

Now, we need to export the data as a `.csv` file after which we can load this file into `stata`.

```
proc export data=out_lib.public_data
  outfile="&out_path./public_data.csv"
  dbms=csv replace;
  delimiter = ",";
run;
```

Then, we can load it into `stata` using the following command.

```
import delimited using "C:\Users\hzhaoar\Downloads\public_data.csv", clear
```

Task (d)

```
count
```

The output in `stata` is 11,667. In the code-book, the 8 variables that I chose have the property `Missing .: 0/11,667`, which means that there is no missing data. Therefore, data is successfully extracted till this step.

Task (e)

In this task, we will convert the two variables which are in Likert scale to two binary variables.

```
gen worse_fin_cf = (fin_cf == 1) | (fin_cf == 2)
gen higher_thi_dis = (thi_dis == 1) | (thi_dis == 2)
```

Task (f)

We use the following code to tell `Stata` that the data is from a complex sample

```
svyset caseid [pw=weight]
```

Then, we can carry out the logistic model. From the code-book, we can know that all predictors are categorical variables.

```
svy: logit worse_fin_cf i.higher_thi_dis i.rate_econ i.have_house i.edu i.race
```

Here is the result.

```
Survey: Logistic regression

Number of strata =      1                         Number of obs   =      11,667
Number of PSUs   = 11,667                         Population size = 255,114,223
                                                  Design df       =      11,666
                                                  F(14, 11653)    =       68.37
                                                  Prob > F        =      0.0000

-------------------------------------------------------------------------------
               |             Linearized
  worse_fin_cf | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
---------------+---------------------------------------------------------------
1.higher_th~s |    .0148348   .0494406     0.30   0.764    -.0820771    .1117466
               |
     rate_econ |
            2  |   -1.110024   .0487947   -22.75   0.000     -1.20567   -1.014378
            3  |   -1.808187   .0796735   -22.69   0.000    -1.964361   -1.652014
            4  |   -2.473934   .3467046    -7.14   0.000    -3.153533   -1.794335
               |
    have_house |
```

```
             2  |   .0711845   .0563518     1.26   0.207    -.0392744    .1816434
             3  |  -.0215269   .0586194    -0.37   0.713    -.1364308    .0933769
             4  |   -.348212   .0993839    -3.50   0.000    -.5430211   -.1534029
                |
           edu  |
             2  |  -.0789397   .1036429    -0.76   0.446    -.2820972    .1242177
             3  |  -.1077575   .1008401    -1.07   0.285     -.305421    .0899061
             4  |  -.2286458   .0996236    -2.30   0.022    -.4239247   -.0333669
                |
          race  |
             2  |  -.7122881   .0809672    -8.80   0.000    -.8709973   -.5535788
             3  |  -.1663969   .0710839    -2.34   0.019    -.3057333   -.0270605
             4  |  -.4589166   .1259744    -3.64   0.000    -.7058474   -.2119857
             5  |   .0214011   .1646227     0.13   0.897    -.3012869    .3440892
                |
         _cons  |   .4143501   .1040557     3.98   0.000     .2103835    .6183167
------------------------------------------------------------------------------
```

We can observe from this table that the p-value for the variable `higher_thi_dis` is 0.764, which is not significant. So, we will not reject the null-hypothesis, which means there is no strong linear relationship between this predictor and the response.

Therefore, our conclusion is that the respondent's family is better off, the same, or worse off finanicially compared to 12 month's ago can NOT be predicted by thinking that the chance of experiencing a natural disaster or severe weather event will be higher, lower or about the same in 5 years.

Task (g)

In this task, we can save the data modified by `stata` to another `.csv` file.

```
export delimited using "C:\Users\hzhaoar\Downloads\public_data_stata.csv", replace
```

Then, we can load it into `R`.

```
pubilc_data <- read.csv("./public_data_stata.csv")
```

Task (h)

In this task, we choose to refit the model using `R` using package `survey`.

```r
library(survey)
```

```
Loading required package: grid

Loading required package: Matrix


Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loading required package: survival


Attaching package: 'survey'

The following object is masked from 'package:graphics':

    dotchart
```

```r
# Set up the complex survey design
design <- svydesign(id = ~ caseid, weight = ~ weight, data = pubilc_data)
```

```r
# Convert variables into categorical
pubilc_data$higher_thi_dis <- factor(pubilc_data$higher_thi_dis)
pubilc_data$rate_econ <- factor(pubilc_data$rate_econ)
pubilc_data$have_house <- factor(pubilc_data$have_house)
pubilc_data$edu <- factor(pubilc_data$edu)
pubilc_data$race <- factor(pubilc_data$race)
```

```r
# Fit the model
model <- svyglm(worse_fin_cf ~ higher_thi_dis + rate_econ + have_house + edu + race,
                design = design, family = quasibinomial(link = "logit"))
```

```r
# Find the pseudo r-squared
psrsq(model)
```

```
[1] 0.0984103
```

We can see the pseudo r-squared is quite small, which means the model is not so reliable and we should include other variables in this survey.