Homework 6

GONG Kuiyuan

Student ID: 39-246182

Student name: GONG Kuiyuan

Preferred name: Eddie

Answers:

Task 1: I first set the working dictionary and use the function `read.csv` to access and store the data.

```
#setwd("~/Desktop/R for empirical research/HW/")
netflix <- read.csv("netflix_titles.csv")
```

Task 2: I use the verb `select()` together with `contains()` to select the columns I need. Here I selected columns that contain *at*. I also demonstrated the other approach where we can use `where(is.character)` to filter those empty columns. I need to use `head()` to restrict the list shown, otherwise it can't not be rendered.

```
library(pacman)
p_load(tidyverse)
netflix |>
  select(contains("at")) |>
  head(5)
```

```
          date_added rating  duration
1 September 25, 2021  PG-13    90 min
2 September 24, 2021  TV-MA 2 Seasons
3 September 24, 2021  TV-MA  1 Season
4 September 24, 2021  TV-MA  1 Season
5 September 24, 2021  TV-MA 2 Seasons
```

```
#netflix |> select(where(is.character))
```

Task 3: I choose to only print the new variables, otherwise the file can't be rendered due to the big tibble.

```
netflix_2 <- netflix |>
  mutate(
    length = if_else(type == "Movie", parse_number(duration), NA),
    seasons = if_else(type == "TV Show", parse_number(duration), NA)
  ) |>
  relocate(length, seasons, .before = 1)

netflix_2 |>
  select(length, seasons) |>
  head() |>
  print()
```

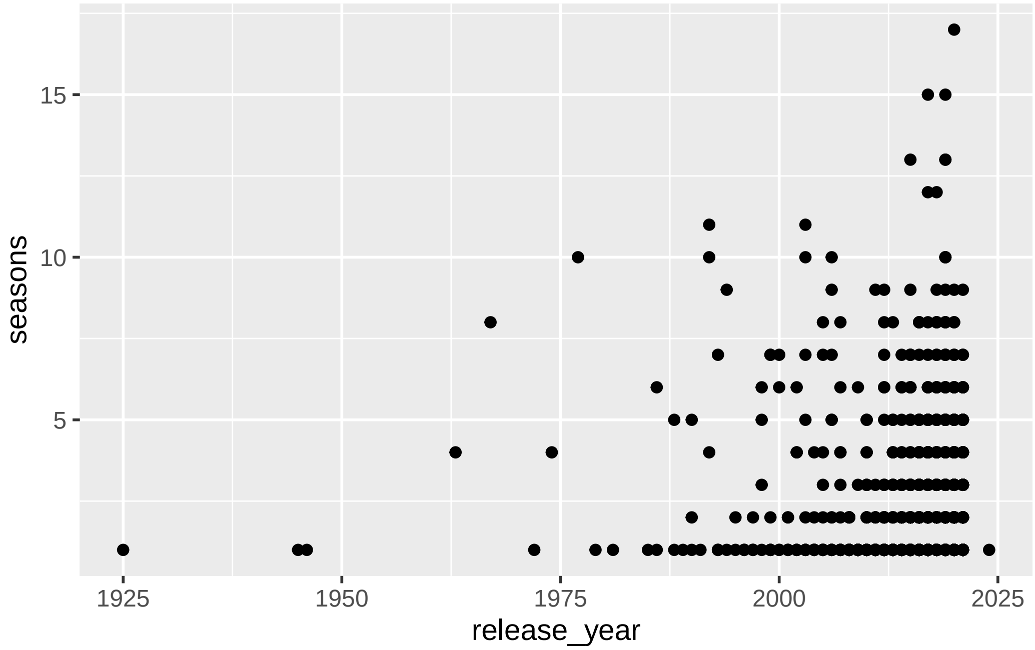
```
  length seasons
1     90      NA
2     NA       2
3     NA       1
4     NA       1
5     NA       2
6     NA       1
```

```
#print(netflix_2)
```

Task 4: Yes, we can observe from the data that there are more TV shows and the number of seasons over the time. A large number of observations are clustered between the years 2000 and 2025. Hypothesis: the number of TV shows and their seasons proliferates as technologies advance over the time.

```
ggplot(
  data = netflix_2,
  mapping = aes(x = release_year, y = seasons)
) +
  geom_point()
```

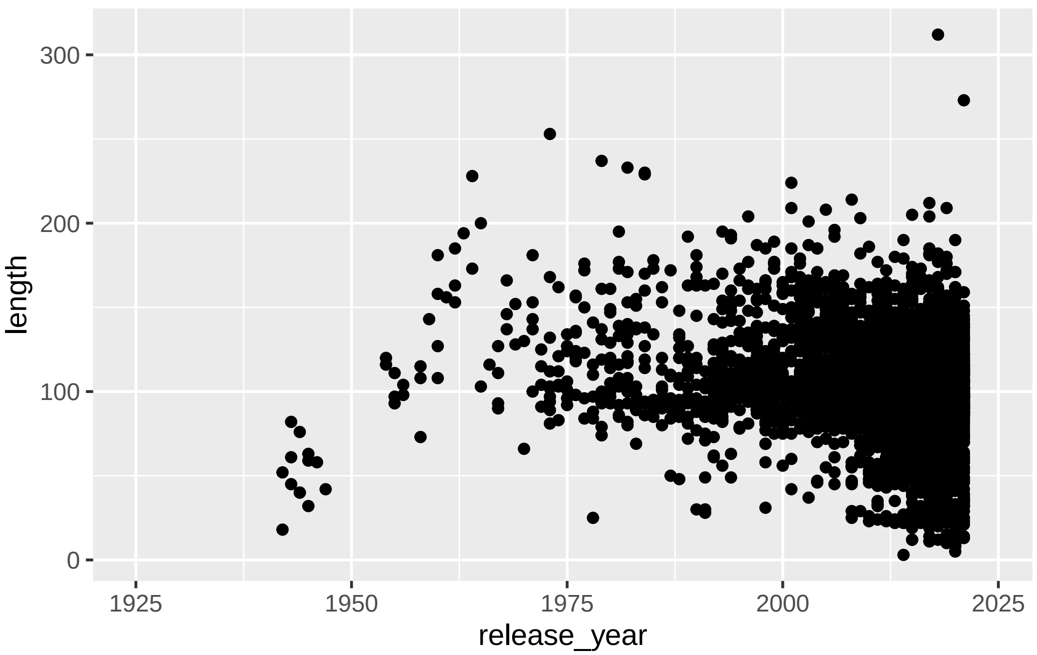
```
Warning: Removed 6132 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Task 5: Yes, we can observe a similar trend with the above plot. The number of movies increase over the time, however, the length of movies is commonly from 0 to 150 minutes, mostly concentrate around 100 minutes. Hypothesis: the number of movies proliferate as technologies advance over the time, however, the length of movies doesn't change a lot but stabilize on a scale.

```
ggplot(
  data = netflix_2,
  mapping = aes(x = release_year, y = length)
) +
  geom_point()
```

```
Warning: Removed 2680 rows containing missing values or values outside the scale range
(`geom_point()`).
```



Task 6: The data is not "tidy" because food, clothing and other categories are column names here. The "tidy" data should be values of a single "category" column, with corresponding amounts in another column. For example, in the column "category" there will be values like "food" and "clothing".

```
library(pacman)
p_load(griffen, griffendata)
print(expenditure_data1)
```

```
# A tibble: 100 x 5
      id  food clothing housing alcohol
   <int> <dbl>    <dbl>   <dbl>   <dbl>
 1     1    95       56     172       8
 2     2   236      183      98      34
 3     3   158      269     521      51
 4     4    69       71     382      37
 5     5    39      171     187      61
 6     6   210      107     666       8
 7     7   319       96     589      17
 8     8   146      189     261      17
 9     9    17      206     184      18
10    10    20      127     115      27
# i 90 more rows
```

Task 7:

```
expenditure_data1 |>
  mutate(total_expenditure = food + clothing + housing + alcohol) |>
  print()
```

```
# A tibble: 100 x 6
      id  food clothing housing alcohol total_expenditure
   <int> <dbl>    <dbl>   <dbl>   <dbl>             <dbl>
 1     1    95       56     172       8               331
 2     2   236      183      98      34               551
 3     3   158      269     521      51               999
 4     4    69       71     382      37               559
 5     5    39      171     187      61               458
 6     6   210      107     666       8               991
 7     7   319       96     589      17              1021
 8     8   146      189     261      17               613
 9     9    17      206     184      18               425
10    10    20      127     115      27               289
# i 90 more rows
```

Task 8:

```
expenditure_pivot <- expenditure_data1 |>
  pivot_longer(
    cols = contains(c("food", "clothing", "housing", "alcohol")),
    names_to = 'category',
    values_to = 'expenditure') |>
  print()
```

```
# A tibble: 400 x 3
      id category expenditure
   <int> <chr>          <dbl>
 1     1 food              95
 2     1 clothing          56
 3     1 housing          172
 4     1 alcohol            8
 5     2 food             236
 6     2 clothing         183
 7     2 housing           98
 8     2 alcohol           34
 9     3 food               158
10     3 clothing           269
# i 390 more rows
```

Task 9:

```
expenditure_pivot <- expenditure_pivot |>
  group_by(id) |>
  summarise(total_expenditrue = sum(expenditure)) |>
  print()
```

```
# A tibble: 100 x 2
      id total_expenditrue
   <int>             <dbl>
 1     1               331
 2     2               551
 3     3               999
 4     4               559
 5     5               458
 6     6               991
 7     7              1021
 8     8               613
 9     9               425
10    10               289
# i 90 more rows
```

Task 10: I prefer to use `group_by()` and `summarise()` to get the target value because it only prints out the total value and mutate other information, which is clearer for me.

Task 11:

```
library(pacman)
p_load(griffen, griffendata)
print(expenditure_data2)
```

```
# A tibble: 100 x 201
      id item1 item2 item3 item4 item5 item6 item7 item8 item9 item10 item11
   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
 1     1    73    15   153   279    22   171   126    90    72     17     73
 2     2   607   116    41    67    37   242    13    14   113     56    312
 3     3   139   335    37   177    84    69    73   337    63    102    197
 4     4    66   612   215   200    60    43    87   693    36     60    113
 5     5   132    12   165    34   178    96   124    67    29    270    115
 6     6   221    59   141    75    36    88    13    57    54     20    970
 7     7    55   286    94    53   215    89   104    24    54     73    104
 8     8   106    21   137    49   240    36   130   244   102    179    187
 9     9   115    45   190    81   231    57   157   208   106    160     48
10    10    29   256   419   156    64   291    31    50    75     89    102
# i 90 more rows
# i 189 more variables: item12 <dbl>, item13 <dbl>, item14 <dbl>, item15 <dbl>,
#   item16 <dbl>, item17 <dbl>, item18 <dbl>, item19 <dbl>, item20 <dbl>,
#   item21 <dbl>, item22 <dbl>, item23 <dbl>, item24 <dbl>, item25 <dbl>,
#   item26 <dbl>, item27 <dbl>, item28 <dbl>, item29 <dbl>, item30 <dbl>,
#   item31 <dbl>, item32 <dbl>, item33 <dbl>, item34 <dbl>, item35 <dbl>,
#   item36 <dbl>, item37 <dbl>, item38 <dbl>, item39 <dbl>, item40 <dbl>, ...
```

Task 12: There are many ways to know the number of columns of a tibble. For example, we can know the size of the tibble when we print it out, or, we can use `ncol()` to report the number for us.

```r
ncol(expenditure_data2)
```

```
[1] 201
```

Task 13: Without pivoting it longer, we need to manually add every term inside of the function in order to get the total expenditures.

```r
#expenditure_data2 |>
  #mutate(
  #total_expenditure = item1 + item2 + item3 + item4...+ item200) |>
  #print()
```

Task 14:

```r
expenditure_data2 |>
  pivot_longer(
    cols = contains("item"),
    names_to = 'category',
    values_to = 'expenditure') |>
  group_by(id) |>
  summarise(total_expenditrue = sum(expenditure)) |>
  print()
```

```
# A tibble: 100 x 2
      id total_expenditrue
   <int>             <dbl>
 1     1             32210
 2     2             31455
 3     3             32024
 4     4             33438
 5     5             32593
 6     6             31102
 7     7             31411
 8     8             31497
 9     9             33735
10    10             32288
# i 90 more rows
```

Task 15: Because the data now is "tidy" in the sense that we can manipulate the data easily using `group_by(id)` and continue to do other operations according to our needs. It transforms multiple expenditure categories (like food…) from separate columns into rows under a single column.

Task 16:

```r
full_cps |>
  filter(state == "Kentacky") |>
  group_by(year, education_category, race) |>
  summarise(
    n = n(),
    wage = mean(wage, na.rm = TRUE)) |>
  filter(n > 10)
```

```
Error in `group_by()`:
! Must group by variables found in `.data`.
x Column `race` is not found.
```

Task 17: First, there is not such variable named "race" in the data, which makes `group_by()` invalid. Second, there is a misspelling of "Kentacky" instead of "Kentucky", which makes the `filter()` invalid. Third, the previous writing style is very difficult for other people to read and debug the code, it may potentially increase the probability of making mistakes.

```r
names(full_cps)
```

```
 [1] "age"                "year"               "wage"
 [4] "hours_lastweek"     "employed"           "education_category"
 [7] "educ_years"         "black"              "white"
[10] "female"             "married"            "single"
[13] "divorced"           "state"              "region"
[16] "sampling_weight"
```

```r
full_cps |>
  filter(state == "Kentacky")
```

```
# A tibble: 0 x 16
# i 16 variables: age <dbl>, year <int>, wage <dbl>, hours_lastweek <dbl>,
#   employed <dbl>, education_category <chr>, educ_years <dbl>, black <dbl>,
#   white <dbl>, female <dbl>, married <dbl>, single <dbl>, divorced <dbl>,
#   state <chr>, region <chr>, sampling_weight <dbl>
```

```r
full_cps |>
  filter(state == "Kentucky")
```

```
# A tibble: 9,159 x 16
     age  year  wage hours_lastweek employed education_category educ_years black
   <dbl> <int> <dbl>          <dbl>    <dbl> <chr>                   <dbl> <dbl>
 1    53  2000 31.2              45        1 college                    18     0
 2    35  2010 17.9              25        1 college                    18     0
 3    28  1984 NA                NA        0 highschool                 NA     0
 4    46  2005  9.86             38        1 somecollege                14     1
 5    27  1980  9.03             50        1 highschool                 10     0
 6    28  1989 12.1              74        1 highschool                 12     0
 7    36  1998 NA                NA        0 highschool                  9     0
 8    57  2008 NA                NA        0 somecollege                14     0
 9    62  2014 NA                20        1 college                    16     0
10    49  1982 25.6              36        1 highschool                 12     0
# i 9,149 more rows
# i 8 more variables: white <dbl>, female <dbl>, married <dbl>, single <dbl>,
#   divorced <dbl>, state <chr>, region <chr>, sampling_weight <dbl>
```