Homework 5

GONG Kuiyuan

Student ID: 39-246182

Student name: GONG Kuiyuan

Preferred name: Eddie

Answers:

Task 1: Note that `dataFrame` here could be any data frame that contains `body_mass_g` and `sex`. This function allows us to pass any data frame as an argument and get the results.

```
library(pacman)
p_load(palmerpenguins)

regression <- function(dataFrame){
  lm (body_mass_g ~ sex, dataFrame)
}

regression(penguins)
```

```
Call:
lm(formula = body_mass_g ~ sex, data = dataFrame)

Coefficients:
(Intercept)      sexmale
     3862.3        683.4
```

Task 2: We first make sure to load all the packages that we need for operations. `names()` could help us to check the variables in the data. `mutate()` is used to generate new variables and I choose to keep the original data and store the change in `penguins2`.

```
library(pacman)
p_load(tidyverse)

library(pacman)
p_load(griffen, griffendata)

names(penguins)
```

```
[1] "species"           "island"            "bill_length_mm"
[4] "bill_depth_mm"     "flipper_length_mm" "body_mass_g"
[7] "sex"               "year"
```

```
penguins2 <- penguins |>
  mutate(new_var = body_mass_g/flipper_length_mm)

penguins2 |> select(new_var)
```

```
# A tibble: 344 x 1
   new_var
     <dbl>
 1    20.7
 2    20.4
 3    16.7
 4    NA
 5    17.9
 6    19.2
 7    20.0
 8    24.0
 9    18.0
10    22.4
# i 334 more rows
```

Task 3: In this case, `relocate()` and `left()` that is from the `griffen` package could do the same job to move the new variable to the front. In the Method 2, I demonstrate the optional operation that can change the position of new variable within the `mutate()` function.

```
library(pacman)
p_load(palmerpenguins)
#Method 1
penguins2 <- penguins |>
```

```
  mutate(new_var = body_mass_g/flipper_length_mm) |>
  left(new_var) #this is from the package 'griffen'
  #relocate(new_var, .before = species)

#Method 2
penguins2 <- penguins |>
  mutate(new_var = body_mass_g/flipper_length_mm, .before=1)
  #mutate(new_var = body_mass_g/flipper_length_mm, .before=species)

names(penguins2)
```

```
[1] "new_var"           "species"           "island"
[4] "bill_length_mm"    "bill_depth_mm"     "flipper_length_mm"
[7] "body_mass_g"       "sex"               "year"
```

Task 4: For this question, we can first use `distinct()` to check the value and names of variables, `print()` or `print_all()` could be used to print out the data.frame. The new variable is named as `mass_flipper`. We want to make the name clear and could be easily understood by other people who may use the data later. We often keep variable names concise but meaningful.

```
penguins2 <- penguins2 |>
  distinct()|>
  rename(mass_flipper = new_var) |>
  print()
```

```
# A tibble: 344 x 9
   mass_flipper species island    bill_length_mm bill_depth_mm flipper_length_mm
          <dbl> <fct>   <fct>              <dbl>         <dbl>             <int>
 1         20.7 Adelie  Torgersen           39.1          18.7               181
 2         20.4 Adelie  Torgersen           39.5          17.4               186
 3         16.7 Adelie  Torgersen           40.3          18                 195
 4         NA   Adelie  Torgersen           NA            NA                  NA
 5         17.9 Adelie  Torgersen           36.7          19.3               193
 6         19.2 Adelie  Torgersen           39.3          20.6               190
 7         20.0 Adelie  Torgersen           38.9          17.8               181
 8         24.0 Adelie  Torgersen           39.2          19.6               195
 9         18.0 Adelie  Torgersen           34.1          18.1               193
10         22.4 Adelie  Torgersen           42            20.2               190
# i 334 more rows
# i 3 more variables: body_mass_g <int>, sex <fct>, year <int>
```

```
  #print_all()

#penguins2 |> distinct(mass_flipper, species)
```

Task 5:

```
penguins2 |>
  group_by(species) |>
  summarise(mean_mass_flipper = mean(mass_flipper))
```

```
# A tibble: 3 x 2
  species   mean_mass_flipper
  <fct>                 <dbl>
1 Adelie                 19.5
2 Chinstrap              19.0
3 Gentoo                 23.3
```

Task 6: According to the results, we can find that *Chinstrap* penguins have the smallest average value and *Gentoo* penguins have the largest average value. *Adelie* penguins are in the middle.

Task 7: I demonstrate how to use `rename()` function to change the name of variable and save it to the data.frame.

```
penguins2 <- penguins2 |>
  rename(mass_flipper2 = mass_flipper)
```

Task 8:

```
print(penguins2)
```

```
# A tibble: 344 x 9
   mass_flipper2 species island   bill_length_mm bill_depth_mm flipper_length_mm
           <dbl> <fct>   <fct>             <dbl>         <dbl>             <int>
 1          20.7 Adelie  Torgers~           39.1          18.7               181
 2          20.4 Adelie  Torgers~           39.5          17.4               186
 3          16.7 Adelie  Torgers~           40.3          18                 195
 4          NA   Adelie  Torgers~           NA            NA                  NA
 5          17.9 Adelie  Torgers~           36.7          19.3               193
 6          19.2 Adelie  Torgers~           39.3          20.6               190
```

```
 7          20.0 Adelie  Torgers~           38.9          17.8               181
 8          24.0 Adelie  Torgers~           39.2          19.6               195
 9          18.0 Adelie  Torgers~           34.1          18.1               193
10          22.4 Adelie  Torgers~           42            20.2               190
# i 334 more rows
# i 3 more variables: body_mass_g <int>, sex <fct>, year <int>
```

```
#Alternatively, we can finish these two steps together with the pipe
#penguins2 <- penguins2 |>
  #rename(mass_flipper2 = mass_flipper) |>
  #print()
```

Task 9: We find that "species" "island" "sex" are factor variables.

```
penguins2 |>
  summarise(
    across(
      where(is.factor)
      )
  )
```

```
Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
dplyr 1.1.0.
i Please use `reframe()` instead.
i When switching from `summarise()` to `reframe()`, remember that `reframe()`
  always returns an ungrouped data frame and adjust accordingly.
```

```
# A tibble: 344 x 3
   species island    sex
   <fct>   <fct>     <fct>
 1 Adelie  Torgersen male
 2 Adelie  Torgersen female
 3 Adelie  Torgersen female
 4 Adelie  Torgersen <NA>
 5 Adelie  Torgersen female
 6 Adelie  Torgersen male
 7 Adelie  Torgersen female
 8 Adelie  Torgersen male
 9 Adelie  Torgersen <NA>
10 Adelie  Torgersen <NA>
# i 334 more rows
```

```
#Alternatively, we can also do this, put `names()` can return only the names.
penguins2 |>
  select(where(is.factor)) |>
  names()
```

```
[1] "species" "island"  "sex"
```

Task 10: The following shows the levels of each variable. I use `is.ordered()` to test where they are ordered. The result says it is not but these three variables are ordered based on the first letter's alphabetical order.

```
levels(penguins2$species)
```

```
[1] "Adelie"    "Chinstrap" "Gentoo"
```

```
is.ordered(penguins2$species)
```

```
[1] FALSE
```

```
levels(penguins2$island)
```

```
[1] "Biscoe"    "Dream"     "Torgersen"
```

```
is.ordered(penguins2$island)
```

```
[1] FALSE
```

```
levels(penguins2$sex)
```

```
[1] "female" "male"
```

```
is.ordered(penguins2$sex)
```

```
[1] FALSE
```

Task 11: I used two functions here to operate this. They are `mutate()` and `factor()`. The first one allows us to change variables in the data.frame and the second one is used to change the order in `sex`.

```r
penguins2 <- penguins2 |>
  mutate(sex = factor(sex, levels = c('male', 'female')))

levels(penguins2$sex)
```

```
[1] "male"   "female"
```

Task 12: After we change the order of `sex`, the coefficient turns into the exact opposite of the former one. Previously, the intercept and coefficient were 3862.3 and 683.4. Now, the intercept and coefficient are 4545.7 and -683.4.

```r
regression <- function(dataFrame){
  lm (body_mass_g ~ sex, dataFrame)
}

regression(penguins2)
```

```

Call:
lm(formula = body_mass_g ~ sex, data = dataFrame)

Coefficients:
(Intercept)    sexfemale
     4545.7       -683.4
```