Coding Assignment 4

Overview

For this assignment, you will continue to work with the sleep study data. As a reminder, the documentation for the data can be found here, but the data are on Github. (Note that the documentation is not entirely accurate, but it should still provide sufficient guidance for our purposes.)

As before, run the following chunk to load `SleepStudy.Rda`. This will load the R object `SleepStudy` (a data frame). For this code chunk to run, be sure that the data file is in a subfolder of your DSC201 folder called "data."

```r
load("data/SleepStudy.Rda")
```

For Questions 1 through 6, I encourage you to NOT use ChatGPT. But do refer to https://dplyr.tidyverse.org/reference/index.html. If you do not know what code to write or if your code does not work, write a comment describing what you are trying to do. Partial credit will be given for descriptions.

(1) In the following code chunk, keep the library statement. This loads the `dplyr` package. In this code chunk, convert the data frame to a tibble called `SleepTibble`. Then get a summary description of the data using `glimpse()`. [1 POINT]

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
SleepTibble <- as_tibble(SleepStudy)
glimpse(SleepTibble)
```

```
## Rows: 253
## Columns: 30
## $ Gender           <int> 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,~
## $ ClassYear        <int> 4, 4, 4, 1, 4, 4, 2, 2, 1, 4, 2, 2, 1, 3, 3, 3, 2, 2,~
## $ LarkOwl          <fct> Neither, Neither, Owl, Lark, Owl, Neither, Lark, Lark~
## $ NumEarlyClass    <int> 0, 2, 0, 5, 0, 0, 2, 0, 2, 2, 1, 0, 4, 2, 5, 0, 2, 5,~
## $ EarlyClass       <int> 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,~
## $ GPA              <dbl> 3.60, 3.24, 2.97, 3.76, 3.20, 3.50, 3.35, 3.00, 4.00,~
## $ ClassesMissed    <int> 0, 0, 12, 0, 4, 0, 2, 0, 0, 0, 0, 0, 0, 6, 1, 0, 5, 0~
## $ CognitionZscore  <dbl> -0.26, 1.39, 0.38, 1.39, 1.22, -0.04, 0.41, -0.59, 1.~
## $ PoorSleepQuality <int> 4, 6, 18, 9, 9, 6, 2, 10, 5, 2, 11, 8, 3, 7, 4, 4, 8,~
## $ DepressionScore  <int> 4, 1, 18, 1, 7, 14, 1, 2, 12, 6, 2, 2, 10, 3, 2, 3, 1~
## $ AnxietyScore     <int> 3, 0, 18, 4, 25, 8, 0, 2, 16, 11, 12, 8, 13, 3, 0, 1,~
## $ StressScore      <int> 8, 3, 9, 6, 14, 28, 1, 3, 20, 31, 13, 11, 18, 2, 3, 1~
## $ DepressionStatus <fct> normal, normal, moderate, normal, normal, moderate, n~
## $ AnxietyStatus    <fct> normal, normal, severe, normal, severe, moderate, nor~
## $ Stress           <fct> normal, normal, normal, normal, normal, high, normal,~
## $ DASScore         <int> 15, 4, 45, 11, 46, 50, 2, 7, 48, 48, 27, 21, 41, 8, 5~
## $ Happiness        <int> 28, 25, 17, 32, 15, 22, 25, 29, 29, 30, 14, 24, 21, 2~
## $ AlcoholUse       <fct> Moderate, Moderate, Light, Light, Moderate, Abstain, ~
## $ Drinks           <int> 10, 6, 3, 2, 4, 0, 6, 3, 3, 6, 10, 10, 4, 5, 0, 2, 4,~
## $ WeekdayBed       <dbl> 25.75, 25.70, 27.44, 23.50, 25.90, 23.80, 25.35, 23.9~
## $ WeekdayRise      <dbl> 8.70, 8.20, 6.55, 7.17, 8.67, 8.95, 8.48, 9.07, 8.75,~
## $ WeekdaySleep     <dbl> 7.70, 6.80, 3.00, 6.77, 6.09, 9.05, 7.73, 9.02, 8.25,~
## $ WeekendBed       <dbl> 25.75, 26.00, 28.00, 27.00, 23.75, 26.00, 25.63, 25.1~
## $ WeekendRise      <dbl> 9.50, 10.00, 12.59, 8.00, 9.50, 10.75, 10.13, 9.75, 9~
## $ WeekendSleep     <dbl> 5.88, 7.25, 10.09, 7.25, 7.00, 9.00, 7.00, 9.00, 9.25~
## $ AverageSleep     <dbl> 7.18, 6.93, 5.02, 6.90, 6.35, 9.04, 7.52, 9.01, 8.54,~
## $ AllNighter       <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Sex              <fct> Female, Female, Female, Female, Female, Male, Male, F~
## $ allNighter       <fct> No, No, No, No, No, No, Yes, No, No, No, No, No, No, ~
## $ earlyClass       <fct> No, Yes, No, Yes, No, No, Yes, No, Yes, Yes, Yes, No,~
```

Data Moves

"Data moves" are a data science framework introduced by Erickson et. al. (2019). They define a data move as an action that alters a dataset's contents, structure, or values. They define six core data moves: (1) filtering, (2) grouping, (3) summarizing, (4) calculating, (5) merging/joining, and (5) making hierarchy. We will cover the first four data moves in this assignment, while practicing `dplyr` functionality.

Filtering

Filtering produces a subset of data. It serves at least two important purposes. First, if a dataset includes extraneous cases, filtering removes the irrelevant ones. This is sometimes called scoping—reducing the scope of the investigation—or focusing. Also, filtering may be used in order to reduce the complexity or quantity of data in order to gain insight. (Largely copied from Erickson et. al., 2019) We learned several `dplyr` functions/verbs that do filtering. Note that filtering does operations on rows.

(2) Filter the sleep study tibble in a way that reduces reduce your rows according to some criteria, following the steps below:

- Before producing the code, write 1-3 sentences describing the filtering you are doing and how such filtering may be helpful in gaining insights from the data. For example, "I am creating a reduced data set that includes only students who have a normal stress level (for whom the variable `Stress` is equal to"normal"). This would allow me to analyze sleep patterns among students who have normal stress-levels." [1 POINT]

First, I am creating a reduced data set that includes only seniors. This would allow me explore sleep patterns among this group of more advanced students. Second, I am creating a dataset of female students (Gender = 0), so that I can understand trends among females.

- Create a code chunk and use a `dplyr` function/verb to do the filtering. Create a data frame (name as you choose) to hold the new, filtered data (the revised version of `SleepStudy`). [1 POINT]

```r
# Filtering to only seniors (showing 2 versions)
SS_seniors_ver1 <- filter(SleepTibble, ClassYear == 4)
SS_seniors_ver2 <- SleepTibble %>%
  filter(ClassYear == 4)

# Filtering to only females (showing 2 versions)
```

```
SS_females_ver1 <- filter(SleepTibble,Gender == 0)
SS_females_ver2 <- SleepTibble %>%
  filter(Gender == 0)
```

- Write 1-2 lines of code that will display results that help you check if your filtering worked as you expected. This may be a print or some other display. If you decide to print data, *limit your print to a subset of rows and columns so that you can view the entire display on your screen.* You do not have to use `dplyr` for the check. [1 POINT]

```
# First, I'm going to check how may observations are for seniors
table(SleepTibble$ClassYear)
```

```
##
##  1  2  3  4
## 47 95 54 57
```

```
# Then I will see if the number of rows matches the number of seniors
nrow(SS_seniors_ver1)
```

```
## [1] 57
```

```
nrow(SS_seniors_ver2)
```

```
## [1] 57
```

```
# Computing number of females in orig tibble by summing number of obs with gender equal to 0
sum(SleepTibble$Gender==0)
```

```
## [1] 151
```

```
# Then I will see if the number of rows matches the number of females
nrow(SS_females_ver1)
```

```
## [1] 151
```

```
nrow(SS_females_ver2)
```

```
## [1] 151
```

```
# Note we can also do this check with dplyr
nfemale <- SS_females_ver1 %>%
  summarise(count = n())
print(paste("number of females =",nfemale))
```

```
## [1] "number of females = 151"
```

Grouping

Grouping is typically used to set up a comparison among different subgroups of a dataset. Just as filtering restricts analysis to a single subset, grouping divides a dataset into multiple subsets. This division is guided by the available value(s) of some variable or variables so that, among the observations within each resulting group, the values of these "grouping" variables are the same. Note that "binning" is a special type of grouping that uses ranges of continuous values (bins or classes) to determine group membership. (Largely copied from Erickson et. al., 2019)

(3) Group the sleep study tibble by following these steps:

- Write 1-3 sentences describing the grouping you are doing and how such grouping may be helpful in gaining insights from the data. For example, "I am grouping students by their gender. This would allow me to compare summary statistics by gender." [1 POINT]

I am grouping students by stress level so that I can compare sleep habbits by stress level. I am also group students by whether they have an early class or not so that I can compare sleep habits among those who do and do not have ealy classes. - Create a code chunk and use a `dplyr` function/verb to do the grouping. Create a data frame (name as you choose) to hold the new, grouped data (the revised version of `SleepStudy`). [1 POINT]

```
# Grouping by stress level (2 versions)
SS_bystress_ver1 <- group_by(SleepTibble,Stress)
SS_bystress_ver2 <- SleepTibble %>%
  group_by(Stress)

# Grouping by whether student has an early class (2 versions)
SS_byearly_ver1 <- group_by(SleepTibble,EarlyClass)
SS_byearly_ver2 <- SleepTibble %>%
  group_by(EarlyClass)
```

- Write 1-2 lines of code that will display some results that help you check if your grouping worked as you expected. This may be a print or some other display. You do not have to use `dplyr` for the check. [1 POINT]

```
# Using glimpse, which tells if a tibble is grouped
  # Just showing for one to save space
glimpse(SS_bystress_ver1)
```

```
## Rows: 253
## Columns: 30
## Groups: Stress [2]
## $ Gender           <int> 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,~
## $ ClassYear        <int> 4, 4, 4, 1, 4, 4, 2, 2, 1, 4, 2, 2, 1, 3, 3, 3, 2, 2,~
## $ LarkOwl          <fct> Neither, Neither, Owl, Lark, Owl, Neither, Lark, Lark~
## $ NumEarlyClass    <int> 0, 2, 0, 5, 0, 0, 2, 0, 2, 2, 1, 0, 4, 2, 5, 0, 2, 5,~
## $ EarlyClass       <int> 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,~
## $ GPA              <dbl> 3.60, 3.24, 2.97, 3.76, 3.20, 3.50, 3.35, 3.00, 4.00,~
## $ ClassesMissed    <int> 0, 0, 12, 0, 4, 0, 2, 0, 0, 0, 0, 0, 0, 6, 1, 0, 5, 0~
## $ CognitionZscore  <dbl> -0.26, 1.39, 0.38, 1.39, 1.22, -0.04, 0.41, -0.59, 1.~
## $ PoorSleepQuality <int> 4, 6, 18, 9, 9, 6, 2, 10, 5, 2, 11, 8, 3, 7, 4, 4, 8,~
## $ DepressionScore  <int> 4, 1, 18, 1, 7, 14, 1, 2, 12, 6, 2, 2, 10, 3, 2, 3, 1~
## $ AnxietyScore     <int> 3, 0, 18, 4, 25, 8, 0, 2, 16, 11, 12, 8, 13, 3, 0, 1,~
## $ StressScore      <int> 8, 3, 9, 6, 14, 28, 1, 3, 20, 31, 13, 11, 18, 2, 3, 1~
## $ DepressionStatus <fct> normal, normal, moderate, normal, normal, moderate, n~
## $ AnxietyStatus    <fct> normal, normal, severe, normal, severe, moderate, nor~
## $ Stress           <fct> normal, normal, normal, normal, normal, high, normal,~
## $ DASScore         <int> 15, 4, 45, 11, 46, 50, 2, 7, 48, 48, 27, 21, 41, 8, 5~
## $ Happiness        <int> 28, 25, 17, 32, 15, 22, 25, 29, 29, 30, 14, 24, 21, 2~
## $ AlcoholUse       <fct> Moderate, Moderate, Light, Light, Moderate, Abstain, ~
## $ Drinks           <int> 10, 6, 3, 2, 4, 0, 6, 3, 3, 6, 10, 10, 4, 5, 0, 2, 4,~
## $ WeekdayBed       <dbl> 25.75, 25.70, 27.44, 23.50, 25.90, 23.80, 25.35, 23.9~
## $ WeekdayRise      <dbl> 8.70, 8.20, 6.55, 7.17, 8.67, 8.95, 8.48, 9.07, 8.75,~
## $ WeekdaySleep     <dbl> 7.70, 6.80, 3.00, 6.77, 6.09, 9.05, 7.73, 9.02, 8.25,~
## $ WeekendBed       <dbl> 25.75, 26.00, 28.00, 27.00, 23.75, 26.00, 25.63, 25.1~
## $ WeekendRise      <dbl> 9.50, 10.00, 12.59, 8.00, 9.50, 10.75, 10.13, 9.75, 9~
## $ WeekendSleep     <dbl> 5.88, 7.25, 10.09, 7.25, 7.00, 9.00, 7.00, 9.00, 9.25~
## $ AverageSleep     <dbl> 7.18, 6.93, 5.02, 6.90, 6.35, 9.04, 7.52, 9.01, 8.54,~
## $ AllNighter       <int> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Sex              <fct> Female, Female, Female, Female, Female, Male, Male, F~
## $ allNighter       <fct> No, No, No, No, No, No, Yes, No, No, No, No, No, No, ~
## $ earlyClass       <fct> No, Yes, No, Yes, No, No, Yes, No, Yes, Yes, Yes, No,~
```

```
# I can compute a summary statistic on another variable that will do so by group
summarize(SS_byearly_ver1,mean_ec=mean(GPA))
```

```
## # A tibble: 2 x 2
##   EarlyClass mean_ec
##        <int>   <dbl>
## 1          0    3.20
## 2          1    3.27
```

```
# Two other ways we did not learn in class
  # Get the grouping variables in a grouped tibble
which.group <- group_vars(SS_bystress_ver2)
print(which.group)
```

```
## [1] "Stress"
```

```
  # Check if tibble is grouped
check.grouped <- is_grouped_df(SS_byearly_ver2)
print(check.grouped)
```

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

Summarizing

Analysts often compute values that summarize a group (even if the group is the entire data set). Summarizing is the process of producing and recording a summary or aggregate value, i.e., a statistic. There are a wide variety of summary measures, and "summary" does not necessarily mean "numerical" or "typical." Often, the point of summarizing is not even the chosen aggregate measure, or the results of that measure across groups. The purpose may be deeper: The value of an aggregate measure summarizes a group, and that summary value can then be used as data in further analysis.

Grouping and summarizing work together to help an analyst get a simpler display or dataset—many fewer points!—that more clearly shows an overall pattern. Note, though, that consolidation into simpler distinct categories leads to a reduction of information. For example, when a display shows only measures of center, variability is lost. (Largely copied from Erickson et. al., 2019)

(4) Summarize the sleep study tibble. Combine your summarizing with grouping. Follow these steps:

- Write 1-3 sentences describing the grouping and summarizing you are doing and how such summarizing may be helpful in gaining insights from the data. For example, "I am grouping students by whether they have an early class (whether the variable `EarlyClass==1`) and then computing the mean time that students go to bed on weekdays, by group (the mean of the variable `WeekdayBed`). This allows me to see if students who have an early class go to bed earlier on average." [1 POINT]

First, I am summing the number of observations with Gender = 1 so I can compute the number of males. Second, I am grouping students by their alcohol use and taking the minimum number of drinks per week so I can see the minimum cut-off for each rating of Moderate, Light and Heavy.

- Create a code chunk and use a `dplyr` function/verb to do the summarizing. Create an object (name as you choose) to hold the summarized data (remember that the summary results are a small data frame, as we discussed in class). [1 POINT]

```
# Counting number of males
# Works because this is a 0/1 variable
n.males <- summarize(SleepTibble,sum_Gender=sum(Gender))
n.males
```

```
## # A tibble: 1 x 1
##   sum_Gender
##        <int>
## 1        102
```

```r
# Grouping by alcohol use and looking min number of drinks in each group
min_by_AlcoholUse <- SleepTibble %>%
  group_by(AlcoholUse) %>%
  summarize(
    min_drinks = min(Drinks)
  )
print(min_by_AlcoholUse)
```

```
## # A tibble: 4 x 2
##   AlcoholUse min_drinks
##   <fct>           <int>
## 1 Abstain             0
## 2 Heavy               5
## 3 Light               1
## 4 Moderate            3
```

- Write a sentence or two interpreting your results. [1 POINT]

There are 102 males in the sleep study data (in SleepTibble). There are four levels of alcohol use. Those who abstain have zero drinks per week. Those who are considered "light" drinkers have at least 1 drink per week but less than 3. Those who are considered "moderate" drinkers have at least 3 drinks per week but less than 5. And those who are considered "heavy" drinkers have at least 5 drinks per week.

Calculating

Another data move is to create a new variable, often represented by a new column in a data table. Because this typically involves calculating the values in this new variable, this data move is called calculating by Erickson et. al. Others refer to calculating as "mutating" or "transforming." Many new variables are calculated using the values from one or more existing variables, which is what you will do below.

(5) Create two new variables in the data frame `SleepStudy`. Each new variable should be created as a calculation or transformation performed on one or more existing variables in `SleepStudy`.

- For each new variable, write 1-3 sentences defining it and how it may be helpful for other analyses or research questions about the data. For example, "I am creating a variable called `HighHappiness` which is equal to one if `Happiness` is greater than 26. This would allow me to group students by whether they have high happiness levels or not." [1 POINT]

I am creating a variable called "Depression." It is equal to 1 if a student experiences "moderate" or "severe" depression. This would allow me to compare results between those with no depression vs those with at least some. I am also creating a variable called "DepressionOrAnxiety." It is equal to 1 if a student experiences "moderate" or "severe" depression or anxiety.

- Use `dplyr` to create both new variables your just described in one code chunk. Create a new data frame (name as you choose) to hold the new, expanded data (identical to `SleepStudy` but with two new variables). [1 POINT]

```r
# Creating revised tibble with additional variables
SleepTibble_expanded <- SleepTibble %>%
  mutate(
    Depression = ifelse(DepressionStatus %in% c("severe","moderate"), 1, 0),
    DepressionOrAnxiety = ifelse(
      DepressionStatus %in% c("moderate", "severe") |
      AnxietyStatus %in% c("moderate", "severe"), 1, 0)
  )
```

- Write 1-2 lines of code that will display some results that help you check if your grouping worked as you expected. This may be a print or some other display. You do not have to use `dplyr` for the check. [1 POINT]

```
# Getting columns involved
selected_columns <- SleepTibble_expanded %>%
  select(DepressionStatus, AnxietyStatus, Depression, DepressionOrAnxiety)

# To view the first few rows of the selected columns
head(selected_columns)
```

```
## # A tibble: 6 x 4
##   DepressionStatus AnxietyStatus Depression DepressionOrAnxiety
##   <fct>            <fct>              <dbl>               <dbl>
## 1 normal           normal                 0                   0
## 2 normal           normal                 0                   0
## 3 moderate         severe                 1                   1
## 4 normal           normal                 0                   0
## 5 normal           severe                 0                   1
## 6 moderate         moderate               1                   1
```

(6) In the last homework (Coding-Assignment3.Rmd), you explored the question: Do students with insufficient sleep have lower GPA's than students with sufficient sleep? You used base R to follow steps to answer this question in a code chunk named `compareGPA`. For extra credit, create a code chunk and use `dplyr` to carry out the same analysis. That is, use `dplyr` to compute the GPA for students with insufficent sleep and the GPA for students with sufficient sleep. [2 POINTS]

```
library(dplyr)

# Create SufficientSleep
SleepTibble <- SleepTibble %>%
  mutate(SufficientSleep = ifelse(AverageSleep >= 8, 1, 0))

# Compute mean GPAs for each group (sufficient and insufficient sleep) in one step
mean_gpa_by_sleep_status <- SleepTibble %>%
  group_by(SufficientSleep) %>%
  summarise(mean_GPA = mean(GPA, na.rm = TRUE))

# To view the result
print(mean_gpa_by_sleep_status)
```

```
## # A tibble: 2 x 2
##   SufficientSleep mean_GPA
##             <dbl>    <dbl>
## 1               0     3.28
## 2               1     3.21
```

(7) With the assistance of ChatGPT:

- `1. Use `dplyr` to count how many people in the data have each stress-level ("normal" or "high").`
- `2. Check your work by using a different approach to count how many people in the data have each str`

Share your prompts and comment on what was helpful or what was confusing. [3 POINTS FOR COMPLETION, ALL ANSWERS OK.]

Here is my prompt for ChatGPT: I have an R data frame called SleepStudy. It has a column named “Stress”, which has character data. How do I use dplyr to count the values of “Stress”?

This is the code that ChatGPT provided:

```r
SleepStudy %>%
  group_by(Stress) %>%
  summarize(count = n()) %>%
  arrange(desc(count))  # Optionally, sort by the count in descending order
```

```
## # A tibble: 2 x 2
##   Stress count
##   <fct>  <int>
## 1 normal   197
## 2 high      56
```

I then followed up with this prompt: Can you provide an alternative way in base R to do the same thing so that I can check my results?

And I got this:

```r
table(SleepStudy$Stress)
```

```
## 
##   high normal 
##     56    197
```

I found ChatGPT to be very helpful in providing code I could just cut and paste into my code chunks. Te results from the two different approaches match. I feel confident the code is providing the right results.

TOTAL OF 20 POINTS # References

Erickson, T., Wilkerson, M., Finzer, W., & Reichsman, F. (2019). Data Moves. Technology Innovations in Statistics Education, 12(1). http://dx.doi.org/10.5070/T5121038001 Retrieved from https://escholarship.org/uc/item/0mg8m7g6