Assignment 1: Manipulation of mammalian sleep data

Your name and student ID

Today's date

Instructions

- Solutions will be released on Tuesday, September 1.
- This semester, homework assignments are for practice only and will not be turned in for marks.

Helpful hints:

- Every function you need to use was taught during lecture! So you may need to revisit the lecture code to help you along by opening the relevant files on Datahub. Alternatively, you may wish to view the code in the condensed PDFs posted on the course website. Good luck!
- Knit your file early and often to minimize knitting errors! If you copy and paste code for the slides, you are bound to get an error that is hard to diagnose. Typing out the code is the way to smooth knitting! We recommend knitting your file each time after you write a few sentences/add a new code chunk, so you can detect the source of knitting errors more easily. This will save you and the GSIs from frustration!
- It is good practice to not allow your code to run off the page. To avoid this, have a look at your knitted PDF and ensure all the code fits in the file. If it doesn't look right, go back to your .Rmd file and add spaces (new lines) using the return or enter key so that the code runs onto the next line.

Begin by knitting this document by pushing the "Knit" button above. As you fill in code and text in the document, you can re-knit (push the button again) and see how the document changes. It is important to re-knit often, because if there is any error in your code, the file will not generate a PDF, so our advice is to knit early and often!

Using `dplyr` to investigate sleep times in mammals

The data file `sleep.csv` contains the sleeptimes and weights for a set of mammals. Hit the green arrow icon in the line below to execute the two lines of code in the code chunk, or execute them line by line by placing your cursor on the first line and hitting cmd + enter on Mac or ctrl + enter on PC.

```
library(dplyr)
library(readr)
sleep <- read_csv("sleep.csv")
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   genus = col_character(),
##   vore = col_character(),
##   order = col_character(),
##   conservation = col_character(),
##   sleep_total = col_double(),
##   sleep_rem = col_double(),
##   sleep_cycle = col_double(),
##   awake = col_double(),
##   brainwt = col_double(),
##   bodywt = col_double()
## )
```

- The `library` command loads the library `dplyr` into memory.
- The `readr` library contains functions to read in the dataset.
- The `dplyr` library contains functions we will use to manipulate data.

Notice that an object called `sleep` appeared in the Environment tab under "Data".

1. [2 points] Use four useful functions discussed in lecture to examine the sleep data set:

```
# Text inside a code chunk that begins with "#" is called a comment.
# We sometimes use comments to explain code to you in plain English.
# Write your four functions below these comments, replacing the placeholder
# text "<<<<YOUR CODE HERE>>>>". Remember, code does *not* begin with a "#"

"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
"<<<<YOUR CODE HERE>>>>"
```

```
## [1] "<<<<YOUR CODE HERE>>>>"
```

```
# BEGIN SOLUTION
dim(sleep)
```

```
## [1] 83 11
```

```
head(sleep)
```

```
## # A tibble: 6 x 11
##   name  genus vore  order conservation sleep_total sleep_rem sleep_cycle
##   <chr> <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl>
## 1 Chee~ Acin~ carni Carn~ lc                  12.1      NA        NA
## 2 Owl ~ Aotus omni  Prim~ <NA>                17         1.8      NA
## 3 Moun~ Aplo~ herbi Rode~ nt                  14.4       2.4      NA
## 4 Grea~ Blar~ omni  Sori~ lc                  14.9       2.3       0.133
## 5 Cow   Bos   herbi Arti~ domesticated         4         0.7       0.667
## 6 Thre~ Brad~ herbi Pilo~ <NA>                14.4       2.2       0.767
## # ... with 3 more variables: awake <dbl>, brainwt <dbl>, bodywt <dbl>
```

```
names(sleep)
```

```
##  [1] "name"         "genus"        "vore"         "order"
##  [5] "conservation" "sleep_total"  "sleep_rem"    "sleep_cycle"
##  [9] "awake"        "brainwt"      "bodywt"
```

```r
str(sleep)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 83 obs. of  11 variables:
##  $ name        : chr  "Cheetah" "Owl monkey" "Mountain beaver" "Greater short-tailed shrew" ...
##  $ genus       : chr  "Acinonyx" "Aotus" "Aplodontia" "Blarina" ...
##  $ vore        : chr  "carni" "omni" "herbi" "omni" ...
##  $ order       : chr  "Carnivora" "Primates" "Rodentia" "Soricomorpha" ...
##  $ conservation: chr  "lc" NA "nt" "lc" ...
##  $ sleep_total : num  12.1 17 14.4 14.9 4 14.4 8.7 7 10.1 3 ...
##  $ sleep_rem   : num  NA 1.8 2.4 2.3 0.7 2.2 1.4 NA 2.9 NA ...
##  $ sleep_cycle : num  NA NA NA 0.133 0.667 ...
##  $ awake       : num  11.9 7 9.6 9.1 20 9.6 15.3 17 13.9 21 ...
##  $ brainwt     : num  NA 0.0155 NA 0.00029 0.423 NA NA NA 0.07 0.0982 ...
##  $ bodywt      : num  50 0.48 1.35 0.019 600 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   name = col_character(),
##   ..   genus = col_character(),
##   ..   vore = col_character(),
##   ..   order = col_character(),
##   ..   conservation = col_character(),
##   ..   sleep_total = col_double(),
##   ..   sleep_rem = col_double(),
##   ..   sleep_cycle = col_double(),
##   ..   awake = col_double(),
##   ..   brainwt = col_double(),
##   ..   bodywt = col_double()
##   .. )
```

```r
# END SOLUTION

# Then, assign p1 to a vector of your function names, in alphabetical order.
# For example, assigning p0 to a vector of fruits looks like this:
# p0 <- c("apple", "banana", "orange")

p1 <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
p1 <- c("dim", "head", "names", "str")

# END SOLUTION
check_problem1()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
##
## Problem 1
## Checkpoints Passed: 2
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

Description of the variables found in the sleep dataset:

Column name	Description
name	common name
genus	taxonomic rank
vore	carnivore, omnivore or herbivore?
order	taxonomic rank
conservation	the conservation status of the mammal
sleep_total	total amount of sleep, in hours
sleep_rem	Rapid eye movement (REM) sleep, in hours
sleep_cycle	length of sleep cycle, in hours
awake	amount of time spent awake, in hours
brainwt	brain weight in kilograms
bodywt	body weight in kilograms

2. [2 points] Write code to select a set of columns. Specifically select the `awake`, `brainwt`, and `bodywt` columns. Assign this smaller dataset to a data frame called `sleep_small`

```
sleep_small <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_small <- sleep %>% select(awake, brainwt, bodywt)

# ALTERNATE SOLUTION
sleep_small <- select(sleep, awake, brainwt, bodywt)

# END SOLUTION
check_problem2()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 2
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

3. [1 point] To select a range of columns by name, use the ":" (colon) operator. Redo the selection for question 1, but use the colon operator. Assign this to `sleep_small_colon`. Note that this returns the same data frame as the previous problem, but is not recommended in practice because it depends on the ordering of the columns and isn't explicit in the columns that are selected, whereas selection by name offers much higher readability for someone else looking at your code later on.

```
sleep_small_colon <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_small_colon <- sleep %>% select(awake:bodywt)

# END SOLUTION
check_problem3()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 3
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

4. [1 point] Select all the columns except for the `vore` variable. Assign this to `sleep_no_vore`.

```
sleep_no_vore <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_no_vore <- sleep %>% select(-vore)

# END SOLUTION
check_problem4()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 4
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

5. [1 point] Run the following chunk of code.

```
select(sleep, starts_with("sl"))
```

```
## # A tibble: 83 x 3
##    sleep_total sleep_rem sleep_cycle
##          <dbl>     <dbl>       <dbl>
##  1        12.1      NA         NA
##  2        17         1.8       NA
##  3        14.4       2.4       NA
##  4        14.9       2.3        0.133
##  5         4         0.7        0.667
##  6        14.4       2.2        0.767
##  7         8.7       1.4        0.383
##  8         7        NA         NA
##  9        10.1       2.9        0.333
## 10         3        NA         NA
## # ... with 73 more rows
```

What does it return? Uncomment one of the possible choices.

```
# p5 <- "returns the number of columns that start with sl"
# p5 <- "returns all columns that start with sl"
# p5 <- "returns all rows that start with sl"
# p5 <- "returns all animals whose names start with sl""

# BEGIN SOLUTION
p5 <- "returns all columns that start with sl"

# END SOLUTION
check_problem5()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
##
## Problem 5
## Checkpoints Passed: 1
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

It returns the columns that start with sl: `sleep_total`, `sleep_rem`, `sleep_cycle`

6. [1 point] Rewrite the previous chunk of code using the pipe operator. Assign this to `sleep_sl`.

```
sleep_sl <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_sl <- sleep %>% select(starts_with("sl"))

# END SOLUTION
check_problem6()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 6
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

7. [1 point] Filter the rows for mammals that sleep a total of more than 16 hours. Assign this to `sleep_over16`.

```
sleep_over16 <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_over16 <- sleep %>% filter(sleep_total > 16)

# END SOLUTION
check_problem7()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 7
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

8. [2 points] Filter the rows for mammals that sleep a total of more than 16 hours and have a body weight of greater than 1 kilogram. Assign this to `sleep_mammals`.

```
sleep_mammals <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_mammals <- sleep %>% filter(sleep_total > 16 & bodywt > 1)

# ALTERNATE SOLUTION
sleep_mammals <- sleep %>% filter(sleep_total > 16, bodywt > 1)

# END SOLUTION
check_problem8()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 8
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

9. [1 point] Suppose you are specifically interested in the sleep of horses and giraffes. Assign `sleep_hg` to a data frame for horses and giraffes only.

```
sleep_hg <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_hg <- sleep %>% filter(name %in% c("Horse", "Giraffe"))
# notice quotes and capitalization!

# END SOLUTION
check_problem9()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
## [1] "Checkpoint 4 Passed: Correct!"
##
## Problem 9
## Checkpoints Passed: 4
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

10. [1 point] Order the dataset by sleep time from shortest sleep time to longest sleep time. Assign this to `sleep_time`.

```
sleep_time <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_time <- sleep %>% arrange(sleep_total)

# END SOLUTION
check_problem10()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 10
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

11. [1 point] Now order for longest sleep time to shortest sleep time. Assign this to `sleep_time_rev`.

```
sleep_time_rev <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_time_rev <- sleep %>% arrange(-sleep_total)

# END SOLUTION
check_problem11()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 11
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

12. [2 points] Suppose you are interested in the order of sleep time, but according to whether the animal is a carnivore, herbivore, or omnivore. Rewrite the above statement to order sleep time according to the type of "-vore" that then animal is:

```
sleep_time_vore <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_time_vore <- sleep %>% arrange(vore, -sleep_total)

# END SOLUTION
check_problem12()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 12
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

13. [1 point] Create a new column called `rem_proportion` which is the ratio of rem sleep to total amount of sleep. Assign this new data frame to `sleep_ratio`.

```
sleep_ratio <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_ratio <- sleep %>% mutate(rem_proportion = sleep_rem/sleep_total)

# END SOLUTION
check_problem13()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 13
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

14. [1 point] Add a second column called `bodywt_grams` which is the bodywt column in grams.

```
sleep_r_bw <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
sleep_r_bw <- sleep %>% mutate(rem_proportion = sleep_rem/sleep_total,
                               bodywt_grams = bodywt * 1000)

# END SOLUTION
check_problem14()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 14
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

15. [1 point] Calculate the average sleep time across all the animals in the dataset using a `dplyr` function and assign it to the variable `avg_sleep_time`. Your answer should be a data frame of 1 observation and 1 variable called `sleep_avg`

```
avg_sleep_time <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
avg_sleep_time <- sleep %>% summarize(sleep_avg = mean(sleep_total))

# END SOLUTION
check_problem15()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
## [1] "Checkpoint 4 Passed: Correct!"
##
## Problem 15
## Checkpoints Passed: 4
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

16. [2 points] Calculate the average sleep time for each type of "-vore". Hint: you'll need to use two `dplyr` functions! The column names should be `vore` and `sleep_avg`.

```
avg_by_vore <- "<<<<YOUR CODE HERE>>>>"

# BEGIN SOLUTION
avg_by_vore <- sleep %>%
                  group_by(vore) %>%
                  summarize(sleep_avg = mean(sleep_total))

# END SOLUTION
check_problem16()
```

```
## [1] "Checkpoint 1 Passed: Correct!"
## [1] "Checkpoint 2 Passed: Correct!"
## [1] "Checkpoint 3 Passed: Correct!"
##
## Problem 16
## Checkpoints Passed: 3
## Checkpoints Errored: 0
## 100% passed
## --------
## Test: PASSED
```

Check your score

```
# Just run this chunk.
total_score()
```

```
##              Test Points_Possible       Type
## Problem 1  PASSED               2 autograded
## Problem 2  PASSED               2 autograded
## Problem 3  PASSED               1 autograded
## Problem 4  PASSED               1 autograded
## Problem 5  PASSED               1 autograded
## Problem 6  PASSED               1 autograded
## Problem 7  PASSED               1 autograded
## Problem 8  PASSED               2 autograded
## Problem 9  PASSED               1 autograded
## Problem 10 PASSED               1 autograded
## Problem 11 PASSED               1 autograded
## Problem 12 PASSED               2 autograded
## Problem 13 PASSED               1 autograded
## Problem 14 PASSED               1 autograded
## Problem 15 PASSED               1 autograded
## Problem 16 PASSED               2 autograded
```