Problem Set 1: Manipulation of mammalian sleep data

Your name and student ID

Today's date

Instructions

- Solutions will be released by Wednesday, January 25th.
- This semester, problem sets 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 sleep times and weights for a set of mammals. Hit the green arrow icon in the line below to execute the 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)
```

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

## The following object is masked from 'package:testthat':
##
##     matches

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

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

```
library(readr)
```

```
##
## Attaching package: 'readr'

## The following objects are masked from 'package:testthat':
##
##     edition_get, local_edition
```

```
sleep <- read_csv("data/sleep.csv")
```

```
## Rows: 83 Columns: 11

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): name, genus, vore, order, conservation
## dbl (6): sleep_total, sleep_rem, sleep_cycle, awake, brainwt, bodywt
##
## 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.
```

- 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` appears in the Environment tab under "Data".

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

```
# 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>>>>"
```

```
# 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")

. = " # BEGIN PROMPT
p1 <- NULL # YOUR CODE HERE
p1
" # END PROMPT

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

```
## [1] "dim"   "head"  "names" "str"
```

```
# END SOLUTION
```

```
test_that("p1a", {
  expect_true(length(p1) == 4)
print("p1a: Checking p1 has 4 items in a list")
})
```

```
## [1] "p1a: Checking p1 has 4 items in a list"
## Test passed
```

```
test_that("p1b", {
  expect_true(p1[1] == "dim" & p1[2] == "head" & p1[3] == "names" & p1[4] == "str")
  print("p1b: Checking the names of the 4 functions in alphabetical order")
})
```

```
## [1] "p1b: Checking the names of the 4 functions in alphabetical order"
## 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 dataframe called `sleep_small`.

```
. = " # BEGIN PROMPT
sleep_small <- NULL # YOUR CODE HERE
sleep_small
" # END PROMPT

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

```
## # A tibble: 83 x 3
##    awake  brainwt  bodywt
##    <dbl>    <dbl>   <dbl>
##  1  11.9 NA        50
##  2   7    0.0155    0.48
##  3   9.6 NA         1.35
##  4   9.1  0.00029   0.019
##  5  20    0.423   600
##  6   9.6 NA         3.85
##  7  15.3 NA        20.5
##  8  17   NA         0.045
##  9  13.9  0.07     14
## 10  21    0.0982   14.8
## # ... with 73 more rows
```

```
# END SOLUTION
```

```
test_that("p2a", {
  expect_true(is.data.frame(sleep_small))
  print("p2a: Checking sleep_small is a dataframe")
})
```

```
## [1] "p2a: Checking sleep_small is a dataframe"
## Test passed
```

```
test_that("p2b", {
  expect_true(ncol(sleep_small) == 3)
  print("p2b: Checking sleep_small has 3 columns")
})
```

```
## [1] "p2b: Checking sleep_small has 3 columns"
## Test passed
```

```
test_that("p2c", {
  expect_true(all(names(sleep_small) == c("awake", "brainwt", "bodywt")))
  print("p2c: Checking sleep_small has columns 'awake', 'brainwt', and 'bodywt'")
})
```

```
## [1] "p2c: Checking sleep_small has columns 'awake', 'brainwt', and 'bodywt'"
## 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 selecting columns by name offers much higher readability for someone else looking at your code later on.

```
. = " # BEGIN PROMPT
sleep_small_colon <- NULL # YOUR CODE HERE
sleep_small_colon
" # END PROMPT

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

```
## # A tibble: 83 x 3
##    awake  brainwt  bodywt
##    <dbl>    <dbl>   <dbl>
##  1  11.9 NA        50
##  2   7    0.0155    0.48
##  3   9.6 NA         1.35
##  4   9.1  0.00029   0.019
##  5  20    0.423   600
##  6   9.6 NA         3.85
##  7  15.3 NA        20.5
##  8  17   NA         0.045
##  9  13.9  0.07     14
## 10  21    0.0982   14.8
## # ... with 73 more rows
```

```
# END SOLUTION
```

```
test_that("p3a", {
  expect_true(is.data.frame(sleep_small_colon))
  print("p3a: Checking sleep_small_colon is a dataframe")
})
```

```
## [1] "p3a: Checking sleep_small_colon is a dataframe"
## Test passed
```

```
test_that("p3b", {
  expect_true(ncol(sleep_small_colon) == 3)
  print("p3b: Checking sleep_small_colon has 3 columns")
})
```

```
## [1] "p3b: Checking sleep_small_colon has 3 columns"
## Test passed
```

```
test_that("p3c", {
  expect_true(all(names(sleep_small_colon) == c("awake", "brainwt", "bodywt")))
print("p3c: Checking sleep_small_colon has columns 'awake', 'brainwt', and 'bodywt'")
})
```

```
## [1] "p3c: Checking sleep_small_colon has columns 'awake', 'brainwt', and 'bodywt'"
## Test passed
```

4. [1 point] Select all of the columns except for the `vore` variable from the original `sleep` dataset. Assign this to `sleep_no_vore`.

```
. = " # BEGIN PROMPT
sleep_no_vore <- NULL # YOUR CODE HERE
sleep_no_vore
" # END PROMPT

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

```
test_that("p4a", {
  expect_true(is.data.frame(sleep_no_vore))
  print("p4a: Checking sleep_small_colon is a dataframe")
})
```

```
## [1] "p4a: Checking sleep_small_colon is a dataframe"
## Test passed
```

```
test_that("p4b", {
  expect_true(ncol(sleep_no_vore) == 10)
  print("p4b: Checking sleep_small_vore has 10 columns")
})
```

```
## [1] "p4b: Checking sleep_small_vore has 10 columns"
## Test passed
```

```
test_that("p4c", {
  expect_true(!("vore" %in% names(sleep_no_vore)))
  print("p4c: Checking sleep_no_vore has no columns with 'vore'")
})
```

```
## [1] "p4c: Checking sleep_no_vore has no columns with 'vore'"
## 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? Copy your choice and assign it to p5.

```
# 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 PROMPT
p5 <- NULL # YOUR CHOICE HERE
p5
" # END PROMPT

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

```
test_that("p5a", {
  expect_true(p5 == "returns all columns that start with sl")
  print("Checking response...")
})
```

```
## [1] "Checking response..."
## Test passed
```

6. [1 point] Rewrite the chunk of code that selects columns starting with "sl" (in question 5) using the pipe operator. Assign this to `sleep_sl`.

```
. = " # BEGIN PROMPT
sleep_sl <- NULL # YOUR CODE HERE
sleep_sl
" # END PROMPT

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

```
# END SOLUTION
```

```
test_that("p6a", {
  expect_true(is.data.frame(sleep_sl))
  print("p6a: Checking sleep_sl is a dataframe")
})
```

```
## [1] "p6a: Checking sleep_sl is a dataframe"
## Test passed
```

```
test_that("p6b", {
  expect_true(ncol(sleep_sl) == 3)
  print("p6b: Checking sleep_sl has 3 columns")
})
```

```
## [1] "p6b: Checking sleep_sl has 3 columns"
## Test passed
```

```
test_that("p6c", {
  expect_true(("sleep_total" %in% names(sleep_sl)) &&
                        ("sleep_rem" %in% names(sleep_sl)) &&
                         ("sleep_cycle" %in% names(sleep_sl)))
  print("p6c: Checking sleep_sl has the 3 columns that start with sl")
})
```

```
## [1] "p6c: Checking sleep_sl has the 3 columns that start with sl"
## Test passed
```

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

```
. = " # BEGIN PROMPT
sleep_over16 <- NULL # YOUR CODE HERE
sleep_over16
" # END PROMPT

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

```
## # A tibble: 8 x 11
##   name    genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##   <chr>   <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
## 1 Owl mo~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
## 2 Long-n~ Dasy~ carni Cing~ lc                  17.4       3.1       0.383   6.6
## 3 North ~ Dide~ omni  Dide~ lc                  18         4.9       0.333   6
## 4 Big br~ Epte~ inse~ Chir~ lc                  19.7       3.9       0.117   4.3
## 5 Thick-~ Lutr~ carni Dide~ lc                  19.4       6.6      NA       4.6
## 6 Little~ Myot~ inse~ Chir~ <NA>                19.9       2         0.2     4.1
## 7 Giant ~ Prio~ inse~ Cing~ en                  18.1       6.1      NA       5.9
## 8 Arctic~ Sper~ herbi Rode~ lc                  16.6      NA        NA       7.4
## # ... with 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# END SOLUTION
```

```
test_that("p7a", {
  expect_true(is.data.frame(sleep_over16))
  print("p7a: Checking sleep_over16 is a dataframe")
})
```

```
## [1] "p7a: Checking sleep_over16 is a dataframe"
## Test passed
```

```
test_that("p7b", {
  expect_true(ncol(sleep_over16) == 11)
  print("p7b: Checking sleep_over16 has 11 columns")
})
```

```
## [1] "p7b: Checking sleep_over16 has 11 columns"
## Test passed
```

```
test_that("p7c", {
  expect_true(nrow(sleep_over16) == 8)
  print("p7c: Checking sleep_over16 has 8 rows")
})
```

```
## [1] "p7c: Checking sleep_over16 has 8 rows"
## Test passed
```

8. [2 points] Filter the rows to include 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`.

```
. = " # BEGIN PROMPT
sleep_mammals <- NULL # YOUR CODE HERE
sleep_mammals
" # END PROMPT

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

```
## # A tibble: 3 x 11
##   name    genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##   <chr>   <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
## 1 Long-n~ Dasy~ carni Cing~ lc                  17.4       3.1       0.383   6.6
## 2 North ~ Dide~ omni  Dide~ lc                  18         4.9       0.333   6
## 3 Giant ~ Prio~ inse~ Cing~ en                  18.1       6.1      NA       5.9
## # ... with 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# END SOLUTION
```

```
test_that("p8a", {
  expect_true(is.data.frame(sleep_mammals))
  print("p8a: Checking sleep_mammals is a dataframe")
})
```

```
## [1] "p8a: Checking sleep_mammals is a dataframe"
## Test passed
```

```
test_that("p8b", {
  expect_true(ncol(sleep_mammals) == 11)
  print("p8b: Checking sleep_mammals has 11 columns")
})
```

```
## [1] "p8b: Checking sleep_mammals has 11 columns"
## Test passed
```

```
test_that("p8c", {
  expect_true(nrow(sleep_mammals) == 3)
  print("p8c: Checking sleep_mammals has 3 rows")
})
```

```
## [1] "p8c: Checking sleep_mammals has 3 rows"
## Test passed
```

9. [1 point] Suppose you are specifically interested in the sleep times of horses and giraffes. Use the original `sleep` dataset and assign `sleep_hg` to a dataframe that only includes horses and giraffes.

```
. = " # BEGIN PROMPT
sleep_hg <- NULL # YOUR CODE HERE
sleep_hg
" # END PROMPT

# BEGIN SOLUTION
sleep_hg <- sleep %>% filter(name %in% c("Horse", "Giraffe"))
sleep_hg
```

```
## # A tibble: 2 x 11
##   name    genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##   <chr>   <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
## 1 Horse   Equus herbi Peri~ domesticated         2.9       0.6           1  21.1
## 2 Giraffe Gira~ herbi Arti~ cd                   1.9       0.4          NA  22.1
## # ... with 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# END SOLUTION
```

```
test_that("p9a", {
  expect_true(is.data.frame(sleep_hg))
  print("p9a: Checking sleep_hg is a dataframe")
})
```

```
## [1] "p9a: Checking sleep_hg is a dataframe"
## Test passed
```

```
test_that("p9b", {
  expect_true(ncol(sleep_hg) == 11)
  print("p9b: Checking sleep_hg has 11 columns")
})
```

```
## [1] "p9b: Checking sleep_hg has 11 columns"
## Test passed
```

```
test_that("p9c", {
  expect_true(nrow(sleep_hg) == 2)
  print("p9c: Checking sleep_hg has 2 rows")
})
```

```
## [1] "p9c: Checking sleep_hg has 2 rows"
## Test passed
```

```
test_that("p9d", {
  expect_true("Horse" %in% sleep_hg$name &&
                          "Giraffe" %in% sleep_hg$name)
  print("p9d: Checking sleep_hg has the correct rows")
})
```

```
## [1] "p9d: Checking sleep_hg has the correct rows"
## Test passed
```

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

```
. = " # BEGIN PROMPT
sleep_time <- NULL # YOUR CODE HERE
sleep_time
" # END PROMPT

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

```
## # A tibble: 83 x 11
##    name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Giraf~ Gira~ herbi Arti~ cd                   1.9       0.4      NA      22.1
##  2 Pilot~ Glob~ carni Ceta~ cd                   2.7       0.1      NA      21.4
##  3 Horse  Equus herbi Peri~ domesticated         2.9       0.6       1      21.1
##  4 Roe d~ Capr~ herbi Arti~ lc                   3        NA        NA      21
##  5 Donkey Equus herbi Peri~ domesticated         3.1       0.4      NA      20.9
##  6 Afric~ Loxo~ herbi Prob~ vu                   3.3      NA        NA      20.7
##  7 Caspi~ Phoca carni Carn~ vu                   3.5       0.4      NA      20.5
##  8 Sheep  Ovis  herbi Arti~ domesticated         3.8       0.6      NA      20.2
##  9 Asian~ Elep~ herbi Prob~ en                   3.9      NA        NA      20.1
## 10 Cow    Bos   herbi Arti~ domesticated         4         0.7       0.667  20
## # ... with 73 more rows, and 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# END SOLUTION
```

```
test_that("p10a", {
  expect_true(is.data.frame(sleep_time))
  print("p10a: Checking sleep_time is a dataframe")
})
```

```
## [1] "p10a: Checking sleep_time is a dataframe"
## Test passed
```

```
test_that("p10b", {
  expect_true(ncol(sleep_time) == 11)
  print("p10b: Checking sleep_time has 11 columns")
})
```

```
## [1] "p10b: Checking sleep_time has 11 columns"
## Test passed
```

```
test_that("p10c", {
  expect_true(nrow(sleep_time) == 83)
  print("p10c: Checking sleep_time has 83 rows")
})
```

```
## [1] "p10c: Checking sleep_time has 83 rows"
## Test passed
```

```
test_that("p10d", {
  expect_true(sleep_time$sleep_total[1] == 1.9)
  print("p10c: Checking sleep_total is in ascending order")
})
```

```
## [1] "p10c: Checking sleep_total is in ascending order"
## Test passed
```

11. [1 point] Now order the original dataset from longest to shortest sleep time. Assign this to sleep_rev.

```
. = " # BEGIN PROMPT
sleep_rev <- NULL # YOUR CODE HERE
sleep_rev
" # END PROMPT

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

```
## # A tibble: 83 x 11
##    name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Littl~ Myot~ inse~ Chir~ <NA>                19.9       2         0.2     4.1
##  2 Big b~ Epte~ inse~ Chir~ lc                  19.7       3.9       0.117   4.3
##  3 Thick~ Lutr~ carni Dide~ lc                  19.4       6.6      NA       4.6
##  4 Giant~ Prio~ inse~ Cing~ en                  18.1       6.1      NA       5.9
##  5 North~ Dide~ omni  Dide~ lc                  18         4.9       0.333   6
##  6 Long-~ Dasy~ carni Cing~ lc                  17.4       3.1       0.383   6.6
##  7 Owl m~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
##  8 Arcti~ Sper~ herbi Rode~ lc                  16.6      NA        NA       7.4
##  9 Golde~ Sper~ herbi Rode~ lc                  15.9       3        NA       8.1
## 10 Tiger  Pant~ carni Carn~ en                  15.8      NA        NA       8.2
## # ... with 73 more rows, and 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# END SOLUTION
```

```
test_that("p11a", {
  expect_true(is.data.frame(sleep_rev))
  print("p11a: Checking sleep_rev is a dataframe")
})
```

```
## [1] "p11a: Checking sleep_rev is a dataframe"
## Test passed
```

```
test_that("p11b", {
  expect_true(ncol(sleep_rev) == 11)
  print("p11b: Checking sleep_rev has 11 columns")
})
```

```
## [1] "p11b: Checking sleep_rev has 11 columns"
## Test passed
```

```
test_that("p11c", {
  expect_true(nrow(sleep_rev) == 83)
  print("p11c: Checking sleep_rev has 83 rows")
})
```

```
## [1] "p11c: Checking sleep_rev has 83 rows"
## Test passed
```

```
test_that("p11d", {
  expect_true(sleep_rev$sleep_total[1] == 19.9)
  print("p11d: Checking sleep_total is in descending order")
})
```

```
## [1] "p11d: Checking sleep_total is in descending order"
## Test passed
```

12. [2 points] Suppose you are interested in the order of sleep time (longest to shortest), but according to whether the animal is a carnivore, herbivore, or omnivore. Write the code that orders sleep time according to the animal's type of `-vore`. Call this `sleep_time_rev`.

```
. = " # BEGIN PROMPT
sleep_time_rev <- NULL # YOUR CODE HERE
sleep_time_rev
" # END PROMPT

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

```
## # A tibble: 83 x 11
##    name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Thick~ Lutr~ carni Dide~ lc                  19.4       6.6      NA       4.6
##  2 Long-~ Dasy~ carni Cing~ lc                  17.4       3.1       0.383   6.6
##  3 Tiger  Pant~ carni Carn~ en                  15.8      NA        NA       8.2
##  4 North~ Onyc~ carni Rode~ lc                  14.5      NA        NA       9.5
##  5 Lion   Pant~ carni Carn~ vu                  13.5      NA        NA      10.5
##  6 Domes~ Felis carni Carn~ domesticated        12.5       3.2       0.417  11.5
##  7 Arcti~ Vulp~ carni Carn~ <NA>                12.5      NA        NA      11.5
##  8 Cheet~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
##  9 Slow ~ Nyct~ carni Prim~ <NA>                11        NA        NA      13
## 10 Jaguar Pant~ carni Carn~ nt                  10.4      NA        NA      13.6
## # ... with 73 more rows, and 2 more variables: brainwt <dbl>, bodywt <dbl>
```

```
# END SOLUTION
```

```
test_that("p12a", {
  expect_true(is.data.frame(sleep_time_rev))
  print("p12a: Checking sleep_time_rev is a dataframe")
})
```

```
## [1] "p12a: Checking sleep_time_rev is a dataframe"
## Test passed
```

```
test_that("p12b", {
  expect_true(ncol(sleep_time_rev) == 11)
  print("p12b: Checking sleep_time_rev has 11 columns")
})
```

```
## [1] "p12b: Checking sleep_time_rev has 11 columns"
## Test passed
```

```
test_that("p12c", {
  expect_true(nrow(sleep_time_rev) == 83)
  print("p12c: Checking sleep_time_rev has 83 rows")
})
```

```
## [1] "p12c: Checking sleep_time_rev has 83 rows"
## Test passed
```

```
test_that("p12d", {
  expect_true(sleep_time_rev$sleep_total[1] == 19.4)
  print("p12c: Correct ordering by -vore type and sleep totals")
})
```

```
## [1] "p12c: Correct ordering by -vore type and sleep totals"
## 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 dataframe to `sleep_ratio`.

```
. = " # BEGIN PROMPT
sleep_ratio <- NULL # YOUR CODE HERE
sleep_ratio
" # END PROMPT

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

```
## # A tibble: 83 x 12
##    name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Cheet~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
##  2 Owl m~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
##  3 Mount~ Aplo~ herbi Rode~ nt                  14.4       2.4      NA       9.6
##  4 Great~ Blar~ omni  Sori~ lc                  14.9       2.3       0.133   9.1
##  5 Cow    Bos   herbi Arti~ domesticated         4         0.7       0.667  20
##  6 Three~ Brad~ herbi Pilo~ <NA>                14.4       2.2       0.767   9.6
##  7 North~ Call~ carni Carn~ vu                   8.7       1.4       0.383  15.3
##  8 Vespe~ Calo~ <NA>  Rode~ <NA>                 7        NA        NA      17
##  9 Dog    Canis carni Carn~ domesticated        10.1       2.9       0.333  13.9
## 10 Roe d~ Capr~ herbi Arti~ lc                   3        NA        NA      21
## # ... with 73 more rows, and 3 more variables: brainwt <dbl>, bodywt <dbl>,
## #   rem_proportion <dbl>
```

```
# END SOLUTION
```

```
test_that("p13a", {
  expect_true(is.data.frame(sleep_ratio))
  print("p13a: Checking sleep_ratio is a dataframe")
})
```

```
## [1] "p13a: Checking sleep_ratio is a dataframe"
## Test passed
```

```
test_that("p13b", {
  expect_true(ncol(sleep_ratio) == 12)
  print("p13b: Checking sleep_time_rev has 12 columns")
})
```

```
## [1] "p13b: Checking sleep_time_rev has 12 columns"
## Test passed
```

```
test_that("p13c", {
  expect_true(nrow(sleep_ratio) == 83)
  print("p13c: Checking sleep_ratio has 83 rows")
})
```

```
## [1] "p13c: Checking sleep_ratio has 83 rows"
## Test passed
```

```
test_that("p13d", {
  expect_true(all.equal(sleep_ratio$rem_proportion[2], 0.10588235, 0.01))
  print("p13c: Checking correct proportions")
})
```

```
## [1] "p13c: Checking correct proportions"
## Test passed
```

14. [1 point] Add another column to the `sleep_ratio` dataset called `bodywt_grams` which is the bodywt column in grams. Call this new dataframe `sleep_bw`.

```
. = " # BEGIN PROMPT
sleep_bw <- NULL # YOUR CODE HERE
sleep_bw
" # END PROMPT

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

```
## # A tibble: 83 x 13
##    name   genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Cheet~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
##  2 Owl m~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
##  3 Mount~ Aplo~ herbi Rode~ nt                  14.4       2.4      NA       9.6
##  4 Great~ Blar~ omni  Sori~ lc                  14.9       2.3       0.133   9.1
##  5 Cow    Bos   herbi Arti~ domesticated         4         0.7       0.667  20
##  6 Three~ Brad~ herbi Pilo~ <NA>                14.4       2.2       0.767   9.6
##  7 North~ Call~ carni Carn~ vu                   8.7       1.4       0.383  15.3
##  8 Vespe~ Calo~ <NA>  Rode~ <NA>                 7        NA        NA      17
##  9 Dog    Canis carni Carn~ domesticated        10.1       2.9       0.333  13.9
## 10 Roe d~ Capr~ herbi Arti~ lc                   3        NA        NA      21
## # ... with 73 more rows, and 4 more variables: brainwt <dbl>, bodywt <dbl>,
## #   rem_proportion <dbl>, bodywt_grams <dbl>
```

```
# END SOLUTION
```

```
test_that("p14a", {
  expect_true(is.data.frame(sleep_bw))
  print("p14a: Checking sleep_bw is a dataframe")
})
```

```
## [1] "p14a: Checking sleep_bw is a dataframe"
## Test passed
```

```
test_that("p14b", {
  expect_true(ncol(sleep_bw) == 13)
  print("p14b: Checking sleep_bw has 13 columns")
})
```

```
## [1] "p14b: Checking sleep_bw has 13 columns"
## Test passed
```

```
test_that("p14c", {
  expect_true(nrow(sleep_bw) == 83)
  print("p14c: Checking sleep_bw has 83 rows")
})
```

```
## [1] "p14c: Checking sleep_bw has 83 rows"
## Test passed
```

```
test_that("p14d", {
  expect_true(all.equal(sleep_bw$bodywt_grams[1], 50000, 0.01))
  print("p14c: Checking bodyweight is in grams")
})
```

```
## [1] "p14c: Checking bodyweight is in grams"
## Test passed
```

15. [1 point] Calculate the average sleep time across all the animals in the dataset using a `dplyr` function and label this value `sleep_avg`. Assign this one value to a dataframe called `avg_sleep_time`.

```
. = " # BEGIN PROMPT
avg_sleep_time <- NULL # YOUR CODE HERE
avg_sleep_time
" # END PROMPT

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

```
## # A tibble: 1 x 1
##   sleep_avg
##       <dbl>
## 1      10.4
```

```
# END SOLUTION
```

```
test_that("p15a", {
  expect_true(is.data.frame(avg_sleep_time))
  print("p15a: Checking avg_sleep_time is a dataframe")
})
```

```
## [1] "p15a: Checking avg_sleep_time is a dataframe"
## Test passed
```

```
test_that("p15b", {
  expect_true(ncol(avg_sleep_time) == 1 &&
                        nrow(avg_sleep_time) == 1)
  print("p15b: Checking avg_sleep_time has 1 row and 1 column")
})
```

```
## [1] "p15b: Checking avg_sleep_time has 1 row and 1 column"
## Test passed
```

```
test_that("p15c", {
  expect_true(is.numeric(avg_sleep_time$sleep_avg))
  print("p15c: Checking sleep_avg column is numeric")
})
```

```
## [1] "p15c: Checking sleep_avg column is numeric"
## Test passed
```

```
test_that("p15d", {
  expect_true(all.equal(avg_sleep_time$sleep_avg, 10.43373, tol = 0.01))
  print("p15d: Checking sleep_avg column is 10.4337")
})
```

```
## [1] "p15d: Checking sleep_avg column is 10.4337"
## Test passed
```

16. [2 points] Calculate the average sleep time for each type of "-vore". Hint: you'll need to use two `dplyr` functions! Name the columns of the dataframe `vore` and `sleep_avg` and assign the dataframe to `avg_by_vore`.

```
. = " # BEGIN PROMPT
avg_by_vore <- NULL # YOUR CODE HERE
avg_by_vore
" # END PROMPT

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

```
## # A tibble: 5 x 2
##   vore    sleep_avg
##   <chr>       <dbl>
## 1 carni       10.4
## 2 herbi        9.51
## 3 insecti     14.9
## 4 omni        10.9
## 5 <NA>        10.2
```

```
# END SOLUTION
```

```
test_that("p16a", {
  expect_true(is.data.frame(avg_by_vore))
  print("p16a: Checking avg_by_vore is a dataframe")
})
```

```
## [1] "p16a: Checking avg_by_vore is a dataframe"
## Test passed
```

```
test_that("p16b", {
  expect_true(ncol(avg_by_vore) == 2 &&
                        nrow(avg_by_vore) == 5)
  print("p16b: Checking avg_by_vore has 5 rows and 2 columns")
})
```

```
## [1] "p16b: Checking avg_by_vore has 5 rows and 2 columns"
## Test passed
```

```
test_that("p16c", {
  expect_true(identical(names(avg_by_vore), c("vore", "sleep_avg")))
  print("p16c: Checking column names are vore and sleep_avg")
})
```

```
## [1] "p16c: Checking column names are vore and sleep_avg"
## Test passed
```

```
test_that("p16d", {
  expect_true(all.equal(avg_by_vore$sleep_avg[1], 10.378947, 0.01))
  print("p16c: Checking correct averages")
})
```

```
## [1] "p16c: Checking correct averages"
## Test passed
```

END