biol827_biological_statistics

UNK Biology

2025-01-20

Table of contents

Preface

Welcome to Biology 827 at the University of Nebraska at Kearney! Material in this class was designed by Drs. Greg Pec, Jayne Jonas, and Jacob C. Cooper for use in *R*.

By the end of this course, you should be able to:

Course learning objectives: By the end of this course, students will be able to: 1. Identify best practices for data management, including creation of metadata, to ensure the longevity of datasets. 2. Construct testable hypotheses about biological systems and identify appropriate experimental designs to statistically test those hypotheses. 3. Calculate descriptive statistics to examine the character of a dataset and evaluate statistical power. 4. Conduct and interpret results of standard statistical tests as appropriate for the experimental design used and data collected. 5. Apply statistical concepts in critically evaluating research conducted by others. 6. Communicate research results that are accurately, concisely, and straightforwardly supported by statistics.

This site will help you navigate different homework assignments to perform the necessary *R* tests.

Welcome to class!

Dr. Jacob C. Cooper, BHS 321

1 Module 1

2 Learning objectives:

To reinforce Module 1 learning objectives and gain familiarity with the R statistical framework by using *RStudio* to:

- Set a working directory
- Import a dataset
- Subset dataframes
- Create and evaluate a frequency histogram

3 Before beginning the walkthrough

3.1 Installing *R* and *RStudio*

If you have not already done so, install both of the following free software programs on your computer before beginning the problem set. If you are using a Mac OS, you *may* be prompted to also download and install XQuartz (also free).

- *R* – behind-the-scenes statistical brain - download from *R* directly
- *RStudio Desktop* - software we'll use to run R - download from Posit

On your computer, create a folder dedicated to work done in R and that you will be able to use all semester. Note the file path to this folder, you will need it in this problem set. **Please make sure that this folder is in your UNK OneDrive - this will back up your data in case anything happens, and make it easier to share files with professors and other students.**

- A *filepath* is like an address your computer uses to organize and find all the information stored on your hard drive. Filepaths are defined within *R* to find where files are stored on your computer. Some filepaths will include `~` which refers to the "base directory" or "default directory" on a machine (for Linux-based operating systems). Examples of filepaths follow for the default `Downloads` folder on a machine:

```
# Linux filepath
"~/Downloads/"

# Mac filepath
"~/Downloads/"

# Windows filepath
"C:\\Users\\[YOUR USER NAME]\\Downloads"
```

3.2 Important notes

3.2.1 Notes on formatting

Throughout these instruction documents, different formats will be used to denote what type of object or data we are talking about. For example, programs like *R* will be italicized, and things related to actual code or coding objects will be formatted `like code`. For example, if a dataset is talking about the number of birds and it has a column called "num_birds", then I will make this clear by formatting it as `num_birds`. If we are talking about the mean of a dataset, I will use "mean", but if I'm talking about the command it will be formatted as "`mean`".

3.2.2 Tab completing

RStudio allows you to do "tab completion". Tab completion is a method that helps prevent you from making mistakes, especially as related to formatting or spelling. For example, if I hit one quote within a coding region in *R*, *RStudio* will automatically complete the quotes and place the cursor in the middle. (This is also true for parentheses, brackets, etc.) Quotes, in *R*, indicates that you will be putting in a filepath. Thus, if I type `"~/Dow"` and hit tab, my computer will autocomplete to `"~/Downloads/"`.

3.2.3 File naming

It is often common for folks to use spaces in their file names, like `bird data.csv`. However, different coding languages, such as *Bash* and *Python*, see spaces as a break between commands. For example, where *R* might see `bird data.csv` as a single file name, *Bash* would interpret this as perform the command `bird` on the object `data.csv`. Thus, it is *always* better to use underscores or dashes instead, such as `bird_data.csv`.

3.2.4 Taking notes in *R*

When writing code, it is important to take notes and document each step of what you are doing. In *R*, anything written after a `#` character is ignored. Thus, anything written after a `#` can be used as notes. Compare the following outputs:

```
mean(1:10)
```

```
[1] 5.5
```

```
# mean(1:10)
```

As we can see, the first format returned a value - the mean - whereas the second example did not run. Thus, we can annotate our code like so:

```
# calculate the mean
mean(1:10)
```

```
[1] 5.5
```

If you want to write a lot of notes, enter `#'` before typing notes. Every time you hit `enter`, the next line will start with `#'`.

3.3 Getting help inside *R*

Every function loaded into *R* has a documentation or help page. This is accessible within *R* by typing the function preceded by `?`. For example, if I am not sure what the `mean` function does or how to use it, I would type `?mean`, as shown below:

```
?mean
```

This will return a window such as the following in the *bottom right* (plot) pane of *RStudio*. This is the *R* documentation page. You can scroll through to see argument explanations, examples of use, and information on how to cite that particular command.

You can also get citations as follows:

```
citation()
```

```
To cite R in publications use:

  R Core Team (2024). _R: A Language and Environment for Statistical
  Computing_. R Foundation for Statistical Computing, Vienna, Austria.
  <https://www.R-project.org/>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {R: A Language and Environment for Statistical Computing},
```

```
    author = {{R Core Team}},
    organization = {R Foundation for Statistical Computing},
    address = {Vienna, Austria},
    year = {2024},
    url = {https://www.R-project.org/},
  }

We have invested a lot of time and effort in creating R, please cite it
when using it for data analysis. See also 'citation("pkgname")' for
citing R packages.
```

4 Datasets:

We will be working with two different datasets as part of the *Module 1: Introduction to R* lecture and this *Problem Set Assignment*. These are fictional data associated with hypothetical studies created to illustrate key points. Look at both datasets, available as comma-separated values (`.csv`) files in *Canvas*.

4.1 Dataset 1

A researcher was interested in the relationship between whether more hours in direct sunlight led to more freckles on the back of hands in humans. They found 10 people who volunteered to participate. Each participant (`Subject`) tracked how much sun exposure they had over the course of the summer (`SunExp`). At the end of the summer, they reported the total number of freckles on the backs of both hands (`Freckles`). *Note* that some of these filenames have spaces, as from previous classes; we are working on reformatting filenames to fit the best practices described above.

- `BIOL827.01_Problem_Set_Data_1.csv`

4.2 Dataset 2

After analyzing dataset 1, the researcher decided to conduct a follow-up study. This time they found 26 volunteers and asked each volunteer to sit in the sun for a specific number of hours every day for the entire summer. Thirteen participants were assigned to each sun exposure regime (`SunExp`): 1 or 3 hours per day (2 regimes x 13 participants per regime = 36 participants). At the end of the summer, the researcher counted the total number of freckles (`Freckles`) on the backs of both hands of each participant (`Subject`).

- `BIOL827.01_Problem_Set_Data_2.csv`

5 Procedure:

2. Download both dataset files (`*.csv`) from *Canvas*. Save them to your *R* folder on your computer.

3. Watch *Module 1: Introduction to R* lecture posted on *Canvas*. There is also a Module 1 tutorial video that walks through the procedures below.

5.1 Section A. Set a working directory

The working directory tells *RStudio* where to look for external files and to save files. This eliminates the need to code the entire file path each time a dataset is imported or an *R* file is saved.

4. Open R, you should have 4 panes.

- The upper-left pane is the "Source" pane should be blank with '`Untitled1`' at the top. This is where you will enter code.
  - If you do not have this pane, you can open it by going to: `File > New File > R Script`.
- The "Console" pane (usually bottom left) is where the code and its associated output will be printed after being Run. (*Running* code is the same as executing the code and having it perform the specified actions).
- The "Global Environment" pane (usually upper right) lists the objects in *R*'s working memory. Any time you import a dataset or write output to an object, it should be listed here.
- The "Plots/Viewer" pane (usually bottom right) is where any graphs and help windows will be displayed.

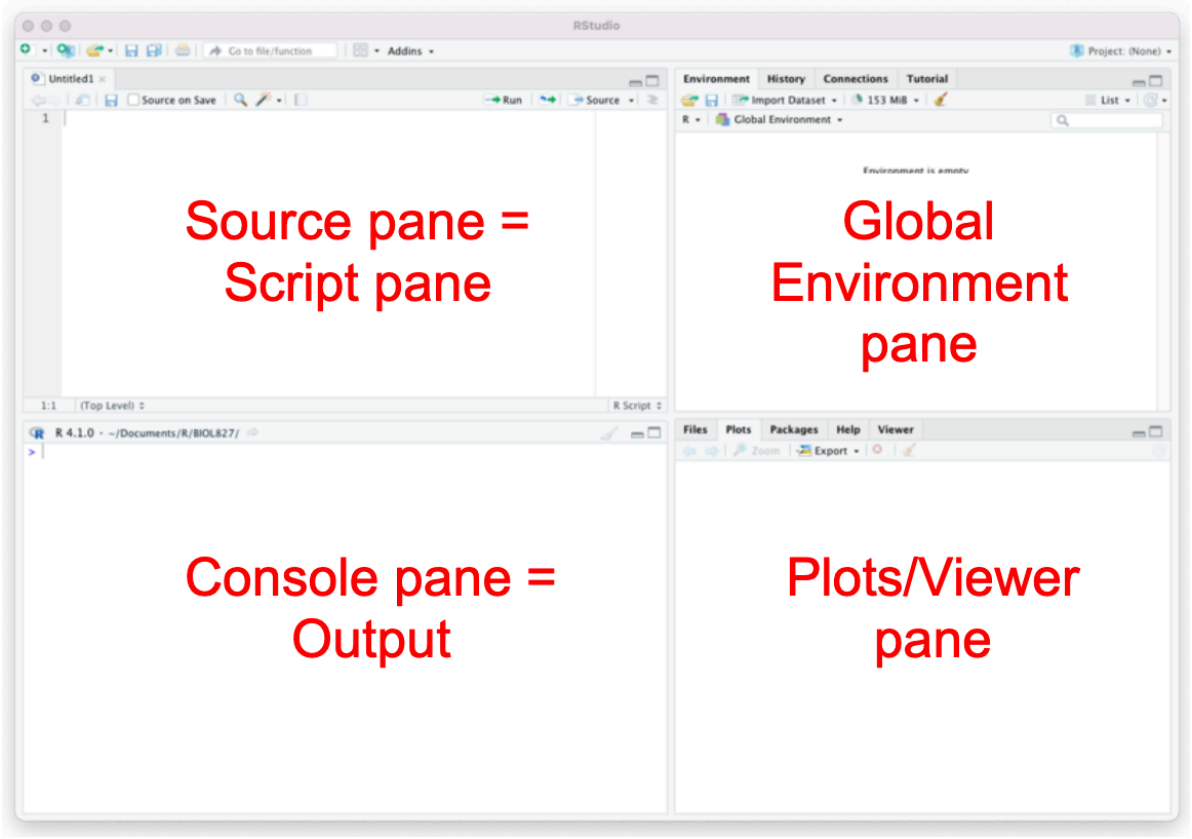


Figure 5.1: The four panes that should appear when you first start RStudio. Note that the top left "Source" pane may be missing; see instructions on opening a new file to initiate this pane.

5. In the source pane, type the command `setwd()`.

- IMPORTANT NOTES:
  - *R* coding is case sensitive! **Use tab completion!**
  - Each new command must begin on a new line.
  - A single command can run onto multiple lines.

```
## EXAMPLES
## INCORRECT: two commands on single line
mean(1:10) var(1:10)

## CORRECT: Each command on a line
mean(1:10)
```

```
var(1:10)

## CORRECT: One command across multiple lines
mean(1,2,3,4,5,
     6,7,8,9,10)
```

6. Inside the parentheses, type the file path to your *R* folder in quotation marks.

- Format of the filepath is operating system dependent. Mac and Linux use `/` to separate portions of the filepath and Windows often uses `\\`. If you are having issues, or two back-slashes (\\) in the path. If one doesn't work, try the other. If both fail, follow these instructions to create a filepath on your computer.
- R reads a single backslash (\) as an "escape" command, so you will receive an error. Escape commands are used frequently in coding languages; for example, `&` may indicate a joining of two objects but `\&` denotes the character "&".
- **Remember, use Tab Complete! This will put the correct format for you.**

```
# Setting the working directory
# Format is shown for Mac
setwd("~/Documents/BIOL827/")
```

7. Click "Run", or place your cursor on the line and click `ctrl`+`enter` (Windows / Linux) or `cmd`+`enter` (Mac) to make the line run

- To run only a portion of the code, highlight that section of code then Run.

8. We can also save the working directory as an *object*. We can do this as follows:

```
filepath <- "~/Desktop/BIOL827/"

filepath
```

```
[1] "~/Desktop/BIOL827/"
```

- After running both lines of code, you will see that `filepath` will appear in the top right pane. This is now in our memory! Every time we type the word `filepath` and have it run, it will print out `"~/Documents/BIOL827/"` or whatever the equivalent is on your machine.

9. See what is in the `filepath` folder. If you have saved both the datasets for this class into that folder, it should look similar to the following:

```
list.files(filepath)
```

```
[1] "BIOL827.01_Problem_Set_Data_1.csv" "BIOL827.01_Problem_Set_Data_2.csv"
```

- We can see a list of all files at the address designated by `filepath`. Using `filepath` ensures that our code will always work, and that we are not reliant on being in the correct working directory with `setwd`. This will become useful when using `rmarkdown`.

5.2 Section B. Import a dataset

10. First, we need to tell *RStudio* a short name for the dataset we are importing. This is called an object name, such as was shown for `filepath` in step 9.

- This object name will be used whenever we want *R* to do something with this dataset. Object names can be anything you choose, but should be short, descriptive, and a single string (**no_spaces**) of characters. The object name ***cannot*** begin with a numeral.
  - I called it object "`Data1`". If you used a different name, be sure to use the name you used in place of `Data1` throughout the instructions.
- In *R*, a "dataframe" is a set of variables arranged in columns with each unique variable in a single column and information associated with a sample unit in a single row. A `data.frame` is also a very specific object type in *R* that can be passed through to certain commands. The dataset we are importing is considered a `data.frame`.

11. Next, type `<-`. In *R* coding, these characters (`<-`) mean "is created from". I also remember as "put this value into this thing".
12. Now, we need to tell RStudio what data we want to import. Because the data is in a comma-separated values (`*.csv`) format, we need to use the function `read.csv`. Syntax for this code is: `read.csv("filename.csv", header = TRUE)`.

- R will look in the *working directory* to find the filename indicated. If the working directory has not been set as directed in Section A. Set a working directory, you will need to include the full filepath with the filename in quotations. There are several ways to do this, as shown below.
- The option "`header = TRUE`" tells *R* that the first row in the `*.csv` file contains the name for each variable in the dataset.
- **Don't forget to use tab complete!**
- Windows computers: If you are using RStudio on a Windows operating system, you may need to include another argument to ensure the *.csv file is read into *RStudio* correctly: `read.csv("filename.csv", header = TRUE, fileEncoding = "UTF-8-BOM")`

13. Highlight and run the code, or place your cursor on the line and hit `ctrl+enter` or `cmd+enter`. This code is telling *RStudio* to import data from the indicated `*.csv` file into *RStudio* using the first row of that file to name each variable (= column). *RStudio* will store the data as an object called `Data1`.

- Whenever we want to do something with this data, we will need to tell *RStudio* to use `Data1`.

```
## Steps 10 through 13 shown here
Data1 <- read.csv("~/Desktop/BIOL827/BIOL827.01_Problem_Set_Data_1.csv")

# view first few rows
head(Data1)
```

```
   Subject     SunExp Freckles
1    James  4.6000000        4
2    Keiko  8.5166667       20
3 Mauricio 14.3500000        3
4   Sharon  6.4166667        9
5    Sonia  8.9500000       13
6  Apoorva  0.1333333       12
```

```
# read the file, but use filepath
# use "paste0" to combine things
# paste0 means "combine, no spaces"
# paste sometimes works with tab complete - not always!

Data1 <- read.csv(paste0(filepath,"BIOL827.01_Problem_Set_Data_1.csv"))

# view first few rows
head(Data1)
```

```
   Subject     SunExp Freckles
1    James  4.6000000        4
2    Keiko  8.5166667       20
3 Mauricio 14.3500000        3
4   Sharon  6.4166667        9
5    Sonia  8.9500000       13
6  Apoorva  0.1333333       12
```

We can also view what *kind* of data this is in *R*:

```
# str = structure
str(Data1)
```

```
'data.frame':   10 obs. of  3 variables:
 $ Subject : chr  "James" "Keiko" "Mauricio" "Sharon" ...
 $ SunExp  : num  4.6 8.52 14.35 6.42 8.95 ...
 $ Freckles: int  4 20 3 9 13 12 14 12 7 6
```

As mentioned above, *R* has created a `data.frame` with these data with three columns imported: `Subject`, `SunExp`, and `Freckles`.

14. View the imported dataset in *RStudio* by clicking on the object name from the list in the environment panel.

- This will show the format and specific setup of the object in your *R* environment.
- Double-clicking this object should open a new tab with a spreadsheet view of the dataset.

5.3 Section C. Subset a dataframe

When working in R, we often want to analyze a subset of observations (= rows) in the `data.frame`. There are many ways to do this, for now we will use the subset() function to create subsets as new objects. The subset function is formatted like so:

`New.dataframe <- subset(dataframe, rows to keep, columns to keep)`

Rows should be identified using criteria specified by a logical argument.

- The logical argument should tell R which variable contains the criteria and values to keep. See examples in Section C.2. Subset observations only.
- If criteria are only given for rows, a comma is not necessary after the criteria.

This will return all variables in the new subset. Variable names must be in quotations and exactly match the spelling and capitalization of the name as given in the dataframe. **Use tab complete when possible, some functions allow for this within the function.**

- More than one column can be selected using `select=` and the `list` function `c("variable 1", "variable 2")`. `c` stands for "concatenate" to combine things into a single object.

```r
# concatenation examples
example_1 <- c("a","b")
example_2 <- c(5,7)

example_1
```

```
[1] "a" "b"
```

```r
example_2
```

```
[1] 5 7
```

- If only subsetting columns and keeping all rows of data, two commas should be between the full `data.frame` name and column variable names so that the row portion of the function is empty. This tells *R* to keep all rows.

5.3.0.1 Anatomy of an *R* command

There are three basic parts to an *R* command.

1. **Object**: set of information (value, variable, dataset, model, etc.) *R* can work with
2. `<-`: separates the object and function; means "is created from".
3. **Function**: tells R to do something.

Command:

```
Data1 <- read.csv("BIOL827.01_Problem_Set_Data_1.csv", header = TRUE)
```

How R reads the command:

Create a new object called `Data1` by importing the `.csv` file into *R*. Use the first row of the `.csv` file to give a variable name to each column.

5.3.1 Section C.1. Subset by columns

Create a new dataframe object containing all observations of the sun exposure variable (`SunExp`) only from existing dataframe `Data1`.

15. Begin the command by providing a name of your choosing for the new dataframe to be created by subsetting `Data1`. For example, `SunExp.Data1` since the new variable will contain only the sun exposure variable.
16. On the same line, enter `SunExp.Data1 <- subset(Data1, , "SunExp")`

- `subset()`: this calls the `subset` function; all arguments related to how we want to subset the dataframe must be made inside these parentheses.
- `Data1`: this is the name of the existing dataframe with information from which the new `data.frame` will be made.
- Two commas (with or without a space between): indicates we are not subsetting by rows. That is, all rows of data will be in our new `data.frame`.
- `SunExp`: this is the name of the variable in `Data1` that we want to copy into the new `data.frame`.

```
# subset data frame
SunExp.Data1 <- subset(Data1, , "SunExp")

# view first few rows
head(SunExp.Data1)
```

```
      SunExp
1  4.6000000
2  8.5166667
3 14.3500000
4  6.4166667
5  8.9500000
6  0.1333333
```

Another way to write this for annotation would be:

```
# subset data frame
SunExp.Data1 <- subset(Data1, # data frame
                       , # row filter condition
                       "SunExp") # column filter condition

# view first few rows
head(SunExp.Data1)
```

```
      SunExp
1  4.6000000
2  8.5166667
3 14.3500000
4  6.4166667
5  8.9500000
6  0.1333333
```

17. Run the above `subset` command.
18. To view contents of the new `data.frame` in the console panel, highlight or type the new dataframe name and click Run.

- Quickly and easily highlight a portion of a line by double-clicking on a word. Double clicking will highlight the object or command name only, which you can then run by clicking "Run" or using `ctrl`+`enter` or `cmd`+`enter`.

5.3.2 Section C.2. Subset observations only

Create a different new `data.frame` from `Data1`. This new `data.frame` should contain all variables but only for participants with 20 or more freckles. Selecting a subset of rows requires a logical statement (*i.e.*, criteria) to let *R* know which rows to copy to the new `data.frame`.

19. Begin the command by providing an object name of your choosing for the new `data.frame`. For example, `Freck20.Data` since we want data from participants with 20 or more freckles in the new `data.frame`.

- Remember, object names cannot have spaces nor begin with a numeral.

20. Enter the function `<- subset(Data1, Freckles >= 20)` after the object name.

- `subset()`: this calls the subset function; all arguments related to how we want to subset the `data.frame` must be made inside these parentheses.
- `Data1`: this is the name of the existing `data.frame` with information from which the new `data.frame` will be made.
- `Freckles >= 20`: this is a logical argument. It tells *R* to look in variable `Freckles` and copy only rows for which the value of `Freckles` is 20 or greater to the new `data.frame`.

```
# subset by rows
# col argument not needed!
Freck20.Data <- subset(Data1, Freckles >= 20)

Freck20.Data
```

```
  Subject   SunExp Freckles
2   Keiko 8.516667       20
```

21. Highlight and run the code, as above.
22. To view contents of this new dataframe in the console panel, highlight or type the new `data.frame` name and click Run.

5.3.3 Section C.3. Subset by both variables and observations

Create a third new dataframe from `Data1`. This new dataframe should contain only the `Subject` and `SunExp` variables and only for participants with 20 or more freckles.

23. Begin the command by providing an object name for the new dataframe.
24. On the same line, enter: `<-subset(Data1, Freckles >= 20, select = c("Subject", "SunExp"))`.

    - `subset()`, `Data1`, and `Freckles >= 20` are the same as described in Section C.2. Subset observations only.
    - `select = c("Subject","SunExp")`: this uses the list function `c()` to select the two variables, `Subject` and `SunExp`, that we want to copy into the new dataframe. Be sure the variable names are in quotations and spelled in the same case as in the `Data1` dataframe.

25. Highlight and run the command.

```
Freck.var.obs.filter <- subset(Data1, Freckles >= 20,
                               select = c("Subject", "SunExp"))
```

25. To view contents of the new dataframe in the console panel, highlight or type the new dataframe name and click Run.

```
Freck.var.obs.filter
```

```
  Subject   SunExp
2   Keiko 8.516667
```

27. Compare the dataframes created in Section C.2. Subset observations only. and Section C.3. Subset by both variables and observations. Notice that the `Freckles` variable is in dataframe `Freck20.Data1` created in Section C.2. Subset observations only, but not in dataframe `SunExp.Freck20.Data1` created in Section C.3. Subset by both variables and observations because it was not included in the select argument in Step 23.

28. Save your R script file before quitting by clicking the disk icon or by selecting File > Save from the menu at the top of the screen.

5.4 Section D. Create frequency histograms

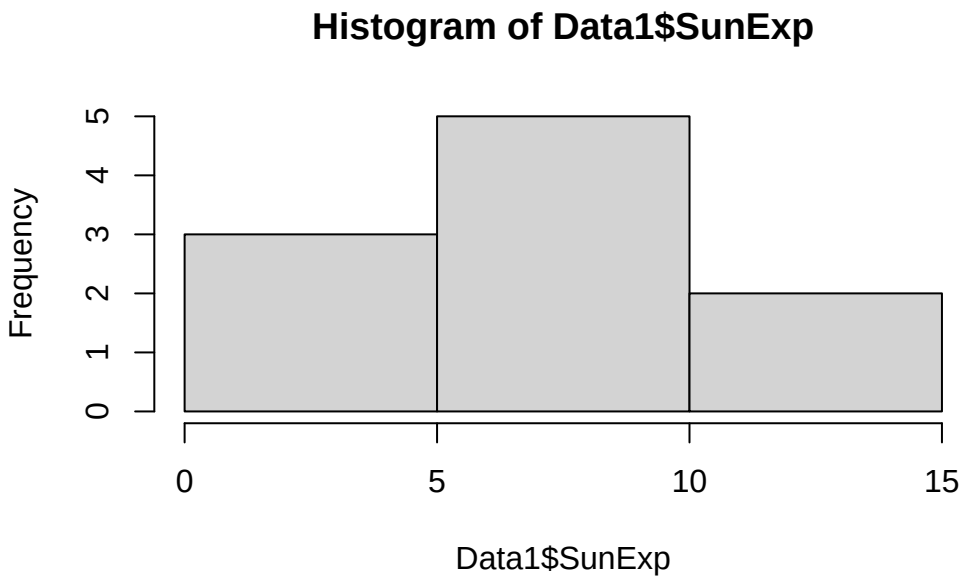
Create a frequency histogram for sun exposure and freckle variables from Data1 to evaluate the frequency distribution for each variable. Here, we will use the hist() function and its default settings to get familiar with the structure and content of histograms.

29. Enter the function: `hist(Data1$SunExp)`. This tells *R* to create a histogram from observations of the `SunExp` variable within the `Data1` dataframe.

    - `hist()`: calls the histogram plotting function; all arguments must be made inside these parentheses.
    - `Data1`: this is the name of the dataframe from which the histogram will be made.
    - `$`: separates an object name from the variable name. The named variable must exist within the named object or you will receive an error from *RStudio*.
    - `SunExp`: identifies the variable in `Data1` to be used for the histogram. A variable with this name must exist within the dataframe identified.

30. Highlight and run the code.

```
hist(Data1$SunExp)
```



31. Repeat step 28 replacing `SunExp` with `Freckles` to create a histogram for the `Freckles` variable within `Data1`.

Basic operators in *R*

- `+` add `^` or `**` exponentiation
- `-` subtract
- `<` Less than
- `*` multiply
- `<=` Less than or equal to
- `/` divide
- `>` Greater than
- `==` equals
- `>=` Greater than or equal to
- `!=` does not equal
- `x|y` `x` or `y` **see note below*
- `!x` is not `x`
- `x&y` `x` and `y` **see note below*
- `*x|y` and `x&y`: `x` and `y` can represent expressions. For example, `SunExp == 1 & Freckles >= 20` could be used to select only those observations meeting both criteria.

32. Highlight and run the code.
33. Both histograms should now be available in the plots panel (lower right).

- To navigate between histograms for `Freckles` and `SunExp`, click the arrows at the top of the plots panel.
- To save graphs as an image file or copy them to paste into a *Canvas* quiz response, click Export (Figure 9, red arrow) and follow the corresponding prompts.

34. Self-assessment. Compare information displayed in each frequency histogram to the raw data for each variable in the Data1 dataframe (or by opening the `.csv` file outside of *RStudio*).

- Looking at the sun exposure variable in the dataframe, how many values between 0 and 5 does it contain?
- Now look at the histogram for sun exposure, what frequency is graphed for the 0-5 bin? How does they compare? Hint: they should be the same.

6 Problem Set Assignment Directions and Questions:

6.1 1. (2 points) In Dataset 1, identify which variable is the response variable and which isthe explanatory variable as well as what type of data each represents (categoricalnominal, categorical ordinal, numeric discrete, numeric continuous). In Canvas, therewill be dropdowns for you to use to identify the type of variable and the type of dataassociated with each of the following.

6.1.1 a. Sun exposure (SunExp)?

6.1.2 b. Freckles (Freckles)?

6.2 2. (2 points) In Dataset 2, identify which variable is the response variable and which is the explanatory variable as well as what type of data each represents (categorical nominal, categorical ordinal, numeric discrete, numeric continuous). In Canvas, there will be dropdowns for you to use to identify the type of variable and the type of data associated with each of the following.

6.2.1 a. Sun exposure (SunExp)?

6.2.2 b. Freckles (Freckles)?

6.3 3. (2 points) Based on descriptions of the studies associated with each dataset, answer the following questions. In Canvas, there will be dropdowns for you to use to identify the type of variable and the type of data associated with each of the following.

6.3.1 a. Was Dataset 1 associated with an observational or experimental study?

6.3.2 b. Was Dataset 2 associated with an observational or experimental study?

6.4 4. (6 points) Using Dataset 2, create one histogram of freckles data for participants exposed to 1 hour of sun daily and another for freckles of participants exposed to 3 hours of sun daily. You will be asked to paste screenshots or insert images of both histograms in Canvas.

(a Canvas 'quiz'). Questions in Canvas may be worded slightly differently than here, usually for brevity, but with the same meaning. If there is a discrepancy in what is being asked between this document and the question form in Canvas, answer based on what is asked in this document. Also, please let me know as soon as possible so I can get it fixed.

Additional resources for R:

If you get stuck performing tasks in R, please reach out to me. There is also a wealth of information, pointers, and discussion boards about R language and RStudio online. Performing an internet search often provides insight and may offer more immediate assistance. Here are a few (of many!) websites that tend to be reliable and helpful:

- RDocumentation
- R-bloggers
- Stackoverflow
- STHDA.com
- GitHub

The following sites are also sometimes useful:

- ChatGPT
- Reddit

7 Module 2

8 Learning objectives:

To continue gaining familiarity with the R statistical framework by using *RStudio* to:

- Create new variables from existing variables
- Calculate simple descriptive statistics
- Install and activate add-on *R* packages
- Use an add-on *R* package to calculate descriptive statistics

9 Datasets:

We will be working with another fictional dataset. These data were created at random to illustrate *R* tasks. Any patterns in these data are random and unintentional. The dataset is available as a comma-separated values (`.csv`) file in *Canvas*.

9.1 Dataset

A researcher was conducting an exercise physiology study. The research team wanted to know whether volunteers of different sexes (female, male, non-binary) differed in physical characteristics, especially Body Mass Index (BMI). For each volunteer, the research team recorded their weight (in kilograms, kg) and height (in meters, m) of each participant.

- BIOL827.02_Problem_Set_Data.csv

10 Procedure:

1. Watch **Module 2: Manipulating and Describing Data in *R*** for a walk-through of these initial procedures.
2. Open a new *R* script in the source pane by clicking on `File menu > New File > R Script`. If so desired, you can also select `Rmarkdown` to create an `rmarkdown` script instead. **Save your script frequently as you work through the problem set (`File menu > Save` or CTRL + S** [Windows] **or CMD + S** [Mac]**).**
3. If necessary, clear the environment and plots panes of your previous work by clicking the broom icon at the top of both panes and clear the console by clicking on `Edit menu > Clear Console`. You can also hit CTRL + L (all operating systems).
4. Set your working directory and import the dataset. Review Module 1 Problem Set procedures as necessary to complete these tasks.
    - I imported the dataset as a dataframe named 'Data'.

Note that I show this intro of loading the file and packages here, but these steps may not be shown in future modules. Below, I use `tidyverse`. If you do not have `tidyverse`, you will have to run `install.packages("tidyverse")` on your machine. This command only needs to be run *once*. If it asks to create a new folder or something, just say "Yes".

```
# load package for data manipulation
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
# where are my files?
# REMEMBER - this is for MAC!
filepath <- "~/Desktop/BIOL827/"

# load my file
# NOTE you will get a multicolored output
# this just tells you about your data
Data <- read_csv(paste0(filepath,
                        "BIOL827.02_Problem_Set_Data.csv"))
```

```
Rows: 30 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Sex
dbl (3): Participant, Weight.kg, Height.m

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.
```

10.1 A. Calculate descriptive statistics

Descriptive statistics refers to simple calculations that summarize or describe a given variable, such as the mean (i.e., average, μ, $\bar{x}$), measures of variation in the data (standard deviation [σ, s], variation, etc.), and spread (minimum, maximum, range, etc.). These values are also one of the tools we can use as part of quality assurance and quality control.

5. To calculate the mean of the `Weight.kg` variable over all sample units, use `mean(Data$Weight.kg)`. This tells *R* to calculate the mean from observations of the `Weight.kg` variable within the `Data` dataframe.

- `mean()`: calls the mean function.
- `Data`: this is the name of the `dataframe` in which the data to be averaged are stored.
- `$`: separates an `object` name from the `variable` name. The named variable *must* exist within the named `object` or you will receive an error from *RStudio*.
- `Weight.kg`: identifies the variable in `Data` from which the average is to be calculated. A variable with this name must exist within the dataframe identified.

```
# view data file - first rows ONLY
head(Data)
```

```
# A tibble: 6 x 4
  Participant Sex    Weight.kg Height.m
        <dbl> <chr>      <dbl>    <dbl>
1          16 Female      61.8     1.87
2          20 Female      64.2     1.89
3          29 Female      64.7     1.76
4          28 Female      67.6     1.86
5           5 Female      71.2     1.96
6           7 Female      72.5     1.8
```

```
mean(Data$Weight.kg)
```

```
[1] 78.929
```

We will need to round values for our assignments.

- `round`: rounds values to a certain number of decimal places. Format is `round(Data, decimals)`.
- `%>%`: "is passed through". The answer from the previous command is put into the next command automatically, and does not need to be declared. This makes coding easier. `round(mean(Data$Weight.kg), 2)` is the same as `mean(Data$Weight.kg) %>% round(2)`. This helps make things easier to follow when lots of commands are used in a row.

```
# round to significant figures
mean(Data$Weight.kg) %>% # get the mean
  round(2) # round to two decimals
```

```
[1] 78.93
```

6. To calculate the standard deviation of the `Weight.kg` variable over all sample units, use `sd(Data$Weight.kg)`. The syntax is similar to the `mean()` function.

```
sd(Data$Weight.kg)
```

```
[1] 16.60132
```

```
# round to two decimal places
sd(Data$Weight.kg) %>%
  round(2)
```

```
[1] 16.6
```

10.2 B. Install an R add-on package

Add-on packages only need to be installed one time. They do not need to be installed each time you open *RStudio*, but packages do have to be activated each time you open *RStudio*. Packages only need to be re-installed when the package developer releases an updated version; you will usually receive a prompt in *RStudio* if/when a package needs to be reinstalled.

The add-on package `pastecs` contains a very handy function to create a large set of descriptive statistics all at once. There are other packages that also can calculate sets of descriptive statistics, but this is the best one I have found to date. Although there is a command-line function for installing packages, this generally only needs to be done once so I prefer to use the installation wizard rather than building it into my script.

7. Type the following into your *R* coding pane. DO NOT put this into your main coding document; your code *will not run* if it is constantly asked to install things.
8. Enter the following text:

```
install.packages("pastecs", dependencies = TRUE)
```

- Package must be in quotes `" "`.
- `dependencies = TRUE` ensures any other needed packages are also installed.

9. After running the code, you should see a lot of output in the code window. **Don't panic**. It is installing a lot. You may be prompted to install extra things or create a new folder; just say "Yes". **See below for an example output from an re-install:**

```
trying URL 'https://cran.rstudio.com/bin/macosx/big-sur-arm64/contrib/4.4/pastecs_1.4.2.tgz'
Content type 'application/x-gzip' length 487644 bytes (476 KB)
==================================================
downloaded 476 KB


The downloaded binary packages are in
    /var/folders/gc/0jz_0k0j2qq75h6ynd6jtb_5xh27x8/T//Rtmp1kkunq/downloaded_packages
```

10. Alternatively, you an go to the "Packages" tab in the plot pane (bottom right) and find `pastecs` there for installation.

11. Whichever method is chosen, you can check installation (and find the citation!) by typing the following:

```
citation("pastecs")
```

```
To cite package 'pastecs' in publications use:

  Grosjean P, Ibanez F (2024). _pastecs: Package for Analysis of
  Space-Time Ecological Series_. R package version 1.4.2,
  <https://CRAN.R-project.org/package=pastecs>.

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {pastecs: Package for Analysis of Space-Time Ecological Series},
    author = {Philippe Grosjean and Frederic Ibanez},
    year = {2024},
    note = {R package version 1.4.2},
    url = {https://CRAN.R-project.org/package=pastecs},
  }
```

10.3 C. Activate an add-on package

An add-on package must be installed before it can be activated (B. Install an R add-on package). Although installation only needs to be done once until a major update is made, you must activate each time you open *RStudio*.

12. In the source pane, run the code `library(pastecs)`.

```
library(pastecs)
```

```
Attaching package: 'pastecs'

The following objects are masked from 'package:dplyr':

    first, last

The following object is masked from 'package:tidyr':

    extract
```

Note there are `masked` warnings. This means that the computer has two commands with the same name for the same package. Imagine if you will the word "solar". In English, this pertains to things related to the sun, but in Spanish this refers to the ground or land. Thus, the computer may assume which one is meant, but this could result in a nonsensical command. Thus, if you have a conflict, you can fix it by designating the package, in this case, `English::solar` and `Spanish::solar`. We will try to let you know if a conflict like this will arise!

13. That's it! You've now got all the additional functions included in this package at your fingertips. So, let's see what it can do!

10.4 D. Use function stat.desc to calculate descriptive statistics

A useful function provided in the `pastecs` package is `stat.desc()` which can return a large set of descriptive statistics quickly and with ease.

10.4.1 D.1. Commands on a single variable

14. Enter an `object.name <-` before your `stat.desc` command. The object name can be of your choosing, this is where output from the `stat.desc()` function will be stored.

15. Enter and run the function `stat.desc(Data$Weight.kg)`.

- `stat.desc()`: this calls the `stat.desc` function; all arguments must be made inside these parentheses.
- `Data`: this is the name of the existing dataframe containing the variable to be described.
- `$Weight.kg`: indicates the variable to be summarized.

```
weight_stats <- stat.desc(Data$Weight.kg)

print(weight_stats)
```

```
     nbr.val     nbr.null       nbr.na          min          max        range
  30.0000000    0.0000000    0.0000000   51.3200000  118.3100000   66.9900000
         sum       median         mean      SE.mean CI.mean.0.95          var
2367.8700000   73.6750000   78.9290000    3.0309730    6.1990358  275.6039197
     std.dev     coef.var
  16.6013228    0.2103324
```

Within `stat.desc`, there are multiple options. The defaults are:

- `basic = TRUE`: Return basic statistics on dataset size? `TRUE` = Yes.
- `desc = TRUE`: Return basic statistics describing the data themselves? `TRUE` = Yes.
- `norm = FALSE`: Return normal distribution statistics related to the data? Things like skewness etc. `FALSE` = No.
- `p = 0.95`: What confidence level should we use? Default is 95%.
- This is based on code by Frédéric Ibanez & Philippe Grosjean; type `?stat.desc` into your bottom left pane to learn more about it.

Abbreviation	Meaning
`nbr.val`	Number of observations containing a number value.
`n`	Sample size.
`median`	Median value among all observations. Half of values are lower and half of values are higher than this value.
`nbr.null`	Number of empty observations mean Mean value across all observations.
`nrb.na`	Number of observations for which data were identified as missing (`NA`).
`se.mean`	Standard error relative to the mean across all observations. Interpret as `mean` ± `se.mean`.
`min`	Minimum value across all observations.
`ci.mean.0.95`	95% confidence interval relative to the mean across all observations (i.e., margin of error). Interpret as `mean` ±`ci.mean.0.95`.
`max`	Maximum value across all observations.
`var`	Variance across all observations.
`range`	Range (maximum – minimum) of values across all observations.
`std.dev`	Standard deviation relative to the `mean` across all observations. Interpret as `mean` ± `st.dev`.
`sum`	Sum of values across all observations.
`coef.var`	Coefficient of variation.

Keep in mind that we also have a few different ways to define columns to be selected. For example:

```
Data$Participant
```

```
 [1] 16 20 29 28  5  7  1 14 13 22 26 30  9 24 11 15 19  4 23  8  3  6 27 25 21
[26] 12 17 10  2 18
```

Above we have returned the data for a single column. But what about multiple columns?

```
Data %>%
  select(Participant, Sex) # column names
```

```
# A tibble: 30 x 2
   Participant Sex
         <dbl> <chr>
 1          16 Female
 2          20 Female
 3          29 Female
 4          28 Female
 5           5 Female
 6           7 Female
 7           1 Female
 8          14 Female
 9          13 Female
10          22 Female
# i 20 more rows
```

```
Data[ , c("Participant", "Sex")]
```

```
# A tibble: 30 x 2
   Participant Sex
         <dbl> <chr>
 1          16 Female
 2          20 Female
 3          29 Female
 4          28 Female
 5           5 Female
 6           7 Female
 7           1 Female
 8          14 Female
 9          13 Female
10          22 Female
# i 20 more rows
```

As we can see above, both methods subset the data similarly.

16. To view contents of the object in the console panel, highlight or type the object name and click Run or hit CTRL + Enter (Linux, Windows) or CMD + ENTER (Mac).

```
weight_stats
```

```
     nbr.val     nbr.null       nbr.na          min          max        range
  30.0000000    0.0000000    0.0000000   51.3200000  118.3100000   66.9900000
         sum       median         mean      SE.mean CI.mean.0.95          var
2367.8700000   73.6750000   78.9290000    3.0309730    6.1990358  275.6039197
     std.dev     coef.var
  16.6013228    0.2103324
```

17. Repeat steps 17 through 20 to run the `stat.desc` function for the `Height.m` variable

10.4.2 D.2. Multiple variables in one command

We can also use `stat.desc()` to calculate descriptive statistics for more than one variable at a time.

18. Enter an `object.name <-`. The object name can be of your choosing that will store output from the `stat.desc()` function.

19. Enter the function `stat.desc(Data[, c("Weight.kg", "Height.m")])`.

- `stat.desc()`: this calls the `stat.desc` function; all arguments must be made inside these parentheses.
- `Data`: this is the name of the existing dataframe containing the variables to be described.
- `[, c("variable1", "variable2")]`: similar to the syntax for the `subset` function (Module 1 Problem Set), we can use hard brackets to specify a group of observations and/or variables on which to run the function. Because we want to run the function on all observations, we use a comma after the first hard bracket. Next, we can use the list function `c()` as we did with the `subset` function in Module 1 Problem Set to identify all the variables we want summarized.

20. Highlight the command and click Run. To view contents of the object in the console panel, highlight or type the object name and click Run. You can also use your keyboard shortcuts.

```
multicol_stats <- stat.desc(Data[, c("Weight.kg", "Height.m")])

multicol_stats
```

```
                Weight.kg     Height.m
nbr.val        30.0000000 30.000000000
nbr.null        0.0000000  0.000000000
nbr.na          0.0000000  0.000000000
min            51.3200000  1.700000000
max           118.3100000  1.970000000
range          66.9900000  0.270000000
sum          2367.8700000 55.140000000
median         73.6750000  1.815000000
mean           78.9290000  1.838000000
SE.mean         3.0309730  0.012405042
CI.mean.0.95    6.1990358  0.025371159
var           275.6039197  0.004616552
std.dev        16.6013228  0.067945211
coef.var        0.2103324  0.036966927
```

10.4.3 D.3. Calculate descriptive statistics for only one group of observations

We need to calculate descriptive statistics separately for each sample. There are different ways to do this in *RStudio*. This is where `tidyverse` pipelines come in handy!

In this dataset, we need to run descriptive statistics separately for each `sex` group (female, male, non-binary). Here, we will demonstrate the process for the non-binary sample.

21. Begin the command by providing an `object.name <-` for the new `dataframe` you will create. You can use an object name of your choosing.
22. We are going to use a pipeline `%>%` series of commands. These will:

    - Start with the `Data` object
    - Pass this through a `filter` command to select only certain data
        - Remember - `==` means "is exactly equal to"
    - Pass this through a `select` command to *remove* non-numeric columns
    - Pass this through `stat.desc` to get our stats information.

23. Run the command below. To view contents of the new dataframe in the console panel, highlight or type the new dataframe name and click Run or use your shortcuts.

```
Data %>%                            # start with dataset
  filter(Sex == "Female") %>%       # select females
  select(-Participant, -Sex) %>%    # remove unneeded columns
  stat.desc() %>%                   # get stats
  round(2)                          # round to two decimals
```

```
             Weight.kg Height.m
nbr.val          10.00    10.00
nbr.null          0.00     0.00
nbr.na            0.00     0.00
min              61.75     1.76
max              91.73     1.96
range            29.98     0.20
sum             734.29    18.51
median           71.85     1.86
mean             73.43     1.85
SE.mean           3.08     0.02
CI.mean.0.95      6.97     0.05
var              95.03     0.00
std.dev           9.75     0.07
coef.var          0.13     0.04
```

24. There we go! We have been able to perform a series of complicated commands quickly, right in a row, by using the pipeline method. This allows us to see things step-by-step through the command, and to annotate appropriately.
25. We can save the above as an object by adding `object.name <-` before the first `Data`.

```
female_data <- Data %>%             # start with dataset
  filter(Sex == "Female") %>%       # select females
  select(-Participant, -Sex) %>%    # remove unneeded columns
  stat.desc() %>%                   # get stats
  round(2)                          # round to two decimals
```

26. To view this, just run `female_data` or use `print(female_data)`.

10.5 E. Create a new variable

Although the research team recorded the height and weight of each participant, they are really interested in the Body Mass Index (BMI) profile of participants. We can have *R* calculate BMI and add it to the `Data` dataframe as a new variable. There are separate formulas for

calculating BMI depending on whether weight and height are in metric units or in English units; BMI will be the same from either formula. The formula for BMI calculated from weight and height in metric units is:

$$BMI = \frac{mass}{height^2}$$

27. In this step, how we name our new object matters! We need to tell *RStudio* the dataframe in which to put the new variable and the name we want it to give to the new variable. Enter the object name as: `Data$BMI <-`.

- `Data`: This identifies the dataframe to which we want the new variable added; in our case, this is named `Data`.
- `$`: Tells *R* that what comes next is the name of the new variable.
- `BMI`: This is the name of the new variable to be added to existing dataframe `Data`.

28. Next enter the mathematical formula (see BMI equation above) for calculating `BMI` with reference to weight and height variables in the existing dataframe `Data`: `Data$Weight.kg/(Data$Height.m^2)`.

- This is also an example of how we can use *R* as a calculator.
- When entering equations, it is best to always use parentheses to ensure the correct order of operations (e.g., PEDMAS). For example: BMI of participant 2:
  - should be $\frac{125.64}{1.922^2} = \frac{125.64}{3.69} = 34.08$
  - should ***not*** be $\frac{125.64}{1.92}^2 = 65.44^2 = 4282.07$

29. Highlight the command and click Run.

30. To view contents of the dataframe 'Data' showing the BMI variable added, highlight or type the object name 'Data' and click Run (Fig. 7).

11 Problem Set

***See the Module 1 portion of the problem set for questions 1 – 5 (Q1-Q5).**

11.1 6. (6 points) Create new variables in the dataframe Data (or whatever object name you gave the imported *.csv file): weight and height in English units.

11.1.1 a. You will be asked to insert an image or screenshot of the dataframe showing they have been added and correctly calculated.

11.1.2 b. Conversions:

- Weight: $1kg = 2.20462lbs$
- Height: $39.3701in = 1m$

11.1.3 c. Coding how-to:

- `Data$Weight.lbs`: `Data$Weight.kg*2.20462`
- `Data$Height.in`: `Data$Height.m*39.3701`

11.2 7. (6 points) Use the `stat.desc()` function to calculate descriptive statistics for `Height.in` and `Weight.lbs` separately for each sample (`Female`, `Male`, `NonBinary`).

11.2.1 a. You will be asked to insert an image or screenshot of the output from the *RStudio* console window in *Canvas*.

11.2.2 b. Hint: You will need to use skills from both the Module 1 and Module 2 portions of the problem set to complete this task in *RStudio*.

11.3 8. (3 points) Look at your *R* scripts from Module 1 and Module 2 portions of this Problem Set. Evaluate how well you did or did not follow best practices for documenting your workflows. Briefly discuss improvements that could be made to the way you document your analysis workflow in future problem sets, if any?

12 Optional extra-credit (+3 points possible):

12.1 9. Create a histogram for each of the following variables in the sample indicated:

- Female Weight.lbs
- Male Height.in
- Non-binary BMI

Compare the histogram to the mean and median of the respective sample calculated using the `stat.desc` function for question 5 above. How, if at all, does the histogram illustrate the relationship of the mean to the median of that sample?

13 Submitting your work for the Modules 1 and 2 Problem Set Assignment:

Enter your responses and materials for each item in the corresponding area of the Modules 1 and 2 Problem Set Assignment quiz page. Having the assignment submitted in this way allows me to be much more consistent by grading each question individually across all students in the course before moving on to grading the next question. The Problem Set Assignment quiz must be submitted in *Canvas* on or before the due date provided in the Course Schedule and in *Canvas*.

Questions in *Canvas* may be worded slightly differently than here, usually for brevity, but with the same meaning. If there is a discrepancy in what is being asked between this document and the question form in *Canvas*, answer based on what is asked in this document. Also, please let me know as soon as possible so I can get it fixed.

You will have the opportunity to upload a `doc(x)`, `html`, or `pdf` file at the end of the 'quiz' in which you show the steps of your work for the possibility of partial credit. `html` from `RMarkdown` is preferred.

14 Module 3

15 Learning objectives:

To continue gaining familiarity with the R statistical framework by using RStudio to:

- Calculate mean and standard deviation
- Calculate median, 25th and 75th percentiles, and interquartile range
- Use functions in `ggplot2` (part of `tidyverse`) to create boxplots, bar charts, scatterplots, and line graphs.

16 Packages

In this problem set, we will use more functions associated with the `tidyverse` package, including a subset of this package - `ggplot2` - which is used for creating publication-quality graphs.

First, we need to load this package. Remember - we installed this in a previous module, so we should not need to install in again and should be able to simply load it.

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

17 Datasets:

We will be working with two real-world datasets this week that are pulled directly from Zenodo, the European Union's open-source repository for scientific data. We will be using the `read_csv` command from `tidyverse` to read files directly from the internet into *R*.

17.1 Dataset 1

Our first dataset looks at the energetic costs associated with nesting in an Arctic seabird, the Common Eider *Somateria mollissima* (**hoyvik_hilde_mind_2017?**). These data represent the open-access permanent archive of a publication based on these data (Høyvik Hilde et al. 2016). The authors constructed artificial shelters around some nests to understand how it affected the physical condition of incubating female eiders. *Note* that I do a few steps below to (1) reduce the size of the data file and (2) rename columns to be more intuitive. All of these steps involve the `tidyverse` steps we've discussed before.

```
bird_data <- read_csv("https://zenodo.org/records/4960831/files/Data%20file.csv") %>%
  # select columns of interest
  select(`Nest ID`, Ring, MeanT, MeanW, MeanH, `Differ. Days`,
         Mass, `Mass recapture`, CS, CSRecapture, `Shelter added (Y/N)`) %>%
  # rename columns
  rename(Nest_ID = `Nest ID`, Bird_ID = Ring,
         Mean_Temp = MeanT, Mean_Wind = MeanW,
         Mean_Humid = MeanH, Days_between_captures = `Differ. Days`,
         Mass_Initial = Mass, Mass_Recapture = `Mass recapture`,
         Clutch_size_initial = CS, Clutch_size_recapture = CSRecapture,
         Shelter_added = `Shelter added (Y/N)`)
```

```
Rows: 87 Columns: 25
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (4): Nest ID, Shelter added (Y/N), COMMENTS, Shelter class
dbl  (19): Year, MeanT, MeanW, Winddir, MeanH, Date capture, Date recapture,...
time  (2): Time deployed, Time retrieved
```

```
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.
```

17.2 Dataset 2

Our second dataset relates to the effects of nitrogen on the growth of Scots Pine and Norway Spruce (Jetsonen et al. 2025). This is a dataset related to a publication on these trees and the volume they add that came out a year prior, and represents another open-access dataset (Jetsonen et al. 2024). Like dataset 1, we are going to select specific columns and rename them for this assignment.

```
tree_data <- read_csv("https://zenodo.org/records/14733005/files/N_fertilization_Finland_data
  select(tree, v_incr_annual_compare, FN, Pa, Tsum, SF) %>%
  rename(Tree_Species = tree,
         Annual_Mean_Volume_Growth = v_incr_annual_compare,
         Nitrogen_Dose = FN, Mean_Annual_Precipitation = Pa,
         Long_term_temperature_sum = Tsum, time_since_fertilization = SF)
```

```
Rows: 165 Columns: 16
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): tree, location, site_type
dbl (13): article_id, GS, GM, GN, Tsum, Pa, FN, v_incr_annual_compare, SF, G...

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.
```

18 Procedure:

1. Watch Module 3: Describing and Visualizing Data in R for a walk-through of these initial procedures.
2. Open a new *R* script or *Rmarkdown* script in the source pane by clicking on the `File menu > New File > R Script` or `Rmarkdown`.
3. If necessary, you can clear the environment and plots panes from *R*'s memory by type `rm(list=ls())` in the code pane and then hitting `CTRL` + `L` to clear the coding pane's text. Alternatively, this can be accomplished by clicking the broom icon at the top of both of these panes and clear the console by clicking on `Editmenu > Clear Console`.
4. Set your working directory and import both datasets. Review Modules 1 and 2 procedures, if necessary, for a reminder of how to complete these tasks.

- For these procedures and video, dataset 1 was imported as a dataframe object named `bird_data`, and dataset 2 as a dataframe object named `tree_data`.

Descriptive statistics refers to simple calculations that summarize or describe a given variable. Measures of central tendency commonly used are the mean (i.e., average) and median (i.e, 50th percentile). Corresponding measures of spread are standard deviation (accompanying the mean) and interquartile range or 25th and 75th percentiles (accompanying the median).

18.1 Section A. Calculate mean and standard deviation

5. Create a new column for the difference in mass as a percent of the birds' initial mass using the command `mutate`.

```
bird_data <- bird_data %>%
  mutate(Percent_mass_change = (Mass_Recapture - Mass_Initial)/Mass_Initial)
```

6. Calculate the mean and standard deviation of the `Percent_mass_change` variable separately for whether the nest was or was not sheltered (`Shelter_added`; `Y` or `N`). First, we will calculate these "by hand" using the following equations:

$$\bar{x} = \frac{\Sigma_1^n x}{n}$$

$$s^2 = \frac{\Sigma_i^n (x_i - \bar{x})^2}{n}$$

In order to do this, we will need to write a `function`. A `function` is a specific R command that creates a pipeline for some sort of input data.

In writing these functions, it's important to know that we can perform subtraction, multiplication, and other mathematical functions across multiple values as once. For example, if we want to know what 1 through 10 all -5.5 are equal, we can do the following:

```
# denote multiple numbers with :
x <- 1:10

x - 5.5
```

```
 [1] -4.5 -3.5 -2.5 -1.5 -0.5  0.5  1.5  2.5  3.5  4.5
```

We can write our own functions in R that will perform these functions as follows:

```
our_mean <- function(x){
  # get sum of input
  sum_x <- sum(x)
  # calculate length of input
  n <- length(x)
  # calculate mean
  xbar <- sum_x/n
  # return mean to console
  # necessary for saving outside function
  return(xbar)
}

our_sd <- function(x){
  # need to calculate mean first!
  # copied from above
  # get sum of input
  sum_x <- sum(x)
  # calculate length of input
  n <- length(x)
  # calculate mean
```

```
  xbar <- sum_x/n

  # calculate deviants
  deviants <- x - xbar
  deviants_sq <- deviants^2
  our_variance <- sum(deviants_sq)/n
  our_sd <- sqrt(our_variance)
  # return answer
  return(our_sd)
}
```

7. Remember, we can use `select` to get a specific set of data from a `dataframe` in *R*. I will calculate these metrics for one set of variables here:

```
# filter data
no_shelter_change <- bird_data %>%
  # get non-sheltered birds
  filter(Shelter_added == "N")

# extract data of interest ONLY
no_shelter_mass_change <- no_shelter_change$Percent_mass_change

# view first few values
head(no_shelter_mass_change)
```

```
[1] -0.1914894 -0.1958763 -0.1978022 -0.1956522 -0.2234043 -0.2134831
```

8. Now that we have our data subsetted, we can calculate "by hand".

```
our_mean(no_shelter_mass_change)
```

```
[1] -0.2068213
```

```
our_sd(no_shelter_mass_change)
```

```
[1] 0.02416277
```

8. Now, calculate the following for individuals *with* shelters. Your answers should be as follows:

```
our_mean(shelter_mass_change)
```

```
[1] -0.1957889
```

```
our_sd(shelter_mass_change)
```

```
[1] 0.0223379
```

9. Now, let's compare our answers to those calculated by *R*. These values may differ slightly based on rounding in *R*, but should be very close. Remember - you can use the same objects that we used before if they are still stored in your memory!

```
mean(no_shelter_mass_change)
```

```
[1] -0.2068213
```

```
sd(no_shelter_mass_change)
```

```
[1] 0.02435685
```

19 Still being formatted; check back soon.

References

Høyvik Hilde, C., C. Pélabon, L. Guéry, G. W. Gabrielsen, and S. Descamps (2016). Mind the wind: Microclimate effects on incubation effort of an arctic seabird. Ecology and Evolution 6:1914–1921.

Jetsonen, J. T., A. Lauren, H. Peltola, K. E. M. Laurén, S. Launiainen, and M. Palviainen (2025). Volume growth responses of Scots pine and Norway spruce to nitrogen fertilization: Quantitative synthesis of fertilization experiments in Finland. [Online.] Available at https://zenodo.org/records/14733005.

Jetsonen, J., A. Laurén, H. Peltola, O. Muhonen, J. Nevalainen, V.-P. Ikonen, A. Kilpeläinen, E.-S. Tuittila, E. Männistö, N. Kokkonen, and M. Palviainen (2024). Effects of nitrogen fertilization on the ground vegetation cover and soil chemical properties in Scots pine and Norway spruce stands. Silva Fennica 58.