R, RStudio, and Machine Learning Tools

R is a powerful software package for statistical computing and graphics. It runs on all popular operating systems and is open source. RStudio is the most commonly used interface for R. Although R can be used without RStudio, it enhances the user experience by simplifying R usage and adding convenient features. This lesson covers installing R and RStudio, using R as a basic calculator, evaluating formulas, and entering data.

Machine learning tools such as ChatGPT, Claude, Gemini, and Llama 3 are valuable for learning statistics and performing statistical analyses. They can provide explanations of complex statistical concepts, assist with problem-solving, and guide users through various analysis techniques. Moreover, they can help users understand how to use statistical software like R, offering suggestions for code improvements or troubleshooting common issues. By engaging with these tools, users can receive personalized support, enabling them to master statistical principles, apply appropriate methods, and interpret the results effectively.

Access to R and RStudio

During this course, you will use the statistical software package R. You have three options:

1. Install R and RStudio on your computer.
2. Use Posit Cloud.
3. Use the COSAM Windows Virtual Desktop (WVD).

I recommend option 1. To install R and RStudio, use the following link: https://rstudio.com/products/rstudio/download/. Make sure to install R first, followed by RStudio.

If you cannot or do not wish to install software on your computer, opt for options 2 or 3. Option 2 is convenient but requires an internet connection and may incur costs. Posit Cloud offers free casual use (less than 25 hours per month), and a $5 monthly plan for up to 75 hours. Additional time costs 10 cents per hour. To use option 3, follow these steps:

1. Visit the short URL https://aub.ie/cosamwvd.
2. At the Microsoft Login Screen, enter your username@auburn.edu (not your email alias).
3. At the Auburn Login Screen, enter your Auburn Password.
4. Complete Duo Multifactor Authentication via Push, Text, or Phone Call.

After launching the WVD, click on the Windows icon in the bottom left corner, scroll down to the RStudio folder, and click on the RStudio app within that folder.

R as a Basic Calculator

You can perform basic arithmetic operations in R, such as addition, subtraction, multiplication, and division. Type the operations into the R Console and press the `Enter` key:

```
(2+3)/5  # Everything after the pound sign is ignored
(2+ 3) / # Spaces are ignored.
5
(2+3)/   # If your submitted input is not complete then a plus sign appears on
           the next line. You can then complete the input. If you would rather
           start over press the Escape key.
3*2      # 3 times 2
```

R offers all the functions of a calculator and many more.

```
3^2           # 3 to the power 2
3^.5          # 3 to the power 0.5 which is the same as the square root of 3
sqrt(3)       # square root of 3
log(10)       # natural log of 10
exp(log(10))  # exponential of the natural log of 10
```

Evaluating Formulas

Suppose 871 of the 1700 babies born at the local hospital last year were boys. Would this be good evidence that a newborn baby is more likely a boy than a girl?

This is a standard textbook problem and one way to answer it involves the evaluation of the following expression

$$\frac{\hat{p}-p_0}{\sqrt{\frac{p_0(1-p_0)}{n}}}$$

where $\hat{p} = x/n$, $x = 871$, $n = 1700$, and $p_0 = 0.5$.

You could enter the values of each variable into the formula, but a better strategy is to store the values into variables and then enter the formula.

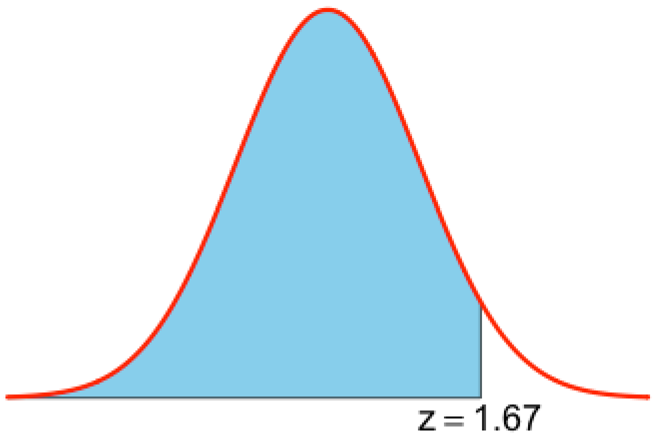
```
x <- 871
n <- 1700
pHat <- x/n
p0 <- 0.5
(pHat-p0)/sqrt(p0*(1-p0)/n)
```

By the way, worldwide the ratio of boys to girls at birth is 1.07:1, that is, there are 107 boys born for every 100 girls. In the United States that ratio is 1.05:1.

Special Functions

Your calculator does not have all the functions needed for doing statistical calculations — which is why all the common statistics books to this date have tables. With R available, these tables are unnecessary.

The most commonly used table in statistics is the one that will give you the area under the *standard normal curve* to the left of some given value z.



What is the area under the curve to the left of $z = 1.67$?

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141

0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633

According to the table the area to the left of 1.67 is 0.9525.

Using R you can find the area with the function *pnorm().*

```
pnorm(1.67)
```

The value in the table is the same as the value reported by R, except that all the values in the table are rounded to 4 digits.

Sometimes one wants to know the value of z such that the area to the left of z is a given value. For example, what is the value of z so that the area to the left of z is 0.8944?

Looking at the table we find that the area to the left of $z = 1.25$ is 0.8944.

Suppose we want to find the value of z such that the area left of z is 0.95. Unfortunately, the value 0.9500 is not in the table and the closest values are 0.9495 and 0.9505. One says that the value 0.9500 is *bracketed* by the values 0.9495 and 0.9505. The area to the left of 1.64 is 0.9495 and the area to the left of 1.65 is 0.9505. The value z we are looking for should be between 1.64 and 1.65 and so we could average these two values to obtain an estimate. Doing so we find that z is approximately 1.645.

With R the solution to this problem is much easier. We can find the value of z such that the area to the left of z is 0.95 using the function *qnorm().*

```
qnorm(0.95)
```

Using a Text Editor

During your first session with R, you may have typed commands directly into the R console and pressed the `Enter` key. If you made a mistake, you had to retype the entire command. However, there are more efficient methods.

After receiving an error message, press the `Up Arrow` key to recall the previous command, correct the mistake, and press the `Enter` key. Use the `Up Arrow` and `Down Arrow` keys to navigate through previous commands.

This approach works well for short commands, but for more complex ones, consider using a text editor.

Type the code you want to submit to R into a text editor, then copy and paste it into the R console. Be aware that using a word processor might introduce extra, invisible characters that could cause issues when pasting code into R. However, using a text editor should not cause any problems.

Text editors offer several advantages, such as easier editing and saving your code as a text file. R has its built-in text editor with additional features for this purpose.

To open R's text editor, select **New Script** from the File menu.

Type and edit commands in the script window as in any other text document.

To submit your code to R, click the icon in the middle of the toolbar. Hovering over the icon displays "Run line or selection." If you highlight a portion of text in your editor, only the highlighted text will be submitted. Otherwise, the entire line containing the cursor will be submitted, regardless of the cursor's position within the line.

Data Analysis in Two Steps

Imagine you want to calculate the average of the following 15 numbers: 55, 60, 65, 68, 72, 64, 83, 63, 66, 72, 95, 62, 83, 58, 72. The simplest and most tedious method would be to add all the numbers and divide by 15:

```
(55+60+65+68+72+64+83+63+66+72+95+62+83+58+72)/15
```

However, a more efficient approach involves two steps:

1. Enter data
2. Process data

Use the `c()` function to combine the data and store the result in a variable. Then, process the data using the `mean()` function:

```
x <- c(55, 58, 60, 62, 63, 64, 65, 66, 68, 72, 72, 72, 83, 83, 95)
mean(x)
```

Variable names should be lowercase and words separated by dots (e.g., variable.name). Choose a name that reflects the variable's purpose. If you only use a few variables, it's common to use short names like `x` or `y`. Storing data in a variable allows you to reference it easily in other calculations. Here are some examples, with explanations provided in other lessons:

```
median(x)       # calculates the median of x
sd(x)           # calculates the standard deviation of x
sort(x)         # sorts x
min(x)          # finds the minimum of x
max(x)          # finds the maximum of x
fivenum(x)      # returns the five-number summary of x
IQR(x)          # calculates the interquartile range of x
hist(x)         # produces a frequency histogram of x
boxplot(x)      # produces a boxplot of x
```

R is case sensitive, so `iqr(x)` and `IQR(x)` are not the same. To learn more about a particular function, type a question mark followed by the function's name. For example, to learn more about the `hist()` function, type:

```
?hist
```

R will open another window with detailed information about the function.

Entering Data into R

You've already learned one way to enter data into R using the *combine* function *c()*. However, this method is not practical for large datasets.

Typically, data is provided to you in a file format. A common format is *comma-separated* text files with a ".csv" extension.

As an example, consider the file class.csv. This file contains scores and course grades for 15 students. The first line has the variable names, and the subsequent lines contain the data for these variables, separated by commas. Save this file to your computer by right-clicking on the link and selecting "Save as".

Save the file to a location where it's easy to find, such as your Desktop. Once you're comfortable with the process, you may want to save the file to a specific folder.

After saving the file, locate it on your computer. The filename should be "class.csv".

To read the data stored in that file into R, use the function *read.csv()*. First, you need to tell R where the file is located. Use the "Session" menu, select "Set Working Directory", then "Choose Directory ...". Navigate to the folder containing your data file and click "Open".

If you encounter any issues, consult the troubleshooting section before continuing.

Now, store the data in a variable. Choose a variable name, such as "myData", and store the data in that variable:

```
myData <- read.csv('class.csv')
```

The variable "myData" now contains the *data frame* created from the file "class.csv". Suppose you want to calculate the mean of the final exam scores, i.e., the mean of the numbers stored in the variable "final" of the data frame "myData". Use the following command:

```
mean(myData$final)
```

If you want to perform other calculations with "myData$final", you can save time by storing these numbers in another variable with a shorter name:

```
x <- myData$final
mean(x)
```

To enter your own data, type it into a spreadsheet and save it as a *comma-separated file* with a ".csv" extension.

Troubleshooting

1. Did you save the file in the correct format? The file should be a text file with a ".csv" extension. To confirm the extension of your file, right-click the file and select *Properties* on Windows or *Get Info* on a Mac.
2. Did you set the *working directory* correctly? To verify, type *getwd()* into the R console. It should return the path to where your file is located on your computer.
3. Did you include the full name of the file, including the ".csv" extension?
4. Did you use single or double quotation marks when typing the file name? Quotation marks indicate to R that you are entering a text string and not a variable name.
5. Did you make any typos? R is case-sensitive.
6. Did you copy and paste the R code from another document? Usually, this works fine, but it may cause issues with single or double quotation marks. Word processors often use different characters for these marks compared to simple text editors.
7. Examine the error message for further guidance.

Example of Using Machine Learning

In a class of 15 students, the scores of exam 1 are:

55, 60, 65, 68, 72, 64, 83, 63, 66, 72, 95, 62, 83, 58, 72.

How spread apart are the data? One idea for measuring the spread or dispersion of the data is to calculate the average distance from its center. More precisely, the measure of spread, called the *mean deviation*, is given by the following formula:

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}|$$

Let's write an R script to evaluate this formula with the given data. To get some help open a machine learning program such as ChatGPT, Claude, Gemini, or Llama 3 and enter the following prompt:

Write an R script that calculates the average distance of the data from its center. Use the data 55, 60, 65, 68, 72, 64, 83, 63, 66, 72, 95, 62, 83, 58, 72.

Copy and paste the script you get into R and execute it. You can then ask the AI to interpret the result.

It would be convenient to create an R function that can calculate the mean deviation if we were to calculate this measure more than once. For that purpose use the following prompt: Write an R function that calculates the average distance of data from its center.