Lab 01

[Some content in this lab is borrowed from UCLA's EDUC 263: Data management and manipulation using R, and D Navarro (2018). Learning statistics with R: A tutorial for psychology students and other beginners.]

R Introduction

We'll be using the R programming language for all the labs and projects in this class. If you haven't programmed before then R can seem intimidating, but we will be using it only as a fancy (graphing) calculator. The labs are only graded for completeness, and are intended to be learning and practice for the Projects - make sure that you understand what you are doing in the labs and *ask questions* so that you will be able to complete the Projects (which are the major component of your grade for the course).

This is an R Markdown Notebook. A Notebook allows us to combine text (like what you are reading right now) with chunks of executable code. When you run code within the notebook, the results appear beneath the code. You will probably want to minimize the console at the bottom left of the screen if/when it is open (using the button that looks like a flattened rectangle). You will want to keep the panels on the right side (Environment, Files, etc.) open most of the time.

Make sure to save your work frequently, using Cmd+S/Ctrl+S or File->Save. Your Notebook is saved to the cloud, and can be accessed by any computer running a modern web browser.

Try executing this chunk by clicking the Run button within the chunk (the green arrow) or by placing your cursor inside it and pressing Cmd+Shift+Enter/Ctrl+Shift+Enter.

```r
2^8
```

```
## [1] 256
```

When you ran this chunk, it sent the command "2+2" to the R server, and it returned the answer 4 below the code chunk. Try editing 2+2 to perform other math operations, like subtraction, multiplication, division, or exponentiation (^).

To create a new code chunk, start with the line `{r} and end with the line` to indicate that the code is in the R programming language. You can also click the Insert->R button above the notebook.

Q1: Create a code chunk below that uses 2 mathematical operators to yield the result 1001.

```r
1000+1
```

```
## [1] 1001
```

```r
5005/5
```

```
## [1] 1001
```

Variables

To save partial results, we can use "variables." A variable is like a box that we can fill with a value, and then re-use later. To store a value to a variable, we use the <- operator. Run this code chunk:

```
x <- 3+5
```

This code first adds 3 and 5, and stores the value into the variable x. Notice that in the upper-right of the screen, the Environment window (in the "Global Environment" drop-down category) lists the variable x as having the value 8. Variables are shared across all code chunks, so we can re-use x again:

```
x * 2
```

```
## [1] 16
```

You can reset your environment to a blank slate by clicking Session->Clear Workspace which will clear all stored variables.

Q2: What names can we use for variables? Using the code chunk below, try replacing the variable name "test" with other possibilities. Which variable names cause an error when you run the code chunk? Can you use spaces? Special characters? Uppercase letters? Periods? Below the code chunk, write 1-2 sentences describing the naming rules you found.

```
#can use
Mikey = 1+1
mikey = 1+2
MIKEY = 1+3
mikey...<-1+4
...mik...ey... = 1+9
mi_key = 1+10
MiKeY = 1+13
Mik3y = 1+14
true = 1+15

#can't use
#mikey! <-1+5
#Mi key <- 1+6
#TRUE <- 1+7
#FALSE <- 1+8
#mi(key) = 1+11
#mi@key = 1+12
```

Answer: It seems as though you can use any combination of upper and lower case letters and numbers, as well as underscores and periods. There are many special characters that you cannot use, nor can you use the terms TRUE or FALSE.

Some tips for naming variables, which are not required but are good practice: -Use informative variable names. As a general rule, using meaningful names like "sales" and "revenue" is preferred over arbitrary ones like "variable1" and "variable2". Otherwise it's very hard to remember what the contents of different variables are, and it becomes hard to understand what your commands actually do.

-Use short variable names. Typing is a pain and no-one likes doing it. So we much prefer to use a name like "sales" over a name like "sales.for.this.book". Obviously there's a bit of a tension between using informative names (which tend to be long) and using short names (which tend to be meaningless), so use a bit of common sense when trading off these two conventions.

-Use one of the conventional naming styles for multi-word variable names. Suppose I want to name a variable that stores "my new salary". We can't include spaces in the variable name, so how should we do this? There are three different conventions that you sometimes see R users employing. Firstly, you can separate the words using periods, which would give you my.new.salary as the variable name. Alternatively, you could separate words using underscores, as in my_new_salary. Finally, you could use capital letters at the beginning of each word (except the first one), which gives you myNewSalary as the variable name.

Functions

The other reason R is powerful is because we can run pre-defined "functions" instead of writing them ourselves.Functions are pre-written bits of code that accomplish some task.

Functions generally follow three sequential steps:

1. take in one or more input variables
2. process the input.
3. output a value and/or produce a plot

One of the simplest functions is the print function, which simply prints out information (i.e. displays it beneath the code chunk)"

```r
print("Hello World")
```

```
## [1] "Hello World"
```

```r
print(100/3)
```

```
## [1] 33.33333
```

We start with the name of the function, print, which is then followed by parentheses. Inside the parentheses we can pass information (called "arguments") to the function. So the first line calls the print function and passes it the string "Hello World", which just prints the string out below the code chunk. The second line does the same thing, but with the numeric result of 100/3.

Here are some other math functions we can call:

```r
sqrt(10)
```

```
## [1] 3.162278
```

```r
factorial(4)
```

```
## [1] 24
```

```r
abs(-2.5)
```

```
## [1] 2.5
```

Q3: Create a code chunk that adds together the square root of 2 and the square root of 5.

```r
sqrt(2)+sqrt(5)
```

```
## [1] 3.650282
```

Some functions will take more than one argument. Each argument should be separated by a comma. You can see all the arguments of a function by highlighting the name of the function (inside a code block) and selecting Code->Go To Help (or pressing F1, i.e. fn->F1 on Mac keyboards).

For example, highlight the name of the round function below and open its help page:

```r
round(5.6925, 3)
```

```
## [1] 5.692
```

The function signature for round in the help file is round(x, digits = 0). This means that round has a single required argument "x" (the number you want to round) and a second argument "digits" giving the number of decimal places you want to round to. Run this code chunk for different values of x and digits to check that the output is what you expect.

Why does the help file have "digits=0" instead of just "digits"? The "=0" indicates that digits has the default value of 0, so if you don't pass in a digits argument then it will assume you want 0 decimal places.

Q4: Create two code chunks below, one that rounds 3.14159 to 2 decimal places, and another that rounds it to 0 decimal places (using only 1 argument).

```
round(3.14159, 2)
```

```
## [1] 3.14
```

```
round(3.14159)
```

```
## [1] 3
```

For function with more than one argument, you can either pass them in the order given in the help file, or name each one so that R knows which one is which. For example, to round 5.34 to one decimal place, you can call the function in any of these ways:

```
round(5.34, 1)
```

```
## [1] 5.3
```

```
round(5.34, digits=1)
```

```
## [1] 5.3
```

```
round(x=5.34, digits=1)
```

```
## [1] 5.3
```

```
round(digits=1, x=5.34)
```

```
## [1] 5.3
```

```
round(1,5.34)
```

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

```
round(1.232443232455,5.34)
```

```
## [1] 1.23244
```

```
round(1.249283749889,5.65)
```

```
## [1] 1.249284
```

Q5: What happens if we try to call round(1, 5.34)? Why?

Answer: It rounds to 1. I believe this is because there aren't any decimal points in 1, so it can only round to the ones place. If we extended it, it would round to 5 decimal places, and I think this is because it rounds the rounding number itself because decimal points can't be fractions.

The seq function generates sequences, and has the default argument values seq(from=1, to=1, by=1).

Q6: Create a code chunk that shows four different ways to generate the sequence 1, 2, 3... 10.

```
seq(1,10,1)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1,10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

To string multiple functions together, one option is to nest them inside each other. This way the value returned by one function is then used as the input to a second function. For example, to take the square root of 2 and then round to 3 decimal places:

```
round(sqrt(2), digits=3)
```

```
## [1] 1.414
```

This first runs the function sqrt(2), and then uses the value of that function as the x input to the round function.

Q7: Create a code chunk that computes the square root of the absolute value of -5, using nesting.

```
sqrt(abs(-5))
```

```
## [1] 2.236068
```

If we are going to do many operations in a row, however, it is usually better to store the results of functions into variables. Then we can reuse those variables as the inputs to later functions. For example, rounding the square root of 2 to 3 decimal places can also be performed as:

```
s2 <- sqrt(2)
round(s2, digits=3)
```

```
## [1] 1.414
```

This yields the same result, and if you look in the Environment pane you can see that we've created a new variable called s2 that has a value stored inside it. The name of this variable "s2" is not important, and can be any value variable name - this is just the label for the box that is holding the value 1.4142135623731.

Q8: Perform the same operation as in Q7, using a variable instead of function nesting.

```
my_variable = abs(-5)
sqrt(my_variable)
```

```
## [1] 2.236068
```

Data frames and mosiac

For most statistical operations, we will want to operate not just on individual numbers but on full datasets, called data frames in R. R provides an example dataset called "iris" that contains measurements of three species of the iris flower. Running a code chunk with the name of this dataset pulls up an interactive viewer showing the dataset:

```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
## 17           5.4         3.9          1.3         0.4     setosa
## 18           5.1         3.5          1.4         0.3     setosa
## 19           5.7         3.8          1.7         0.3     setosa
## 20           5.1         3.8          1.5         0.3     setosa
## 21           5.4         3.4          1.7         0.2     setosa
## 22           5.1         3.7          1.5         0.4     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 24           5.1         3.3          1.7         0.5     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 26           5.0         3.0          1.6         0.2     setosa
## 27           5.0         3.4          1.6         0.4     setosa
## 28           5.2         3.5          1.5         0.2     setosa
## 29           5.2         3.4          1.4         0.2     setosa
## 30           4.7         3.2          1.6         0.2     setosa
## 31           4.8         3.1          1.6         0.2     setosa
## 32           5.4         3.4          1.5         0.4     setosa
## 33           5.2         4.1          1.5         0.1     setosa
## 34           5.5         4.2          1.4         0.2     setosa
## 35           4.9         3.1          1.5         0.2     setosa
## 36           5.0         3.2          1.2         0.2     setosa
## 37           5.5         3.5          1.3         0.2     setosa
## 38           4.9         3.6          1.4         0.1     setosa
## 39           4.4         3.0          1.3         0.2     setosa
## 40           5.1         3.4          1.5         0.2     setosa
## 41           5.0         3.5          1.3         0.3     setosa
## 42          4.5         2.3          1.3         0.3     setosa
## 43          4.4         3.2          1.3         0.2     setosa
## 44          5.0         3.5          1.6         0.6     setosa
## 45          5.1         3.8          1.9         0.4     setosa
## 46          4.8         3.0          1.4         0.3     setosa
## 47          5.1         3.8          1.6         0.2     setosa
## 48          4.6         3.2          1.4         0.2     setosa
## 49          5.3         3.7          1.5         0.2     setosa
## 50          5.0         3.3          1.4         0.2     setosa
## 51          7.0         3.2          4.7         1.4 versicolor
## 52          6.4         3.2          4.5         1.5 versicolor
## 53          6.9         3.1          4.9         1.5 versicolor
## 54          5.5         2.3          4.0         1.3 versicolor
## 55          6.5         2.8          4.6         1.5 versicolor
## 56          5.7         2.8          4.5         1.3 versicolor
## 57          6.3         3.3          4.7         1.6 versicolor
## 58          4.9         2.4          3.3         1.0 versicolor
## 59          6.6         2.9          4.6         1.3 versicolor
## 60          5.2         2.7          3.9         1.4 versicolor
## 61          5.0         2.0          3.5         1.0 versicolor
## 62          5.9         3.0          4.2         1.5 versicolor
## 63          6.0         2.2          4.0         1.0 versicolor
## 64          6.1         2.9          4.7         1.4 versicolor
## 65          5.6         2.9          3.6         1.3 versicolor
## 66          6.7         3.1          4.4         1.4 versicolor
## 67          5.6         3.0          4.5         1.5 versicolor
## 68          5.8         2.7          4.1         1.0 versicolor
## 69          6.2         2.2          4.5         1.5 versicolor
## 70          5.6         2.5          3.9         1.1 versicolor
## 71          5.9         3.2          4.8         1.8 versicolor
## 72          6.1         2.8          4.0         1.3 versicolor
## 73          6.3         2.5          4.9         1.5 versicolor
## 74          6.1         2.8          4.7         1.2 versicolor
## 75          6.4         2.9          4.3         1.3 versicolor
## 76          6.6         3.0          4.4         1.4 versicolor
## 77          6.8         2.8          4.8         1.4 versicolor
## 78          6.7         3.0          5.0         1.7 versicolor
## 79          6.0         2.9          4.5         1.5 versicolor
## 80          5.7         2.6          3.5         1.0 versicolor
## 81          5.5         2.4          3.8         1.1 versicolor
## 82          5.5         2.4          3.7         1.0 versicolor
## 83          5.8         2.7          3.9         1.2 versicolor
## 84          6.0         2.7          5.1         1.6 versicolor
## 85          5.4         3.0          4.5         1.5 versicolor
## 86          6.0         3.4          4.5         1.6 versicolor
## 87          6.7         3.1          4.7         1.5 versicolor
## 88          6.3         2.3          4.4         1.3 versicolor
## 89          5.6         3.0          4.1         1.3 versicolor
## 90          5.5         2.5          4.0         1.3 versicolor
## 91          5.5         2.6          4.4         1.2 versicolor
## 92          6.1         3.0          4.6         1.4 versicolor
## 93          5.8         2.6          4.0         1.2 versicolor
## 94          5.0         2.3          3.3         1.0 versicolor
## 95          5.6         2.7          4.2         1.3 versicolor
## 96           5.7         3.0          4.2         1.2 versicolor
## 97           5.7         2.9          4.2         1.3 versicolor
## 98           6.2         2.9          4.3         1.3 versicolor
## 99           5.1         2.5          3.0         1.1 versicolor
## 100          5.7         2.8          4.1         1.3 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
## 105          6.5         3.0          5.8         2.2  virginica
## 106          7.6         3.0          6.6         2.1  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 114          5.7         2.5          5.0         2.0  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 124          6.3         2.7          4.9         1.8  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 127          6.2         2.8          4.8         1.8  virginica
## 128          6.1         3.0          4.9         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 139          6.0         3.0          4.8         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 147          6.3         2.5          5.0         1.9  virginica
## 148          6.5         3.0          5.2         2.0  virginica
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica
```

We can see that this data frame has 5 variables (columns), each of which has 150 observations (rows).

To perform operations on datasets, we will be using functions from the mosaic library. This is a package of functions that is not part of R itself, but was created to make simple statistical operations in R easy to perform in a standardized way. To import the mosaic library, you must run the "library" function before trying to use mosaic functions. Nesting this in the suppressMessages function just silences a bunch of informational messages we don't need to worry about.

```
suppressMessages(library(mosaic))
```

Almost all mosiac functions have the format: goal(y ~ x | grouping, data=dataframe_name)

Here goal is the name of the function, and x, y, and grouping are names of variables inside dataframe_name. Both y and grouping are often optional, and can be left blank. If we don't use the grouping variable, we don't use the vertical line.

For example, to compute the average sepal length, we can use the mean function:

```
mean( ~ Sepal.Length, data=iris)
```

```
## [1] 5.843333
```

We can also compute the mean sepal length for each species, using species as the grouping variable:

```
mean( ~ Sepal.Length | Species, data=iris)
```

```
##     setosa versicolor  virginica
##      5.006      5.936      6.588
```

Q9: Create a code chunk to determine which iris species has the widest sepals. Hint: check what other variables besides Sepal.Length are in the iris dataset.

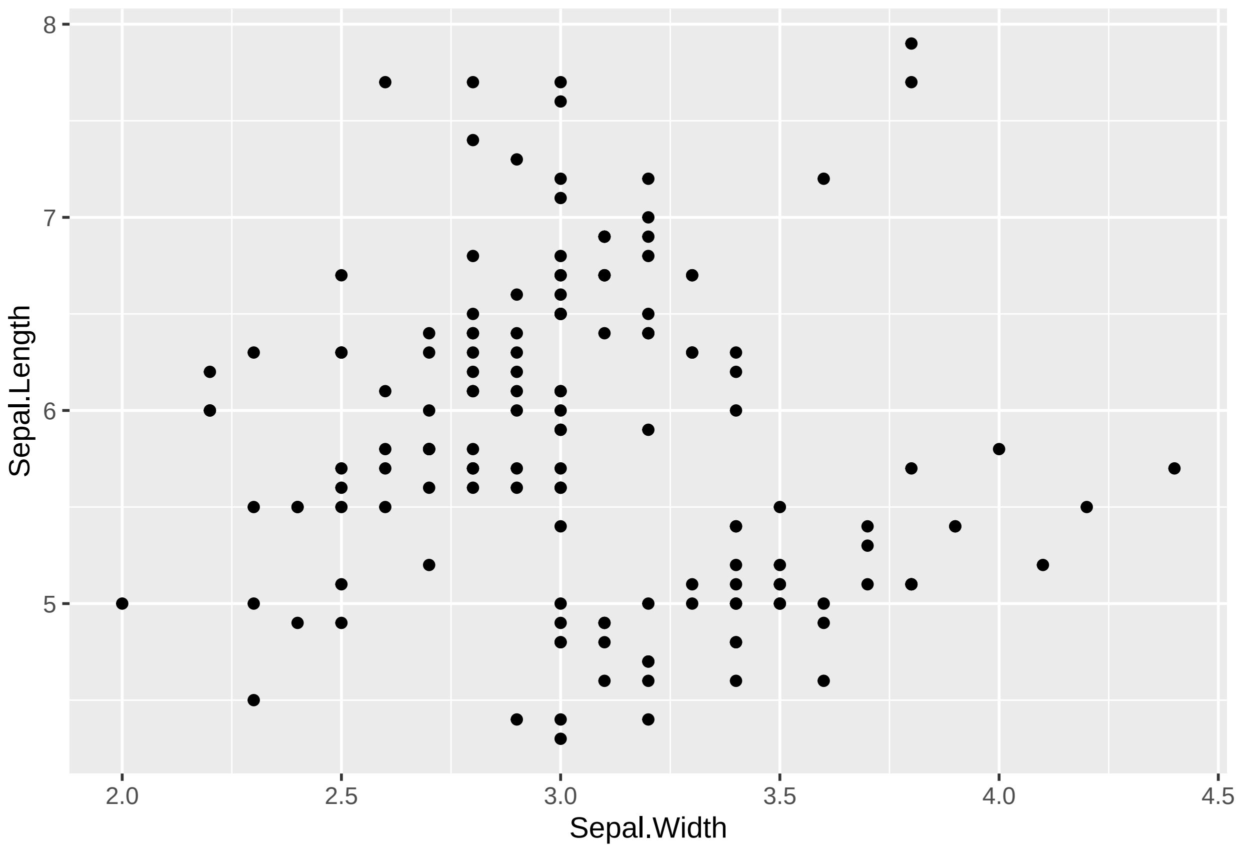
```
mean( ~ Sepal.Width | Species, data=iris)
```

```
##     setosa versicolor  virginica
##      3.428      2.770      2.974
```

Answer: species with widest sepal width is setosa

Mosaic also allows us to make different kinds of data plots. For example, we can make scatter plots with sepal width as the x axis and sepal length as the y axis:

```
gf_point(Sepal.Length ~ Sepal.Width , data=iris)
```



We will cover data visualization in more detail next week, and will be learning more about using mosiac functions throughout the course.

R Practice (time permitting)

[Feel free to skip this section if you are running out of time for your lab section, and proceed directly to "Submitting your lab notebook" below.]

Using everything you've learned above, create or complete a code chunk for each of these questions:

Q10: Compute the length of the hypotenuse of a right triangle with sides 3 and 7, to 2 decimal places.

```
a <- 3
b <- 7

hypotenuse <- round(sqrt((3^2)+(7^2)), 2)

hypotenuse
```

```
## [1] 7.62
```

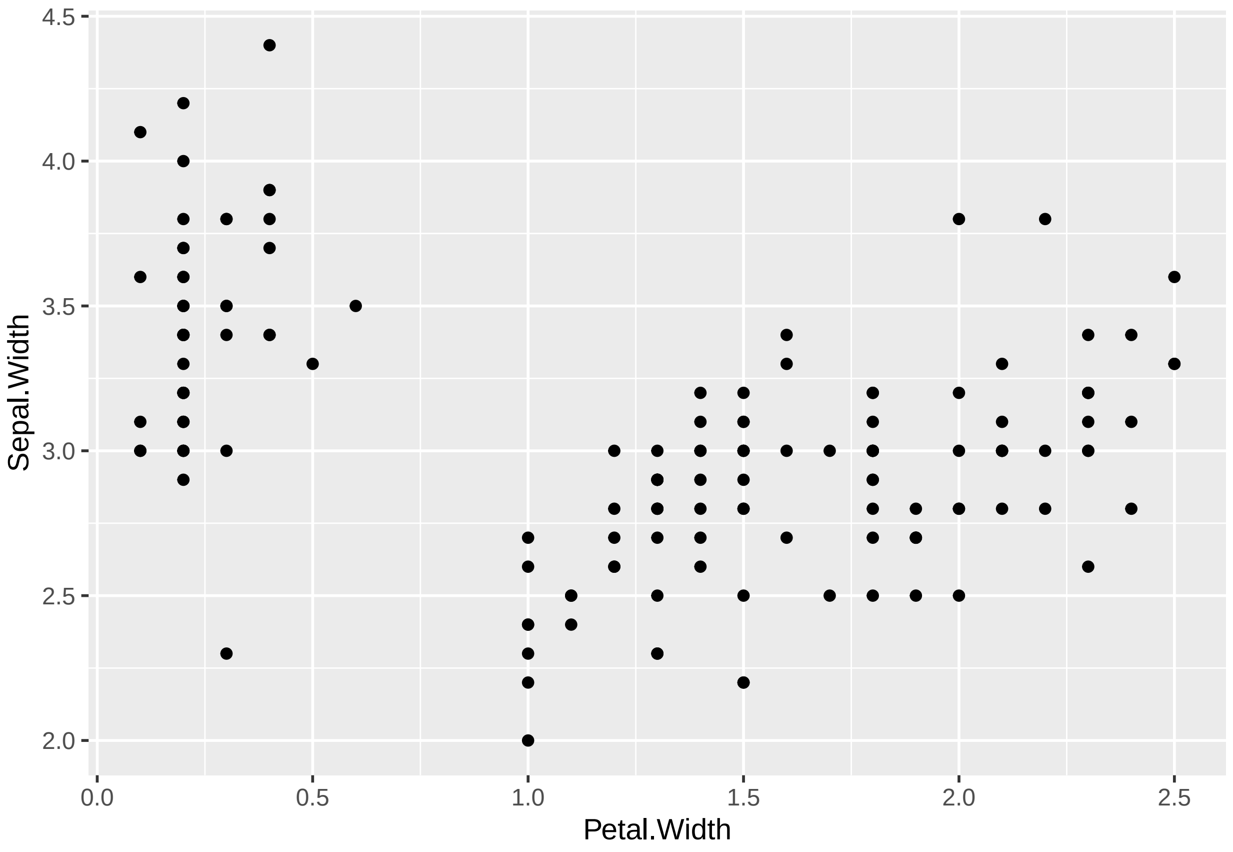
Q11: Generate the sequence 2, 4, 6... 20.

```
seq(2,20,2)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

Q12: Plot x = petal width vs. y = sepal width for the iris dataset.

```
gf_point(Sepal.Width~Petal.Width, data=iris)
```



Submitting your lab notebook

Once you've completed your notebook for a lab, you can generate a pdf by using the "knit to pdf" button, in the dropdown list at the top of the notebook (between the magnifying glass and gear icons). This will re-run all the code blocks from scratch (in a clean R environment), so inspect the pdf it creates to make sure that your code blocks are all still producing the correct answers to the questions. Then download this pdf, and submit it via courseworks.