Lab 2: data wrangling and basic plotting

Entomology 5126 - Spring 2026

Jeremy Hemberger and Brian Aukema

January 30, 2026

Table of contents

1 Introduction

Last week we entered data into R and performed some basic summary statisitics. Practice makes perfect - this week we will enter the data again, but do a few more steps within R to manipulate specific parts of the data.

You will begin to find that there are multiple ways of tackling problems within R, any of which will give you acceptable results. (For instance, last week we used the `read.table()` command. You could use `scan()` instead, with the proper arguments). In general, for these labs, I will provide helpful commands that I use. You may, however, find other commands that make more sense to you. At this stage, it might be a good idea to keep a separate Word or text file of your favourite commands - things that you know worked! It may become invaluable later on in the semester or for your thesis!

2 R Studio scripts

Last week many class members used R Studio, but not everyone was familiar with R scripts. Use File/New File/R Script or File/New File/R Notebook to launch a new document that is a notepad for your R code. Here, you can write code and save the file as a *.R (or, for R Notebooks, an `.Rmd`) file. Whenever you want to pass the command along to R, simply move to that line and hit Ctrl-Enter (or Cmd-Enter on a Mac). A good way to annotate code in a script file is to use the # sign before any comments.

3 Manipulating Data

It can sometimes be easiest to do data manipulation directly in Microsoft Excel. That said, older 32-bit versions of Excel could only handle 65,500 rows of data. Often, when you are working on an analysis, you will have to transform data on the fly. Doing transformations and reimportations of Excel data can get tedious. It's also a great way to accidentally change a value that you *don't* want to change, not realize it, and end up forever editing your data file. Remember; keep a totally raw, untouched, and locked data file in multiple locations to prevent such a mistake! In today's lab, we will also work a little with manipulating data directly within R. In general, this is a preferred way to manipulate data, because it is **documented** and **repeatable**.

4 Useful Commands

4.1 Reading data

After exporting data from Excel (or Google Sheets) (File/Save As... for a worksheet and choosing text format(tab delimited)) you can import it with the `read.table()` command, which we learned last week. I also encourage folks to look into the readR package in the Tidyverse. The `read_csv()` function here is equivalent to `read.table()` but has some subtle differences that make the code more readable and quickly interpretable.

4.2 Creating a list of numbers

A list of numbers, which you may need to specify the rows that you'd like, for example, can be created by concatenation or the `c()` command. Recall, to stick numbers together in a "vector" (list of numbers, like a spreadsheet column), name the vector, and use the concatenate command:

```
myniftyvector <- c(1,2,3,4,5)
```

As an aside, a short form for a sequence of numbers in a row, as above, is simply `1:5`. There are two other commands you may find useful at some point in this course: `seq()` and `rep()`. I will let you experiment with these using the help commands `?seq` and `?rep`. It may be a little strange at first, but if you read the examples at the bottom of the instructions and try one or two test cases, it becomes clear.

4.3 Subsetting data

Quite often, we want to perform an operation or analysis on only one portion of the data. R uses indexing to select variables, in a scheme like this:

```
myniftydata[rowsIwant, columnsIwant]
```

It's quite simple to select various rows or columns. You can either name the the columns you want, or use the indexing position. A comma separates the row from the column arguments.

`myniftydata[1:5,2]` selects the first 5 rows in the 2nd column.

`myniftydata[-3,2]` selects all rows EXCEPT the 3rd, in the 2nd column.

`myniftydata[,2:6]` selects all rows and columns 2 through 6.

`myniftydata[1:5, c("year","trial")]` selects the first 5 rows of two columns named "year" and "trial."

A quick way to select one entire column is using a dollar sign. The dollar sign specifies which column within a dataframe, by name:

`myniftydata$trial`

If you want the 5th observation, you could index that using brackets. If you are using the dollar sign to specify a column, the bracket index does not use a comma separator as above because you are only specifying the rows within the specific column:

`myniftydata$trial[5]` picks the 5th observation in the trial column.

4.4 Creating a new column

You can create a new column in a dataframe by using the dollar sign and naming the new column. For example,
`myniftydata$diameter <- 1.9` creates a new column named `diameter` that is full of the values 1.9. You can put text into a new column simply by using quotations around the text value.

4.5 Deleting an old column

Specify the offending column as NULL:

```
myniftydata$junk <- NULL
```

4.6 Selecting elements of a dataframe

Now, often we do not want just any rows or columns, but we want all rows that have a certain value. This can be accomplished using mathematical operators. For example,

`myniftydata[myniftydata$trial==4,]` selects all of the rows in the dataframe where the trial column has a value of 4 (and all columns of those specific rows).

`myniftydata[myniftydata$diameter >= 30,]` selects all rows in the dataframe where the diameter is greater than or equal to 30 (and all columns of those specific rows).

`myniftydata[(myniftydata$dbh >= 30 & myniftydata$trial==4),]` selects all rows with dbh greater than or equal to thirty in the fourth trial (and all columns of those specific rows). The ampersand & is the AND argument; use a pipe | for the OR argument.

4.7 Factors vs. continuous or discrete variables

Last class, we examined summary statistics on the data after we imported it. You may have transformed the 'trial' column in the rodent data set so that instead of "first" or "second" it said "1" or "2". When you performed a summary on the data, you received summary statistics for the trial variable: a minimum, mean, maximum.... Of course, that made little sense: who cares if the mean of the trial is 1.5? It was a categorical variable, not a numerical variable. Instead of having values of "1" or "2" the trials could have (and probably should have) been named "A" and "B".

When you read in data, R makes its best guess at what type of variables you have. The `summary()` command is useful: if you have a numerical or discrete variable, you will see a numerical summary (min, max, mean, quantiles...). If you have a categorical or text variable, you will see the different values followed by a colon and the frequency with which those values appear.

If you have a numeric variable (like trials 1 and 2) and you would like to convert it to a factor, use the `as.factor()` command:

```
myniftydata$trial <- as.factor(myniftydata$trial)
```

This command actually creates a new variable, trial, in the dataframe, and puts in the values of `myniftydata$trial` converted to a factor. Because the new variable has the same name as an existing variable, the old one is overwritten.

If you ever need to turn a categorical variable into a numeric variable (rare, but it may happen…) use this command:

```
myniftydata$trial <- as.numeric(as.character(myniftydata$trial))
```

4.8 Checking data

Last week, we examined the commands `dim()`, `nrow()`, and `summary()`. Remember, you can do the `summary()` command on a single column just as well as the entire dataframe!

5 Graphing data

R can be used to graph data, as well as analyze it. R's graphing capabilities are extremely powerful, and the `ggplot2` package has become very popular over the last decade. I *highly* encourage folks to explore `ggplot()` as the main method for plotting and visualizing your data. It's an extremely powerful and flexible package, and you can produce stunning figures with just a little bit of practice. I'll do my best to carve out some time to introduce ggplot and cover the basics of its use either today or next week.

It helps during EDA (Exploratory Data Analysis) to graph the data and see what sorts of trends might be lurking about. Brian Joiner, the founder of Minitab statistical software, has been quoted:

> Regression without first plotting the data is *always* a regression!

If you have a dataset named `myniftydata`, typing

`plot(mymyniftydata)` plots all of the variables in your data set against each other in a scatterplot matrix. This is a good way to spot trends between continuous variables. If you are only interested in a few variables (let's say columns 3,4, and 5), just specify those columns, e.g., `plot(myniftydata[,3:5])`.

If you are working with one or two specific variables in the dataframe, `plot(mydata$y~mydata$x, main="Title", xlab="x label", ylab="y label")` plots an x-y plot (with a title, x label, and y label that you specify).

`barplot(mydata$y)` plots a barplot. Labels can be added as above.

`boxplot(mydata$y)` plots a box-and-whisker (with labels if you like).

`hist(mydata$y)` plots a histogram (with labels if you like).

`plot(density(mydata$y))` plots a density plot (with labels if you like).

6 Assignment

1. Load the Excel data named Lab2datal.xls into `R`. Like last week, separate the metadata into its own sheet. I do not need to see the metadata this time, however.
2. Perform a summary. Are any adjustments necessary?
3. Plot the data in a *scatterplot matrix*. Why do graphs corresponding to the number of insects eaten by the pitcher plants appear to have straight horizontal or vertical lines?
4. You will notice that no measurement was recorded for plant #5. Let's pretend that you send your assistant out to the field and record a measurement of 3.496 after the fact. Replace the missing data point with this value (yes, you could do it very easily in Excel or in `R`'s data viewer, but see if you can do it from the `R` command line for the practice).
5. Convert the plant size in cm to mm.
6. Create a new column with percentage of insects eaten.
7. Try reproducing the above scatterplot matrix in ggplot instead of Base R. Do the same for the boxplot below.
8. Please turn in
    1. A summary of your final dataframe. (The data set, when you read it into R, is stored in what is called a - essentially a table with rows and columns).
    2. A boxplot plot of the percentage variable.