```
---
title: "Station 1 - Get Ready to Use R"
author: "Your name here"
output: html_document
---

A. Accessing RStudio
-----------------------------------------
You can access Wooster's RStudio server at [rstudio.wooster.edu](rstudio.wooster.edu).  Login
using your Wooster username and password.  You can access *your* RStudio folder from any
computer connected to the internet -- no need to download anything, or transfer files between
computers!  However, if off-campus you will need to VPN to the campus network; directions on
how to do that are [https://technology.wooster.edu/virtual-private-network-vpn/](here).

You may also download R and RStudio on your personal machine, although I don't recommend
doing this right now.  You can download R at [http://cran.us.r-project.org/](CRAN).  Then
download Rstudio Desktop at [https://rstudio.com/products/rstudio/](RStudio).

As you can see, the RStudio environment contains four quadrants: the upper-left contains the
files (including data sets) you're working on; the lower-left contains the **Console**, where
you can run R code; the lower-right contains several tabs including Plots, Packages, and
Help; and the upper-right contains your History, including a list of loaded data sets.

### Creating a space for your documents
I recommend creating a folder to keep all your DATA 230/231 activities and handouts in.  You
will have a LOT of them this semester, and it's best to start an organizational system now.
If you're working with RStudio on your desktop, you know how to create folders/subfolders.
If you're working on Wooster's RStudio server, click on "Files" in the lower-right corner,
then create a New Folder called "DATA231" or "DATA230" and within that, a folder called
"Activities".  **Save this document in that folder now.**  Call it "Station1.Rmd"

B. R Markdown
----------------------
The thing you are reading right now is an **R Markdown** document. R Markdown runs *inside* R
Studio.  Markdown is a simple formatting syntax for authoring web pages, word documents, and
pdfs. On Moodle, you can find the link to a useful R Markdown guide.  You can also click the
**?** toolbar button for more details on using R Markdown.

Fist, [watch this short video to introduce yourself to RStudio]
(https://rmarkdown.rstudio.com/lesson-1.html) (https://rmarkdown.rstudio.com/lesson-1.html).

When you click the **Knit** button at the top of this window a web page will be generated
that includes both content as well as the output of any embedded R code chunks within the
document. Not only can you imbed the R *code*, you can imbed the *output* produced by the R
code.  In this way, your analysis is fully reproducible and updatable.  **All of your
homework assignments must be prepared using R Markdown, and submitted as a pdf or Word
document**.

The nice thing about R Markdown is that you can write prose (as I am doing now), mathematical
equations using LaTeX syntax (like $y_i = a + b x_i$), and R code/output/plots all in one
synthesized document.  This makes it approximately 10,000 times easier to use than doing the
same thing in Word or LaTeX.  As you go through this introduction, I recommend that you look
also at the .Rmd file and .html file side-by-side, to get an idea of how R Markdown works.
(Since you're going to have to use R Markdown this semester, you might as well start learning
it now!)

In the future, when you want to start a new R Markdown document, click "File > New File > R
Markdown..."  Put yourself as author, and make sure to give it a descriptive title!


**The goal of this station is to get you ready to read, visualize, and use data.  You will
also become comfortable navigating the RStudio interface and running code from within a
document.**
```

````
C. Using R as a Calculator and Running Code within a RMarkdown Document
---------------------------
With R by your side, you will never need your TI-84 again. Consider the simple arithmetic in
the chunk below.  You can run this code chunk all at once by by clicking the *Run* button
(sideways green arrow) within the chunk. Notice the interactive nature of the output.  You
can click through to see the different pieces of output.

**As you work through this document, you should Run each chunk as you come to it.**

```{r}
5 + 3

5.3 * 23.4

sqrt(16) # this is a comment.  R will not 'see' anything past a #, but you can use it to
explain your code
```
Look closely at how R Markdown denotes the R code and the output. Also note in the .Rmd file
how I include R code as separate from  prose.  These are called "chunks".  The easiest way to
add a new chunk is to click on the green "C Insert" icon above, then choose "R".  R code that
is not inside of a chunk *will not* be run by R Markdown!

You can also save values to named variables, to be used later:
```{r}
product <- 15.3 * 23.4 #save result
```
If you save something like this, R will not show the output unless you expressly ask for it:
```{r}
product #show result
```
Alternatively, I could use "=" instead of "<-":
```{r}
product = 15.3 * 23.4  #save result
product #show result
```

The symbol "<-" is the *assignment operator*.  It's essentially the same as "=" in this
instance.  When we get into more complicated work, there are cases where <- works and =
doesn't, so it's good to get into the habit of using <- now.

Once variables are defined, they can be referenced with other operators and functions. Try
executing each line of code individually by placing your cursor on line 67 and pressing
*Ctrl+Enter* (*Cmd +Enter* for Mac users); then do the same for line 68. (This is how you can
run a single line within a larger chunk.)
```{r}
.5 * product # half of the product
product^(1/4) # fourth root of the product
```
You can also use in-line R code in R Markdown, which can be useful when calling defined
variables.  Did you know that the natural log of `r product` is `r log(product)`?

The semi-colon can be used to place multiple commands on one line. One frequent use of this
is to save and print a value all in one go, as below:
```{r}
product <- 15.3 * 23.4 ; product  # save result and show it
```

*Only R code (and comments) should be inside chunks.*  Prose
(interpretations/explanations/descriptions) should never be put inside a chunk; prose should
be below or above the chunk, as I have done above (and continue to do throughout this
document).  You should also never cut-and-paste output or graphs into the chunks.  The whole
point of code chunks is that they contain the code *and* they'll run the code (resulting in
the output and/or graphs).
````

If you want to run something in R but don't want it to appear in the R Markdown document, simply run it in the **Console** in the lower-left quadrant of RStudio. Type the last line of R code above into the Console and see what happens.

D. Six Things to Know About R and RStudio
---

1. R is case-sensitive.
If you mis-capitalize something, it won't do what you want.

2. TAB completion and arrows can improve typing speed and accuracy.
If you begin a command and hit the TAB key, R will show you a list of possible ways to complete the command. If you hit TAB after the opening parenthesis of a function, it will show you the list of arguments it expects. The up and down arrows can be used to retrieve past commands.

3. If you see a `+` prompt, it means R is waiting for more input.
Often this means that you have forgotten a closing parenthesis or made some other syntax error. If you have completely messed up and just want to get back to the console, hit the ESC key and start the command fresh.

4. The Console and the Markdown document are *separate entities*. Suppose you've run something, saved something, opened a data set, or opened loaded a package in the Console. It exists *only* in the Console, not in the R Markdown document. Think of the Console as a "practice area" -- a place where you can play around and try stuff. Once you've settled on a piece of code and you want it to appear in the "official" document (that you'll turn in to me), put it into the Markdown document (within a "chunk", of course).

Remember! The Markdown document (.Rmd) and the Console are *separate* entities. Just because you've done something (run some code, loaded some data, etc.) in one place, it **does not** transfer to the other place.

E. R packages

In addition to its core features, R provides many more features through a (large) number of packages. To use a package, it must be installed (one time), and loaded (each session). You install the package by clicking the "Packages" tab in the lower-right quadrant, then clicking "Install" and type the name of the package. To load the package in the Console, you can type `library(package)`, or just click the button next to the package in the "Packages" tab. However, if you are going to use a package within an R Markdown document, you must "call" the package within your particular R Markdown document. (Why? Because the Console and the Markdown document are separate entities!)

For example: We will be using the "tidyverse" package a lot this term. The package is already installed on Wooster's RStudio server. (If you're using a desktop version of RStudio, install this package now.) Load the package in the Console by typing `library(tidyverse)` in the Console. Now we still need to load it *in this R Markdown document*.

```{r}
library(tidyverse) #this loads the tidyverse package into this R Markown document
#We could also use the code require("tidyverse") to do this
```

The "tidyverse" package is now loaded. But it sure takes up a lot of space on the html file, doesn't it? If we wanted to supress the output from this particular chunk on the html file, we could use `include=FALSE` in the chunk header, as below. (Note that the chunk still runs; but the code and the output is supressed when it knits to html. There are other useful chunk headers that we will see as we continue learning R Markdown...)

```{r include=FALSE}
library(mosaic)
library(ggformula)
library(Stat2Data)
```

We will use these four packages a lot this semester, so you should become accustomed to loading them all in every session.

F. Getting Help

If something doesn't go quite right, or if you can't remember something, it's good to know where to turn for help. In addition to asking your friends and neighbors, you can use the R help system.

1. To get help on a specific function or data set, simply precede its name with a `?`. Type "?histogram" into the Console and see what happens. Alternatively, you can type "histogram" into the search bar in the Help tab in the lower-right quadrant.

2. If you don't know the exact name of a function, you can give part of the name to "apropos" and R will find all functions that match. Quotation marks inside the `apropos()` function (single or double) are mandatory here. Type "apropos('hist')" into the Console and see what happens. Alternatively, you can type "hist" into the search bar in the Help tab in the lower-right quadrant.

3. If that fails, you can do a broader search using `??` or `help.search()`, which will find matches not only in the names of functions and data sets, but also in the documentation for them. Quotations marks are optional here. Try typing "??histogram" and/or "help.search('histogram')" into the Console.

4. Examples and Demos. Many functions and data sets in R include example code demonstrating typical uses. For example, "example(histogram)" will generate a number of example plots (and provide you with the commands used to create them). Examples such as this are intended to help you learn how specific R functions work. These examples also appear at the end of the documentation for functions and data sets.

Some packages include demos. Demos are bits of R code that can be executed using the `demo()` command with the name of the demo. Demos are intended to illustrate a concept, a method, or some such thing, and are independent of any particular function or data set. You can get a list of available demos using "demo()", or demos from just the package of interest using "demo(package='NAME OF PACKAGE')".

5. The Internet. One of the great things about R is how well-documented it is online. Since it's free and open-source and people across disciplines use it, there is an astounding amount of information available on the web. If you get stuck on something, Google it: I can almost guarantee you will find someone who has posted a solution to either your exact problem, or something close to it.

NOTE: There are usually MANY different ways to do things in R. The ways I use, and the ways I show you, are (almost) never the *only* way, and are not necessarily the *best* or *simplest* way. Feel free to find different/better/easier ways to do things! I (almost) never care about the details of the code, as long as you're getting the "right" result.

G. Loading Data

As long as they are in the right format, data sets can be loaded from files saved on your computer, or straight from webpages. Data sets can be whitespace-delimited, comma-delimited, or tab-delimited. RStudio is smart enough that it can *usually* tell which kind of delimiter is being used.

Many of the data sets that we will be using in class this semester are in the `Stat2Data` package. We loaded this package above. (...As long as you Ran the chunk at line 108. Remember that you should be running every chunk as you work through this document!)

The `Day1Survey` dataset lives within the Stat2Data package. To load this data set, type `data(Day1Survey)` in the chunk below. Note that as you type, R will start to auto-complete with the appropriate data set! Then run the chunk below, so that the data set has also been loaded in the Console. (Because the RMarkdown document and the Console are separate entities, *right*?)
```{r}

```

Let's load another dataset from the Stat2Data package, and do a couple fun with things with it. Notice how, when you Knit this document, the results (the means and side-by-side boxplots) show up just below the code!

```{r}
# this dataset is from an experiment studying rats and weight gain as a function of diet
data("FatRats")

#calculate means for each of three protein sources
mean(Gain ~  Source, data = FatRats)

# next, create side-by-side boxplots by protein level (hi vs. lo) *and* source (beef vs cereal vs pork)
gf_boxplot(Gain ~ Protein | Source, data = FatRats)
```

Data can also be loaded from files on your computer (.txt, .xls, .csv), or from the internet. If working on the College's RStudio server, you often need to upload the data on the server first. Let's try this now:

- Download the "USStates_fall2021.csv" file from Moodle.

- In the lower-right quadrant of RStudio, click on "Files", then your new "DATA231" or "DATA230" folder that you created in Section A.

- Click "Upload > Choose File", and select the USStates_fall2021.csv file that you downloaded. Then click the Upload button.

Now you can Import the data set into your Console and/or RMarkdown document.

- In the top-right corner of RStudio, click "Environment > Import Dataset".

- Navigate to your DATA231/DATA230 folder and welect USStates_fall2021.csv and click "Open".

- As I said above, RStudio is smart enough that it can usually tell which kind of delimiter is being used. It is showing you a preview of the data set in the "Data Frame" window. Make sure it looks okay: columns and rows, with column names given by the Headings (not V1, V2, etc). Then hit "Import".

- The data set is now loaded in the Console, and the Console now has the code to load the data set. But of course, that doesn't mean it's loaded in the RMarkdown document! Copy that line (the read.csv line, not the "View" line) in the code chunk below.

```{r}

```

Now use the code in lines 151 and 154 as a guide to calculate mean household income by region and make side-by-side boxplots of household income by region.

```{r}

```

FINALLY

...Knit this entire document and make sure it renders correctly. In general, I suggest "knitting as you go": knitting every few chunks, to make sure things are rendering correctly, rather than waiting until the end of a document to knit the whole thing (and potentially encountering lots of errors that you have to unpack)! I also ALWAYS suggest "saving as you go": it's a good idea to save every couple of minutes. (This is good practice for all your files.)