

# Introduction to R Programming

## Red Rock Data Science

W. Evan Johnson, Ph.D.  
Professor, Division of Infectious Disease  
Director, Center for Data Science  
Rutgers University – New Jersey Medical School  
[w.evan.johnson@rutgers.edu](mailto:w.evan.johnson@rutgers.edu)

2024-05-23

## R Tutorials (GitHub and YouTube)

Dr. Johnson will provide an online R tutorial on GitHub:  
[https://github.com/wevanjohnson/2024\\_04\\_R\\_tutorial](https://github.com/wevanjohnson/2024_04_R_tutorial)

# R Tutorials (GitHub and YouTube)

| Lecture    | Topics                                |
|------------|---------------------------------------|
| Lecture 1  | Installing R, RStudio, and R packages |
| Lecture 2  | Introduction to R/RStudio             |
| Lecture 3  | R basics, Part 1                      |
| Lecture 4  | R basics, Part 2                      |
| Lecture 5  | R basics, Part 3                      |
| Lecture 6  | Programming Basics                    |
| Lecture 7  | R Markdown                            |
| Lecture 8  | Input/output data, Data structures    |
| Lecture 9  | The tidyverse                         |
| Lecture 10 | Visualization with ggplot2, Part 1    |
| Lecture 11 | Visualization with ggplot2, Part 2    |
| Lecture 12 | Visualization with ggplot2, Part 3    |
| Lecture 13 | Creating R Packages                   |
| Lecture 14 | Shiny Programming, Part 1             |
| Lecture 15 | Shiny Programming, Part 2             |

# Section 1

## Installation Details

# Important installations

You will need to install the following: \vskip .1in

## Mac Users

- R and R Studio
- Know how to access a terminal (Rstudio or Terminal)
- git (type `git --version` in the terminal)

\vskip .1in

## Windows Users:

- R and R Studio
- A terminal app (Git Bash, MobaXterm, Putty)
- Git for Windows

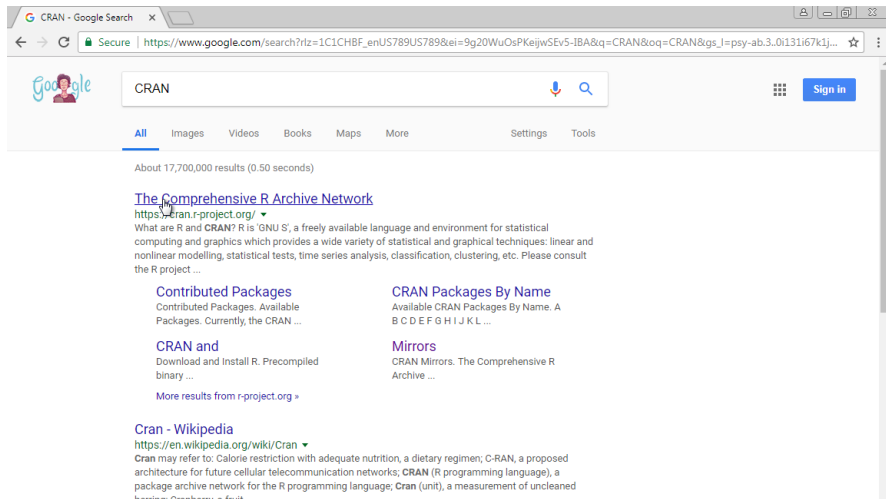
# R and Rstudio

**R** is a language for statistical computing and graphics. **RStudio** is an interactive desktop environment (IDE), but it is not R, nor does it include R when you download and install it. Therefore, to use RStudio, we first need to install R.



# Installing R (Windows and Mac)

You can download R from the Comprehensive R Archive Network (CRAN)<sup>1</sup>. Search for CRAN on your browser:



A screenshot of a web browser showing a Google search for "CRAN". The search bar contains the text "CRAN". Below the search bar, the results are displayed. The top result is "The Comprehensive R Archive Network" with the URL "https://cran.r-project.org/". Below this result, there are several links: "Contributed Packages", "CRAN Packages By Name", "CRAN and Mirrors", and "Cran - Wikipedia".

CRAN - Google Search

Secure | [https://www.google.com/search?rlz=1C1CHBF\\_enUS789US789&ei=9g20WuOsPKiejwSEv5-IBA&q=CRAN&oq=CRAN&gs\\_l=psy-ab.3..0i131i67k1j...](https://www.google.com/search?rlz=1C1CHBF_enUS789US789&ei=9g20WuOsPKiejwSEv5-IBA&q=CRAN&oq=CRAN&gs_l=psy-ab.3..0i131i67k1j...)

CRAN

All Images Videos Books Maps More Settings Tools

About 17,700,000 results (0.50 seconds)

[The Comprehensive R Archive Network](https://cran.r-project.org/)  
<https://cran.r-project.org/> ▼  
 What are R and CRAN? R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project ...

**Contributed Packages**  
 Contributed Packages. Available Packages. Currently, the CRAN ...

**CRAN Packages By Name**  
 Available CRAN Packages By Name. A B C D E F G H I J K L ...

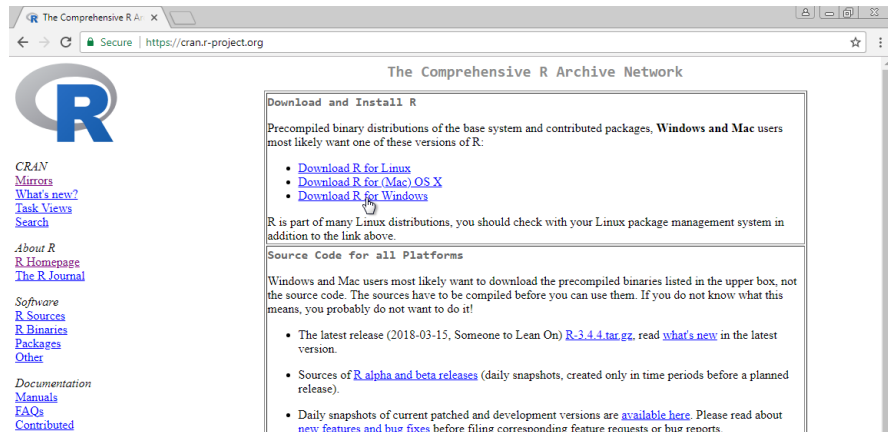
**CRAN and Mirrors**  
 Download and Install R. Precompiled binary ...  
 CRAN Mirrors. The Comprehensive R Archive ...

[More results from r-project.org »](#)

**Cran - Wikipedia**  
<https://en.wikipedia.org/wiki/Cran> ▼  
 Cran may refer to: Calorie restriction with adequate nutrition, a dietary regimen; C-RAN, a proposed architecture for future cellular telecommunication networks; **CRAN** (R programming language), a package archive network for the R programming language; **Cran** (unit), a measurement of uncleaned berries; Cranberry, a fruit.

# Installing R (Windows and Mac)

Once on the CRAN page, select the version for your operating system: Linux, Mac OS X, or Windows. Here we show screenshots for Windows, but the process is similar for the other platforms. When they differ, we will also show screenshots for Mac OS X.



The screenshot shows a web browser window with the address bar displaying "https://cran.r-project.org". The page title is "The Comprehensive R Archive Network". On the left side, there is a navigation menu with links: CRAN, Mirrors, What's new?, Task Views, Search, About R, R Homepage, The R Journal, Software, R Sources, R Binaries, Packages, Other, Documentation, Manuals, FAQs, and Contributed. The main content area is titled "Download and Install R" and contains the following text: "Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:". Below this text are three bullet points with links: "Download R for Linux", "Download R for (Mac) OS X", and "Download R for Windows". A mouse cursor is hovering over the "Download R for Windows" link. Below the links, it says "R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above." There is another section titled "Source Code for all Platforms" which contains the text: "Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!". Below this text are three bullet points: "The latest release (2018-03-15, Someone to Lean On) [R-3.4.4.tar.gz](#), read [what's new](#) in the latest version.", "Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).", and "Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports."



# Installing RStudio (Windows and Mac)

To install RStudio, start by searching for “RStudio” on your browser:

The screenshot shows a Safari browser window with the address bar set to a Google search for "rstudio". The search results page displays "About 76,400,000 results (0.27 seconds)". The top result is "RStudio Desktop" from Posit, with a link to <https://posit.co/download/rstudio-desktop>. The description states: "RStudio Desktop. Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive ...". Below this, there are links for "R Packages", "Enterprise", "Cheatsheets", and "Resources".

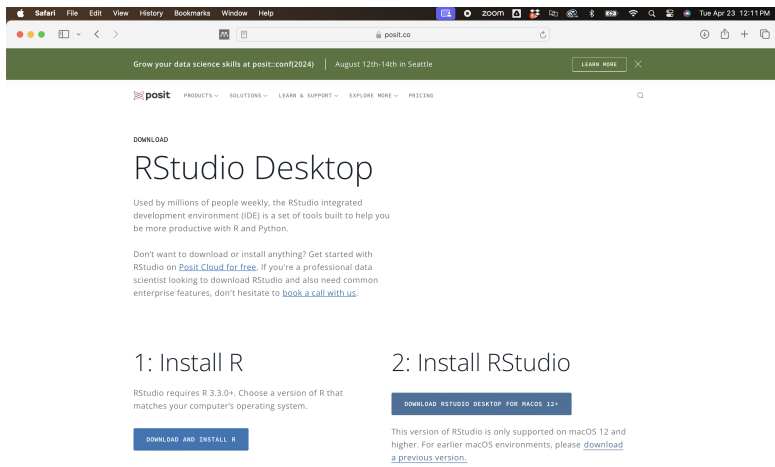
The right sidebar provides more details about RStudio:

- RStudio** (Computer program)
- RStudio is an integrated development environment for R, a programming language for statistical computing and graphics. It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser. [Wikipedia](#)
- Programming languages:** Java, C++, JavaScript
- Developer:** Posit, Joseph J. Allaire
- Inventor:** JJ Allaire [doubt](#)
- Initial release:** 28 February 2011; 13 years ago
- License:** GNU Affero General Public License v3
- Operating system:** Ubuntu, Fedora, Red Hat Linux, openSUSE, macOS, Windows NT
- Platform:** IA-32, x86-64; Qt

Below the sidebar, there are more search results from Posit, including "Download RStudio | The Popular Open-Source IDE from ..." and "Posit | The Open-Source Data Science Company".

# Installing RStudio (Mac)

You should find the Posit/RStudio website as shown above. Once there, click on “Download RStudio Desktop for Mac OS 12+” below the *2: Install RStudio* header.



The screenshot shows a web browser window displaying the Posit website. The browser's address bar shows 'posit.co'. The website has a green header with the Posit logo and navigation links: PRODUCTS, SOLUTIONS, LEARN & SUPPORT, EXPLORE MORE, and PRICING. A search icon is also present. Below the header, the page is titled 'RStudio Desktop' under the 'DOWNLOAD' section. The text describes RStudio as an integrated development environment (IDE) for R and Python. It offers two options: downloading or installing RStudio, or getting started with RStudio on Posit Cloud for free. A button labeled 'DOWNLOAD AND INSTALL R' is visible. To the right, under the heading '2: Install RStudio', there is a button labeled 'DOWNLOAD RSTUDIO DESKTOP FOR MACOS 12+'. Below this button, a note states that this version of RStudio is only supported on macOS 12 and higher, with a link to 'download a previous version' for earlier environments.

Grow your data science skills at [posit::conf\(2024\)](#) | August 12th-14th in Seattle [LEARN MORE](#)

**posit** PRODUCTS SOLUTIONS LEARN & SUPPORT EXPLORE MORE PRICING

DOWNLOAD

## RStudio Desktop

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.

Don't want to download or install anything? Get started with RStudio on [Posit Cloud for free](#). If you're a professional data scientist looking to download RStudio and also need common enterprise features, don't hesitate to [book a call with us](#).

### 1: Install R

RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

[DOWNLOAD AND INSTALL R](#)

### 2: Install RStudio

[DOWNLOAD RSTUDIO DESKTOP FOR MACOS 12+](#)

This version of RStudio is only supported on macOS 12 and higher. For earlier macOS environments, please [download a previous version](#).

## More on R and Rstudio

See more detailed instructions at in Lecture 1 at:  
[https://github.com/wevanjohnson/2024\\_04\\_R\\_tutorial](https://github.com/wevanjohnson/2024_04_R_tutorial)

# Accessing the terminal and installing Git

For Wednesday (4/24), you will also need to install a terminal and install Git (and get a GitHub account).

Detailed instructions: <https://rafalab.github.io/dsbook/accessing-the-terminal-and-installing-git.html>

## Why R?

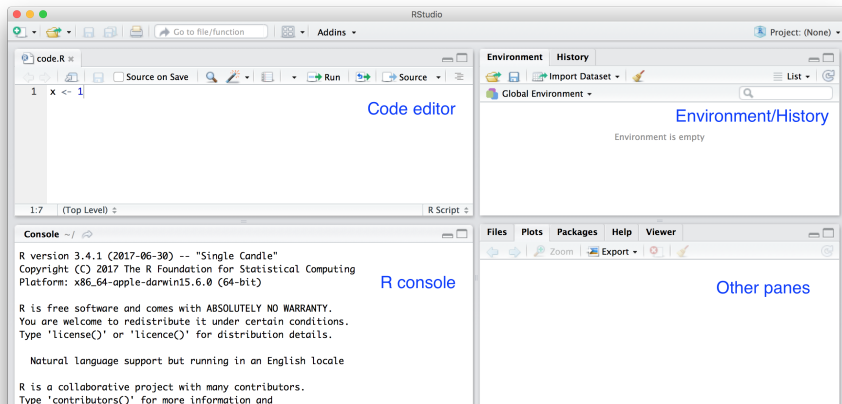
R is not a programming language for software development like C or Java. It was created by statisticians as an environment for data analysis. A history of R is summarized here: [A Brief History of S](#).



The **interactivity** of R (more later), is an indispensable feature in data science because, as you will learn, the ability to quickly explore data is a necessity for success in this field.

# RStudio

One of the great advantages of R over point-and-click analysis software is that you can save your work as scripts. You can edit and save these scripts using a text editor. We will use the interactive *Integrated Development Environment* (IDE) RStudio.



# Objects

Suppose we are asked to solve the quadratic equation  $x^2 + x - 1 = 0$ , we can define:

```
a <- 1  
b <- 1  
c <- -1
```

which stores the values for later use. We use `<-` to assign values to the variables. We can also assign values using `=` instead of `<-`, but we recommend against using `=` to avoid confusion.

# Objects

To see the value stored in a variable, we simply ask R to evaluate `a` and it shows the stored value:

```
a
```

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

A more explicit way to ask R to show us the value stored in `a` is using `print` like this:

```
print(a)
```

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



# The Workspace

Now since these values are saved in variables, to obtain a solution to our equation, we use the quadratic formula:

```
(-b + sqrt(b^2 - 4*a*c) ) / ( 2*a )
```

```
## [1] 0.618034
```

```
(-b - sqrt(b^2 - 4*a*c) ) / ( 2*a )
```

```
## [1] -1.618034
```

# Scripts

To solve another equation such as  $3x^2 + 2x - 1$ , we can copy and paste the code above and then redefine the variables and recompute the solution:

```
a <- 3
b <- 2
c <- -5
(-b + sqrt(b^2 - 4*a*c)) / (2*a)
(-b - sqrt(b^2 - 4*a*c)) / (2*a)
```

By creating and saving a script with the code above, we would not need to retype everything each time and, instead, simply change the variable names. Try writing the script above into an editor and notice how easy it is to change the variables and receive an answer.

# Functions

Once you define variables, the data analysis process can usually be described as a series of **functions** applied to the data. R includes several predefined functions and most of the analysis pipelines we construct make extensive use of these.

# Functions

Most functions require one or more **arguments**. Below is an example of how we assign an object to the argument of the function `log`. Remember that we earlier defined `a` to be 1:

```
log(8)
```

```
## [1] 2.079442
```

```
log(a)
```

```
## [1] 0
```

# Functions

You can change the default values by simply assigning another object:

```
log(8, base = 2)
```

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

Note that we have not been specifying the argument `x` as such:

```
log(x = 8, base = 2)
```

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

# Installing R packages

The functionality provided by a fresh install of R is only a small fraction of what is possible. In fact, we refer to what you get after your first install as **base R**. The extra functionality comes from add-ons available from developers.

There are currently hundreds of these available from CRAN and many others shared via other repositories such as GitHub. However, because not everybody needs all available functionality, R instead makes different components available via **packages**.

# Installing R packages

R makes it very easy to install packages from within R. For example, to install the **dslabs** package, which we use to share datasets and code related to this book, you would type:

```
install.packages("dslabs")
```

We can install more than one package at once by feeding a character vector to this function:

```
install.packages(c("tidyverse", "dslabs"))
```

# Vectors

In R, the most basic objects available to store data are **vectors**. As we have seen, complex datasets can usually be broken down into components that are vectors. For example, in a data frame, each column is a vector. Here we learn more about this important class.



# Creating Vectors

We can create vectors using the function `c`, which stands for **concatenate**. We use `c` to concatenate entries in the following way:

```
codes <- c(380, 124, 818)
codes
```

```
## [1] 380 124 818
```

# Creating Vectors

We can also create character vectors. We use the quotes to denote that the entries are characters rather than variable names.

```
country <- c("italy", "canada", "egypt")
```

# Names

Sometimes it is useful to name the entries of a vector. For example, when defining a vector of country codes, we can use the names to connect the two:

```
codes <- c("italy" = 380, "canada" = 124, "egypt" = 818)
```

```
codes
```

```
##   italy canada  egypt
```

```
##    380    124    818
```

# Subsetting elements

The elements of vectors can be obtained using the following:

```
codes[1]
```

```
## italy
```

```
## 380
```

```
codes["italy"]
```

```
## italy
```

```
## 380
```

```
codes[2:3]
```

```
## canada egypt
```

```
## 124 818
```

# Data Frames

The most common way of storing a dataset in R is in a **data frame**, which is a combination of several (columns of) vectors.

Data frames are particularly useful for datasets because we can combine different data types into one object.

# Data Frames

A large proportion of data analysis challenges start with data stored in a data frame. For example, we stored the data for our motivating example in a data frame. You can access this dataset by loading the **dslabs** library and loading the murders dataset using the data function:

```
library(dslabs)  
data(murders)
```

# Data Frames

We can show the first six lines using the function `head`:

```
head(murders)
```

| ##   | state      | abb | region | population | total |
|------|------------|-----|--------|------------|-------|
| ## 1 | Alabama    | AL  | South  | 4779736    | 135   |
| ## 2 | Alaska     | AK  | West   | 710231     | 19    |
| ## 3 | Arizona    | AZ  | West   | 6392017    | 232   |
| ## 4 | Arkansas   | AR  | South  | 2915918    | 93    |
| ## 5 | California | CA  | West   | 37253956   | 1257  |
| ## 6 | Colorado   | CO  | West   | 5029196    | 65    |

## The Accessor: \$

For our analysis, we will need to access the different variables represented by columns included in this data frame. To do this, we use the accessor operator \$ in the following way:

```
murders$population
```

```
## [1] 4779736 710231 6392017 2915918 37253956 5029196 3574097 89
## [9] 601723 19687653 9920000 1360301 1567582 12830632 6483802 304
## [17] 2853118 4339367 4533372 1328361 5773552 6547629 9883640 530
## [25] 2967297 5988927 989415 1826341 2700551 1316470 8791894 205
## [33] 19378102 9535483 672591 11536504 3751351 3831074 12702379 105
## [41] 4625364 814180 6346105 25145561 2763885 625741 8001024 672
## [49] 1852994 5686986 563626
```



# Subsetting Columns and Rows

In addition the columns and rows of a data frame can be subsetting using the following syntax:

```
murders[1,1]
```

```
## [1] "Alabama"
```

```
murders[1,]
```

```
##      state abb region population total
## 1 Alabama  AL  South     4779736    135
```

```
murders[,4]
```

```
## [1] 4779736  710231 6392017 2915918 37253956 5029196 3574097 89
## [9] 601723 19687653 9920000 1360301 1567582 12830632 6483802 304
## [17] 2853118 4339367 4533372 1328361 5773552 6547629 9883640 530
## [25] 2967297 5988927 989415 1826341 2700551 1316470 8791894 205
## [33] 19378102 9535483 672591 11536504 3751351 3831074 12702379 105
## [41] 4625364 814180 6346105 25145561 2763885 625741 8001024 672
## [49] 1852994 5686986 563626
```

# Programming Basics

By coding in R, we can efficiently perform exploratory data analysis, build data analysis pipelines, and prepare data visualization to communicate results. However, R is not just a data analysis environment but a programming language.

You should also understand the following three key programming concepts: **conditional expressions**, **for-loops**, and **functions**. These are not just key building blocks for advanced programming, but are sometimes useful during data analysis.

# R markdown

R markdown is a format for **literate programming** documents. Literate programming weaves instructions, documentation, equations, and detailed comments among executable code.

It is based on **markdown**, a markup language that is widely used to generate html pages. You can learn more about markdown with the following tutorial: **[click here](#)**

# R markdown

You can start an R markdown document in RStudio by clicking on **File**, **New File**, then **R markdown**. You will then be asked for a title and author.

Once you gain experience with R markdown, you will be able to do this without the template and can simply start from a blank template.

As a convention, we use the `.Rmd` suffix for R markdown files

# Compiling the document using knitr

With R markdown, you need to **compile** the document into the final report. We use the `knitr` package to do this. The specific function used to compile is the `knit` function, which takes a filename as input.

RStudio provides a `Knit` button that makes it easy and convenient to compile the document.

# The Header

At the top of the document is the R markdown header:

```
---  
title: "Nanostring Analysis"  
author: "Evan Johnson"  
date: "12/5/2019"  
output: html_document:  
---
```

R markdown reports can be to be in HTML, PDF, Microsoft Word, or presentation formats. By changing the output to, for example `pdf_document` or `word_document`, we can control the type of output that is produced.

# The Header

Other output options include code folding, themes, etc.:

```
---  
title: "Nanostring Analysis"  
author: "Evan Johnson"  
date: "12/5/2019"  
output:  
  html_document:  
    code_folding: hide  
    toc: true  
    toc_float: true  
    theme: "flatly"  
editor_options:  
  chunk_output_type: console  
---
```

# R Code Chunks

In various places in the document, we see something like this:

```
```{r}  
summary(pressure)  
```
```

These are the **code chunks**. When you compile the document, the R code inside the chunk, in this case `summary(pressure)`, will be evaluated and the result included in that position in the final document.



# R Code Chunks

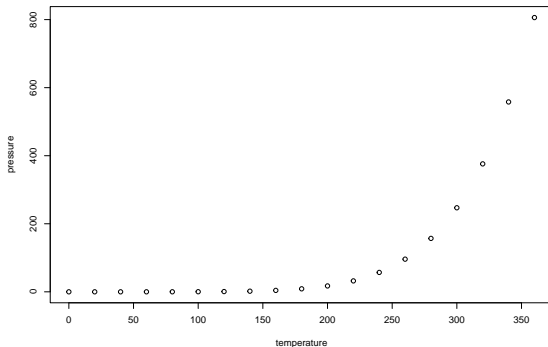
Its a good habit to adding a label to the R code chunks. This will be very useful when debugging, among other situations. You do this by adding a descriptive word like this:

```
```{r pressure-summary}  
summary(pressure)  
```
```

## R Code Chunks

We can also add plots to our report:

```
```{r, out.width = '60%', fig.align = 'center'}  
plot(pressure)  
```
```



The options `out.width` and `fig.align` adjust the figure size and location.

## R Code Chunks

By default, the code will show up as well. To avoid having the code show up, you can use an argument. To avoid this, you can use the argument `echo=FALSE`. For example:

```
```{r, echo=FALSE}  
summary(pressure)  
```
```

If you want to include the code in the document, but not run the code and/or include results, you can use the argument `eval=FALSE`:

```
```{r, eval=FALSE}  
# You can install the tidyverse using:  
install.packages(tidyverse)  
```
```

## R Code Chunks (global knitr options)

One of the R chunks may contain a complex looking call:

```
```{r setup, include=FALSE}  
knitr::opts_chunk$set(echo = TRUE)  
```
```

The `include=FALSE` option in the chunk call will tell the system to run the R code, but not include it in the html/pdf/word report.

The R code in the chunk sets the `knitr` default to `echo=TRUE` in the call for any R chunks following this one.

## Other Code Chunks

knitr can execute code in many languages besides R:

```
```{python}
x = [2, 1, 3]
print(x[0], "Hello Python!")
```
```

## 2 Hello Python!

Some of the available language engines include: Python, SQL, Bash, Rcpp, Stan, JavaScript, and CSS. These will also have `echo=F`, so they are not evaluated.

# L<sup>A</sup>T<sub>E</sub>X Equations

One exciting feature about R markdown is that it allows you to include L<sup>A</sup>T<sub>E</sub>X equations to be rendered in html, pdf, or Word.

For example, to show the probability under the normal curve in the interval  $(a, b)$ , you can use the L<sup>A</sup>T<sub>E</sub>X code:

```
\mbox{Pr}(a < x < b) =
  \int_a^b \frac{1}{\sqrt{2\pi}\sigma}
  e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \, dx
```

This will be rendered by knitr as:

$$\Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$$

## Other Options (tabsets)

One very useful tool in Markdown is to add tabs to the html document:

```
## This is a header {.tabset}  
### This is Tab 1  
### This is Tab 2  
## This is a new header (not in the tabs)
```

The `{.tabset}` option will make all lower level subheadings tabs in the html. The next same or higher level header will break out of the tabset.

## Other Options (tabsets)

**Pro tip:** tabs can be generated dynamically:

```
## My Header {.tabset}  
```${r, results = 'asis'}  
n <- 10  
for (i in 1:n){  
  cat("###" , "Tab", i, "\n")  
  print(i)  
  cat("\n\n")  
}  
```
```



## Other Options

We will explore other R markdown options: headers, tabsets, and  $\text{\LaTeX}$  equations in class and in your Exercises. `\vskip .2` in Note: From now on, all R-based Exercises should preferably be done using an R markdown document. Your document should include headers, descriptive text, equations, R code, and plots/figures!

## More on R markdown

There is a lot more you can do with R markdown. We highly recommend you continue learning as you gain more experience writing reports in R. There are many free resources on the internet including:

- 1 RStudio's tutorial: <https://rmarkdown.rstudio.com>
- 2 The cheat sheet: <https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf>
- 3 The knitR book: <https://yihui.name/knitr/>

# Exercises

R markdown is a powerful tool for literate programming. To gain more practice, recreate the .Rmd file for the example file: "Rmarkdown\_example.html."

# Session info

```
sessionInfo()
```

```
## R version 4.4.0 (2024-04-24)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib; LAPACK version 3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats      graphics  grDevices utils      datasets  methods   base
##
## other attached packages:
## [1] dslabs_0.8.0
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.35    fastmap_1.1.1    xfun_0.43        Matrix_1.7-0
##  [5] lattice_0.22-6   reticulate_1.36.1 knitr_1.46        htmltools_0.5.8.1
##  [9] png_0.1-8        rmarkdown_2.26   tinytex_0.50      cli_3.6.2
## [13] grid_4.4.0       compiler_4.4.0   rprojroot_2.0.4   here_1.0.1
## [17] rstudioapi_0.16.0 tools_4.4.0      evaluate_0.23     Rcpp_1.0.12
## [21] yaml_2.3.8       jsonlite_1.8.8   rlang_1.1.3
```