

# STATS 266 Handout - Introduction & Basics

Qi Wang

2025-02-27

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Installing R . . . . .	2
1.2	Installing RStudio . . . . .	3
1.3	RStudio Interface Overview . . . . .	3
<b>2</b>	<b>Basic R Commands</b>	<b>4</b>
2.1	Using R as a calculator . . . . .	4
2.2	Assign Values . . . . .	5
2.3	Logical Operations with R . . . . .	7
<b>3</b>	<b>Vectorization</b>	<b>7</b>
<b>4</b>	<b>Remove a variable</b>	<b>8</b>
<b>5</b>	<b>Project Management</b>	<b>8</b>
<b>6</b>	<b>Version Control - GitHub</b>	<b>9</b>
6.1	Prerequisites . . . . .	9
6.2	Create a Repository on GitHub . . . . .	9
6.3	Clone the Repository in RStudio . . . . .	10
6.4	Make Local Changes and Commit . . . . .	10
6.5	Push Changes to GitHub . . . . .	10
6.6	Verify Changes on GitHub . . . . .	10
6.7	Additional Resources . . . . .	10
<b>7</b>	<b>Acknowledgement</b>	<b>10</b>

# 1 Introduction

Welcome to **STATS 266: Introduction to R**. This handout provides an overview of R, including its installation, basic operations, and data handling capabilities. By the end of this document, you should be able to:

- Understand the purpose of R and RStudio.
- Install and set up R for your coursework.
- Learn basic R syntax, project managements.

## 1.1 Installing R

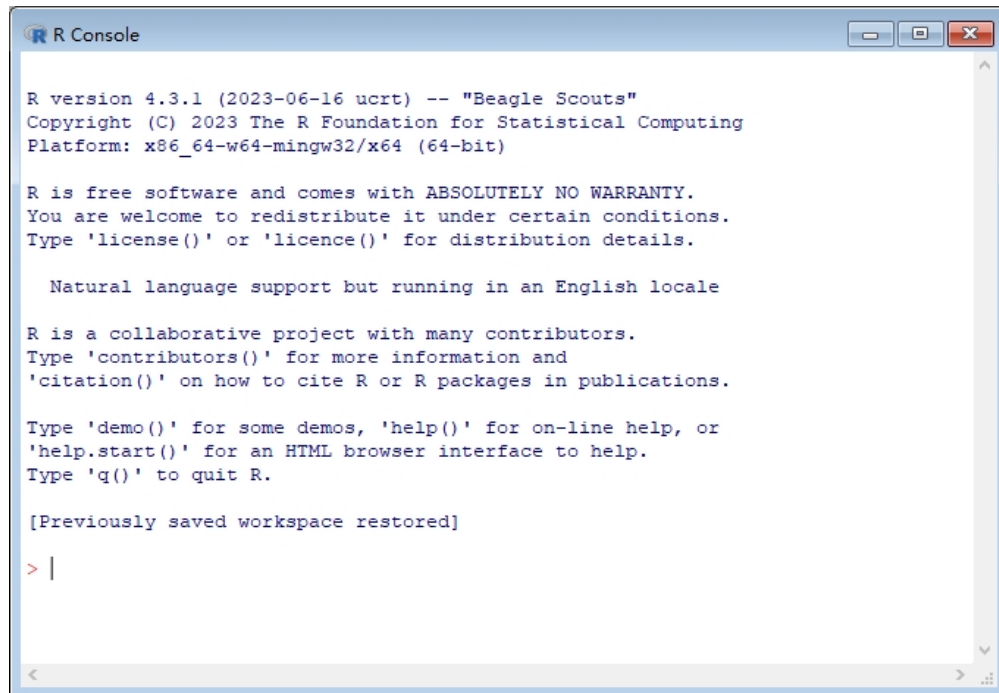
**R** is a programming language specifically designed for **statistical computing, data analysis, and visualization**. It is widely used in academia, research, and industry. R is:

- Open-source and free to use.
- Strong statistical capabilities with built-in functions.
- Data visualization tools like ‘ggplot2’.
- Extensive community support with thousands of packages available on CRAN.

Installing R To install **R**, follow these steps:

- Visit the [CRAN website](#).
- Select your operating system (Windows / macOS / Linux).
- Download and install the latest version.

Find **R** on your computer and open it, it shows like:

A screenshot of the R Console window. The title bar says "R Console". The text inside the console is as follows:

```
R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"
Copyright (C) 2023 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

## 1.2 Installing RStudio

RStudio is an **Integrated Development Environment (IDE)** for R that provides:

- A user-friendly interface.
- Code organization tools.
- An embedded console and visualization panel.

Installation Steps:

- Visit [RStudio Download Page](#)
- Choose your operating system and download RStudio Desktop.
- Install and open RStudio.

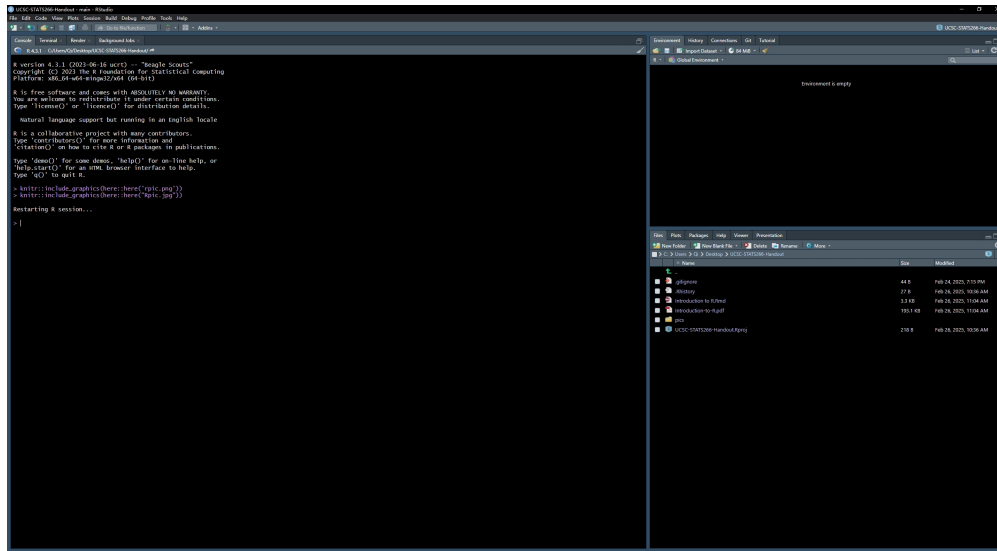
**Warning:** You will need to install R before installing RStudio.

## 1.3 RStudio Interface Overview

After opening **RStudio**, you will see four main panels:

- Console: Where you enter R commands.
- Environment/History: Displays variables and past commands.

- Files/Plots/Packages: Manages files, plots, and installed packages.
- Script Editor: Write and save R scripts (‘.R’ files).



## 2 Basic R Commands

### 2.1 Using R as a calculator

R is a computer language for scientific computing, so we can use it as a calculator.

```
(10+23-45)/6^7
```

```
## [1] -4.286694e-05
```

What does  $e - 05$  mean? It actually means  $10^{-5}$ . By writing in this way, the result is easier to read. In all,

- plus: +
- minus: -
- times: \*
- divide: /
- logarithm: log()
- exponentiation: exp()
- sine, cosine, tangent: sin(), cos(), tan()

Some more operations:

- modulo: `%%`
- absolute value: `abs()`
- round: `round()`
- round up: `ceiling()`
- round down: `floor()`

```
print(c(5%%2,abs(-5),round(1/3),ceiling(3/4),floor(3/4)))
```

```
## [1] 1 5 0 1 0
```

Take `abs()` as an example—this differs from simple arithmetic operations. In R, such constructs are called **functions**. We will explore how to write functions in detail later. Similar to mathematical functions, a function in R takes an input, processes it, and returns an output. In this case, `abs()` takes a number as input and returns its absolute value. Later in this course, you will learn how to write your own custom functions. For now, if you don't know how to use some function, just type `?` in front of the function. For example:

```
?abs()
```

```
## starting httpd help server ... done
```

You will see what arguments it needs, and what values it outputs.

## 2.2 Assign Values

We definitely want to use some symbols to represent the values, that improves the readability of the code. We use “<-” or “=”. See the following example:

```
x <- 0.5
y = log(x)
print(c(x, log(x), log(0.5)))
```

```
## [1] 0.5000000 -0.6931472 -0.6931472
```

```
print(c(y, exp(y), exp(log(0.5))) )
```

```
## [1] -0.6931472 0.5000000 0.5000000
```

Note: <- is generally recommended since it indicates the direction.

So if we want to repetitively use the value, we had better assign it to some variable, and it will be saved for future use. However, not all names are legal for variables:

- Variable names can contain letters, numbers, underscores and periods but no spaces.
- They must start with a letter or a period followed by a letter (they cannot start with a number nor an underscore).
- Variables beginning with a period are hidden variables.

Different people use different conventions for long variable names, these include:

- periods.between.words (Sometimes mess up with Python!)
- underscores\_between\_words (My personal recommendation.)
- camelCaseToSeparateWords (Hard to read!)
- periods.between.words (Sometimes mess up with Python!)
- underscores\_between\_words (My personal recommendation.)
- camelCaseToSeparateWords (Hard to read!)

Exercise: Which of the following are valid R variable names?

- min\_height (Y)
- max.height (Y)
- \_\_age
- .mass
- MaxLength (Y)
- min-length
- 2widths
- celsius2kelvin (Y)

Warning: Some dangerous assigning will cause trouble.

```
print(c(pi,sin(pi/6)))
```

```
## [1] 3.141593 0.500000
```

```
pi <- 2
print(c(pi, sin(pi/6)))
```

```
## [1] 2.0000000 0.3271947
```

Avoid assigning values to names that conflict with built-in variables or functions. This applies to variables, functions, and data frames. Additionally, when using packages, be mindful that your variable names do not coincide with function or object names within those packages. This is dangerous because although it will not report a bug, your future experiential result could be affected.

## 2.3 Logical Operations with R

We can also do comparisons in R, there are some logical operators:

```
print(c(1==1, 1==2))
```

```
## [1] TRUE FALSE
```

```
print(c(1>2, 1<2))
```

```
## [1] FALSE TRUE
```

```
print(c(1>2 & 1<2))
```

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

```
print(c(1>2 | 1<2))
```

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

- and: &
- or: |
- compares: >, <, ==, >=, <=, !=

Logical values in R has 0 (FALSE) or 1 (TRUE) values.

## 3 Vectorization

One final thing to be aware of is that R is vectorized, meaning that variables and functions can have vectors as values. In contrast to physics and mathematics, a vector in R describes a set of values in a certain order of the same data type. For example:

```
print(c(1:5))
```

```
## [1] 1 2 3 4 5
```

```
print(c(1:5) - c(6:10))
```

```
## [1] -5 -5 -5 -5 -5
```

```
print(2^(1:5))
```

```
## [1] 2 4 8 16 32
```

## 4 Remove a variable

There are sometimes after you assign values to a variable, you want to remove it from the environment to save the RAM:

```
x <- 1 # See the "Environment" panel in RStudio  
rm(x) # Check the "Environment" panel in RStudio again
```

One important thing is to remove all variables before running a session:

```
rm(list = ls())
```

## 5 Project Management

We're going to create a new project in RStudio:

1. Click the “**File**” menu button, then “**New Project**”.
2. Click “**New Directory**”.
3. Click “**New Project**”.
4. Type in the name of the directory to store your project, e.g., `my_project`.
5. If available, select the checkbox for “**Create a git repository**.”
6. Click the “**Create Project**” button.

Sometimes, we need to read some data into R to make further analysis. To begin, we need to know, where we are now:



```
getwd()
```

```
## [1] "C:/Users/Qi/Desktop/UCSC-STATS266-Handout"
```

We can also set a new path to work at:

```
setwd("./")
```

By making a project, we can assign data together with the code and use some functions to read data under the file path of the project.

```
here::here()
```

```
## [1] "C:/Users/Qi/Desktop/UCSC-STATS266-Handout"
```

## 6 Version Control - GitHub

This part of materials are from <https://happygitwithr.com/rstudio-git-github>.

### 6.1 Prerequisites

Ensure you have:

- A registered free GitHub account.
- Installed or updated R and RStudio.
- Installed Git on your system.
- Configured Git with your username and email.
- Verified the ability to push to and pull from GitHub via the command line.

### 6.2 Create a Repository on GitHub

1. Log in to your GitHub account.
2. Click the "New" button near "Repositories" to create a new repository.
3. Fill in the repository details:
  - Repository name: e.g., `myrepo`
  - Description: "Repository for testing my Git/GitHub setup"
  - Public: Select this option.
  - Initialize this repository with\*: Add a README file.
4. Click "Create repository".
5. Click the "Code" button and copy the repository's HTTPS URL.

### 6.3 Clone the Repository in RStudio

1. Open RStudio.
2. Go to File, New Project, Version Control, Git.
3. In the "Repository URL" field, paste the copied URL from GitHub.
4. Specify the local directory where you want to store the project.
5. Click "Create Project".

### 6.4 Make Local Changes and Commit

1. In RStudio's Files pane, open README.md.
2. Add a line, e.g., `This is a line from RStudio.`
3. Save the file.
4. Click the "Git" tab.
5. Stage the README.md file by checking its box.
6. Click "Commit".
7. Enter a commit message, e.g., `"Commit from RStudio."`
8. Click "Commit."

### 6.5 Push Changes to GitHub

1. In the Git tab, click "Push" to upload your commit to GitHub.
2. If prompted, enter your GitHub credentials.

### 6.6 Verify Changes on GitHub

1. Navigate to your GitHub repository in a web browser.
2. Confirm that the changes to README.md appear in your repository.

### 6.7 Additional Resources

For more details, refer to:

[Happy Git and GitHub for the useR](#)

## 7 Acknowledgement

This teaching material is adapted from the previous material of this course made by [Marcela Alfaro-Córdoba](#) and [Sheng Jiang](#).