

Lecture 1: Introduction: Variables, data types, and data structures

Stats 32: Introduction to R for Undergraduates

Harrison Li

April 2, 2024

Agenda

- 1 Course Logistics
- 2 Introduction to R
- 3 Variables and data types
- 4 Data structures
- 5 Indexing

Reading: Sections 1.1, 1.2

Course Logistics

Course outline

4 major units:

- Fundamentals (1 week)
 - Data structures, functions, packages
- Data cleaning and wrangling (1 week)
 - Tidy data, dplyr verbs
- Data visualization (1 week)
 - Basic univariate and multivariate visualizations using ggplot2
- Data analysis (2 weeks)
 - Linear regression and extensions, A/B testing, model validation and selection

See syllabus on Canvas for more detailed information

Meetings and textbook

This class will meet for the first five weeks of the quarter on Tuesdays and Thursdays from 12 noon to 1:20 pm in Mitchell Earth Sciences B67.

Laptops required for each class session — much of the learning in this class will be through interactive labs.

This course is loosely based on the textbook *Statistical Inference via Data Science: A ModernDive into R and the Tidyverse* by Chester Ismay and Albert Y. Kim. It is freely available online at <https://moderndive.com>.

Assignments and grading

- Completion of in-class labs in groups (20%, starting 4/4)
- Weekly homework assignments (80%, 4 total)

Although this is a 1 unit course, since we only meet for half of the quarter and for 3 hours/week (instead of 1 hour), expect the workload for the course to be closer to that of a 2-3 unit course during the first five weeks.

Introduction to R

What is R?

- A free programming language specifically designed for statistical computing.
- The language of choice for statisticians; increasingly popular for anyone who needs to wrangle with, visualize, and/or analyze data!
- A modern implementation of the earlier language S, developed at Bell Labs in the 1970s.

R's strengths

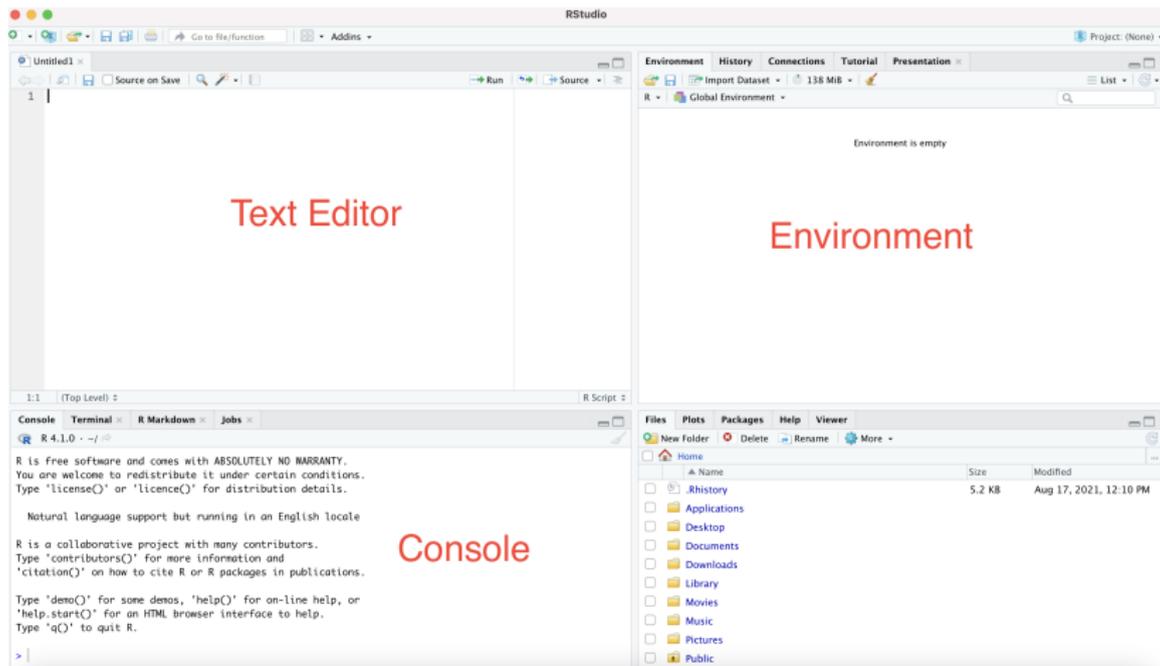
- *Packages*. R has a comprehensive collection of statistical packages; new statistical methods are often first implemented in R.
- *Versatility*. Use R to write code, develop online data dashboards (R Shiny), generate professional technical reports (R Markdown), and create presentation slides like this one (Beamer).
- *Open source*. R is free to use for everyone, and anyone can view the underlying source code, or (conversely) contribute code.
- *Easier syntax*. Relative to other languages, R's syntax is less nitpicky, making it easier to pick up.

R's weaknesses

- *Slowness.* Due to the high-level functionality R provides, it tends to run significantly more slowly than lower-level languages (C, Java, etc.). We will not focus on speed in this class.
- *Uneven documentation.* By virtue of being open source, R's documentation is less polished than that of some other similar languages (MATLAB, Maple, etc.). Internet forums are your friend!

RStudio

RStudio is an Integrated Development Environment (IDE) providing a user-friendly way to interact with R.



The **console** (bottom left section of the RStudio window) allows you to type commands directly into R, one line at a time.

The top left section of the RStudio window is a **text editor**, for creating and saving longer files (say with `.R` or `.Rmd` extensions)

Finally, the top right section displays the **environment**, which requires an understanding of **variables** or **objects**.

Variables and data types

Variables and data types

A **variable** (more formally an **object**) is a *named container* for data. This data can be of various **types**:

- *Numeric*: A number, e.g. -4, 100, or 0.1273
- *Logical*: Either TRUE or FALSE
- *Character*: Text, e.g. "I love Stats 32" (note the quotes)
- *Factor*: A non-numeric variable taking on one of several pre-specified values, known as the *levels*, e.g. eye color can be "blue", "brown", or "green"

Assignment

A variable is **assigned** (given a value) using the `<-` operator, or alternatively, the `=` sign (not recommended):

```
myVariable <- 3
```

To display the value of a variable, you can simply enter its name in the console:

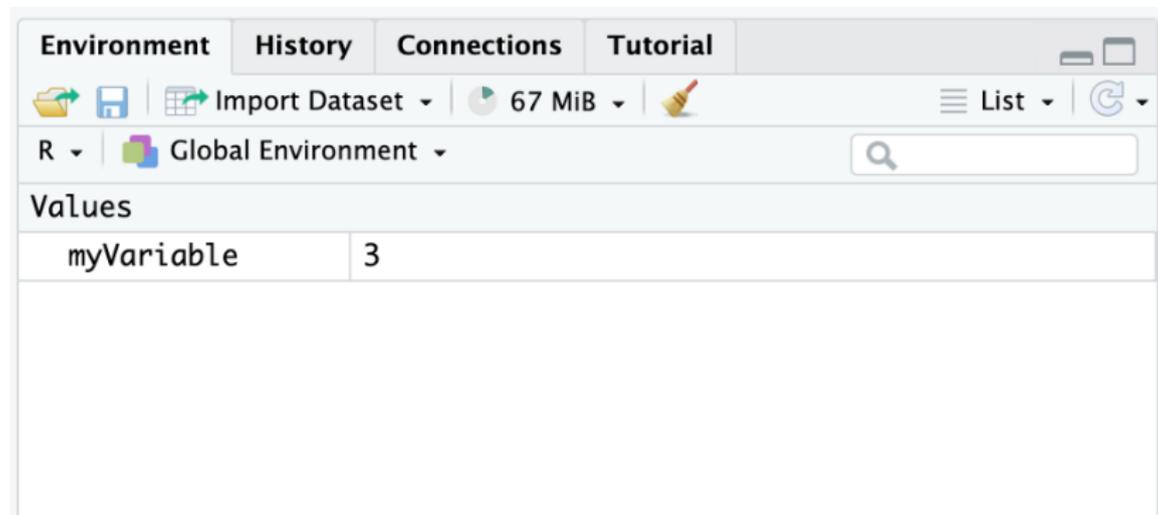
```
myVariable
```

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

Assignment

Once a variable is assigned, it becomes a part of the **environment**: the space of all objects that have been defined.

Recall the environment section is the top right corner of RStudio. It displays the value of all variables in the environment.



The screenshot shows the RStudio Environment pane. At the top, there are tabs for 'Environment', 'History', 'Connections', and 'Tutorial'. Below the tabs, there are icons for file operations and a status bar showing '67 MiB'. The main area displays 'R' and 'Global Environment' with a search box. Below this, the 'Values' section contains a table with one row: 'myVariable' with the value '3'.

Values	
myVariable	3

Variable reassignment

```
myVariable <- 3  
myVariable
```

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

```
myVariable <- 4  
myVariable
```

```
## [1] 4
```

Basic arithmetic operations

R supports basic arithmetic operations for *numeric* variables:

- *Addition* using `+`
- *Subtraction* using `-`
- *Multiplication* using `*`
- *Division* using `/`
- *Exponentiation* using `^`
- *Modular division* using `%%`: given integers `a` and `b`, `a %% b` returns the remainder when `a` is divided by `b`

Basic arithmetic operations

```
x <- 2  
y <- -4  
z <- 17
```

```
x-y
```

```
## [1] 6
```

```
3+2*x^2
```

```
## [1] 11
```

```
z %% x
```

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

Converting and verifying data types

```
x <- "32"  
y <- as.numeric(x)  
y
```

```
## [1] 32
```

```
is.numeric(y)
```

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

```
as.character(y)
```

```
## [1] "32"
```

```
is.character(as.character(y))
```

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

Data structures

Data structures

Data can be organized into a **data structure** for ease of manipulation. Here are some basic data structures in R:

- *Vector*: A 1-D collection of data of the *same* type, created using `c()`
- *Matrix*: A 2-D array of data of the *same* type, created using `matrix()`
- *List*: A 1-D collection of data of *arbitrary* types, created using `list()`
- *Data frame*: A 2-D array of data with named columns; each column is a vector. Created using `data.frame()`. We will learn more about data frames in Lecture 2. Useful fact: a data frame is in fact a list under the hood, where each element of the list is a column of the data frame (which is itself a vector)

Vectors

```
myNumericVector <- c(2, -1, 5)
myCharacterVector <- c("S", "t", "a", "t", "s")
myNumericVector
```

```
## [1] 2 -1 5
```

```
myCharacterVector
```

```
## [1] "S" "t" "a" "t" "s"
```

Vectorization

Note that arithmetic operations in R are “vectorized” for convenience, meaning they will apply to each element of a vector in the expected way:

```
myNumericVector + 1
```

```
## [1] 3 0 6
```

```
myNumericVector + c(1, 1, 1)
```

```
## [1] 3 0 6
```

```
myNumericVector^2
```

```
## [1] 4 1 25
```

```
myNumericVector * c(0, 3, 2)
```

```
## [1] 0 -3 10
```

Matrices

```
myMatrix <- matrix(data=c(4,3,1,2), nrow=2,  
                    ncol=2, byrow=TRUE)
```

```
myMatrix
```

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

Lists

```
myList <- list(2, "cat", c(3, 1))  
myList
```

```
## [[1]]  
## [1] 2  
##  
## [[2]]  
## [1] "cat"  
##  
## [[3]]  
## [1] 3 1
```

Named lists

Note that each entry in a list can also be given a name:

```
myNamedList <- list(integer=2, characterVector="cat", numericVector=c(3,1))
myNamedList
```

```
## $integer
## [1] 2
##
## $characterVector
## [1] "cat"
##
## $numericVector
## [1] 3 1
```

```
names(myNamedList)
```

```
## [1] "integer"          "characterVector" "numericVector"
```

Indexing

Indexing

You can access individual entries or other chunks of a larger data structure via *indexing*.

```
myNumericVector
```

```
## [1] 2 -1 5
```

```
myNumericVector[2]
```

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

```
myMatrix
```

```
##      [,1] [,2]
```

```
## [1,]  4  3
```

```
## [2,]  1  2
```

```
myMatrix[2,1]
```

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

```
myMatrix[,1]
```

```
## [1] 4 1
```

Indexing

```
myList
```

```
## [[1]]  
## [1] 2  
##  
## [[2]]  
## [1] "cat"  
##  
## [[3]]  
## [1] 3 1
```

```
myList[2]
```

```
## [[1]]  
## [1] "cat"
```

```
myList[c(1,3)]
```

```
## [[1]]  
## [1] 2  
##  
## [[2]]  
## [1] 3 1
```

Indexing

```
myList[2]
```

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

```
## [1] "cat"
```

Note `myList[2]` returns a *list* with a single element that is the character object “cat”. You need to use double square brackets to “fully” index into a list, i.e. `myList[[2]]` returns the character vector “cat”, rather than a list.

Indexing

For a named list, you can also index by name using the \$ operator:

```
myNamedList$characterVector
```

```
## [1] "cat"
```

This is equivalent to the following:

```
myNamedList[["characterVector"]]
```

```
## [1] "cat"
```