# Lab 0: An introduction to the R environment

Guoliang Ma

STAT 473/573 lab session
Spring 2023

# Setup and Installation

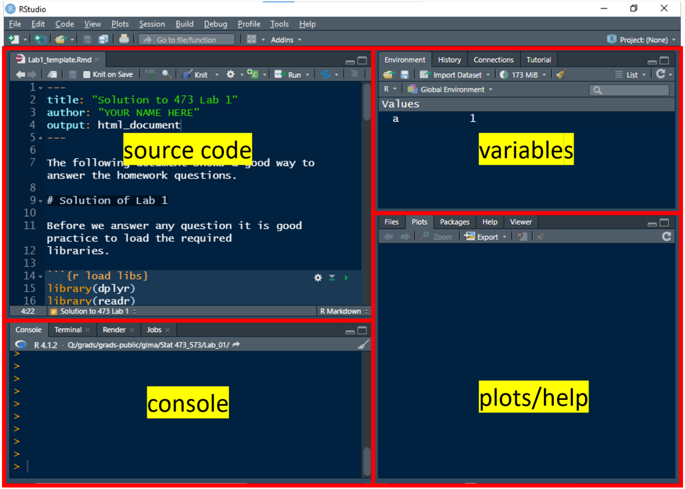
# The R executable and the IDE

This tutorial is for Windows OS. For mac OS, the installation might be slightly different, but the R code shouls work the same way.

1. R is a statistical computing language. You need to install it before install RStudio.
2. Go to `https://cran.r-project.org/bin/windows/base/` to download and install R .
3. An Integrated Development Environment (IDE)gives you more convenience to write your code. RStudio is the IDE we use for this course.
4. Go to `https://posit.co/download/rstudio-desktop/` and download RStudio once you have installed R .

# Layout of RStudio

1. The default layout of RStudio



~~~
---
title: "Solution to 473 Lab 1"
author: "YOUR NAME HERE"
output: html_document
---

The following document shows a good way to
answer the homework questions.

# Solution of Lab 1

Before we answer any question it is good
practice to load the required
libraries.

```{r load libs}
library(dplyr)
library(readr)
~~~

Values

| | |
|---|---|
| a | 1 |

```
R 4.1.2 · Q:/grads/grads-public/glma/Stat 473_573/Lab_01/
>
>
>
>
>
>
>
>
>
>
```

# Layout of RStudio

2. The source code area shows your current file. Select a chunk of code and press Ctrl + Enter to execute them.
3. The console area helps you write short (typically one-line) testing code. When this console is activated (by clicking anywhere in the console), pressing Enter executes the current line of code.
4. The variable area shows you the variables generated, including the data you loaded from elsewhere.
5. The plots/help area shows the plots you make. In the console, type ? followed by any command, and the help documentation will pomp up in this area.

# R source code and R markdown

Once you install R and RStudio, run RStudio. An empty file will be opened for you. This file is the `.R` file. In this course, we work with two types of files: `.R` and `.Rmd`.
⚠mind the capitalization!

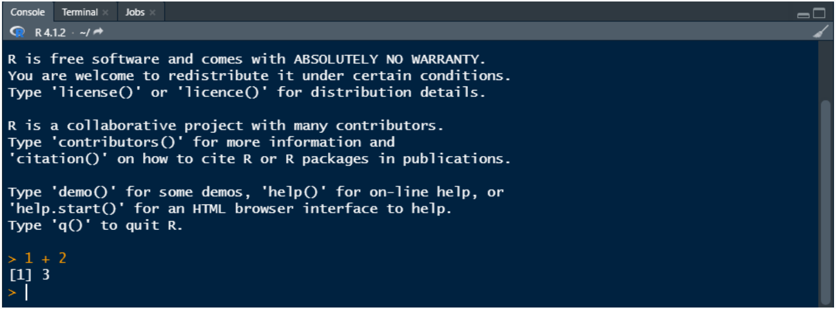
1. `.R` file is used for running “normal” code. When the intention is just coding and computing, use `.R`. You will use .R most of the time for your own statistical analysis/computing task.
2. `.Rmd` file is used for documentation. It can be compiled and a .pdf or .html file will be the output. You will use .Rmd most of the time for your lab homework.

# Basics

# Use the Console

The Console is a powerful interpreter of the R language. Meaning that you don't have to wait before your code to be translated into machine code. Instead, simply hit Enter and see the results.

1. basic calculation

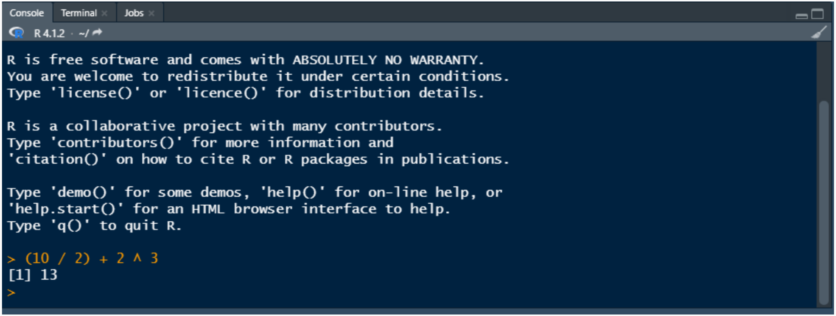


```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> 1 + 2
[1] 3
> |
```

# Use the Console

The Console is a powerful interpreter of the R language. Meaning that you don't have to wait before your code to be translated into machine code. Instead, simply hit Enter and see the results.

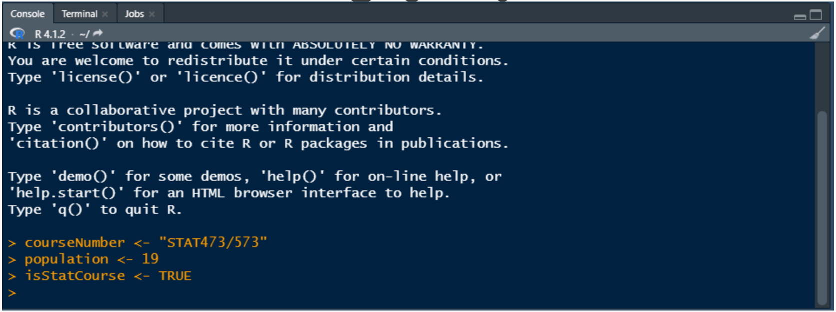
1. basic calculation



```
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.

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.

> (10 / 2) + 2 ^ 3
[1] 13
>
```

# Data types

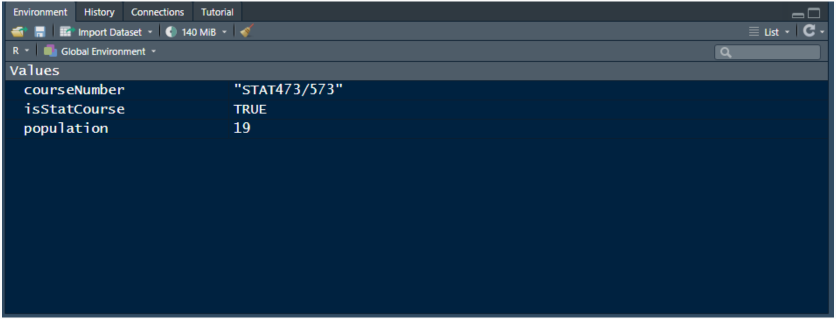
We mainly works with variables in R . The data types we'll see include: numeric, character, and logical. You need a variable name to hold the values of the variable. Use `<-` (`<` and `-`) to assign values to a variable.



```
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.

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.

> courseNumber <- "STAT473/573"
> population <- 19
> isStatCourse <- TRUE
>
```

# Data types

When a variable is created, it is recorded in the variable area.

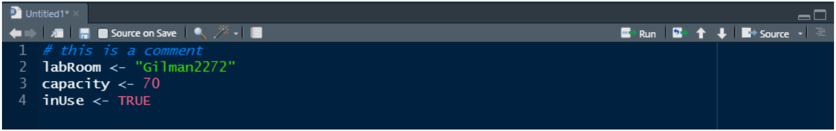


```
Values
  courseNumber          "STAT473/573"
  isStatCourse          TRUE
  population            19
```

1. Note the value of a character variable is quoted with ” and ”.
2. Note also that logical values are TRUE and FALSE instead of `True` or `true`.

# Use the source code area

Commands executed in the Console area will not be stored once you terminate the Console. However, we need to save the code for further uses. We write the code in the `.R` file in the source code area.

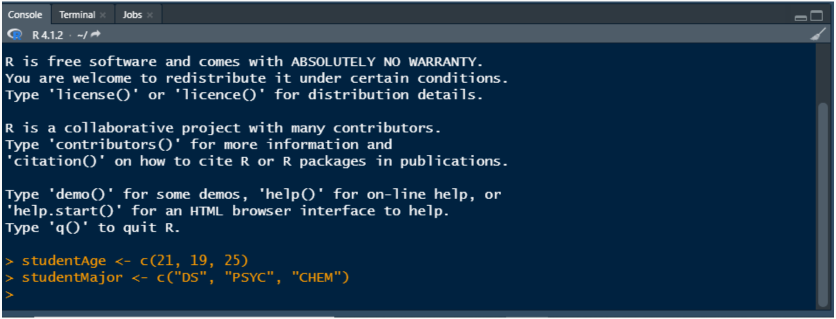


```
# this is a comment
labRoom <- "Gilman2272"
capacity <- 70
inUse <- TRUE
```

# Data structures: vector

Data structure in R differs from data types we just saw. We'll use vector, list, matrix, and data frame.

1. Vectors in R can save a sequence of values of the same type. Use c( ) and , to create a vector.

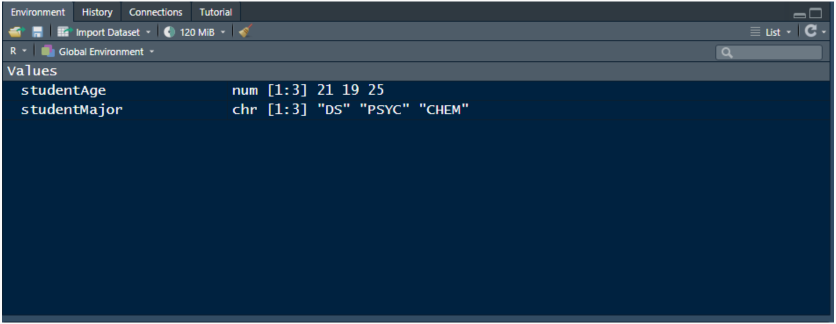


```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> studentAge <- c(21, 19, 25)
> studentMajor <- c("DS", "PSYC", "CHEM")
>
```

# Data structures: vector

Data structure in R differs from data types we just saw. We'll use vector, list, matrix, and data frame.

1. Vectors in R can save a sequence of values of the same type. Use c( ) and , to create a vector.



```
Values
  studentAge                num [1:3] 21 19 25
  studentMajor              chr [1:3] "DS" "PSYC" "CHEM"
```

# Data structures: vector

1. A simpler way to create sequences is to use `:` .

   ```
   sequence <- 1:5
   ```

2. How to create a sequence starting from 1, to 99, with step size 2? Type `?seq` in the Console to find out.

3. What will happen if I assign different types to a vector?

   ```
   difType <- c(1, "a", 5)
   ```

# Data structure: vector

1. Slicing a vector:
   1.1 Use [ and ] to slice a vector:
   ```
   difType[2]
   ```
   1.2 Note the starting index of a vector in R is 1
   1.3 For the vector from 1 to 99 with step size 2, how can I get the last 25 numbers?
2. Adding new elements: How do I append `100` as the 51st element of the vector?
3. Modifying existing values of a vector
   3.1 How to change the first element to 2?
   Hint: use the index.
4. Deleting elements from a vector

# Data structure: matrix

In R , a matrix is a two-dimensional structure with fixed numbers of rows and columns.

```
nums <- 0:11+1
```

Try the following two code. Can you see the difference?

```
matrix(nums, nrow=3)
```

```
matrix(nums, nrow=3, byrow=TRUE
```

�what is the difference between the following

```
nums <- 0:11+1
```

```
nums <- 0:(11+1)
```

�what is the output of

```
matrix(0:12), nrow=3
```

# Data structure: matrix

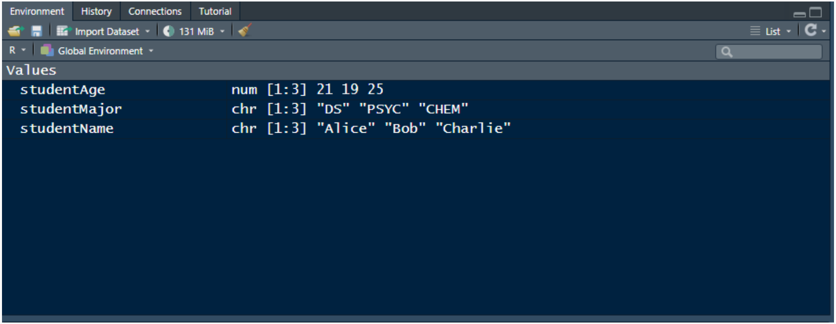
1. size of a matrix: `dim(mat)`
2. slicing a matrix: use double indexing: `mat[1, 2]` instead of `mat[1][2]`.

   �How would you select a submatrix?

   �How would you select the second columns of a matrix?
3. adding a row/column to a matrix by using `rbind`/`cbind`

# Data structure: data frame

We have a class of three students: Alice, Bob, and Charlie. Their ages are 21, 19, 25. Their majors are DS, PSYC, and CHEM. How do we store this information?



```
Values
  studentAge                  num [1:3] 21 19 25
  studentMajor                chr [1:3] "DS" "PSYC" "CHEM"
  studentName                 chr [1:3] "Alice" "Bob" "Charlie"
```

We can use the `cbind` function:

```
d <- cbind(studentName, studentAge, studentMajor)
```

# Data structure: data frame

�What is the type of `d`?

```
students <- data.frame(studentName, studentAge,
studentMajor)
```

�What happens in the variable area?

# Data structure: data frame

We will seldom add rows to a data frame. We frequently add columns to a data frame.

```
students$studentGrade <- c(95, 90, 100)
```

◆? What happens to your `student` data frame?

⚠ Note the $ operator in R .

# Data structure: data frame

Loading and saving data from and to `.csv` files.

1. 
   ```
   d <- read.csv("./data.csv")
   ```
2. 
   ```
   write.csv(students, "./students.csv")
   ```
   �Why is there no `<-` in the output command?

   �How would you read data from other sources, e.g., `.dat`, `.h5`, etc?

# R packages

# R packages

1. R is most powerful when you use its packages. We will use `knitr`, `dplyr`, `tidyr`, `readr`, `ggplot2`, `purrr`.
2. To install `dplyr`, in the Console, stype

   ```
   install.packages("dplyr")
   ```

   and Enter . Do the same for the other packages.
3. We introduce two ways to use an R package. You can

   3.1 load the whole package and use its functions by

   ```
   library("dplyr")
   ```

   ```
   select(data, colname)
   ```

   3.2 call the function of an installed package without loading the whole package

   ```
   dplyr::select(data, colname)
   ```

# The **knitr** package: Writing R markdown

The `knit` package is required to convert a source `.Rmd` to a .pdf or .html.

1. The title is at the beginning of a `.Rmd` file, surrounded by two “triple hyphens”:

   ```
   ---
   ```

   An example title is:

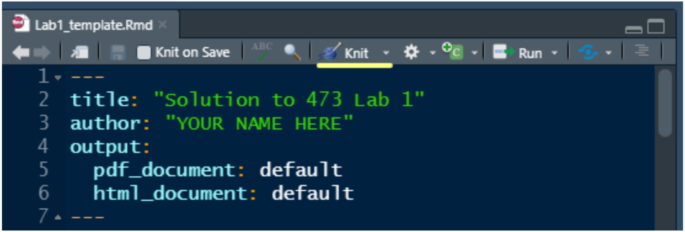
   ```
   ---
   title: "Solution to 473 Lab 1"
   author: "Guoliang Ma"
   output:
    pdf_document: default
    html_document: default
   ---
   ```

# The **knitr** package: Writing R markdown

2. Some keyword letters
   2.1 `#` : gives you level-one header
   2.2 `##` : gives you a level-two header
   2.3 `` ```{r} `` and `` ``` `` (the one above `Tab` ): are used to surround R code.
   2.4 `#` within an R code chunk defined by `` ```{r} `` and `` ``` `` : comments of the R code.

   Otherwise, type "normally" as you do with MS word or any other text editors.

3. Compile your `.Rmd` with `knit`. Use ▽ to select output format.



```
---
title: "Solution to 473 Lab 1"
author: "YOUR NAME HERE"
output:
  pdf_document: default
  html_document: default
---
```

# The **dplyr** package and the Pipe workflow in R

1. In R , `=` is less frequently used as an assignment operator. Instead, `<-` is more popular. For example, you want to create a variable `a` with value 3, use

   ```
   a <- 3
   ```

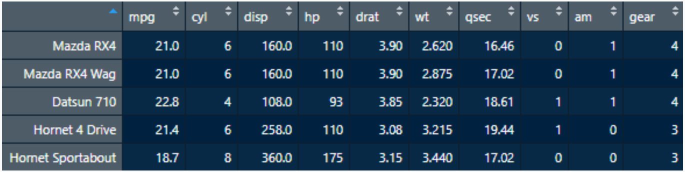
   Check out
   `https://stackoverflow.com/questions/1741820/what-are-the-differences-between-and-assignment-o`
   for reasons.

2. When indicating default function parameters, use `=` .

3. Built in `dplyr`, `tidyr`, and many other packages, is the operator `%>%` . This is called the Pipe workflow.

# The **dplyr** package and the Pipe workflow in

`dplyr` is a powerful tool for handling data. For example, consider this data set.



|  | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear |
|---|---|---|---|---|---|---|---|---|---|---|
| Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 |
| Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 |
| Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.215 | 19.44 | 1 | 0 | 3 |
| Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 |

We would like to calculate the group mean of `mpg` grouped by `gear`, number of gears.

# The **dplyr** package and the Pipe workflow in R

The workflow is group by gear
⇒ calculate means for grouped data.

The pipe workflow uses %>% to combine the workflow into one chunk of code:

```
data %>%
 group_by(gear) %>%
 summarise(meanByGear=mean(mpg))
```

There are many other useful functions in `dplyr`. When you want to achieve specific aims, search e.g., “group mean dplyr.”

# Other **`dplyr`** functions

1. `select`: select columns from a data frame
2. `mutate`: create/change the values of a column of a data frame
3. `subset`: select observations (rows)