




Lab 0: An introduction to the R environment

Guoliang Ma

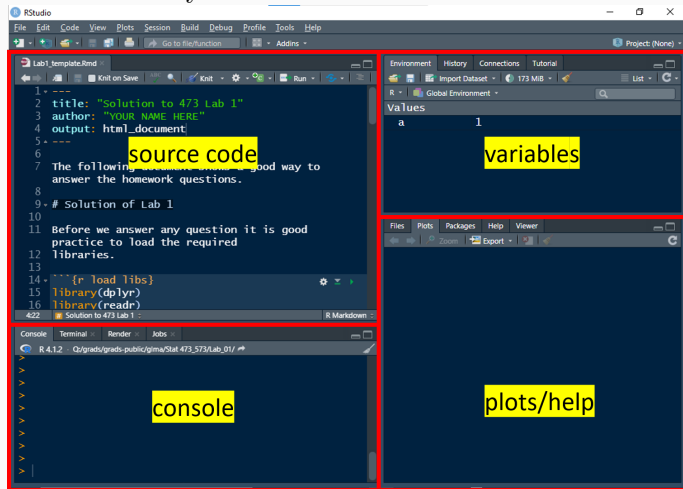
STAT 473/573 lab session
Spring 2023

The executable and the IDE

1.  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 .
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/downloads/` and download RStudio once you have installed .

Layout of RStudio

1. The default layout of RStudio



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 pop up in this area.


R source code and R markdown

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.

R packages

1.  is most powerful when you use its packages. We will use `knitr`, `dplyr`, `tidyr`, `readr`, `ggplot2`, `purrr`.

2. To install `dplyr`, type

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

and . 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 **knit** 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 **knit** package: Writing R markdown

2. Some keyword letters

2.1 `#`: gives you level-one header

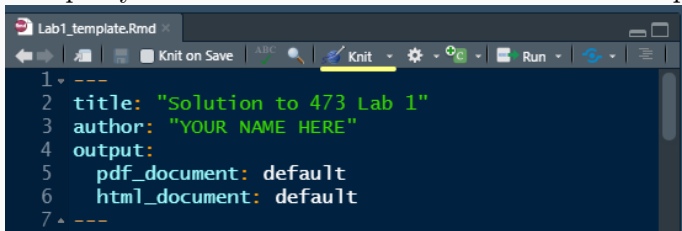
2.2 `##`: gives you a level-two header

2.3 ```{r}` and ````` (the one above `Tab`): are used to surround  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.



```
1 ---
2 title: "Solution to 473 Lab 1"
3 author: "YOUR NAME HERE"
4 output:
5   pdf_document: default
6   html_document: default
7 ---
```




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

1. In  ,  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-operator-in-r> 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

The workflow is `group by gear` \Rightarrow `calculate means for grouped data`. The pipe workflow uses `%>%` to combine the workflow into one chunk of code:

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

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