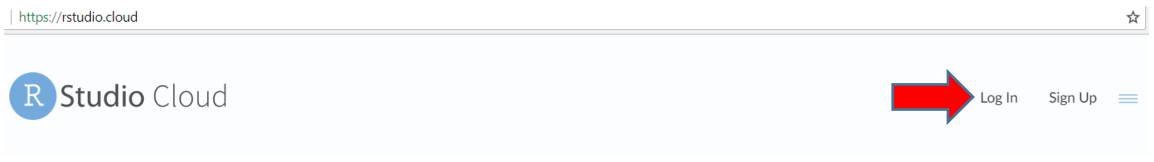
Business Data Mining Lab 1

Overview of R, RStudio, RStudio Cloud

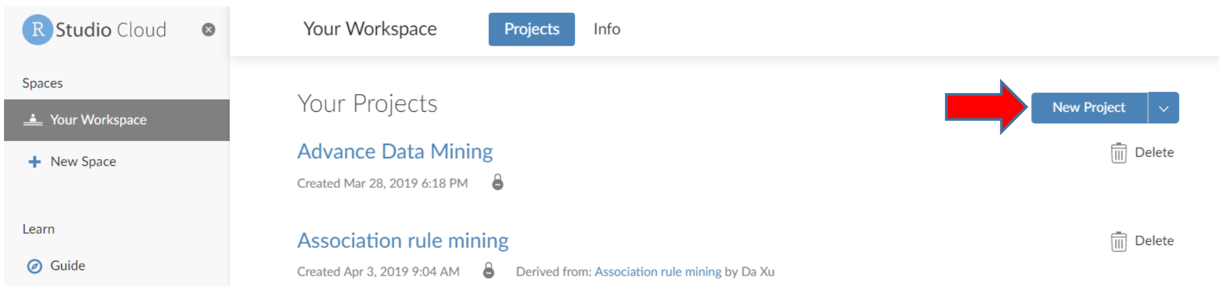
- R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical methods for data mining and modeling, such as linear and nonlinear regression, classical statistical tests, time-series analysis, classification, clustering and association rule mining.
- RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.
- RStudio Cloud is a cloud-based RStudio that allows people to use R directly from web browser and store everything on web server.

Introduction of RStudio Cloud

1. To access RStudio Cloud, go to https://rstudio.cloud/.
2. In the main page, click on "Log In" and sign in with your Gmail or GitHub account. Alternatively, you can also sign up for a new account.



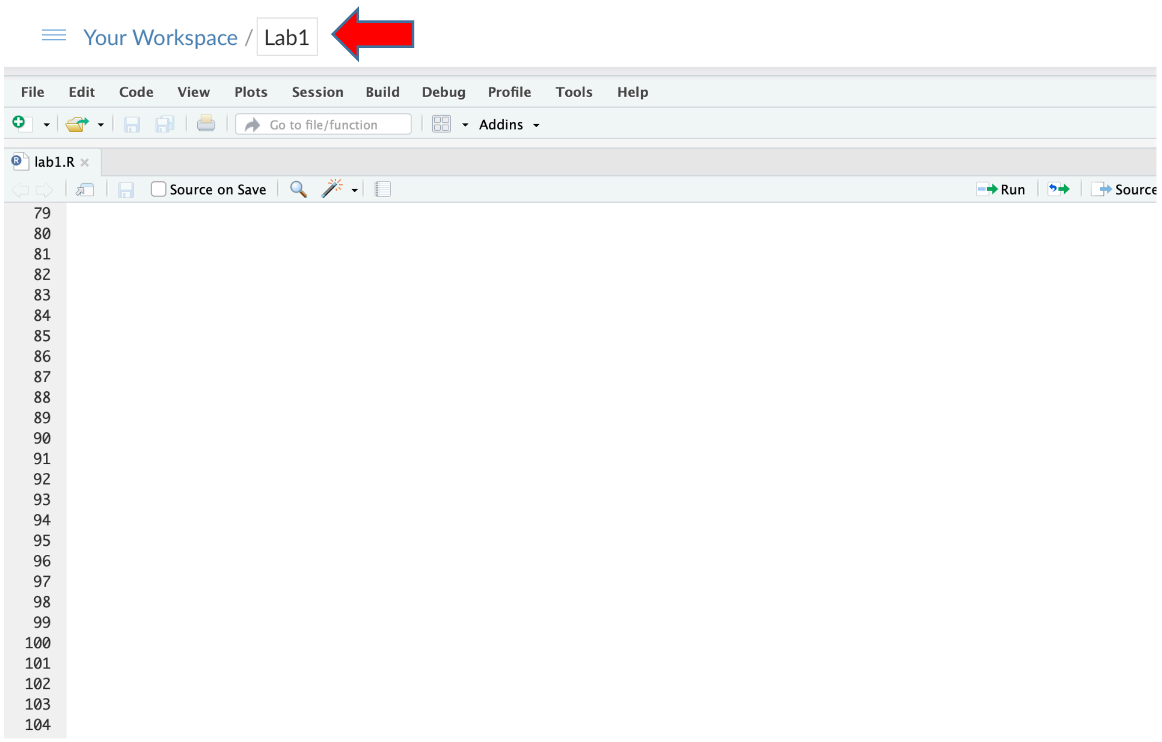
3. Once logged in, the main page shows your workspace that contains several existing projects. You can create project by clicking on "New Project" button.



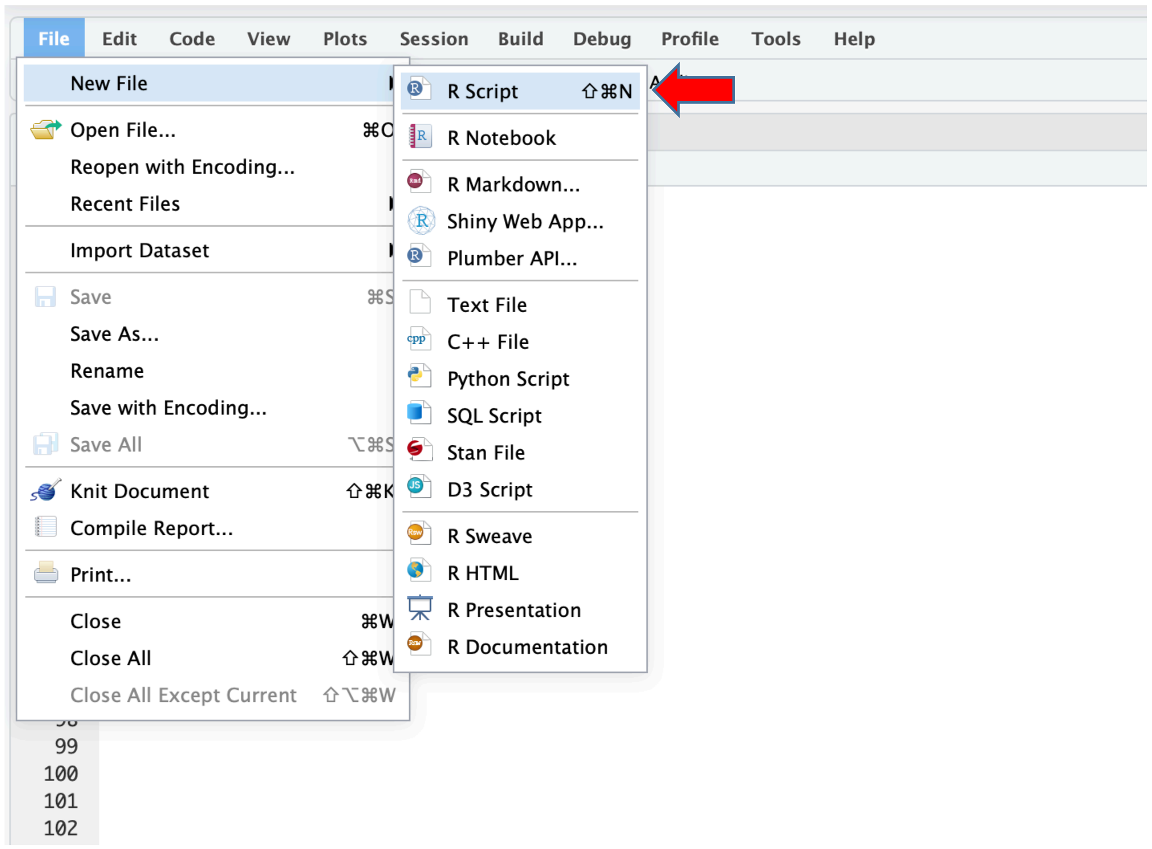
Advance Data Mining
Created Mar 28, 2019 6:18 PM

Association rule mining
Created Apr 3, 2019 9:04 AM Derived from: Association rule mining by Da Xu

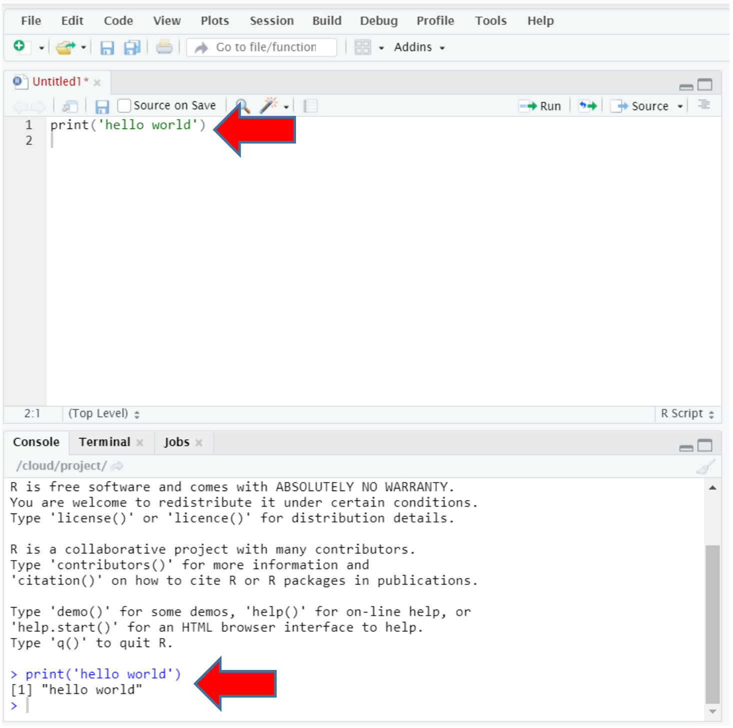
4. Once a new project created, you can change the name of project by clicking on "Untitled Project" and RStudio Cloud will automatically save the project for you in your workspace.



5. Since we are mainly writing R script to communicate with R, we need to create a new R script file in this project. To do that, click on “File” tab -> “New File” -> “R Script”.



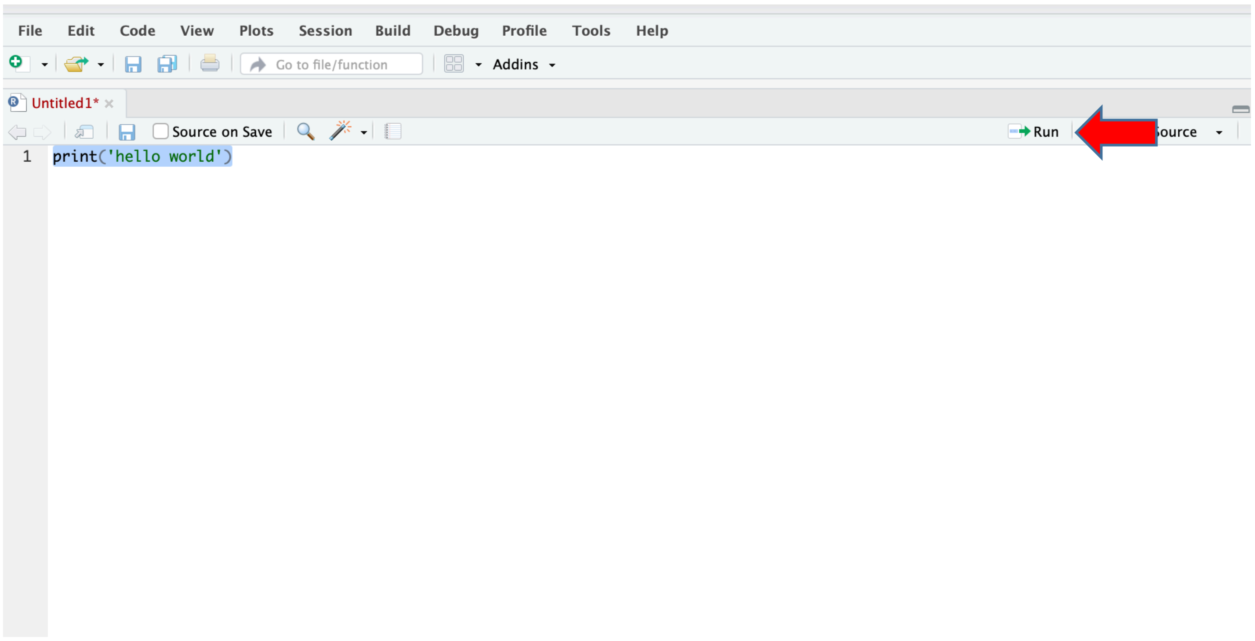
6. In the R script we just created, type *print('hello world')* and press "Ctrl + Enter" ("Command + Enter" for Mac user). R will execute this line of code and print "hello world" accordingly in the console below.



```
print('hello world')
```

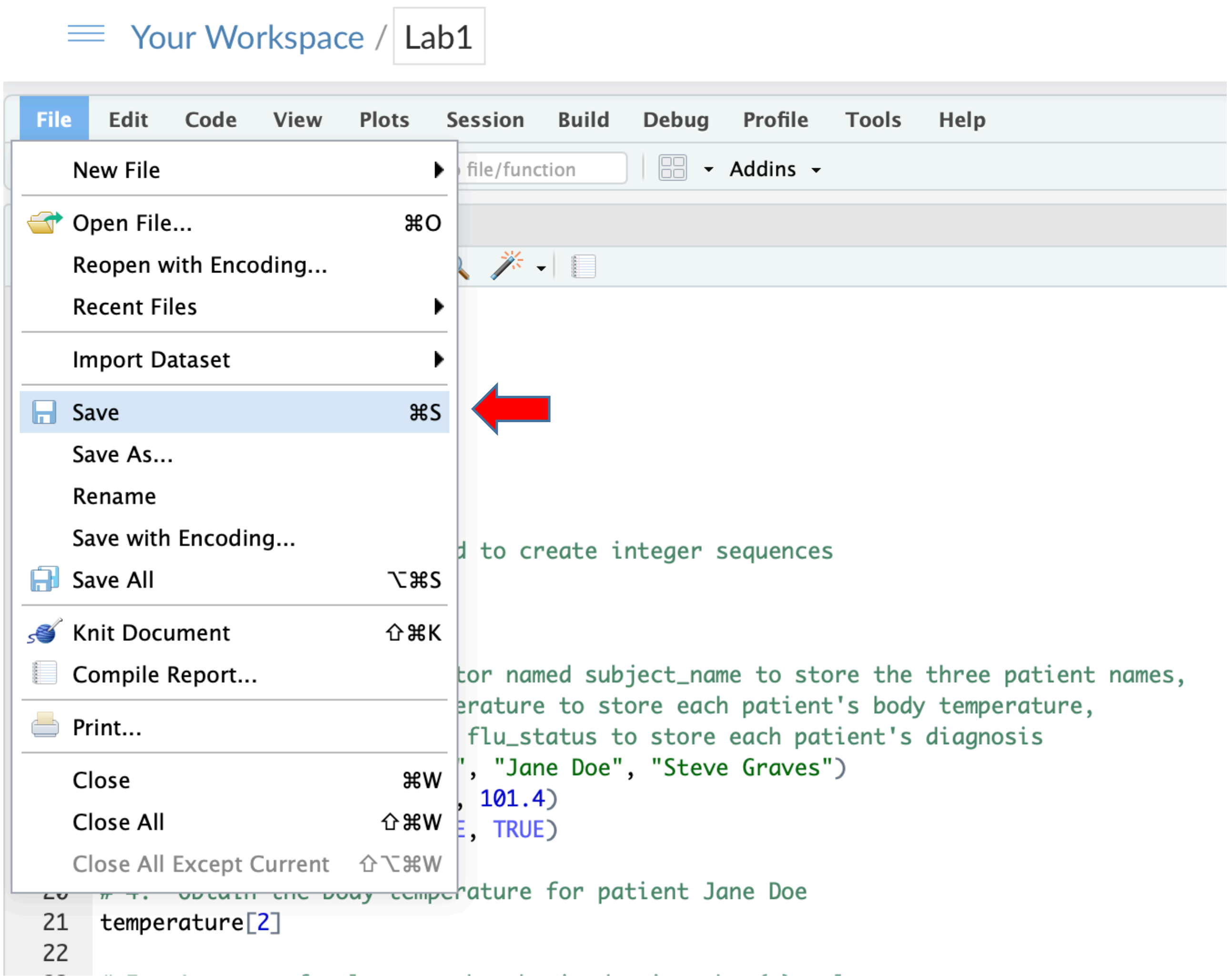
```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

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.

> print('hello world')
[1] "hello world"
>
```

7. Alternatively, we can run this line of code by selecting the code. Then, click on "Code" tab and select "Run Selected Line(s)".


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Untitled1*
Source on Save
Run
ource

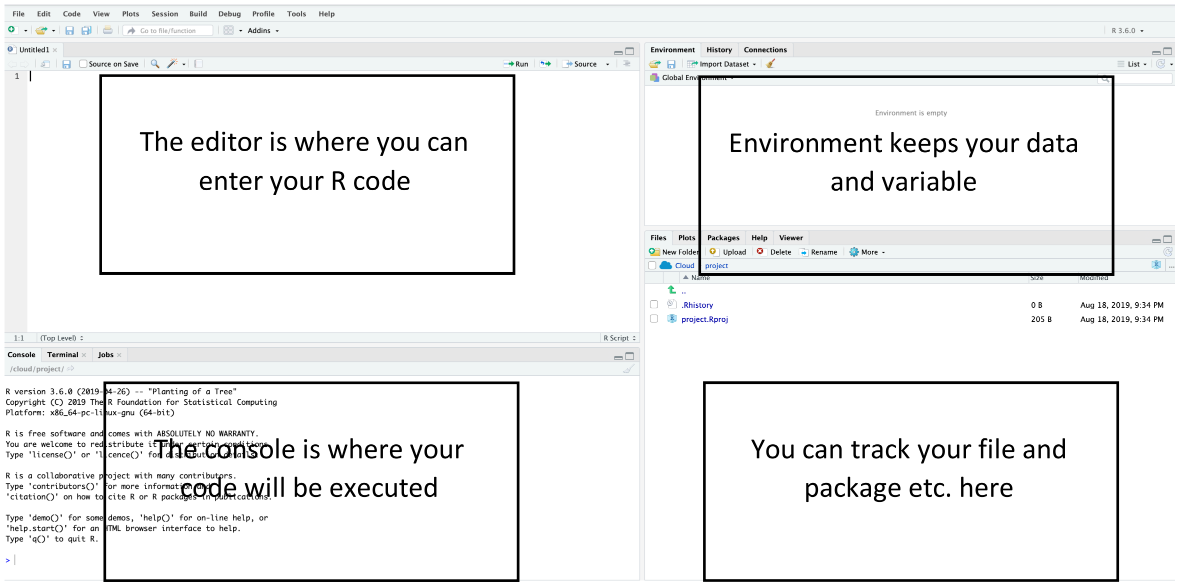
```
print('hello world')
```

8. To save the R script file, press "Ctrl + S". Alternatively, we can save R script by clicking on "File" tab and "Save".



```
d to create integer sequences

tor named subject_name to store the three patient names,
erature to store each patient's body temperature,
 flu_status to store each patient's diagnosis
", "Jane Doe", "Steve Graves")
, 101.4)
E, TRUE)

# 4.  Obtain the body temperature for patient Jane Doe
temperature[2]
```

9. Overall, the RStudio interface can be divided into 4 sections: Editor, Console, Environment and File section.



The editor is where you can enter your R code

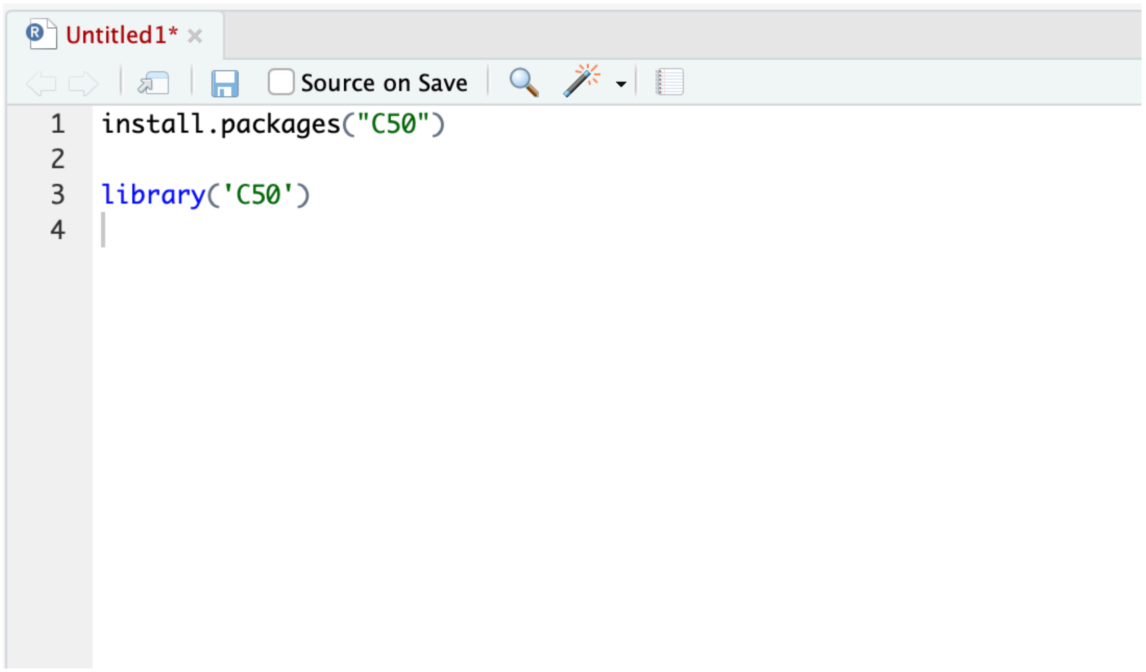
Environment keeps your data and variable

```
R version 3.6.0 (2019-04-26) -- "Planting of a Tree"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (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.

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.

>
```

The console is where your code will be executed

You can track your file and package etc. here

10. Using (Installing and loading) R Packages.
R Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into a session to be used. All R functions and datasets are stored in packages. Only when a package is loaded are its contents available.
Type install.packages("package name") then type library("package name").



```r
install.packages("C50")

library('C50')

```

11. Errors Common to R beginners:

1) object 'xyz' not found:

Check for typos and upper/lower case

Look back in your script to see where xyz was created and run that bit again.

2) There is no package called 'XYZ' when you do library("XYZ").

Check the spelling for typos and lower/upper case

Look for the package in the list of packages installed: use Packages > Load package... from the main menu or do rownames(installed.packages()). If it's not there, you need to install it using install.packages("package name")

3) Could not find function "xyz".

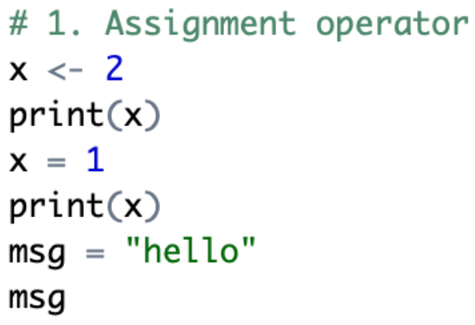
Check the spelling for typos and lower/upper case

Check if install the required packages.

Basic operators and data structures in R

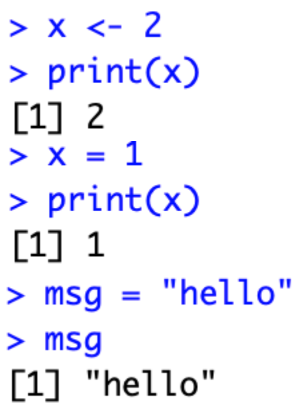
Complete the following tasks, save and name your R script as Lab1.R, and submit on Canvas.

Please include a task's description using comment lines that start with # before you start the R code that performs the task. For example,



```
# 1. Assignment operator
x <- 2
print(x)
x = 1
print(x)
msg = "hello"
msg
```

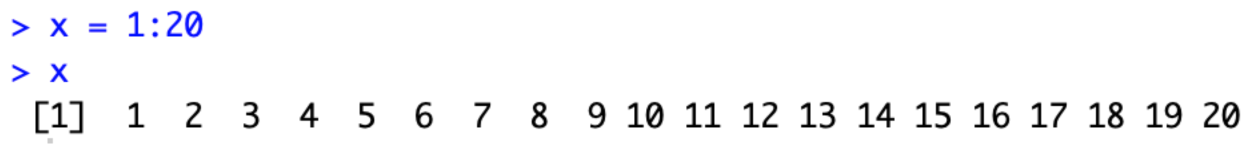
1. Assignment operator: <- or =



```
> x <- 2
> print(x)
[1] 2
> x = 1
> print(x)
[1] 1
> msg = "hello"
> msg
[1] "hello"
```

The [1] indicates 2 is the first element printed.

2. Use the colon (:) operator to create integer sequences.



```
> x = 1:20
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Vector: The fundamental R data structure is the vector, which stores an ordered set of values called elements. A vector can contain any number of elements, but all of the elements must be of the same type of values. For instance, a vector cannot contain both numbers and text. vectors can be created by using the c() combine function.

3. Create a character vector named subject_name to store the three patient names, a double vector named temperature to store each patient's body temperature, and a logical vector named flu_status to store each patient's diagnosis

```
> subject_name <- c("John Doe", "Jane Doe", "Steve Graves")
> temperature <- c(98.1, 98.6, 101.4)
> flu_status <- c(FALSE, FALSE, TRUE)
```

4. Obtain the body temperature for patient Jane Doe

```
> temperature[2]
[1] 98.6
```

5. A range of values can be obtained using the (:) colon operator.
   obtain the body temperature of Jane Doe and Steve Graves

```
> temperature[2:3]
[1]  98.6 101.4
```

6. Exclude Jane Doe's temperature data.

```
> temperature[-2]
[1]  98.1 101.4
```

Factor: A factor is a special case of vector that is solely used to represent categorical variables.

7. Create a factor from a character vector

```
> gender <- factor(c("MALE", "FEMALE", "MALE"))
> gender
[1] MALE   FEMALE MALE
Levels: FEMALE MALE
```

8. When we create factors, we can add additional levels that may not appear in the data.
   Create a factor for the blood type

```
> blood <- factor(c("O", "AB", "A"), levels = c("A", "B", "AB", "O"))
> blood[1:2]
[1] O  AB
Levels: A B AB O
```

List: A list is a data structure, much like a vector. However, where a vector requires all its elements to be the same type, a list allows different types of elements to be collected.

9. Create a list named components for all of the first patient's data.

```
> # 9.  Create a list with named components for all of the first patient's data.
> subject1 <- list(fullname = subject_name[1],
+                  temperature = temperature[1],
+                  flu_status = flu_status[1],
+                  gender = gender[1],
+                  blood = blood[1])
> subject1
$fullname
[1] "John Doe"

$temperature
[1] 98.1

$flu_status
[1] FALSE

$gender
[1] MALE
Levels: 1 MALE

$blood
[1] O
Levels: A B AB O
```

10. Access the temperature value in the created list.

```
> subject1[2]
$temperature
[1] 98.1

> subject1$temperature
[1] 98.1
```

Data frames: A structure analogous to a spreadsheet or database, since it has both rows and columns of data.

11. Create a data frame for our patient dataset, by using the patient data vectors we created.

```
> pt_data <- data.frame(subject_name, temperature, flu_status, gender, blood, stringsAsFactors = FALSE)
> pt_data
  subject_name temperature flu_status gender blood
1     John Doe        98.1      FALSE   MALE     O
2     Jane Doe        98.6      FALSE FEMALE    AB
3 Steve Graves       101.4       TRUE   MALE     A
```

12. Obtain the subject_name vector from the created data frame.

```
> pt_data$subject_name
[1] "John Doe"      "Jane Doe"      "Steve Graves"
```

13. Extract the first and second columns from the data frame.

```
> pt_data[c("subject_name","temperature")]
  subject_name temperature
1     John Doe        98.1
2     Jane Doe        98.6
3 Steve Graves       101.4
> pt_data[,c(1, 2)]
  subject_name temperature
1     John Doe        98.1
2     Jane Doe        98.6
3 Steve Graves       101.4
```

14. Extract the value in the first row and second column of the patient data frame.

```
> pt_data[1, 2]
[1] 98.1
```

15. Extract the first column from data frame.

```
> pt_data[, 1]
[1] "John Doe"     "Jane Doe"     "Steve Graves"
```

16. Extract the first row from data frame.

```
> pt_data[1, ]
  subject_name temperature flu_status gender blood
1     John Doe        98.1      FALSE   MALE     O
```

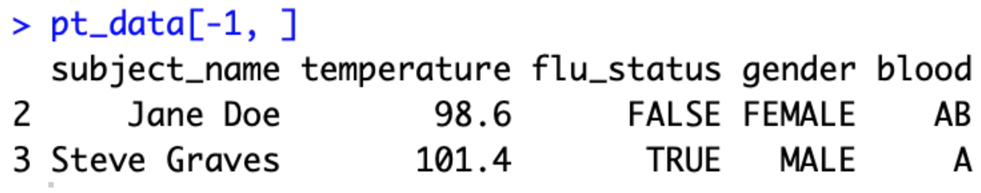
17. Extract everything from data frame.

```
> pt_data[ , ]
  subject_name temperature flu_status gender blood
1     John Doe        98.1      FALSE   MALE     O
2     Jane Doe        98.6      FALSE FEMALE    AB
3 Steve Graves       101.4       TRUE   MALE     A
```

18. Exclude the first column from data frame.

```
> pt_data[, -1]
  temperature flu_status gender blood
1        98.1      FALSE   MALE     O
2        98.6      FALSE FEMALE    AB
3       101.4       TRUE   MALE     A
```

19. Exclude the first row from data frame.



```
> pt_data[-1, ]
  subject_name temperature flu_status gender blood
2     Jane Doe        98.6      FALSE FEMALE    AB
3 Steve Graves       101.4       TRUE   MALE     A
```