Tidyweek: DataAnalysis Bootcamp

Module 1 : Basic R

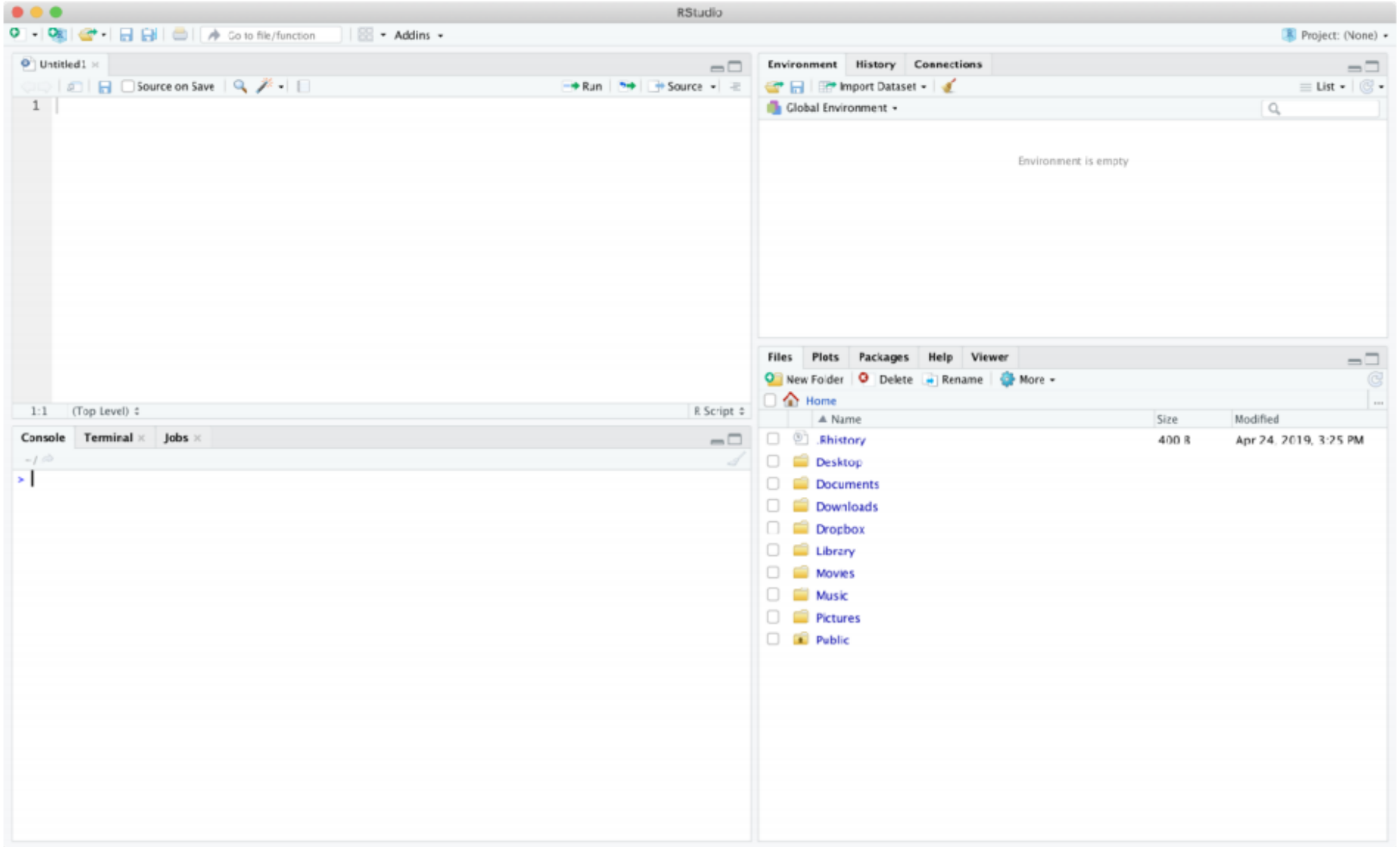
Diwash Shrestha

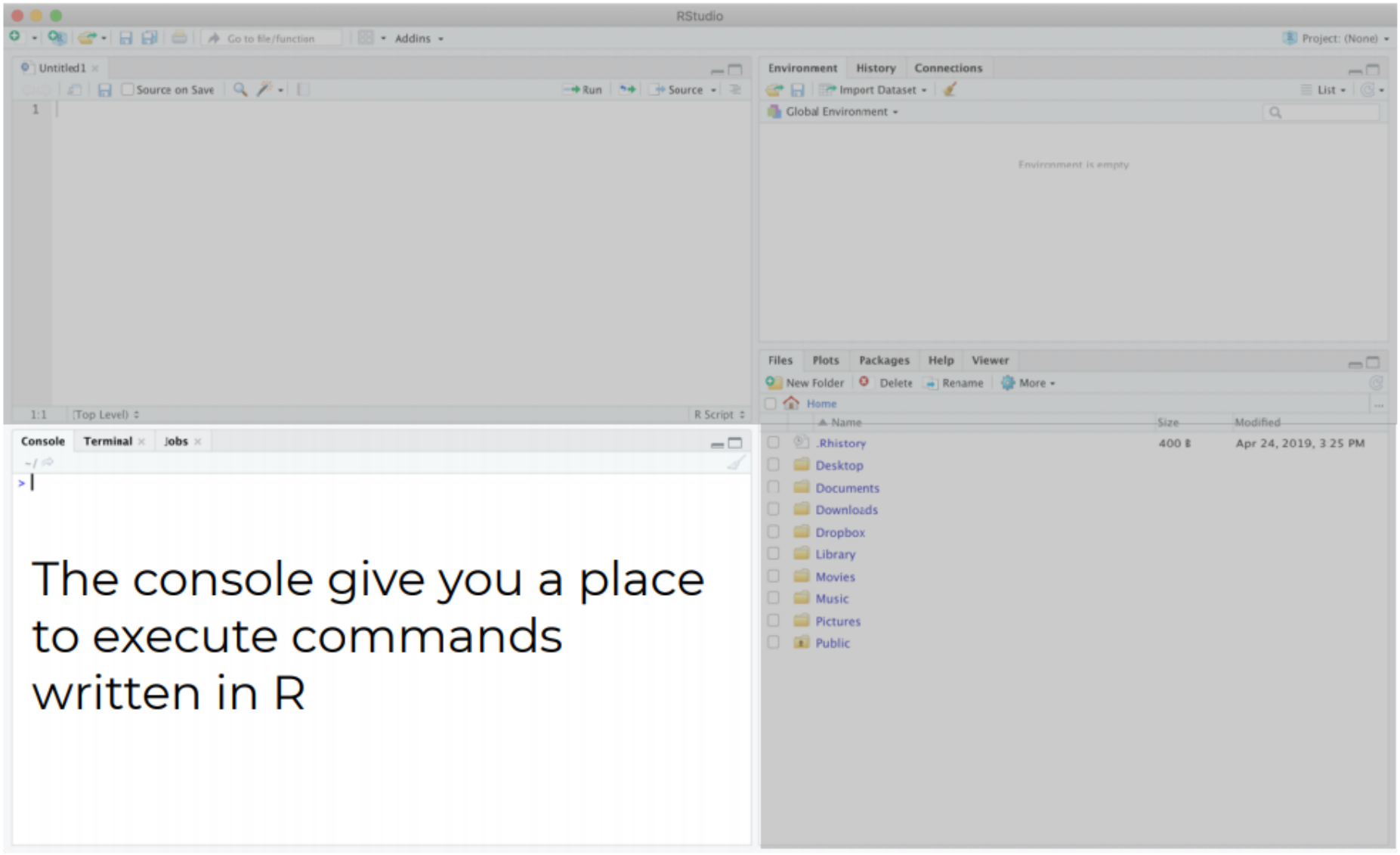
R Nepal

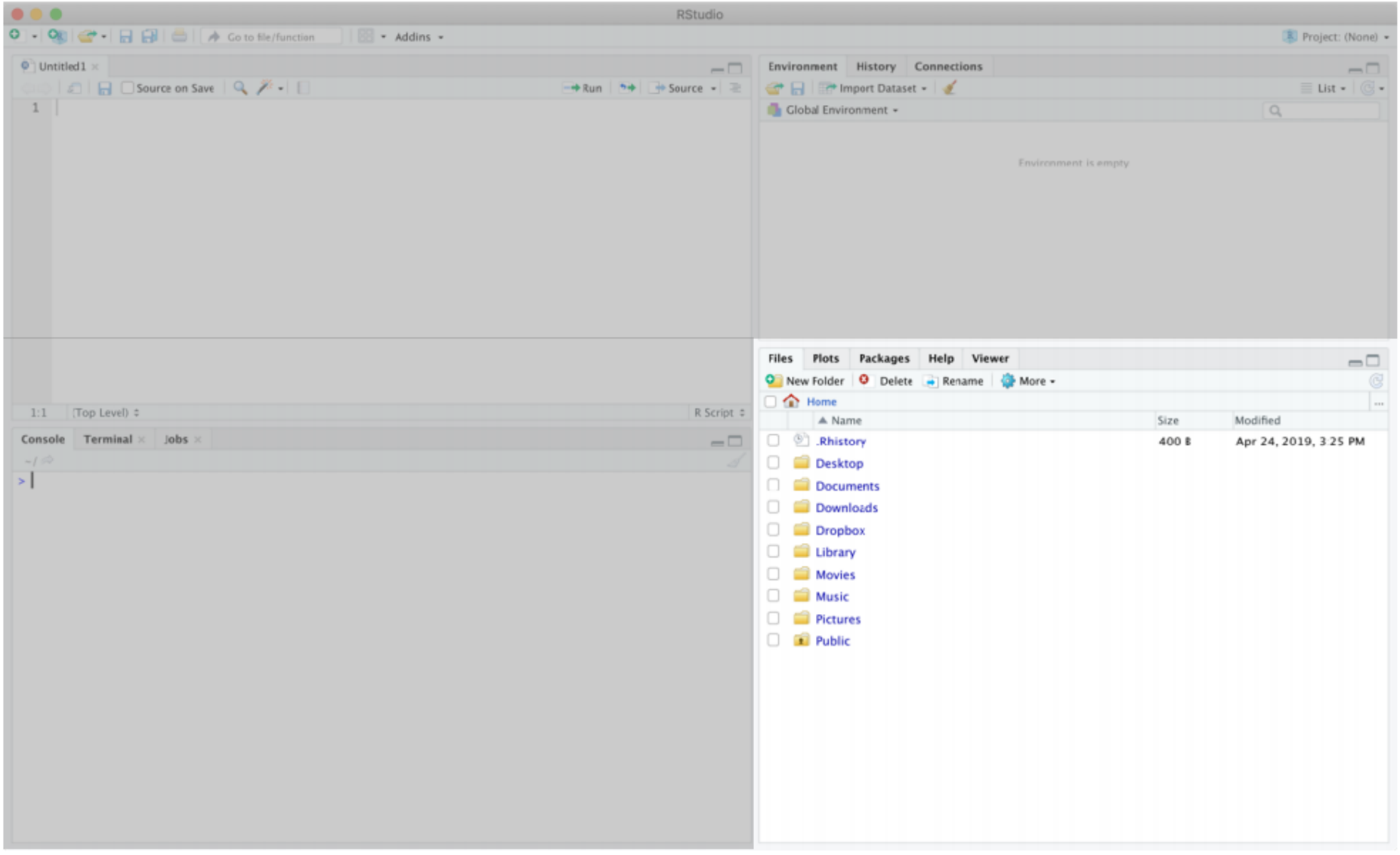
2019/Sep/20

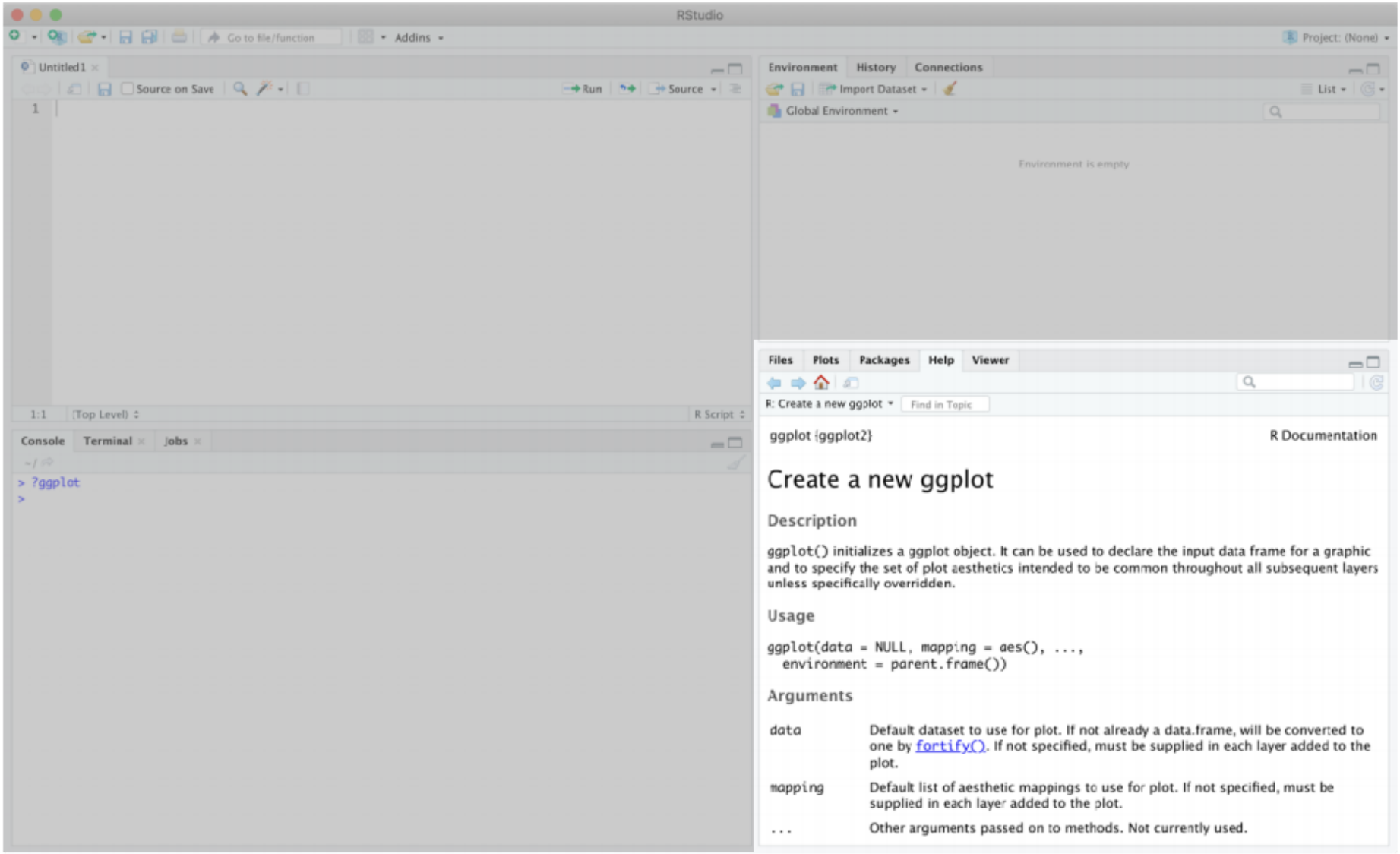
Hello to R and RStudio

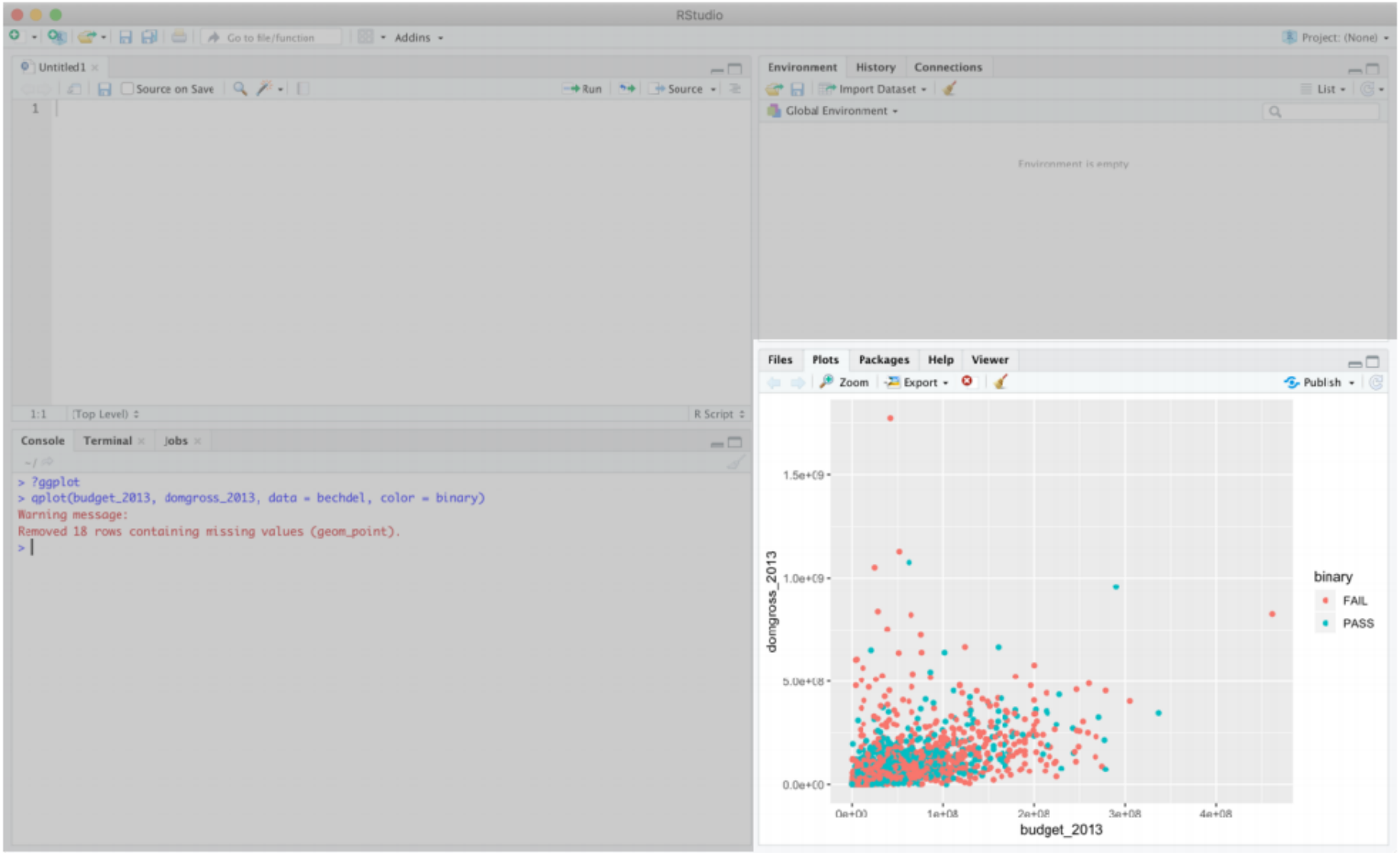
- Install **R** and **Rstudio**
- Install **Tidyverse** Package

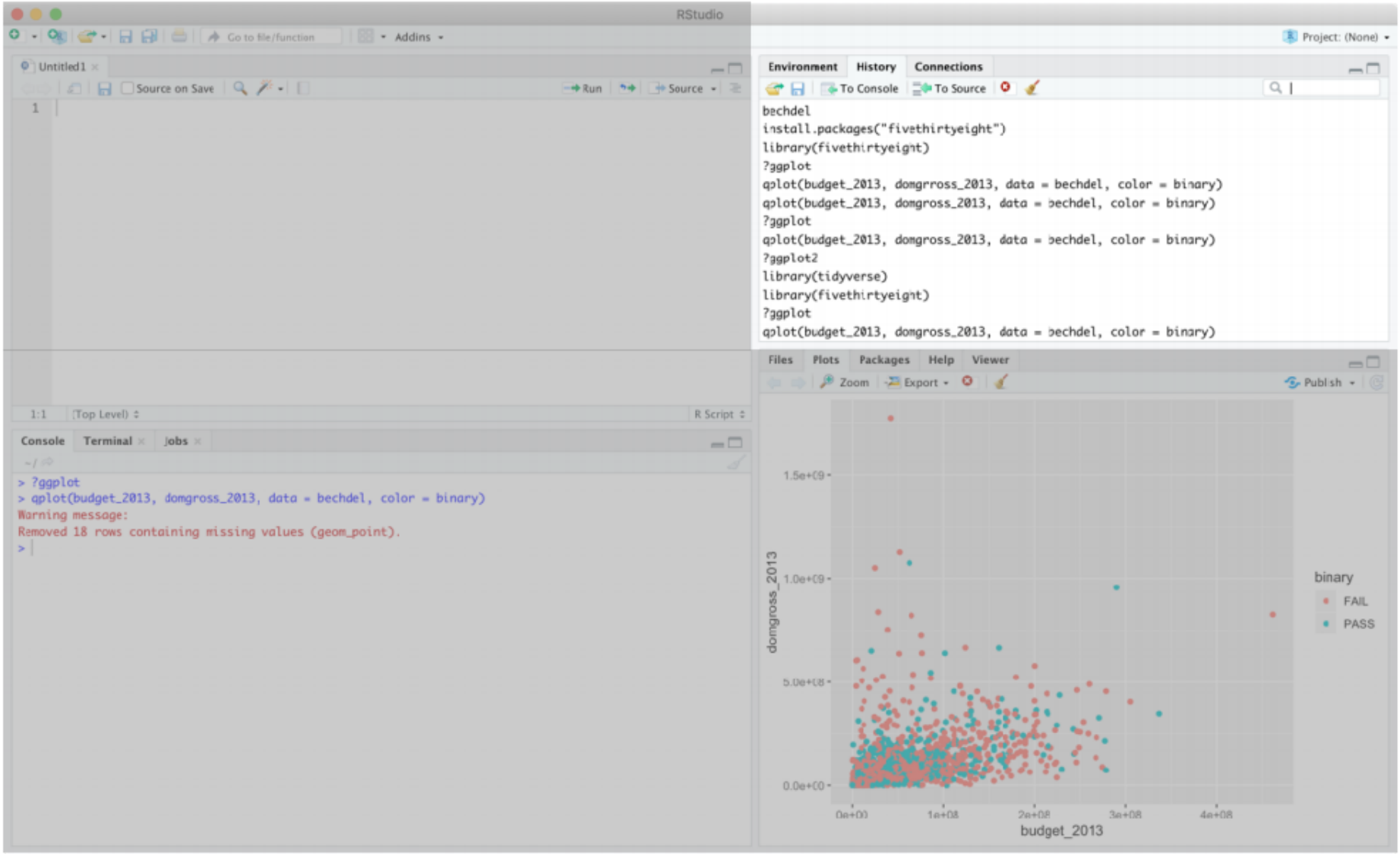

RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
Jobs
~/
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.Rhistory
400 B
Apr 24, 2019, 3:25 PM
Desktop
Documents
Downloads
Dropbox
Library
Movies
Music
Pictures
Public

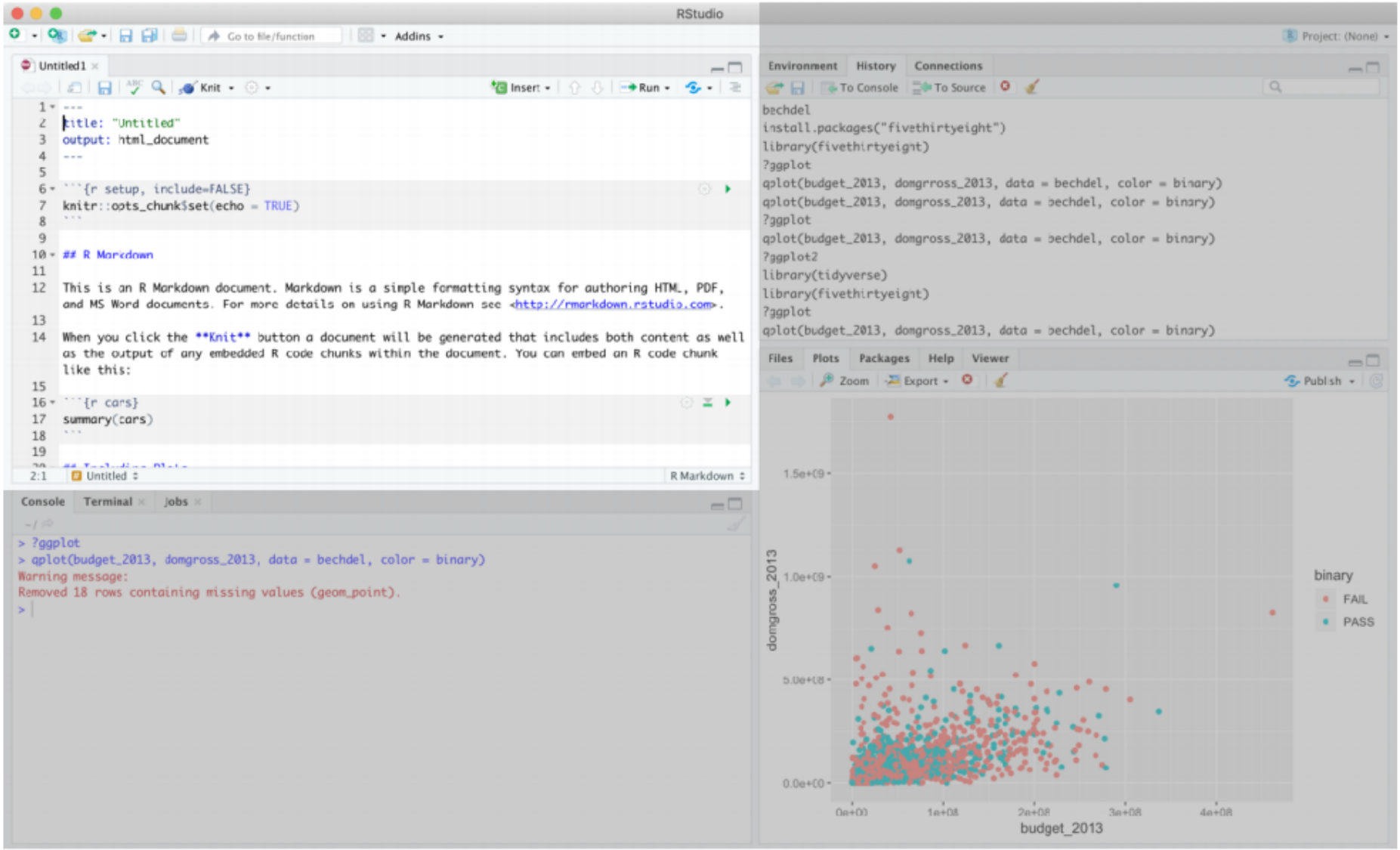

RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.Rhistory
400 B
Apr 24, 2019, 3:25 PM
Desktop
Documents
Downloads
Dropbox
Library
Movies
Music
Pictures
Public
Console
Terminal
Jobs
~/
>
The console give you a place to execute commands written in R


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
1:1
(Top Level)
R Script
Console
Terminal
Jobs
~/
>
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
.Rhistory
400 B
Apr 24, 2019, 3:25 PM
Desktop
Documents
Downloads
Dropbox
Library
Movies
Music
Pictures
Public

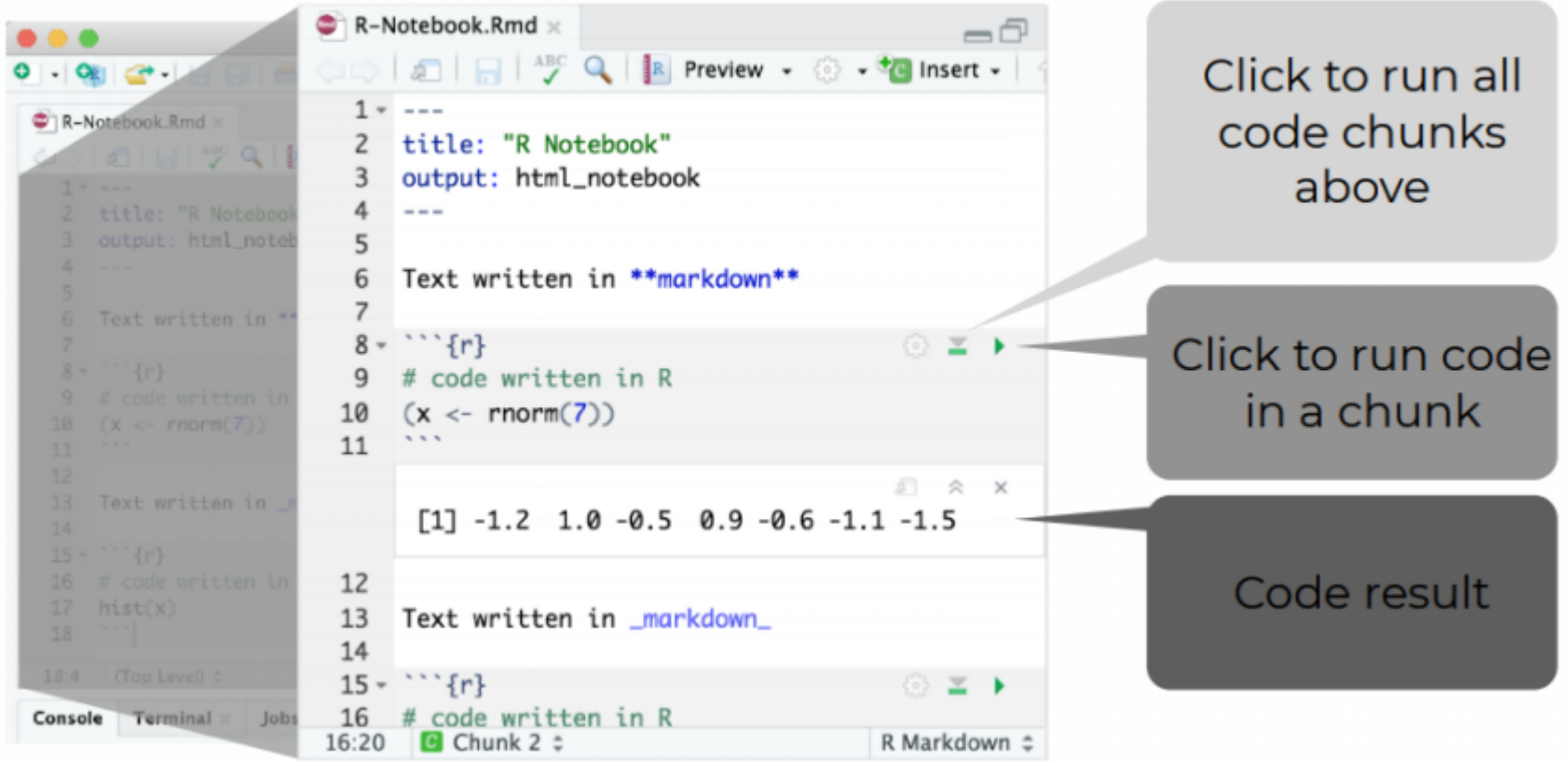

RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
Jobs
~/
> ?ggplot
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
R: Create a new ggplot
Find in Topic
ggplot {ggplot2}
R Documentation
Create a new ggplot
Description
ggplot() initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.
Usage
ggplot(data = NULL, mapping = aes(), ...,
environment = parent.frame())
Arguments
data
Default dataset to use for plot. If not already a data.frame, will be converted to one by fortify(). If not specified, must be supplied in each layer added to the plot.
mapping
Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
...
Other arguments passed on to methods. Not currently used.


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
Jobs
~/
> ?ggplot
> qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
Warning message:
Removed 18 rows containing missing values (geom_point).
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
1.5e+09
1.0e+09
5.0e+08
0.0e+00
domgross_2013
0e+00
1e+08
2e+08
3e+08
4e+08
budget_2013
binary
FAIL
PASS


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1
Environment
History
Connections
To Console
To Source
bechdel
install.packages("fivethirtyeight")
library(fivethirtyeight)
?ggplot
qplot(budget_2013, domgrross_2013, data = bechdel, color = binary)
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
?ggplot
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
?ggplot2
library(tidyverse)
library(fivethirtyeight)
?ggplot
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
1:1
(Top Level)
R Script
Console
Terminal
Jobs
> ?ggplot
> qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
Warning message:
Removed 18 rows containing missing values (geom_point).
>
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
1.5e+09
1.0e+09
5.0e+08
0.0e+00
domgross_2013
0e+00
1e+08
2e+08
3e+08
4e+08
budget_2013
binary
FAIL
PASS


RStudio
Go to file/function
Addins
Project: (None)
Untitled1
Knit
Insert
Run
1 ---
2 title: "Untitled"
3 output: html_document
4 ---
5
6 ```{r setup, include=FALSE}
7 knitr::opts_chunk$set(echo = TRUE)
8 ```
9
10 ## R Markdown
11
12 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
13
14 When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
15
16 ```{r cars}
17 summary(cars)
18 ```
19
2:1 Untitled
R Markdown
Console
Terminal
Jobs
> ?ggplot
> qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
Warning message:
Removed 18 rows containing missing values (geom_point).
>
Environment
History
Connections
To Console
To Source
bechdel
install.packages("fivethirtyeight")
library(fivethirtyeight)
?ggplot
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
?ggplot
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
?ggplot2
library(tidyverse)
library(fivethirtyeight)
?ggplot
qplot(budget_2013, domgross_2013, data = bechdel, color = binary)
Files
Plots
Packages
Help
Viewer
Zoom
Export
Publish
domgross_2013
1.5e+09
1.0e+09
5.0e+08
0.0e+00
0e+00
1e+08
2e+08
3e+08
4e+08
budget_2013
binary
FAIL
PASS

R Notebooks


R-Notebook.Rmd
Preview
Insert
1 ---
2 title: "R Notebook"
3 output: html_notebook
4 ---
5
6 Text written in **markdown**
7
8 ```{r}
9 # code written in R
10 (x <- rnorm(7))
11 ```
[1] -1.2 1.0 -0.5 0.9 -0.6 -1.1 -1.5
12
13 Text written in _markdown_
14
15 ```{r}
16 # code written in R
16:20
Chunk 2
R Markdown
Click to run all code chunks above
Click to run code in a chunk
Code result

Your Turn

05:00

- Copy the File from Pendrive
- Open **01_BasicR.Rmd** and look around.

Basic Syntax

- **Assign value to a variable using "<-" operator.**
- **Assigned a number to the variable but can assign many type of data types.**

```
num <- 123
print(num)
```

```
## [1] 123
```

```
alp <- "abc"
print(alp)
```

```
## [1] "abc"
```

Comment

- Comments are helping text in R program
- Ignored by the interpreter while executing your actual program.
- Written using # in the begining of the statement.
- Write comments as it help to make code readable ,reuseable.

```
# this is a comment
# assigning 1 to a
a <- 1
a
```

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

Your Turn 1

02:00

- Assign some number to **num1** variable and print out.
- Assign alphabet to **alpha1** variable and print out.
- Dont forget to write some comments.

Your Answer 1

- Assign some number to **num1** variable and print out

```
# assiging 123 to num1
num1 <- 123
print(num1)
```

```
## [1] 123
```

- Assign alphabet to **alpha1** variable and print out

```
#assigning apple to alpha1
alpha1 <- "apple"
print(alpha1)
```

```
## [1] "apple"
```

Naming Convention

- variable name must start with a letter(A-Z, a-z)
- can contain letters, numbers and symbols(_ .)
- R is case sensitive language **mydata** and **MyData** is different

```
mydata <- 234 # correct
1_my_data <- 345 # in correct
my_data_1 <- 456 # correct
```

```
mydata <- 234
MyData <- 567
print(mydata)
```

```
## [1] 234
```

```
print(MyData)
```

```
## [1] 567
```

Data Types

- **types or kind of information or data a variable is holding.**
- **class()** function is used to determine the datatype of variable

Data Types	Values
Logical	TRUE / FALSE
Numeric	123 , 2.55
Character	"a" , "abc"

Data Types

```
class(a)
```

```
## [1] "numeric"
```

```
class(alp)
```

```
## [1] "character"
```

```
class(mydata)
```

```
## [1] "numeric"
```

Your Turn 2

02:00

Find the **class** of the variable you defined in Your Turn 1

Your Answer 2

```
# datatype in num1 and alpha1
class(num1)
```

```
## [1] "numeric"
```

```
class(alpha1)
```

```
## [1] "character"
```

Data Structures

Data Structure are the R Objects that is used to store some kind of data.

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

Vectors

- Vector is collection of same type of values
- created using **c()** function

```
# create new vector
vec1 <- c(1,2,3)
vec1
```

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

--

```
# datatype of vector
class(vec1)
```

```
## [1] "numeric"
```

```
vec2 <- c("ram","shyam",1)
vec2
```

```
## [1] "ram"   "shyam" "1"
```

--

```
# Data type of the vector
class(vec2)
```

```
## [1] "character"
```

Your Turn 3

- create a new vector
- find the datatype of vector

01:00

Operators

- Arthmetic Operators
- Relational Operators
- Logical Operators

Arthmetic Operators

Operator	Meaning
+	Add
-	Subtract
*	Multiplication
/	Division
^	Exponent
%%	Modulus

Addition

```
#Addition
a <- c(5,6,7)
b <- c(2,1,3)
print(a + b)
```

```
## [1]  7  7 10
```

--

Multiply

```
# Multiply
a <- c(1,2,3)
b <- c(2,3,4)
print(a * b)
```

```
## [1]  2  6 12
```

--

subtraction

```
#subtraction
a <- c(9,8,7)
b <- c(2,4,9)
print(a - b)
```

```
## [1]  7  4 -2
```

Division

```
# Division
a <- c(22,13,24)
b <- c(2,3,4)
print(a / b)
```

```
## [1] 11.000000  4.333333  6.000000
```

Exponent

```
4 ^ 2
```

```
## [1] 16
```

```
99 ^ 3
```

```
## [1] 970299
```

Modulus

```
7 %% 2
```

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

```
99 %% 8
```

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

Relational Operators

- operators used to compare two elements or vectors.
- compare vector **a** with vector **b**.

Operator	Definition
==	Equal to
>=	Greater than or equal to
<=	Lesser than or equal to
>	Greater than
<	Lesser than
!=	Not equal to

Greater than (>)

```
# greater than

a = c(5,2,3,4)
b = c(10,1,2,9)
print(a > b)
```

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

Less Than (<)

```
#less than <
a <- c(9,10,11,21)
b <- c(11,2,3,55)
print(a<b)
```

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

Equals to (==)

```
#equals to ==

a <- c(2,3,4)
b <- c(2,3,5)
print(a == b)
```

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

NotEqual (!=)

```
# notequal to
a <- c(2,3,4,10)
b <- c(1,3,4,9)
print(a!=b)
```

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

Less than equal to (<=)

```
#less than equal to
a <- c(2,3,4,9)
b <- c(2,5,7,8)
print(a<=b)
```

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

Greater than equal to(>=)

```
# greater than equal to

a <- c(2,3,4,10)
b <- c(5,7,4,9)
print(a>=b)
```

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

Logical Operators

AND operator

It gives **TRUE** value only when both condition result are **TRUE**.

```
# AND operator
a <- c(3,9,5)
b <- c(5,6,7)
print(a>5 & b>5)
```

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

OR operator

It gives **TRUE** value when one of the condition result is **TRUE**.

```
# OR operator
a <- c(1,5,3)
b <- c(4,5,6)
print(a<5 | b>5)
```

Logical NOT (!)

- It makes the **TRUE** to **FALSE** and Viceversa.

```
# logical not
a <- c(3,4,5)

print(!(a>3))
```

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

DataFrame

- table with rows and columns.
- each column contains values of one variable
- each row contains set of values from each column

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Lets Create Dataframe

We can create a dataframe using different type of vector in r using **data.frame()**. R by default converts string to factor while reading or creating dataframe , to keep string as string we use stringsAsFactors.

```
data <- data.frame(id = c(1:5),
                   name = c("Richa","Deepak","Manisha","Ryan","Ram"),
                   age = c(23,25,45,30,57),
                   gender=as.factor(c("female","male","female","male","male")),
                   stringsAsFactors = FALSE)
```

id	name	age	gender
1	Richa	23	female
2	Deepak	25	male
3	Manisha	45	female
4	Ryan	30	male

Get the Structure of the Data Frame

```
# Get the structure of the data frame.
str(data)
```

```
## 'data.frame':    5 obs. of  4 variables:
##  $ id    : int  1 2 3 4 5
##  $ name  : chr  "Richa" "Deepak" "Manisha" "Ryan" ...
##  $ age   : num  23 25 45 30 57
##  $ gender: Factor w/ 2 levels "female","male": 1 2 1 2 2
```

Summary of Data in DataFrame

```
# Print the summary
summary(data)
```

Extract Data From DataFrame

head

- **head()** extract first 6 rows of the dataframe

```
head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1

tail

- **tail()** extract last 6 rows of the dataframe

```
tail(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.7	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.9	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.5	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.5	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.6	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.6	1	1	4	2

Extract Data

- dataframe_name[row_id_from : row_id_to, col_id_from : col_id_to]

Extract first two rows and first two columns

```
# extract first two rows and column
data[1:2,1:2]
```

	id	name
1	1	Richa
2	2	Deepak

Extract 2nd row and all columns

```
# extract last row and all column
data[2,]
```

	id	name	age	gender
2	2	Deepak	25	male

Extract 3rd row with 2nd and 4th column

```
# extract 3rd and 5th row with 2nd and 4th column
data[3,c(2,4)]
```

	name	gender
3	Manisha	female

Extract 2nd and 4th row with 2nd,3rd and 4th column

```
# extract 3rd and 5th row with 2nd and 4th column
data[c(2,4),c(3,4)]
```

	age	gender
2	25	male
4	30	male

Your Turn 4

05:00

- Extract **name** and **age** column
- Extract 2nd row
- Extract 3rd and 4th row and name, gender,age column

Your Answer 4

Extract **name** and **age** column

```
data[,c("name","age")]
```

name <chr>	age <dbl>
Richa	23
Deepak	25
Manisha	45
Ryan	30
Ram	57

5 rows

Extract **2nd** row

```
data[2,]
```

	id	name	age	gender
2	2	Deepak	25	male

Extract 3rd and 4th row and name, gender,age column

```
data[3:4,2:4]
```

	name	age	gender
3	Manisha	45	female
4	Ryan	30	male

Add Column

```
#Add the 'address' column
data$address <- c("butwal","kathmandu","brigunj","illam","karnali")
data
```

id <int>	name <chr>	age <dbl>	gender <fctr>	address <chr>
1	Richa	23	female	butwal
2	Deepak	25	male	kathmandu
3	Manisha	45	female	brigunj
4	Ryan	30	male	illam
5	Ram	57	male	karnali

5 rows

Your Turn 5

02:00

- Add **"blood_group"** column to the dataframe

```
# Add 'blood_group' column
data$blood_group <- c()
data
```

Your Answer 5

```
# Add 'blood_group' column
data$blood_group <- c("A+","B+","AB-","O+","A-")
```

```
head(data)
```

id	name	age	gender	address	blood_group
1	Richa	23	female	butwal	A+
2	Deepak	25	male	kathmandu	B+
3	Manisha	45	female	brigunj	AB-
4	Ryan	30	male	illam	O+
5	Ram	57	male	karnali	A-

Add Row

We can add rows to dataframe using **rbind()** function.

```
# Create second data frame
newdata <- data.frame(id = c(6:8),
                      name  = as.character(c("Deepika","Manish","Brion")),
                      age  = c(33,15,40),
                      gender =  as.factor(c("female","male","male")),
                      address = c("patan","dharan","butwal"),
                      blood_group =   c("A+","B+","AB-"),
                      stringsAsFactors = FALSE)
```

```
newdata
```

id	name	age	gender	address	blood_group
6	Deepika	33	female	patan	A+
7	Manish	15	male	dharan	B+
8	Brion	40	male	butwal	AB-

```
final_data <- rbind(data,newdata)
final_data
```

id	name	age	gender	address	blood_group
1	Richa	23	female	butwal	A+
2	Deepak	25	male	kathmandu	B+
3	Manisha	45	female	brigunj	AB-
4	Ryan	30	male	illam	O+
5	Ram	57	male	karnali	A-
6	Deepika	33	female	patan	A+
7	Manish	15	male	dharan	B+
8	Brion	40	male	butwal	AB-

Merge two dataframe

- Connect two dataframe with different columns using a **key**
- **merge()** is used tp merge the two dataframe
- must have a common column in two dataframe

```
prof_data <- data.frame(
  name = as.character(c("Deepika", "Manish", "Brion", "Richa", "Deepak", "Manisha",
"Ryan", "Ram")),
  phone = c(9129345550, 9129365570, 9129346789, 9129343457, 9129342344, 9129343451,
9129344670, 9129345650),

  profession = as.character(c("Doctor", "Nurse", "Driver", "Scientist", "Computer
Technician", "Teacher", "Student", "Lawyer")),
  stringsAsFactors = FALSE
)
```

```
merge(prof_data,final_data, by = 'name')
```

name	phone	profession	id	age	gender	address	blood_group
Brion	9129346789	Driver	8	40	male	butwal	AB-
Deepak	9129342344	Computer Technician	2	25	male	kathmandu	B+
Deepika	9129345550	Doctor	6	33	female	patan	A+
Manish	9129365570	Nurse	7	15	male	dharan	B+
Manisha	9129343451	Teacher	3	45	female	brigunj	AB-
Ram	9129345650	Lawyer	5	57	male	karnali	A-
Richa	9129343457	Scientist	1	23	female	butwal	A+
Ryan	9129344670	Student	4	30	male	illam	O+

Rename column

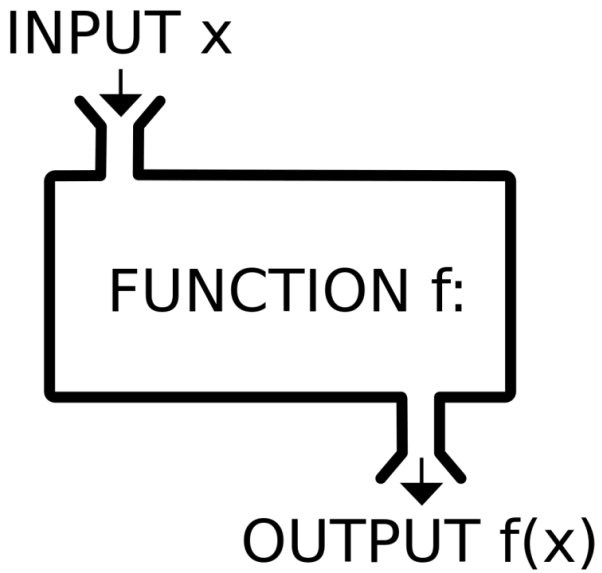
- select column with its name and replace with new name
- **name()** function gives the name of the dataframe column

```
names(final_data)[names(final_data)=="name"] <- "first_name"
final_data
```

id	first_name	age	gender	address	blood_group
1	Richa	23	female	butwal	A+
2	Deepak	25	male	kathmandu	B+
3	Manisha	45	female	brigunj	AB-
4	Ryan	30	male	illam	O+

Function

- set of statements organized together to perform a specific task
- R has a large number of in-built functions



Function Definition

```
function_name <- function(arg_1,arg_2)
  {
  Function Body
  }
```

Function Components

- Function Name: Name of the Function, stored in R environment as an object
- Arguments : Argument are the inputs
- Function Body : Collection of operations performed on the input
- Return Value: Final output from the function

```
# function for adding number
add_me <- function(num1,num2){
  sum <- num1 + num2
  return(sum)
}
```

```
add_me(99,123)
```

```
## [1] 222
```

Built In function

- R has a large number of in-built functions

```
# Create a sequence of numbers from 32 to 44.
print(seq(32,44))
```

```
##  [1] 32 33 34 35 36 37 38 39 40 41 42 43 44
```

```
# Find mean of numbers from 25 to 82.
print(mean(25:82))
```

```
## [1] 53.5
```

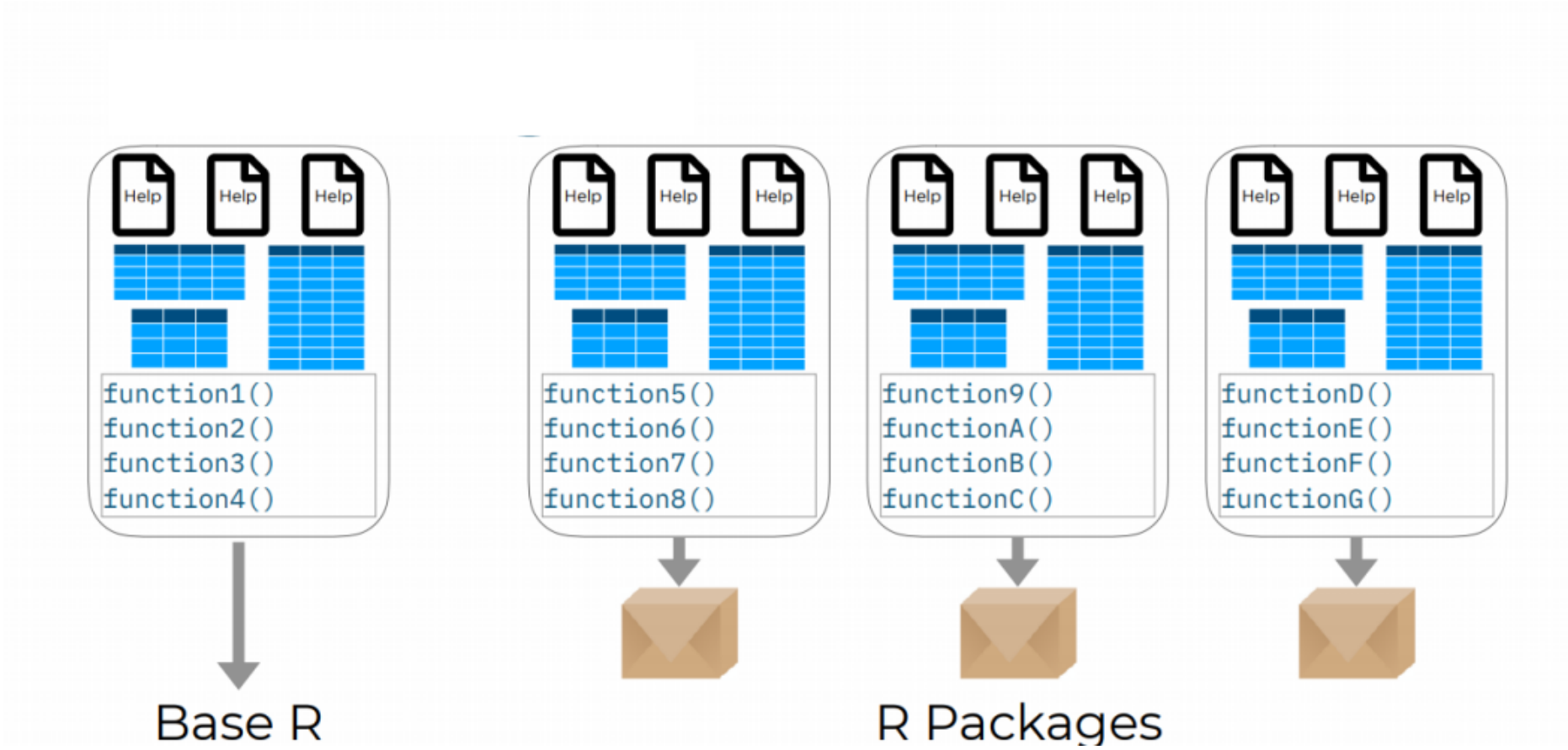
```
# Find sum of numbers frm 41 to 68.
print(sum(41:68))
```

```
## [1] 1526
```

Package

- Packages are a collection of R functions, complied code and sample data
- Stored under a directory called "library" in the R environment
- R installs a set of packages during installation
- Comprehensive R Archive Network" (CRAN) has collection of package

Package


Help
Help
Help
function1()
function2()
function3()
function4()
Base R
Help
Help
Help
function5()
function6()
function7()
function8()
Help
Help
Help
function9()
functionA()
functionB()
functionC()
Help
Help
Help
functionD()
functionE()
functionF()
functionG()
R Packages


android


CRAN
The Comprehensive R Archive Network


RStudio
tidyverse
dplyr
readr
ggplot2
tidyr
tibble
knitr
rmarkdown
feather
purrr
lubridate
broom
Shiny

Install Package

- **install.package()** is used to install package.

```
install.packages("package name")
```

- **library()** is used to load the package.

```
library("package name")
```

Asking for Help

- **?** and **help()** function is used to ask help
- help provides details about r objects like function, data, package.

```
# asking help about the mean() function
?mean
```

```
# asking help about the mtcars datasets
?mtcars
```

```
# asking help about the dplyr package
?dplyr
```

Lets Explore