

P8477 EPI MODELING FOR INFECTIOUS DISEASES

Lab 1: R101

Housekeeping issues

- ▶ Add/drop class
- ▶ Waitlist

Learning Objectives

- ▶ R intro
 - Setting it up
 - Getting started
- ▶ Variables/Data types in R
- ▶ Basic math operations in R
- ▶ Basic functions in R
- ▶ Plotting (data visualization)
- ▶ Input and Output in R

Why programing?

- ▶ For ID modeler, can't write code is like can't drive



<http://www.funnyfidos.com/just-out-for-a-sunday-drive/>

R 101

► What is R?

- <https://www.youtube.com/watch?v=ZCQHm63xc4s>

► Why R?

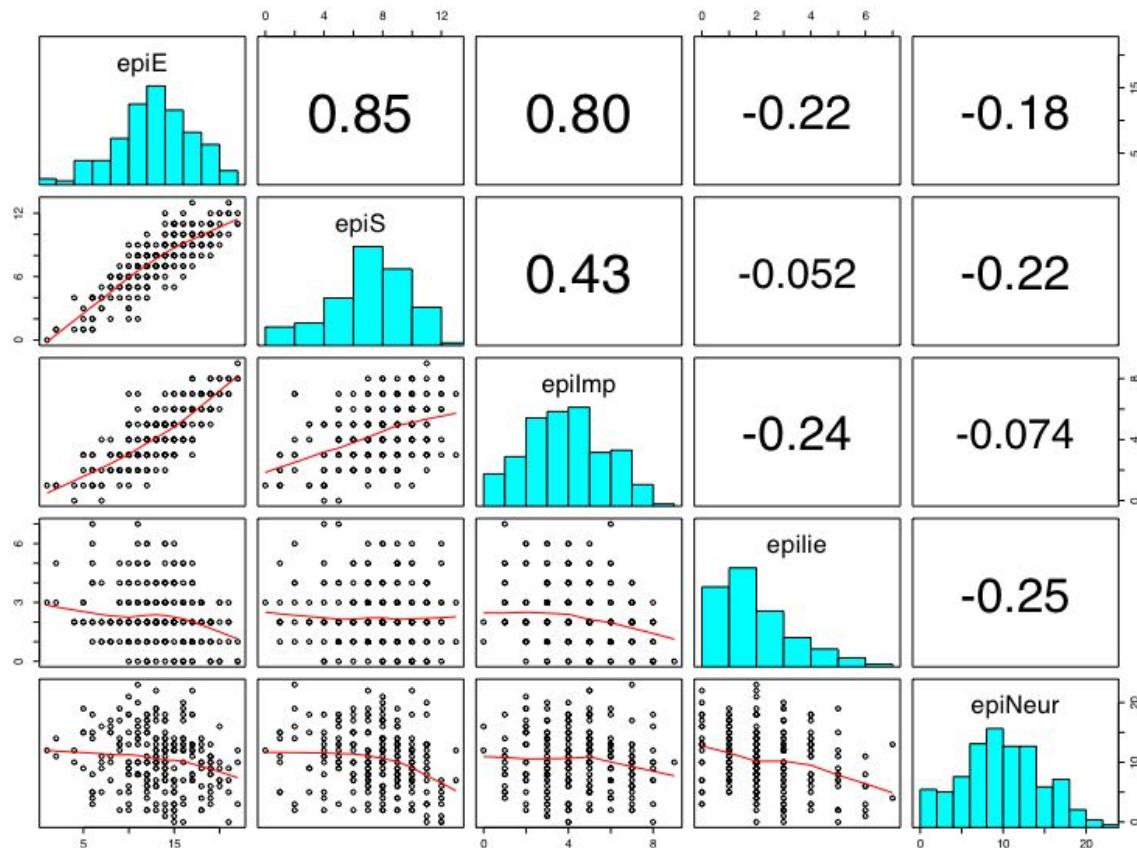
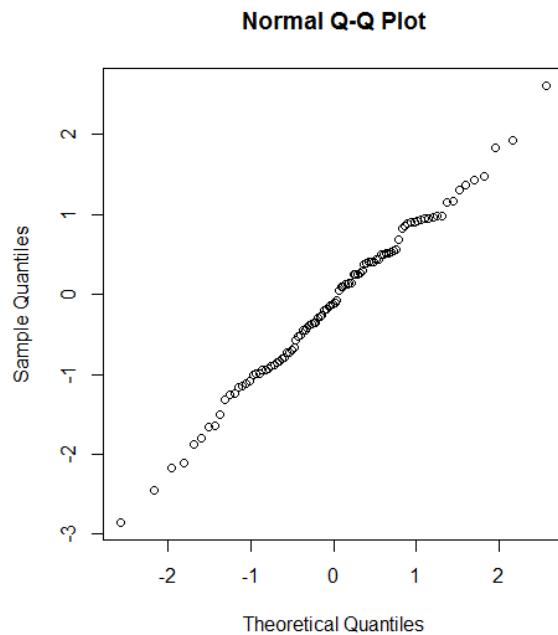
- Statistical programming: powerful analytical tool
 - ❖ Not just for this class
- Open source:
 - ❖ Many developers & functionalities (packages)
 - ❖ And you don't have to pay
- Easy to use (cp. other languages)

What can you do with R?

- ▶ Data analysis
- ▶ Data visualization
- ▶ Simulation (that's we will do with it mostly)
- ▶ Mapping
- ▶ ...

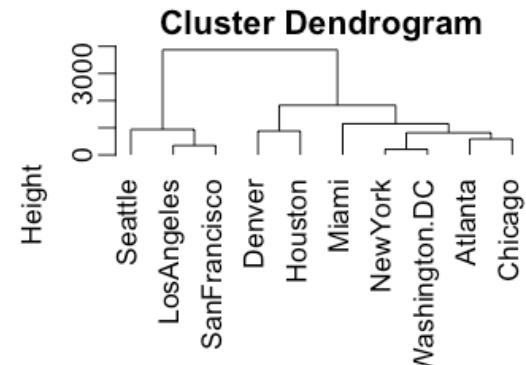
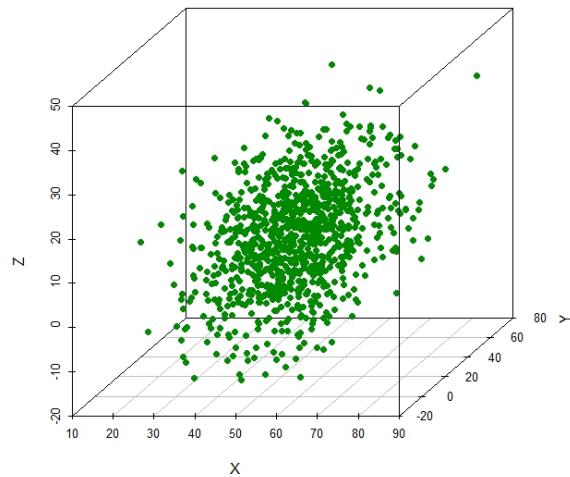
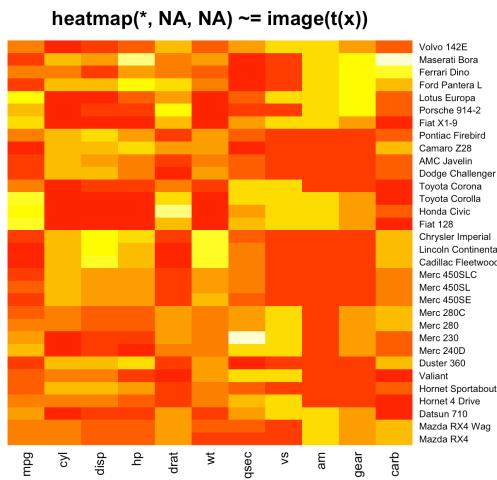
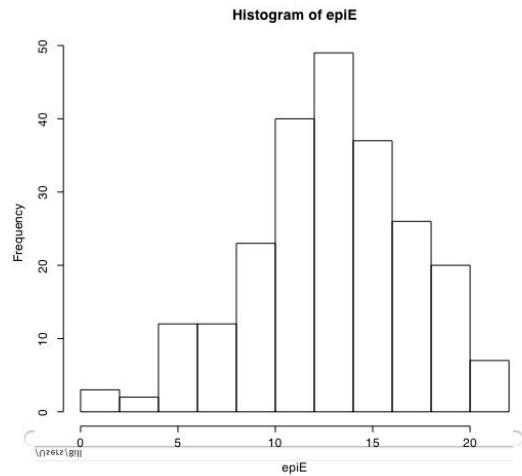
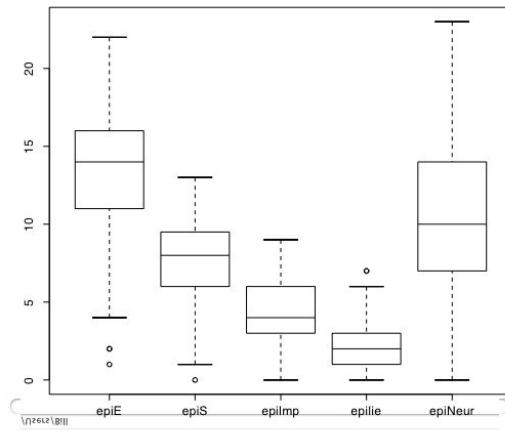
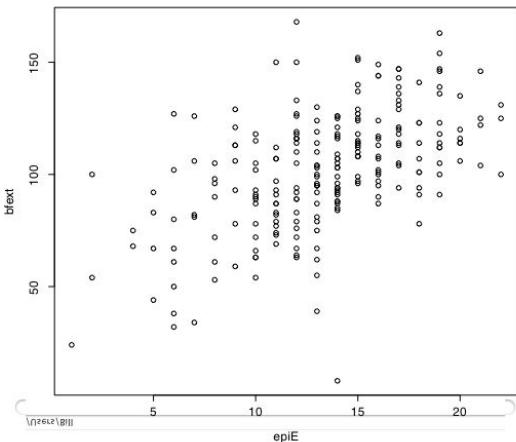
What can you do with R?

- ▶ Data analysis
 - Distributions
 - Hypothesis tests
 - Regression models
- ...



What can you do with R?

► Data visualization



UScitiesD
hclust (*, "ward.D2")

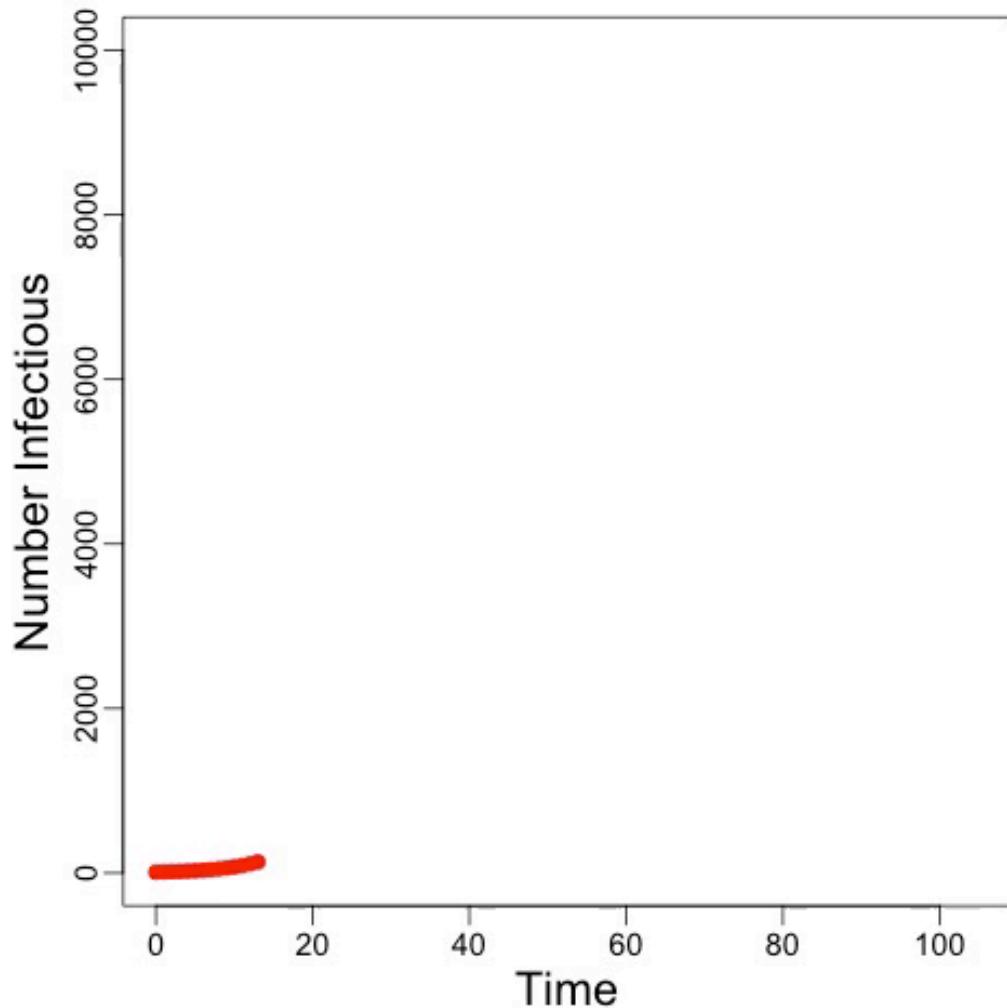
What can you do with R?

- ▶ Simulation (that's we will do with it mostly)

$$\frac{dS}{dt} = -\frac{\beta IS}{N} \quad (1)$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I \quad (2)$$

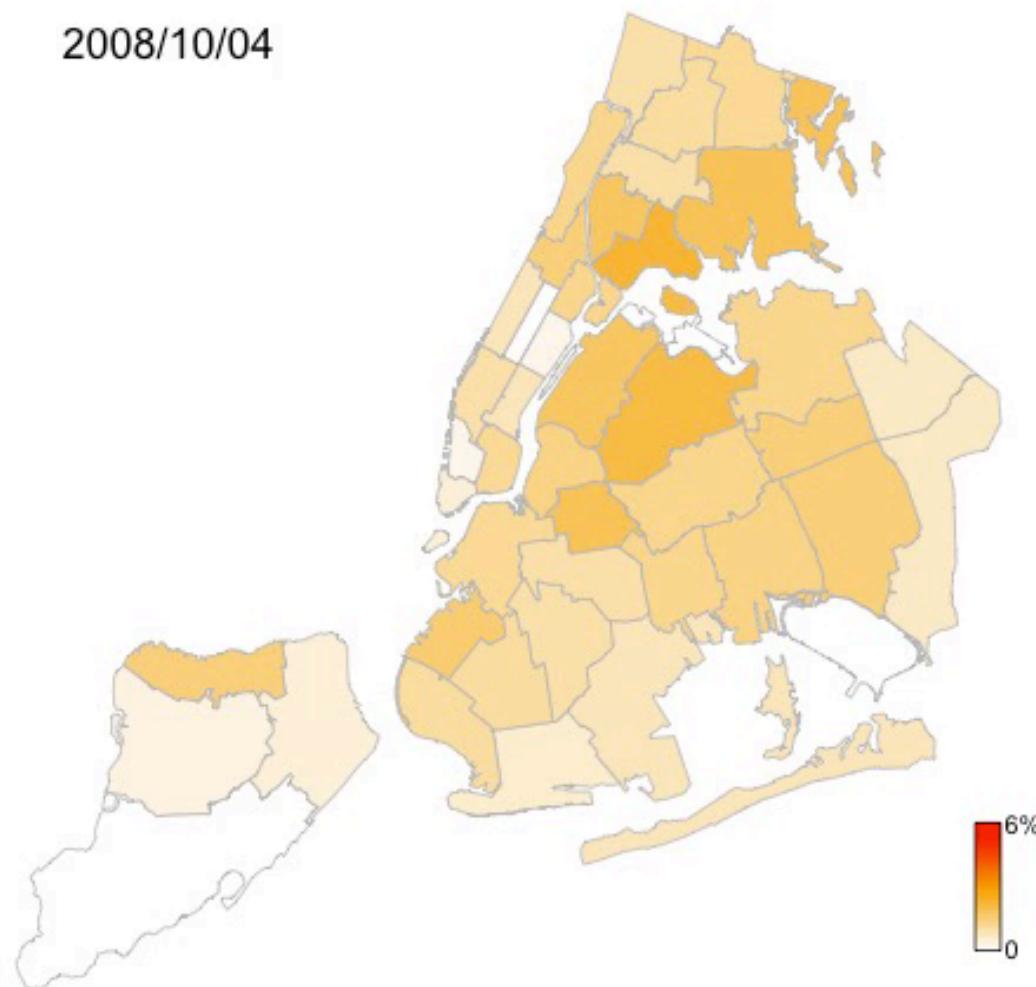
$$\frac{dR}{dt} = \gamma I \quad (3)$$



What can you do with R?

► Mapping

2008/10/04



Let's get started...

▶ Installation

- R: <https://www.r-project.org>



```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (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.
```

```
Natural language support but running in an English locale
```

```
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.
```

```
[R.app GUI 1.68 (7288) x86_64-apple-darwin13.4.0]
```

```
[Workspace restored from /Users/wan/.RData]
[History restored from /Users/wan/.Rapp.history]
```

```
> ← Enter your command here
```

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It con variety of UNIX platforms, Windows and MacOS. To [download R](#), please cho mirror.

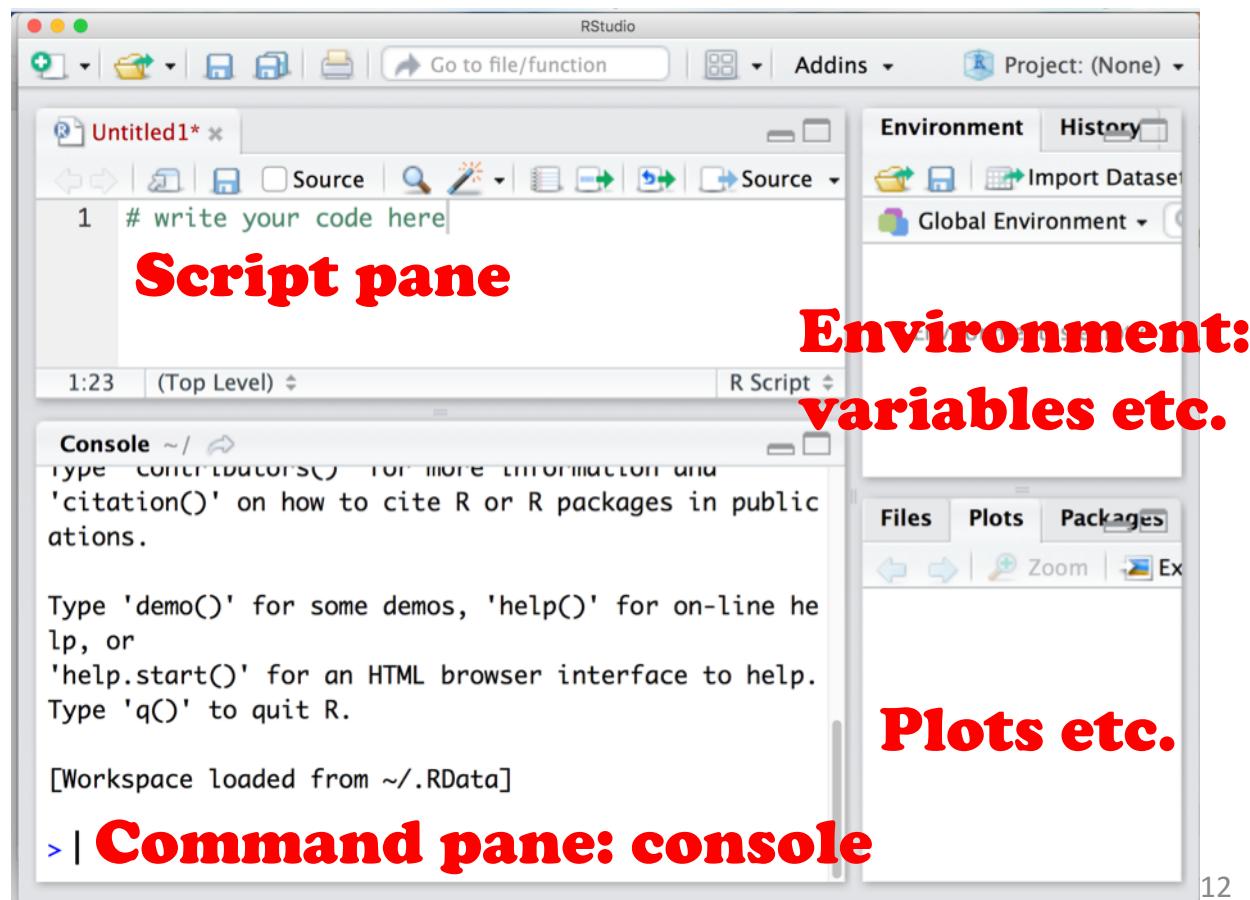
If you have questions about R like how to download and install the software, are, please read our [answers to frequently asked questions](#) before you send a

Let's get started...

► R studio: a user friendly version of R

- <https://www.rstudio.com/products/rstudio/download/>
- You'll need to install R first

► R studio interface

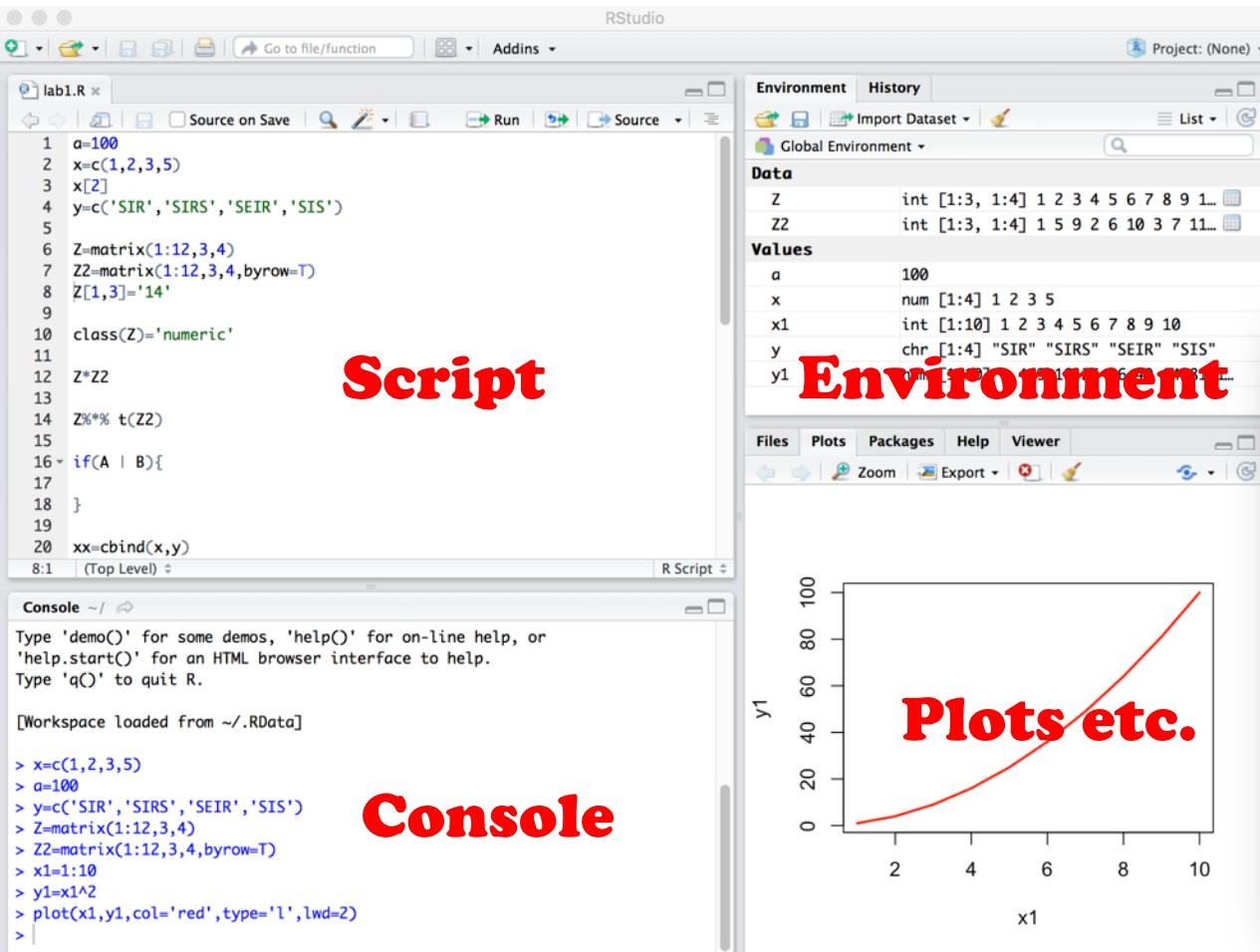


Let's get started...

► R studio: a user friendly version of R

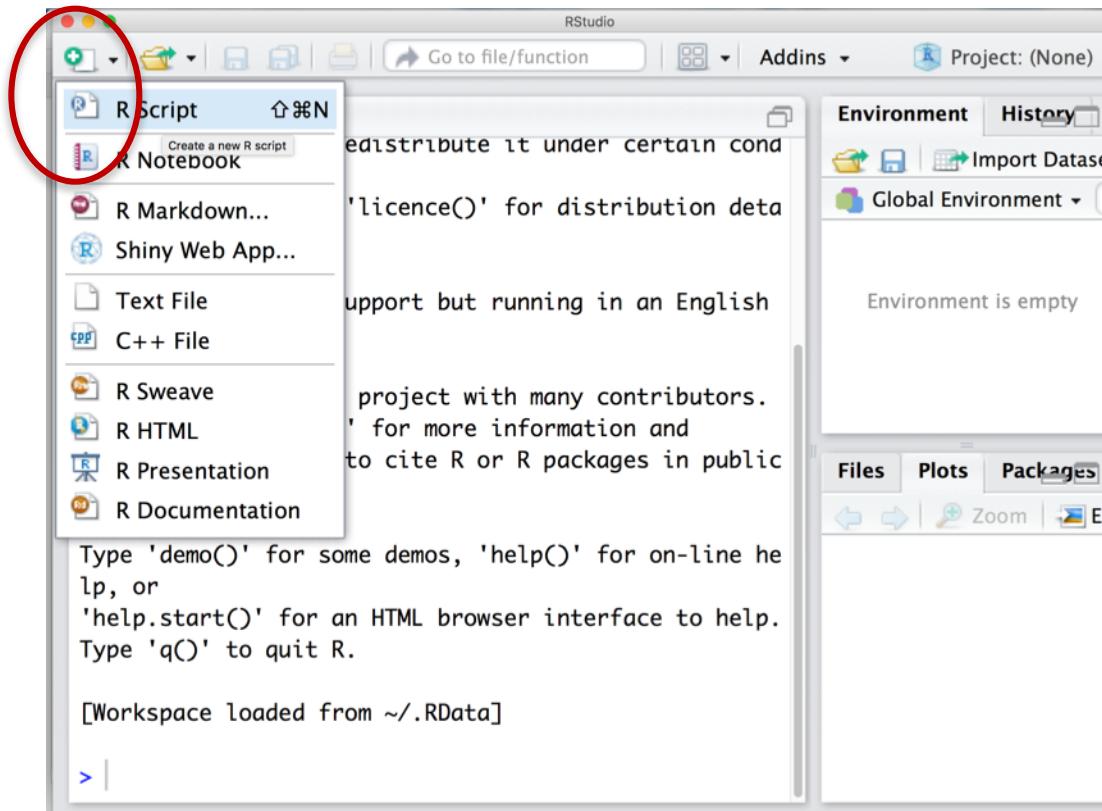
- <https://www.rstudio.com/products/rstudio/download/>
- You'll need to install R first

► R studio interface



Write and Execute Commands

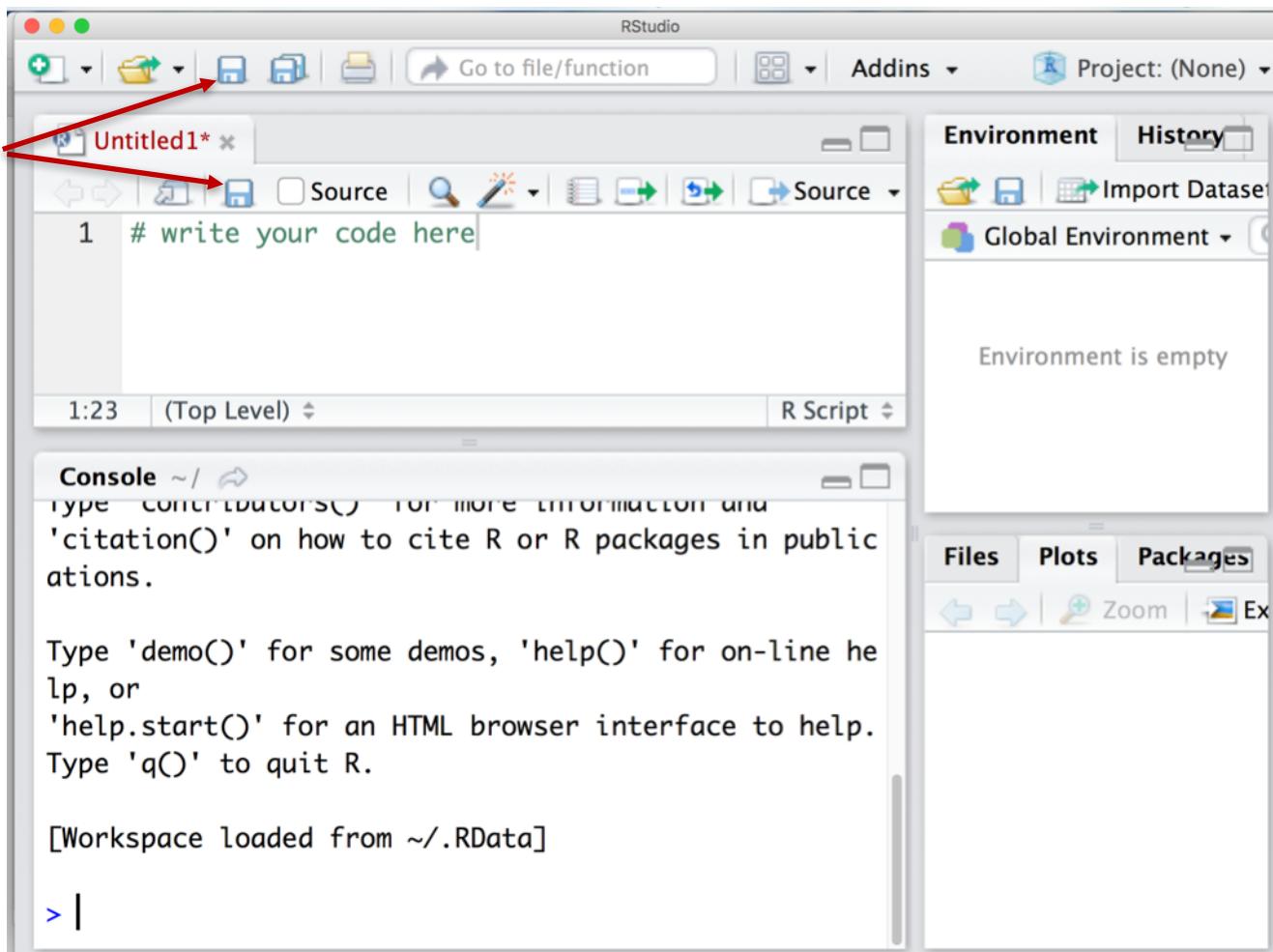
- ▶ Open a script pane to write your code
 - Tip #1: always save your script (so you can re-use it)



Write and Execute Commands

- ▶ Open a script pane to write your code
 - Tip #1: always save your script (so you can re-use it)

Save script



Variables/Data types

▶ Numbers: 1, 2, 3, ...

- Assign numbers: `a=1; b=2;`

Execute commands

(or press

"Ctrl+Return"(Windows)/"Command+Return"(Mac)

- Or a `<- 1`

Comments:
starting with `#`
(won't execute)

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays the script file `basic_programming.R` containing the following code:

```
1 # Script for R tutorial
2
3 # numbers
4 a=1; b=2;
```
- Console:** Shows the output of running the script:

```
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.

[Workspace loaded from ~/.RData]

> a=1; b=2;
>
```
- Environment View:** Shows the variables `a` and `b` with values 1 and 2 respectively.
- Message Bar:** A red arrow points to the "Run" button in the toolbar, with the text "Execute commands (or press 'Ctrl+Return'(Windows)/'Command+Return'(Mac))".
- Comments:** A red arrow points to the first line starting with `#`, with the text "Comments: starting with '#' (won't execute)".
- Bottom Navigation:** Shows tabs for Files, Plots, Packages, and Help.

Variables/Data types

- ▶ Strings:
 - Assign strings: aa="I'm a string"; bb="Columbia"

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays the script file `basic_programing.R` containing the following R code:

```
3 # numbers
4 a=1; b=2;
5
6 # strings:
7 aa="I'm a string"
8 bb="Columbia"
```

The lines `aa` and `bb` are highlighted in blue.
- Console:** Shows the output of the R code:

```
> aa="I'm a string"
> bb="Columbia"
> aa
[1] "I'm a string"
> |
```
- Environment View:** Shows the assigned variables and their values:

Values
a 1
aa "I'm a str...
b 2
bb "Columbia"

Variables/Data types

- ▶ Logical/Boolean:
 - Assign Boolean: TRUE (T), FALSE (F)

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays the script file `basic_programming.R*` containing the following R code:

```
9  
10 # logical/Boolean:  
11 flag1(TRUE); # all in cap  
12 flag2=T; # or just T  
13 flag3(FALSE); # all in cap  
14 flag4=F # or just T  
15  
16
```
- Console:** Shows the output of the executed code:

```
> flag1=TRUE; # all in cap  
> flag2=T; # or just T  
> flag3(FALSE); # all in cap  
> flag4=F # or just T  
> flag1  
[1] TRUE  
> flag3  
[1] FALSE  
>
```
- Environment:** Shows the current environment variables:

a	1
aa	"I'm a str...
b	2
bb	"Columbia"
flag1	TRUE
- Packages:** Shows the current packages loaded.

Variables/Data types

► Vectors: >=1 element

○ Assignment:

- ❖ Numbers: `x=c(1,2,3)`
- ❖ Strings: `y=c('SIR', 'SEIR', 'SIRS')`
- ❖ `c()`: concatenate

○ Access element:

- ❖ use `[]`, e.g. `x[1]`
- ❖ Index in R starts from 1



The screenshot shows the RStudio interface. On the left, a script editor window displays the following R code:

```
# vectors
x=c(1,2,3,4,5,6,7,8,9,10);
y=c('SIR','SEIR','SIRS')
```

On the right, the Environment pane shows variables and their values:

a	1
aa	"I'm a str...
b	2
bb	"Columbia"
flag1	TRUE

The Console pane at the bottom shows the execution of the code and the results of the indexing:

```
> # vectors
> x=c(1,2,3,4,5,6,7,8,9,10);
> y=c('SIR','SEIR','SIRS')
> x[2]
[1] 2
> y[1]
[1] "SIR"
>
```

Variables/Data types

► Vectors: >=1 element

○ Sequence:

❖ `z2=seq(1,100,by=2)`

✓ `seq()`: sequence

✓ `by`: increment

❖ `z1=1:100`

✓ `:`: increase by 1

The screenshot shows the RStudio interface with the following components:

- Code Editor (Top Left):** Displays the script file `basic_programming.R*` containing the following code:

```
20
21 # sequence
22 z1=1:100;
23 z2=seq(1,100,by=2);
```
- Console (Bottom Left):** Shows the output of the executed code:

```
> z1=1:100;
> z2=seq(1,100,by=2);
> z1[10]
[1] 10
> z2[10]
[1] 19
> |
```
- Environment (Right Side):** Shows the global environment with variables defined:

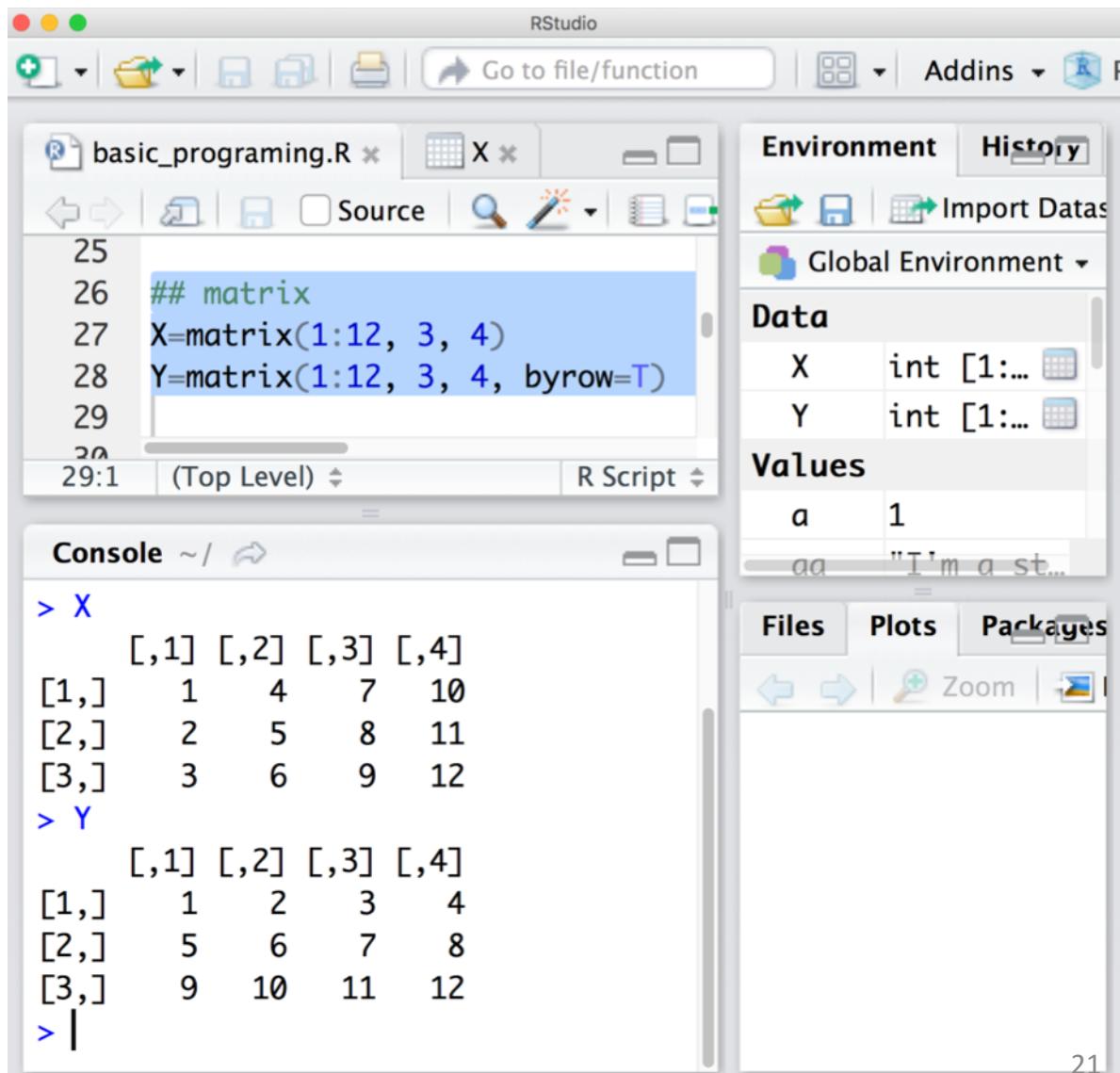
flag4	FALSE
x	num [1:10...]
y	chr [1:3...]
z1	int [1:10...]
z2	num [1:50...]
- Plots (Bottom Right):** Shows a blank plot area.

Two red arrows point from the text "increase by 1" to the `z2[10]` and `z1[10]` lines in the Console window.

Variables/Data types

► Matrix:

- `X=matrix(1:12,3,4)`
matrix(elements to use in the matrix, #Row, #Col)
- in R, elements are filled by column by default
- To tell R to fill by row, use:
`Y=matrix(1:12,3,4,
byrow=T)`
- Access element
 - ❖ e.g.: `X[1,2]`
Row index, Col index



The screenshot shows the RStudio interface with the following details:

- Code Editor:** Shows a script named "basic_programming.R" with the following code:

```
## matrix
X=matrix(1:12, 3, 4)
Y=matrix(1:12, 3, 4, byrow=T)
```
- Console:** Displays the output of the executed code:

```
> X
 [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> Y
 [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
> |
```
- Data View:** Shows the variables defined in the environment:

Variable	Type	Value
X	int [1:12]	[1, 4, 7, 10, 2, 5, 8, 11, 3, 6, 9, 12]
Y	int [1:12]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
- Global Environment:** Shows the variable "aa" with the value "I'm a st...".

Variables/Data types

► Matrix:

- `X=matrix(1:12,3,4)`
- in R, elements are filled by column by default
- To tell R to fill by row, use:
`Y=matrix(1:12,3,4,
byrow=T)`
- Contains the same type of elements

The screenshot shows the RStudio interface with the following details:

- Environment Panel:** Shows variables `X` and `Y` as integer matrices [1:12]. `X` is highlighted with a red box.
- Data View:** Displays the contents of matrix `X` in a grid format:

	v1	v2	v3	v4
1	1	4	7	10
2	2	5	8	11
3	3	6	9	12

- Console View:** Shows the R code used to create matrix `X` and its printed output.

```
> X
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> Y
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Variables/Data types

▶ Array: could be >2 dimensions

- `array()`
- `arr=array(0,c(2,4,5))`

`array(elements to use in the array, c(dim1, dim2,...))`

- To access elements:
 - ❖ `arr[1,1,1]`
 - ❖ `arr[1,,]`

The screenshot shows the RStudio interface with the following components:

- Top Bar:** Shows the RStudio logo, file menu, and "Project: (None)".
- Left Panel:** Shows a file named "basic_programming.R" and other tabs like "da" and "X".
- Code Editor:** Displays R code:

```
29
30
31 # array: when your dataset has >2 dimension
32 arr=array(0,c(2,3,4))
33
34
35
36
```
- Console:** Displays the output of the R code:

```
> View(da)
> # array: when your dataset has >2 dimensions
> arr=array(0,c(2,3,4))
> arr[1,,]
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0
[3,]    0    0    0    0
>
```
- Environment:** Shows variables defined in the global environment:

aa	"I'm a string"
arr	num [1:2, 1:4]
b	2
bb	"Columbia"
flag1	TRUE
flag2	TRUE
- Plots:** Shows a small plot icon.
- Packages:** Shows a small package icon.

Variables/Data types

- ▶ List: an object consisting of an ordered collection of objects known as its components.
 - `list()`
 - Elements can be of different data types vs. same data type in a vector

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays the script `basic_programming.R`. The code defines a list `LL` with three components: `model`, `Susceptible`, and `Infected`. The `Susceptible` component is a sequence from $1e5$ to $.9e5$ with a step of -500 . The `Infected` component is a sequence from $1,1000$ to $1,000$ with a step of 500 .
- Console:** Shows the output of running `LL`. It prints the value of `$model` as "SIR". Then it prints the first seven elements of the `$Susceptible` vector: [1] 100000 99500 99000 98500 98000 97500 [7] 97000 96500 96000 95500 95000 94500.
- Environment:** Shows the variables defined in the global environment: `LL` (List of 3), `x` (num [1:10]), and `v` (chr [1:3] "S").

Variables/Data types

- ▶ Data Frame: an organized list
 - `data.frame(...)`
 - allow different data types for different columns
 - ❖ vs matrix (only 1 data type for all elements)
 - e.g.: put different features together (name, weight, height, etc.); rows are records

Index	Weight	Group
1	92.9	Control
2	86.8	Control
3	81.3	Control
4	73.1	Experiment
5	94.3	Control
6	94.9	Experiment
7	67.4	Control
8	81.7	Control
9	93.7	Experiment
10	60.0	Control

Variables/Data types

- ▶ Special variables:
 - NA: missing value
 - Inf: infinity (e.g. divide by 0)
 - pi: 3.1415...

(Math) Operations

- summation: $1 + 3$
- subtraction: $3 - 2$
- multiplication: $3 * 2$
- division: $3 / 2$
- square root: `sqrt()`
- exponential: x^y
- logarithm:
 - ✓ natural log: `log()`;
 - ✓ other base: `log(x, base=Base)`
- matrix multiplication: `%*%`
- mod (residual): `%%`
- logic: `&` (and); `|` (or)
- Use `()` to arrange the order of operations: $(100+20)*5$ v. $100+20*5$

Don't forget `*`: e.g., x times y
should be ' $x*y$ ', not ' xy '

Built-in functions

- c(); seq(); matrix(); array()
- head(); tail()
- rep();
- sum(); mean(); sd();
median(); var();
- min(); max();
- sort(); order();
- rbind(); cbind()
- paste(); paste0()
- summary(); table()
- Learn more:
 - ❖ help()
 - ❖ ? Function name

The screenshot shows the RStudio interface. The top panel is the 'Script' editor, displaying an R script named 'basic_programing.R'. The code in the script includes comments and several calls to the 'head' and 'tail' functions. The bottom panel is the 'Console' window, which shows the output of running the 'head(da)' command on a dataset 'da'. The output displays the first six rows of the dataset.

```
## Inputs and outputs
da=read.csv('~/Documents/Teaching/IDmodelin'
head(da) # by default: the first 6 rows
head(da,2); # the first 2 rows
tail(da); # by default: the last 6 rows
tail(da,1); # the last row
```

time	S	I
1	0	99990.00 10.00000
2	1	99984.47 12.21326
3	2	99977.71 14.91591
4	3	99969.45 18.21594
5	4	99959.38 22.24507
6	5	99947.07 27.16385

Customized functions

► Customized functions

- EX: Name of the function [choose as you like, but should start with a character (not number)]

Function input(s)

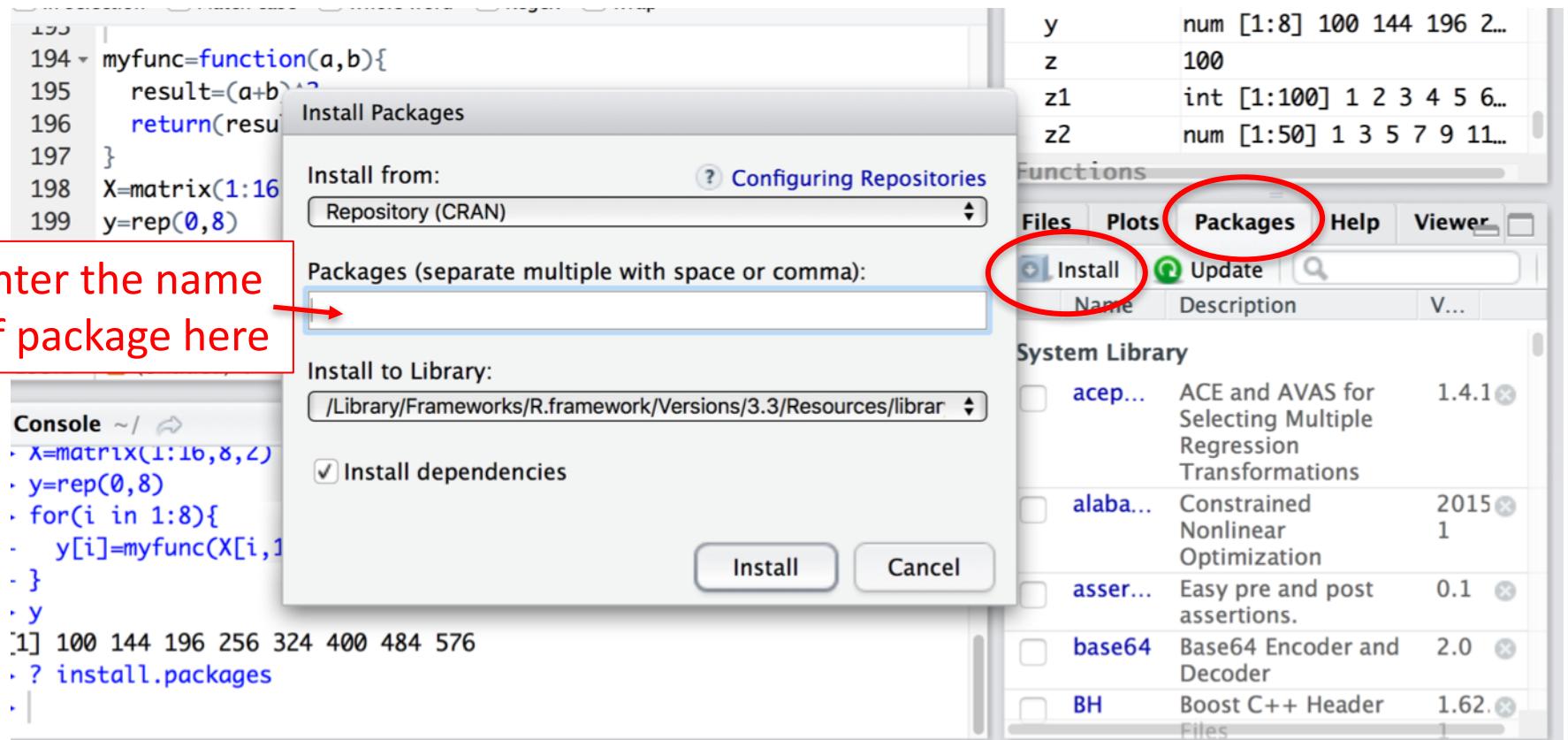
```
194 myfunc=function(a,b){  
195   result=(a+b)^2  
196   return(result)  
197 }  
198 X=matrix(1:16,8,2)  
199 y=rep(0,8)  
200 for(i in 1:8){  
201   y[i]=myfunc(X[i,1],X[i,2])  
202 }  
193:1 # (Untitled) ▾
```

Console ~ / ↗

```
+ }  
> X=matrix(1:16,8,2)  
> y=rep(0,8)  
> for(i in 1:8){  
+   y[i]=myfunc(X[i,1],X[i,2])  
+ }  
> y  
[1] 100 144 196 256 324 400 484 576  
> |
```

Install a “library” (“package”)

- ▶ What is a library/package?
 - All R functions and datasets are stored in packages
- ▶ Install a library
 - command: `install.packages("NAME OF PACKAGE")`

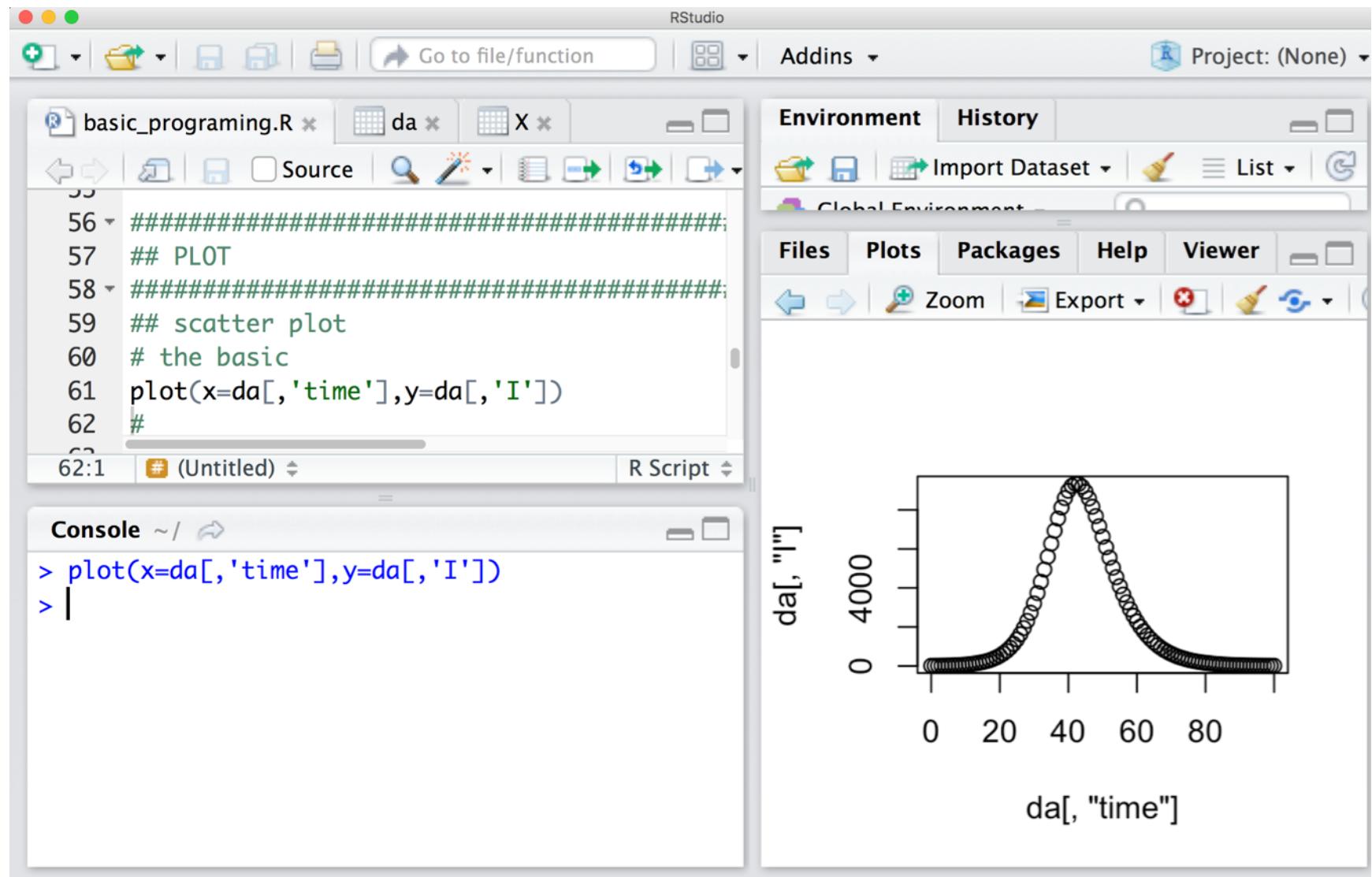


Install a “library” (“package”)

- ▶ What is a library/package?
 - All R functions and datasets are stored in packages
- ▶ Install a library
 - command: `install.packages` (“NAME OF PACKAGE”)
- ▶ Load a library: **need to load it before use**
 - command: `library` (“NAME OF LIBRARY”)

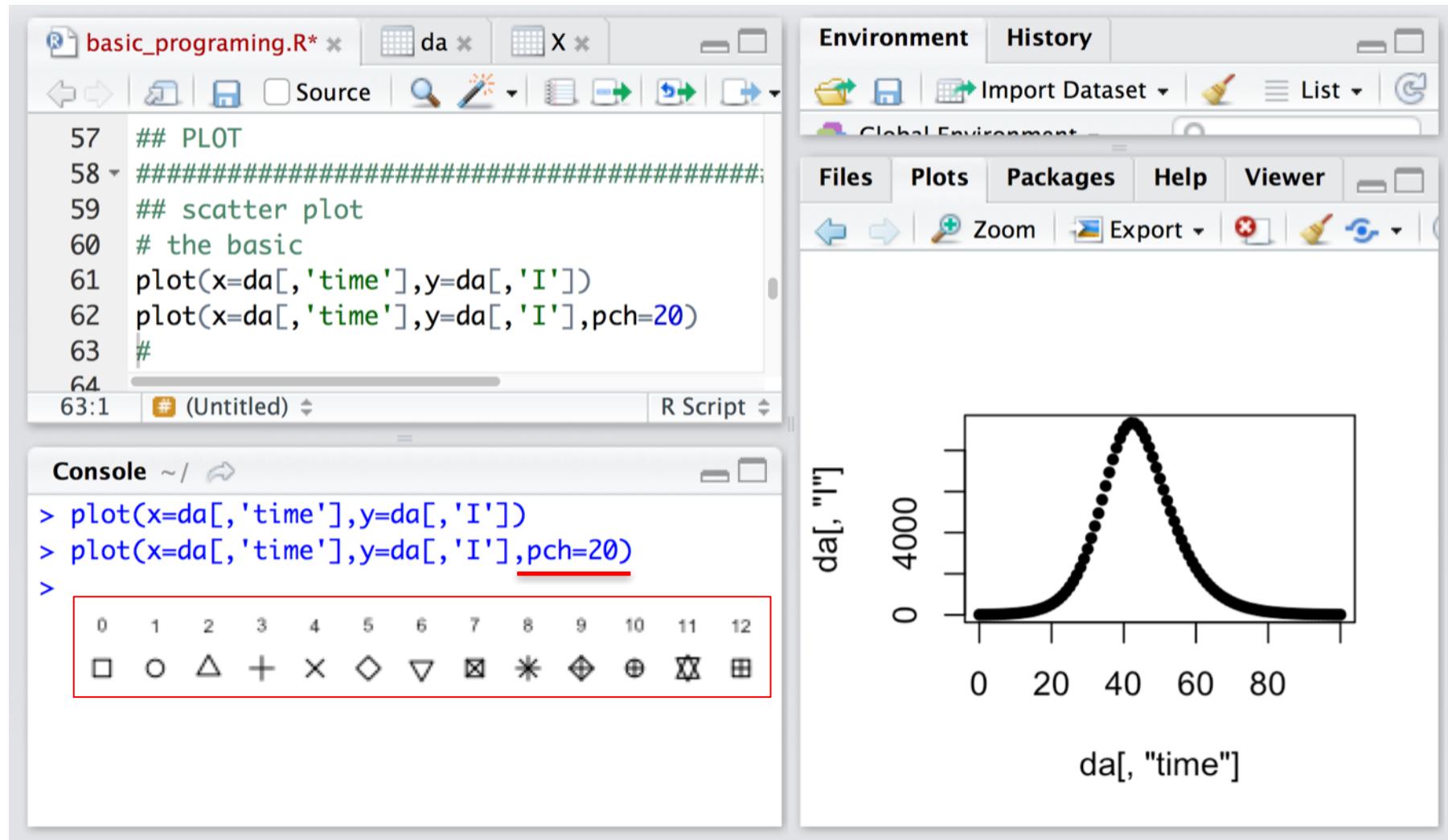
Plots

▶ Scatter plot: `plot(x,y,...)`



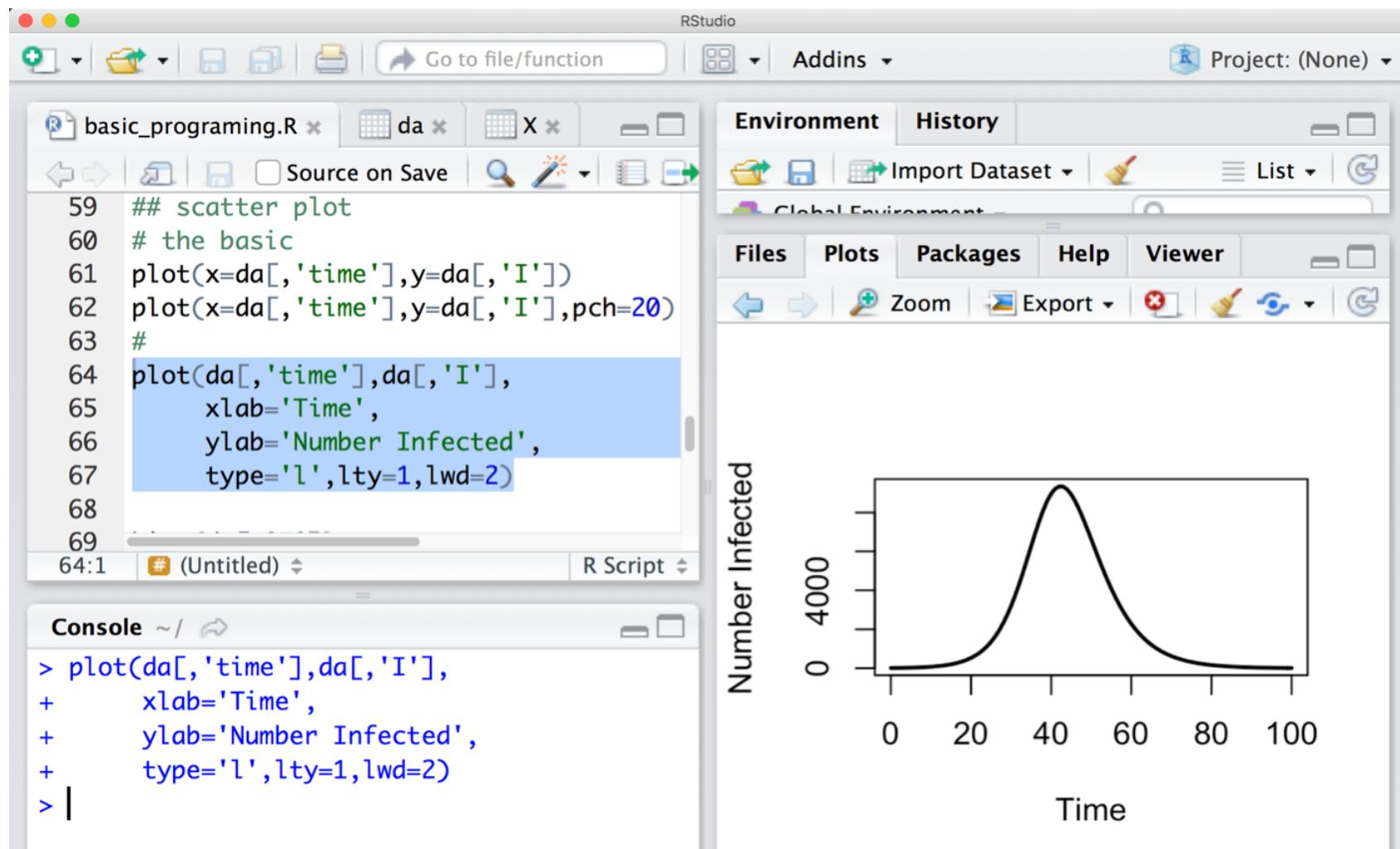
Plots

▶ Scatter plot: `plot(x,y,...)`



Plots

- ▶ Scatter plot: `plot(x,y,...)`
 - Add another line to the same plot: `lines (x, y, ...)`



Plots

▶ Scatter plot: `plot(x,y,...)`

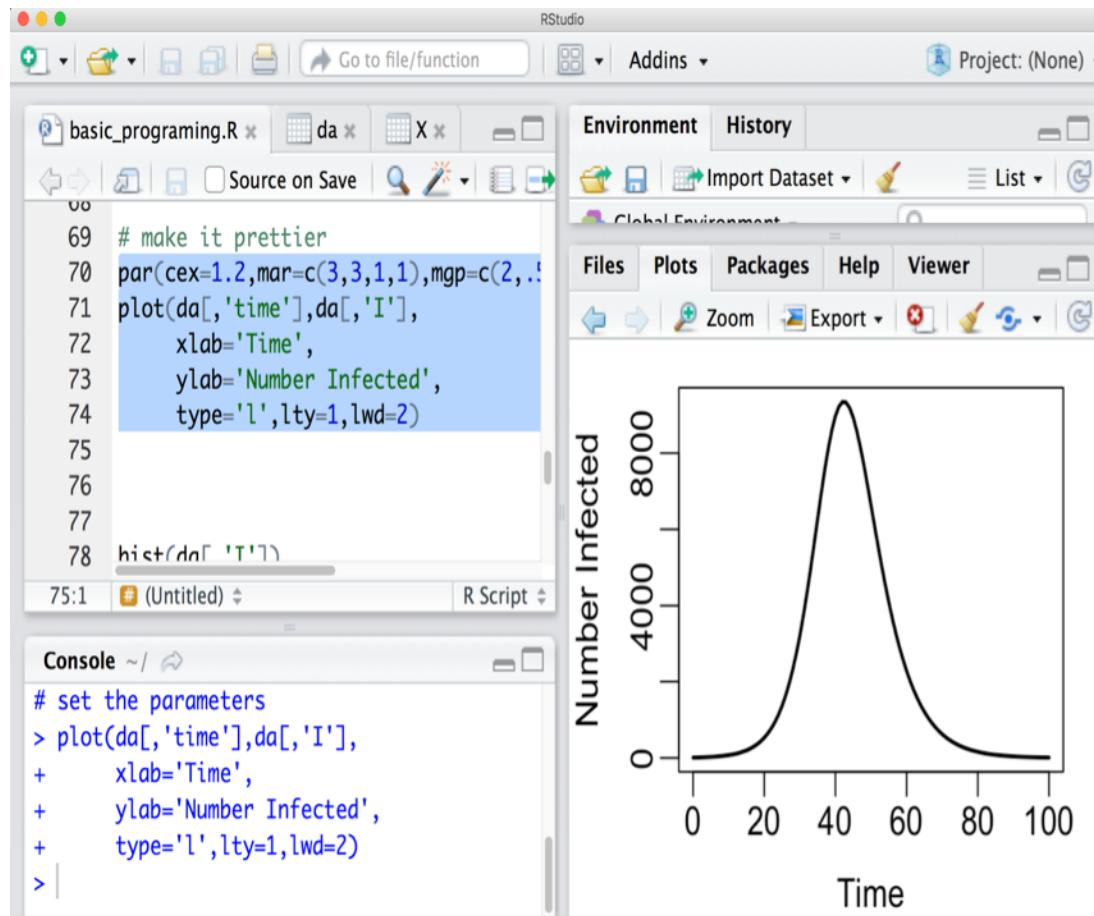
○ Make it prettier: `par()`

❖ E.g.: `par(mar=c(3,3,1,1),
cex=1.2, mgp=c(2,.5,0))`

✓ Margin: `mar=c(3,3,1,1)`

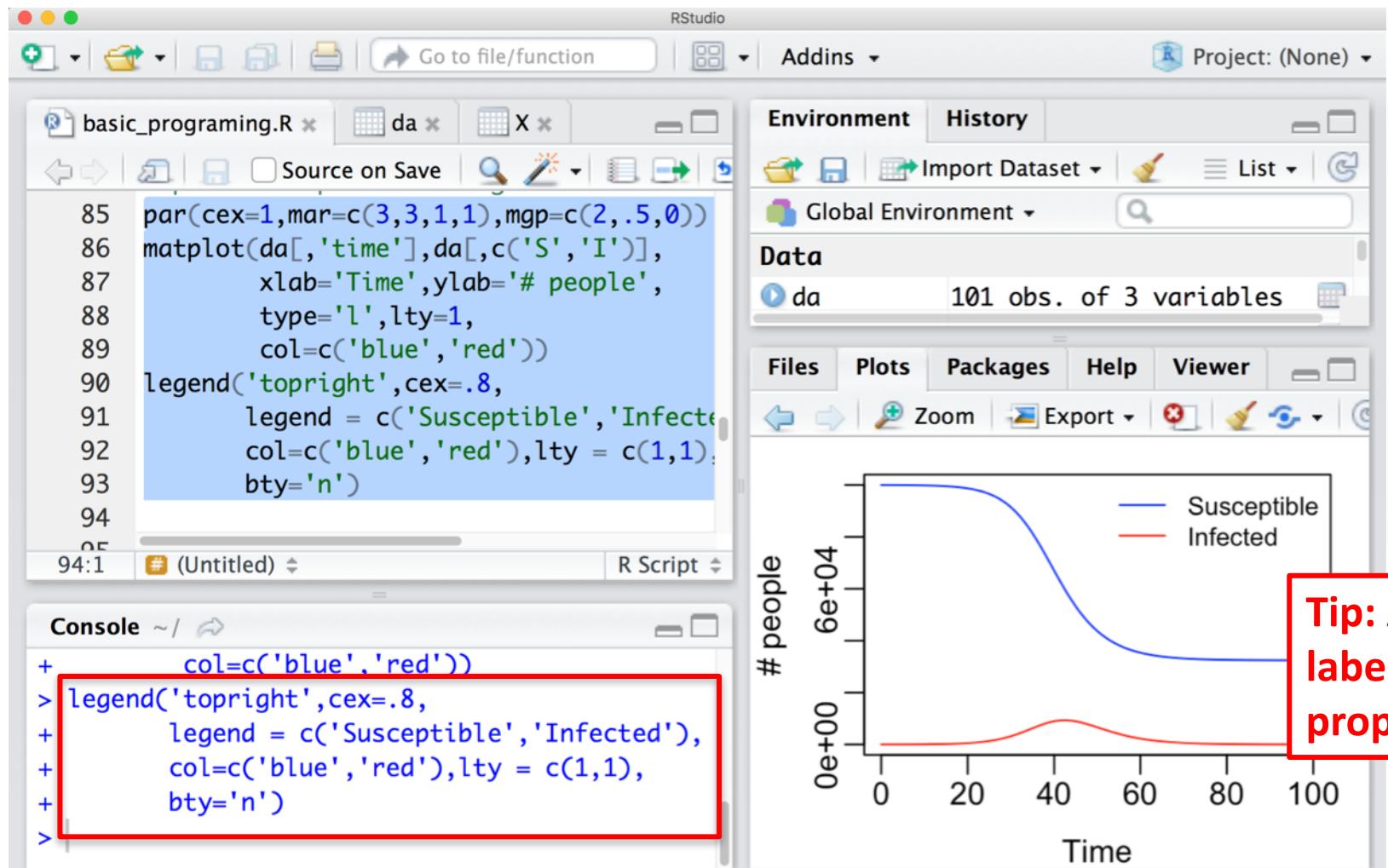
✓ Font size: `cex=1.2`

✓ The margin to axis title,
axis labels and axis line:
`mgp=c(2,0.5,0)`



The Basic: Plots

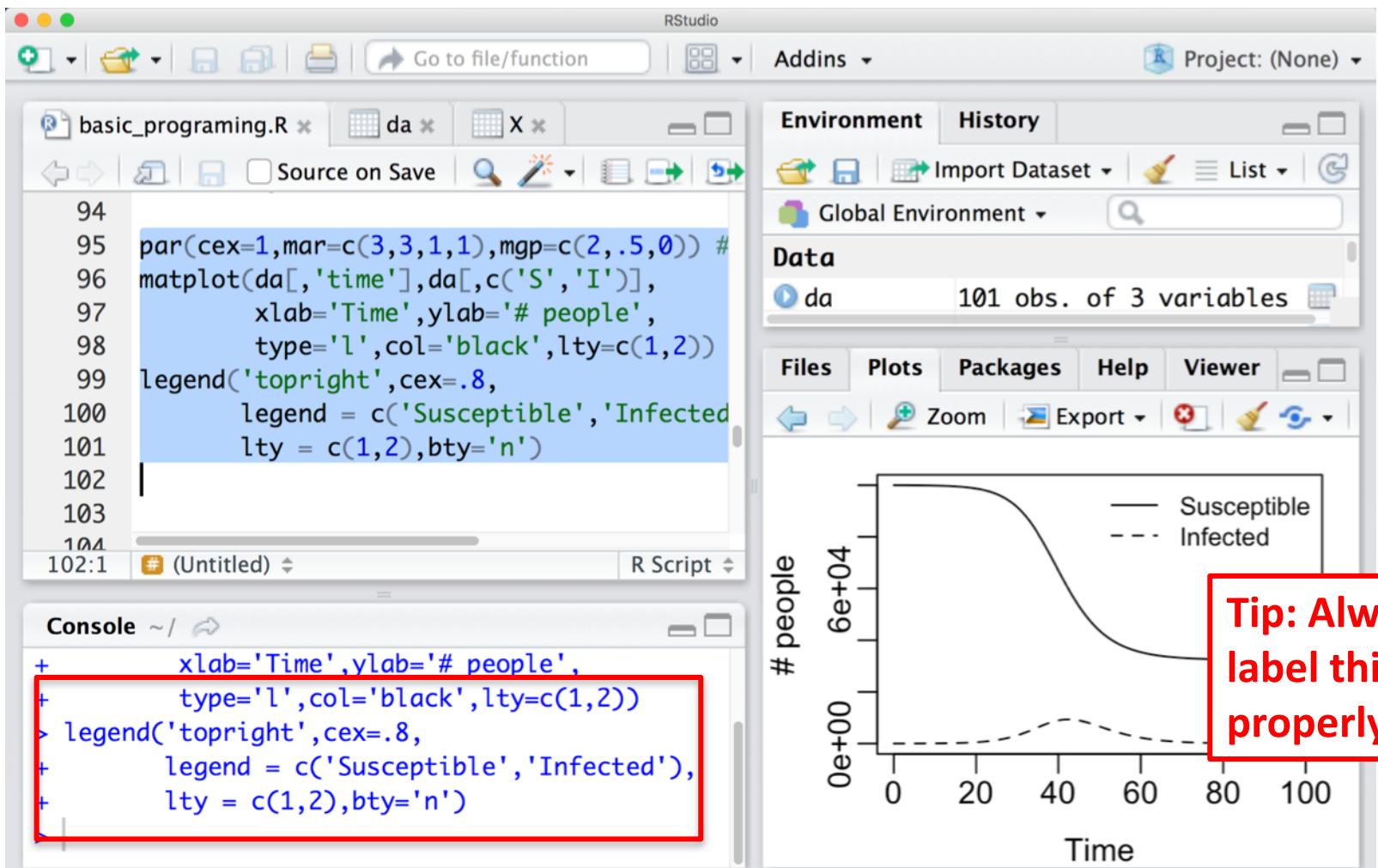
- ▶ Scatter plot & legend: plot multiple lines together
 - ❖ `matplot(x,y,...)`



Tip: Always
label things
properly

Plots

- ▶ Scatter plot & legend: plot multiples line together
 - ❖ `matplot(x,y,...)`



Tip: Always
label things
properly

Plots

► Histogram: `hist(x,...)`

The screenshot shows the RStudio interface with the following components:

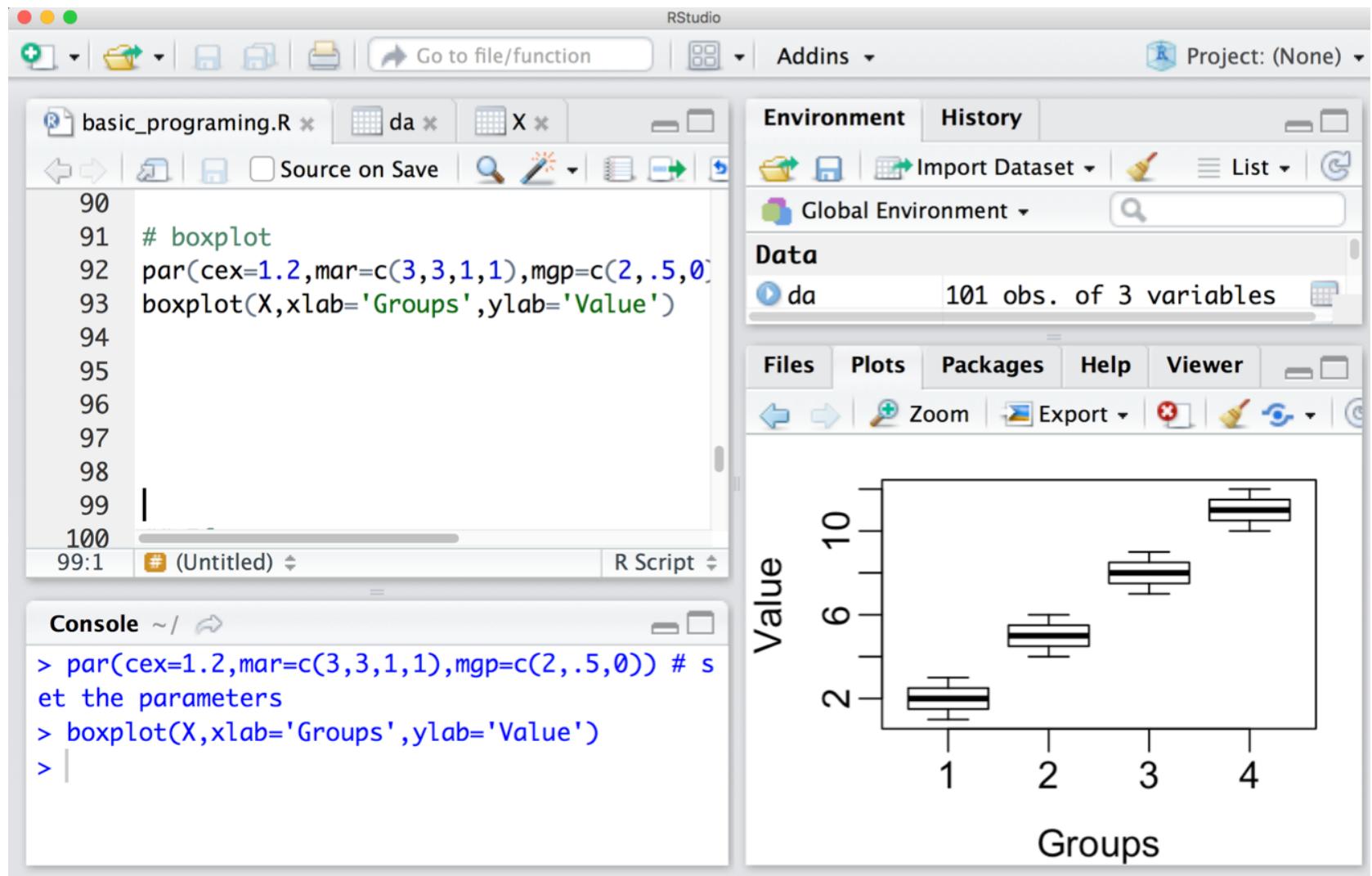
- Code Editor:** Displays the R script `basic_programming.R` containing the following code:

```
84  
85  
86 # histogram  
87 par(cex=1,mar=c(3,3,3,3),mgp=c(2,.5,0))  
88 hist(da[, 'I'],xlab='Infected',  
     main='Histogram of the Infected')  
89  
90  
91  
92  
93 ## If statements  
94
```
- Console:** Shows the command entered in the R console:

```
> par(cex=1,mar=c(3,3,3,3),mgp=c(2,.5,0)) #  
set the parameters  
> hist(da[, 'I'],xlab='Infected',  
+       main='Histogram of the Infected')  
> |
```
- Environment Tab:** Shows the global environment with tabs for Files, Plots, Packages, Help, and Viewer.
- Plots Tab:** Shows a histogram titled "Histogram of the Infected". The x-axis is labeled "Infected" and ranges from 0 to 10,000. The y-axis is labeled "Frequency" and ranges from 0 to 60. The histogram has approximately 10 bins, with the first bin having a frequency of about 58 and subsequent bins decreasing rapidly.

Plots

- ▶ Boxplot: `boxplot (x,...)`
- ▶ Barplot: `barplot (...)`



Plots

► Put multiple plots together

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays R code for generating multiple plots. The code includes:
 - `par(mfrow=c(2,2), cex=.7, mar=c(3,3,2,1), mgp=c(2, .5, 0))` # set the parameters
 - `plot(da[, 'time'], da[, 'I'], main='Infected', xlab='Time', ylab='#Infected', type='l', lty=1, lwd=2)`
 - `plot(da[, 'time'], da[, 'S'], main='Susceptible', xlab='Time', ylab='#Susceptible', type='l', lty=1, lwd=2, col='blue')`
 - `hist(da[, 'I'], xlab='Infected', main='Hist of the Infected')`
 - `boxplot(X, main='boxplot', xlab='Groups', yl`
- Console:** Displays the R command history corresponding to the code in the editor.
- Environment:** Shows the global environment with a data frame named `da` containing 101 observations of 3 variables.
- Plots:** Four plots are displayed side-by-side:
 - Infected:** A line plot showing the number of infected individuals over time, peaking around 40 units of time.
 - Susceptible:** A line plot showing the number of susceptible individuals over time, decreasing from approximately 3e+04 to near zero.
 - Hist of the Infected:** A histogram showing the frequency distribution of infected individuals.
 - boxplot:** A boxplot showing the distribution of values across four groups.

The Basic: Input and Output

- ▶ Read in a file
 - read a .csv file: `read.csv('path')`
 - read a table: `read.table('path')`
- ▶ Save files
 - save as .csv: `write.table(x, 'pathToSave.csv', col.names=T, row.names=F, sep=',') # comma separated values`
 - save R project:
 - ❖ `save.image('path/filename.RData')`

Things to Know: General Programming Tips

- ▶ R is case sensitive
 - $Aa \neq aa$
- ▶ Index in R starts from 1 (not 0)
- ▶ Save your code: write in the script pane, not the console directly
- ▶ Write comments:
 - `#...SO YOU KNOW WHAT YOU ARE DOING 3 MONTHS LATER...`
- ▶ Use the ‘help’ function to learn more about each function
 - Or Google
- ▶ Practice, practice, practice
 - You will remember the syntax after using it many times
 - To do: work through Worksheet Part I
 - Homework (Lab report #1)

More Resources on R

- ▶ More R tutorials:
 - edX (Date Science: R Basics)
<https://www.edx.org/course/data-science-r-basics-harvardx-ph125-1x-0>
 - Youtube, Coursera (<https://www.coursera.org>)
 - <http://www.cyclismo.org/tutorial/R/index.html>
- ▶ Google/ R help
- ▶ CourseWorks:
 - Modules -> R Tutorial -> Short-R-Intro.pdf
 - ❖ Short-R-intro.pdf
 - ❖ Long-R-intro.pdf