UTSA®

ALVAREZ
College of Business
The University of Texas at San Antonio

Introduction to Programming in R

Arka Roy, Ph.D.
arkajyoti.roy@utsa.edu
www.arkajyotiroy.com
MSDA Bootcamp

UTSA®

ALVAREZ
College of Business
The University of Texas at San Antonio

Introduction to Programming in R: Module 1

Basics of R & R studio

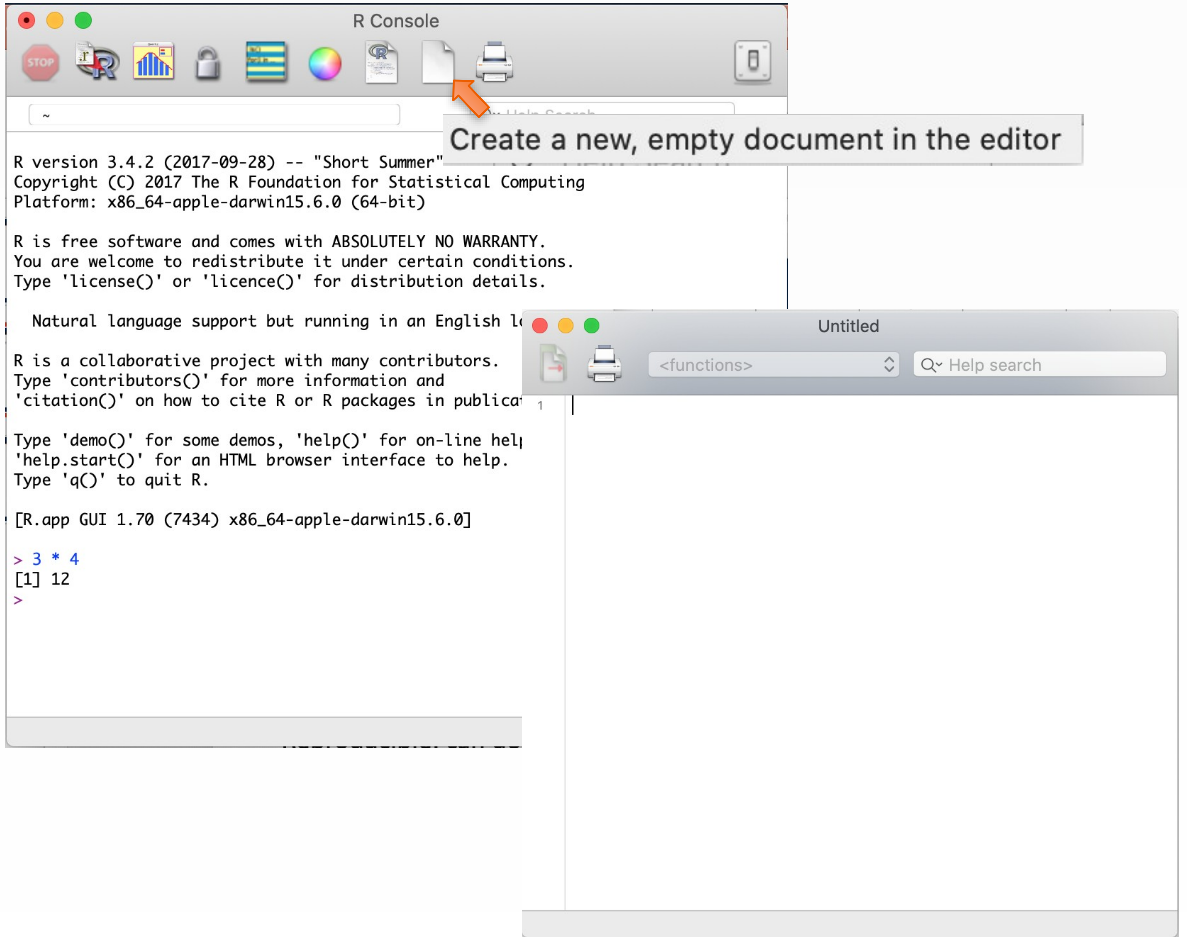
Learning Objectives

- What is *R* and *R*-studio
- Customizing *R*-Studio
- Help in *R*-Studio
- Common commands
- Creating, storing and removing objects
- Input and output in Base *R*
- *R* projects

Note: For install instructions, please see the R install videos and walkthroughs.

What is R?

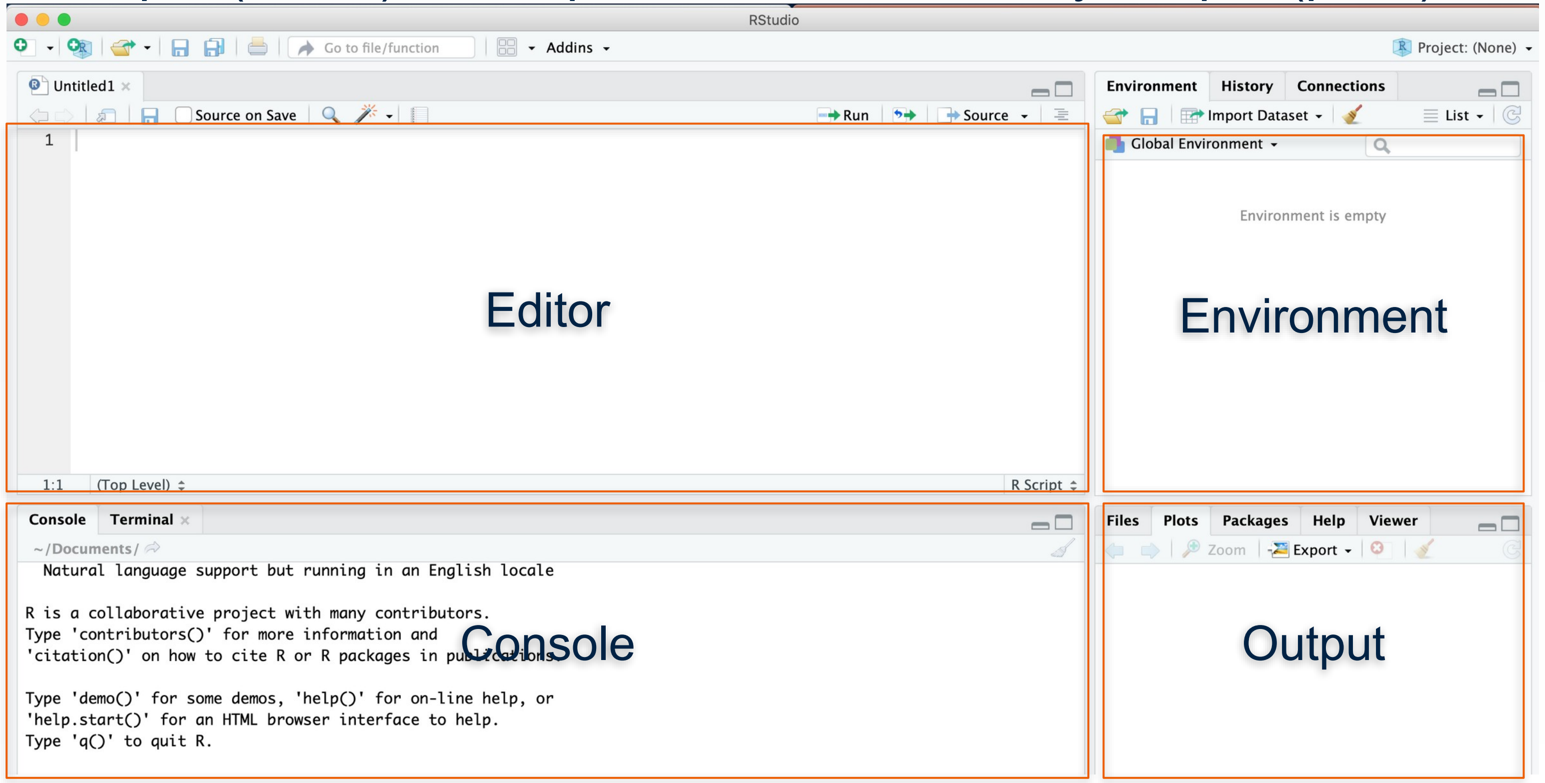
- Programming language
- Interactive environment for Data Science
- Free, open-source platform
- Reproducible, can document (comment) within code
- Contains sophisticated graphical tools for presentation
- Large collection of packages
- Basic layout: console and editor



```
R version 3.4.2 (2017-09-28) -- "Short Summer"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (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 l

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publica

Type 'demo()' for some demos, 'help()' for on-line hel
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.70 (7434) x86_64-apple-darwin15.6.0]

> 3 * 4
[1] 12
>
```

R studio

- GUI for R
- Shows input (editor), workspace variables, history, output (plots), etc.



```
~/Documents/
 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.
```

Customizing R studio

Changing font size/type and appearance of R Studio:

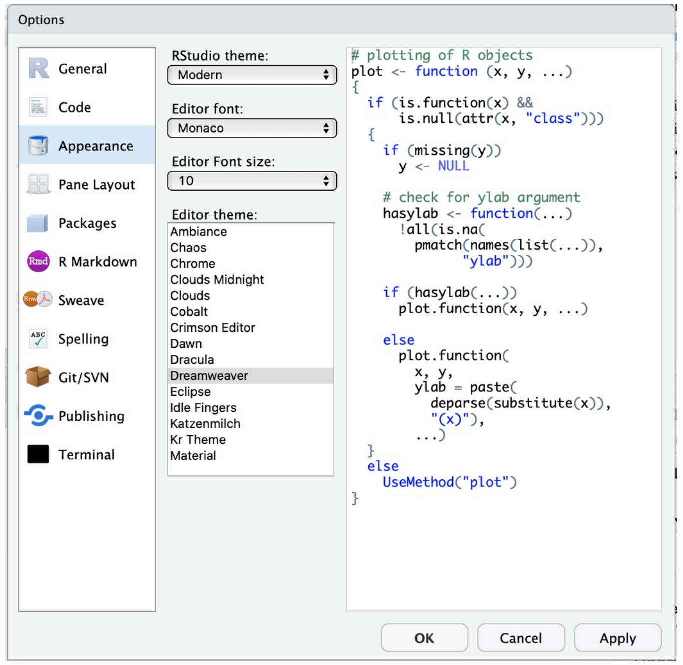
- Tools > Global Options > Appearance

Changing default saving and loading procedures:

- Tools > Global Options > General
  - Uncheck “Restore .Rdata into workspace at startup”.
  - Change “Save workspace to .Rdata on exit” to “Never”.

Changing code display settings:

- Tools > Global Options > Code > Display
  - Check “Show indent guides”.



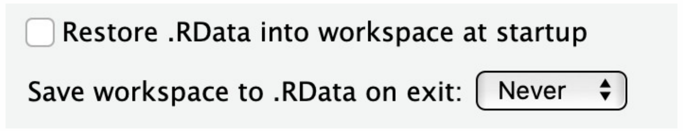
RStudio theme: Modern

Editor font: Monaco

Editor Font size: 10

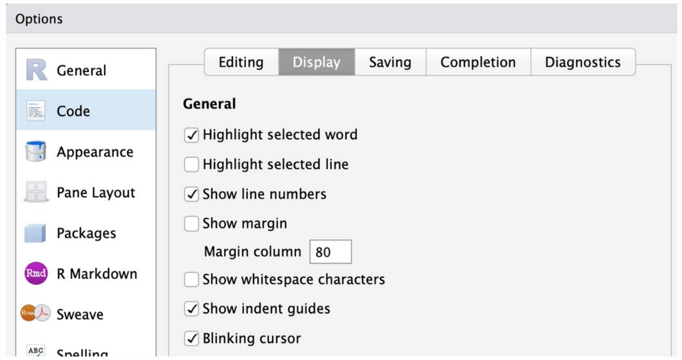
Editor theme:
Ambiance
Chaos
Chrome
Clouds Midnight
Clouds
Cobalt
Crimson Editor
Dawn
Dracula
Dreamweaver
Eclipse
Idle Fingers
Katzenmilch
Kr Theme
Material

```
# plotting of R objects
plot <- function (x, y, ...)
{
  if (is.function(x) &&
      is.null(attr(x, "class")))
  {
    if (missing(y))
      y <- NULL

    # check for ylab argument
    hasylab <- function(...)
      !all(is.na(
        pmatch(names(list(...)),
               "ylab")))

    if (hasylab(...))
      plot.function(x, y, ...)

    else
      plot.function(
        x, y,
        ylab = paste(
          deparse(substitute(x)),
          "(x)"),
        ...)
  }
  else
    UseMethod("plot")
}
```



☐ Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

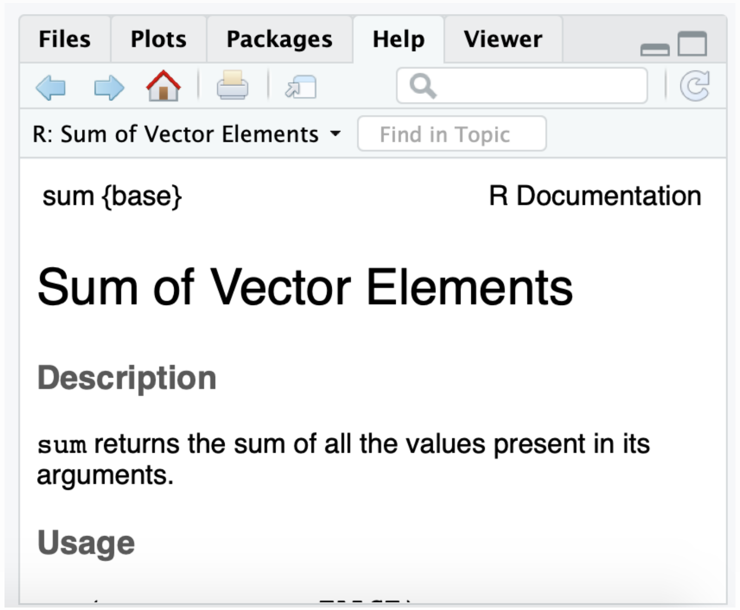


General

☑ Highlight selected word
☐ Highlight selected line
☑ Show line numbers
☐ Show margin
Margin column 80
☐ Show whitespace characters
☑ Show indent guides
☑ Blinking cursor

Help

- Extensive built-in help facility.
- To get help on a function, for e.g., sum, type help(sum) in console.
  - Alternatively, type ?sum for the same output.
- Google it! Add “R” to the query.
- Stack overflow, include [R] to query.
- Common errors:
  - R and R studio are case-sensitive
  - Check spelling!



sum {base} R Documentation

Sum of Vector Elements

Description

`sum` returns the sum of all the values present in its arguments.

Usage

Commands

- Alphanumeric characters for names, or '.' and '_'.

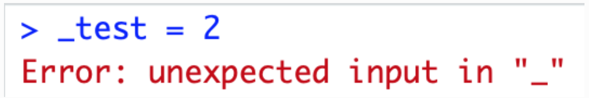
Exceptions:

  - names cannot start with '_'
  - if it starts with '.' then the 2nd letter cannot be a digit.

- Names have "unlimited" length
- Elementary commands are *expressions* or *assignments.*
- Expressions are evaluated, printed (unless made invisible) and value is lost.
- Assignments evaluates and passes value to variable, without printing automatically.
- Commands are separated by newline or ';'
- Comments starts with hash '#'
- If a command is incomplete at the end of the line, R will prompt a '+' on the following line.
- In R and R-studio use '<-' to assign, but in R-studio you can also use '='.
- In console, press return to run the command.
  - From editor, press Run to run selected code, or command + return (Mac) / control + return (Windows).

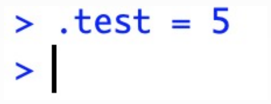
Exercises

- Name a variable '_test' and assign it a value of 2.



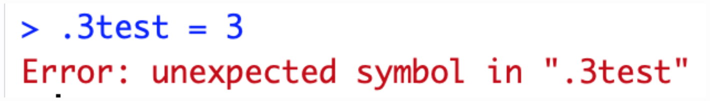
```
> _test = 2
Error: unexpected input in "_"
```

- Name a variable '.test' and assign it a value of 5.
  - Putting period first makes the variable hidden



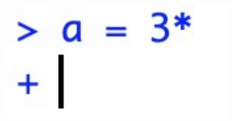
```
> .test = 5
> |
```

- Name a variable .3test and assign it a value of 3.



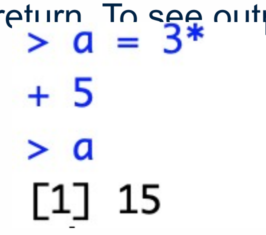
```
> .3test = 3
Error: unexpected symbol in ".3test"
```

- For variable *a*, assign it '3*' and run it without completing.



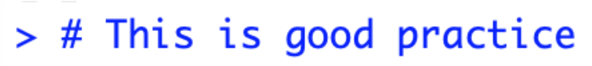
```
> a = 3*
+ |
```

  - Then complete it by typing in '5' and pressing return. To see output, type 'a' and hit return.



```
> a = 3*
+ 5
> a
[1] 15
```

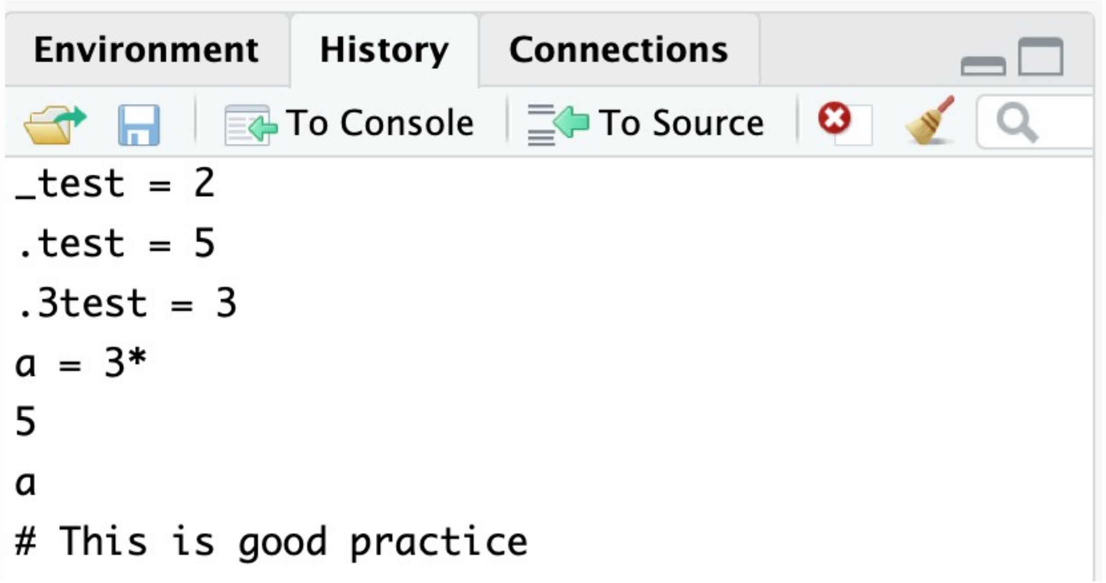
- Type '# This is good practice' in console or editor and run it.



```
> # This is good practice
```

Recalling and Edits

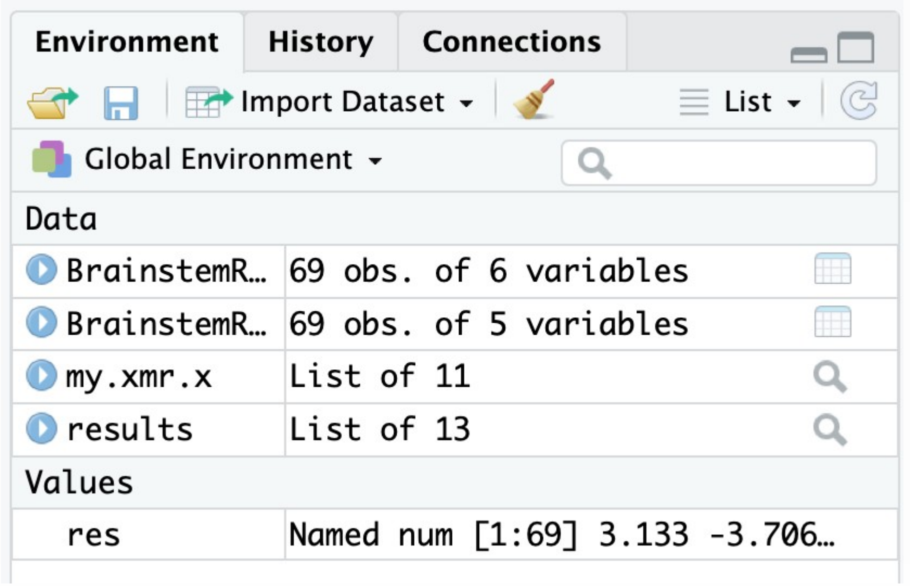
- Commands can be recalled in console using up and down arrow keys or by using comma
  - The use left and rights arrows to move between characters to edit.
- You can also see your previous executed commands in History.
- Function sink(“output.txt”) will divert all output to a text file called output.txt.
  - The function sink() will restore it to the console.



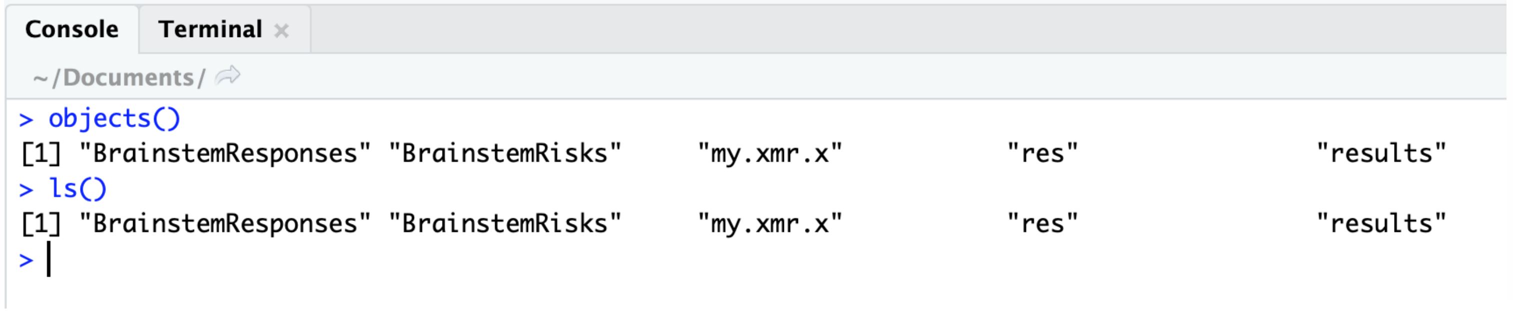
```
_test = 2
.test = 5
.3test = 3
a = 3*
5
a
# This is good practice
```

Objects

- The *Environment*, i.e., workspace, shows all the *objects* created in R.
- Objects can be variables, arrays of numbers, character strings, functions or more general structures such components.
- Functions 'objects()' or 'ls()' will show you the currents objects stored in the workspace.



Data	
BrainstemR…	69 obs. of 6 variables
BrainstemR…	69 obs. of 5 variables
my.xmr.x	List of 11
results	List of 13
Values	
res	Named num [1:69] 3.133 -3.706…



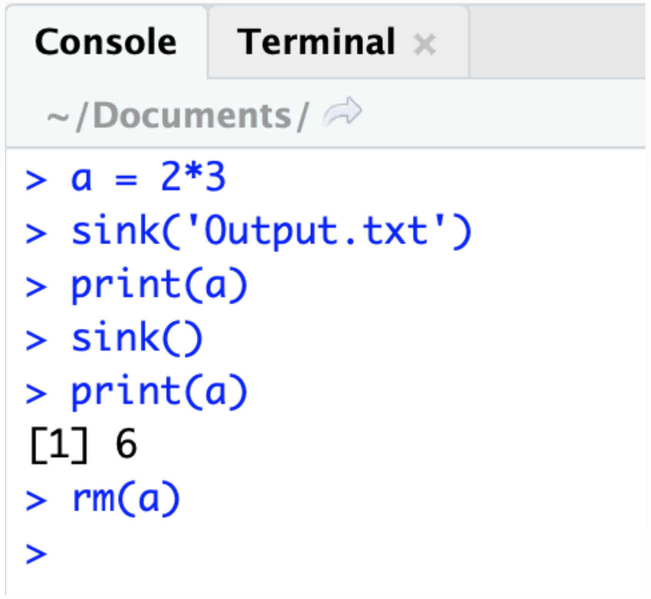
```
> objects()
[1] "BrainstemResponses" "BrainstemRisks"     "my.xmr.x"           "res"                "results"
> ls()
[1] "BrainstemResponses" "BrainstemRisks"     "my.xmr.x"           "res"                "results"
> |
```

Removing and Storing Objects

- Objects can be removed from the workspace using ‘rm()’.
  - For examples to remove the object ‘results’ use rm(results).
- To clear the entire workspace use ‘rm(list=ls())’.
- To clear console, hold down ‘control’ button and hit ‘*l*’ (that is l for like).
- At the end of a session, you can choose to save all the objects in the workspace to a .RData file and all the command lines to a .Rhistory file.
  - When R is restarted from the same directory, all the objects from the .RData file and all the command lines from the .Rhistory file are reloaded automatically.
  - These choices can be changed through customization.

Exercise 1

- Using the console, first create a variable *a* = 2 x *4.*
- Create *b = 3 x 5*.
- Clear the variable *a* from the workspace.
- Finally, clear the console.



```
> a = 2*3
> sink('Output.txt')
> print(a)
> sink()
> print(a)
[1] 6
> rm(a)
>
```

Hint: *Control* + *l* clears console.

Input/Output in Base R

- Before you can read in data (commonly .csv), we need to check what is our working folder.
  - Use *getwd()*
- If the data is located somewhere else, we have two options.

  We can move the working directory to the data folder and the load it.
  - Use *setwd("*path*")* and *read.csv("*filename*”, header* = *T/F)*

  Or we can load the data without moving by supplying the path including filename.
  - Use *read.csv(("*path/filename*”, header* = *T/F)*
- Alternative common functions for reading in text or .csv:

df <- read.table(“pathname/filename", header = T/F, sep = ",") ⇝ use sep = “\t” for tab-separated

df <- read.delim(“pathname/filename", header = T/F) ⇝ wrapper function for read.table with default tab-separated

Input/Output in Base R (cont.)

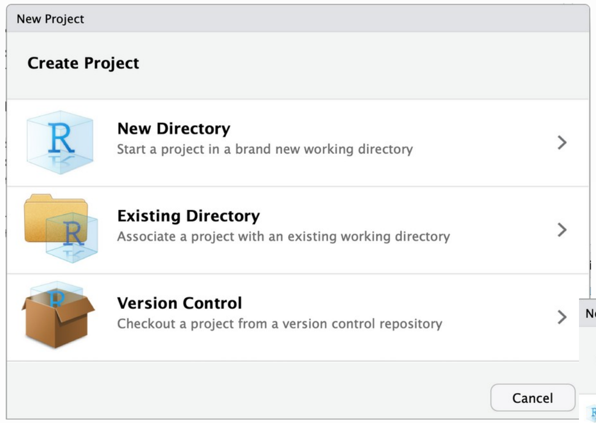
- Like importing data, exporting can be done by supplying the full path with the filename, or just the filename to store it in the working directory.
  - Use *write.csv(variable, "path/filename")* or *setwd("path")* + *write.csv(variable, "filename")*
  - Use *write.table(variable, "path/filename", sep = "\t", row.names = T, col.names = NA)*
    - By default, some importing functions may or may not load column names.
- *Let's import the file "hmda_2017_tx_all_40.csv" with the appropriate headers.*

```
> df <- read.csv("/Users/arkaroy1/Downloads/Texas_data/hmda_2017_tx_all_40.csv", header = T)
```

- *We will look at more advanced importing/exporting data functions in Module 3 using Tidyverse (package).*

R Projects

- Project management is important!
- Working on many projects simultaneously can become messy if the codes and location of codes are not organized well.
- R has a nice built-in project management structure.
    - When starting a new project, go to *File > New Project* from the menu.
    - Select *New Directory*.
    - Select *New Project*.
    - Give a descriptive name and provide a directory and click *Create Project*.
- A folder will be made in that location with a *.Rproj* file that manages everything.


New Project

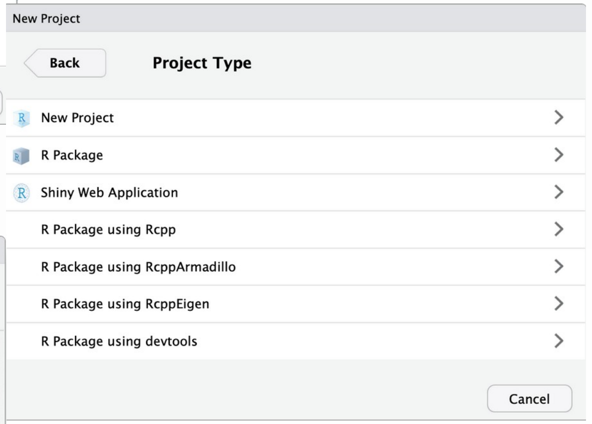
Create Project

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

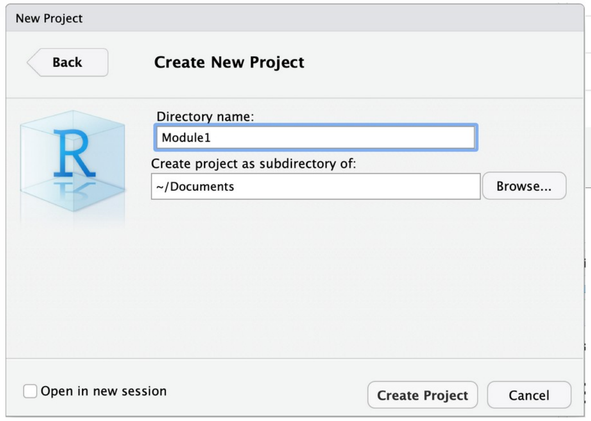
Cancel


New Project

Back — Project Type

New Project
R Package
Shiny Web Application
R Package using Rcpp
R Package using RcppArmadillo
R Package using RcppEigen
R Package using devtools

Cancel


New Project

Back — Create New Project

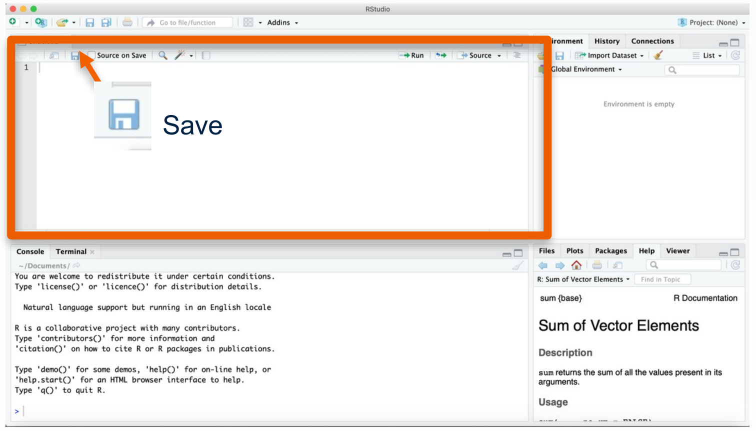
Directory name:
Module1
Create project as subdirectory of:
~/Documents
Browse...

Open in new session

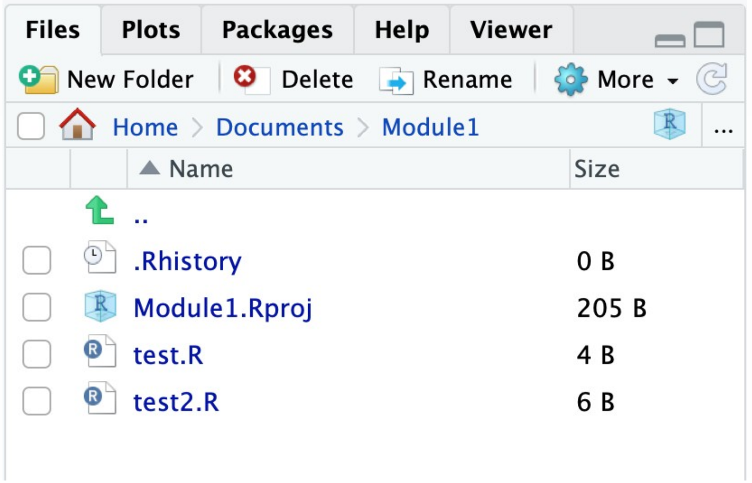
Create Project — Cancel

R Projects (cont.)

- You can write multiple commands in a single document, i.e. scripts, using the editor.
    - To create one, go to File > New File > R script. To save it, click on the floppy disk icon (save current document).
- All files that you work on, including editor files, data files, command history, etc. should be <u>saved</u> in the folder created for the project.
- You can open the project file *.Rproj*, either by double clicking the *.Rproj* files or by going to File > Open Project.
    - All the associated files will be visible on the right-hand side File window.
    - To open them, simply click on the desired file.



```
~/Documents/
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.

>
```



Name	Size
..	
.Rhistory	0 B
Module1.Rproj	205 B
test.R	4 B
test2.R	6 B

Exercise 2

- Create a R project for the next the next Lesson, which will start off with data types.
- Call the project *'Data Types'* and store it in an appropriate location.
- Create a script file named '*datatypes.R'* and store it in the Data Types folder.
    - We will use this in the next section on Data Types.
- *Save the df variable in your environment in the Data Types folder.*

Next module is Data Types.