Introduction to R
Data Science Skills Day 2022

Anjali Silva, PhD

Summer Undergraduate Data Science Research Program
University of Toronto
03 June 2022

a.silva@utoronto.ca anjalisilva.github.io @Silva_Anjali

Welcome!

- Instructor: Anjali Silva, PhD
  - Researcher and Lecturer, Department of Cell & Systems Biology, U of T
  - Data Analyst, University of Toronto Libraries
  - Pronouns: she/her
  - Name Phonetic: Un-j-li Sil-va

Course Description

- Introduction to R
  Data Science Skills Day

  - The vast amount of data produced by evolving information technology requires tools and skills. Among the many tools, R is a free, open-source language for data sciences. R is a programming language that can aid in the process of data analysis. This course is a beginner level, introductory course for R for data analysis. We will learn about R, RStudio (the environment use to work in R), including installation, and apply R for beginner-level data modeling and visualization. By the end of the course, you'll have a introduction to the flexibility of R, different functionalities, and understand how to apply it for basic data exploration.

  - Friday 10:00 am – 4 pm EST; online - synchronous.

Material

- Instructor Slides:
  - https://github.com/anjalisilva/DSI_IntroductionToR
  - SlideIntroR2022.pdf
- Instructor R Script:
  - https://github.com/anjalisilva/DSI_IntroductionToR
  - Script.R

Course Objectives

- Learning Objectives:
    - Install R and RStudio
    - Navigate the RStudio environment
    - Discover how to use RStudio to apply R to your analysis.
    - Importing data from a spreadsheet
    - View attributes of a dataset
    - Understand differences in varying data types and structure
    - Write and test functions
    - Generate simple visualizations
    - Be aware of sources for getting help in R
    - Be aware of sources for expanding skills in R

Course Expectations

- Be respectful.
- One speaker at a time.
- Keep yourself on mute, unless you need to speak or ask a question.
- You may save your questions to 'Any questions?' section.
- If you have a question, use raise hand feature. First say your name, then ask the question.
- If you have a question, you may type it to chat as well.

Course Expectations

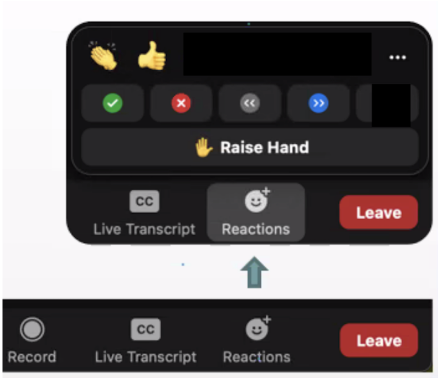
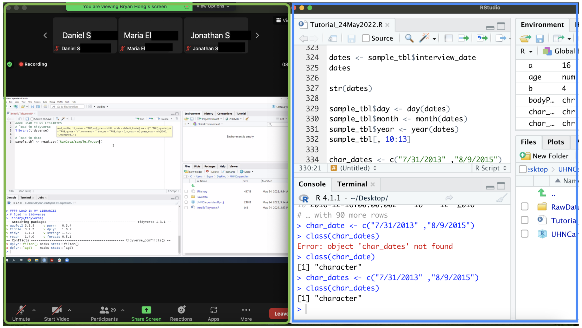


Figure: Zoom 'Reactions' that you may use.

Navigating an Online Code-Along Workshop



```
#### LOAD IN MY LIBRARIES
# load in tidyverse
library(tidyverse)

# load in data
sample_tbl <- read_csv('RawData/sample_fw.csv')
```

```
> #### LOAD IN MY LIBRARIES
> # load in tidyverse
> library(tidyverse)
-- Attaching packages --------------------------------------- tidyverse 1.3.1 --
v ggplot2 3.3.5     v purrr   0.3.4
v tibble  3.1.2     v dplyr   1.0.7
v tidyr   1.1.3     v stringr 1.4.0
v readr   1.4.0     v forcats 0.5.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
>
```

Tutorial_24May2022.R

```
dates <- sample_tbl$interview_date
dates

str(dates)

sample_tbl$day <- day(dates)
sample_tbl$month <- month(dates)
sample_tbl$year <- year(dates)
sample_tbl[, 10:13]

char_dates <- c("7/31/2013" ,"8/9/2015")
```

```
R 4.1.1 · ~/Desktop/
# … with 90 more rows
> char_date <- c("7/31/2013" ,"8/9/2015")
> class(char_dates)
Error: object 'char_dates' not found
> class(char_date)
[1] "character"
> char_dates <- c("7/31/2013" ,"8/9/2015")
> class(char_dates)
[1] "character"
>
```

a	16
age	num
b	4
bodyP…	chr
char_…	chr
char_…	chr

Figure: It is recommended that windows are resized so that both the user RStudio window and Instructor Zoom window (with RStudio) is visible at the same time. User may collapse panels of their RStudio not in current use.

Outline

Time	Topic
10.00 -10.10 am	Introduction
10.10 - 11.00 am	Setup and RStudio
11.00 - 11.15 am	Short Break
11.15 - 12.15 pm	Analyzing Patient Data
12.15 - 1.00 pm	Lunch
1.00 - 2.15 pm	Data Types and Structures
2.15 - 2.30 pm	Short Break
2.30 - 3.45 pm	Creating Functions
3.45 - 4.00 pm	Next Steps and Final Remarks

Any questions?

R and RStudio

What is R?

- A language and environment for statistical computing and graphics.
- R was initially written by Ross Ihaka and Robert Gentleman.
- Since mid-1997, the R Core Team modify the R source.
- R runs on a wide variety of UNIX platforms, Windows and MacOS.

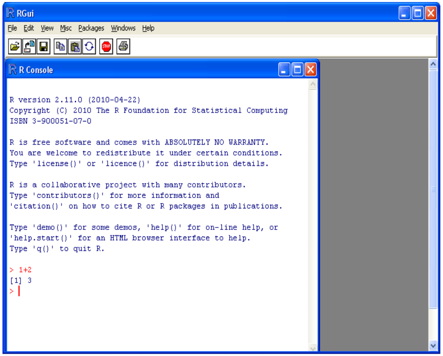
R continue...

- R is a scripting language, thus an interpreter executes commands one line at a time.
- A Free software under the terms of the GNU General Public License.
- R home page: https://www.R-project.org/
- How can R be obtained?
  - Via CRAN, the "Comprehensive R Archive Network".
  - https://cran.r-project.org/

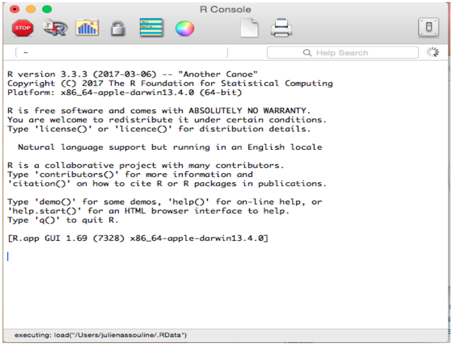
R continue...

- How can R be installed?
    - Unix
        - https://cran.r-project.org/doc/FAQ/R-FAQ.html#How-can-R-be-installed-_0028Unix_002dlike_0029
    - Windows
        - https://cran.r-project.org/bin/windows/base/
    - Mac
        - https://cran.r-project.org/bin/macosx/

R continue...



```
R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

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.

> 1+2
[1] 3
>
```



```
R version 3.3.3 (2017-03-06) -- "Another Canoe"
Copyright (C) 2017 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 1.69 (7328) x86_64-apple-darwin13.4.0]

executing: load("/Users/julienassouline/.RData")
```

R continue...

- R can be used interactively or non-interactively.
    - Interactively, with or without an integrated development environment (IDE): RStudio.
    - Non-interactively via scripts.
- R is designed with interactive data exploration in mind.
- A version of R is released each year. Current release is 4.2.0.

R continue...

- Ihaka,R. and Gentleman,R. (1996) R: a language for data analysis and graphics. *J. Comput. Graph. Statist.*, 5, 299–314.

RStudio

- RStudio is the Graphical User Interface for R.
- Not only for R, but can also use Python.
- An integrated development environment with:
  - A console
  - Syntax-highlighting editor for code execution
  - Tools for plotting, viewing history, debugging and workspace management

Why learn R?

- Not only is R free, but it is also open-source and cross-platform.
- R code is great for reproducibility.
- R is interdisciplinary and extensible.
- R works on data of all shapes and sizes.
- R produces high-quality graphics.
- R has a large and welcoming community.

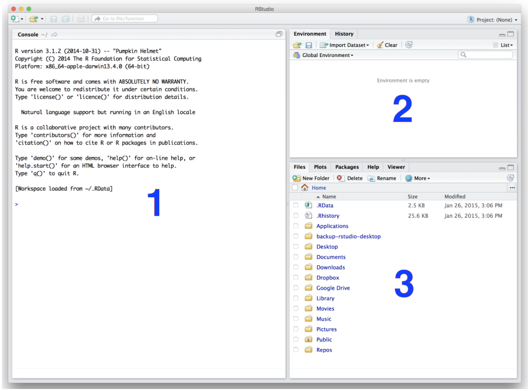
Documentation for R

- Online documentation for functions and variables in R exists.
- Obtained by typing *help(FunctionName)* or *?FunctionName* at the R prompt, where FunctionName is name of function.
- E.g., if ‘sum’ is the function then:

```
> help(sum)
> ?sum
```

RStudio

- RStudio contains many features that make the development process easier and faster.

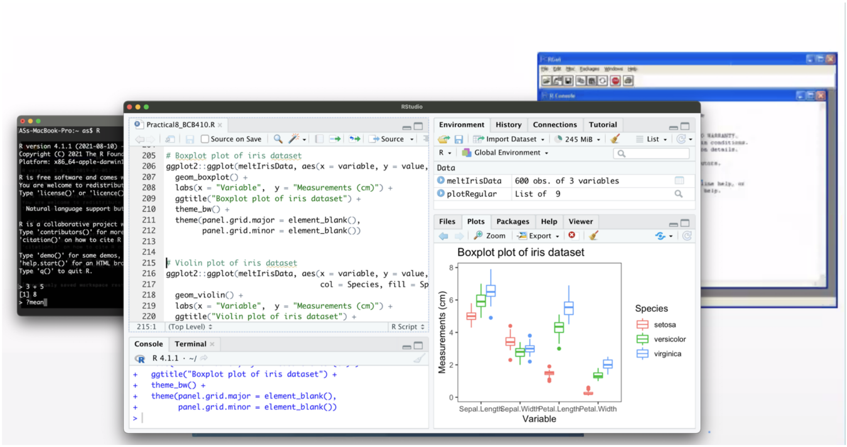


```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)

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.

[Workspace loaded from ~/.RData]

>
```

Name	Size	Modified
.RData	2.5 KB	Jan 26, 2015, 3:06 PM
.Rhistory	25.6 KB	Jan 26, 2015, 3:06 PM
Applications		
backup-rstudio-desktop		
Desktop		
Documents		
Downloads		
Dropbox		
Google Drive		
Library		
Movies		
Music		
Pictures		
Public		
Repos		

Figure: Anatomy of default RStudio. 1. This is the Console. 2. Environment and History. 3. Files, Plots, Packages, Help and Viewer. If a script is opened up, it will appear on top of Console.

Options To Work With R



```
ASs-MacBook-Pro:~ as$ R

R version 4.1.1 (2021-08-10) -
Copyright (C) 2021 The R Found
Platform: x86_64-apple-darwin1

R is free software and comes w
You are welcome to redistribut
Type 'license()' or 'licence()

  Natural language support but

R is a collaborative project w
Type 'contributors()' for more
'citation()' on how to cite R

Type 'demo()' for some demos,
'help.start()' for an HTML bro
Type 'q()' to quit R.

> 3 + 5
[1] 8
> ?mean
```

```
# Boxplot plot of iris dataset
ggplot2::ggplot(meltIrisData, aes(x = variable, y = value,
  geom_boxplot() +
  labs(x = "Variable",  y = "Measurements (cm)") +
  ggtitle("Boxplot plot of iris dataset") +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())


# Violin plot of iris dataset
ggplot2::ggplot(meltIrisData, aes(x = variable, y = value,
                                  col = Species, fill = Sp
  geom_violin() +
  labs(x = "Variable",  y = "Measurements (cm)") +
  ggtitle("Violin plot of iris dataset") +
```

```
+   ggtitle("Boxplot plot of iris dataset") +
+   theme_bw() +
+   theme(panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank())
> 
```

```
Data
meltIrisData   600 obs. of 3 variables
plotRegular    List of 9
```

Figure: Options to work with R.

Options To Work With R

Any questions?

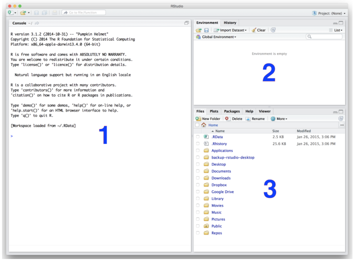
Practical - Setup

- This lesson assumes you have (current) versions of the following installed on your computer:
  1. the R software itself, and
  2. RStudio Desktop
- Any questions with R or RStudio setup?
- If you have downloaded R and RStudio, read about RStudio `http://swcarpentry.github.io/r-novice-inflammation/09-supp-intro-rstudio/index.html`

Practical - Explore RStudio

RStudio

- By now, you should have RStudio installed.



```
R version 3.1.2 (2014-10-31) -- "Pumpkin Helmet"
Copyright (C) 2014 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.

[Workspace loaded from ~/.RData]

>
```

- There are two main ways of interacting with R:
  - Using the console
  - By using script files

- Click on 'Tools' → 'Keyboard Shortcuts Help' for shortcuts.

Interacting with R

- Console:
  - Type commands directly into the console and press ‘Enter’ to execute.
- Script:
  - Put cursor at the end of the line to execute OR highlight the section.
  - Press ‘Ctrl’ + ‘Enter’ on Windows, Mac OR ‘Cmd’ + ‘Return’ on Mac.
- Clear console with ‘Ctrl’ + ‘L’.
- If R is still waiting for you to enter more text, the console will show a + prompt.

R Project

- Good to keep data, analyses, and text in a single folder.
- RStudio interface for this is Projects.
    - File → New project; choose New directory → New project
- Enter a name for this new folder ("directory") and choose a convenient location for it. This will be your working directory.
- On Desktop, save as 'DSI$_{I}ntroR'$
- Click on 'Create' project.
- Create a new file where we will type our scripts.
    - Go to File → New File → R script. Click the save icon on your toolbar and save your script as "script.R".

Any questions?

Some Basics

Organize Working Directory

- Structure of the working directory is very important.

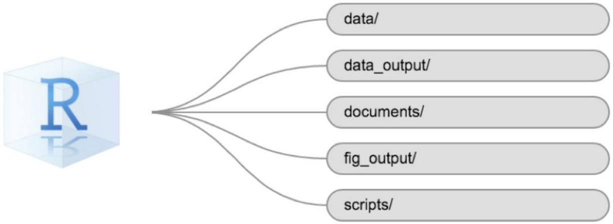


Figure: Examples of suggested directories within working directory or R Project. Figure from: `https://datacarpentry.org/r-socialsci/00-intro/index.html`

R Coding Style

- **Very important** when writing R packages.
- Review the R coding style (content) outlined here:
  - `http://steipe.biochemistry.utoronto.ca/abc/index.php/RPR-Coding_style`

R Features

- In R, the indexing begins from 1.
- R is case sensitive ("X" is not the same as "x").
- R uses dynamic variable typing, so variables can be used over and over again.

Assignment and Commenting

- The $\leftarrow$ symbol is the assignment operator.
- To assign a value to a variable or object called 'test1'

```
test1 <- 123
test1
```

- Comment using # character

```
test1 <- 123  # This is a comment
test1 # This is called auto-printing
```

R Built-in Functions

- There are many built-in functions. You will learn these as you go.
- Functions combine a sequence of expressions that are executed to achieve a goal.
- The “argument” of the function is provided inside the brackets.
- The “return value” of the function is the value provided back.

R Built-in Functions

- We will cover some basic functions:

```
x <- 5

x # auto-printing

print(x) # explicit printing using print() function

typeof(x) # "double" obtained using typeof() function

length(x) # 1 obtained using length() function
```

R Packages

- Packages are collections of R functions, data, and compiled code.
- Libraries are directories in R where the packages are stored.
- Built-in functions are part of R standard or base packages and do not need to be downloaded.

```
library(help = "base")
library(help = "stats")
```

- Function print() is part of base R package.

R Built-in Functions

- Standard or base R package contains basic functions for R to function as a language: arithmetic, input/output, basic graphics, etc.
- Some functions are not built-in. To get these, need to download pacakges.
- We will cover downloading of packages later.

R Version

- To obtain session information:
  `sessionInfo()`
- Version information:
  `R.Version()`
- Show objects in workspace
  `ls()`

R Help Function

- Getting help:

```
?"<-"  # help on assignment operator
help("<-") # help on assignment operator

?typeof # help on typeof function

?class # help on class function

?print # help on print function
```

Any questions?

R Data Types

- Numeric: floating types (double precision).
- Logicals: booleans = TRUE/FALSE or T/F.
- Character strings.
- Examples:

```
xValue <- 100
xValue

yVariable <- FALSE
yVariable

zVariable <- "hello"
zVariable
```

R Class

- Numbers in R are usually treated as numeric objects (i.e., double precision real numbers).
- To explicitly assign an integer, need to specify the L suffix.

```
x <- 1L
x
class(x) # "integer"
```

R Class

- Complex class:

```
x <- c(2 + 0i, 5 + 4i)
class(x) # "complex"
```

- Inf represents infinity:

```
Inf
1 / Inf # 0
```

- NaN represents an undefined value/missing value:

```
NaN # not a number
0 / 0 # NaN
```

Concatenating

- c() function concatenating elements together:

```
x <- c(0.5, 0.6)
class(x) # "numeric"

x <- c("a", "b", "c")
class(x) # "character"

x <- c(TRUE, FALSE)
class(x) # "logical"
```

Character Strings

- Character strings are collections of characters.
- Provided as values in single or double quotes.

```
xVariable <- ‘hello’
class(xVariable) # "character"

zVariable <- "hello"
class(zVariable) # "character"
```

- “paste” converts inputs to strings, concatenate and return:

```
paste(xVariable)
```

Character Strings

- "cat" concatenates and prints the arguments to the screen:

  ```
  cat("\n", xVariable, zVariable) # "\n" adds new line
  ```

- "print" prints the argument:

  ```
  print(c(zVariable, xVariable))
  ```

Missing Values

- Missing values are denoted by NA (Not Available) or NaN (Not a Number).

```
x <- c(1, 3, NA, 4, 5)
class(x) # "numeric"

y <- c(1, 3, NaN, 4, 5)
class(y) # "numeric"

# is.na() is used to test objects if they are NA
# is.nan() is used to test for NaN

is.na(x) # FALSE FALSE  TRUE FALSE FALSE
is.nan(x) # FALSE FALSE FALSE FALSE FALSE
```

Question: What is the difference between NA and NaN in R?

Any questions?

Tips for Solving Issues

- Copy and paste the entire **exact** error message into Google.
  - Someone else may have gotten this same error and has asked a question.
- Copy and paste the entire error message into Google, followed by ‘r’.
- Google the name of the function with term ‘tutorial r’ to see tutorials.
- If struggling with code for a plot, Google ‘r plot plotname’, then click on Images.
- If errors with reading files, ensure path is correct. Check using getwd().

Where To Get Help

- RStudio Community: `https://community.rstudio.com/`.
- Carpentries: `https://datacarpentry.org/r-socialsci/`.
- StackOverflow: `https://stackoverflow.com/`.
- R-help mailing list: `https://stat.ethz.ch/mailman/listinfo/r-help`.

Vectors

Vectors

- Vector is a basic data structure in R.
- An R vector can only contain objects of the same class.
- There are multipel ways to create a vector:

```
y <- 1L:5L  # vector using : operator
is.vector(y)

y <- c(1, 2, 3, 4, 5) # vector using c() function
is.vector(y)

y <- seq(1, 5, by = 1) # vector using seq() function
is.vector(y)
```

Vectors

- There are multipel ways to create a vector:

```
# vector using paste() function
y <- paste("A", 1:5, sep = "")
is.vector(y)

# vector using rep() function
y <- rep(letters[1:5], 3)
is.vector(y)
```

Vectors

- There are multipel ways to create a vector:

```
?vector # vector using vector() function
# vector(mode, length )
```

- The atomic modes are "logical", "integer", "numeric" (synonym "double"), "complex", "character" and "raw":

```
# Initialize vector of certain length
x <- vector(mode = "numeric", length = 5)
x # 0 0 0 0 0

# Initialize vector of certain length
x <- vector(mode = "character", length = 10)
x # "" "" "" "" "" "" "" "" "" ""
```

Vectors

- If mixing objects of two different classes in a vector, every element in the vector is forced to be same class.

```
y <- c(2.2, "a")
class(y) # "character"
y
```

Vectors

- Sometimes objects can be coerced from one class to another using the as.* functions:

```
x <- 1L:10L
x # 1  2  3  4  5  6  7  8  9 10
class(x) # "integer"

z <- as.character(x)
z # "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
class(z) # "character"
```

Vectors

- Sometimes objects can be coerced from one class to another using the as.* functions:

```
w <- c("a", "b", "c")
w
class(w)
q <- as.numeric(w)
q # NA NA NA
```

Question:

Vector index in R starts from ____.

Content of Vectors

- To access the contents of the vector use [ ]:

```
x <- 20:30 # vector
x # 20 21 22 23 24 25 26 27 28 29 30
length(x) # 11
x[1] # 20
x[15] # NA

x[c(1, 2, 4)] # 20 21 23
```

- To remove elements:

```
x[c(-2, -4)]
```

Question:
Can you mix positive and negative integers when accessing elements of a vector?

Content of Vectors

- Can you mix positive and negative integers when accessing elements of a vector?

```
x <- 20:30 # vector
x # 20 21 22 23 24 25 26 27 28 29 30

x[c(2, -4)] # ?
```

Content of Vectors

- There are several ways to modify vectors:

```
x <- 20:30 # vector
x # 20 21 22 23 24 25 26 27 28 29 30

x[1] <- 10
x # 10 21 22 23 24 25 26 27 28 29 30

x[1:3] <- 10
x # 10 10 10 23 24 25 26 27 28 29 30

x[x < 25] <- 5
x # 5  5  5  5  5 25 26 27 28 29 30
```

Any questions?

Matrices & Lists

Matrices

- Matrices are constructed column-wise.
- Entries can be thought of starting in the “upper left” corner and running down the columns.
- Matrices must have every element be the same class (e.g., all integers).

```
?matrix
matrixOne <- matrix(data = 1:5, nrow = 2, ncol = 3)
matrixOne <- matrix(data = 1:6, nrow = 2, ncol = 3)
matrixOne

dim(matrixOne) # dimension 2 3
nrow(matrixOne) # 2
ncol(matrixOne) # 3
attributes(matrixOne)
```

Matrices

- Column-binding or row-binding can be done by cbind() and rbind() functions.

```
a <- 1:4
b <- 5:8
c <- cbind(a, b)
c
dim(c) # 4 2

d <- rbind(a, b)
d
dim(d) # 2 4
```

Lists

- A list is represented as a vector but can contain objects of different classes.

```
listOne <- list(16, "abc", TRUE, 5 + 4i)
listOne
length(listOne) # 4
typeof(listOne) # "list"
class(listOne) # "list"

# access the contents of the list
listOne[[1]] # 16
listOne[[2]] # "abc"
listOne[[4]] # "5+4i"
```

Lists

- Empty lists can be created using vector() function:

```
listTwo <- vector(mode = "list", length = 5)
listTwo
length(listTwo) # 5
```

- Lists can have names:

```
listDestinations1 <- list(1, 2, 3)
listDestinations1
names(listDestinations1) # NULL
names(listDestinations1) <- c("Canada", "Alaska",
"England")
listDestinations1
```

Any questions?