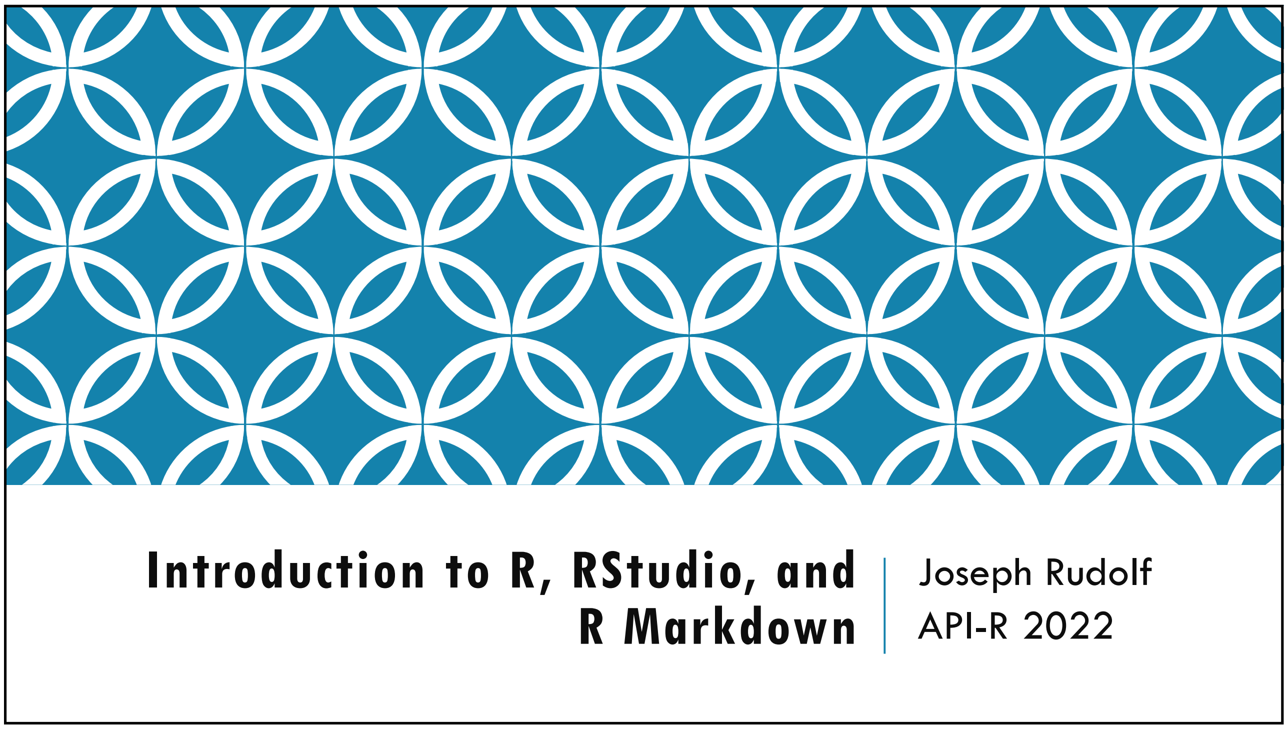
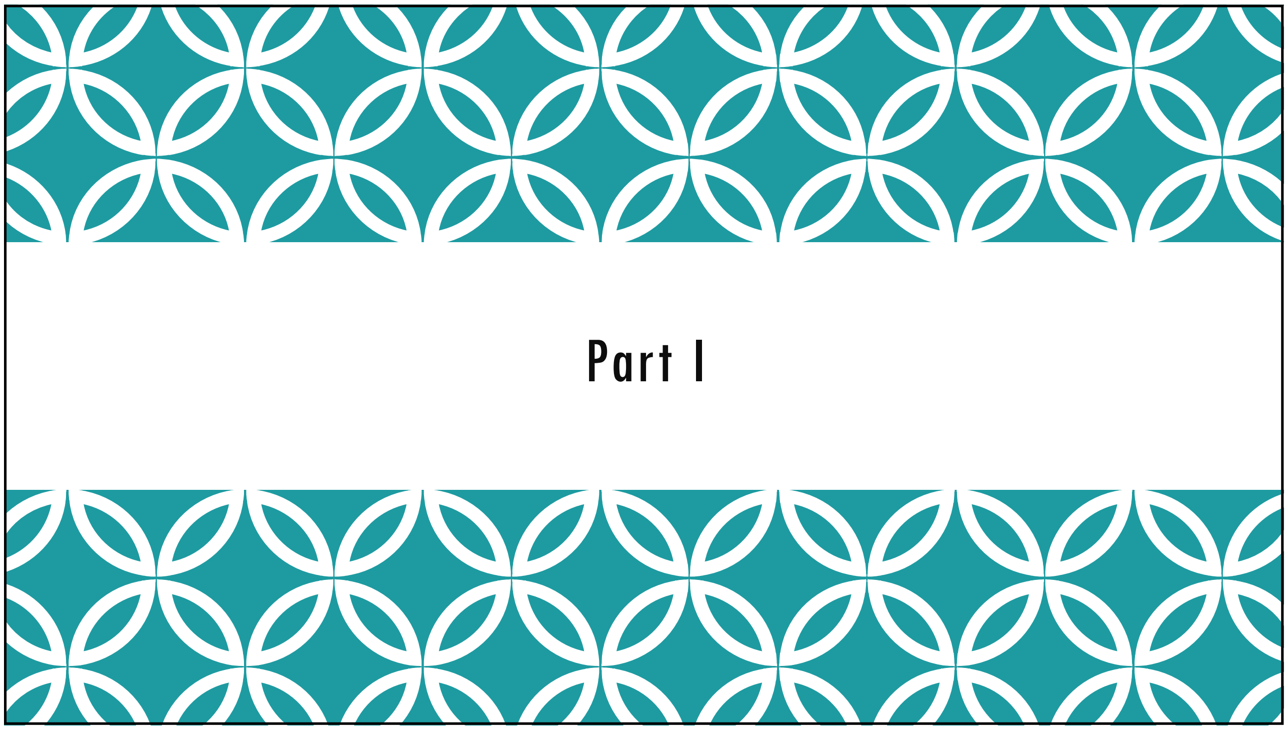


Introduction to R, RStudio, and R Markdown

Joseph Rudolf
API-R 2022

1



Part I

2

R

Programming language for data analysis

RStudio

Interactive development environment (IDE)

R Markdown

Computational document format

3

Getting Started with RStudio

4

RStudio: On the Web and In Your Home

RStudio Server
Hosted on a server
(in the cloud)

RStudio Desktop
Installed locally on
your computer

Note: Use Rstudio Server only for this course. Do not upload protected health information to the cloud!

5

5

Your Turn #1

Go to https://api-r.cloud in your browser and log in using the username and password provided in the course email.

Click "thumbs up" in zoom once you see the RStudio panes.

If you can't access the site click "thumbs down" and we will set you up in a backup configuration shortly.

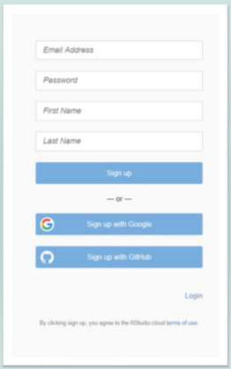
6

If You Can't Access api-r.cloud site

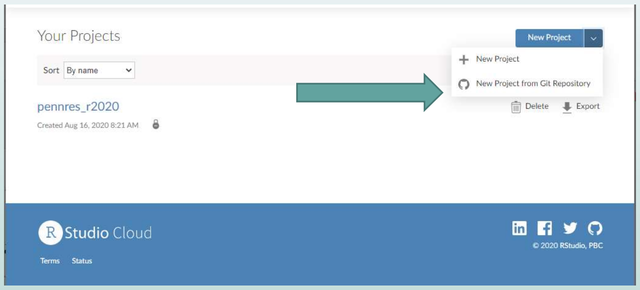
1.

rstudio.cloud

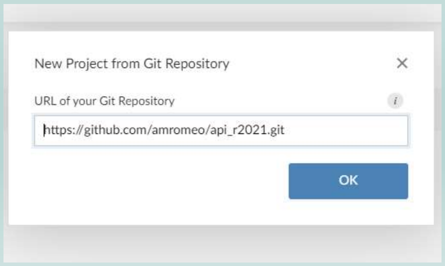
2.



3.



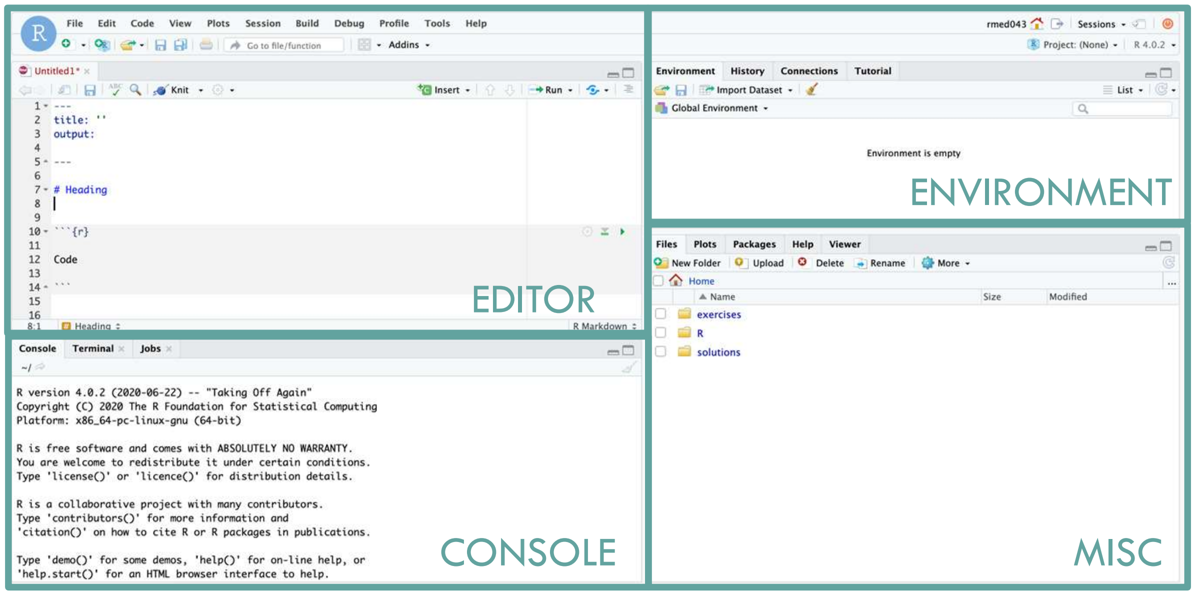
4.



URL of your Git Repository

https://github.com/amromeo/api_r2021.git

7



~~~
---
title: ''
output:

---

# Heading


```{r}

Code

```
~~~

```
~/
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 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.
```

8

8

Reproducible Data Analysis and R Markdown

9

The Duke Cancer Scandal

- Chemo sensitivity from microarrays
- Errors first, then cover-up
- Clinical trials based on flawed models
- Papers retracted, lawsuits settled

10

Duke	MD Anderson
"18881_at"	"18882_g_at"
"31321_at"	"31322_at"
"31725_s_at"	"31726_at"
"32307_r_at"	"32308_r_at"
...	

Off-by-one indexing error

11

"Common problems are simple…

Off-by-one **indexing error**

Sensitive / resistant **label reversal**

Confounding in experimental design

Inclusion of data from **non-reported sources**

Wrong figure shown

… and simple problems are common."

12

Point-and-click is not reproducible

Computer code can precisely document each step of the analysis

13

Why YOU should analyze your data reproducibly

"Can we redo the analysis with this month's data?"

"Why do the data in Table 1 not seem to agree with Figure 2?"

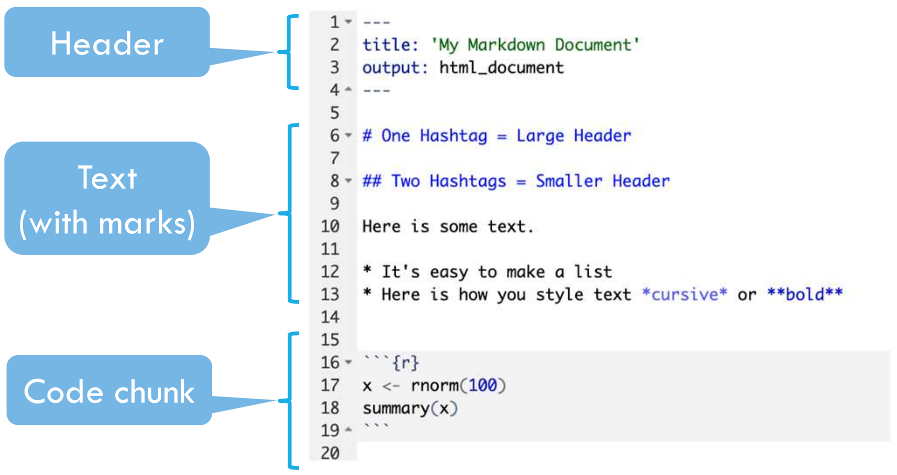
"Why did I decide to omit these six samples from my analysis?"

YOUR CLOSEST COLLABORATOR IS **YOU** FROM 6 MONTHS AGO

14

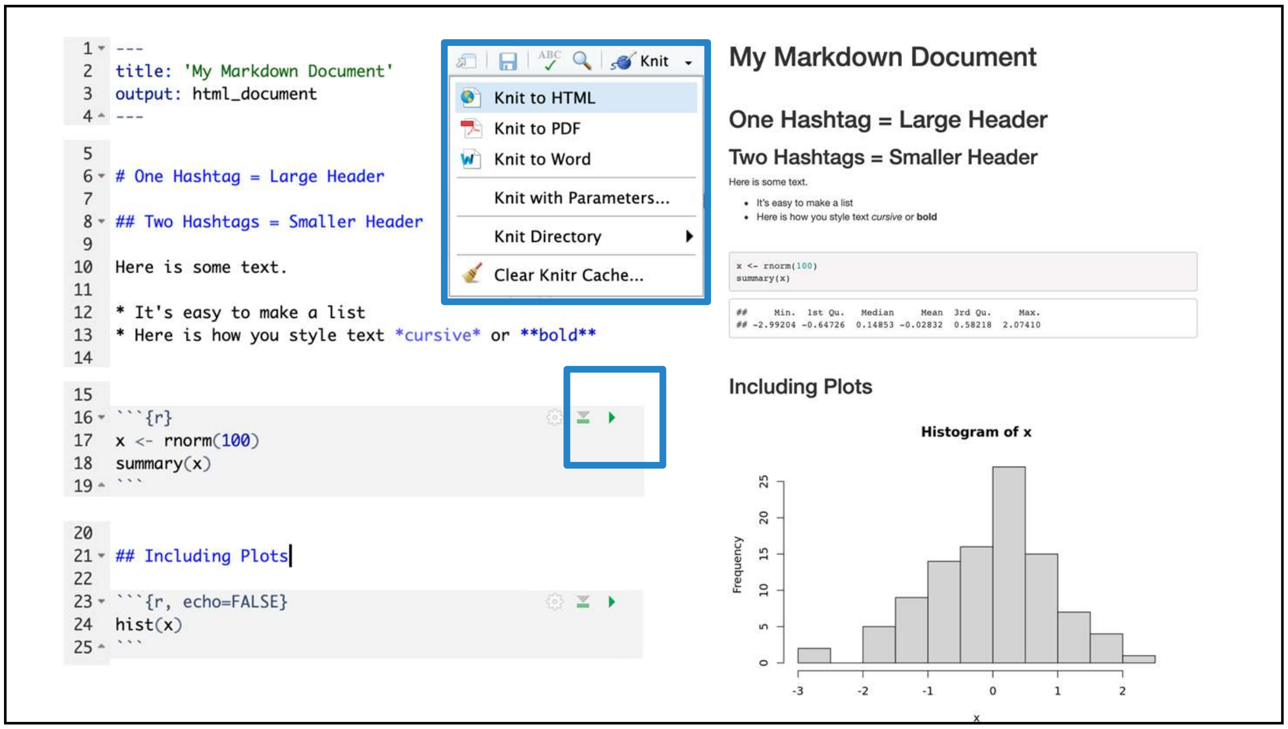
15

Anatomy of an R Markdown Document



~~~
---
title: 'My Markdown Document'
output: html_document
---

# One Hashtag = Large Header

## Two Hashtags = Smaller Header

Here is some text.

* It's easy to make a list
* Here is how you style text *cursive* or **bold**


```{r}
x <- rnorm(100)
summary(x)
```

~~~

16



~~~
---
title: 'My Markdown Document'
output: html_document
---

# One Hashtag = Large Header

## Two Hashtags = Smaller Header

Here is some text.

* It's easy to make a list
* Here is how you style text *cursive* or **bold**

```{r}
x <- rnorm(100)
summary(x)
```

## Including Plots

```{r, echo=FALSE}
hist(x)
```
~~~

My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

Here is some text.

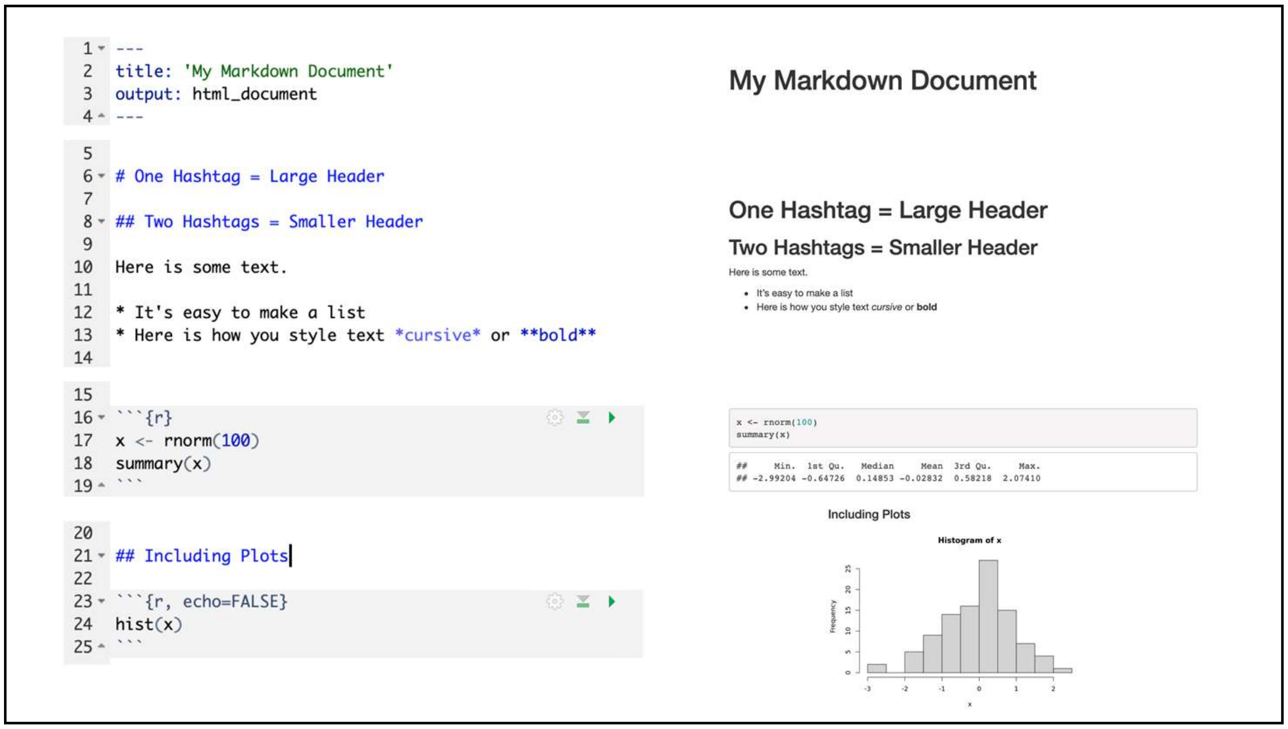
- It's easy to make a list
- Here is how you style text *cursive* or **bold**

```
x <- rnorm(100)
summary(x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.99204 -0.64726  0.14853 -0.02832  0.58218  2.07410
```

Including Plots

17



~~~
---
title: 'My Markdown Document'
output: html_document
---

# One Hashtag = Large Header

## Two Hashtags = Smaller Header

Here is some text.

* It's easy to make a list
* Here is how you style text *cursive* or **bold**

```{r}
x <- rnorm(100)
summary(x)
```

## Including Plots

```{r, echo=FALSE}
hist(x)
```
~~~

My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

Here is some text.

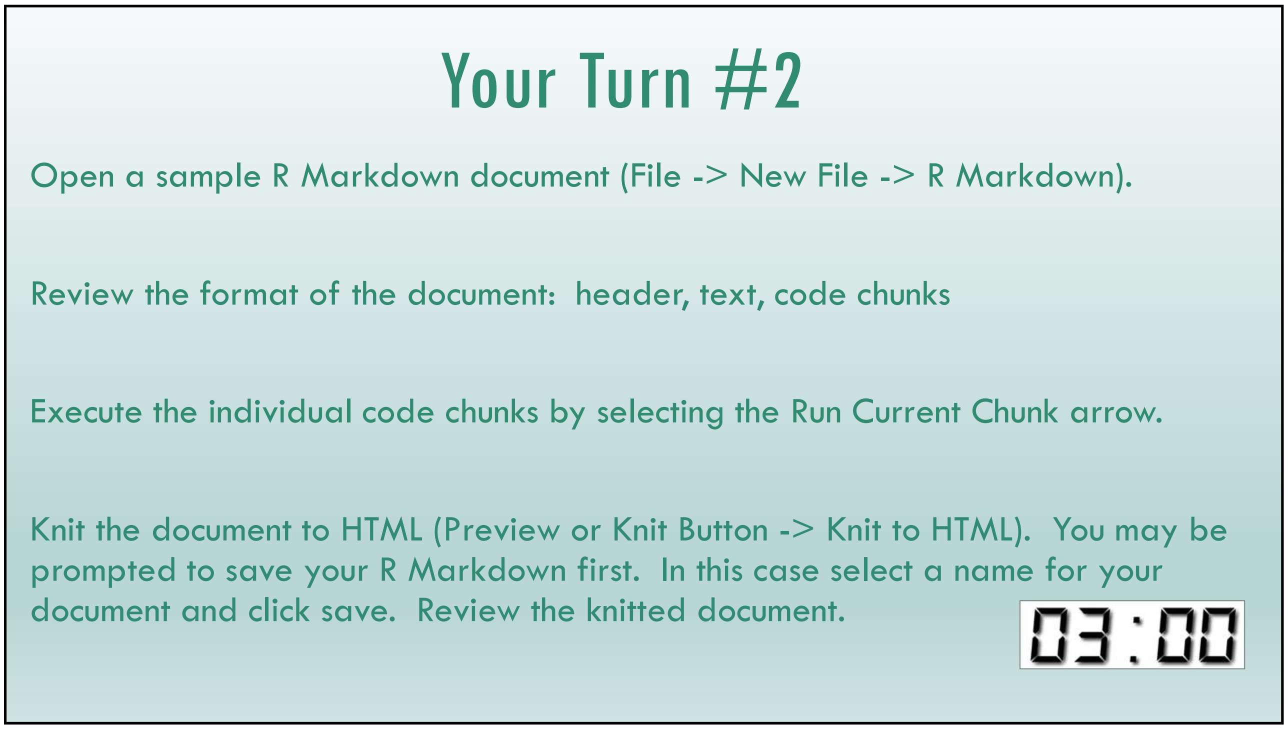
- It's easy to make a list
- Here is how you style text *cursive* or **bold**

```
x <- rnorm(100)
summary(x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.99204 -0.64726  0.14853 -0.02832  0.58218  2.07410
```

Including Plots

18



Your Turn #2

Open a sample R Markdown document (File -> New File -> R Markdown).

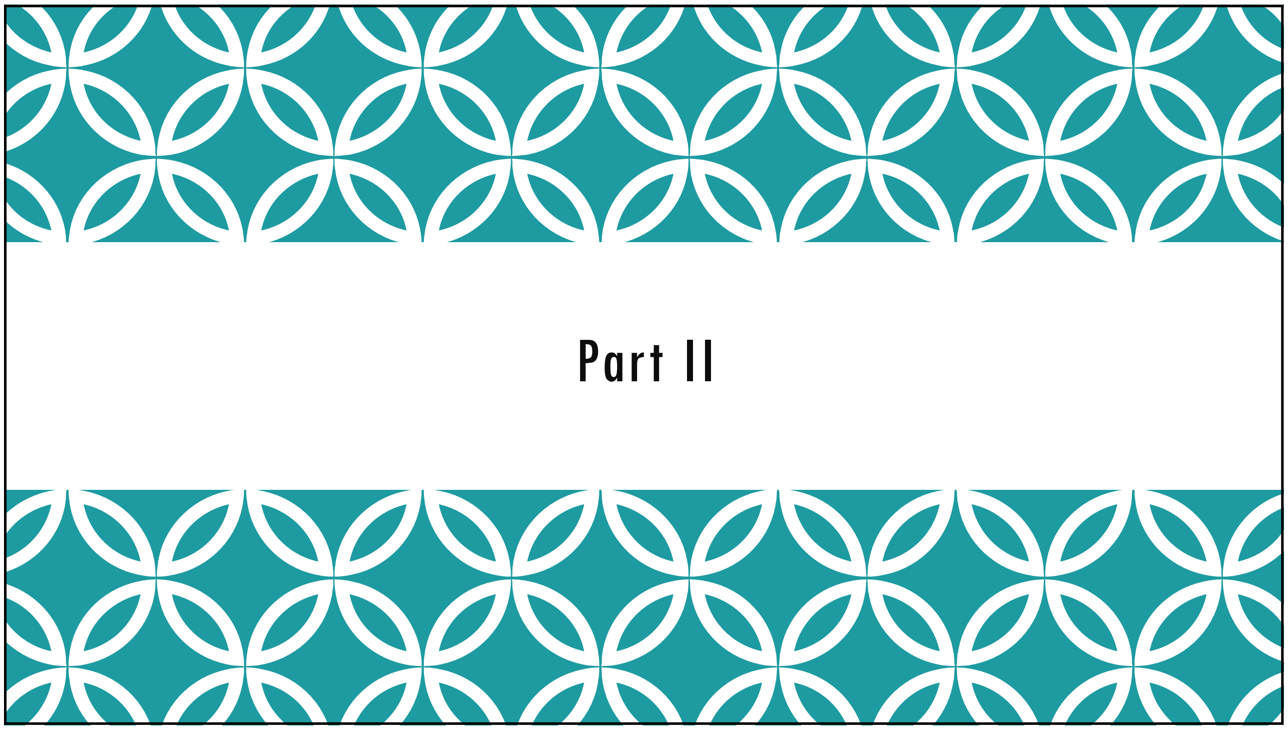
Review the format of the document: header, text, code chunks

Execute the individual code chunks by selecting the Run Current Chunk arrow.

Knit the document to HTML (Preview or Knit Button -> Knit to HTML). You may be prompted to save your R Markdown first. In this case select a name for your document and click save. Review the knitted document.

03:00

19



Part II

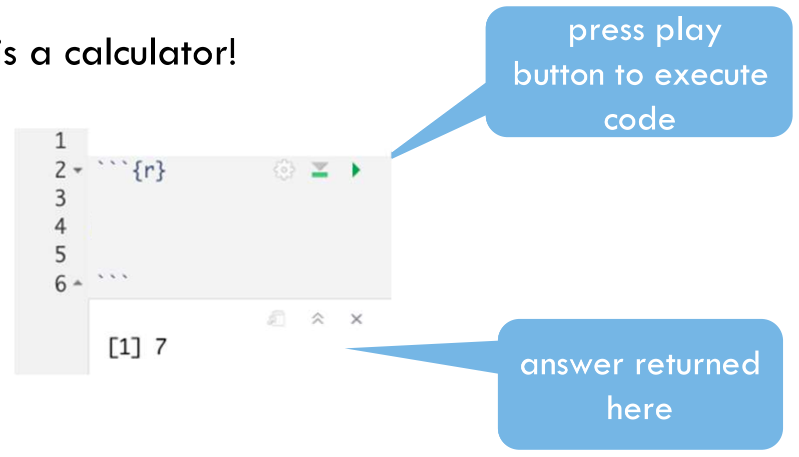
20

The Basics of Coding

21

The Basics of Coding: Calculation

- R is a calculator!

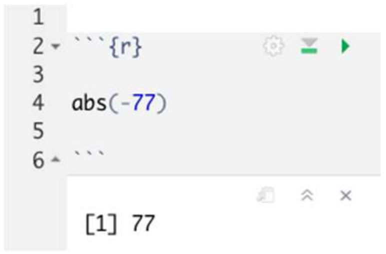


~~~
```{r}


```
[1] 7
~~~

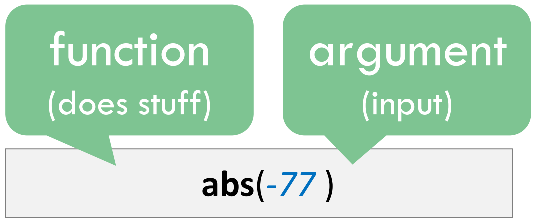
22

The Basics of Coding: Functions

- Code that extends our reach beyond the basic operators



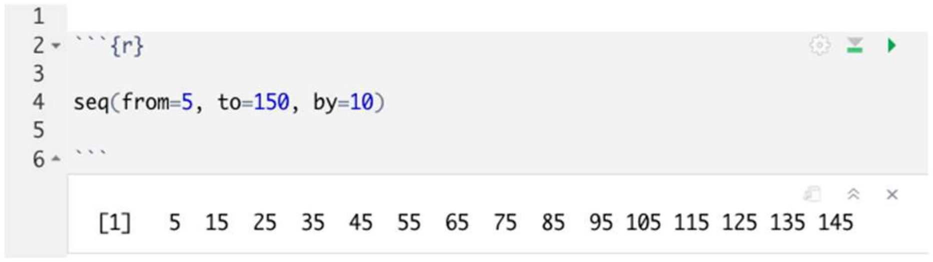
~~~
```{r}

abs(-77)

```

[1] 77
~~~



23

Putting Functions to Work

- We can use functions to do more than simple math, we can make things!
- We can create a series of integers (a vector) using the seq() function

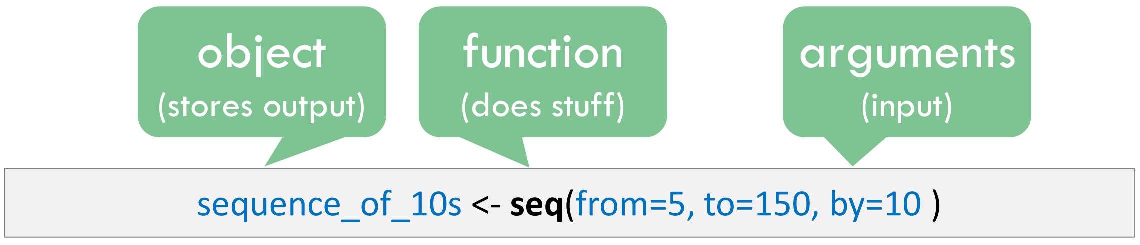


~~~
```{r}

seq(from=5, to=150, by=10)

```

[1]   5  15  25  35  45  55  65  75  85  95 105 115 125 135 145
~~~

24

The Basics of Coding: Objects

- Objects are the container for your output

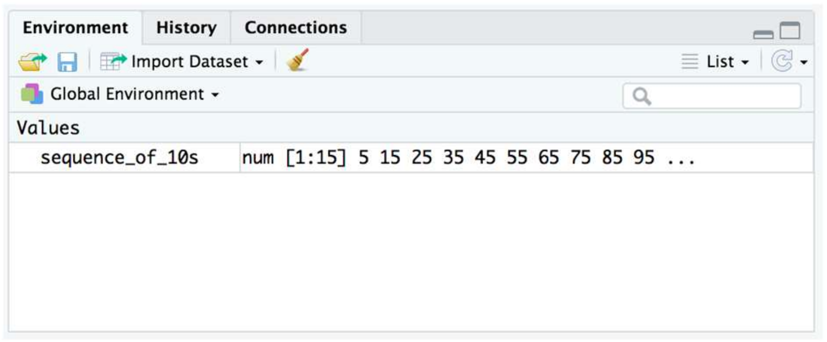


```
sequence_of_10s <- seq(from=5, to=150, by=10 )
```

25

Checking the contents of an object

- The environment tab shows us the objects we have created.

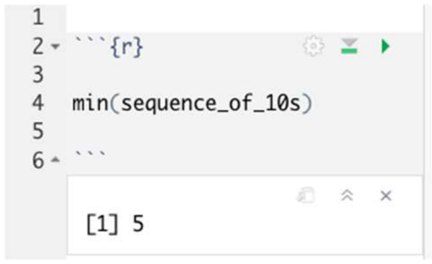


```
Values
  sequence_of_10s    num [1:15] 5 15 25 35 45 55 65 75 85 95 ...
```

26

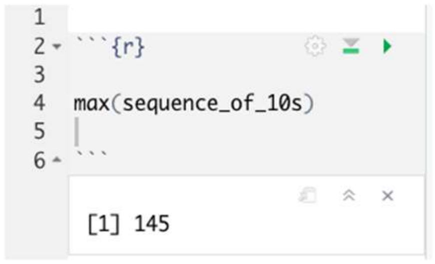
Bending objects to your will

- Once we have created an object we can start to interact with it.
- This includes passing our objects to other functions… Whoa!



~~~
```{r}

min(sequence_of_10s)

```
~~~

```
[1] 5
```



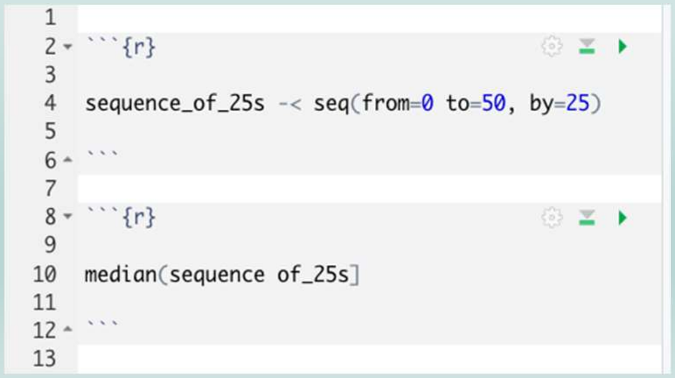
~~~
```{r}

max(sequence_of_10s)

```
~~~

```
[1] 145
```

27

Your Turn #3

I've written some code to create a sequence from 0 to 500 in increments of 25 called sequence_of_25s. Ultimately I want to calculate the median value of this sequence. Unfortunately I've made some mistakes in my code and I am hoping you can help me find them.



~~~
```{r}

sequence_of_25s -< seq(from=0 to=50, by=25)

~~~

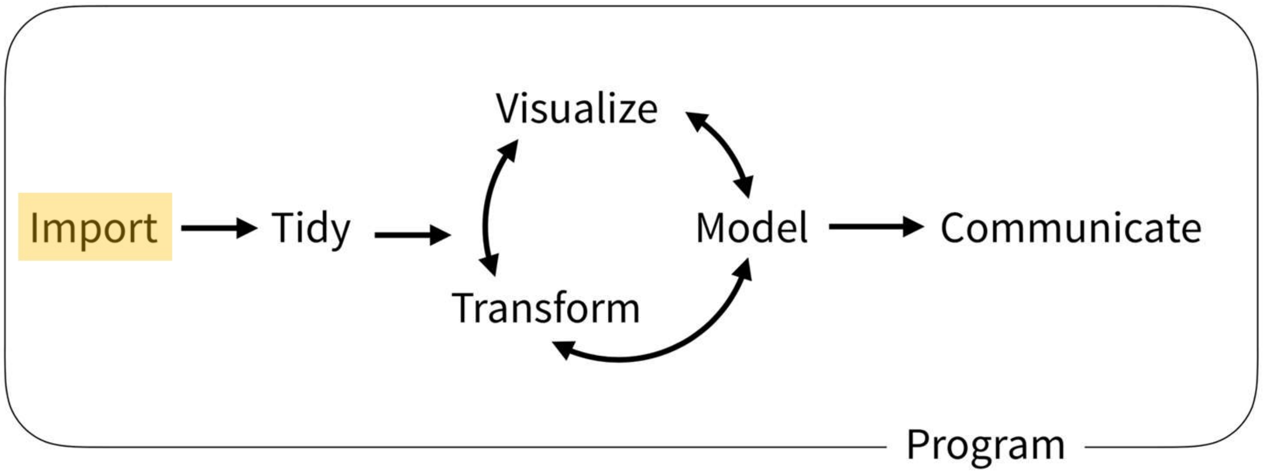
```{r}

median(sequence of_25s]

```

```

01:00

28
```

Importing Data

29

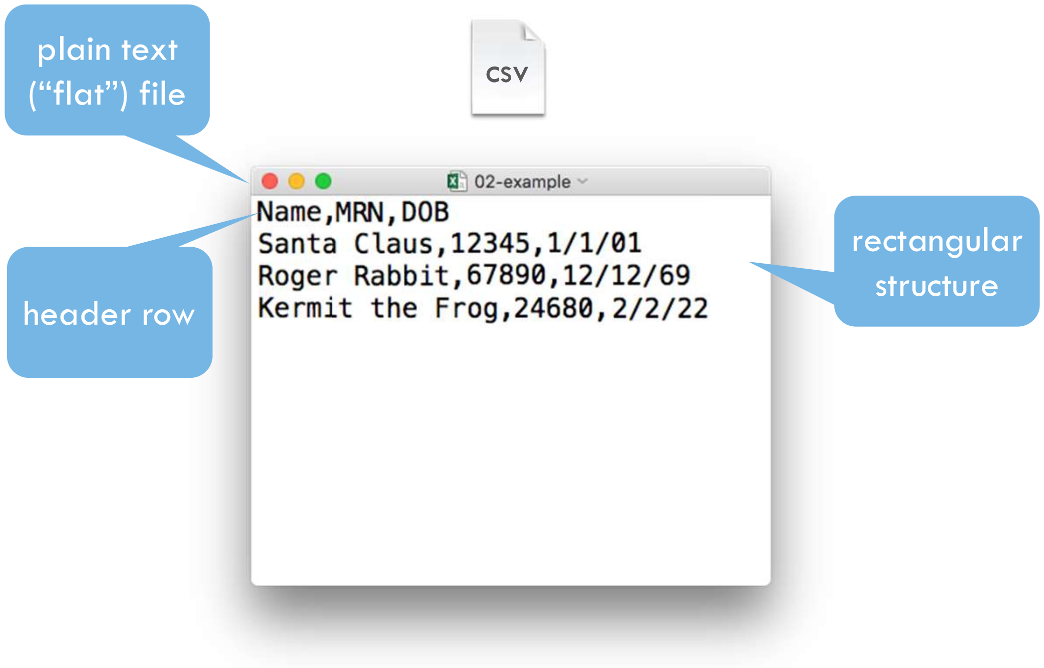
The Data Analysis Pipeline



From *R for Data Science* (https://r4ds.had.co.nz/introduction.html)

30

30



```
Name,MRN,DOB
Santa Claus,12345,1/1/01
Roger Rabbit,67890,12/12/69
Kermit the Frog,24680,2/2/22
```

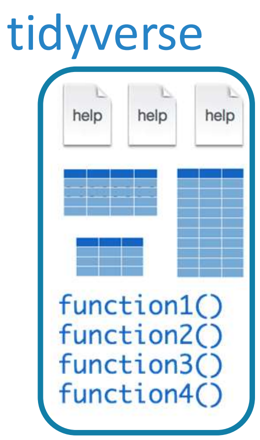
31

Tidyverse: R Packages for Data Science

- A consistent way to organize data
- Human readable, concise, consistent code
- Build pipelines from atomic data analysis steps

32

Installing and loading R packages



```
install.packages("tidyverse")
```

Downloads files to computer
1 x per computer

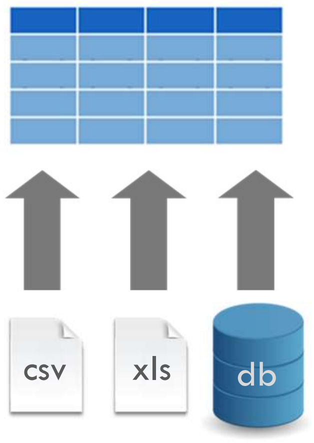
```
library("tidyverse")
```

Loads package
1 x per R Session

33

Dataframes: Beyond the Vector

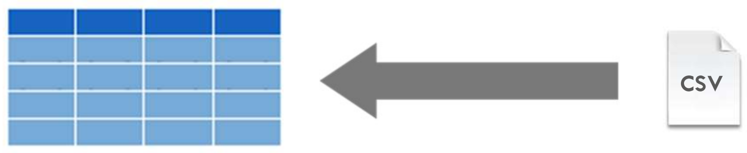
- Dataframe is the term for a table
- Dataframes are composed:
  Columns (Variables)
  Rows (Observations)
- Dataframes are objects and can be acted on like other objects



34

read_csv()

```
data_frame <- read_csv(file_name)
```

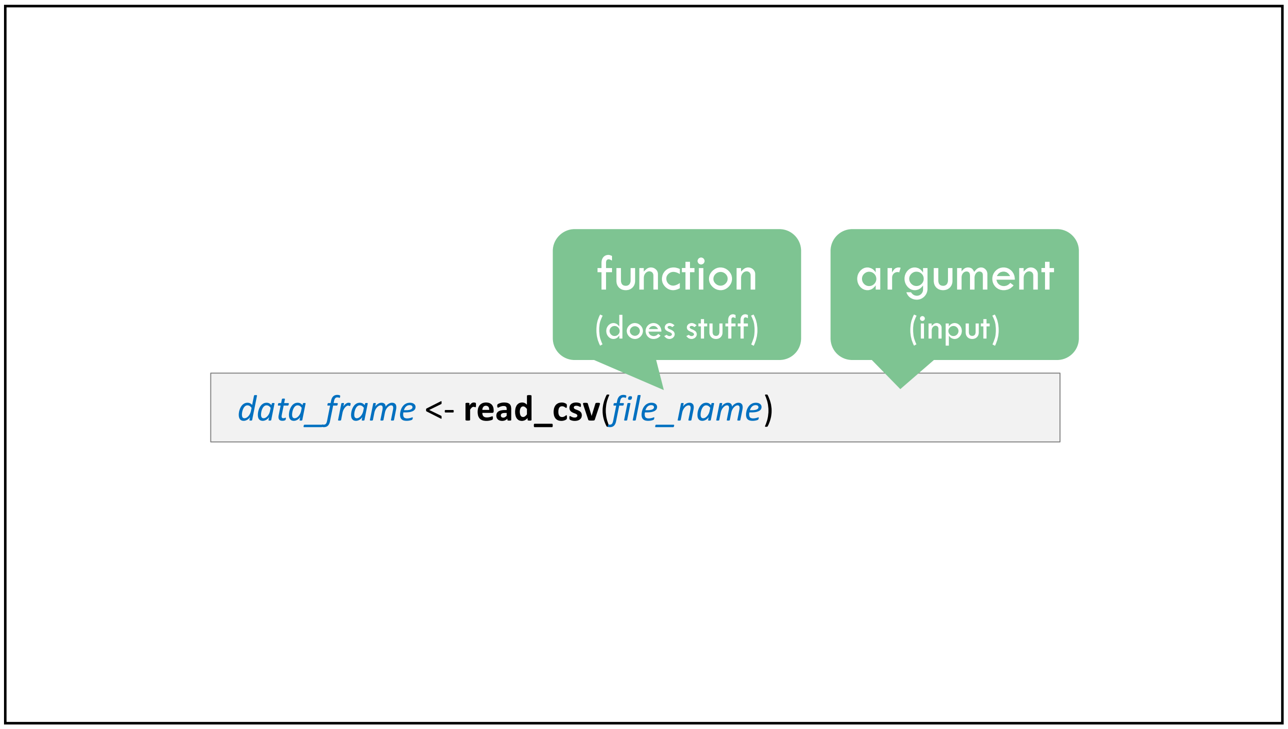


35

function
(does stuff)

```
data_frame <- read_csv(file_name)
```

36

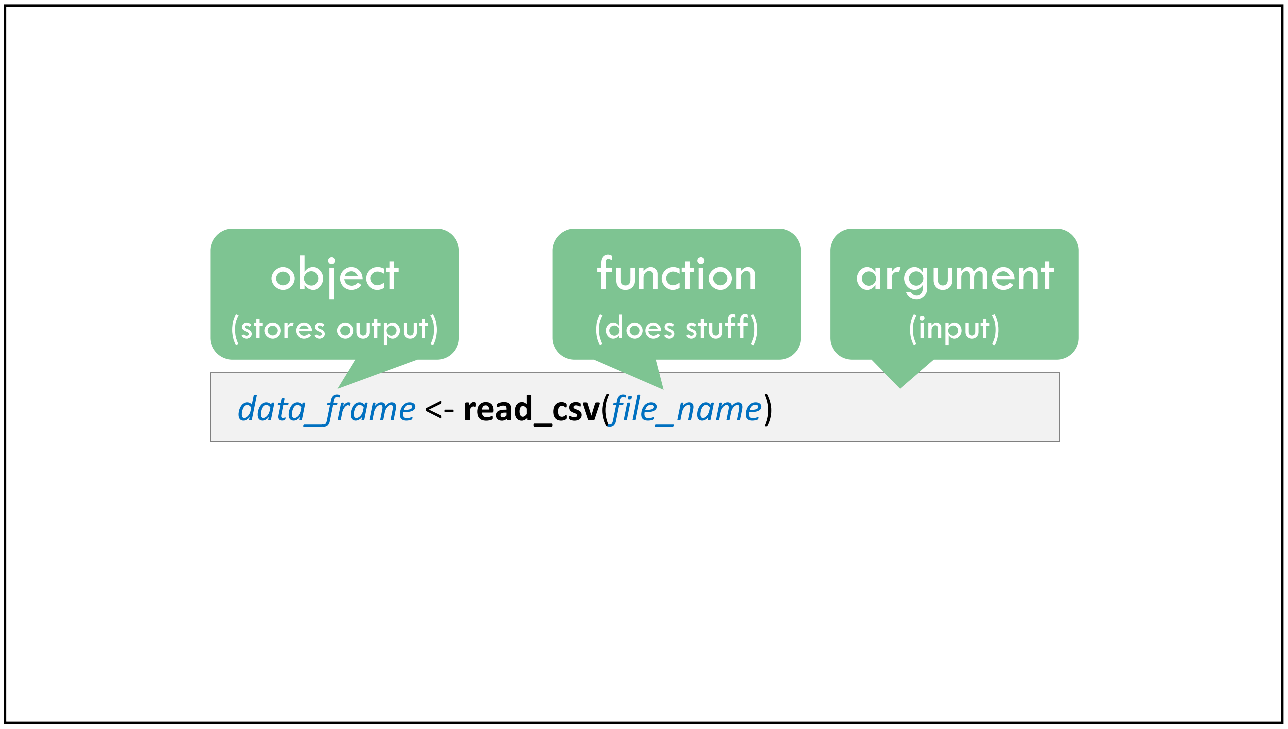


function
(does stuff)

argument
(input)

```
data_frame <- read_csv(file_name)
```

37



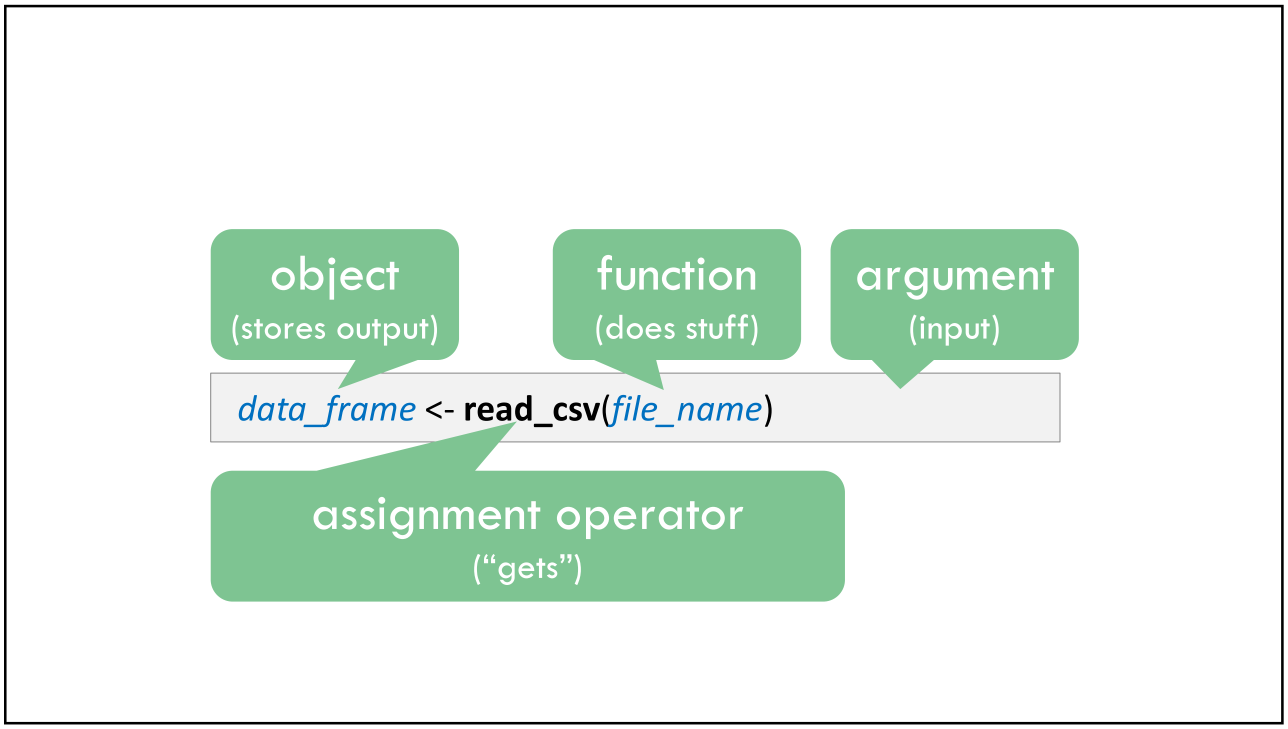
object
(stores output)

function
(does stuff)

argument
(input)

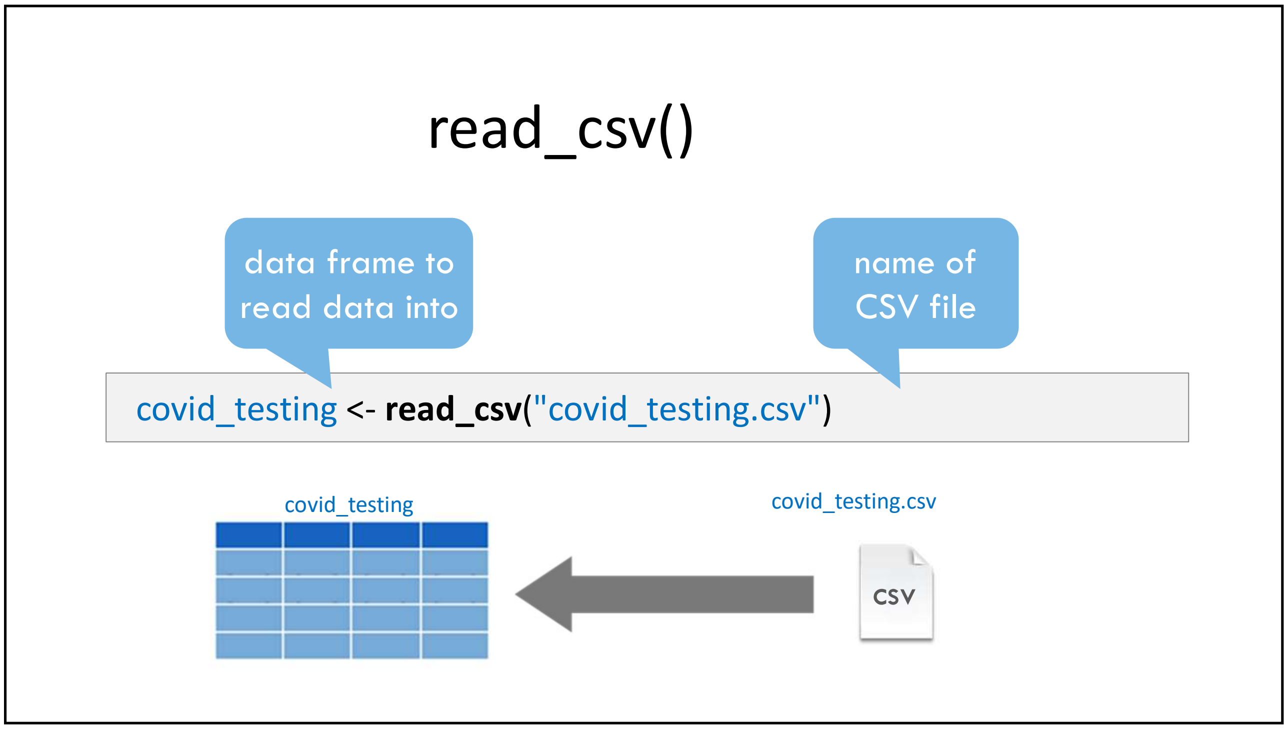
```
data_frame <- read_csv(file_name)
```

38



```
data_frame <- read_csv(file_name)
```

39



read_csv()

```
covid_testing <- read_csv("covid_testing.csv")
```

40

Your Turn #4

In the MISC pane, select the folder: “exercises”

Select the R Markdown file: “01 - Importing and Exploring Data.Rmd”

In the Editor pane, follow the instructions to complete the exercise.

05:00

41

Recap

Programming Language

IDE

Document Format

Packages extend the functionality of R. They need to be installed once per computer and loaded each session.

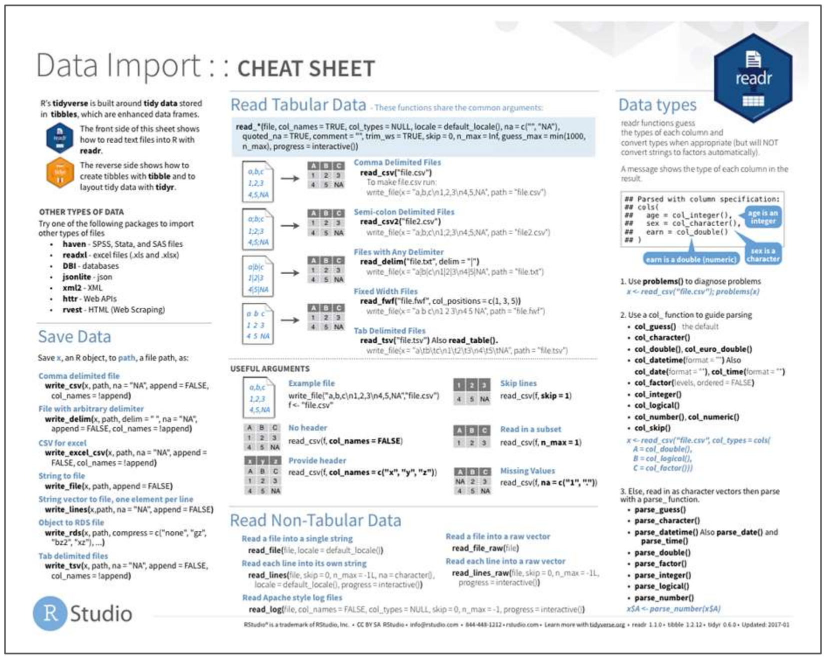
Functions do stuff. They accept **Arguments** to define parameters. We can store the output of functions in **Objects** using the assignment operator (<-).

Importing Data is the first step data analysis pipeline. read_csv() is a function from the tidyverse that we can use for importing data.

42

What else?

43



Data Import : : CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored in **tibbles**, which are enhanced data frames.

The front side of this sheet shows how to read text files into R with **readr**.

The reverse side shows how to create tibbles with **tibble** and to layout tidy data with **tidyr**.

OTHER TYPES OF DATA

Try one of the following packages to import other types of files

- **haven** - SPSS, Stata, and SAS files
- **readxl** - excel files (.xls and .xlsx)
- **DBI** - databases
- **jsonlite** - json
- **xml2** - XML
- **httr** - Web APIs
- **rvest** - HTML (Web Scraping)

Save Data

Save x, an R object, to path, a file path, as:

Comma delimited file
write_csv(x, path, na = "NA", append = FALSE, col_names = !append)

File with arbitrary delimiter
write_delim(x, path, delim = " ", na = "NA", append = FALSE, col_names = !append)

CSV for excel
write_excel_csv(x, path, na = "NA", append = FALSE, col_names = !append)

String to file
write_file(x, path, append = FALSE)

String vector to file, one element per line
write_lines(x,path, na = "NA", append = FALSE)

Object to RDS file
write_rds(x, path, compress = c("none", "gz", "bz2", "xz"), ...)

Tab delimited files
write_tsv(x, path, na = "NA", append = FALSE, col_names = !append)

Read Tabular Data

- These functions share the common arguments:

read_*(file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())

Comma Delimited Files
read_csv("file.csv")
To make file.csv run:
write_file(x = "a,b,c\n1,2,3\n4,5,NA", path = "file.csv")

Semi-colon Delimited Files
read_csv2("file2.csv")
write_file(x = "a;b;c\n1;2;3\n4;5;NA", path = "file2.csv")

Files with Any Delimiter
read_delim("file.txt", delim = "|")
write_file(x = "a|b|c\n1|2|3\n4|5|NA", path = "file.txt")

Fixed Width Files
read_fwf("file.fwf", col_positions = c(1, 3, 5))
write_file(x = "a b c\n1 2 3\n4 5 NA", path = "file.fwf")

Tab Delimited Files
read_tsv("file.tsv") Also **read_table()**.
write_file(x = "a\tb\tc\n1\t2\t3\n4\t5\tNA", path = "file.tsv")

USEFUL ARGUMENTS

Example file
write_file("a,b,c\n1,2,3\n4,5,NA","file.csv")
f <- "file.csv"

Skip lines
read_csv(f, **skip = 1**)

No header
read_csv(f, **col_names = FALSE**)

Read in a subset
read_csv(f, **n_max = 1**)

Provide header
read_csv(f, **col_names = c("x", "y", "z")**)

Missing Values
read_csv(f, **na = c("1", ".")**)

Read Non-Tabular Data

Read a file into a single string
read_file(file, locale = default_locale())

Read each line into its own string
read_lines(file, skip = 0, n_max = -1L, na = character(), locale = default_locale(), progress = interactive())

Read Apache style log files
read_log(file, col_names = FALSE, col_types = NULL, skip = 0, n_max = -1, progress = interactive())

Read a file into a raw vector
read_file_raw(file)

Read each line into a raw vector
read_lines_raw(file, skip = 0, n_max = -1L, progress = interactive())

Data types

readr functions guess the types of each column and convert types when appropriate (but will NOT convert strings to factors automatically).

A message shows the type of each column in the result.

```
## Parsed with column specification:
## cols(
##   age = col_integer(),
##   sex = col_character(),
##   earn = col_double()
## )
```

1. Use **problems()** to diagnose problems
   x <- read_csv("file.csv"); problems(x)

2. Use a col_ function to guide parsing
   - **col_guess()** the default
   - **col_character()**
   - **col_double()**, **col_euro_double()**
   - **col_datetime(**format = ""**)** Also **col_date(**format = ""**)**, **col_time(**format = ""**)**
   - **col_factor(**levels, ordered = FALSE**)**
   - **col_integer()**
   - **col_logical()**
   - **col_number()**, **col_numeric()**
   - **col_skip()**

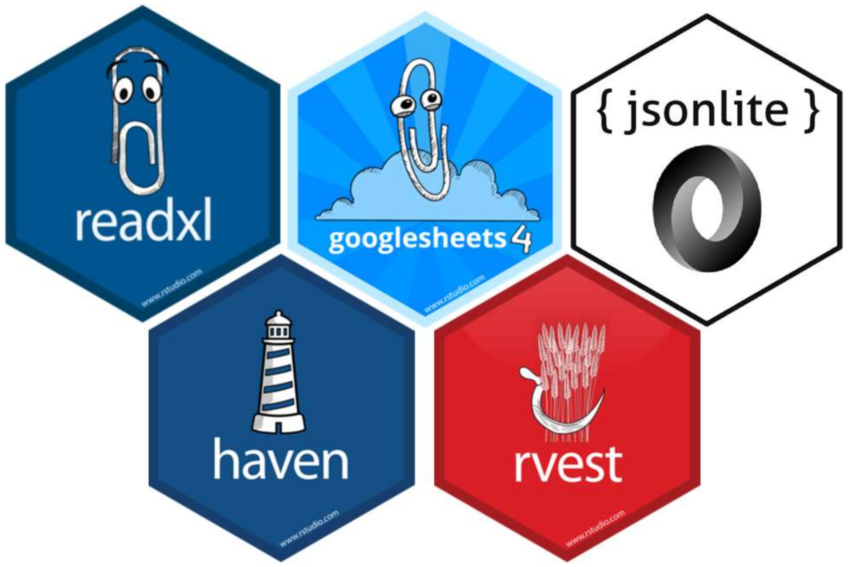
   *x <- read_csv("file.csv", col_types = cols(
   A = col_double(),
   B = col_logical(),
   C = col_factor()))*

3. Else, read in as character vectors then parse with a parse_ function.
   - **parse_guess()**
   - **parse_character()**
   - **parse_datetime()** Also **parse_date()** and **parse_time()**
   - **parse_double()**
   - **parse_factor()**
   - **parse_integer()**
   - **parse_logical()**
   - **parse_number()**

   x$A <- parse_number(x$A)

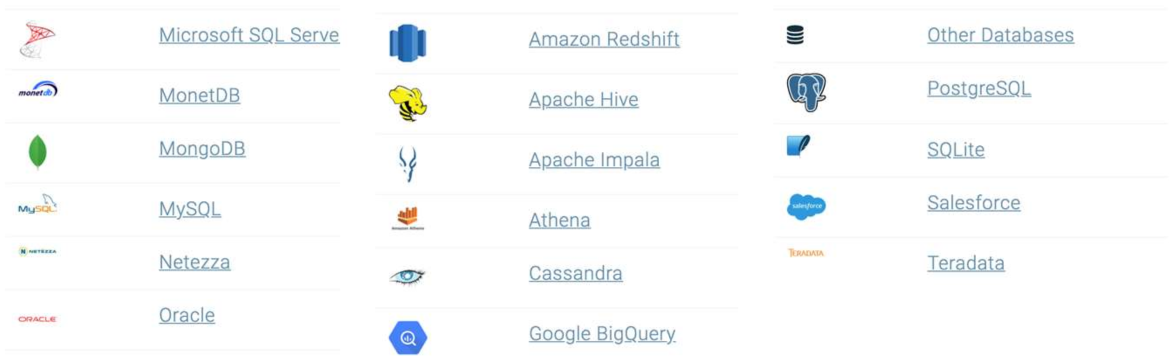
RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with tidyverse.org • readr 1.1.0 • tibble 1.2.12 • tidyr 0.6.0 • Updated: 2017-01

44



45

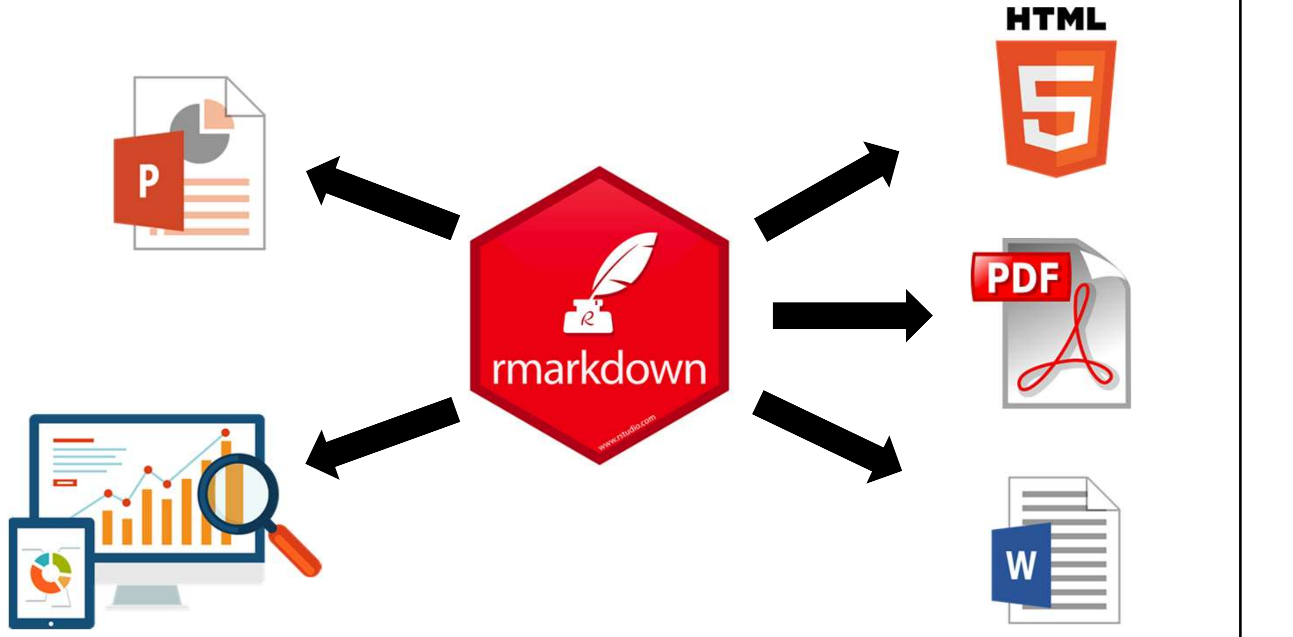
Databases



- Microsoft SQL Serve
- MonetDB
- MongoDB
- MySQL
- Netezza
- Oracle
- Amazon Redshift
- Apache Hive
- Apache Impala
- Athena
- Cassandra
- Google BigQuery
- Other Databases
- PostgreSQL
- SQLite
- Salesforce
- Teradata

https://db.rstudio.com/databases/

46



47

R Interface to Python

~~~
```{python}
import pandas
covid_testing.info()
```
~~~

48