



Introduction to R, RStudio, and R Markdown

Joseph Rudolf
API-R 2021

July 15, 2021	Session	Instructor
1:00 pm - 1:30 pm	Instructor Introductions, Introduction to technology	Amrom Obstfeld
1:35 pm - 2:05 pm	Introduction to R and RStudio	Joe Rudolf
2:20 pm - 3:15 pm	Reproducible Reporting	Joe Rudolf
3:30 pm - 5:00 pm	Data Visualization	Stephan Kadauke
July 16, 2021		
1:00 pm - 2:30 pm	Data Transformation	Amrom Obstfeld
2:45 pm - 4:15 pm	Statistical Analysis	Dan Herman
4:30 pm - 5:00 pm	Advanced Reporting	Patrick Mathias



Part I





R

Programming
language for
data analysis



RStudio

Interactive
development
environment (IDE)



R Markdown

Computational
document format



Getting Started with RStudio

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!

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.

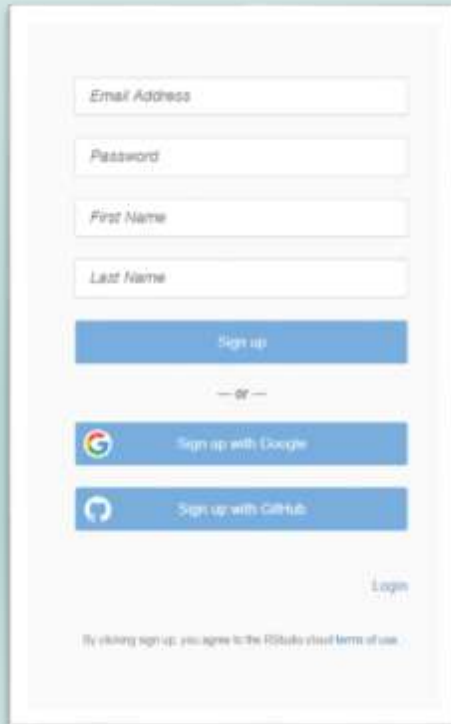
If you can't access the site click “thumbs down” and we will set you up in a backup configuration shortly.

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

1.

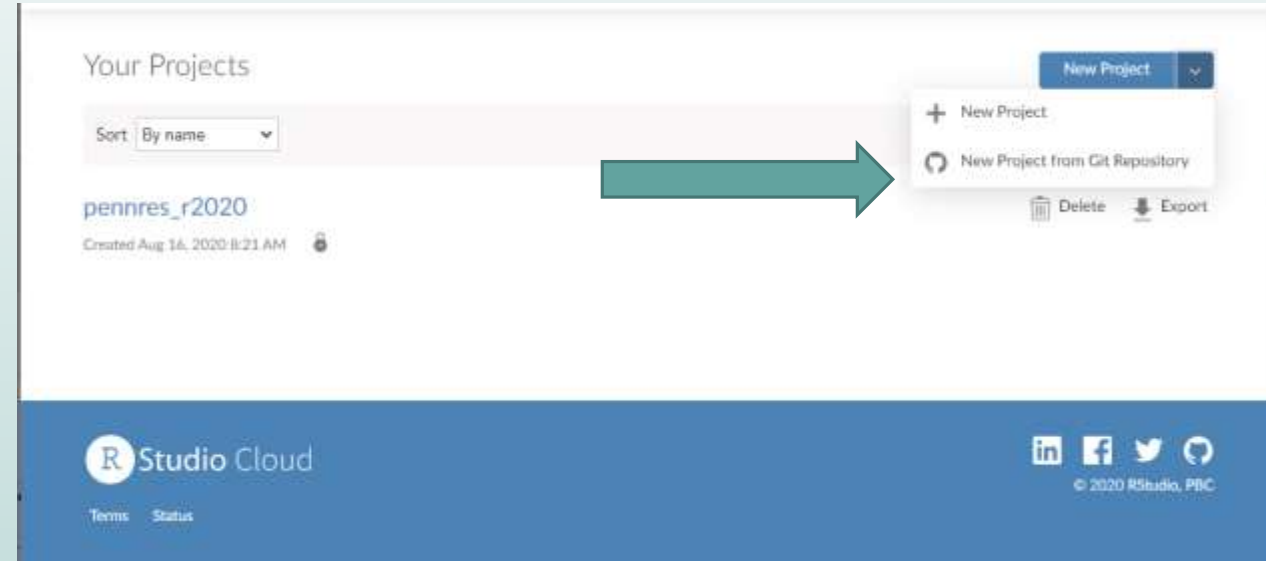
rstudio.cloud

2.

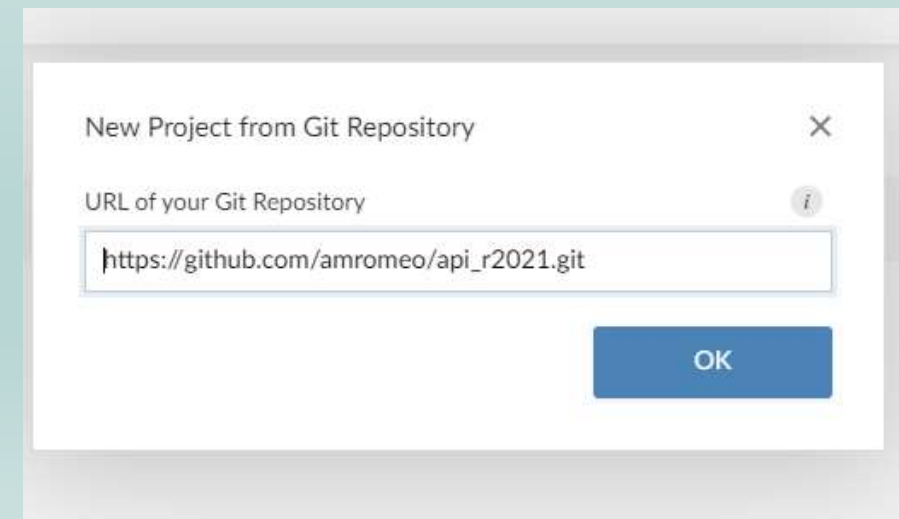


A sign-up form for RStudio Cloud. It contains four input fields: 'Email Address', 'Password', 'First Name', and 'Last Name'. Below these is a blue 'Sign up' button. Underneath the button is a separator line with '— or —' in the middle. Below the separator are two buttons: 'Sign up with Google' (with the Google logo) and 'Sign up with GitHub' (with the GitHub logo). At the bottom right is a 'Login' link. At the very bottom, there is a small line of text: 'By clicking sign up, you agree to the RStudio Cloud terms of use.'

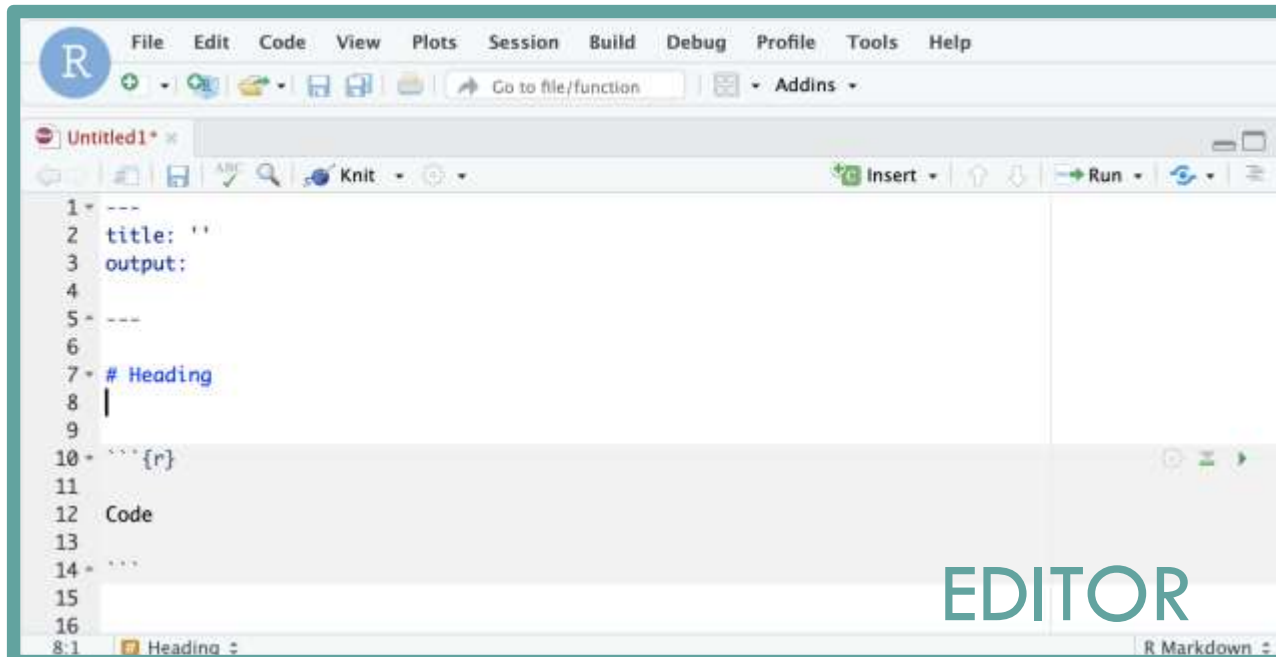
3.



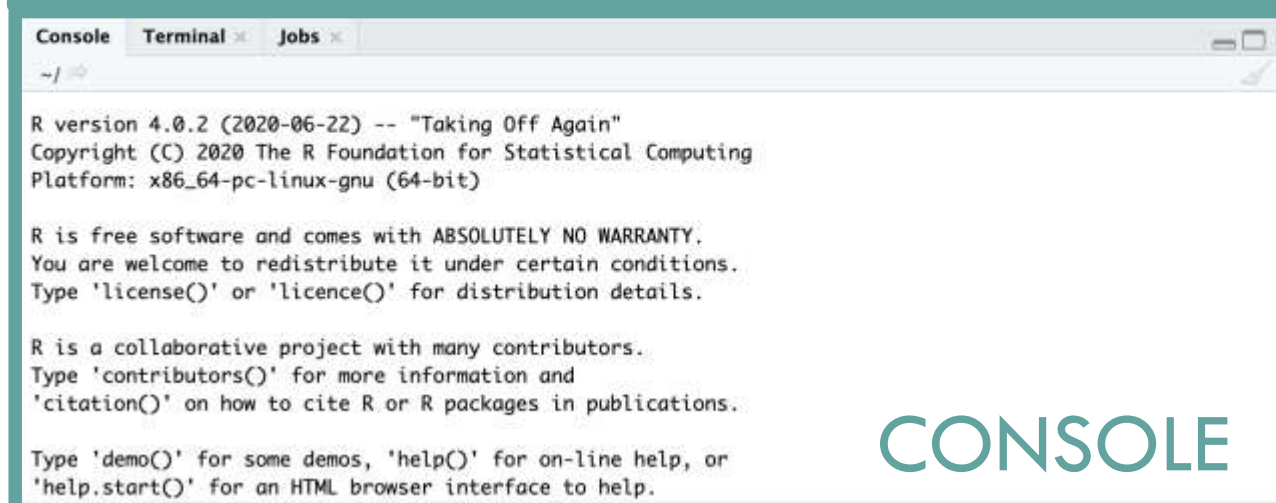
4.



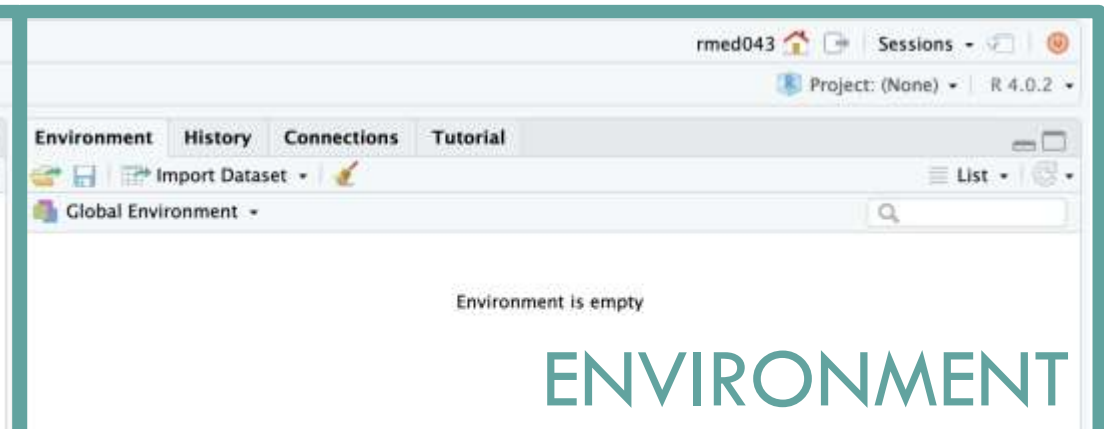
A dialog box titled 'New Project from Git Repository'. It has a close button (X) in the top right corner. Below the title is a label 'URL of your Git Repository' with an information icon (i) to its right. Below the label is a text input field containing the URL 'https://github.com/amromeo/api_r2021.git'. At the bottom right of the dialog is a blue 'OK' button.



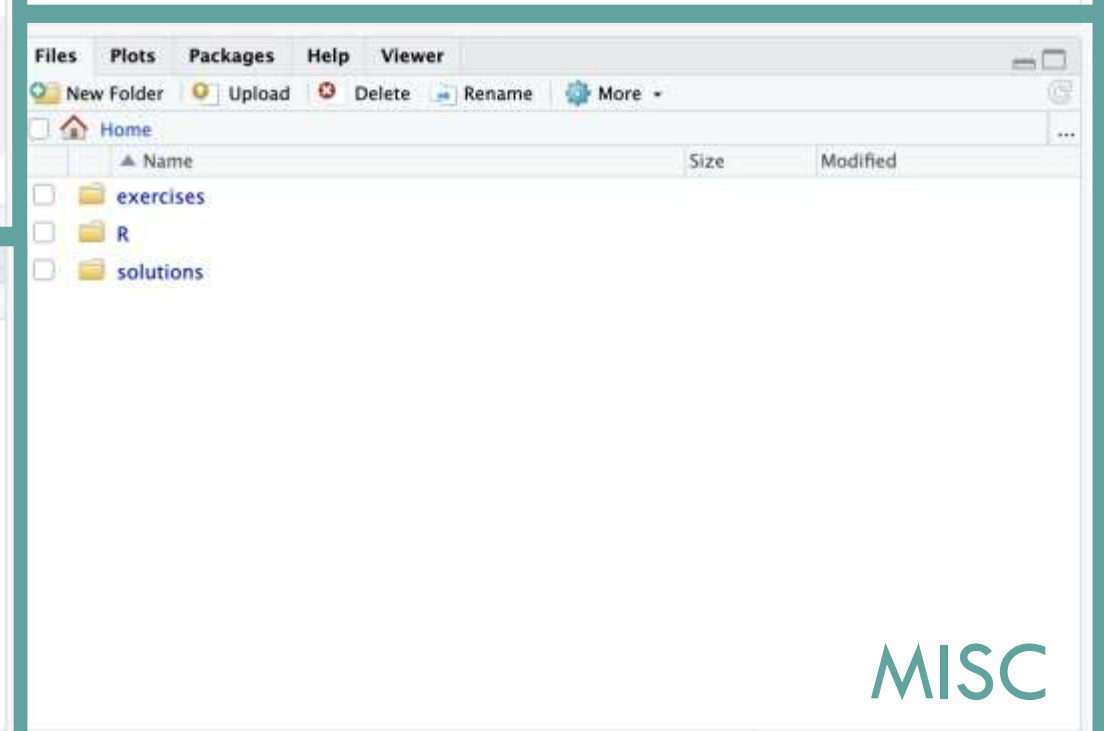
EDITOR



CONSOLE



ENVIRONMENT



MISC



Reproducible Data Analysis and R Markdown

The Duke Cancer Scandal

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



Duke

MD Anderson

"1881_at"

"1882_g_at"

"31321_at"

"31322_at"

"31725_s_at"

"31726_at"

"32307_r_at"

"32308_r_at"

...

Off-by-one indexing error

“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.”

Point-and-click is not reproducible



Computer code can precisely document each step of the analysis

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



Anatomy of an R Markdown Document

Header

Text
(with marks)

Code chunk

```
1 ---  
2 title: 'My Markdown Document'  
3 output: html_document  
4 ---  
5  
6 # One Hashtag = Large Header  
7  
8 ## Two Hashtags = Smaller Header  
9  
10 Here is some text.  
11  
12 * It's easy to make a list  
13 * Here is how you style text cursive or bold  
14  
15  
16 ```{r}  
17 x <- rnorm(100)  
18 summary(x)  
19 ```  
20
```

```
1 ---
2 title: 'My Markdown Document'
3 output: html_document
4 ---
```

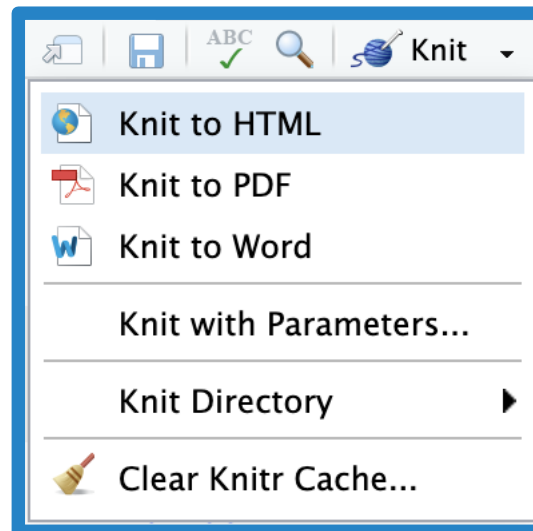
```
5
6 # One Hashtag = Large Header
7
8 ## Two Hashtags = Smaller Header
```

```
9
10 Here is some text.
```

```
11
12 * It's easy to make a list
13 * Here is how you style text *cursive* or **bold**
14
```

```
15
16 ```{r}
17 x <- rnorm(100)
18 summary(x)
19 ```
```

```
20
21 ## Including Plots
22
23 ```{r, echo=FALSE}
24 hist(x)
25 ```
```



My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

Here is some text.

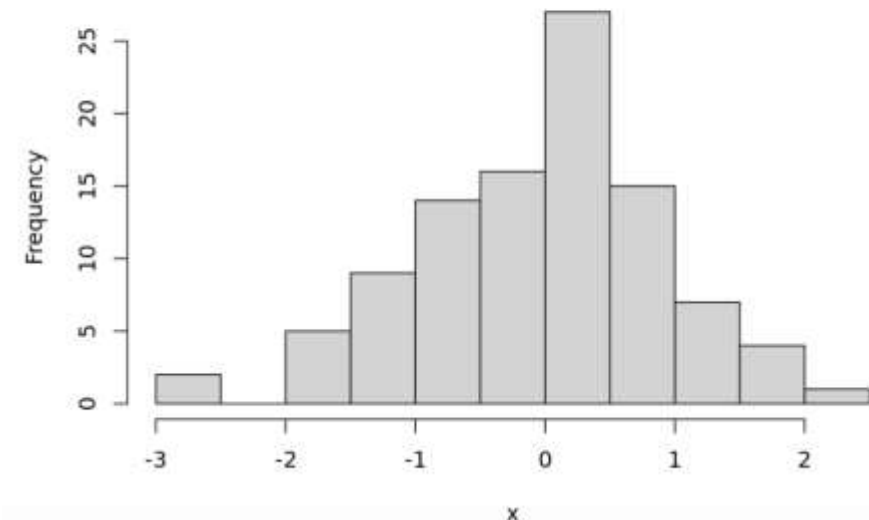
- It's easy to make a list
- Here is how you style text *cursive* or **bold**

```
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

Histogram of x



```
1 ---
2 title: 'My Markdown Document'
3 output: html_document
4 ---
```

```
5
6 # One Hashtag = Large Header
7
8 ## Two Hashtags = Smaller Header
9
```

```
10 Here is some text.
```

```
11
12 * It's easy to make a list
13 * Here is how you style text *cursive* or **bold**
14
```

```
15
16 ```{r}
17 x <- rnorm(100)
18 summary(x)
19 ```
```

```
20
21 ## Including Plots
22
23 ```{r, echo=FALSE}
24 hist(x)
25 ```
```

My Markdown Document

One Hashtag = Large Header

Two Hashtags = Smaller Header

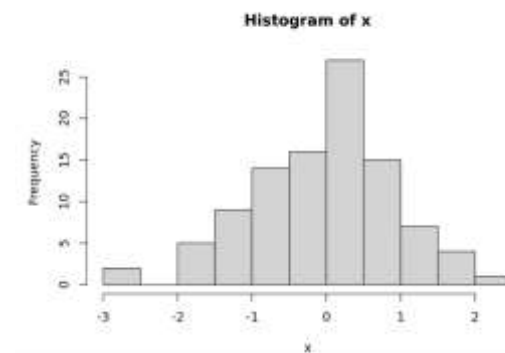
Here is some text.

- 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



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.

A digital timer with a black border and a light blue background, displaying the time 03:00 in a black, segmented digital font.



Part II





The Basics of Coding

The Basics of Coding: Calculation

- R is a calculator!



A screenshot of an R console window. The console has a light gray background with line numbers 1 through 6 on the left. Line 2 contains the code `{r}`. To the right of the code are three icons: a gear, a green minus sign, and a green play button. Below the code, the output `[1] 7` is displayed. At the bottom of the console window are three small icons: a trash can, an up arrow, and a close button (X).

press play
button to execute
code

answer returned
here

The Basics of Coding: Functions

- Code that extends our reach beyond the basic operators

```
1  
2  ```{r}  
3  
4  abs(-77)  
5  
6  ```  
[1] 77
```

function
(does stuff)

argument
(input)

abs(-77)

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

```
1  
2 ``{r}  
3  
4 seq(from=5, to=150, by=10)  
5  
6 ```
```

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

The Basics of Coding: Objects

- Objects are the container for your output

object
(stores output)

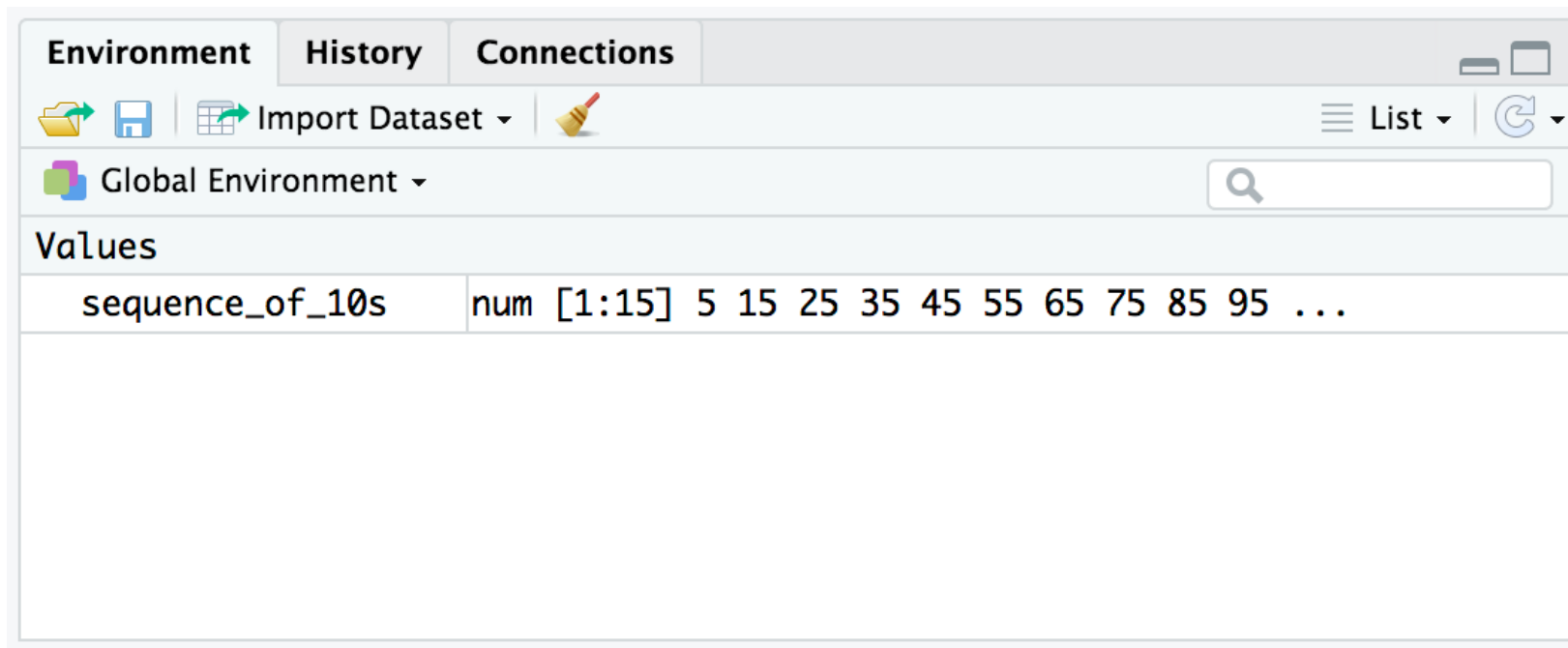
function
(does stuff)

arguments
(input)

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

Checking the contents of an object

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



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!

```
1  
2  ```{r}  
3  
4  min(sequence_of_10s)  
5  
6  ```
```

[1] 5

```
1  
2  ```{r}  
3  
4  max(sequence_of_10s)  
5  
6  ```
```

[1] 145

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.

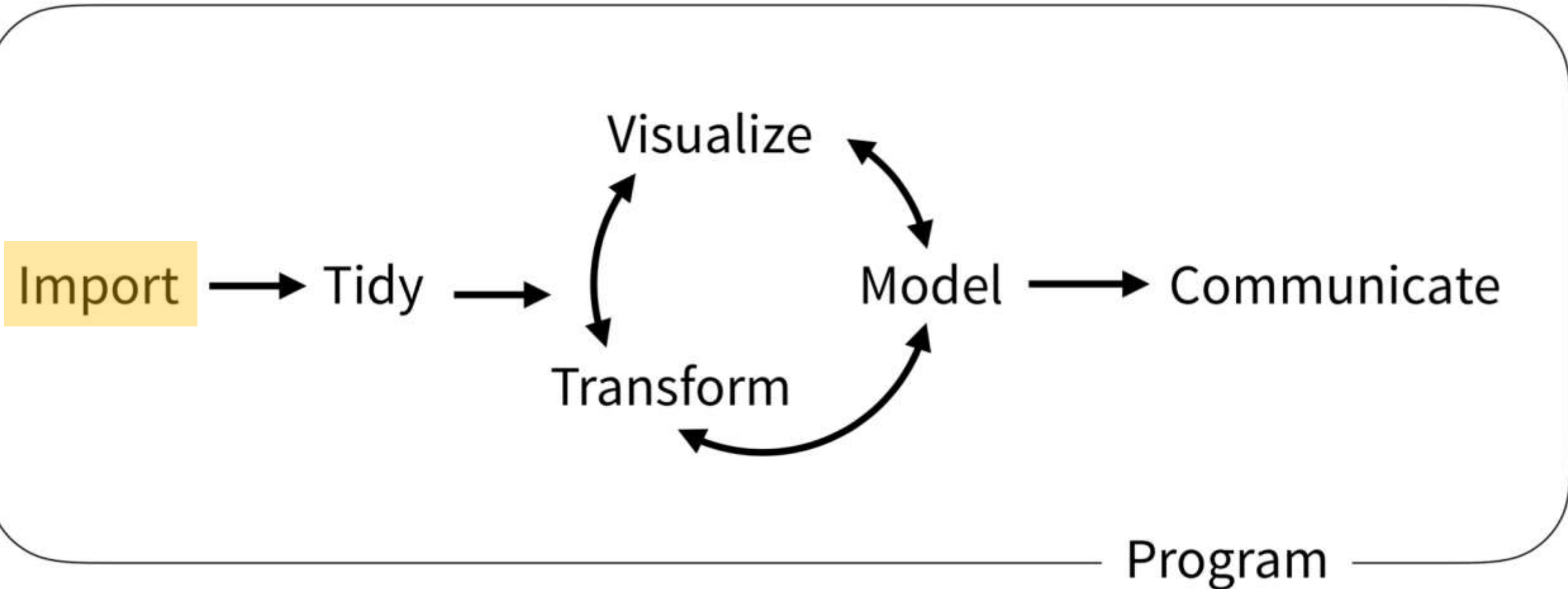
```
1  
2  ```{r}  
3  
4  sequence_of_25s <- seq(from=0 to=50, by=25)  
5  
6  ```  
7  
8  ```{r}  
9  
10 median(sequence of_25s]  
11  
12  ```  
13
```

01:00



Importing Data

The Data Analysis Pipeline



plain text
("flat") file



header row

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

rectangular
structure

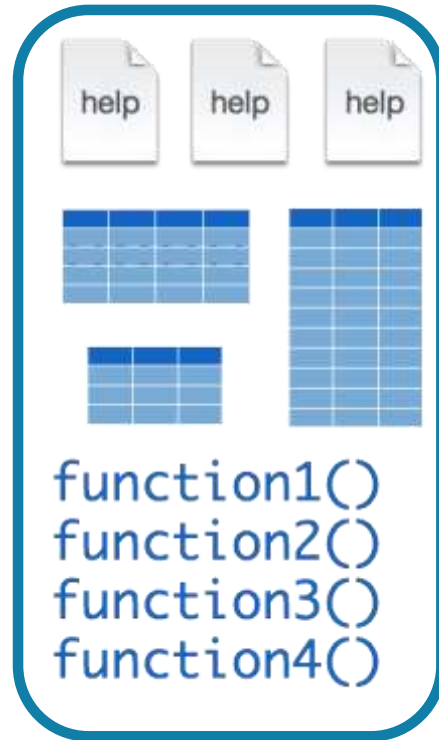
Tidyverse: R Packages for Data Science

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



Installing and loading R packages

tidyverse



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

Downloads files to computer

1 x per computer

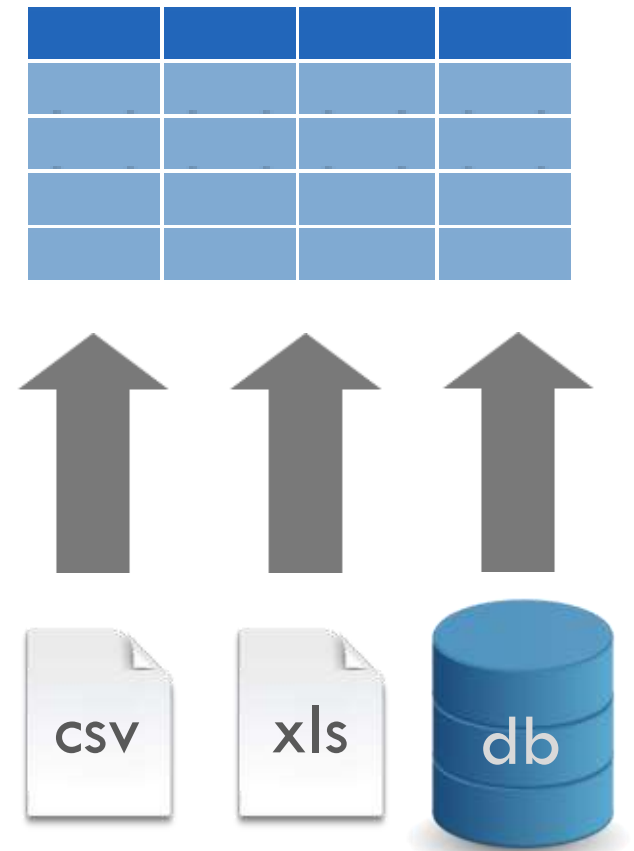
```
library("tidyverse")
```

Loads package

1 x per R Session

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



read_csv()

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



function
(does stuff)

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

function
(does stuff)

argument
(input)

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

object
(stores output)

function
(does stuff)

argument
(input)

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

object
(stores output)

function
(does stuff)

argument
(input)

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

assignment operator
("gets")

read_csv()

data frame to
read data into

name of
CSV file

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

covid_testing

covid_testing.csv



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

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.



What else?



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* = *lappend*)

File with arbitrary delimiter

write_delim(*x*, *path*, *delim* = " ", *na* = "NA", *append* = FALSE, *col_names* = *lappend*)

CSV for excel

write_excel_csv(*x*, *path*, *na* = "NA", *append* = FALSE, *col_names* = *lappend*)

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* = *lappend*)

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())
```

a,b,c
1,2,3
4,5,NA

A	B	C
1	2	3
4	5	NA

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")

a,b,c
1;2;3
4;5;NA

A	B	C
1	2	3
4	5	NA

Semi-colon Delimited Files

read_csv2("file2.csv")

write_file(*x* = "a;b;c\n1;2;3\n4;5;NA", *path* = "file2.csv")

a|b|c
1|2|3
4|5|NA

A	B	C
1	2	3
4	5	NA

Files with Any Delimiter

read_delim("file.txt", *delim* = "|")

write_file(*x* = "a|b|c\n1|2|3\n4|5|NA", *path* = "file.txt")

a b c
1 2 3
4 5 NA

A	B	C
1	2	3
4	5	NA

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

a,b,c
1,2,3
4,5,NA

Example file

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

1	2	3
4	5	NA

Skip lines

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

A	B	C
1	2	3
4	5	NA

No header

read_csv(*f*, *col_names* = FALSE)

A	B	C
1	2	3

Read in a subset

read_csv(*f*, *n_max* = 1)

x	y	z
1	2	3
4	5	NA

Provide header

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

A	B	C
NA	2	3
4	5	NA

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()
## )
```

age is an integer

sex is a character

earn is a double (numeric)

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()**

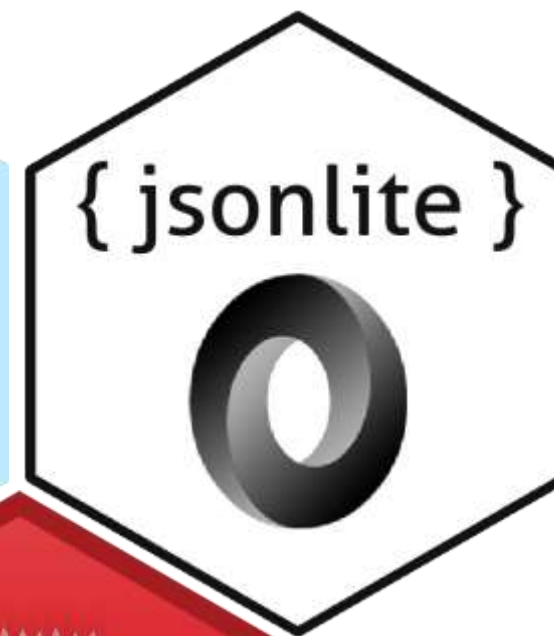
```
x <- read_csv("file.csv", col_types = cols(  
  A = col_double(),  
  B = col_logical(),  
  C = col_factor()))
```

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)
```





Databases



[Microsoft SQL Serve](#)



[MonetDB](#)



[MongoDB](#)



[MySQL](#)



[Netezza](#)



[Oracle](#)



[Amazon Redshift](#)



[Apache Hive](#)



[Apache Impala](#)



[Athena](#)



[Cassandra](#)



[Google BigQuery](#)



[Other Databases](#)



[PostgreSQL](#)



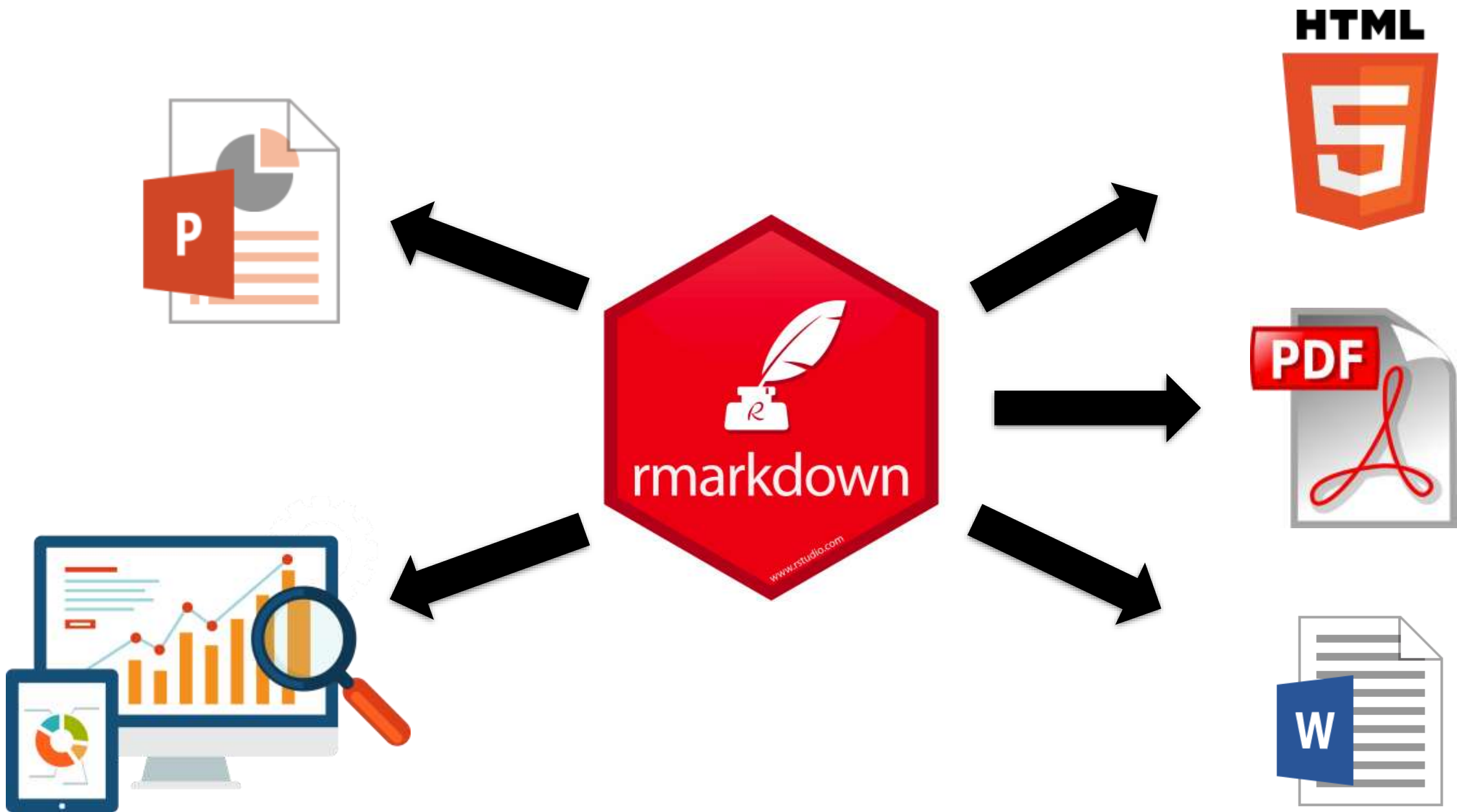
[SQLite](#)



[Salesforce](#)



[Teradata](#)



R Interface to Python



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