Introduction to R and R Studio

Session 1
August 16, 2020
Amrom Obstfeld

August 16 2020	**Session**	**Instructor**
9:00 am - 9:30 am	Instructor Introductions, Introduction to technology	Amrom Obstfeld
9:30 am - 10:15 am	Introduction to R and RStudio	Amrom Obstfeld
10:30 pm - 11:15 am	Reproducible Reporting	Amrom Obstfeld
11:30 am - 1:00 am	Data Visualization	Stephan Kadauke
August 23 2020		
9:00 am - 10:30 pm	Data Transformation	Amrom Obstfeld
10:45 am - 12:15 pm	Statistical Analysis	Dan Herman
12:30 pm - 1:00 pm	Workshop Close out	Amrom Obstfeld

Lesson Goals

1. Get oriented to R and RStudio
2. Learn some fundamentals of coding

Lesson Objectives

1. Log in and tour RStudio Cloud
2. Execute code at the console
3. Define and use functions
4. Define and create objects in the environment
5. Load data into R and interact with a dataframe

Getting Oriented to R

What is R?

- R is a statistical programming language.
- Using R you can load, analyze, and visualize data.
- R also provides an environment in which we can conduct reproducible data analysis.
  - Documented
  - Revisable
  - Shareable

RStudio: The Portal to R

- RStudio is an integrated development environment (IDE)

- Using RStudio we can interact with the R programming language to:

  - Write and execute code interactively
  - View data
  - Debug and fix errors
  - Author our code

RStudio: In the Cloud… In Your Home

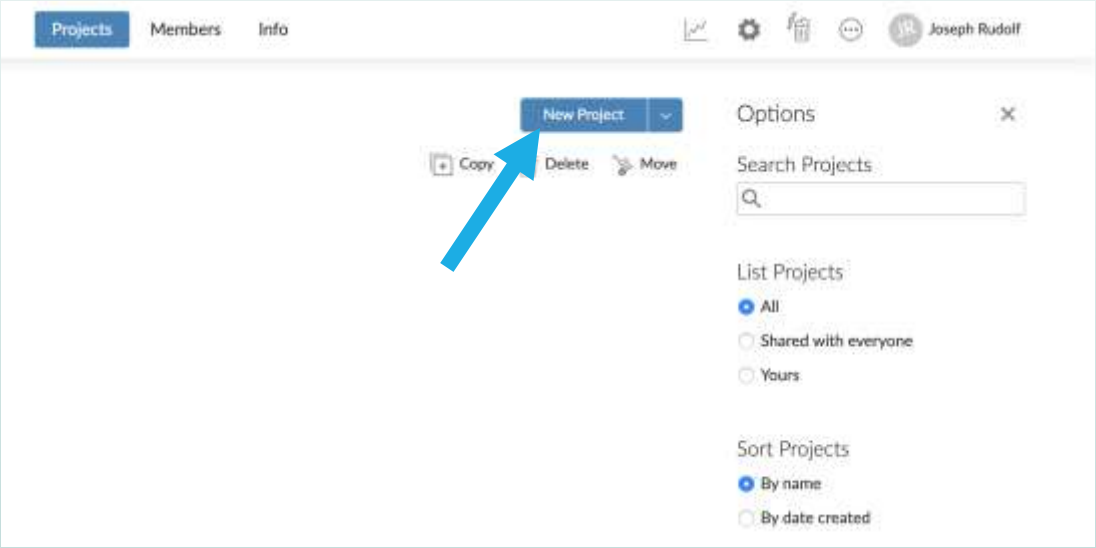
- RStudio Cloud: An online hosted version of RStudio that we will use for these course sessions

- RStudio Desktop: A locally installed version of RStudio that you will use when you get home to continue your learning

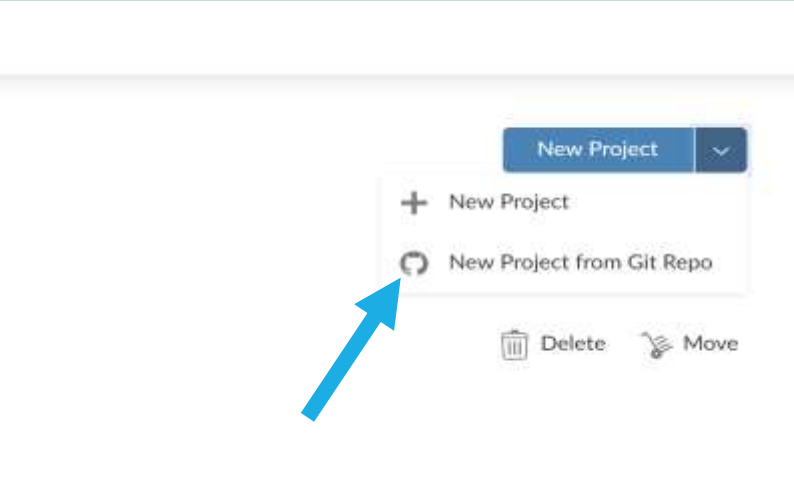
Note: Use Rstudio Cloud only for this course. Do not upload protected health information to the cloud!

Your Turn

Navigate to:
rstudio.cloud



Register for an account



Type in this link at the prompt:
https://tinyurl.com/pennres2020

Studio Cloud

API R Workshop 2020 / API R Project 2020

Joseph Rudolf

Spaces

- Your Workspace
- AACC 2019 Introduction to R
- API R Workshop 2020
- New Space

Learn

- Guide
- What's New
- Primers
- Cheat Sheets
- Feedback and Questions

Info

- Plans & Pricing
- Terms and Conditions
- System Status

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | R 4.0.0

Console Terminal Jobs

/cloud/project/

```
R version 4.0.0 (2020-04-24) -- "Arbor Day"
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.
Type 'q()' to quit R.

>
```

Environment History Connections Git

Import Dataset | List

Global Environment

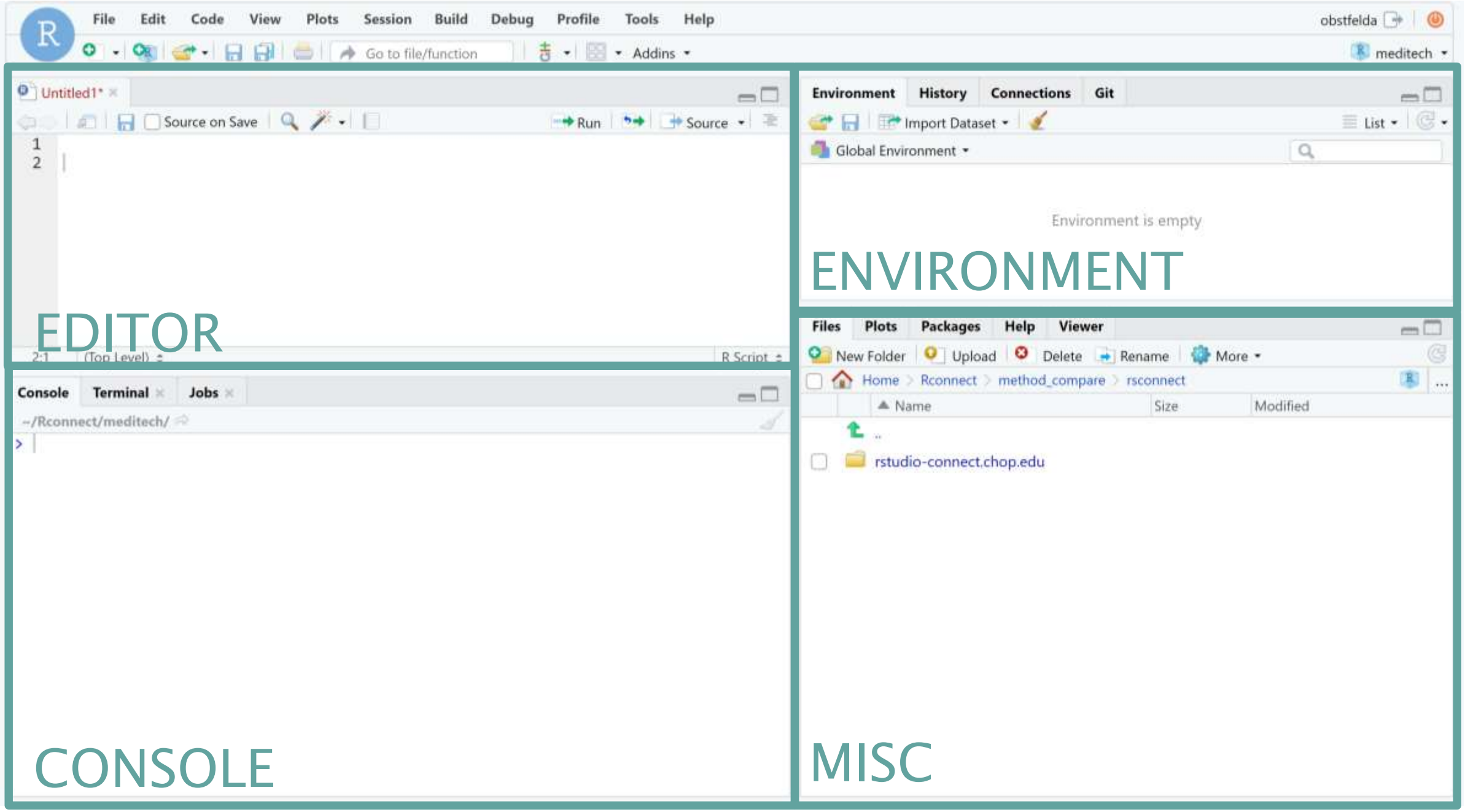
Environment is empty

Files Plots Packages Help Viewer

New Folder | Upload | Delete | Rename | More

Cloud > project

Name	Size	Modified
..		
.gitignore	621 B	Jun 23, 2020, 9:21 PM
.Rhistory	0 B	Jul 13, 2020, 1:26 PM
.Rprofile	88 B	Jun 23, 2020, 9:25 PM
03 – Visualize.Rmd	3 KB	Jul 10, 2020, 8:46 AM
04 – Transform.Rmd	4.8 KB	Jul 13, 2020, 12:08 PM
05 – Stats.Rmd	5.8 KB	Jul 10, 2020, 8:46 AM
06 – Advanced Reporting.Rmd	871 B	Jul 13, 2020, 7:08 AM
coursepack		
data		
LICENSE	1 KB	Jun 23, 2020, 9:21 PM
presentations		
project.Rproj	205 B	Jul 13, 2020, 1:26 PM
README.md	6.9 KB	Jul 12, 2020, 3:42 PM


File Edit Code View Plots Session Build Debug Profile Tools Help
obstfelda
Go to file/function
Addins
meditech
Untitled1*
Source on Save
Run
Source
EDITOR
Environment History Connections Git
Import Dataset
List
Global Environment
Environment is empty
ENVIRONMENT
Console Terminal Jobs
~/Rconnect/meditech/
CONSOLE
Files Plots Packages Help Viewer
New Folder
Upload
Delete
Rename
More
Home > Rconnect > method_compare > rsconnect
Name
Size
Modified
rstudio-connect.chop.edu
MISC

The Basics of Coding

The Basics of Coding: Calculation

- R is a calculator!

```
> 2 + 3 + 2
[1] 7
>
>
> 4 * 20
[1] 80
>
>
> 6 ^ 8
[1] 1679616
>
```

enter/return to execute code

answer returned here

Your Turn #1

Place your cursor at the console and click to enter the console.

Complete the following calculation:

-For the date 12-29-1974

-Take the four digit year

-Subtract the month then multiply by the day

What did you get?

- A four digit number? A five digit number?

```
> 1974 - 12 * 29
[1] 1626
>
>
> (1974 - 12) * 29
[1] 56898
```

- Order of operations matters!

The Basis of Coding: 3 Players

1. Functions
2. Arguments
3. Objects

The Basics of Coding: Functions

- Code that extends our reach beyond the basic operators

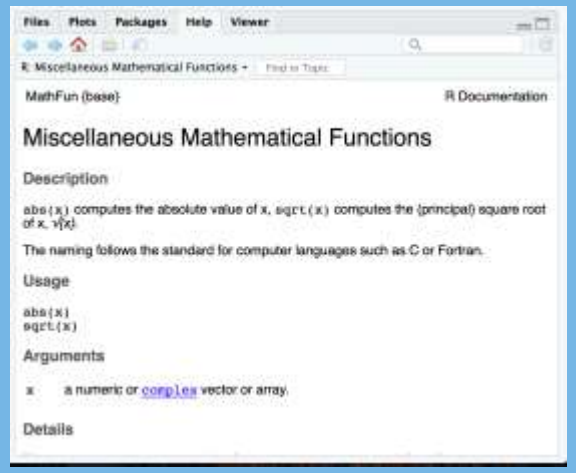
```
> abs(-77)
[1] 77
>
```

function
(does stuff)

abs(-77)

- What if I don't know what a function does?

```
> ?abs()
>
```



The Basics of Coding: Arguments

- The input that defines what the function should do

```
> abs(-77)
[1] 77
>
```

argument
(input)

abs(-77)

The Basics of Coding: Objects

- Objects are the container for your output

object
(stores output)

```
my_abs <- abs(-77 )
```

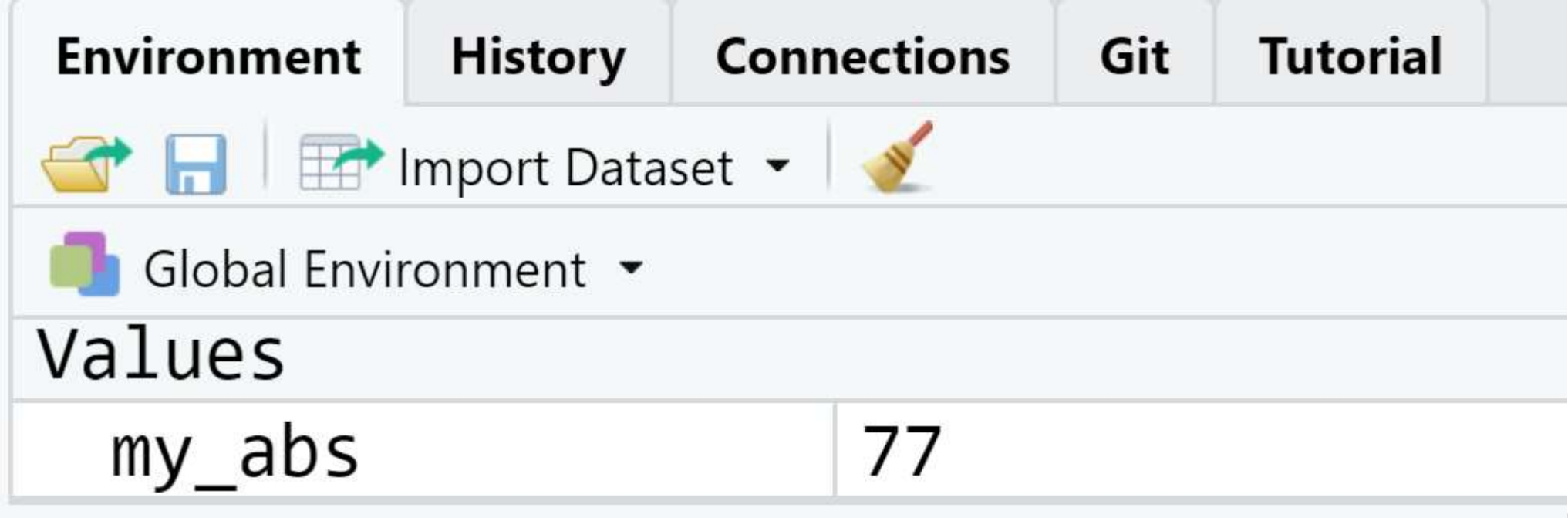
Checking the contents of an object

- Entering the object name at the console allows us to output the contents of an object.

```
> abs(-77)
[1] 77
> my_abs <- abs(-77)
> my_abs
[1] 77
>
```

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!

```
> log(my_abs,2)
[1] 6.266787
>
```

Knowledge Check

Consider this code:.

```
mean_age <- mean(age, na.rm=TRUE)
```

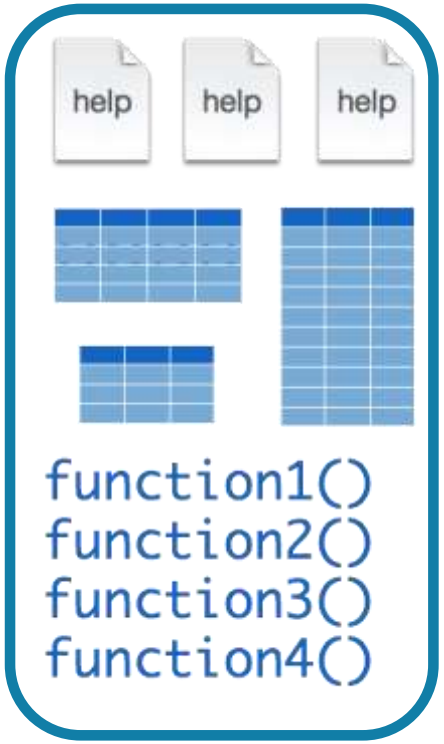
Which is the function?
Which is the argument?
Which is the objects?

The Basics of Coding: Packages

- A package is a collection of functions.
- Packages extend the capabilities of the base R programming language.
- The **tidyverse** includes functions for reading data into the R environment, cleaning and manipulating data, and plotting our results.

A Word About Packages

tidyverse



1

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

Downloads files to computer

1 x per computer

2

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

Loads package

1 x per R Session

Your Turn #2

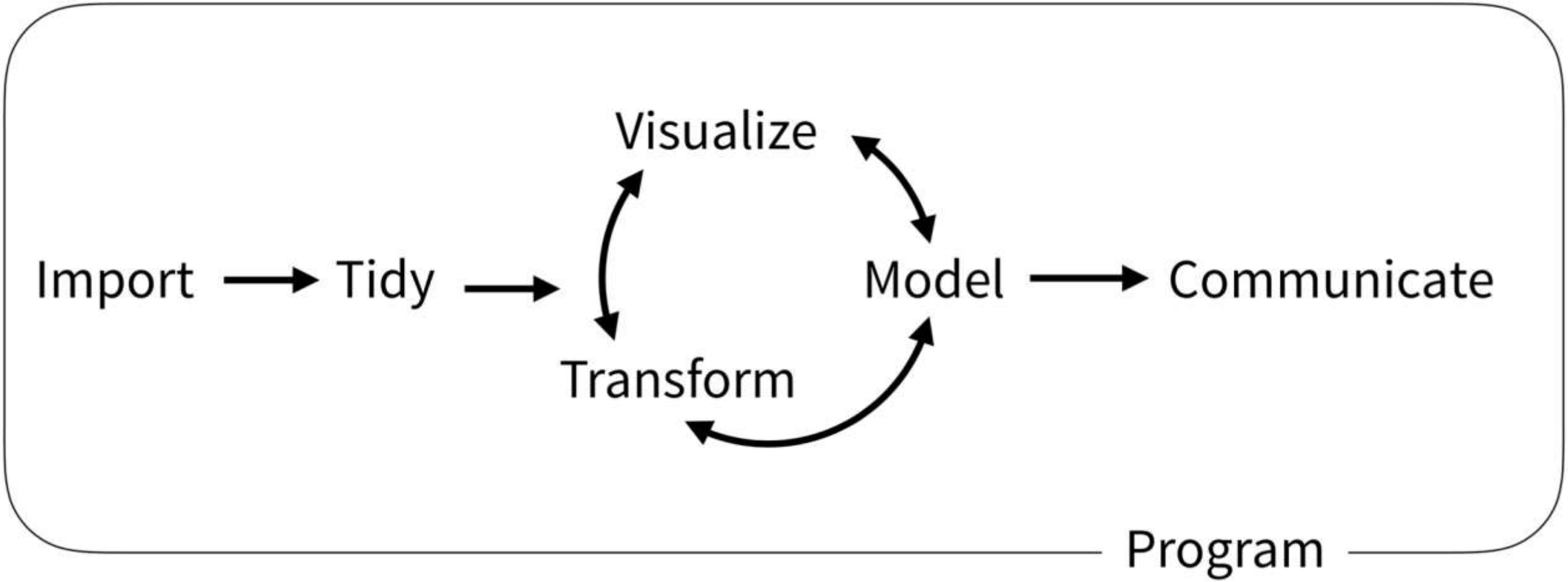
Run the following in the console:

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

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

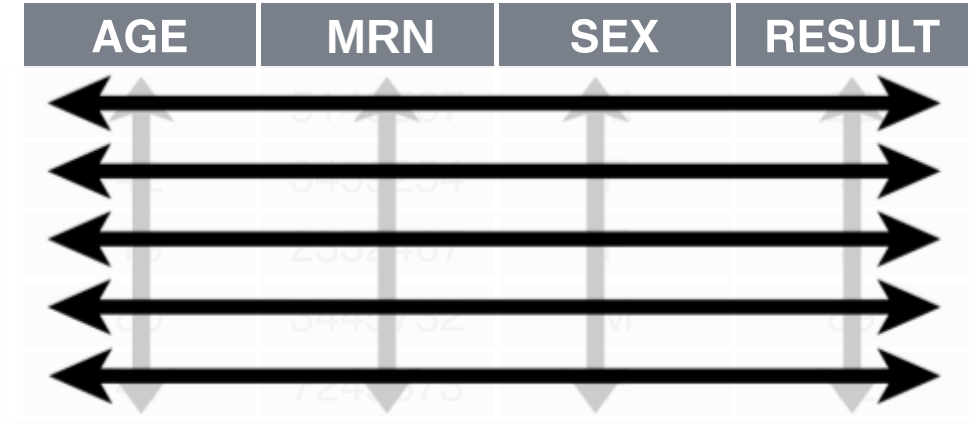
The Data Analysis Process

Typical Data Analysis Pipeline



From *R for Data Science*

What is a “Tidy” Data Frame

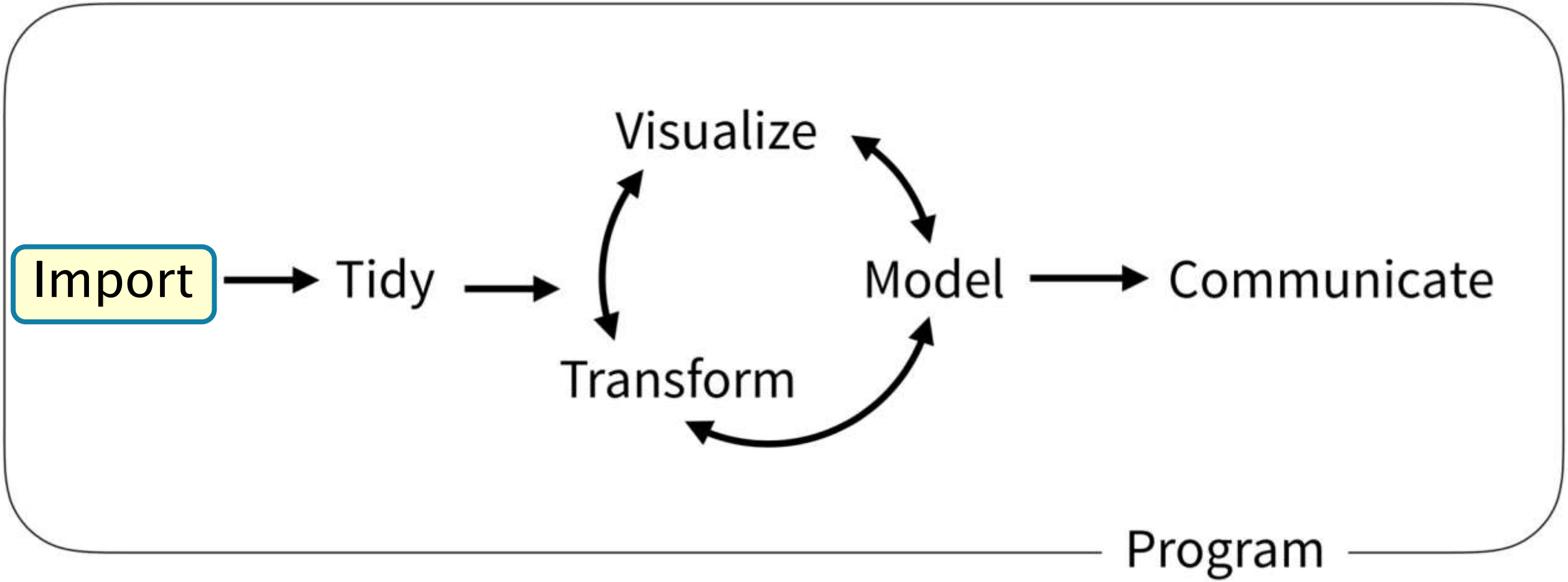


A data set is **tidy** if:

1. Each variable is in its own column
2. Each observation is in its own row
3. Each value is in its own cell

Importing Data

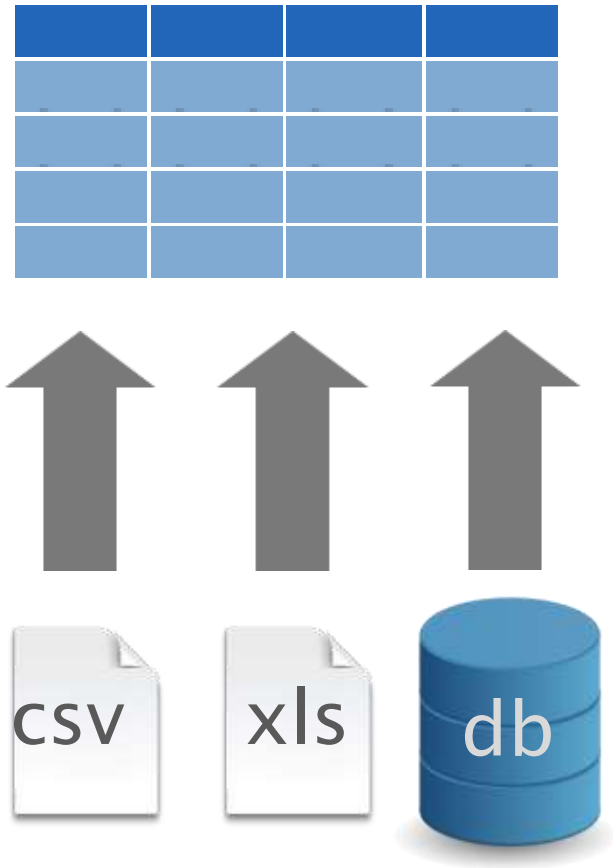
Typical Data Analysis Pipeline



From *R for Data Science*

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



plain text ("flat") file

header row

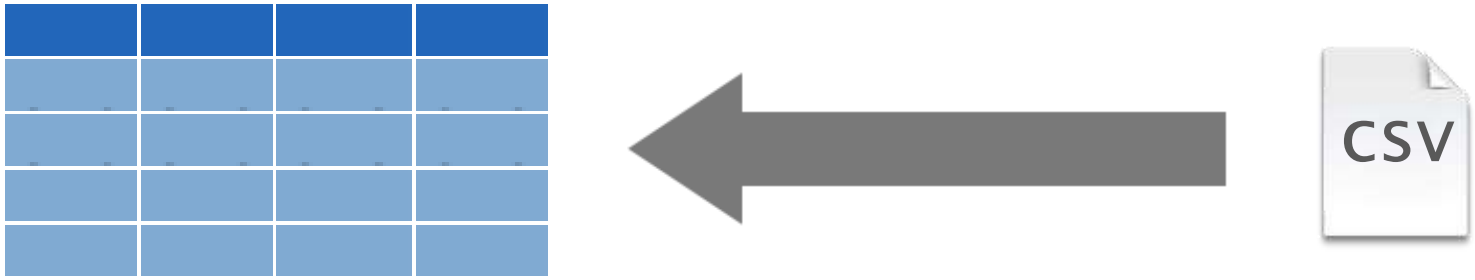
02-example

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

rectangular structure

Loading Data to Create a Dataframe

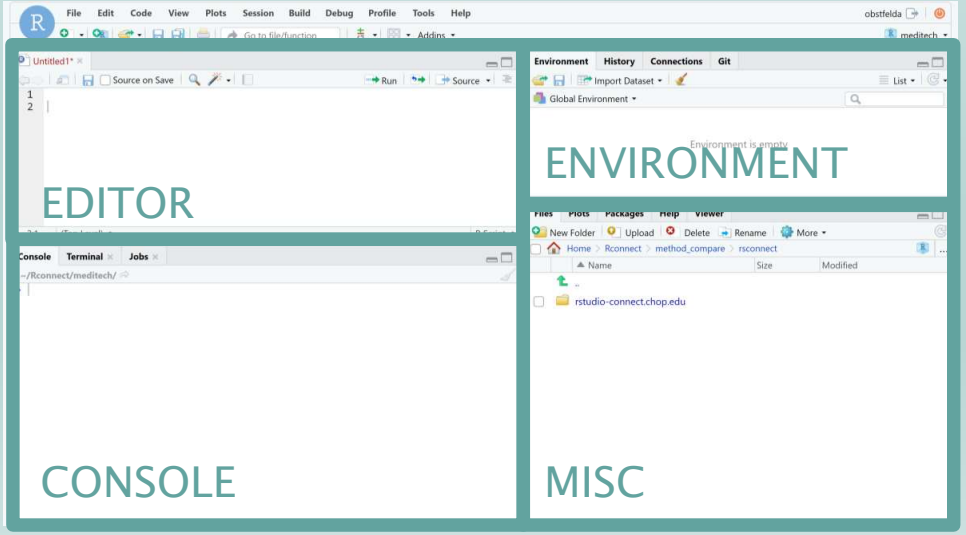
```
data_frame <- read_csv("file_name")
```



Memory Check

After reading in data using code such as this, where will you the data appear?

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



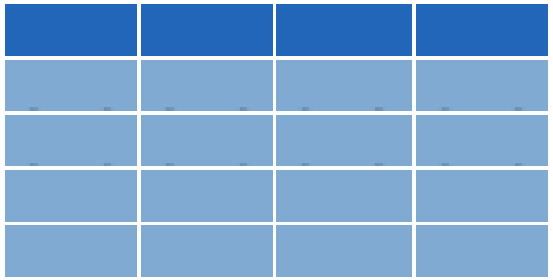
read_csv()

data frame to read data into

name of CSV file

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

covid_testing



covid_testing.csv

Your Turn #3

Configure environment and load the Covid Testing CSV:

Load the tidyverse library using library(tidyverse)

Use the **read_csv()** function to load the data

-File_name argument: "data/covid_testing.csv"

-Object name: covid_testing

What's in a name?

- Capitalization matters

covid_testing ≠ Covid_testing ≠ COVID_TESTING

- Strive for names that are concise and meaningful (not easy!)

Bad	Still not great	Good
p	name	patient_name

Viewing the Contents of a Dataframe

single click to explore the data

Viewing the Contents of a Dataframe

covid_testing

Filter

	mrn	first_name	last_name	gender	pan_day	test_id	clinic_name	re
1	5001412	jhezane	westerling	female	4	covid	inpatient ward	
2	5000533	penny	targaryen	female	7	covid	clinical lab	
3	5009134	grunt	rivers	male	7	covid	clinical lab	
4	5008518	melisandre	swyft	female	8	covid	clinical lab	
5	5008967	rolley	karstark	male	8	covid	emergency dept	
6	5011048	megga	karstark	female	8	covid	oncology day hosp	negat
7	5000663	ithoke	targaryen	male	9	covid	clinical lab	negat
8	[illegible]2158	ravella	frey	female	9	covid	emergency dept	negat
[illegible]	5003794	styr	tyrell	male	9	covid	clinical lab	negat
[illegible]	5004706	wvnafrvd	seaworth	male	9	covid	clinical lab	negat

[illegible]owing 1 to 11 of 15,524 entries, 17 total columns

17 Attributes (Columns)

15,524 Observations (Rows)

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)
```

RStudio

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"
```

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

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

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

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

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()**

```
x1A <- parse_number(x1A)
```

What Else?

Import Excel files (.xls, .xlsx)

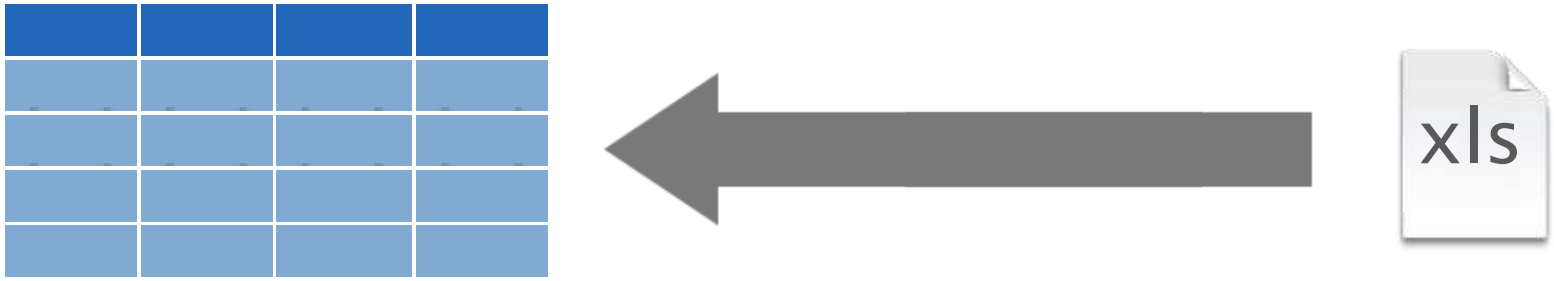
```
library(readxl)
```

read_excel()

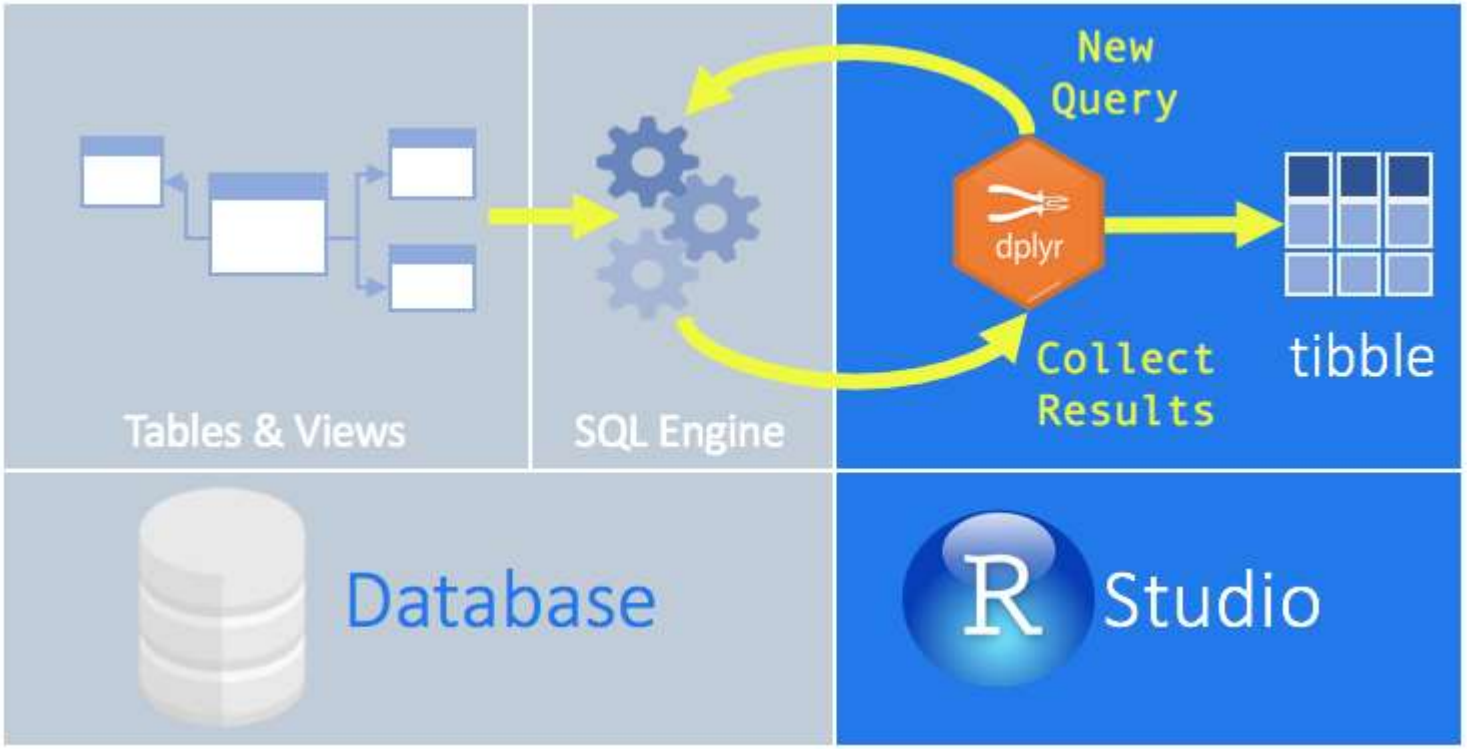
data frame to read data into

name of Excel file

```
data_frame <- read_excel(file_name)
```



Read Directly From Database



Reads SPSS, Stata, and SAS files

Format	Typical Extension	Import Package	Export Package	Installed by Default
Comma-separated data	.csv	data.table	data.table	Yes
Pipe-separated data	.psv	data.table	data.table	Yes
Tab-separated data	.tsv	data.table	data.table	Yes
CSVY (CSV + YAML metadata header)	.csvy	data.table	data.table	Yes
SAS	.sas7bdat	haven	haven	Yes
SPSS	.sav	haven	haven	Yes
Stata	.dta	haven	haven	Yes
SAS XPORT	.xpt	haven	haven	Yes
SPSS Portable	.por	haven		Yes
Excel	.xls	readxl		Yes
Excel	.xlsx	readxl	openxlsx	Yes
R syntax	.R	base	base	Yes
Saved R objects	.RData, .rda	base	base	Yes
Serialized R objects	.rds	base	base	Yes

Lesson Goals

1. Get oriented to R and RStudio
2. Learn some fundamentals of coding

Lesson Objectives

1. Log in and tour RStudio Cloud
2. Execute code at the console
3. Define and use functions
4. Define and create objects in the environment
5. Load data into R and interact with a dataframe