

R MIEF Skills Workshop

Session 1

January 12, 2026

Introduction

Purpose

Learning R

Purpose of this course: Put you on the right track to use R for data or policy analysis.

- As data work has become near-ubiquitous in the research/policy world, so have basic tasks like aggregating, analyzing, summarizing, and visualizing data.
- **The vast majority** of research assistant/analyst (RA) work consists of cleaning and constructing datasets for analysis
- **Entry-level RA positions rarely require complex econometric/regression skills**

Introduction

Purpose

Learning R

You should think of learning R like learning a language.

- Taking a six-hour course won't make you proficient in it
- If you don't practice it, you'll forget it
- Solution — Find ways to use R in your life, either personally or professionally

Why R?

Jobs

Beautiful Tables

Beautiful Graphs

Beautiful Maps

- Many entry-level research jobs in policy, economic, development, or political science institutions now expect quantitative work using Stata, R, or Python
- Coding skills make you more valuable in any position — data adds value to nearly every kind of research!

Why R?

Jobs

Beautiful Tables

Beautiful Graphs

Beautiful Maps

2021 Expected vs. Actual Fantasy Points

Top 40 Running Backs

Player	Team	Expected FP	Expected FP Rank	Actual FP	Actual FP Rank
Jonathan Taylor	 IND	301.20	RB 1	324.20	RB 1
Najee Harris	 PIT	292.67	RB 2	228.10	RB 4
Joe Mixon	 CIN	251.89	RB 3	254.80	RB 3
Leonard Fournette	 TB	232.12	RB 4	221.10	RB 5
Ezekiel Elliott	 DAL	225.03	RB 5	214.76	RB 6
Austin Ekeler	 LAC	217.88	RB 6	263.90	RB 2
Antonio Gibson	 WAS	216.49	RB 7	186.70	RB 11

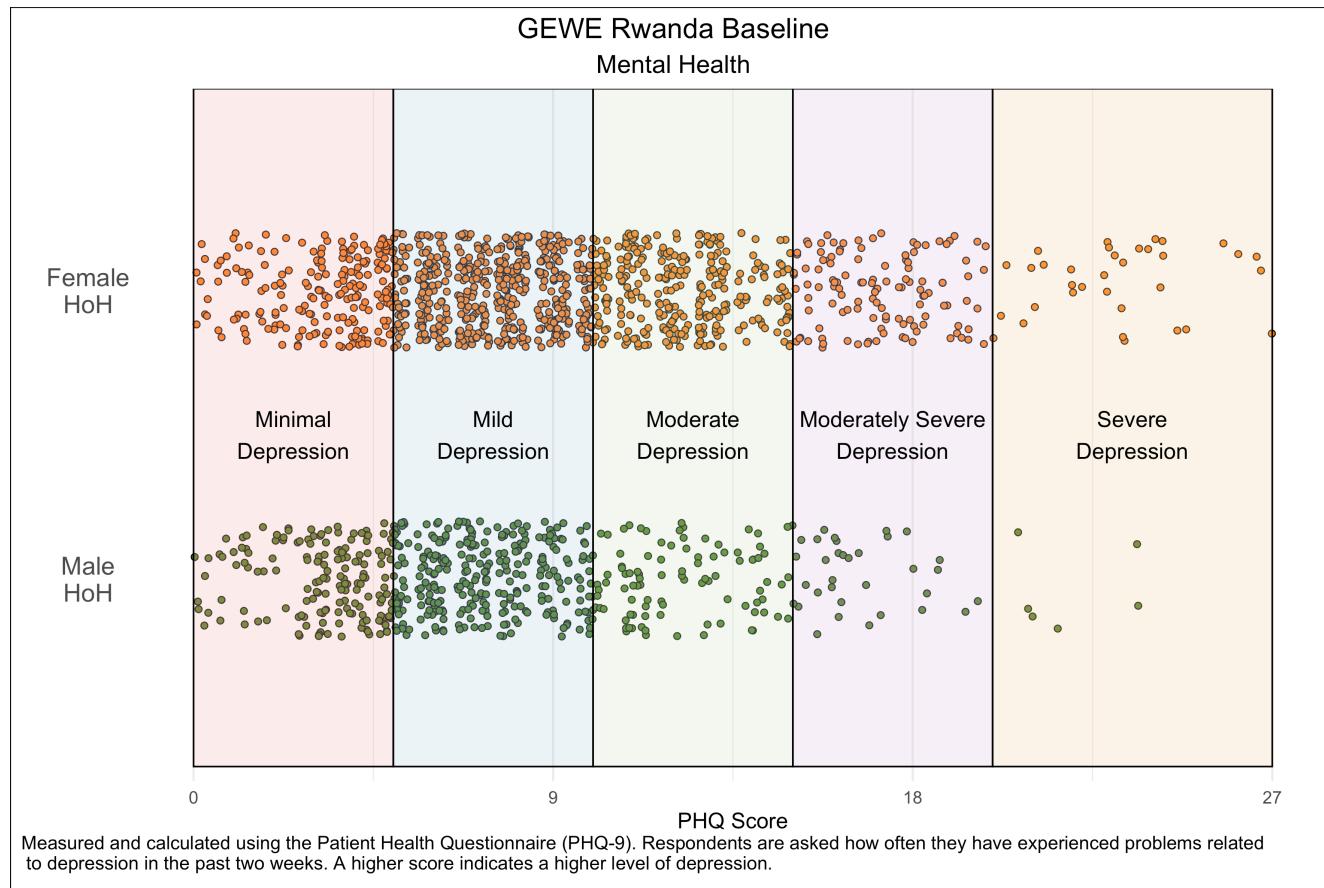
Why R?

Jobs

Beautiful Tables

Beautiful Graphs

Beautiful Maps



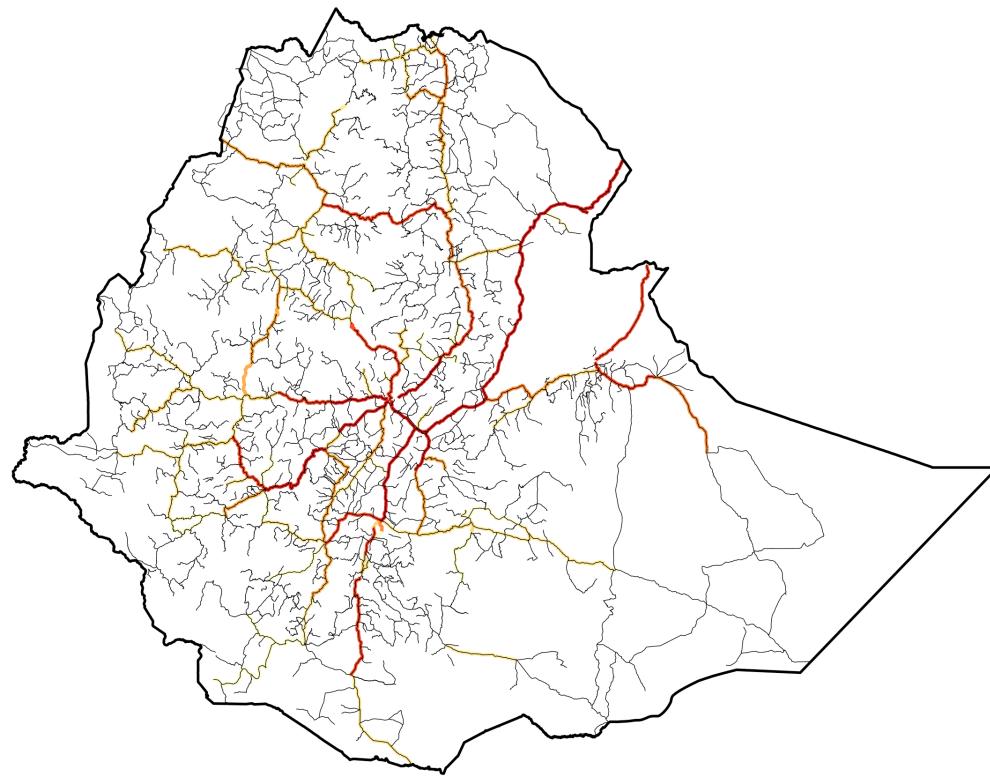
Why R?

Jobs

Beautiful Tables

Beautiful Graphs

Beautiful Maps



2002 Km of Traffic	0-10,000	10,000-20,000	20,000-30,000	30,000-40,000	40,000-50,000	50,000-100,000	100,000-1,000,000
--------------------	----------	---------------	---------------	---------------	---------------	----------------	-------------------

Today

- Learn how to:
 - Think as a coder
 - Identify the basic components of data analysis
 - Set up your environment to use R and RStudio
 - Identify and address basic errors in your R setup
- Be introduced to:
 - The R coding language
 - The building blocks of coding in R: scalars, vectors, lists, and data frames
 - Creating a scatter plot, density plot, and bar chart using the ggplot2 package
 - Creating flexible and easy-to-read tables of any dataset using the gt package
 - Creating simple academic-standard regression output tables using the stargazer package

Think As A Coder

Recipe Analogy

- Your data are your ingredients
- Your script is your recipe
- Your output is your... cake? Whatever you're making

What this means:

- **Make sure you have all of the ingredients you need.**
- **Follow every step of the recipe — you can't skip any steps!**
- If you mess up somewhere, you have to start from scratch to make sure that you get the correct outcome.

Identify the Basic Components of Data Analysis

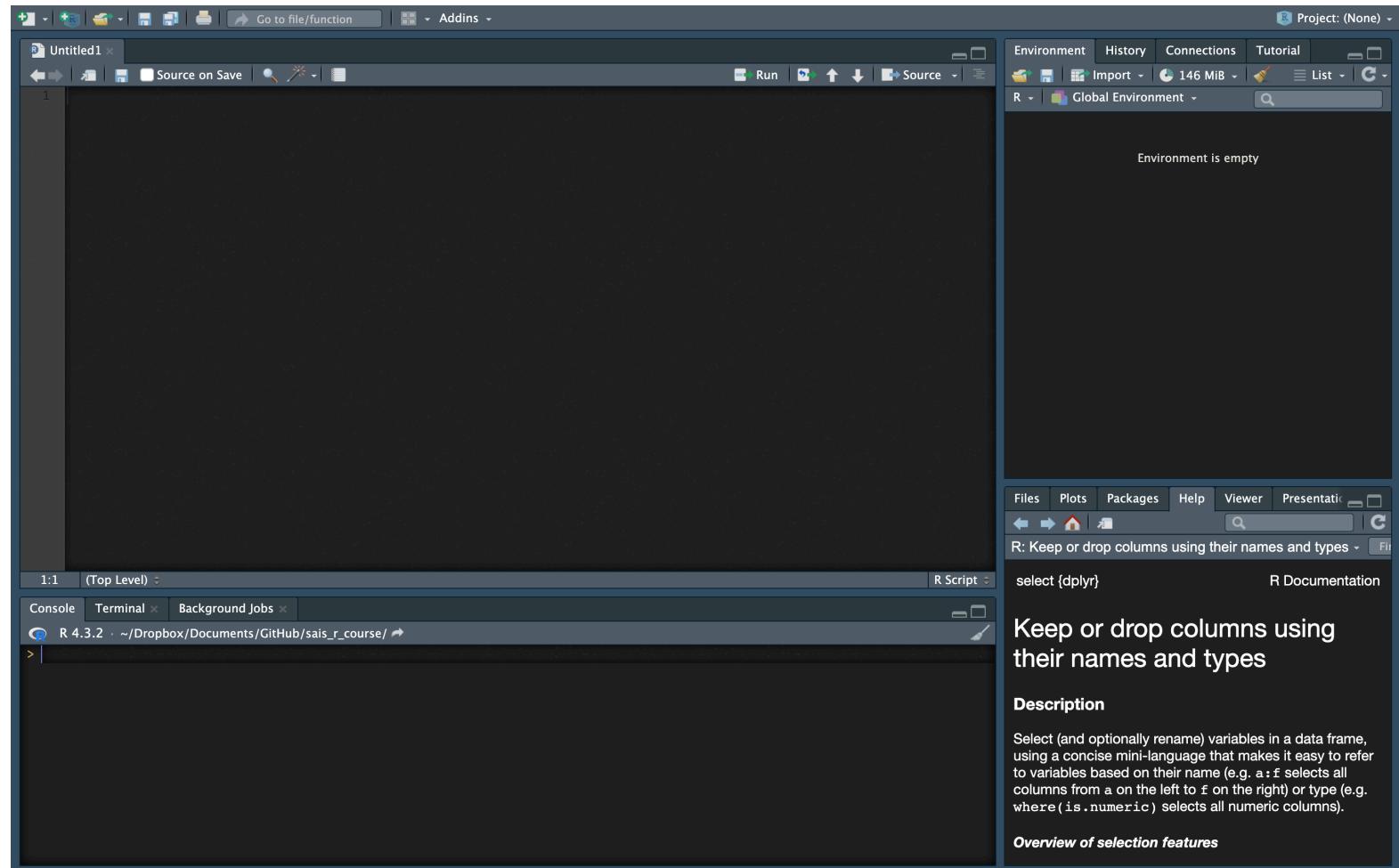
Three main components to your data analysis:

- Data
- Code
- Outputs

Questions to ask:

- Where is the file stored?
- In which format is the file?
- How easy is it for you/other people to access it?
- Is it intuitive for you to navigate to the file?

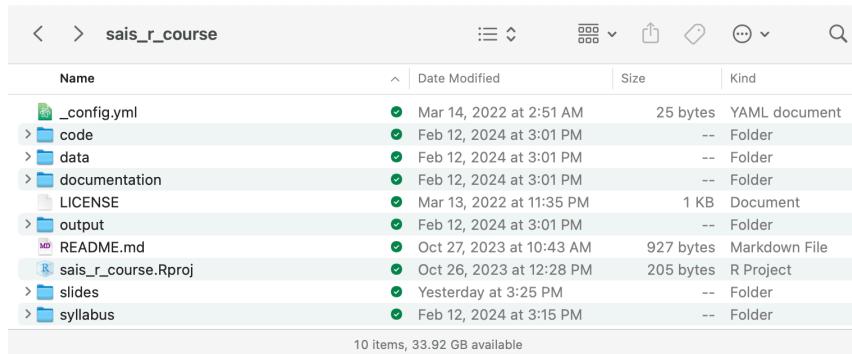
Your RStudio Environment



Setting Up Your RStudio Environment

Setting Up Your Data Structure

Setting Up Your Script



The screenshot shows a file explorer window with the following directory structure and file list:

Name	Date Modified	Size	Kind
_config.yml	Mar 14, 2022 at 2:51 AM	25 bytes	YAML document
code	Feb 12, 2024 at 3:01 PM	--	Folder
data	Feb 12, 2024 at 3:01 PM	--	Folder
documentation	Feb 12, 2024 at 3:01 PM	--	Folder
LICENSE	Mar 13, 2022 at 11:35 PM	1 KB	Document
output	Feb 12, 2024 at 3:01 PM	--	Folder
README.md	Oct 27, 2023 at 10:43 AM	927 bytes	Markdown File
sais_r_course.Rproj	Oct 26, 2023 at 12:28 PM	205 bytes	R Project
slides	Yesterday at 3:25 PM	--	Folder
syllabus	Feb 12, 2024 at 3:15 PM	--	Folder

10 items, 33.92 GB available

- code
- data
 - raw
 - intermediate
- output
- documentation (Optional)
- slides (Optional)

Setting Up Your RStudio Environment

Setting Up Your Data Structure

Setting Up Your Script

Defining your code's purpose — What are its inputs? What are its outputs?

```
## Programming for Professional Research Using R -- Session 1

#### Session Description

# Basic R Use - Introduction to R and RStudio, How to import/export data
# Using Basic Tidyverse Functions - how to subset, mutate, summarize, and reshape data
# Using R in a Collaborative Setting - Introduction to downloading and publishing data using
# Github
```

Set up your packages — Which user-created functions do you plan to use?

```
## 1. Setup ----

### Packages Random

# NOTE -- Unlike using library(), the 'pacman::p_load()' function installs the package if it is
# already not present in the user's R environment.

if(!require(pacman)) install.packages("pacman")

pacman::p_load(tidyverse, data.table, janitor, usethis)
```

Import your data — Which data frames do you need to fulfill the script's purpose?

```
## 2. Import Data ----

# This is session 1 so we're going to import the data from Dropbox.

if(!(file.exists("data/final/wvs_values_norms_data.csv"))){ # Checks whether the data has been
  # downloaded already
  usethis::use_zip(
    "https://www.dropbox.com/scl/fo/vnxjbqyqlg9z368coh1vq/h?rlkey=zc66o2ll7613b5e9ipp915ynk8dl=1"
  )
}

norms_values_data <- data.table::fread( # Other options are base R's read.csv() and data.table::fread()
  "data/final/wvs_values_norms_data.csv", na.strings = ""
```

R Basics

Scalars

Vectors

More Vectors

Selecting Vector Elements

This is a scalar:

```
a ← 2
```

It's an object (a) with a single value (2). Notice that we assign the value 2 to a using an arrow ←.

R Basics

Scalars

Vectors

More Vectors

Selecting Vector Elements

These are vectors:

```
vector <- c(1, 2, 5)
vector
```

```
## [1] 1 2 5
```

```
vector <- 2:6
vector
```

```
## [1] 2 3 4 5 6
```

R Basics

Scalars

Vectors

More Vectors

Selecting Vector Elements

These are more vectors:

```
vector <- seq(2, 3, by = 0.5)  
vector
```

```
## [1] 2.0 2.5 3.0
```

Vectors take multiple values. Notice that vectors have a specific **order**: `c(1, 2)` is not the same as `c(2, 1)`.

R Basics

Scalars

Vectors

More Vectors

Selecting Vector Elements

```
## By Position

x[4]          # Fourth element

x[-4]         # Everything but the fourth element

x[2:4]        # Elements two to four

x[-(2:4)]    # Everything but elements two to four

x[c(1, 5)]   # Elements one and five
```

```
# By Value

x[x == 10]      # Elements which are equal to 10

x[x < 0]        # Elements that are less than zero

x[x %in% c(1, 2, 5)] # Elements in the set 1, 2, 5
```

R Basics

Lists

Some Classes

Data Frames

Assignment vs. Printing

This is a list:

```
y <- list("a", 1, "b")
```

Lists are different from vectors in one key manner: **they can take objects of different classes.** Vectors can only take objects of the same class.

R Basics

[Lists](#)[Some Classes](#)

[Data Frames](#)[Assignment vs. Printing](#)

Here are some classes in R:

```
a <- "a"      # Character
b <- 1         # Integer
c <- 1.2       # Numeric
d <- 1 + 2i    # Complex
e <- TRUE      # Logical
```

R Basics

Lists

Some Classes

Data Frames

Assignment vs. Printing

This is a data frame:

```
df <- data.frame(  
  first_name = c("Mark", "Mary", "July"),  
  last_name  = c("Smith", "John", "Sanchez"),  
  age        = c(21, 34, 55)  
)
```

At their core, data frames are just a group of named vectors of the same length. In practice, we visualize data frames as **tables** where each vector is a **column**, or variable, and each group of nth elements of the vectors is a **row**, or observation.

R Basics

Lists

Some Classes

Data Frames

Assignment vs. Printing

KEY — We can classify the code we write in our script into two general categories:

1- Assignment code. We are **assigning** a value to an object:

```
a ← 2
```

2- Printing code. We are **printing** an object's value:

```
a
```

```
## [1] 2
```

```
print(a)
```

```
## [1] 2
```

Only 2- results in something appearing in your console. When you assign a value to an object, nothing prints in the console.

R Basics

NOTE — R works in a manner that allows to write a specific function over multiple lines. This is called a **code chunk**.

This:

```
a ← mean(c(seq(1, 4, by = 0.5)))  
a
```

```
## [1] 2.5
```

Is the same as this:

```
a ← mean(  
  c(  
    seq(1, 4, by = 0.5)  
  )  
)  
a
```

```
## [1] 2.5
```

R Basics

Functions

Packages

Like in mathematics, a function is an expression or rule that takes an input, or *argument*, and returns an *output*. A function is any f such that $f(x) = y$.

In R, functions take the following form:

```
# The function is called  
sum_function <- function(x, y) { # x and y are the function's arguments  
  # Within the brackets {}, we define the function's output  
}  
  
sum_function(2, 3)
```

```
## [1] 5
```

The vast majority of using R is done using functions, either included with R or written by external users. For instance, to find the sum of 2 and 3, we can use the "base R" function `sum()`.

R Basics

Functions

Packages

A package is a bundle of functions created by external users. They innovate and, in many cases, improve upon base R functions. Most work in R is done using functions from external packages.

We will look at how to install and load packages shortly. Examples of commonly-used packages are `dplyr`, `tidyr`, `stringr`, `ggplot2`, and `data.table`.

R Basics

You can click anywhere in the code chunk and click "run" or Cmd+Enter (Mac)/Ctrl+Enter (Windows), and the whole chunk will run.

KEY — If you only select a portion of the code chunk and run that, then R will identify the chunk as unfinished and refuse to let you do anything else until you've "completed" it.

Coding Set Up

Packages are groups of user-created functions that help us accomplish tasks that would be harder/impossible using base R functions.

Easy

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

Better

The pacman package installs packages if they aren't installed yet, loads them otherwise

```
if(!require(pacman)) install.packages("pacman")
pacman :: p_load(tidyverse)
```

File paths help R identify where the files you want to use are located.

You want your code to be **reproducible** and **easy to use by other people**

Simple solution: Create an `.rproj` file that people can open to access your R environment

Better solution:

```
# Set User (this allows us to use fixed file paths but to adapt them  
# for multiple possible users)  
# 1 - Marc-Andrea Fiorina  
# 2 - Enter here if needed  
  
user <- 1  
  
if(user == 1) {  
  # Absolute file path  
  main_filepath <- "/Users/marc-andreafiorina/Dropbox/SAIS R Course/"  
}  
  
# Notice the relative file paths  
data_filepath <- paste0(main_filepath, "data/")
```

Importing Data

Easiest file type to import into R is a .csv file. But you can also import .xlsx, .dta (Stata), etc.

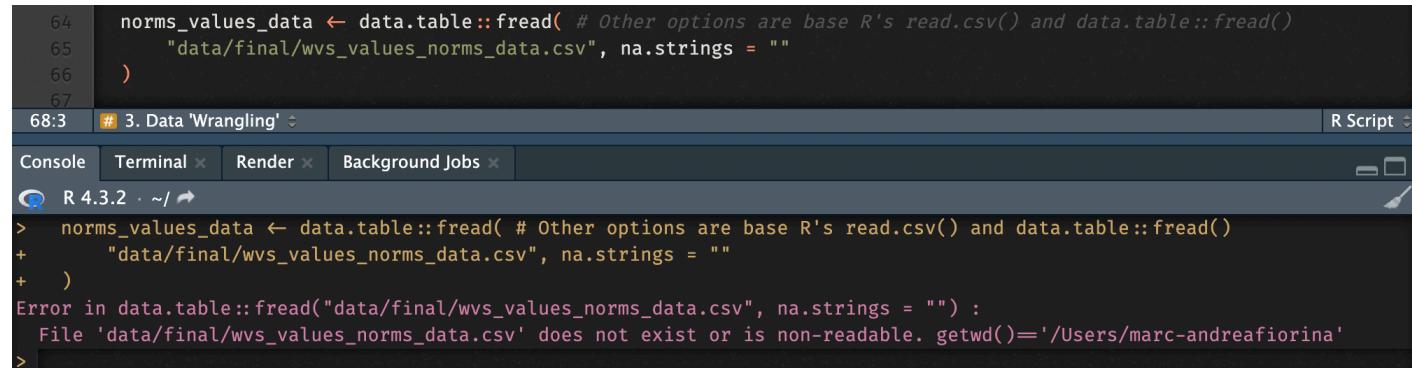
- Easy → `read.csv()`
- Harder (faster) → `data.table:: fread()`

```
norms_values_data ← data.table::fread(  
  "raw/wvs_values_norms_data.csv",  
  na.strings = ""  
)
```

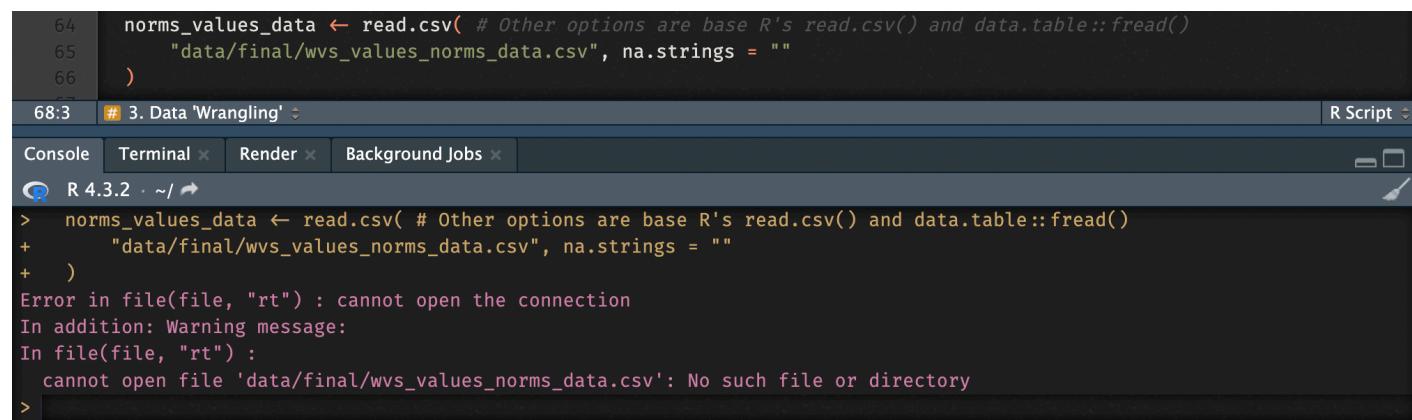
Note the construction of the "harder" method, with the " :: ". This is used to refer to the package from which the function originates. The structure is `package:: function()`.

Basic Issues in RStudio

Data Not Loading



64 norms_values_data ← data.table::fread(# Other options are base R's read.csv() and data.table::fread()
65 "data/final/wvs_values_norms_data.csv", na.strings = ""
66)
67
68:3 # 3. Data 'Wrangling' ◊ R Script ◊
Console Terminal × Render × Background Jobs ×
R 4.3.2 · ~/ ↻ Error in data.table::fread("data/final/wvs_values_norms_data.csv", na.strings = "") :
> norms_values_data ← data.table::fread(# Other options are base R's read.csv() and data.table::fread()
+ "data/final/wvs_values_norms_data.csv", na.strings = ""
+)
Error in data.table::fread("data/final/wvs_values_norms_data.csv", na.strings = "") :
File 'data/final/wvs_values_norms_data.csv' does not exist or is non-readable. getwd()= '/Users/marc-andreaefiorina'
>



64 norms_values_data ← read.csv(# Other options are base R's read.csv() and data.table::fread()
65 "data/final/wvs_values_norms_data.csv", na.strings = ""
66)
67
68:3 # 3. Data 'Wrangling' ◊ R Script ◊
Console Terminal × Render × Background Jobs ×
R 4.3.2 · ~/ ↻ Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
cannot open file 'data/final/wvs_values_norms_data.csv' : No such file or directory
>

Basic Issues in RStudio

Data Not Loading

Check:

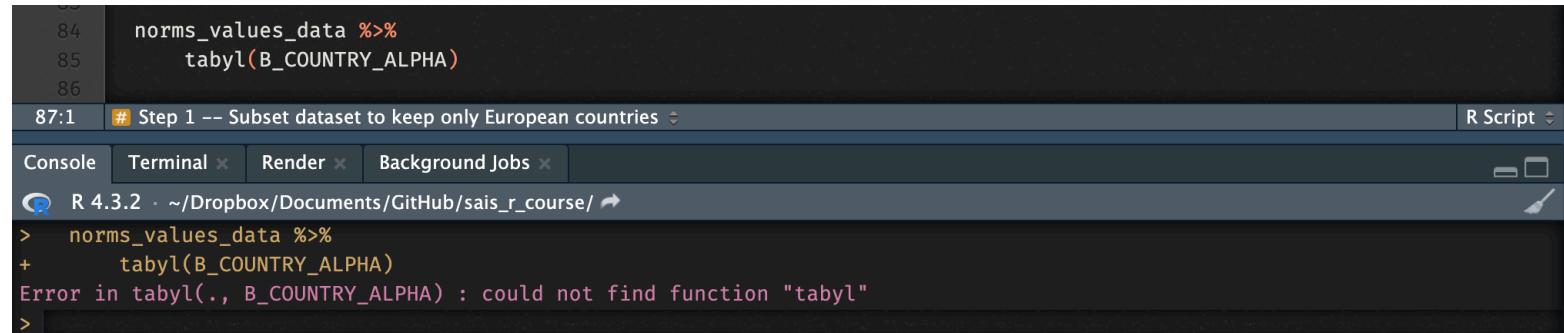
- Working directory — `getwd()` or check the top of the RStudio console.
- File path — are there any typos? Is your file where you expect it to be?

Solutions:

- If you were provided with an `.Rproj` file alongside your script, make sure that you opened the project.
- Modify the working directory using `setwd()` or correct the file path if need be!

Basic Issues in RStudio

Function Not Found



A screenshot of the RStudio interface. The code editor shows lines 84, 85, and 86 of a script, followed by line 87:1 which contains a comment: "# Step 1 -- Subset dataset to keep only European countries". The R console window below shows the same code being run. The command "norms_values_data %>% tabyl(B_COUNTRY_ALPHA)" is entered, followed by a prompt ">". An error message is displayed: "Error in tabyl(., B_COUNTRY_ALPHA) : could not find function "tabyl"".

```
84 norms_values_data %>%
85   tabyl(B_COUNTRY_ALPHA)
86
87:1 # Step 1 -- Subset dataset to keep only European countries
>
Console Terminal × Render × Background Jobs ×
R 4.3.2 · ~/Dropbox/Documents/GitHub/sais_r_course/ ↵
> norms_values_data %>%
+   tabyl(B_COUNTRY_ALPHA)
Error in tabyl(., B_COUNTRY_ALPHA) : could not find function "tabyl"
>
```

Basic Issues in RStudio

Function Not Found

Check:

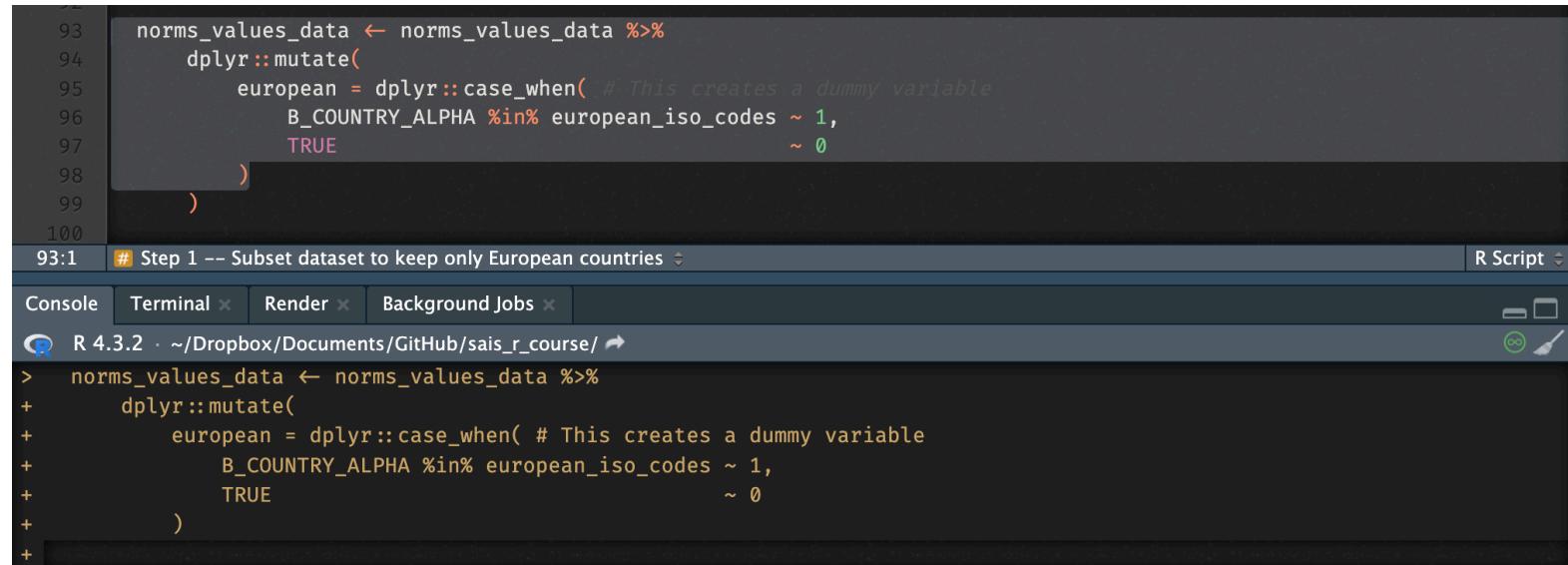
- From which package the function comes. You can do this using `?? FUNCTION NAME` (e.g. `??tabyl`) or through a Google search.
- That the package is (1) installed in your environment and (2) loaded. Having the package installed isn't sufficient!

Solutions:

- If the package isn't installed, use `install.packages("tabyl")`. If the package isn't loaded, use `library(tabyl)` or `pacman::p_load(tabyl)`.

Basic Issues in RStudio

Code Not Running



The screenshot shows the RStudio interface. The top part displays a script with line numbers 93 through 100. Lines 93-98 show the creation of a new column 'european' using dplyr::mutate(). Line 99 shows a closing parenthesis. Line 100 is a blank line. Below the script, a status bar indicates '# Step 1 -- Subset dataset to keep only European countries'. The bottom part of the interface shows a dark-themed R console window with the following text:

```
R 4.3.2 · ~/Dropbox/Documents/GitHub/sais_r_course/ ↗
> norms_values_data ← norms_values_data %>%
+   dplyr::mutate(
+     european = dplyr::case_when( # This creates a dummy variable
+       B_COUNTRY_ALPHA %in% european_iso_codes ~ 1,
+       TRUE                      ~ 0
+     )
+   )
```

Basic Issues in RStudio

Code Not Running

Check:

- That you didn't miss a parenthesis () or bracket {} in your code! This is the most common reason.
- If you missed it, the console will show a + at the start of the console line instead of the expected >.

Solutions:

- Type gibberish and/or the missing parenthesis/bracket until the > reappears. More likely, you'll have to rerun the code chunk to make sure it works!

Basic Issues in RStudio

Object Not Found

The screenshot shows the RStudio interface with two panes. The left pane is the R Script editor containing R code. The right pane is the Global Environment browser.

R Script Editor Content:

```
97     B_COUNTRY_ALPHA ~ european ~ ~ 0
98   TRUE
99   )
100
101 # Check that it worked
102
103 norms_values_data %>%
104   janitor::tabyl(european, B_COUNTRY_ALPHA)
105
106 # Now subset using filter()
107
108 european_data ← norms_values_data %>%
109   dplyr::filter(european == 1)
110
111 ### Step 2 -- Select relevant variables —
112
113 # We want to look at what people find important in life. Those are questions Q1-Q6.
114
115 # So we keep those questions, as well as D_INTERVIEW (unique ID, always keep) and B_COUNTRY_ALPHA
116
117 european_data ← european_data %>%
118   dplyr::select(
119     D_INTERVIEW, B_COUNTRY_ALPHA, dplyr::matches("^Q0[1-6]")
120     # matches() allows us to select multiple variables at once using a common string in
121     # their name
122   )
123
124:5 # Step 3 -- Clean variables
```

Global Environment Browser:

- norms_values... 84638 obs. of 60 variables

Console Tab:

```
R 4.3.2 · ~/Dropbox/Documents/GitHub/sais_r_course/
> european_data ← european_data %>%
+   dplyr::select(
+     D_INTERVIEW, B_COUNTRY_ALPHA, dplyr::matches("^Q0[1-6]")
+     # matches() allows us to select multiple variables at once using a common string in
+     # their name
+   )
Error: object 'european_data' not found
>
```

R Documentation:

Keep or drop columns using their names and types

Description

Select (and optionally rename) variables in a data frame, using a concise mini-language that makes it easy to refer to variables based on their name (e.g. `a:f` selects all columns from `a` on the left to `f` on the right) or type (e.g.

Basic Issues in RStudio

Object Not Found

Check:

- Whether the object exists in your environment. More likely than not, you either misspelt the name of the object, or you skipped the code that creates it (remember that a script is like a recipe and steps can't be skipped!).

Solutions:

- Backtrack in your code and run the chunk that creates the object.
- If a typo is at fault, correct the typo.

Practical Exercise

Using the World Values Survey Dataset

World Values Survey

Background

"The survey, which started in 1981, seeks to use the most rigorous, high-quality research designs in each country. The WVS consists of nationally representative surveys conducted in almost 100 countries which contain almost 90 percent of the world's population, using a common questionnaire. [...] WVS seeks to help scientists and policy makers understand changes in the beliefs, values and motivations of people throughout the world."

Survey Contents

- Social values, attitudes & stereotypes
- Societal well-being
- Social capital, trust and organizational membership
- Economic values
- Corruption
- Migration
- Post-materialist index
- Science & technology
- Religious values
- Security
- Ethical values & norms
- Political interest and political participation
- Political culture and political regimes
- Demography

Today's practical component

1— Download the World Values Survey (Wave 7) at this link:

<https://www.worldvaluessurvey.org/WVSDocumentationWV7.jsp>

2— Set up your data analysis folder. Make sure that it contains the following folders:

- data
 - raw
 - intermediate
- code
- output
- documentation

3— Place the data and documentation files that you find appropriate for your analysis in the corresponding folders.

4— Open RStudio and click on "New Project" in the "File" dropdown menu. Navigate to your data folder and select it.

Today's practical component

5— Set up your R Script. It should have the following components:

- Package setup. For now, have the script load the `dplyr`, `data.table`, and `skimr` packages.
- Data import. Import your data into the script using `read.csv()` or `fread()`. Remember to give your data frame an intuitive name!

6— Test your R Script. Type and run the following line in your code:

`skim([[DATASETNAME]]).`

Replace `[[DATASETNAME]]` with the name you gave your dataset. If this works without issue, you're free to take a break!

BREAK

Data Visualization – Descriptive Statistics – Plots

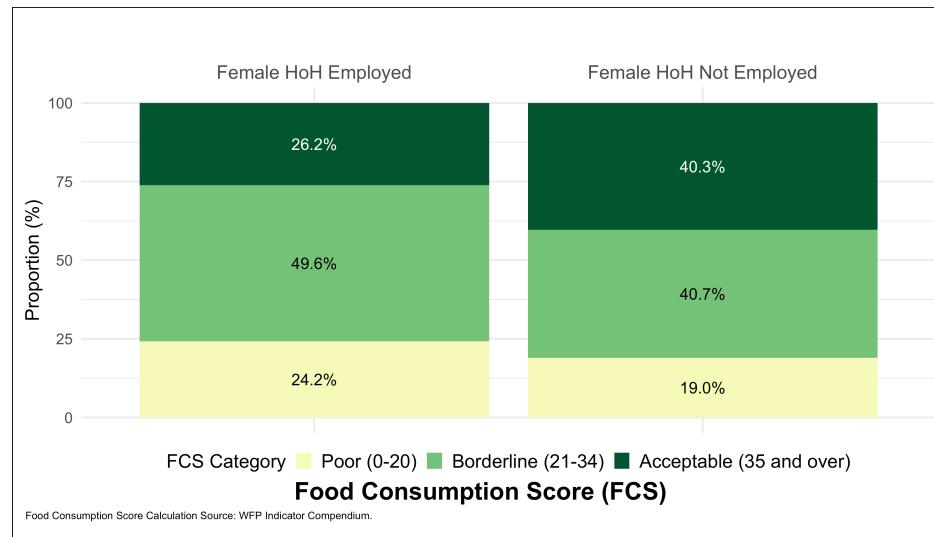
Descriptive Stats Plots

ggplot2 is the gold standard in data visualization in data work. It's one of the main reason that people use R over other programming languages.

Very simple syntax and allows you to add elements very easily.

You can use ggplot2 to create any type of plot you can think of.

I've included a lot of links at the end of these slides to explore the possibilities of ggplot2 further. Strongly recommend you use them or at least save them somewhere.



The Magic of ggplot2

Using ggplot2 to create plots is great because the **structure** it sets up makes plot creation intuitive.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(  
    mapping = aes(<MAPPINGS>),  
    stat = <STAT>,  
    position = <POSITION>  
) +  
  <SCALE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <THEME_FUNCTION>
```

1. Data: The data that you want to visualize
2. Layers: geom_ and stat_ → The geometric shapes and statistical summaries representing the data
3. Aesthetics: aes() → Aesthetic mappings of the geometric and statistical objects
4. Scales: scale_ → Maps between the data and the aesthetic dimensions
5. Facets: facet_ → The arrangement of the data into a grid of plots
6. Visual themes: theme() and theme_ → The overall visual defaults of a plot

Scatter Plot – Step-by-Step

[Dataset](#)[Convert to Plot](#)[Add Something](#)

Start with a dataset you want to visualize

```
head(mtcars)
```

```
##                                     mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160 110 3.90 2.620 16.46  0  1     4     4
## Mazda RX4 Wag       21.0   6 160 110 3.90 2.875 17.02  0  1     4     4
## Datsun 710          22.8   4 108  93 3.85 2.320 18.61  1  1     4     1
## Hornet 4 Drive      21.4   6 258 110 3.08 3.215 19.44  1  0     3     1
## Hornet Sportabout   18.7   8 360 175 3.15 3.440 17.02  0  0     3     2
## Valiant             18.1   6 225 105 2.76 3.460 20.22  1  0     3     1
```

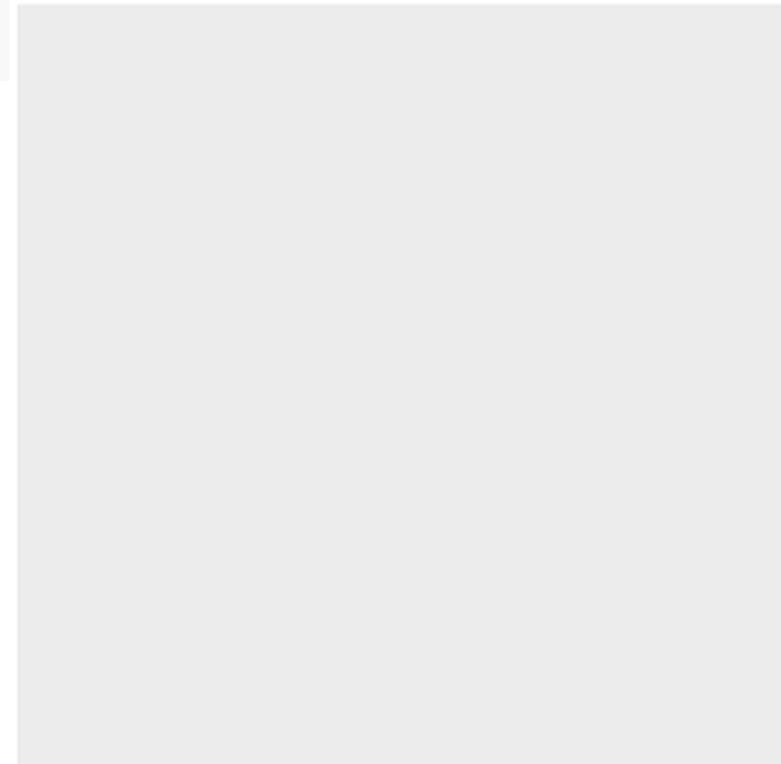
Scatter Plot – Step-by-Step

Dataset

Convert to Plot

Add Something

```
ggplot(mtcars)
```



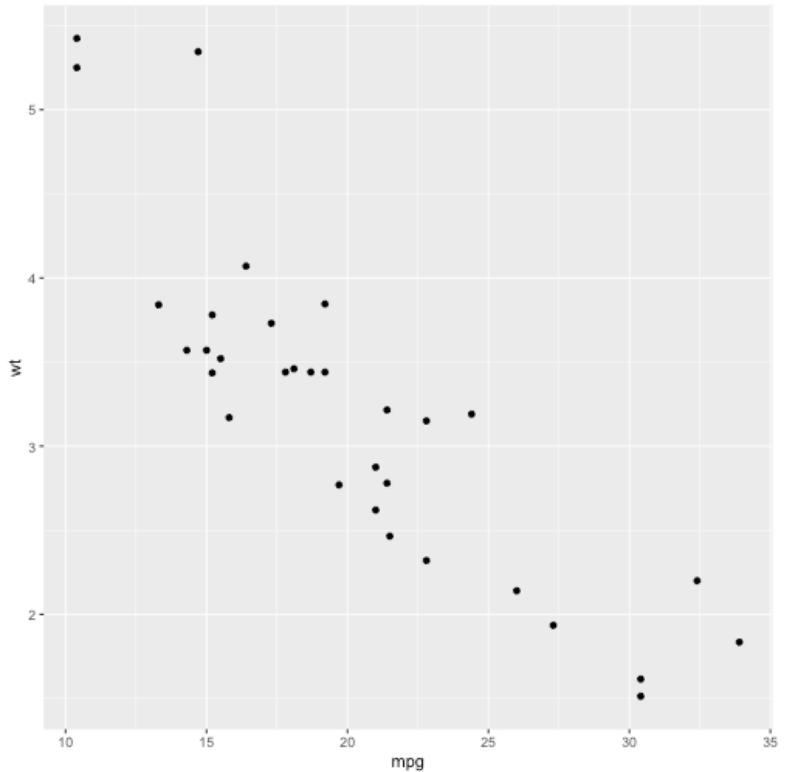
Scatter Plot – Step-by-Step

Dataset

Convert to Plot

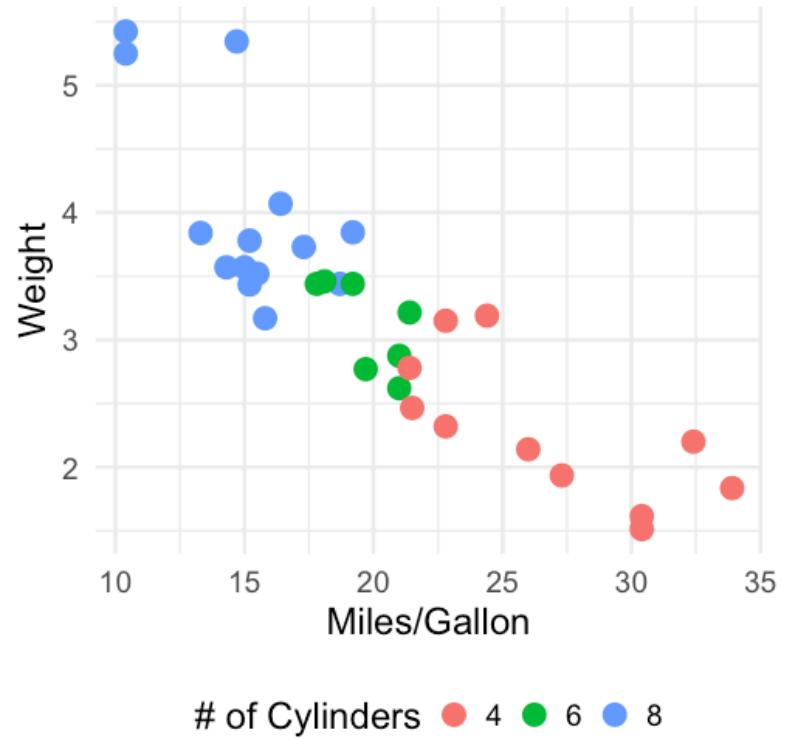
Add Something

```
ggplot(mtcars) +  
  geom_point(  
    aes(x = mpg, y = wt)  
)
```



Scatter Plot – Make It Better

```
ggplot(mtcars) +  
  geom_point(  
    aes(  
      x = mpg, y = wt,  
      color = factor(cyl)  
    ),  
    size = 6  
  ) +  
  xlab("Miles/Gallon") +  
  ylab("Weight") +  
  scale_color_discrete(  
    name = "# of Cylinders"  
  ) +  
  theme_minimal(base_size = 24) +  
  theme(  
    legend.position = "bottom"  
  )
```



Bar Plot – Step-by-Step

[Dataset](#)[Convert to Plot](#)[Add Something](#)[Fix Class Issue](#)

Start with a dataset you want to visualize

```
mtcars_summary ← mtcars %>%  
  group_by(cyl) %>%  
  summarize(  
    mpg = mean(mpg)  
  ) %>%  
  ungroup()  
  
mtcars_summary
```

```
## # A tibble: 3 × 2  
##       cyl   mpg  
##   <dbl> <dbl>  
## 1     4  26.7  
## 2     6  19.7  
## 3     8  15.1
```

Bar Plot – Step-by-Step

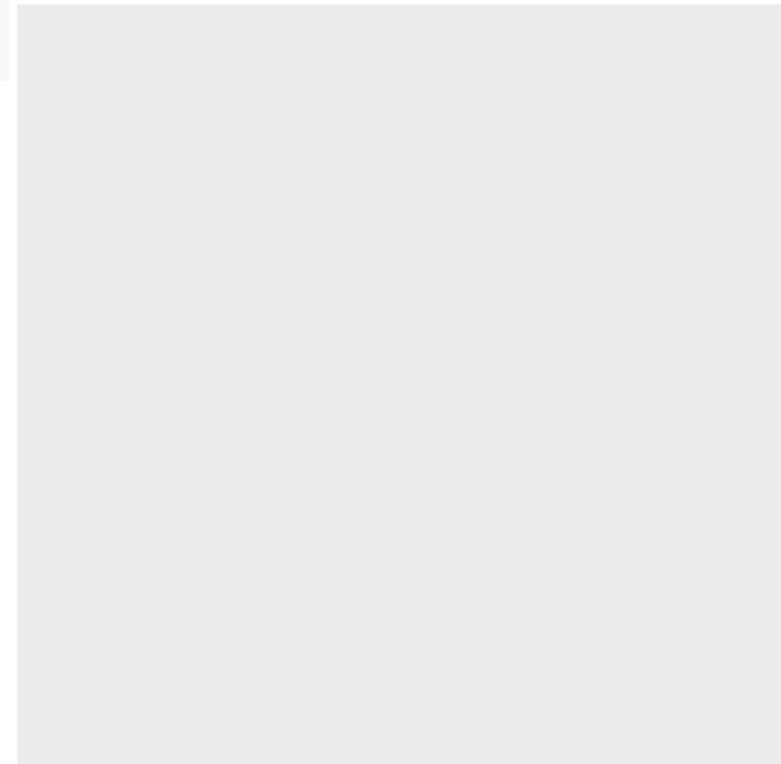
Dataset

Convert to Plot

Add Something

Fix Class Issue

```
ggplot(mtcars_summary)
```



Bar Plot – Step-by-Step

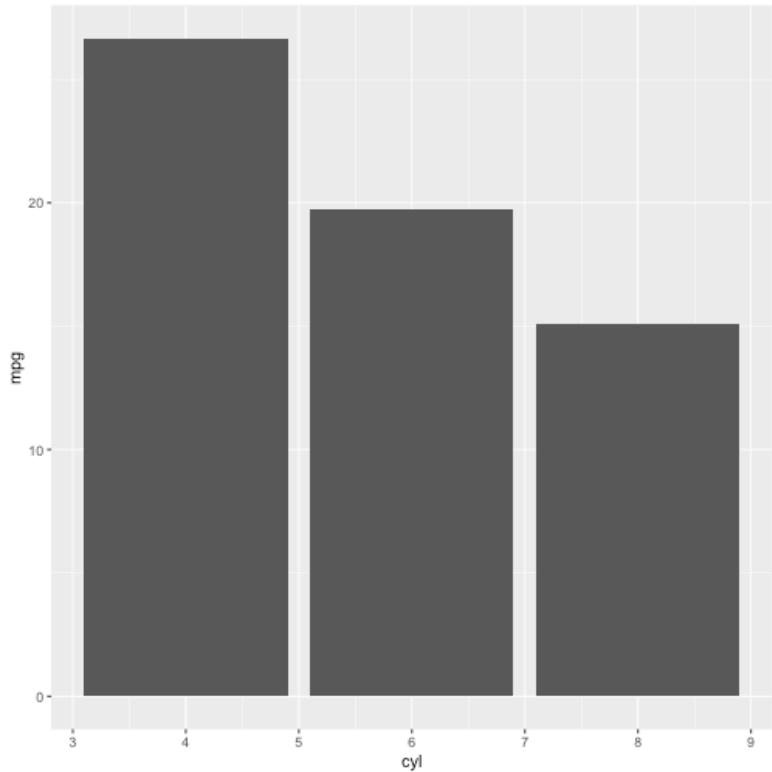
Dataset

Convert to Plot

Add Something

Fix Class Issue

```
ggplot(mtcars_summary) +  
  geom_col(  
    aes(  
      x = cyl,  
      y = mpg  
    )  
  )
```

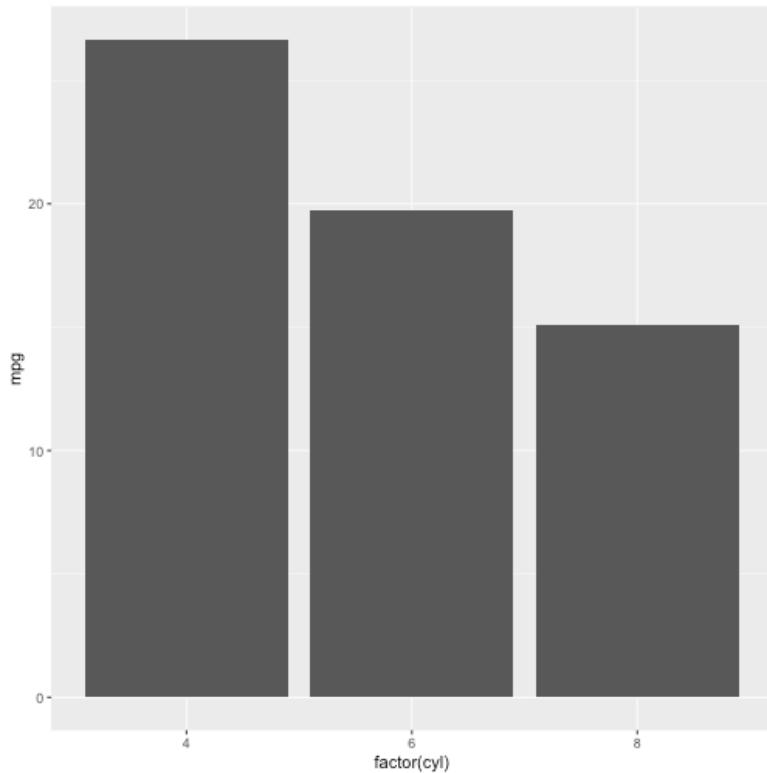


Bar Plot – Step-by-Step

[Dataset](#)[Convert to Plot](#)[Add Something](#)[Fix Class Issue](#)

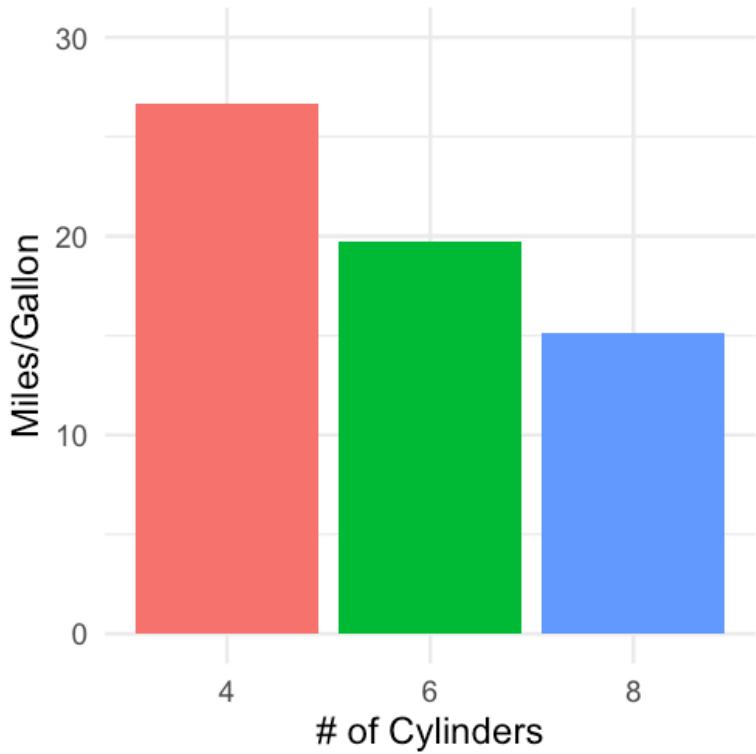
```
ggplot(mtcars_summary) +  
  geom_col(  
    aes(  
      x = factor(cyl),  
      y = mpg  
    )  
  )
```

cyl categorizes cars by number of cylinders. Although the values are numbers, it is a **categorical** variable. We communicate this to ggplot() using the factor() function.



Bar Plot – Make It Better

```
ggplot(mtcars_summary) +  
  geom_col(  
    aes(  
      x      = factor(cyl),  
      y      = mpg,  
      fill   = factor(cyl)  
    )  
  ) +  
  xlab("# of Cylinders") +  
  ylab("Miles/Gallon") +  
  scale_y_continuous(  
    limits = c(0, 30)  
  ) +  
  theme_minimal(base_size = 24) +  
  theme(  
    legend.position = "none"  
)
```



Plot Standards

1. Your plot should be **properly labeled**:

- The plot should have a title describing its content
- Axes should be labeled
- Legend (if any) should have a title and labels

2. Your plot should be **properly formatted**:

- Axis dimensions should be appropriate. What is appropriate varies depending on context, but usually you should aim to fill the plot space with data
- Text size should be large enough for text to be legible

3. Your plot should be **self contained**. People should be able to understand your plot and its data without any other context or explanatory text. That means:

- A caption note that includes data source and any important data construction notes
- Title and subtitle that deliver the plot's *message*

Data Visualization – Descriptive Statistics – Tables

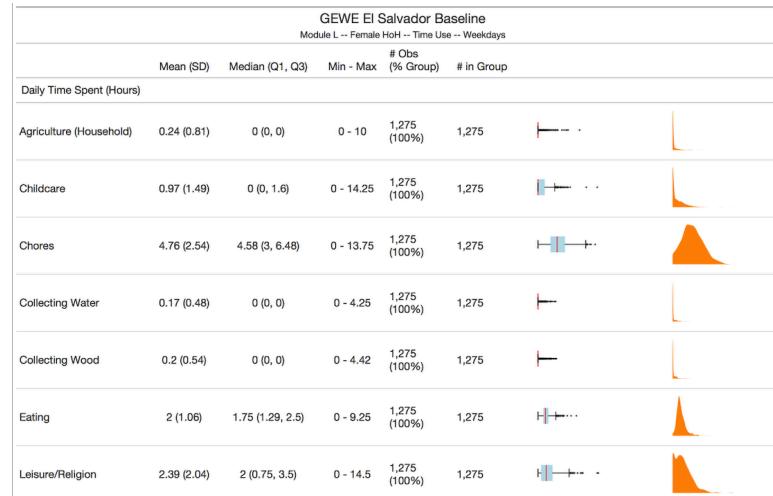
Descriptive Statistics Tables

Thankfully, not every RA position requires academic-standard tables or use of LaTeX.

It is still useful, however, to be able to communicate descriptive statistics about data.

There are countless R packages to help do this. Today, we're looking at the `gt` package. It's simple to use and it's very easy to create good-looking tables using it.

`gt` exports into `.png`, `.pdf`, or `.html`. You can add interactive elements, plots within columns.



Descriptive Statistics Table – Step-by-Step

We will mainly use the example in the script for this. To summarize, the steps are:

- Create a dataset you want to export
- Run the dataset through the `gt()` function to create a `gt` object
- Customize the table using functions from the `gt` package (see online for further things you can do). Examples of what you can do include:
 - Modify column names – `cols_label()`
 - Modify borders – `tab_style()`, `cell_borders()`
 - Add colors conditional on cell value – `data_color()`
 - Add title/subtitle – `tab_header()`
- Export the table using `gtsave()`

Data Visualization — Simple Regression Table

Regression Tables

Regression tables are very common in economic/policy analysis.

They're very simple to create using R and a software called **LateX** (pronounced latek).

Unless you're getting into academic research, you don't need to know how to properly use LateX. Just enough to:

- Export the LateX script from R
- Copy/paste it into a LateX-reading software, e.g. Overleaf
- Export the pdf or png to share

Predicted Consumption per Capita (2019 PPP USD)

	Any Treatment vs. Control (1)	Women Working Treatment vs. Any Treatment (2)
Any Treatment	12.049** (5.330)	12.155* (6.600)
Women Working Treatment		-0.222 (8.463)
Baseline Control	0.249** (0.101)	0.249** (0.101)
Constant	22.788*** (3.483)	22.791*** (3.489)
Control Mean	27.91	27.91
Observations	761	761
R ²	0.028	0.028
Adjusted R ²	0.025	0.024
Residual Std. Error	44.983 (df = 758)	45.013 (df = 757)
F Statistic	10.925*** (df = 2; 758)	7.275*** (df = 3; 757)

Note:

*p<0.1; **p<0.05; ***p<0.01

Regression Table – Step by Step

Run Regression in R

Convert to Exportable Table

```
# Simplest regression format in R

reg_example <- lm(
  outcome_variable ~ independent_variable + control_variables,
  data = dataset
)

# Observe results

reg_example %>% summary()
```

Regression Table – Step by Step

Run Regression in R

Convert to Exportable Table

Simply do one of these!

```
reg_example_ht ← reg_example %>%  
  huxtable::huxreg()
```

OR

```
reg_example_sg ← reg_example %>%  
  stargazer::stargazer() # Many options to make prettier
```

Regression Table – Step by Step

[Export Huxtable Table](#)[Export Stargazer Table](#)

Some simple options for the Huxtable table:

```
huxtable::quick_latex(  
  reg_example_ht,  
  file = "filepath/filepath/filepath/reg_example_ht.tex"  
)  
  
huxtable::quick_pdf  
  reg_example_ht,  
  file = "filepath/filepath/filepath/reg_example_ht.pdf"  
)  
  
huxtable::quick_html(  
  reg_example_ht,  
  file = "filepath/filepath/filepath/reg_example_ht.html"  
)
```

Regression Table – Step by Step

Export Huxtable Table

Export Stargazer Table

```
# You can export a LaTeX script using the 'writeLines' function  
writeLines(  
  reg_example_sg,  
  "filepath/filepath/filepath/reg_example_sg.tex"  
)
```

To visualize your table, the easiest solution is to:

- Create a free Overleaf account on overleaf.com
- Open a new document
- Copy/paste your .tex output in between the begin{document} and end{document} lines
- Click compile and then save!

You can also install the `tinytex` package and use `pdftotex` to save a PDF file.

Wednesday

- Practice visualizations
- Learn how to wrangle data

Links

Thomas Mock, “A Gentle Introduction to Tidy Statistics in R” ([blog post](#) and [video](#))

Hadley Wickham, Mine Çetinkaya-Rundel & Garrett Grolemund, [R for Data Science \(2e\)](#)

RStudio, [RStudio Cheatsheets](#)

Links

Tables

Marek Hlavac, “[stargazer: beautiful LATEX, HTML and ASCII tables from R statistical output](#)”

Thomas Mock, “[gt - a \(G\)rammar of \(T\)ables](#)”

Plots

Alicia Horsch, “[A quick introduction to ggplot2](#)”

RStudio, [RStudio Cheatsheets](#)