Introduction to R Markdown, `ggplot`, & `tidyverse`

Amanda Loehrke

University of California, Davis
Department of Political Science

ICPSR 2025

Today's goals

- Introduce:
  - R Markdown file type
  - Key functions in the `tidyverse` package
  - The `ggplot2` package
- Example code & plot output

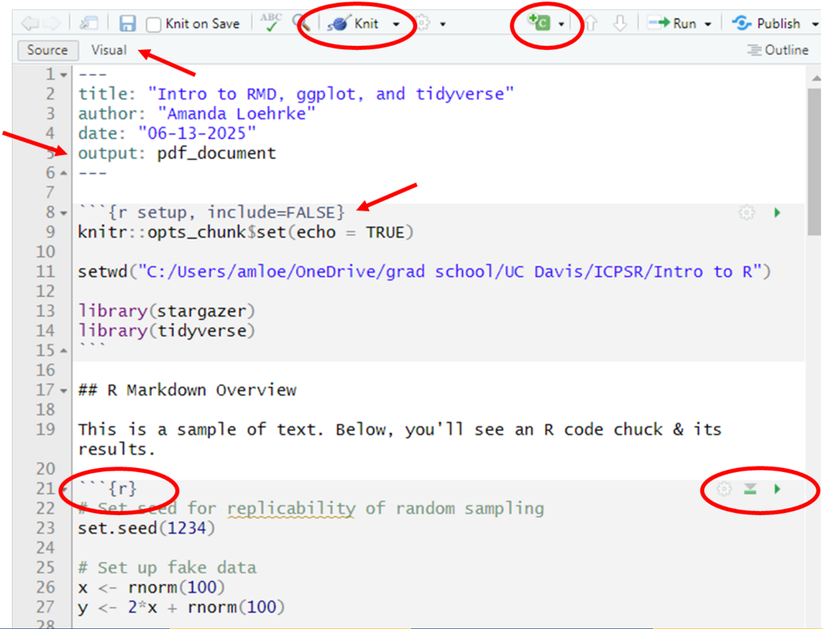
What is R Markdown?

- A combination of text and code saved as a .Rmd file
- Uses plain text (with some formatting instructions) plus chunks of code
- Customizable to include text, code, and/or output
- Can render to PDF, HTML, Word, etc.
- Many, many applications

RMD workflow

- Open a new .Rmd file
- Select the output you want to render, add title, and save file to your working directory
- Write text, titles, section headings, etc.
- Embed code
- Choose output formatting options
- Save & render (to PDF, HTML, etc.)

R Markdown Basics



Example output (PDF)

Intro to RMD, ggplot, and tidyverse

Amanda Loehrke

06-13-2025

R Markdown Overview

This is a sample of text. Below, you'll see an R code chuck & its results.

```
# Set seed for replicability of random sampling
set.seed(1234)

# Set up fake data
x <- rnorm(100)
y <- 2*x + rnorm(100)

# Make a dataframe
df <- data.frame(x, y)

# Run a model
model <- lm(y ~ x, df)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Residuals:
```

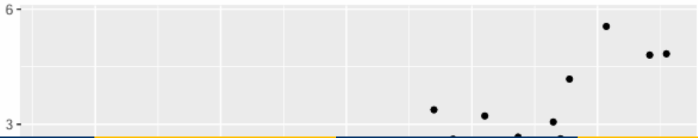
Table 1:

	Dependent variable:
	y
x	1.974^{***}
	(0.104)
Constant	0.037
	(0.105)
Observations	100
R^2	0.787
Adjusted R^2	0.785
Residual Std. Error	1.037 (df = 98)
F Statistic	361.796^{***} (df = 1; 98)
Note:	*p<0.1; **p<0.05; ***p<0.01

Including Plots

You can also embed plots, for example:

```r
ggplot(df, aes(x, y)) +
  geom_point()
```



Text formatting (basics)

Code:

```
# Main title
## Section title
### Subsection title

*italic text*

**bold text**

* This
* Is
* A List

1. This
2. Is A
3. Numbered list
```

Output:

Main title

Section title

Subsection title

italic text

bold text

- This
- Is
- A List

1. This
2. Is A
3. Numbered list

Code chunks (basics)

````
```{r}

```
````

There are default options for a code chunk; some common options to change:

- `include = FALSE` no code or results will render
- `echo = FALSE` results will render, code will not
- `message = FALSE` no code generated messages will render
- `warning = FALSE` no code generated messages will render
- `error = TRUE` display error message in rendered doc

Other options

- Equations with TeX math mode
  ```
  $Y = \beta_1 X_1 + \beta_2 X_2 + \alpha + \epsilon$
  ```
  - $Y = \beta_1 X_1 + \beta_2 X_2 + \alpha + \epsilon$
- Bibliographies
  - Can add BibTex, Zotero, DOI citations and bibliography
- Output tables
  - Packages like `knitr`, `stargazer`, `flextable`, `kableExtra`
- Insert a url: < `url` >
- And much, much more!

Introduction to the `tidyverse`

- This big package includes many important packages like `ggplot2`, `tidyr`, and `dplyr`
- Useful for data wrangling, combining functions, and visualization

Data wrangling with `dplyr`

- `filter()`
  - Subset the data, select rows that meet certain criteria
- `select()`
  - Choose certain variables to keep
- `group_by()`
  - Prep data for grouped operations/summaries

Wrangling, continued

- `case_when()`
  - Create conditional values (like nested `ifelse()`)
- `mutate()`
  - Add or modify variables
- `summarize()`
  - Calculate summary statistics for one or more variables
- Pipes %>%
  - For a sequence of operations
  - A useful Base R function used here, too: `%in%`

Example of `filter()`, `select()`, and pipes

- Say we want to subset the ANES data to be four feeling thermometers for people in the South:

```
## Filter and select functions
# Without pipes
anes.filtered <- filter(anes.data, region == "South")

anes.filtered <- filter(anes.filtered, pid7 != 8)

anes.filtered <- select(anes.filtered, cand_biden,
                        cand_trump, group_nra, group_blm, pid7)

variable.names(anes.filtered)
head(anes.filtered)
```

Example continued

```
# Now, with pipes
anes.filtered.pipes <- anes.data %>%
  filter(region == "South") %>%
  filter(pid7!= 8) %>%
  select(cand_biden, cand_trump, group_nra, group_blm, pid7)
```

```
# Or this way (just combine two filter arguments into one)
anes.filtered.pipes <- anes.data %>%
  filter(region == "South", pid7!= 8) %>%
  select(cand_biden, cand_trump, group_nra, group_blm, pid7)

variable.names(anes.filtered.pipes)
head(anes.filtered.pipes)
```

Example of `group_by()`, `mutate()`, and `summarize()`

- Say we want the average Trump and Biden Feeling thermometer rating by region (Northeast, South, Midwest, West)
- We need to clean the data (get rid of "No Answer" and "not asked" rows), break up the data by region, and calculate the average feeling thermometer ratings for each region
- To do this with Base R:

Base R Example

```
# In base R:
anes.data$cand_biden_clean <- ifelse(anes.data$cand_biden == "No Answer", NA,
                                     ifelse(anes.data$cand_biden == "not asked", NA,
anes.data$cand_biden))

anes.data$cand_biden_clean <- as.numeric(anes.data$cand_biden_clean)

anes.data$cand_trump_clean <- ifelse(anes.data$cand_trump == "No Answer", NA,
                                     ifelse(anes.data$cand_trump == "not asked", NA,
anes.data$cand_trump))

anes.data$cand_trump_clean <- as.numeric(anes.data$cand_trump_clean)

# Check cleaned data
table(anes.data$cand_biden_clean)
table(anes.data$cand_trump_clean)

# Find average ft score by region
aggregate(cbind(cand_biden_clean, cand_trump_clean) ~ region,
          data = anes.data,
          FUN = mean,
          na.rm = TRUE)
```

tidyverse example

```
# With dplyr:
anes.clean <- anes.data %>%
  filter(!cand_biden %in% c("No Answer", "not asked")) %>%
  filter(!cand_trump %in% c("No Answer", "not asked")) %>%
  mutate(cand_biden = as.numeric(cand_biden),
         cand_trump = as.numeric(cand_trump))

# Check cleaned data
table(anes.clean$cand_biden)
table(anes.clean$cand_trump)

# Find average ft score by region
anes.clean %>%
  group_by(region) %>%
  summarize(avg.bidenft = mean(cand_biden),
            avg.trumpft = mean(cand_trump))
```

R output from each example

- With Base R:

region <chr>	cand_biden_clean <dbl>	cand_trump_clean <dbl>
Midwest	35.00522	45.06005
Northeast	46.01767	43.95053
South	40.81639	46.39302
West	45.70833	40.72135

- With `tidyverse`:

region <chr>	avg.bidenft <dbl>	avg.trumpft <dbl>
Midwest	35.00522	45.06005
Northeast	46.01767	43.95053
South	40.81639	46.39302
West	45.70833	40.72135

The ggplot2 package

- "Grammar of graphics"
- Think about grammar in language: it is a set of rules that govern how language is structured (words, phrases, clauses)
- Used to produce layered, statistical graphics
- Uses a "grammar" to build graphs layer-by-layer
- For more package information, click here.

Grammar of graphics (basics)

- **Data**: what information to map
- **Aesthetics**: which data is on your x-axis? y-axis? color? size? fill?
- **Geometries**: the type of plot you want to create (scatterplot, bar plot, time series, etc.)
- **Scales**: mappings of aesthetic values to data values
- **Stats**: transformations with statistics to summarize data
- **Faceting**: splitting data to create multiple variations of the same graph

Using `ggplot()`

- All graphics begin with the `ggplot()` function
- In this function, we specify the dataset and variables to map to the aesthetics

  ```
  ggplot(data, aes(x = variable1, y = variable2)
  ```
- **Data**: name of the data frame object that holds the variables we want to plot
- **X and Y**: aesthetics that position objects on the graph
- Running this will produce a plot with x and y axes, no data will be plotted

Aesthetics

- Commonly used aesthetics:
  - **x**: positioning on the x-axis
  - **y**: positioning on the y-axis
  - **color**: color of the data objects
  - **fill**: fill color of points, boxes, etc.
  - **linetype**: solid, dashed, dotted, etc.
  - **shape**: circle, square, solid, etc.
  - **size**: how large objects appear
  - **alpha**: transparency of objects

Using `ggplot()` + `geom_*`

- We add layers with + and our next layer will be the geometry
  `ggplot(data, aes(x)) + geom_*()`
- You'll specify the type of geometry you want, based on the type of plot you want to create (scatterplot, histogram, barplot, etc.)
- The geometry will take the mapping from `ggplot()` but can also be overwritten in `geom_*(aes(variable))`
  - Can specify line type, colors, sizes, etc.

Geometries

- * There are many geometries available, some common ones are:
  - `geom_histogram()`
  - `geom_density()`
  - `geom_boxplot()`
  - `geom_bar()`
  - `geom_point()`
  - `geom_line()`

Using the stats layer

- Can map on statistical functions to transform data, just add a layer with $+$
- Each stat function is associated with a geometry, so they can be used without specifying a geometry
- Ex) `stat_summary()` will give a point for the mean and error bars
- Many ways to use this layer depending on what you are trying to do!

Using the scales layer

- Scales define which aesthetic values are mapped to the data values
- This layer is added to `ggplot()` and the geometry layer

```
ggplot(data, aes(x)) +
    geom_bar() +
    scale_*()
```

- Many different scaling layers depending on what you are trying to accomplish

Scales

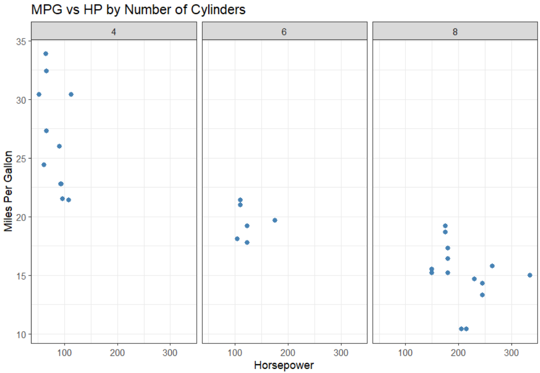
- Specify that you are using scale, then name the primary aesthetic (x, y, color, etc.), then name the scale you want to use (continuous, discrete, etc.)
- Some common scale specifications
  - `scale_x_continuous()`
  - `scale_y_continuous()`
  - `scale_color_manual()`
- Axis scales specify the breaks, labels, and names

More on axes, limits, and labeling

- Can also control the limits and titles of the axes using shortcut functions instead of scaling
- Some examples:
  - `lims()`: set axis limits
  - `xlim()`: set x axis limit
  - `ylim()`: set y axis limit
  - `xlab()`: label x axis
  - `ylab()`: label y axis
  - `ggtitle()`: label title
  - Or, use `lab()` to specify each axis label, title, subtitle, etc.
    ```
    lab(title = "Insert your title here", x = "X axis label", y = "Y axis label")
    ```

Faceting

- Used to split plots into multiple panels
- `facet_wrap()` and `facet_grid()`



Themes and colors

- Themes control the parts of the graph not related to the data (background color, font size, gridlines, label colors, etc.)
- Can set one theme for the entire plot (complete themes)
- Can modify specific components of a theme (modify components)
- There are various default colors in R, for a full list, click here

Interactive plots

- You can make interactive charts with ggplot
- Also uses the package `plotly`
- Zoom, hover, etc.
- This website is a great introduction to interactive plots in R
- Or this website that introduces dynamic, interactive plots with the `ggiraph` package

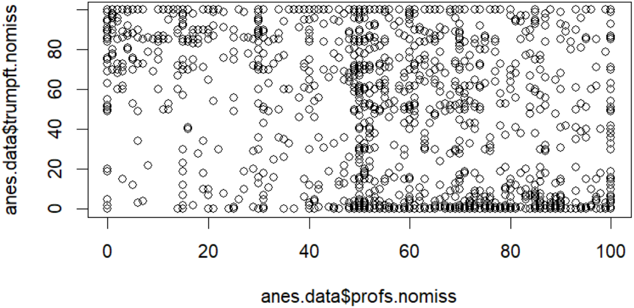
Saving plots to files

1. Export from the Plots tab in RStudio
2. Right-click on an image and save it
3. Use `ggsave()` in your code to save the plot object to your working directory
   - Easy to change the default width, height, filename, etc.

```
myplot <- ggplot(data, aes(x)) + geom_hist()
ggsave("myplot.png", plot = myplot, width = 10,
height = 6)
```

Example scatterplot

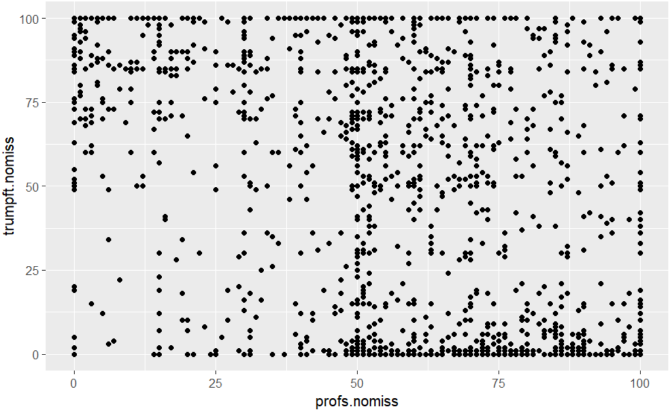
- Remember when we made a scatterplot with `profs.nomiss` and `trumpft.nomiss`? It looked like this in Base R:



- We can made a similar plot using `ggplot`

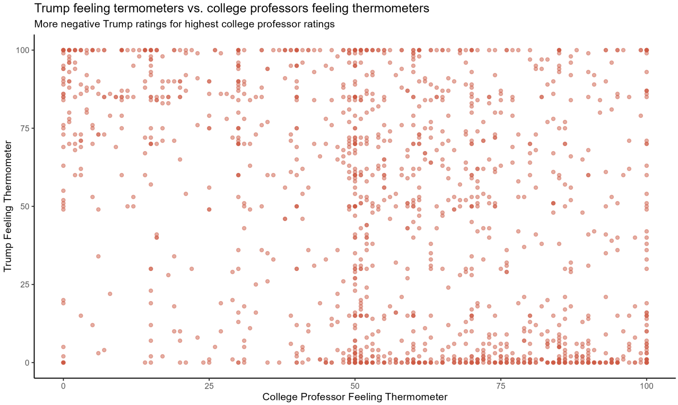
Scatterplot example default

```
# Plot with tidyverse
anes.data <- anes.data %>%
  mutate(profs.nomiss = ifelse(group_colprofs %in% c(999, -7), NA, group_colprofs),
         trumpft.nomiss = ifelse(cand_trump %in% c(999, -7), NA, cand_trump))

ggplot(anes.data, aes(x = profs.nomiss, y = trumpft.nomiss)) +
  geom_point()
```



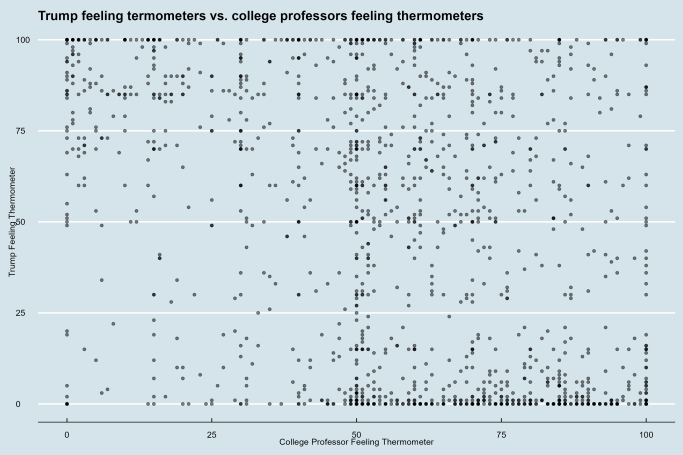
Scatterplot example stylized

```
prof.trump.plot <- ggplot(anes.data, aes(x = profs.nomiss, y = trumpft.nomiss)) +
  geom_point(alpha = 0.5, color = "coral3") +
  labs(title = "Trump feeling termometers vs. college professors feeling thermometers",
       subtitle = "More negative Trump ratings for highest college professor ratings",
       x = "College Professor Feeling Thermometer",
       y = "Trump Feeling Thermometer") +
  theme_classic()

ggsave("ggplot_ft.png", plot = prof.trump.plot, width = 10, height = 6)
```



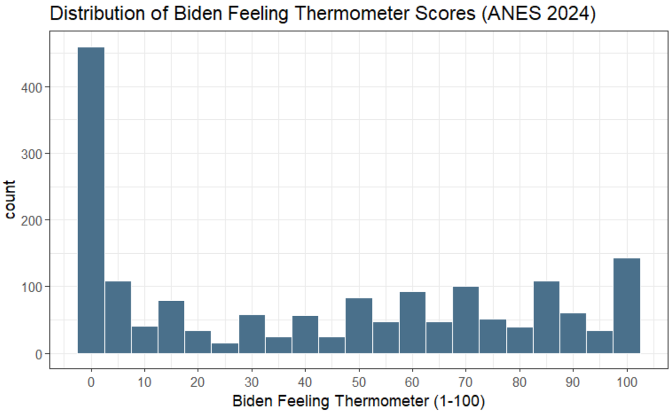
Scatterplot example, continued

```
econ.plot <- ggplot(anes.data, aes(x = profs.nomiss, y = trumpft.nomiss)) +
  geom_point(alpha = 0.5) +
  labs(title = "Trump feeling termometers vs. college professors feeling thermometers",
       x = "College Professor Feeling Thermometer",
       y = "Trump Feeling Thermometer") +
  theme_economist() +
  scale_color_economist()
```



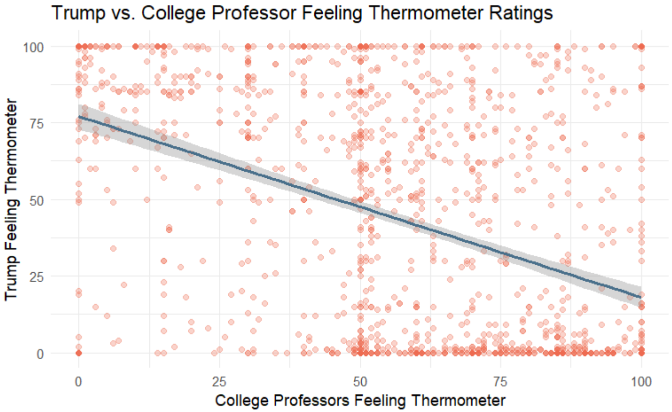
Histogram example

```
# Histogram of Biden feeling thermometer
ggplot(anes.clean, aes(cand_biden_clean)) +
  geom_histogram(binwidth = 5, fill = "skyblue4", color = "white") +
  labs(title = "Distribution of Biden Feeling Thermometer Scores (ANES 2024)",
       x = "Biden Feeling Thermometer (1-100)") +
  scale_x_continuous(breaks = seq(0, 100, 10)) +
  theme_bw()
```



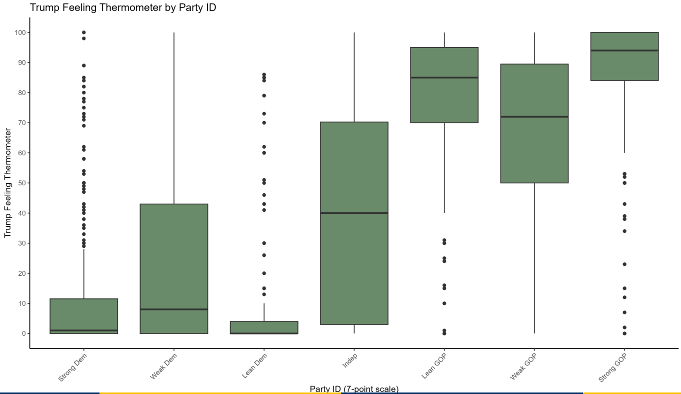
Regression line scatter plot example

```
ggplot(anes.clean, aes(x = group_colprofs, y = cand_trump)) +
  geom_point(alpha = 0.3, color = "coral2") +
  geom_smooth(method = "lm", se = TRUE, color = "skyblue4", size = 1) +
  labs(title = "Trump vs. College Professor Feeling Thermometer Ratings",
       x = "College Professors Feeling Thermometer",
       y = "Trump Feeling Thermometer") +
  theme_minimal()
```



Box plot example

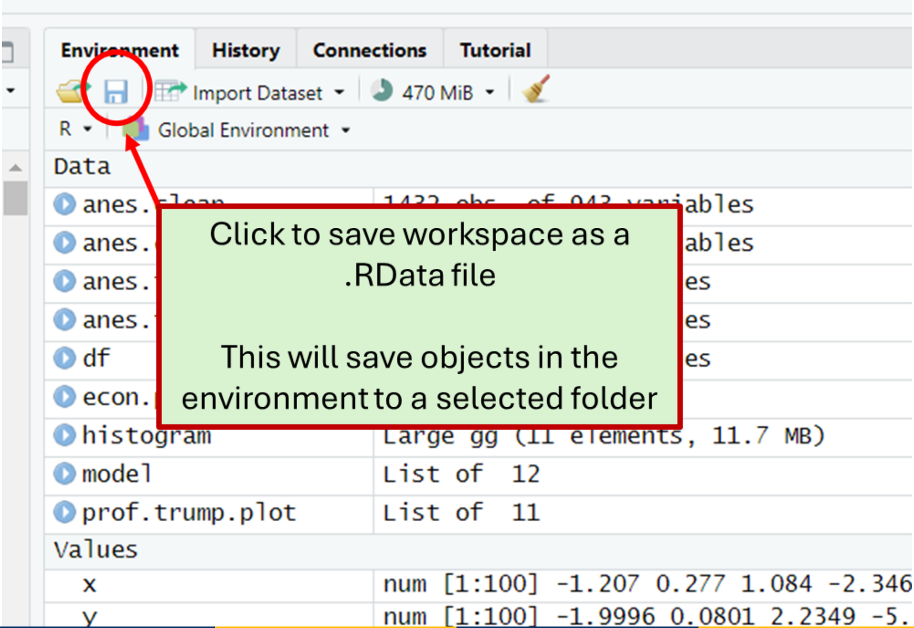
```
histogram <- ggplot(anes.clean, aes(x = pid7, y = cand_trump)) +
  geom_boxplot(fill = "darkseagreen4") +
  labs(title = "Trump Feeling Thermometer by Party ID",
       x = "Party ID (7-point scale)",
       y = "Trump Feeling Thermometer") +
  scale_y_continuous(breaks = seq(0, 100, 10)) +
  scale_x_discrete(labels = c("Strong Dem", "Weak Dem", "Lean Dem", "Indep",
"Lean GOP", "Weak GOP", "Strong GOP")) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

ggsave("histogram_example.png", plot = histogram, width = 12)
```



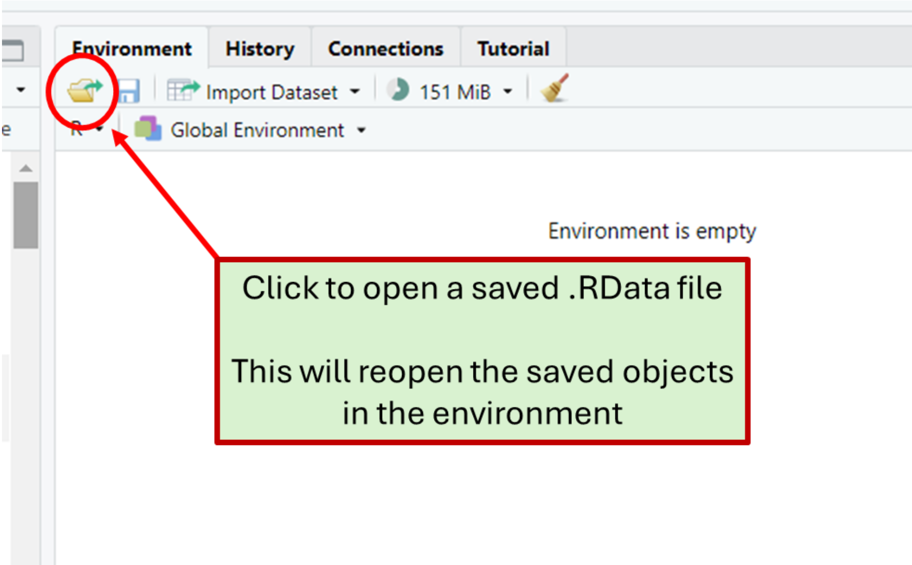
Some `ggplot` considerations

- You can make a plot do (probably) anything you want it to
- Think:
  - What dataset am I plotting from?
  - What variables do I want to visualize?
  - What type of plot do I want to create?
  - What I can do to make the plot look nicer?
    - Labels, colors, scaling, etc.
- Try different styles and layers until you produce a clear, professional-style plot that presents the data accurately & meaningfully

Saving environment (.RData)



Opening environment (.RData)



Opening RMD file in R Studio

- Open the file on Canvas, save in folder, open in R Studio
- Set your working directory (Session → Set Working Directory → Choose Directory)
- Download `tinytex` package in order to render to a PDF file

```
install.packages("tinytex")
tinytex::install_tinytex()
```

- Make sure `anes_pilot_2024_20240319.csv` and `anes_pilot_2024_20240319.dta` are in your working directory

Final thoughts

- When in doubt, ask questions!
- There are endless resources online and in print about R, R Studio, R Markdown, specific packages, etc.
- Talk to colleagues, instructors, TAs, etc.
- Coding in R is like learning a new language, so it takes practice

(Some) Resources

- Data Analysis for Social Science (Llaudet & Imai)
- R Workflow
- R Markdown, RMD Cheatsheet
- `ggplot2`
- Tidyverse & more
- DataCamp

Thank you! Feel free to send me an email with any questions/comments!

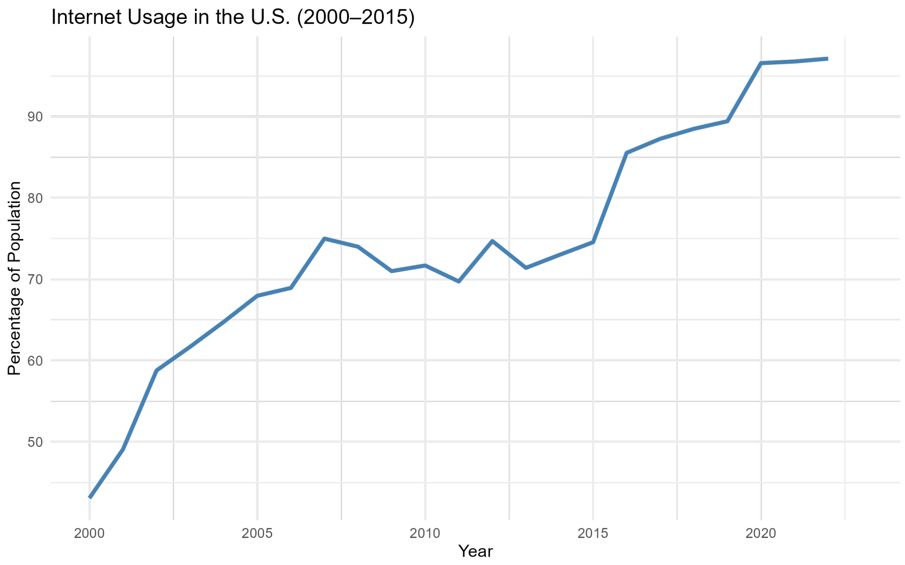
aloehrke@umich.edu
aloehrke@ucdavis.edu

Appendix

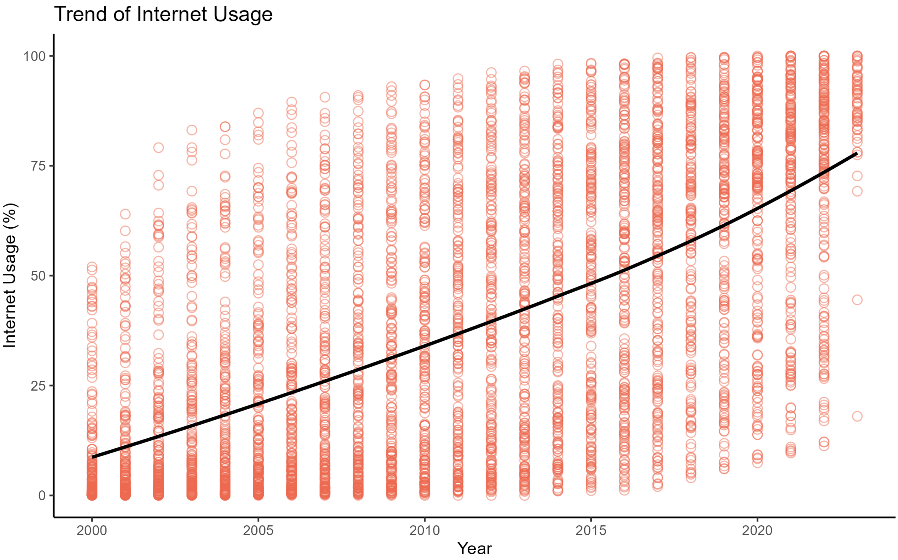
(1) Example plots
(2) Intro to R and R Studio

Examples

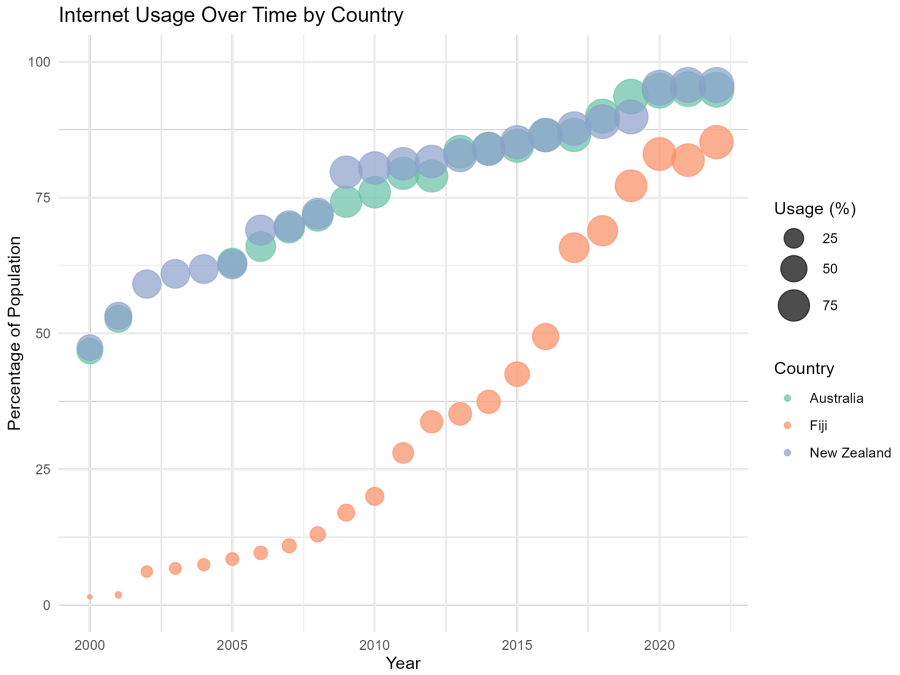
```
# Time series for US internet usage
us <- df %>% filter(Country.Name == "United States")

timeseries <- ggplot(us, aes(x = Year, y = Percentage)) +
  geom_line(color = "steelblue", size = 1.2) +
  labs(
    title = "Internet Usage in the U.S. (2000-2015)",
    x = "Year",
    y = "Percentage of Population"
  ) +
  theme_minimal()

timeseries

ggsave("timeseries.png", width = 8, height = 5, plot = timeseries)
```


Internet Usage in the U.S. (2000–2015)
Percentage of Population
90
80
70
60
50
2000
2005
2010
2015
2020
Year

```r
# Scatterplot with trendline
scatter <- ggplot(df, aes(x = Year, y = Percentage)) +
  geom_point(color = "coral2", size = 2.5, shape = 1, alpha = 0.5) +
  geom_smooth(method = "loess", color = "black", se = FALSE) +
  labs(
    title = "Trend of Internet Usage",
    x = "Year",
    y = "Internet Usage (%)"
  ) +
  theme_classic()

scatter

ggsave("scatter.png", width = 8, height = 5, plot = scatter)
```


Trend of Internet Usage
Internet Usage (%)
100
75
50
25
0
2000
2005
2010
2015
2020
Year

```r
# Bubble plot for usage over time, three countries
clean <- df %>%
  filter(Country.Name %in% c("Australia", "New Zealand", "Fiji")) %>%
  na.omit()

bubble <- ggplot(clean, aes(x = Year, y = Percentage,
                            color = Country.Name, size = Percentage)) +
  geom_point(alpha = 0.7) +
  scale_size(range = c(1, 10), name = "Usage (%)") +
  scale_x_continuous(by = 5) +
  scale_color_brewer(palette = "Set2") +
labs(
    title = "Internet Usage Over Time by Country",
    x = "Year",
    y = "Percentage of Population",
    color = "Country"
  ) +
  theme_minimal()

bubble

ggsave("bubble.png", width = 8, height = 6, plot = bubble)
```


Internet Usage Over Time by Country
Percentage of Population
100
75
50
25
0
2000
2005
2010
2015
2020
Year
Usage (%)
25
50
75
Country
Australia
Fiji
New Zealand

R and RStudio

- RStudio is a user-friendly graphical interface for R
- Open source, packages for almost everything (data processing, cleaning, visualization, etc.)
- For help, click here.

Downloading R and RStudio

- Downloading R
  - R for Windows: https://cran.r-project.org/bin/windows/base/
  - R for macOS: https://cran.r-project.org/bin/macosx/
- Downloading R Studio
  - RStudio: https://posit.co/downloads/
- Open the downloaded file (check your Downloads folder), click yes through all of the prompts to install like any other program

Downloading R and RStudio

- Downloading R
  - R for Windows: https://cran.r-project.org/bin/windows/base/
  - R for macOS: https://cran.r-project.org/bin/macosx/
- Downloading R Studio
  - RStudio: https://posit.co/downloads/
- Open the downloaded file (check your Downloads folder), click yes through all of the prompts to install like any other program

- *If* your computer is an older model, or you are using an iPad or similar, you can access RStudio online:
  - Go to https://posit.cloud/
  - Create an account using your UC Davis email
  - Click on "New Project" and select "New Project from Template"
  - Select "Data Analysis in R with the Tidyverse and click "OK"

Getting Started With RStudio

- Create a folder on your computer where you want to save your files
  - This is where you will store your R files, download data to use in R, saved visualization, tables, etc.
- Open R Studio
  - Create a new R script and save to your folder
  - Set working directory to your folder in R script
    - It will look something like this:

```
setwd("C:/Desktop/R Folder")
getwd()
```

- For a helpful RStudio tutorial, click here.

More on R Studio