

Introduction to R, R Studio, and ggplot2

Amanda Loehrke

University of California, Davis

2025

Contents

- R and R Studio
 - ▶ Introduction
 - ▶ Downloading and installing programs
- ggplot2 package
 - ▶ Aesthetics
 - ▶ Geometries
 - ▶ Stats
 - ▶ Scales
 - ▶ Labeling
 - ▶ Faceting
 - ▶ Themes
 - ▶ Saving plots
- Example code & plot output

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 POL 114 folder on your computer
 - ▶ This is where you will store your R files, download data to use in R, etc.
- Open R Studio
 - ▶ Create a new R script and save to POL 114 folder
 - ▶ Set working directory to your POL 114 folder in R script
 - ★ It will look something like this:

```
setwd("C:/Desktop/POL 114")  
getwd()
```
- For a helpful RStudio tutorial, click [here](#).

More on R Studio

1. script/source
This is where you write and edit code.
To evaluate your code, you have to click "Run" to evaluate them in the console.

2. environment
In here you can see the objects in your working space (environment) or view your command history (history)

3. console
This is where the script code is evaluated by R. You can also perform calculations, etc. in here that you don't need to save in your script.

4. files/plots/packages/help
Here you can see file directories, view plots you produce, install and update packages, and access R help.
R help is very useful tool! If you need help you can search for a function or type "?functionname" in the console to see that function's help page.

Your Workspace / eat-cake.R

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

```
1 library(tidyverse)
2 library(unvotes)
3 library(lubridate)
4 library(scales)
5
6 # Make a plot
7
8 # calculate % votes yes each year by country by issue
9
10 un_yes = un_votes %>%
11   filter(country == "United States") %>%
12   inner_join(un_votes, by = "year") %>%
13   inner_join(un_votes, by = "issue") %>%
14   group_by(country, year, issue) %>%
15   summarize(
16     votes = n(),
17     percent_yes = votes / n()
18   ) %>%
19   filter(votes > 5) # only use records where there are more than 5 votes
20
21 # make plot
22 ggplot(un_yes, aes(x = year, y = percent_yes, color = country)) +
23   facet_wrap(~ issue)
24
25 # Make a plot
```

Environment History Connections Tutorial

R 4.3.3

Data

un_yes

657 obs. of 5 variables

Files Plots Packages Help Viewer Presentation

Folder Blank File Upload Delete Rename

Cloud project

Rhistory 0.8 Aug 8, 2023, 5:35 PM

installing "binary" package

DONE (remotes)

The downloaded source file is located at: /tmp/Rtmpq6w2/00/downloaded_packages/

```
> # make plot
> ggplot(un_yes, aes(x = year, y = percent_yes, color = country)) +
+   geom_point() +
+   geom_smooth(method = "lm") +
+   facet_wrap(~ issue) +
+   labs(
+     title = "Percent Yes by Year and Country",
+     subtitle = "1946-2014",
+     y = "% Yes",
+     x = "Year",
+     color = "Country"
+   ) +
+   scale_y_continuous(labels = percent)
+   geom_smooth() using formula = 'y ~ x'
```

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 - the 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 - `aes()`
`ggplot(data, aes(x = variable1, y = variable2))`
- **Data**: name of the data frame that holds the variables we want to plot
- **X and Y**: aesthetics that position objects on the graph
- We would specify then `variable1` and `variable2`
- 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 objects
 - ▶ **fill**: fill color of objects
 - ▶ **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))`

Geometries

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

Using the stats layer

- Can map on statistical functions to transform adata
- 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.

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](#)

Saving plots to files

- A few ways to save a plot
- You can export from the Plots tab in R Studio (default is the bottom right panel, Plots tab)
- Can right click on an image and save it
- Can use `ggsave()` in your code to save the plot object
 - ▶ Easy to specify width, height, name of file

```
myplot <- ggplot(data, aes(x) + geom_bar())  
ggsave("myplot.png", plot = myplot)
```

Examples

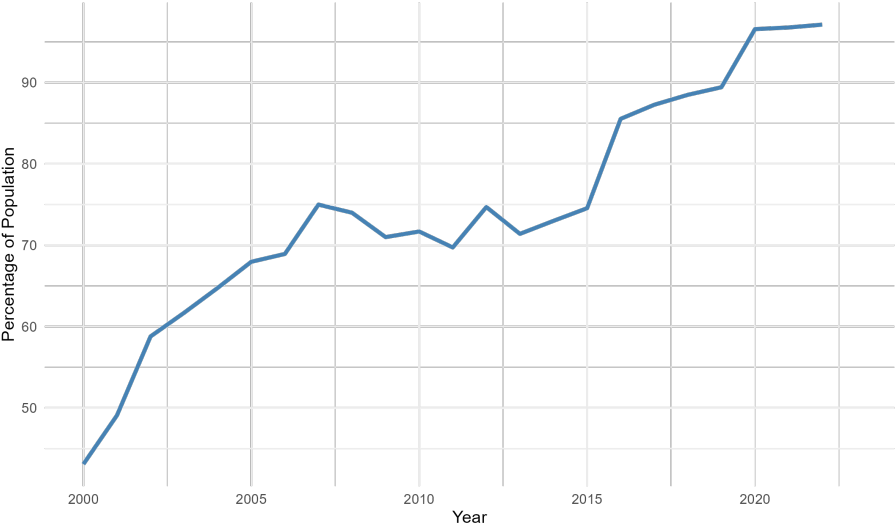
```
# Time series for US internet usage
us <- df %>% filter(Country.Name == "United States")

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)



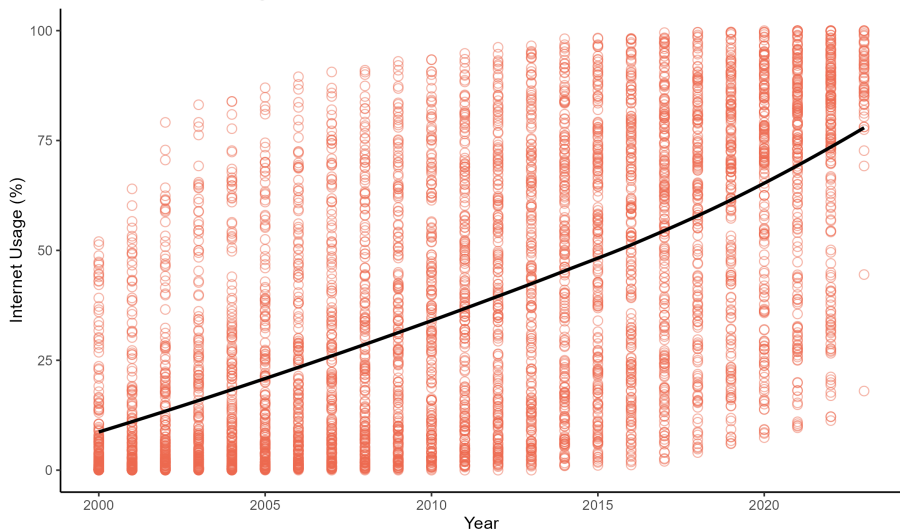
```
# Scatterplot with trendline
```

```
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



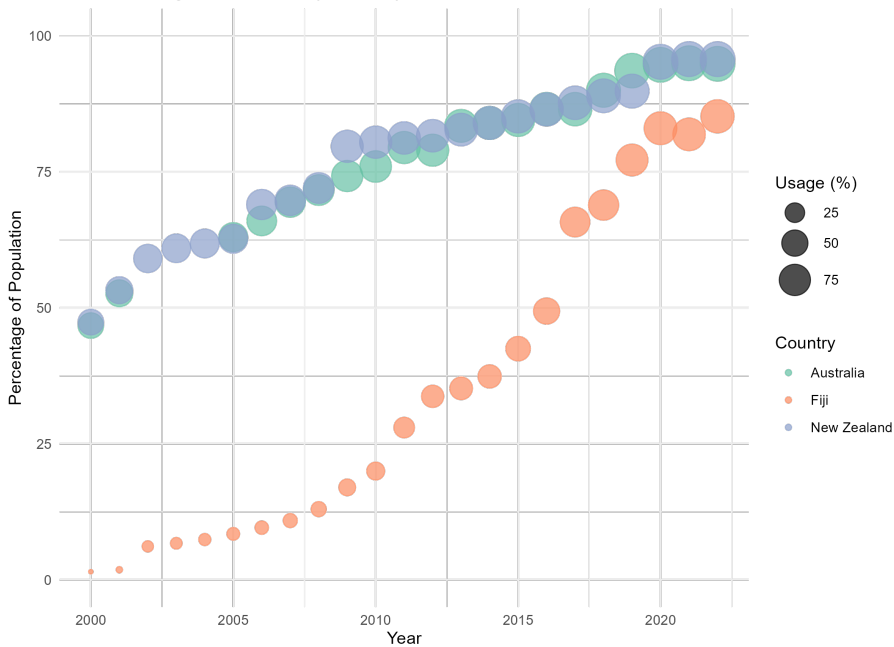

```
# Bubble plot for usage over time, three countries
clean <- df %>%
  filter(Country.Name %in% c("Australia", "New Zealand", "Fiji")) %>%
  na.omit()

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



Final thoughts

- When in doubt, ask questions! There are endless resources online about plotting with ggplot2. Talk to your professor, TA, or classmates.
- Don't get overwhelmed!
- 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