ggplot2

You will NOT be tested on writing R code

Grinnell College

September 2025

Note on the use of AI

It's important that you learn how to do this on your own with other resources before using AI.

You won't actually learn to code if you have an AI do the questions for you.

As the instructor of the course I'd honestly rather you get an 80% on your own two feet over finished perfectly with minimal effort (ie extensive AI use). It unlikely your grade will come down to a fraction of a percent you gain by it's use.

R Packages

Before we get to ggplot2 let's talk about *packages* in R

- A package in R must be downloaded
  - install.packages('PACKAGE NAME')
- It contains useful (??) functions/capabilities that you'd have to code yourself otherwise
- For gamers, packages are like mods
  - User made/open source
  - Edit capabilities
  - Can be toggled on and off
- To load a package we use library(PACKAGE NAME)
  - No quote marks for library(), quotes for install.packages()

Package Examples

Here's are my favorites/memorable ones:

- From Hadley Wickham we have...
  - ggplot2 (graphing)
  - dplyr (data manipulation)
  - Roxygen2 (easier to build your own packages)
  - stringr (manipulate chracter strings)
  - reshape2 (he's abandoned this one...)
- beepr
  - Makes a beep/noise at the end of code
  - Useful to indicate your code chunk is done
  - It's cute
- And experimental design research specific
  - FrF2 (2-level fractional factorial designs)
  - rsm (response surface methods)
  - AlgDesign (searchs for optimal designs)

R Packages: Common Pitfalls

So important tidbits to make your code more efficient:

- You only have to install a package once
  - So don't add it to your markdown file
  - Run it in your console instead
  - (Full disclosure: occasionally re-installs of packages are needed but those are few and far between)

- You have to load your package with library() before you can use the functions in the package
  - Errors for this are often "cannot find function 'ggplot' "
  - Have to load the library in the markdown ONCE
    - ★ Best practice is generally to put all the libraries at the top of the file together

Heads Up

Don't worry about understanding the following slides...big take away is that ggplot2 works by adding commands/desired edits on top of each other via the + symbol

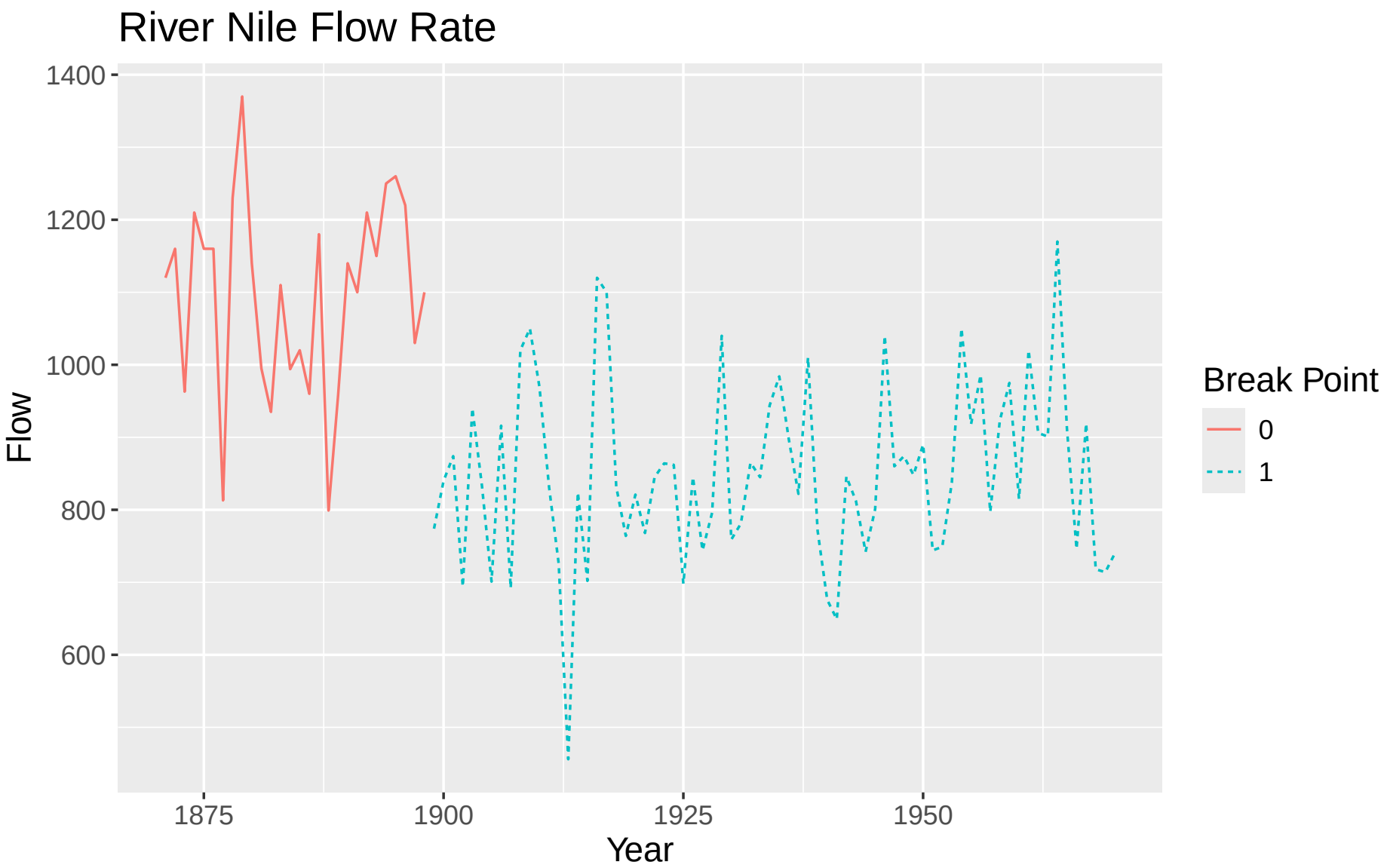
Different commands do different things

Grammar of Graphics (Leland Wilkinson 2004)

Graphics are independent elements that are added to each other similar to grammar

- A statistical graph maps data
- Mapping is made up of...
  - Layers
    - ★ Geometries (lines, bar, dots)
    - ★ Stat summaries (means, binning data in histograms)
  - Scales
    - ★ Colors, shapes, sizes, etc...
  - Coords
    - ★ Coordinate system (eg mpg vs kpl)
    - ★ Default is usually fine
  - Facets
    - ★ Breaks the graphs into subsets (eg a grid of histograms)
  - Themes (minor aesthetics)

Example


River Nile Flow Rate
1400
1200
1000
800
600
Flow
1875
1900
1925
1950
Year
Break Point
0
1

Code to Make Example

```
ggplot(data = Nile, #function is ggplot, data is Nile
       aes(x = year, #x-axis and y-axis define inside aes()
           y = flow)) +

  geom_line(aes(color = changepoint, #make a graph of lines
                linetype = changepoint)) + #make the colors and line
                                           #type based on the change point

  theme(text = element_text(size = 14)) + #boost font size

  labs(title = "River Nile Flow Rate", #give main title
       color = "Break Point", #and give the legend for color and line
       linetype = 'Break Point')  + #type a title as well

  xlab('Year') + #labels for x and y axis

  ylab("Flow")
```

Breakdown of Code

- Data: first line defined as data = Nile
- Coordinates: x = year, y = flow define our axis and ggplot2 will make the coordinates for those two
- Scales: Color and line type are encoded with information on the change point
- Geometry: geom_line in that we want to connect our data points
- Theme: up the font size to make it easier to read
- Facets: we didn't facet

Useful Geometries

- One Numeric variable
  - geom_histogram (makes a histogram)
- Two Numeric variables
  - geom_point (scatterplot)
  - geom_jitter (scatterplot + white noise)
  - geom_abline (draws a straight line)
  - geom_smooth ("free-hand" draws a line; usually curvy)
- One Categorical variable
  - geom_bar (bar chart)
- A Numeric variable and a Categorical variable
  - geom_boxplot (Guess the plot)

Rules of Thumb

- When using geom_ () functions, we usually reference the variables in the data set inside the aes() function
  - geom_point(aes(shape = MY_VARIABLE))

- We use + to string together the different parts

Coding Better Practices

- Save early, save often
- Name variables something easy to understand but also easy to type
  - "flow" is better than "Flow Rate measured in 10^8 m^3"
- Don't write all your code on a single line
- A new line after comma's or +'s is roughly standard (see examples above)
- Don't edit working code directly; copy and paste the code and then edit it
- Stretch goal: comment (with #) on the code as you go
  - You don't have to be as detailed as mine
  - Don Knuth insisted that this will save time in the long run