Programming Camp Day 1

Augustus Kmetz

2025-09-10

Welcome to Stanford Economics!

Introductions

- Gus - Norman, OK; board games...

Introduce yourselves

- Name
- Where you're from
- Hobby *or* a book/movie/other thing you did recently

Some Advice

Free Disposal

- You are at the start of a very formative phase of your life
- You are here for a marathon not a sprint
- Your cohort and the other students in the department are your colleagues
- You are all smart you don't need to prove that to anyone
- Asking questions is a way to learn not to signal
- Find some non-econ/non-grad student friends!
- SF is fun but Stanford's comparative advantage is nature

Famous Nature!

Yosemite 1

Yosemite 2

Local Nature!

Insert Russian Ridge or something else here

Grabtown Gulch

Henry Cowell Redwoods

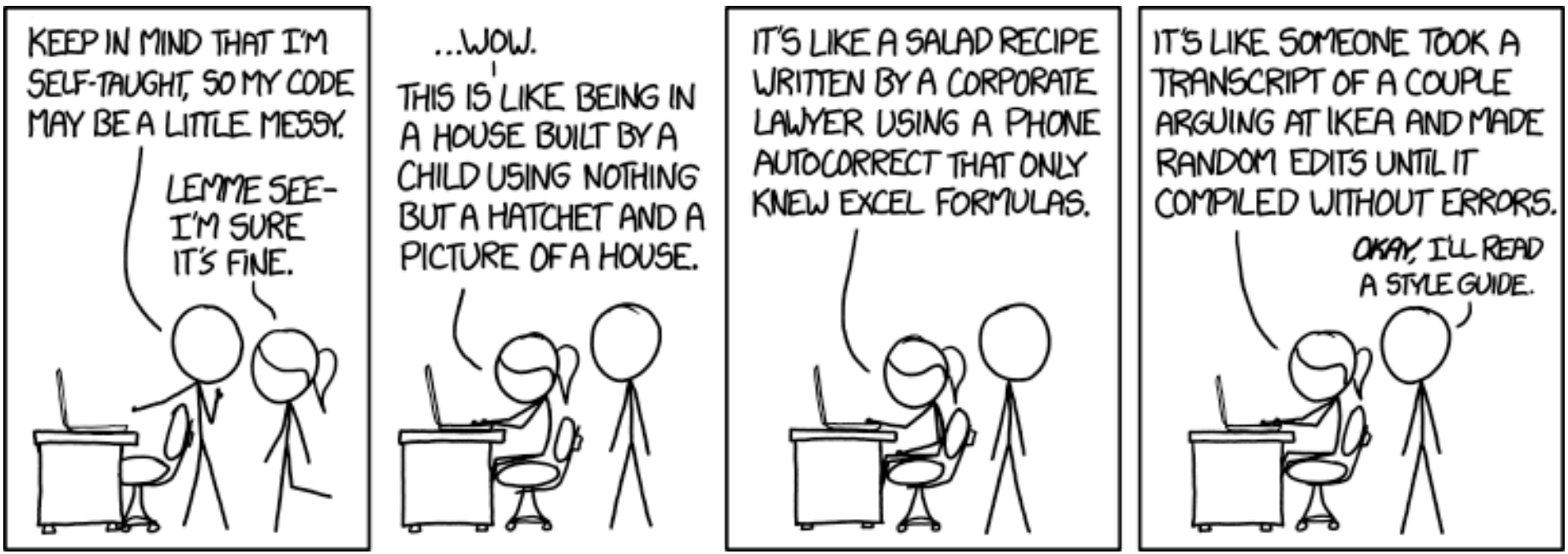
Why do we have programming camp?

- Modern economics **requires** more than pen and paper
- Need to be able to communicate with a computer to help you with data/model/simulations
- Many of you already know how to code in some language...but I suspect you learned in an ad hoc way
- Attempt to bring everyone onto even footing so that you can spend the time in first year classes learning the economics not how to code

Why do we have programming camp?

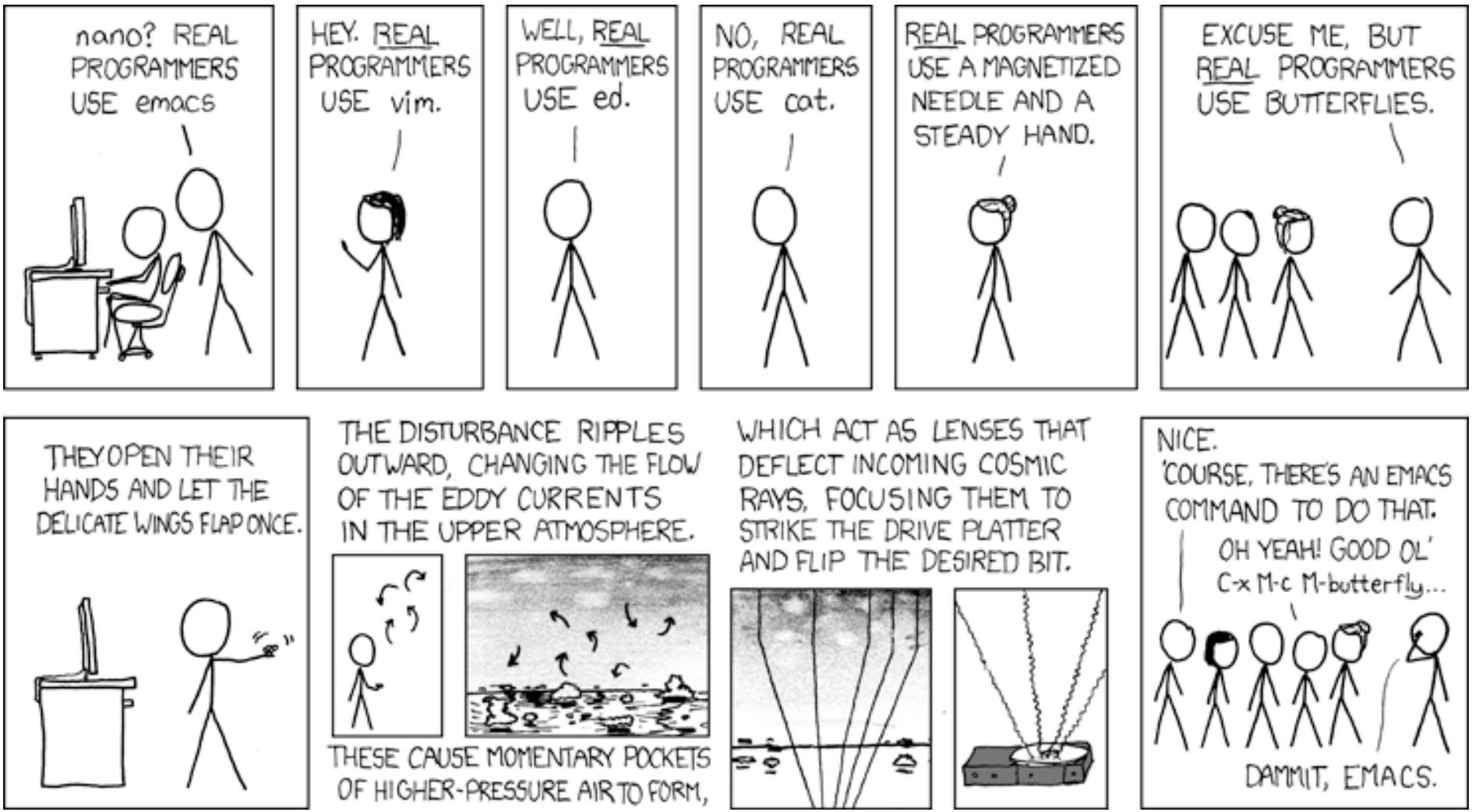
- Modern economics **requires** more than pen and paper
- Need to be able to communicate with a computer to help you with data/model/simulations
- Many of you already know how to code in some language...but I suspect you learned in an ad hoc way
- Attempt to bring everyone onto even footing so that you can spend the time in first year classes learning the economics not how to code
- Sessions in **R** and Sessions in **Python**
  - Get you ready for metrics/research going forward
  - Get you ready for dynamic programming in first year Macro
- Very low stakes - you are the best judge of the value of your time

Why do we have programming camp?



xkcd 1513

Use an appropriate tool for the job at hand



xkcd 378

Intro to R

Why R as an economist?

Pros:

- Free and user driven
- Widely used in fields outside of economics
- New "exotic estimators" and ML methods appear in R first
- Great for working with messy, complex, and multiple data sets
- Low bar to produce beautiful graphics
- Easy to integrate with Latex and Beamer

Cons:

- If you have a pristine data set and want to run a few regressions maybe use STATA
- If you want to use **really big** data

Other useful resources for learning R

- R for Data Science - Freely available at: http://r4ds.had.co.nz/
- *Advanced R* by Hadley Wickham for intermediate programmers
- Data Visualization: A practical introduction - Freely available at https://socviz.co/
- Data Camp - free courses for R, python and more

Plan for first sessions

- Nuts and Bolts/Grammar of Graphics
- Importing/Transforming/Tidy Data
- Relational Data/Regressions/Rmarkdown
- Control Flow/Functions/Functionals

Getting started with R (at home)

Installing R (the engine)

- Download and Install
  - pick the latest version for your OS
  - each year a new version of R is available, and 2-3 minor releases

Installing RStudio (the car)

- Download and install
  - scroll down to "Installers for Supported Platforms" near the bottom
  - pick the download link corresponding to your computer's operating system

Installing Tex

- To use Tex for typesetting with RStudio download and install:
  - Windows: MiKTeX
  - Mac: MacTeX
  - Linux: Tex Live

Getting started with R (today)

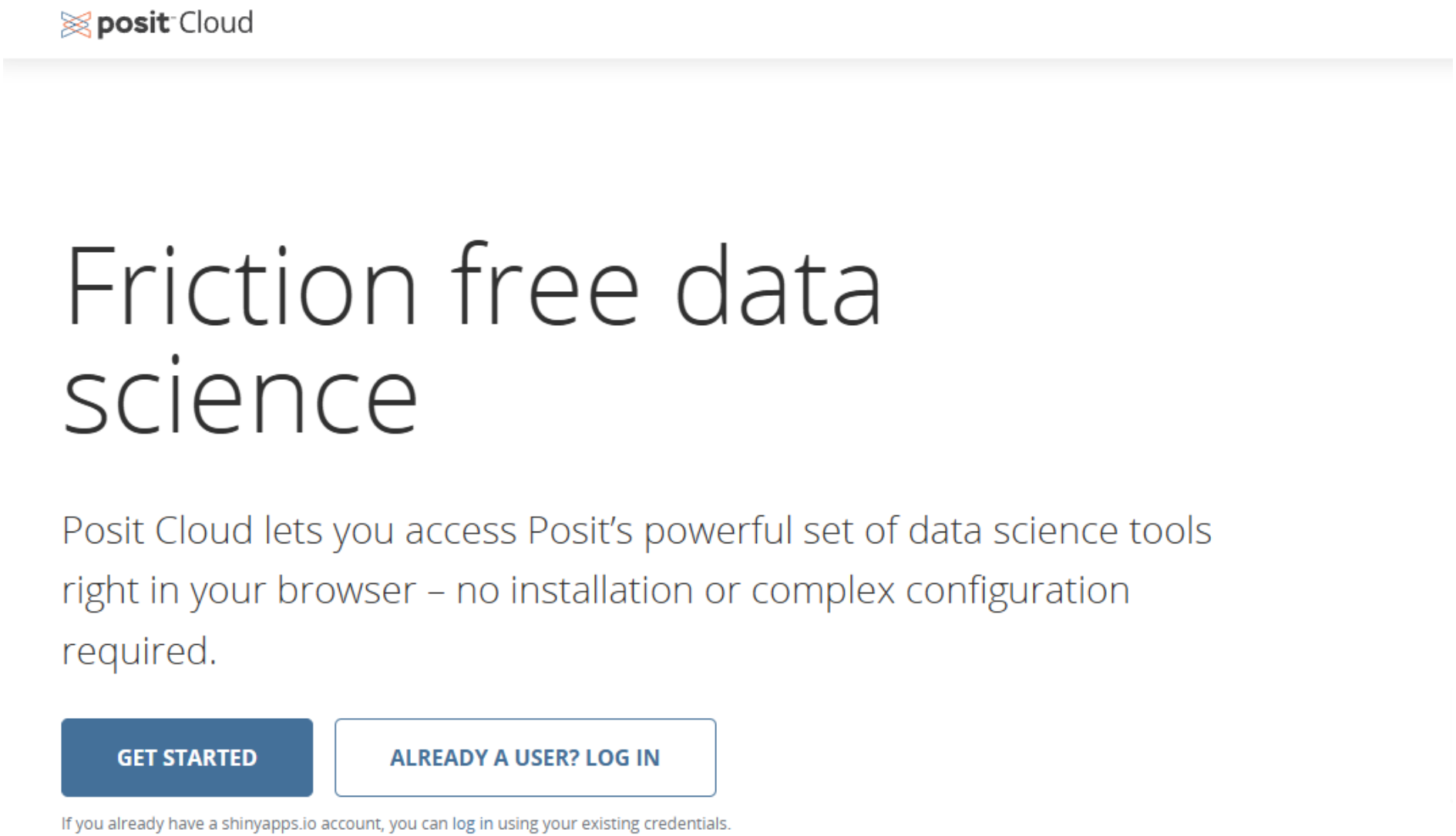
RStudio is a user-friendly graphical interface for the R software

- Lots of nice built in helpfulness
- Easy to interface with git

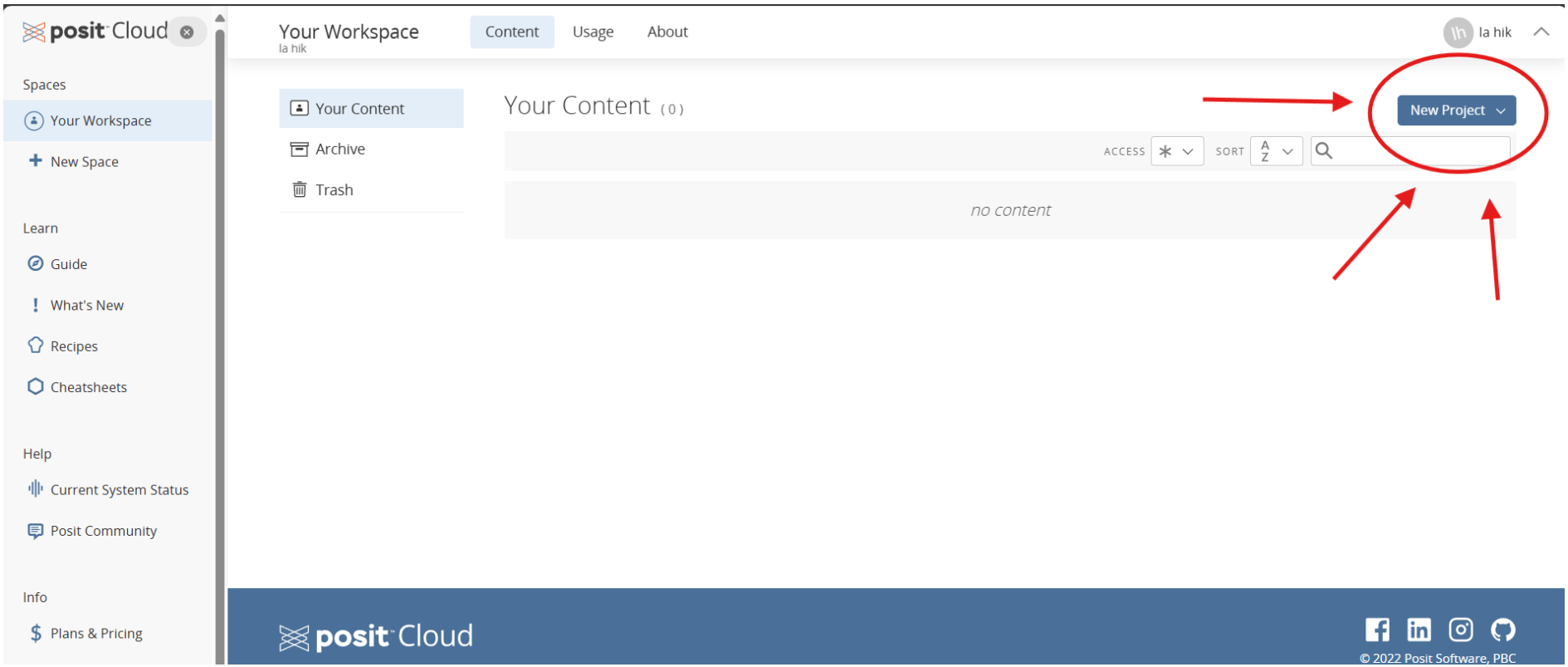
We will use Posit Cloud (formerly RStudio Cloud)

- Cloud version that can be accessed directly from your browser
- Waste as little time as possible getting things set up
- Happy to provide support for you setting it up on your own machine
- Have you all set up your accounts?

Getting Started with Posit Cloud 1

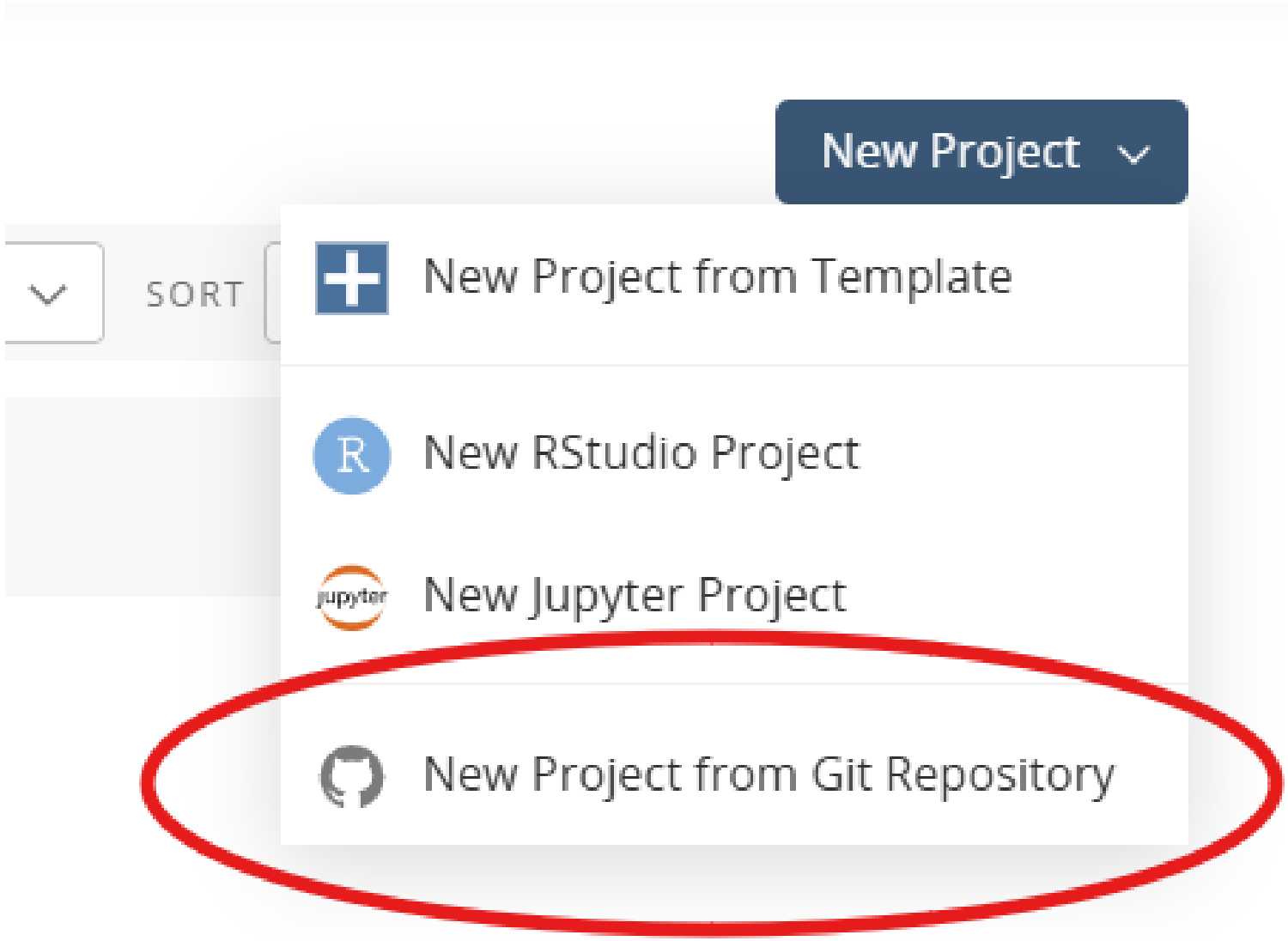


Getting Started with Posit Cloud 2



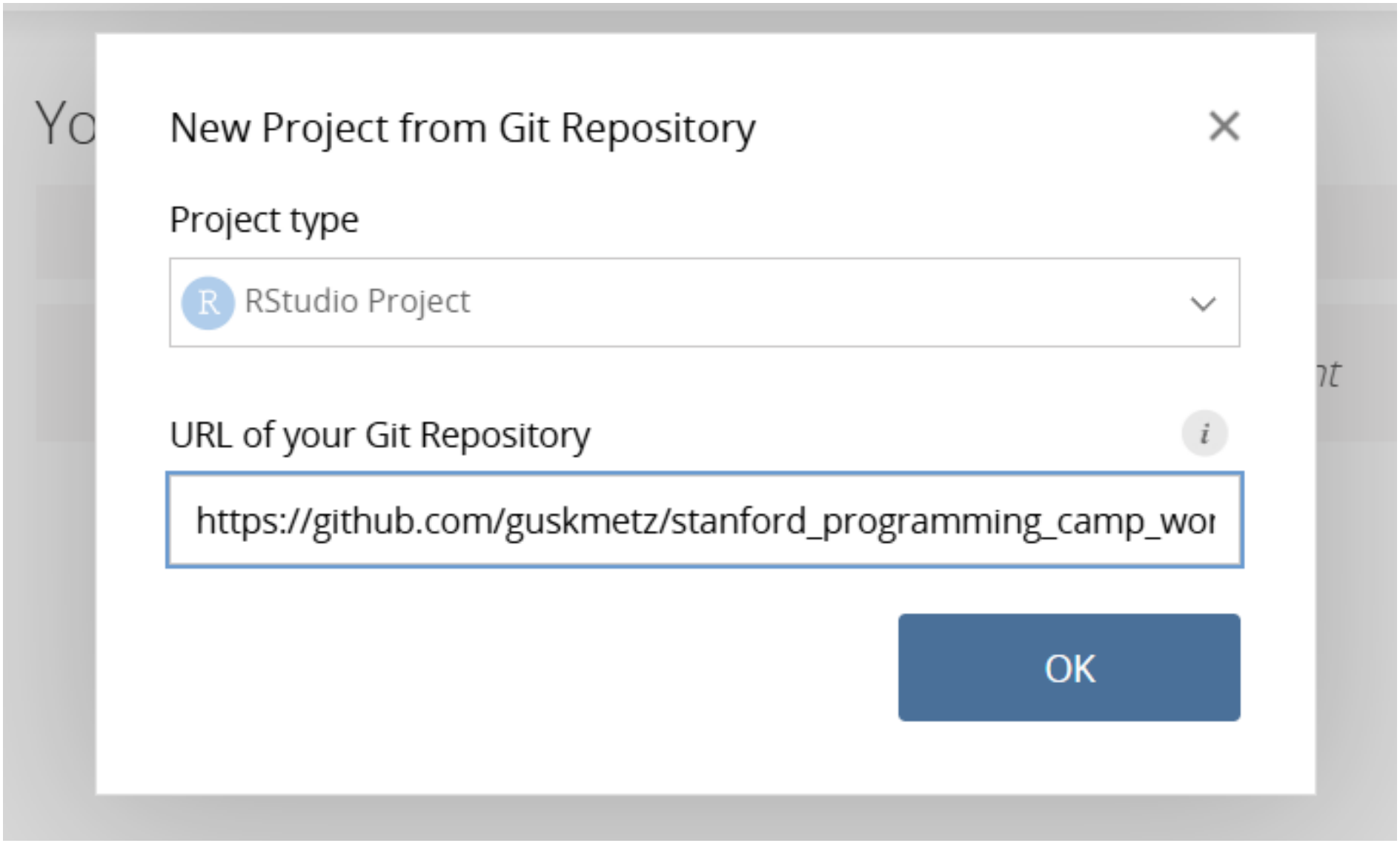
Empty workspace

Getting Started with Posit Cloud 3



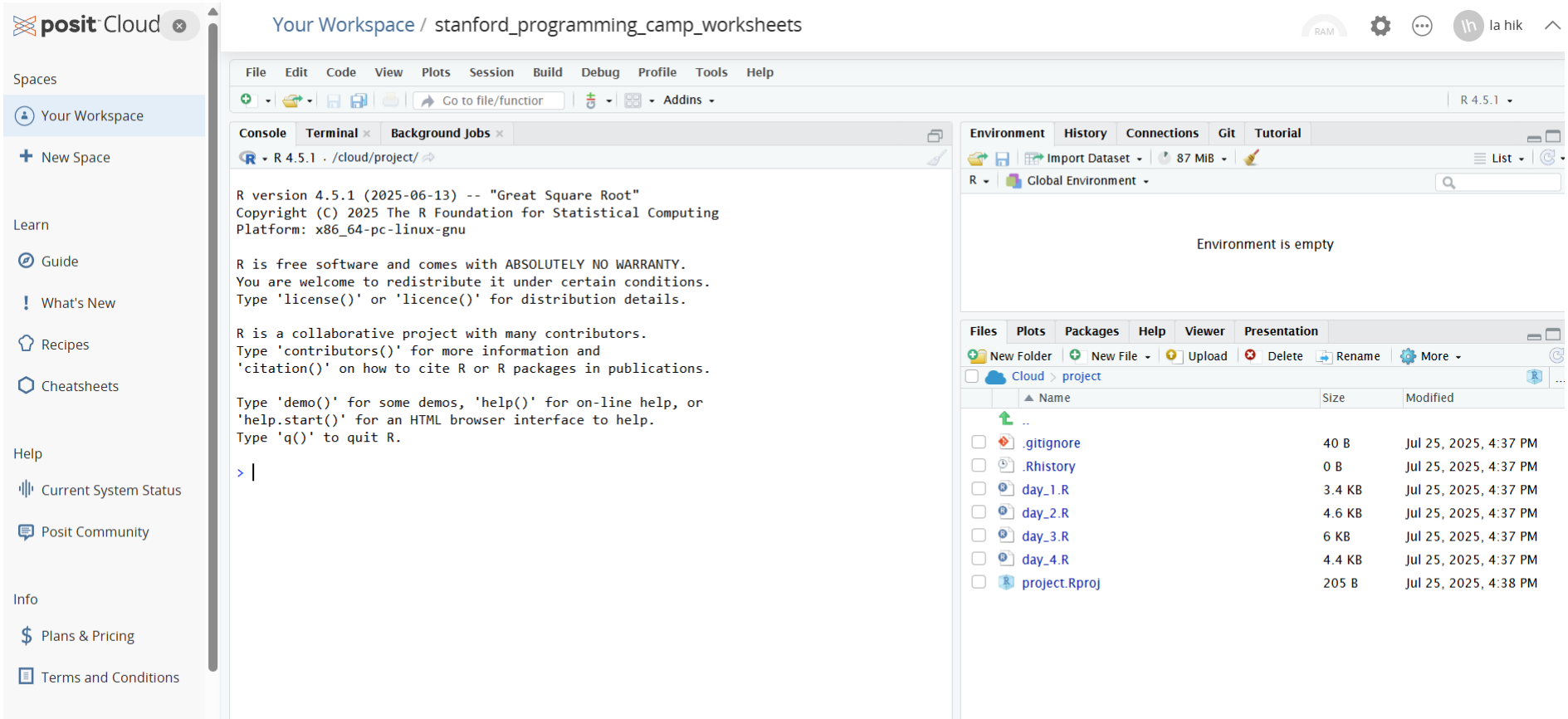
Empty workspace

Getting Started with Posit Cloud 4



Empty workspace

Getting Started with Posit Cloud 5



Empty workspace

Running R code

Interpreter mode

- The console is good for seeing results interactively
- Try typing:

```
print("hello world!")
```

- The console can also work as a calculator
- R understands some math already
- You can also run system commands that you would run in your terminal

Scripting mode

- To save our work/do more complicated things we use text files called Rscripts
- Click on File > New File > R Script

Nuts and Bolts

R as a calculator

- R can be used as a calculator
- Intuitive arithmetic operators: addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^), modulus (%%)
- Built in constants: `pi`, `LETTERS`, `state.abb`, `state.name`, etc.

Creating new objects

- Variables are objects used to store various information
- In R you do not declare the variable type
- You can create new objects with <- (also ->)

```
x <- 42
```

- Combine items using c()

```
x <- c(42, 43)
x[1]
```

```
[1] 42
```

- All R statements when you create objects have the same form

```
object_name <- value
```

- Best practice not to use = - reserved for use within functions

Making your code readable

- Use informative and descriptive object names
- Surround your <- with spaces
- In general use spaces to improve readability
- R does not care if you break code across lines

Naming Objects

- i_use_snake_case
- otherPeopleUseCamelCase
- some.people.use.periods
- And_aFew.People_RENOUNCEconvention
- Very helpful to distinguish your objects from built in objects/functions/etc

Naming Objects

- RStudio is really helpful
- Create an object with the name `really_long_and_painful_to_type` and assign it the value of 42
- To inspect the object start to type "really" and then press TAB
- Oops the value was supposed to be 43...press the up arrow when typing at the command line

Naming Objects

Make yet another variable

```
r_rocks <- 3
```

Let's try to inspect it?

```
r_rock
R_rocks
```

Why don't these work?

Naming Objects

- There's a contract between you and R (and really most programming languages)
- It will do tedious and hard computations for you but in return you must be completely precise in your instructions
- Typos matter
- Case matters
- Indexing matters (R starts with 1)

Calling functions

- R and all the associated packages have many functions that are all called in a similar manner:

```
function_name(argument_1 = value_1,
              argument_2 = value_2,
              ...,
              argument_n = value_n)
```

- Let's try using `seq()` which makes regular sequences of numbers
- What are the arguments to `seq()`? How could you find out?

Calling functions

- Create a sequence of even numbers from 1 to 10 and assign it to an object called `even`
- Create a sequence of odd numbers from 1 to 10 in two ways: using `seq()` and using your already created object `even`
- Use `sum()`, `length()`, and `mean()` to calculate the average value

Packages

- Packages are a collection of functions, compiled code, and sample data
- They are stored under a directory called library in the R environment
- Some packages are installed by default during R installation and are always automatically loaded at the beginning of an R session
- You can install and load additional packages

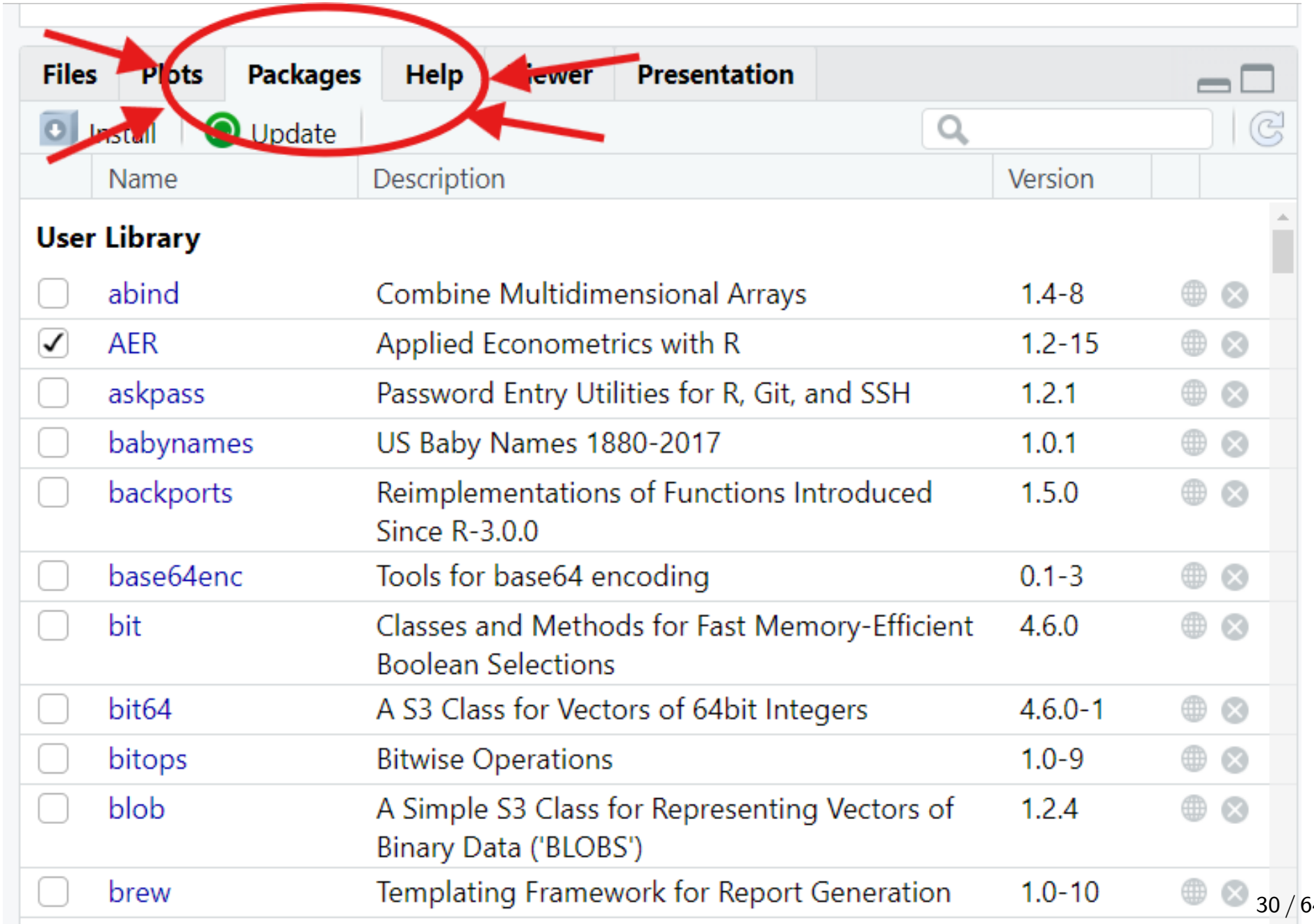
Packages

R comes pre-loaded with a number of packages, appropriately called the base packages. We can see which packages are installed by running the following command:

```
rownames(installed.packages(priority="base"))
```

```
 [1] "base"      "compiler"  "datasets"  "graphics"  "grDevice
 [7] "methods"   "parallel"  "splines"   "stats"     "stats4"
[13] "tools"     "utils"
```

Packages


Files
Plots
Packages
Help
Viewer
Presentation
Install
Update
Name
Description
Version
User Library
abind
Combine Multidimensional Arrays
1.4-8
AER
Applied Econometrics with R
1.2-15
askpass
Password Entry Utilities for R, Git, and SSH
1.2.1
babynames
US Baby Names 1880-2017
1.0.1
backports
Reimplementations of Functions Introduced Since R-3.0.0
1.5.0
base64enc
Tools for base64 encoding
0.1-3
bit
Classes and Methods for Fast Memory-Efficient Boolean Selections
4.6.0
bit64
A S3 Class for Vectors of 64bit Integers
4.6.0-1
bitops
Bitwise Operations
1.0-9
blob
A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')
1.2.4
brew
Templating Framework for Report Generation
1.0-10

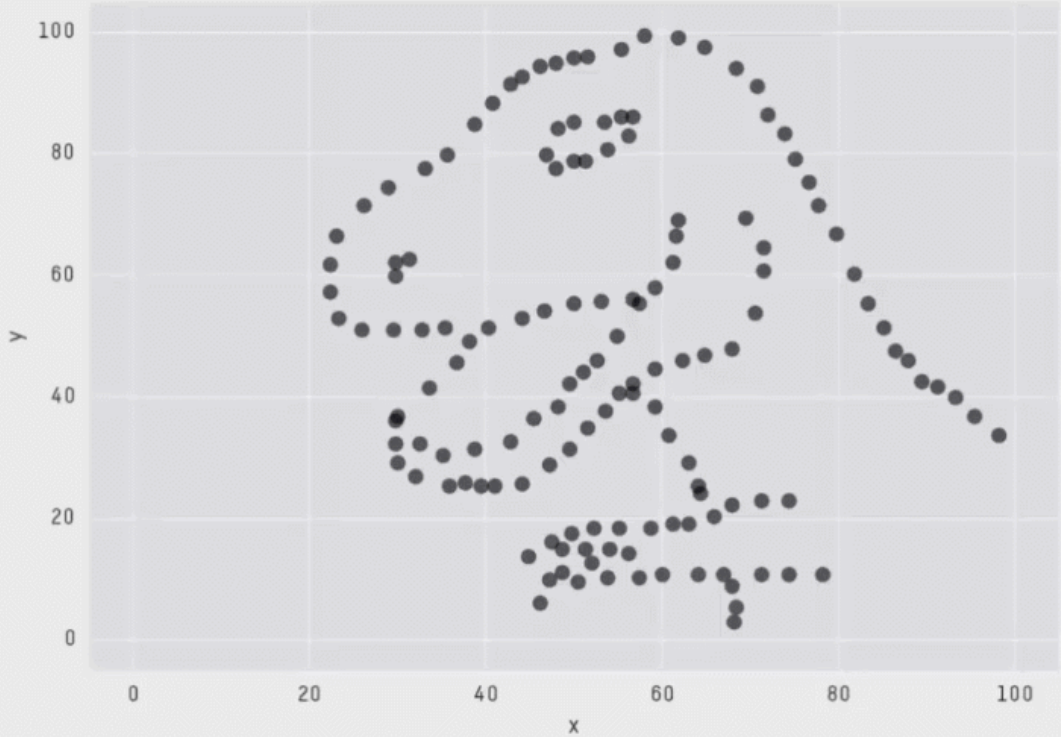
Packages

```
# Install a package on your computer
# (also this is how you write comments in R)
# You only need to run this command once!
# (...you need to install all packages for every new Posit Clo
install.packages("ggplot2")
install.packages("AER")

# Load an installed package in the R session
# You need to load the package every time!
library(ggplot2)
library(AER)
```

Grammar of Graphics

Why do we need to visualize data?



X Mean: 54.2659224
Y Mean: 47.8313999
X SD : 16.7649829
Y SD : 26.9342120
Corr. : -0.0642526

Matejka & Fitzmaurice 2017

Introduction

- What is a graphic? How can we succinctly describe a graphic? How can we create the graphic we described?
- A grammar is an abstraction which makes thinking reasoning and communicating graphics easier
- Developed by Lealand Wilkinson (1999/2005) refined and implemented in R by Hadley Wickham (2006)
- Basic idea: building up a graphic from multiple layers of data

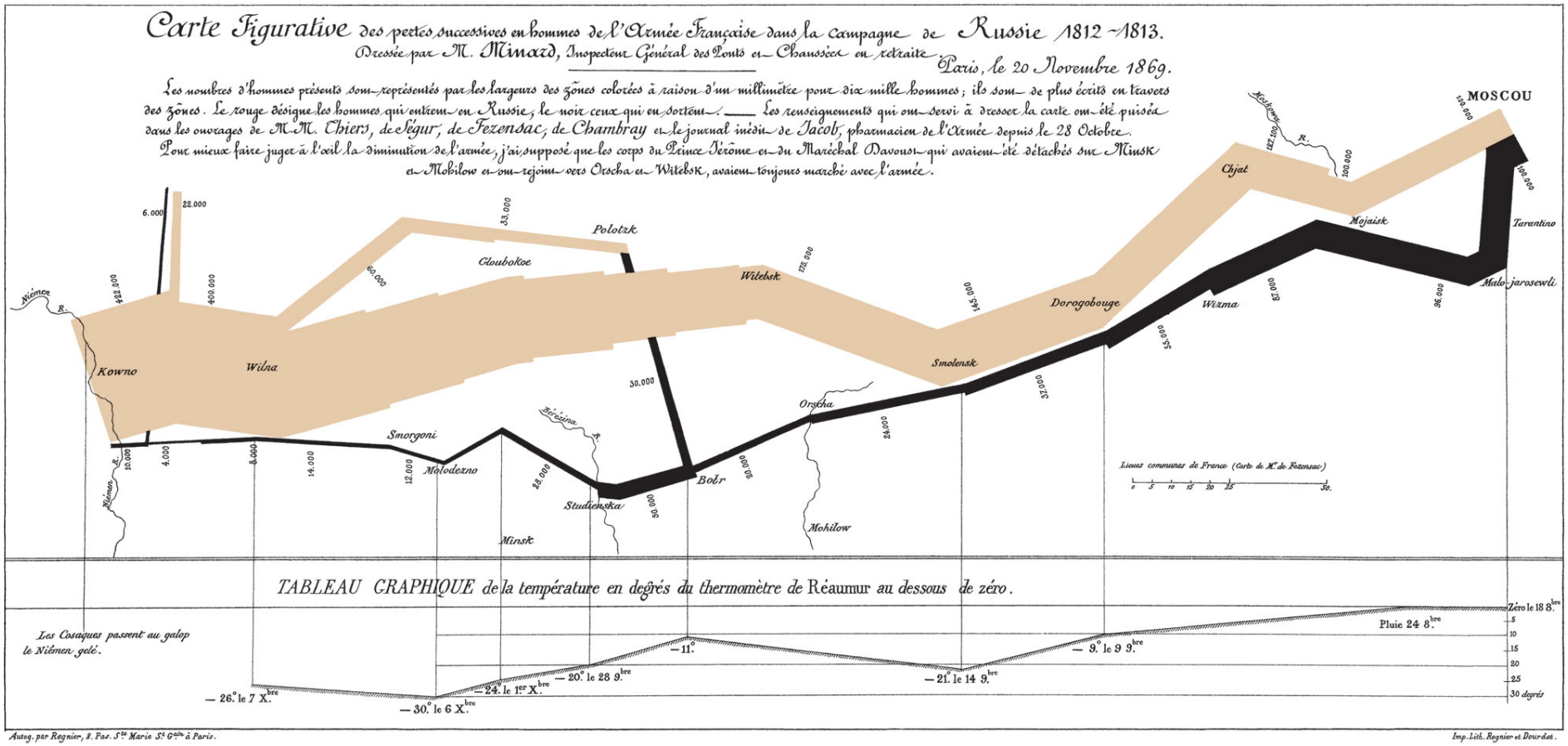
ggplot2 packages

- Advantages of ggplot2
  - layered grammar of graphics that allows you to specify building blocks of graphics and combine them to create just about any kind of graphical display
  - very flexible can make plots ranging from simple scatter/bar/line to histogram/density/boxplot to maps
  - documentation is well-written and online support is plentiful
  - many packages exist which extend the functionality

ggplot2 packages

- Advantages of ggplot2
  - layered grammar of graphics that allows you to specify building blocks of graphics and combine them to create just about any kind of graphical display
  - very flexible can make plots ranging from simple scatter/bar/line to histogram/density/boxplot to maps
  - documentation is well-written and online support is plentiful
  - many packages exist which extend the functionality
- limitations of ggplot2
  - does not handle 3D graphics - generally 3d plots do not translate well to 2d presentations
  - does not offer interactive plots
    - use `plotly` instead
  - inefficient for graph/network plots with nodes and edges
    - use `igraph` instead

A great graph

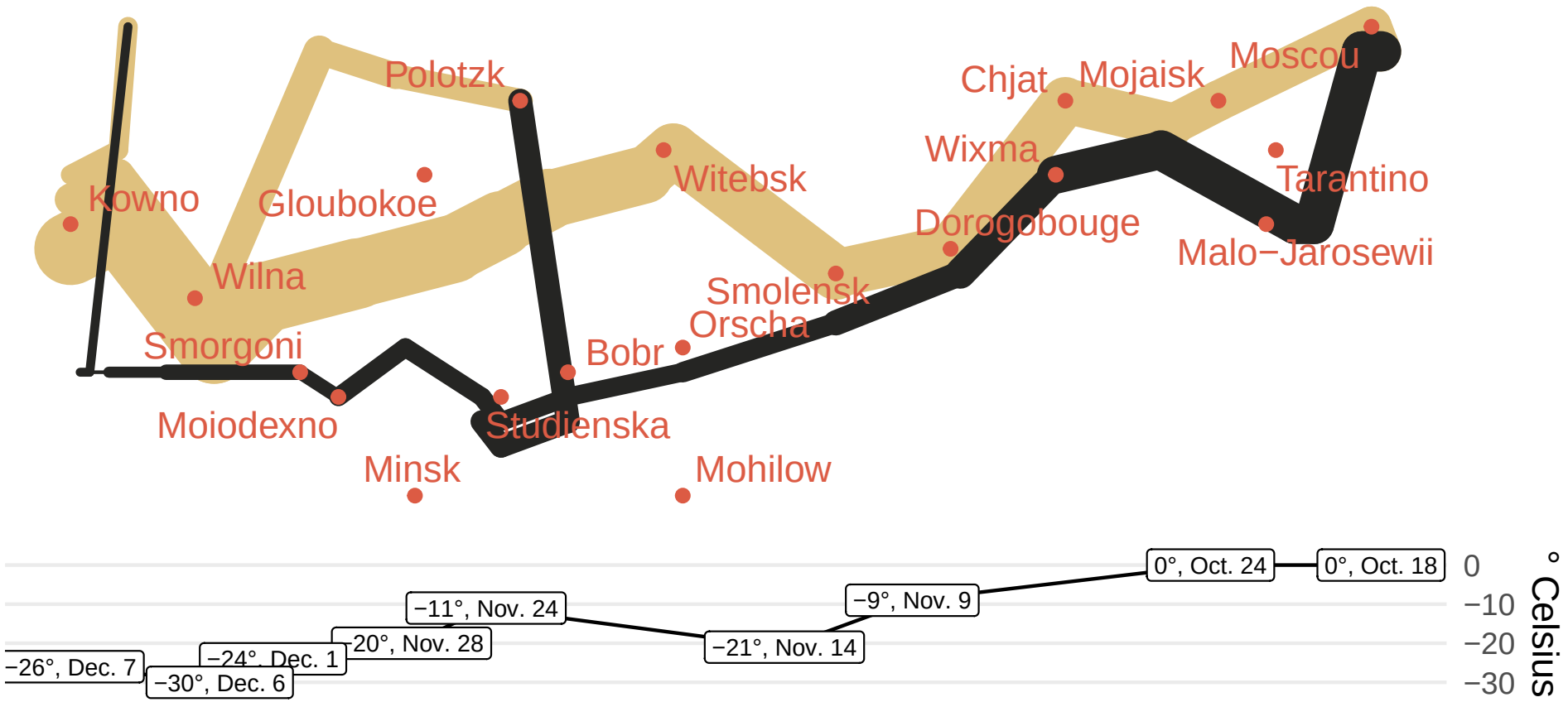


Napoleon's March

Unpacking the aesthetics

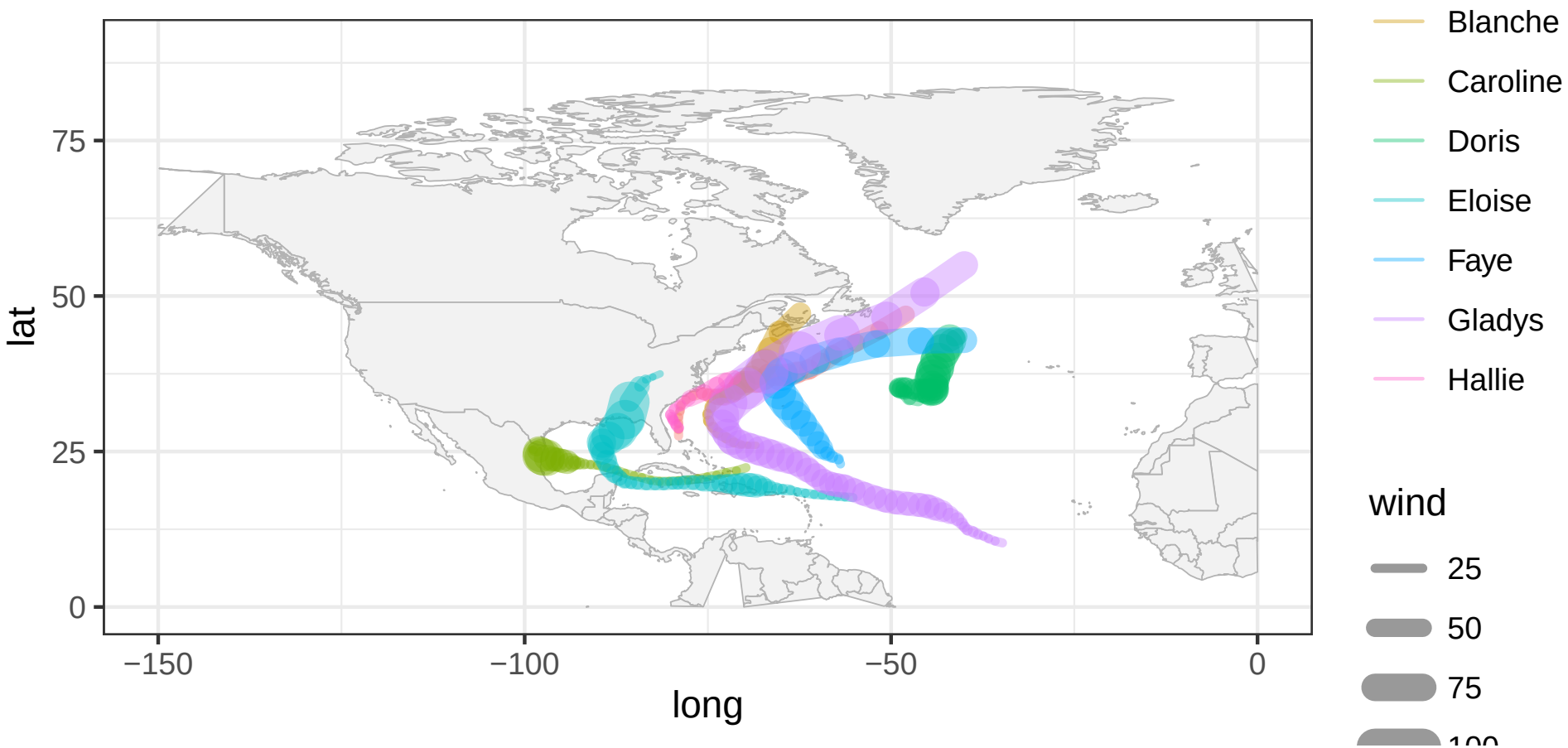
Information	Aesthetic
Size of Napoleon's Grande Armée	Width of path
Longitude of the army's position	x-axis
Latitude of the army's position	y-axis
Direction of the army's movement	Color of path
Date of points along retreat path	Text below plot
Temperature during the army's retreat	Line below plot

Reproduced in ggplot2



Minard Reproduced with ggplot2

Map graphics



[1975 Atlantic Hurricane Tracks](https://www.gastonsanchez.com/R-tidy-hurricanes/maps-ggplot.html)

How to build a plot

When creating a plot we start with data

- Let's check out the dataset `CigarettesSW`
- This is a built in dataset from the `AER` packages
- Step 1 is to load in the data

```
data("CigarettesSW")
```

We can look at the data using the functions: `head()`, `tail()`, `summary()`, and `help()`

How to build a plot

Goal Simple scatter plot of cigarette taxes (`tax`) vs per capita consumption of cigarettes (`packs`)

How to build a plot

Goal Simple scatter plot of cigarette taxes (`tax`) vs per capita consumption of cigarettes (`packs`)

- More precisely we are mapping:
  - x-position to cigarette taxes
  - y-position to per capita consumption of cigarettes
  - color to the year of the observation
- x-position, y-position, and color are all examples of *aesthetics*, things we can perceive on the graphic

How to build a plot

Goal Simple scatter plot of cigarette taxes (`tax`) vs per capita consumption of cigarettes (`packs`)

- More precisely we are mapping:
  - x-position to cigarette taxes
  - y-position to per capita consumption of cigarettes
  - color to the year of the observation
- x-position, y-position, and color are all examples of *aesthetics*, things we can perceive on the graphic
- To create a complete plot we need to combine:
  - the data - represented by the point geom
  - the scales and coordinate system - which generates axes and legends so we can read values from the graph
  - plot annotations - such as the background and plot title

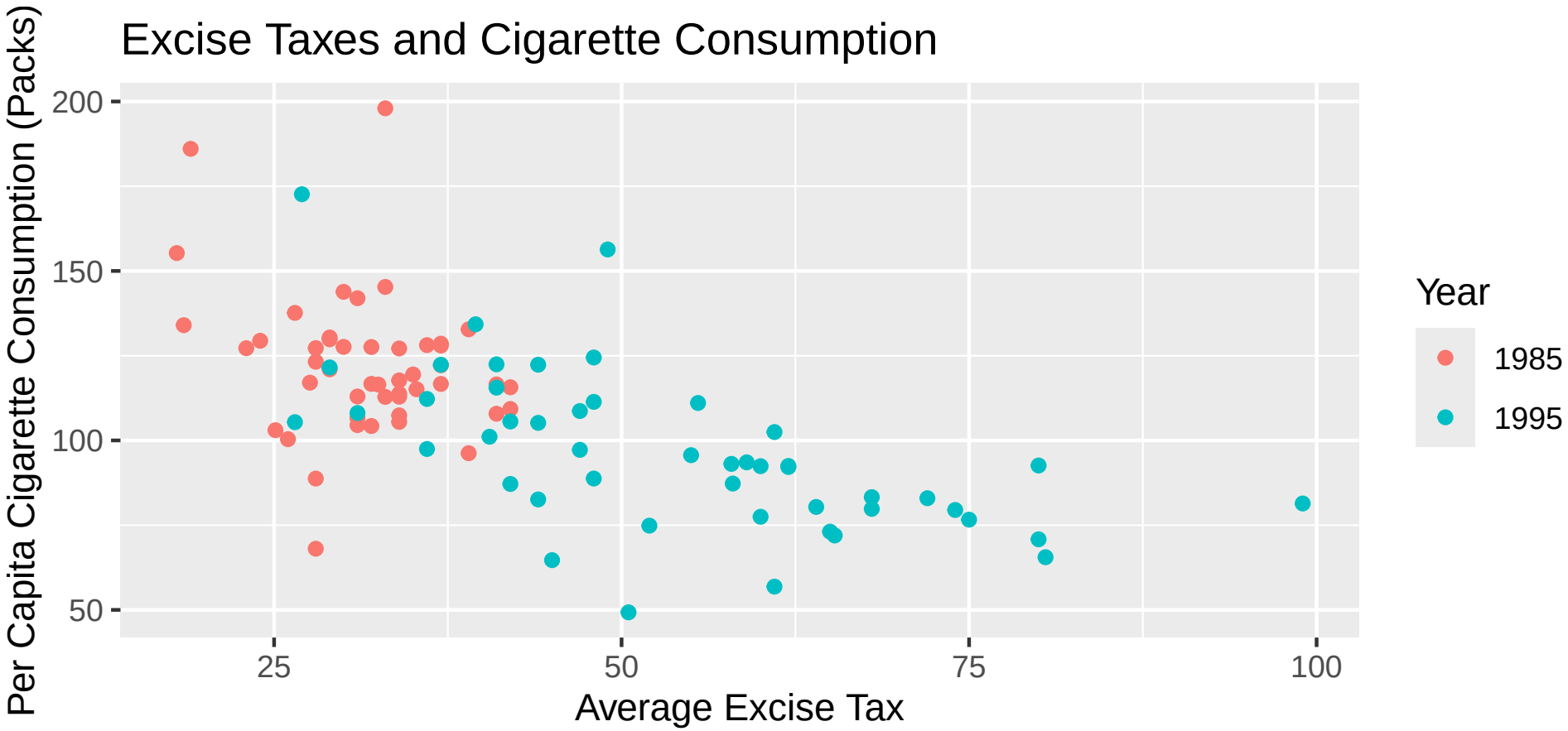
Actually building the plot

How do we implement this grammar using code?

- `ggplot()` function initializes a basic graph structure
- Different parts of a plot can be added together using +
- Any data or arguments you supply to the initializing `ggplot()` function call apply to all additional layers

```
ggplot(CigarettesSW, aes(x = tax, y = packs, color = year)) +
  geom_point() +
  labs(x = "Average Excise Tax",
       y = "Per Capita Cigarette Consumption (Packs)",
       color = "Year",
       title = "Excise Taxes and Cigarette Consumption")
```

Actually building the plot



Try some things out for yourself

- How would you change the code so that different years are represented by different shapes?
- Try changing the plot so that instead of points each observation is represented by the state abb `geom_text()`
- Install and load the package `ggthemes`
- Try **adding** a theme to your plot such as `theme_stata()`, `theme_economist()`

Discrete vs continuous aesthetics

The nature of the data determines the type of aesthetics and graphics you can use.

In `ggplot2`, there are two types of aesthetics: discrete and continuous.

- Discrete aesthetics are used to represent categories/classes.
- Continuous aesthetics are used to represent numbers.

Sometimes, the aesthetic is neither perfectly discrete or continuous. You could map a category to color (like retreat status in the Minard plot), but you could also map a number to the color (for instance temperature). This is where your judgement comes in (which matters since we are social scientists).

Layers

1. data and aesthetic mapping
   - data are what turn an abstract graphic into a concrete graphic
   - along with data we need a specification of which variables are mapped to which aesthetics
   - redundant mappings are important for making accessible graphics for men (colorblind individuals)
2. statistical transformation
3. geometric object (geoms)
   - control the type of plot you create
   - each geom can only display certain aesthetics
4. position adjustment

Layers in the wild

```
ggplot() +
    geom_histogram(data = CigarettesSW,
                   aes(x = tax, fill = year),
                   bins = 30,
                   position = "identity",
                   alpha = 0.5) +
    labs(x = "Average Excise Tax",
         y = "Count(States)",
         fill = "Year",
         title = "Distribution of Excise Taxes over Time")
```

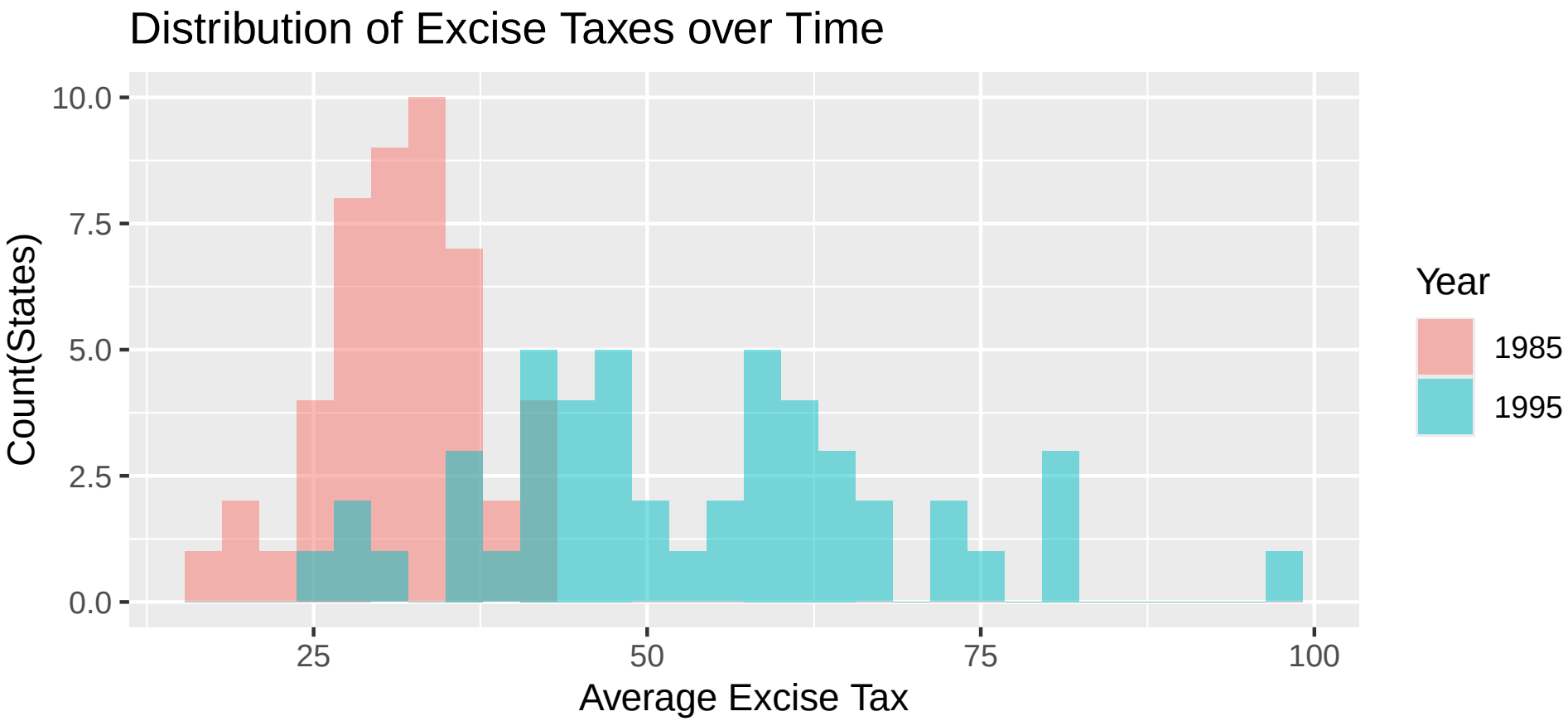
Using a chart to make a point

How has the distribution of taxes changed over time?

```
ggplot() +
    geom_histogram(data = CigarettesSW,
                   aes(x = tax, fill = year),
                   bins = 30,
                   position = "identity",
                   alpha = 0.5) +
    labs(x = "Average Excise Tax",
         y = "Count(States)",
         fill = "Year",
         title = "Distribution of Excise Taxes over Time")
```

Using a chart to make a point

How has the distribution of taxes changed over time?

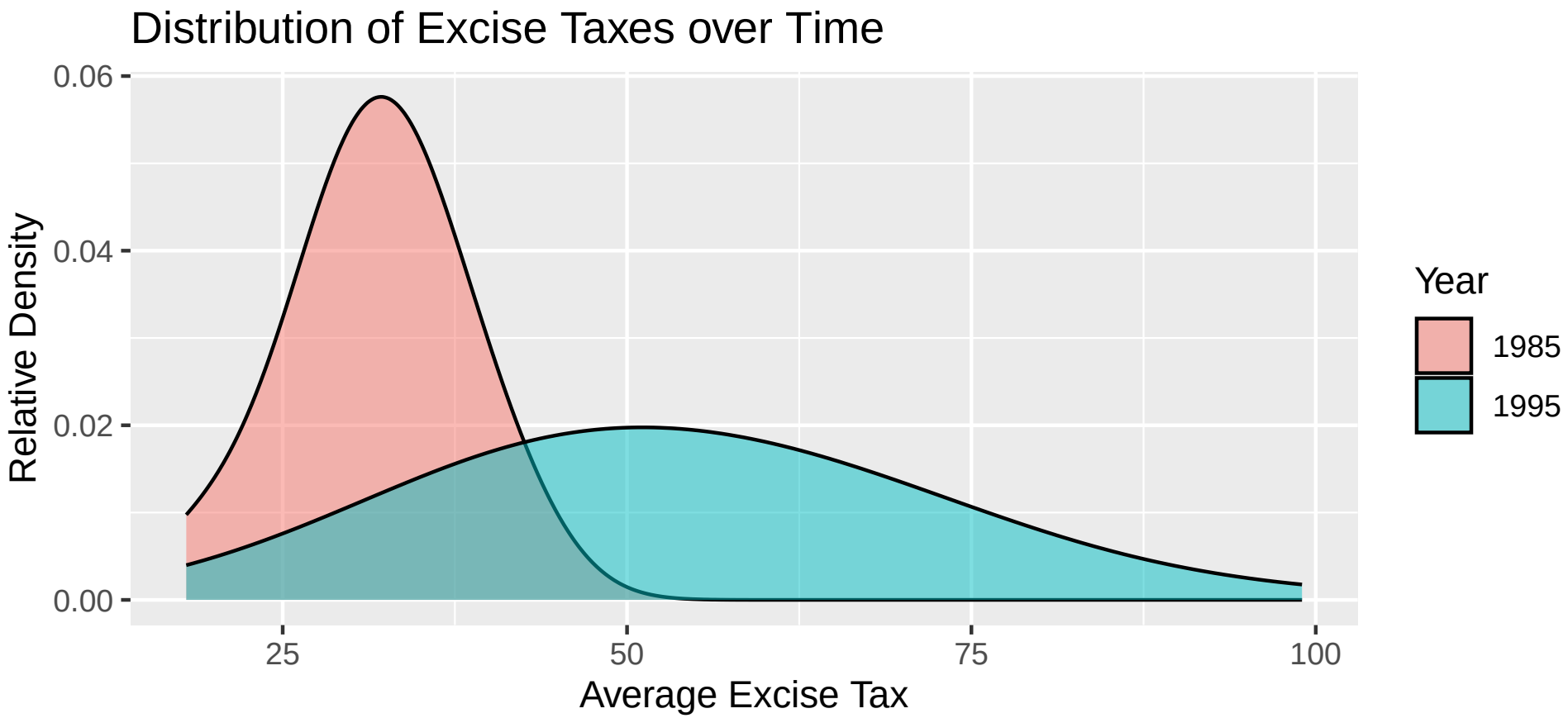


Using a chart to make a point

```
ggplot(CigarettesSW, aes(x = tax, fill = year)) +
    geom_density(alpha = 0.5, adjust = 2) +
    labs(x = "Average Excise Tax",
         y = "Relative Density",
         fill = "Year",
         title = "Distribution of Excise Taxes over Time")
```

Using a chart to make a point

- What chart you use/how much of the data you show depends on what point you are trying to make
- Can use different graphics to make your messaging more "crisp"



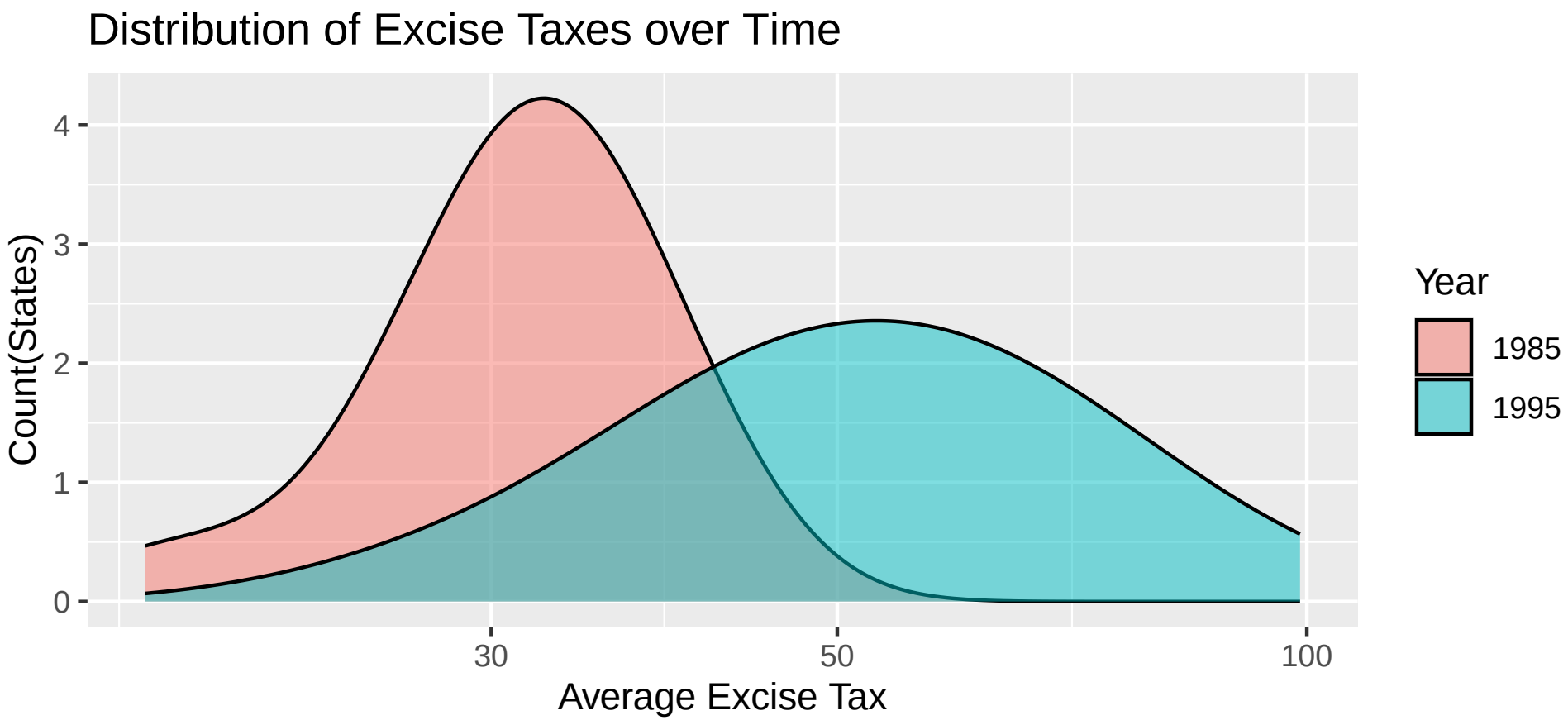
Scales

- A scale controls the mapping from data to aesthetic attributes
- Need one scale for each aesthetic property used in a layer
- Scales are common across layers to ensure a consistent mapping
- Scales typically map from a single variable to a single aesthetic with some exceptions

Scales in the wild

```
ggplot(CigarettesSW, aes(x = tax, fill = year)) +
    geom_density(alpha = 0.5, adjust = 2) +
    labs(x = "Average Excise Tax",
         y = "Count(States)",
         fill = "Year",
         title = "Distribution of Excise Taxes over Time") +
  scale_x_log10()
```

Scales in the wild



More plots

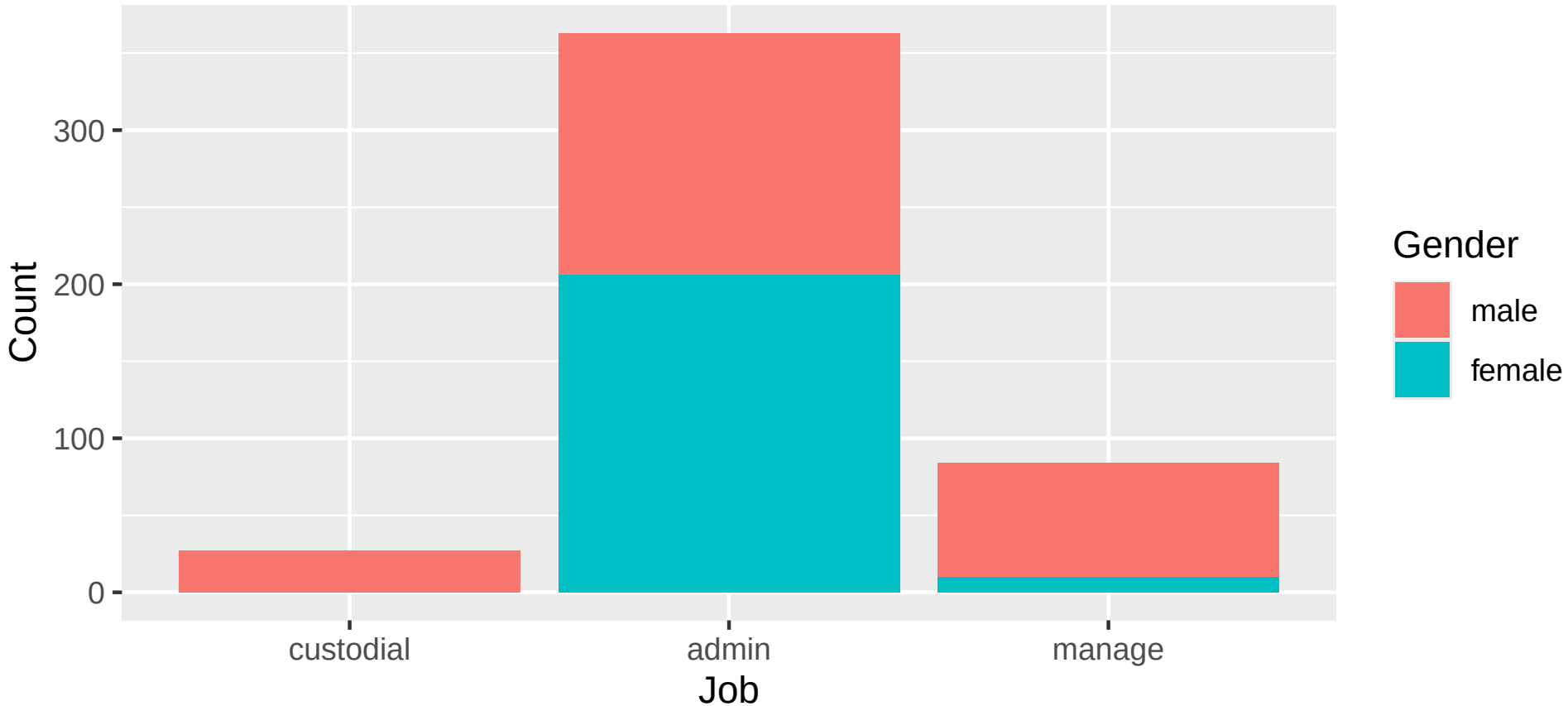
- Load in the `BankWages` dataset `data(BankWages)`
- What are the variables? hint(`head()`, `summary()`, `help()`)
- Let's make a bar plot of the gender distribution by job
  - What do we want our plot to look like?
  - What aesthetics should we use? What variables to map to each aesthetic?

Bar plots

```
data(BankWages)

ggplot(BankWages, aes(x = job, fill = gender)) +
  geom_bar() +
  labs(x = "Job",
       y = "Count",
       fill = "Gender")
```

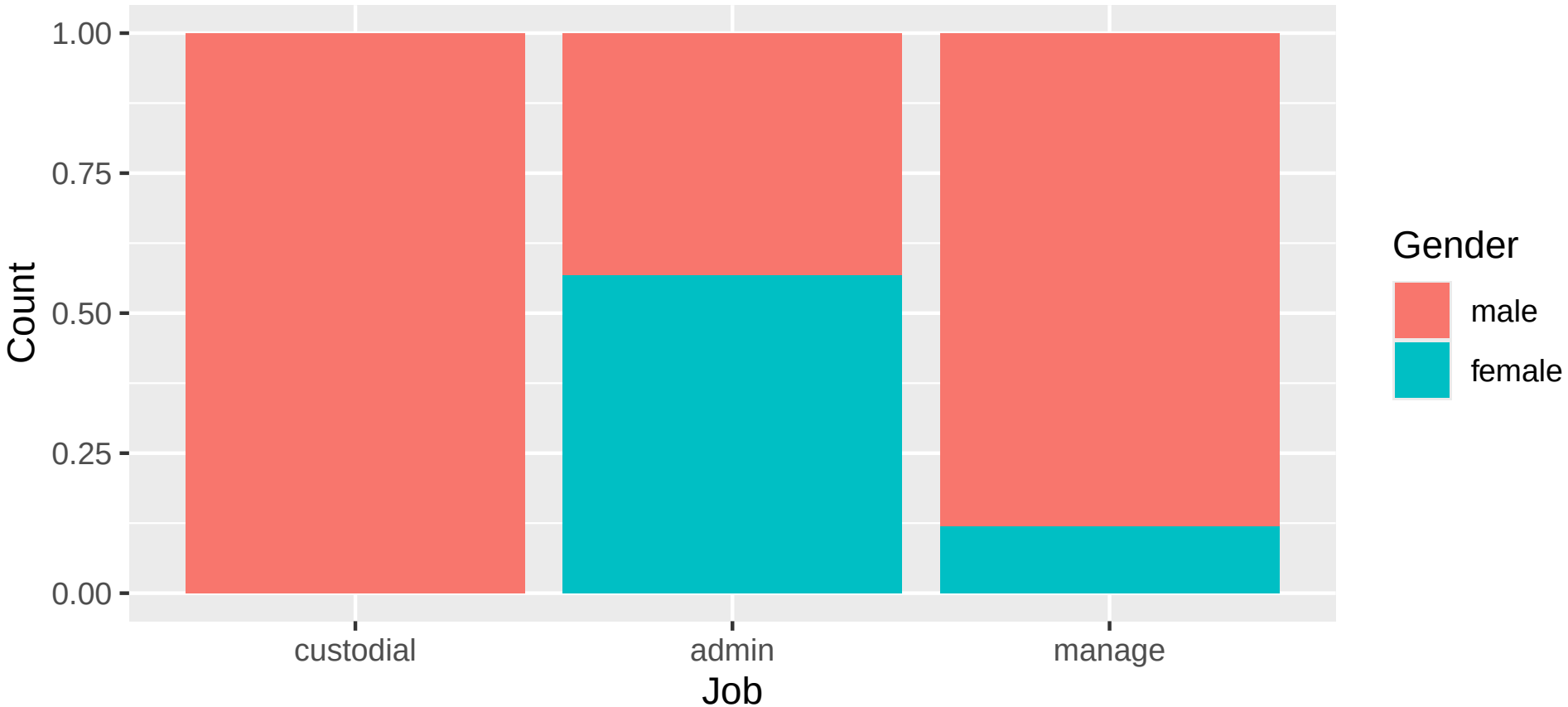
Bar plots


300
200
100
0
Count
custodial
admin
manage
Job
Gender
male
female

Bar plots

```
ggplot(BankWages, aes(x = job, fill = gender)) +
  geom_bar(position = "fill") +
  labs(x = "Job",
       y = "Count",
       fill = "Gender")
```

Bar plots


1.00
0.75
0.50
0.25
0.00
Count
custodial
admin
manage
Job
Gender
male
female

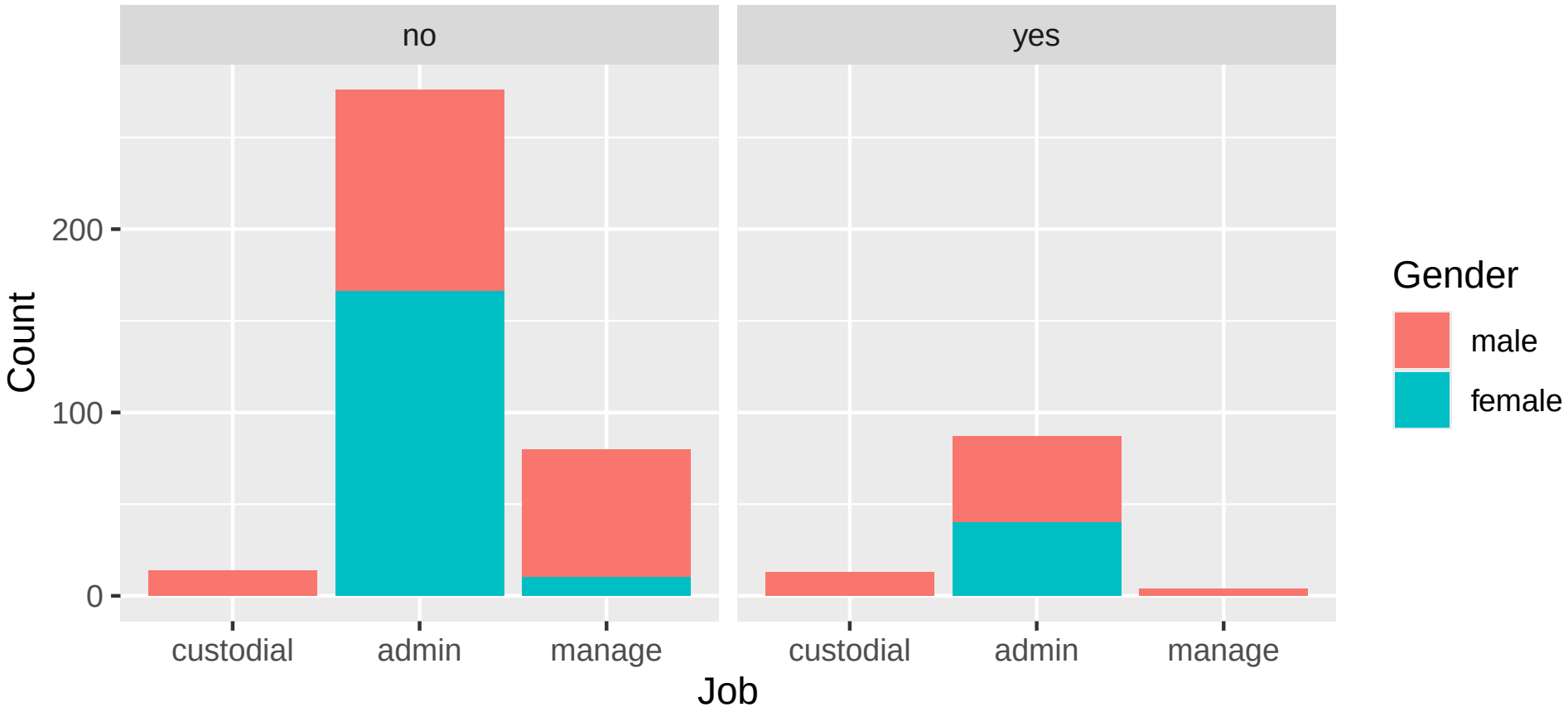
Faceting

- We can show more of the data using faceting
- Faceting is a way to produce multiple charts showing different subset of the data
- Splits the data by different variable(s) values
- We can change the scales to retail comparisons across facets or apply the scales only locally

Faceting

```
ggplot(BankWages, aes(x = job, fill = gender)) +
  geom_bar() +
  labs(x = "Job",
       y = "Count",
       fill = "Gender") +
  facet_wrap("minority")
```

Faceting


no
yes
200
100
0
Count
custodial
admin
manage
custodial
admin
manage
Job
Gender
male
female

Saving a chart

- Whenever possible use a vectorized format (pdf, eps, etc)

```
p_job_gender <-
  ggplot(BankWages, aes(x = job, fill = gender)) +
  geom_bar() +
  labs(x = "Job",
       y = "Count",
       fill = "Gender")

p_job_gender

ggsave(p_job_gender, filename = "job_gender.pgf",
       width = 6.5, height = 3, units = "in")
```

Review

The layered grammar defines the following components of a graphic:

- a default dataset and set of mappings from variables to aesthetics
- one or more layers, with each layer having one geometric object, one statistical transformations, one position adjustment, and optionally one dataset and set of aesthetic mappings
- one scale for each aesthetic mapping used
- a coordinate system
- the facet specification

Review

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