

# Big Data and Economics

## Lecture 1: Introduction

---

Kyle Coombs (he/him/his)  
Bates College | [EC/DCS 368](#)

# Table of contents

1. Prologue
2. Veggies: Syllabus highlights
3. Dessert: What is data science?
4. Appetizer: Getting started
5. Entree: R for data science
6. Second Dessert: Data visualization with ggplot2

# Prologue

---

# Introductions

## Course

 <https://github.com/big-data-and-economics>

You'll soon receive access to this GitHub organization, where we submit assignments, upload presentations, etc.

## Me

 [Kyle Coombs](#)

 [kcoombs@bates.edu](mailto:kcoombs@bates.edu)

 Assistant Professor (economics)

 From Scotia, New York

 Live in Maine and Massachusetts

 Research fields: Public and Labor, interested in applied econometrics and data science

# Syllabus highlights

---

(Read the full document [here](#).)

# Why this course?

Fill in the gaps left by traditional econometrics and methods classes.

- Practical skills that tools that will benefit your thesis and future career.
- Neglected skills like how to actually find datasets in the wild and clean them.
- Apply skills to analyze empirical questions on economic and social problems.

Data science skills are largely distinct from (and complementary to) the core 'metrics familiar to economists.

- Acquiring data; scraping; maintaining databases; etc.
- Data viz, cleaning and wrangling; programming; cloud computation; relational databases; machine learning; etc.

*"In short, we will cover things that I wish someone had taught me when I was starting out in college."*

# Caveat

- This course will be **hard**. You will need to:
  - Teach yourself new skills I cannot cover in 12 weeks
  - Be entrepreneurial: If you find a better way to do something, do (and share) it!
  - Be patient: You will encounter bugs and errors, and you will need to learn how to fix them
- This course will also be **rewarding**
  - You can avoid the mistakes you make here on your thesis and in your career
  - You will learn skills that employers, pre-doc programs, and grad schools want
  - You will learn how to be a better researcher and citizen
  - Seriously, a little data science goes a long way in helping you see through BS

# Tips for coding fruitfully

You're gonna write a lot of code for this class, which means you're gonna troubleshoot a lot of bugs.

- Some of these will be bugs of your own making, some will be bugs of open source R tools
1. Try to describe in plain words/simple pictures what you want code to do before you write it
    - Read in a CSV file with variables for annual population at the county level and calculate the change in population from the previous year
  2. Break this description into smaller steps (1: Read in data, 2: Drop rows with NA County, etc.)
  3. Write code "modularly" (in pieces) to do each step, then you can troubleshoot modularly
  4. The `help` documentation for R functions is the best place to start for troubleshooting
  5. Google is your dictionary, AI is your weird friend who knows a lot of words but sometimes uses them wrong<sup>1</sup>
    - Be precise in your Google searches and ChatGPT instructions
    - I encourage you to use Generative AI ([ChatGPT](#) and [GitHub CoPilot](#)) to help learn R and Git
  6. If you're stuck, ask for help from me and classmates on GitHub Discussions/Issues
    - Write a minimally reproducible example of the bug



# Assignments and grading

1. Exercises: low-stakes completion exercises
2. 4 group problem sets, drop the lowest, grade improvement allowed
3. Group presentations
4. Final project: TBD
5. GitHub Participation
6. Lewiston Hack-a-thon: 2-day data science competition at end of semester
  - Details TK

# Class outline

## Data science basics

- Version control with Git and GitHub
- R language basics
- Data cleaning and wrangling
- Webscraping
- Data visualization

## Analysis and Programming

- Spatial analysis
- Regression analysis
- Functions and loops
- Parallel programming

## Causal inference

- Regression discontinuity design
- Panel data and fixed effects
- Difference-in-difference design

## Scaling up: Big data, ML, and cloud computation

- Machine Learning techniques
- Text analysis

What is Data Science?

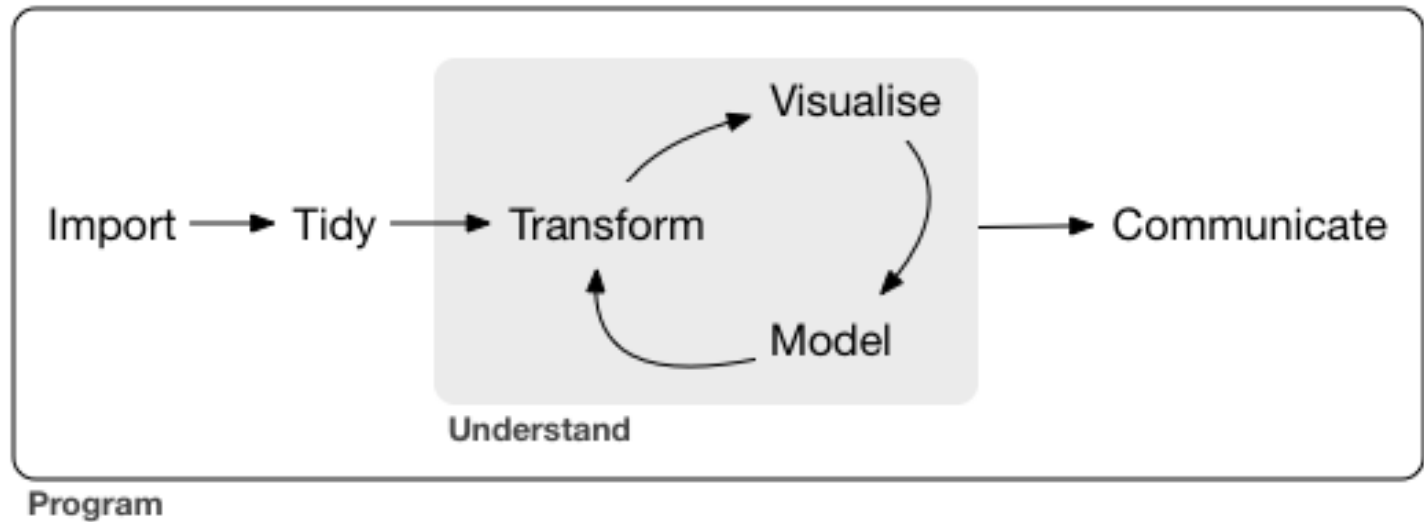
# What is Data Science?

- **Data science (DS):** The scientific discipline that deals with transforming data into useful information ("insights") using a variety of stats/ML techniques
  - Facebook: Collects data on search history, friendship links, site clicks, occupation, etc.
  - Chetty et al. (2022) used FB data to estimate users' SES and social network ([Social Capital Atlas](#))
- The rise of data science has come because of the so-called "Big" Data revolution
  - The rise of the internet in the late-1990s and 2000s  $\Rightarrow$   $\uparrow$  opportunities for companies and governments to collect data on consumers & citizens
  - Spread of mobile devices & social media from late 2000s until now generated even more data

## Pillars of data science

- Programming (automation of data collection, manipulation, cleaning, visualization, and modeling)
- Visualization & exploration
- Causal inference (to be able to make a policy prescription)
- Machine learning (to select models, compress data, predict outcomes)

# The data science workflow



Source: [R for Data Science](#)

# "Big" Data

Statistical information is currently accumulating at an unprecedented rate. But no amount of statistical information, however complete and exact, can by itself explain economic phenomena. If we are not to get lost in the overwhelming, bewildering mass of statistical data that are now becoming available, we need the guidance and help of a powerful theoretical framework. Without this no significant interpretation and coordination of our observations will be possible.

# "Big" Data

Statistical information is currently accumulating at an unprecedented rate. But no amount of statistical information, however complete and exact, can by itself explain economic phenomena. If we are not to get lost in the overwhelming, bewildering mass of statistical data that are now becoming available, we need the guidance and help of a powerful theoretical framework. Without this no significant interpretation and coordination of our observations will be possible.

Source: Frisch, Ragnar. 1933. "Editor's Note" *Econometrica* 1(1): 1-4

# Types of data

1. "Long" data (a.k.a. "Big- $N$ " data because  $N$  very, very large [and may not all fit onto a single hard drive!], government tax records, Medicare claims data, etc.)
2. "Wide" data (a.k.a. "Big- $K$ " data because  $K > N$ , customer data sets where each click is a variable)
3. "Wild" data (unstructured; happenstance; collected without a particular intention; e.g. twitter, contrast with Census surveys)
4. "Big Data" is a catch-all for any combination of the above data types that is hard to analyze with classical methods like OLS regression
  - Too many variables
  - Too many observations
  - Needs special wrangling or analysis



# Long data

The screenshot shows a Microsoft Excel spreadsheet titled "data\_examples - Excel". The spreadsheet contains a long dataset with the following columns: person\_id, income, years of education, and gender. The data is organized in a long format, with each row representing an individual. The first column (A) is labeled "person\_id", the second (B) is "income", the third (C) is "years of education", and the fourth (D) is "gender". The data spans from row 1 to row 44. The income values are in dollars, and the years of education are integers. The gender values are "F" for female and "M" for male. The spreadsheet interface includes the standard Excel ribbon with tabs for File, Home, Insert, Page Layout, Formulas, Data, Review, and View. The Home tab is active, showing options for Clipboard, Font, Paragraph, Styles, and Cells. The status bar at the bottom indicates "Ready" and "long wide Sheet3".

person_id	income	years of education	gender
101	\$ 8,825.23	12	F
102	\$38,356.11	14	M
103	\$ 8,641.73	13	F
104	\$10,024.09	13	M
105	\$79,923.36	12	M
106	\$57,007.00	14	M
107	\$59,494.84	15	F
108	\$92,150.41	13	M
109	\$75,373.30	13	F
110	\$15,680.30	13	M
111	\$46,593.41	13	F
112	\$71,386.71	15	M
113	\$72,674.96	11	M
114	\$58,535.12	12	M
115	\$11,968.91	12	F
116	\$99,265.27	14	M
117	\$46,181.11	11	F
118	\$74,175.59	15	M
119	\$73,409.86	11	F
120	\$65,784.26	14	M
121	\$ 3,532.26	14	M
122	\$33,836.95	15	M
123	\$56,806.58	13	F
124	\$68,478.31	13	M
125	\$60,566.22	15	F
126	\$98,447.41	13	F
127	\$79,397.90	11	F
128	\$17,594.75	12	F
129	\$84,667.93	13	M
130	\$87,953.71	13	M
131	\$68,423.74	14	F
132	\$51,357.62	13	M
133	\$82,233.86	12	F
134	\$92,901.91	14	M
135	\$75,153.35	13	M
136	\$29,740.94	15	M
137	\$ 795.36	13	F
138	\$27,283.46	12	M
139	\$ 1,137.37	12	F
140	\$61,127.80	13	M
141	\$33,153.06	12	F
142	\$19,774.73	15	M
143	\$55,925.97	13	M
144	\$15,508.81	15	M

- Main application: *identifying causal effects*
- Example: effects of improving schools on income

# Wide data

data\_examples (1) - Excel

FileHomeInsertPage LayoutFormulasDataReviewViewTell me what you want to do...

CutCopyFormat Painter

Clipboard

Calibri11

</

- Main application: *prediction*
- Example: predicting income to target ads from tons of information like location, links clicked, etc.

# Wild Data

```
<caption>List of men's Olympic records in athletics
</caption>
<tbody><tr>
<th scope="col" width="12%">Event
</th>
<th class="unsortable" width="5%">Record
</th>
<th scope="col" width="10%">Athlete(s)
</th>
<th scope="col" width="15%">Nation
```

# Why does data type matter?

1. Data type determines how much memory is required to store information
2. Data type determines what method you can use to read and analyze the data
  - A difference-in-difference model requires a different data shape than a regression discontinuity model
  - You cannot have a wide data set with one row per unit and one column per year

# The data you need depend on the

- Any dataset, no matter how big, has simplified the world in some way
- You want the simplification to match the question
- How do you record where a person is?
  - County? Lots of people have same location.
  - IP address? Changes frequently
  - GPS coordinates? Too precise, and changes every second!
- Your question and theory should guide your data collection
  - Are you curious about the effect of local government policies or firms on people?
  - Are you looking to measure the effect of air pollution on health?
  - Do you want to see how people change their commute patterns over time? When there is a road closure? <!--

# Big data & machine learning

- You'll often hear the phrase "big data and machine learning"
- This is because many machine learning algorithms are helpful for big data problems:
  - Selecting which  $k < K$  covariates should enter your model
  - Streamlined techniques for processing "wild" data
  - New modeling approaches that can leverage the greater amount of information that Big Data has

# What is machine learning? What is AI?

- **Machine learning (ML):** Allowing computers to learn for themselves without explicitly being programmed
  - USPS: Computer to read handwriting on envelopes
  - Google: AlphaGo, computer that defeated world champion Go player
  - Apple/Amazon/Microsoft: Siri, Alexa, Cortana, Talon voice assistants
- **Artificial intelligence (AI):** Constructing machines (robots, computers) to think and act like human beings
- ML is a subset of AI -->

# Getting started

---



# Software installation and registration

1. Download [R](#).
2. Download [RStudio](#).
3. Download [Git](#).
4. Create an account on [GitHub](#) and register for a student/educator [discount](#).
  - I will use GitHub to disseminate and submit assignments, receive feedback, etc.
5. Make a folder on your computer for this class. Any and all repositories for this class should be cloned into this folder.

If you had trouble completing any of these steps, please raise your hand.

- My go-to place for installation guidance and troubleshooting is Jenny Bryan's <http://happygitwithr.com>.

# Some OS-specific extras

I'll detail further software requirements as and when the need arises. However, to help smooth some software installation issues further down the road, please also do the following (depending on your OS):

- **Windows:** Install [Rtools](#). I also recommend that you install [Chocolatey](#) and [Windows Subsystem for Linux](#).
- **Mac:** Install [Homebrew](#). I also recommend that you configure/open your C++ toolchain (see [here](#).)
- **Linux:** None (you should be good to go).

# Checklist

- ☑ Do you have the most recent version of R?

```
version$version.string
```

```
## [1] "R version 4.3.3 (2024-02-29 ucrt)"
```

- ☑ Do you have the most recent version of RStudio?

```
RStudio.Version()$version
```

```
## Requires an interactive session but should return something like "[1] '1.4.1100'"
```

- ☑ Have you updated all of your R packages?

```
update.packages(ask = FALSE, checkBuilt = TRUE)
```

# Checklist (cont.)

Open up the [shell](#).

- Windows users, make sure that you installed a Bash-compatible version of the shell. If you installed [Git for Windows](#), then you should be good to go.

☒ Which version of Git have you installed?

```
git --version
```

```
## git version 2.34.1
```

☒ Did you introduce yourself to Git? (Substitute in your details.)

```
git config --global user.name 'kgcsport'  
git config --global user.email 'kcoombs@bates.edu'  
git config --global --list
```

☒ Did you register an account in GitHub?

# Checklist (cont.)

- Navigate to the [class materials repository](#)
- Click the green "Code" button and copy the HTTPS link.
- Under Codespaces, click `Create codespace on main`. This will create a cloud-based server for you to work on.
- It may take a few minutes to get up and running.
- Once inside, navigate to `PORTS` at the bottom, and click the link under "Local Address" for the RStudio Port. The username and password are rstudio/rstudio.
- You should now be in RStudio!
- Open up test.Rmd and click `Knit` at the top to test that it works!
- Navigate back to the main GitHub repository page and click `Code`, the three dots next to the codespace name, and `Delete codespace`.
- I'll encourage you to use Codespaces on some problem sets when I want to hit the ground running instead of troubleshooting package installation issues.
  - But you can use them for any problem set or exercise
- Use it today if your R installation is having issues!

# Checklist (cont.)

We will make sure that everything is working properly with your R and GitHub setup next lecture.

For the rest of today's lecture, I want to go over some very basic ChatGPT and R concepts.

PS — Just so you know where we're headed: We'll return to these R concepts (and delve much deeper) next week after a brief, but important detour to the lands of coding best practices and Git(Hub).

# R and Generative AI for data science

---

# Why R and RStudio?

## Data science positivism

- Alongside Python, R has become the *de facto* language for data science.
  - See: [The Impressive Growth of R](#), [The Popularity of Data Science Software](#)
- Open-source (free!) with a global user-base spanning academia and industry.
  - "Do you want to be a profit source or a cost center?"

## Bridge to applied economics and other tools

- Already has all of the statistics and econometrics support, and is amazingly adaptable as a “glue” language to other programming languages and APIs.
- The RStudio IDE and ecosystem allow for further, seamless integration.

## Path dependency

- It's also the language that I know best.
- (Learning multiple languages is a good idea, though.)



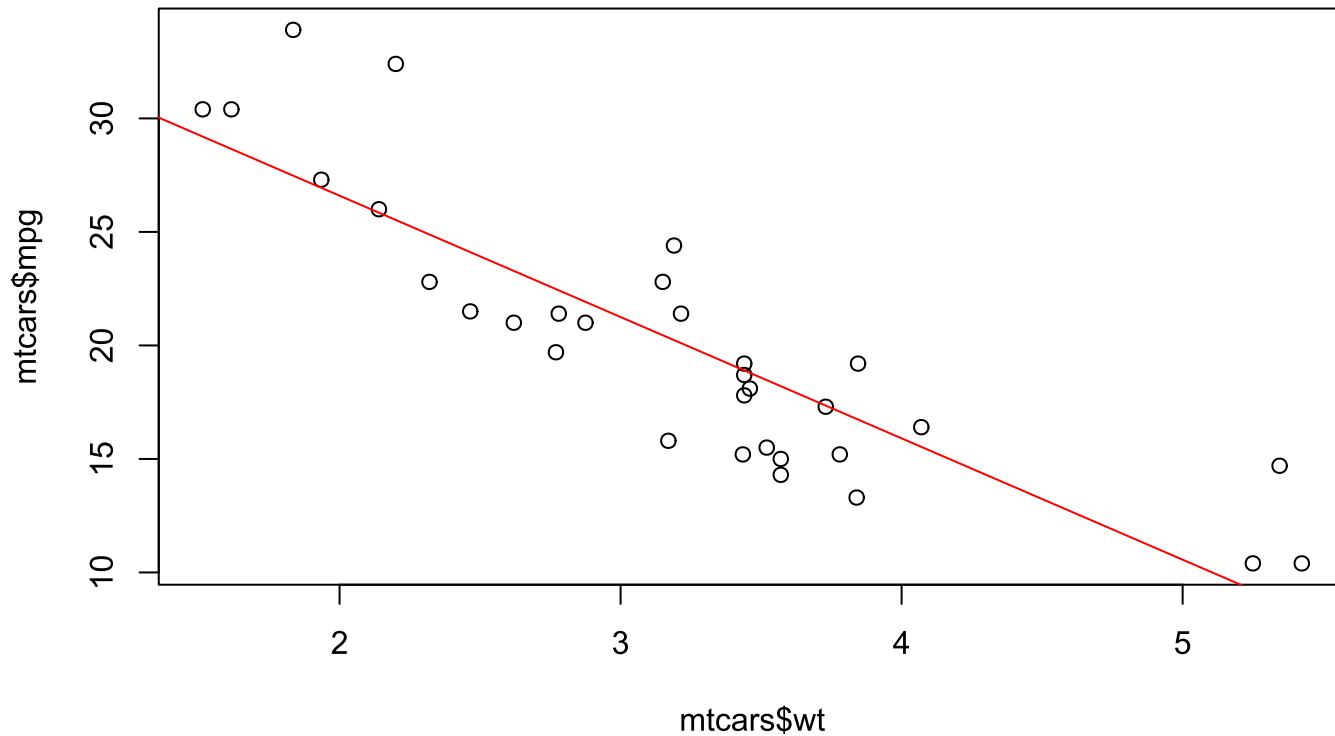
# R code example (linear regression)

```
fit = lm(mpg ~ wt, data = mtcars)
summary(fit)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776   19.858  < 2e-16 ***
## wt          -5.3445     0.5591   -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

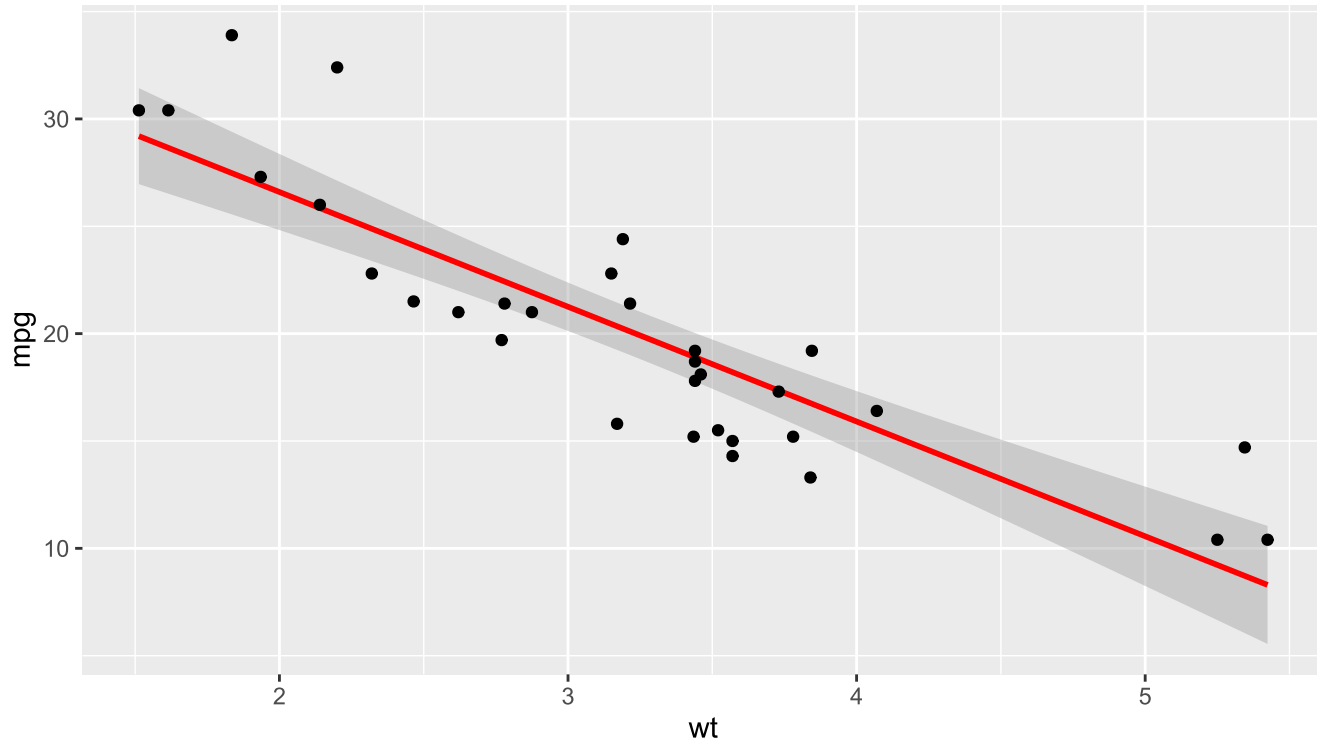
# Base R plot

```
par(mar = c(4, 4, 1, .1)) ## Just for nice plot margins on this slide deck  
plot(mtcars$wt, mtcars$mpg)  
abline(fit, col = "red") # Add in the object fit from before!
```



# ggplot2

```
library(ggplot2)
ggplot(data = mtcars, aes(x = wt, y = mpg)) +
  geom_smooth(method = "lm", col = "red") +
  geom_point()
```



# Do ↑ in GDP cause life expectancy to ↑?

- Let's use our new-found R knowledge to try to separate correlation from causation for a critical question in economics:
  - Does increasing the economic pie (GDP) lead to longer lives (life expectancy)?
- We can use the gapminder dataset to explore this question
- The [gapminder](#) dataset contains panel data on life expectancy, population size, and GDP per capita for 142 countries since the 1950s
- Any predictions about what we'll learn?
- Also, we can tap into generative AI to help us climb the steep learning curve of coding in a new language

# More ggplot2

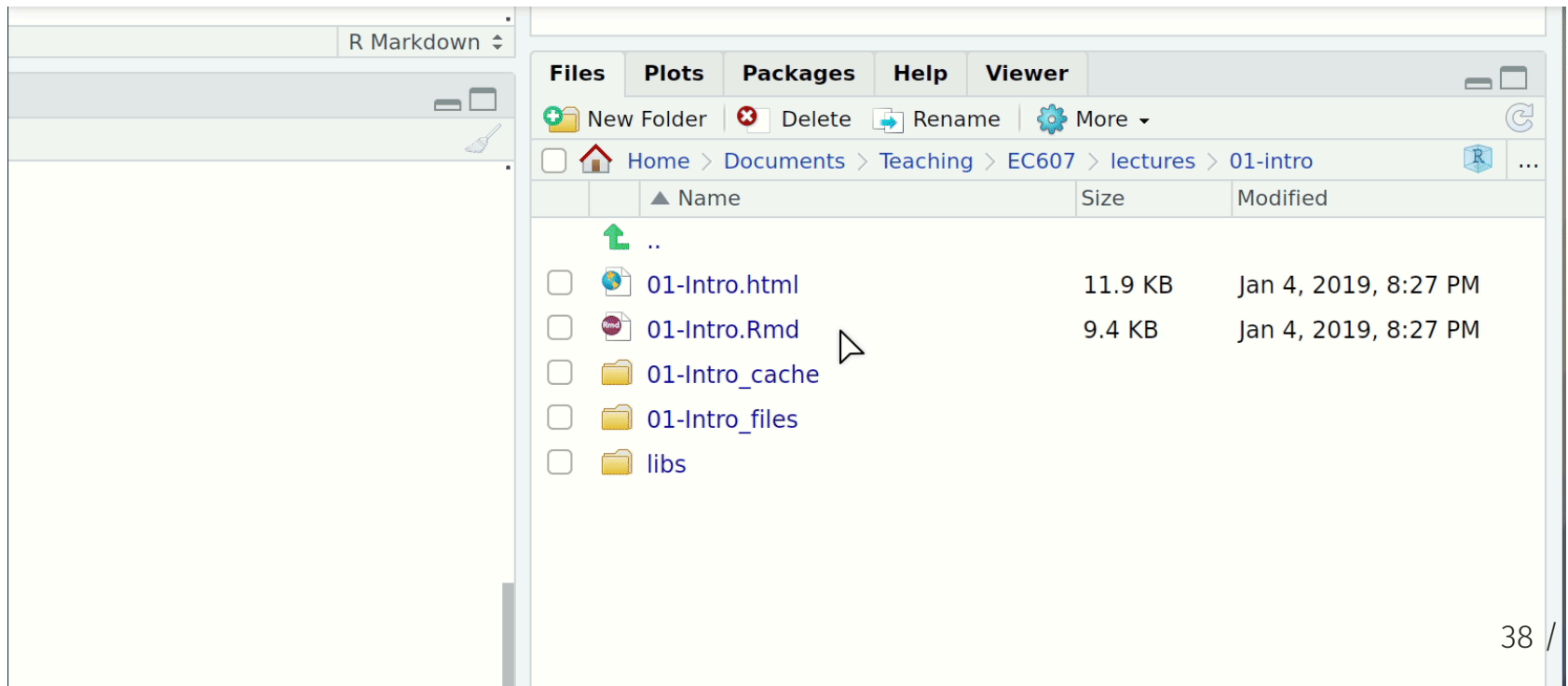
---

# Install and load

Open up your laptops. For the remainder of this first lecture, we're going to play around with **ggplot2** (i.e. livecoding) to explore our question about GDP and life expectancy.

If you don't have them already, install the **ggplot2** and **gapminder** and **swirl** packages via either:

- **Console:** Enter `install.packages(c("ggplot2", "gapminder", "swirl"), dependencies=T)`.
- **RStudio:** Click the "Packages" tab in the bottom-right window pane. Then click "Install" and search for these two packages.



# Install and load (cont.)

Once the packages are installed, load them into your R session with the `library()` function.

```
library(ggplot2)
library(gapminder) ## We're just using this package for the gapminder data
library(swirl)
```

Notice too that you don't need quotes around the package names any more. Reason: R now recognises these packages as defined objects with given names.

PS — A convenient way to combine the package installation and loading steps is with the **pacman** package's `p_load()` function. If you run `pacman::p_load(ggplot2, gapminder)` it will first look to see whether it needs to install either package before loading them. Clever.

- We'll get to this later, but if you want to run a function from an (installed) package without loading it, you can use the `PACKAGE::package_function()` syntax.

# Exploratory analysis

- What initial exploratory questions do we have? Shout them out!
- Here's what I want to know:
  1. How are the data organized?
  2. What are the unique countries in the dataset?
  3. What does the relationship between GDP per capita and life expectancy look like?
- We'll tackle the third today since it is the most visually interesting



# Swirl installation

- Swirl is an R package that teaches you R interactively
- It's a great way to learn R if you're new to the language
- Use this code to get it up and running

```
library(swirl)
install_course_github('big-data-and-economics','ECON368-R-Swirls')
swirl()
```

- Tell swirl what it should call you (e.g. "Wizard of R")
- Then select the course `ECON368-R-Swirls` and the lesson `GGPlot Demo` (#13 last time I checked)
- Work through interactively

# Leverage ChatGPT and GitHub CoPilot

- I'm going to ask you to use a number of new technical tools in this class
- The documentation will be concise and oftentimes you won't know where to start
  - For example, how in the world are you supposed to make a ggplot plot of the gapminder data if you've never used ggplot before?
  - What if you've never even opened R? How do you even look at the gapminder data?
- I strongly encourage you to use ChatGPT and GitHub CoPilot in all aspects of this class
- ChatGPT is a large language model (LLM) developed by OpenAI
- Create an account at <http://chat.openai.com/>
- ChatGPT is excellent at producing basic code and helping resolve bugs
- It serves as a bridge between the abstract ideas in your head and code you write

# ChatGPT example

First, submit a [prompt on Mentimeter](#) that loads the gapminder data, displays some key checks, then makes a scatterplot of GDP per capita and life expectancy

## **Human:**

Write an R script that does the following:

- (a) Reads in the gapminder dataset using the library gapminder.
- (b) Prints out the first 10 rows of the gapminder dataset.
- (c) Lists which countries are in the sample (i.e. list unique values of the country variable)
- (d) Plots a scatterplot of the gdpPercap by lifeExp using the gapminder dataset using ggplot.

# ChatGPT example (continued)

## Assistant:

Certainly! Here's an R script that accomplishes these tasks using the gapminder dataset:

```
# Load necessary libraries  
library(gapminder)  
library(ggplot2)  
  
# Step (a) - Read in the gapminder dataset  
data("gapminder")
```

This script assumes that the gapminder package is installed and loaded. It reads in the gapminder dataset, prints the first 10 rows, lists unique countries present in the dataset, and then creates a scatterplot of gdpPercap (GDP per Capita) against lifeExp (Life Expectancy) using ggplot2. Adjust the plot aesthetics as needed to suit your preferences.

---

Exported on 1/8/2024.

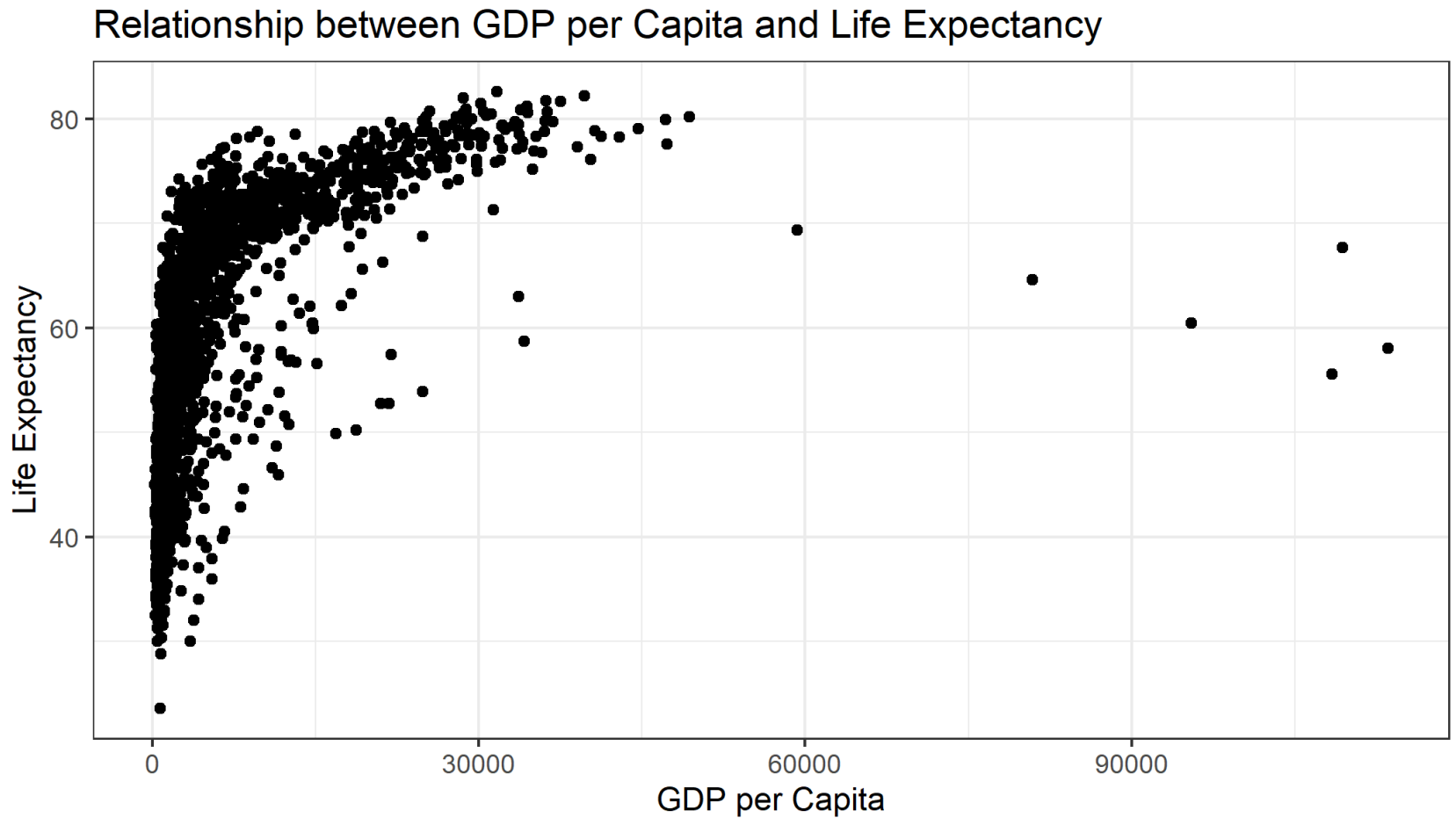
# How's the code run?

```
## # A tibble: 10 × 6
##   country      continent  year lifeExp      pop gdpPercap
##   <fct>        <fct>    <int>  <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952   28.8  8425333    779.
## 2 Afghanistan Asia      1957   30.3  9240934    821.
## 3 Afghanistan Asia      1962   32.0 10267083    853.
## 4 Afghanistan Asia      1967   34.0 11537966    836.
## 5 Afghanistan Asia      1972   36.1 13079460    740.
## 6 Afghanistan Asia      1977   38.4 14880372    786.
## 7 Afghanistan Asia      1982   39.9 12881816    978.
## 8 Afghanistan Asia      1987   40.8 13867957    852.
## 9 Afghanistan Asia      1992   41.7 16317921    649.
## 10 Afghanistan Asia      1997   41.8 22227415    635.
```

```
##   [1] Afghanistan      Albania      Algeria
##   [4] Angola            Argentina    Australia
##   [7] Austria           Bahrain      Bangladesh
##  [10] Belgium           Benin        Bolivia
##  [13] Bosnia and Herzegovina Botswana     Brazil
##  [16] Bulgaria          Burkina Faso Burundi
##  [19] Cambodia          Cameroon     Canada
##  [22] Central African Republic Chad           Chile
##  [25] China             Colombia     Comoros
##  [28] Congo, Dem. Rep.  Congo, Rep.  Costa Rica
##  [31] Cote d'Ivoire     Croatia      Cuba
##  [34] Czech Republic   Denmark      Djibouti
##  [37] Dominican Republic Ecuador        Egypt
##  [40] El Salvador      Equatorial Guinea Eritrea
##  [43] Ethiopia         Finland      France
```

# How's the code run? (cont.)



# Tips for using ChatGPT

- Think of it as a more interactive version of Googling for the solution to a bug
  - It is not a replacement for you, the programmer
  - Think through the basic coding tasks first, then ask AI to fill in the blanks
- Be as specific as possible in your instructions
  - If you know the name of the variables in your dataset, use them
- Think of it as a more interactive version of Googling for the solution to a bug
- Try things iteratively and in small steps
  - If you're not sure how to do something, try to break it down into smaller steps
  - This is a good tip for coding in general
- Your brain is still the most powerful tool you have
  - ChatGPT is a tool to help you, not replace you
  - You will not get much mileage if you say, "Read in the gapminder dataset and do something interesting with it"
- Often it only provides "skeleton code", so you'll need to fill in the blanks

# Elements of ggplot2

Hadley Wickham's ggplot2 is one of the most popular packages in the entire R canon.

- It also happens to be built upon some deep visualization theory: i.e. Leland Wilkinson's *The Grammar of Graphics*.

There's a lot to say about ggplot2's implementation of this "grammar of graphics" approach, but the three key elements are:

1. Your plot ("the visualization") is linked to your variables ("the data") through various **aesthetic mappings**.
2. Once the aesthetic mappings are defined, you can represent your data in different ways by choosing different **geoms** (i.e. "geometric objects" like points, lines or bars).
3. You build your plot in **layers**.

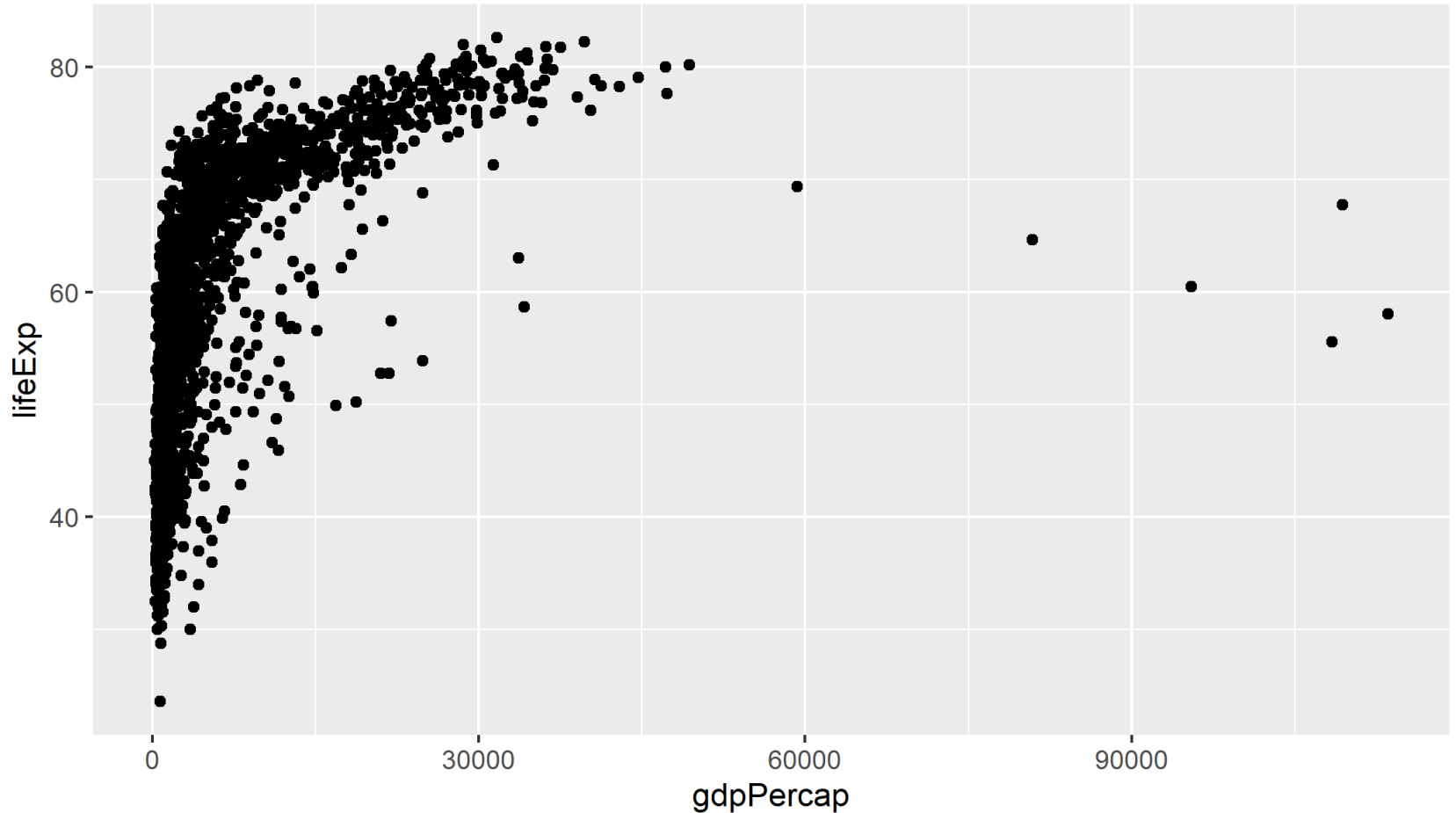
That's kind of abstract. Let's break down the elements of ggplot2 in turn with some actual plots.

- As a shortcut, we'll use AI to write the basic code for us then we'll fill in the blanks.



# 1. Aesthetic mappings

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +  
  geom_point()
```



# 1. Aesthetic mappings (cont.)

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +  
  geom_point()
```

Focus on the top line, which contains the initialising `ggplot()` function call. This function accepts various arguments, including:

- Where the data come from (i.e. `data = gapminder`).
- What the aesthetic mappings are (i.e. `mapping = aes(x = gdpPercap, y = lifeExp)`).

The aesthetic mappings here are pretty simple: They just define an x-axis (GDP per capita) and a y-axis (life expectancy).

- To get a sense of the power and flexibility that comes with this approach, however, consider what happens if we add more aesthetics to the plot call...

# 1. Aesthetic mappings (cont.)

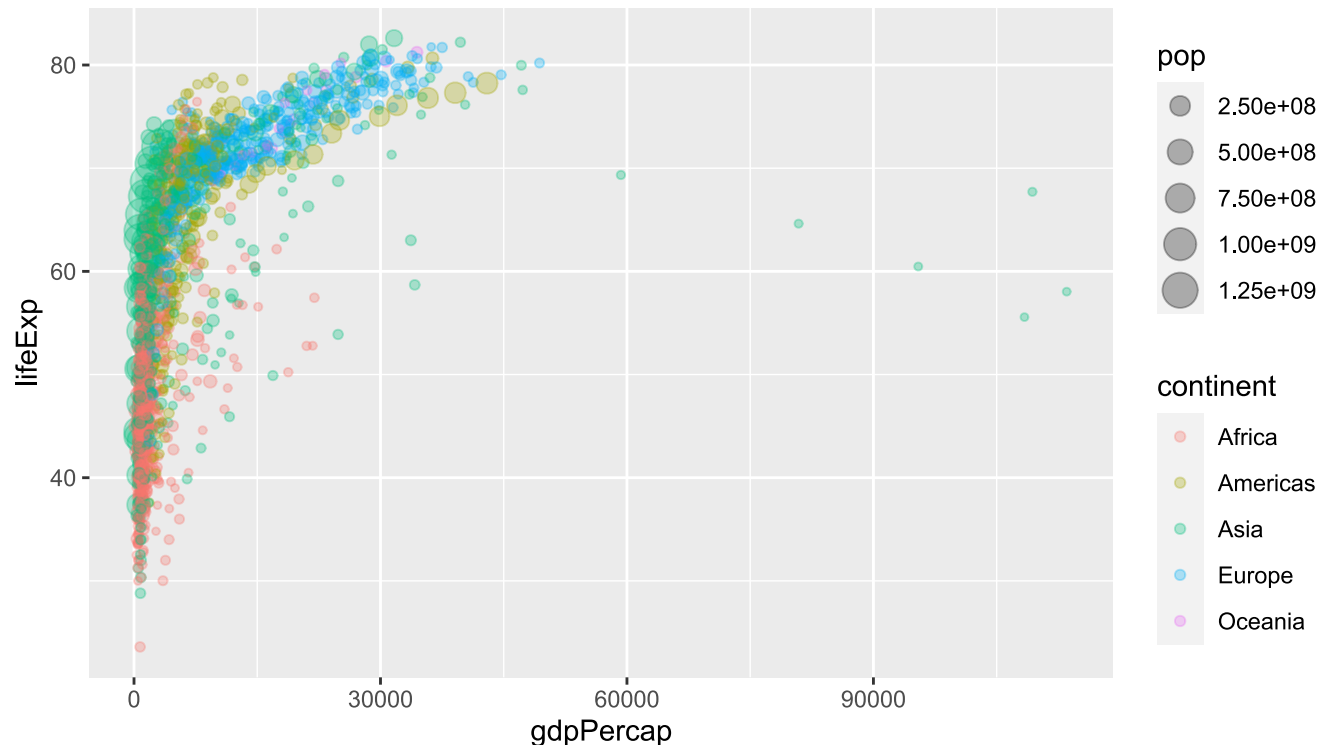
- Can you ask ChatGPT to add color the dots by continent?

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +  
  geom_point()
```

- Submit your guesses on [Mentimeter](#).

# 1. Aesthetic mappings (cont.)

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, col = continent)) +  
  geom_point(alpha = 0.3) ## "alpha" controls transparency. Takes a value between 0 and 1.
```



- I've dropped the "mapping =" part of the ggplot call. `ggplot2` knows the order of the arguments.

# 1. Aesthetic mappings (cont.)

The aesthetics expect "long" data. This is distinct from the "wide" format, where each series has its own column.

## Long will work

```
## # A tibble: 6 × 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int>  <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952   28.8  8425333    779.
## 2 Afghanistan Asia      1957   30.3  9240934    821.
## 3 Afghanistan Asia      1962   32.0 10267083    853.
## 4 Afghanistan Asia      1967   34.0 11537966    836.
## 5 Afghanistan Asia      1972   36.1 13079460    740.
## 6 Afghanistan Asia      1977   38.4 14880372    786.
```

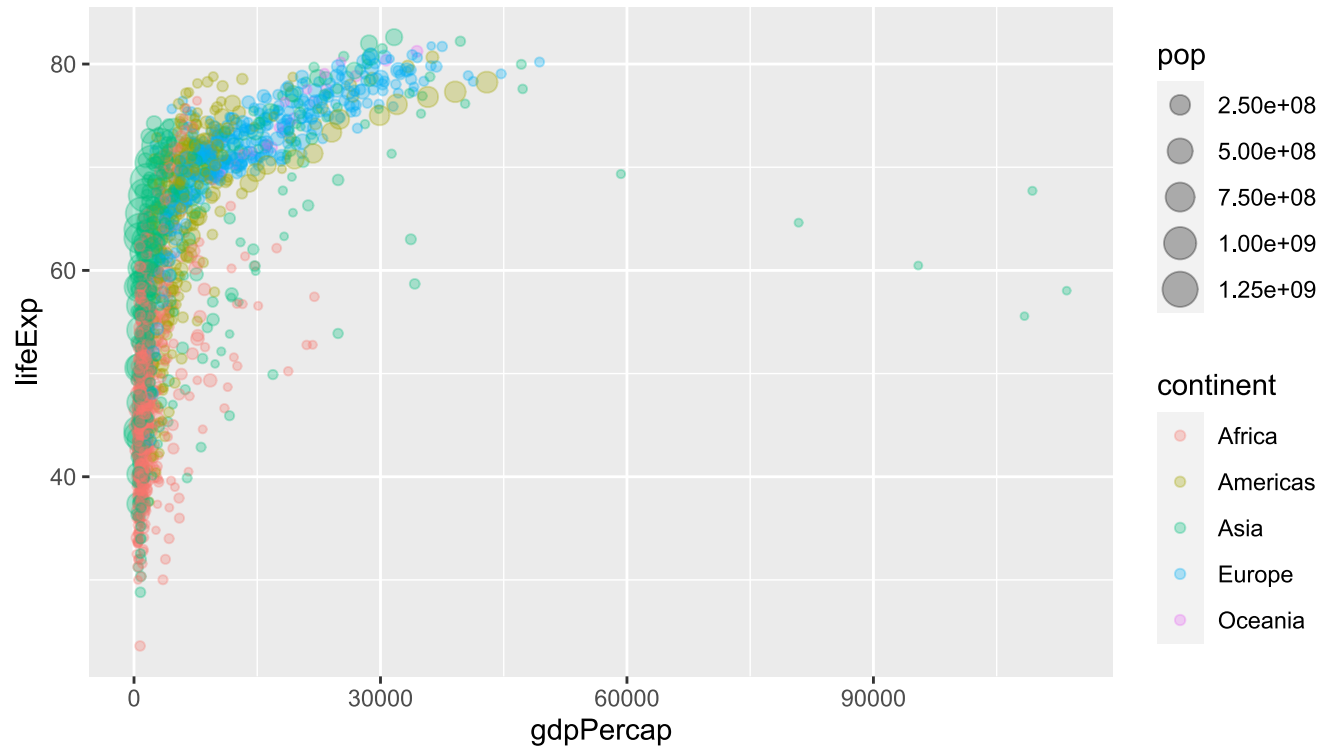
## Wide would not work

```
## # A tibble: 142 × 26
##   country    continent lifeExp_1952 lifeExp_1957 lifeExp_1962 lifeExp_1967
##   <fct>      <fct>          <dbl>      <dbl>      <dbl>      <dbl>
## 1 Afghanistan Asia          28.8        30.3        32.0        34.0
## 2 Albania     Europe          55.2        59.3        64.8        66.2
## 3 Algeria     Africa          43.1        45.7        48.3        51.4
## 4 Angola      Africa          30.0        32.0        34          36.0
## 5 Argentina   Americas        62.5        64.4        65.1        65.6
## 6 Australia   Oceania         69.1        70.3        70.9        71.1
## 7 Austria     Europe          66.8        67.5        69.5        70.1
```

# 1. Aesthetic mappings (cont.)

We can specify aesthetic mappings in the geom layer too.

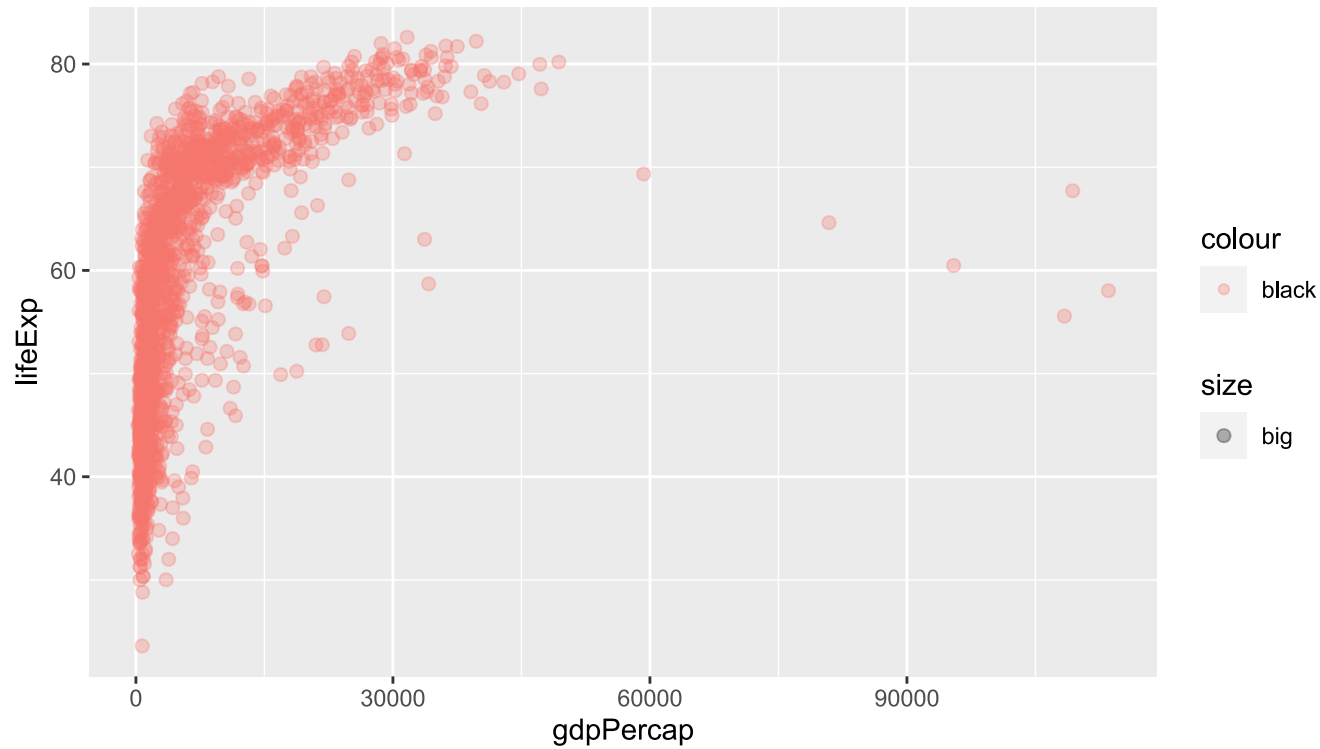
```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) + ## Applicable to all geoms  
  geom_point(aes(size = pop, col = continent), alpha = 0.3) ## Applicable to this geom only
```



# 1. Aesthetic mappings (cont.)

Oops. What went wrong here?

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +  
  geom_point(aes(size = "big", col="black"), alpha = 0.3)
```

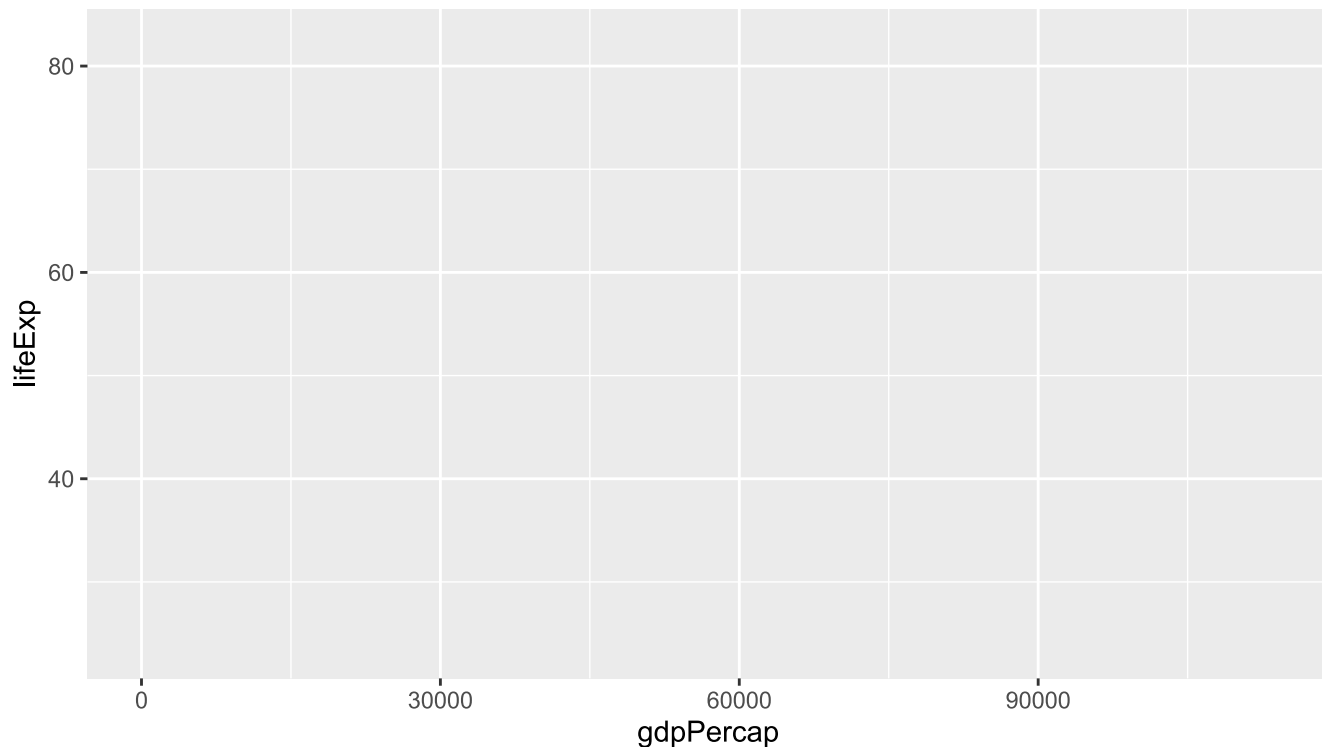


**Answer:** Aesthetics must be mapped to variables, not descriptions!

# 1. Aesthetic mappings (cont.)

At this point, instead of repeating the same ggplot2 call every time, it will prove convenient to define an intermediate plot object that we can re-use.

```
p = ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp))  
p
```

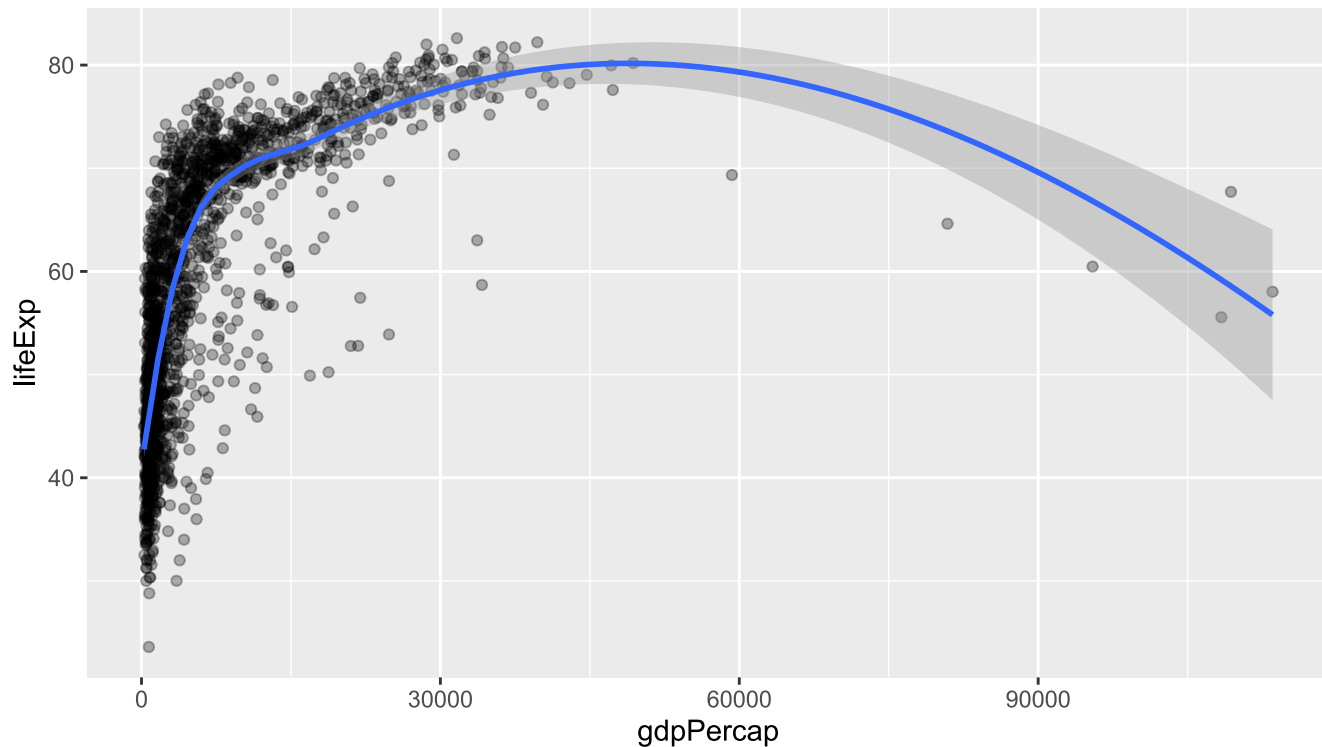




## 2. Geoms

Once your variable relationships have been defined by the aesthetic mappings, you can invoke and combine different geoms to generate different visualizations.

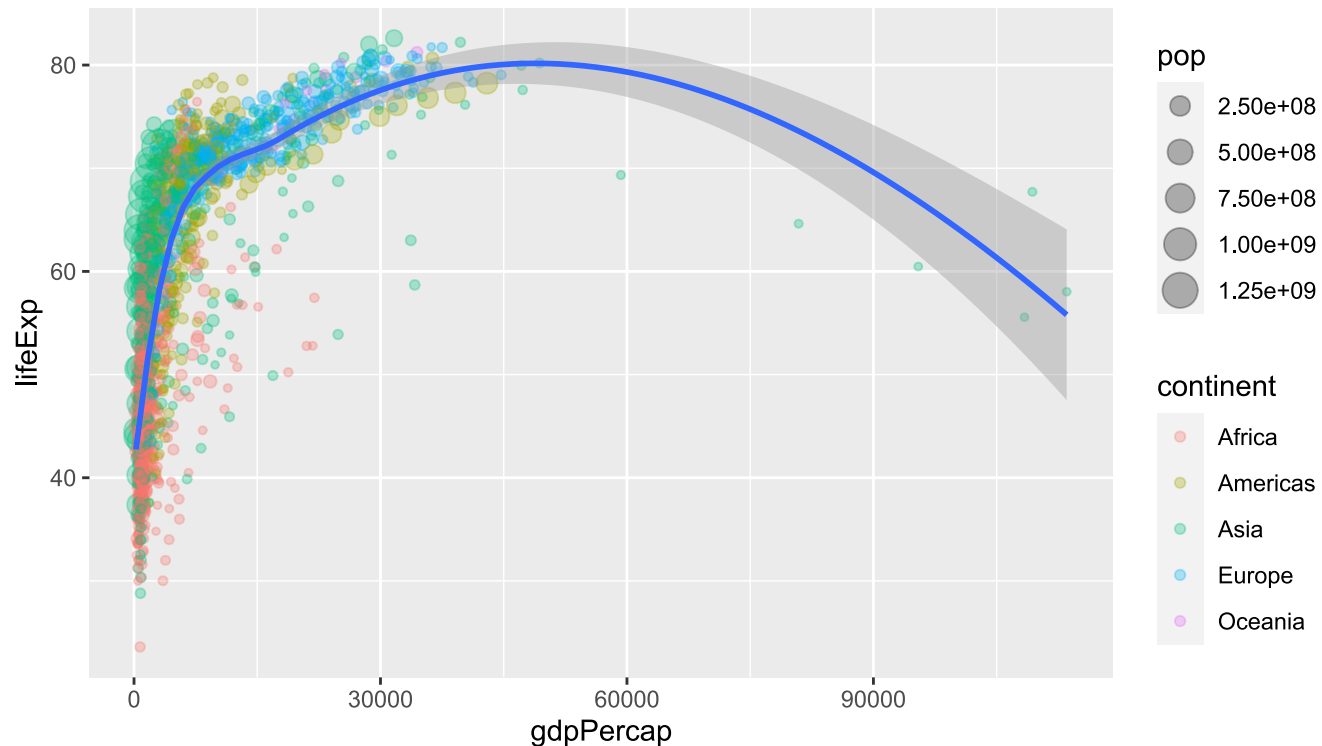
```
p +  
  geom_point(alpha = 0.3) +  
  geom_smooth(method = "loess") # A smoothed "locally estimated scatterplot smoothing" line
```



## 2. Geoms (cont.)

Aesthetics can be applied differentially across geoms.

```
p +  
  geom_point(aes(size = pop, col = continent), alpha = 0.3) +  
  geom_smooth(method = "loess")
```



## 2. Geoms (cont.)

The previous plot provides a good illustration of the power (or effect) that comes from assigning aesthetic mappings "globally" vs in the individual geom layers.

- Compare: What happens if you run the below code chunk?

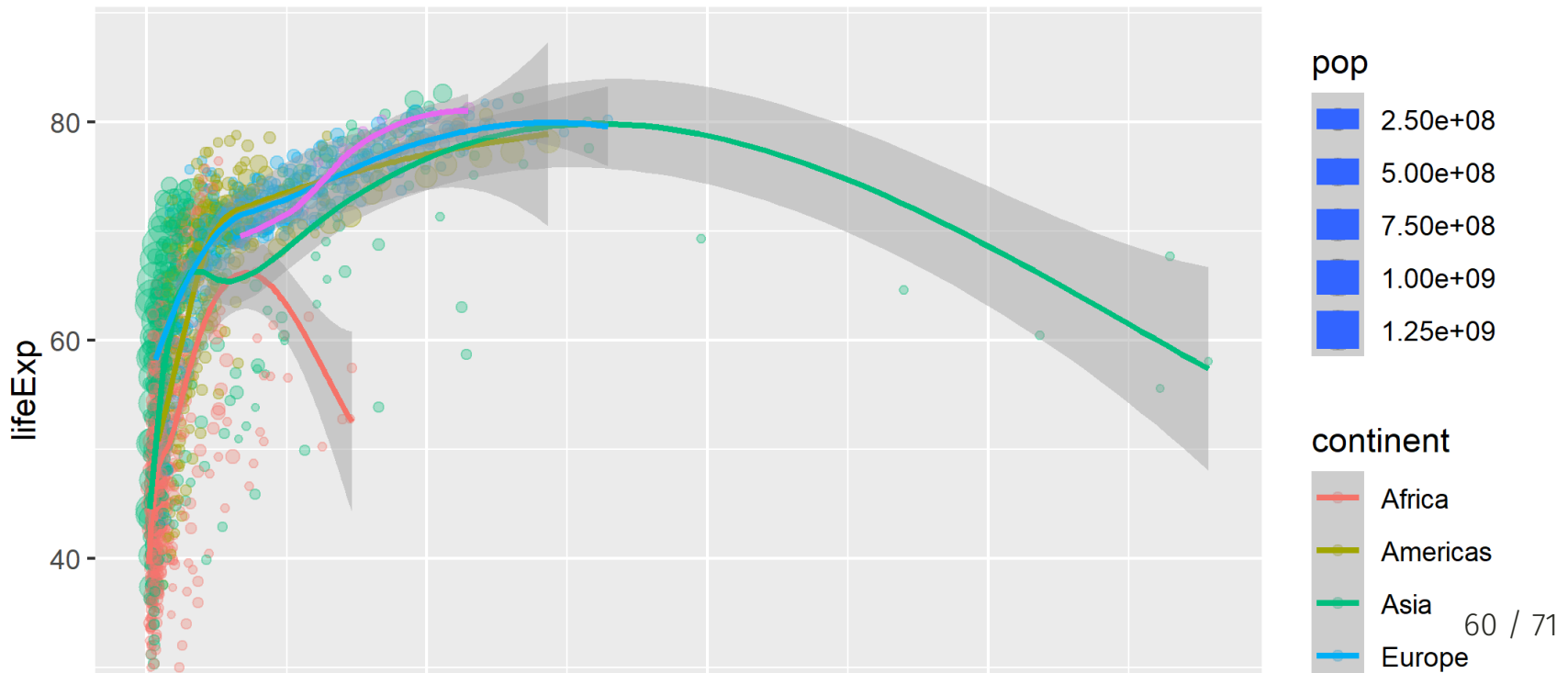
```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, col = continent)) +  
  geom_point(alpha = 0.3) +  
  geom_smooth(method = "loess")
```

## 2. Geoms (cont.)


The previous plot provides a good illustration of the power (or effect) that comes from assigning aesthetic mappings "globally" vs in the individual geom layers.

- Compare: What happens if you run the below code chunk?

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, col = continent)) +  
  geom_point(alpha = 0.3) +  
  geom_smooth(method = "loess")
```



# 3. Build your plot in layers

We've already seen how we can chain (or "layer") consecutive plot elements using the  connector.

- The fact that we can create and then re-use an intermediate plot object (e.g. "p") is testament to this.

But it bears repeating: You can build out some truly impressive complexity and transformation of your visualization through this simple layering process.

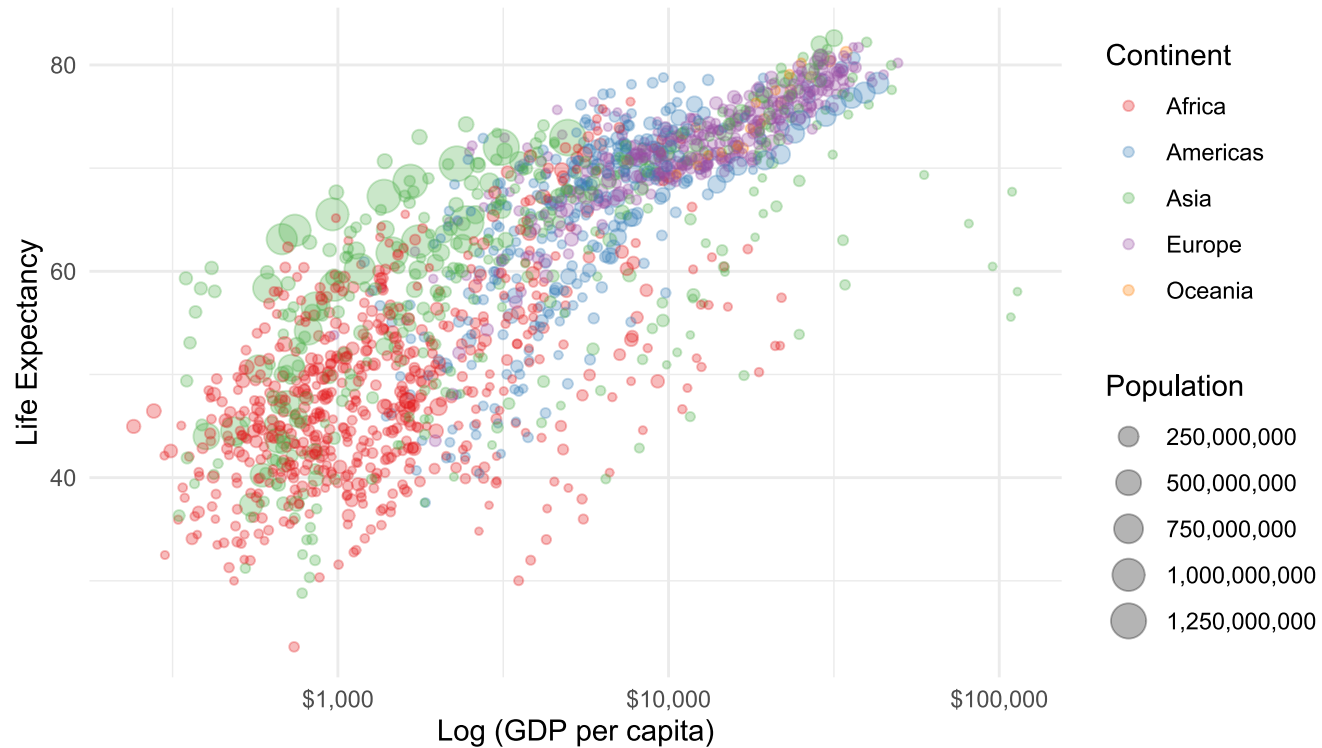
- You don't have to transform your original data; ggplot2 takes care of all of that.
- For example (see next slide for figure).
- **Bonus:** Maybe this will help make sense of the non-linear relationship between GDP per capita and life expectancy?

# Build your plot in layers (cont.)

```
p2 =  
  p +  
  geom_point(aes(size = pop, col = continent), alpha = 0.3) +  
  scale_color_brewer(name = "Continent", palette = "Set1") + ## Different colour scale  
  scale_size(name = "Population", labels = scales::comma) + ## Different point (i.e. legend) scale  
  scale_x_log10(labels = scales::dollar) + ## Switch to logarithmic scale on x-axis. Use dollar units.  
  labs(x = "Log (GDP per capita)", y = "Life Expectancy") + ## Better axis titles  
  theme_minimal() ## Try a minimal (b&w) plot theme
```

- Before executing, what will this do?
- The comments help, as will Google and ChatGPT, but the function names are somewhat intuitive too.

# 3. Build your plot in layers (cont.)



# What else?

We have barely scratched the surface of ggplot2's or ChatGPT's functionality... let alone talked about the entire ecosystem of packages that has been built around it.

- Here's are an additional example to whet your appetite

Note that you will need to install and load some additional packages if you want to recreate the next two figures on your own machine. A quick way to do this:

```
if (!require("pacman")) install.packages("pacman")
pacman::p_load(hrbrthemes, gganimate)
```

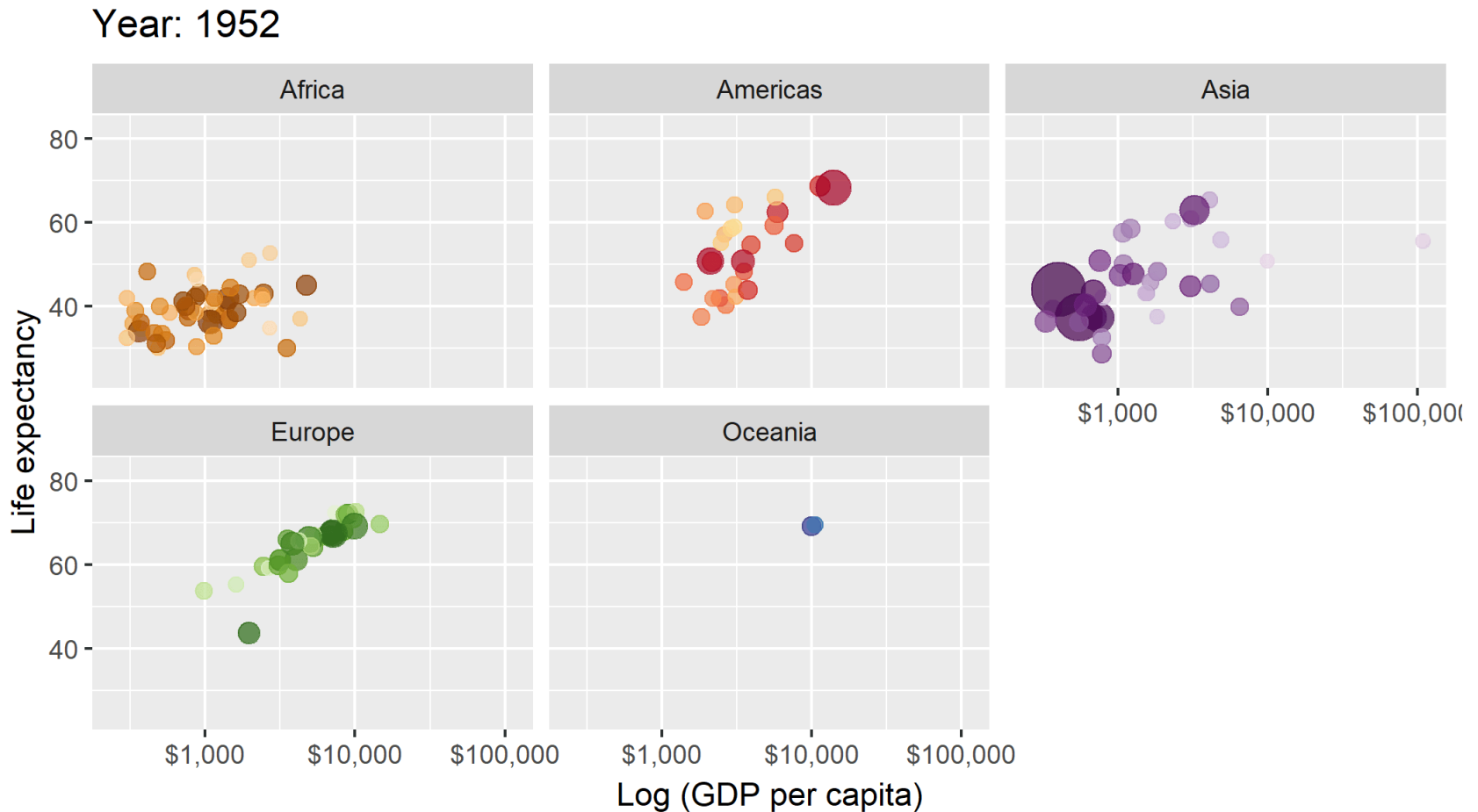
Animation! (See the next slide for the resulting GIF.)

```
# library(gganimate)
ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10(labels = scales::dollar) +
  facet_wrap(~continent) +
  # Here comes the gganimate specific bits
  labs(title = 'Year: {frame_time}', x = 'Log (GDP per capita)', y = 'Life expectancy') +
  transition_time(year) +
  ease_aes('linear')
```

- What is different about this code?



# What else? (cont.)



Note that this animated plot provides a much more intuitive understanding of the underlying data. Just as [Hans Rosling](#) intended.

# But do GDPPC increases cause life

- We can't answer this question with a simple plot.
- We also can't answer this question with a very fancy plot.
- What do we need?
  - A model
  - A causal identification strategy
  - More granular (bigger) data

# What else? (cont.)

We also haven't touched on ggplot2's relationship to "tidy" data.

- It actually forms part of a suite of packages collectively known as the **tidyverse**.
- We will get back to this.

Rest assured, you will be using ggplot2 throughout the rest of this course and developing your skills along the way.

- Your second assignment (coming up) is a chance specifically to hone some of those skills.

You can do some reading and practice on your own. Pick any of the following (or choose among the litany of online resources) and work through their examples:

- **Chapter 3** of *R for Data Science* by Hadley Wickham and Garrett Grolemund.
- ***Data Visualization: A Practical Guide*** by Kieran Healy.
- **Designing ggplots** by Malcom Barrett.

For next class on GitHub, please complete the following:

- Exercise 0
- Work through Chapters 1-14 of <https://happygitwithr.com/>
- Read through the **Git fundamentals** unit

Next lecture: Deep dive into Git(Hub).

---

# Appendix

# Some R basics

1. Everything is an object.
2. Everything has a name.
3. You do things using functions.
4. Functions come pre-written in packages (i.e. "libraries"), although you can — and should — write your own functions too.

Points 1. and 2. can be summarised as an **object-oriented programming** (OOP) approach.

- This may sound super abstract now, but we'll see *lots* of examples over the coming weeks that will make things clear.

# R vs Stata

If you're coming from Stata, some additional things worth emphasizing:

- Multiple objects (e.g. data frames) can exist happily in the same workspace.
  - No more `keep`, `preserve`, `restore` hackery. (Though, props to [Stata 16](#).)
  - This is a direct consequence of the OOP approach.
- You will load packages at the start of every new R session. Make peace with this.
  - "Base" R comes with tons of useful in-built functions. It also provides all the tools necessary for you to write your own functions.
  - However, many of R's best data science functions and tools come from external packages written by other users.
- R easily and infinitely parallelizes. For free.
  - Compare the cost of a [Stata/MP](#) license, nevermind the fact that you effectively pay per core...
- You don't need to `tsset` or `xtset` your data. (Although you can too.)