

Data Visualization in R

The Very (Very!) Basics

Ciara Zogheib

iSkills Workshop 2023

I wish to acknowledge this land on which the University of Toronto operates. For thousands of years it has been the traditional land of the Huron-Wendat, the Seneca, and the Mississaugas of the Credit. Today, this meeting place is still the home to many Indigenous people from across Turtle Island and we are grateful to have the opportunity to work on this land.

Indigenous Data Science - Resources

- U of T's Indigenous Data Sovereignty Guide <https://guides.library.utoronto.ca/indigenoustudies/datasovereignty>
- CARE Principles for Indigenous Data Governance <https://www.gida-global.org/care>
- First Nations Principles of OCAP <https://fnigc.ca/ocap-training/>

Hello!

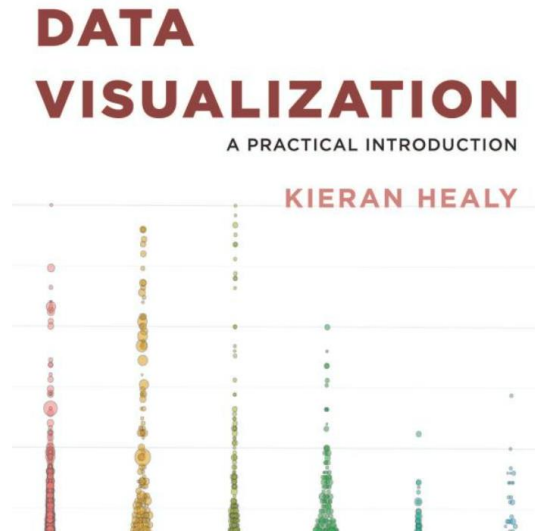
- Ciara Zogheib (She/Her)
- PhD Student at the Faculty of Information
- Working as a data scientist/researcher with the government

What are we doing today?

- Data viz concepts speedrun
- Ggplot basics!

Reference

- Practice code is modified from Healy, K. (2018). Data Visualization: A Practical Introduction. Princeton University Press.



RStudio

- Should be already installed, but if not, don't worry!
- Alternatives for this workshop:
 - <https://rdr.io/snippets/>
 - <https://rstudio-with-jupyterhub-uoft.netlify.app/#3>

Data Viz Concepts Speedrun

Qualities of Data Visualizations

- Is the visualization pleasing to look at? → **Aesthetic**
- Does the visualization accurately and honestly present data? → **Substantive**
- Can we understand what message the maker of the visualization is attempting to convey? → **Perceptual**

We need to consider all of these qualities when evaluating and designing ‘good’ data visualizations

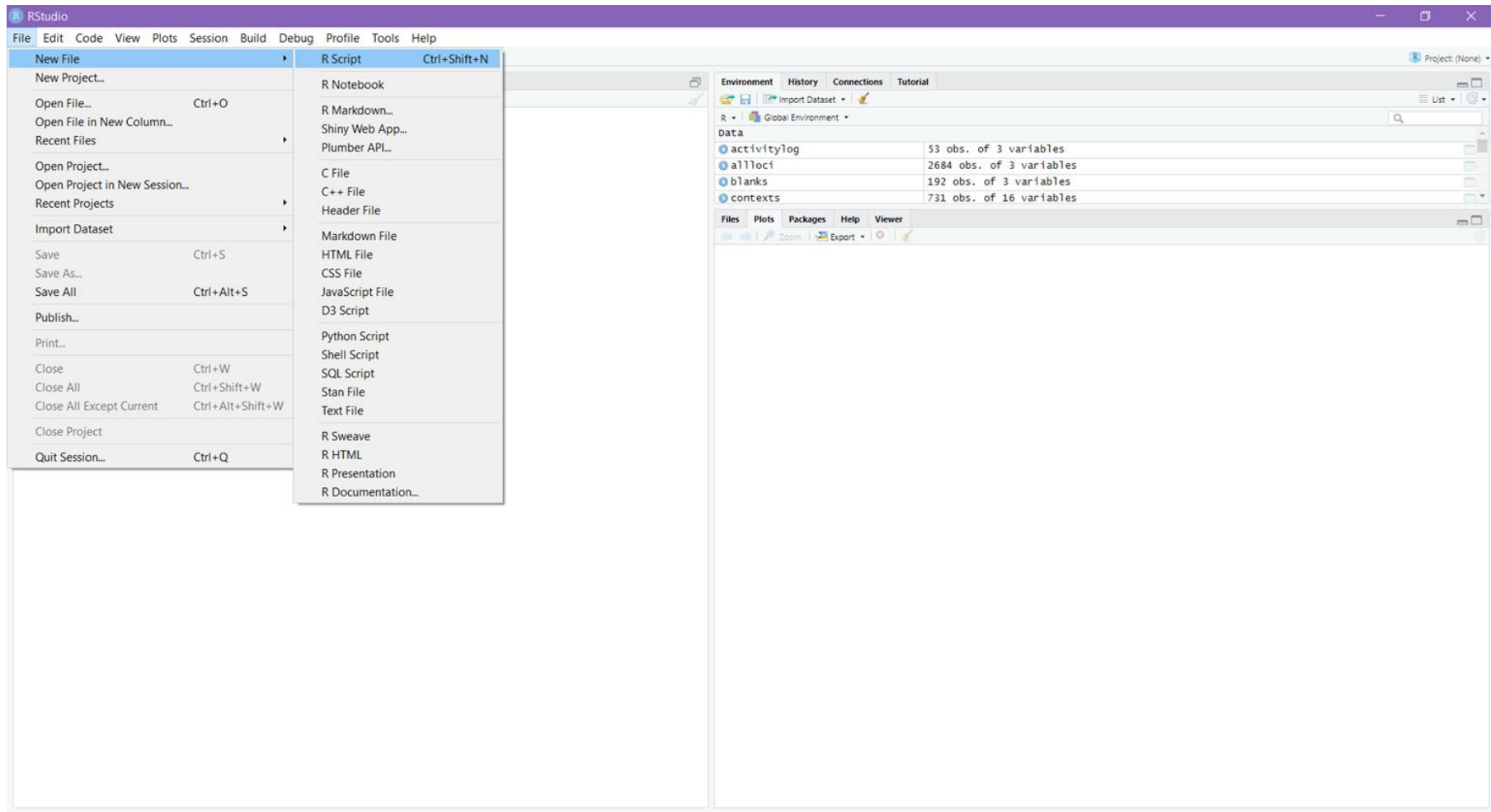
What data visualization IS

- Dependent on:
 - **Context** → **where and how** will our visualization be used? (eg. academic journal, poster, infographic)
 - **Audience** → **who** is intended to use our visualization? (eg. subject experts, general public)
 - **Data structure** → **what** information do our data capture? (eg. quantities, relationships)

What data visualization is NOT

- Hard and fast rules for every situation → visualizing data means making decisions
 - Type of graph (<https://www.data-to-viz.com/>)
 - Colour (Sequential, Qualitative, Diverging)
 - Alt text and captions (<https://datasf.gitbook.io/public-data-visualization-guide/accessibility/alt-text>)

Setting up R for data visualization



The image shows the RStudio desktop environment. A large blue arrow points from the text 'This is our R Script where we type our code' to the 'Untitled1.R' script editor. Another blue arrow points from the text 'Plots and figures will appear in the 'Plots' tab' to the 'Plots' tab in the 'Environment' pane.

This is our R Script where we type our code

Plots and figures will appear in the 'Plots' tab

Environment

Object	Class	Attributes
activitylog	data.frame	53 obs. of 3 variables
allicci	data.frame	2684 obs. of 3 variables
blanks	data.frame	192 obs. of 3 variables
contexts	data.frame	192 obs. of 16 variables

Files

Plots

Console

```
R version 4.0.3 (2021-03-31) -- "Snake and Throw"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale


R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

warning: namespace 'pool' is not available and has been replaced
by .GlobalEnv when processing object 'pool'
[workspace loaded from ~/.RData]

> |
```

We click 'Install' in the 'Packages' tab (next to 'Plots') to install the packages we need



The image shows the RStudio interface with the 'Packages' tab selected. A dialog box titled 'Install Packages' is open, showing the 'Repository (CRAN)' and the 'tidyverse' package selected. The 'Install' button is highlighted.

Environment History Connections Tutorial

R - Global Environment

Data

Variable	Obs.	Vars.
activitylog	53	3
allloci	2684	3
blanks	192	3
contexts	731	16

Files Plots Packages Help Viewer

☒ Install ☒ Update

Name	Description	Version
User Library		
abind	Combine Multidimensional Arrays	1.4-5
animation	A Gallery of Animations in Statistics and Utilities to Create Animations	2.6
analyses	Analyses of Phylogenetics and Evolution	5.5
ssh	Safe Password Entry for R, Git, and SSH	1.1
assertions	Easy Pre and Post Assertions	0.2.1
reim	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1
base64	Base64 Encoder and Decoder	2.0
tools	Tools for base64 encoding	0.1-3
boost	Boost C++ Header Files	1.75.0-0
classes	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.4
s3	A S3 Class for Vectors of 64bit Integers	4.0.5
simple	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.1
convert	Convert Statistical Objects into Tidy Tibbles	0.7.6
bootstrap	Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'	0.2.4
cache	Cache R Objects with Automatic Pruning	1.0.4
call	Call R from R	3.7.0
companion	Companion to Applied Regression	3.0-10
carData	Companion to Applied Regression Data Sets	3.0-4
cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
cli	Helpers for Developing Command Line Interfaces	2.5.0
clipr	Read and Write from the System Clipboard	0.7.1
coda	Output Analysis and Diagnostics for MCMC	0.19-4
colorspace	A Toolbox for Manipulating and Assessing Colors and Palettes	2.0-1
commonmark	High Performance CommonMark and Github Markdown Rendering in R	1.7
conquer	Convolution-Type Smoothed Quantile Regression	1.0.2
cpp11	A C++11 Interface for R's C Interface	0.2.7
crayon	Colored Terminal Output	1.4.1
crosstalk	Inter-Widget Interactivity for HTML Widgets	1.1.1
curl	A Modern and Flexible Web Client for R	4.3.1
data.table	Extension of 'data.frame'	1.14.0
DBI	R Database Interface	1.1.1
dbplyr	A 'dplyr' Back End for Databases	2.1.1

Install Packages

Install from: ☒ Configuring Repositories

Repository (CRAN)

Packages (separate multiple with space or comma):

tidyverse

tidyverse binary:

C:/Users/OWNER/Documents/R/win-library/4.0 [Default]

☒ Install dependencies

```
> library(tidyverse)
-- Attaching packages ----- tidyverse 1.3.1 --
v ggplot2 3.3.3      v purrr  0.3.4
v tibble  3.1.1      v dplyr  1.0.5
v tidyr   1.1.3      v stringr 1.4.0
v readr   1.4.0      v forcats 0.5.1

-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()

warning messages:
1: In function(kind = NULL, normal.kind = NULL, sample.kind = NULL) :
  non-uniform 'Rounding' sampler used
2: In function(kind = NULL, normal.kind = NULL, sample.kind = NULL) :
  non-uniform 'Rounding' sampler used
> library(socviz)
+
+
> library(socviz)
Error in library(socviz) : there is no package called 'socviz'
>
```

Load required packages

- Once our packages have been installed, type the following in your script:

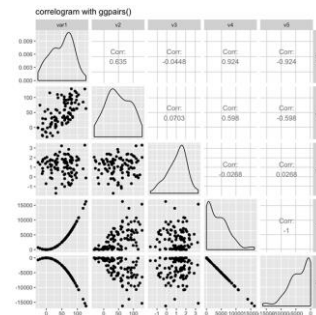
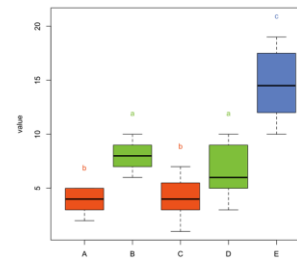
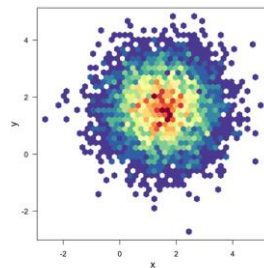
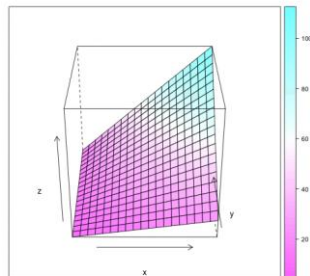
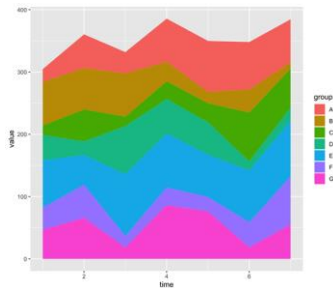
```
library(tidyverse)  
library(socviz)  
library(ggplot2)
```

- ctrl+enter (Windows) or cmd+enter (Mac) over highlighted code to run it and load the libraries

What is ggplot?

What is ggplot?

- An open source package for data visualization in R
- [Developed in 2005 by Hadley Wickham](#); now one of the most used R packages
- One package, [a LOT](#) of different types of data visualizations



How does ggplot work?

- Data visualization is when we represent our data using lines, shapes, colours, etc
- A **mapping** is the relationship between the variables in our data and the representation of those variables in our visualization
- Our mappings in ggplot are called aesthetic mappings, or **aesthetics**

How does ggplot work?

- Once we make our mappings, ggplot lets us choose what type of plot we want. Each type of plot in ggplot is called a **geom** (eg. `geom_bar()` for bar plots, `geom_point()` for scatterplots, etc)
- We make our plot by adding a `ggplot()` object + a `geom()` object, then adding labels, legends, etc

Preparing our data for visualizing

- For graphing with ggplot (AKA what we want to do), our data should be in **long** format as opposed to **wide** format
- That is: **in long formatted data, every observation should be a row, and every variable should be a column**

Make a figure with ggplot

Load our sample dataset

- We will use a preloaded sample dataset called 'gapminder', which can be loaded as follows:

```
install.packages("gapminder")  
  
library(gapminder)  
  
gapminder
```

- We can see that our dataset contains data about countries over several years

Making a figure with a sample dataset

- First, we need to make a `ggplot()` object that tells ggplot which dataset we are using:

```
p <- ggplot(data = gapminder)
```

- Next, we need to establish our mapping (which variable corresponds to which visual element). To do this, we use the `aes()` function:

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


Making a figure with a sample dataset - aes() function

- The `aes()` function, in this case, maps variables to the x and y axes.
- It can also map variables to other things you will see on the plot, such as colour, shape, size, line type (dashed vs solid), etc
- Within the `aes()` function, we do not have to say where our variables are found, because ggplot will assume they are in the dataset we assigned as our data object (in this case, `gapminder`)

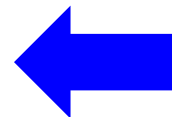
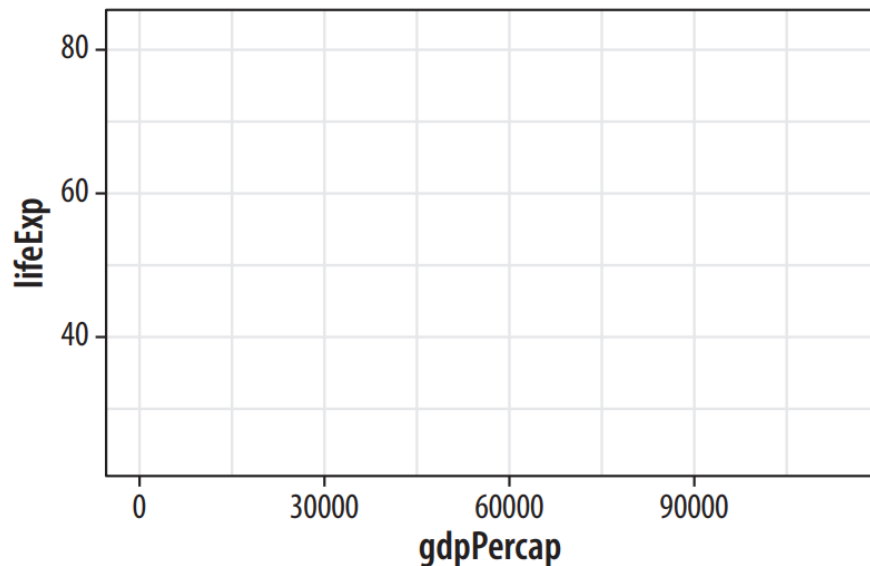
```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp))
```

Making a figure with a sample dataset

- At this point, if we just type 'p' (our ggplot object) and run the code, what output do we get?

Making a figure with a sample dataset

- At this point, if we just type 'p' (our ggplot object) and run the code, what output do we get?



We get an output that has a mapping, but does not know what type of plot to make, since we have not yet chosen a `geom()` function

Making a figure with a sample dataset

- To produce an actual plot, we need to pick a `geom()` function that tells `ggplot` what type of plot to make.
- To make a scatterplot:

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

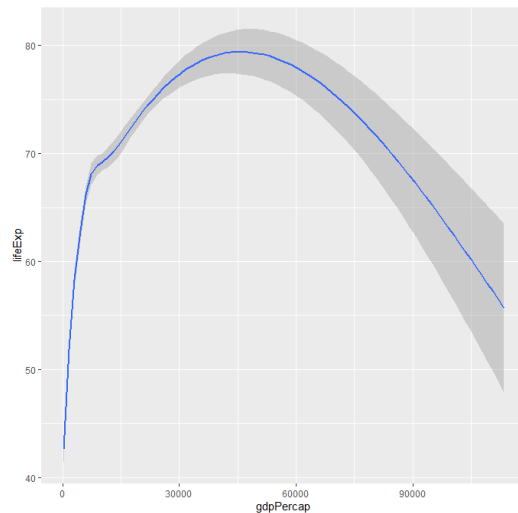
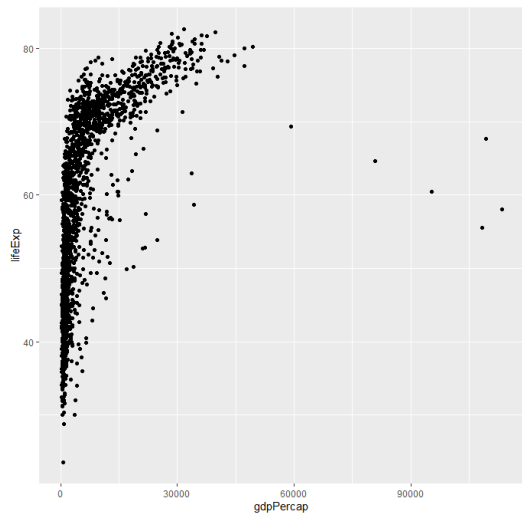
Changing our plot

- We can choose a different geom to make a different kind of plot
- For example, by using `geom_smooth()` instead of `geom_point()`:

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_smooth()
```

Changing our plot

- We go from a scatterplot to a line plot with shaded standard error



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

```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y =  
  lifeExp))  
  
p + geom_smooth()
```

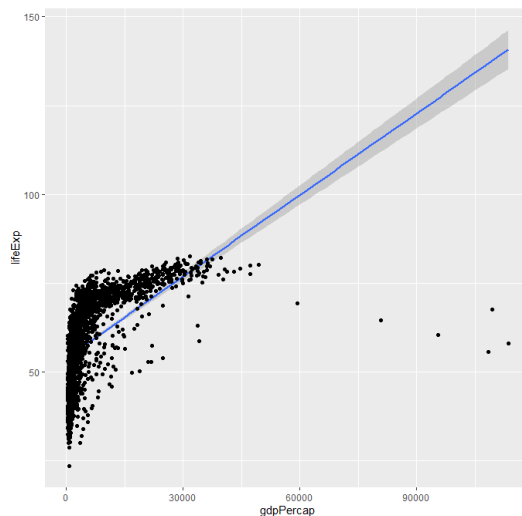
Changing our plot

- By default, the `geom_smooth()` function is using a generalized additive model (gam).
- We can add an argument to the function to change the method it is using to fit a line to a linear model (lm):

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_smooth(method = "lm")
```

Changing our plot

- We can also combine our geoms to see both types at once:



```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_point() + geom_smooth(method = "lm")
```

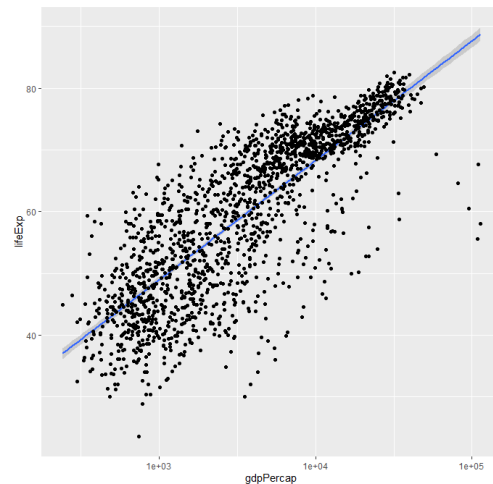
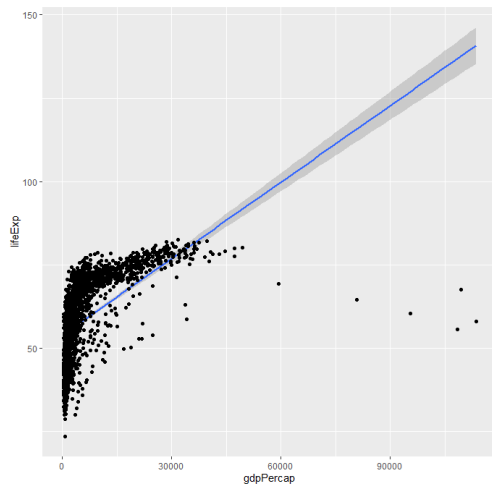

Scale

- We can see that our GDP data is very skewed to the left of our plot, because it is not normally distributed.
- We can address this by transforming our x axis from a linear scale to a log scale using the `scale_x_log10()` function:

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_smooth(method = "lm") + geom_point() +  
  scale_x_log10()
```

Scale

- We go from an ill-fitting linear plot to a logarithmic plot:



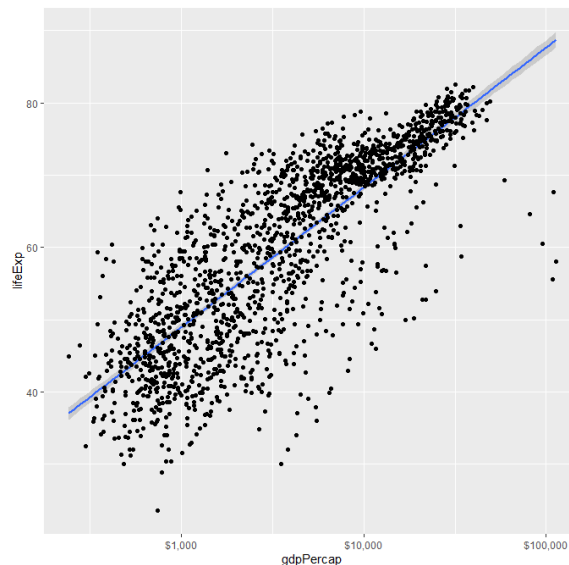
```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_point() + geom_smooth(method = "lm")
```

```
p <- ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_point() + geom_smooth(method = "lm") + scale_x_log10()
```

Labels

- If we want to view our GDP (on the x axis) in dollar amounts rather than in scientific notation, we can add an argument from the scales package to our `scale_x_log10()` function:

```
p <- ggplot(data = gapminder,  
            mapping = aes(x =  
                          gdpPercap, y =  
                          lifeExp))  
  
p + geom_smooth(method="lm") +  
  geom_point() +  
  scale_x_log10(labels =  
               scales::dollar)
```



The ggplot process

- We will follow the same basic process (with some more details) to make just about every ggplot.
- **To visualize data with ggplot, we:**
 1. Tell `ggplot()` what dataset we want to use
 2. Tell `ggplot()` what mapping we want to see
 3. Decide what type of plot we want to see using a `geom()`
 4. Add geoms to our ggplot object as needed to customize our visualization
 5. Customize labels, titles, scales, etc

Mapping aesthetics versus setting aesthetics

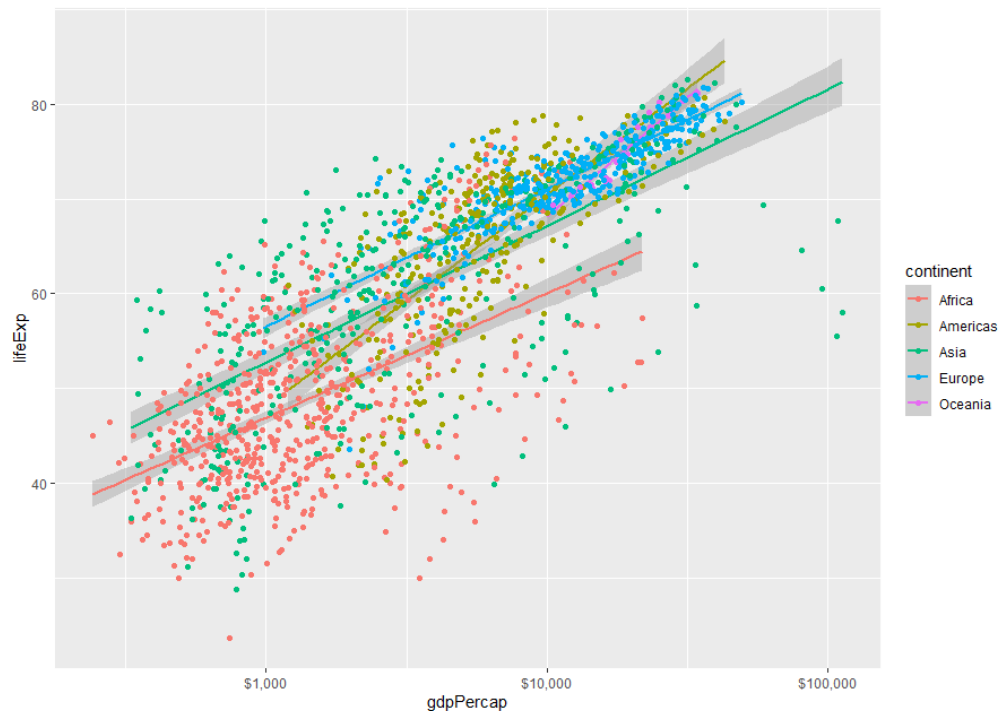
Mapping a variable

- **Remember:** aesthetic mapping lets us tell ggplot to express a variable with a given visual element (size, colour, shape, etc).
- For example, if we want our 'continent' variable in the gapminder dataset to be represented by colour, we type:

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp,  
                          color = continent))
```

Colour!

- Our resultant plot looks like:



Mapping a variable

IMPORTANT:

Mapping a variable to color is NOT the same thing as assigning a specific colour to a variable.

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp, color =  
                          continent))
```

(So, for example, the above code will not let us make all our points purple.)

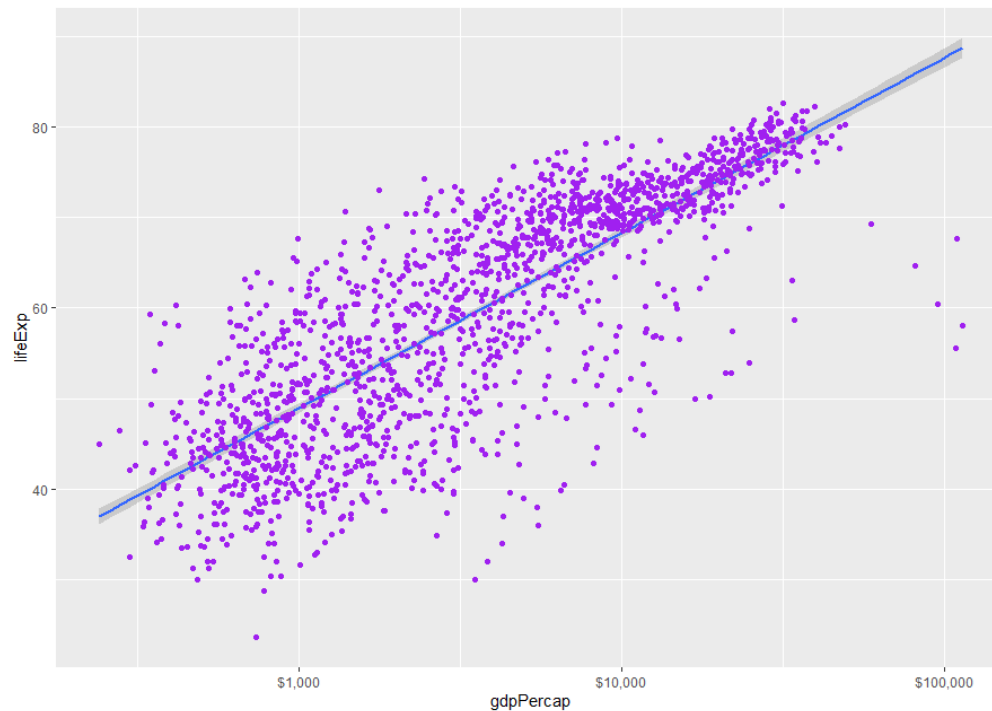
Setting a variable

- If we want to **set** (not map) a visual element to a particular value (eg. purple), we do that in our `geom()` object, not in the `ggplot()`
- For example, if we want to make our combined graph and colour data points purple, we do:

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp))  
p + geom_point(color = "purple") + geom_smooth(method =  
"lm") + scale_x_log10()
```

Colour!

- Our resultant plot looks like:



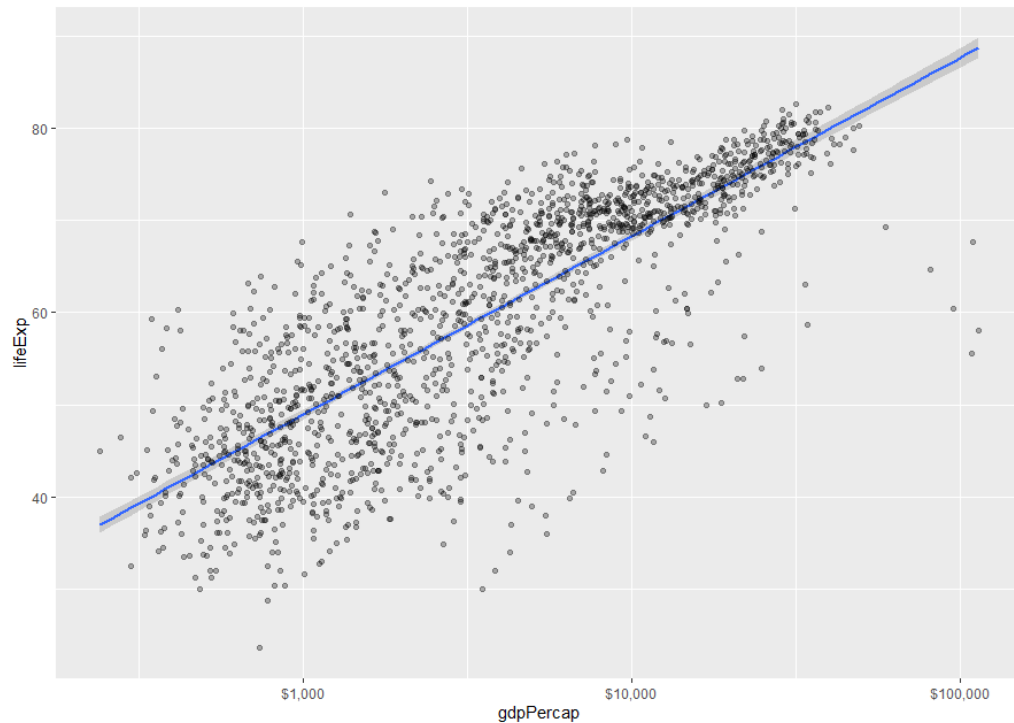
Opacity

- The same principle applies to altering the opacity of our visual elements.
- We do this using the **alpha** argument in our geom, where an alpha of zero is completely transparent, while an alpha of one is completely opaque.

```
p <- ggplot(data = gapminder,  
            mapping = aes(x = gdpPercap, y = lifeExp))  
  
p + geom_point(alpha = 0.3) + geom_smooth(method =  
"lm") + scale_x_log10()
```

Opacity

- Our resultant plot looks like:



Grouping data with ggplot

- This time, we want to plot the trajectory of GDP over time for each country in the dataset
- **What happens if we try to produce the desired graph using what we have previously learned?**
 - Variables of interest: 'year' and 'gdpPercap'
 - Use `geom_line()`

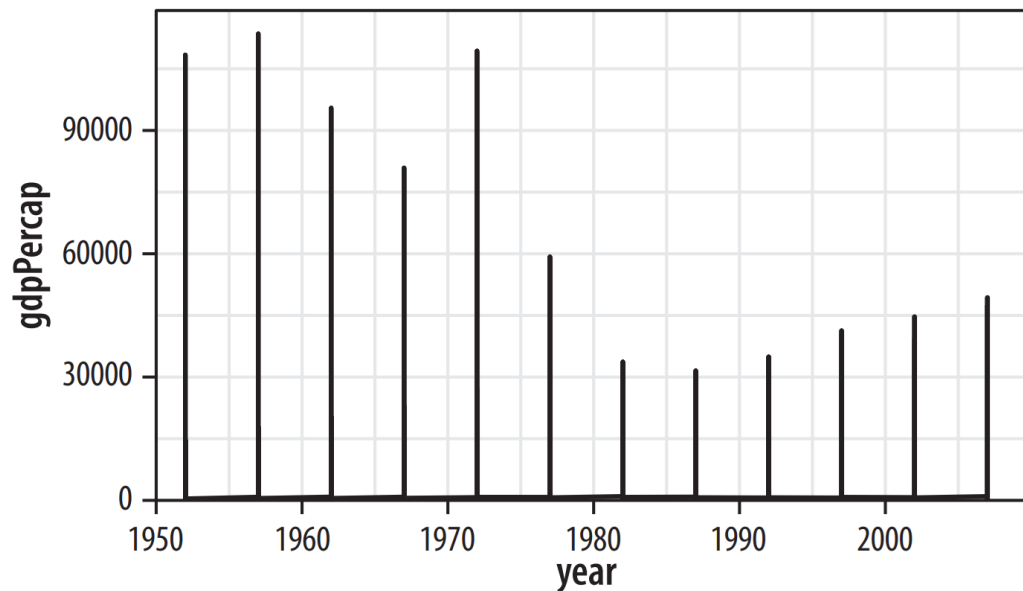
Graphing GDP over time per country

- If we write our code as:

```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))  
p + geom_line()
```

- Our result will look like...

Graphing GDP over time per country - Incorrectly!



```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))  
p + geom_line()
```

- This is **not** the GDP by country plot we wanted to produce
- We did not tell ggplot that yearly observations are grouped by country, so it is showing all countries' GDP for 1952, then all for 1957, and so on

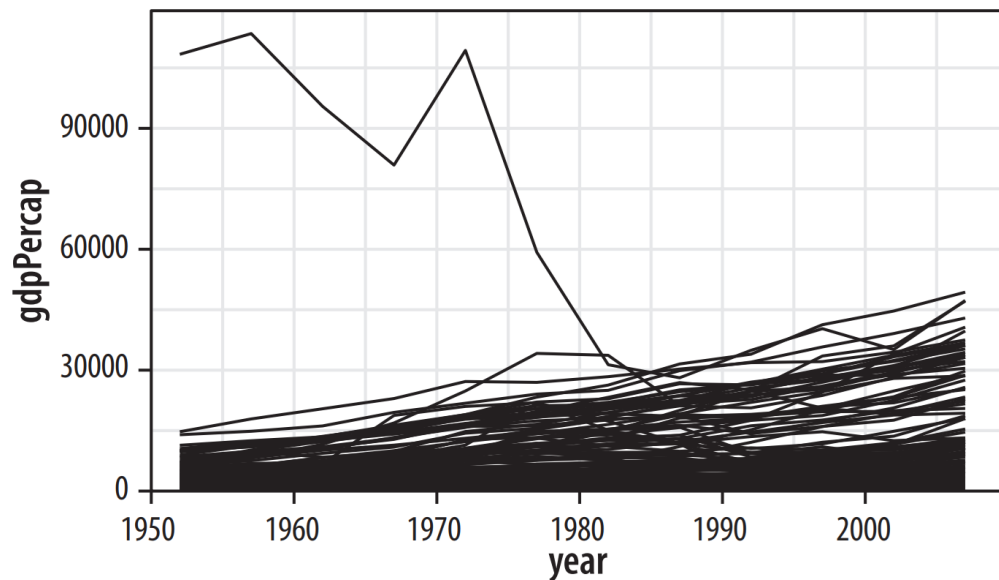
Graphing GDP over time per country - Group aesthetic

- We can fix this issue by telling ggplot about the country-level grouping in the dataset using a **group aesthetic**:

```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))  
p + geom_line(aes(group = country))
```

- This time, our result will look like...

Graphing GDP over time per country - Group aesthetic



```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))  
p + geom_line(aes(group = country))
```

- Our graph is still hard to read, but now shows the data as we wanted (each line represents GDP of a country over time)
- **How can we modify our plot to make the trend in our data and our message more clear?**

Faceting data with ggplot

Graphing GDP over time per country - Faceting our data

- **Faceting** is when we break our data up into pieces to make a **small multiple** plot
- When we facet, we split the data by some third variable, and our plot will have a separate panel for each value of the faceting variable
- **Note:** Facets are not a `geom` object, but are a way of organizing a series of geoms

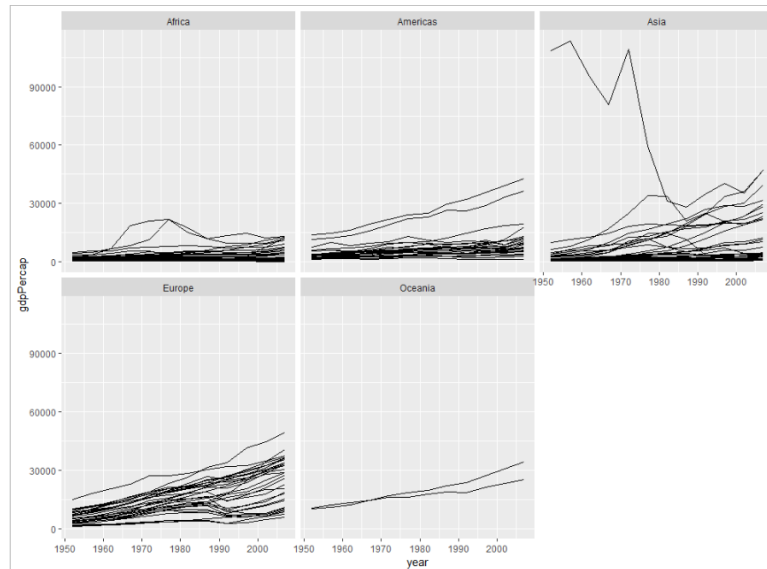
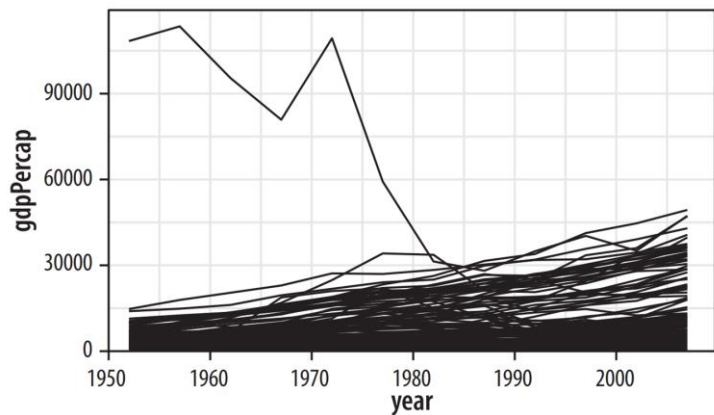
Graphing GDP over time per country - Faceting our data

- We can facet our Gapminder data using the `facet_wrap()` function
- In our case, we want to break up our plot by the variable 'continent', so we facet as follows:

```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))  
p + geom_line(aes(group = country)) + facet_wrap(~continent)
```

Faceted plots

- Faceting by continent changes our plot output:



```
p <- ggplot(data=gapminder, mapping = aes(x=year,
y=gdpPercap))

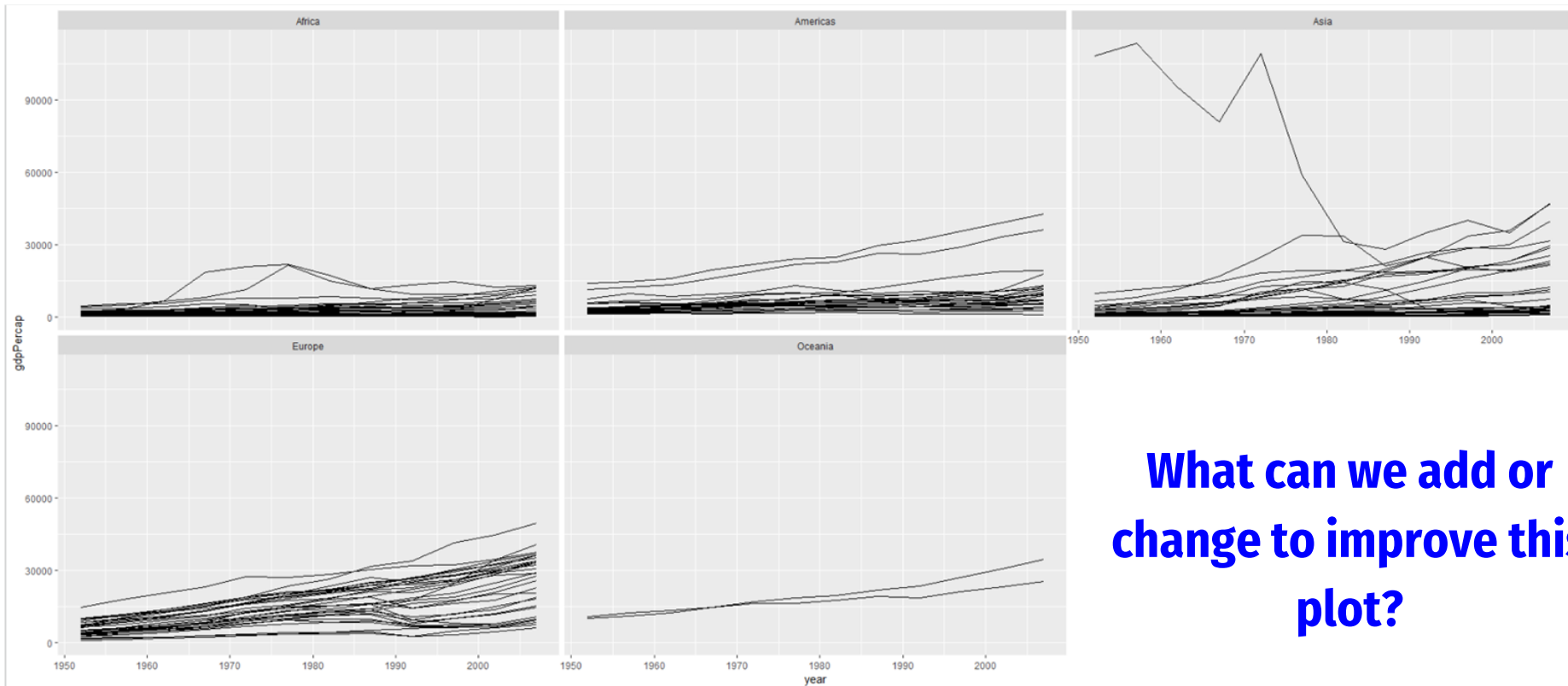
p + geom_line(aes(group = country))
```

```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))

p + geom_line(aes(group = country)) + facet_wrap(~continent)
```

Activity: Improving our grouped and faceted plot

Activity



**What can we add or
change to improve this
plot?**

Activity - Improving our plot

- Healy (2018) offers several suggestions for ways we can improve the aesthetic, substantive, and perceptual characteristics of our plot:
 - Make country trends light grey colour
 - Add a trend line
 - Make y axis logarithmic and show that values are in dollars
 - Try to fit all five facets on a single row (5 columns)
 - Add axis labels and graph title

Activity - Improving our plot

- **Try to identify which parts of our updated code correspond to each of the suggested changes:**
- Make country trends light grey
- Add a trend line
- Make y axis logarithmic and show that values are in dollars
- Try to fit all five facets on a single row (5 columns)
- Add axis labels and graph title

```
p<-ggplot(data=gapminder,mapping=aes(x=year, y=gdpPercap))  
p + geom_line(color="gray70", aes(group = country)) +  
      geom_smooth(linewidth=1.1,method="loess",se=FALSE) +  
      scale_y_log10(labels=scales::dollar) +  
      facet_wrap(~continent,ncol=5) +  
      labs(x = "Year",  
           y = "GDP per capita",  
           Title = "GDP per capita on Five Continents")
```

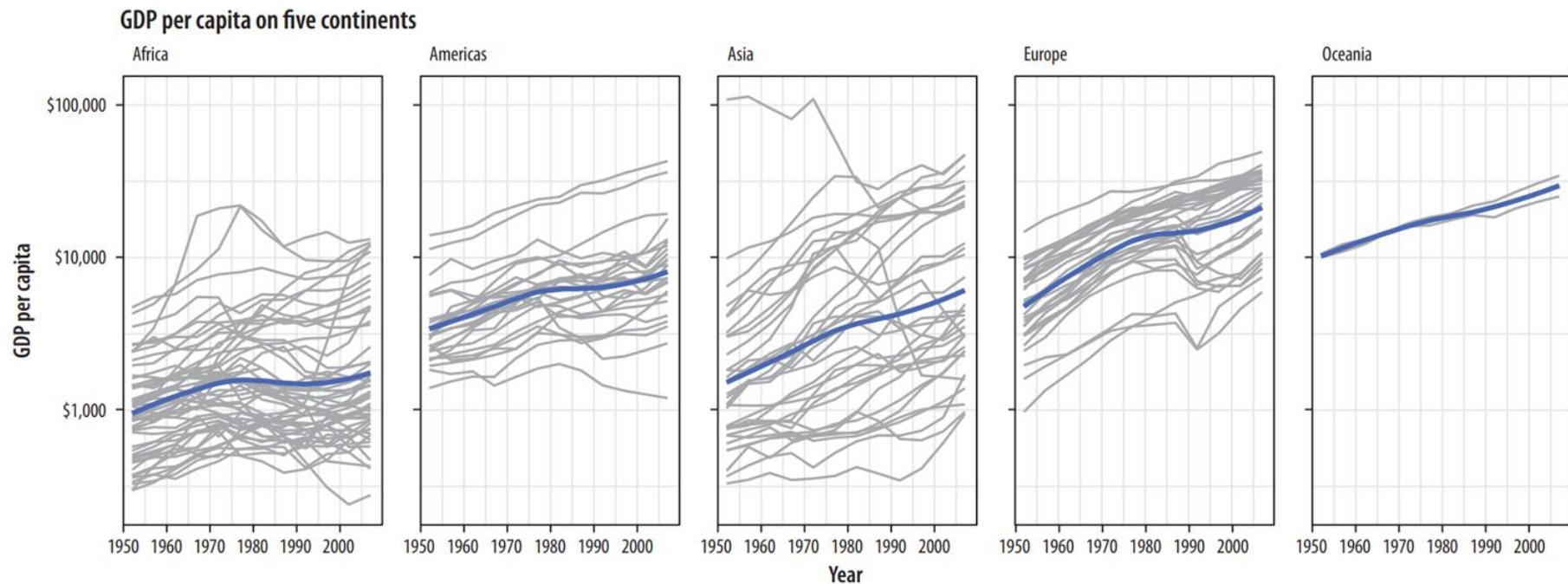
Activity - Improving our plot

- Try to identify which parts of our updated code correspond to each of the suggested changes:

- Make country trends light grey
- Add a trend line
- Make y axis logarithmic and show that values are in dollars
- Try to fit all five facets on a single row (5 columns)
- Add axis labels and graph title

```
p<-ggplot(data=gapminder,mapping=aes(x=year, y=gdpPercap))  
p + geom_line(color="gray70", aes(group = country)) +  
      geom_smooth(linewidth=1.1,method="loess",se=FALSE) +  
      scale_y_log10(labels=scales::dollar) +  
      facet_wrap(~continent,ncol=5) +  
      labs(x = "Year",  
           y = "GDP per capita",  
           Title = "GDP per capita on Five Continents")
```

Activity - Improved group/facet plot



Plotting text

Labeling outliers

- Adding labels to the points in our scatterplots can help to make our plots more informative
- We can add text to our plots using `geom_text()` and label only specific points using `subset()`

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

p + geom_point(alpha = 0.3) +
  scale_x_log10(labels = scales::dollar) +
  geom_text(data = subset(gapminder, gdpPercap > 98000),
    aes(label = country))
```

Annotating plots

- It can sometimes be useful to annotate a figure or place arbitrary text on a plot
 - That is, text that is not mapped to a variable
- We do this using the `annotate()` function, which is not a geom, but uses the features of geoms (such as the ability to modify size, colour, and x or y position)

Annotating plots with text

```
p + geom_point(alpha = 0.3) +  
  scale_x_log10(labels = scales::dollar) +  
  annotate(geom = "text", x = 21000, y = 45, label = "A  
surprisingly high \n per cap value.", hjust = 0)
```

- The above code
 - Uses `annotate()` to add text (`geom = "text"`)
 - Positions the text at (21000, 45) on our plot
 - Using the **newline code** (`'\n'`) to force a line break
 - Left-justifies our text

Annotating plots with shapes

- We can also use `annotate` to add shapes to our plot
- For example, we can add a red rectangle to our previous code by calling `annotate()` again with “rect” instead of “text”

```
p <- ggplot(data=gapminder, aes(x = gdpPercap, y = lifeExp))  
  
p + geom_point(alpha = 0.3) +  
  scale_x_log10(labels = scales::dollar) +  
  annotate(geom = "rect", xmin = 20000, xmax = 140000, ymin = 40, ymax  
= 50, fill = "red", alpha = 0.2) +  
  annotate(geom = "text", x = 21000, y = 45, label = "A surprisingly  
high \n per cap value.", hjust = 0)
```

Some ggplot resources

- Access the “[Data Visualization with ggplot2 Cheatsheet](#)” for a review of the basics of constructing a ggplot
- Consult the “[R Graphics Cookbook](#)” for ‘recipes’ for specific visualization types

Thank you!

