Data Visualization

First Steps: Make a Plot

Ciara Zogheib

Data Sciences Institute, University of Toronto

Prerequisites

- You have installed and loaded the tidyverse, socviz, and ggplot2 packages in RStudio

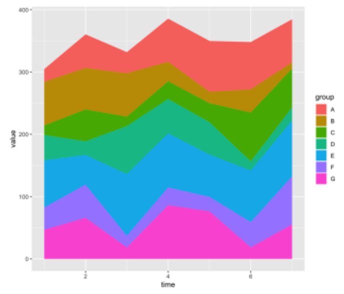
Today we will...

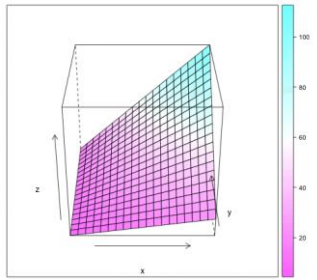
- Learn about the ggplot library and aesthetic mapping versus setting
- Work through practice code from Chapter 3 (*Make a Plot*) of Healy, K. (2018). Data Visualization: A Practical Introduction. Princeton University Press. This code will let us
    - Make a plot with ggplot
    - Adjust the type of plot, scale, and labels of our plot
    - Add colour and opacity to our plots
    - Save and export our plots as image files
- Apply our visualization evaluation skills to ggplot images

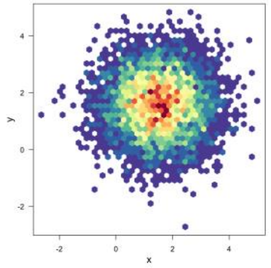
ggplot

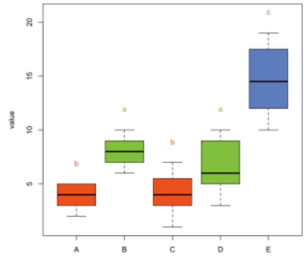
What is ggplot?

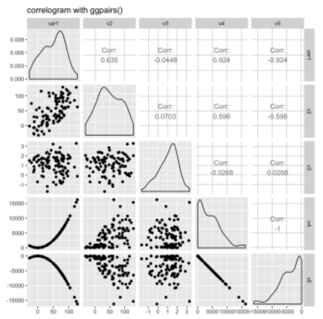
- An open source package for data visualization in R
- Developed in 2005 by Hadley Wickham; now one of the most used R packages
- One package, a LOT of different types of data visualizations











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

(Healy, 2018)

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

(Healy, 2018)

Making a figure with ggplot

Making a figure with a sample dataset

- We will once again use the sample dataset of country information that we used last class

```
library(gapminder)

gapminder
```

- Before, we plotted life expectancy against per capita GDP for every country and year in the dataset
- Now we will break the plot down, step by step

(Healy, 2018)

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))
```

(Healy, 2018)

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))
```

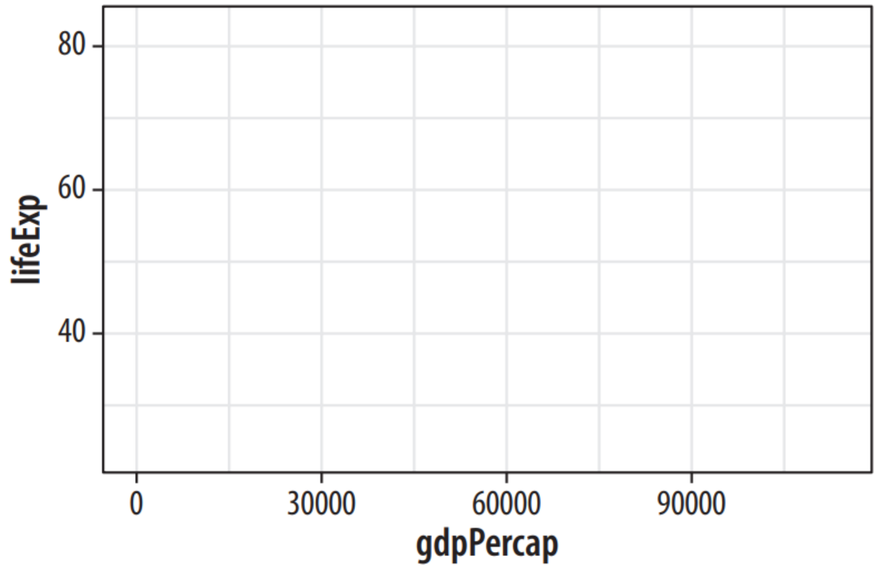
(Healy, 2018)

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

(Healy, 2018)

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()
```

(Healy, 2018)

Changing our plot

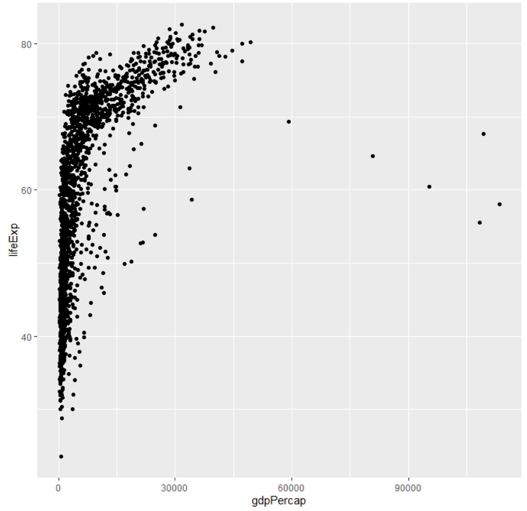
- We can choose a different geom to make a different kind of plot
- For example, by using `geom_smooth()` instead of `geom_point()`:

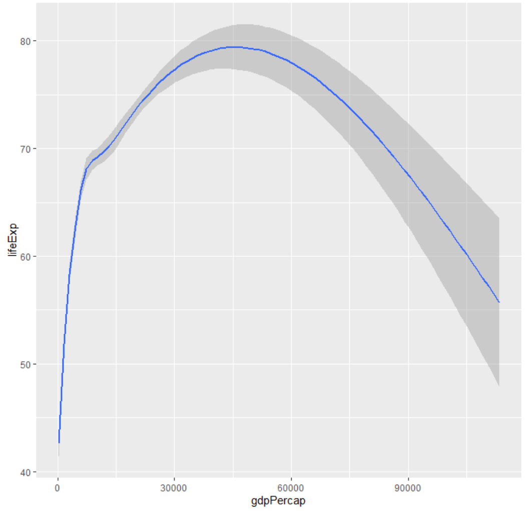
```
p <- ggplot(data = gapminder,

         mapping = aes(x = gdpPercap, y = lifeExp))

p + geom_smooth()
```

(Healy, 2018)

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

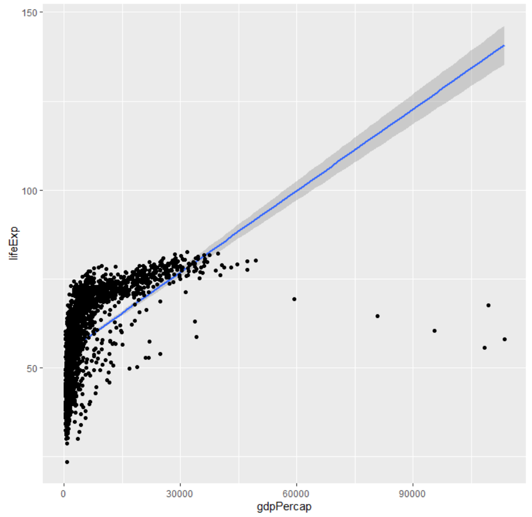
- By default, the `geom_smooth()` function is using a generalized additive model (gam).
- We can add an argument to the function to change the method it is using to fit a line to a linear model (lm):

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

          mapping = aes(x = gdpPercap, y = lifeExp))

p + geom_smooth(method = “lm”)
```

(Healy, 2018)

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”)
```

(Healy, 2018)

Scale

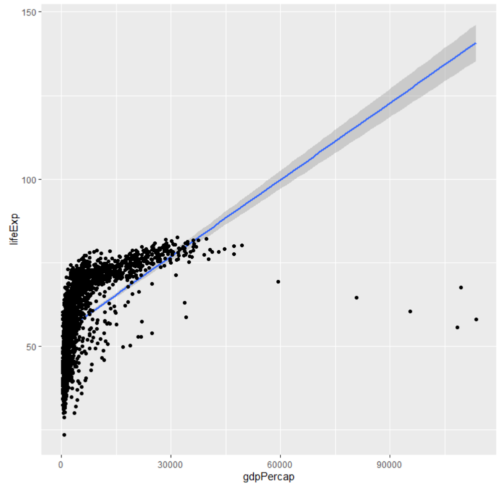
- We can see that our GDP data is very skewed to the left of our plot, because it is not normally distributed.
- We can address this by transforming our x axis from a linear scale to a log scale using the `scale_x_log10()` function:

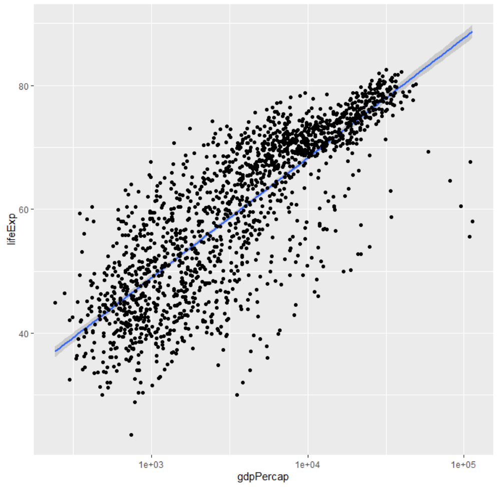
```
p <- ggplot(data = gapminder,

          mapping = aes(x = gdpPercap, y = lifeExp))

p + geom_smooth(method = “lm”)+ geom_point() +
scale_x_log10()
```

(Healy, 2018)

Scale

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





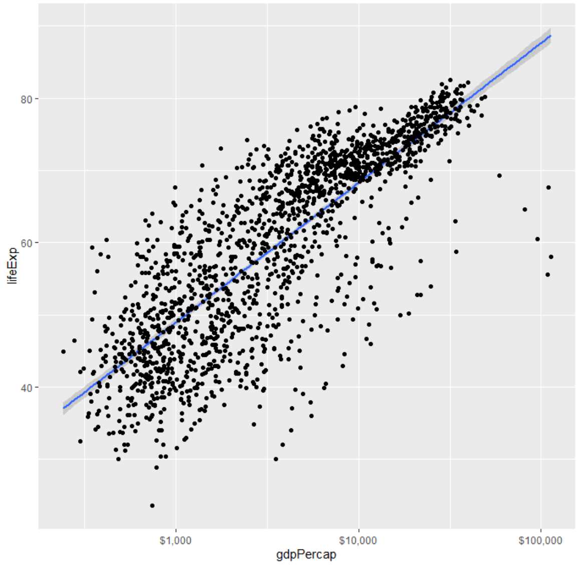
```
p <- ggplot(data = gapminder,mapping = aes(x =
gdpPercap, y = lifeExp))

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)
```



(Healy, 2018)

The ggplot process

- We will follow the same basic process (with some more details) to make just about every ggplot we saw in the gallery
- **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

(Healy, 2018)

Mapping aesthetics versus setting aesthetics

Mapping a variable

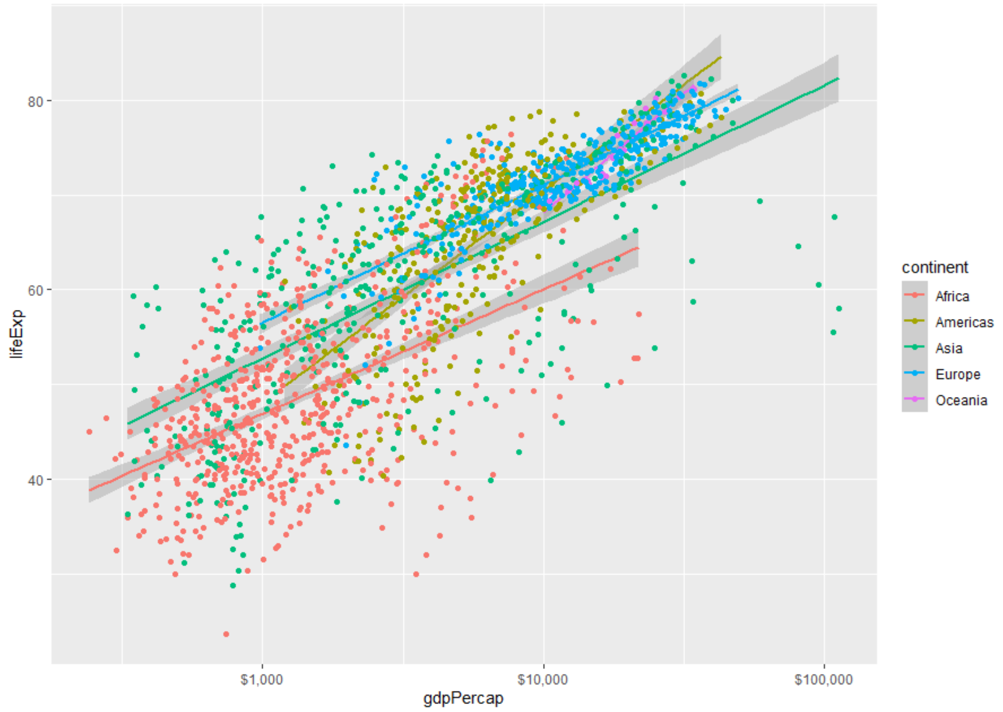
- **Remember**: aesthetic mapping lets us tell ggplot to express a variable with a given visual element (size, colour, shape, etc).
- For example, if we want our 'continent' variable in the gapminder dataset to be represented by colour, we type:

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

          mapping = aes(x = gdpPercap, y = lifeExp,
          color = continent))
```

(Healy, 2018)

Colour!

- Our resultant plot looks like:



(Healy, 2018)

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

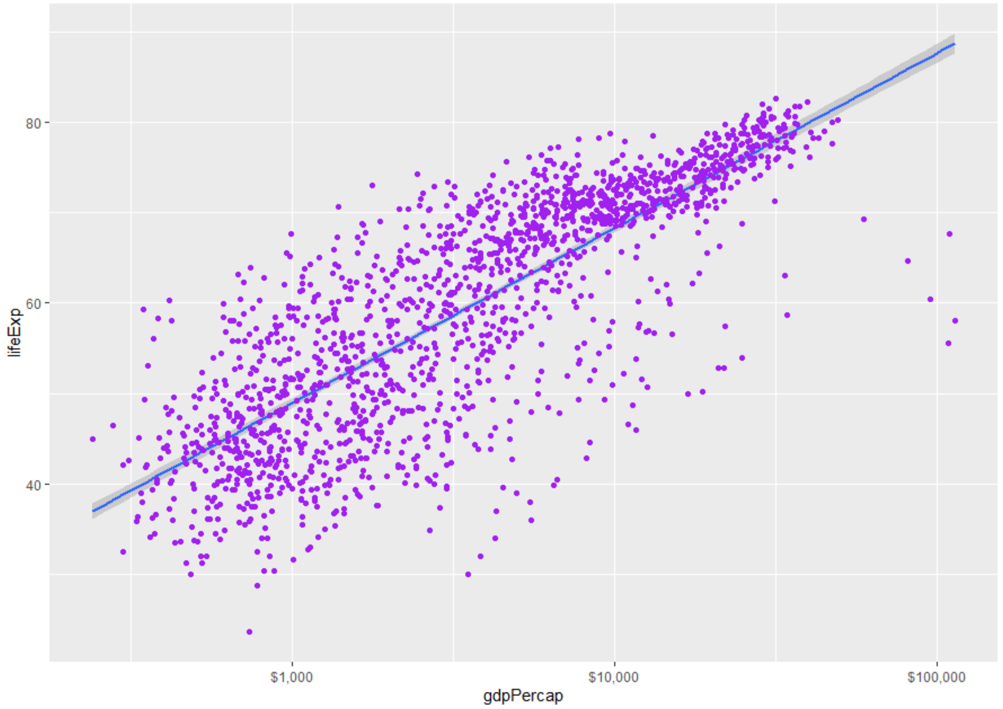
- If we want to set (not map) a visual element to a particular value (eg. purple), we do that in our `geom()` object, not in the `ggplot()`
- For example, if we want to make our combined graph and colour data points purple, we do:

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

          mapping = aes(x = gdpPercap, y = lifeExp))

p + geom_point(color = “purple”) + geom_smooth(method
= "lm”) + scale_x_log10()
```

(Healy, 2018)

Colour!

- Our resultant plot looks like:



(Healy, 2018)

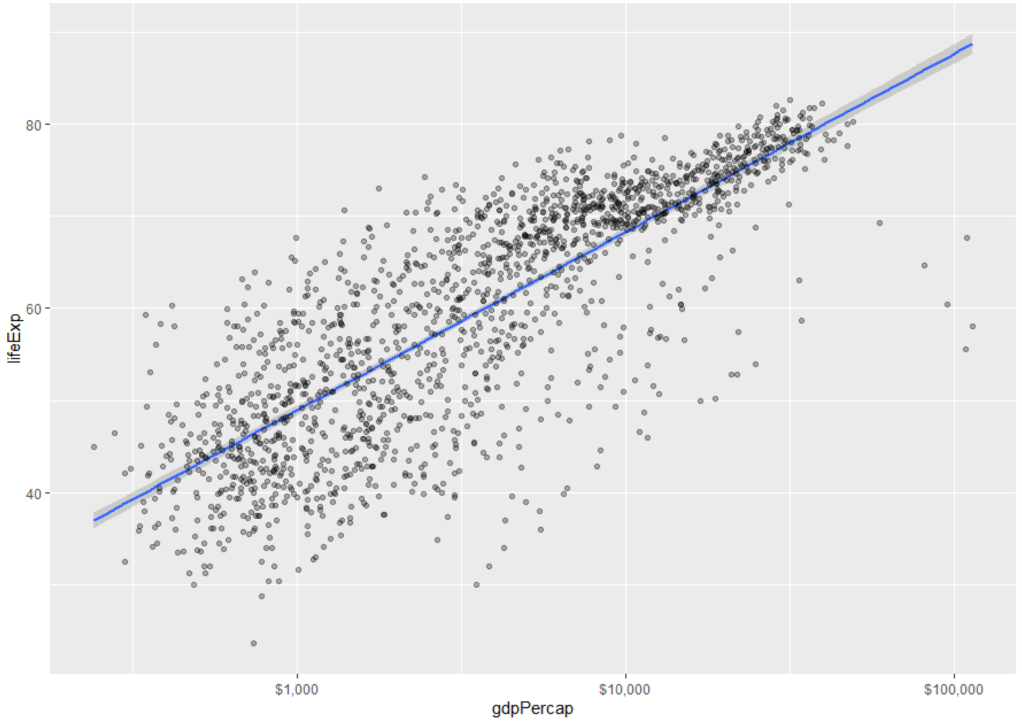
Opacity

- 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:



(Healy, 2018)

Saving our ggplots

Saving our plots

- We can easily save our most recent plot using the ggsave() function, specifying the name and file type we want to save as:

```
ggsave(filename = “sampleimage.png”)

ggsave(filename = “sampleimage.pdf”)
```

- We can also set the size of our saved plot in the units of our choice:

```
ggsave(filename = “sampleimage.png”,height = 8,
width = 10, units = “in”)
```

(Healy, 2018)

SAVE YOUR SCRIPT HERE

(We’ll use it again in the next classes!)

Activity: Exploring ggplot

Activity

- Visit the R Graphs Gallery at https://www.r-graph-gallery.com/all-graphs.html
- Select one of the visualization types that you find most interesting
- For your visualization of choice:
    1. Copy the provided code and attempt to replicate the output in RStudio. **NOTE:** You may have to install packages.
    2. Recall the aesthetic, substantive, and perceptual qualities of data visualizations. Does your visualization of choice succeed in each area?

For more information about ggplot...

- We will go into (a lot!) more detail in later classes, but for extra resources or reminders as we work through the rest of this course, you can:
    - 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

Next...

- What is reproducible data visualization?
- How can we incorporate ideas about reproducibility into our data visualization practices? (Ethics)