Data Visualization

Graphing our Data: Show the Right Numbers

Ciara Zogheib

Data Sciences Institute, University of Toronto

Prerequisites

- You have installed and loaded the tidyverse, socviz, and ggplot2 packages in RStudio
- You have the code from last class (or a copy) open in RStudio

Today, we will...

- Expand the kinds of ggplots we can make
- Work through practice code from Chapter 4 (*Show the Right Numbers*) of Healy, K. (2018). Data Visualization: A Practical Introduction. Princeton University Press. This code will let us
    - Group data
    - Facet data
    - Transform data
    - Make frequency plots
    - Make histograms and density plots

Grouping data with ggplot

Recall our ‘Gapminder’ dataset

- We will once again use the Gapminder sample dataset

```
library(gapminder)

gapminder
```

- This time, we want to plot the trajectory of GDP over time for each country in the dataset
- **Activity: Try to produce the desired graph using what we have previously learned in this course**

(Healy, 2018)

Graphing GDP over time per country

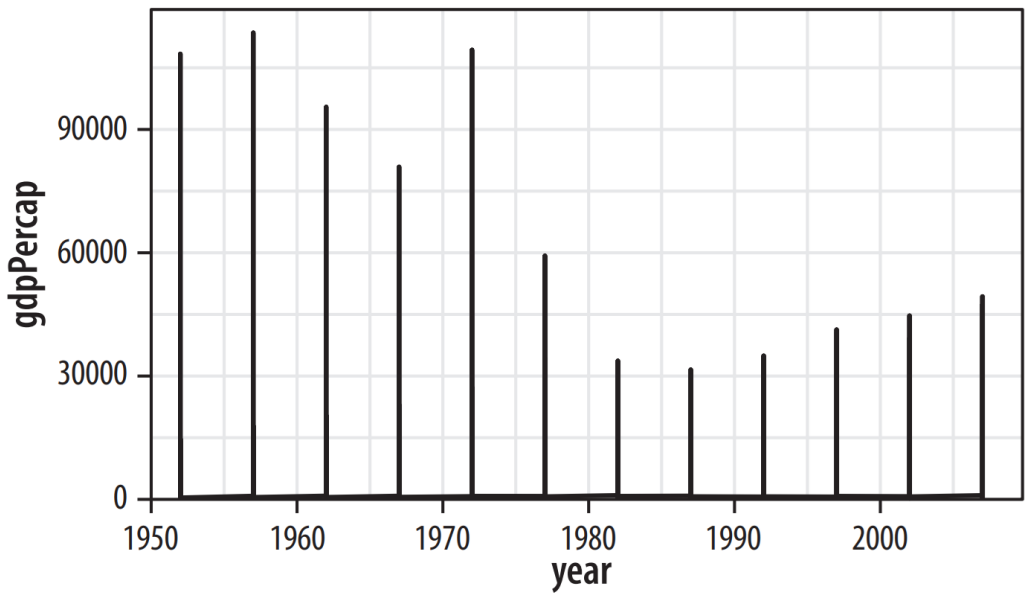
- If we write our code as:

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

p + geom_line()
```

- Our result will look like...

(Healy, 2018)

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 1952, and so on

(Healy, 2018)

Graphing GDP over time per country - Group aesthetic

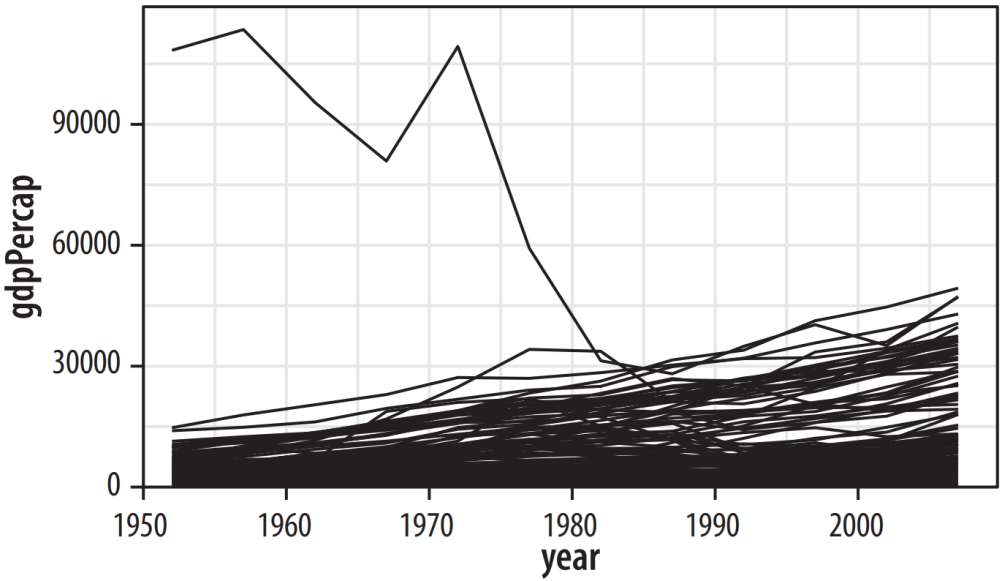
- We can fix this issue by telling ggplot about the country-level grouping in the dataset using a group aesthetic:

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

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

(Healy, 2018)

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?**

(Healy, 2018)

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

(Healy, 2018)

Graphing GDP over time per country - Faceting our data

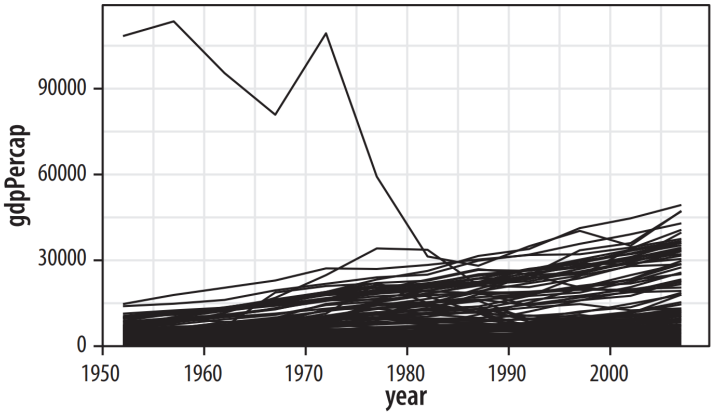
- We can facet our Gapminder data using the `facet_wrap()` function
- In our case, we want to break up our plot by the variable 'continent', so we facet as follows:

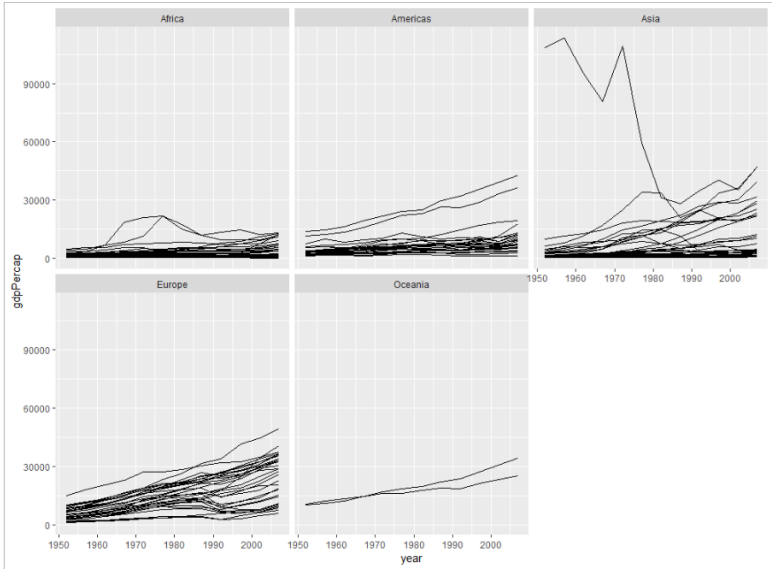
```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))
p + geom_line(aes(group = country)) + facet_wrap(~continent)
```

(Healy, 2018)

Faceted plots

- Faceting by continent changes our plot output:



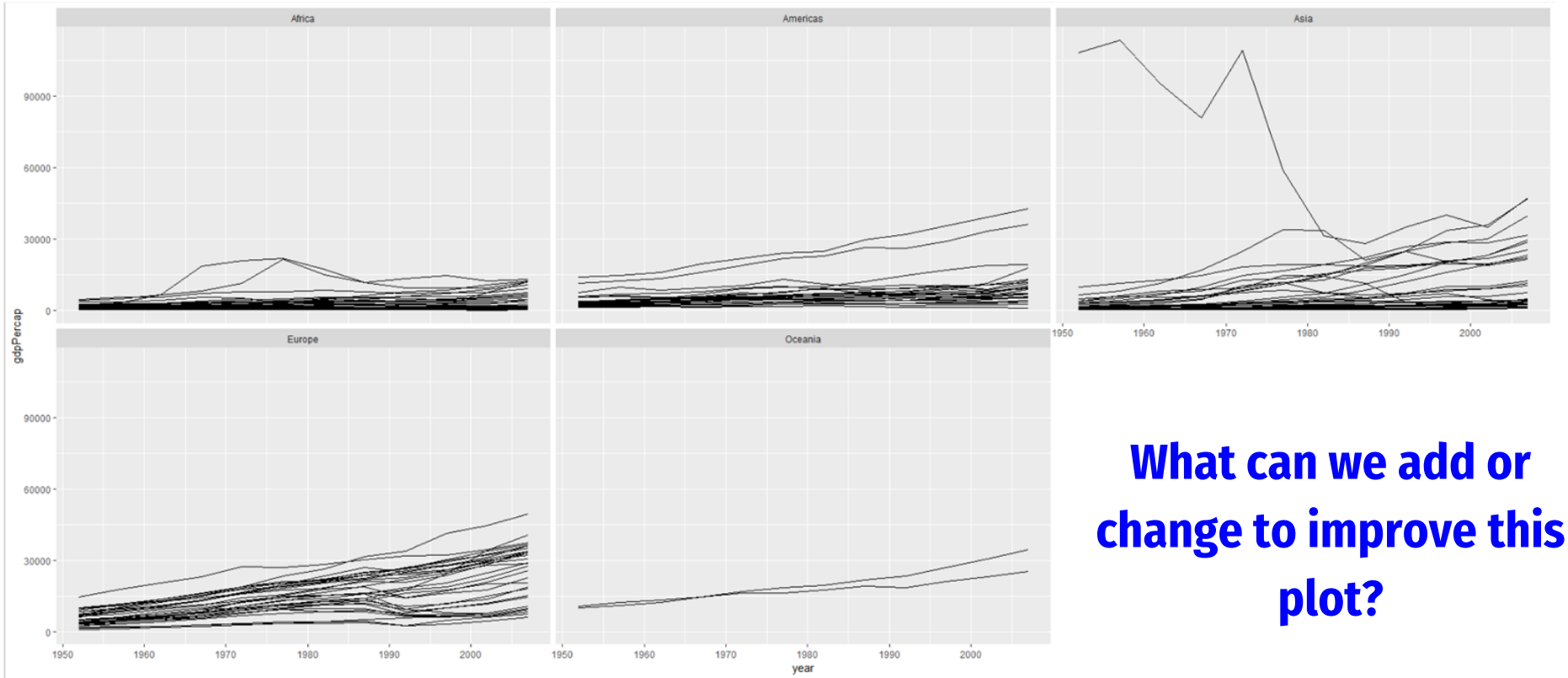



```
p <- ggplot(data=gapminder, mapping = aes(x=year, y=gdpPercap))

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(size=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")
```

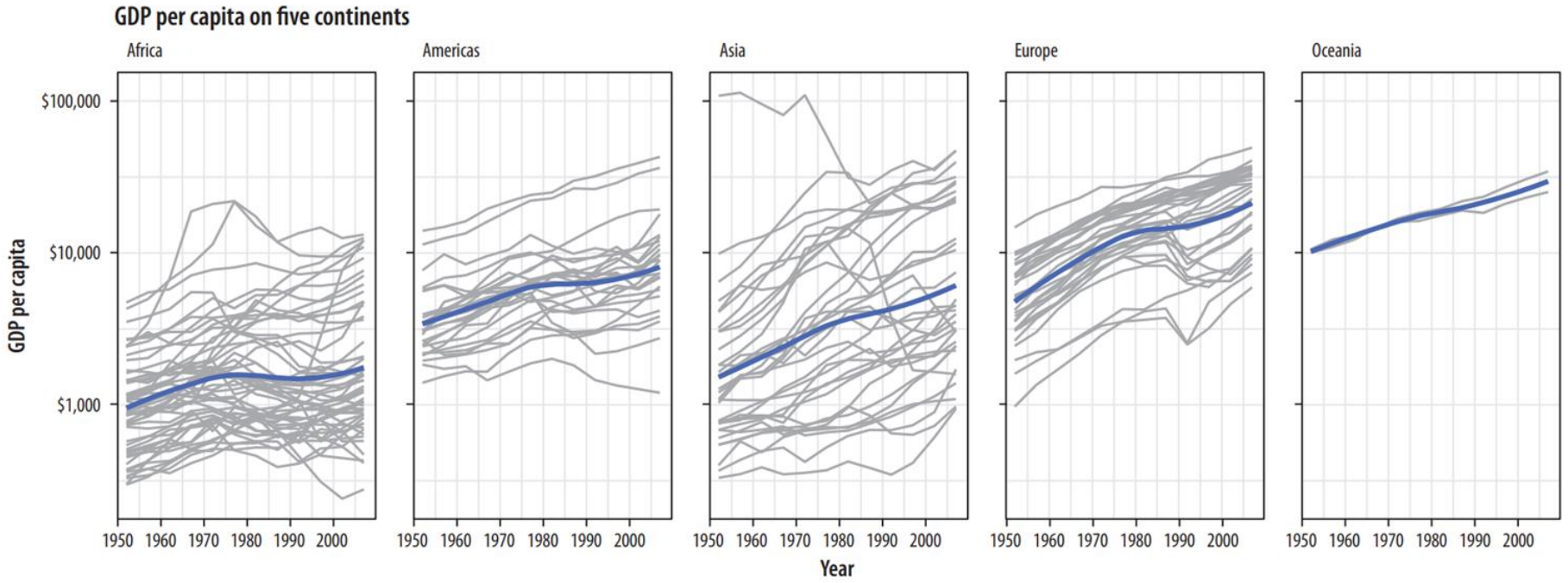
Activity - Improving our plot

- **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(size=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



Faceting data with ggplot (Part 2!)

facet_wrap() vs facet_grid()

- The `facet_wrap()` function is useful when we want a series of small multiples based on a single categorical variable
- If we want to cross-classify data by two categorical variables, we can use `facet_grid()` instead
- To explore `facet_grid()`, we will load a new sample dataset (**gss_sm**) that contains several categorical measures

(Healy, 2018)

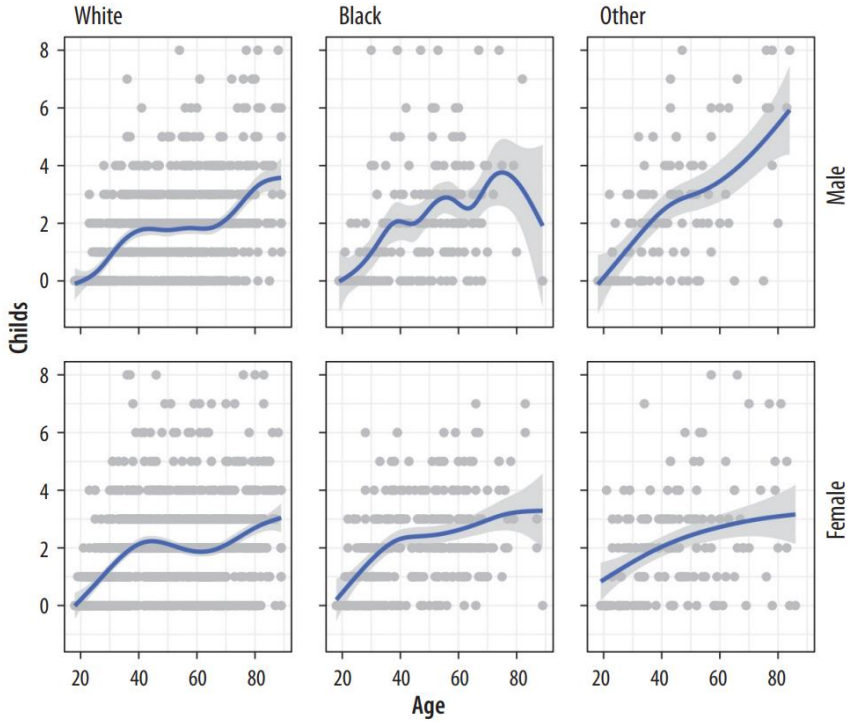
Using facet_grid()

- We will use the gss_sm dataset to make a smoothed scatterplot of the relationship between respondent age and number of children they have
  - The `age` variable is the age of the respondent
  - The `childs` variable is a numeric count of respondent's children
  - We will facet our plot by `sex` and `race` of the respondent

```
p <- ggplot(data=gss_sm, mapping = aes(x=age, y=childs))

p + geom_point(alpha=0.2) + geom_smooth() + facet_grid(sex~race)
```

(Healy, 2018)

Using facet_grid() - Result




```
p <- ggplot(data=gss_sm, mapping = aes(x=age, y=childs))

p + geom_point(alpha=0.2) + geom_smooth() + facet_grid(sex~race)
```

(Healy, 2018)

Transforming data with ggplot

Geoms can transform data

- Throughout this class, we have mostly seen geoms such as `geom_point()` which plot our data directly on our figures
- We have also seen geoms such as `geom_smooth()` that transform our data (eg. plotting a LOESS line versus plotting a line from ordinary least squares regression)
- The `geom_smooth()` function performs these transformations without us explicitly specifying → This is because each geom has an associated `stat_` function that it uses by default
- We can change this `stat_` if we want to calculate a different statistic for the geom

(Healy, 2018)

How do stat_ functions work?

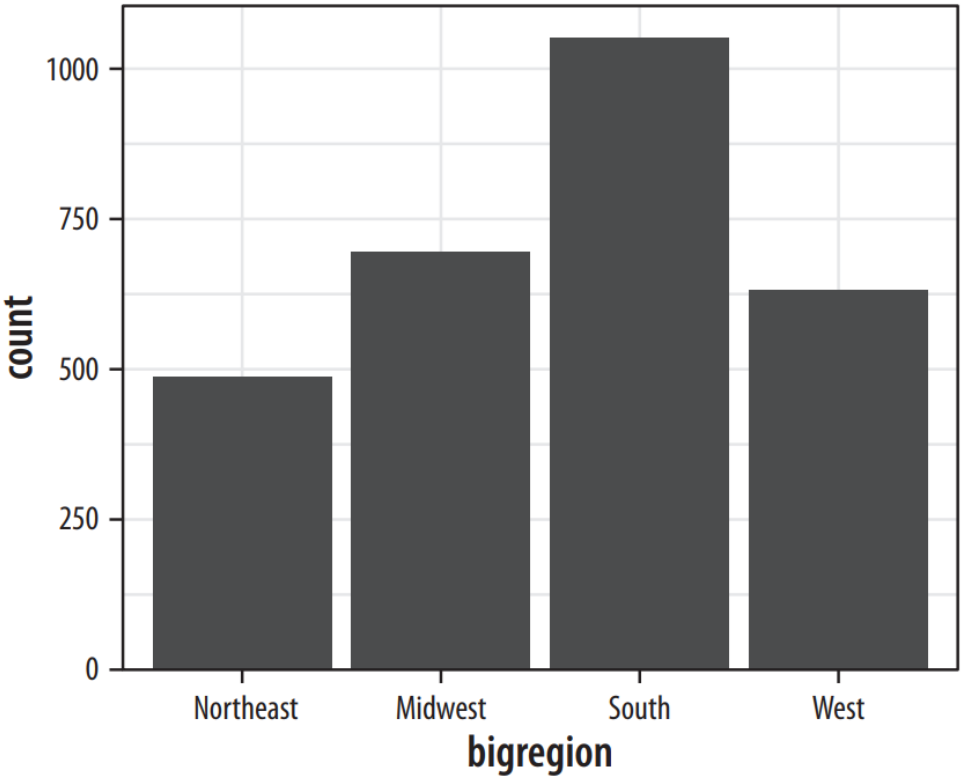
- We can make a default `geom_bar()` plot using our gss_sm dataset

```
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p + geom_bar()
```

- Note that we only specified one mapping, `aes(x = bigregion)`

(Healy, 2018)

How do stat_ functions work?



- Even though we only specified an x variable, there is a y variable ('count') that has been calculated automatically for us by the default `stat_count()` function associated with `geom_bar()`

(Healy, 2018)

Using a non-default stat_

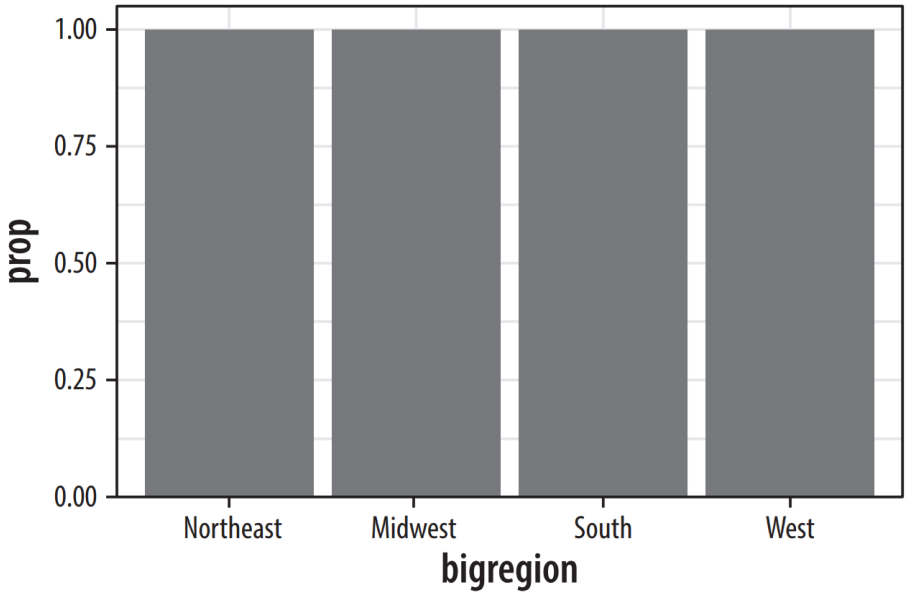
- If we want our chart to show relative frequencies rather than counts, we can use the prop ('proportion') statistic instead

```
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p + geom_bar(mapping = aes(y = ..prop..))
```

- `..prop..` is a temporary variable calculated by ggplot that we can use as a mapping in our plot
  - The two periods at the beginning and end of our temporary variable show that it was calculated as our statistic
  - The general format is `<mapping> = <..statistic..>`

(Healy, 2018)

Using a non-default stat_ - Initial result

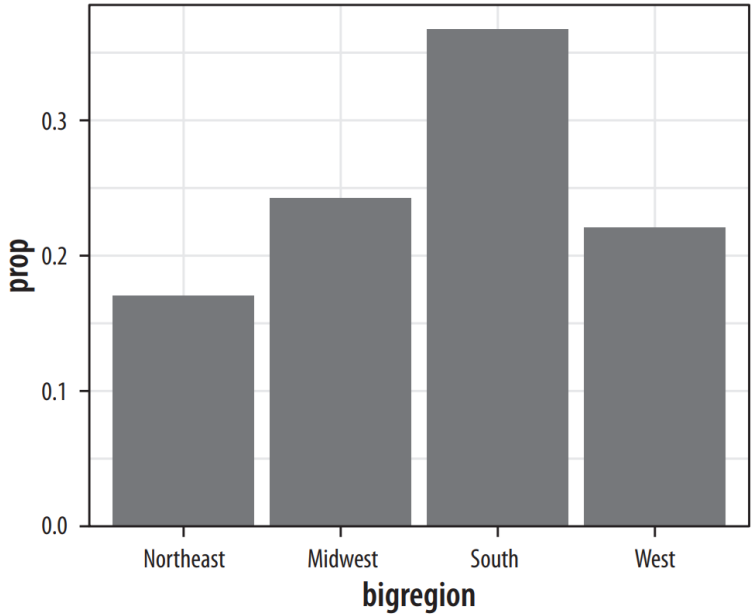



```
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p + geom_bar(mapping = aes(y = ..prop..))
```

- This plot is still not right. All bars have a value of 1, but we want them to *sum* to 1 so we get number of observations per region as a proportion of the total
- This is a grouping issue

(Healy, 2018)

Using a non-default stat_ - Result




```
p <- ggplot(data=gss_sm, mapping = aes(x = bigregion))
p + geom_bar(mapping = aes(y = ..prop.., group = 1))
```

- We create a 'dummy group' with `group = 1` to tell ggplot to use the whole dataset when establishing the denominator for `..prop..` calculations
- Now our chart has correct proportions

(Healy, 2018)

Frequency plots with ggplot

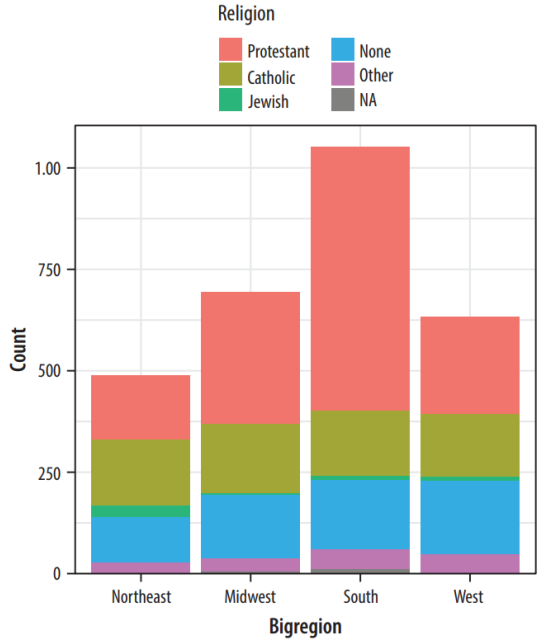
Setting up our new plot

- The variable `religion` is a categorical variable about participants' religious affiliation
- We can plot religious affiliation by census region as a bar chart, and colour the bars differently for each religion:

```
p <- ggplot(data = gss_sm, mapping = aes(x = bigregion, fill = religion))

p + geom_bar()
```

(Healy, 2018)

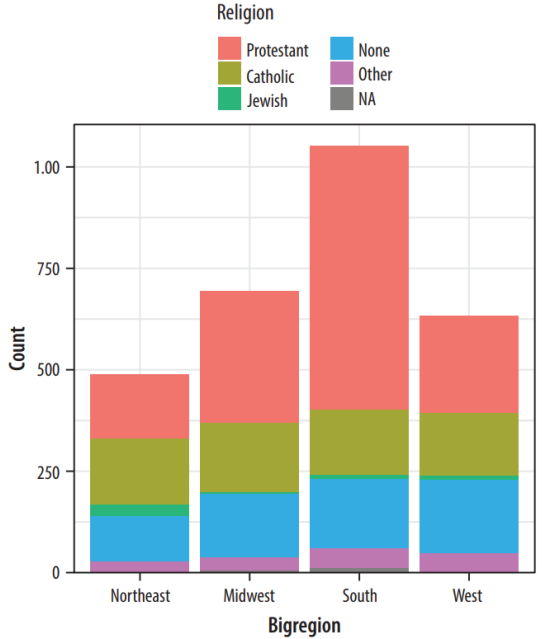
Setting up our new plot - Initial result




```
p ← ggplot(data = gss_sm, mapping = aes(x = bigregion, fill = religion))

p + geom_bar()
```

(Healy, 2018)

Setting up our new plot - Initial result

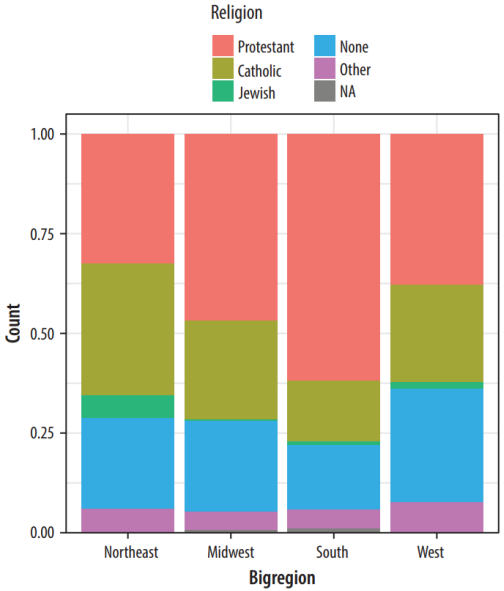


- **It is challenging for readers to compare relative lengths and areas on an unaligned scale** (as shown in this graph)
- We can remedy this by setting the 'position' argument of our geom_bar() to 'fill'

```
p ← ggplot(data = gss_sm, mapping = aes(x = bigregion, fill = religion))

p + geom_bar()
```

(Healy, 2018)

Setting up our new plot - Initial result

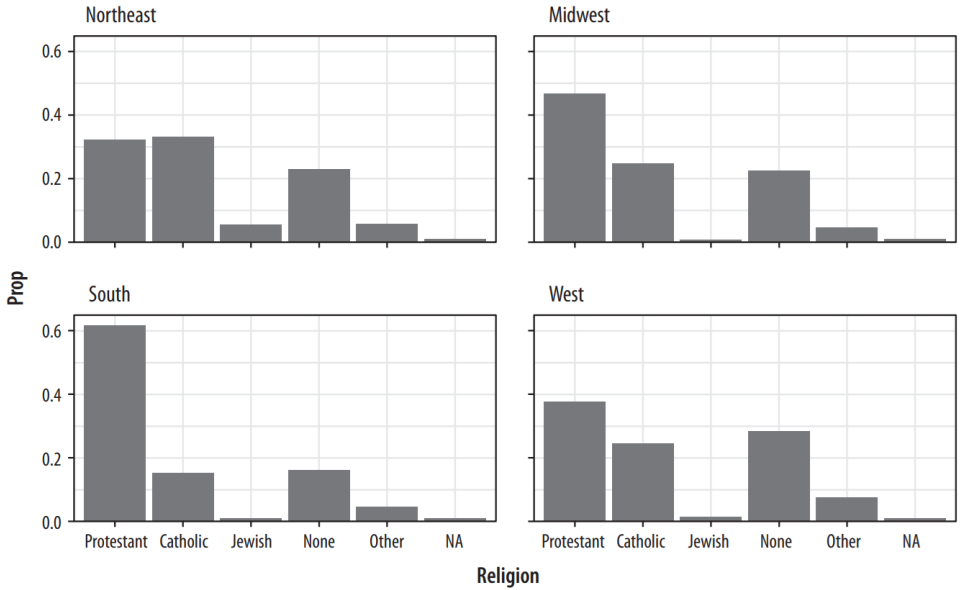


- Now we can compare proportions across groups
- BUT we cannot see the relative size of each cut with respect to the overall total
- Our frequency chart can benefit from faceting!

```
p ← ggplot(data = gss_sm, mapping = aes(x = bigregion, fill = religion))

p + geom_bar(position = "fill")
```

(Healy, 2018)

A faceted frequency plot



- By using facets, we see the breakdown we want of proportional religious affiliation by region

```
p <- ggplot(data = gss_sm, mapping = aes(x = religion))

p + geom_bar(position = "dodge", mapping = aes(y = ..prop.., group =
bigregion)) + facet_wrap(~bigregion, ncol = 1)
```

(Healy, 2018)

Histograms and density plots with ggplot

Histograms

- A histogram summarizes a continuous variable by dividing it into ‘bins’ and counting how many observations are found in each bin
- Bar charts use pre-existing categories (eg. religious affiliation), but with histograms, we decide how many bins to use
- We will use the ggplot sample dataset **midwest** to explore histograms of distribution of geographic areas of counties in the Midwestern United States

(Healy, 2018)

Histograms - Choosing bins

- By default, ggplot's `geom_histogram()` function will choose a bin size for us:

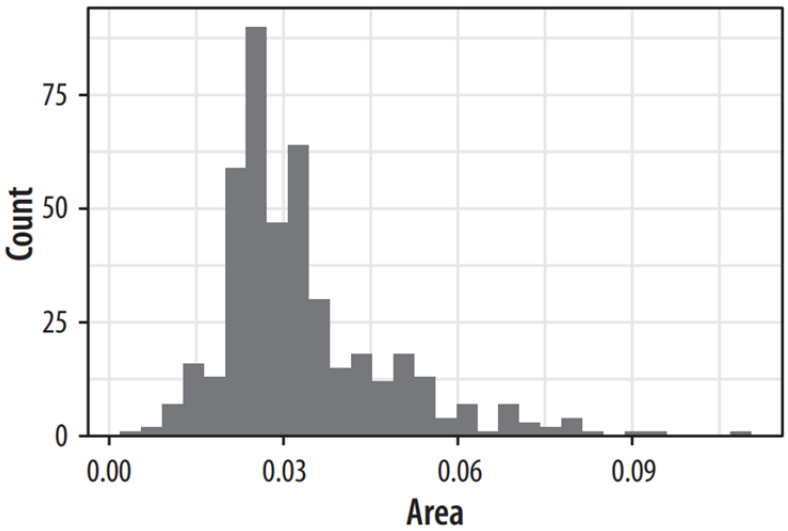
```
p <- ggplot(data = midwest, mapping = aes(x = area))

p + geom_histogram()
```

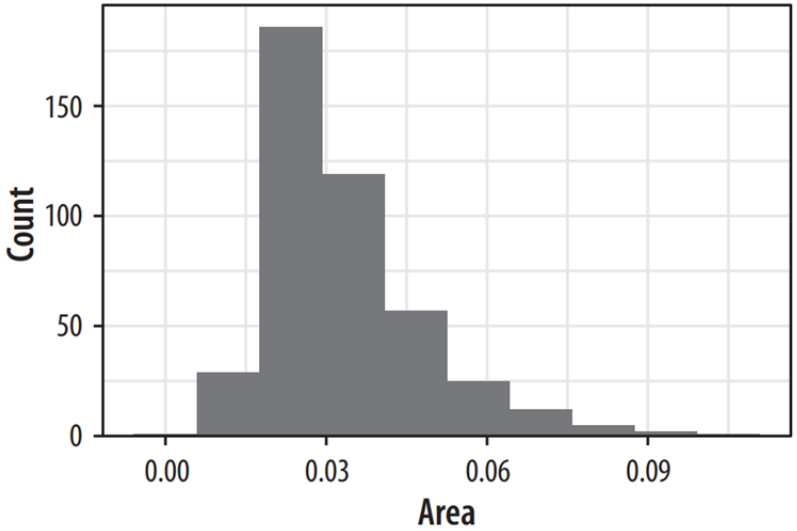
- We can also manually set the number of bins we want:

```
p ← ggplot(data = midwest, mapping = aes(x = area))

p + geom_histogram(bins = 10)
```

(Healy, 2018)

Histograms - Choosing bins






```
p ← ggplot(data = midwest, mapping = aes(x = area))

p + geom_histogram()
```

```
p ← ggplot(data = midwest, mapping = aes(x = area))

p + geom_histogram(bins = 10)
```

(Healy, 2018)

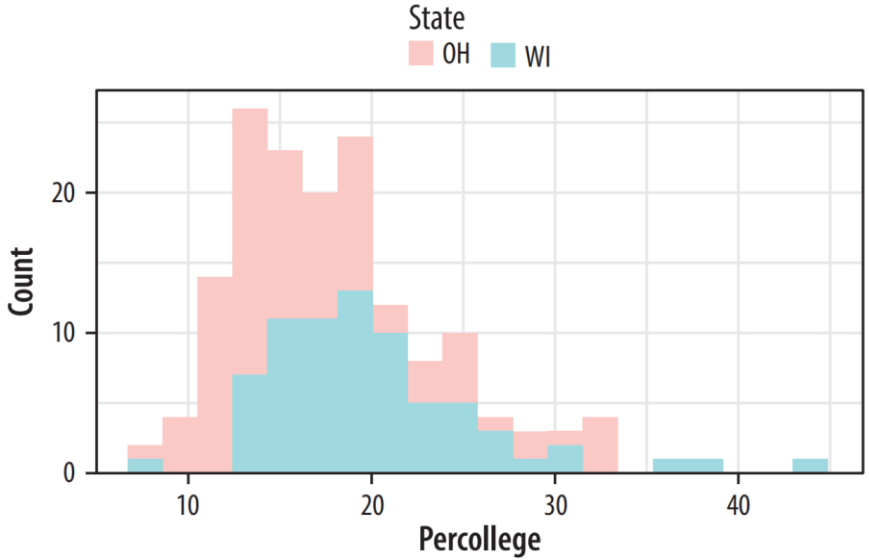
Histograms - Comparing distributions

- We can use several histograms at once to compare distributions, either by faceting or by showing them in the same plot using the fill mapping
- Here, we subset our data to choose from a character vector of two states (“OH” and “WI”), then filter so we only view observations whose state name is part of that vector:

```
oh_wi <- c("OH", "WI")

p <- ggplot(data = subset(midwest, subset = state %in% oh_wi),
mapping = aes(x = percollege, fill = state))

p + geom_histogram(alpha = 0.4, bins = 20)
```

(Healy, 2018)

Histograms - Comparing distributions

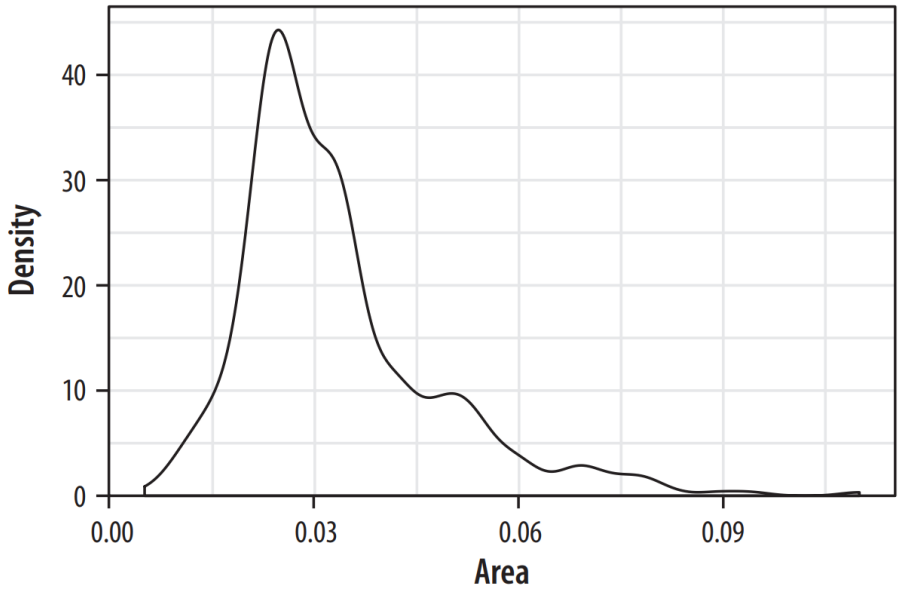



```
oh_wi ← c("OH", "WI")

p ← ggplot(data = subset(midwest, subset = state %in% oh_wi), mapping = aes(x = percollege, fill = state))

p + geom_histogram(alpha = 0.4, bins = 20)
```

- The ‘fill’ argument colours our histograms based on their state
- The `subset()` function ensures we only plot data from one of two states (from our oh_wi vector)

(Healy, 2018)

Density plots



- If our variable is continuous, an alternative to binning the data in a histogram is to use the `geom_density()` function to estimate the underlying distribution

```
p <- ggplot(data = midwest,mapping = aes(x = area))

p + geom_density()
```

(Healy, 2018)

Modifying density plots

- We can alter the colour and transparency of our density plots to make them easier to read

```
oh_wi <- c("OH", "WI")

p <- ggplot(data = subset(midwest, subset = state %in% oh_wi),
mapping = aes(x = area, fill = state, color = state))

p + geom_density(alpha = 0.3)
```

(Healy, 2018)

Next...

- How do we know what kind of data visualization to make for a given situation? (Professional skills)
- Perceptual qualities of data visualization, ‘neutral’ data visualization, using data visualization to make a point