Data Science Visualization

The textbook for the Data Science course series is freely available online.

Learning Objectives

- Data visualization principles to better communicate data-driven findings
- How to use ggplot2 to create custom plots
- The weaknesses of several widely used plots and why you should avoid them

Course Overview

Section 1: Introduction to Data Visualization and Distributions

You will get started with data visualization and distributions in R.

Section 2: Introduction to ggplot2

You will learn how to use ggplot2 to create plots.

Section 3: Summarizing with dplyr

You will learn how to summarize data using dplyr.

Section 4: Gapminder

You will see examples of ggplot2 and dplyr in action with the Gapminder dataset.

Section 5: Data Visualization Principles

You will learn general principles to guide you in developing effective data visualizations.

Section 1 Overview

Section 1 introduces you to Data Visualization and Distributions.

After completing Section 1, you will:

- understand the importance of data visualization for communicating data-driven findings.
- be able to use distributions to summarize data.
- be able to use the average and the standard deviation to understand the normal distribution.
- be able to assess how well a normal distribution fits the data using a quantile-quantile plot.
- be able to interpret data from a boxplot.

Introduction to Data Visualization

The textbook for this section is available here

Key points

- Plots of data easily communicate information that is difficult to extract from tables of raw values.
- Data visualization is a key component of exploratory data analysis (EDA), in which the properties of data are explored through visualization and summarization techniques.
- Data visualization can help discover biases, systematic errors, mistakes and other unexpected problems in data before those data are incorporated into potentially flawed analysis.
- This course covers the basics of data visualization and EDA in R using the **ggplot2** package and motivating examples from world health, economics and infectious disease.

Code

```
if(!require(dslabs)) install.packages("dslabs")
```

```
## Loading required package: dslabs
```

```
library(dslabs)
data(murders)
head(murders)
```

```
##        state abb region population total
## 1    Alabama  AL  South    4779736   135
## 2     Alaska  AK   West     710231    19
## 3    Arizona  AZ   West    6392017   232
## 4   Arkansas  AR  South    2915918    93
## 5 California  CA   West   37253956  1257
## 6   Colorado  CO   West    5029196    65
```

Introduction to Distributions

The textbook for this section is available here

Key points

- The most basic statistical summary of a list of objects is its distribution.
- We will learn ways to visualize and analyze distributions in the upcoming videos.
- In some cases, data can be summarized by a two-number summary: the average and standard deviation. We will learn to use data visualization to determine when that is appropriate.

Data Types

The textbook for this section is available here

Key points

- Categorical data are variables that are defined by a small number of groups.
  - Ordinal categorical data have an inherent order to the categories (mild/medium/hot, for example).

  - Non-ordinal categorical data have no order to the categories.
- Numerical data take a variety of numeric values.
  - Continuous variables can take any value.
  - Discrete variables are limited to sets of specific values.

Assessment - Data Types

1. The type of data we are working with will often influence the data visualization technique we use.

We will be working with two types of variables: categorical and numeric. Each can be divided into two other groups: categorical can be ordinal or not, whereas numerical variables can be discrete or continuous.

We will review data types using some of the examples provided in the `dslabs` package. For example, the `heights` dataset.

```r
library(dslabs)
data(heights)
```

```r
data(heights)
names(heights)
```

```
## [1] "sex"    "height"
```

2. We saw that `sex` is the first variable. We know what values are represented by this variable and can confirm this by looking at the first few entires:

```r
head(heights)
```

```
##      sex height
## 1   Male     75
## 2   Male     70
## 3   Male     68
## 4   Male     74
## 5   Male     61
## 6 Female     65
```

What data type is the `sex` variable?

- ☐ A. Continuous
- ☒ B. Categorical
- ☐ C. Ordinal
- ☐ D. None of the above

3. Keep in mind that discrete numeric data can be considered ordinal.

Although this is technically true, we usually reserve the term ordinal data for variables belonging to a small number of different groups, with each group having many members.

The `height` variable could be ordinal if, for example, we report a small number of values such as short, medium, and tall. Let's explore how many unique values are used by the heights variable. For this we can use the `unique` function:

```
x <- c(3, 3, 3, 3, 4, 4, 2)
unique(x)
```

```
x <- heights$height
length(unique(x))
```

```
## [1] 139
```

4. One of the useful outputs of data visualization is that we can learn about the distribution of variables.

For categorical data we can construct this distribution by simply computing the frequency of each unique value. This can be done with the function `table`. Here is an example:

```
x <- c(3, 3, 3, 3, 4, 4, 2)
table(x)
```

```
x <- heights$height
tab <- table(x)
```

5. To see why treating the reported heights as an ordinal value is not useful in practice we note how many values are reported only once.

In the previous exercise we computed the variable tab which reports the number of times each unique value appears. For values reported only once tab will be 1. Use logicals and the function sum to count the number of times this happens.

```
tab <- table(heights$height)
sum(tab==1)
```

```
## [1] 63
```

6. Since there are a finite number of reported heights and technically the height can be considered ordinal, which of the following is true:

- ☒ A. It is more effective to consider heights to be numerical given the number of unique values we observe and the fact that if we keep collecting data even more will be observed.
- ☐ B. It is actually preferable to consider heights ordinal since on a computer there are only a finite number of possibilities.
- ☐ C. This is actually a categorical variable: tall, medium or short.
- ☐ D. This is a numerical variable because numbers are used to represent it.

Describe Heights to ET

The textbook for this section is available:

- Case Study describing student heights
- Distribution Function
- CDF Intro
- Histograms

Key points

- A distribution is a function or description that shows the possible values of a variable and how often those values occur.
- For categorical variables, the distribution describes the proportions of each category.
- A *frequency table* is the simplest way to show a categorical distribution. Use `prop.table` to convert a table of counts to a frequency table. *Barplots* display the distribution of categorical variables and are a way to visualize the information in frequency tables.
- For continuous numerical data, reporting the frequency of each unique entry is not an effective summary as many or most values are unique. Instead, a distribution function is required.
- The *cumulative distribution function (CDF)* is a function that reports the proportion of data below a value a for all values of a: $F(a) = Pr(x \leq a)$.
- The proportion of observations between any two values a and b can be computed from the CDF as $F(b) - F(a)$.
- A *histogram* divides data into non-overlapping bins of the same size and plots the counts of number of values that fall in that interval.

Code

```
# load the dataset
library(dslabs)
data(heights)
```

```
# make a table of category proportions
prop.table(table(heights$sex))
```

```
##
##    Female      Male
## 0.2266667 0.7733333
```

Smooth Density Plots

The textbook for this section is available here

Key points

- *Smooth density plots* can be thought of as histograms where the bin width is extremely or infinitely small. The smoothing function makes estimates of the true continuous trend of the data given the available sample of data points.
- The degree of smoothness can be controlled by an argument in the plotting function. (We will learn functions for plotting later.)
- While the histogram is an assumption-free summary, the smooth density plot is shaped by assumptions and choices you make as a data analyst.
- The y-axis is scaled so that the area under the density curve sums to 1. This means that interpreting values on the y-axis is not straightforward. To determine the proportion of data in between two values, compute the area under the smooth density curve in the region between those values.
- An advantage of smooth densities over histograms is that densities are easier to compare visually.

A further note on histograms: note that the choice of binwidth has a determinative effect on shape. There is no "true" choice for binwidth, and you can sometimes gain insights into the data by experimenting with binwidths.

Assessment - Distributions

1. You may have noticed that numerical data is often summarized with the average value.

For example, the quality of a high school is sometimes summarized with one number: the average score on a standardized test. Occasionally, a second number is reported: the standard deviation. So, for example, you might read a report stating that scores were 680 plus or minus 50 (the standard deviation). The report has summarized an entire vector of scores with with just two numbers. Is this appropriate? Is there any important piece of information that we are missing by only looking at this summary rather than the entire list? We are going to learn when these 2 numbers are enough and when we need more elaborate summaries and plots to describe the data.

Our first data visualization building block is learning to summarize lists of factors or numeric vectors. The most basic statistical summary of a list of objects or numbers is its distribution. Once a vector has been summarized as distribution, there are several data visualization techniques to effectively relay this information. In later assessments we will practice to write code for data visualization. Here we start with some multiple choice questions to test your understanding of distributions and related basic plots.

In the murders dataset, the region is a categorical variable and on the right you can see its distribution. To the closest 5%, what proportion of the states are in the North Central region?

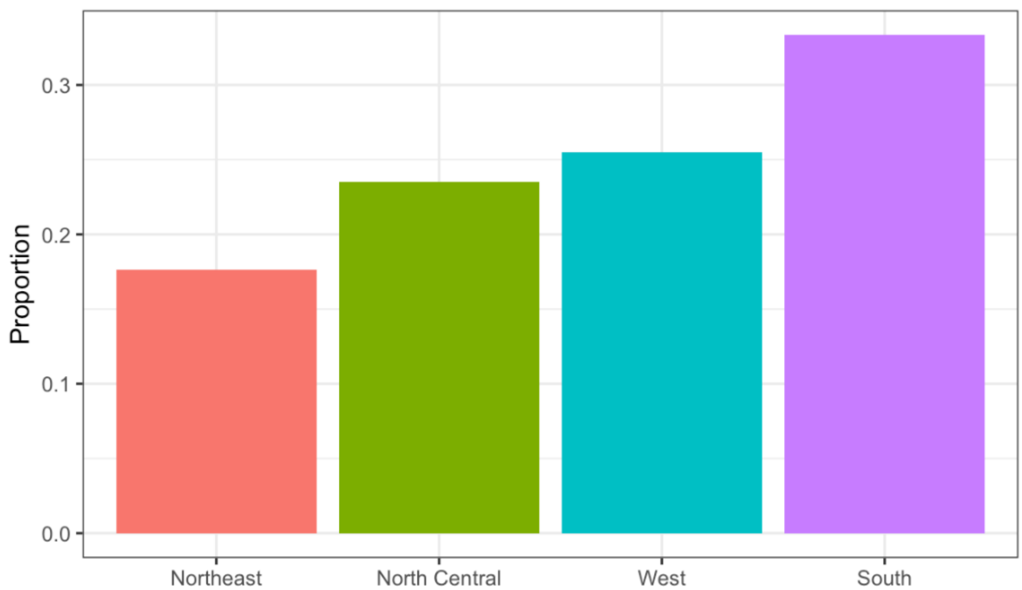


Figure 1: Region vs. Proportion

- ☐ A. 75%
- ☐ B. 50%
- ☒ C. 20%
- ☐ D. 5%

2. In the murders dataset, the region is a categorical variable and to the right is its distribution.

Which of the following is true:

- ☐ A. The graph above is a histogram.

☒ B. The graph above shows only four numbers with a bar plot.
☐ C. Categories are not numbers, so it does not make sense to graph the distribution.
☐ D. The colors, not the height of the bars, describe the distribution.

3. The plot shows the eCDF for male heights.

Based on the plot, what percentage of males are shorter than 75 inches?

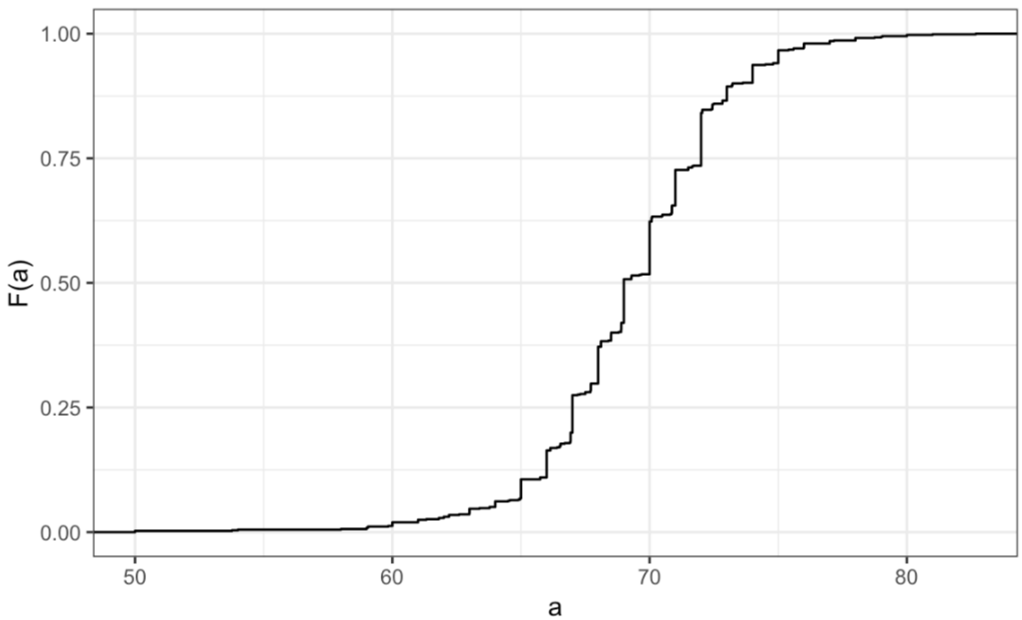


Figure 2: eCDF for male heights

☐ A. 100%
☒ B. 95%
☐ C. 80%
☐ D. 72 inches

4. To the closest inch, what height `m` has the property that 1/2 of the male students are taller than `m` and 1/2 are shorter?

☐ A. 61 inches
☐ B. 64 inches
☒ C. 69 inches
☐ D. 74 inches

5. Here is an eCDF of the murder rates across states.

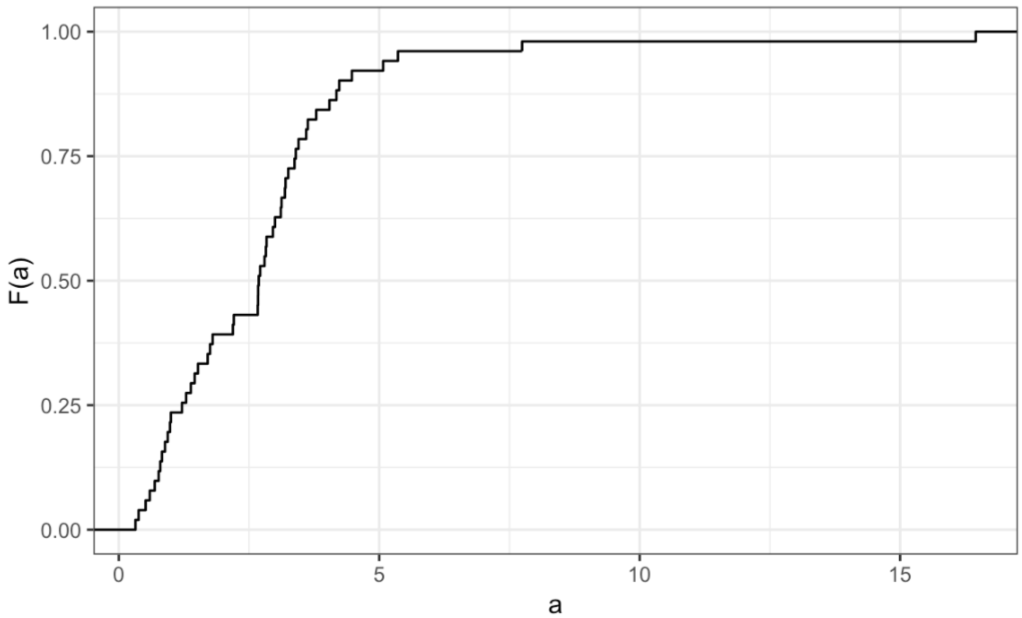


Figure 3: eCDF of the murder rates across states

Knowing that there are 51 states (counting DC) and based on this plot, how many states have murder rates larger than 10 per 100,000 people?

☒ A. 1
☐ B. 5
☐ C. 10
☐ D. 50

6. Based on the eCDF above, which of the following statements are true.

☐ A. About half the states have murder rates above 7 per 100,000 and the other half below.
☐ B. Most states have murder rates below 2 per 100,000.
☐ C. All the states have murder rates above 2 per 100,000.
☒ D. With the exception of 4 states, the murder rates are below 5 per 100,000.

7. Here is a histogram of male heights in our `heights` dataset.

Based on this plot, how many males are between 62.5 and 65.5?

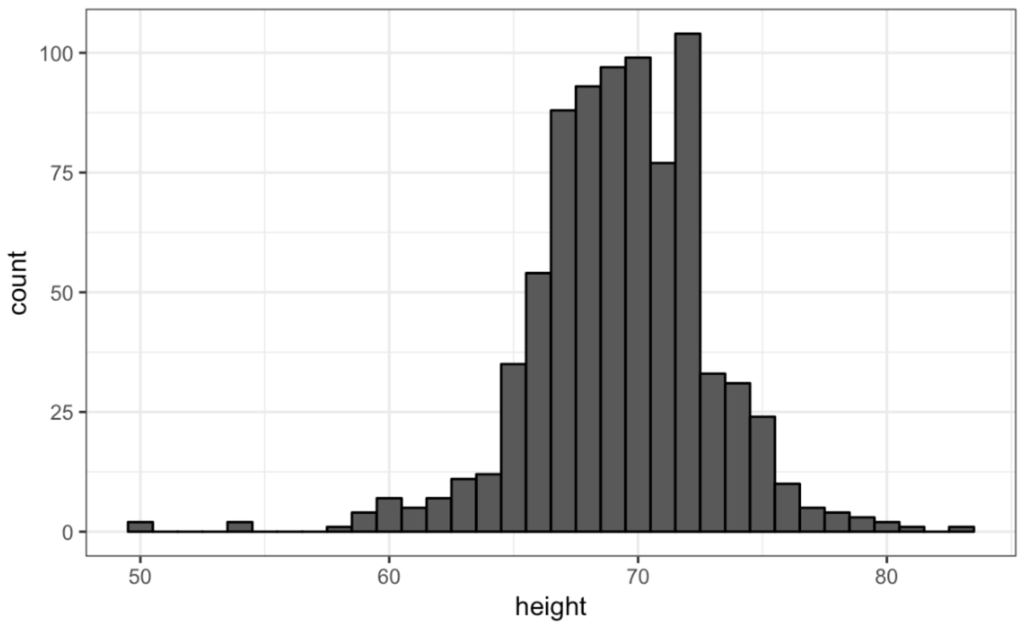


Figure 4: Histogram of male heights

☐ A. 11
☐ B. 29
☒ C. 58
☐ D. 99

8. About what percentage are shorter than 60 inches?

☒ A. 1%
☐ B. 10%
☐ C. 25%
☐ D. 50%

9. Based on this density plot, about what proportion of US states have populations larger than 10 million?

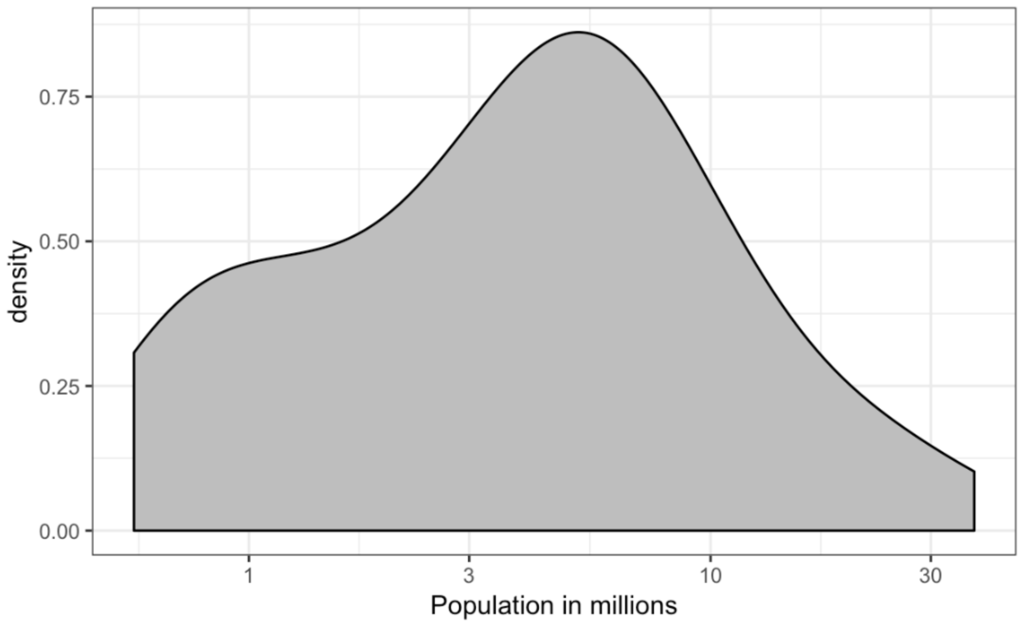


Figure 5: Density plot population

☐ A. 0.02
☒ B. 0.15
☐ C. 0.50
☐ D. 0.55

10. Below are three density plots. Is it possible that they are from the same dataset?

Which of the following statements is true?

☐ A. It is impossible that they are from the same dataset.
☐ B. They are from the same dataset, but the plots are different due to code errors.
☐ C. They are the same dataset, but the first and second plot undersmooth and the third oversmooths.
☒ D. They are the same dataset, but the first is not in the log scale, the second undersmooths and the third oversmooths.

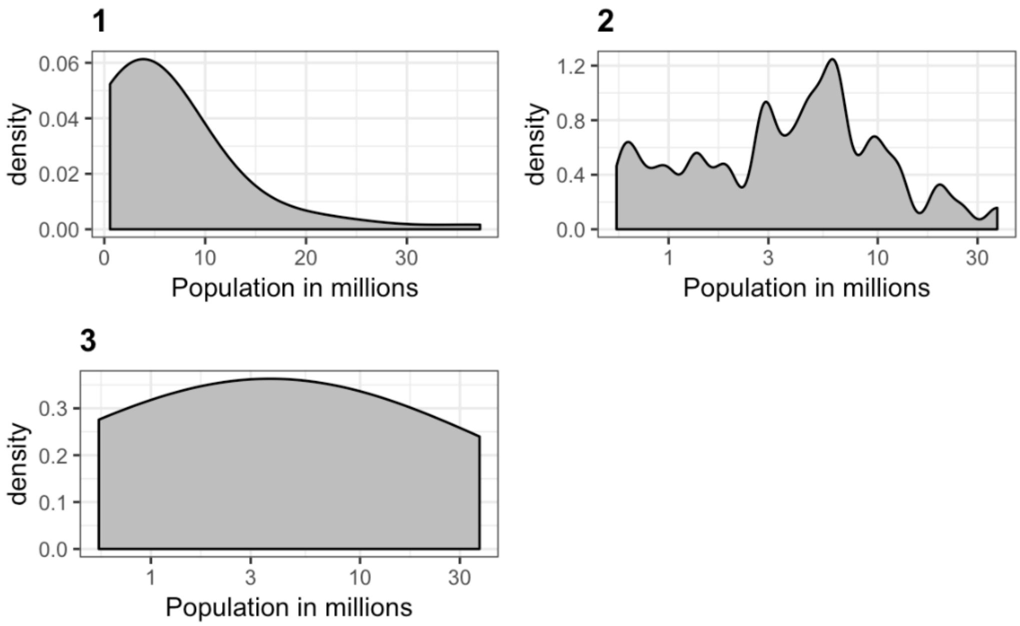


Figure 6: Three density plots

Normal Distribution

The textbook for this section is available here

Key points

- The normal distribution:
  - Is centered around one value, the *mean*
  - Is symmetric around the mean
  - Is defined completely by its mean (μ) and standard deviation (σ)
  - Always has the same proportion of observations within a given distance of the mean (for example, 95% within 2 σ)
- The standard deviation is the average distance between a value and the mean value.
- Calculate the mean using the `mean` function.
- Calculate the standard deviation using the `sd` function or manually.
- Standard units describe how many standard deviations a value is away from the mean. The z-score, or number of standard deviations an observation x is away from the mean (μ):

  $Z = \frac{x-\mu}{\sigma}$
- Compute standard units with the `scale` function.
- **Important:** to calculate the proportion of values that meet a certain condition, use the `mean` function on a logical vector. Because TRUE is converted to 1 and FALSE is converted to 0, taking the mean of this vector yields the proportion of TRUE.

Equation for the normal distribution

The normal distribution is mathematically defined by the following formula for any mean μ and standard deviation σ:

$Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx$

Code

```
if(!require(tidyverse)) install.packages("tidyverse")
```

```
## Loading required package: tidyverse

## -- Attaching packages ------------------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ----------------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# define x as vector of male heights
library(tidyverse)
index <- heights$sex=="Male"
x <- heights$height[index]

# calculate the mean and standard deviation manually
average <- sum(x)/length(x)
SD <- sqrt(sum((x - average)^2)/length(x))

# built-in mean and sd functions - note that the audio and printed values disagree
average <- mean(x)
SD <- sd(x)
c(average = average, SD = SD)
```

```
##   average        SD
## 69.314755  3.611024
```

```
# calculate standard units
z <- scale(x)

# calculate proportion of values within 2 SD of mean
mean(abs(z) < 2)
```

```
## [1] 0.9495074
```

Note about the sd function: The built-in R function `sd` calculates the standard deviation, but it divides by `length(x)-1` instead of `length(x)`. When the length of the list is large, this difference is negligible and you can use the built-in `sd` function. Otherwise, you should compute σ by hand. For this course series, assume that you should use the `sd` function unless you are told not to do so.

Assessment - Normal Distribution

1. Histograms and density plots provide excellent summaries of a distribution.

But can we summarize even further? We often see the average and standard deviation used as summary statistics: a two number summary! To understand what these summaries are and why they are so widely used, we need to understand the normal distribution.

The normal distribution, also known as the bell curve and as the Gaussian distribution, is one of the most famous mathematical concepts in history. A reason for this is that approximately normal distributions occur in many situations. Examples include gambling winnings, heights, weights, blood pressure, standardized test scores, and experimental measurement errors. Often data visualization is needed to confirm that our data follows a normal distribution.

Here we focus on how the normal distribution helps us summarize data and can be useful in practice.

One way the normal distribution is useful is that it can be used to approximate the distribution of a list of numbers without having access to the entire list. We will demonstrate this with the heights dataset.

Load the height data set and create a vector `x` with just the male heights:

```
library(dslabs)
data(heights)
x <- heights$height[heights$sex == "Male"]
```

What proportion of the data is between 69 and 72 inches (taller than 69 but shorter or equal to 72)? A proportion is between 0 and 1.

```
x <- heights$height[heights$sex == "Male"]
mean(x > 69 & x <= 72)
```

```
## [1] 0.3337438
```

2. Suppose all you know about the height data from the previous exercise is the average and the standard deviation and that its distribution is approximated by the normal distribution.

We can compute the average and standard deviation like this:

```
library(dslabs)
data(heights)
x <- heights$height[heights$sex=="Male"]
avg <- mean(x)
stdev <- sd(x)
```

Suppose you only have `avg` and `stdev` below, but no access to `x`, can you approximate the proportion of the data that is between 69 and 72 inches?

Given a normal distribution with a mean `mu` and standard deviation `sigma`, you can calculate the proportion of observations less than or equal to a certain `value` with `pnorm(value, mu, sigma)`. Notice that this is the CDF for the normal distribution. We will learn much more about `pnorm` later in the course series, but you can also learn more now with `?pnorm`.

```r
x <- heights$height[heights$sex=="Male"]
avg <- mean(x)
stdev <- sd(x)
pnorm(72, avg, stdev) - pnorm(69, avg, stdev)
```

```
## [1] 0.3061779
```

3. Notice that the approximation calculated in the second question is very close to the exact calculation in the first question.

The normal distribution was a useful approximation for this case. However, the approximation is not always useful. An example is for the more extreme values, often called the "tails" of the distribution. Let's look at an example. We can compute the proportion of heights between 79 and 81.

```r
library(dslabs)
data(heights)
x <- heights$height[heights$sex == "Male"]
mean(x > 79 & x <= 81)
```

```r
x <- heights$height[heights$sex == "Male"]
avg <- mean(x)
stdev <- sd(x)
exact <- mean(x > 79 & x <= 81)
approx <- pnorm(81, avg, stdev) - pnorm(79, avg, stdev)
exact
```

```
## [1] 0.004926108
```

```r
approx
```

```
## [1] 0.003051617
```

```r
exact/approx
```

```
## [1] 1.614261
```

4. Someone asks you what percent of seven footers are in the National Basketball Association (NBA). Can you provide an estimate? Let's try using the normal approximation to answer this question.

First, we will estimate the proportion of adult men that are 7 feet tall or taller.

Assume that the distribution of adult men in the world as normally distributed with an average of 69 inches and a standard deviation of 3 inches.

```r
# use pnorm to calculate the proportion over 7 feet (7*12 inches)
1 - pnorm(7*12, 69, 3)
```

```
## [1] 2.866516e-07
```

5. Now we have an approximation for the proportion, call it `p`, of men that are 7 feet tall or taller.

We know that there are about 1 billion men between the ages of 18 and 40 in the world, the age range for the NBA.

Can we use the normal distribution to estimate how many of these 1 billion men are at least seven feet tall?

```
p <- 1 - pnorm(7*12, 69, 3)
round(p*10^9)
```

```
## [1] 287
```

6. There are about 10 National Basketball Association (NBA) players that are 7 feet tall or higher.

```
p <- 1 - pnorm(7*12, 69, 3)
N <- round(p*10^9)
10/N
```

```
## [1] 0.03484321
```

7. In the previous exerceise we estimated the proportion of seven footers in the NBA using this simple code:

```
p <- 1 - pnorm(7*12, 69, 3)
N <- round(p * 10^9)
10/N
```

Repeat the calculations performed in the previous question for Lebron James' height: 6 feet 8 inches. There are about 150 players, instead of 10, that are at least that tall in the NBA.

```
## Change the solution to previous answer
p <- 1 - pnorm(7*12, 69, 3)
N <- round(p * 10^9)
10/N
```

```
## [1] 0.03484321
```

```
p <- 1 - pnorm(6*12+8, 69, 3)
N <- round(p * 10^9)
150/N
```

```
## [1] 0.001220842
```

8. In answering the previous questions, we found that it is not at all rare for a seven footer to become an NBA player.

What would be a fair critique of our calculations?

- ☐ A. Practice and talent are what make a great basketball player, not height.
- ☐ B. The normal approximation is not appropriate for heights.
- ☒ C. As seen in exercise 3, the normal approximation tends to underestimate the extreme values. It's possible that there are more seven footers than we predicted.
- ☐ D. As seen in exercise 3, the normal approximation tends to overestimate the extreme values. It's possible that there are less seven footers than we predicted.

Quantile-Quantile Plots

The textbook for this section is available here

Key points

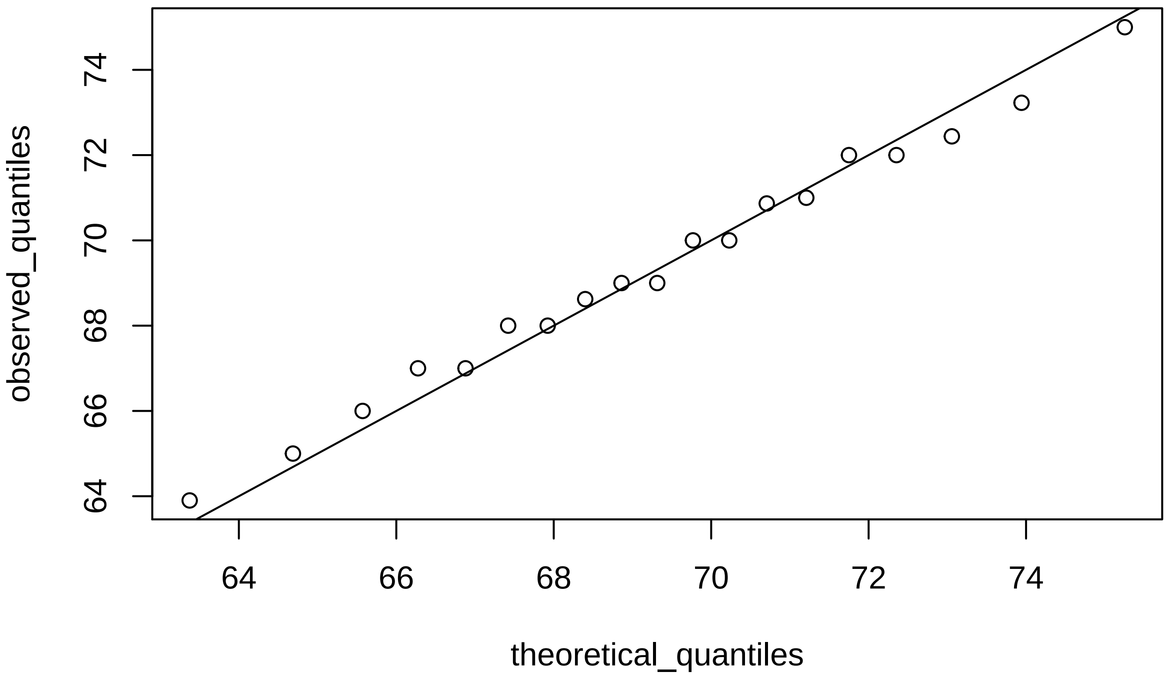
- Quantile-quantile plots, or QQ-plots, are used to check whether distributions are well-approximated by a normal distribution.
- Given a proportion p, the quantile q is the value such that the proportion of values in the data below q is p.
- In a QQ-plot, the sample quantiles in the observed data are compared to the theoretical quantiles expected from the normal distribution. If the data are well-approximated by the normal distribution, then the points on the QQ-plot will fall near the identity line (sample = theoretical).
- Calculate sample quantiles (observed quantiles) using the `quantile` function.
- Calculate theoretical quantiles with the `qnorm` function. `qnorm` will calculate quantiles for the standard normal distribution ($\mu = 0$, $\sigma = 1$) by default, but it can calculate quantiles for any normal distribution given mean and `sd` arguments. We will learn more about `qnorm` in the probability course.
- Note that we will learn alternate ways to make QQ-plots with less code later in the series.

Code

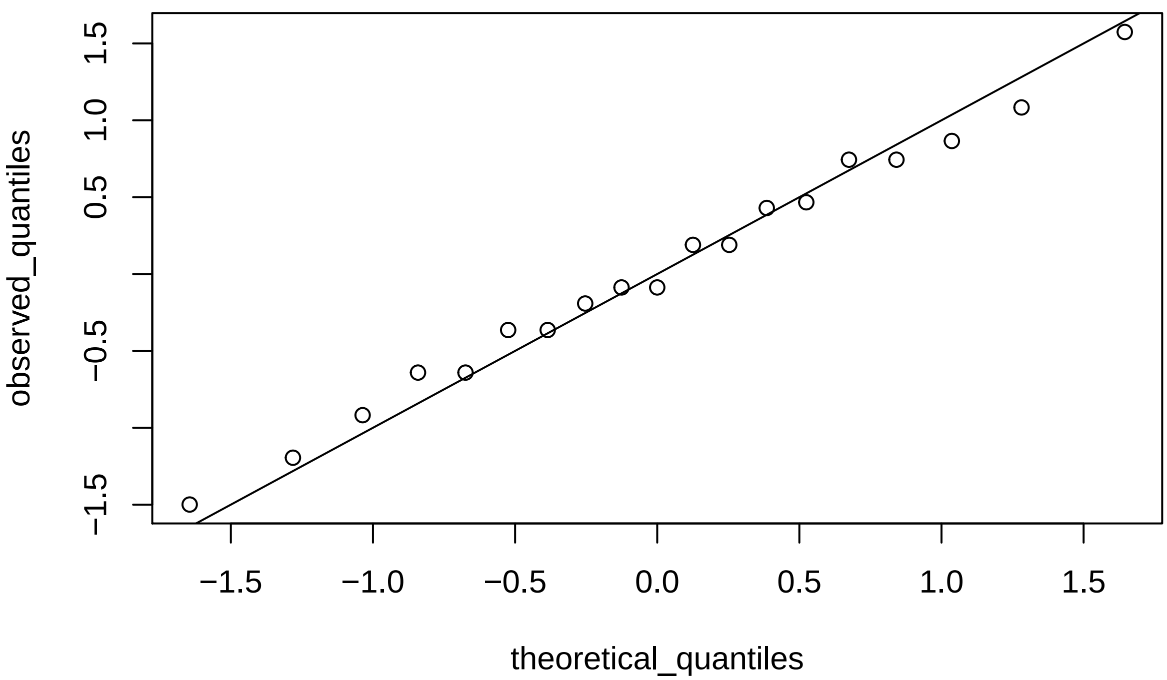
```
# define x and z
index <- heights$sex=="Male"
x <- heights$height[index]
z <- scale(x)

# proportion of data below 69.5
mean(x <= 69.5)
```

```
## [1] 0.5147783
```

```
# calculate observed and theoretical quantiles
p <- seq(0.05, 0.95, 0.05)
observed_quantiles <- quantile(x, p)
theoretical_quantiles <- qnorm(p, mean = mean(x), sd = sd(x))

# make QQ-plot
plot(theoretical_quantiles, observed_quantiles)
abline(0,1)
```




```
# make QQ-plot with scaled values
observed_quantiles <- quantile(z, p)
theoretical_quantiles <- qnorm(p)
plot(theoretical_quantiles, observed_quantiles)
abline(0,1)
```



Percentiles

The textbook for this section is available here

Key points

- *Percentiles* are the quantiles obtained when defining p as `0.01,0.02,...,0.99`. They summarize the values at which a certain percent of the observations are equal to or less than that value.
- The 50th percentile is also known as the *median*.
- The *quartiles* are the 25th, 50th and 75th percentiles.

Boxplots

The textbook for this section is available here

Key points

- When data do not follow a normal distribution and cannot be succinctly summarized by only the mean and standard deviation, an alternative is to report a five-number summary: range (ignoring outliers) and the quartiles (25th, 50th, 75th percentile).
- In a *boxplot*, the box is defined by the 25th and 75th percentiles and the median is a horizontal line through the box. The whiskers show the range excluding outliers, and outliers are plotted separately as individual points.
- The *interquartile* range is the distance between the 25th and 75th percentiles.
- Boxplots are particularly useful when comparing multiple distributions.
- We discuss outliers later.

Assessment - Quantiles, percentiles, and boxplots

1. When analyzing data it's often important to know the number of measurements you have for each category.

```
male <- heights$height[heights$sex=="Male"]
female <- heights$height[heights$sex=="Female"]
length(male)
```

```
## [1] 812
```

```
length(female)
```

```
## [1] 238
```

2. Suppose we can't make a plot and want to compare the distributions side by side. If the number of data points is large, listing all the numbers is inpractical. A more practical approach is to look at the percentiles. We can obtain percentiles using the `quantile` function like this

```
library(dslabs)
data(heights)
quantile(heights$height, seq(.01, 0.99, 0.01))
```

```
male <- heights$height[heights$sex=="Male"]
female <- heights$height[heights$sex=="Female"]
female_percentiles <- quantile(female, seq(0.1, 0.9, 0.2))
male_percentiles <- quantile(male, seq(0.1, 0.9, 0.2))
df <- data.frame(female = (female_percentiles), male = (male_percentiles))
df
```

```
##       female     male
## 10% 61.00000 65.00000
## 30% 63.00000 68.00000
## 50% 64.98031 69.00000
## 70% 66.46417 71.00000
## 90% 69.00000 73.22751
```

3. Study the boxplots summarizing the distributions of populations sizes by country.

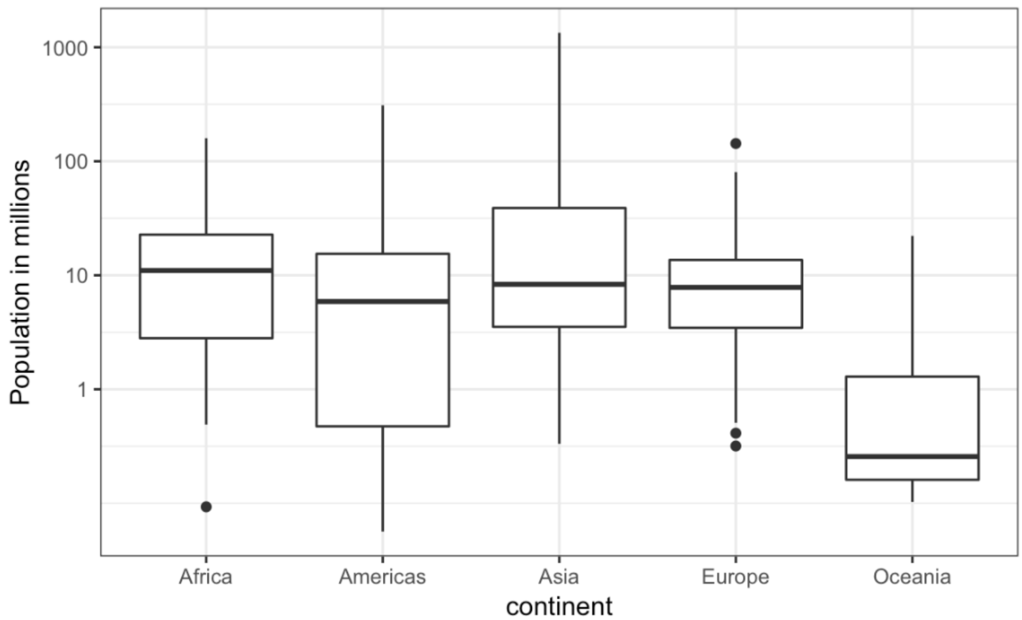


Figure 7: Continent vs Population

Which continent has the country with the largest population size?

- ☐ A. Africa
- ☐ B. Americas
- ☒ C. Asia
- ☐ D. Europe
- ☐ E. Oceania

4. Study the boxplots summarizing the distributions of populations sizes by country.

Which continent has median country with the largest population?

- ☒ A. Africa
- ☐ B. Americas
- ☐ C. Asia
- ☐ D. Europe
- ☐ E. Oceania

5. Again, look at the boxplots summarizing the distributions of populations sizes by country.

To the nearest million, what is the median population size for Africa?

- ☐ A. 100 million
- ☐ B. 25 million
- ☒ C. 10 million
- ☐ D. 5 million
- ☐ E. 1 million

6. Examine the following boxplots and report approximately what proportion of countries in Europe have populations below 14 million?

- ☒ A. 0.75
- ☐ B. 0.50
- ☐ C. 0.25
- ☐ D. 0.01

7. Based on the boxplot, if we use a log transformation, which continent shown below has the largest interquartile range?

- ☐ A. Africa
- ☒ B. Americas
- ☐ C. Asia
- ☐ D. Europe
- ☐ E. Oceania

Distribution of Female Heights

The textbook for this section is available here

Key points

- If a distribution is not normal, it cannot be summarized with only the mean and standard deviation. Provide a histogram, smooth density or boxplot instead.
- A plot can force us to see unexpected results that make us question the quality or implications of our data.

Assessment - Robust Summaries With Outliers

1. For this chapter, we will use height data collected by Francis Galton for his genetics studies. Here we just use height of the children in the dataset:

```
library(HistData)
data(Galton)
x <- Galton$child
```

```
if(!require(HistData)) install.packages("HistData")
```

```
## Loading required package: HistData
```

```
## Warning: package 'HistData' was built under R version 4.0.2
```

```
library(HistData)
data(Galton)
x <- Galton$child
mean(x)
```

```
## [1] 68.08847
```

```
median(x)
```

```
## [1] 68.2
```

2. Now for the same data compute the standard deviation and the median absolute deviation (MAD).

```
x <- Galton$child
sd(x)
```

```
## [1] 2.517941
```

```
mad(x)
```

```
## [1] 2.9652
```

3. In the previous exercises we saw that the mean and median are very similar and so are the standard deviation and MAD. This is expected since the data is approximated by a normal distribution which has this property.

Now suppose that suppose Galton made a mistake when entering the first value, forgetting to use the decimal point. You can imitate this error by typing:

```
library(HistData)
data(Galton)
x <- Galton$child
x_with_error <- x
x_with_error[1] <- x_with_error[1]*10
```

The data now has an outlier that the normal approximation does not account for. Let's see how this affects the average.

```
x <- Galton$child
x_with_error <- x
x_with_error[1] <- x_with_error[1]*10
gem <- mean(x)
gem_error <- mean(x_with_error)
gem_error - gem
```

```
## [1] 0.5983836
```

4. In the previous exercise we saw how a simple mistake in 1 out of over 900 observations can result in the average of our data increasing more than half an inch, which is a large difference in practical terms.

Now let's explore the effect this outlier has on the standard deviation.

```
x_with_error <- x
x_with_error[1] <- x_with_error[1]*10
sd(x_with_error)- sd(x)
```

```
## [1] 15.6746
```

5. In the previous exercises we saw how one mistake can have a substantial effect on the average and the standard deviation.

Now we are going to see how the median and MAD are much more resistant to outliers. For this reason we say that they are *robust* summaries.

```
x_with_error <- x
x_with_error[1] <- x_with_error[1]*10
mediaan <- median(x)
mediaan_error <- median(x_with_error)
mediaan_error - mediaan
```

```
## [1] 0
```

6. We saw that the median barely changes. Now let's see how the MAD is affected.

We saw that the median barely changes. Now let's see how the MAD is affected.

```
x_with_error <- x
x_with_error[1] <- x_with_error[1]*10
mad_normal <- mad(x)
mad_error <- mad(x_with_error)
mad_error - mad_normal
```

```
## [1] 0
```

7. How could you use exploratory data analysis to detect that an error was made?

- ☐ A. Since it is only one value out of many, we will not be able to detect this.
- ☐ B. We would see an obvious shift in the distribution.
- ☒ C. A boxplot, histogram, or qq-plot would reveal a clear outlier.
- ☐ D. A scatter plot would show high levels of measurement error.

8. We have seen how the average can be affected by outliers.

But how large can this effect get? This of course depends on the size of the outlier and the size of the dataset.

To see how outliers can affect the average of a dataset, let's write a simple function that takes the size of the outlier as input and returns the average.

```
x <- Galton$child
error_avg <- function(k){
x[1] = k
mean(x)
}
error_avg(10000)
```

```
## [1] 78.79784
```

```r
error_avg(-10000)
```

```
## [1] 57.24612
```

Section 2 Overview

In Section 2, you will learn how to create data visualizations in R using ggplot2.

After completing Section 2, you will:

- be able to use ggplot2 to create data visualizations in R.
- be able to explain what the data component of a graph is.
- be able to identify the geometry component of a graph and know when to use which type of geometry.
- be able to explain what the aesthetic mapping component of a graph is.
- be able to understand the scale component of a graph and select an appropriate scale component to use.

Note that it can be hard to memorize all of the functions and arguments used by ggplot2, so we recommend that you have a cheat sheet handy to help you remember the necessary commands.

ggplot

The textbook for this section is available here

Key points

- Throughout the series, we will create plots with the **ggplot2** package. ggplot2 is part of the tidyverse, which you can load with `library(tidyverse)`.
- Note that you can also load ggplot2 alone using the command `library(ggplot2)`, instead of loading the entire tidyverse.
- ggplot2 uses a *grammar of graphics* to break plots into building blocks that have intuitive syntax, making it easy to create relatively complex and aesthetically pleasing plots with relatively simple and readable code.
- ggplot2 is designed to work exclusively with tidy data (rows are observations and columns are variables).

Graph Components

The textbook for this section is available here

Key points

- Plots in ggplot2 consist of 3 main components:
  - Data: The dataset being summarized
  - Geometry: The type of plot (scatterplot, boxplot, barplot, histogram, qqplot, smooth density, etc.)
  - Aesthetic mapping: Variables mapped to visual cues, such as x-axis and y-axis values and color
- There are additional components:
  - Scale
  - Labels, Title, Legend
  - Theme/Style

Creating a New Plot

The textbook for this section is available here

Key points

- You can associate a dataset x with a ggplot object with any of the 3 commands:
  - `ggplot(data = x)`
  - `ggplot(x)`
  - `x %>% ggplot()`
- You can assign a ggplot object to a variable. If the object is not assigned to a variable, it will automatically be displayed.
- You can display a ggplot object assigned to a variable by printing that variable.

Code

```
ggplot(data = murders)

murders %>% ggplot()
```

```
p <- ggplot(data = murders)
class(p)
```

```
## [1] "gg"     "ggplot"
```

```
print(p)    # this is equivalent to simply typing p
```

The functions above render a plot, in this case a blank slate since no geometry has been defined. The only style choice we see is a grey background.

Layers

The textbook for this section is available:

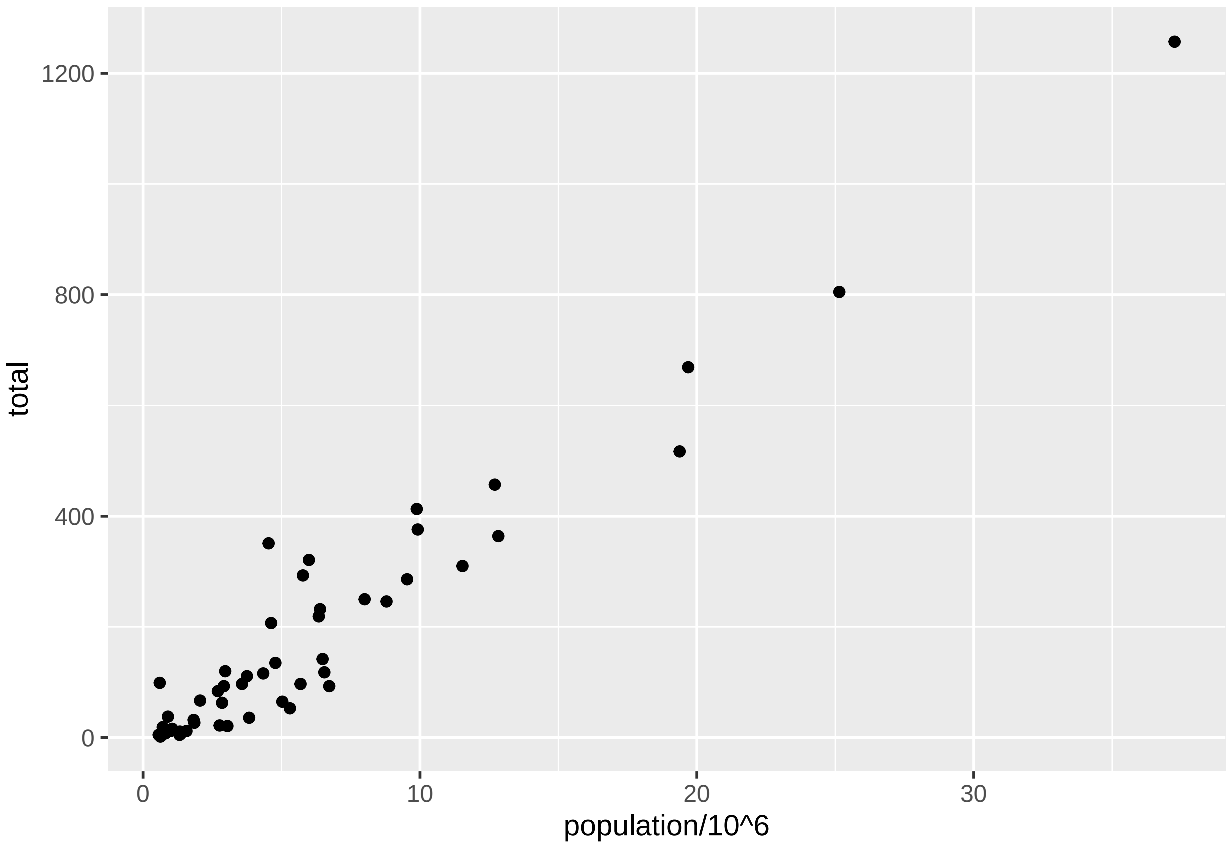
- Geometries
- Aesthetic mappings
- Layers

Key points

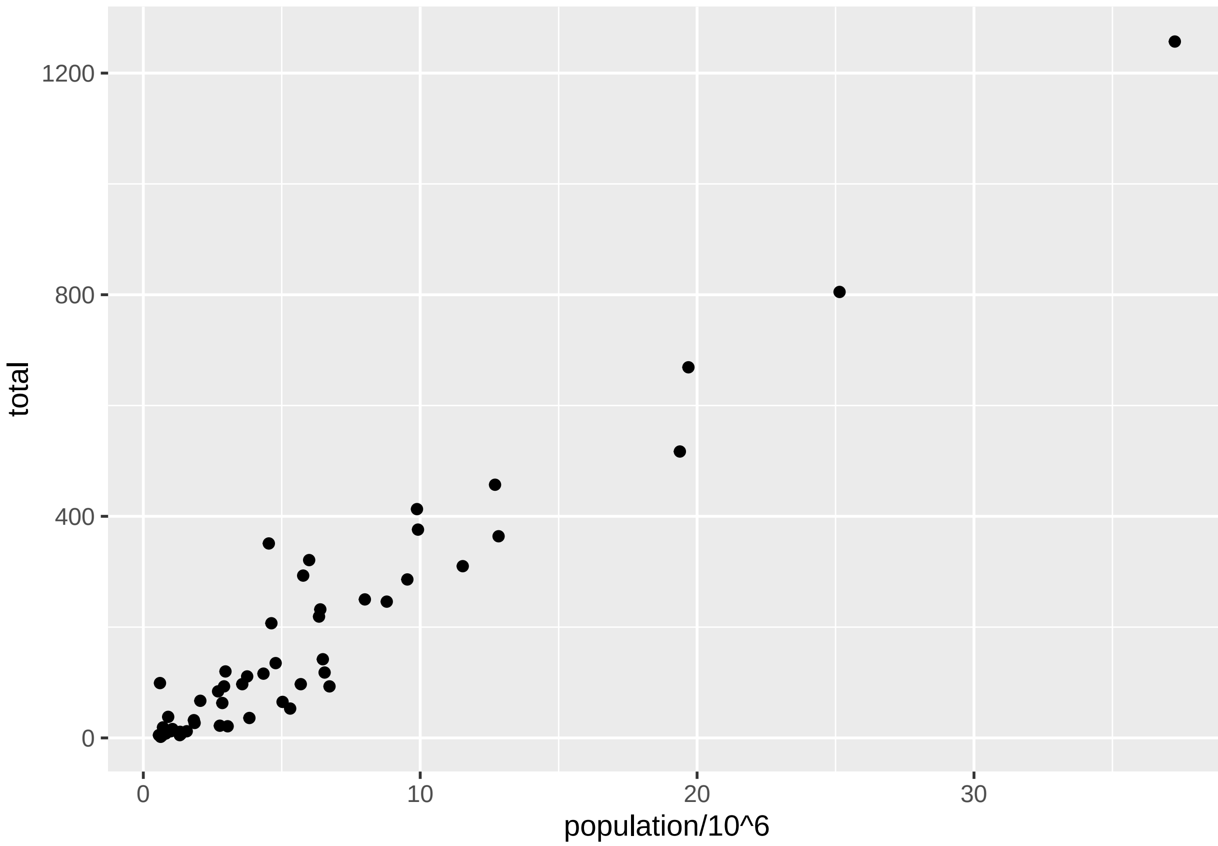
- In ggplot2, graphs are created by adding *layers* to the ggplot object: `DATA %>% ggplot() + LAYER_1 + LAYER_2 + ... + LAYER_N`
- The *geometry layer* defines the plot type and takes the format `geom_X` where X is the plot type.
- *Aesthetic mappings* describe how properties of the data connect with features of the graph (axis position, color, size, etc.) Define aesthetic mappings with the `aes` function.
- aes uses variable names from the object component (for example, `total` rather than `murders$total`).
- `geom_point` creates a scatterplot and requires x and y aesthetic mappings.
- `geom_text` and `geom_label` add text to a scatterplot and require x, y, and label aesthetic mappings.
- To determine which aesthetic mappings are required for a geometry, read the help file for that geometry.
- You can add layers with different aesthetic mappings to the same graph.

Code: Adding layers to a plot

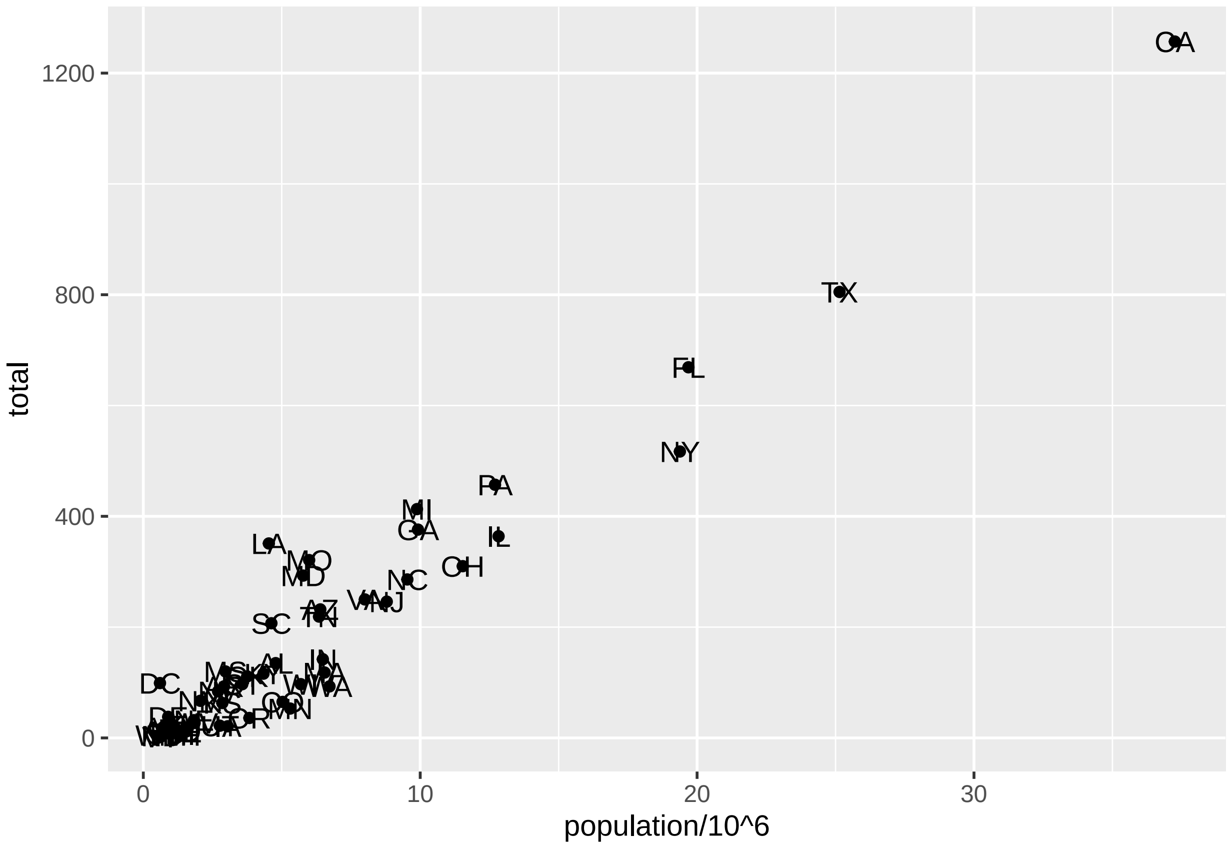
```
murders %>% ggplot() +
    geom_point(aes(x = population/10^6, y = total))
```




```
# add points layer to predefined ggplot object
p <- ggplot(data = murders)
p + geom_point(aes(population/10^6, total))
```




```r
# add text layer to scatterplot
p + geom_point(aes(population/10^6, total)) +
    geom_text(aes(population/10^6, total, label = abb))
```



*Code: Example of **aes** behavior*

```
# no error from this call
p_test <- p + geom_text(aes(population/10^6, total, label = abb))
```

```
# error - "abb" is not a globally defined variable and cannot be found outside of aes
p_test <- p + geom_text(aes(population/10^6, total), label = abb)
```

Tinkering

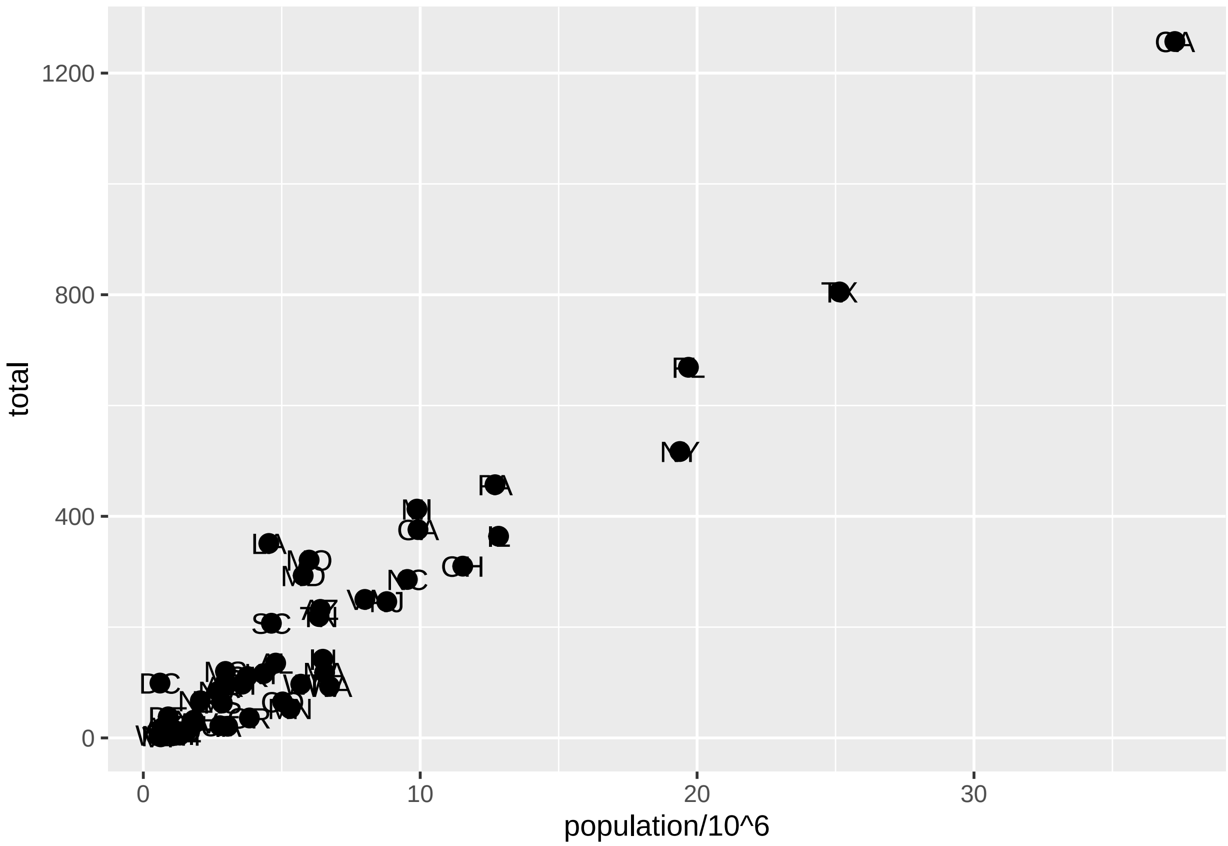
The textbook for this section is available here and here

Key points

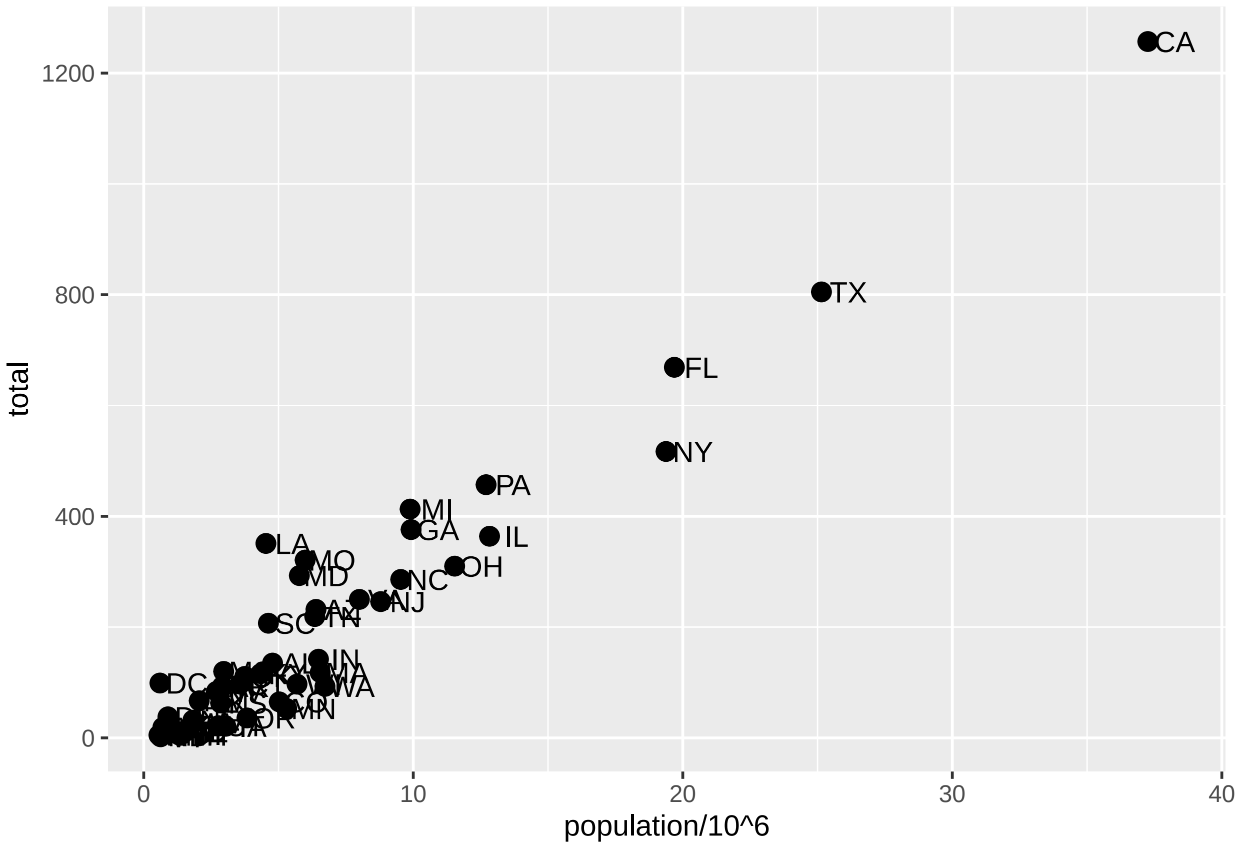
- You can modify arguments to geometry functions other than aes and the data. Additional arguments can be found in the documentation for each geometry.
- These arguments are not aesthetic mappings: they affect all data points the same way.
- *Global aesthetic mappings* apply to all geometries and can be defined when you initially call `ggplot`. All the geometries added as layers will default to this mapping. Local aesthetic mappings add additional information or override the default mappings.

Code

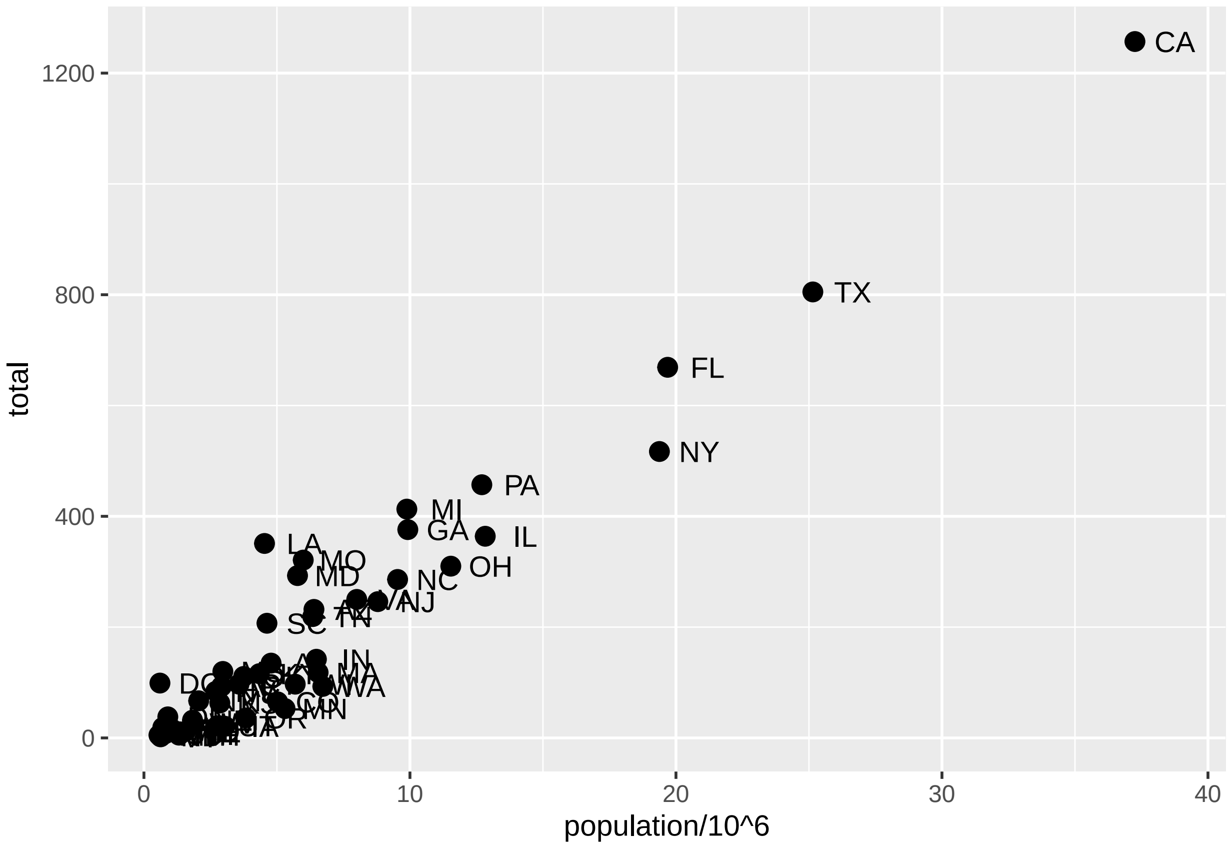
```
# change the size of the points
p + geom_point(aes(population/10^6, total), size = 3) +
    geom_text(aes(population/10^6, total, label = abb))
```



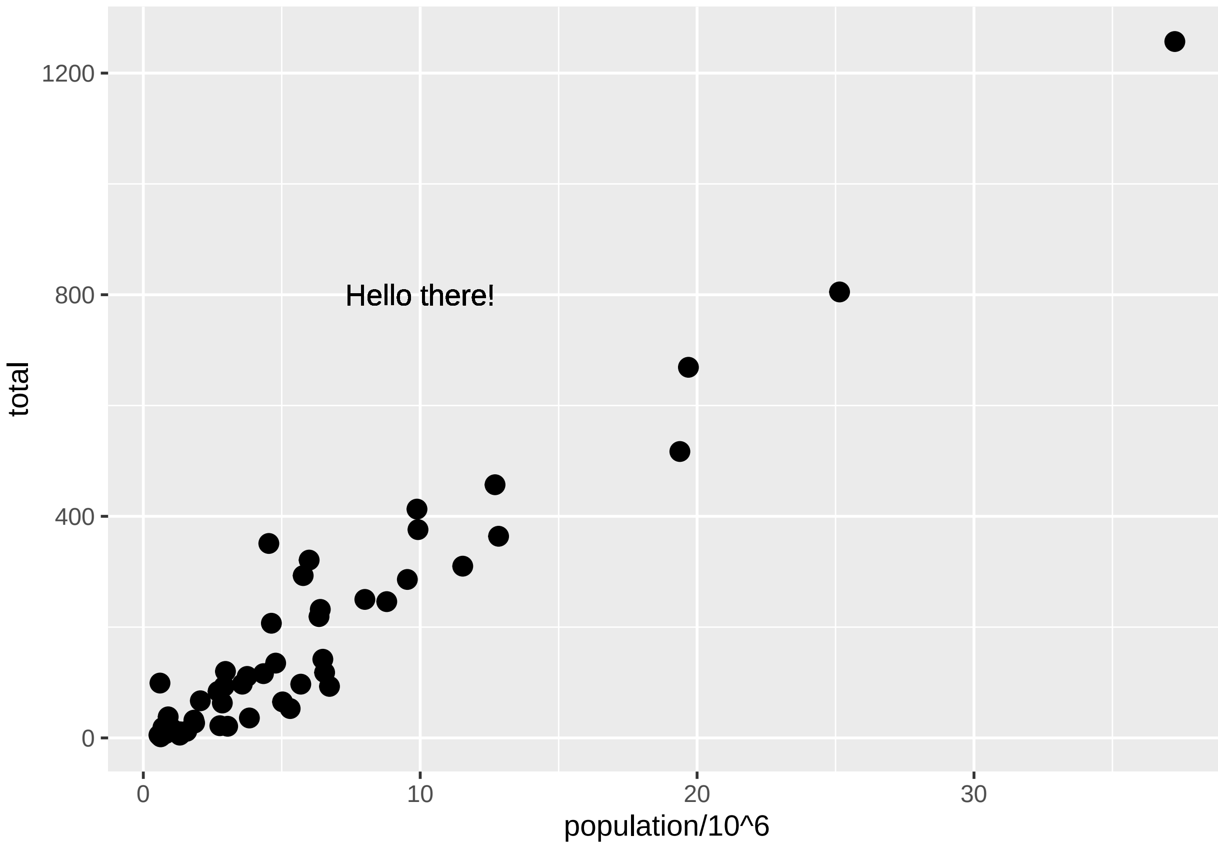

```
# move text labels slightly to the right
p + geom_point(aes(population/10^6, total), size = 3) +
    geom_text(aes(population/10^6, total, label = abb), nudge_x = 1)
```




```
# simplify code by adding global aesthetic
p <- murders %>% ggplot(aes(population/10^6, total, label = abb))
p + geom_point(size = 3) +
    geom_text(nudge_x = 1.5)
```




```
# local aesthetics override global aesthetics
p + geom_point(size = 3) +
    geom_text(aes(x = 10, y = 800, label = "Hello there!"))
```



Scales, Labels, and Colors

The textbook for this section is available:

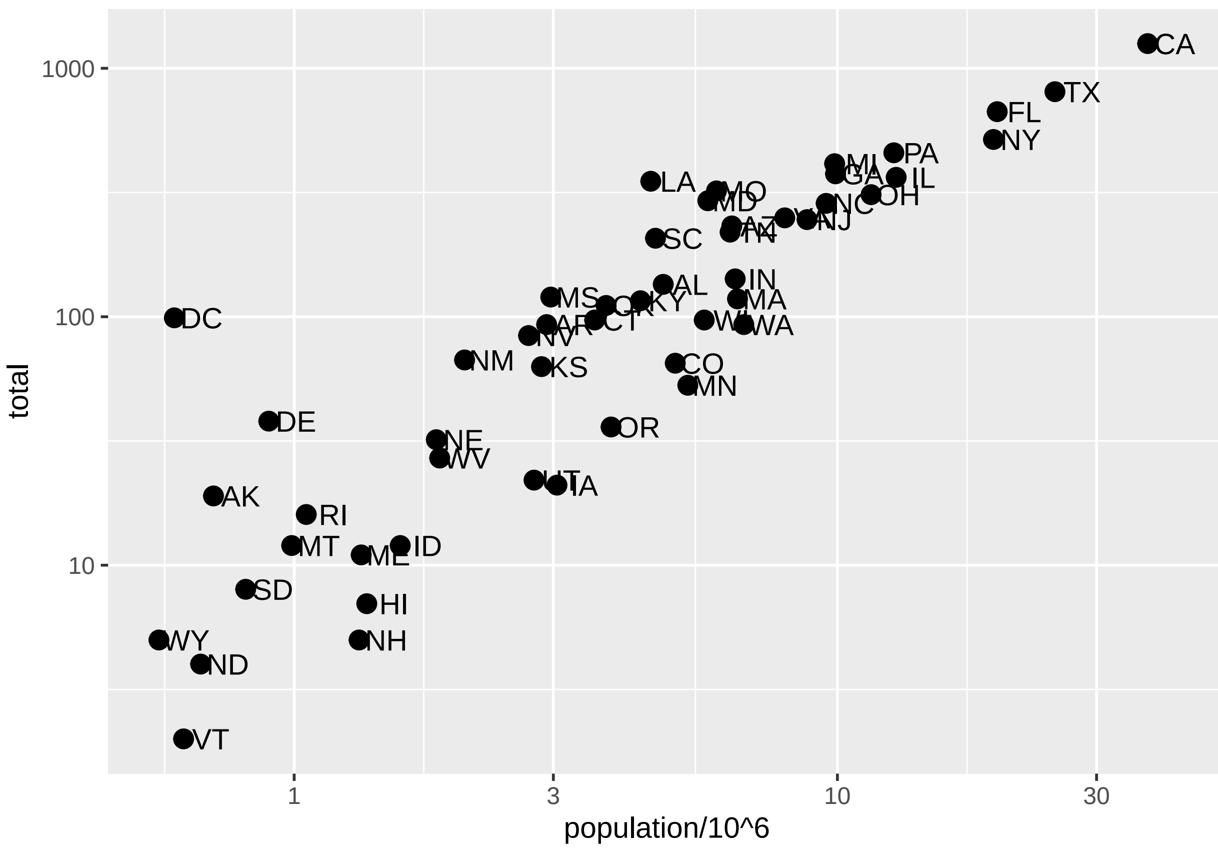
- Scales
- Labels and titles
- Categories as colors
- Annotation, shapes and adjustments

Key points

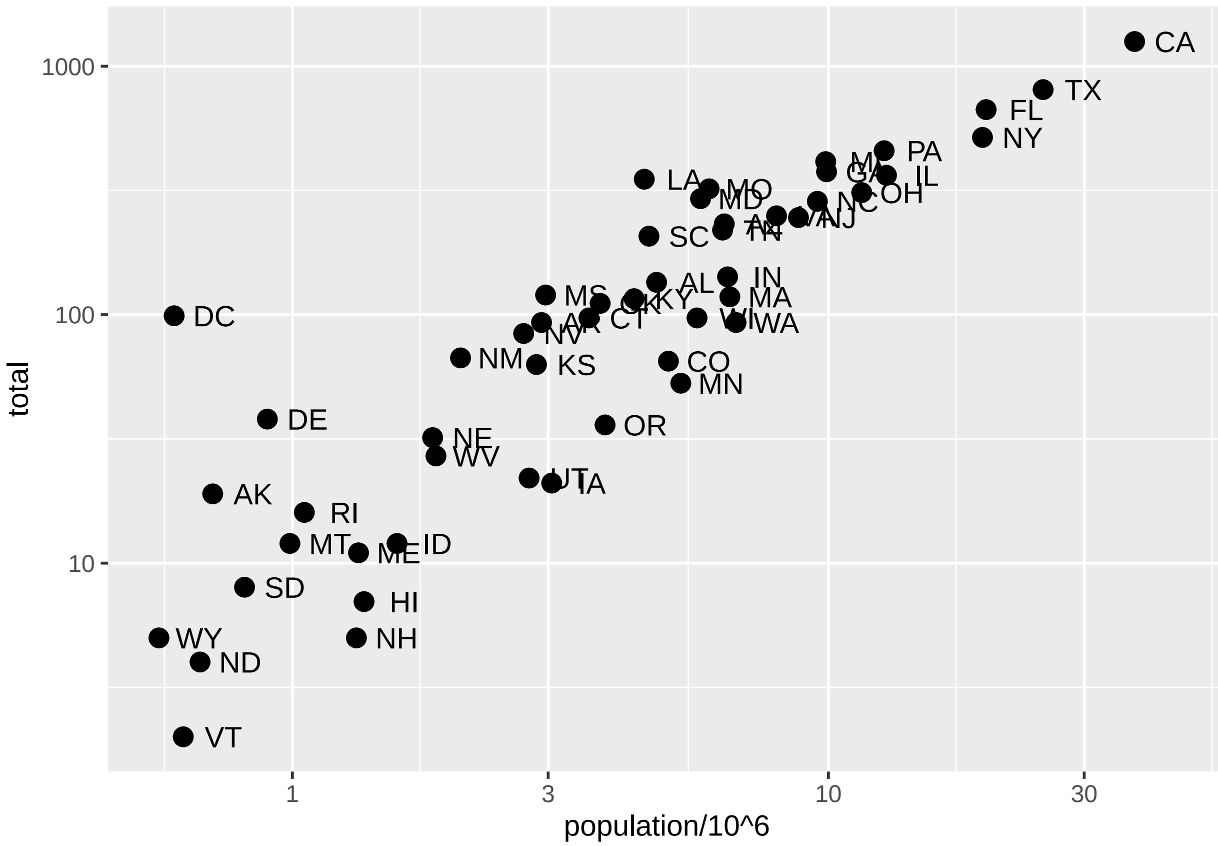
- Convert the x-axis to log scale with `scale_x_continuous(trans = "log10")` or `scale_x_log10`. Similar functions exist for the y-axis.
- Add axis titles with `xlab` and `ylab` functions. Add a plot title with the `ggtitle` function.
- Add a color mapping that colors points by a variable by defining the `col` argument within `aes`. To color all points the same way, define `col` outside of `aes`.
- Add a line with the `geom_abline` geometry. `geom_abline` takes arguments `slope` (default = 1) and `intercept` (default = 0). Change the color with `col` or `color` and line type with `lty`.
- Placing the line layer after the point layer will overlay the line on top of the points. To overlay points on the line, place the line layer before the point layer.
- There are many additional ways to tweak your graph that can be found in the ggplot2 documentation, cheat sheet, or on the internet. For example, you can change the legend title with `scale_color_discrete`.

Code: Log-scale the x- and y-axis

```r
# define p
p <- murders %>% ggplot(aes(population/10^6, total, label = abb))

# log base 10 scale the x-axis and y-axis
p + geom_point(size = 3) +
    geom_text(nudge_x = 0.05) +
    scale_x_continuous(trans = "log10") +
    scale_y_continuous(trans = "log10")
```

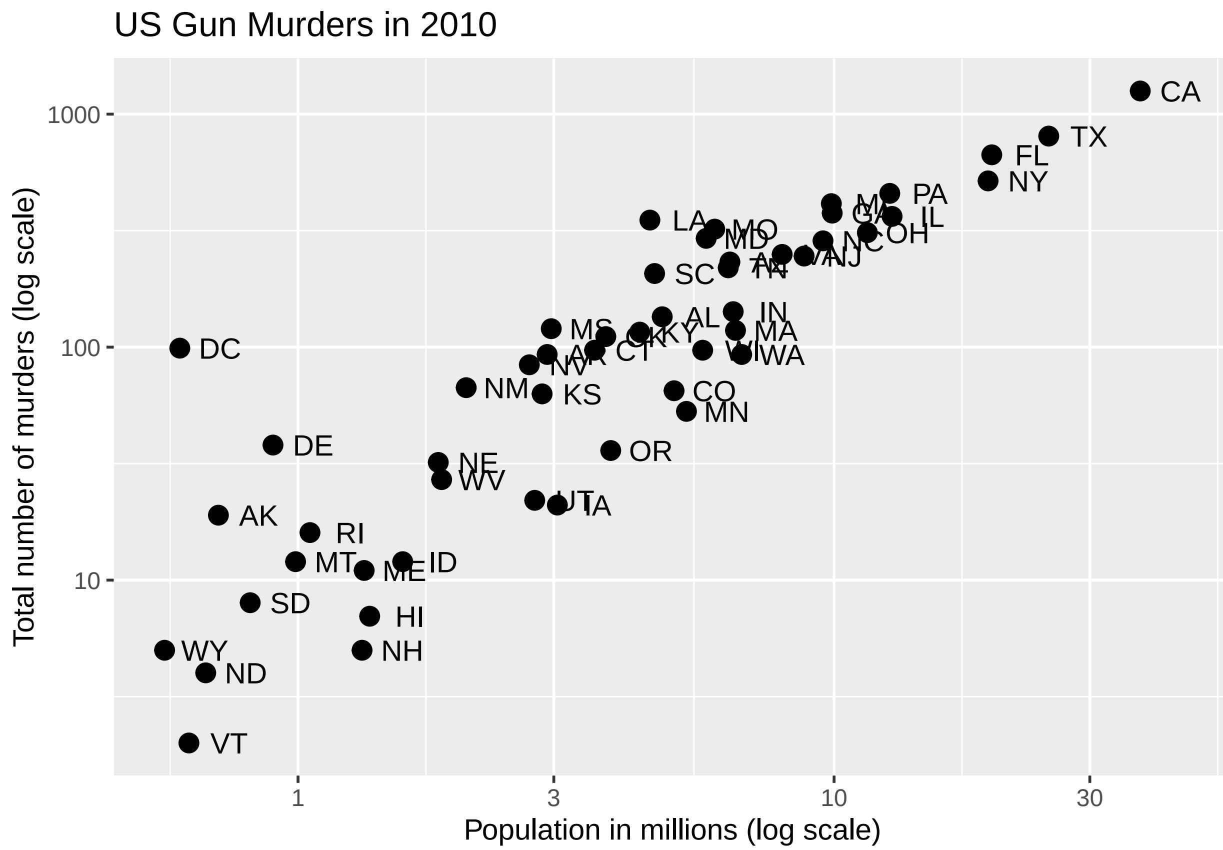



```r
# efficient log scaling of the axes
p + geom_point(size = 3) +
    geom_text(nudge_x = 0.075) +
    scale_x_log10() +
    scale_y_log10()
```



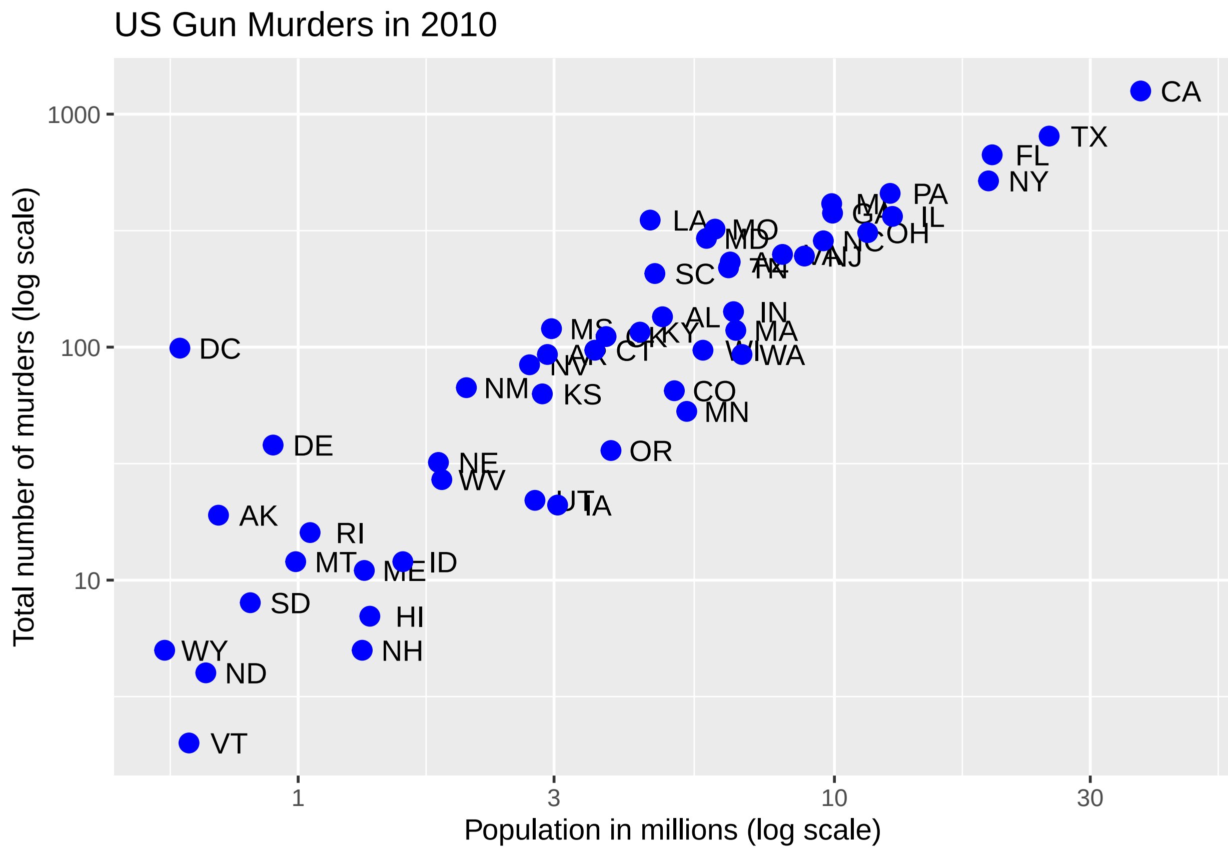
Code: Add labels and title

```
p + geom_point(size = 3) +
    geom_text(nudge_x = 0.075) +
    scale_x_log10() +
    scale_y_log10() +
    xlab("Population in millions (log scale)") +
    ylab("Total number of murders (log scale)") +
    ggtitle("US Gun Murders in 2010")
```

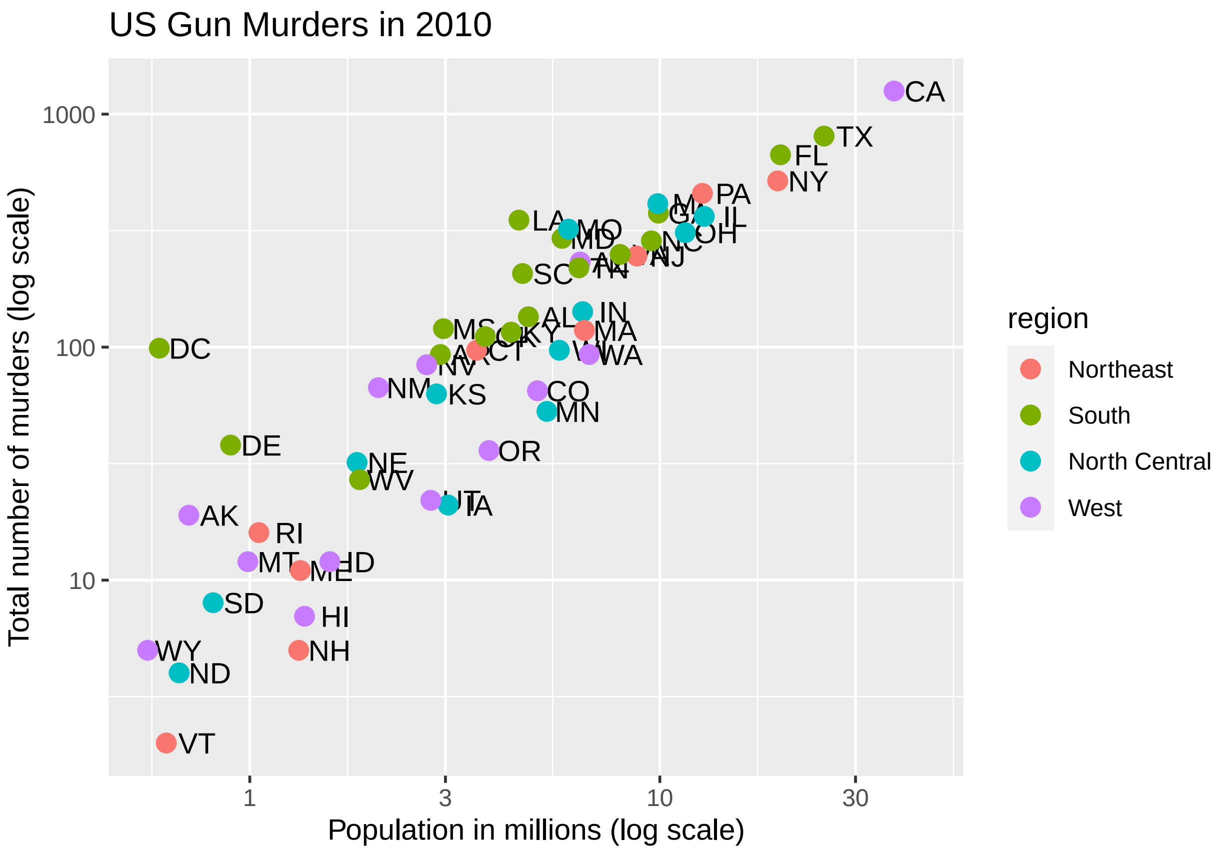


Code: Change color of the points

```
# redefine p to be everything except the points layer
p <- murders %>%
    ggplot(aes(population/10^6, total, label = abb)) +
    geom_text(nudge_x = 0.075) +
    scale_x_log10() +
    scale_y_log10() +
    xlab("Population in millions (log scale)") +
    ylab("Total number of murders (log scale)") +
    ggtitle("US Gun Murders in 2010")

# make all points blue
p + geom_point(size = 3, color = "blue")
```

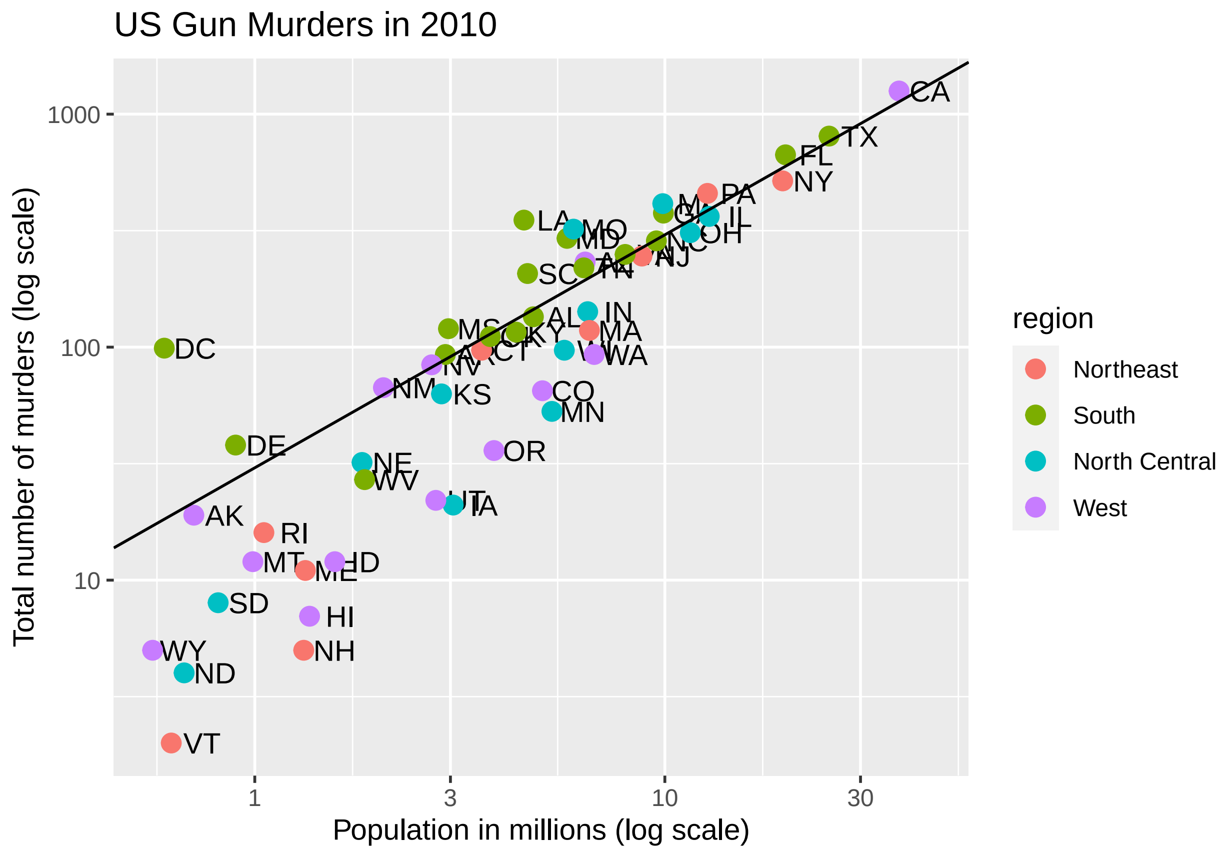



```
# color points by region
p + geom_point(aes(col = region), size = 3)
```

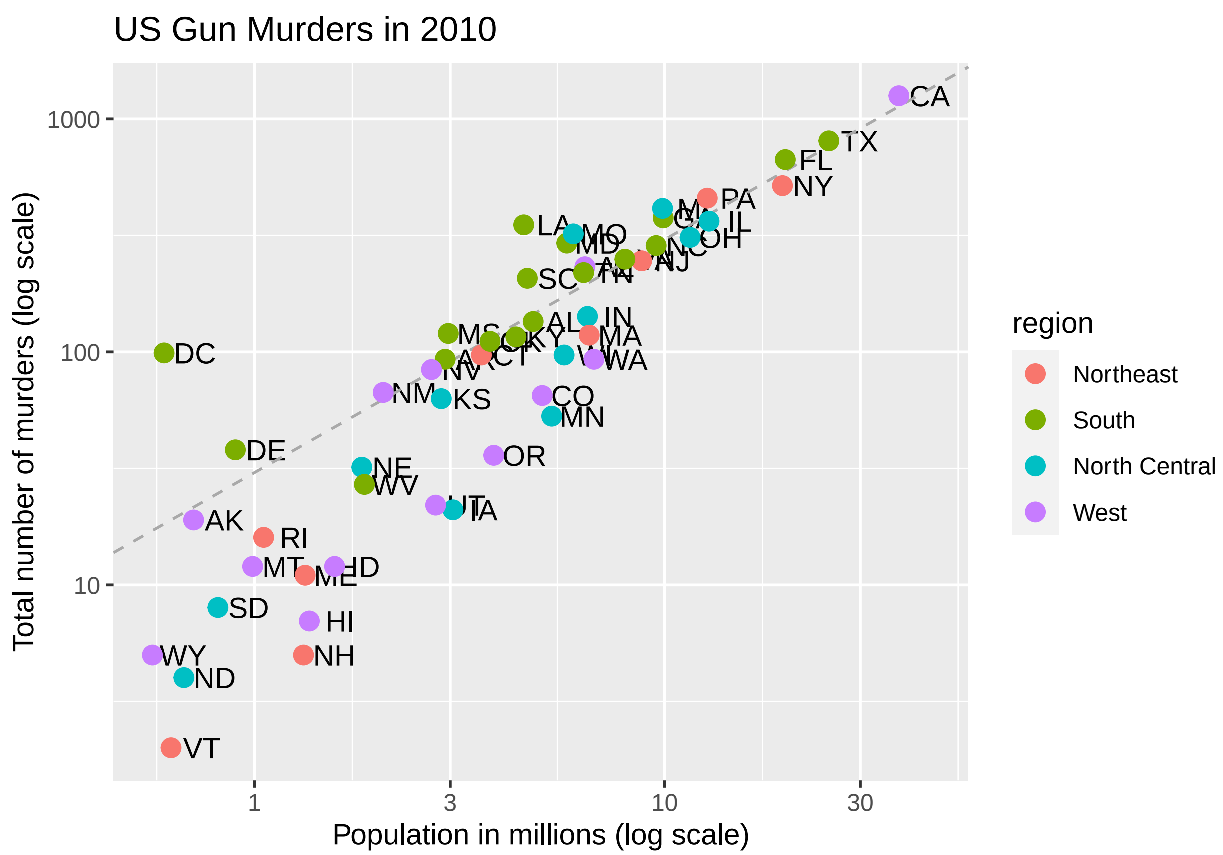


Code: Add a line with average murder rate

```
# define average murder rate
r <- murders %>%
    summarize(rate = sum(total) / sum(population) * 10^6) %>%
    pull(rate)

# basic line with average murder rate for the country
p + geom_point(aes(col = region), size = 3) +
    geom_abline(intercept = log10(r))     # slope is default of 1
```




```r
# change line to dashed and dark grey, line under points
p +
    geom_abline(intercept = log10(r), lty = 2, color = "darkgrey") +
    geom_point(aes(col = region), size = 3)
```



Code: Change legend title

```
p <- p + scale_color_discrete(name = "Region")    # capitalize legend title
```

Add-on Packages

The textbook for this section is available here and here

Key points

- The style of a ggplot graph can be changed using the `theme` function.
- The **ggthemes** package adds additional themes.
- The **ggrepel** package includes a geometry that repels text labels, ensuring they do not overlap with each other: `geom_text_repel`.

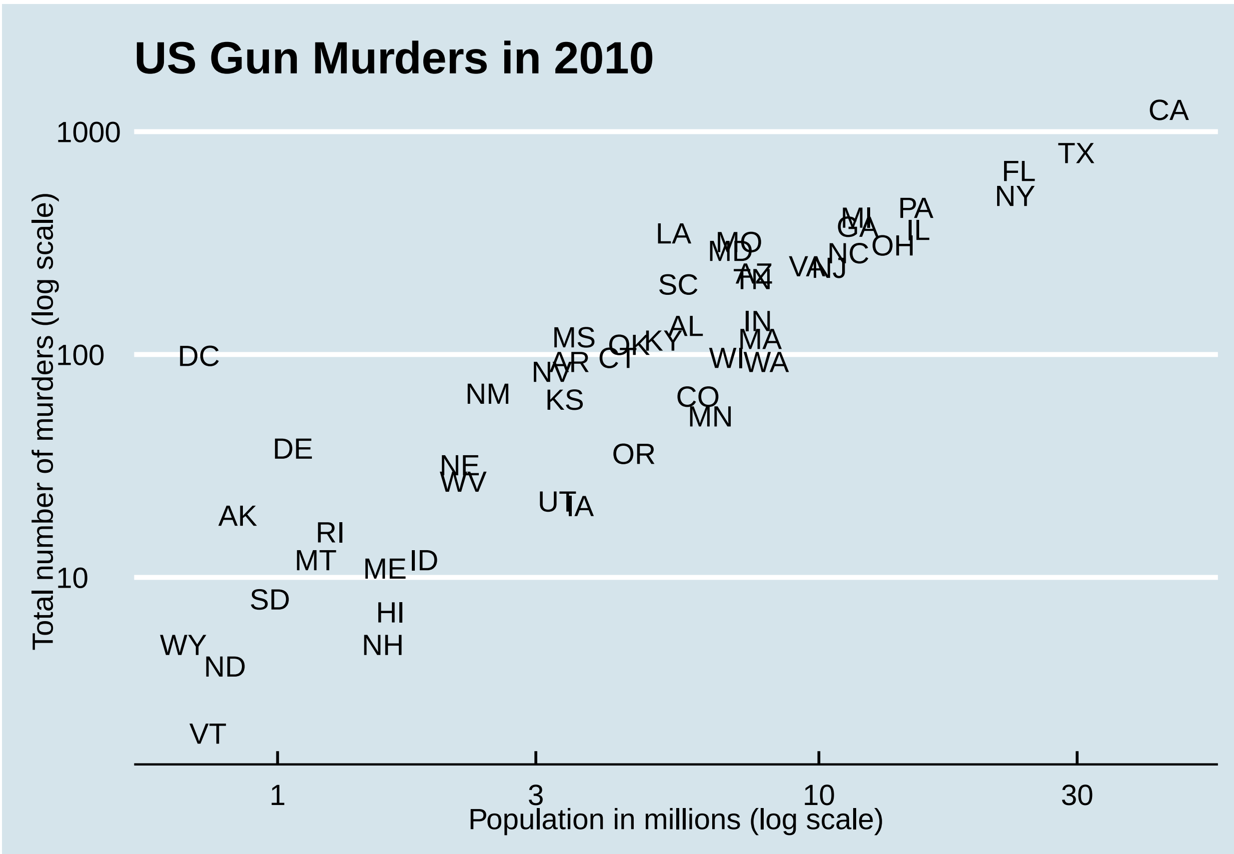
Code: Adding themes

```
if(!require(ggthemes)) install.packages("ggthemes")
```

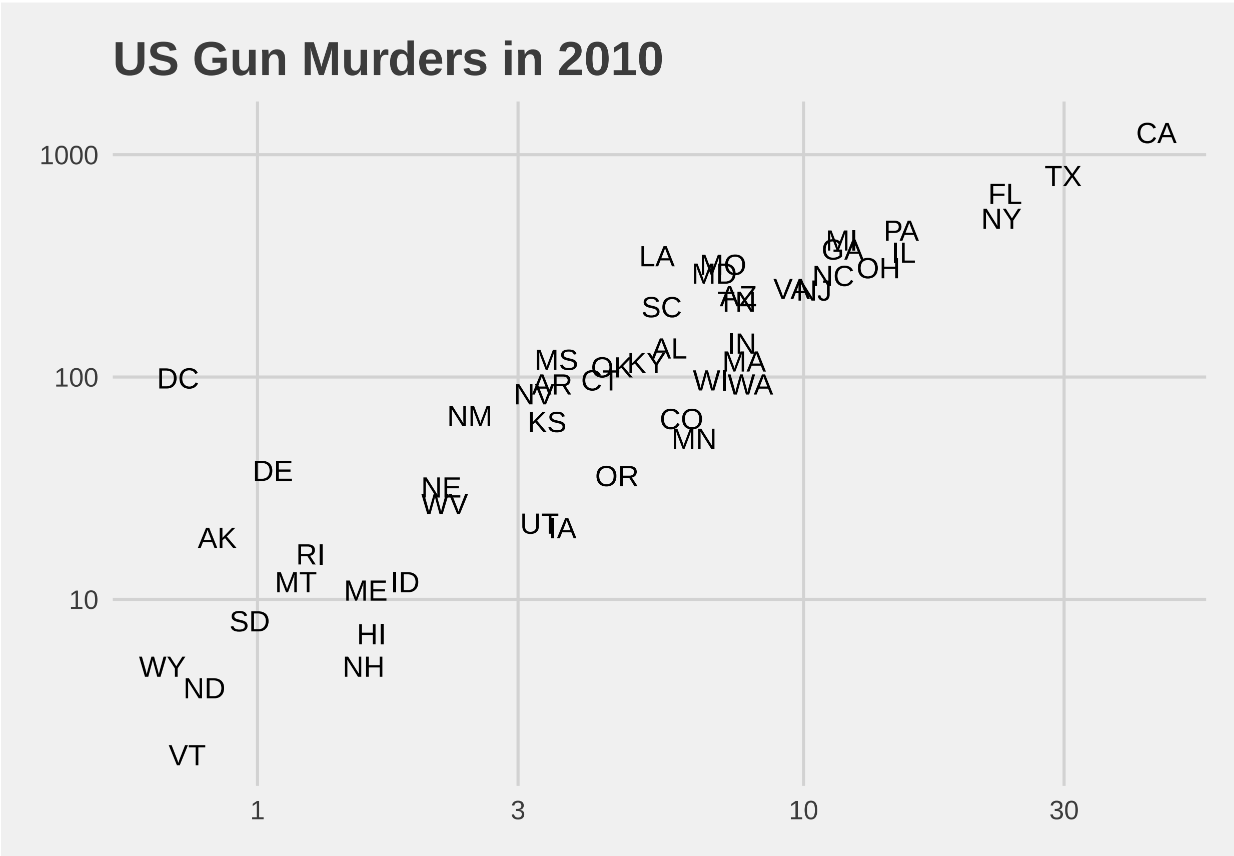
```
## Loading required package: ggthemes
```

```
## Warning: package 'ggthemes' was built under R version 4.0.2
```

```r
# theme used for graphs in the textbook and course
ds_theme_set()

# themes from ggthemes
library(ggthemes)
p + theme_economist()     # style of the Economist magazine
```




```r
p + theme_fivethirtyeight()     # style of the FiveThirtyEight website
```



Code: Putting it all together to assemble the plot

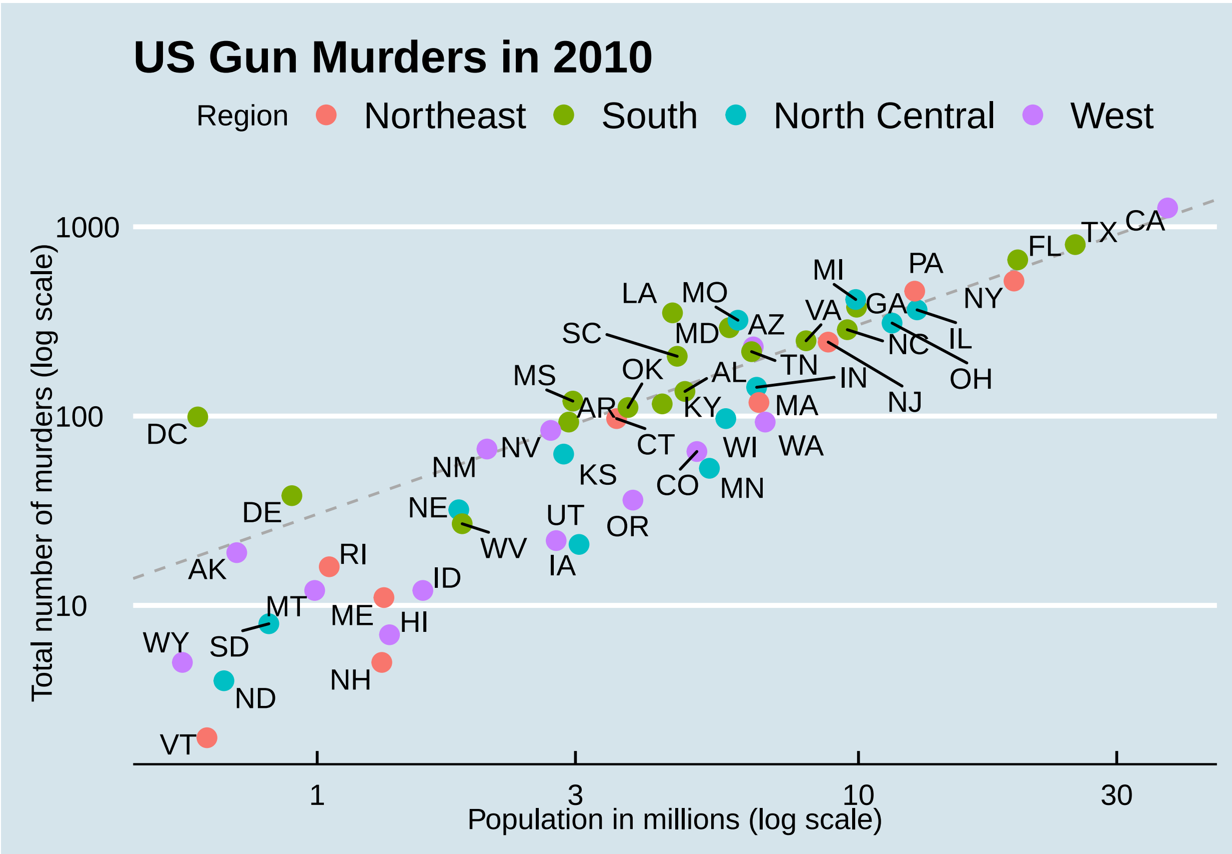
```
if(!require(ggrepel)) install.packages("ggrepel")
```

```
## Loading required package: ggrepel
```

```
## Warning: package 'ggrepel' was built under R version 4.0.2
```

```
# load libraries
library(ggrepel)

# define the intercept
r <- murders %>%
    summarize(rate = sum(total) / sum(population) * 10^6) %>%
    .$rate

# make the plot, combining all elements
murders %>%
    ggplot(aes(population/10^6, total, label = abb)) +
    geom_abline(intercept = log10(r), lty = 2, color = "darkgrey") +
    geom_point(aes(col = region), size = 3) +
    geom_text_repel() +
    scale_x_log10() +
    scale_y_log10() +
    xlab("Population in millions (log scale)") +
```

```r
  ylab("Total number of murders (log scale)") +
  ggtitle("US Gun Murders in 2010") +
  scale_color_discrete(name = "Region") +
  theme_economist()
```



Other Examples

The textbook for this section is available:

- Histograms
- Density plots
- QQ-plots
- Grids of plots

Key points

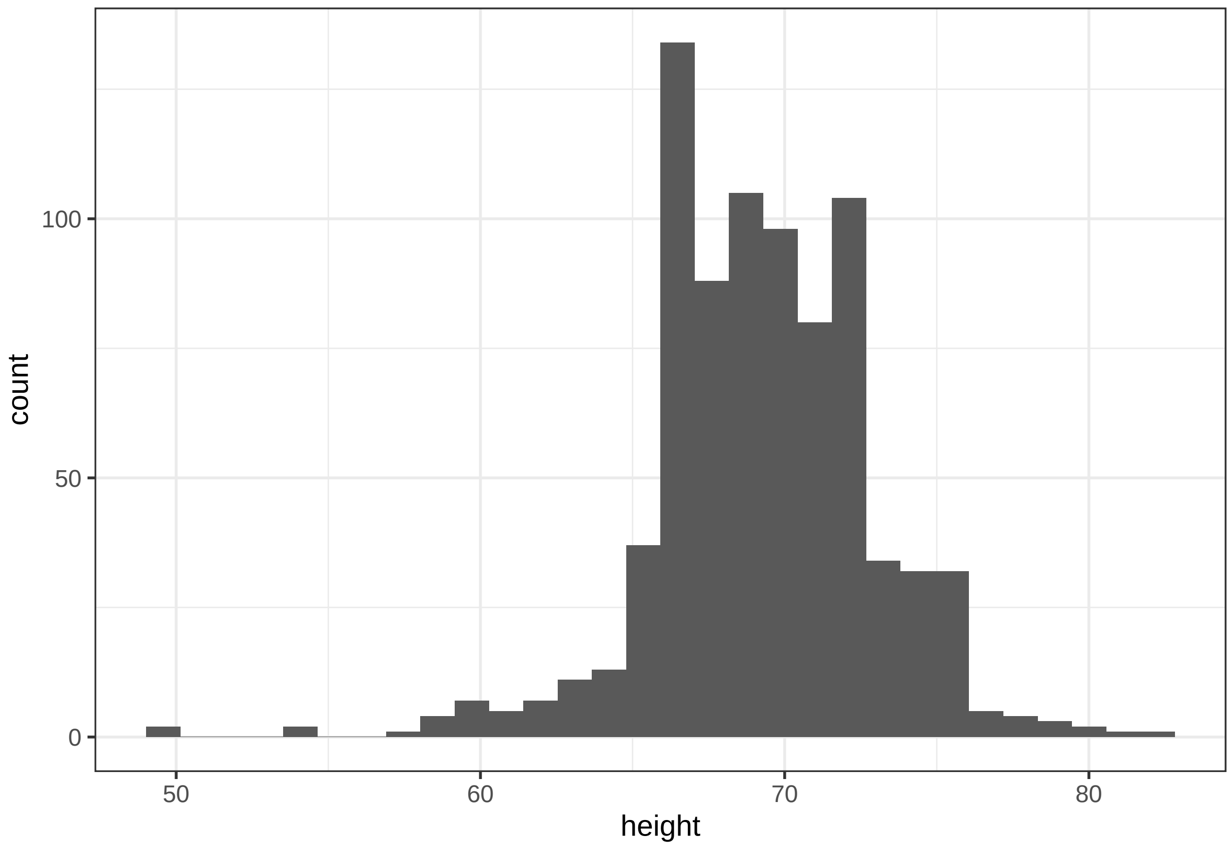
- `geom_histogram` creates a histogram. Use the `binwidth` argument to change the width of bins, the `fill` argument to change the bar fill color, and the `col` argument to change bar outline color.
- `geom_density` creates smooth density plots. Change the fill color of the plot with the `fill` argument.
- `geom_qq` creates a quantile-quantile plot. This geometry requires the `sample` argument. By default, the data are compared to a standard normal distribution with a mean of 0 and standard deviation of 1. This can be changed with the `dparams` argument, or the sample data can be scaled.

- Plots can be arranged adjacent to each other using the `grid.arrange` function from the `gridExtra` package. First, create the plots and save them to objects (p1, p2, ...). Then pass the plot objects to `grid.arrange`.

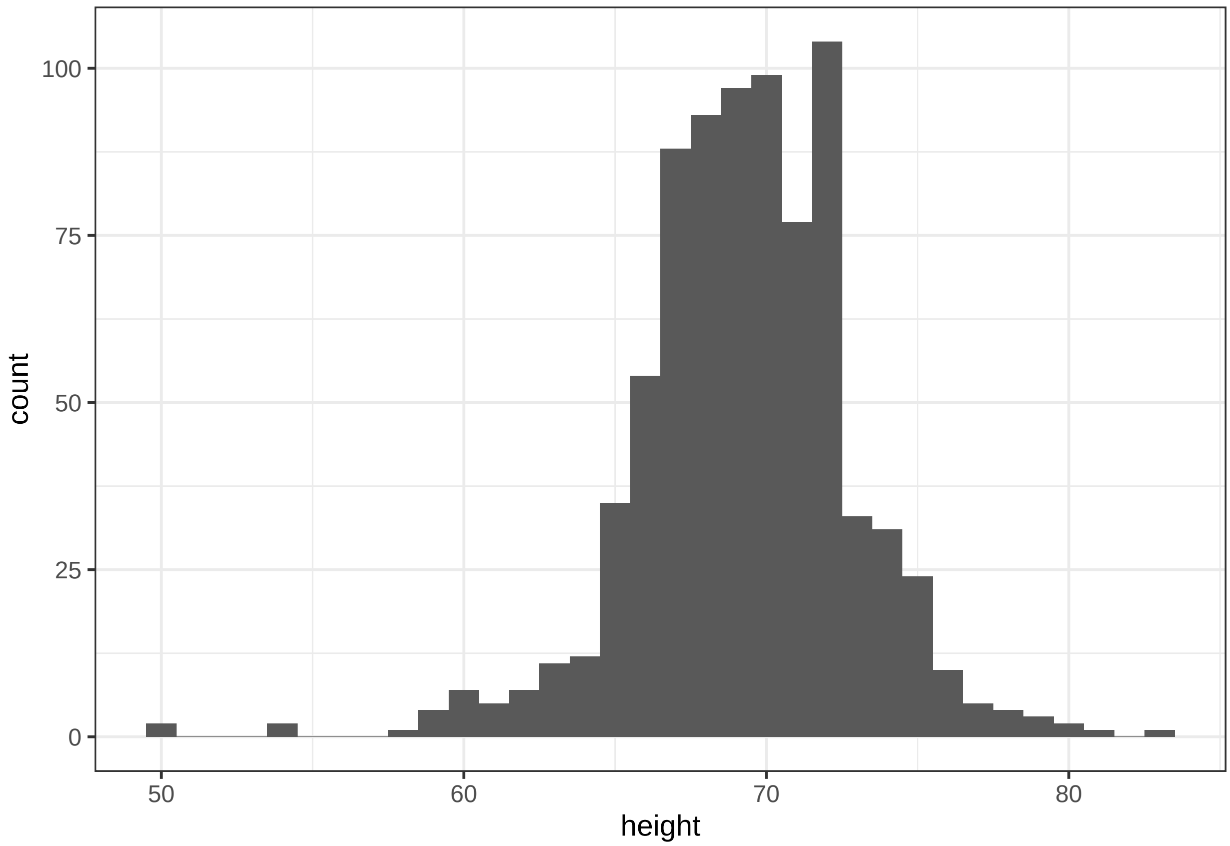
Code: Histograms in ggplot2

```
# define p
p <- heights %>%
    filter(sex == "Male") %>%
    ggplot(aes(x = height))

# basic histograms
p + geom_histogram()
```

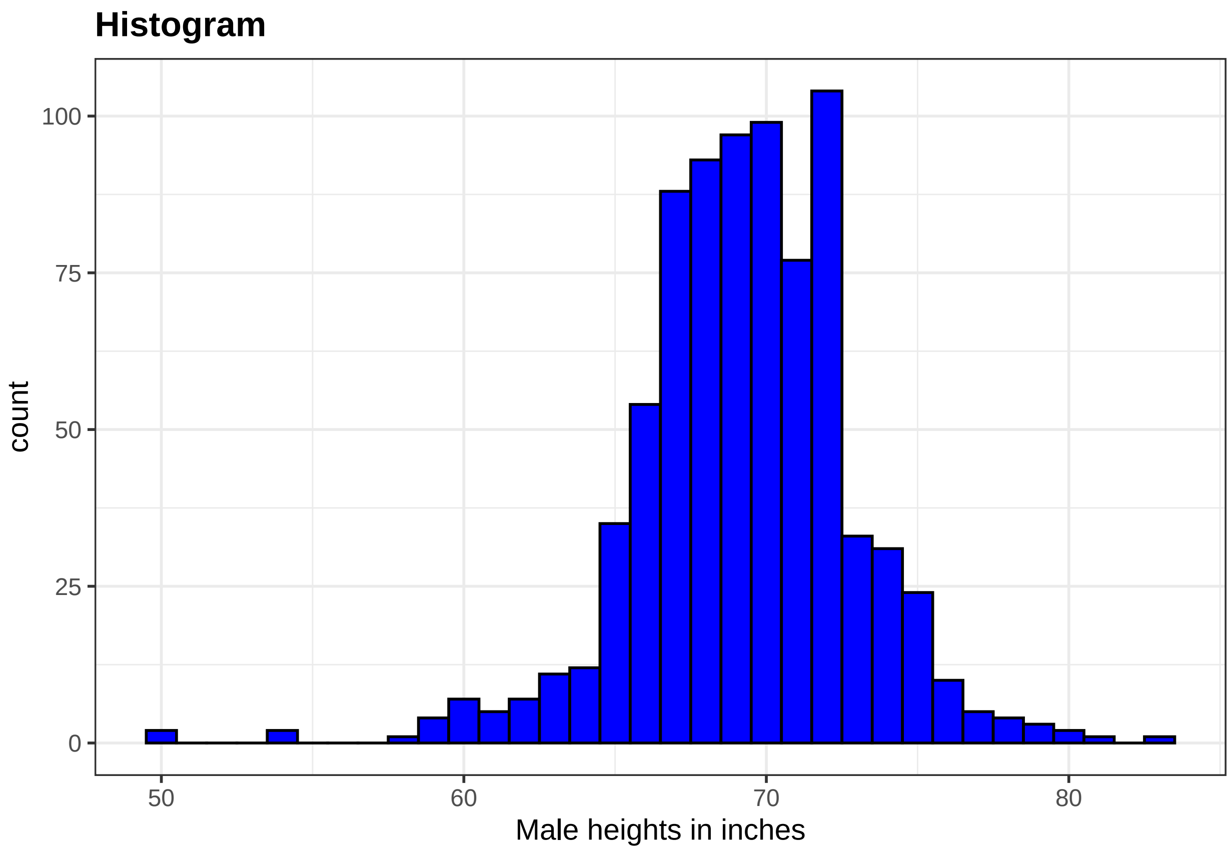
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
p + geom_histogram(binwidth = 1)
```

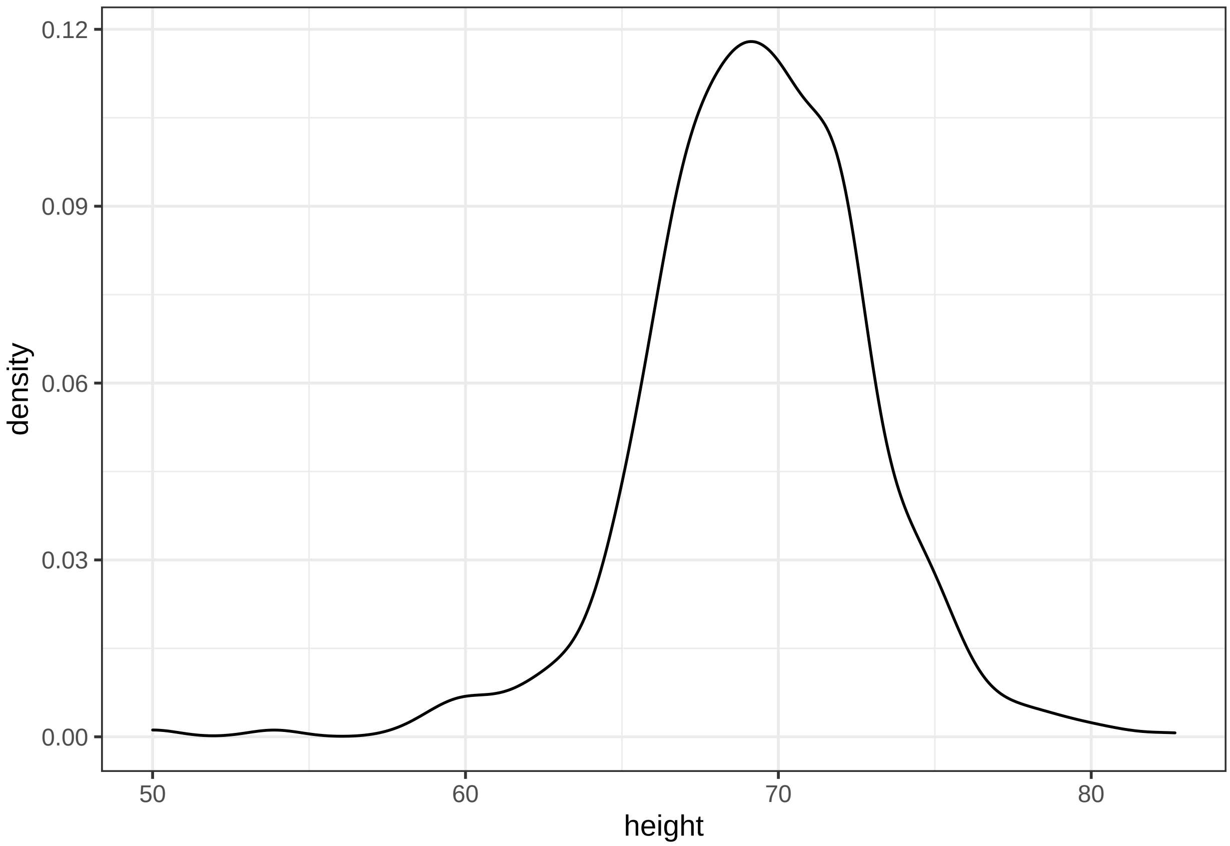



```r
# histogram with blue fill, black outline, labels and title
p + geom_histogram(binwidth = 1, fill = "blue", col = "black") +
    xlab("Male heights in inches") +
    ggtitle("Histogram")
```

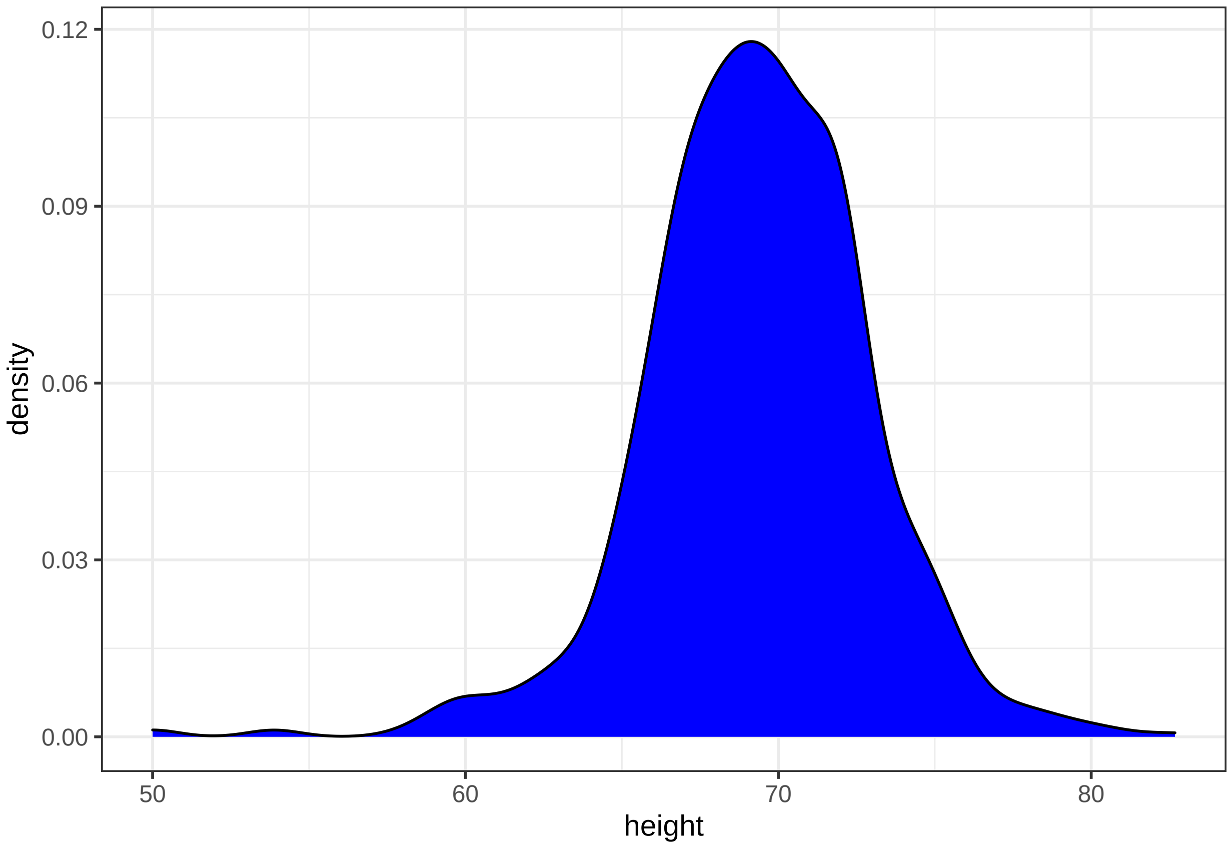


Code: Smooth density plots in ggplot2

```
p + geom_density()
```

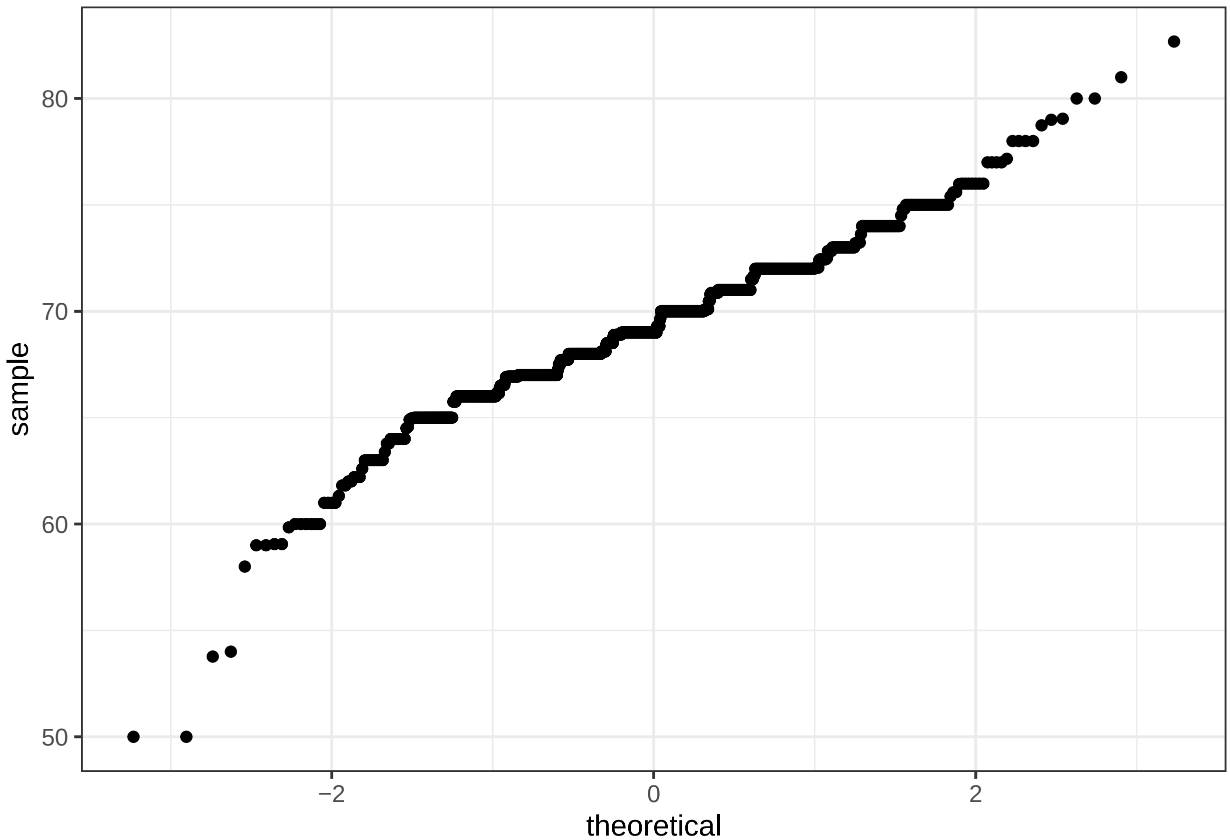



```
p + geom_density(fill = "blue")
```

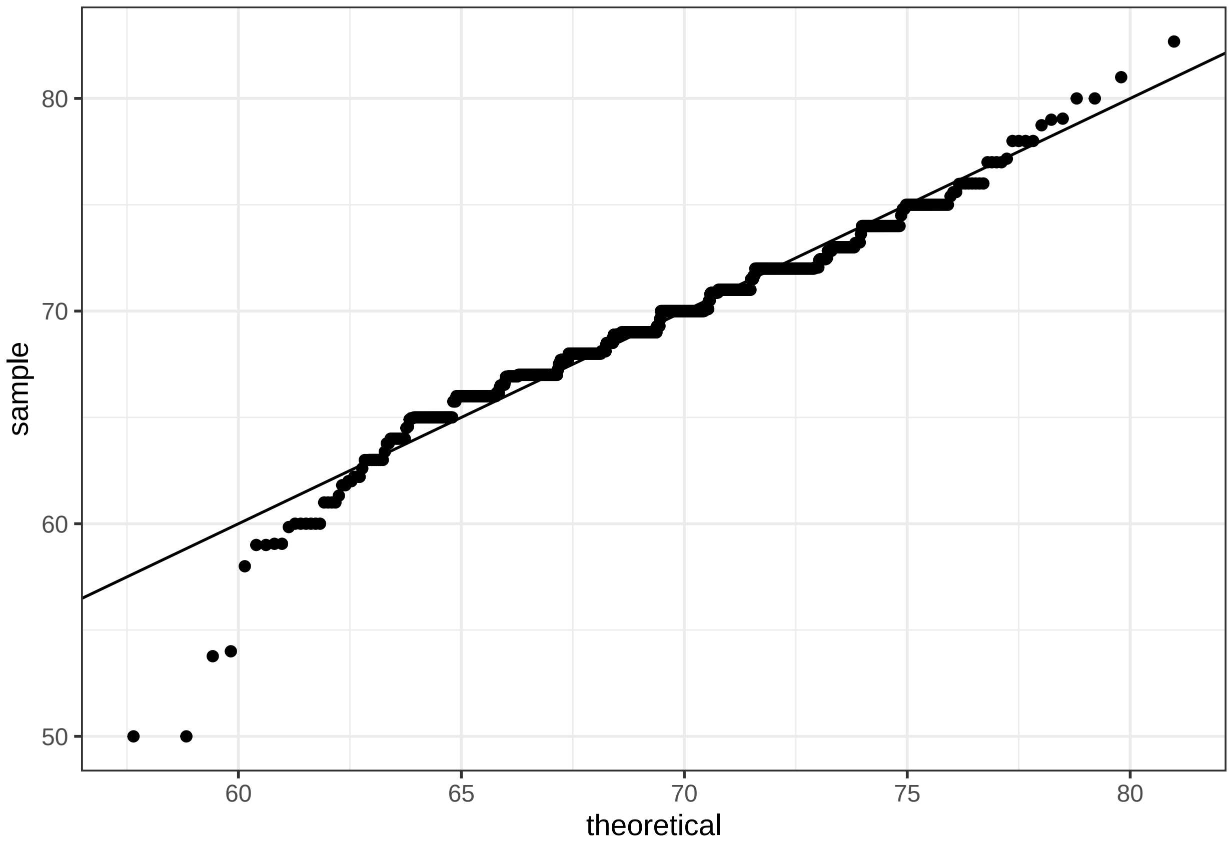


Code: Quantile-quantile plots in ggplot2

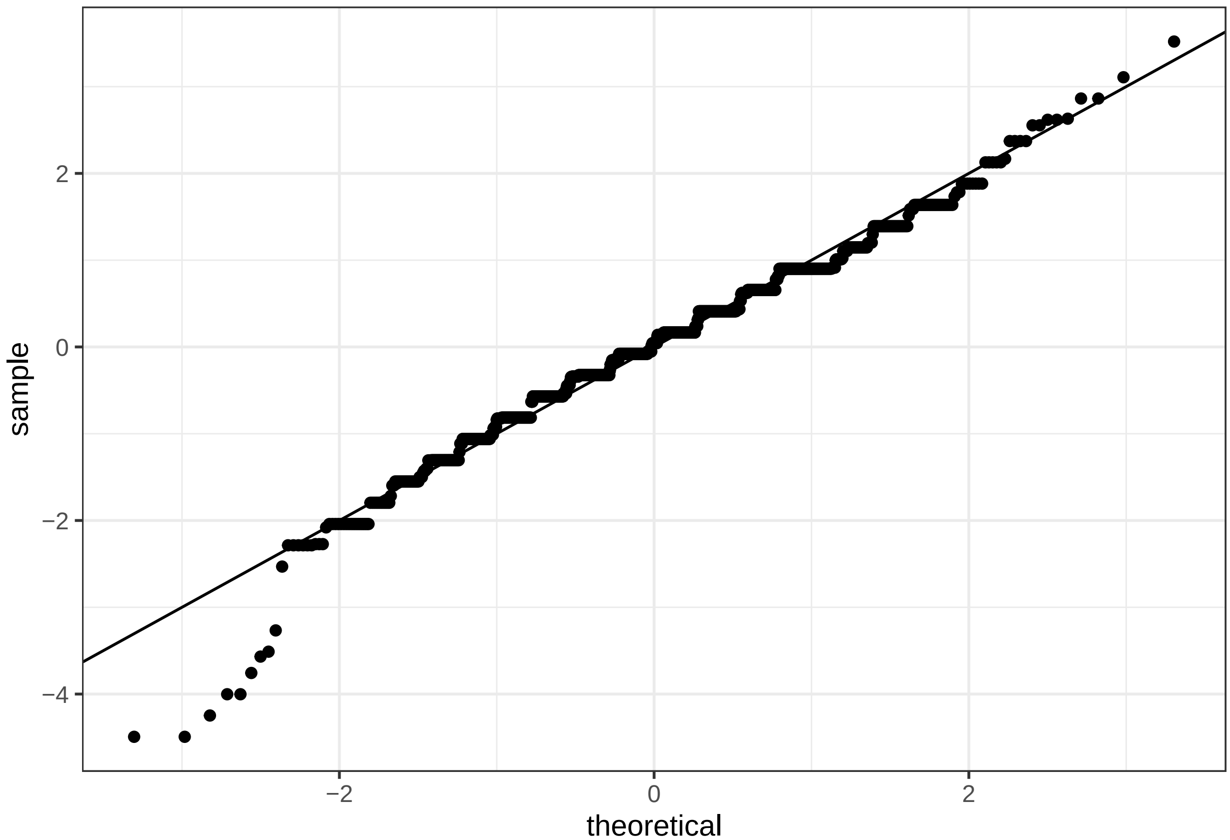
```
# basic QQ-plot
p <- heights %>% filter(sex == "Male") %>%
    ggplot(aes(sample = height))
p + geom_qq()
```




```r
# QQ-plot against a normal distribution with same mean/sd as data
params <- heights %>%
    filter(sex == "Male") %>%
    summarize(mean = mean(height), sd = sd(height))
p + geom_qq(dparams = params) +
    geom_abline()
```




```
# QQ-plot of scaled data against the standard normal distribution
heights %>%
    ggplot(aes(sample = scale(height))) +
    geom_qq() +
    geom_abline()
```

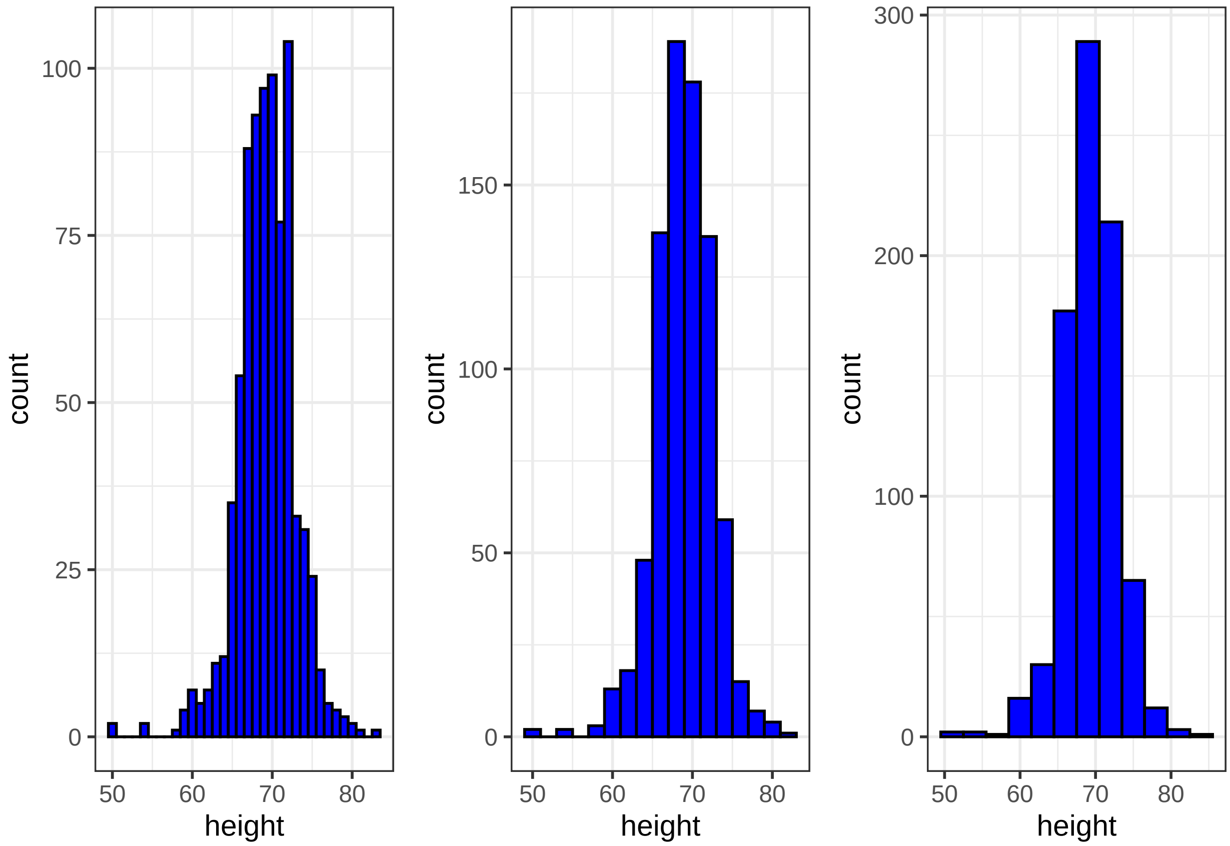


Code: Grids of plots with the grid.extra package

```
if(!require(gridExtra)) install.packages("gridExtra")
```

```
## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# define plots p1, p2, p3
p <- heights %>% filter(sex == "Male") %>% ggplot(aes(x = height))
p1 <- p + geom_histogram(binwidth = 1, fill = "blue", col = "black")
p2 <- p + geom_histogram(binwidth = 2, fill = "blue", col = "black")
p3 <- p + geom_histogram(binwidth = 3, fill = "blue", col = "black")

# arrange plots next to each other in 1 row, 3 columns
library(gridExtra)
grid.arrange(p1, p2, p3, ncol = 3)
```



Assessment - ggplot2

1. Start by loading the dplyr and ggplot2 libraries as well as the `murders` data.

```
library(dplyr)
library(ggplot2)
library(dslabs)
data(murders)
```

Note that you can load both dplyr and ggplot2, as well as other packages, by installing and loading the tidyverse package.

With ggplot2 plots can be saved as objects. For example we can associate a dataset with a plot object like this

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

Because `data` is the first argument we don't need to spell it out. So we can write this instead:

```
p <- ggplot(murders)
```

or, if we load `dplyr`, we can use the pipe:

```
p <- murders %>% ggplot()
```

Remember the pipe sends the object on the left of `%>%` to be the first argument for the function the right of `%>%`.

Now let's get an introduction to `ggplot`.

```
if(!require(dplyr)) install.packages("dplyr")

library(dplyr)
p <- ggplot(murders)
class(p)
```

```
## [1] "gg"     "ggplot"
```

2. Remember that to print an object you can use the command `print` or simply type the object. For example, instead of

```
x <- 2
print(x)
```

you can simply type

```
x <-2
x
```

Print the object `p` defined in exercise one

```
p <- ggplot(murders)
```

and describe what you see.

- ☐ A. Nothing happens.
- ☒ B. A blank slate plot.
- ☐ C. A scatter plot.
- ☐ D. A histogram.

3. Now we are going to review the use of pipes by seeing how they can be used with `ggplot`.

```
# define ggplot object called p like in the previous exercise but using a pipe
p <- heights %>% ggplot()
p # a blank slate plot
```

4. Now we are going to add layers and the corresponding aesthetic mappings. For the murders data, we plotted total murders versus population sizes in the videos.

Explore the **murders** data frame to remind yourself of the names for the two variables (total murders and population size) we want to plot and select the correct answer.

- ☐ A. state and abb.
- ☐ B. total_murders and population_size.
- ☒ C. total and population.
- ☐ D. murders and size.

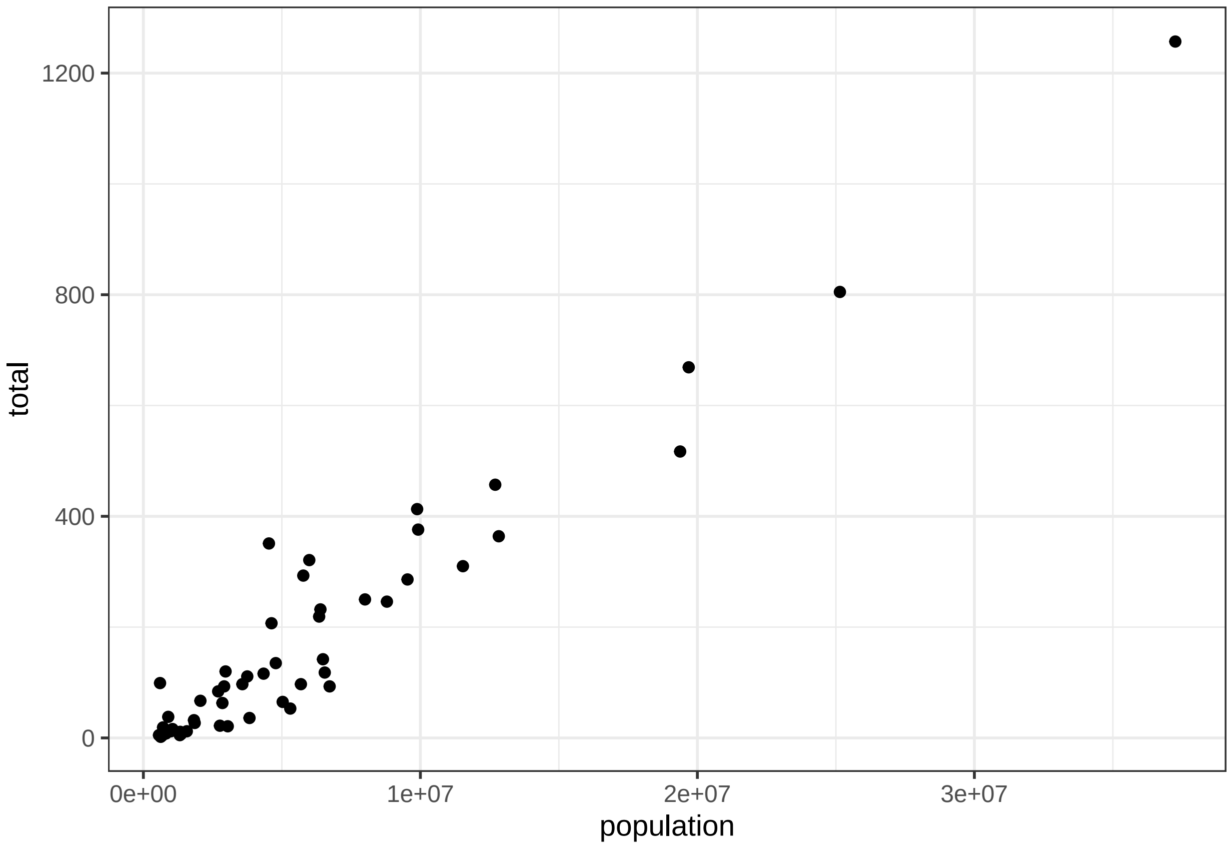
5. To create a scatter plot, we add a layer with the function `geom_point`.

The aesthetic mappings require us to define the x-axis and y-axis variables respectively. So the code looks like this:

```
murders %>% ggplot(aes(x = , y = )) +
  geom_point()
```

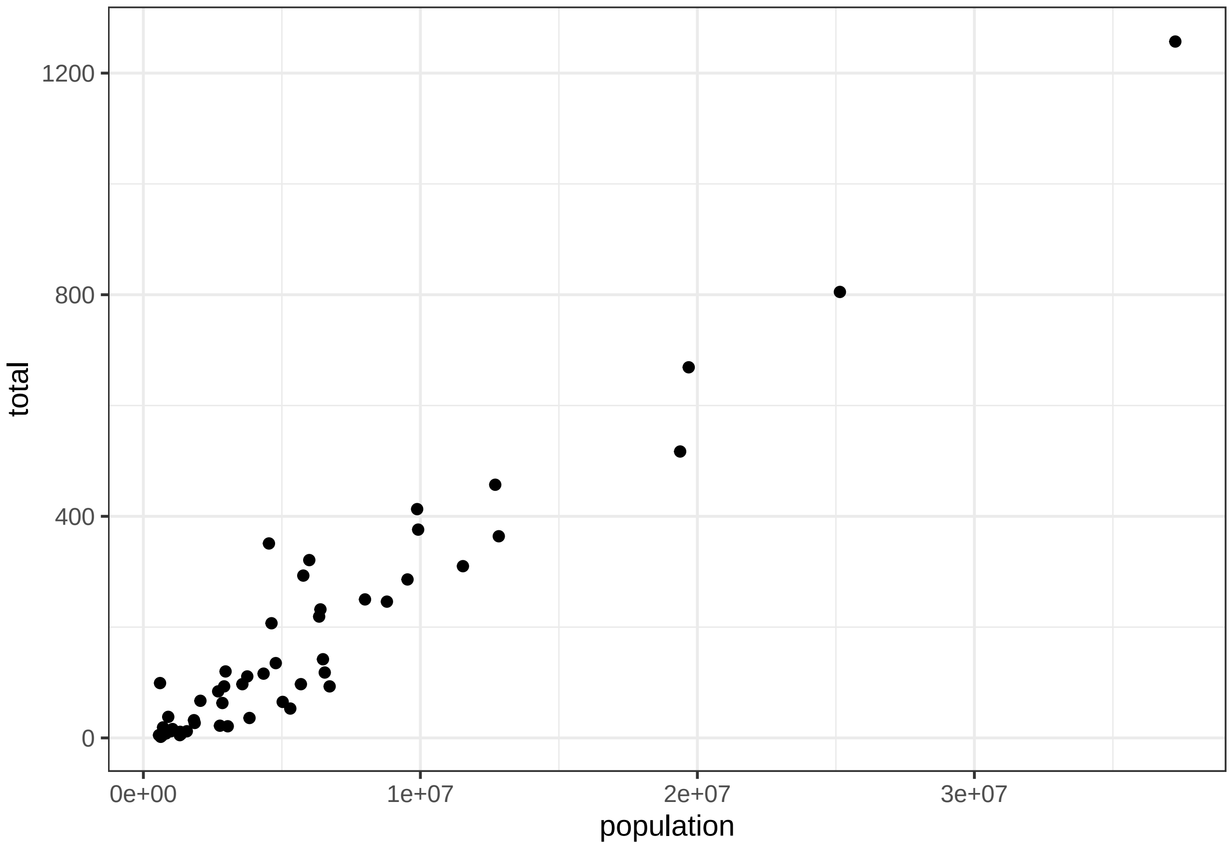
except we have to fill in the blanks to define the two variables `x` and `y`.

```
## Fill in the blanks
murders %>% ggplot(aes(x =population , y =total )) +
  geom_point()
```



6. Note that if we don't use argument names, we can obtain the same plot by making sure we enter the variable names in the desired order.

```
murders %>% ggplot(aes(population, total)) +
  geom_point()
```



7. If instead of points we want to add text, we can use the geom_text() or geom_label() geometries.

However, note that the following code

```
murders %>% ggplot(aes(population, total)) +
  geom_label()
```

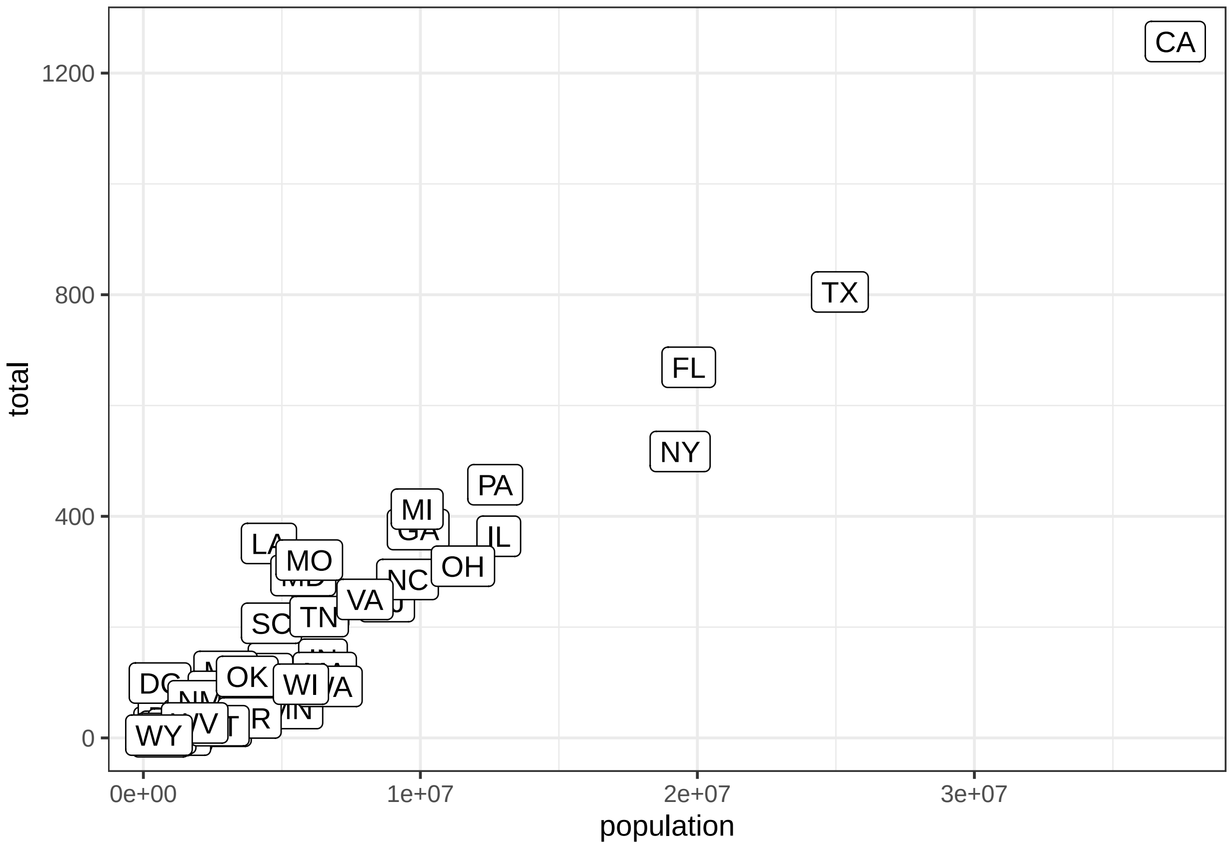
will give us the error message: `Error: geom_label requires the following missing aesthetics: label`

Why is this?

- ☒ A. We need to map a character to each point through the label argument in aes.
- ☐ B. We need to let geom_label know what character to use in the plot.
- ☐ C. The geom_label geometry does not require x-axis and y-axis values.
- ☐ D. geom_label is not a ggplot2 command.

8. You can also add labels to the points on a plot.

```
## edit the next line to add the label
murders %>% ggplot(aes(population, total, label = abb)) + geom_label()
```



9. Now let's change the color of the labels to blue. How can we do this?

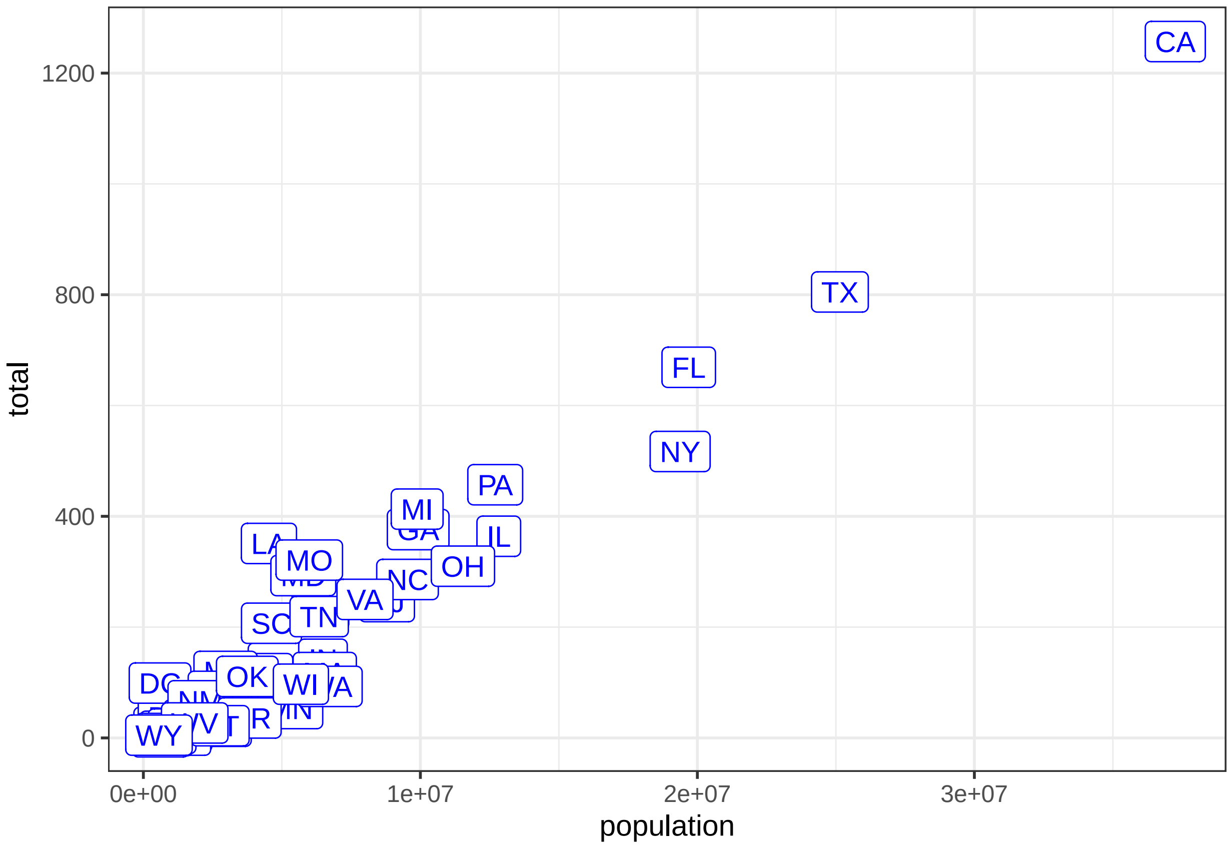
☐ A. By adding a column called blue to murders
☐ B. By mapping the colors through aes because each label needs a different color
☐ C. By using the color argument in ggplot
☒ D. By using the color argument in geom_label because we want all colors to be blue so we do not need to map colors

10. Now let's go ahead and make the labels blue. We previously wrote this code to add labels to our plot:

```
murders %>% ggplot(aes(population, total, label= abb)) +
  geom_label()
```

Now we will edit this code.

```
murders %>% ggplot(aes(population, total,label= abb)) +
  geom_label(color="blue")
```



11. Now suppose we want to use color to represent the different regions.

So the states from the West will be one color, states from the Northeast another, and so on.

In this case, which of the following is most appropriate:

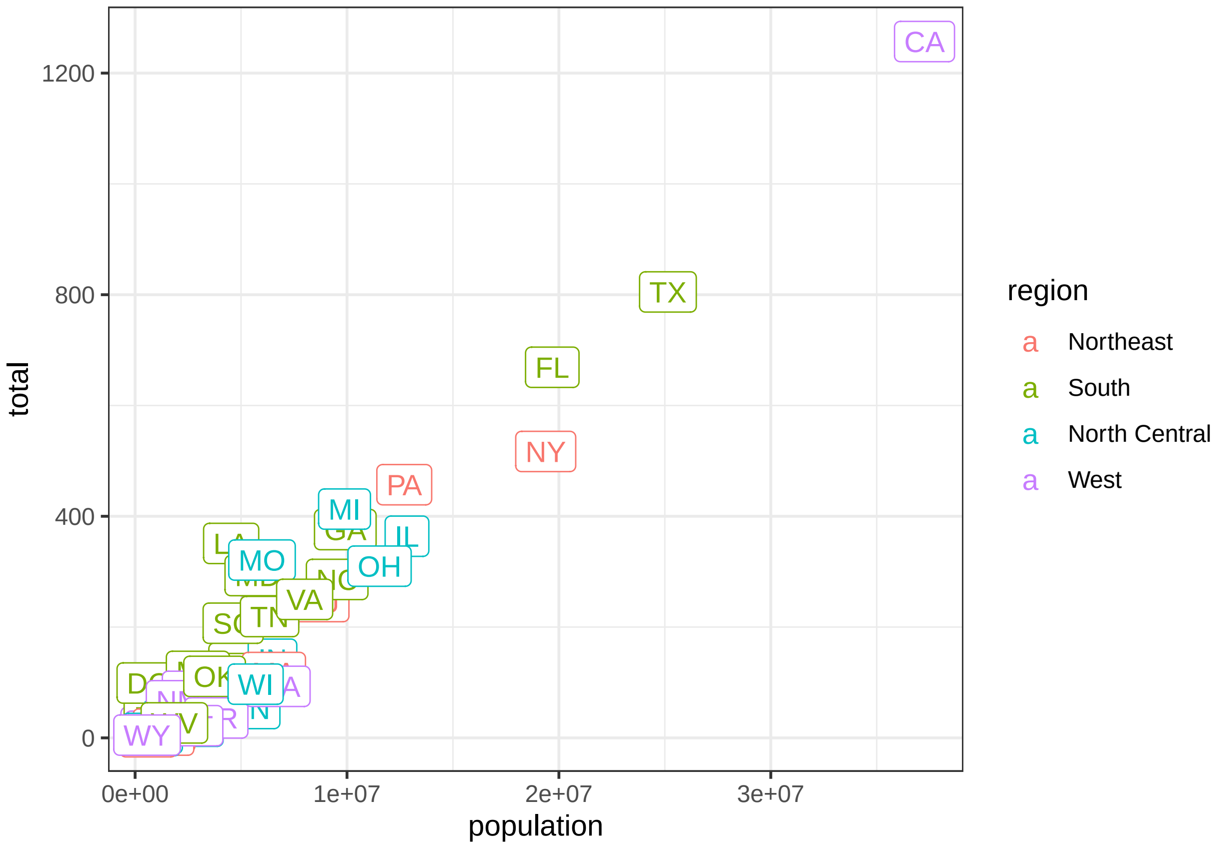
- ☐ A. Adding a column called color to murders with the color we want to use
- ☒ B. Mapping the colors through the color argument of aes because each label needs a different color
- ☐ C. Using the color argument in ggplot
- ☐ D. Using the color argument in geom_label because we want all colors to be blue so we do not need to map colors

12. We previously used this code to make a plot using the state abbreviations as labels:

```
murders %>% ggplot(aes(population, total, label = abb)) +
  geom_label()
```

We are now going to add color to represent the region.

```
## edit this code
murders %>% ggplot(aes(population, total, label = abb, color=region)) +
  geom_label()
```



13. Now we are going to change the axes to log scales to account for the fact that the population distribution is skewed.

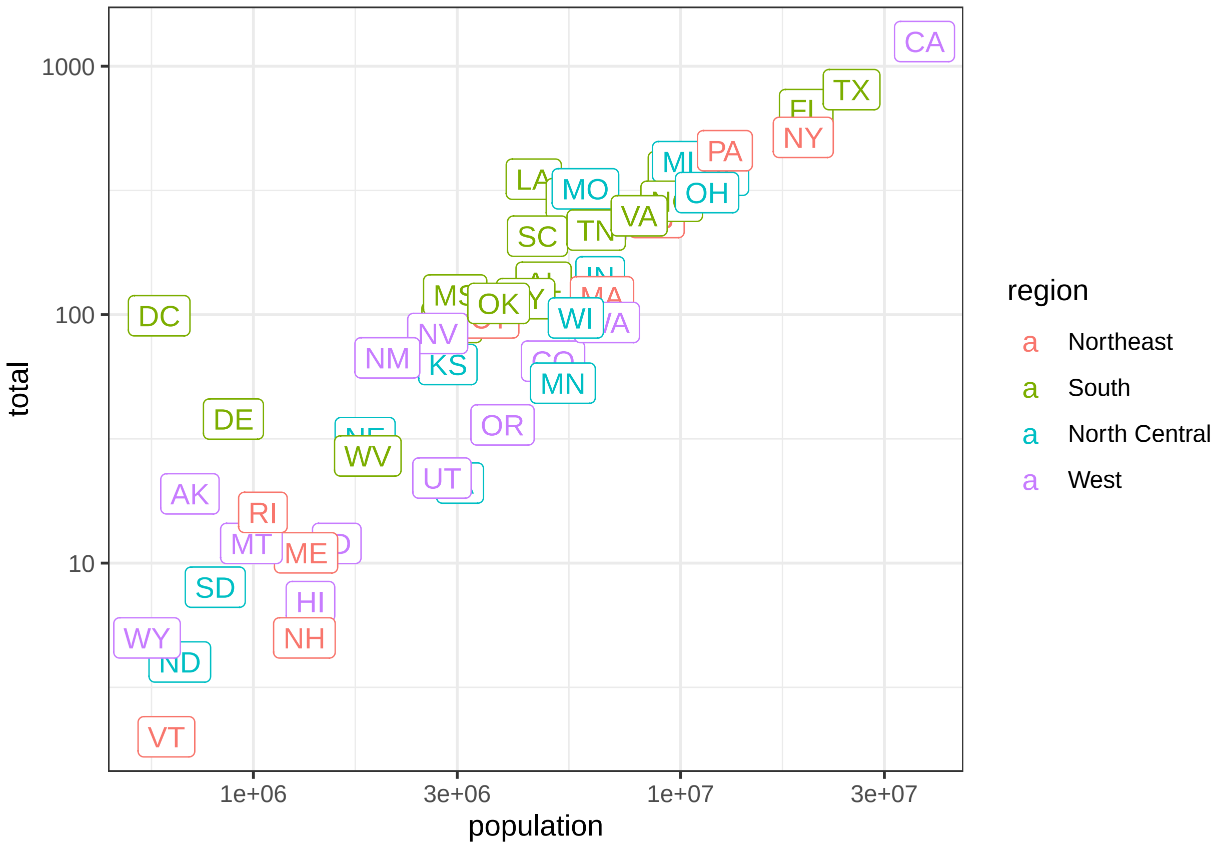
Let's start by defining an object `p` that holds the plot we have made up to now:

```
p <- murders %>% ggplot(aes(population, total, label = abb, color = region)) +
  geom_label()
```

To change the x-axis to a log scale we learned about the `scale_x_log10()` function. We can change the axis by adding this layer to the object `p` to change the scale and render the plot using the following code:

```
p + scale_x_log10()
```

```
p <- murders %>% ggplot(aes(population, total, label = abb, color = region)) + geom_label()
## add layers to p here
p + scale_x_log10() + scale_y_log10()
```

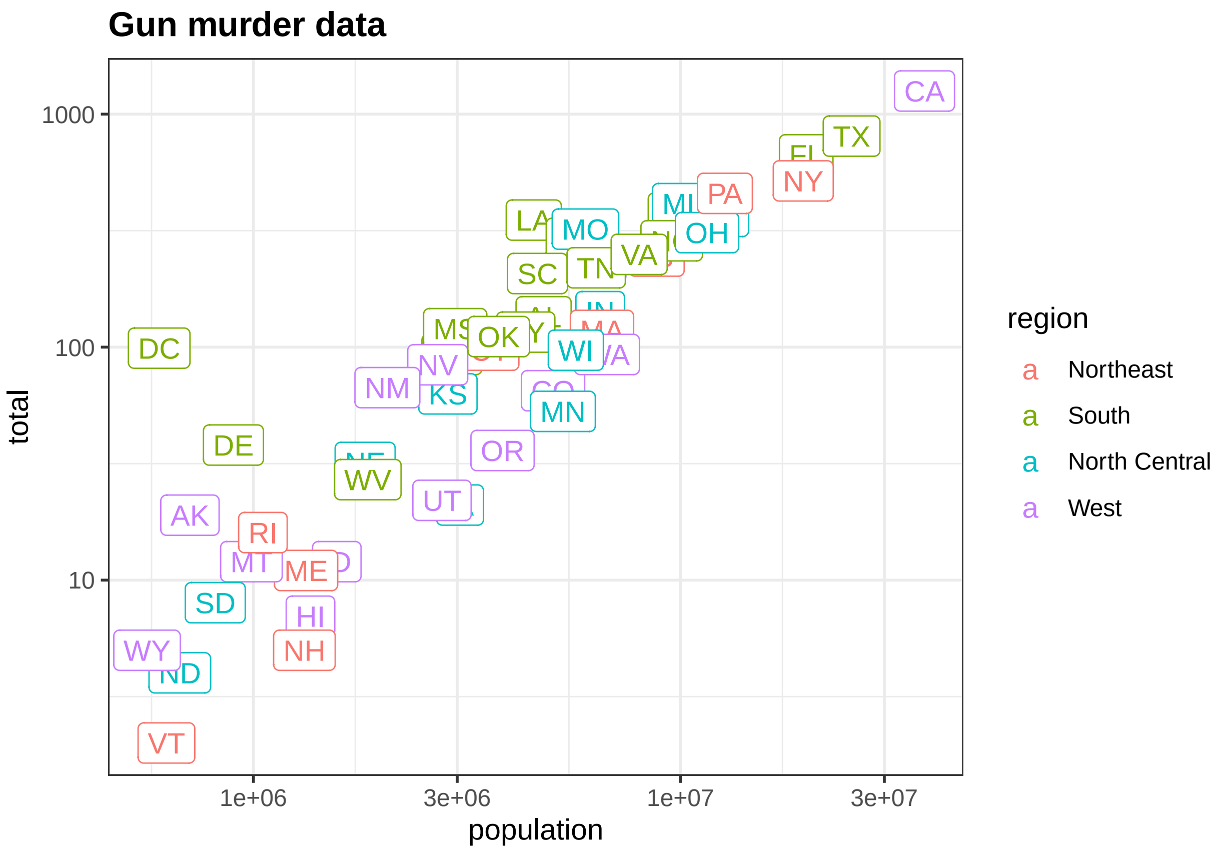


14. In the previous exercises we created a plot using the following code:

```
library(dplyr)
library(ggplot2)
library(dslabs)
data(murders)
p<- murders %>% ggplot(aes(population, total, label = abb, color = region)) +
  geom_label()
p + scale_x_log10() + scale_y_log10()
```

We are now going to add a title to this plot. We will do this by adding yet another layer, this time with the function `ggtitle`.

```
p <- murders %>% ggplot(aes(population, total, label = abb, color = region)) + geom_label()
# add a layer to add title to the next line
p + scale_x_log10() + scale_y_log10() + ggtitle("Gun murder data")
```



15. We are going to shift our focus from the `murders` dataset to explore the `heights` dataset.

We use the `geom_histogram` function to make a histogram of the heights in the `heights` data frame. When reading the documentation for this function we see that it requires just one mapping, the values to be used for the histogram.

What is the variable containing the heights in inches in the `heights` data frame?

- ☐ A. sex
- ☐ B. heights
- ☒ C. height
- ☐ D. heights$height

16. We are now going to make a histogram of the heights so we will load the heights dataset.

The following code has been pre-run for you to load the heights dataset:

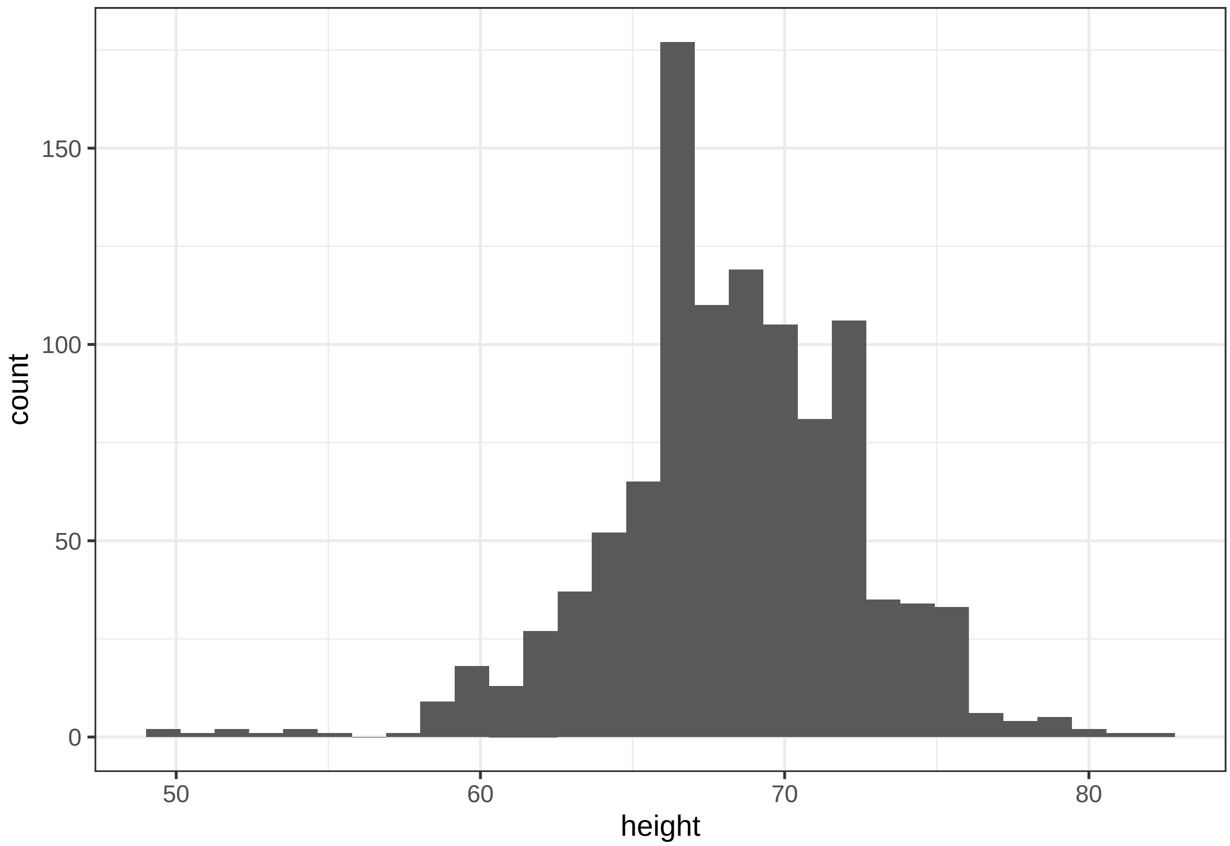
```
library(dplyr)
library(ggplot2)
library(dslabs)
data(heights)
```

```
# define p here
p <- heights %>% ggplot(aes(height))
```

17. Now we are ready to add a layer to actually make the histogram.

```r
p <- heights %>%
  ggplot(aes(height))
## add a layer to p
p + geom_histogram()
```

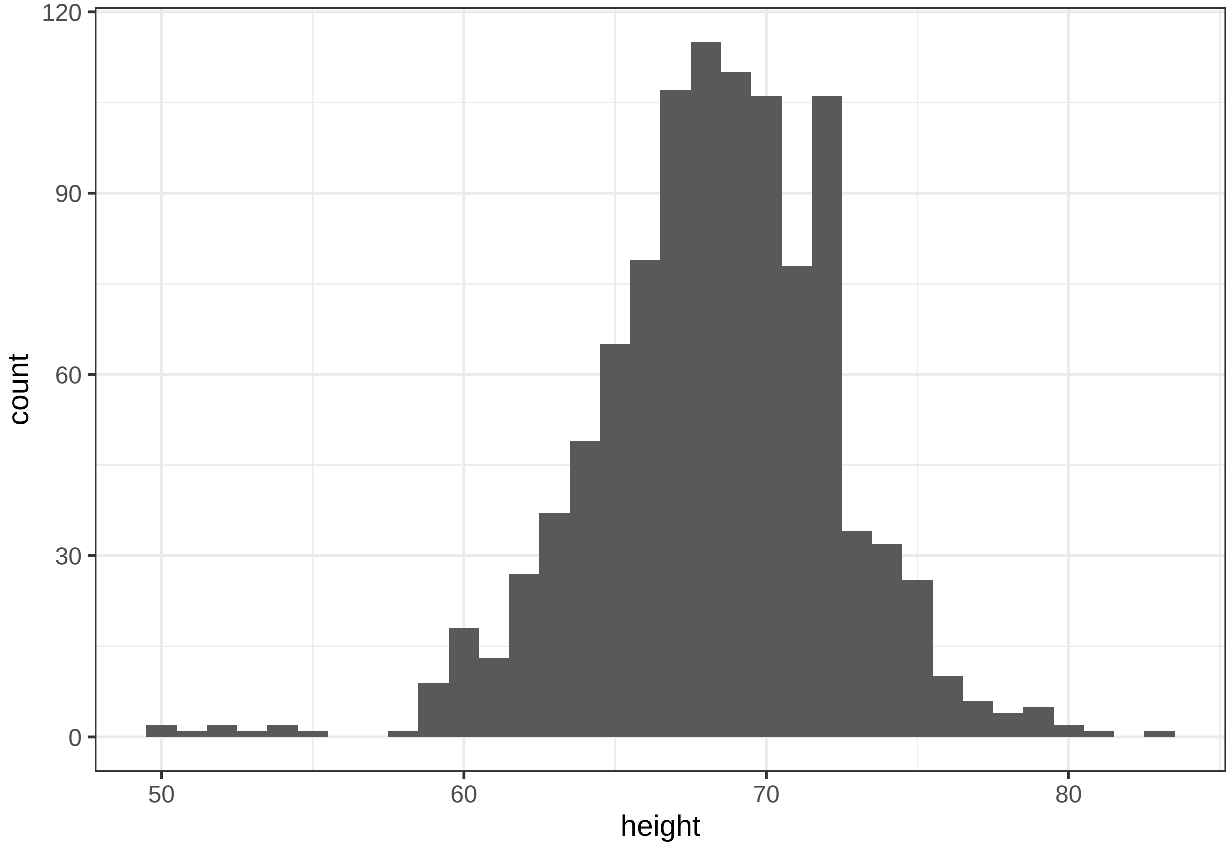
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



18. Note that when we run the code from the previous exercise we get the following warning:

```
stat_bin() using bins = 30. Pick better value with binwidth.
```

```r
p <- heights %>%
  ggplot(aes(height))
## add the geom_histogram layer but with the requested argument
p + geom_histogram(binwidth = 1)
```

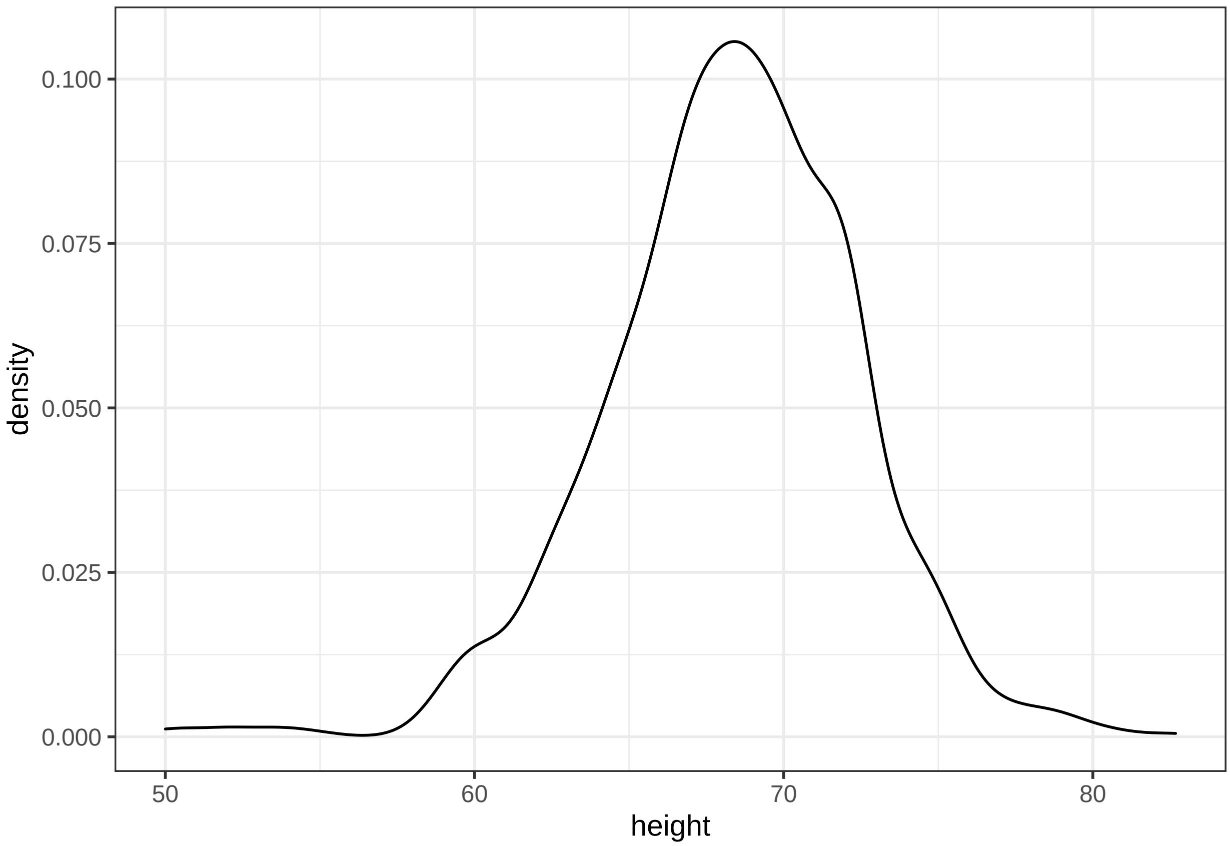


19. Now instead of a histogram we are going to make a smooth density plot.

In this case, we will not make an object `p`. Instead we will render the plot using a single line of code. In the previous exercise, we could have created a histogram using one line of code like this:

```
heights %>%
  ggplot(aes(height)) +
  geom_histogram()
```

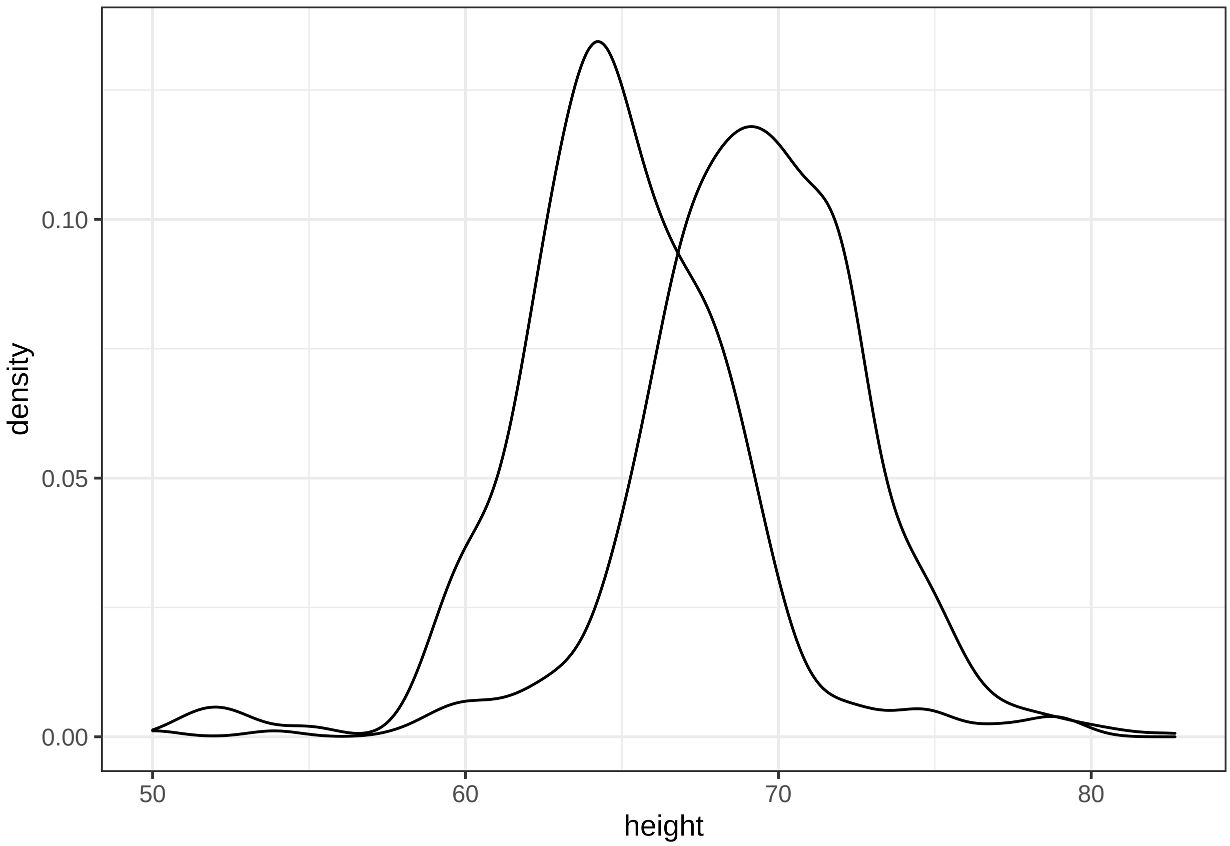
```
## add the correct layer using +
heights %>%
  ggplot(aes(height)) + geom_density()
```



20. Now we are going to make density plots for males and females separately.

We can do this using the `group` argument within the `aes` mapping. Because each point will be assigned to a different density depending on a variable from the dataset, we need to map within `aes`.

```
## add the group argument then a layer with +
heights %>%
  ggplot(aes(height, group = sex)) + geom_density()
```

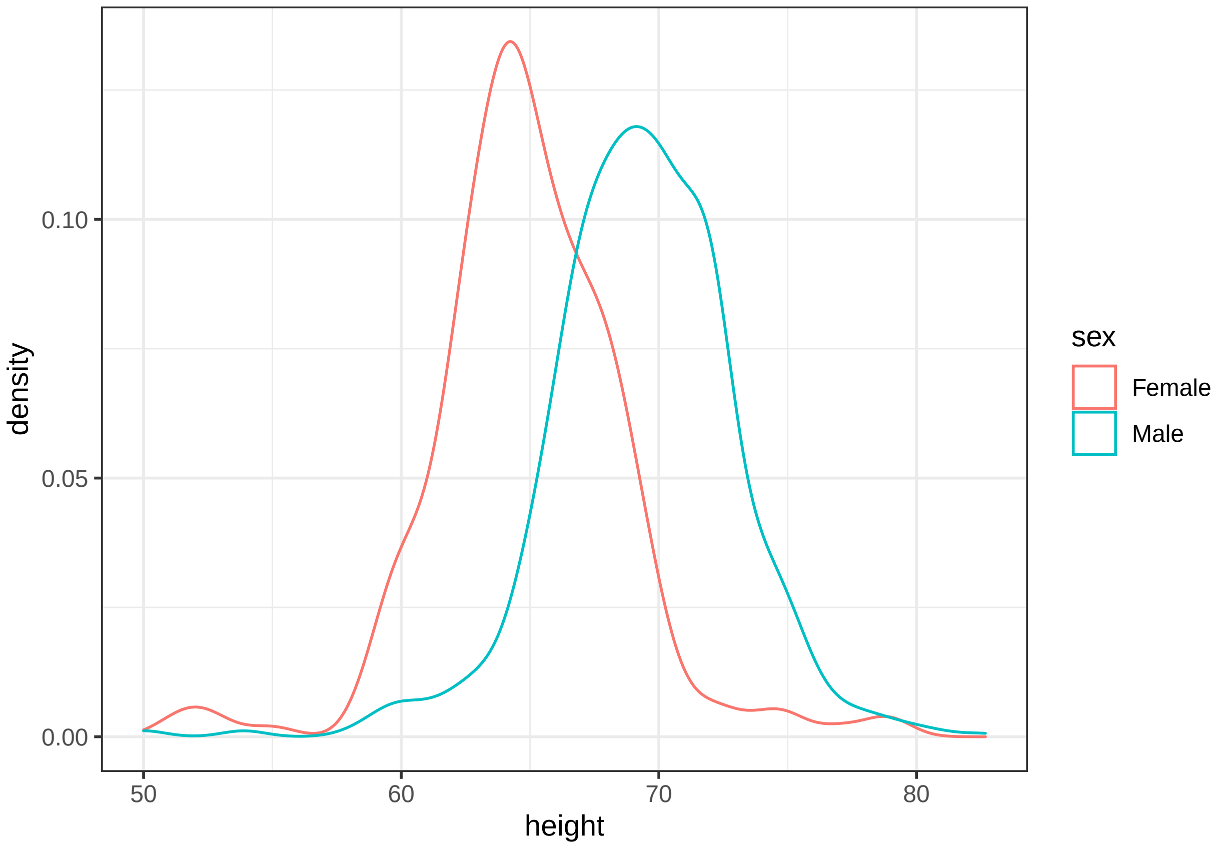


21. In the previous exercise we made the two density plots, one for each sex, using:

```
heights %>%
  ggplot(aes(height, group = sex)) +
  geom_density()
```

We can also assign groups through the `color` or `fill` argument. For example, if you type `color = sex` ggplot knows you want a different color for each sex. So two densities must be drawn. You can therefore skip the `group = sex` mapping. Using `color` has the added benefit that it uses color to distinguish the groups. Change the density plots from the previous exercise to add color.

```
## edit the next line to use color instead of group then add a density layer
heights %>%
  ggplot(aes(height, color = sex)) + geom_density()
```



22. We can also assign groups using the `fill` argument.

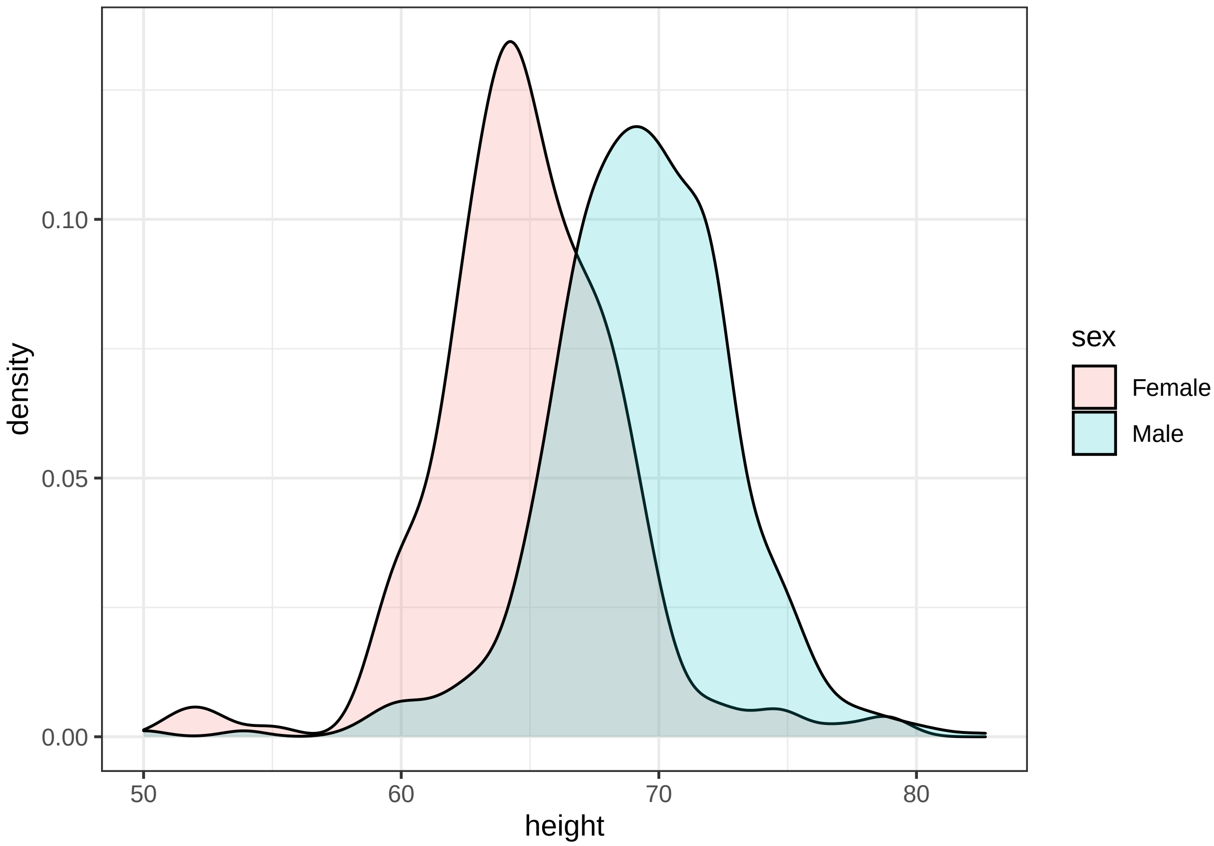
When using the `geom_density` geometry, `color` creates a colored line for the smooth density plot while `fill` colors in the area under the curve.

We can see what this looks like by running the following code:

```
heights %>%
  ggplot(aes(height, fill = sex)) +
  geom_density()
```

However, here the second density is drawn over the other. We can change this by using something called *alpha blending*.

```
heights %>%
  ggplot(aes(height, fill = sex)) +
  geom_density(alpha=0.2)
```



Section 3 Overview

Section 3 introduces you to summarizing with dplyr.

After completing Section 3, you will:

- understand the importance of summarizing data in exploratory data analysis.
- be able to use the "summarize" verb in dplyr to facilitate summarizing data.
- be able to use the "group_by" verb in dplyr to facilitate summarizing data.
- be able to access values using the dot placeholder.
- be able to use "arrange" to examine data after sorting.

dplyr

The textbook for this section is available here

Key points

- `summarize` from the dplyr/tidyverse package computes summary statistics from the data frame. It returns a data frame whose column names are defined within the function call.
- `summarize` can compute any summary function that operates on vectors and returns a single value, but it cannot operate on functions that return multiple values.
- Like most dplyr functions, `summarize` is aware of variable names within data frames and can use them directly.

Code

```
# compute average and standard deviation for males
s <- heights %>%
    filter(sex == "Male") %>%
    summarize(average = mean(height), standard_deviation = sd(height))

# access average and standard deviation from summary table
s$average
```

```
## [1] 69.31475
```

```
s$standard_deviation
```

```
## [1] 3.611024
```

```
# compute median, min and max
heights %>%
    filter(sex == "Male") %>%
    summarize(median = median(height),
                      minimum = min(height),
                      maximum = max(height))
```

```
##   median minimum  maximum
## 1     69      50 82.67717
```

```
# alternative way to get min, median, max in base R
quantile(heights$height, c(0, 0.5, 1))
```

```
##       0%      50%     100%
## 50.00000 68.50000 82.67717
```

```
# generates an error: summarize can only take functions that return a single value
heights %>%
    filter(sex == "Male") %>%
    summarize(range = quantile(height, c(0, 0.5, 1)))
```

The Dot Placeholder

The textbook for this section is available here

Note that a common replacement for the dot operator is the pull function. Here is the textbook section on the pull function.

Key points

- The dot operator allows you to access values stored in data that is being piped in using the `%>%` character. The dot is a placeholder for the data being passed in through the pipe.
- The dot operator allows `dplyr` functions to return single vectors or numbers instead of only data frames.
- `us_murder_rate %>% .$rate` is equivalent to `us_murder_rate$rate`.

- Note that an equivalent way to extract a single column using the pipe is `us_murder_rate %>% pull(rate)`. The pull function will be used in later course material.

Code

```
murders <- murders %>% mutate(murder_rate = total/population*100000)
summarize(murders, mean(murder_rate))
```

```
##   mean(murder_rate)
## 1          2.779125
```

```
# calculate US murder rate, generating a data frame
us_murder_rate <- murders %>%
    summarize(rate = sum(total) / sum(population) * 100000)
us_murder_rate
```

```
##       rate
## 1 3.034555
```

```
# extract the numeric US murder rate with the dot operator
us_murder_rate %>% .$rate
```

```
## [1] 3.034555
```

```
# calculate and extract the murder rate with one pipe
us_murder_rate <- murders %>%
    summarize(rate = sum(total) / sum(population * 100000)) %>%
    .$rate
```

Group By

The textbook for this section is available here

Key points

- The `group_by` function from **dplyr** converts a data frame to a grouped data frame, creating groups using one or more variables.
- `summarize` and some other **dplyr** functions will behave differently on grouped data frames.
- Using `summarize` on a grouped data frame computes the summary statistics for each of the separate groups.

Code

```
# compute separate average and standard deviation for male/female heights
heights %>%
    group_by(sex) %>%
    summarize(average = mean(height), standard_deviation = sd(height))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 2 x 3
##   sex    average standard_deviation
##   <fct>    <dbl>              <dbl>
## 1 Female    64.9               3.76
## 2 Male      69.3               3.61
```

```
# compute median murder rate in 4 regions of country
murders <- murders %>%
    mutate(murder_rate = total/population * 100000)
murders %>%
    group_by(region) %>%
    summarize(median_rate = median(murder_rate))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 4 x 2
##   region        median_rate
##   <fct>               <dbl>
## 1 Northeast            1.80
## 2 South                3.40
## 3 North Central        1.97
## 4 West                 1.29
```

Sorting Data Tables

The textbook for this section is available here

Key points

- The `arrange` function from **dplyr** sorts a data frame by a given column.
- By default, `arrange` sorts in ascending order (lowest to highest). To instead sort in descending order, use the function `desc` inside of `arrange`.
- You can `arrange` by multiple levels: within equivalent values of the first level, observations are sorted by the second level, and so on.
- The `top_n` function shows the top results ranked by a given variable, but the results are not ordered. You can combine `top_n` with `arrange` to return the top results in order.

Code

```
# set up murders object
murders <- murders %>%
    mutate(murder_rate = total/population * 100000)

# arrange by population column, smallest to largest
murders %>% arrange(population) %>% head()
```

```
##                  state abb        region population total murder_rate
## 1              Wyoming  WY          West     563626     5   0.8871131
## 2 District of Columbia  DC         South     601723    99  16.4527532
## 3              Vermont  VT     Northeast     625741     2   0.3196211
## 4         North Dakota  ND North Central     672591     4   0.5947151
## 5               Alaska  AK          West     710231    19   2.6751860
## 6         South Dakota  SD North Central     814180     8   0.9825837
```

```r
# arrange by murder rate, smallest to largest
murders %>% arrange(murder_rate) %>% head()
```

```
##            state abb        region population total murder_rate
## 1        Vermont  VT     Northeast     625741     2   0.3196211
## 2 New Hampshire  NH     Northeast    1316470     5   0.3798036
## 3         Hawaii  HI          West    1360301     7   0.5145920
## 4  North Dakota  ND North Central     672591     4   0.5947151
## 5           Iowa  IA North Central    3046355    21   0.6893484
## 6          Idaho  ID          West    1567582    12   0.7655102
```

```r
# arrange by murder rate in descending order
murders %>% arrange(desc(murder_rate)) %>% head()
```

```
##                   state abb        region population total murder_rate
## 1 District of Columbia  DC         South     601723    99   16.452753
## 2            Louisiana  LA         South    4533372   351    7.742581
## 3             Missouri  MO North Central    5988927   321    5.359892
## 4             Maryland  MD         South    5773552   293    5.074866
## 5       South Carolina  SC         South    4625364   207    4.475323
## 6             Delaware  DE         South     897934    38    4.231937
```

```r
# arrange by region alphabetically, then by murder rate within each region
murders %>% arrange(region, murder_rate) %>% head()
```

```
##            state abb    region population total murder_rate
## 1        Vermont  VT Northeast     625741     2   0.3196211
## 2 New Hampshire  NH Northeast    1316470     5   0.3798036
## 3          Maine  ME Northeast    1328361    11   0.8280881
## 4  Rhode Island  RI Northeast    1052567    16   1.5200933
## 5 Massachusetts  MA Northeast    6547629   118   1.8021791
## 6       New York  NY Northeast   19378102   517   2.6679599
```

```r
# show the top 10 states with highest murder rate, not ordered by rate
murders %>% top_n(10, murder_rate)
```

```
##                   state abb        region population total murder_rate
## 1               Arizona  AZ          West    6392017   232    3.629527
## 2              Delaware  DE         South     897934    38    4.231937
## 3  District of Columbia  DC         South     601723    99   16.452753
## 4               Georgia  GA         South    9920000   376    3.790323
## 5             Louisiana  LA         South    4533372   351    7.742581
## 6              Maryland  MD         South    5773552   293    5.074866
## 7              Michigan  MI North Central    9883640   413    4.178622
## 8           Mississippi  MS         South    2967297   120    4.044085
## 9              Missouri  MO North Central    5988927   321    5.359892
## 10       South Carolina  SC         South    4625364   207    4.475323
```

```r
# show the top 10 states with highest murder rate, ordered by rate
murders %>% arrange(desc(murder_rate)) %>% top_n(10)
```

```
## Selecting by murder_rate
```

```
##                   state abb        region population total murder_rate
## 1  District of Columbia  DC         South     601723    99   16.452753
## 2             Louisiana  LA         South    4533372   351    7.742581
## 3              Missouri  MO North Central    5988927   321    5.359892
## 4              Maryland  MD         South    5773552   293    5.074866
## 5        South Carolina  SC         South    4625364   207    4.475323
## 6              Delaware  DE         South     897934    38    4.231937
## 7              Michigan  MI North Central    9883640   413    4.178622
## 8           Mississippi  MS         South    2967297   120    4.044085
## 9               Georgia  GA         South    9920000   376    3.790323
## 10              Arizona  AZ          West    6392017   232    3.629527
```

Assessment - Summarizing with dplyr

To practice our dplyr skills we will be working with data from the survey collected by the United States National Center for Health Statistics (NCHS). This center has conducted a series of health and nutrition surveys since the 1960's.

Starting in 1999, about 5,000 individuals of all ages have been interviewed every year and then they complete the health examination component of the survey. Part of this dataset is made available via the NHANES package which can be loaded this way:

```r
if(!require(NHANES)) install.packages("NHANES")
```

```
## Loading required package: NHANES
```

```
## Warning: package 'NHANES' was built under R version 4.0.2
```

```r
library(NHANES)
data(NHANES)
```

The NHANES data has many missing values. Remember that the main summarization function in R will return `NA` if any of the entries of the input vector is an `NA`. Here is an example:

```r
data(na_example)
mean(na_example)
```

```
## [1] NA
```

```r
sd(na_example)
```

```
## [1] NA
```

To ignore the `NA`s, we can use the `na.rm` argument:

```r
mean(na_example, na.rm = TRUE)
```

```
## [1] 2.301754
```

```r
sd(na_example, na.rm = TRUE)
```

```
## [1] 1.22338
```

Try running this code, then let us know you are ready to proceed with the analysis.

1. Let's explore the `NHANES` data. We will be exploring blood pressure in this dataset.

First let's select a group to set the standard. We will use 20-29 year old females. Note that the category is coded with `20-29`, with a space in front of the `20`! The `AgeDecade` is a categorical variable with these ages.

To know if someone is female, you can look at the `Gender` variable.

```r
## fill in what is needed
tab <- NHANES %>% filter(AgeDecade == " 20-29" & Gender == "female")
head(tab)
```

```
## # A tibble: 6 x 76
##      ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
##   <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
## 1 51710 2009_10  female    26 " 20-29"        319 White <NA>  College ~
## 2 51731 2009_10  female    28 " 20-29"        346 Black <NA>  High Sch~
## 3 51741 2009_10  female    21 " 20-29"        253 Black <NA>  Some Col~
## 4 51741 2009_10  female    21 " 20-29"        253 Black <NA>  Some Col~
## 5 51760 2009_10  female    27 " 20-29"        334 Hisp~ <NA>  9 - 11th~
## 6 51764 2009_10  female    29 " 20-29"        357 White <NA>  College ~
## # ... with 67 more variables: MaritalStatus <fct>, HHIncome <fct>,
## #   HHIncomeMid <int>, Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>,
## #   Work <fct>, Weight <dbl>, Length <dbl>, HeadCirc <dbl>, Height <dbl>,
## #   BMI <dbl>, BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>,
## #   BPSysAve <int>, BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>,
## #   BPDia2 <int>, BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>,
## #   DirectChol <dbl>, TotChol <dbl>, UrineVol1 <int>, UrineFlow1 <dbl>,
## #   UrineVol2 <int>, UrineFlow2 <dbl>, Diabetes <fct>, DiabetesAge <int>,
## #   HealthGen <fct>, DaysPhysHlthBad <int>, DaysMentHlthBad <int>,
## #   LittleInterest <fct>, Depressed <fct>, nPregnancies <int>, nBabies <int>,
## #   Age1stBaby <int>, SleepHrsNight <int>, SleepTrouble <fct>,
## #   PhysActive <fct>, PhysActiveDays <int>, TVHrsDay <fct>, CompHrsDay <fct>,
## #   TVHrsDayChild <int>, CompHrsDayChild <int>, Alcohol12PlusYr <fct>,
## #   AlcoholDay <int>, AlcoholYear <int>, SmokeNow <fct>, Smoke100 <fct>,
## #   Smoke100n <fct>, SmokeAge <int>, Marijuana <fct>, AgeFirstMarij <int>,
## #   RegularMarij <fct>, AgeRegMarij <int>, HardDrugs <fct>, SexEver <fct>,
## #   SexAge <int>, SexNumPartnLife <int>, SexNumPartYear <int>, SameSex <fct>,
## #   SexOrientation <fct>, PregnantNow <fct>
```

2. Now we will compute the average and standard deviation for the subgroup we defined in the previous exercise (20-29 year old females), which we will use reference for what is typical.

You will determine the average and standard deviation of systolic blood pressure, which are stored in the `BPSysAve` variable in the NHANES dataset.

```r
## complete this line of code.
ref <- NHANES %>% filter(AgeDecade == " 20-29" & Gender == "female") %>% summarize(average = mean(BPSys
ref
```

```
## # A tibble: 1 x 2
##   average standard_deviation
##     <dbl>              <dbl>
## 1    108.               10.1
```

3. Now we will repeat the exercise and generate only the average blood pressure for 20-29 year old females.

For this exercise, you should review how to use the place holder `.` in dplyr or the `pull` function.

```r
## modify the code we wrote for previous exercise.
ref_avg <- NHANES %>%
  filter(AgeDecade == " 20-29" & Gender == "female") %>%
  summarize(average = mean(BPSysAve, na.rm = TRUE),
            standard_deviation = sd(BPSysAve, na.rm=TRUE)) %>% .$average
ref_avg
```

```
## [1] 108.4224
```

4. Let's continue practicing by calculating two other data summaries: the minimum and the maximum.

Again we will do it for the `BPSysAve` variable and the group of 20-29 year old females.

```r
## complete the line
NHANES %>%
      filter(AgeDecade == " 20-29"  & Gender == "female") %>% summarize(minbp = min(BPSysAve, na.rm = T
            maxbp = max(BPSysAve, na.rm=TRUE))
```

```
## # A tibble: 1 x 2
##   minbp maxbp
##   <int> <int>
## 1    84   179
```

5. Now let's practice using the `group_by` function.

What we are about to do is a very common operation in data science: you will split a data table into groups and then compute summary statistics for each group.

We will compute the average and standard deviation of systolic blood pressure for females for each age group separately. Remember that the age groups are contained in `AgeDecade`.

```r
##complete the line with group_by and summarize
NHANES %>%
      filter(Gender == "female") %>% group_by(AgeDecade) %>% summarize(average = mean(BPSysAve, na.rm =
            standard_deviation = sd(BPSysAve, na.rm=TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 9 x 3
##   AgeDecade average standard_deviation
##   <fct>       <dbl>              <dbl>
## 1 " 0-9"       100.               9.07
## 2 " 10-19"     104.               9.46
## 3 " 20-29"     108.              10.1
## 4 " 30-39"     111.              12.3
## 5 " 40-49"     115.              14.5
## 6 " 50-59"     122.              16.2
## 7 " 60-69"     127.              17.1
## 8 " 70+"       134.              19.8
## 9  <NA>        142.              22.9
```

6. Now let's practice using `group_by` some more.

We are going to repeat the previous exercise of calculating the average and standard deviation of systolic blood pressure, but for males instead of females.

This time we will not provide much sample code. You are on your own!

```r
NHANES %>%
      filter(Gender == "male") %>% group_by(AgeDecade) %>% summarize(average = mean(BPSysAve, na.rm = TI
            standard_deviation = sd(BPSysAve, na.rm=TRUE))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 9 x 3
##   AgeDecade average standard_deviation
##   <fct>       <dbl>              <dbl>
## 1 " 0-9"       97.4               8.32
## 2 " 10-19"    110.               11.2
## 3 " 20-29"    118.               11.3
## 4 " 30-39"    119.               12.3
## 5 " 40-49"    121.               14.0
## 6 " 50-59"    126.               17.8
## 7 " 60-69"    127.               17.5
## 8 " 70+"      130.               18.7
## 9  <NA>       136.               23.5
```

7. We can actually combine both of these summaries into a single line of code.

This is because `group_by` permits us to group by more than one variable.

We can use `group_by(AgeDecade, Gender)` to group by both age decades and gender.

```r
NHANES %>% group_by(AgeDecade, Gender) %>% summarize(average = mean(BPSysAve, na.rm = TRUE),
            standard_deviation = sd(BPSysAve, na.rm=TRUE))
```

```
## `summarise()` regrouping output by 'AgeDecade' (override with `.groups` argument)
```

```
## # A tibble: 18 x 4
## # Groups:   AgeDecade [9]
```

```
##    AgeDecade Gender average standard_deviation
##    <fct>     <fct>    <dbl>              <dbl>
##  1 " 0-9"    female   100.                9.07
##  2 " 0-9"    male      97.4               8.32
##  3 " 10-19"  female   104.                9.46
##  4 " 10-19"  male     110.               11.2
##  5 " 20-29"  female   108.               10.1
##  6 " 20-29"  male     118.               11.3
##  7 " 30-39"  female   111.               12.3
##  8 " 30-39"  male     119.               12.3
##  9 " 40-49"  female   115.               14.5
## 10 " 40-49"  male     121.               14.0
## 11 " 50-59"  female   122.               16.2
## 12 " 50-59"  male     126.               17.8
## 13 " 60-69"  female   127.               17.1
## 14 " 60-69"  male     127.               17.5
## 15 " 70+"    female   134.               19.8
## 16 " 70+"    male     130.               18.7
## 17  <NA>     female   142.               22.9
## 18  <NA>     male     136.               23.5
```

8. Now we are going to explore differences in systolic blood pressure across races, as reported in the `Race1` variable.

We will learn to use the `arrange` function to order the outcome acording to one variable.

Note that this function can be used to order any table by a given outcome. Here is an example that arranges by systolic blood pressure.

```
NHANES %>% arrange(BPSysAve)
```

If we want it in descending order we can use the `desc` function like this:

```
NHANES %>% arrange(desc(BPSysAve))
```

In this example, we will compare systolic blood pressure across values of the `Race1` variable for males between the ages of 40-49.

```
NHANES %>% filter(AgeDecade == " 40-49" & Gender == "male") %>% group_by(Race1) %>% summarize(average =
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 5 x 3
##   Race1    average standard_deviation
##   <fct>      <dbl>              <dbl>
## 1 White       120.               13.4
## 2 Other       120.               16.2
## 3 Hispanic    122.               11.1
## 4 Mexican     122.               13.9
## 5 Black       126.               17.1
```

Section 4 Overview

In Section 4, you will look at a case study involving data from the Gapminder Foundation about trends in world health and economics.

After completing Section 4, you will:

- understand how Hans Rosling and the Gapminder Foundation use effective data visualization to convey data-based trends.
- be able to apply the ggplot2 techniques from the previous section to answer questions using data.
- understand how fixed scales across plots can ease comparisons.
- be able to modify graphs to improve data visualization.

Case Study: Trends in World Health and Economics

The textbook for this section is available here

More about Gapminder

The original Gapminder TED talks are available and we encourage you to watch them.

- The Best Stats You've Ever Seen
- New Insights on Poverty

You can also find more information and raw data (in addition to what we analyze in class) at.

Key points

- Data visualization can be used to dispel common myths and educate the public and contradict sensationalist or outdated claims and stories.
- We will use real data to answer the following questions about world health and economics:
  - Is it still fair to consider the world as divided into the West and the developing world?
  - Has income inequality across countries worsened over the last 40 years?

Gapminder Dataset

The textbook for this section is available here

Key points

- A selection of world health and economics statistics from the Gapminder project can be found in the **dslabs** package as `data(gapminder)`.
- Most people have misconceptions about world health and economics, which can be addressed by considering real data.

Code

```
# load and inspect gapminder data
data(gapminder)
head(gapminder)
```

```
##                country year infant_mortality life_expectancy fertility
## 1             Albania 1960           115.40           62.87      6.19
## 2             Algeria 1960           148.20           47.50      7.65
## 3              Angola 1960           208.00           35.98      7.32
## 4 Antigua and Barbuda 1960               NA           62.97      4.43
## 5           Argentina 1960            59.87           65.39      3.11
## 6             Armenia 1960               NA           66.86      4.55
##   population          gdp continent          region
## 1    1636054           NA    Europe Southern Europe
## 2   11124892  13828152297    Africa Northern Africa
## 3    5270844           NA    Africa   Middle Africa
## 4      54681           NA  Americas       Caribbean
## 5   20619075 108322326649  Americas   South America
## 6    1867396           NA      Asia    Western Asia
```

```
# compare infant mortality in Sri Lanka and Turkey
gapminder %>%
    filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>%
    select(country, infant_mortality)
```

```
##     country infant_mortality
## 1 Sri Lanka              8.4
## 2    Turkey             11.6
```

Life Expectancy and Fertility Rates

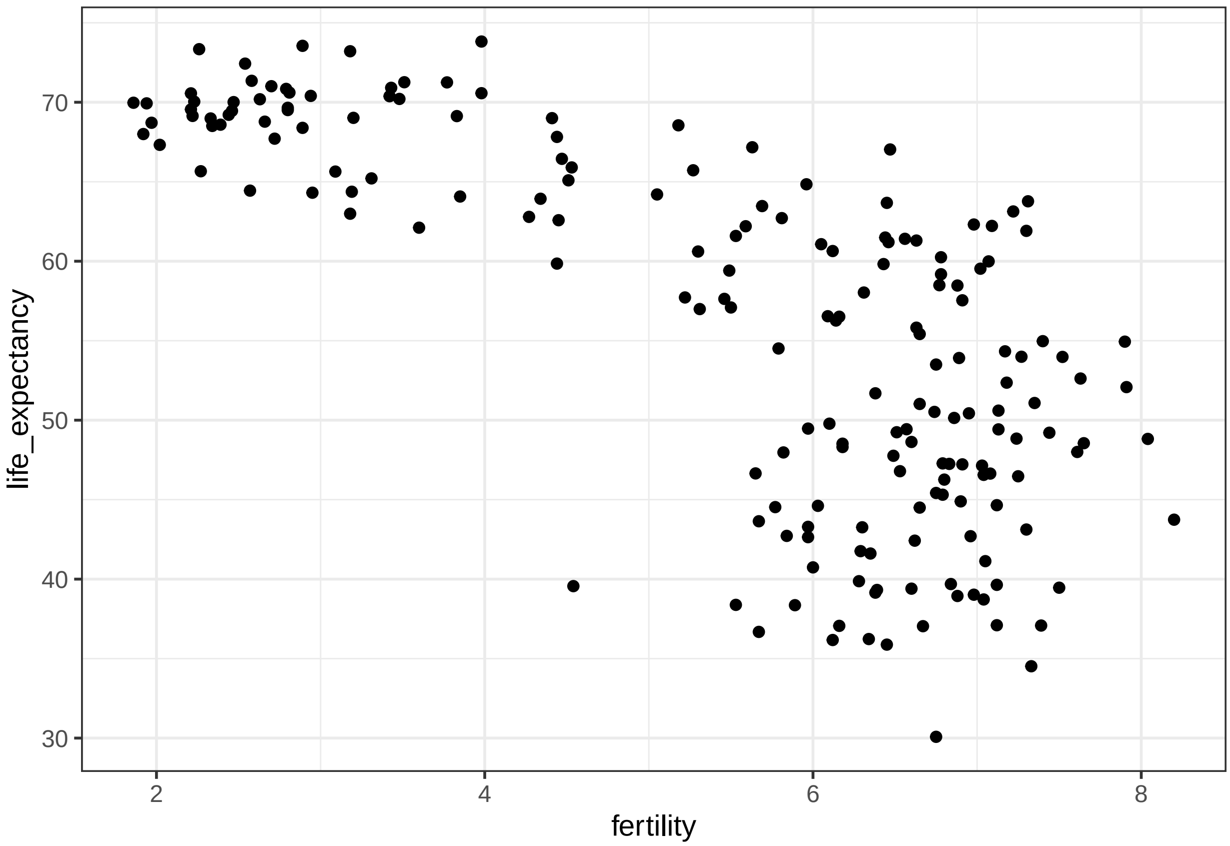
The textbook for this section is available here

Key points

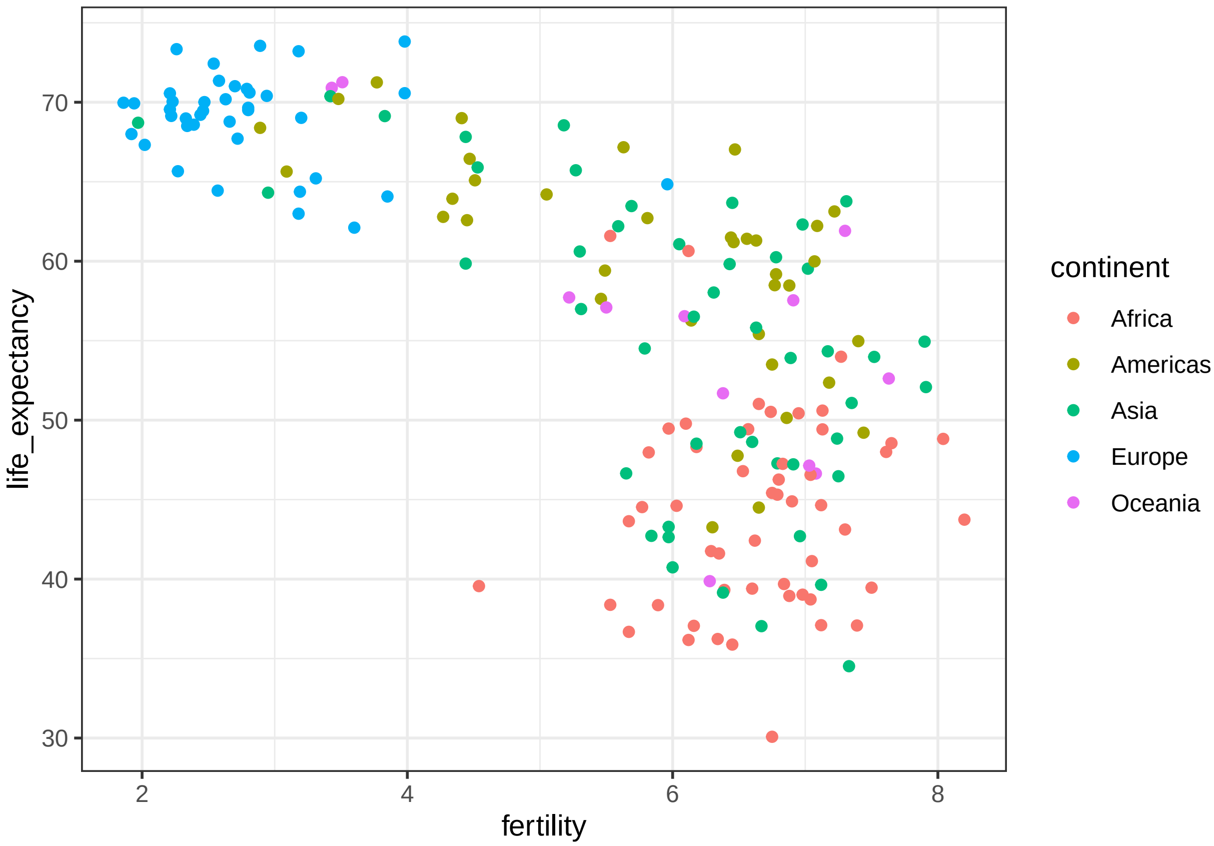
- A prevalent worldview is that the world is divided into two groups of countries:
  - Western world: high life expectancy, low fertility rate
  - Developing world: lower life expectancy, higher fertility rate
- Gapminder data can be used to evaluate the validity of this view.
- A scatterplot of life expectancy versus fertility rate in 1962 suggests that this viewpoint was grounded in reality 50 years ago. Is it still the case today?

Code

```
# basic scatterplot of life expectancy versus fertility
ds_theme_set()    # set plot theme
filter(gapminder, year == 1962) %>%
    ggplot(aes(fertility, life_expectancy)) +
    geom_point()
```




```
# add color as continent
filter(gapminder, year == 1962) %>%
    ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point()
```



Faceting

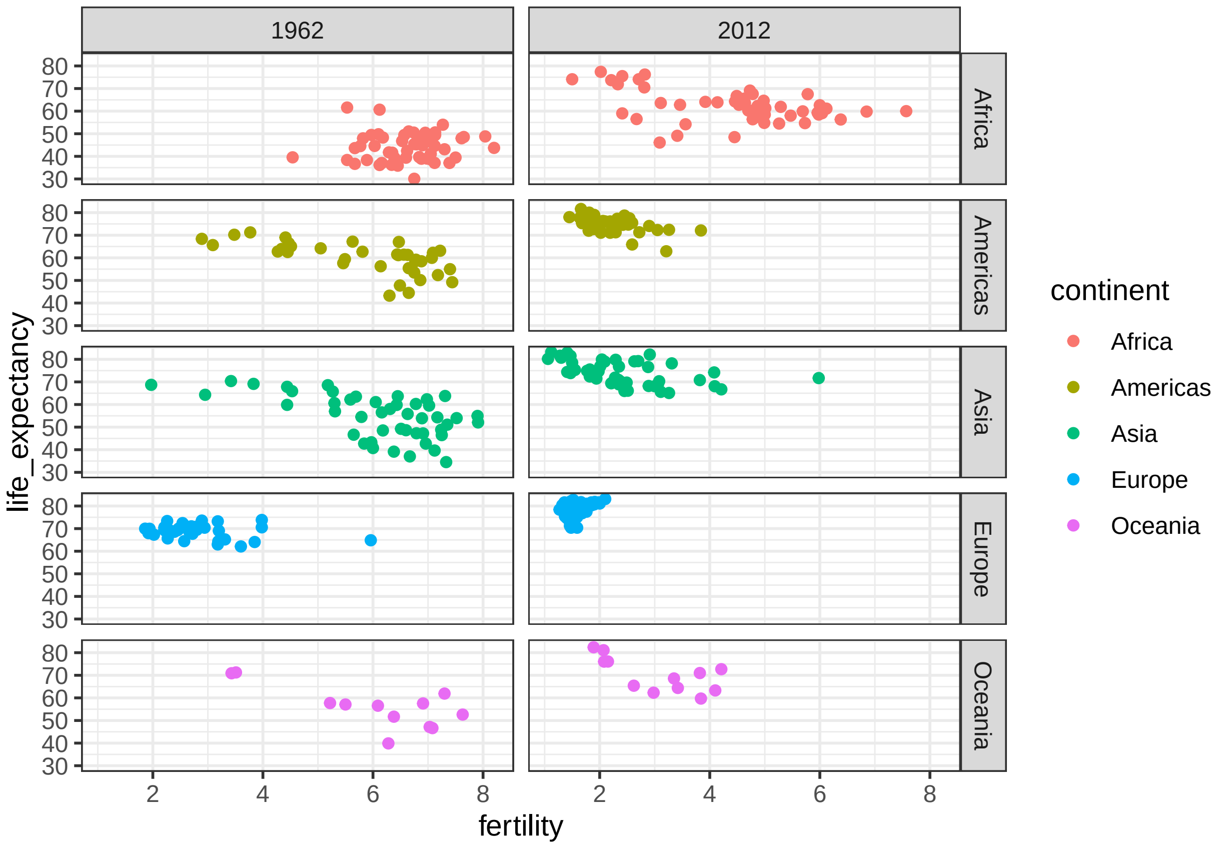
The textbook for this section is available here

Key points

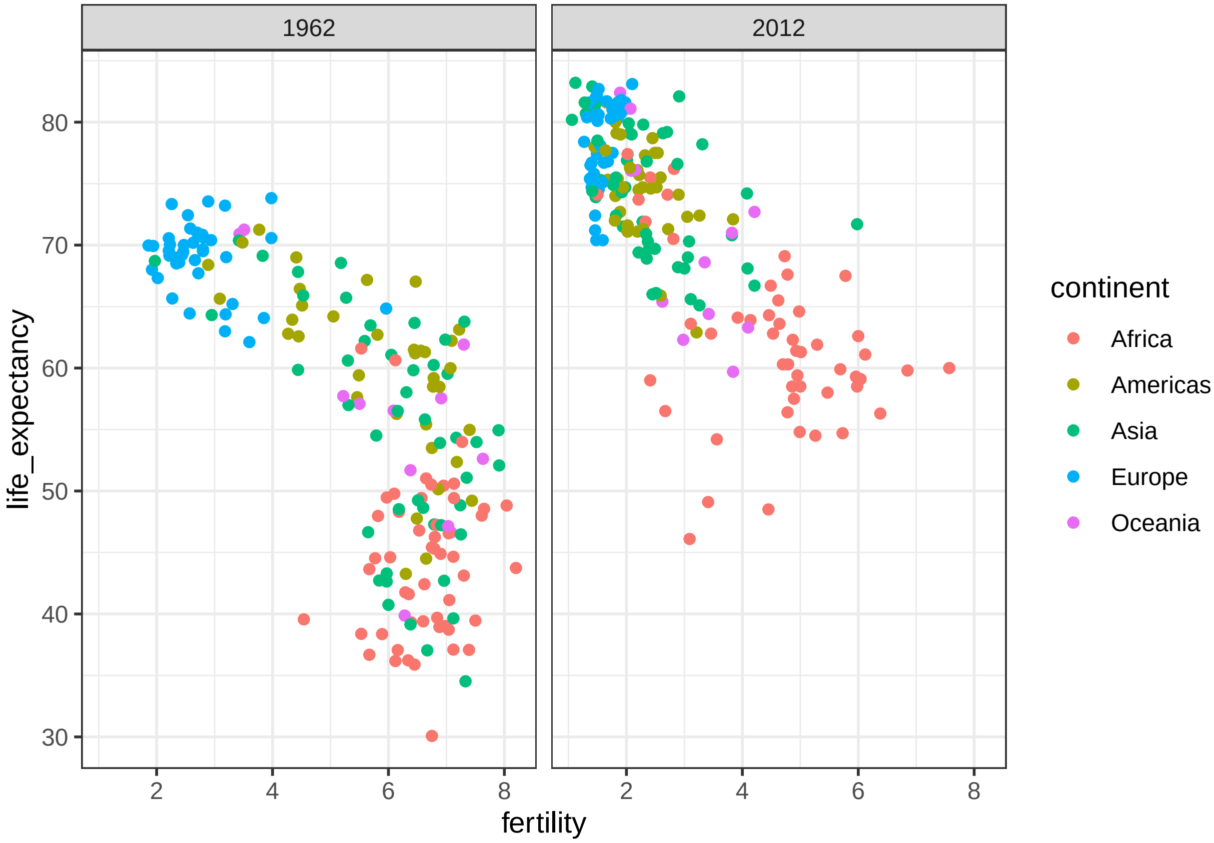
- Faceting makes multiple side-by-side plots stratified by some variable. This is a way to ease comparisons.
- The `facet_grid` function allows faceting by up to two variables, with rows faceted by one variable and columns faceted by the other variable. To facet by only one variable, use the dot operator as the other variable.
- The `facet_wrap` function facets by one variable and automatically wraps the series of plots so they have readable dimensions.
- Faceting keeps the axes fixed across all plots, easing comparisons between plots.
- The data suggest that the developing versus Western world view no longer makes sense in 2012.

Code

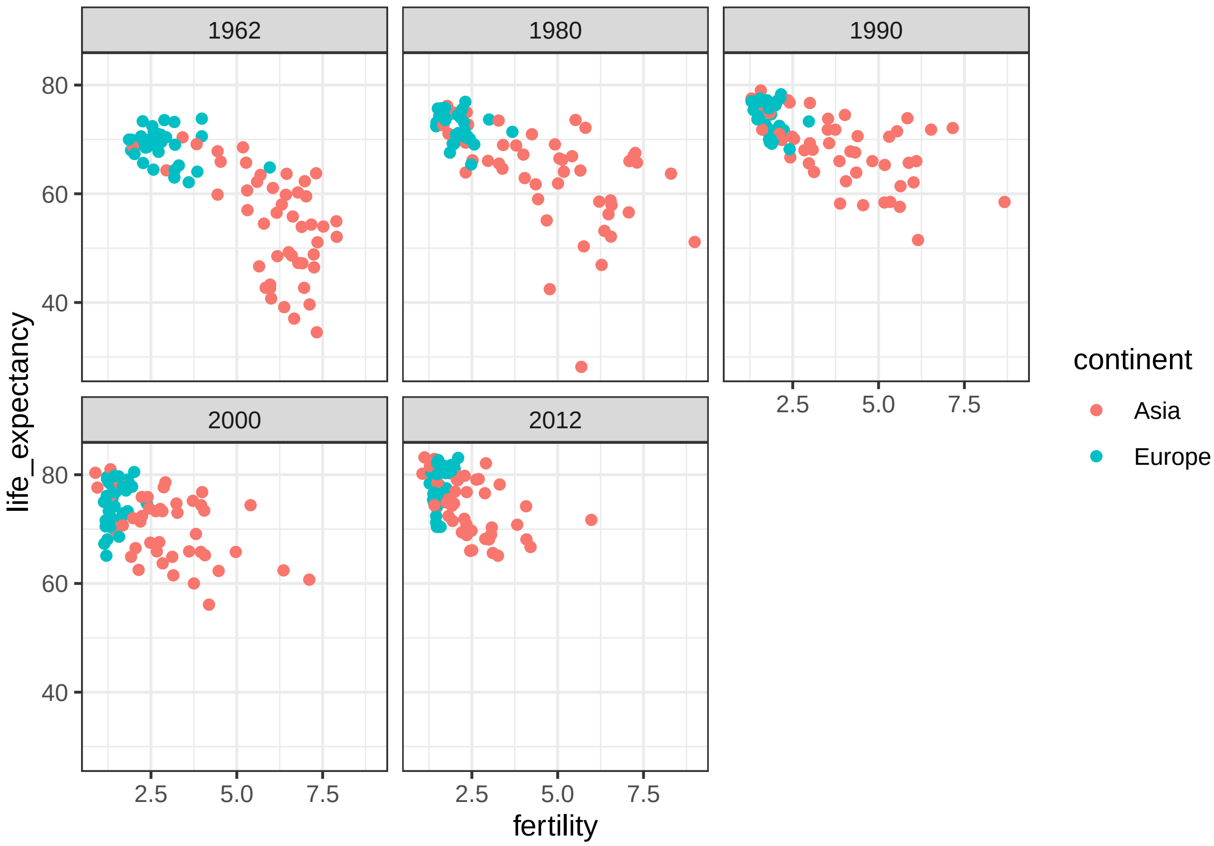
```
# facet by continent and year
filter(gapminder, year %in% c(1962, 2012)) %>%
    ggplot(aes(fertility, life_expectancy, col = continent)) +
    geom_point() +
    facet_grid(continent ~ year)
```




```r
# facet by year only
filter(gapminder, year %in% c(1962, 2012)) %>%
    ggplot(aes(fertility, life_expectancy, col = continent)) +
    geom_point() +
    facet_grid(. ~ year)
```




```
# facet by year, plots wrapped onto multiple rows
years <- c(1962, 1980, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
gapminder %>%
    filter(year %in% years & continent %in% continents) %>%
    ggplot(aes(fertility, life_expectancy, col = continent)) +
    geom_point() +
    facet_wrap(~year)
```



Time Series Plots

The textbook for this section is available here

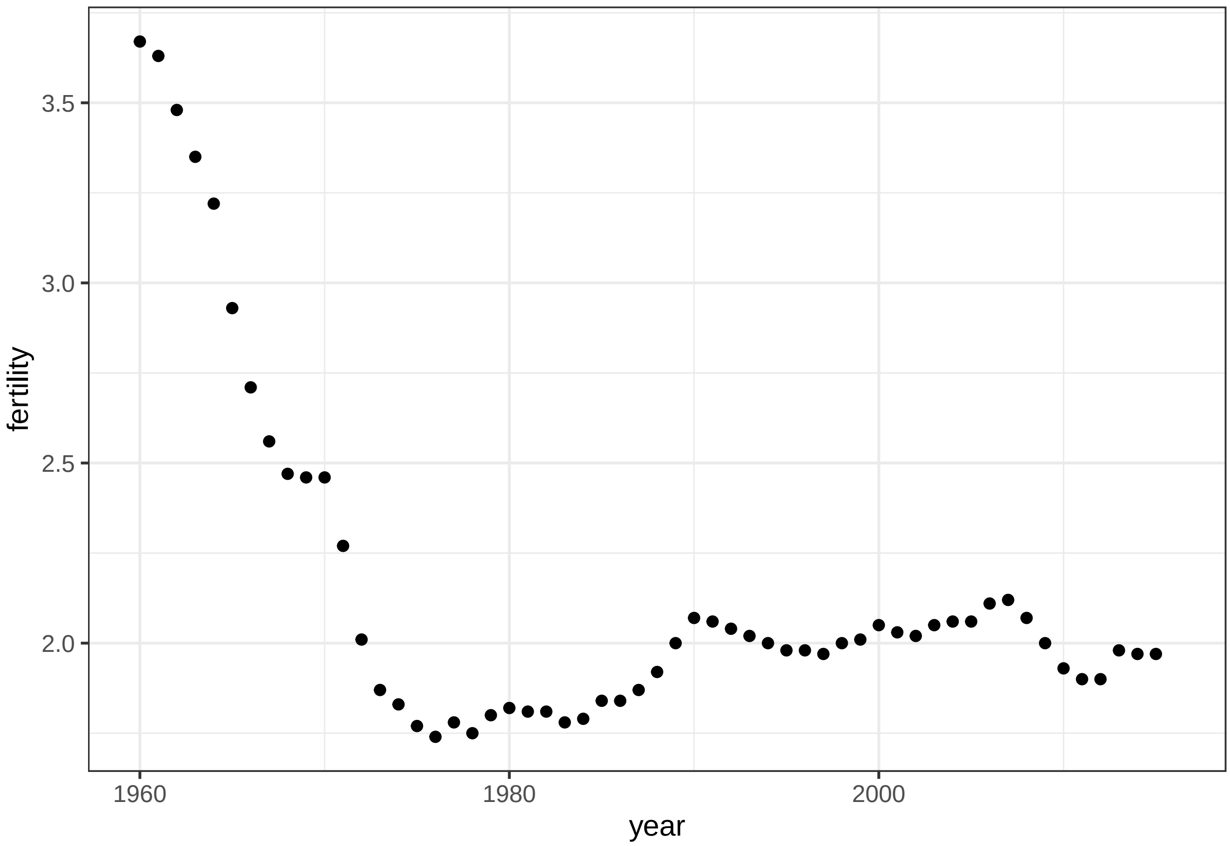
Key points

- Time series plots have time on the x-axis and a variable of interest on the y-axis.
- The `geom_line` geometry connects adjacent data points to form a continuous line. A line plot is appropriate when points are regularly spaced, densely packed and from a single data series.
- You can plot multiple lines on the same graph. Remember to group or color by a variable so that the lines are plotted independently.
- Labeling is usually preferred over legends. However, legends are easier to make and appear by default. Add a label with `geom_text`, specifying the coordinates where the label should appear on the graph.

Code: Single time series

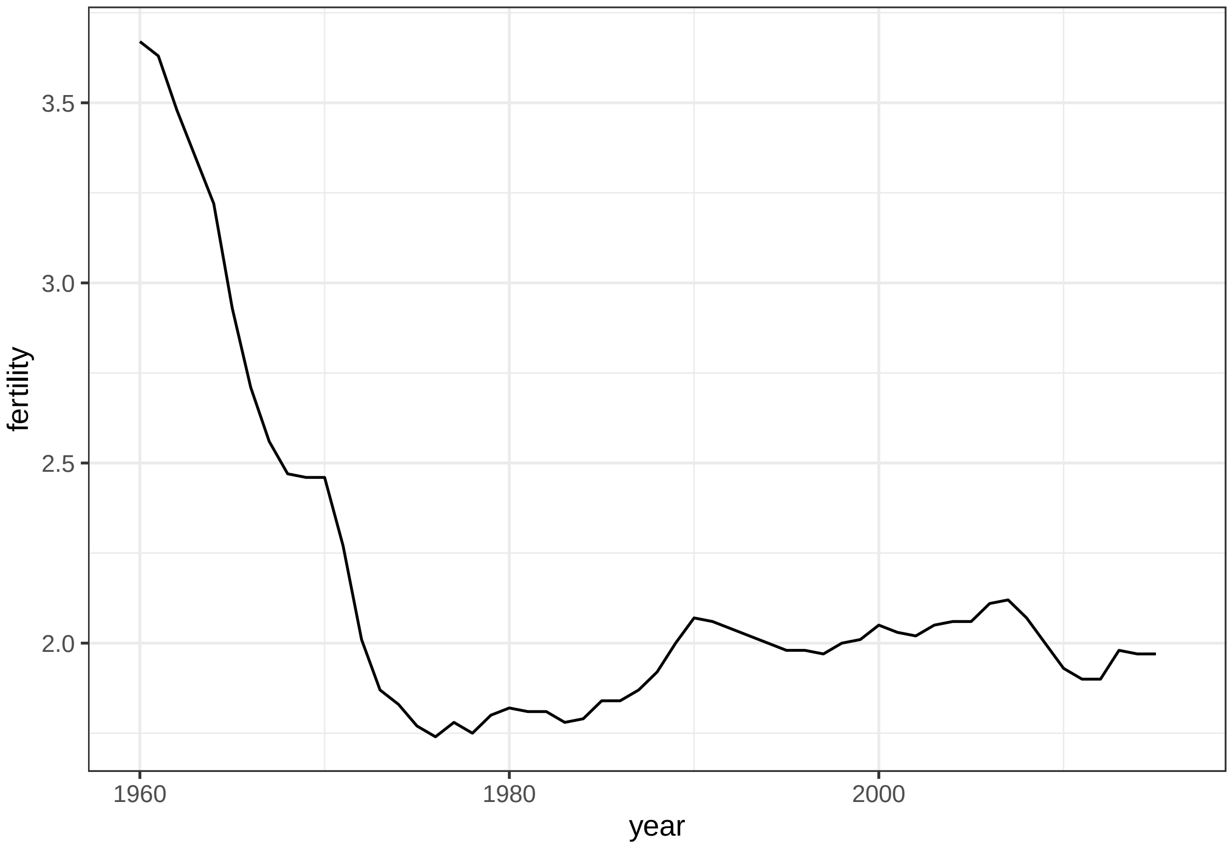
```
# scatterplot of US fertility by year
gapminder %>%
    filter(country == "United States") %>%
    ggplot(aes(year, fertility)) +
    geom_point()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




```
# line plot of US fertility by year
gapminder %>%
    filter(country == "United States") %>%
    ggplot(aes(year, fertility)) +
    geom_line()
```

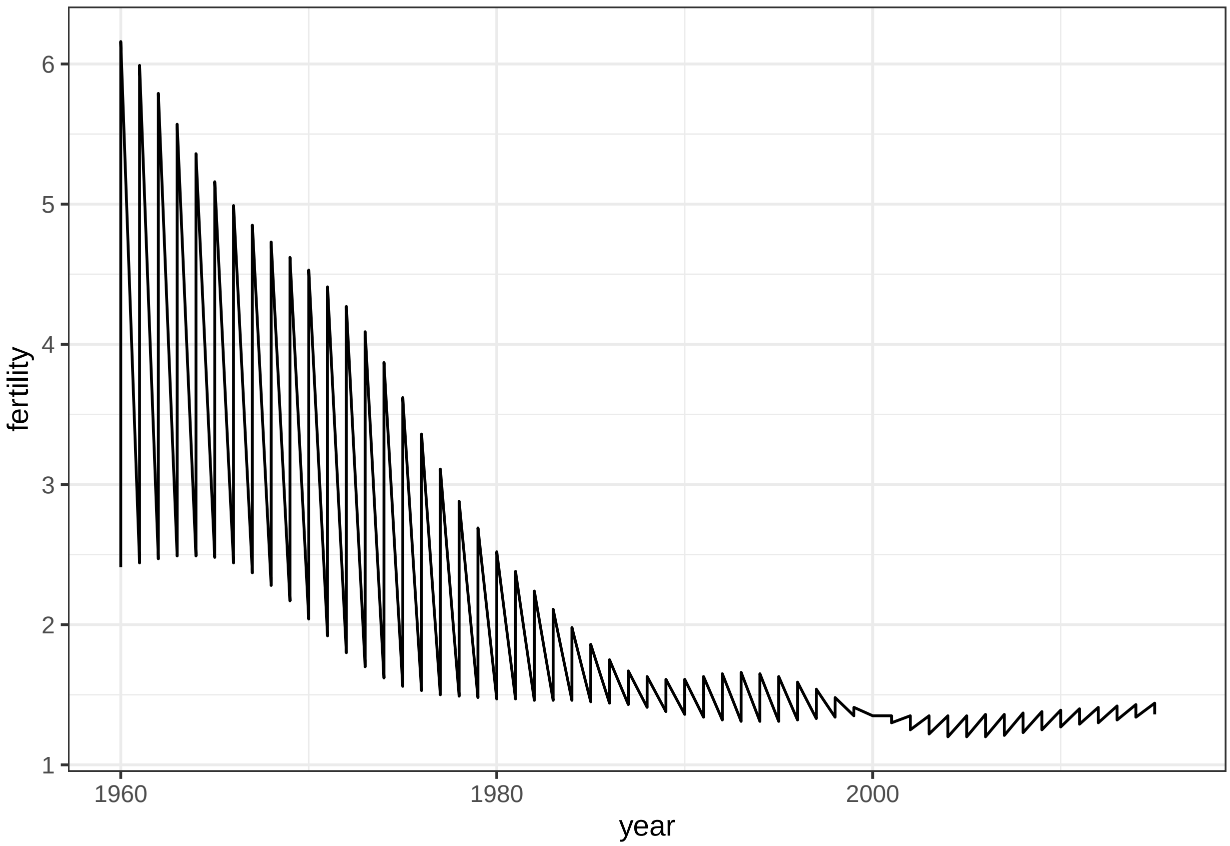
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Code: Multiple time series

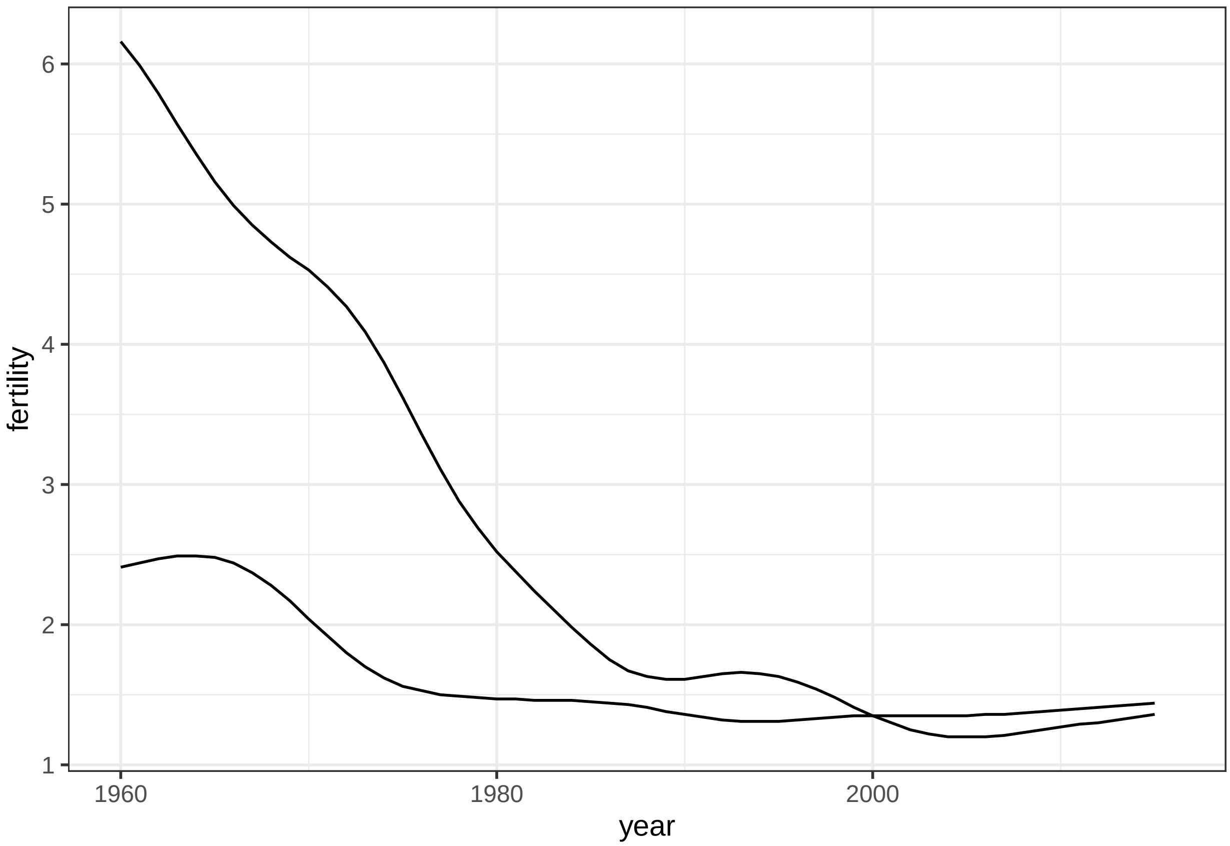
```
# line plot fertility time series for two countries- only one line (incorrect)
countries <- c("South Korea", "Germany")
gapminder %>% filter(country %in% countries) %>%
    ggplot(aes(year, fertility)) +
    geom_line()
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```



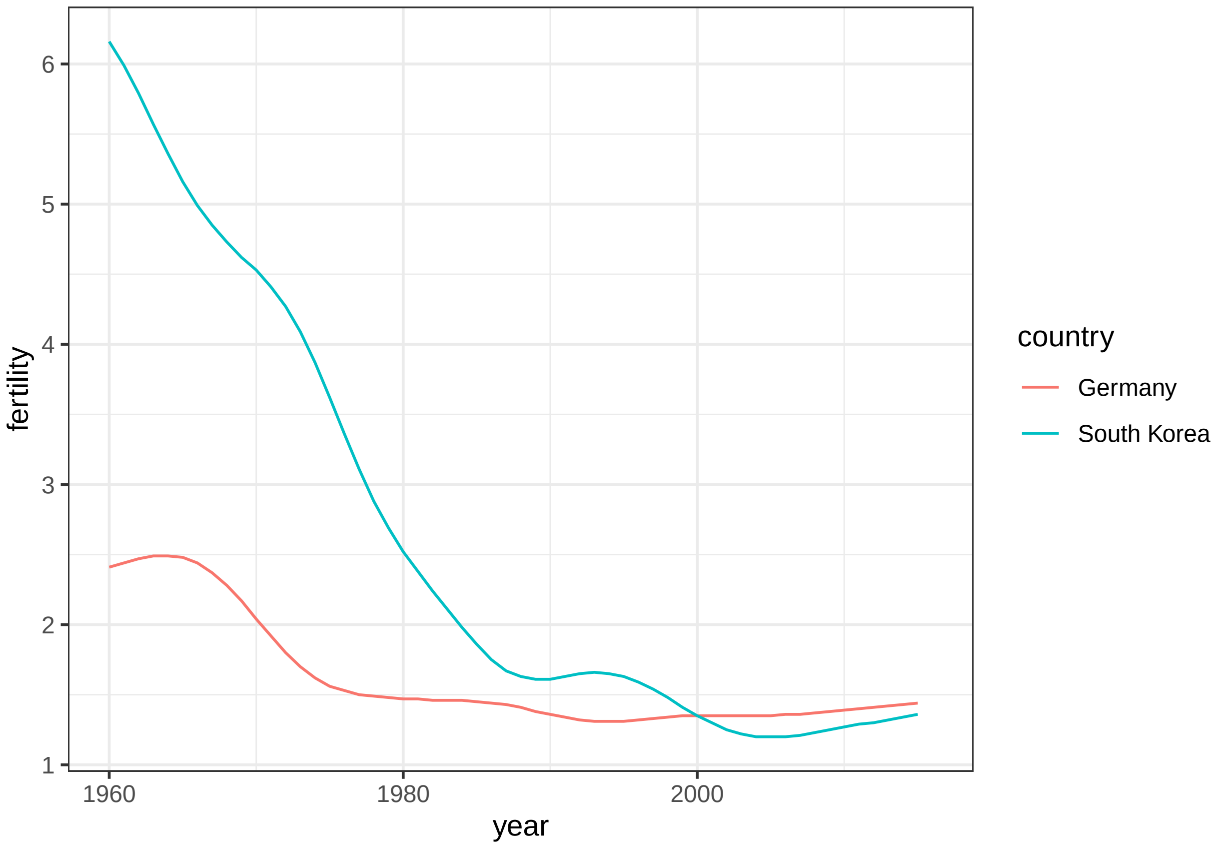

```
# line plot fertility time series for two countries - one line per country
gapminder %>% filter(country %in% countries) %>%
    ggplot(aes(year, fertility, group = country)) +
    geom_line()
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```



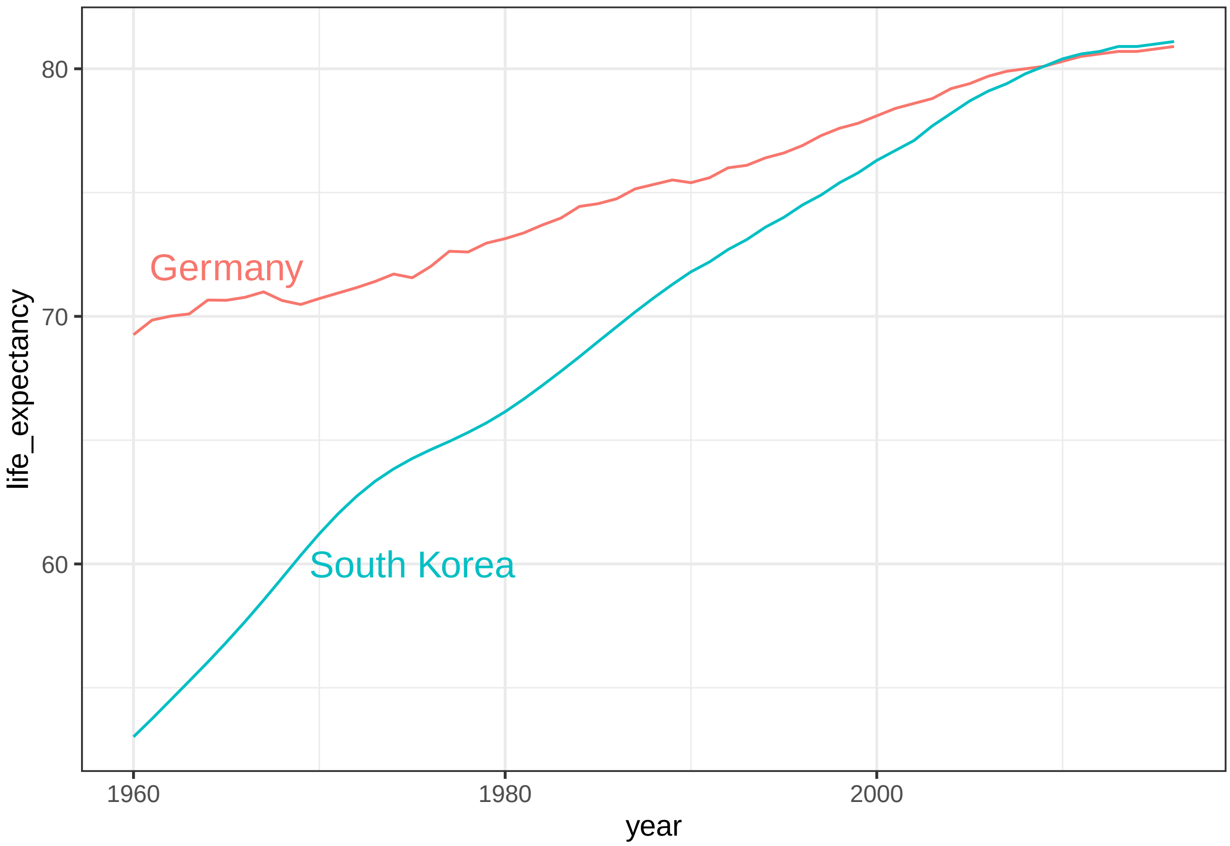

```r
# fertility time series for two countries - lines colored by country
gapminder %>% filter(country %in% countries) %>%
    ggplot(aes(year, fertility, col = country)) +
    geom_line()
```

```
## Warning: Removed 2 row(s) containing missing values (geom_path).
```



Code: Adding text labels to a plot

```
# life expectancy time series - lines colored by country and labeled, no legend
labels <- data.frame(country = countries, x = c(1975, 1965), y = c(60, 72))
gapminder %>% filter(country %in% countries) %>%
    ggplot(aes(year, life_expectancy, col = country)) +
    geom_line() +
    geom_text(data = labels, aes(x, y, label = country), size = 5) +
    theme(legend.position = "none")
```



Transformations

The textbook for this section is available here and here

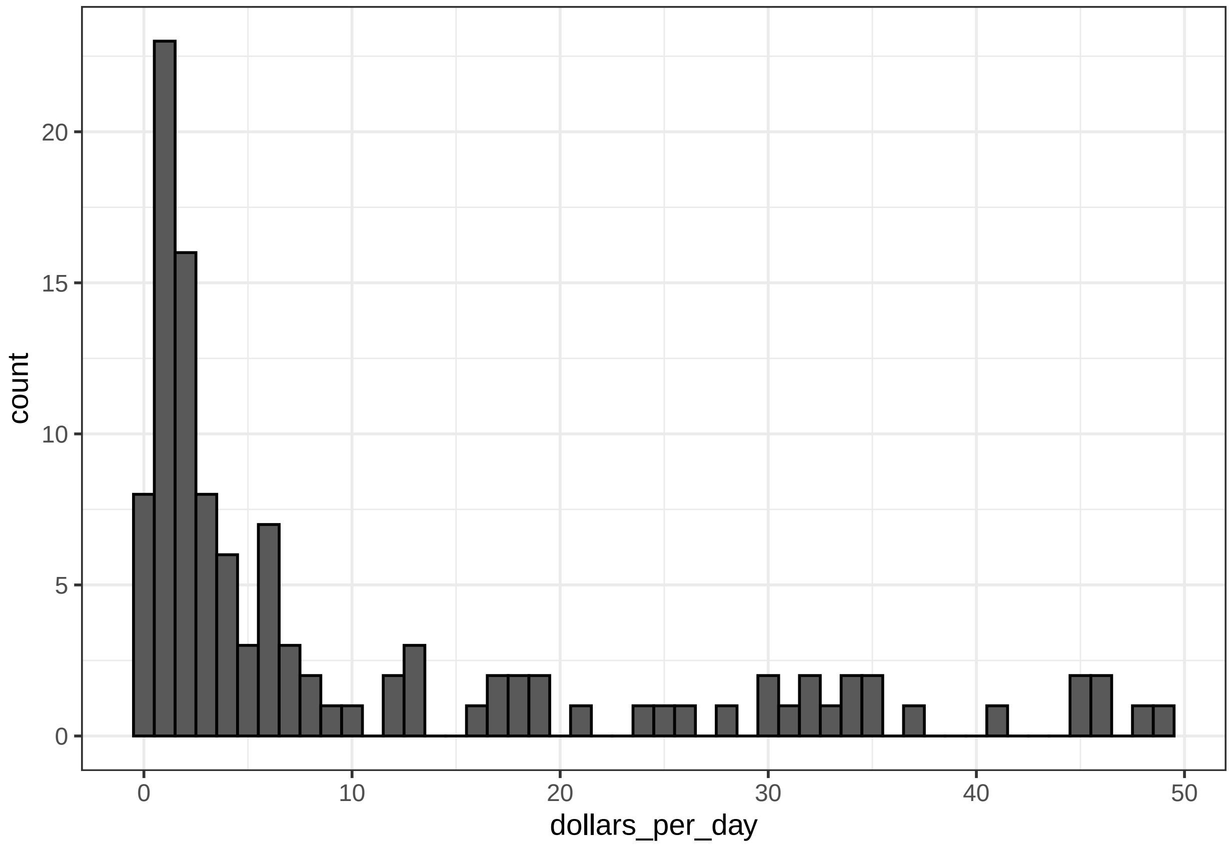
Key points

- We use GDP data to compute income in US dollars per day, adjusted for inflation.
- Log transformations convert multiplicative changes into additive changes.
- Common transformations are the log base 2 transformation and the log base 10 transformation. The choice of base depends on the range of the data. The natural log is not recommended for visualization because it is difficult to interpret.
- The mode of a distribution is the value with the highest frequency. The mode of a normal distribution is the average. A distribution can have multiple local modes.
- There are two ways to use log transformations in plots: transform the data before plotting or transform the axes of the plot. Log scales have the advantage of showing the original values as axis labels, while log transformed values ease interpretation of intermediate values between labels.
- Scale the x-axis using `scale_x_continuous` or `scale_x_log10` layers in ggplot2. Similar functions exist for the y-axis.
- In 1970, income distribution is bimodal, consistent with the dichotomous Western versus developing worldview.

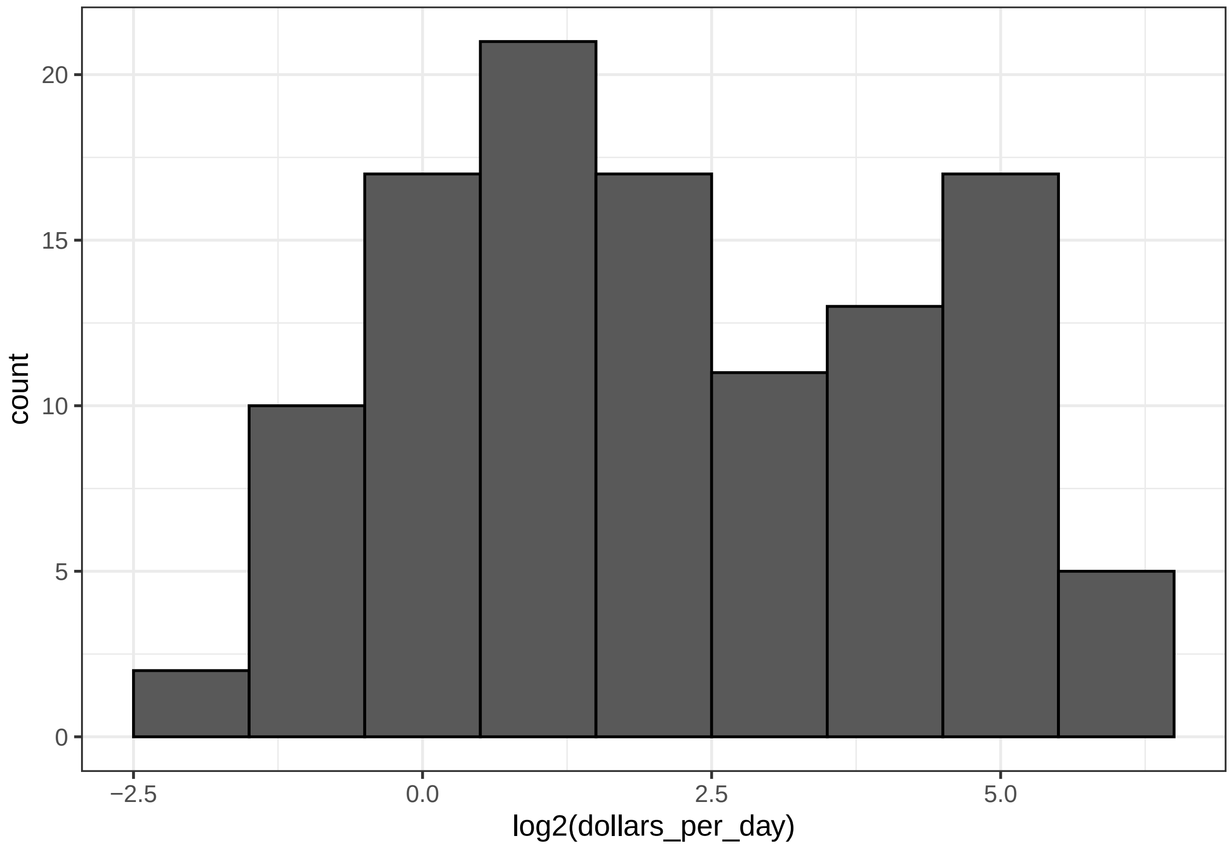
Code

```
# add dollars per day variable
gapminder <- gapminder %>%
    mutate(dollars_per_day = gdp/population/365)
```

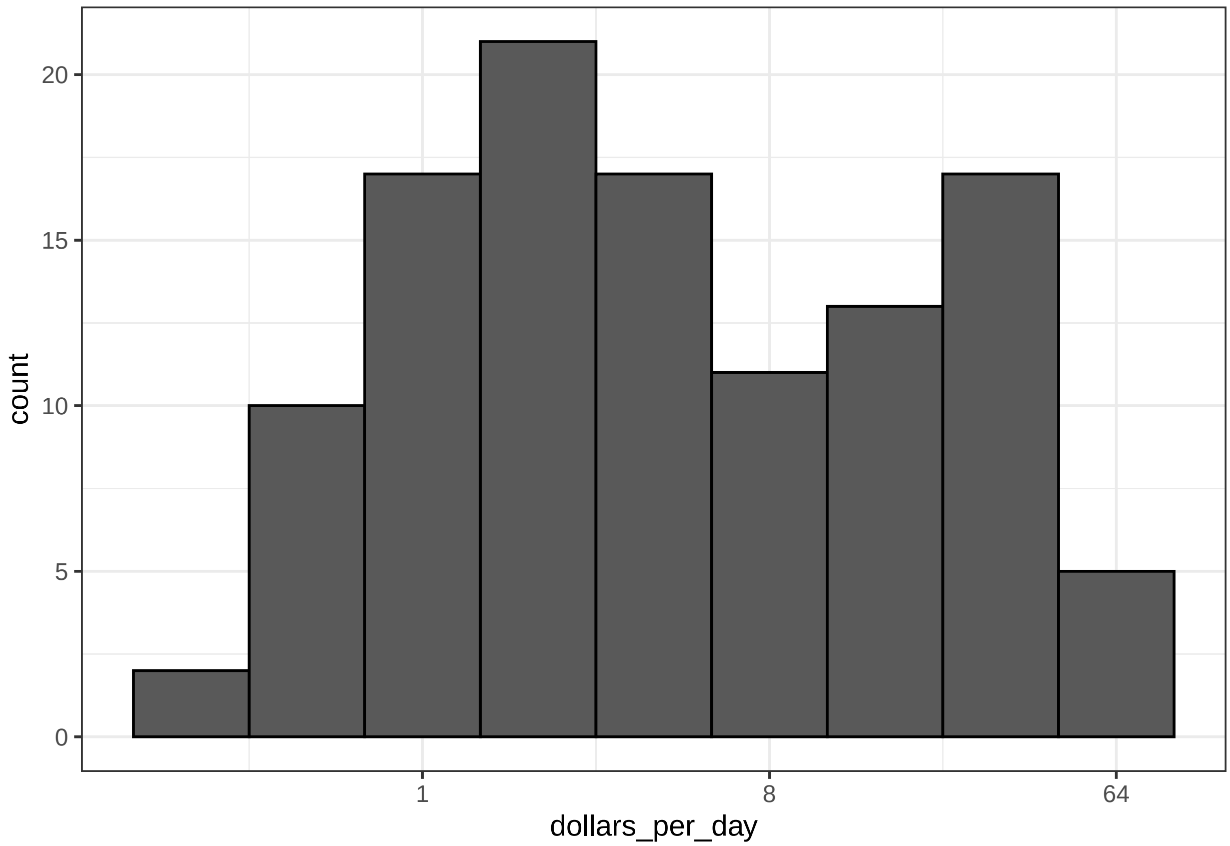
```r
# histogram of dollars per day
past_year <- 1970
gapminder %>%
    filter(year == past_year & !is.na(gdp)) %>%
    ggplot(aes(dollars_per_day)) +
    geom_histogram(binwidth = 1, color = "black")
```




```r
# repeat histogram with log2 scaled data
gapminder %>%
    filter(year == past_year & !is.na(gdp)) %>%
    ggplot(aes(log2(dollars_per_day))) +
    geom_histogram(binwidth = 1, color = "black")
```




```r
# repeat histogram with log2 scaled x-axis
gapminder %>%
    filter(year == past_year & !is.na(gdp)) %>%
    ggplot(aes(dollars_per_day)) +
    geom_histogram(binwidth = 1, color = "black") +
    scale_x_continuous(trans = "log2")
```



Stratify and Boxplot

The textbook for this section is available here. Note that many boxplots from the video are instead dot plots in the textbook and that a different boxplot is constructed in the textbook. Also read that section to see an example of grouping factors with the `case_when` function.

Key points

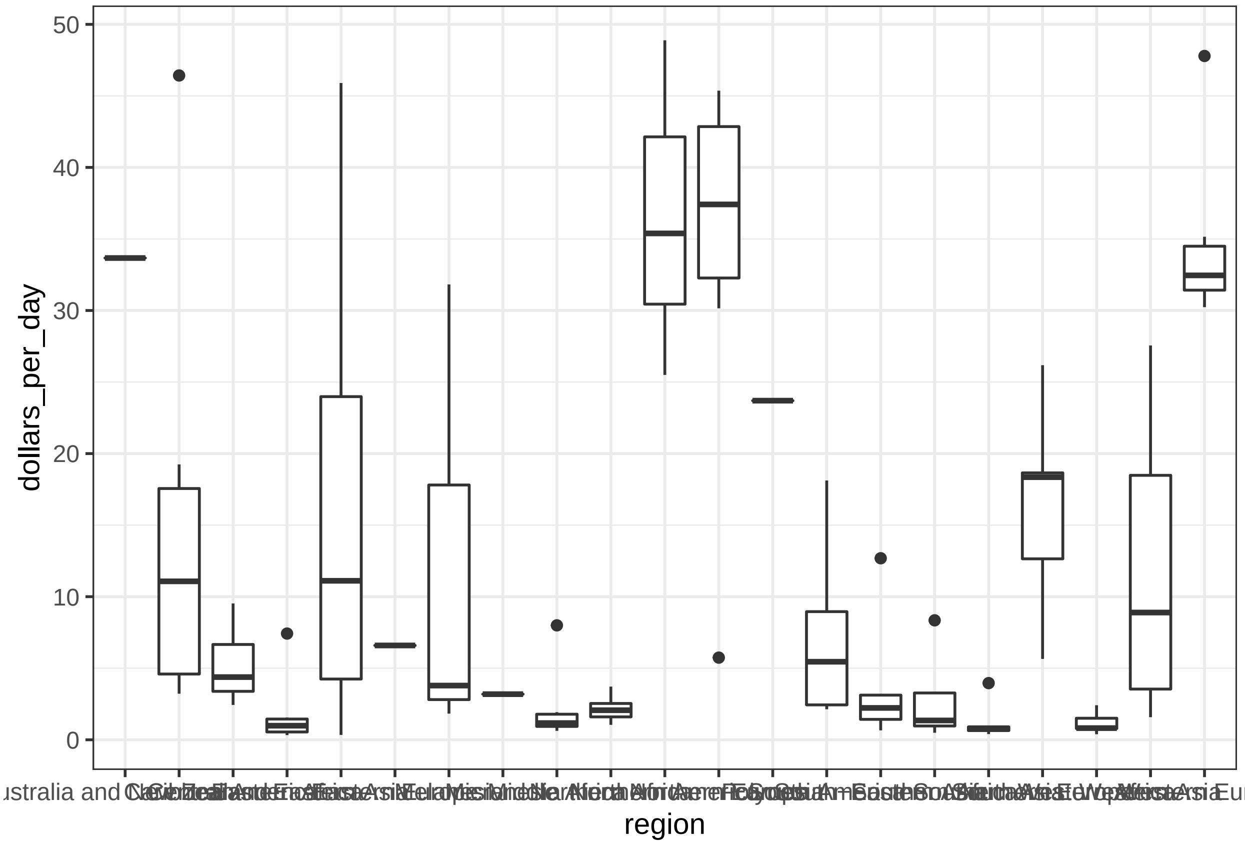
- Make boxplots stratified by a categorical variable using the `geom_boxplot` geometry.
- Rotate axis labels by changing the theme through `element_text`. You can change the angle and justification of the text labels.
- Consider ordering your factors by a meaningful value with the `reorder` function, which changes the order of factor levels based on a related numeric vector. This is a way to ease comparisons.
- Show the data by adding data points to the boxplot with a `geom_point` layer. This adds information beyond the five-number summary to your plot, but too many data points it can obfuscate your message.

Code: Boxplot of GDP by region

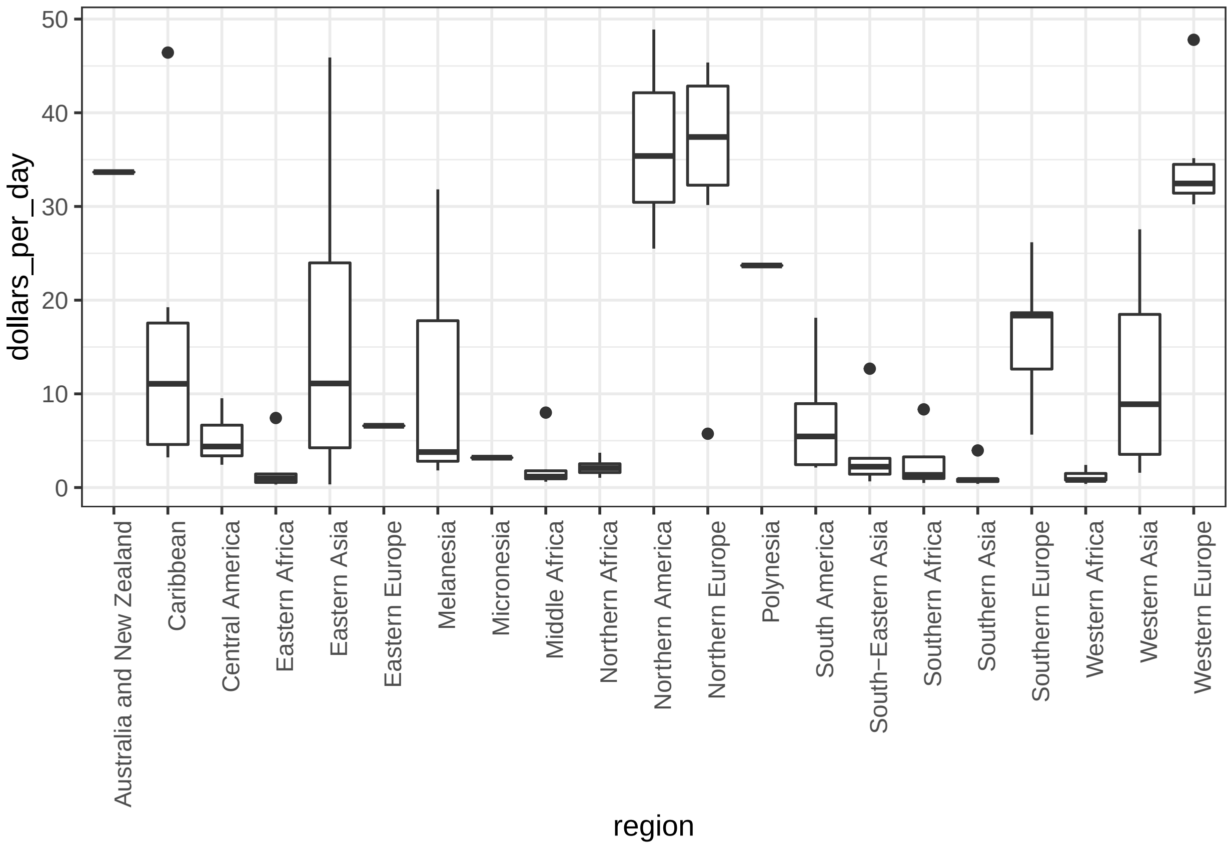
```
# add dollars per day variable
gapminder <- gapminder %>%
    mutate(dollars_per_day = gdp/population/365)

# number of regions
length(levels(gapminder$region))
```

```
## [1] 22
```

```r
# boxplot of GDP by region in 1970
past_year <- 1970
p <- gapminder %>%
    filter(year == past_year & !is.na(gdp)) %>%
    ggplot(aes(region, dollars_per_day))
p + geom_boxplot()
```




```r
# rotate names on x-axis
p + geom_boxplot() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Code: The reorder function

```
# by default, factor order is alphabetical
fac <- factor(c("Asia", "Asia", "West", "West", "West"))
levels(fac)
```

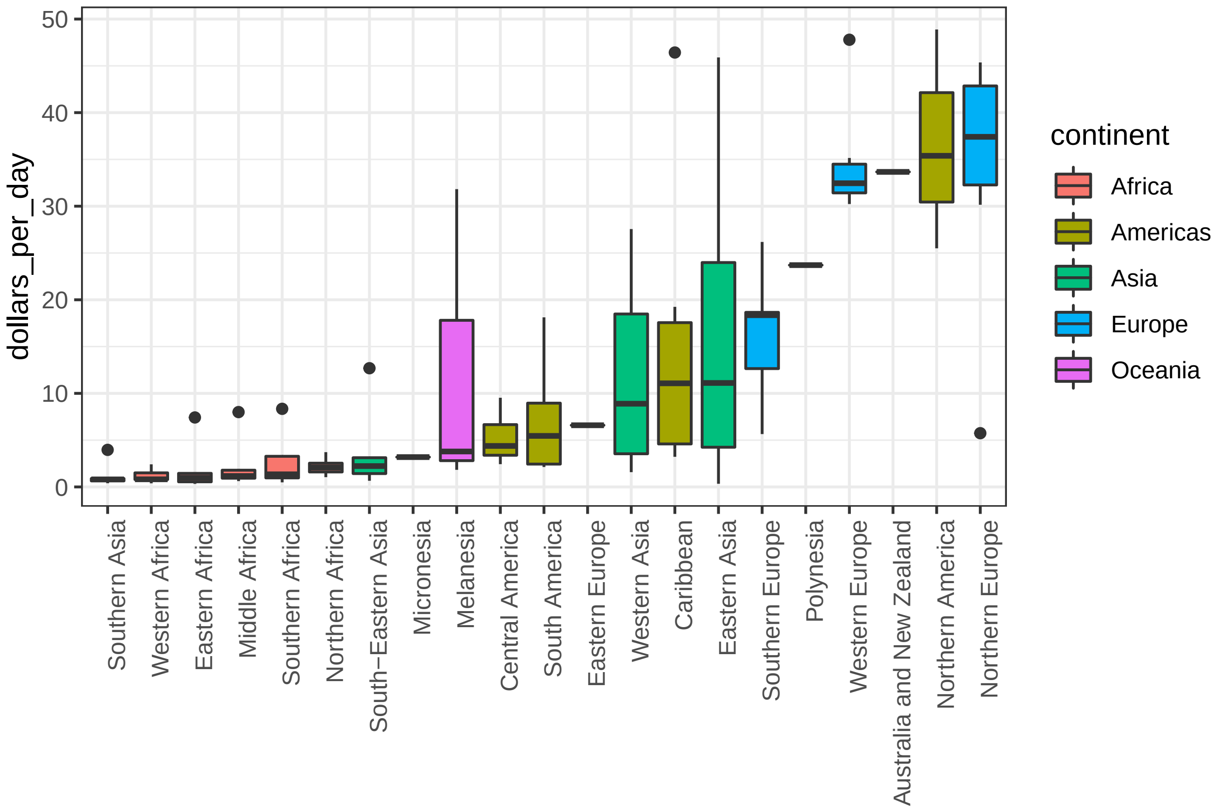
```
## [1] "Asia" "West"
```

```
# reorder factor by the category means
value <- c(10, 11, 12, 6, 4)
fac <- reorder(fac, value, FUN = mean)
levels(fac)
```

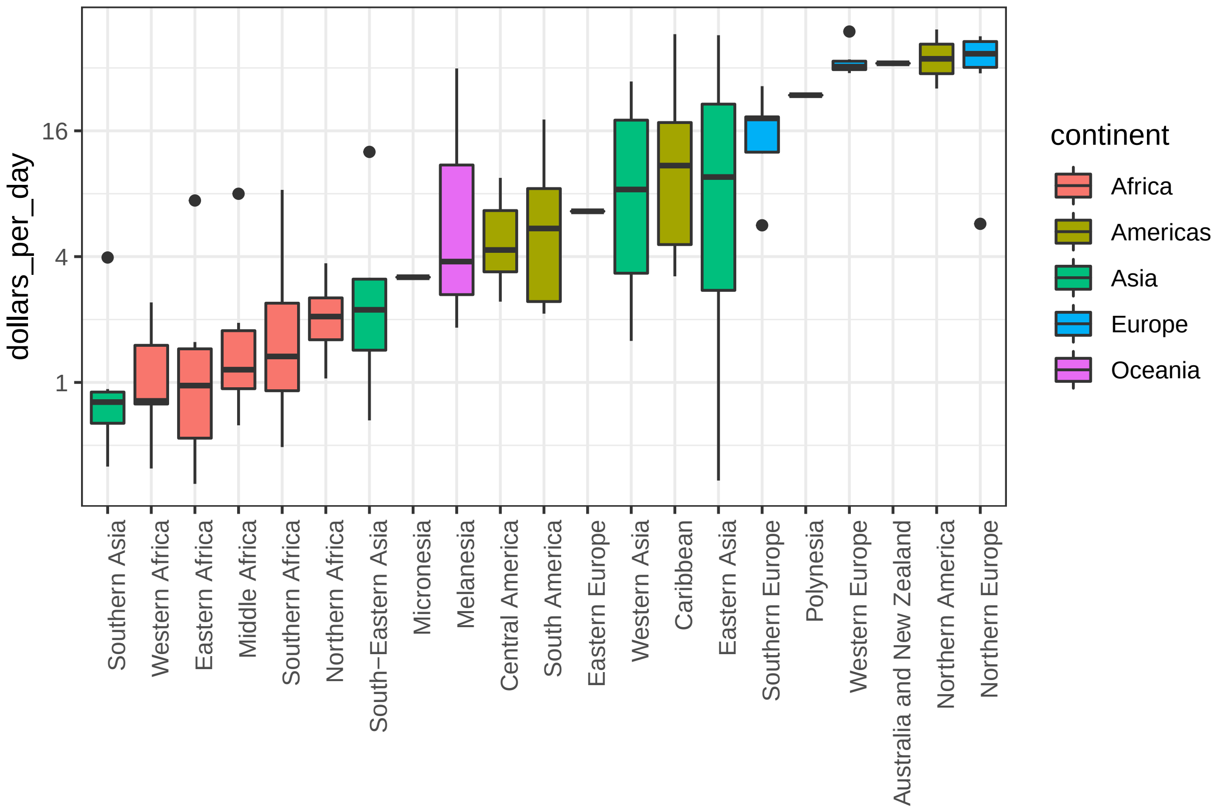
```
## [1] "West" "Asia"
```

Code: Enhanced boxplot ordered by median income, scaled, and showing data

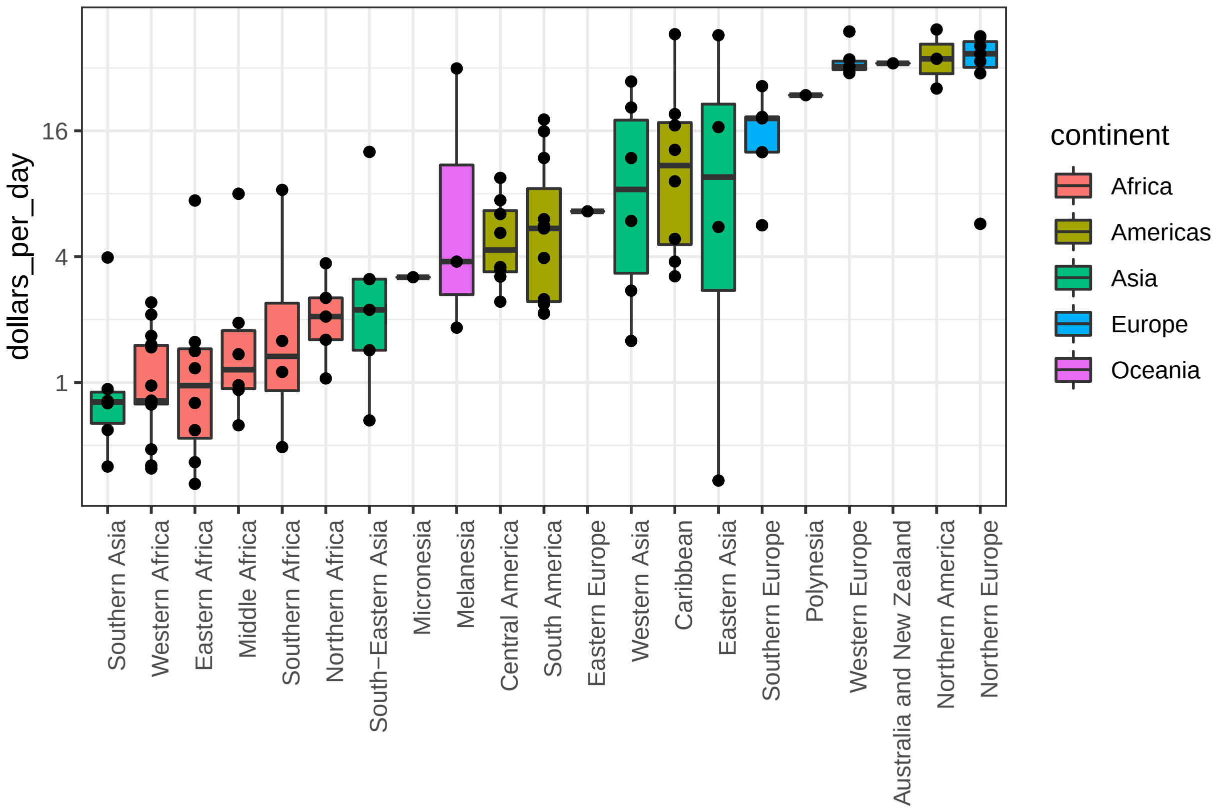
```
# reorder by median income and color by continent
p <- gapminder %>%
    filter(year == past_year & !is.na(gdp)) %>%
    mutate(region = reorder(region, dollars_per_day, FUN = median)) %>%    # reorder
    ggplot(aes(region, dollars_per_day, fill = continent)) +    # color by continent
    geom_boxplot() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    xlab("")
p
```




```
# log2 scale y-axis
p + scale_y_continuous(trans = "log2")
```




```r
# add data points
p + scale_y_continuous(trans = "log2") + geom_point(show.legend = FALSE)
```



Comparing Distributions

The textbook for this section is available here. Note that the boxplots are slightly different.

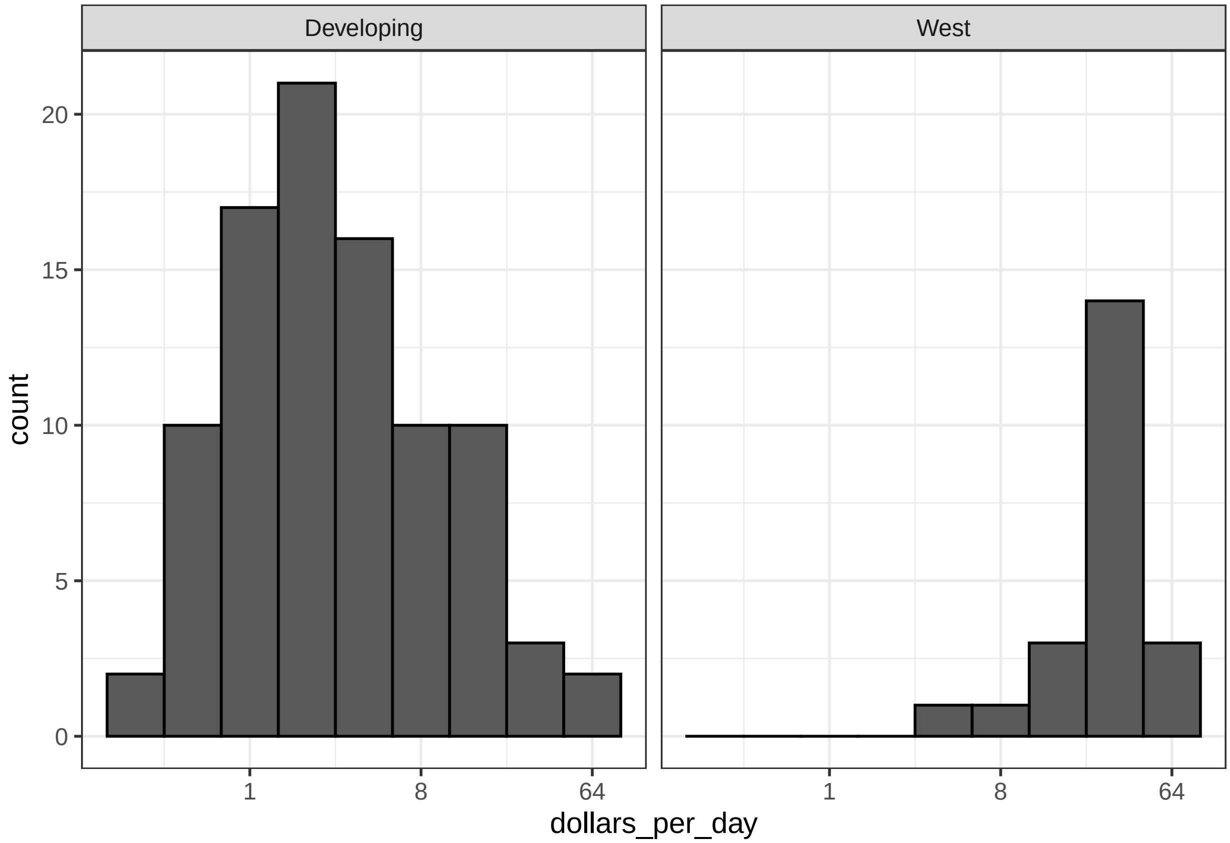
Key points

- Use `intersect` to find the overlap between two vectors.
- To make boxplots where grouped variables are adjacaent, color the boxplot by a factor instead of faceting by that factor. This is a way to ease comparisons.
- The data suggest that the income gap between rich and poor countries has narrowed, not expanded.

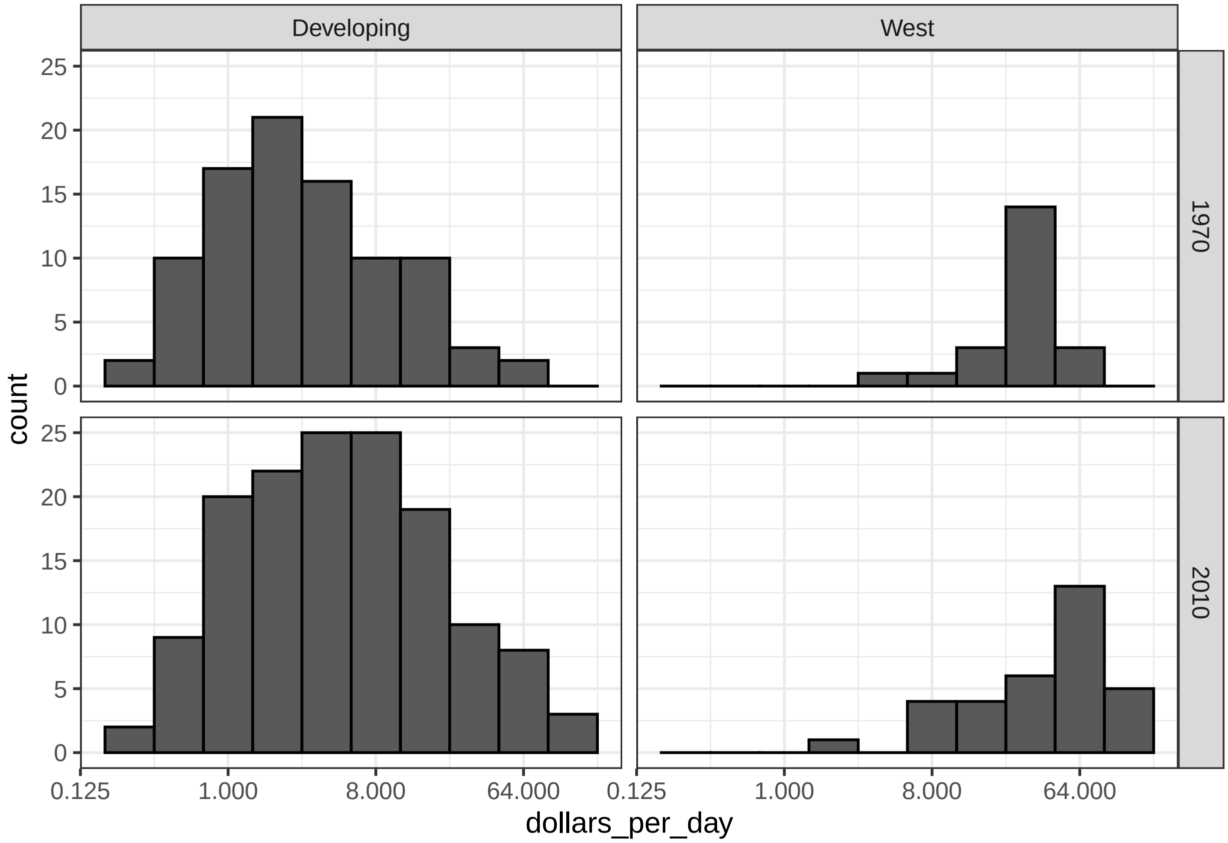
Code: Histogram of income in West versus developing world, 1970 and 2010

```
# add dollars per day variable and define past year
gapminder <- gapminder %>%
    mutate(dollars_per_day = gdp/population/365)
past_year <- 1970

# define Western countries
west <- c("Western Europe", "Northern Europe", "Southern Europe", "Northern America", "Australia and Ne

# facet by West vs devloping
gapminder %>%
    filter(year == past_year & !is.na(gdp)) %>%
    mutate(group = ifelse(region %in% west, "West", "Developing")) %>%
    ggplot(aes(dollars_per_day)) +
    geom_histogram(binwidth = 1, color = "black") +
```

```r
    scale_x_continuous(trans = "log2") +
    facet_grid(. ~ group)
```

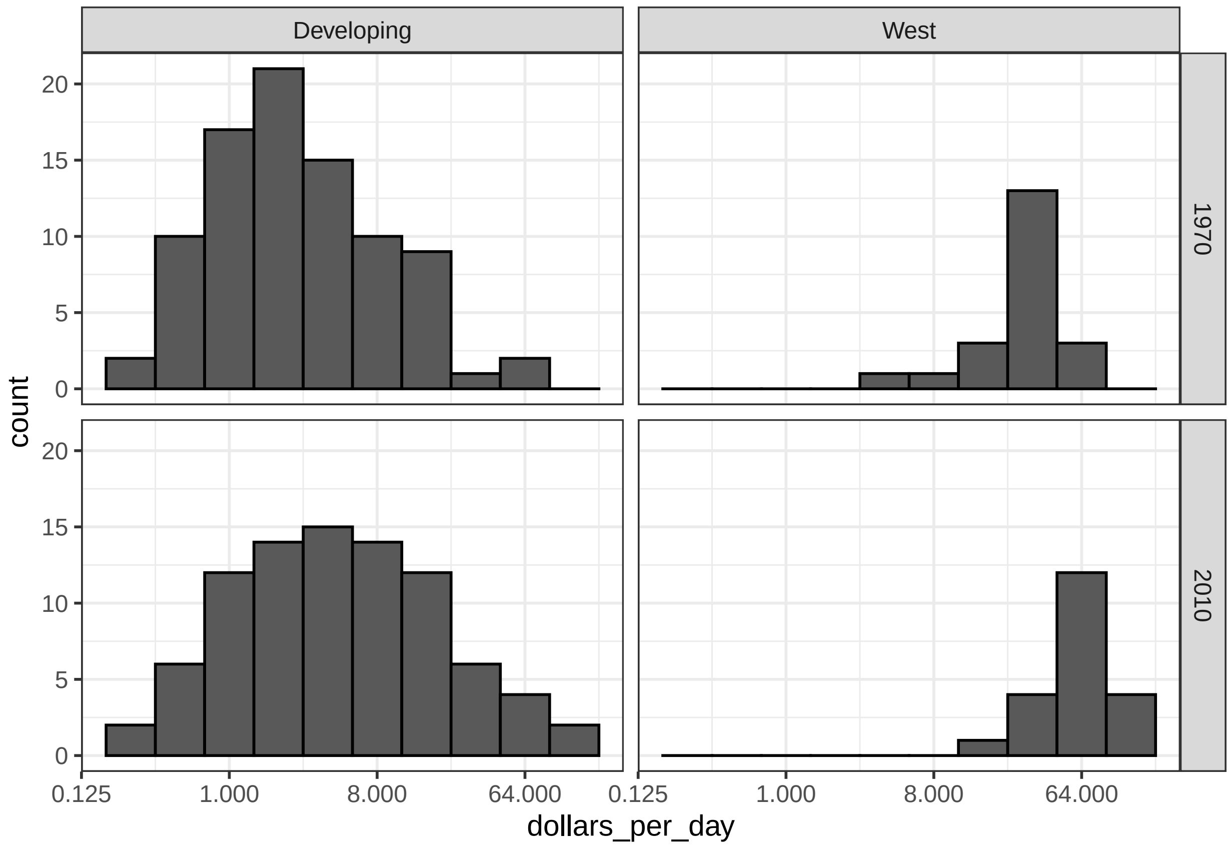



```r
# facet by West/developing and year
present_year <- 2010
gapminder %>%
    filter(year %in% c(past_year, present_year) & !is.na(gdp)) %>%
    mutate(group = ifelse(region %in% west, "West", "Developing")) %>%
    ggplot(aes(dollars_per_day)) +
    geom_histogram(binwidth = 1, color = "black") +
    scale_x_continuous(trans = "log2") +
    facet_grid(year ~ group)
```



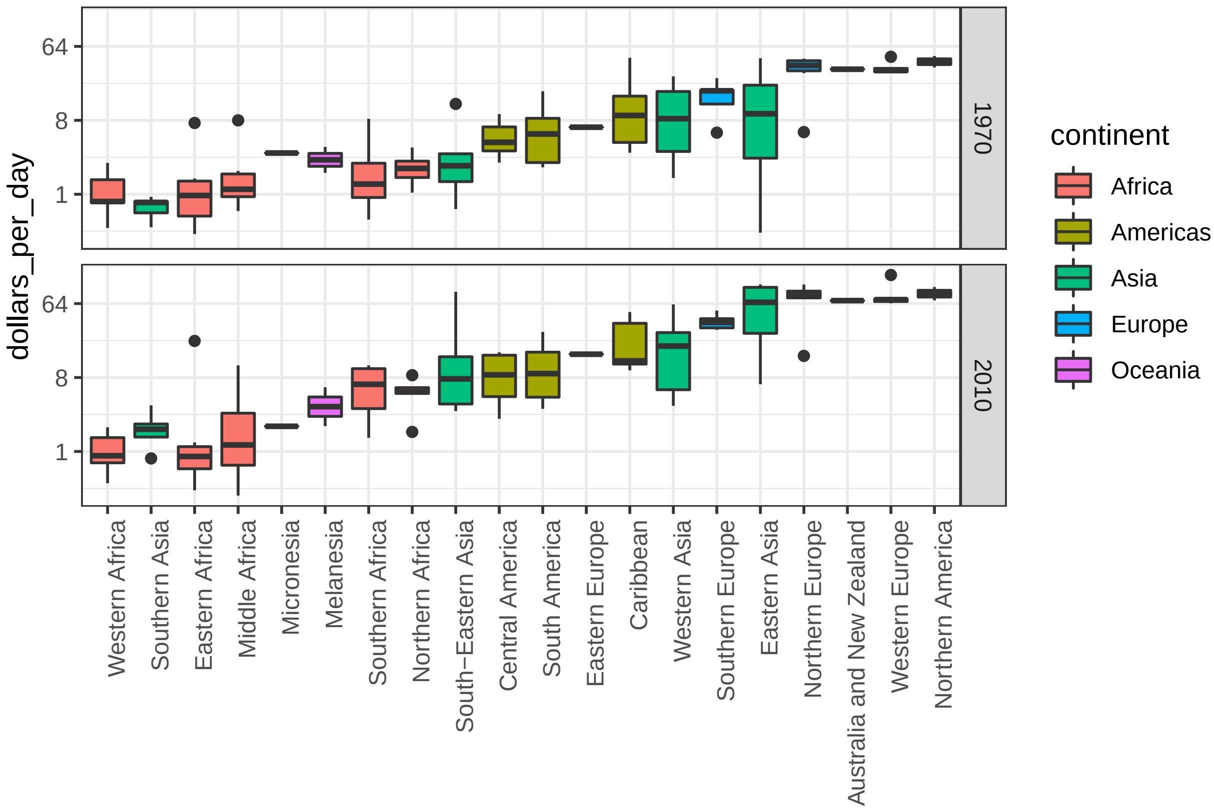
Code: Income distribution of West versus developing world, only countries with data

```
# define countries that have data available in both years
country_list_1 <- gapminder %>%
    filter(year == past_year & !is.na(dollars_per_day)) %>% .$country
country_list_2 <- gapminder %>%
    filter(year == present_year & !is.na(dollars_per_day)) %>% .$country
country_list <- intersect(country_list_1, country_list_2)

# make histogram including only countries with data available in both years
gapminder %>%
    filter(year %in% c(past_year, present_year) & country %in% country_list) %>%    # keep only selecte
    mutate(group = ifelse(region %in% west, "West", "Developing")) %>%
    ggplot(aes(dollars_per_day)) +
    geom_histogram(binwidth = 1, color = "black") +
    scale_x_continuous(trans = "log2") +
    facet_grid(year ~ group)
```

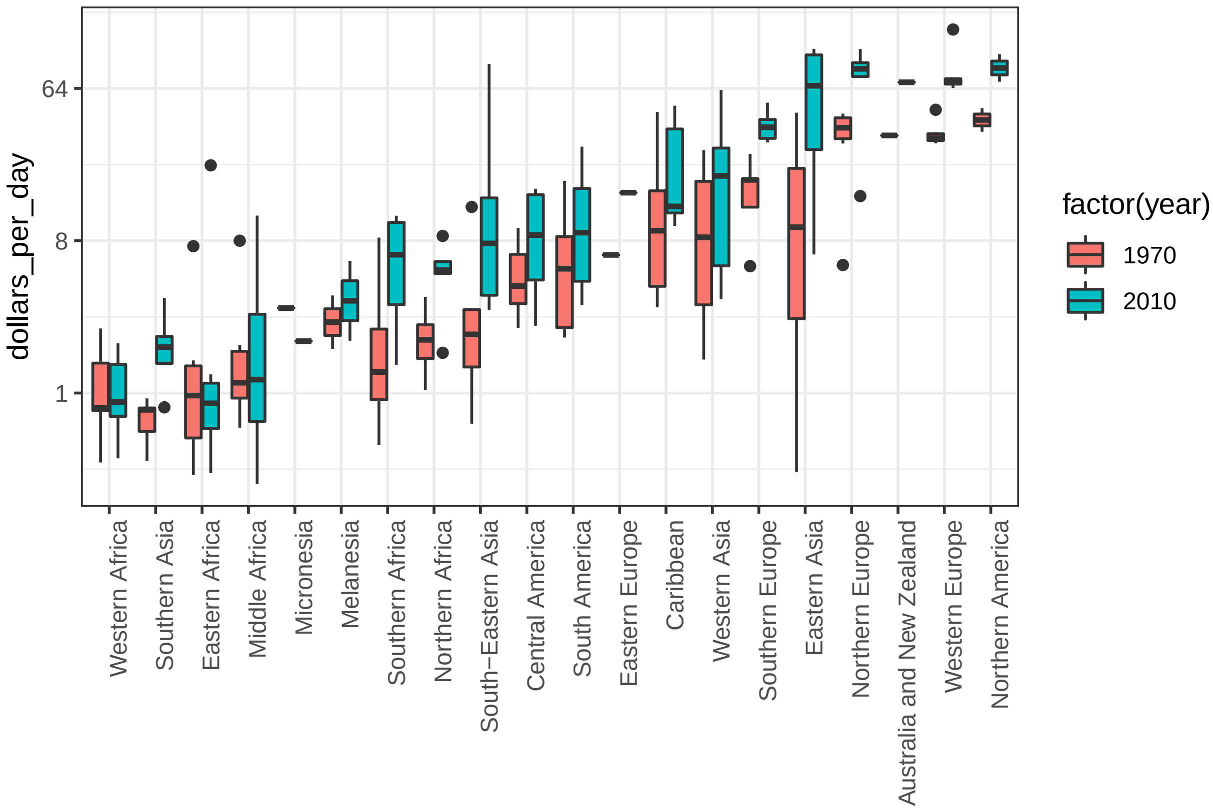


Code: Boxplots of income in West versus developing world, 1970 and 2010

```
p <- gapminder %>%
    filter(year %in% c(past_year, present_year) & country %in% country_list) %>%
    mutate(region = reorder(region, dollars_per_day, FUN = median)) %>%
    ggplot() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
    xlab("") + scale_y_continuous(trans = "log2")
p + geom_boxplot(aes(region, dollars_per_day, fill = continent)) +
    facet_grid(year ~ .)
```




```r
# arrange matching boxplots next to each other, colored by year
p + geom_boxplot(aes(region, dollars_per_day, fill = factor(year)))
```



Density Plots

The textbook for this section is available:

- 1970 versus 2010 income distributions
- Accessing computed variables
- Weighted densities

Key points

- Change the y-axis of density plots to variable counts using `..count..` as the y argument.
- The `case_when` function defines a factor whose levels are defined by a variety of logical operations to group data.
- Plot stacked density plots using `position="stack"`.
- Define a weight `aesthetic` mapping to change the relative weights of density plots - for example, this allows weighting of plots by population rather than number of countries.

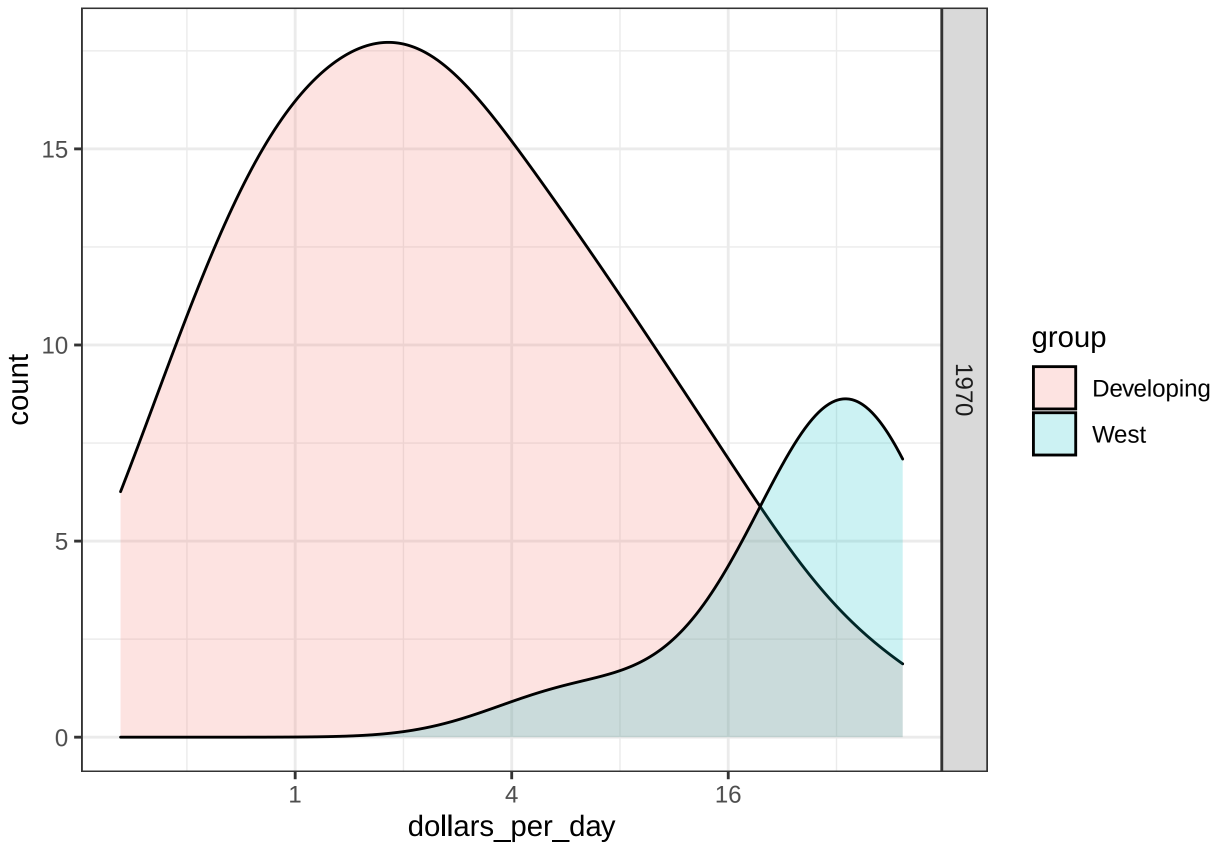
Code: Faceted smooth density plots

```
# smooth density plots - area under each curve adds to 1
gapminder %>%
    filter(year == past_year & country %in% country_list) %>%
    mutate(group = ifelse(region %in% west, "West", "Developing")) %>% group_by(group) %>%
    summarize(n = n()) %>% knitr::kable()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

group	n
Developing	87
West	21

```r
# smooth density plots - variable counts on y-axis
p <- gapminder %>%
    filter(year == past_year & country %in% country_list) %>%
    mutate(group = ifelse(region %in% west, "West", "Developing")) %>%
    ggplot(aes(dollars_per_day, y = ..count.., fill = group)) +
    scale_x_continuous(trans = "log2")
p + geom_density(alpha = 0.2, bw = 0.75) + facet_grid(year ~ .)
```

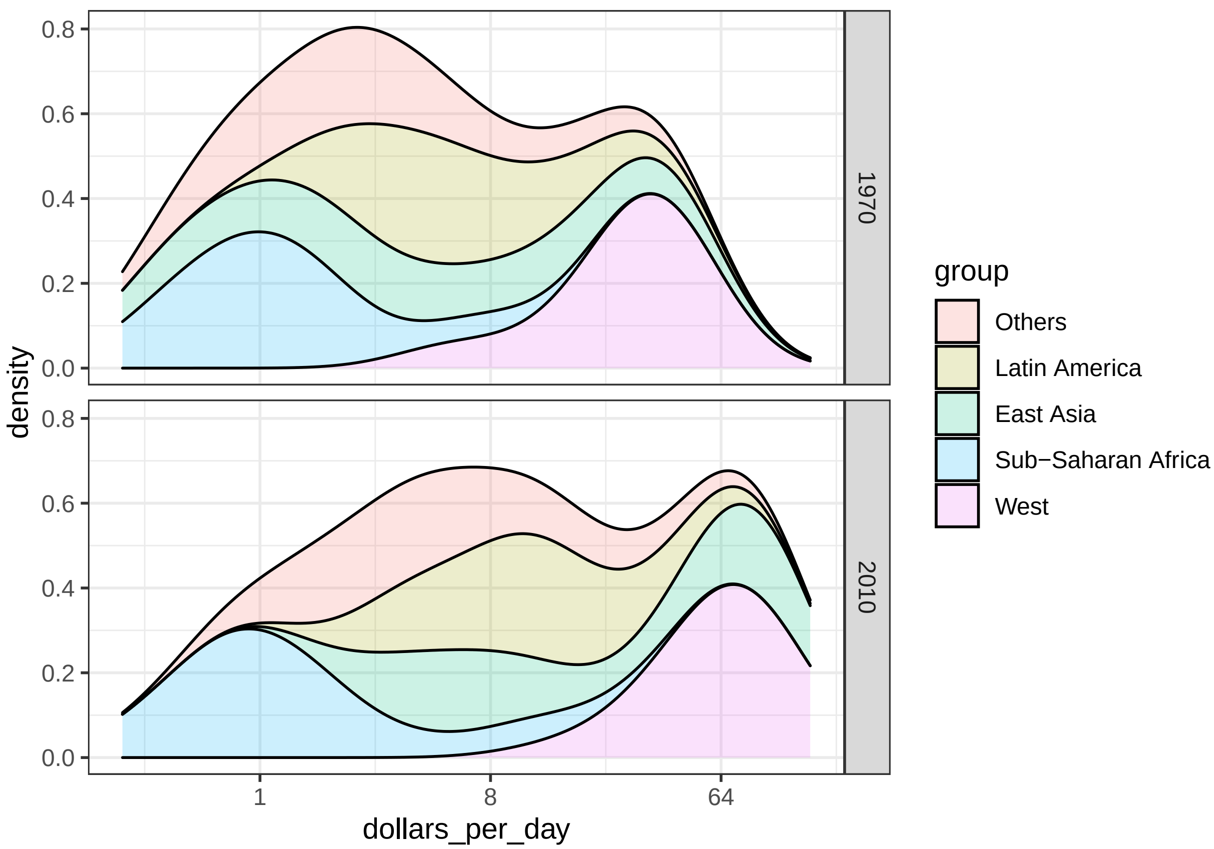


Code: Add new region groups with case_when

```r
# add group as a factor, grouping regions
gapminder <- gapminder %>%
    mutate(group = case_when(
        .$region %in% west ~ "West",
        .$region %in% c("Eastern Asia", "South-Eastern Asia") ~ "East Asia",
        .$region %in% c("Caribbean", "Central America", "South America") ~ "Latin America",
        .$continent == "Africa" & .$region != "Northern Africa" ~ "Sub-Saharan Africa",
        TRUE ~ "Others"))

# reorder factor levels
gapminder <- gapminder %>%
    mutate(group = factor(group, levels = c("Others", "Latin America", "East Asia", "Sub-Saharan Africa
```

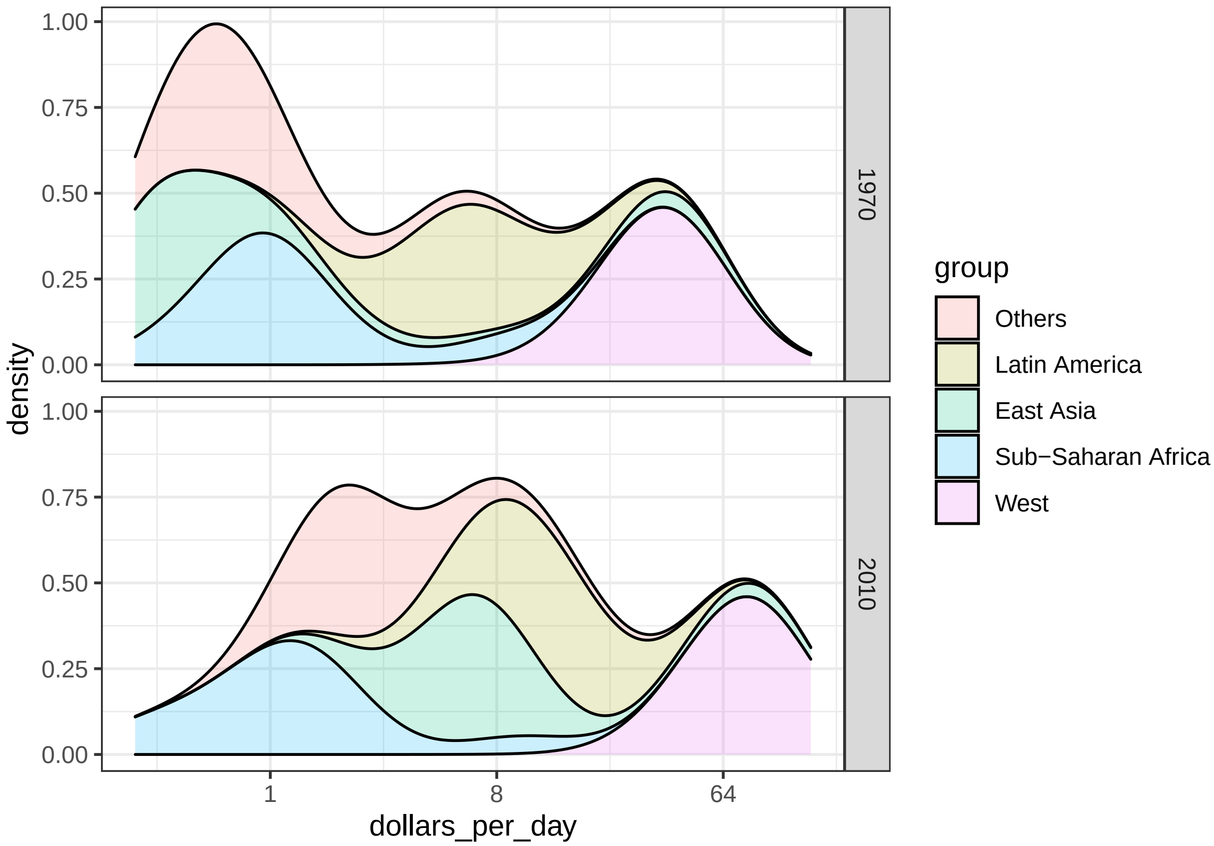
Code: Stacked density plot

```
# note you must redefine p with the new gapminder object first
p <- gapminder %>%
  filter(year %in% c(past_year, present_year) & country %in% country_list) %>%
    ggplot(aes(dollars_per_day, fill = group)) +
    scale_x_continuous(trans = "log2")

# stacked density plot
p + geom_density(alpha = 0.2, bw = 0.75, position = "stack") +
    facet_grid(year ~ .)
```



Code: Weighted stacked density plot

```
# weighted stacked density plot
gapminder %>%
    filter(year %in% c(past_year, present_year) & country %in% country_list) %>%
    group_by(year) %>%
    mutate(weight = population/sum(population*2)) %>%
    ungroup() %>%
    ggplot(aes(dollars_per_day, fill = group, weight = weight)) +
    scale_x_continuous(trans = "log2") +
    geom_density(alpha = 0.2, bw = 0.75, position = "stack") + facet_grid(year ~ .)
```



Ecological Fallacy

The textbook for this section is available here

Key points

- The *breaks* argument allows us to set the location of the axis labels and tick marks.
- The *logistic* or *logit transformation* is defined as $f(p) = log\frac{p}{1-p}$, or the log of odds. This scale is useful for highlighting differences near 0 or near 1 and converts fold changes into constant increases.
- The *ecological fallacy* is assuming that conclusions made from the average of a group apply to all members of that group.

Code

```
# add additional cases
gapminder <- gapminder %>%
    mutate(group = case_when(
        .$region %in% west ~ "The West",
        .$region %in% "Northern Africa" ~ "Northern Africa",
        .$region %in% c("Eastern Asia", "South-Eastern Asia") ~ "East Asia",
        .$region == "Southern Asia" ~ "Southern Asia",
        .$region %in% c("Central America", "South America", "Caribbean") ~ "Latin America",
        .$continent == "Africa" & .$region != "Northern Africa" ~ "Sub-Saharan Africa",
        .$region %in% c("Melanesia", "Micronesia", "Polynesia") ~ "Pacific Islands"))

# define a data frame with group average income and average infant survival rate
```

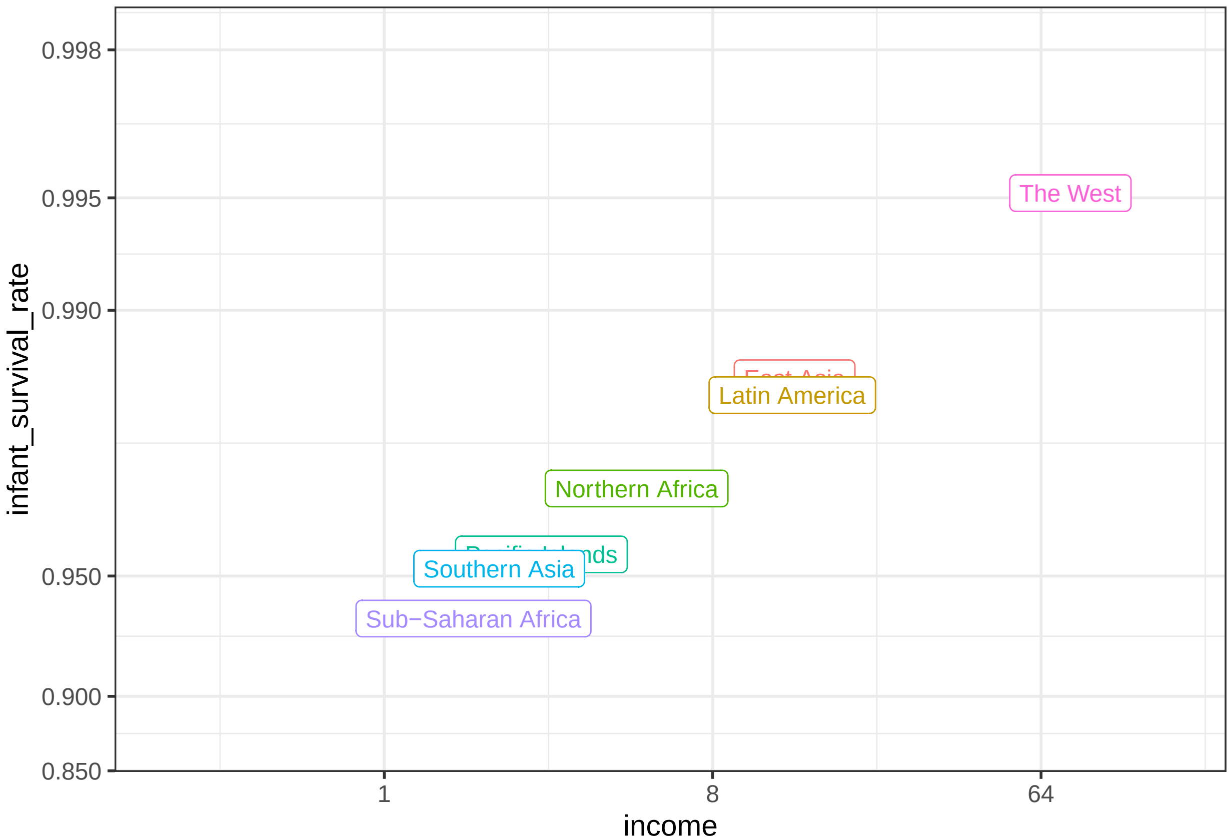
```r
surv_income <- gapminder %>%
    filter(year %in% present_year & !is.na(gdp) & !is.na(infant_mortality) & !is.na(group)) %>%
    group_by(group) %>%
    summarize(income = sum(gdp)/sum(population)/365,
                        infant_survival_rate = 1 - sum(infant_mortality/1000*population)/sum(population)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
surv_income %>% arrange(income)
```

```
## # A tibble: 7 x 3
##   group              income infant_survival_rate
##   <chr>               <dbl>                <dbl>
## 1 Sub-Saharan Africa   1.76                0.936
## 2 Southern Asia        2.07                0.952
## 3 Pacific Islands      2.70                0.956
## 4 Northern Africa      4.94                0.970
## 5 Latin America       13.2                 0.983
## 6 East Asia           13.4                 0.985
## 7 The West            77.1                 0.995
```

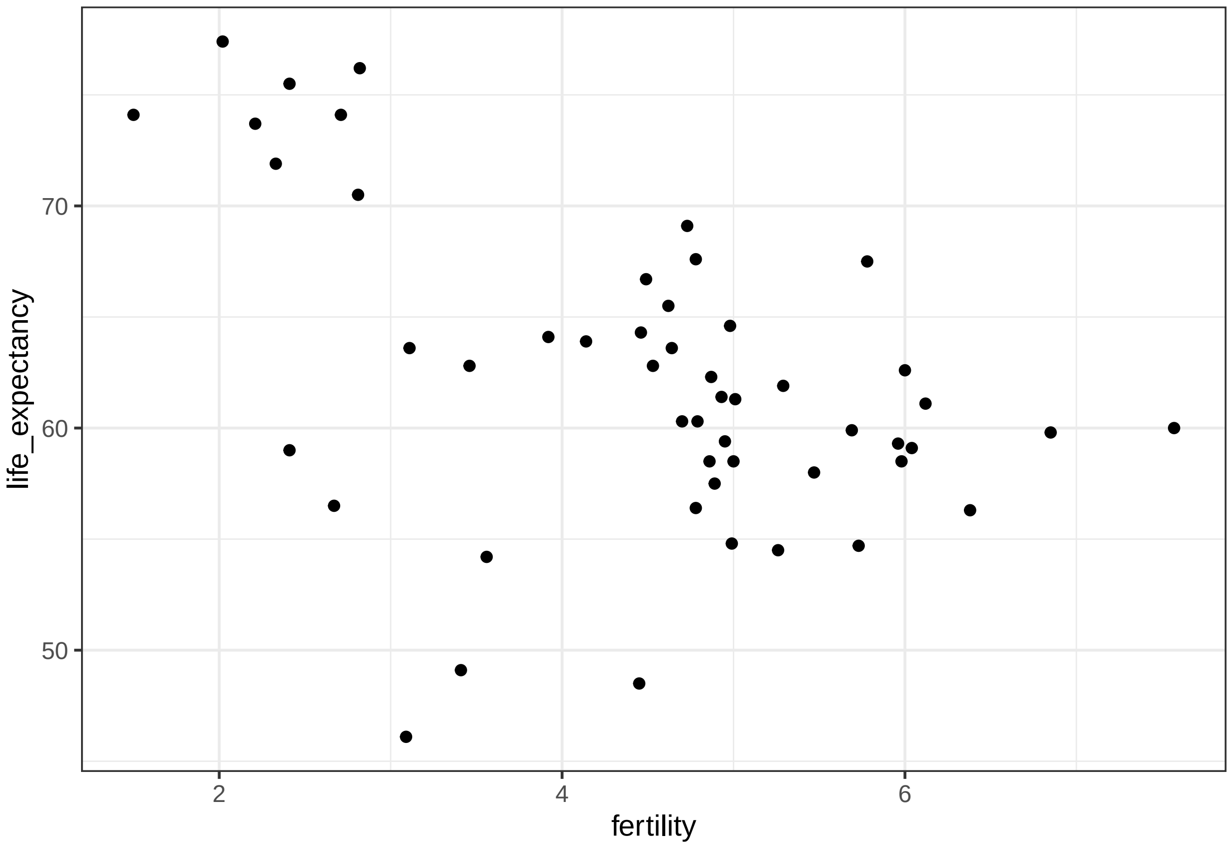
```r
# plot infant survival versus income, with transformed axes
surv_income %>% ggplot(aes(income, infant_survival_rate, label = group, color = group)) +
    scale_x_continuous(trans = "log2", limit = c(0.25, 150)) +
    scale_y_continuous(trans = "logit", limit = c(0.875, .9981),
                                       breaks = c(.85, .90, .95, .99, .995, .998)) +
    geom_label(size = 3, show.legend = FALSE)
```



Assessment - Exploring the Gapminder Dataset

1. The Gapminder Foundation is a non-profit organization based in Sweden that promotes global development through the use of statistics that can help reduce misconceptions about global development.

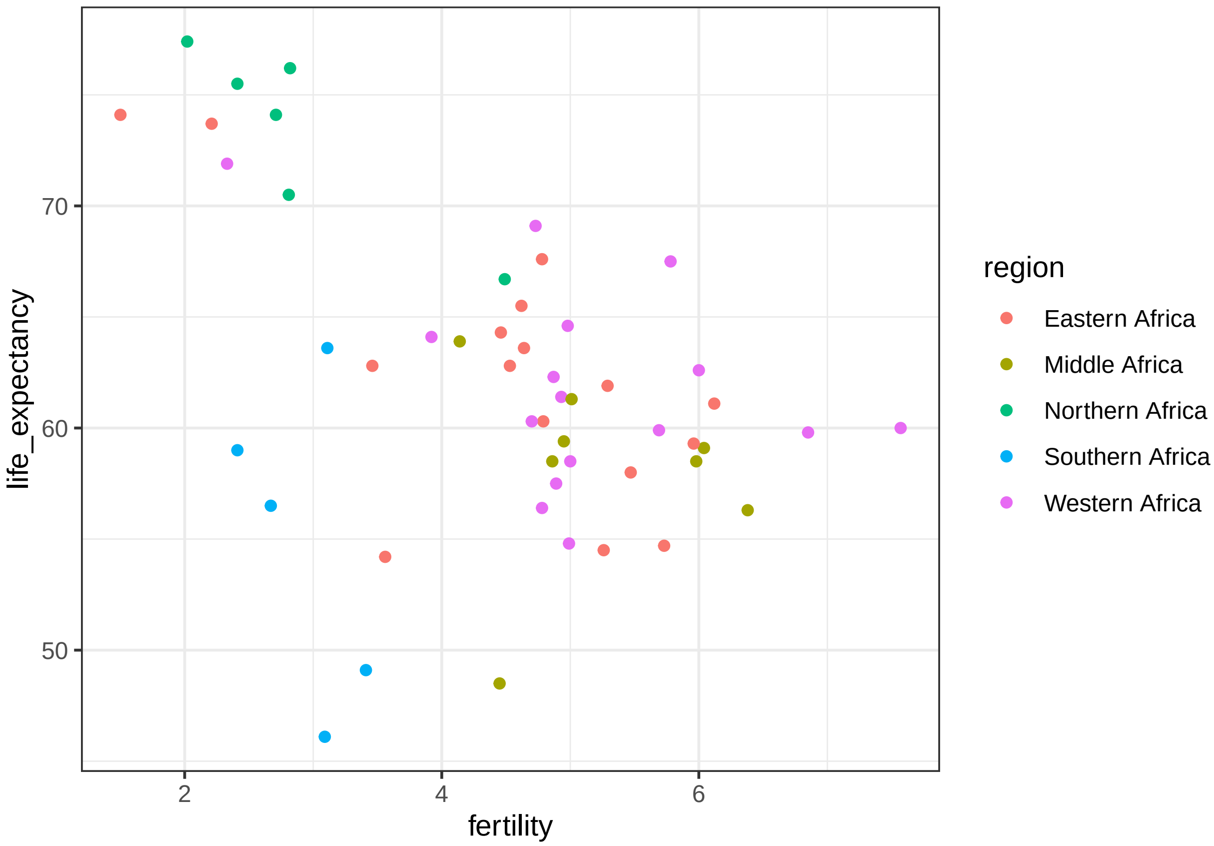
```r
## fill out the missing parts in filter and aes
gapminder %>% filter(year == 2012 & continent == "Africa") %>%
  ggplot(aes(fertility, life_expectancy)) +
  geom_point()
```



2. Note that there is quite a bit of variability in life expectancy and fertility with some African countries having very high life expectancies.

There also appear to be three clusters in the plot.

```
gapminder %>% filter(year == 2012 & continent == "Africa") %>%
  ggplot(aes(fertility, life_expectancy, color = region)) +
  geom_point()
```



3. While many of the countries in the high life expectancy/low fertility cluster are from Northern Africa, three countries are not.

```
df <- gapminder %>% filter(year == 2012 & continent == "Africa", fertility <= 3 & life_expectancy >= 70)
df
```

```
##      country          region
## 1    Algeria Northern Africa
## 2 Cape Verde  Western Africa
## 3      Egypt Northern Africa
## 4      Libya Northern Africa
## 5  Mauritius  Eastern Africa
## 6    Morocco Northern Africa
## 7 Seychelles  Eastern Africa
## 8    Tunisia Northern Africa
```

4. The Vietnam War lasted from 1955 to 1975.

Do the data support war having a negative effect on life expectancy? We will create a time series plot that covers the period from 1960 to 2010 of life expectancy for Vietnam and the United States, using color to distinguish the two countries. In this start we start the analysis by generating a table.

```
tab <- gapminder %>% filter(year >= 1960 & year <= 2010 & country%in%c("Vietnam", "United States"))
tab
```

```
##           country year infant_mortality life_expectancy fertility population
## 1   United States 1960             25.9           69.91      3.67  186176524
## 2         Vietnam 1960             75.6           58.52      6.35   32670623
## 3   United States 1961             25.4           70.32      3.63  189077076
## 4         Vietnam 1961             72.6           59.17      6.39   33666768
## 5   United States 1962             24.9           70.21      3.48  191860710
## 6         Vietnam 1962             69.9           59.82      6.43   34684164
## 7   United States 1963             24.4           70.04      3.35  194513911
## 8         Vietnam 1963             67.3           60.42      6.45   35722092
## 9   United States 1964             23.8           70.33      3.22  197028908
## 10        Vietnam 1964             61.7           60.95      6.46   36780984
## 11  United States 1965             23.3           70.41      2.93  199403532
## 12        Vietnam 1965             60.7           61.32      6.48   37860014
## 13  United States 1966             22.7           70.43      2.71  201629471
## 14        Vietnam 1966             59.9           61.36      6.49   38959335
## 15  United States 1967             22.0           70.76      2.56  203713082
## 16        Vietnam 1967             59.0           61.06      6.49   40074695
## 17  United States 1968             21.3           70.42      2.47  205687611
## 18        Vietnam 1968             58.2           60.45      6.49   41195833
## 19  United States 1969             20.6           70.66      2.46  207599308
## 20        Vietnam 1969             57.3           59.63      6.49   42309662
## 21  United States 1970             19.9           70.92      2.46  209485807
## 22        Vietnam 1970             56.4           58.78      6.47   43407291
## 23  United States 1971             19.1           71.24      2.27  211357912
## 24        Vietnam 1971             55.5           58.17      6.42   44485910
## 25  United States 1972             18.3           71.34      2.01  213219515
## 26        Vietnam 1972             54.7           58.00      6.35   45549487
## 27  United States 1973             17.5           71.54      1.87  215092900
## 28        Vietnam 1973             53.8           58.35      6.25   46604726
## 29  United States 1974             16.7           72.08      1.83  217001865
## 30        Vietnam 1974             52.8           59.23      6.13   47661770
## 31  United States 1975             16.0           72.68      1.77  218963561
## 32        Vietnam 1975             51.8           60.54      5.97   48729397
## 33  United States 1976             15.2           72.99      1.74  220993166
## 34        Vietnam 1976             50.9           62.07      5.80   49808071
## 35  United States 1977             14.5           73.38      1.78  223090871
## 36        Vietnam 1977             49.8           63.58      5.61   50899504
## 37  United States 1978             13.8           73.58      1.75  225239456
## 38        Vietnam 1978             48.8           64.86      5.42   52015279
## 39  United States 1979             13.2           74.03      1.80  227411604
## 40        Vietnam 1979             47.8           65.84      5.23   53169674
## 41  United States 1980             12.6           73.93      1.82  229588208
## 42        Vietnam 1980             46.8           66.49      5.05   54372518
## 43  United States 1981             12.1           74.36      1.81  231765783
## 44        Vietnam 1981             45.8           66.86      4.87   55627743
## 45  United States 1982             11.7           74.65      1.81  233953874
## 46        Vietnam 1982             44.8           67.10      4.69   56931822
## 47  United States 1983             11.2           74.71      1.78  236161961
## 48        Vietnam 1983             43.9           67.30      4.52   58277391
## 49  United States 1984             10.9           74.81      1.79  238404223
## 50        Vietnam 1984             43.0           67.51      4.36   59653092
## 51  United States 1985             10.6           74.79      1.84  240691557
## 52        Vietnam 1985             42.0           67.77      4.21   61049370
## 53  United States 1986             10.4           74.87      1.84  243032017
```

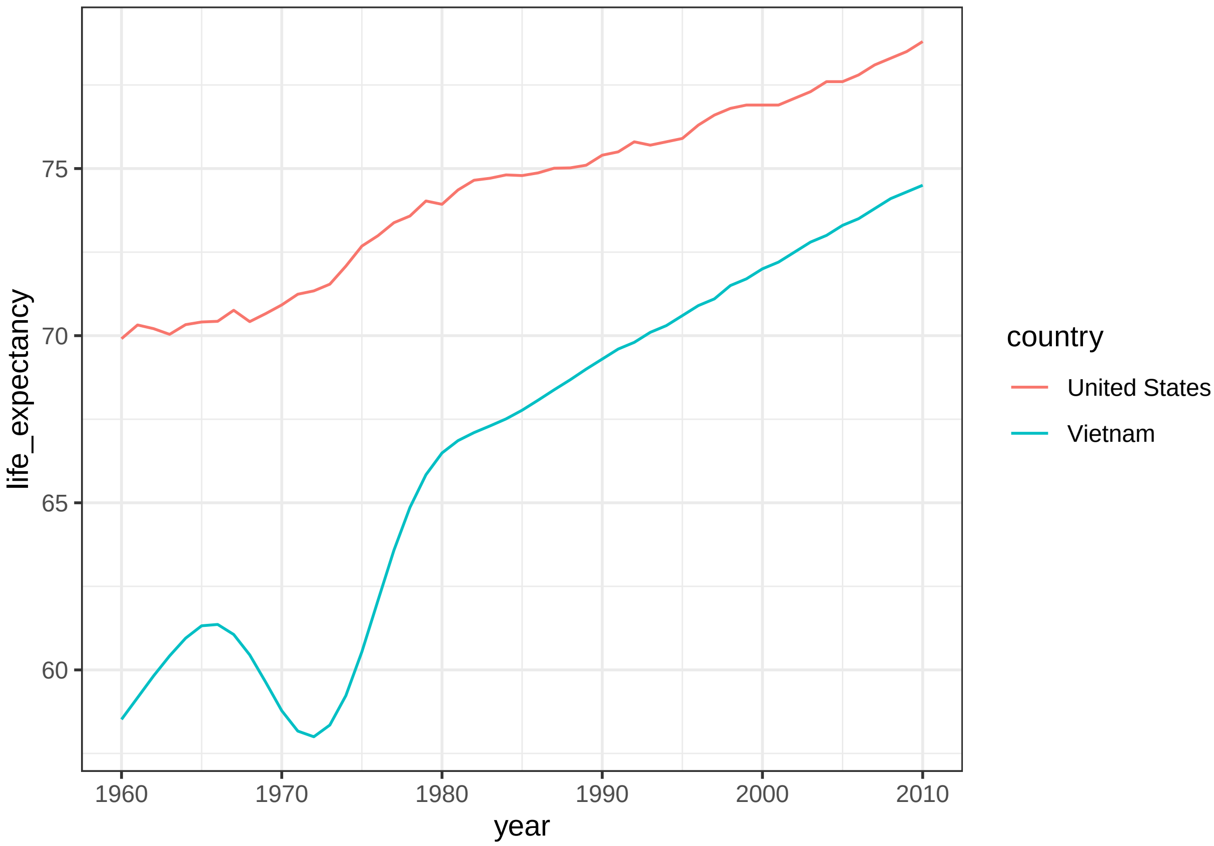
```
## 54        Vietnam 1986            41.0           68.07      4.06   62459557
## 55  United States 1987            10.2           75.01      1.87  245425409
## 56        Vietnam 1987            40.0           68.38      3.93   63881296
## 57  United States 1988            10.0           75.02      1.92  247865202
## 58        Vietnam 1988            38.9           68.68      3.81   65313709
## 59  United States 1989             9.7           75.10      2.00  250340795
## 60        Vietnam 1989            37.7           69.00      3.68   66757401
## 61  United States 1990             9.4           75.40      2.07  252847810
## 62        Vietnam 1990            36.6           69.30      3.56   68209604
## 63  United States 1991             9.1           75.50      2.06  255367160
## 64        Vietnam 1991            35.4           69.60      3.42   69670620
## 65  United States 1992             8.8           75.80      2.04  257908206
## 66        Vietnam 1992            34.3           69.80      3.26   71129537
## 67  United States 1993             8.5           75.70      2.02  260527420
## 68        Vietnam 1993            33.1           70.10      3.07   72558986
## 69  United States 1994             8.2           75.80      2.00  263301323
## 70        Vietnam 1994            32.0           70.30      2.88   73923849
## 71  United States 1995             8.0           75.90      1.98  266275528
## 72        Vietnam 1995            30.9           70.60      2.68   75198975
## 73  United States 1996             7.7           76.30      1.98  269483224
## 74        Vietnam 1996            29.9           70.90      2.48   76375677
## 75  United States 1997             7.5           76.60      1.97  272882865
## 76        Vietnam 1997            28.9           71.10      2.31   77460429
## 77  United States 1998             7.3           76.80      2.00  276354096
## 78        Vietnam 1998            27.9           71.50      2.17   78462888
## 79  United States 1999             7.2           76.90      2.01  279730801
## 80        Vietnam 1999            27.0           71.70      2.06   79399708
## 81  United States 2000             7.1           76.90      2.05  282895741
## 82        Vietnam 2000            26.1           72.00      1.98   80285563
## 83  United States 2001             7.0           76.90      2.03  285796198
## 84        Vietnam 2001            25.3           72.20      1.94   81123685
## 85  United States 2002             6.9           77.10      2.02  288470847
## 86        Vietnam 2002            24.6           72.50      1.92   81917488
## 87  United States 2003             6.8           77.30      2.05  291005482
## 88        Vietnam 2003            23.9           72.80      1.91   82683039
## 89  United States 2004             6.9           77.60      2.06  293530886
## 90        Vietnam 2004            23.2           73.00      1.90   83439812
## 91  United States 2005             6.8           77.60      2.06  296139635
## 92        Vietnam 2005            22.6           73.30      1.90   84203817
## 93  United States 2006             6.7           77.80      2.11  298860519
## 94        Vietnam 2006            22.0           73.50      1.89   84979667
## 95  United States 2007             6.6           78.10      2.12  301655953
## 96        Vietnam 2007            21.4           73.80      1.88   85770717
## 97  United States 2008             6.5           78.30      2.07  304473143
## 98        Vietnam 2008            20.8           74.10      1.86   86589342
## 99  United States 2009             6.4           78.50      2.00  307231961
## 100       Vietnam 2009            20.3           74.30      1.84   87449021
## 101 United States 2010             6.3           78.80      1.93  309876170
## 102       Vietnam 2010            19.8           74.50      1.82   88357775
##              gdp continent             region dollars_per_day     group
## 1   2.479391e+12  Americas   Northern America      36.4860841  The West
## 2             NA      Asia South-Eastern Asia              NA East Asia
## 3   2.536417e+12  Americas   Northern America      36.7526728  The West
## 4             NA      Asia South-Eastern Asia              NA East Asia
```

```
## 5   2.691139e+12  Americas   Northern America      38.4288283  The West
## 6             NA      Asia South-Eastern Asia              NA East Asia
## 7   2.809549e+12  Americas   Northern America      39.5724576  The West
## 8             NA      Asia South-Eastern Asia              NA East Asia
## 9   2.972502e+12  Americas   Northern America      41.3332358  The West
## 10            NA      Asia South-Eastern Asia              NA East Asia
## 11  3.162743e+12  Americas   Northern America      43.4548382  The West
## 12            NA      Asia South-Eastern Asia              NA East Asia
## 13  3.368321e+12  Americas   Northern America      45.7684897  The West
## 14            NA      Asia South-Eastern Asia              NA East Asia
## 15  3.452529e+12  Americas   Northern America      46.4328711  The West
## 16            NA      Asia South-Eastern Asia              NA East Asia
## 17  3.618250e+12  Americas   Northern America      48.1945141  The West
## 18            NA      Asia South-Eastern Asia              NA East Asia
## 19  3.730416e+12  Americas   Northern America      49.2309826  The West
## 20            NA      Asia South-Eastern Asia              NA East Asia
## 21  3.737877e+12  Americas   Northern America      48.8852142  The West
## 22            NA      Asia South-Eastern Asia              NA East Asia
## 23  3.867133e+12  Americas   Northern America      50.1276977  The West
## 24            NA      Asia South-Eastern Asia              NA East Asia
## 25  4.080668e+12  Americas   Northern America      52.4338121  The West
## 26            NA      Asia South-Eastern Asia              NA East Asia
## 27  4.321881e+12  Americas   Northern America      55.0495657  The West
## 28            NA      Asia South-Eastern Asia              NA East Asia
## 29  4.299437e+12  Americas   Northern America      54.2819231  The West
## 30            NA      Asia South-Eastern Asia              NA East Asia
## 31  4.291009e+12  Americas   Northern America      53.6901599  The West
## 32            NA      Asia South-Eastern Asia              NA East Asia
## 33  4.523528e+12  Americas   Northern America      56.0796900  The West
## 34            NA      Asia South-Eastern Asia              NA East Asia
## 35  4.733337e+12  Americas   Northern America      58.1289879  The West
## 36            NA      Asia South-Eastern Asia              NA East Asia
## 37  4.999656e+12  Americas   Northern America      60.8138968  The West
## 38            NA      Asia South-Eastern Asia              NA East Asia
## 39  5.157035e+12  Americas   Northern America      62.1290351  The West
## 40            NA      Asia South-Eastern Asia              NA East Asia
## 41  5.142220e+12  Americas   Northern America      61.3632291  The West
## 42            NA      Asia South-Eastern Asia              NA East Asia
## 43  5.272896e+12  Americas   Northern America      62.3314167  The West
## 44            NA      Asia South-Eastern Asia              NA East Asia
## 45  5.168479e+12  Americas   Northern America      60.5256797  The West
## 46            NA      Asia South-Eastern Asia              NA East Asia
## 47  5.401886e+12  Americas   Northern America      62.6675327  The West
## 48            NA      Asia South-Eastern Asia              NA East Asia
## 49  5.790542e+12  Americas   Northern America      66.5445377  The West
## 50  1.145347e+10      Asia South-Eastern Asia       0.5260311 East Asia
## 51  6.028651e+12  Americas   Northern America      68.6224765  The West
## 52  1.188938e+10      Asia South-Eastern Asia       0.5335622 East Asia
## 53  6.235265e+12  Americas   Northern America      70.2908174  The West
## 54  1.222101e+10      Asia South-Eastern Asia       0.5360622 East Asia
## 55  6.432743e+12  Americas   Northern America      71.8098149  The West
## 56  1.265894e+10      Asia South-Eastern Asia       0.5429137 East Asia
## 57  6.696490e+12  Americas   Northern America      74.0182447  The West
## 58  1.330898e+10      Asia South-Eastern Asia       0.5582742 East Asia
```

```
## 59  6.935219e+12  Americas   Northern America      75.8989379  The West
## 60  1.428912e+10      Asia South-Eastern Asia       0.5864260 East Asia
## 61  7.063943e+12  Americas   Northern America      76.5411775  The West
## 62  1.501800e+10      Asia South-Eastern Asia       0.6032171 East Asia
## 63  7.045491e+12  Americas   Northern America      75.5880837  The West
## 64  1.591320e+10      Asia South-Eastern Asia       0.6257703 East Asia
## 65  7.285373e+12  Americas   Northern America      77.3915942  The West
## 66  1.728906e+10      Asia South-Eastern Asia       0.6659299 East Asia
## 67  7.494650e+12  Americas   Northern America      78.8143037  The West
## 68  1.868476e+10      Asia South-Eastern Asia       0.7055104 East Asia
## 69  7.803020e+12  Americas   Northern America      81.1926662  The West
## 70  2.033630e+10      Asia South-Eastern Asia       0.7536931 East Asia
## 71  8.001917e+12  Americas   Northern America      82.3322348  The West
## 72  2.227648e+10      Asia South-Eastern Asia       0.8115996 East Asia
## 73  8.304875e+12  Americas   Northern America      84.4322774  The West
## 74  2.435711e+10      Asia South-Eastern Asia       0.8737312 East Asia
## 75  8.679071e+12  Americas   Northern America      87.1373006  The West
## 76  2.634272e+10      Asia South-Eastern Asia       0.9317253 East Asia
## 77  9.061073e+12  Americas   Northern America      89.8298924  The West
## 78  2.786124e+10      Asia South-Eastern Asia       0.9728441 East Asia
## 79  9.502248e+12  Americas   Northern America      93.0664656  The West
## 80  2.919122e+10      Asia South-Eastern Asia       1.0072573 East Asia
## 81  9.898800e+12  Americas   Northern America      95.8657062  The West
## 82  3.117252e+10      Asia South-Eastern Asia       1.0637548 East Asia
## 83  1.000703e+13  Americas   Northern America      95.9303301  The West
## 84  3.332183e+10      Asia South-Eastern Asia       1.1253518 East Asia
## 85  1.018996e+13  Americas   Northern America      96.7782269  The West
## 86  3.568108e+10      Asia South-Eastern Asia       1.1933517 East Asia
## 87  1.045007e+13  Americas   Northern America      98.3841464  The West
## 88  3.830049e+10      Asia South-Eastern Asia       1.2690975 East Asia
## 89  1.081371e+13  Americas   Northern America     100.9317862  The West
## 90  4.128394e+10      Asia South-Eastern Asia       1.3555482 East Asia
## 91  1.114630e+13  Americas   Northern America     103.1195945  The West
## 92  4.476905e+10      Asia South-Eastern Asia       1.4566432 East Asia
## 93  1.144269e+13  Americas   Northern America     104.8978847  The West
## 94  4.845303e+10      Asia South-Eastern Asia       1.5621152 East Asia
## 95  1.166093e+13  Americas   Northern America     105.9078868  The West
## 96  5.255039e+10      Asia South-Eastern Asia       1.6785876 East Asia
## 97  1.161905e+13  Americas   Northern America     104.5511719  The West
## 98  5.586668e+10      Asia South-Eastern Asia       1.7676470 East Asia
## 99  1.120919e+13  Americas   Northern America      99.9574489  The West
## 100 5.884079e+10      Asia South-Eastern Asia       1.8434472 East Asia
## 101 1.154791e+13  Americas   Northern America     102.0991582  The West
## 102 6.283222e+10      Asia South-Eastern Asia       1.9482502 East Asia
```

5. Now that you have created the data table in Exercise 4, it is time to plot the data for the two countries.

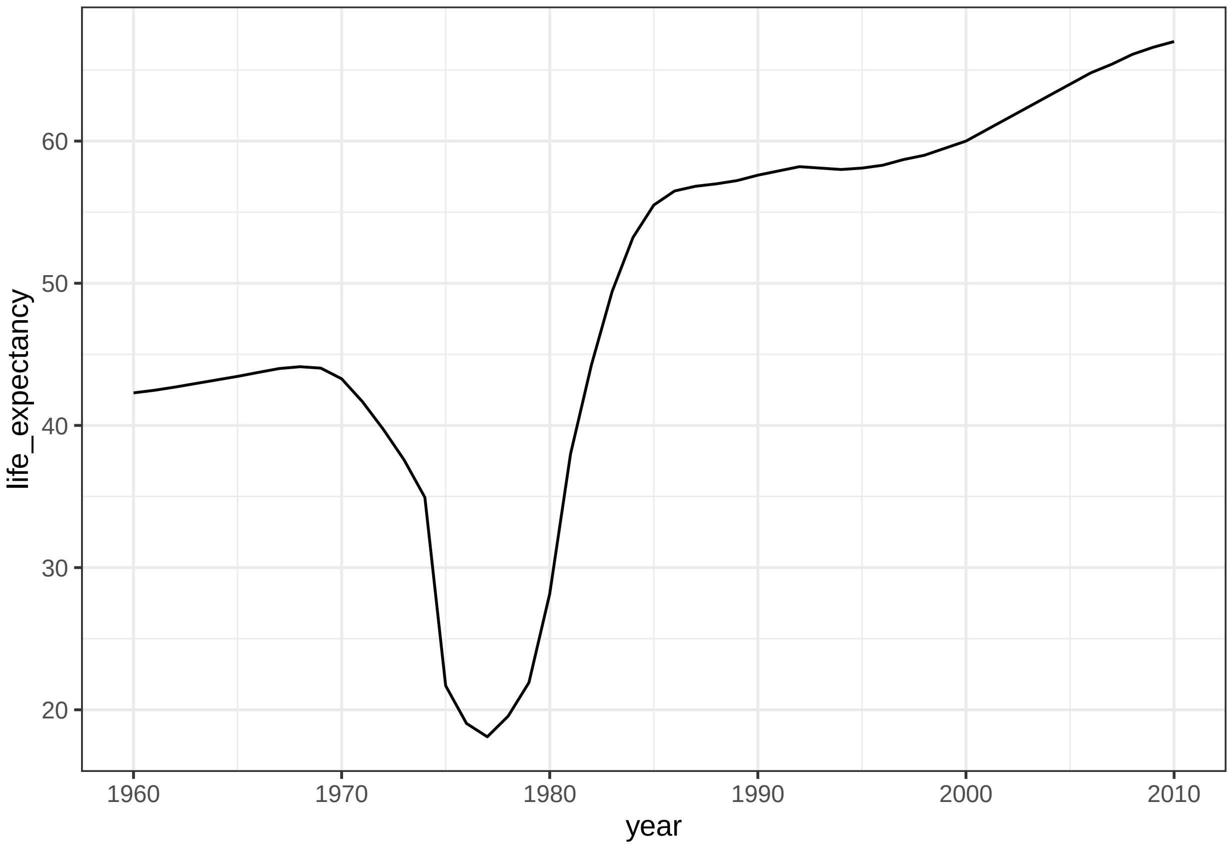
```
p <- tab %>% ggplot(aes(year,life_expectancy,color=country)) + geom_line()
p
```



6. Cambodia was also involved in this conflict and, after the war, Pol Pot and his communist Khmer Rouge took control and ruled Cambodia from 1975 to 1979.

He is considered one of the most brutal dictators in history. Do the data support this claim?

```
p <- gapminder %>% filter(year >= 1960 & year <= 2010 & country == "Cambodia") %>% ggplot(aes(year, life
p
```



7. Now we are going to calculate and plot dollars per day for African countries in 2010 using GDP data.

In the first part of this analysis, we will create the dollars per day variable.

```
daydollars <- gapminder %>%
mutate(dollars_per_day = gdp/population/365) %>% filter(continent == "Africa" & year == 2010 & !is.na(g
daydollars
```

```
##                       country year infant_mortality life_expectancy fertility
## 1                     Algeria 2010             23.5            76.0      2.82
## 2                      Angola 2010            109.6            57.6      6.22
## 3                       Benin 2010             71.0            60.8      5.10
## 4                    Botswana 2010             39.8            55.6      2.76
## 5                Burkina Faso 2010             69.7            59.0      5.87
## 6                     Burundi 2010             63.8            60.4      6.30
## 7                    Cameroon 2010             66.2            57.8      5.02
## 8                  Cape Verde 2010             23.3            71.1      2.43
## 9    Central African Republic 2010            101.7            47.9      4.63
## 10                       Chad 2010             93.6            55.8      6.60
## 11                    Comoros 2010             63.1            67.7      4.92
## 12           Congo, Dem. Rep. 2010             84.8            58.4      6.25
## 13                Congo, Rep. 2010             42.2            60.4      5.07
## 14              Cote d'Ivoire 2010             76.9            56.6      4.91
## 15                      Egypt 2010             24.3            70.1      2.88
## 16          Equatorial Guinea 2010             78.9            58.6      5.14
## 17                    Eritrea 2010             39.4            60.1      4.97
```

```
## 18            Ethiopia 2010         50.8           62.1      4.90
## 19               Gabon 2010         42.8           63.0      4.21
## 20              Gambia 2010         51.7           66.5      5.80
## 21               Ghana 2010         50.2           62.9      4.05
## 22              Guinea 2010         71.2           57.9      5.17
## 23       Guinea-Bissau 2010         73.4           54.3      5.12
## 24               Kenya 2010         42.4           62.9      4.62
## 25             Lesotho 2010         75.2           46.4      3.21
## 26             Liberia 2010         65.2           60.8      5.02
## 27          Madagascar 2010         42.1           62.4      4.65
## 28              Malawi 2010         57.5           55.4      5.64
## 29                Mali 2010         82.9           59.2      6.84
## 30          Mauritania 2010         70.1           68.6      4.84
## 31           Mauritius 2010         13.3           73.4      1.52
## 32             Morocco 2010         28.5           73.7      2.58
## 33          Mozambique 2010         71.9           54.4      5.41
## 34             Namibia 2010         37.5           61.4      3.23
## 35               Niger 2010         66.1           59.2      7.58
## 36             Nigeria 2010         81.5           61.2      6.02
## 37              Rwanda 2010         43.8           65.1      4.84
## 38             Senegal 2010         46.7           64.2      5.05
## 39          Seychelles 2010         12.2           73.1      2.26
## 40        Sierra Leone 2010        107.0           55.0      4.94
## 41        South Africa 2010         38.2           54.9      2.47
## 42               Sudan 2010         53.3           66.1      4.64
## 43           Swaziland 2010         59.1           46.4      3.56
## 44            Tanzania 2010         42.4           61.4      5.43
## 45                Togo 2010         59.3           58.7      4.79
## 46             Tunisia 2010         14.9           77.1      2.04
## 47              Uganda 2010         49.5           57.8      6.16
## 48              Zambia 2010         52.9           53.1      5.81
## 49            Zimbabwe 2010         55.8           49.1      3.72
##    population          gdp continent          region dollars_per_day
## 1    36036159  79164339611    Africa Northern Africa       6.0186382
## 2    21219954  26125663270    Africa   Middle Africa       3.3731063
## 3     9509798   3336801340    Africa  Western Africa       0.9613161
## 4     2047831   8408166868    Africa Southern Africa      11.2490111
## 5    15632066   4655655008    Africa  Western Africa       0.8159650
## 6     9461117   1158914103    Africa  Eastern Africa       0.3355954
## 7    20590666  13986616694    Africa   Middle Africa       1.8610130
## 8      490379    971606715    Africa  Western Africa       5.4283242
## 9     4444973   1054122016    Africa   Middle Africa       0.6497240
## 10   11896380   3369354207    Africa   Middle Africa       0.7759594
## 11     698695    247231031    Africa  Eastern Africa       0.9694434
## 12   65938712   6961485000    Africa   Middle Africa       0.2892468
## 13    4066078   5067059617    Africa   Middle Africa       3.4141881
## 14   20131707  11603002049    Africa  Western Africa       1.5790537
## 15   82040994 160258746162    Africa Northern Africa       5.3517764
## 16     728710   5979285835    Africa   Middle Africa      22.4802803
## 17    4689664    771116883    Africa  Eastern Africa       0.4504905
## 18   87561814  18291486355    Africa  Eastern Africa       0.5723232
## 19    1541936   6343809583    Africa   Middle Africa      11.2717391
## 20    1693002   1217357172    Africa  Western Africa       1.9700066
## 21   24317734   8779397392    Africa  Western Africa       0.9891194
```

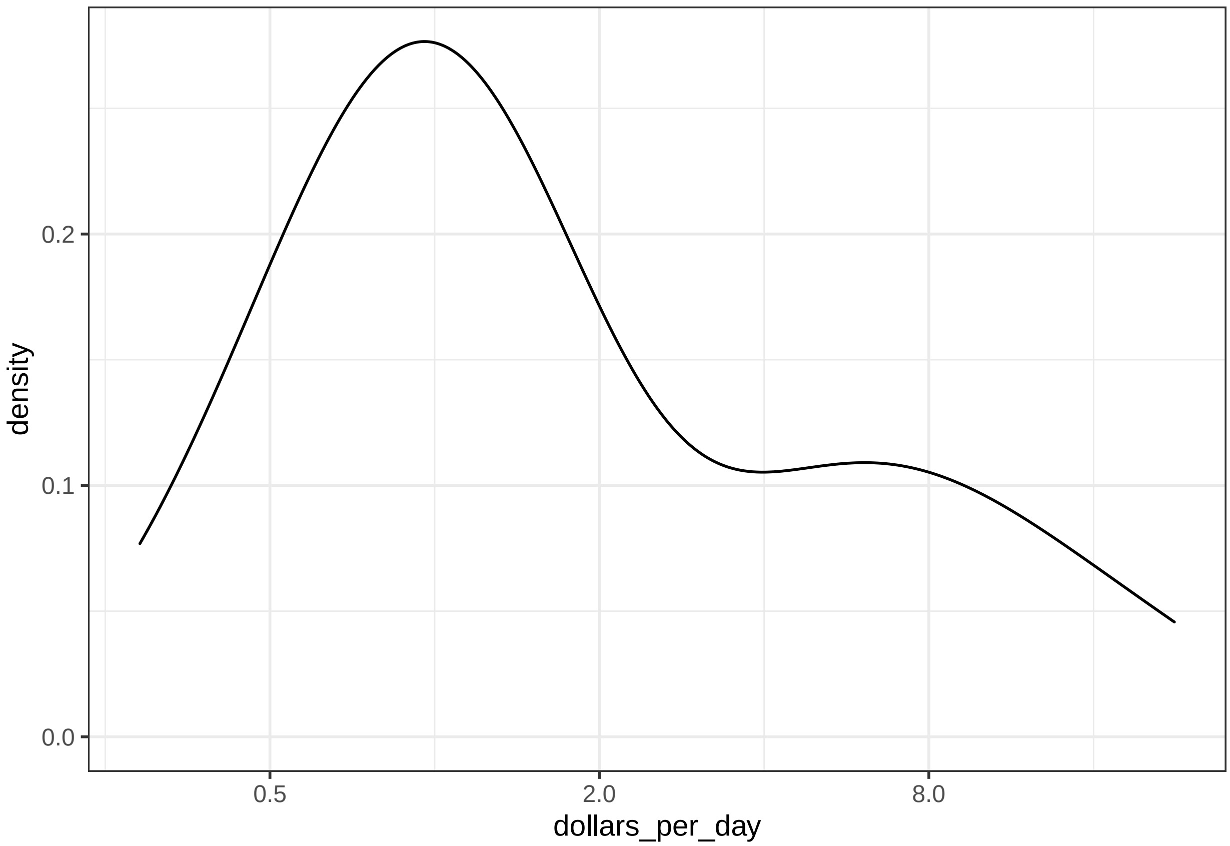
```
## 22    11012406   5493989673    Africa  Western Africa       1.3668245
## 23     1634196    244395463    Africa  Western Africa       0.4097285
## 24    40328313  18988282813    Africa  Eastern Africa       1.2899794
## 25     2010586   1076239050    Africa Southern Africa       1.4665377
## 26     3957990   1040653199    Africa  Western Africa       0.7203416
## 27    21079532   5026822443    Africa  Eastern Africa       0.6533407
## 28    14769824   2758392725    Africa  Eastern Africa       0.5116676
## 29    15167286   4199858651    Africa  Western Africa       0.7586368
## 30     3591400   2107593972    Africa  Western Africa       1.6077936
## 31     1247951   6636426093    Africa  Eastern Africa      14.5694737
## 32    32107739  59908047776    Africa Northern Africa       5.1119027
## 33    24321457   8972305823    Africa  Eastern Africa       1.0106985
## 34     2193643   6155469329    Africa Southern Africa       7.6878050
## 35    16291990   2781188119    Africa  Western Africa       0.4676957
## 36   159424742  85581744176    Africa  Western Africa       1.4707286
## 37    10293669   3583713093    Africa  Eastern Africa       0.9538282
## 38    12956791   6984284544    Africa  Western Africa       1.4768337
## 39       93081    760361490    Africa  Eastern Africa      22.3803157
## 40     5775902   1574302614    Africa  Western Africa       0.7467505
## 41    51621594 187639624489    Africa Southern Africa       9.9586457
## 42    36114885  22819076998    Africa Northern Africa       1.7310873
## 43     1193148   1911603442    Africa Southern Africa       4.3894552
## 44    45648525  19965679449    Africa  Eastern Africa       1.1982970
## 45     6390851   1595792895    Africa  Western Africa       0.6841085
## 46    10639194  33161453137    Africa Northern Africa       8.5394905
## 47    33149417  12701095116    Africa  Eastern Africa       1.0497174
## 48    13917439   5587389858    Africa  Eastern Africa       1.0999091
## 49    13973897   4032423429    Africa  Eastern Africa       0.7905980
##                  group
## 1     Northern Africa
## 2  Sub-Saharan Africa
## 3  Sub-Saharan Africa
## 4  Sub-Saharan Africa
## 5  Sub-Saharan Africa
## 6  Sub-Saharan Africa
## 7  Sub-Saharan Africa
## 8  Sub-Saharan Africa
## 9  Sub-Saharan Africa
## 10 Sub-Saharan Africa
## 11 Sub-Saharan Africa
## 12 Sub-Saharan Africa
## 13 Sub-Saharan Africa
## 14 Sub-Saharan Africa
## 15    Northern Africa
## 16 Sub-Saharan Africa
## 17 Sub-Saharan Africa
## 18 Sub-Saharan Africa
## 19 Sub-Saharan Africa
## 20 Sub-Saharan Africa
## 21 Sub-Saharan Africa
## 22 Sub-Saharan Africa
## 23 Sub-Saharan Africa
## 24 Sub-Saharan Africa
## 25 Sub-Saharan Africa
```

```
## 26 Sub-Saharan Africa
## 27 Sub-Saharan Africa
## 28 Sub-Saharan Africa
## 29 Sub-Saharan Africa
## 30 Sub-Saharan Africa
## 31 Sub-Saharan Africa
## 32    Northern Africa
## 33 Sub-Saharan Africa
## 34 Sub-Saharan Africa
## 35 Sub-Saharan Africa
## 36 Sub-Saharan Africa
## 37 Sub-Saharan Africa
## 38 Sub-Saharan Africa
## 39 Sub-Saharan Africa
## 40 Sub-Saharan Africa
## 41 Sub-Saharan Africa
## 42    Northern Africa
## 43 Sub-Saharan Africa
## 44 Sub-Saharan Africa
## 45 Sub-Saharan Africa
## 46    Northern Africa
## 47 Sub-Saharan Africa
## 48 Sub-Saharan Africa
## 49 Sub-Saharan Africa
```

8. Now we are going to calculate and plot dollars per day for African countries in 2010 using GDP data.

In the second part of this analysis, we will plot the smooth density plot using a log (base 2) x axis.

```
p <- daydollars %>% ggplot(aes(dollars_per_day)) +
scale_x_continuous(trans = "log2") + geom_density()
p
```



9. Now we are going to combine the plotting tools we have used in the past two exercises to create density plots for multiple years.

```
daydollars <- gapminder %>%
mutate(dollars_per_day = gdp/population/365) %>% filter(continent == "Africa" & year%in%c(1970,2010) &
daydollars
```

```
##                       country year infant_mortality life_expectancy fertility
## 1                     Algeria 1970            146.0           52.41      7.64
## 2                       Benin 1970            157.1           43.93      6.75
## 3                    Botswana 1970             85.3           54.30      6.64
## 4                Burkina Faso 1970            149.3           40.27      6.62
## 5                     Burundi 1970            146.4           42.76      7.31
## 6                    Cameroon 1970            126.2           48.97      6.21
## 7    Central African Republic 1970            137.0           43.36      5.95
## 8                        Chad 1970            135.9           45.72      6.53
## 9            Congo, Dem. Rep. 1970            149.0           48.13      6.21
## 10                Congo, Rep. 1970             88.5           52.85      6.26
## 11              Cote d'Ivoire 1970            161.0           45.38      7.91
## 12                      Egypt 1970            162.0           52.54      5.94
## 13                      Gabon 1970               NA           45.55      5.08
## 14                     Gambia 1970            126.0           43.31      6.09
## 15                      Ghana 1970            120.1           50.08      6.95
## 16              Guinea-Bissau 1970               NA           45.50      6.07
## 17                      Kenya 1970             91.3           53.83      8.08
## 18                    Lesotho 1970            131.6           49.67      5.81
```

```
## 19                  Liberia 1970            191.3          40.10     6.70
## 20              Madagascar 1970             93.2          47.77     7.33
## 21                   Malawi 1970            207.7          41.62     7.30
## 22                     Mali 1970            195.7          34.51     6.90
## 23               Mauritania 1970            108.5          49.77     6.78
## 24                  Morocco 1970            120.8          54.34     6.69
## 25                    Niger 1970            137.6          38.24     7.42
## 26                  Nigeria 1970            168.9          41.79     6.47
## 27                   Rwanda 1970            129.4          45.58     8.23
## 28                  Senegal 1970            121.7          39.59     7.34
## 29               Seychelles 1970             54.1          64.62     5.76
## 30             Sierra Leone 1970            191.0          43.15     6.70
## 31             South Africa 1970               NA          52.77     5.59
## 32                    Sudan 1970             94.7          54.26     6.89
## 33                Swaziland 1970            119.3          48.79     6.88
## 34                     Togo 1970            132.8          47.72     7.08
## 35                  Tunisia 1970            122.2          52.94     6.44
## 36                   Zambia 1970            109.3          53.88     7.44
## 37                 Zimbabwe 1970             72.4          57.22     7.42
## 38                  Algeria 2010             23.5          76.00     2.82
## 39                   Angola 2010            109.6          57.60     6.22
## 40                    Benin 2010             71.0          60.80     5.10
## 41                 Botswana 2010             39.8          55.60     2.76
## 42             Burkina Faso 2010             69.7          59.00     5.87
## 43                  Burundi 2010             63.8          60.40     6.30
## 44                 Cameroon 2010             66.2          57.80     5.02
## 45               Cape Verde 2010             23.3          71.10     2.43
## 46 Central African Republic 2010            101.7          47.90     4.63
## 47                     Chad 2010             93.6          55.80     6.60
## 48                  Comoros 2010             63.1          67.70     4.92
## 49         Congo, Dem. Rep. 2010             84.8          58.40     6.25
## 50              Congo, Rep. 2010             42.2          60.40     5.07
## 51            Cote d'Ivoire 2010             76.9          56.60     4.91
## 52                    Egypt 2010             24.3          70.10     2.88
## 53        Equatorial Guinea 2010             78.9          58.60     5.14
## 54                  Eritrea 2010             39.4          60.10     4.97
## 55                 Ethiopia 2010             50.8          62.10     4.90
## 56                    Gabon 2010             42.8          63.00     4.21
## 57                   Gambia 2010             51.7          66.50     5.80
## 58                    Ghana 2010             50.2          62.90     4.05
## 59                   Guinea 2010             71.2          57.90     5.17
## 60            Guinea-Bissau 2010             73.4          54.30     5.12
## 61                    Kenya 2010             42.4          62.90     4.62
## 62                  Lesotho 2010             75.2          46.40     3.21
## 63                  Liberia 2010             65.2          60.80     5.02
## 64               Madagascar 2010             42.1          62.40     4.65
## 65                   Malawi 2010             57.5          55.40     5.64
## 66                     Mali 2010             82.9          59.20     6.84
## 67               Mauritania 2010             70.1          68.60     4.84
## 68                Mauritius 2010             13.3          73.40     1.52
## 69                  Morocco 2010             28.5          73.70     2.58
## 70               Mozambique 2010             71.9          54.40     5.41
## 71                  Namibia 2010             37.5          61.40     3.23
## 72                    Niger 2010             66.1          59.20     7.58
```

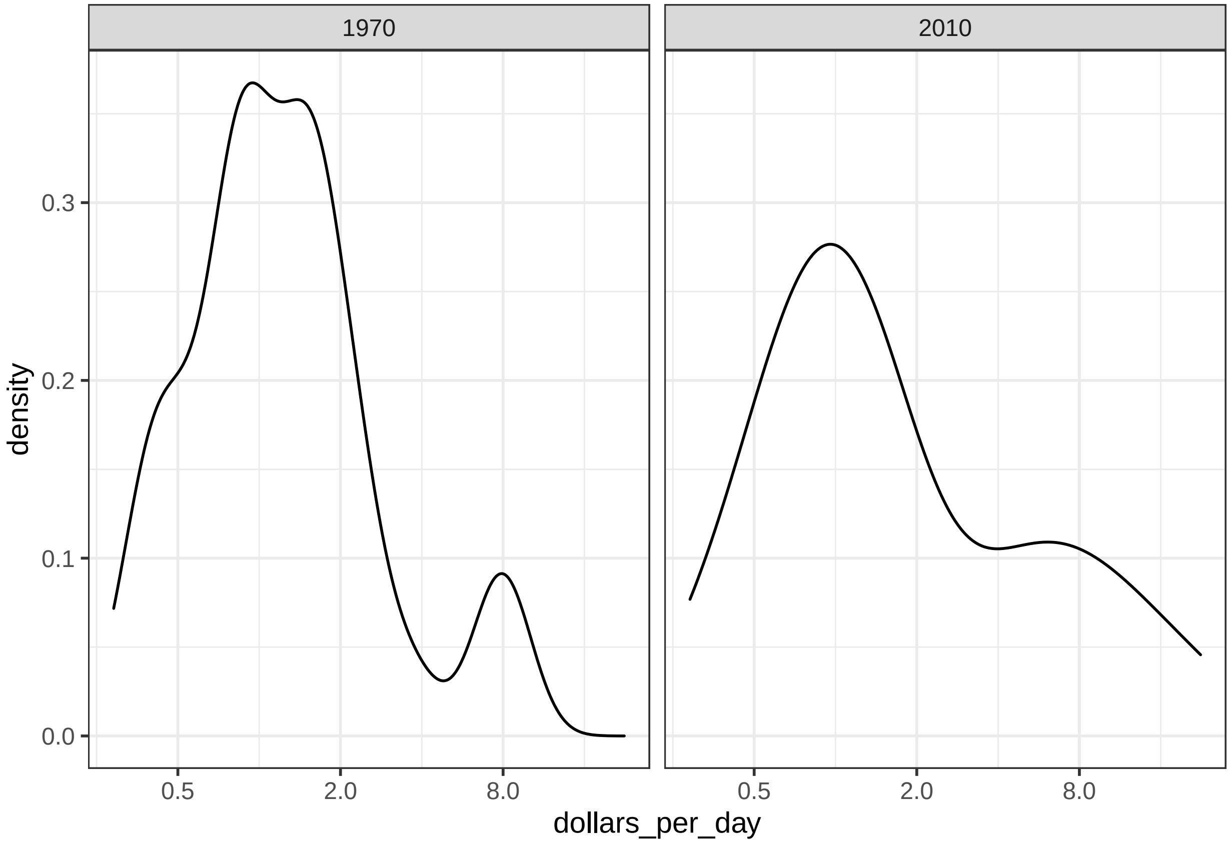
```
## 73               Nigeria 2010            81.5            61.20      6.02
## 74                Rwanda 2010            43.8            65.10      4.84
## 75               Senegal 2010            46.7            64.20      5.05
## 76            Seychelles 2010            12.2            73.10      2.26
## 77          Sierra Leone 2010           107.0            55.00      4.94
## 78          South Africa 2010            38.2            54.90      2.47
## 79                 Sudan 2010            53.3            66.10      4.64
## 80             Swaziland 2010            59.1            46.40      3.56
## 81              Tanzania 2010            42.4            61.40      5.43
## 82                  Togo 2010            59.3            58.70      4.79
## 83               Tunisia 2010            14.9            77.10      2.04
## 84                Uganda 2010            49.5            57.80      6.16
## 85                Zambia 2010            52.9            53.10      5.81
## 86              Zimbabwe 2010            55.8            49.10      3.72
##    population         gdp continent          region dollars_per_day
## 1    14550033 19741305571    Africa Northern Africa       3.7172265
## 2     2907769   831774871    Africa  Western Africa       0.7837057
## 3      693021   283867117    Africa Southern Africa       1.1222144
## 4     5624597   795164207    Africa  Western Africa       0.3873223
## 5     3457113   524049198    Africa  Eastern Africa       0.4153035
## 6     6770967  3372153343    Africa   Middle Africa       1.3644693
## 7     1828710   647622869    Africa   Middle Africa       0.9702518
## 8     3644911   829387598    Africa   Middle Africa       0.6234157
## 9    20009902  6728080745    Africa   Middle Africa       0.9211988
## 10    1335090   939633199    Africa   Middle Africa       1.9282127
## 11    5241914  4619775632    Africa  Western Africa       2.4145607
## 12   34808599 20331718433    Africa Northern Africa       1.6002752
## 13     590119  1722664256    Africa   Middle Africa       7.9977566
## 14     447283   247459869    Africa  Western Africa       1.5157568
## 15    8596977  2549677064    Africa  Western Africa       0.8125434
## 16     711828   104038537    Africa  Western Africa       0.4004297
## 17   11252466  3276361787    Africa  Eastern Africa       0.7977215
## 18    1032240   184783955    Africa Southern Africa       0.4904454
## 19    1419728  1094083642    Africa  Western Africa       2.1113125
## 20    6576301  2807129955    Africa  Eastern Africa       1.1694670
## 21    4603739   549382768    Africa  Eastern Africa       0.3269426
## 22    5949043  1038617256    Africa  Western Africa       0.4783167
## 23    1148908   700627427    Africa  Western Africa       1.6707406
## 24   16039600 12097898528    Africa Northern Africa       2.0664435
## 25    4497355  1343819364    Africa  Western Africa       0.8186360
## 26   56131844 19793025795    Africa  Western Africa       0.9660732
## 27    3754546   809941587    Africa  Eastern Africa       0.5910217
## 28    4217754  2266115562    Africa  Western Africa       1.4720005
## 29      52364   141888524    Africa  Eastern Africa       7.4237202
## 30    2514151   739785784    Africa  Western Africa       0.8061610
## 31   22502502 68558449204    Africa Southern Africa       8.3471326
## 32   10232758  3901968151    Africa Northern Africa       1.0447158
## 33     445844   257078586    Africa Southern Africa       1.5797564
## 34    2115521   618863063    Africa  Western Africa       0.8014646
## 35    5060393  4688590613    Africa Northern Africa       2.5384301
## 36    4185378  2384401746    Africa  Eastern Africa       1.5608166
## 37    5206311  2682438620    Africa  Eastern Africa       1.4115843
## 38   36036159 79164339611    Africa Northern Africa       6.0186382
## 39   21219954 26125663270    Africa   Middle Africa       3.3731063
```

```
## 40    9509798   3336801340    Africa  Western Africa       0.9613161
## 41    2047831   8408166868    Africa Southern Africa      11.2490111
## 42   15632066   4655655008    Africa  Western Africa       0.8159650
## 43    9461117   1158914103    Africa  Eastern Africa       0.3355954
## 44   20590666  13986616694    Africa   Middle Africa       1.8610130
## 45     490379    971606715    Africa  Western Africa       5.4283242
## 46    4444973   1054122016    Africa   Middle Africa       0.6497240
## 47   11896380   3369354207    Africa   Middle Africa       0.7759594
## 48     698695    247231031    Africa  Eastern Africa       0.9694434
## 49   65938712   6961485000    Africa   Middle Africa       0.2892468
## 50    4066078   5067059617    Africa   Middle Africa       3.4141881
## 51   20131707  11603002049    Africa  Western Africa       1.5790537
## 52   82040994 160258746162    Africa Northern Africa       5.3517764
## 53     728710   5979285835    Africa   Middle Africa      22.4802803
## 54    4689664    771116883    Africa  Eastern Africa       0.4504905
## 55   87561814  18291486355    Africa  Eastern Africa       0.5723232
## 56    1541936   6343809583    Africa   Middle Africa      11.2717391
## 57    1693002   1217357172    Africa  Western Africa       1.9700066
## 58   24317734   8779397392    Africa  Western Africa       0.9891194
## 59   11012406   5493989673    Africa  Western Africa       1.3668245
## 60    1634196    244395463    Africa  Western Africa       0.4097285
## 61   40328313  18988282813    Africa  Eastern Africa       1.2899794
## 62    2010586   1076239050    Africa Southern Africa       1.4665377
## 63    3957990   1040653199    Africa  Western Africa       0.7203416
## 64   21079532   5026822443    Africa  Eastern Africa       0.6533407
## 65   14769824   2758392725    Africa  Eastern Africa       0.5116676
## 66   15167286   4199858651    Africa  Western Africa       0.7586368
## 67    3591400   2107593972    Africa  Western Africa       1.6077936
## 68    1247951   6636426093    Africa  Eastern Africa      14.5694737
## 69   32107739  59908047776    Africa Northern Africa       5.1119027
## 70   24321457   8972305823    Africa  Eastern Africa       1.0106985
## 71    2193643   6155469329    Africa Southern Africa       7.6878050
## 72   16291990   2781188119    Africa  Western Africa       0.4676957
## 73  159424742  85581744176    Africa  Western Africa       1.4707286
## 74   10293669   3583713093    Africa  Eastern Africa       0.9538282
## 75   12956791   6984284544    Africa  Western Africa       1.4768337
## 76      93081    760361490    Africa  Eastern Africa      22.3803157
## 77    5775902   1574302614    Africa  Western Africa       0.7467505
## 78   51621594 187639624489    Africa Southern Africa       9.9586457
## 79   36114885  22819076998    Africa Northern Africa       1.7310873
## 80    1193148   1911603442    Africa Southern Africa       4.3894552
## 81   45648525  19965679449    Africa  Eastern Africa       1.1982970
## 82    6390851   1595792895    Africa  Western Africa       0.6841085
## 83   10639194  33161453137    Africa Northern Africa       8.5394905
## 84   33149417  12701095116    Africa  Eastern Africa       1.0497174
## 85   13917439   5587389858    Africa  Eastern Africa       1.0999091
## 86   13973897   4032423429    Africa  Eastern Africa       0.7905980
##                group
## 1     Northern Africa
## 2  Sub-Saharan Africa
## 3  Sub-Saharan Africa
## 4  Sub-Saharan Africa
## 5  Sub-Saharan Africa
## 6  Sub-Saharan Africa
```

```
## 7  Sub-Saharan Africa
## 8  Sub-Saharan Africa
## 9  Sub-Saharan Africa
## 10 Sub-Saharan Africa
## 11 Sub-Saharan Africa
## 12    Northern Africa
## 13 Sub-Saharan Africa
## 14 Sub-Saharan Africa
## 15 Sub-Saharan Africa
## 16 Sub-Saharan Africa
## 17 Sub-Saharan Africa
## 18 Sub-Saharan Africa
## 19 Sub-Saharan Africa
## 20 Sub-Saharan Africa
## 21 Sub-Saharan Africa
## 22 Sub-Saharan Africa
## 23 Sub-Saharan Africa
## 24    Northern Africa
## 25 Sub-Saharan Africa
## 26 Sub-Saharan Africa
## 27 Sub-Saharan Africa
## 28 Sub-Saharan Africa
## 29 Sub-Saharan Africa
## 30 Sub-Saharan Africa
## 31 Sub-Saharan Africa
## 32    Northern Africa
## 33 Sub-Saharan Africa
## 34 Sub-Saharan Africa
## 35    Northern Africa
## 36 Sub-Saharan Africa
## 37 Sub-Saharan Africa
## 38    Northern Africa
## 39 Sub-Saharan Africa
## 40 Sub-Saharan Africa
## 41 Sub-Saharan Africa
## 42 Sub-Saharan Africa
## 43 Sub-Saharan Africa
## 44 Sub-Saharan Africa
## 45 Sub-Saharan Africa
## 46 Sub-Saharan Africa
## 47 Sub-Saharan Africa
## 48 Sub-Saharan Africa
## 49 Sub-Saharan Africa
## 50 Sub-Saharan Africa
## 51 Sub-Saharan Africa
## 52    Northern Africa
## 53 Sub-Saharan Africa
## 54 Sub-Saharan Africa
## 55 Sub-Saharan Africa
## 56 Sub-Saharan Africa
## 57 Sub-Saharan Africa
## 58 Sub-Saharan Africa
## 59 Sub-Saharan Africa
## 60 Sub-Saharan Africa
```

```
## 61 Sub-Saharan Africa
## 62 Sub-Saharan Africa
## 63 Sub-Saharan Africa
## 64 Sub-Saharan Africa
## 65 Sub-Saharan Africa
## 66 Sub-Saharan Africa
## 67 Sub-Saharan Africa
## 68 Sub-Saharan Africa
## 69    Northern Africa
## 70 Sub-Saharan Africa
## 71 Sub-Saharan Africa
## 72 Sub-Saharan Africa
## 73 Sub-Saharan Africa
## 74 Sub-Saharan Africa
## 75 Sub-Saharan Africa
## 76 Sub-Saharan Africa
## 77 Sub-Saharan Africa
## 78 Sub-Saharan Africa
## 79    Northern Africa
## 80 Sub-Saharan Africa
## 81 Sub-Saharan Africa
## 82 Sub-Saharan Africa
## 83    Northern Africa
## 84 Sub-Saharan Africa
## 85 Sub-Saharan Africa
## 86 Sub-Saharan Africa
```

```
p <- daydollars %>% ggplot(aes(dollars_per_day)) +
scale_x_continuous(trans = "log2") + geom_density() + facet_grid(.~year)
p
```



10. Now we are going to edit the code from Exercise 9 to show stacked histograms of each region in Africa.

```
daydollars <- gapminder %>%
mutate(dollars_per_day = gdp/population/365) %>% filter(continent == "Africa" & year%in%c(1970,2010) &
daydollars
```

```
##                      country year infant_mortality life_expectancy fertility
## 1                    Algeria 1970            146.0           52.41      7.64
## 2                      Benin 1970            157.1           43.93      6.75
## 3                   Botswana 1970             85.3           54.30      6.64
## 4               Burkina Faso 1970            149.3           40.27      6.62
## 5                    Burundi 1970            146.4           42.76      7.31
## 6                   Cameroon 1970            126.2           48.97      6.21
## 7   Central African Republic 1970            137.0           43.36      5.95
## 8                       Chad 1970            135.9           45.72      6.53
## 9            Congo, Dem. Rep. 1970            149.0           48.13      6.21
## 10               Congo, Rep. 1970             88.5           52.85      6.26
## 11             Cote d'Ivoire 1970            161.0           45.38      7.91
## 12                     Egypt 1970            162.0           52.54      5.94
## 13                     Gabon 1970               NA           45.55      5.08
## 14                    Gambia 1970            126.0           43.31      6.09
## 15                     Ghana 1970            120.1           50.08      6.95
## 16             Guinea-Bissau 1970               NA           45.50      6.07
## 17                     Kenya 1970             91.3           53.83      8.08
## 18                   Lesotho 1970            131.6           49.67      5.81
## 19                   Liberia 1970            191.3           40.10      6.70
```

```
## 20              Madagascar 1970              93.2           47.77      7.33
## 21                  Malawi 1970             207.7           41.62      7.30
## 22                    Mali 1970             195.7           34.51      6.90
## 23              Mauritania 1970             108.5           49.77      6.78
## 24                 Morocco 1970             120.8           54.34      6.69
## 25                   Niger 1970             137.6           38.24      7.42
## 26                 Nigeria 1970             168.9           41.79      6.47
## 27                  Rwanda 1970             129.4           45.58      8.23
## 28                 Senegal 1970             121.7           39.59      7.34
## 29              Seychelles 1970              54.1           64.62      5.76
## 30            Sierra Leone 1970             191.0           43.15      6.70
## 31            South Africa 1970                NA           52.77      5.59
## 32                   Sudan 1970              94.7           54.26      6.89
## 33               Swaziland 1970             119.3           48.79      6.88
## 34                    Togo 1970             132.8           47.72      7.08
## 35                 Tunisia 1970             122.2           52.94      6.44
## 36                  Zambia 1970             109.3           53.88      7.44
## 37                Zimbabwe 1970              72.4           57.22      7.42
## 38                 Algeria 2010              23.5           76.00      2.82
## 39                  Angola 2010             109.6           57.60      6.22
## 40                   Benin 2010              71.0           60.80      5.10
## 41                Botswana 2010              39.8           55.60      2.76
## 42            Burkina Faso 2010              69.7           59.00      5.87
## 43                 Burundi 2010              63.8           60.40      6.30
## 44                Cameroon 2010              66.2           57.80      5.02
## 45              Cape Verde 2010              23.3           71.10      2.43
## 46 Central African Republic 2010             101.7           47.90      4.63
## 47                    Chad 2010              93.6           55.80      6.60
## 48                 Comoros 2010              63.1           67.70      4.92
## 49        Congo, Dem. Rep. 2010              84.8           58.40      6.25
## 50             Congo, Rep. 2010              42.2           60.40      5.07
## 51           Cote d'Ivoire 2010              76.9           56.60      4.91
## 52                   Egypt 2010              24.3           70.10      2.88
## 53       Equatorial Guinea 2010              78.9           58.60      5.14
## 54                 Eritrea 2010              39.4           60.10      4.97
## 55                Ethiopia 2010              50.8           62.10      4.90
## 56                   Gabon 2010              42.8           63.00      4.21
## 57                  Gambia 2010              51.7           66.50      5.80
## 58                   Ghana 2010              50.2           62.90      4.05
## 59                  Guinea 2010              71.2           57.90      5.17
## 60           Guinea-Bissau 2010              73.4           54.30      5.12
## 61                   Kenya 2010              42.4           62.90      4.62
## 62                 Lesotho 2010              75.2           46.40      3.21
## 63                 Liberia 2010              65.2           60.80      5.02
## 64              Madagascar 2010              42.1           62.40      4.65
## 65                  Malawi 2010              57.5           55.40      5.64
## 66                    Mali 2010              82.9           59.20      6.84
## 67              Mauritania 2010              70.1           68.60      4.84
## 68               Mauritius 2010              13.3           73.40      1.52
## 69                 Morocco 2010              28.5           73.70      2.58
## 70              Mozambique 2010              71.9           54.40      5.41
## 71                 Namibia 2010              37.5           61.40      3.23
## 72                   Niger 2010              66.1           59.20      7.58
## 73                 Nigeria 2010              81.5           61.20      6.02
```

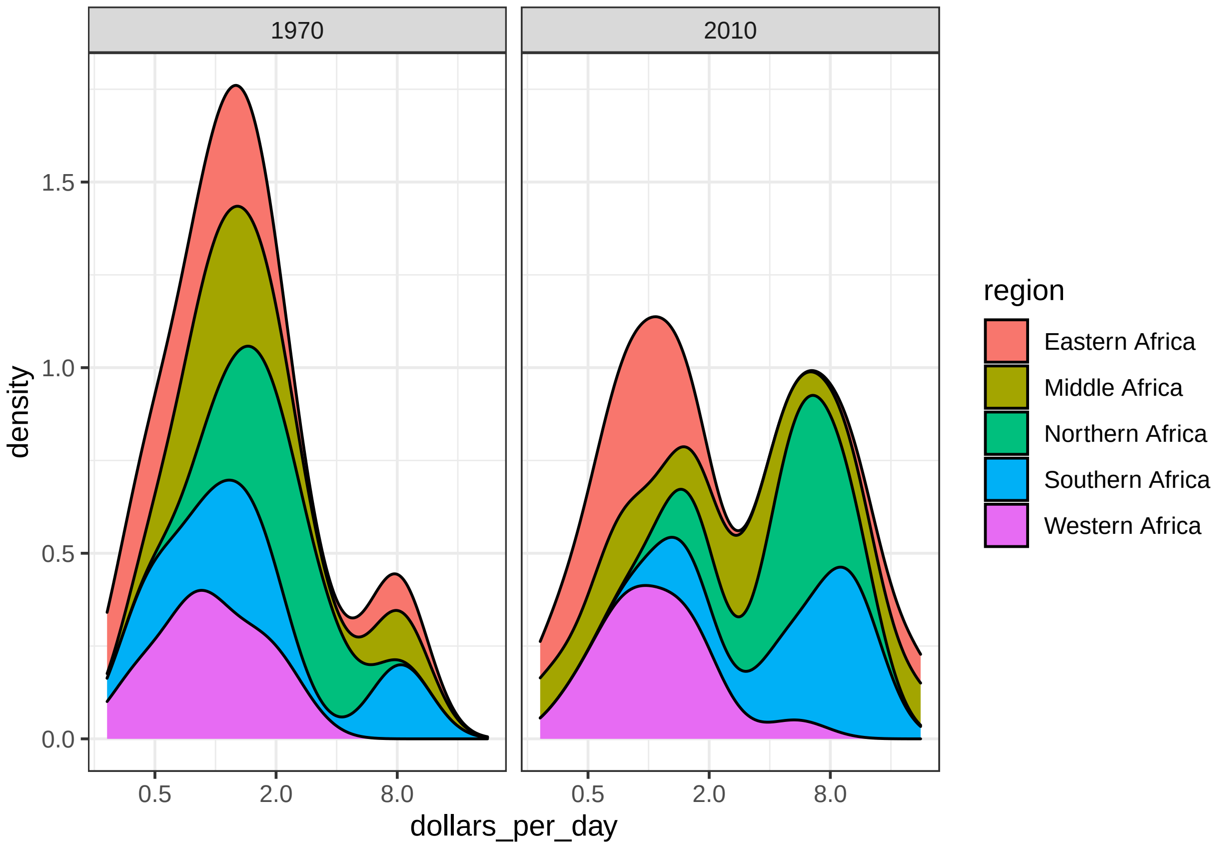
```
## 74                  Rwanda 2010            43.8           65.10      4.84
## 75                 Senegal 2010            46.7           64.20      5.05
## 76              Seychelles 2010            12.2           73.10      2.26
## 77            Sierra Leone 2010           107.0           55.00      4.94
## 78            South Africa 2010            38.2           54.90      2.47
## 79                   Sudan 2010            53.3           66.10      4.64
## 80               Swaziland 2010            59.1           46.40      3.56
## 81                Tanzania 2010            42.4           61.40      5.43
## 82                    Togo 2010            59.3           58.70      4.79
## 83                 Tunisia 2010            14.9           77.10      2.04
## 84                  Uganda 2010            49.5           57.80      6.16
## 85                  Zambia 2010            52.9           53.10      5.81
## 86                Zimbabwe 2010            55.8           49.10      3.72
##    population         gdp continent          region dollars_per_day
## 1    14550033 19741305571    Africa Northern Africa       3.7172265
## 2     2907769   831774871    Africa  Western Africa       0.7837057
## 3      693021   283867117    Africa Southern Africa       1.1222144
## 4     5624597   795164207    Africa  Western Africa       0.3873223
## 5     3457113   524049198    Africa  Eastern Africa       0.4153035
## 6     6770967  3372153343    Africa   Middle Africa       1.3644693
## 7     1828710   647622869    Africa   Middle Africa       0.9702518
## 8     3644911   829387598    Africa   Middle Africa       0.6234157
## 9    20009902  6728080745    Africa   Middle Africa       0.9211988
## 10    1335090   939633199    Africa   Middle Africa       1.9282127
## 11    5241914  4619775632    Africa  Western Africa       2.4145607
## 12   34808599 20331718433    Africa Northern Africa       1.6002752
## 13     590119  1722664256    Africa   Middle Africa       7.9977566
## 14     447283   247459869    Africa  Western Africa       1.5157568
## 15    8596977  2549677064    Africa  Western Africa       0.8125434
## 16     711828   104038537    Africa  Western Africa       0.4004297
## 17   11252466  3276361787    Africa  Eastern Africa       0.7977215
## 18    1032240   184783955    Africa Southern Africa       0.4904454
## 19    1419728  1094083642    Africa  Western Africa       2.1113125
## 20    6576301  2807129955    Africa  Eastern Africa       1.1694670
## 21    4603739   549382768    Africa  Eastern Africa       0.3269426
## 22    5949043  1038617256    Africa  Western Africa       0.4783167
## 23    1148908   700627427    Africa  Western Africa       1.6707406
## 24   16039600 12097898528    Africa Northern Africa       2.0664435
## 25    4497355  1343819364    Africa  Western Africa       0.8186360
## 26   56131844 19793025795    Africa  Western Africa       0.9660732
## 27    3754546   809941587    Africa  Eastern Africa       0.5910217
## 28    4217754  2266115562    Africa  Western Africa       1.4720005
## 29      52364   141888524    Africa  Eastern Africa       7.4237202
## 30    2514151   739785784    Africa  Western Africa       0.8061610
## 31   22502502 68558449204    Africa Southern Africa       8.3471326
## 32   10232758  3901968151    Africa Northern Africa       1.0447158
## 33     445844   257078586    Africa Southern Africa       1.5797564
## 34    2115521   618863063    Africa  Western Africa       0.8014646
## 35    5060393  4688590613    Africa Northern Africa       2.5384301
## 36    4185378  2384401746    Africa  Eastern Africa       1.5608166
## 37    5206311  2682438620    Africa  Eastern Africa       1.4115843
## 38   36036159 79164339611    Africa Northern Africa       6.0186382
## 39   21219954 26125663270    Africa   Middle Africa       3.3731063
## 40    9509798  3336801340    Africa  Western Africa       0.9613161
```

```
## 41    2047831   8408166868    Africa Southern Africa      11.2490111
## 42   15632066   4655655008    Africa  Western Africa       0.8159650
## 43    9461117   1158914103    Africa  Eastern Africa       0.3355954
## 44   20590666  13986616694    Africa   Middle Africa       1.8610130
## 45     490379    971606715    Africa  Western Africa       5.4283242
## 46    4444973   1054122016    Africa   Middle Africa       0.6497240
## 47   11896380   3369354207    Africa   Middle Africa       0.7759594
## 48     698695    247231031    Africa  Eastern Africa       0.9694434
## 49   65938712   6961485000    Africa   Middle Africa       0.2892468
## 50    4066078   5067059617    Africa   Middle Africa       3.4141881
## 51   20131707  11603002049    Africa  Western Africa       1.5790537
## 52   82040994 160258746162    Africa Northern Africa       5.3517764
## 53     728710   5979285835    Africa   Middle Africa      22.4802803
## 54    4689664    771116883    Africa  Eastern Africa       0.4504905
## 55   87561814  18291486355    Africa  Eastern Africa       0.5723232
## 56    1541936   6343809583    Africa   Middle Africa      11.2717391
## 57    1693002   1217357172    Africa  Western Africa       1.9700066
## 58   24317734   8779397392    Africa  Western Africa       0.9891194
## 59   11012406   5493989673    Africa  Western Africa       1.3668245
## 60    1634196    244395463    Africa  Western Africa       0.4097285
## 61   40328313  18988282813    Africa  Eastern Africa       1.2899794
## 62    2010586   1076239050    Africa Southern Africa       1.4665377
## 63    3957990   1040653199    Africa  Western Africa       0.7203416
## 64   21079532   5026822443    Africa  Eastern Africa       0.6533407
## 65   14769824   2758392725    Africa  Eastern Africa       0.5116676
## 66   15167286   4199858651    Africa  Western Africa       0.7586368
## 67    3591400   2107593972    Africa  Western Africa       1.6077936
## 68    1247951   6636426093    Africa  Eastern Africa      14.5694737
## 69   32107739  59908047776    Africa Northern Africa       5.1119027
## 70   24321457   8972305823    Africa  Eastern Africa       1.0106985
## 71    2193643   6155469329    Africa Southern Africa       7.6878050
## 72   16291990   2781188119    Africa  Western Africa       0.4676957
## 73  159424742  85581744176    Africa  Western Africa       1.4707286
## 74   10293669   3583713093    Africa  Eastern Africa       0.9538282
## 75   12956791   6984284544    Africa  Western Africa       1.4768337
## 76      93081    760361490    Africa  Eastern Africa      22.3803157
## 77    5775902   1574302614    Africa  Western Africa       0.7467505
## 78   51621594 187639624489    Africa Southern Africa       9.9586457
## 79   36114885  22819076998    Africa Northern Africa       1.7310873
## 80    1193148   1911603442    Africa Southern Africa       4.3894552
## 81   45648525  19965679449    Africa  Eastern Africa       1.1982970
## 82    6390851   1595792895    Africa  Western Africa       0.6841085
## 83   10639194  33161453137    Africa Northern Africa       8.5394905
## 84   33149417  12701095116    Africa  Eastern Africa       1.0497174
## 85   13917439   5587389858    Africa  Eastern Africa       1.0999091
## 86   13973897   4032423429    Africa  Eastern Africa       0.7905980
##                 group
## 1     Northern Africa
## 2  Sub-Saharan Africa
## 3  Sub-Saharan Africa
## 4  Sub-Saharan Africa
## 5  Sub-Saharan Africa
## 6  Sub-Saharan Africa
## 7  Sub-Saharan Africa
```

```
## 8  Sub-Saharan Africa
## 9  Sub-Saharan Africa
## 10 Sub-Saharan Africa
## 11 Sub-Saharan Africa
## 12    Northern Africa
## 13 Sub-Saharan Africa
## 14 Sub-Saharan Africa
## 15 Sub-Saharan Africa
## 16 Sub-Saharan Africa
## 17 Sub-Saharan Africa
## 18 Sub-Saharan Africa
## 19 Sub-Saharan Africa
## 20 Sub-Saharan Africa
## 21 Sub-Saharan Africa
## 22 Sub-Saharan Africa
## 23 Sub-Saharan Africa
## 24    Northern Africa
## 25 Sub-Saharan Africa
## 26 Sub-Saharan Africa
## 27 Sub-Saharan Africa
## 28 Sub-Saharan Africa
## 29 Sub-Saharan Africa
## 30 Sub-Saharan Africa
## 31 Sub-Saharan Africa
## 32    Northern Africa
## 33 Sub-Saharan Africa
## 34 Sub-Saharan Africa
## 35    Northern Africa
## 36 Sub-Saharan Africa
## 37 Sub-Saharan Africa
## 38    Northern Africa
## 39 Sub-Saharan Africa
## 40 Sub-Saharan Africa
## 41 Sub-Saharan Africa
## 42 Sub-Saharan Africa
## 43 Sub-Saharan Africa
## 44 Sub-Saharan Africa
## 45 Sub-Saharan Africa
## 46 Sub-Saharan Africa
## 47 Sub-Saharan Africa
## 48 Sub-Saharan Africa
## 49 Sub-Saharan Africa
## 50 Sub-Saharan Africa
## 51 Sub-Saharan Africa
## 52    Northern Africa
## 53 Sub-Saharan Africa
## 54 Sub-Saharan Africa
## 55 Sub-Saharan Africa
## 56 Sub-Saharan Africa
## 57 Sub-Saharan Africa
## 58 Sub-Saharan Africa
## 59 Sub-Saharan Africa
## 60 Sub-Saharan Africa
## 61 Sub-Saharan Africa
```

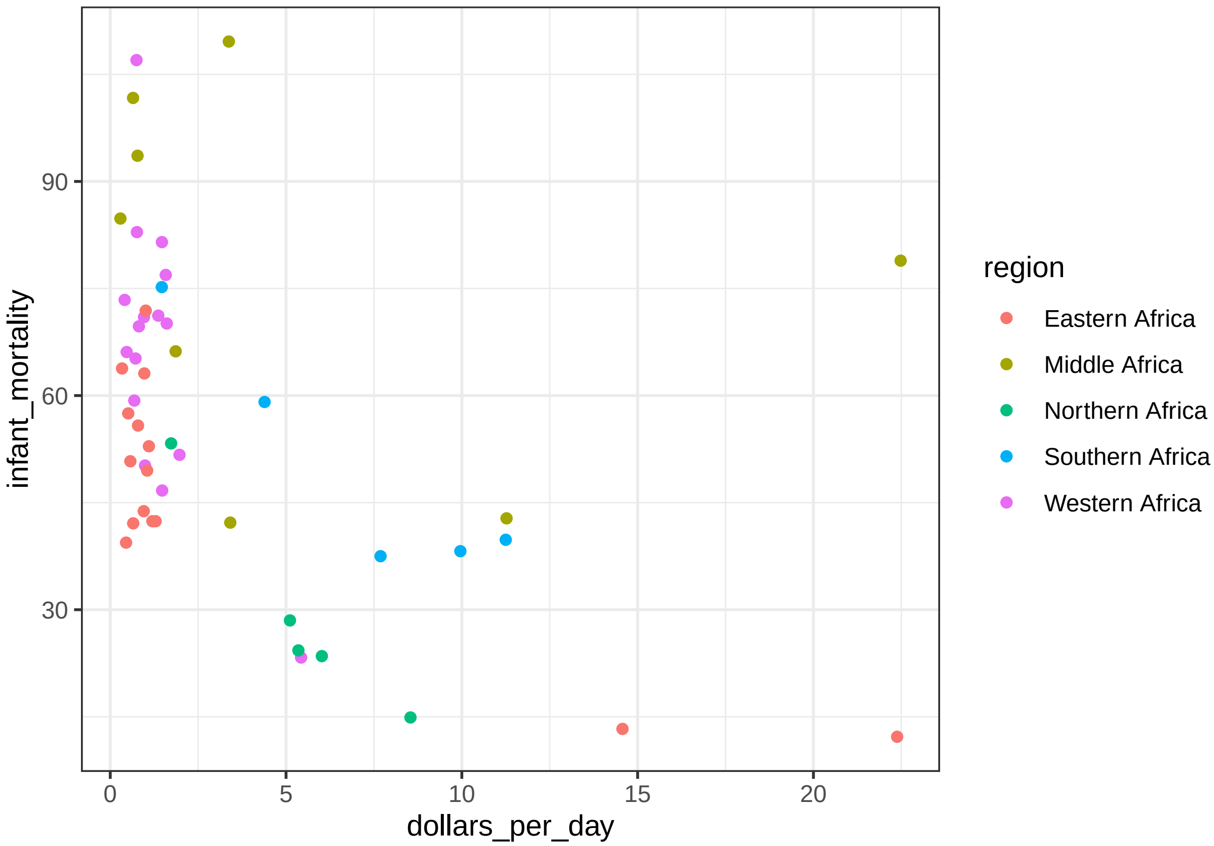
```
## 62 Sub-Saharan Africa
## 63 Sub-Saharan Africa
## 64 Sub-Saharan Africa
## 65 Sub-Saharan Africa
## 66 Sub-Saharan Africa
## 67 Sub-Saharan Africa
## 68 Sub-Saharan Africa
## 69    Northern Africa
## 70 Sub-Saharan Africa
## 71 Sub-Saharan Africa
## 72 Sub-Saharan Africa
## 73 Sub-Saharan Africa
## 74 Sub-Saharan Africa
## 75 Sub-Saharan Africa
## 76 Sub-Saharan Africa
## 77 Sub-Saharan Africa
## 78 Sub-Saharan Africa
## 79    Northern Africa
## 80 Sub-Saharan Africa
## 81 Sub-Saharan Africa
## 82 Sub-Saharan Africa
## 83    Northern Africa
## 84 Sub-Saharan Africa
## 85 Sub-Saharan Africa
## 86 Sub-Saharan Africa
```

```
daydollars %>% ggplot(aes(dollars_per_day, fill = region)) +
scale_x_continuous(trans = "log2") + geom_density(bw = 0.5, position = "stack") + facet_grid(.~year)
```



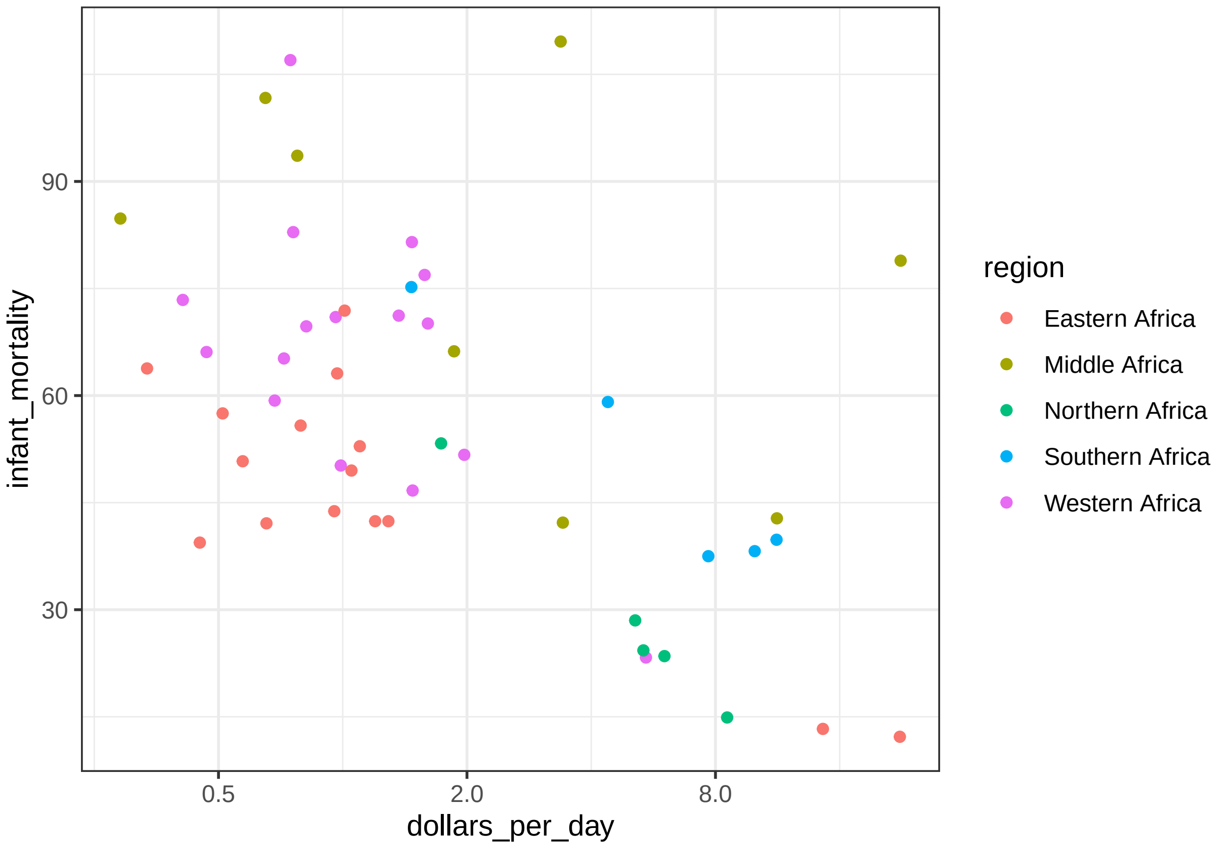
11. We are going to continue looking at patterns in the gapminder dataset by plotting infant mortality rates versus dollars per day for African countries.

```
gapminder_Africa_2010 <- gapminder %>%
mutate(dollars_per_day = gdp/population/365) %>% filter(continent == "Africa" & year == 2010 & !is.na(gd
# now make the scatter plot
gapminder_Africa_2010 %>% ggplot(aes(dollars_per_day, infant_mortality, color = region)) + geom_point()
```



12. Now we are going to transform the x axis of the plot from the previous exercise.

```
gapminder_Africa_2010 %>% ggplot(aes(dollars_per_day, infant_mortality, color = region)) + scale_x_conti
```

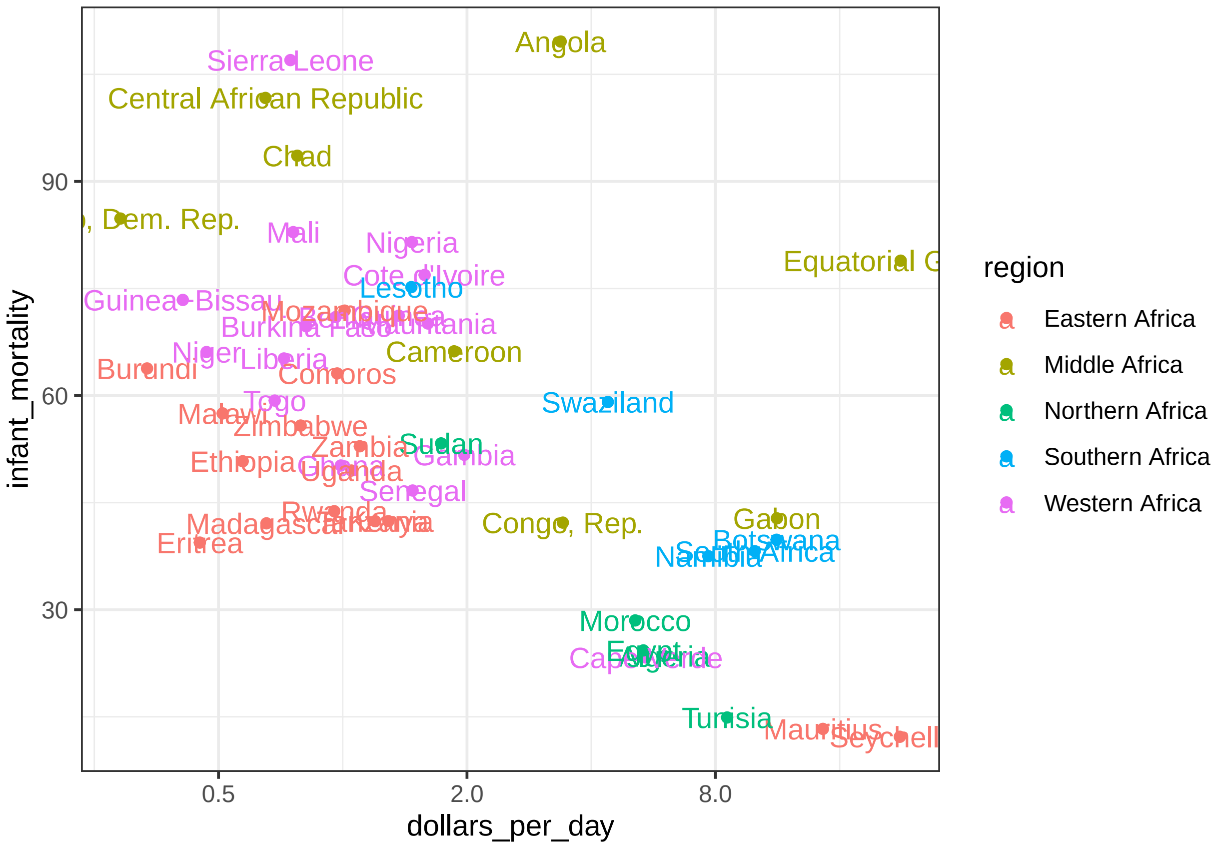


13. Note that there is a large variation in infant mortality and dollars per day among African countries.

As an example, one country has infant mortality rates of less than 20 per 1000 and dollars per day of 16, while another country has infant mortality rates over 10% and dollars per day of about 1.

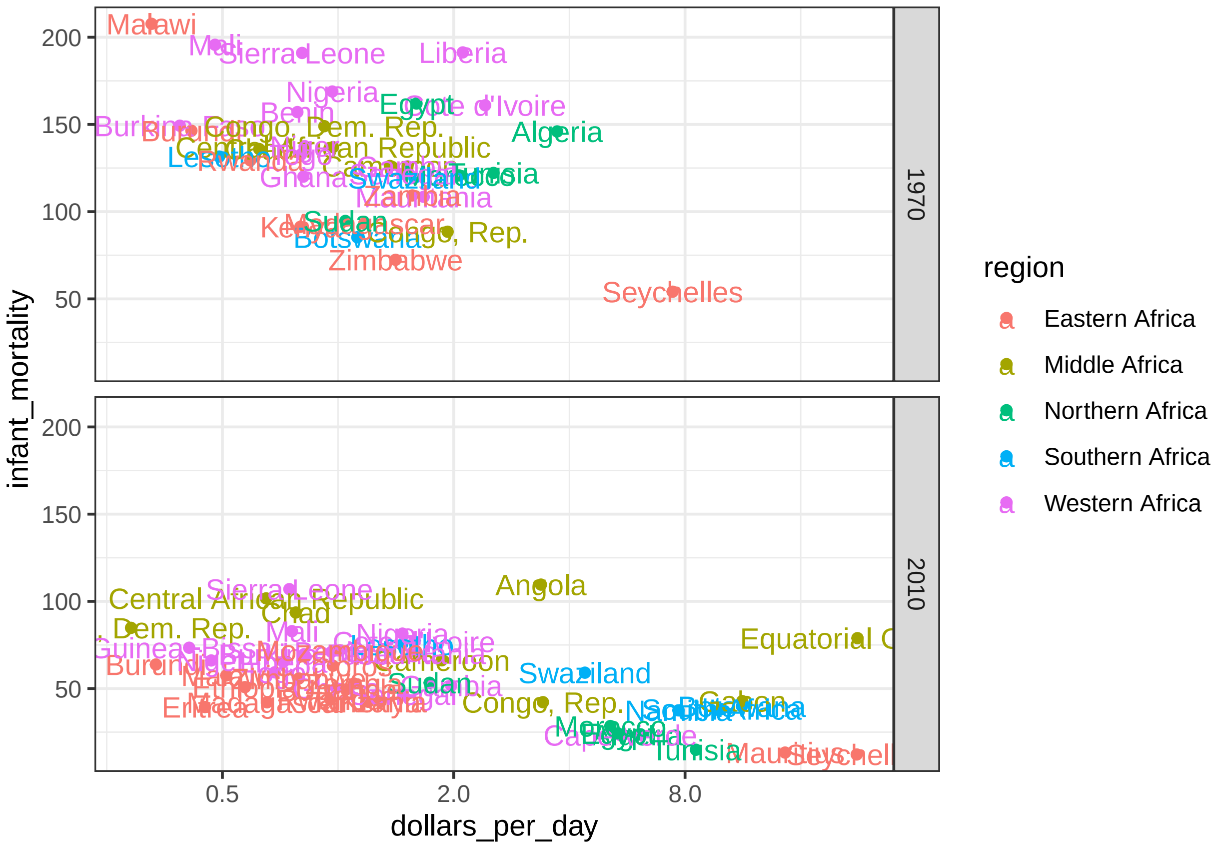
In this exercise, we will remake the plot from Exercise 12 with country names instead of points so we can identify which countries are which.

```
gapminder_Africa_2010 %>% ggplot(aes(dollars_per_day, infant_mortality, color = region, label = country
```



14. Now we are going to look at changes in the infant mortality and dollars per day patterns African countries between 1970 and 2010.

```
gapminder_Africa_1970_2019 <- gapminder %>% mutate(dollars_per_day = gdp/population/365) %>% filter(con
gapminder_Africa_1970_2019 %>% ggplot(aes(dollars_per_day, infant_mortality, color = region, label = co
```



Section 5 Overview

Section 5 covers some general principles that can serve as guides for effective data visualization.

After completing Section 5, you will:

- understand basic principles of effective data visualization.
- understand the importance of keeping your goal in mind when deciding on a visualization approach.
- understand principles for encoding data, including position, aligned lengths, angles, area, brightness, and color hue.
- know when to include the number zero in visualizations.
- be able to use techniques to ease comparisons, such as using common axes, putting visual cues to be compared adjacent to one another, and using color effectively.

Introduction to Data Visualization Principles

The textbook for this section is available here

Key points

- We aim to provide some general guidelines for effective data visualization.
- We show examples of plot styles to avoid, discuss how to improve them, and use these examples to explain research-based principles for effective visualization.
- When choosing a visualization approach, keep your goal and audience in mind.

Encoding Data Using Visual Cues

The textbook for this section is available here

You can learn more about barplots in the textbook section on barplots

Key points

- Visual cues for encoding data include position, length, angle, area, brightness and color hue.
- Position and length are the preferred way to display quantities, followed by angles, which are preferred over area. Brightness and color are even harder to quantify but can sometimes be useful.
- Pie charts represent visual cues as both angles and area, while donut charts use only area. Humans are not good at visually quantifying angles and are even worse at quantifying area. Therefore pie and donut charts should be avoided - use a bar plot instead. If you must make a pie chart, include percentages as labels.
- Bar plots represent visual cues as position and length. Humans are good at visually quantifying linear measures, making bar plots a strong alternative to pie or donut charts.

Know When to Include Zero

The textbook for this section is available here

Key points

- When using bar plots, always start at 0. It is deceptive not to start at 0 because bar plots imply length is proportional to the quantity displayed. Cutting off the y-axis can make differences look bigger than they actually are.
- When using position rather than length, it is not necessary to include 0 (scatterplot, dot plot, boxplot).

Do Not Distort Quantities

The textbook for this section is available here

Key points

- Make sure your visualizations encode the correct quantities.
- For example, if you are using a plot that relies on circle area, make sure the area (rather than the radius) is proportional to the quantity.

Order by a Meaningful Value

The textbook for this section is available here

Key points

- It is easiest to visually extract information from a plot when categories are ordered by a meaningful value. The exact value on which to order will depend on your data and the message you wish to convey with your plot.
- The default ordering for categories is alphabetical if the categories are strings or by factor level if factors. However, we rarely want alphabetical order.

Assessment - Data Visualization Principles, Part 1

1: Pie charts are appropriate:

- ☐ A. When we want to display percentages.
- ☐ B. When ggplot2 is not available.
- ☐ C. When I am in a bakery.
- ☒ D. Never. Barplots and tables are always better.

2. What is the problem with this plot?

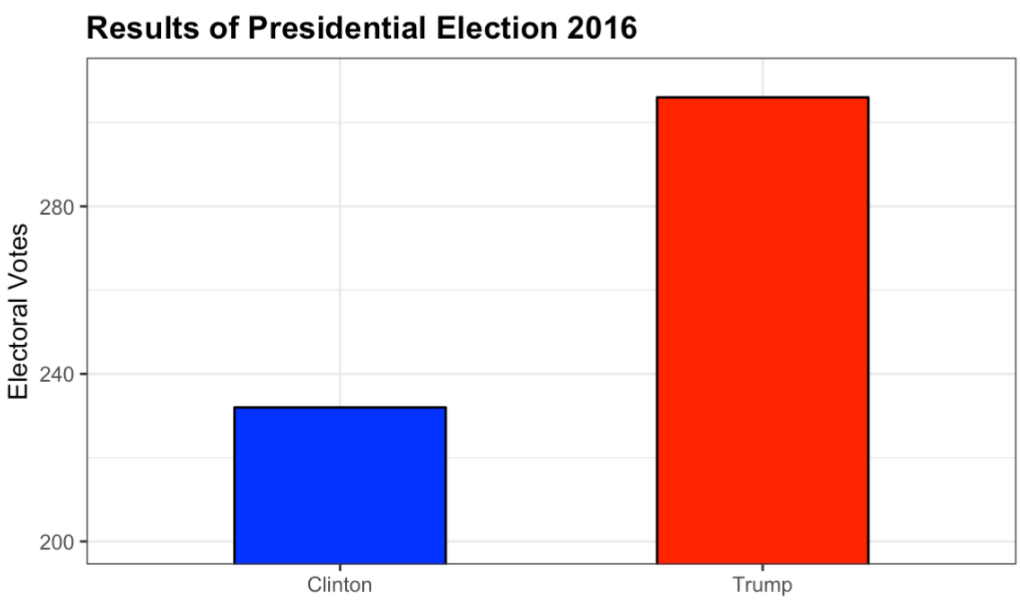


Figure 8: Result of Presidential Election 2016

- ☐ A. The values are wrong. The final vote was 306 to 232.
- ☒ B. The axis does not start at 0. Judging by the length, it appears Trump received 3 times as many votes when in fact it was about 30% more.
- ☐ C. The colors should be the same.
- ☐ D. Percentages should be shown as a pie chart.

3. Take a look at the following two plots. They show the same information: rates of measles by state in the United States for 1928.

- ☐ A. Both plots provide the same information, so they are equally good.
- ☐ B. The plot on the left is better because it orders the states alphabetically.
- ☒ C. The plot on the right is better because it orders the states by disease rate so we can quickly see the states with highest and lowest rates.
- ☐ D. Both plots should be pie charts instead.

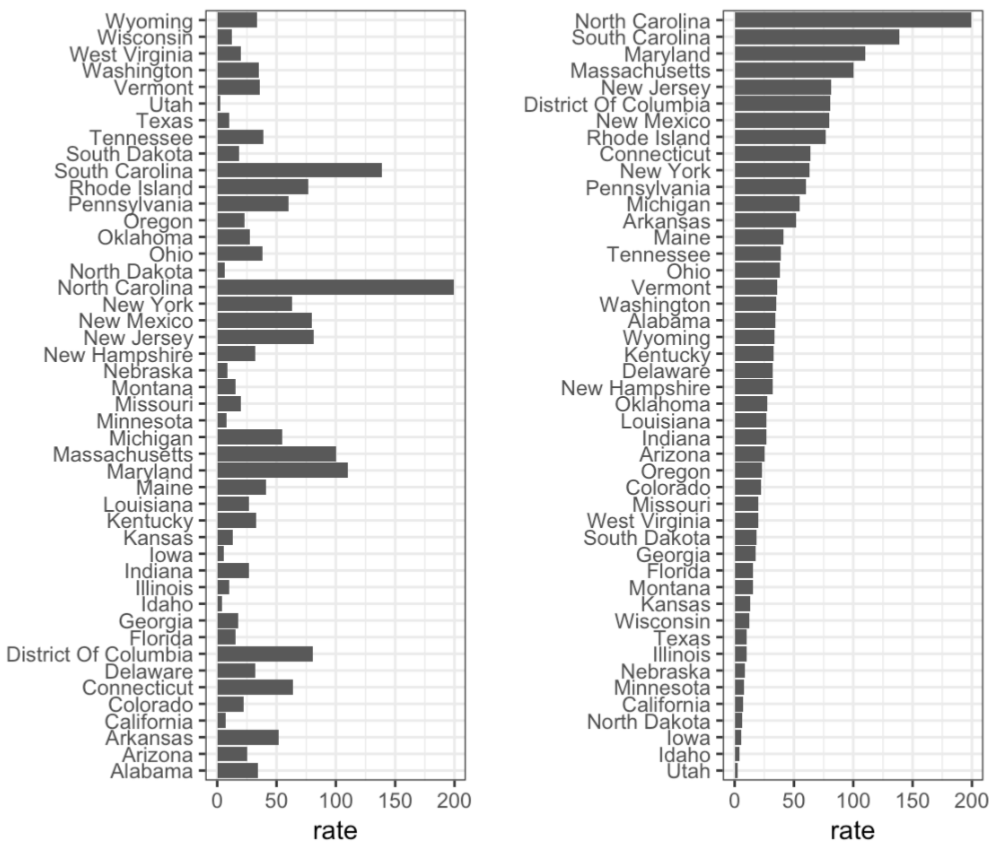


Figure 9: Rates of measles in the US for 1928

Show the Data

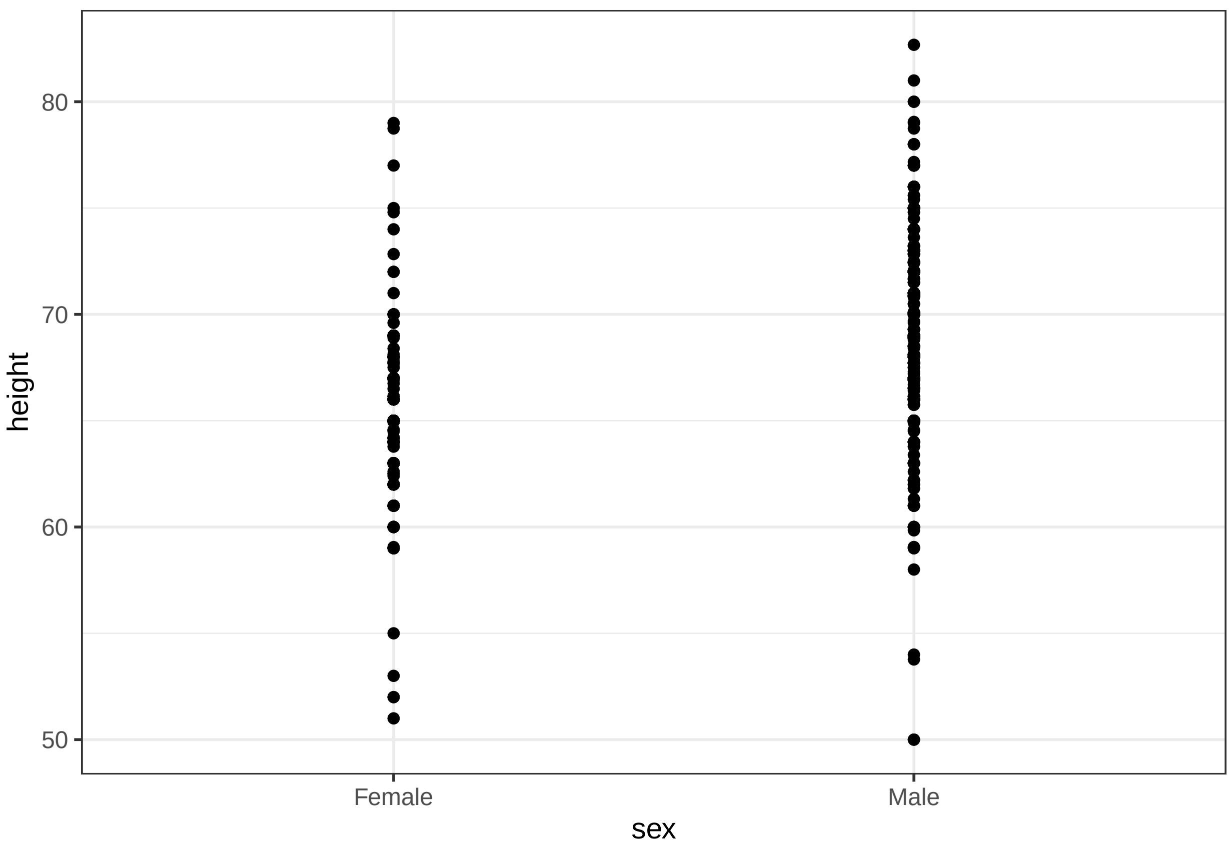
The textbook for this section is available here

Key points

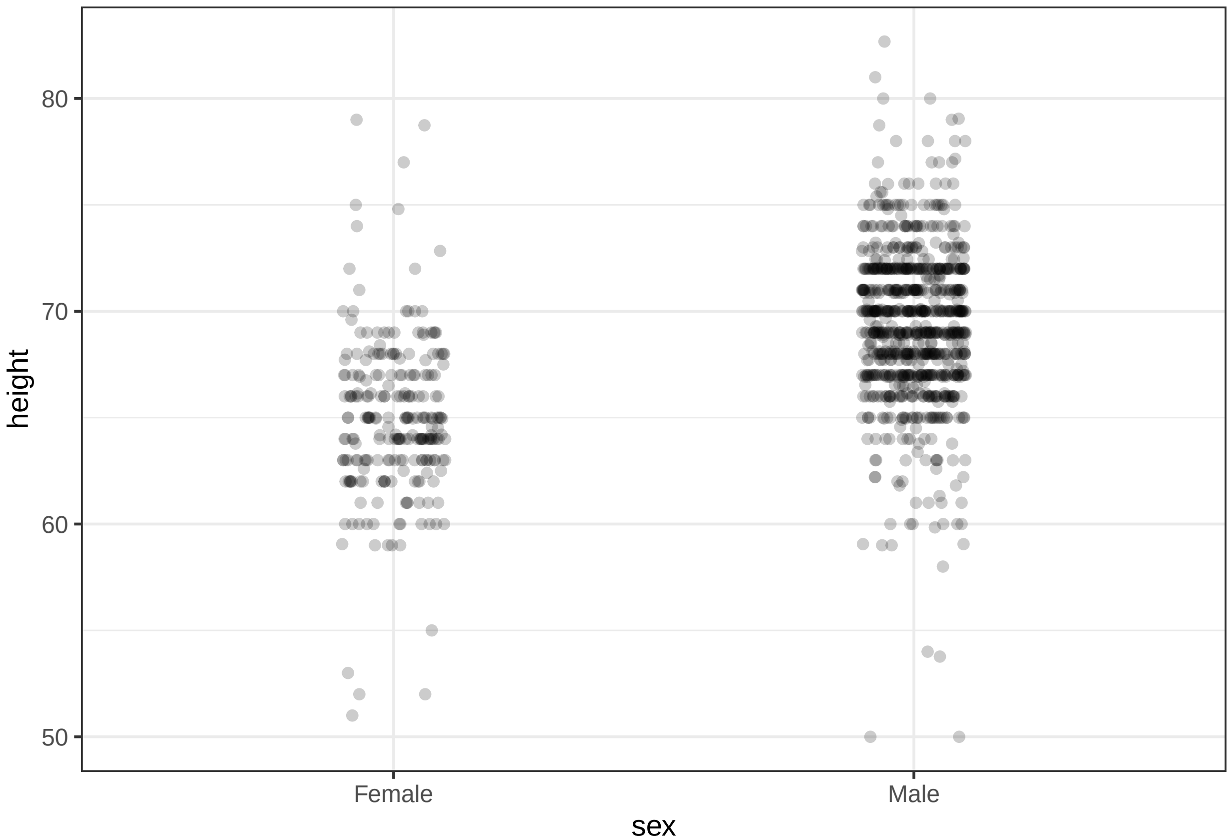
- A dynamite plot - a bar graph of group averages with error bars denoting standard errors - provides almost no information about a distribution.
- By showing the data, you provide viewers extra information about distributions. Jitter is adding a small random shift to each point in order to minimize the number of overlapping points. To add jitter, use the `geom_jitter` geometry instead of `geom_point`.
- Alpha blending is making points somewhat transparent, helping visualize the density of overlapping points. Add an `alpha` argument to the geometry.

Code

```
# dot plot showing the data
heights %>% ggplot(aes(sex, height)) + geom_point()
```




```
# jittered, alpha blended point plot
heights %>% ggplot(aes(sex, height)) + geom_jitter(width = 0.1, alpha = 0.2)
```



Ease Comparisons: Use Common Axes

The textbook for this section is available here

Key points

- Ease comparisons by keeping axes the same when comparing data across multiple plots.
- Align plots vertically to see horizontal changes. Align plots horizontally to see vertical changes.
- Bar plots are useful for showing one number but not useful for showing distributions.

Consider Transformations

The textbook for this section is available here

Key points

- Use transformations when warranted to ease visual interpretation.
- The log transformation is useful for data with multiplicative changes. The logistic transformation is useful for fold changes in odds. The square root transformation is useful for count data.
- We learned how to apply transformations earlier in the course.

Ease Comparisons: Compared Visual Cues Should Be Adjacent

The textbook for this section is available:

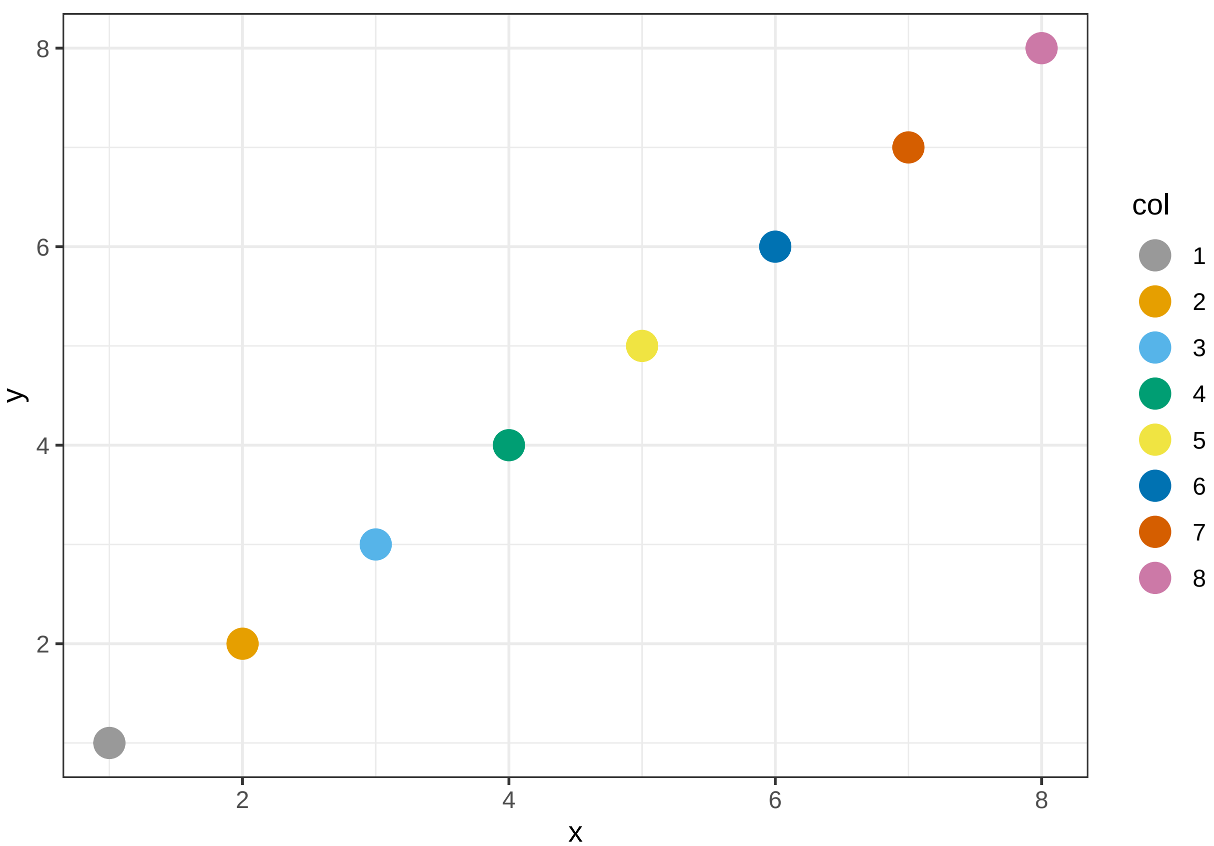
- Compared visual cues being adjacent
- Using color
- Considering the color blind

Key points

- When two groups are to be compared, it is optimal to place them adjacent in the plot.
- Use color to encode groups to be compared.
- Consider using a color blind friendly palette.

Code

```
color_blind_friendly_cols <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E0

p1 <- data.frame(x = 1:8, y = 1:8, col = as.character(1:8)) %>%
    ggplot(aes(x, y, color = col)) +
    geom_point(size = 5)
p1 + scale_color_manual(values = color_blind_friendly_cols)
```



Assessment - Data Visualization Principles, Part 2

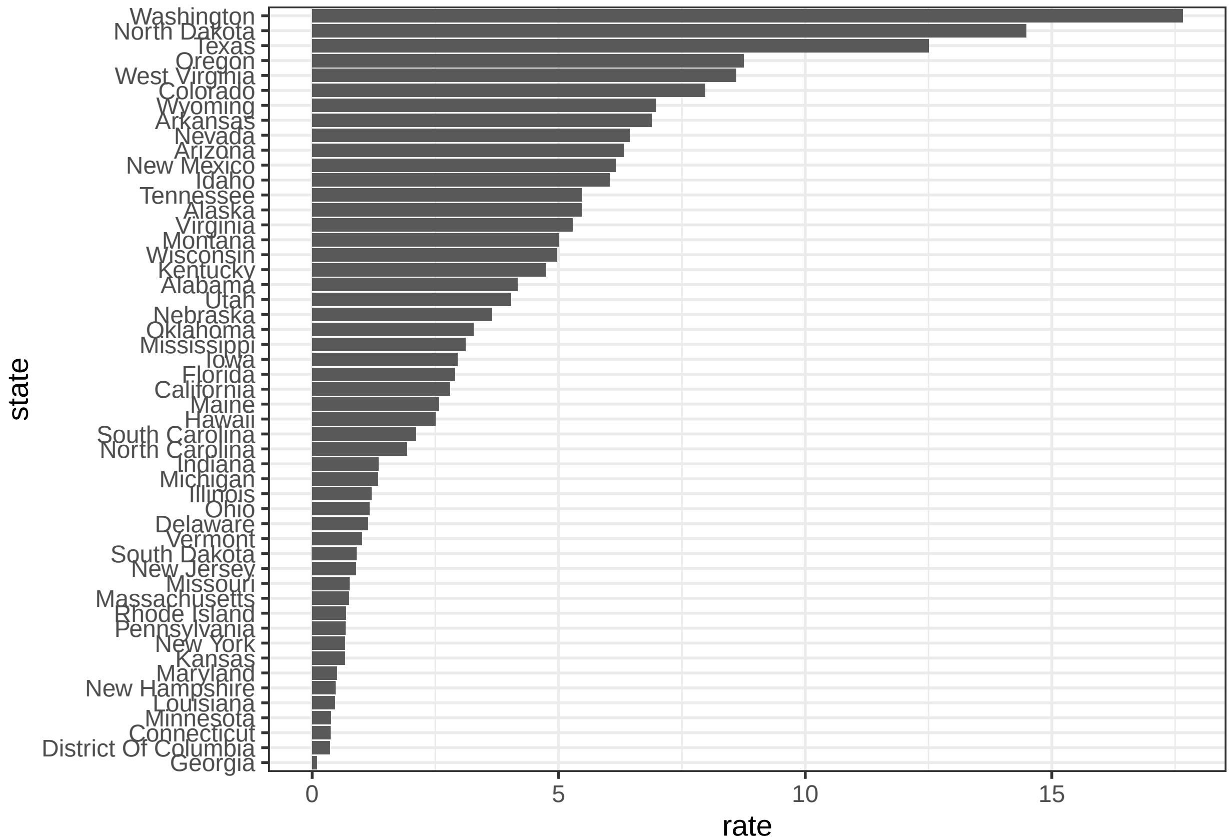
1. To make the plot on the right in the exercise from the last set of assessments, we had to reorder the levels of the states' variables.

```r
dat <- us_contagious_diseases %>%
filter(year == 1967 & disease=="Measles" & !is.na(population)) %>% mutate(rate = count / population * 10
state <- dat$state
rate <- dat$count/(dat$population/10000)*(52/dat$weeks_reporting)
state = reorder(state, rate)
levels(state)
```

```
##  [1] "Georgia"              "District Of Columbia" "Connecticut"
##  [4] "Minnesota"            "Louisiana"            "New Hampshire"
##  [7] "Maryland"             "Kansas"               "New York"
## [10] "Pennsylvania"         "Rhode Island"         "Massachusetts"
## [13] "Missouri"             "New Jersey"           "South Dakota"
## [16] "Vermont"              "Delaware"             "Ohio"
## [19] "Illinois"             "Michigan"             "Indiana"
## [22] "North Carolina"       "South Carolina"       "Hawaii"
## [25] "Maine"                "California"           "Florida"
## [28] "Iowa"                 "Mississippi"          "Oklahoma"
## [31] "Nebraska"             "Utah"                 "Alabama"
## [34] "Kentucky"             "Wisconsin"            "Montana"
## [37] "Virginia"             "Alaska"               "Tennessee"
## [40] "Idaho"                "New Mexico"           "Arizona"
## [43] "Nevada"               "Arkansas"             "Wyoming"
## [46] "Colorado"             "West Virginia"        "Oregon"
## [49] "Texas"                "North Dakota"         "Washington"
```

2. Now we are going to customize this plot a little more by creating a rate variable and reordering by that variable instead.

```r
dat <- us_contagious_diseases %>% filter(year == 1967 & disease=="Measles" & count>0 & !is.na(populatio
  mutate(rate = count / population * 10000 * 52 / weeks_reporting)
dat %>% mutate(state = reorder(state, rate)) %>% ggplot(aes(state, rate)) +
  geom_bar(stat="identity") +
  coord_flip()
```



3. Say we are interested in comparing gun homicide rates across regions of the US.

We see this plot and decide to move to a state in the western region.

What is the main problem with this interpretation?

```
library(dplyr)
library(ggplot2)
library(dslabs)
data("murders")
murders %>% mutate(rate = total/population*100000) %>%
  group_by(region) %>%
  summarize(avg = mean(rate)) %>%
  mutate(region = factor(region)) %>%
  ggplot(aes(region, avg)) +
  geom_bar(stat="identity") +
  ylab("Murder Rate Average")
```

- ☐ A. The categories are ordered alphabetically.
- ☐ B. The graph does not show standard errors.
- ☒ C. It does not show all the data. We do not see the variability within a region and it's possible that the safest states are not in the West.
- ☐ D. The Northeast has the lowest average.

4. To further investigate whether moving to the western region is a wise decision, let's make a box plot of murder rates by region, showing all points.

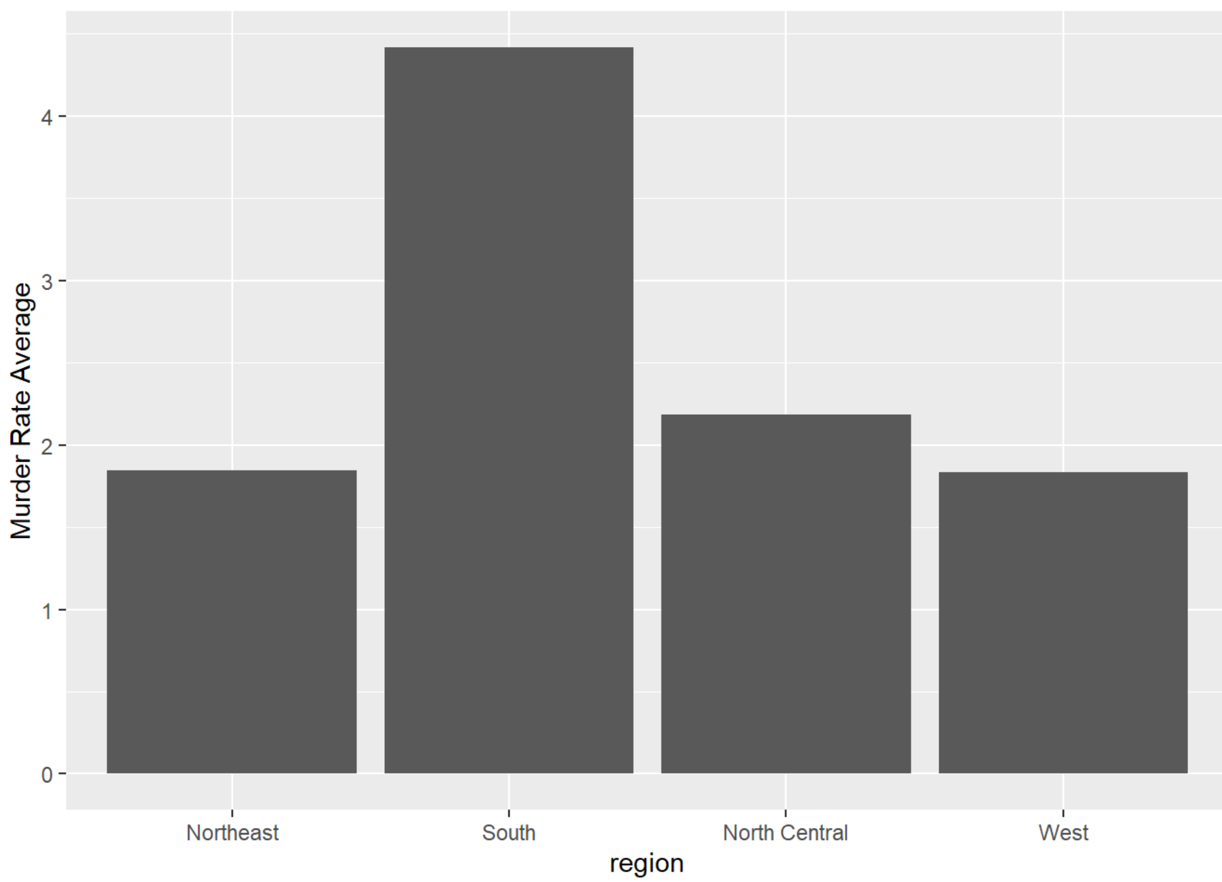
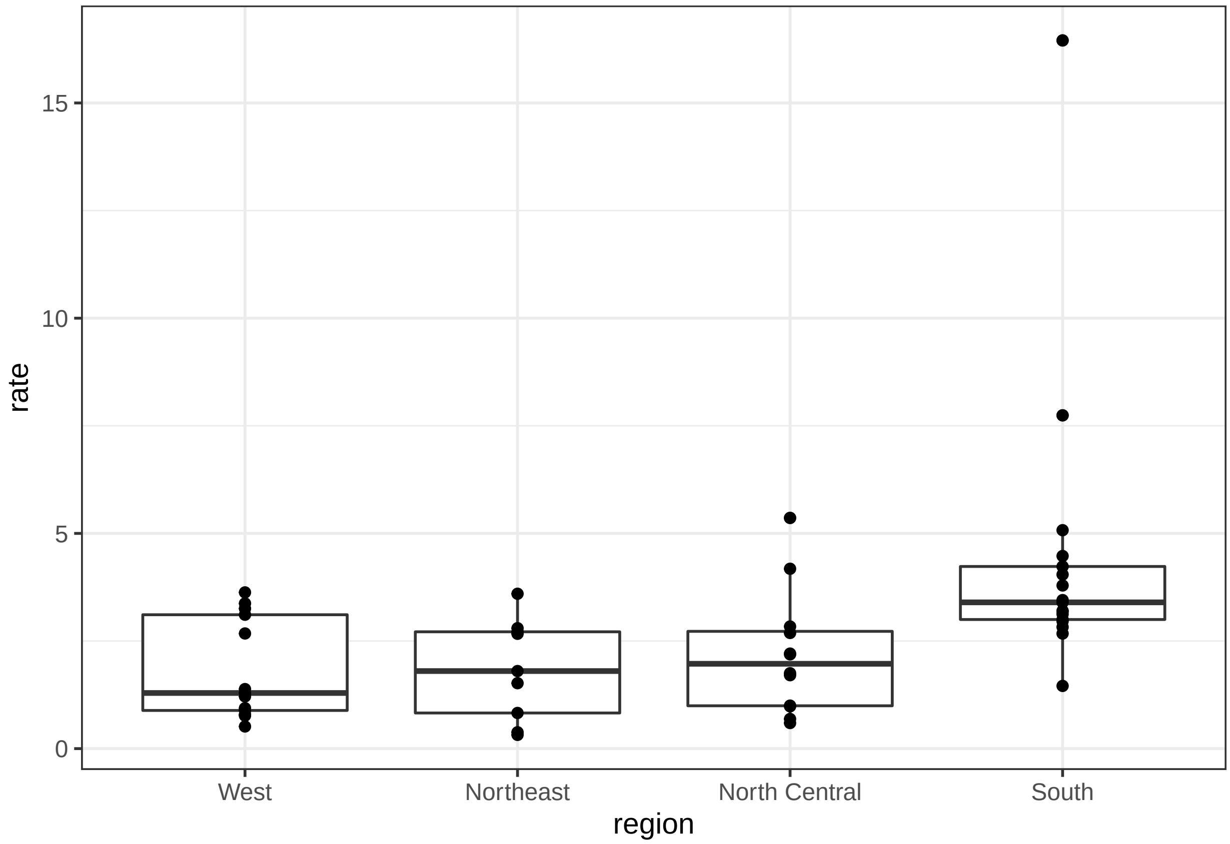


Figure 10: Rates of measles in the US for 1928

```
murders %>% mutate(rate = total/population*100000) %>% mutate(region = reorder(region, rate, FUN = medi
```



Slope Charts

The textbook for this section is available here

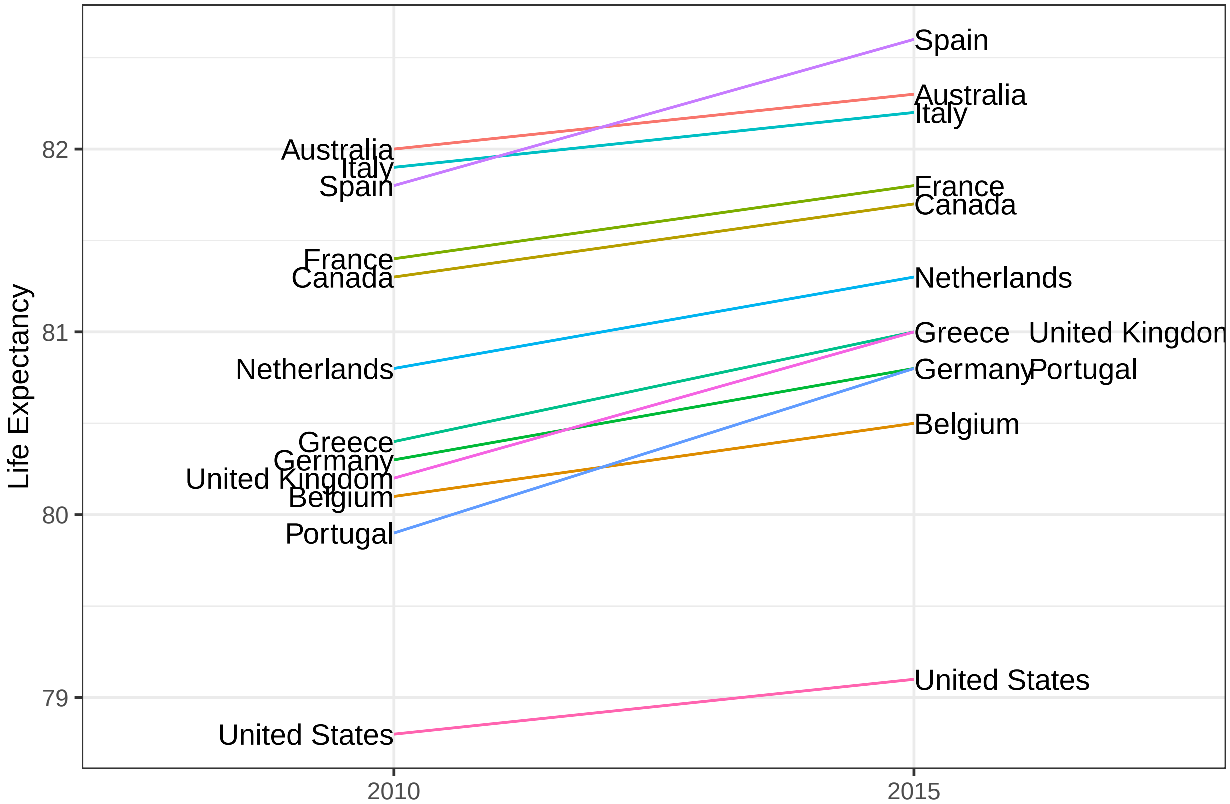
Key points

- Consider using a slope chart or Bland-Altman plot when comparing one variable at two different time points, especially for a small number of observations.
- Slope charts use angle to encode change. Use `geom_line` to create slope charts. It is useful when comparing a small number of observations.
- The Bland-Altman plot (Tukey mean difference plot, MA plot) graphs the difference between conditions on the y-axis and the mean between conditions on the x-axis. It is more appropriate for large numbers of observations than slope charts.

Code: Slope chart

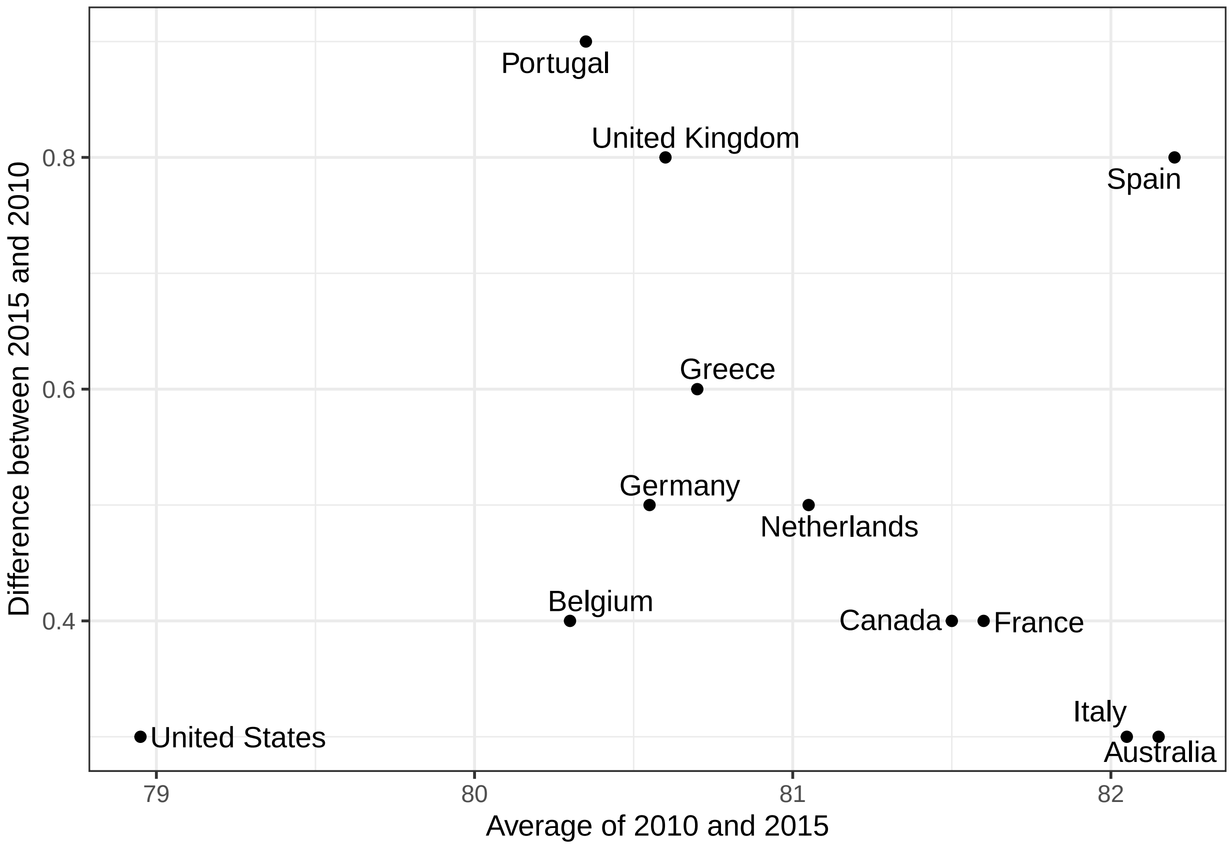
```
west <- c("Western Europe", "Northern Europe", "Southern Europe", "Northern America", "Australia and Ne

dat <- gapminder %>%
    filter(year %in% c(2010, 2015) & region %in% west & !is.na(life_expectancy) & population > 10^7)

dat %>%
    mutate(location = ifelse(year == 2010, 1, 2),
```

```
                 location = ifelse(year == 2015 & country %in% c("United Kingdom", "Portugal"), locatio
                 hjust = ifelse(year == 2010, 1, 0)) %>%
    mutate(year = as.factor(year)) %>%
    ggplot(aes(year, life_expectancy, group = country)) +
    geom_line(aes(color = country), show.legend = FALSE) +
    geom_text(aes(x = location, label = country, hjust = hjust), show.legend = FALSE) +
    xlab("") +
    ylab("Life Expectancy")
```



Code: Bland-Altman plot

```
dat %>%
    mutate(year = paste0("life_expectancy_", year)) %>%
    select(country, year, life_expectancy) %>% spread(year, life_expectancy) %>%
    mutate(average = (life_expectancy_2015 + life_expectancy_2010)/2,
                difference = life_expectancy_2015 - life_expectancy_2010) %>%
    ggplot(aes(average, difference, label = country)) +
    geom_point() +
    geom_text_repel() +
    geom_abline(lty = 2) +
    xlab("Average of 2010 and 2015") +
    ylab("Difference between 2015 and 2010")
```



Encoding a Third Variable

The textbook for this section is available here

Key points

- Encode a categorical third variable on a scatterplot using color hue or shape. Use the shape argument to control shape.
- Encode a continuous third variable on a using color intensity or size.

Case Study: Vaccines

The textbook for this section is available here. Information on color palettes can be found in the textbook section on encoding a third variable

Key points

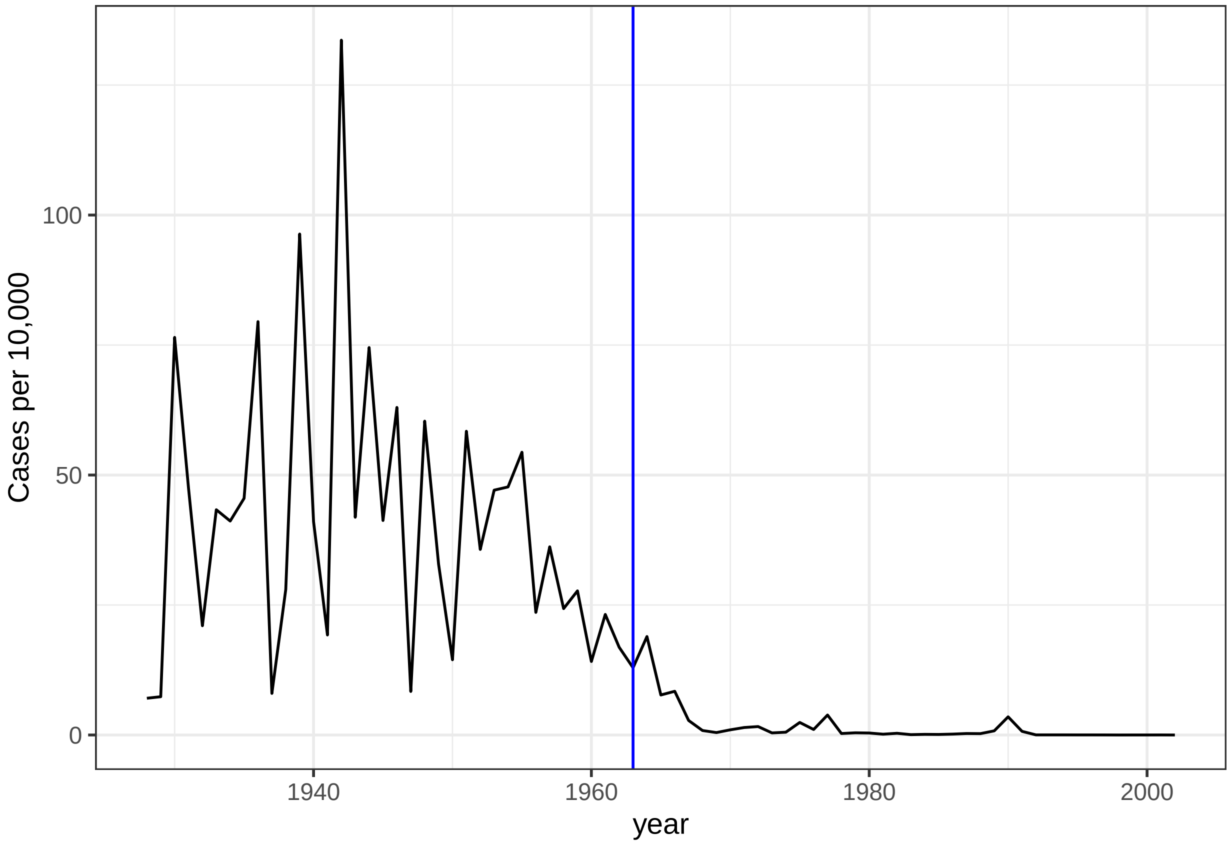
- Vaccines save millions of lives, but misinformation has led some to question the safety of vaccines. The data support vaccines as safe and effective. We visualize data about measles incidence in order to demonstrate the impact of vaccination programs on disease rate.
- The **RColorBrewer** package offers several color palettes. Sequential color palettes are best suited for data that span from high to low. Diverging color palettes are best suited for data that are centered and diverge towards high or low values.
- The `geom_tile` geometry creates a grid of colored tiles.
- Position and length are stronger cues than color for numeric values, but color can be appropriate sometimes.

Code: Tile plot of measles rate by year and state

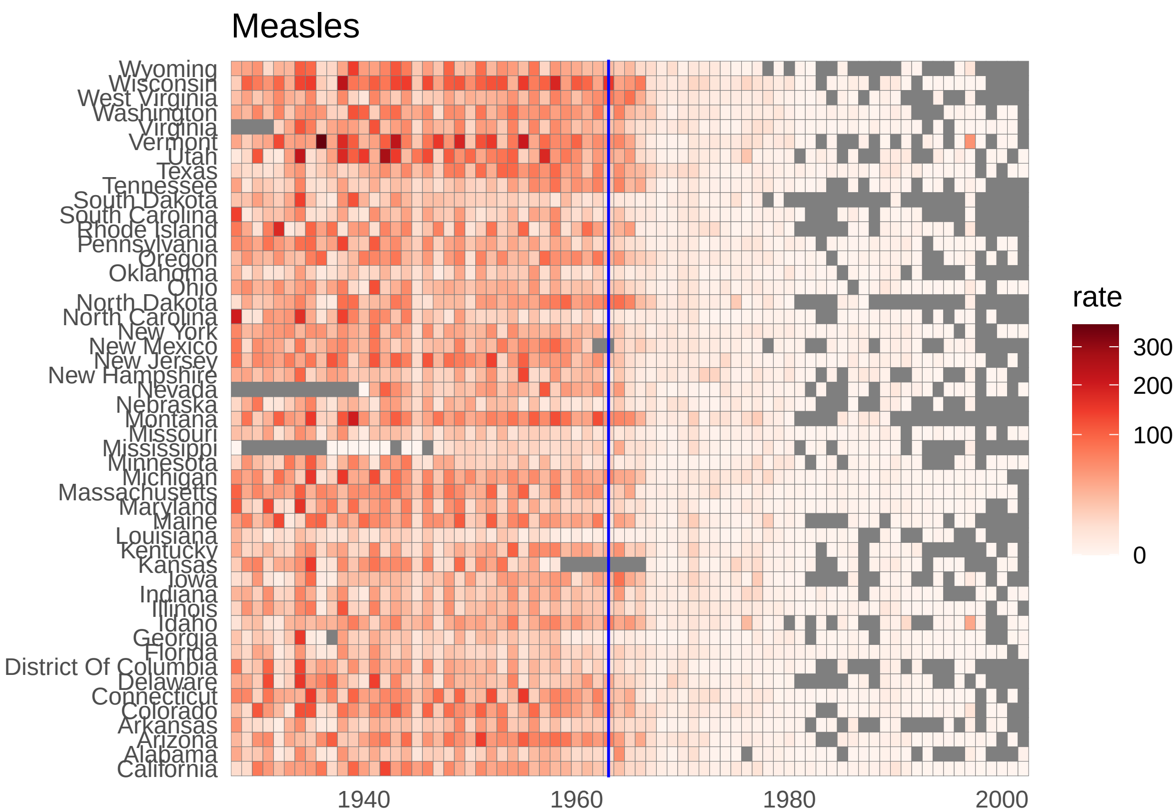
```
# import data and inspect
str(us_contagious_diseases)
```

```
## 'data.frame':    16065 obs. of  6 variables:
##  $ disease        : Factor w/ 7 levels "Hepatitis A",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ state          : Factor w/ 51 levels "Alabama","Alaska",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year           : num  1966 1967 1968 1969 1970 ...
##  $ weeks_reporting: num  50 49 52 49 51 51 45 45 45 46 ...
##  $ count          : num  321 291 314 380 413 378 342 467 244 286 ...
##  $ population     : num  3345787 3364130 3386068 3412450 3444165 ...
```

```
# assign dat to the per 10,000 rate of measles, removing Alaska and Hawaii and adjusting for weeks repo
the_disease <- "Measles"
dat <- us_contagious_diseases %>%
    filter(!state %in% c("Hawaii", "Alaska") & disease == the_disease) %>%
    mutate(rate = count / population * 10000 * 52/weeks_reporting) %>%
    mutate(state = reorder(state, rate))

# plot disease rates per year in California
dat %>% filter(state == "California" & !is.na(rate)) %>%
    ggplot(aes(year, rate)) +
    geom_line() +
    ylab("Cases per 10,000") +
    geom_vline(xintercept=1963, col = "blue")
```



```r
# tile plot of disease rate by state and year
dat %>% ggplot(aes(year, state, fill=rate)) +
    geom_tile(color = "grey50") +
    scale_x_continuous(expand = c(0,0)) +
    scale_fill_gradientn(colors = RColorBrewer::brewer.pal(9, "Reds"), trans = "sqrt") +
    geom_vline(xintercept = 1963, col = "blue") +
    theme_minimal() + theme(panel.grid = element_blank()) +
    ggtitle(the_disease) +
    ylab("") +
    xlab("")
```



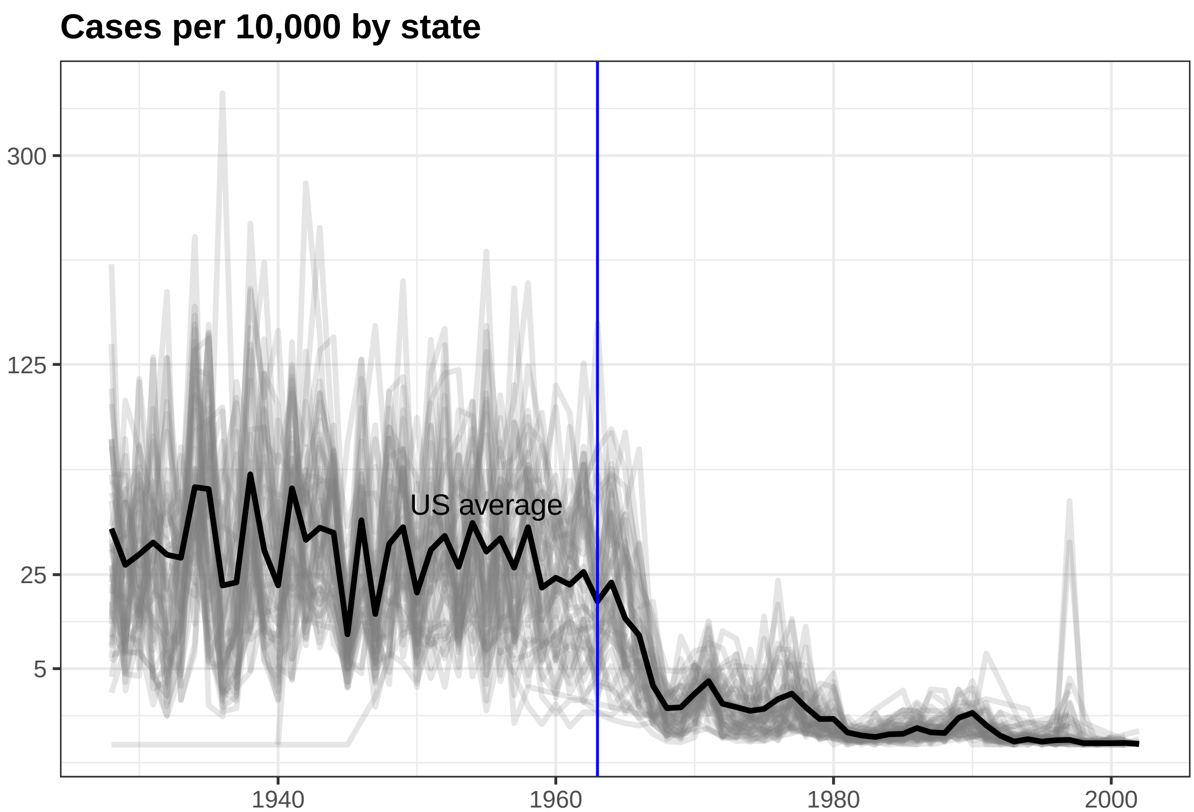
Code: Line plot of measles rate by year and state

```r
# compute US average measles rate by year
avg <- us_contagious_diseases %>%
    filter(disease == the_disease) %>% group_by(year) %>%
    summarize(us_rate = sum(count, na.rm = TRUE)/sum(population, na.rm = TRUE)*10000)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# make line plot of measles rate by year by state
dat %>%
    filter(!is.na(rate)) %>%
    ggplot() +
    geom_line(aes(year, rate, group = state), color = "grey50",
        show.legend = FALSE, alpha = 0.2, size = 1) +
    geom_line(mapping = aes(year, us_rate), data = avg, size = 1, col = "black") +
```

```r
  scale_y_continuous(trans = "sqrt", breaks = c(5, 25, 125, 300)) +
  ggtitle("Cases per 10,000 by state") +
  xlab("") +
  ylab("") +
  geom_text(data = data.frame(x = 1955, y = 50),
      mapping = aes(x, y, label = "US average"), color = "black") +
  geom_vline(xintercept = 1963, col = "blue")
```



Avoid Pseudo and Gratuitous 3D Plots

The textbook for this section is available here

Key point

In general, pseudo-3D plots and gratuitous 3D plots only add confusion. Use regular 2D plots instead.

Avoid Too Many Significant Digits

The textbook for this section is available here

Key points

- In tables, avoid using too many significant digits. Too many digits can distract from the meaning of your data.
- Reduce the number of significant digits globally by setting an option. For example, `options(digits = 3)` will cause all future computations that session to have 3 significant digits.
- Reduce the number of digits locally using `round` or `signif`.

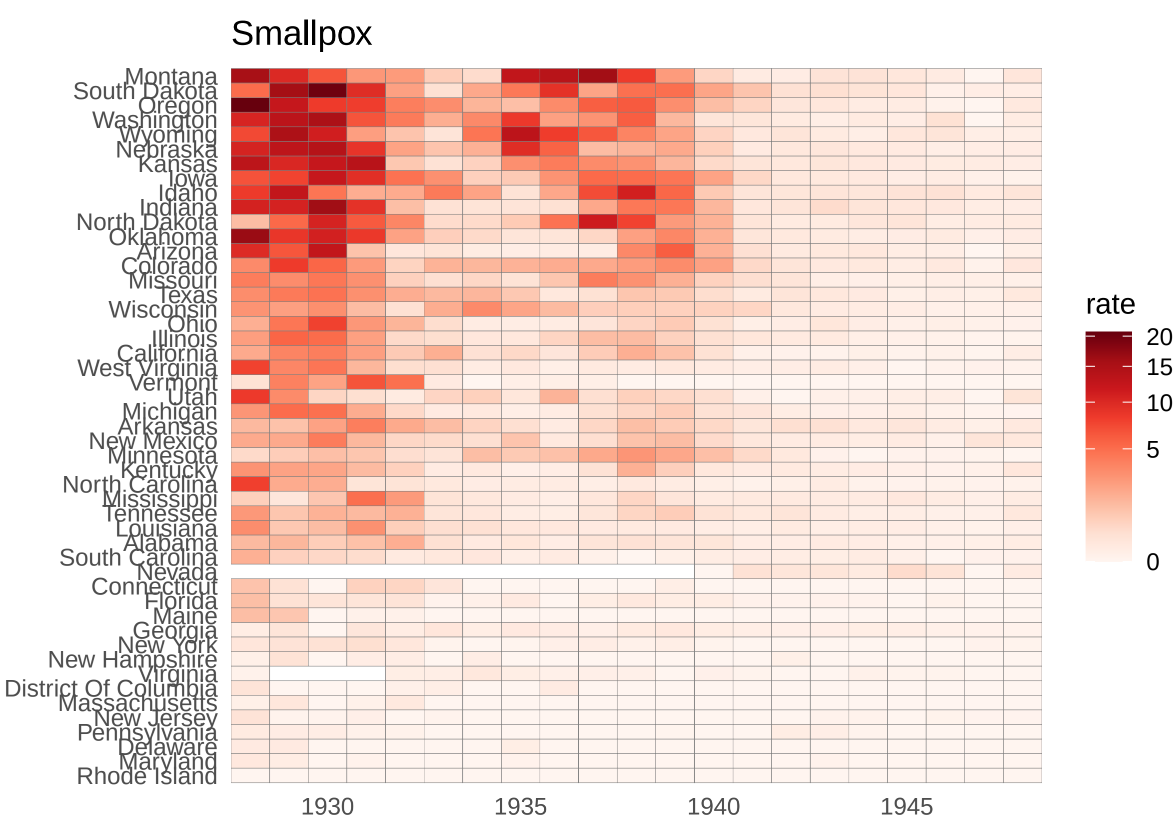
Assessment - Data Visualization Principles, Part 3

1. The sample code given creates a tile plot showing the rate of measles cases per population. We are going to modify the tile plot to look at smallpox cases instead.

```
if(!require(RColorBrewer)) install.packages("RColorBrewer")
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)

the_disease = "Smallpox"
dat <- us_contagious_diseases %>%
   filter(!state%in%c("Hawaii","Alaska") & disease == the_disease & weeks_reporting >= 10) %>%
   mutate(rate = count / population * 10000) %>%
   mutate(state = reorder(state, rate))

dat %>% ggplot(aes(year, state, fill = rate)) +
  geom_tile(color = "grey50") +
  scale_x_continuous(expand=c(0,0)) +
  scale_fill_gradientn(colors = brewer.pal(9, "Reds"), trans = "sqrt") +
  theme_minimal() +
  theme(panel.grid = element_blank()) +
  ggtitle(the_disease) +
  ylab("") +
  xlab("")
```



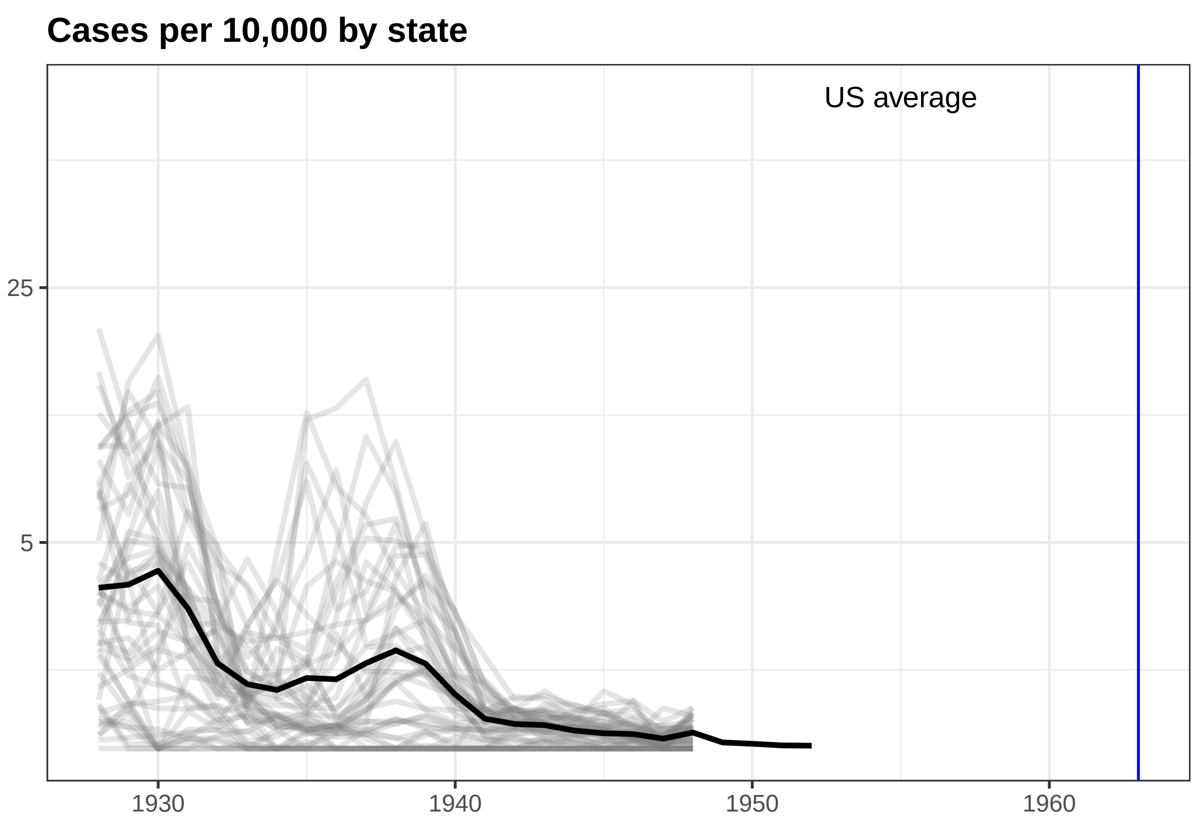
2. The sample code given creates a time series plot showing the rate of measles cases per population by state.

We are going to again modify this plot to look at smallpox cases instead.

```
the_disease = "Smallpox"
dat <- us_contagious_diseases %>%
   filter(!state%in%c("Hawaii","Alaska") & disease == the_disease & weeks_reporting >= 10) %>%
   mutate(rate = count / population * 10000) %>%
   mutate(state = reorder(state, rate))

avg <- us_contagious_diseases %>%
  filter(disease==the_disease) %>% group_by(year) %>%
  summarize(us_rate = sum(count, na.rm=TRUE)/sum(population, na.rm=TRUE)*10000)
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
dat %>% ggplot() +
  geom_line(aes(year, rate, group = state),  color = "grey50",
            show.legend = FALSE, alpha = 0.2, size = 1) +
  geom_line(mapping = aes(year, us_rate),  data = avg, size = 1, color = "black") +
  scale_y_continuous(trans = "sqrt", breaks = c(5,25,125,300)) +
  ggtitle("Cases per 10,000 by state") +
  xlab("") +
  ylab("") +
  geom_text(data = data.frame(x=1955, y=50), mapping = aes(x, y, label="US average"), color="black") +
  geom_vline(xintercept=1963, col = "blue")
```

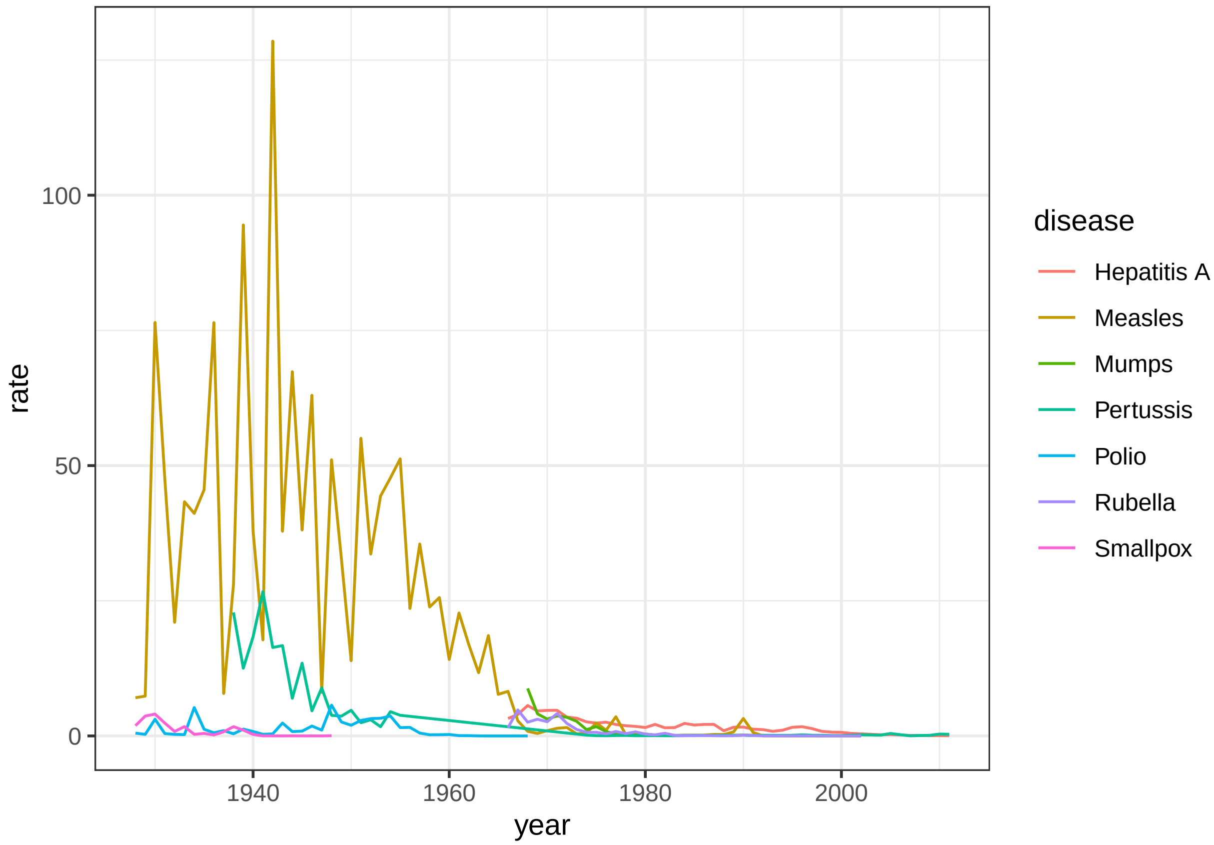


3. Now we are going to look at the rates of all diseases in one state.

Again, you will be modifying the sample code to produce the desired plot.

```
us_contagious_diseases %>% filter(state=="California" & weeks_reporting >= 10) %>%
  group_by(year, disease) %>%
  summarize(rate = sum(count)/sum(population)*10000) %>%
  ggplot(aes(year, rate, color = disease)) +
  geom_line()
```

```
## `summarise()` regrouping output by 'year' (override with `.groups` argument)
```

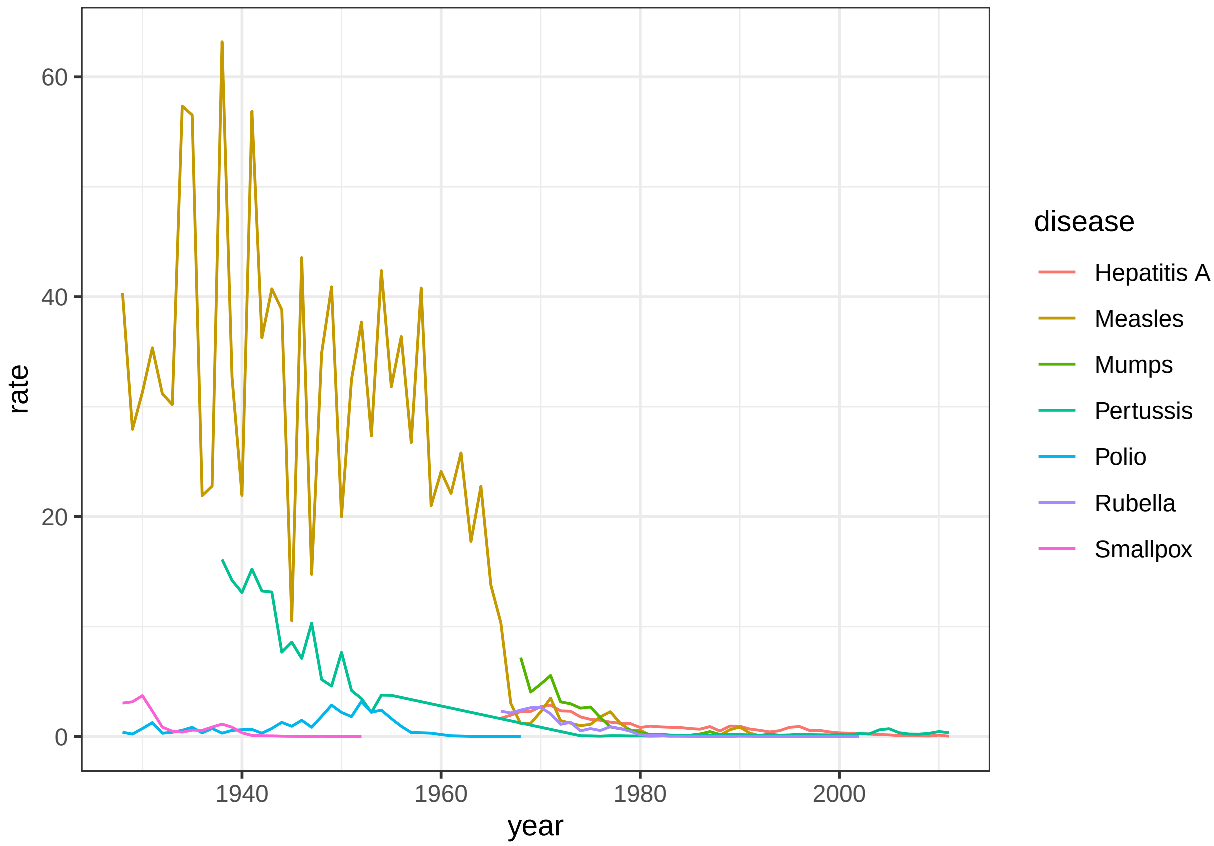


4. Now we are going to make a time series plot for the rates of all diseases in the United States.

For this exercise, we have provided less sample code - you can take a look at the previous exercise to get you started.

```
us_contagious_diseases %>% filter(!is.na(population)) %>%
  group_by(year, disease) %>%
  summarize(rate = sum(count)/sum(population)*10000) %>%
  ggplot(aes(year, rate, color = disease)) +
  geom_line()
```

```
## `summarise()` regrouping output by 'year' (override with `.groups` argument)
```



Titanic Survival Exercises

Put all your new skills together to perform exploratory data analysis on a classic machine learning dataset: Titanic survival!

Background

The Titanic was a British ocean liner that struck an iceberg and sunk on its maiden voyage in 1912 from the United Kingdom to New York. More than 1,500 of the estimated 2,224 passengers and crew died in the accident, making this one of the largest maritime disasters ever outside of war. The ship carried a wide range of passengers of all ages and both genders, from luxury travelers in first-class to immigrants in the lower classes. However, not all passengers were equally likely to survive the accident. We use real data about a selection of 891 passengers to learn who was on the Titanic and which passengers were more likely to survive.

Libraries, Options, and Data

Define the titanic dataset starting from the **titanic** library with the following code:

```
if(!require(titanic)) install.packages("titanic")
```

```
## Loading required package: titanic
```

```
## Warning: package 'titanic' was built under R version 4.0.2
```

```
options(digits = 3)    # report 3 significant digits
library(tidyverse)
library(titanic)
```

```r
titanic <- titanic_train %>%
    select(Survived, Pclass, Sex, Age, SibSp, Parch, Fare) %>%
    mutate(Survived = factor(Survived),
           Pclass = factor(Pclass),
           Sex = factor(Sex))
```

1. Variable Types

Inspect the data and also use `?titanic_train` to learn more about the variables in the dataset. Match these variables from the dataset to their variable type. There is at least one variable of each type (ordinal categorical, non-ordinal categorical, continuous, discrete).

- `Survived` non-ordinal categorical
- `Pclass` ordinal categorical
- `Sex` non-ordinal categorical
- `SibSp` discrete
- `Parch` discrete

- `Fare` continuous

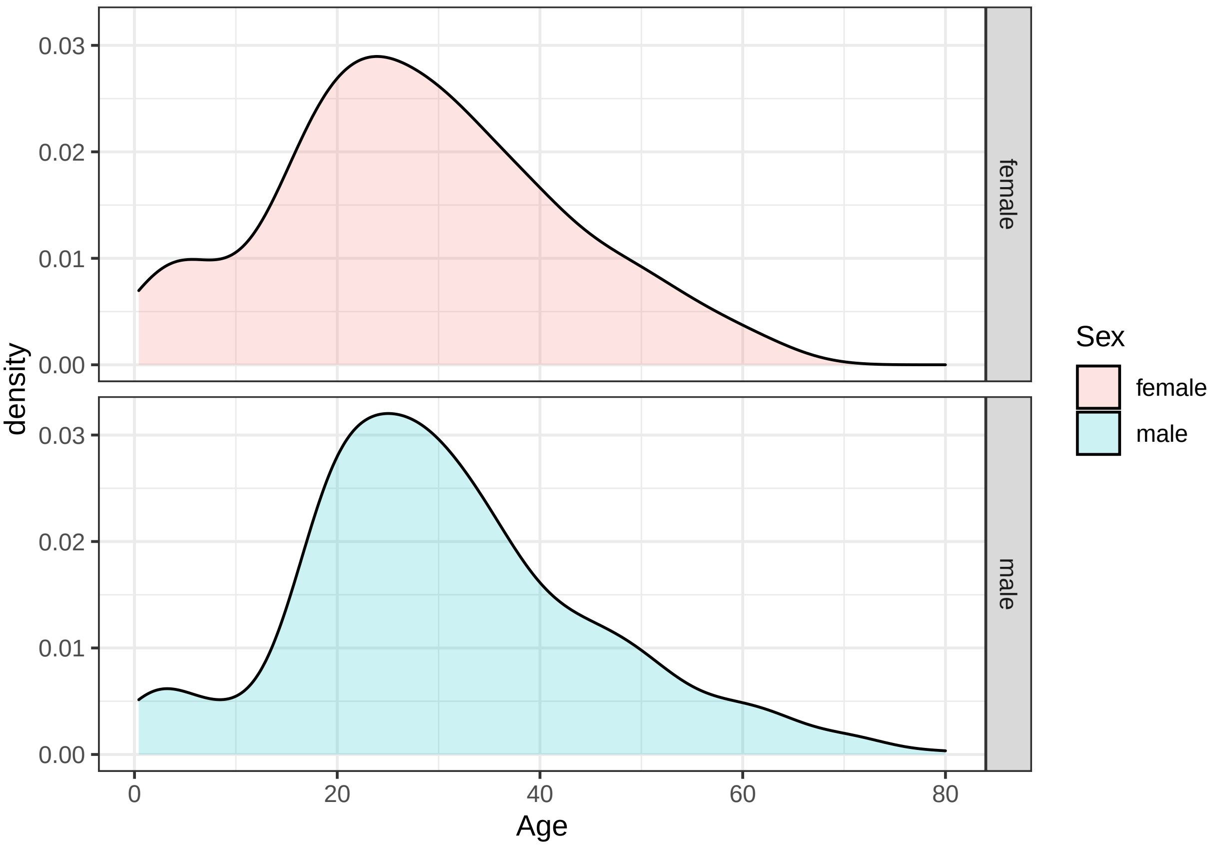
2. Demographics of Titanic Passengers

Make density plots of age grouped by sex. Try experimenting with combinations of faceting, alpha blending, stacking and using variable counts on the y-axis to answer the following questions. Some questions may be easier to answer with different versions of the density plot.

A faceted plot is useful for comparing the distributions of males and females for A. Each sex has the same general shape with two modes at the same locations, though proportions differ slightly across ages and there are more males than females.

```r
titanic %>%
    ggplot(aes(Age, fill = Sex)) +
    geom_density(alpha = 0.2) +
    facet_grid(Sex ~ .)
```

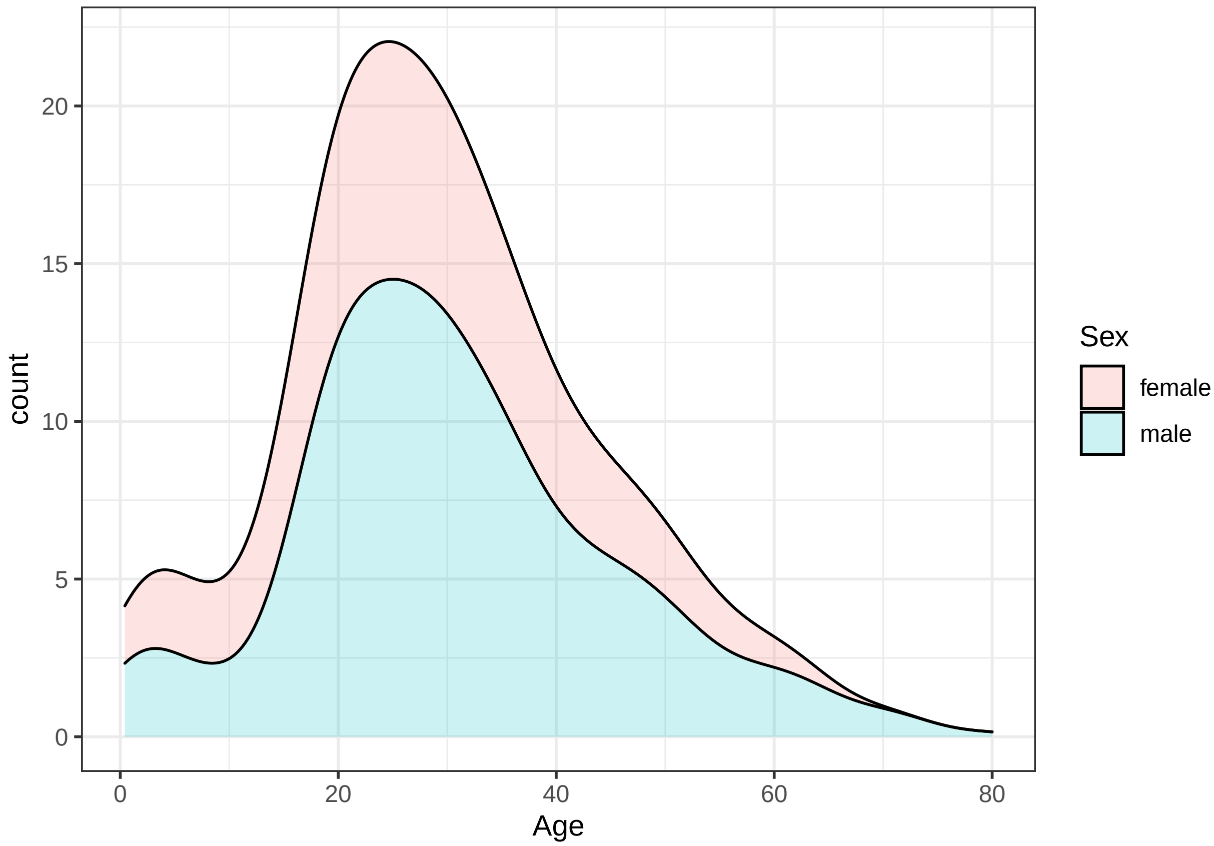
```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```



A stacked density plot with count on the y-axis is useful for answering B, C and D. The main mode is around age 25 and a second smaller mode is around age 4-5. There are more males than females as indicated by a higher total area and higher counts at almost all ages. With count on the y-axis, it is clear that more males than females are age 40.

```
titanic %>%
    ggplot(aes(Age, y = ..count.., fill = Sex)) +
    geom_density(alpha = 0.2, position = "stack")
```

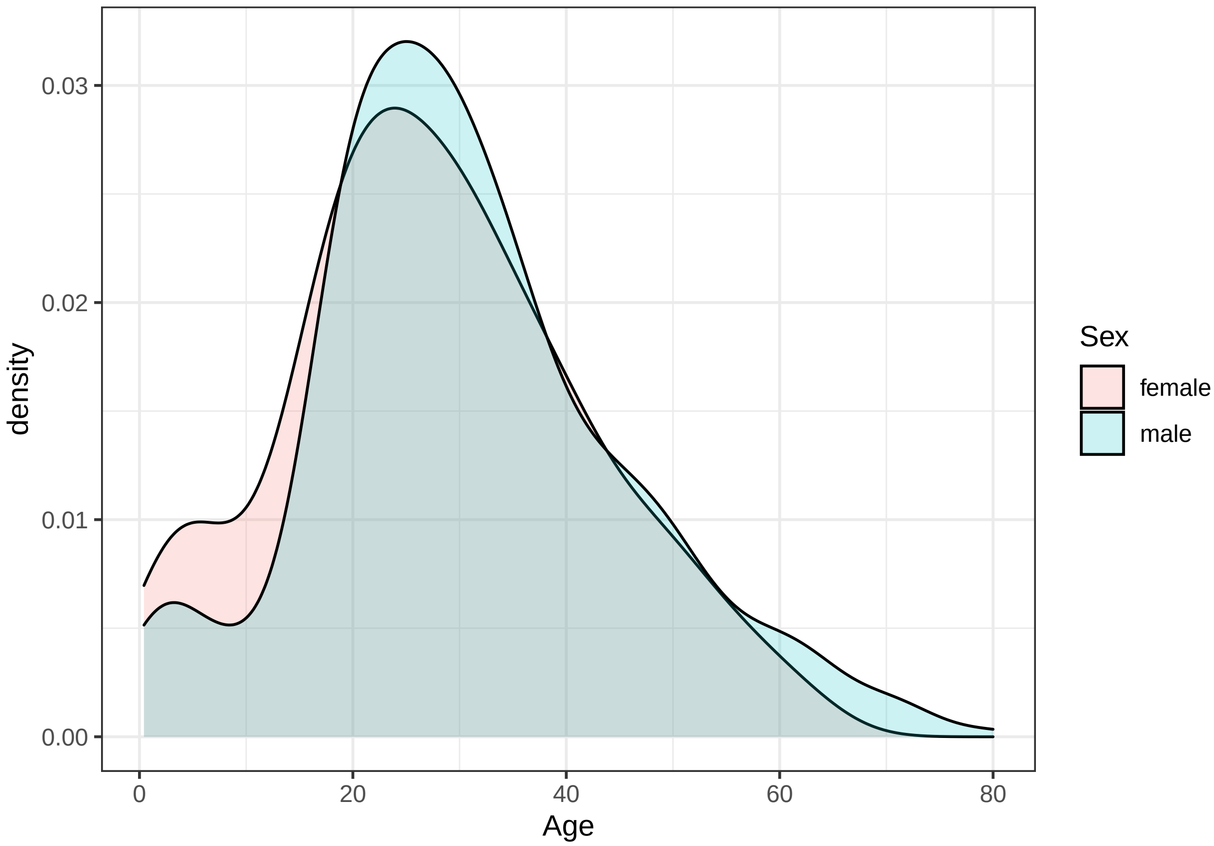
```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```



A plot filled by sex with alpha blending helps reveal the answers to E, F and G. There is a higher proportion of females than males below age 17, a higher proportion of males than females for ages 18-35, approximately the same proportion of males and females age 35-55, and a higher proportion of males over age 55. The oldest individuals are male.

```
titanic %>%
    ggplot(aes(Age, fill = Sex)) +
    geom_density(alpha = 0.2)
```

```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```



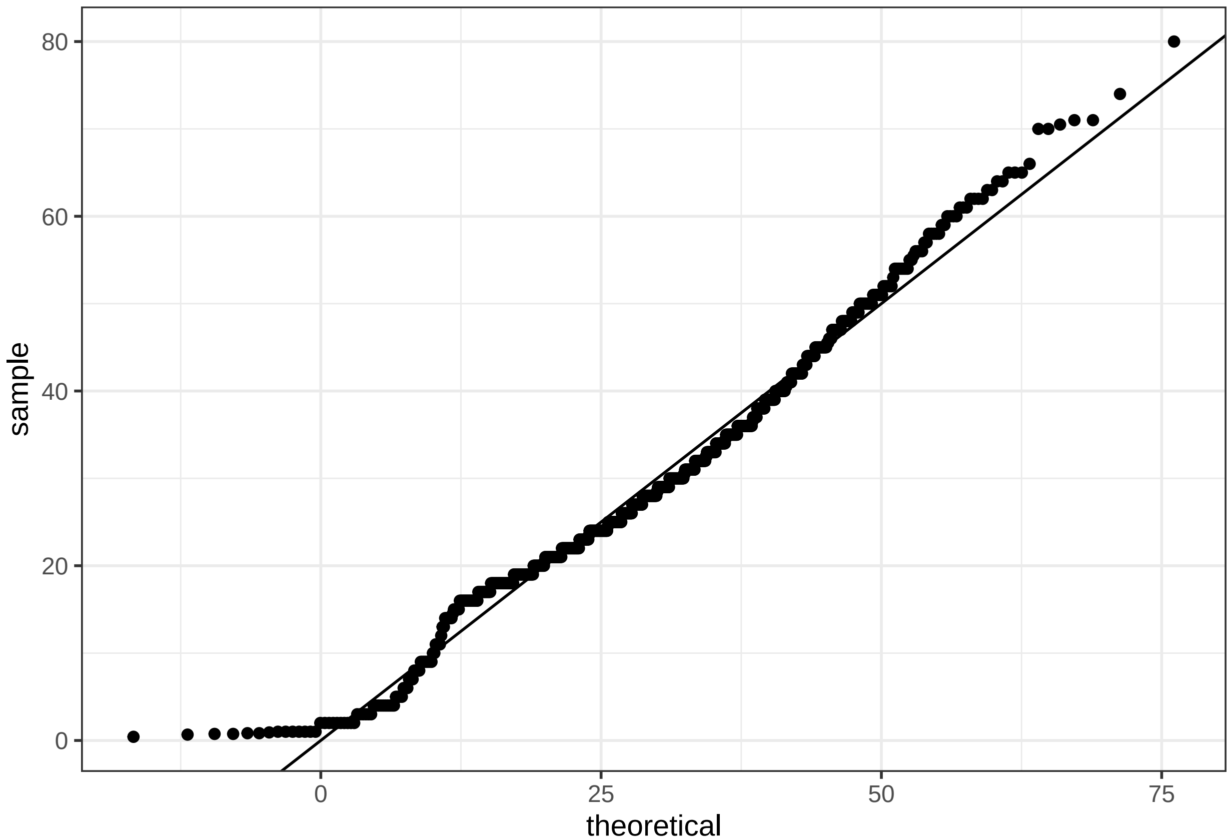
Which of the following are true? Select all correct answers.

- ☒ A. Females and males had the same general shape of age distribution.
- ☒ B. The age distribution was bimodal, with one mode around 25 years of age and a second smaller mode around 5 years of age.
- ☐ C. There were more females than males.
- ☒ D. The count of males of age 40 was higher than the count of females of age 40.
- ☒ E. The proportion of males age 18-35 was higher than the proportion of females age 18-35.
- ☒ F. The proportion of females under age 17 was higher than the proportion of males under age 17.
- ☐ G. The oldest passengers were female.

3. QQ-plot of Age Distribution

Use `geom_qq` to make a QQ-plot of passenger age and add an identity line with `geom_abline`. Filter out any individuals with an age of NA first. Use the following object as the `dparams` argument in `geom_qq`:

```
params <- titanic %>%
    filter(!is.na(Age)) %>%
    summarize(mean = mean(Age), sd = sd(Age))

titanic %>%
    filter(!is.na(Age)) %>%
    ggplot(aes(sample = Age)) +
    geom_qq(dparams = params) +
    geom_abline()
```



What is the correct plot according to the instructions above? `QQ-plot C`

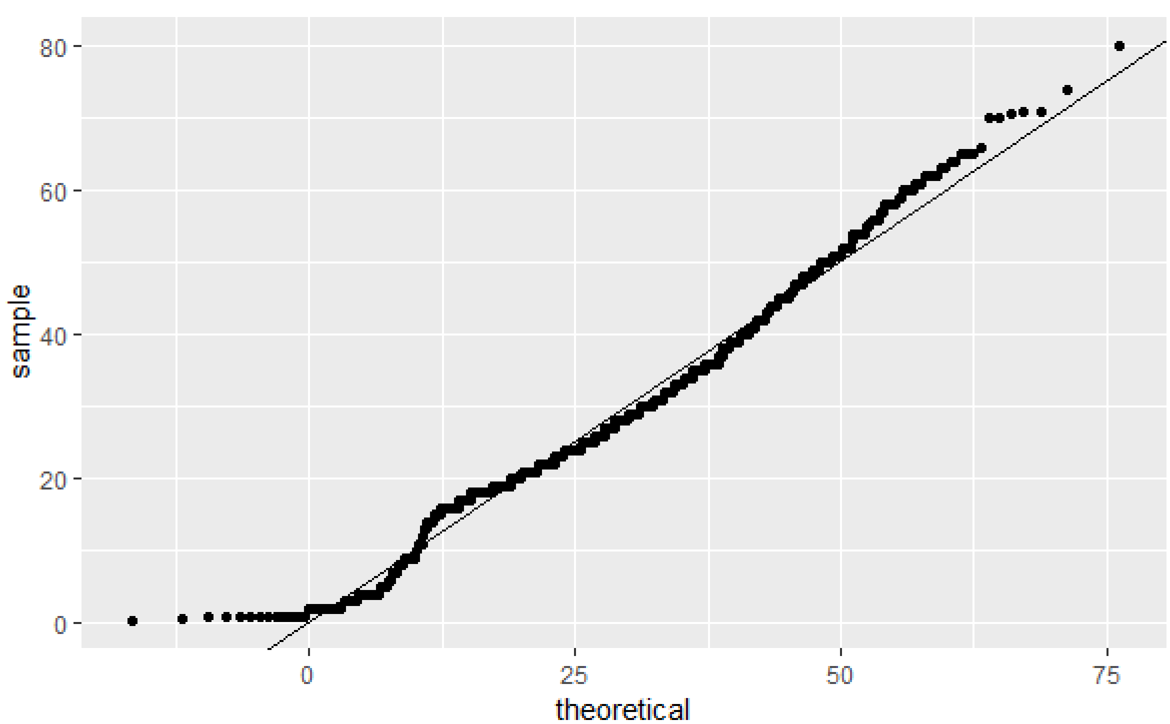


Figure 11: QQ-plot C

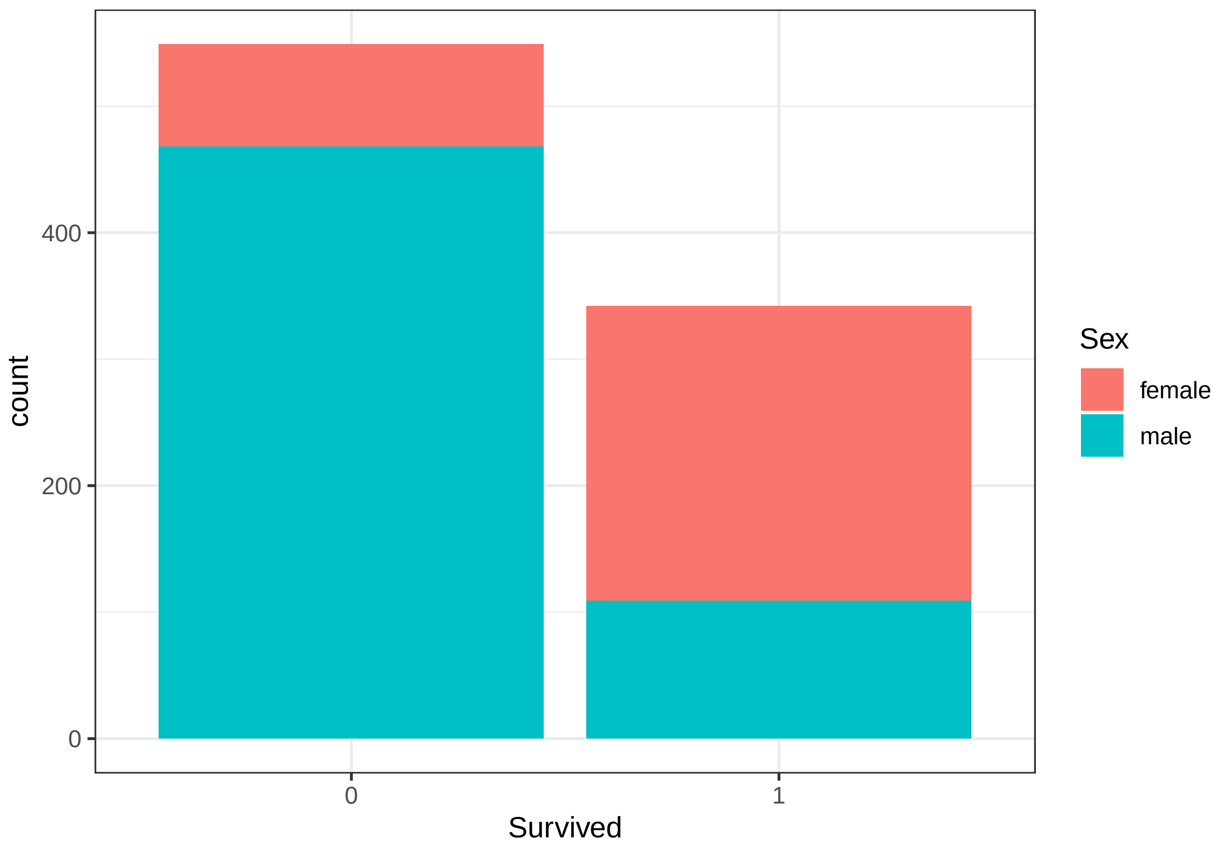
4. Survival by Sex

To answer the following questions, make barplots of the `Survived` and `Sex` variables using `geom_bar`. Try plotting one variable and filling by the other variable. You may want to try the default plot, then try adding `position = position_dodge()` to `geom_bar` to make separate bars for each group.

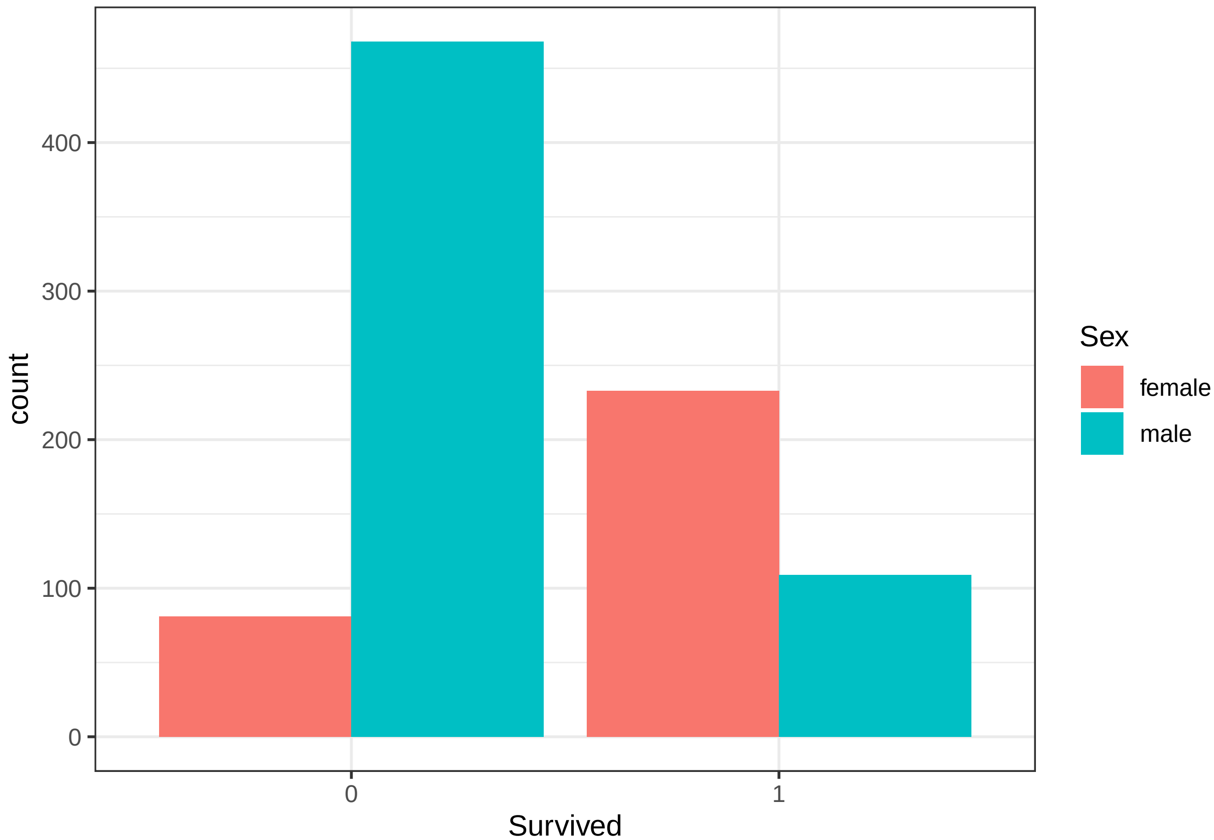
You can read more about making barplots in the textbook section on ggplot2 geometries.

A and B can be clearly seen in the barplot of survival status filled by sex. The count of survivors is lower than the count of non-survivors. The bar of survivors is more than half filled by females. Alternatively, the bars can be split by sex with `position_dodge`, showing the "Female, Survived" bar has a greater height than the "Male, survived" bar. C and D are more clearly seen in the barplot of sex filled by survival status, though they can also be determined from the first barplot. Most males did not survive, but most females did survive.

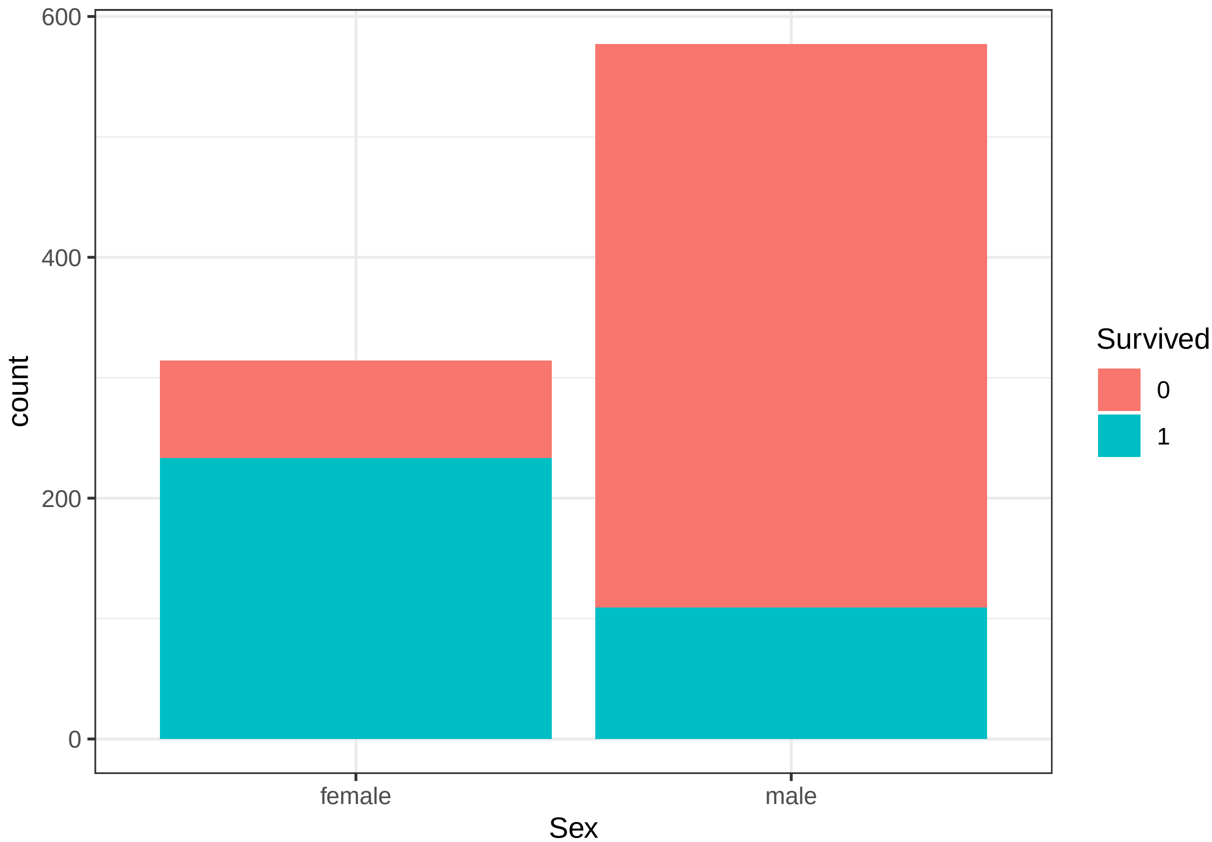
```
#plot 1 - survival filled by sex
titanic %>%
    ggplot(aes(Survived, fill = Sex)) +
    geom_bar()
```




```
# plot 2 - survival filled by sex with position_dodge
titanic %>%
    ggplot(aes(Survived, fill = Sex)) +
    geom_bar(position = position_dodge())
```




```r
#plot 3 - sex filled by survival
titanic %>%
    ggplot(aes(Sex, fill = Survived)) +
    geom_bar()
```



Which of the following are true? Select all correct answers.

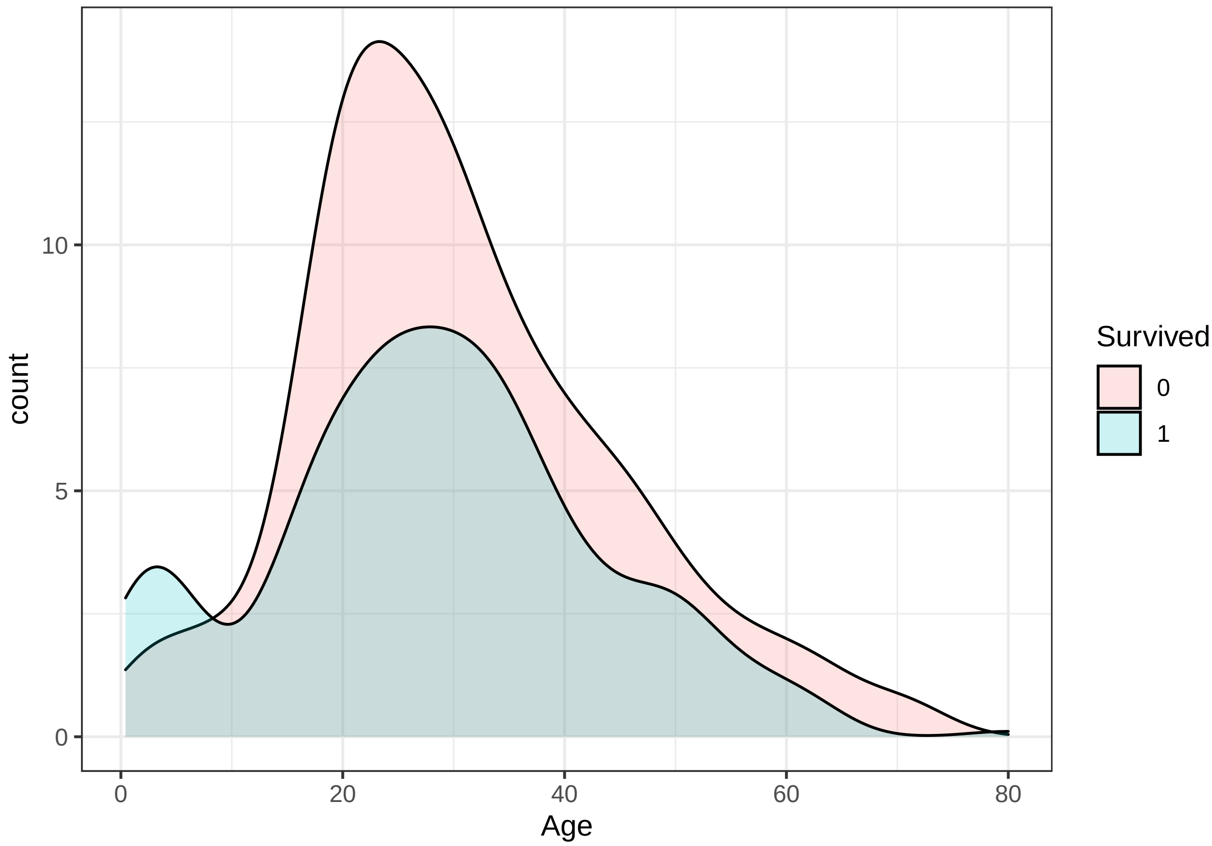
- ☒ A. Less than half of passengers survived.
- ☒ B. Most of the survivors were female.
- ☐ C. Most of the males survived.
- ☒ D. Most of the females survived.

5. Survival by Age

Make a density plot of age filled by survival status. Change the y-axis to count and set `alpha = 0.2`.

```
titanic %>%
    ggplot(aes(Age, y = ..count.., fill = Survived)) +
    geom_density(alpha = 0.2)
```

```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```



Which age group is the only group more likely to survive than die?

- ☒ A. 0-8
- ☐ B. 10-18
- ☐ C. 18-30
- ☐ D. 30-50
- ☐ E. 50-70
- ☐ F. 70-80

Which age group had the most deaths?

- ☐ A. 0-8
- ☐ B. 10-18
- ☒ C. 18-30
- ☐ D. 30-50
- ☐ E. 50-70
- ☐ F. 70-80

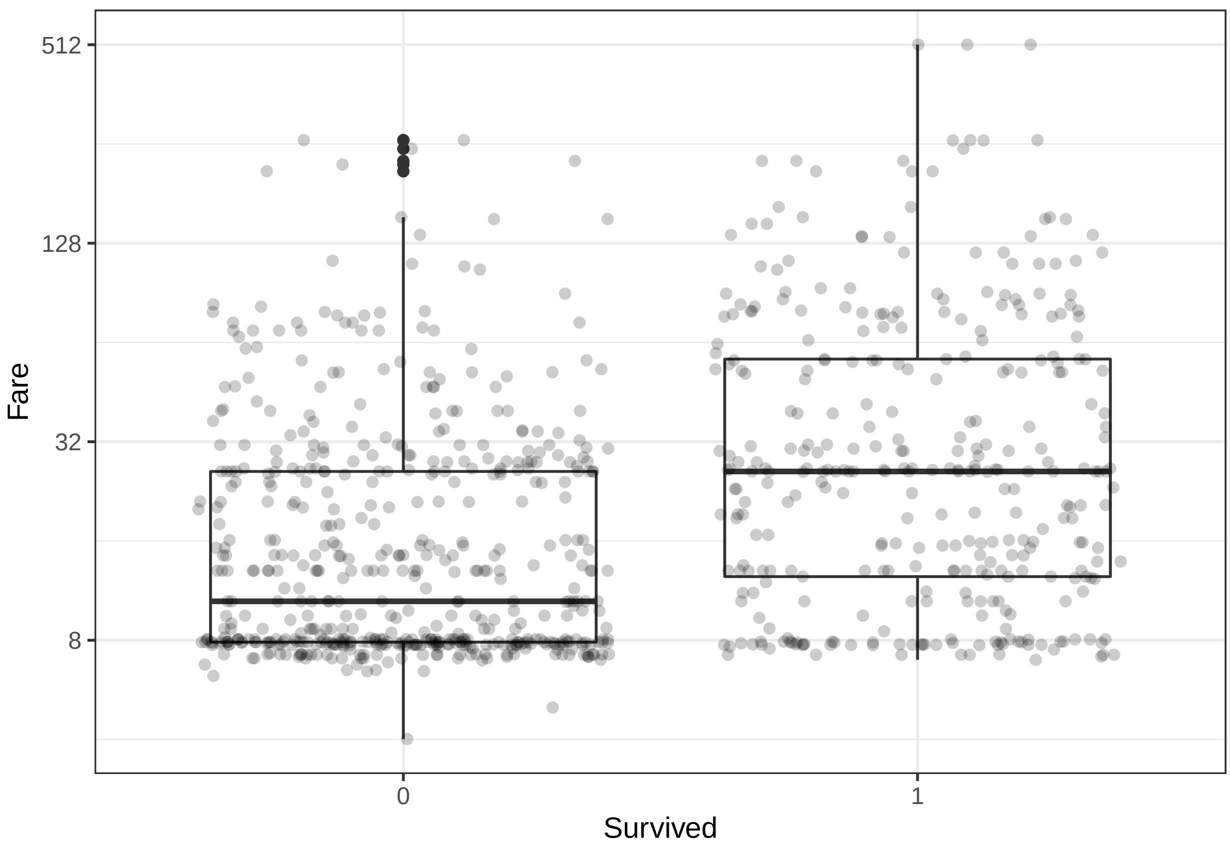
Which age group had the highest proportion of deaths?

- ☐ A. 0-8
- ☐ B. 10-18
- ☐ C. 18-30
- ☐ D. 30-50
- ☐ E. 50-70
- ☒ F. 70-80

6. Survival by Fare

Filter the data to remove individuals who paid a fare of 0. Make a boxplot of fare grouped by survival status. Try a log2 transformation of fares. Add the data points with jitter and alpha blending.

```
titanic %>%
    filter(Fare > 0) %>%
    ggplot(aes(Survived, Fare)) +
    geom_boxplot() +
    scale_y_continuous(trans = "log2") +
    geom_jitter(alpha = 0.2)
```



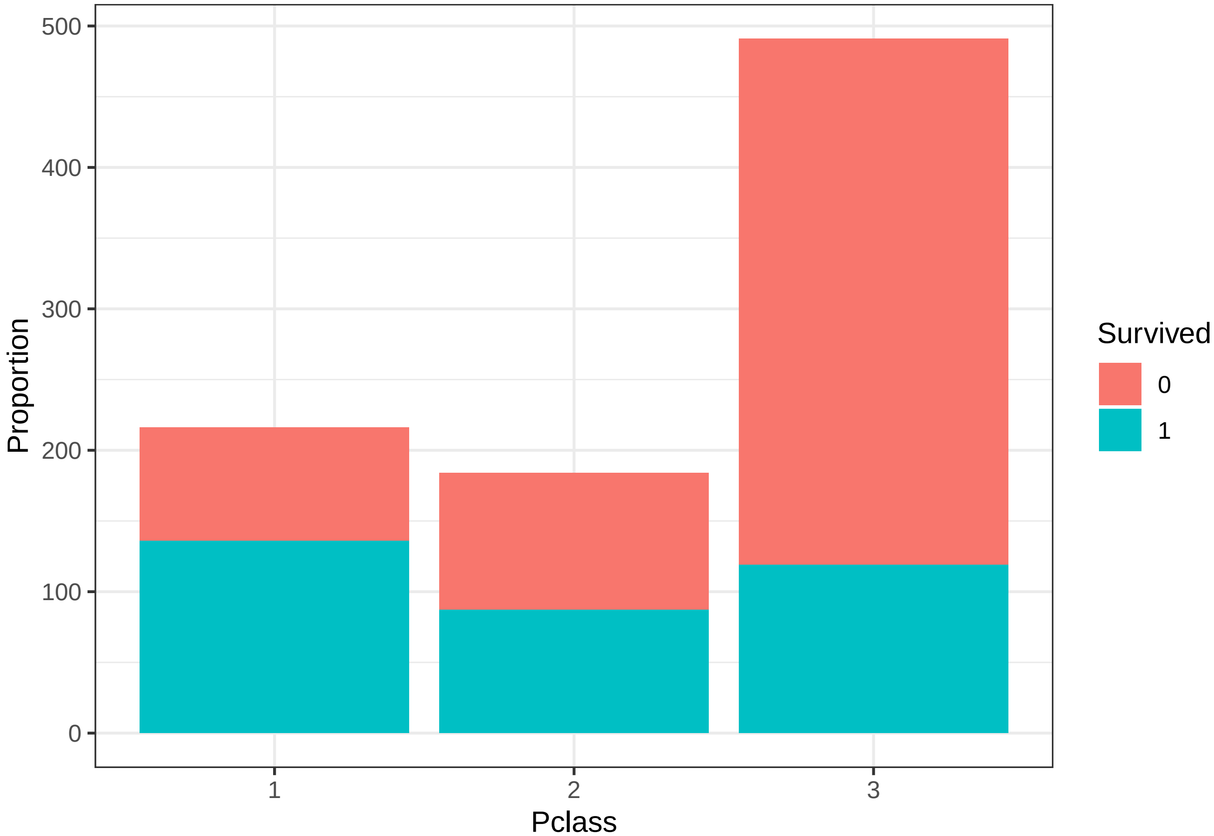
Which of the following are true? Select all correct answers.

- ☒ A. Passengers who survived generally payed higher fares than those who did not survive.
- ☐ B. The interquartile range for fares was smaller for passengers who survived.
- ☒ C. The median fare was lower for passengers who did not survive.
- ☐ D. Only one individual paid a fare around $500. That individual survived.
- ☒ E. Most individuals who paid a fare around $8 did not survive.

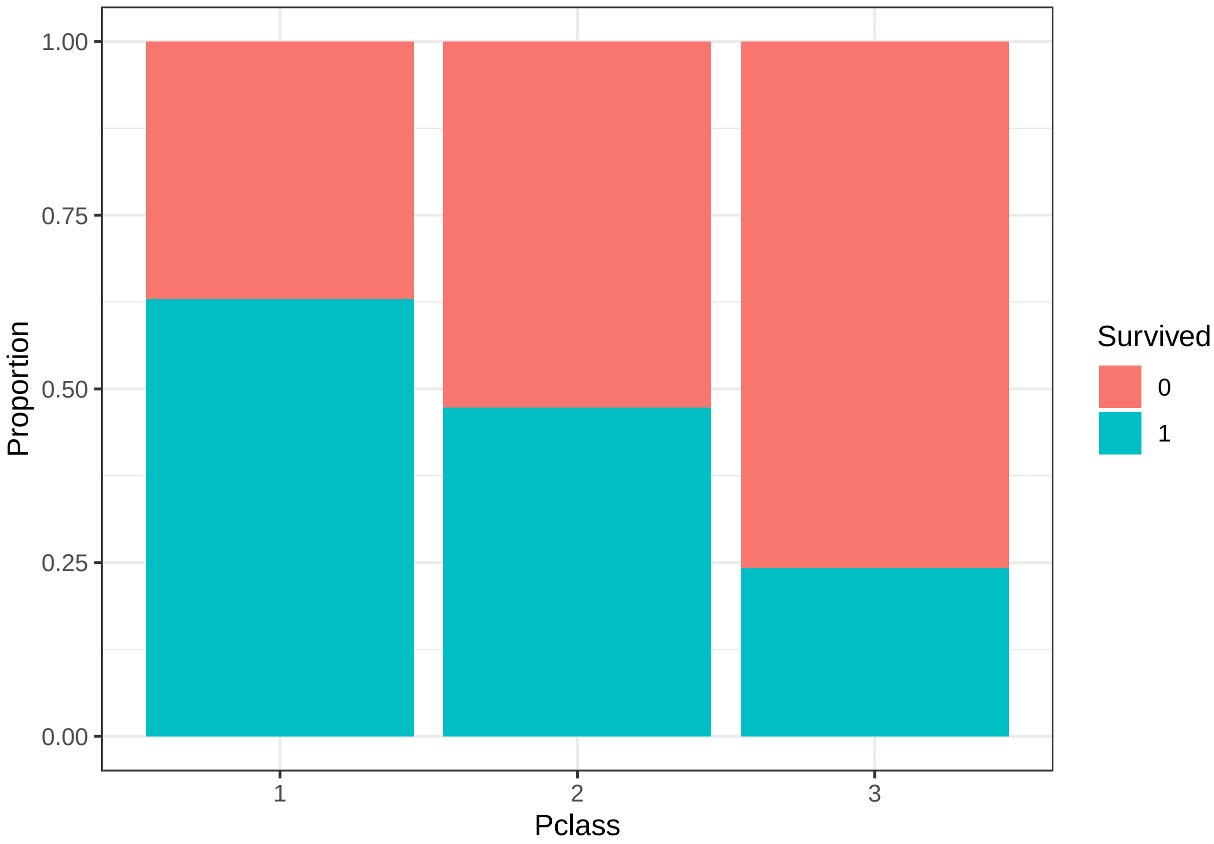
7. Survival by Passenger Class

The `Pclass` variable corresponds to the passenger class. Make three barplots. For the first, make a basic barplot of passenger class filled by survival. For the second, make the same barplot but use the argument `position = position_fill()` to show relative proportions in each group instead of counts. For the third, make a barplot of survival filled by passenger class using `position = position_fill()`.

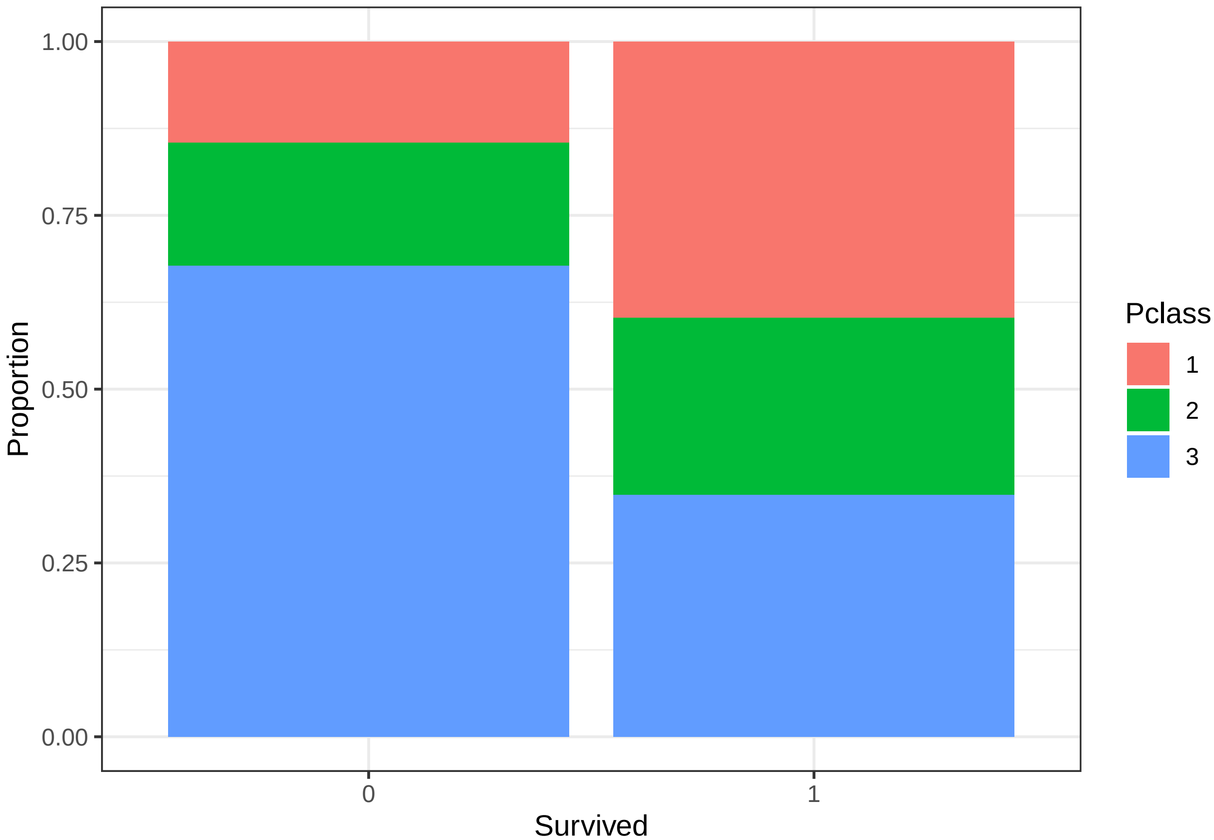
```r
# barplot of passenger class filled by survival
titanic %>%
    ggplot(aes(Pclass, fill = Survived)) +
    geom_bar() +
    ylab("Proportion")
```




```r
# barplot of passenger class filled by survival with position_fill
titanic %>%
    ggplot(aes(Pclass, fill = Survived)) +
    geom_bar(position = position_fill()) +
    ylab("Proportion")
```




```r
# barplot of survival filled by passenger class with position_fill
titanic %>%
    ggplot(aes(Survived, fill = Pclass)) +
    geom_bar(position = position_fill()) +
    ylab("Proportion")
```



Which of the following are true? Select all correct answers.

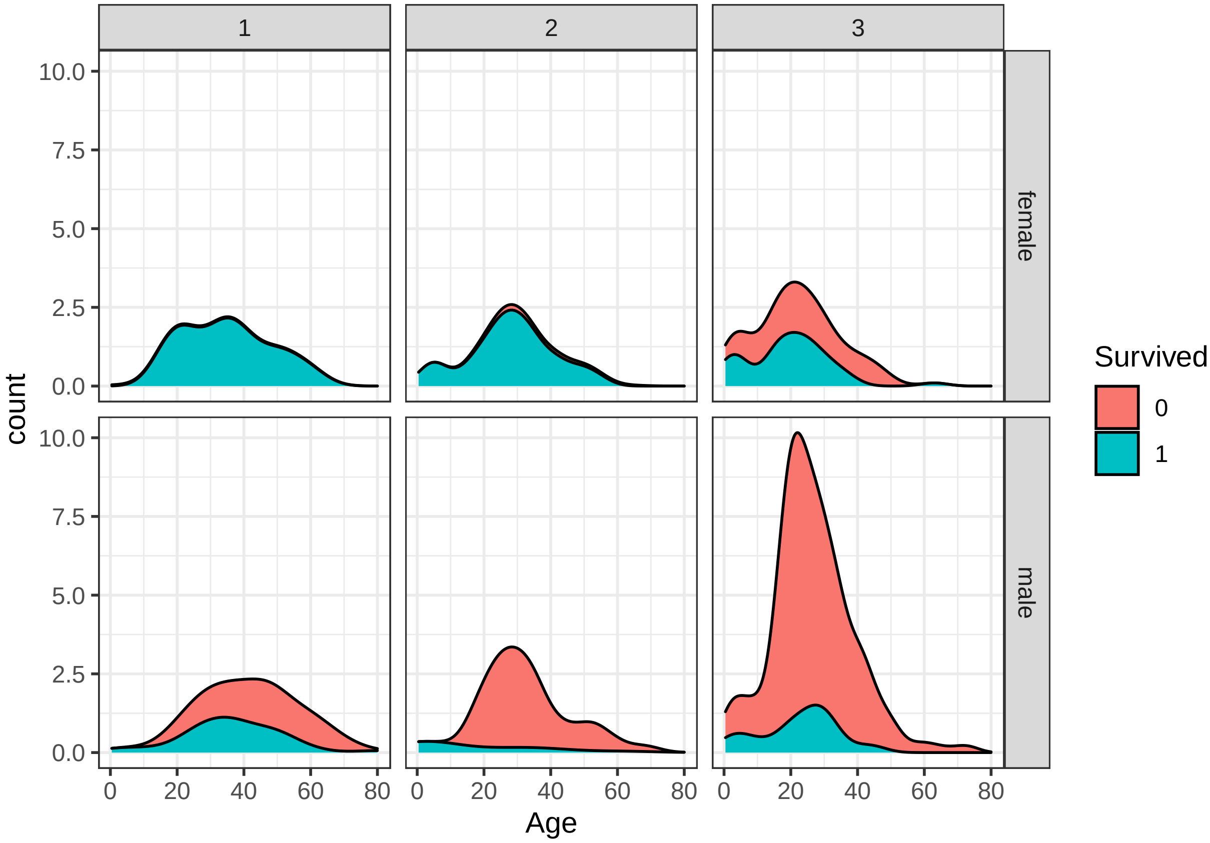
- ☒ A. There were more third class passengers than passengers in the first two classes combined.
- ☐ B. There were the fewest passengers in first class, second-most passengers in second class, and most passengers in third class.
- ☒ C. Survival proportion was highest for first class passengers, followed by second class. Third-class had the lowest survival proportion.
- ☒ D. Most passengers in first class survived. Most passengers in other classes did not survive.
- ☐ E. The majority of survivors were from first class. (Majority means over 50%.)
- ☒ F. The majority of those who did not survive were from third class.

8. Survival by Age, Sex and Passenger Class

Create a grid of density plots for age, filled by survival status, with count on the y-axis, faceted by sex and passenger class.

```
titanic %>%
    ggplot(aes(Age, y = ..count.., fill = Survived)) +
    geom_density(position = "stack") +
    facet_grid(Sex ~ Pclass)
```

```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```



Which of the following are true? Select all correct answers.

☒ A. The largest group of passengers was third-class males.
☐ B. The age distribution is the same across passenger classes.
☐ C. The gender distribution is the same across passenger classes.
☒ D. Most first-class and second-class females survived.
☒ E. Almost all second-class males did not survive, with the exception of children.

Properties of Stars Exercises

Background

Astronomy is one of the oldest data-driven sciences. In the late 1800s, the director of the Harvard College Observatory hired women to analyze astronomical data, which at the time was done using photographic glass plates. These women became known as the Harvard Computers. They computed the position and luminosity of various astronomical objects such as stars and galaxies. (If you are interested, you can learn more about the Harvard Computers). Today, astronomy is even more of a data-driven science, with an inordinate amount of data being produced by modern instruments every day.

In the following exercises we will analyze some actual astronomical data to inspect properties of stars, their absolute magnitude (which relates to a star's **luminosity**, or brightness), temperature and type (spectral class).

Libraries and Options

```
data(stars)
options(digits = 3)    # report 3 significant digits
```

IMPORTANT: These exercises use **dslabs** datasets that were added in a July 2019 update. Make sure your package is up to date with the command `update.packages("dslabs")`. You can also update all packages on your system by running `update.packages()` with no arguments, and you should consider doing this routinely.

1. Load the `stars` data frame from **dslabs**. This contains the name, absolute magnitude, temperature in degrees Kelvin, and spectral class of selected stars. Absolute magnitude (shortened in these problems to simply "magnitude") is a function of star luminosity, where **negative** values of magnitude have higher luminosity.

What is the mean magnitude?

```
mean(stars$magnitude)
```

```
## [1] 4.26
```

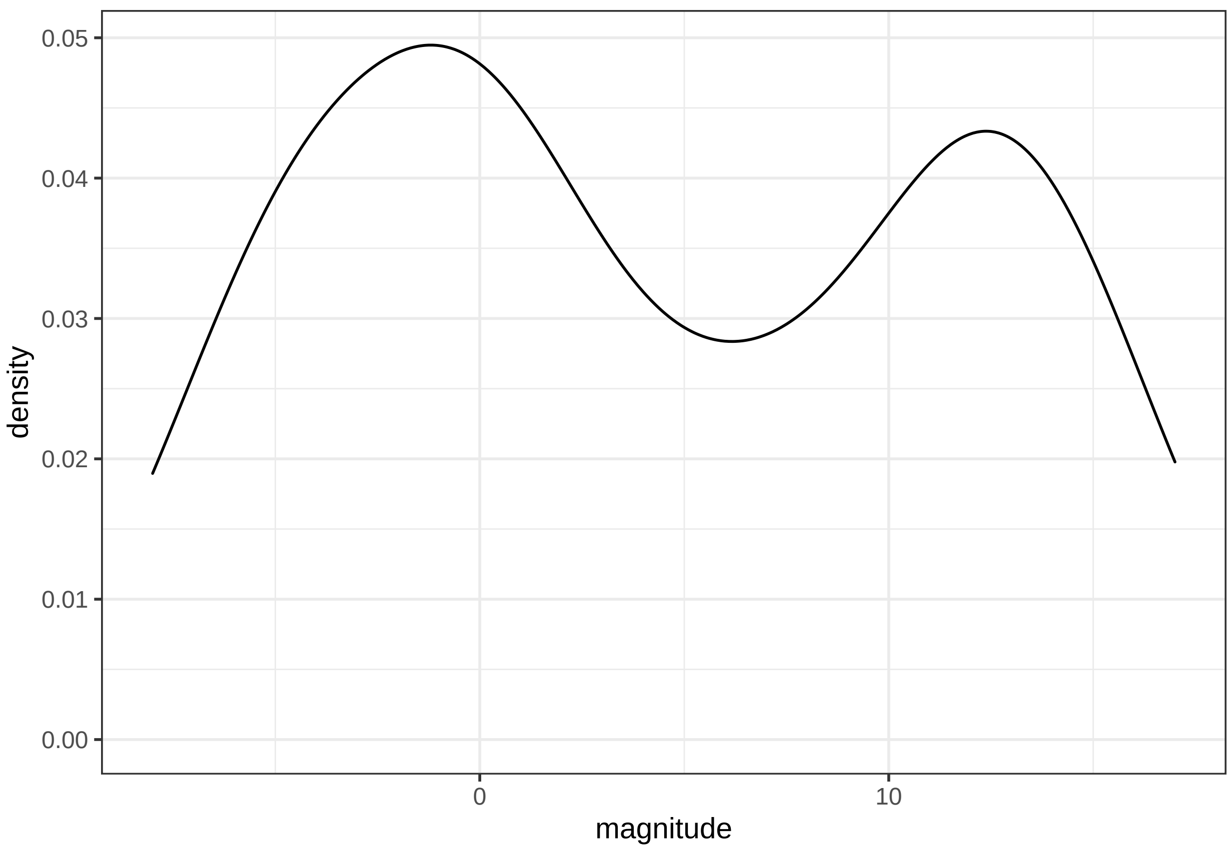
What is the standard deviation of magnitude?

```
sd(stars$magnitude)
```

```
## [1] 7.35
```

2. Make a density plot of the magnitude.

```
stars %>%
  ggplot(aes(magnitude)) +
  geom_density()
```

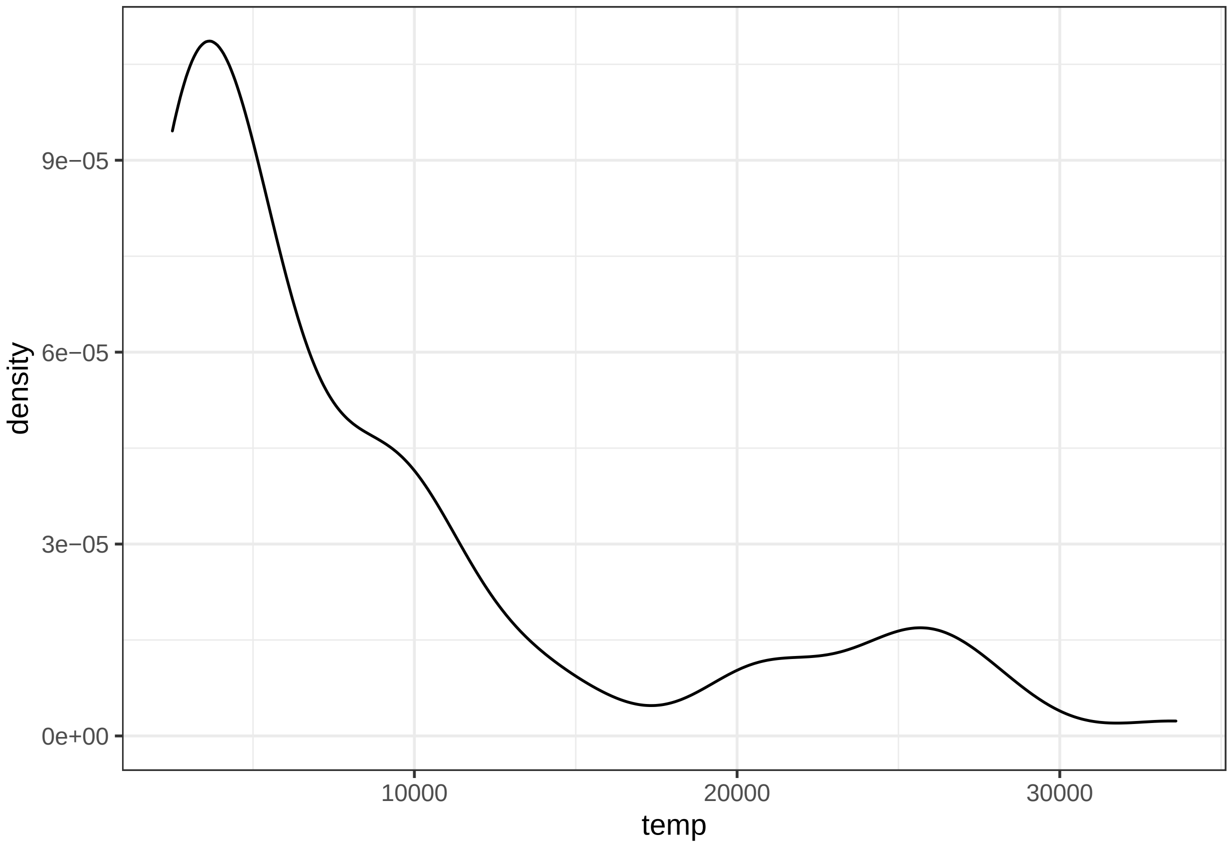


How many peaks are there in the data?

- ☐ A. 1
- ☒ B. 2
- ☐ C. 3
- ☐ D. 4

3. Examine the distribution of star temperature.

```
stars %>%
  ggplot(aes(temp)) +
  geom_density()
```



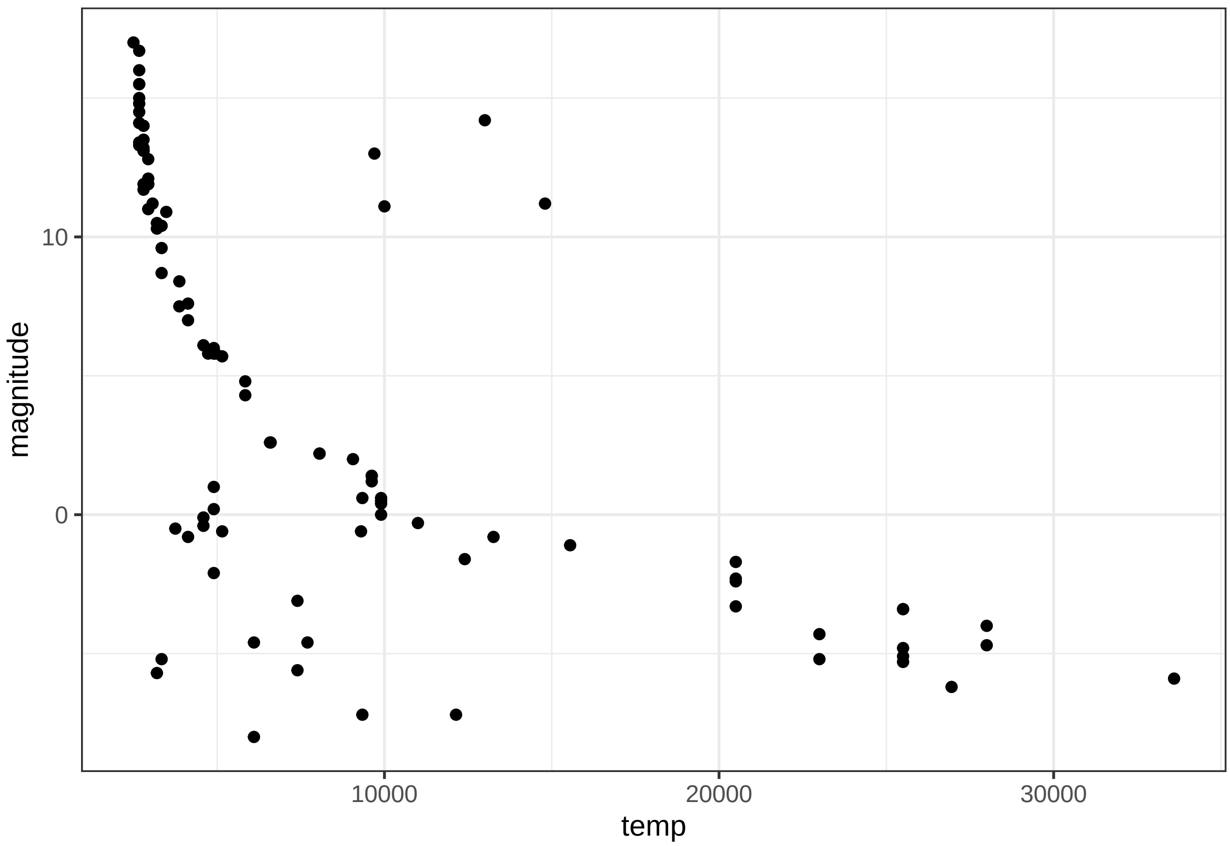
Which of these statements best characterizes the temperature distribution?

- ☐ A. The majority of stars have a high temperature.
- ☒ B. The majority of stars have a low temperature.
- ☐ C. The temperature distribution is normal.
- ☐ D. There are equal numbers of stars across the temperature range.

4. Make a scatter plot of the data with temperature on the x-axis and magnitude on the y-axis and examine the relationship between the variables. Recall that lower magnitude means a more luminous (brighter) star.

Most stars follow a ________________________ trend. These are called main sequence stars.

```r
stars %>%
  ggplot(aes(temp, magnitude)) +
  geom_point()
```

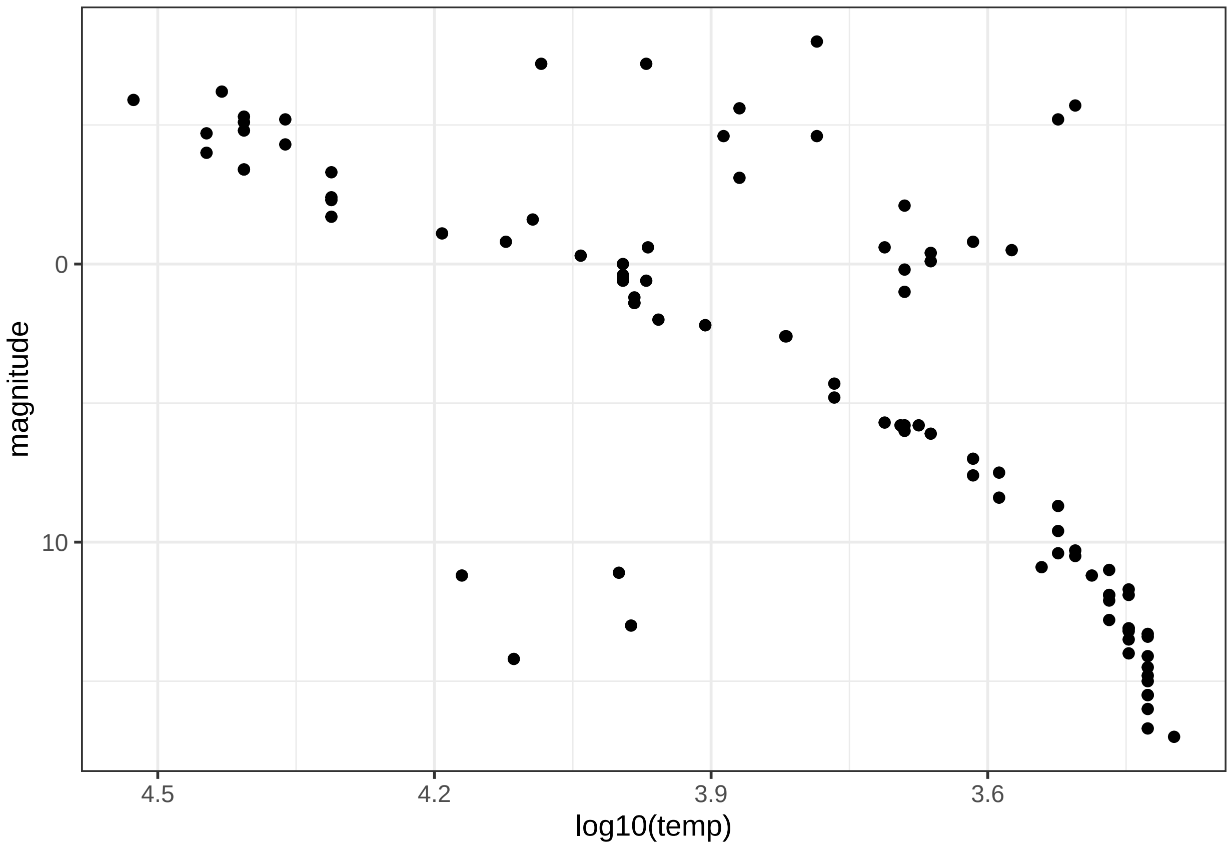


Fill in the blank:

- ☐ A. decreasing linear
- ☐ B. increasing linear
- ☒ C. decreasing exponential
- ☐ D. increasing exponential

5. For various reasons, scientists do not always follow straight conventions when making plots, and astronomers usually transform values of star luminosity and temperature before plotting. Flip the y-axis so that lower values of magnitude are at the top of the axis (recall that **more luminous stars have lower magnitude**) using `scale_y_reverse`. Take the log base 10 of temperature and then also flip the x-axis.

```r
stars %>%
  ggplot(aes(x=log10(temp), magnitude)) +
  scale_x_reverse() +
  scale_y_reverse() +
  geom_point()
```



Fill in the blanks in the statements below to describe the resulting plot.

The brighest, highest temperature stars are in the ____________________ corner of the plot.

☐ A. lower left
☐ B. lower right
☒ C. upper left
☐ D. upper right

For main sequence stars, hotter stars have _________________ luminosity.

☒ A. higher
☐ B. lower

6. The trends you see allow scientists to learn about the evolution and lifetime of stars. The primary group of stars to which most stars belong we will call the main sequence stars (discussed in question 4). Most stars belong to this main sequence, however some of the more rare stars are classified as “old” and “evolved” stars. These stars tend to be **hotter** stars, but also have **low luminosity**, and are known as white dwarfs.

How many white dwarfs are there in our sample?

`These stars are in the lower left of the plot from question 5. There are 4 stars in this region.`

7. Consider stars which are not part of the Main Group but are not old/evolved (white dwarf) stars. These stars must also be unique in certain ways and are known as giants. Use the plot from Question 5 to estimate the average temperature of a giant.

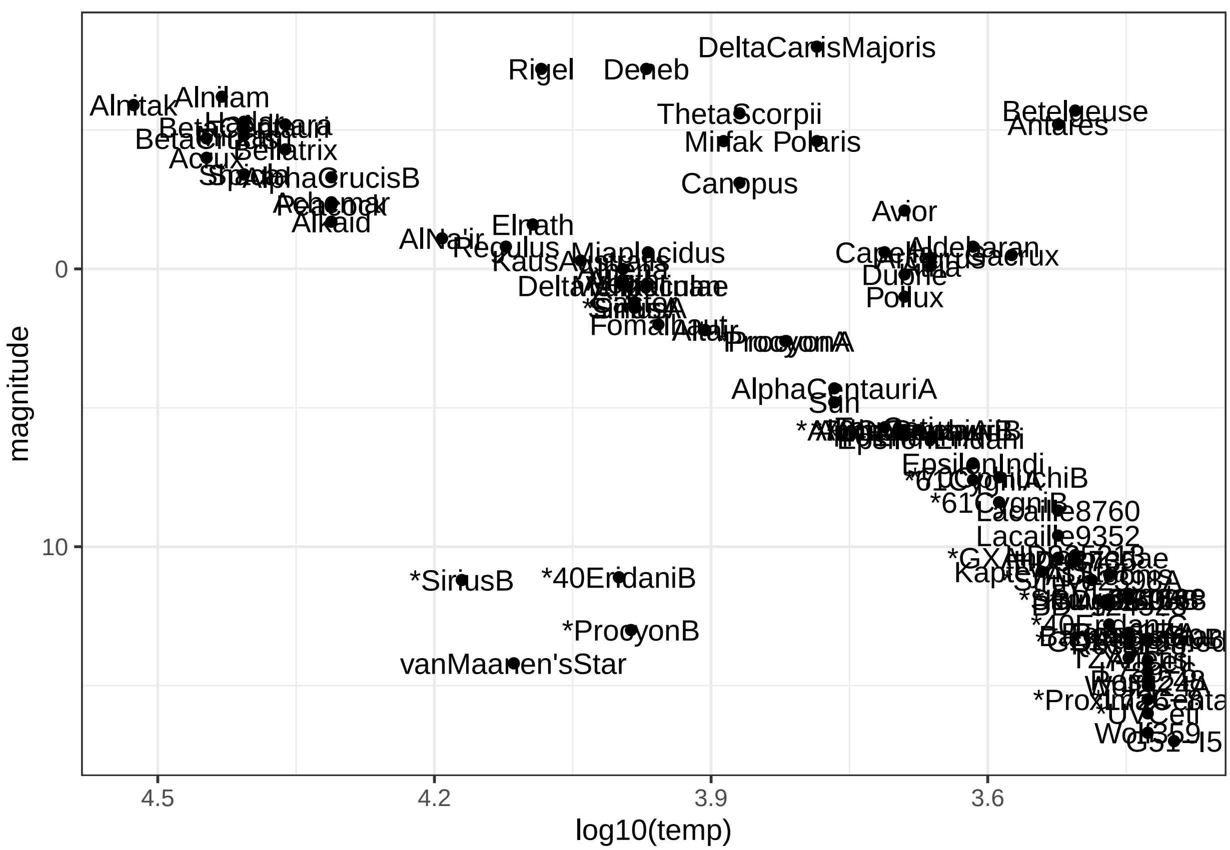
`Giants are in the upper right corner of the plot and generally have temperatures below 6000K.`

Which of these temperatures is closest to the average temperature of a giant?:

- ☒ A. 5000K
- ☐ B. 10000K
- ☐ C. 15000K
- ☐ D. 20000K

8. We can now identify whether specific stars are main sequence stars, red giants or white dwarfs. Add text labels to the plot to answer these questions. You may wish to plot only a selection of the labels, repel the labels, or zoom in on the plot in RStudio so you can locate specific stars.

```
stars %>%
    ggplot(aes(log10(temp), magnitude)) +
    geom_point() +
    geom_text(aes(label = star)) +
    scale_x_reverse() +
    scale_y_reverse()
```



Fill in the blanks in the statements below:

The least lumninous star in the sample with a surface temperature over 5000K is ______________.

- ☐ A. Antares
- ☐ B. Castor

☐ C. Mirfak
☐ D. Polaris
☒ E. van Maanen's Star

The two stars with lowest temperature and highest luminosity are known as supergiants. The two supergiants in this dataset are ________________.

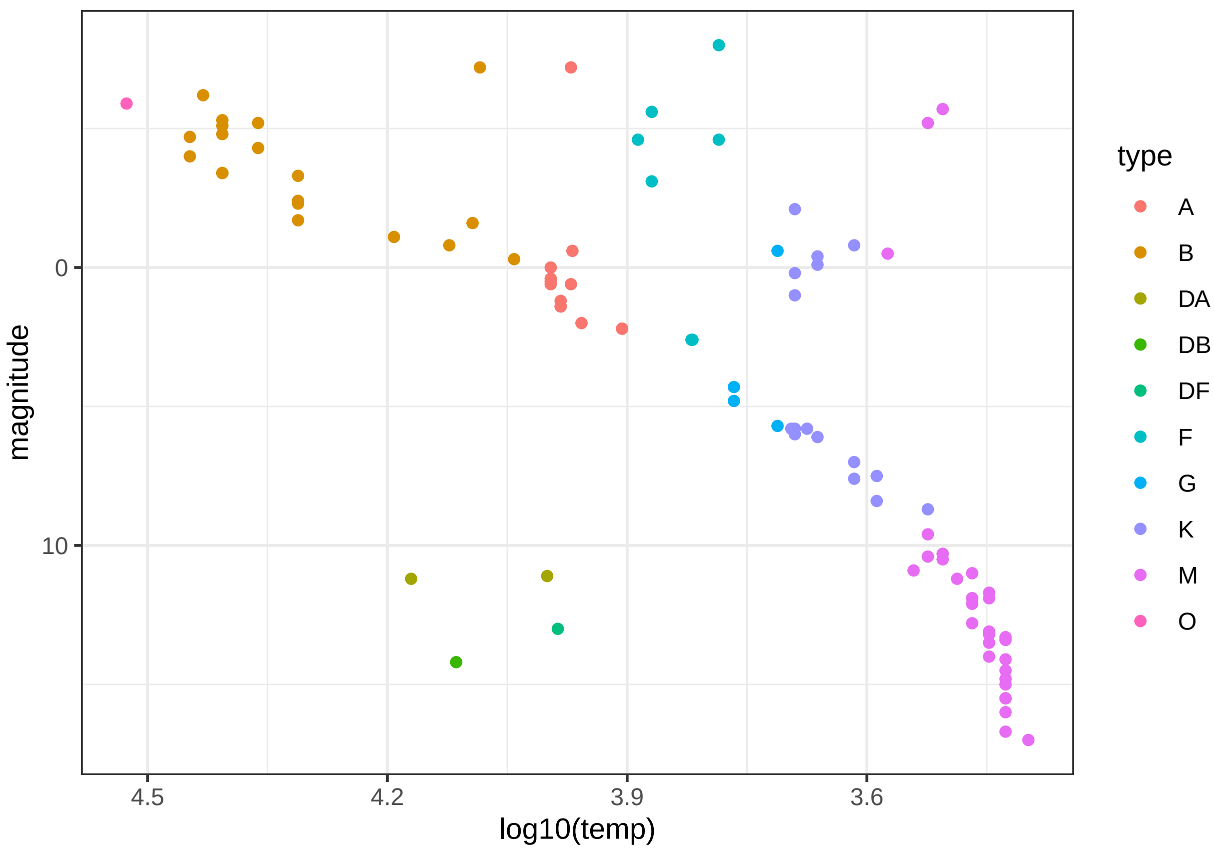
☐ A. Rigel and Deneb
☐ B. *SiriusB and van Maanen's Star
☐ C. Alnitak and Alnitam
☒ D. Betelgeuse and Antares
☐ E. Wolf359 and G51-I5

The Sun is a ___________________.

☒ A. main sequence star
☐ B. giant
☐ C. white dwarf

9. Remove the text labels and color the points by star type. This classification describes the properties of the star's spectrum, the amount of light produced at various wavelengths.

```
stars %>%
    ggplot(aes(log10(temp), magnitude, col = type)) +
    geom_point() +
    scale_x_reverse() +
    scale_y_reverse()
```



Which star type has the lowest temperature? `M`

Which star type has the highest temperature? `O`

The Sun is classified as a G-type star. Is the most luminous G-type star in this dataset also the hottest? `No`

Climate Change Exercises

Background

The planet's surface temperature is increasing due to human greenhouse gas emissions, and this global warming and carbon cycle disruption is wreaking havoc on natural systems. Living systems that depend on current temperature, weather, currents and carbon balance are jeopardized, and human society will be forced to contend with widespread economic, social, political and environmental damage as the temperature continues to rise. Although most countries recognize that global warming is a crisis and that humans must act to limit its effects, little action has been taken to limit or reverse human impact on the climate.

One limitation is the spread of misinformation related to climate change and its causes, especially the extent to which humans have contributed to global warming. In these exercises, we examine the relationship between global temperature changes, greenhouse gases and human carbon emissions using time series of actual atmospheric and ice core measurements from the National Oceanic and Atmospheric Administration (NOAA) and Carbon Dioxide Information Analysis Center (CDIAC).

Libraries and Options

```
data(temp_carbon)
data(greenhouse_gases)
data(historic_co2)
```

IMPORTANT: These exercises use **dslabs** datasets that were added in a July 2019 update. Make sure your package is up to date with the command `update.packages("dslabs")`. You can also update all packages on your system by running `update.packages()` with no arguments, and you should consider doing this routinely.

1. Load the `temp_carbon` dataset from **dslabs**, which contains annual global temperature anomalies (difference from 20th century mean temperature in degrees Celsius), temperature anomalies over the land and ocean, and global carbon emissions (in metric tons). Note that the date ranges differ for temperature and carbon emissions.

Which of these code blocks return the latest year for which carbon emissions are reported?

- [ ] A.

```
temp_carbon %>%
    .$year %>%
    max()
```

- [x] B.

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    pull(year) %>%
    max()
```

```
## [1] 2014
```

☐ C.

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    max(year)
```

☒ D.

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    .$year %>%
    max()
```

```
## [1] 2014
```

☒ E.

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    select(year) %>%
    max()
```

```
## [1] 2014
```

☐ F.

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    max(.$year)
```

2. Inspect the difference in carbon emissions in `temp_carbon` from the first available year to the last available year.

What is the first year for which carbon emissions (`carbon_emissions`) data are available?

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    .$year %>%
    min()
```

```
## [1] 1751
```

What is the last year for which carbon emissions data are available?

```
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    .$year %>%
    max()
```

```
## [1] 2014
```

How many times larger were carbon emissions in the last year relative to the first year?

```
carbon1 <- temp_carbon %>%
    filter(year == 1751) %>%
    .$carbon_emissions

carbon2 <- temp_carbon %>%
    filter(year == 2014) %>%
    .$carbon_emissions

carbon2/carbon1
```

```
## [1] 3285
```

3. Inspect the difference in temperature in `temp_carbon` from the first available year to the last available year.

What is the first year for which global temperature anomaly (`temp_anomaly`) data are available?

```
temp_carbon %>%
    filter(!is.na(temp_anomaly)) %>%
    .$year %>%
    min()
```

```
## [1] 1880
```

What is the last year for which global temperature anomaly data are available?

```
temp_carbon %>%
    filter(!is.na(temp_anomaly)) %>%
    .$year %>%
    max()
```

```
## [1] 2018
```

How many degrees Celsius has temperature increased over the date range? Compare the temperatures in the most recent year versus the oldest year.

```
temp1 <- temp_carbon %>%
    filter(year == "1880") %>%
    .$temp_anomaly

temp2 <- temp_carbon %>%
    filter(year == "2018") %>%
    .$temp_anomaly

temp2 - temp1
```

```
## [1] 0.93
```

4. Create a time series line plot of the temperature anomaly. Only include years where temperatures are reported. Save this plot to the object `p`.

Which command adds a blue horizontal line indicating the 20th century mean temperature?

☐ A.

```
p <- p + geom_vline(aes(xintercept = 0), col = "blue")
```

☐ B.

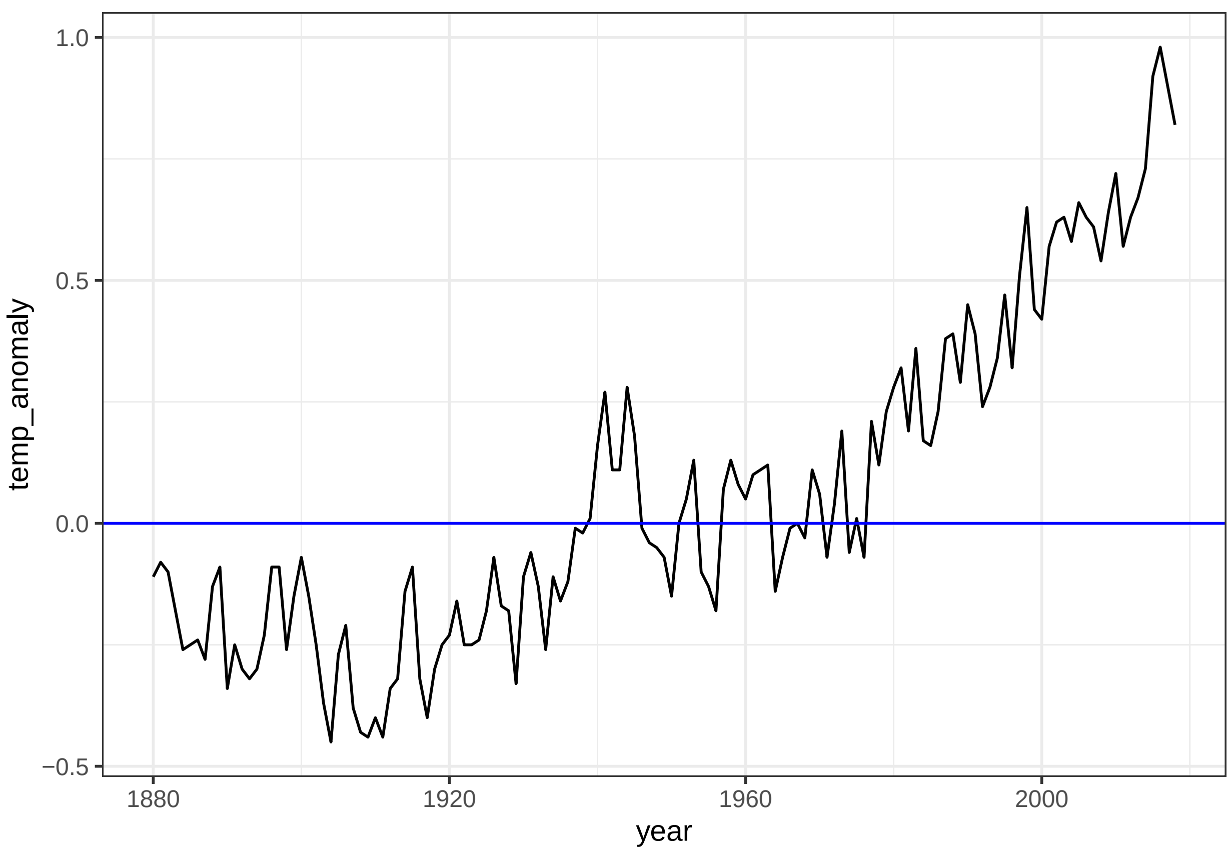
```
p <- p + geom_hline(aes(y = 0), col = "blue")
```

☐ C.

```
p <- p + geom_hline(aes(yintercept = 0, col = blue))
```

☒ D.

```
p <- temp_carbon %>%
    filter(!is.na(temp_anomaly)) %>%
    ggplot(aes(year, temp_anomaly)) +
    geom_line()

p <- p + geom_hline(aes(yintercept = 0), col = "blue")
p
```



5. Continue working with `p`, the plot created in the previous question.

Change the y-axis label to be “Temperature anomaly (degrees C)”. Add a title, “Temperature anomaly relative to 20th century mean, 1880-2018”. Also add a text layer to the plot: the x-coordinate should be 2000, the y-coordinate should be 0.05, the text should be “20th century mean”, and the text color should be blue.

☐ A.

```
p + ylab("Temperature anomaly (degrees C)") +
  title("Temperature anomaly relative to 20th century mean, 1880-2018") +
  geom_text(aes(x = 2000, y = 0.05, label = "20th century mean", col = "blue"))
```

☐ B.

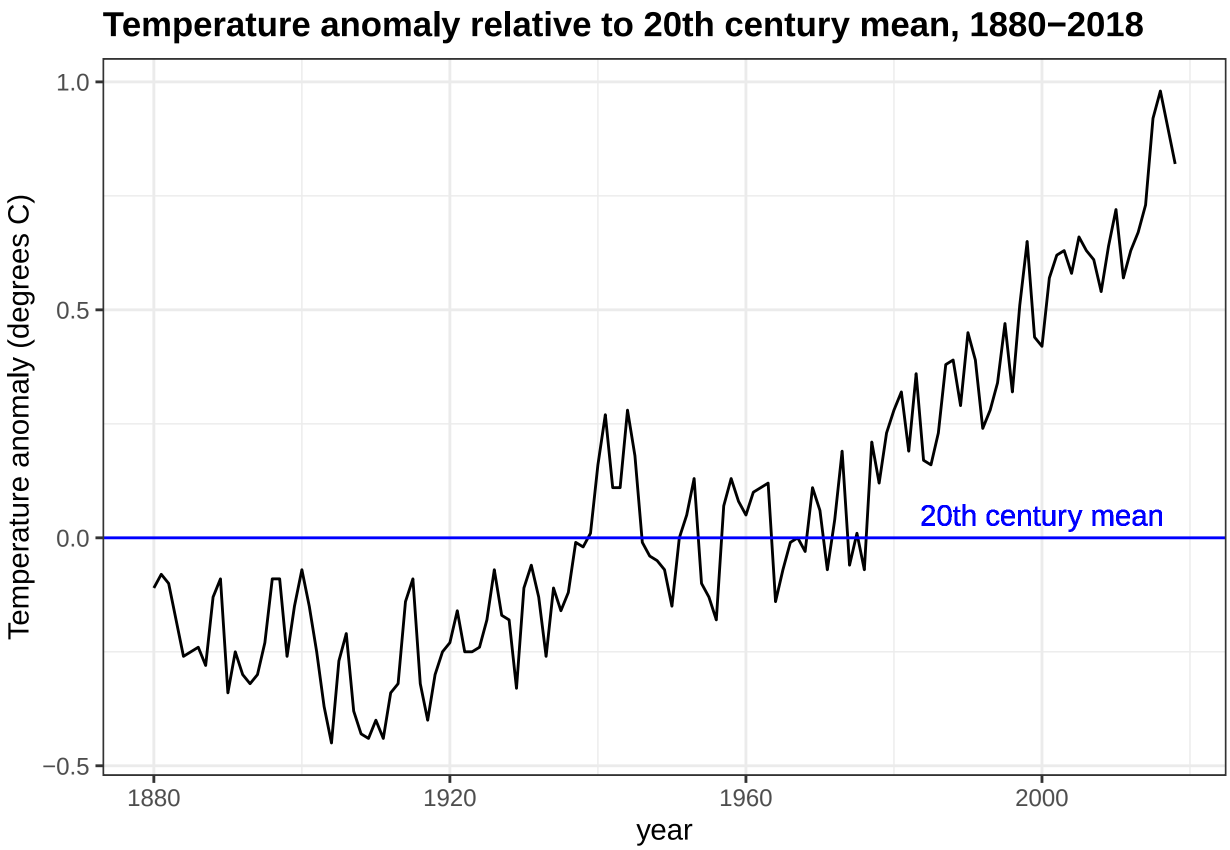
```
p + ylim("Temperature anomaly (degrees C)") +
  ggtitle("Temperature anomaly relative to 20th century mean, 1880-2018") +
  geom_text(aes(x = 2000, y = 0.05, label = "20th century mean"), col = "blue")
```

☐ C.

```
p + ylab("Temperature anomaly (degrees C)") +
  ggtitle("Temperature anomaly relative to 20th century mean, 1880-2018") +
  geom_text(aes(x = 2000, y = 0.05, label = "20th century mean", col = "blue"))
```

☒ D.

```
p <- temp_carbon %>%
  filter(!is.na(temp_anomaly)) %>%
  ggplot(aes(year, temp_anomaly)) +
  geom_line() +
  geom_hline(aes(yintercept=0), col='blue') +
  ylab("Temperature anomaly (degrees C)") +
  ggtitle("Temperature anomaly relative to 20th century mean, 1880-2018") +
  geom_text(aes(x = 2000, y = 0.05, label="20th century mean"), col='blue')
p
```



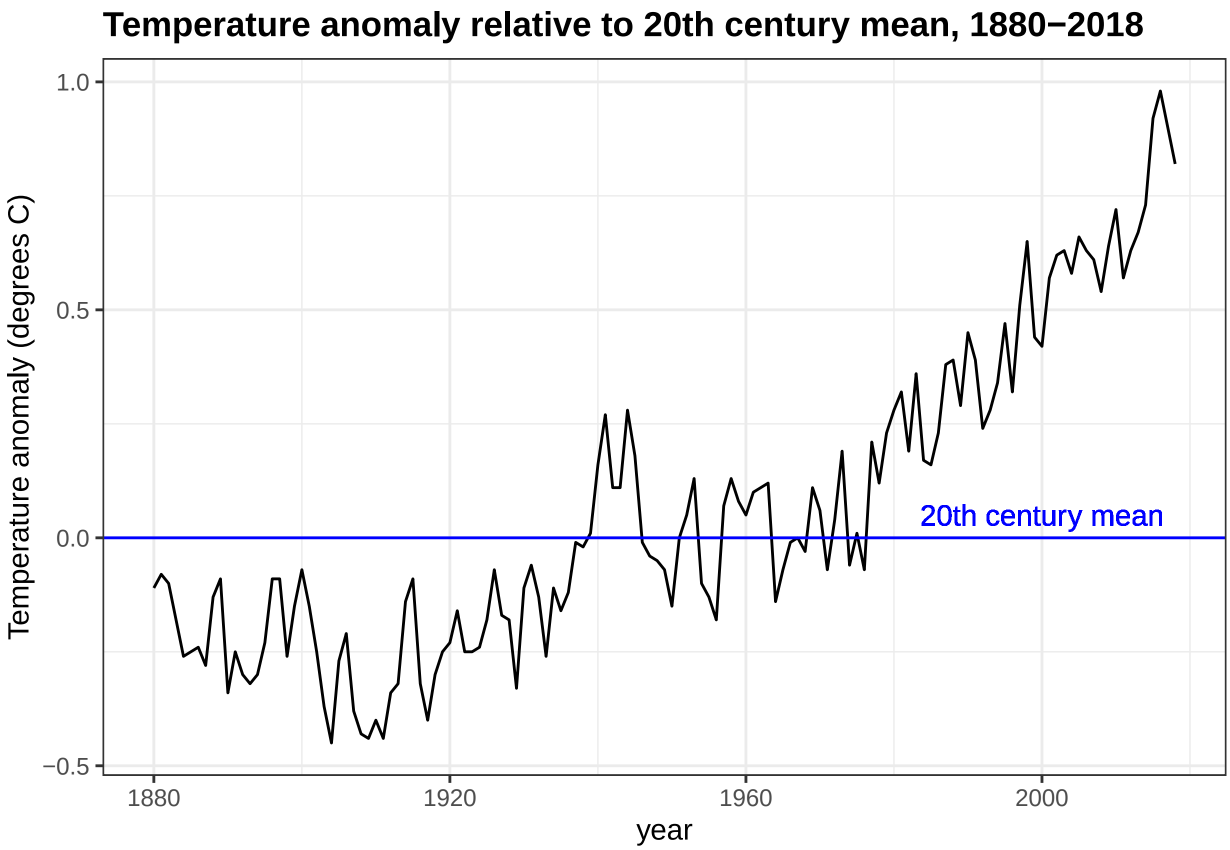
☐ E.

```
p + ylab("Temperature anomaly (degrees C)") +
    title("Temperature anomaly relative to 20th century mean, 1880-2018") +
    geom_text(aes(x = 2000, y = 0.05, label = "20th century mean"), col = "blue")
```

6. Use the plot created in the last two exercises to answer the following questions.

Answers within 5 years of the correct answer will be accepted.

```
temp_carbon %>%
    filter(!is.na(temp_anomaly)) %>%
    ggplot(aes(year, temp_anomaly)) +
    geom_line() +
    geom_hline(aes(yintercept = 0), col = "blue") +
    ylab("Temperature anomaly (degrees C)") +
    geom_text(aes(x = 2000, y = 0.05, label = "20th century mean"), col = "blue") +
    xlim(c(1880, 2018)) +
    ggtitle("Temperature anomaly relative to 20th century mean, 1880-2018")
```



When was the earliest year with a temperature above the 20th century mean? `1940`

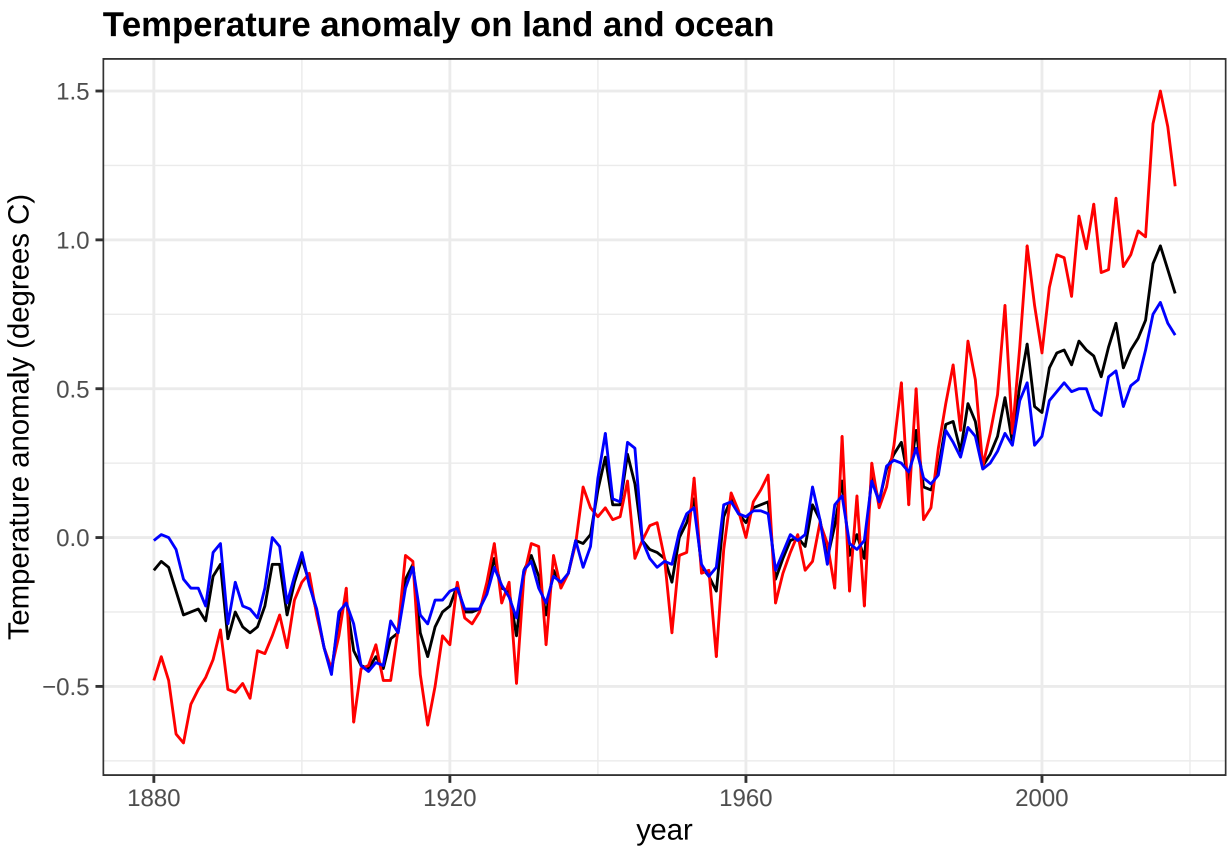
When was the last year with an average temperature below the 20th century mean? `1976`

In what year did the temperature anomaly exceed 0.5 degrees Celsius for the first time? `1997`

7. Add layers to the previous plot to include line graphs of the temperature anomaly in the ocean (`ocean_anomaly`) and on land (`land_anomaly`).

Assign different colors to the lines. Compare the global temperature anomaly to the land temperature anomaly and ocean temperature anomaly.

```
temp_carbon %>%
    filter(!is.na(temp_anomaly)) %>%
    ggplot(aes(year, temp_anomaly)) +
    geom_line() +
    geom_line(aes(year, land_anomaly), col = "red") +
    geom_line(aes(year, ocean_anomaly), col = "blue") +
    ylab("Temperature anomaly (degrees C)") +
    xlim(c(1880, 2018)) +
    ggtitle("Temperature anomaly on land and ocean")
```



Which region has the largest 2018 temperature anomaly relative to the 20th century mean? `Land`

Which region has the largest change in temperature since 1880? `Land`

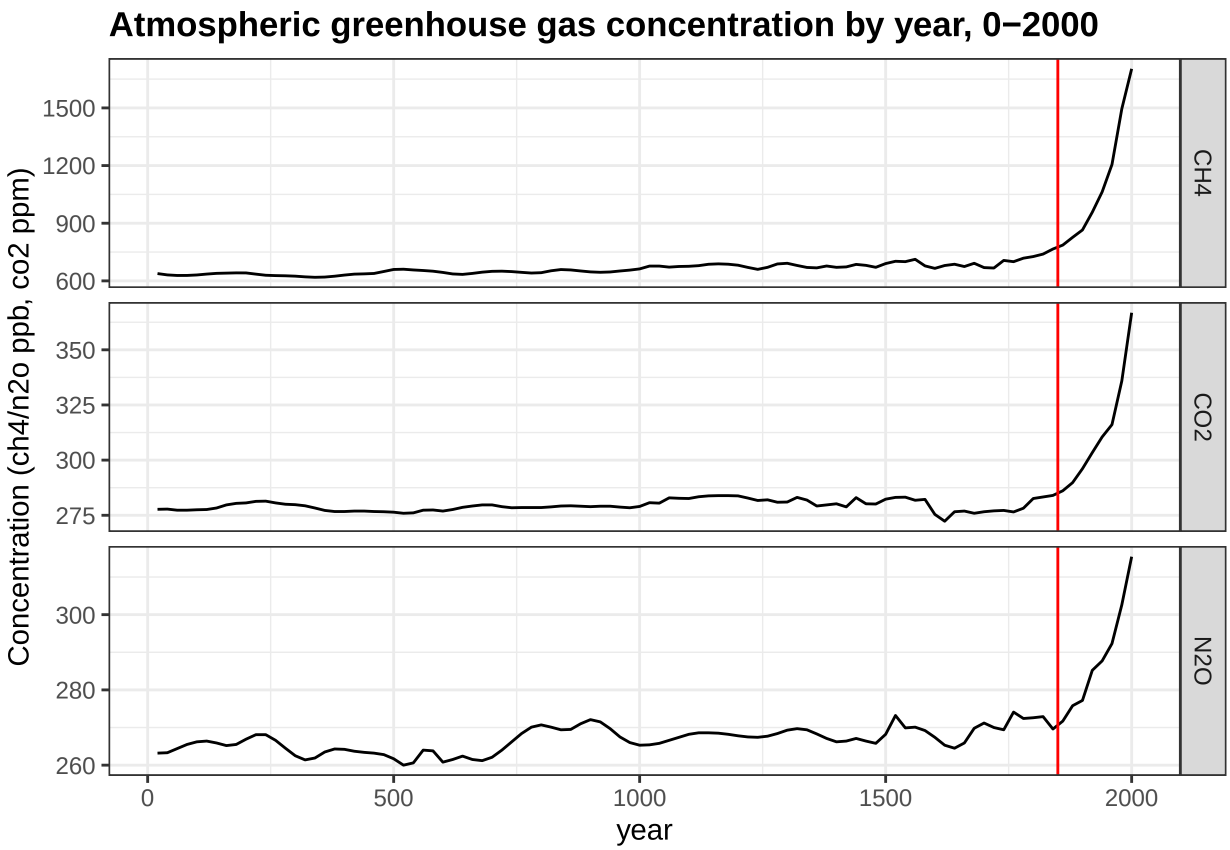
Which region has a temperature anomaly pattern that more closely matches the global pattern? `Ocean`

8. A major determinant of Earth's temperature is the greenhouse effect. Many gases trap heat and reflect it towards the surface, preventing heat from escaping the atmosphere. The greenhouse effect is vital in keeping Earth at a warm enough temperature to sustain liquid water and life; however, changes in greenhouse gas levels can alter the temperature balance of the planet.

The `greenhouse_gases` data frame from **dslabs** contains concentrations of the three most significant greenhouse gases: carbon dioxide (CO_2, abbreviated in the data as `co2`), methane (CH_4, `ch4` in the data), and nitrous oxide (N_2O, `n2o` in the data). Measurements are provided every 20 years for the past 2000 years.

Complete the code outline below to make a line plot of `concentration` on the y-axis by year on the x-axis. Facet by `gas`, aligning the plots vertically so as to ease comparisons along the year axis. Add a vertical line with an x-intercept at the year 1850, noting the unofficial start of the industrial revolution and widespread fossil fuel consumption. Note that the units for `ch4` and `n2o` are ppb while the units for `co2` are ppm.

```
greenhouse_gases %>%
    ggplot(aes(year, concentration)) +
    geom_line() +
    facet_grid(gas ~ ., scales = "free") +
    geom_vline(xintercept = 1850, col='red') +
    ylab("Concentration (ch4/n2o ppb, co2 ppm)") +
    ggtitle("Atmospheric greenhouse gas concentration by year, 0-2000")
```



What code fills the first blank? `year, concentration`

What code fills the second blank? Make sure to align plots vertically. `gas ~ .`

What code fills the third blank? `geom_vline(xintercept = 1850)`

9. Interpret the plot of greenhouse gases over time from the previous question. You will use each answer exactly once `ch4`, `co2`, `n2o`, all, none).

Which gas was stable at approximately 275 ppm/ppb until around 1850? `co2`

Which gas more than doubled in concentration since 1850? `ch4`

Which gas decreased in concentration since 1850? `none`

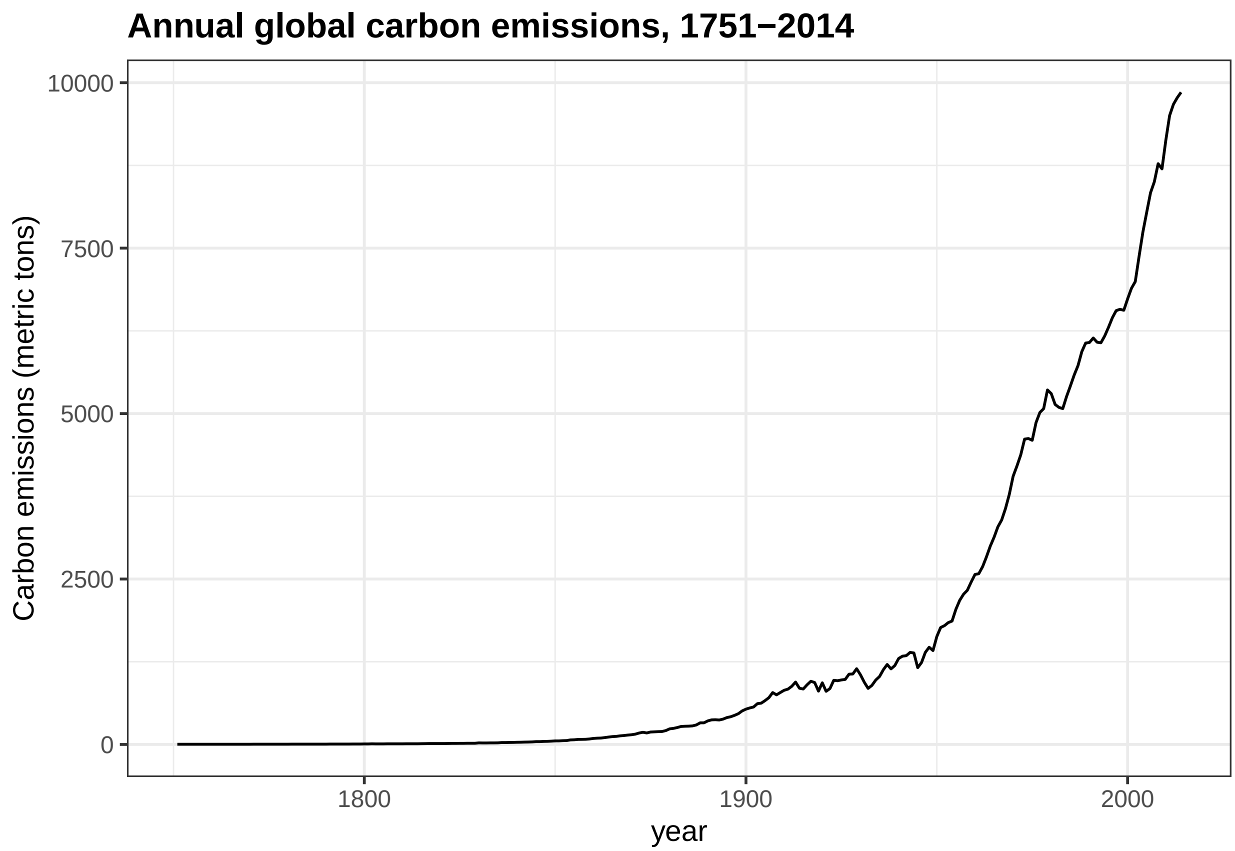
Which gas had the smallest magnitude change since 1850? `n2o`

Which gas increased exponentially in concentration after 1850? `all`

10. While many aspects of climate are independent of human influence, and `co2` levels can change without human intervention, climate models cannot reconstruct current conditions without incorporating the effect of manmade carbon emissions. These emissions consist of greenhouse gases and are mainly the result of burning fossil fuels such as oil, coal and natural gas.

Make a time series line plot of carbon emissions (`carbon_emissions`) from the `temp_carbon` dataset. The y-axis is metric tons of carbon emitted per year.

```r
temp_carbon %>%
    filter(!is.na(carbon_emissions)) %>%
    ggplot(aes(year, carbon_emissions)) +
    geom_line() +
    ylab("Carbon emissions (metric tons)") +
    ggtitle("Annual global carbon emissions, 1751-2014")
```



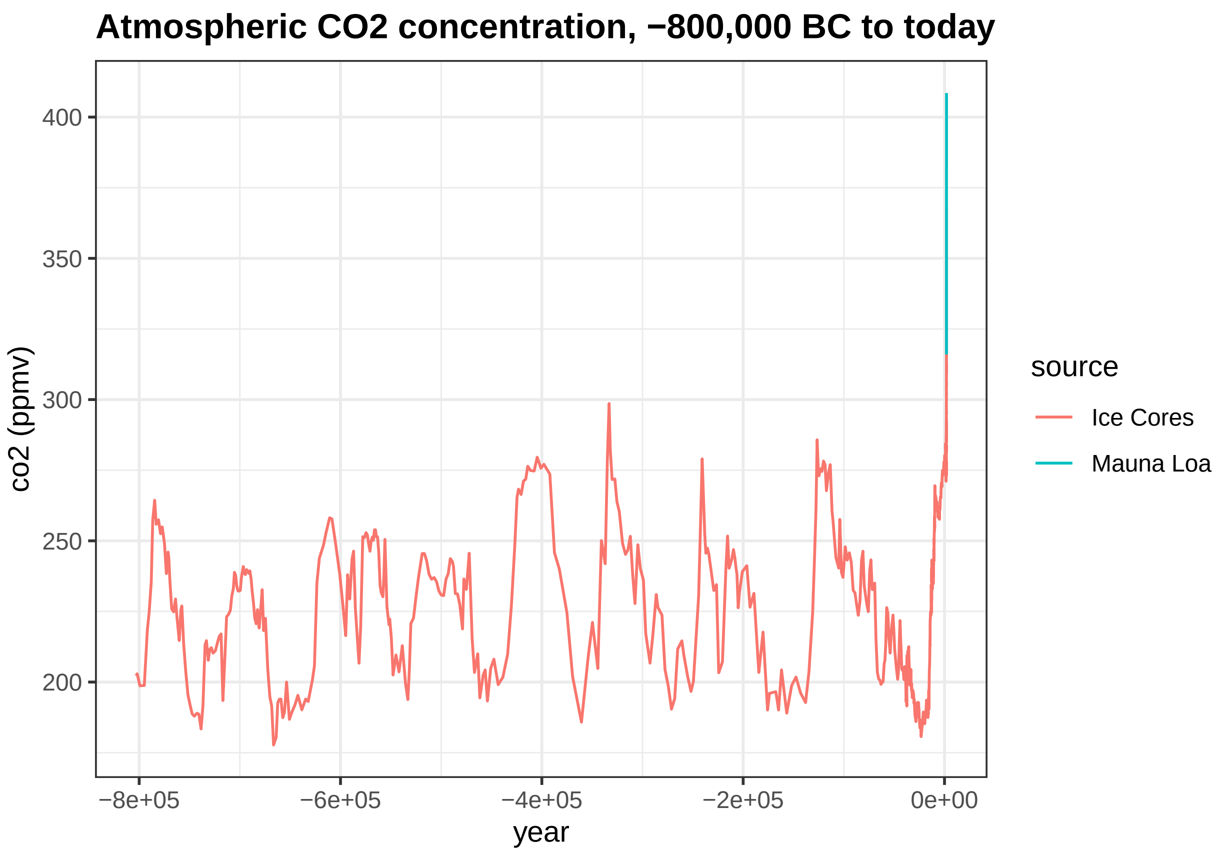
Which of the following are true about the trend of carbon emissions? Check all correct answers.

- ☒ A. Carbon emissions were essentially zero before 1850 and have increased exponentially since then.
- ☐ B. Carbon emissions are reaching a stable level.
- ☐ C. Carbon emissions have increased every year on record.
- ☒ D. Carbon emissions in 2014 were about 4 times as large as 1960 emissions.
- ☒ E. Carbon emissions have doubled since the late 1970s.
- ☒ F. Carbon emissions change with the same trend as atmospheric greenhouse gas levels (`co2`, `ch4`, `n2o`)

11. We saw how greenhouse gases have changed over the course of human history, but how has CO_2 (`co2` in the data) varied over a longer time scale? The `historic_co2` data frame in **dslabs** contains direct measurements of atmospheric `co2` from Mauna Loa since 1959 as well as indirect measurements of atmospheric `co2` from ice cores dating back 800,000 years.

Make a line plot of `co2` concentration over time (`year`), coloring by the measurement source (`source`). Save this plot as `co2_time` for later use.

```r
co2_time <- historic_co2 %>%
  filter(!is.na(co2)) %>%
  ggplot(aes(year, co2, col=source)) +
  geom_line() +
  ggtitle("Atmospheric CO2 concentration, -800,000 BC to today") +
  ylab("co2 (ppmv)")
co2_time
```



Which of the following are true about `co2_time`, the time series of `co2` over the last 800,000 years? Check all correct answers.

- ☒ A. Modern co2 levels are higher than at any point in the last 800,000 years.
- ☒ B. There are natural cycles of co2 increase and decrease lasting 50,000-100,000 years per cycle.
- ☒ C. In most cases, it appears to take longer for co2 levels to decrease than to increase.
- ☐ D. co2 concentration has been at least 200 ppm for the last 800,000 years.

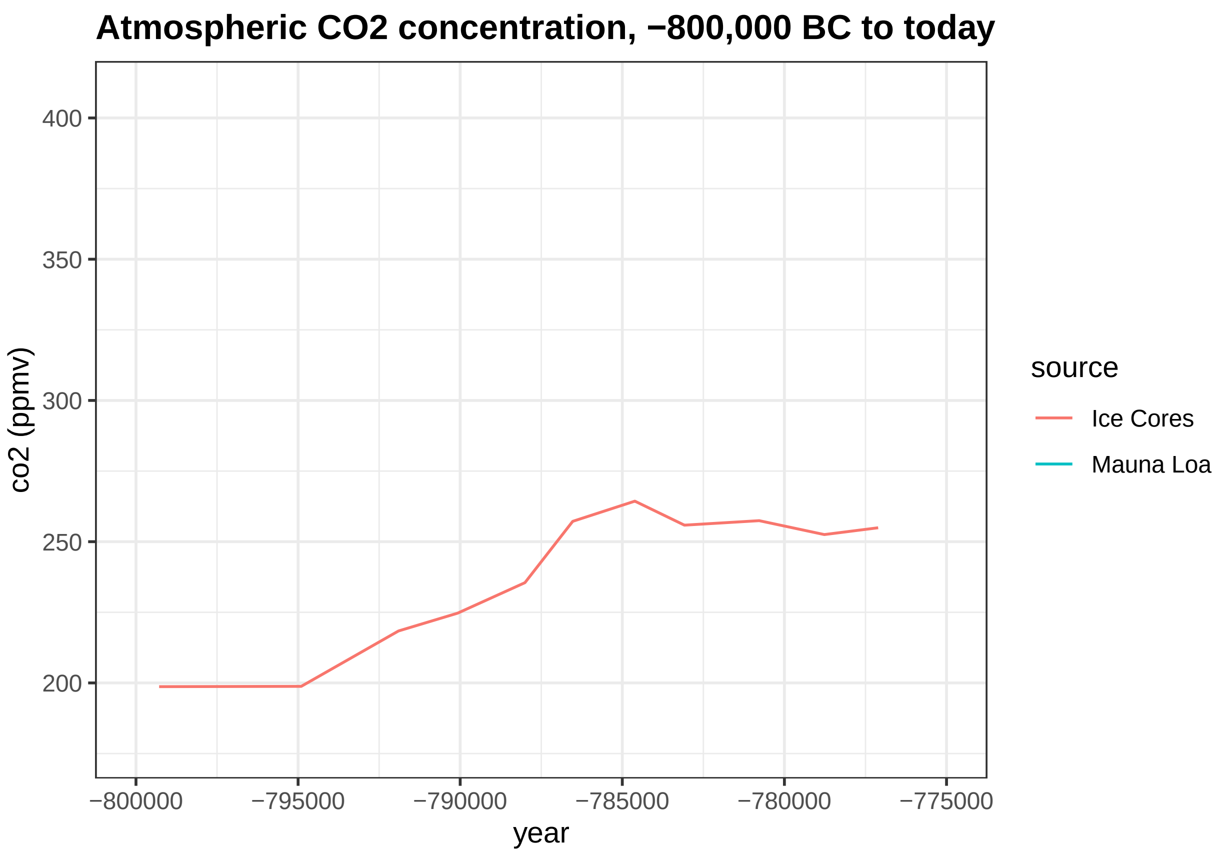
12. One way to differentiate natural `co2` oscillations from today's manmade `co2` spike is by examining the rate of change of `co2`. The planet is affected not only by the absolute concentration of `co2` but also by its rate of change. When the rate of change is slow, living and nonliving systems have time to adapt to new temperature and gas levels, but when the rate of change is fast, abrupt differences can overwhelm natural systems. How does the pace of natural `co2` change differ from the current rate of change?

Use the `co2_time` plot saved above. Change the limits as directed to investigate the rate of change in `co2` over various periods with spikes in `co2` concentration.

Change the x-axis limits to -800,000 and -775,000. About how many years did it take for `co2` to rise from 200 ppmv to its peak near 275 ppmv?

```r
co2_time <- historic_co2 %>%
    ggplot(aes(year, co2, col = source)) +
    geom_line() +
    ggtitle("Atmospheric CO2 concentration, -800,000 BC to today") +
    ylab("co2 (ppmv)")
co2_time + xlim(-800000, -775000)
```

```
## Warning: Removed 683 row(s) containing missing values (geom_path).
```

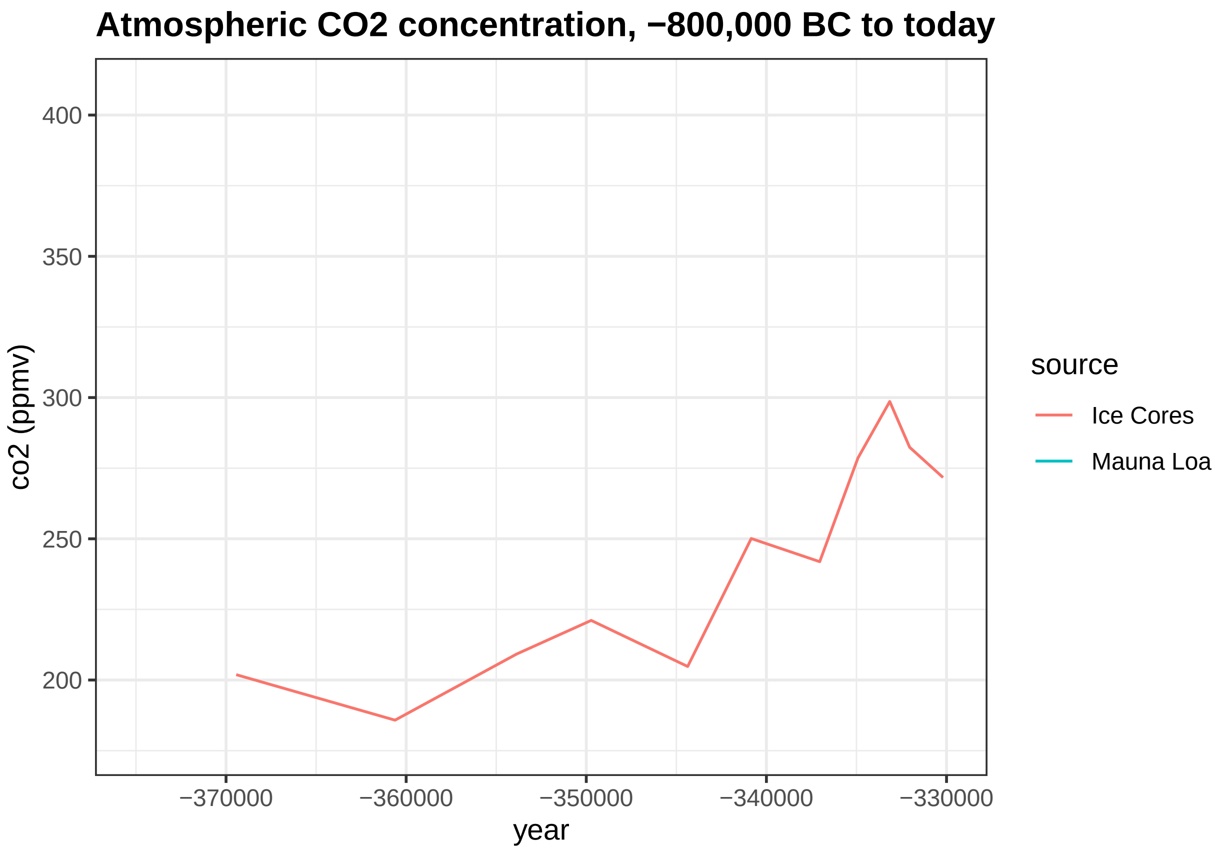


- ☐ A. 100
- ☐ B. 3,000
- ☐ C. 6,000
- ☒ D. 10,000

Change the x-axis limits to -375,000 and -330,000. About how many years did it take for co2 to rise from the minimum of 180 ppm to its peak of 300 ppmv?

```r
co2_time <- historic_co2 %>%
    ggplot(aes(year, co2, col = source)) +
    geom_line() +
    ggtitle("Atmospheric CO2 concentration, -800,000 BC to today") +
    ylab("co2 (ppmv)")
co2_time + xlim(-375000, -330000)
```

```
## Warning: Removed 683 row(s) containing missing values (geom_path).
```

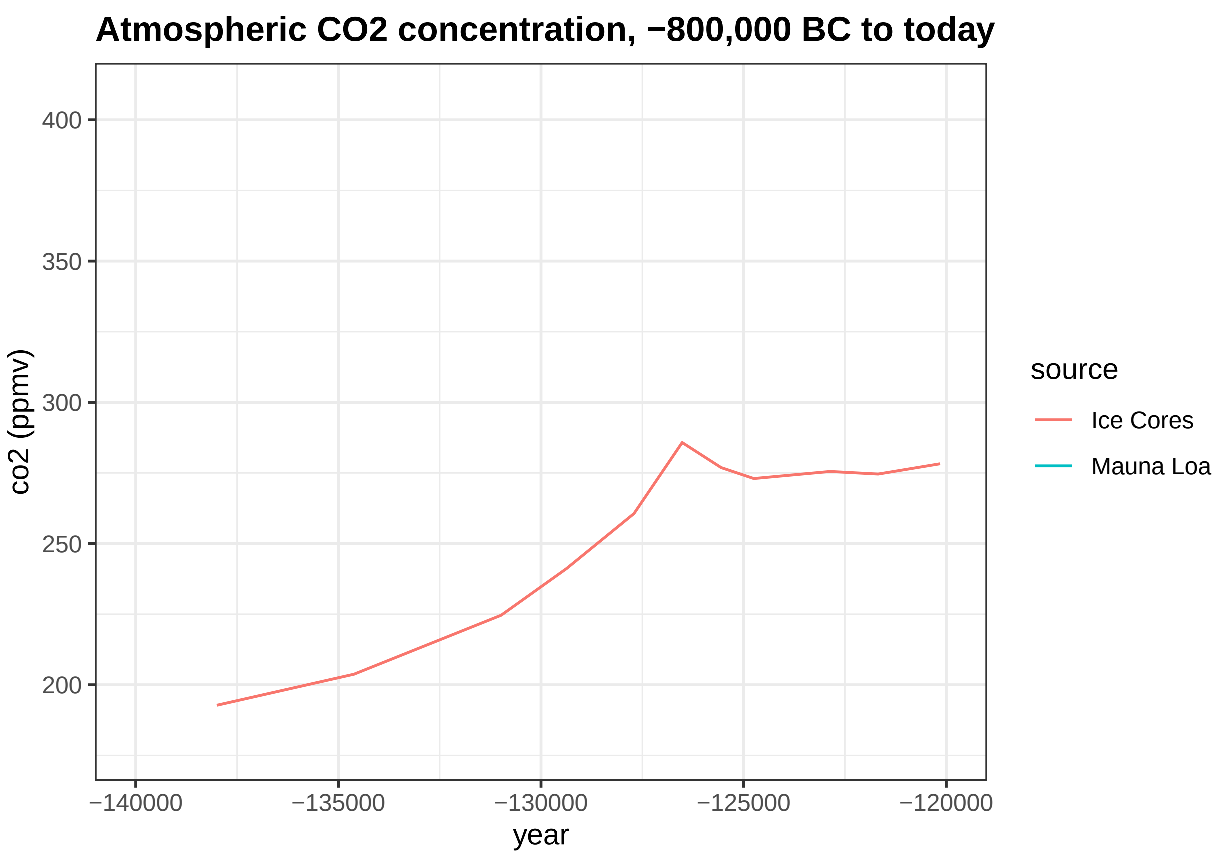


☐ A. 3,000
☐ B. 6,000
☐ C. 12,000
☒ D. 25,000

Change the x-axis limits to -140,000 and -120,000. About how many years did it take for co2 to rise from 200 ppmv to its peak near 280 ppmv?

```
co2_time <- historic_co2 %>%
    ggplot(aes(year, co2, col = source)) +
    geom_line() +
    ggtitle("Atmospheric CO2 concentration, -800,000 BC to today") +
    ylab("co2 (ppmv)")
co2_time + xlim(-140000, -120000)
```

```
## Warning: Removed 683 row(s) containing missing values (geom_path).
```

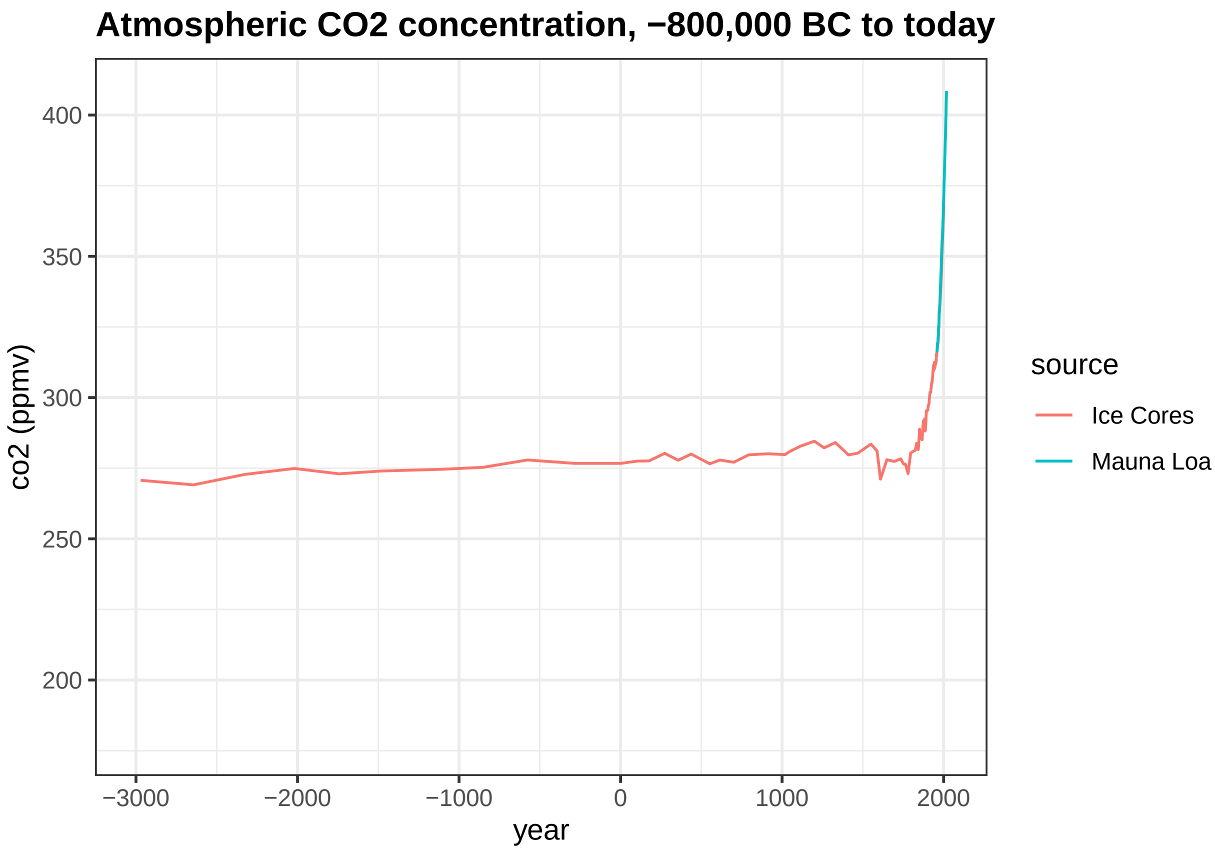


- ☐ A. 3200
- ☐ B. 1,500
- ☐ C. 5,000
- ☒ D. 9,000

Change the x-axis limits to -3000 and 2018 to investigate modern changes in co2. About how many years did it take for co2 to rise from its stable level around 275 ppmv to the current level of over 400 ppmv?

```
co2_time <- historic_co2 %>%
    ggplot(aes(year, co2, col = source)) +
    geom_line() +
    ggtitle("Atmospheric CO2 concentration, -800,000 BC to today") +
    ylab("co2 (ppmv)")
co2_time + xlim(-3000, 2018)
```

```
## Warning: Removed 539 row(s) containing missing values (geom_path).
```



- ☒ A. 250
- ☐ B. 1,000
- ☐ C. 2,000
- ☐ D. 5,000