

# 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.

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

```
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 entries:

```
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 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?

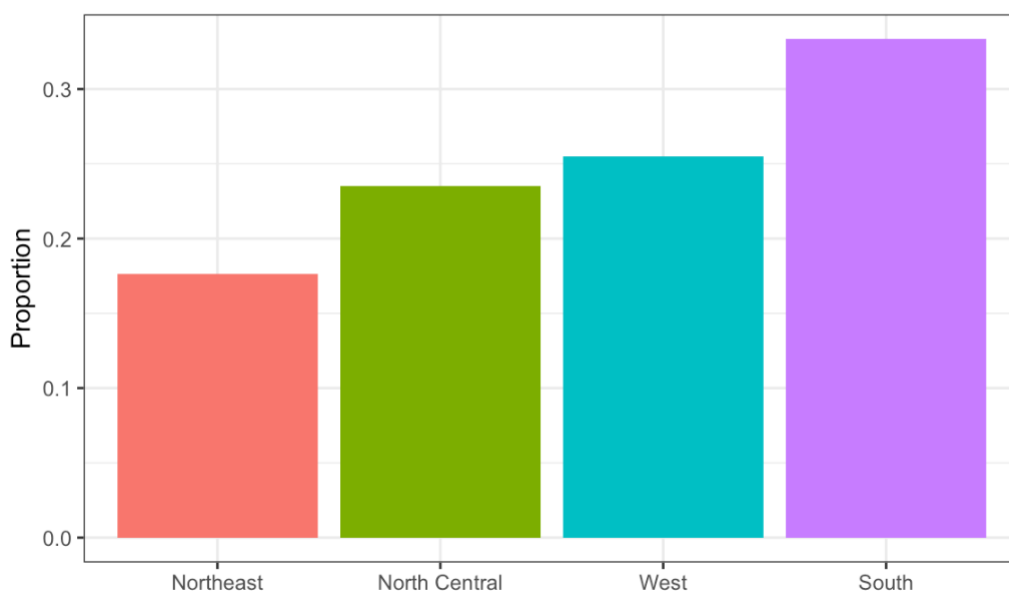


Figure 1: Region vs. Proportion

- ☐ 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?

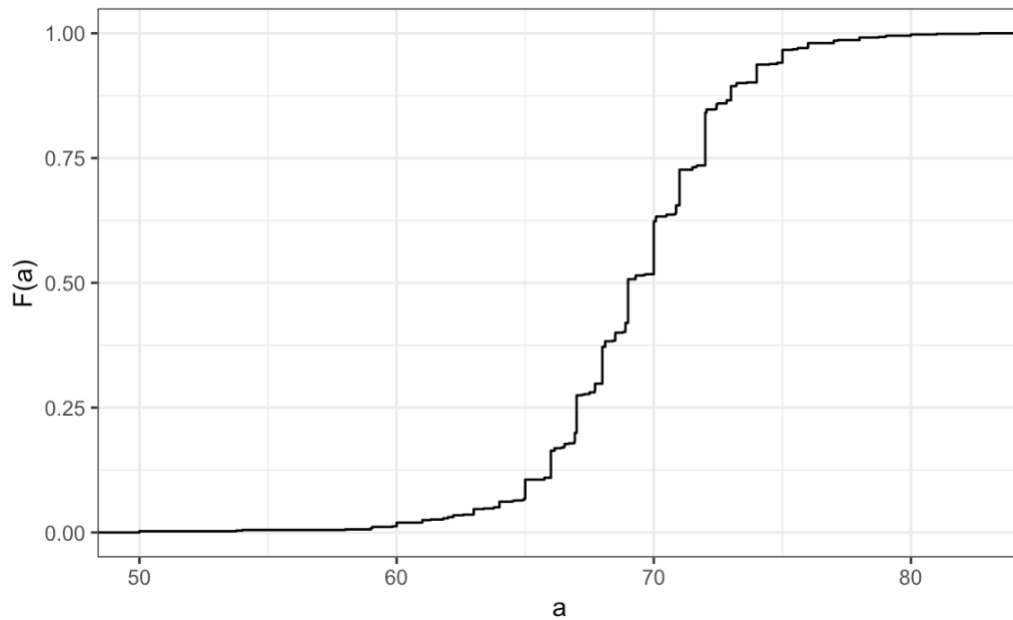


Figure 2: eCDF for male heights

- ☐ 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.

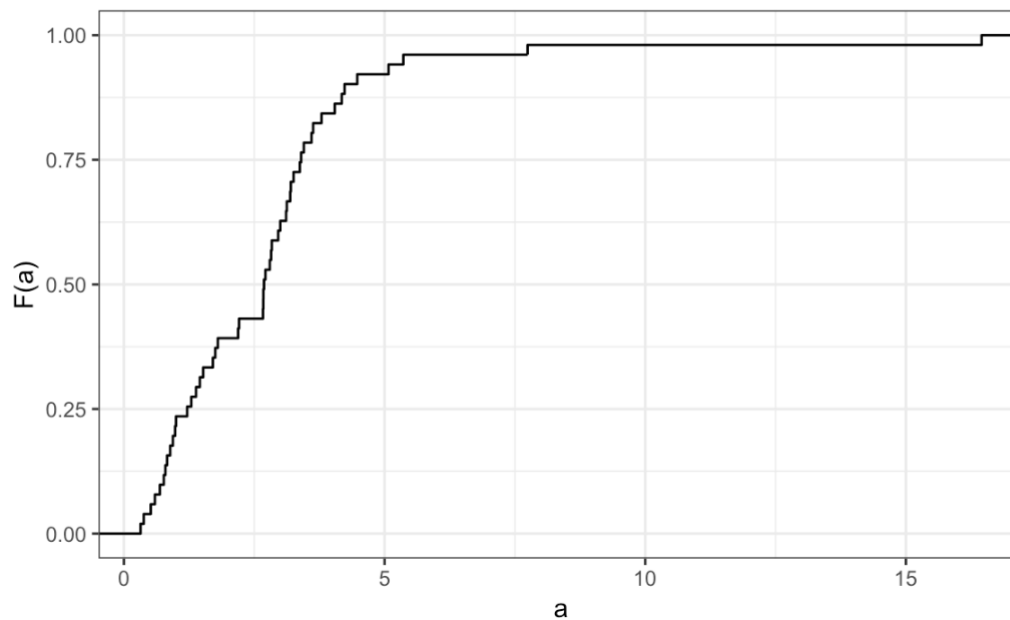


Figure 3: eCDF of the murder rates across states

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?

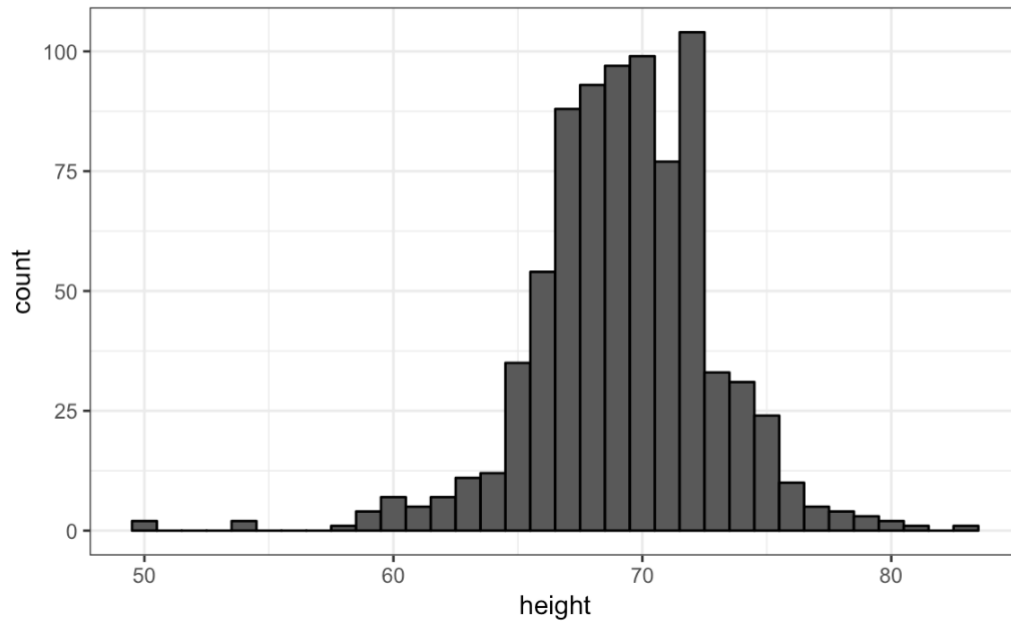


Figure 4: Histogram of male heights

- ☐ 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?

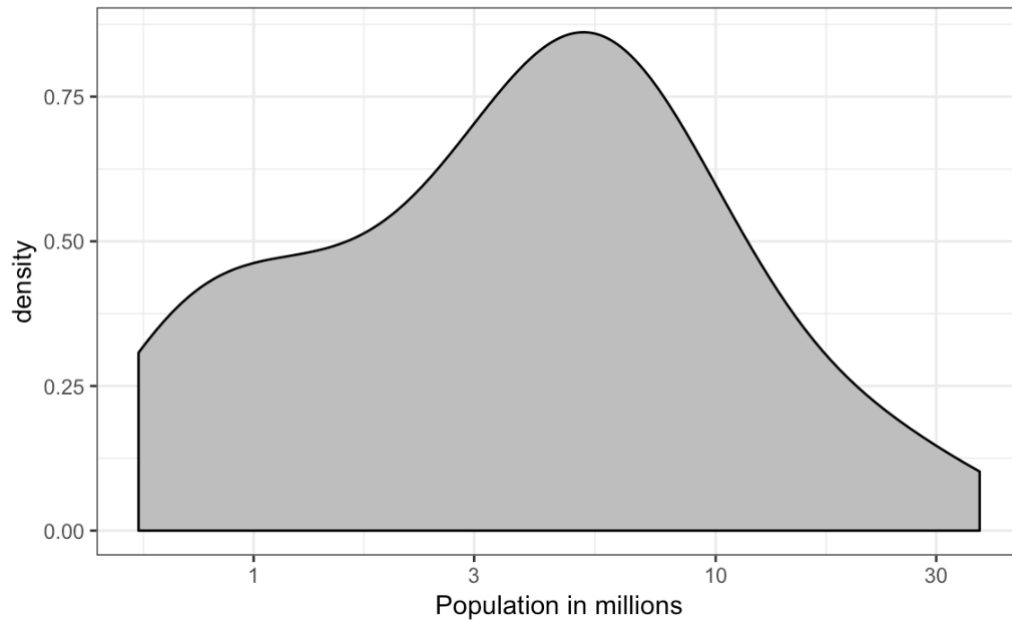


Figure 5: Density plot population

- ☐ 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.

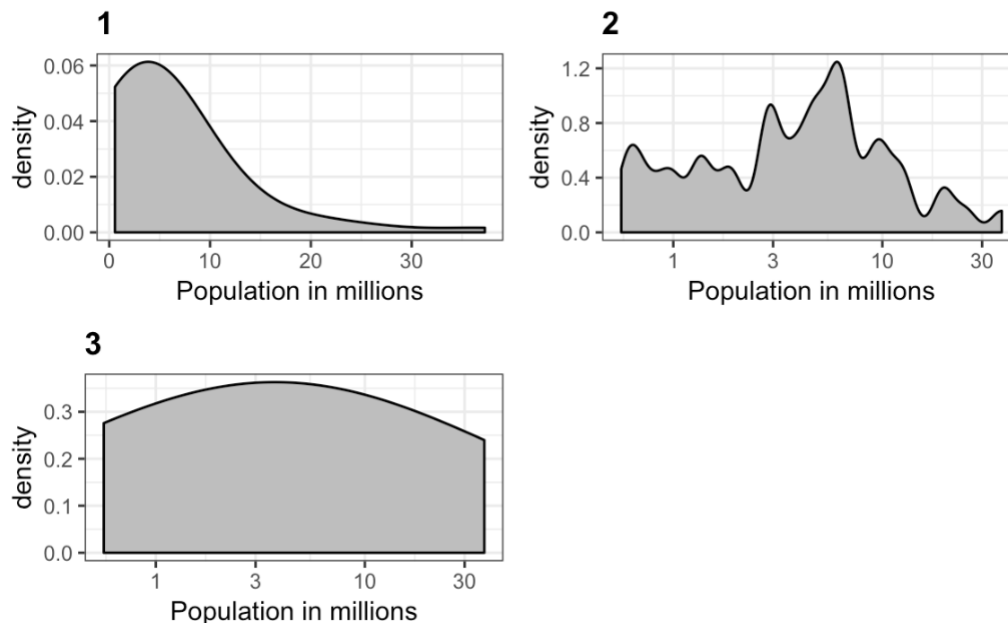


Figure 6: Three density plots

## 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 ( $\mu$ ) and standard deviation ( $\sigma$ )
  - Always has the same proportion of observations within a given distance of the mean (for example, 95% within  $2\sigma$ )
- 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 ( $\mu$ ):
 
$$Z = \frac{x - \mu}{\sigma}$$
- 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  $\mu$  and standard deviation  $\sigma$ :

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

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  $\sigma$  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`.

```
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.

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

```
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
```

```
approx
```

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

```
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 is normally distributed with an average of 69 inches and a standard deviation of 3 inches.

```
# 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 exercise 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)

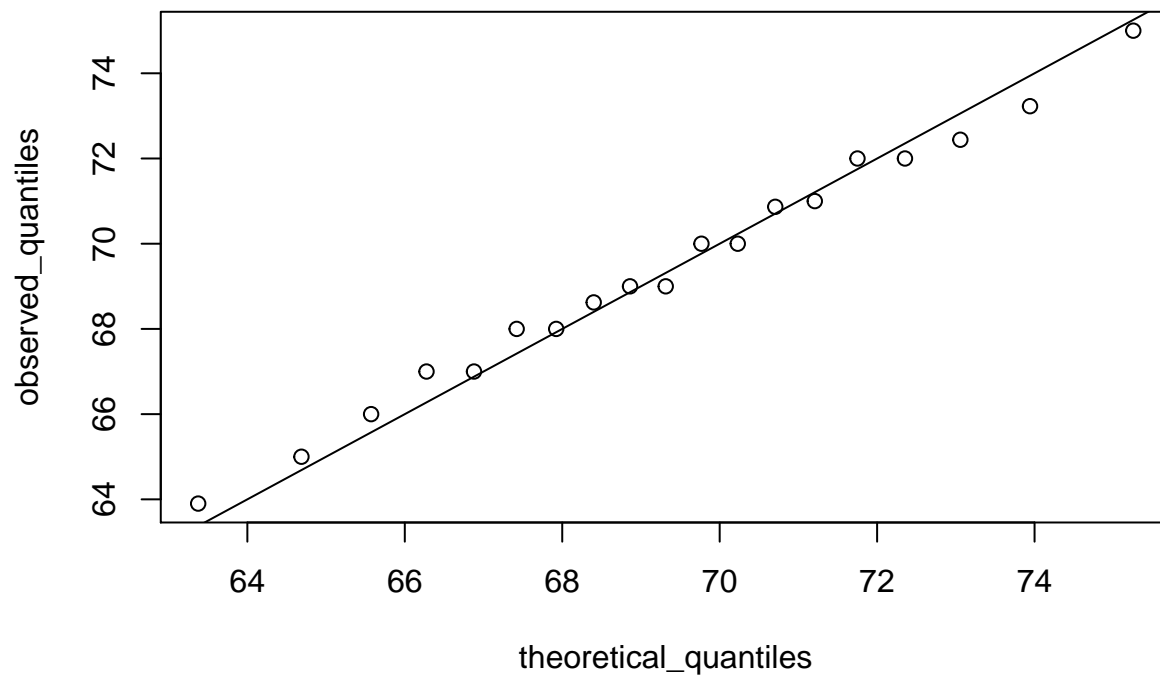
# proportion of data below 69.5
mean(x <= 69.5)
```

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

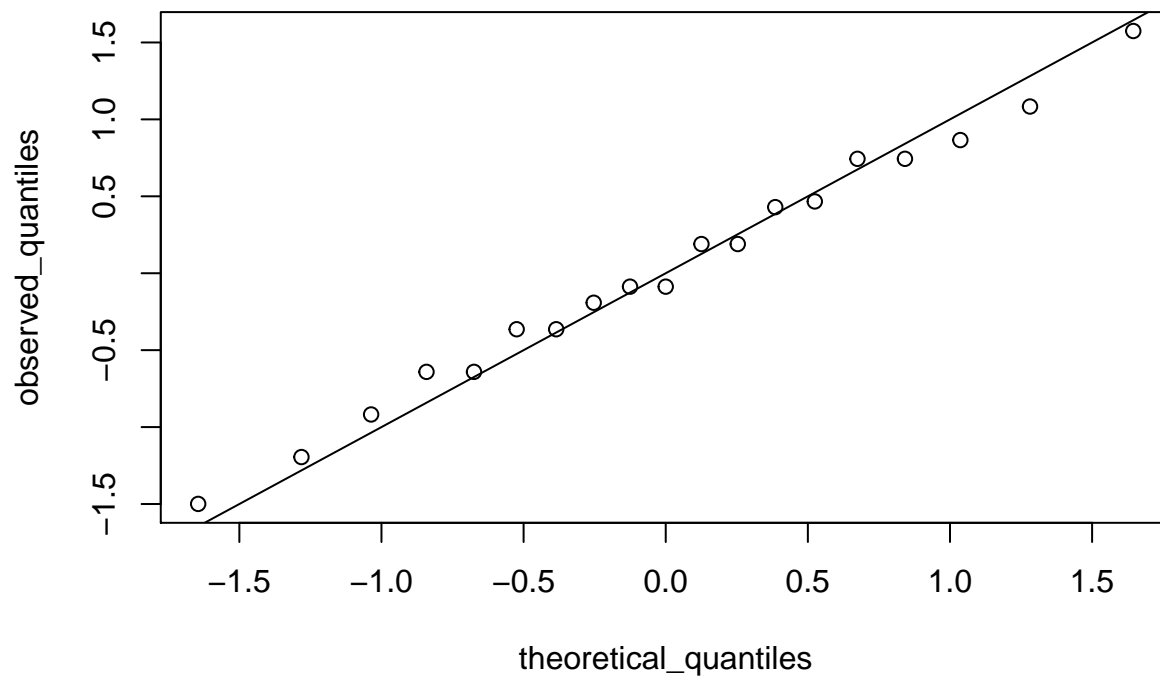
```
# calculate observed and theoretical quantiles
p <- seq(0.05, 0.95, 0.05)
observed_quantiles <- quantile(x, p)
theoretical_quantiles <- qnorm(p, mean = mean(x), sd = sd(x))

# 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  $0.01, 0.02, \dots, 0.99$ . 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 impractical. 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.

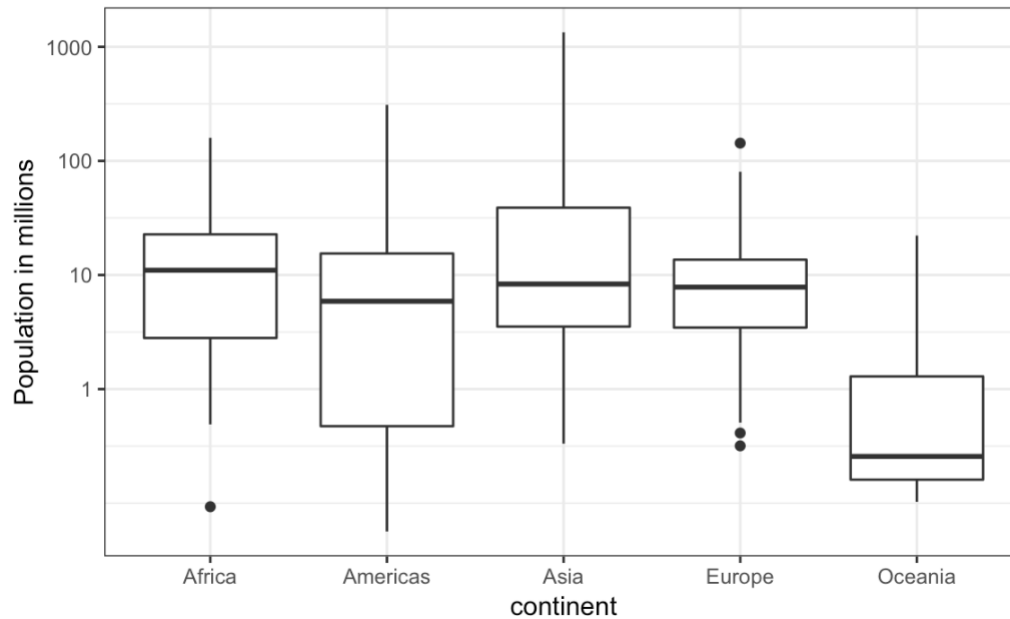


Figure 7: Continent vs Population

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 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
```

```
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

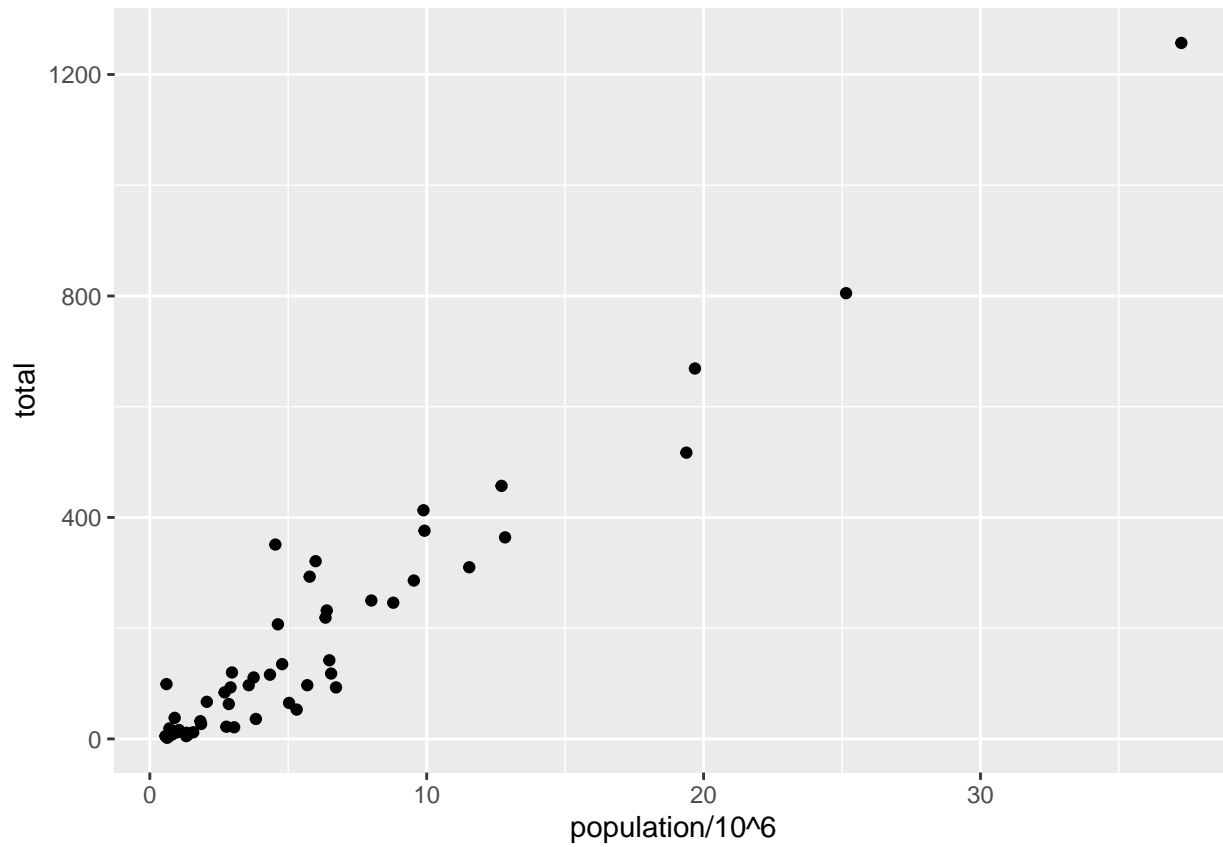
### Key points

- 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.

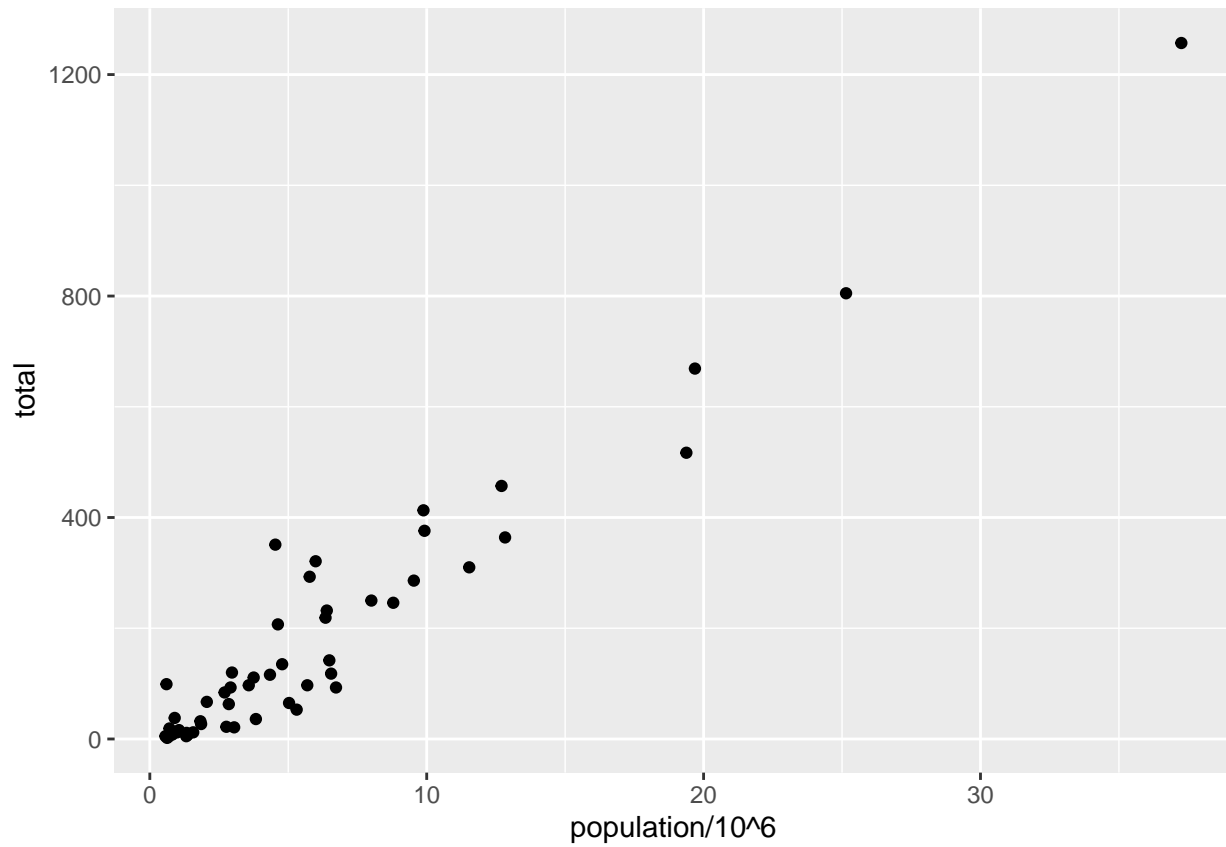


Code: Adding layers to a plot

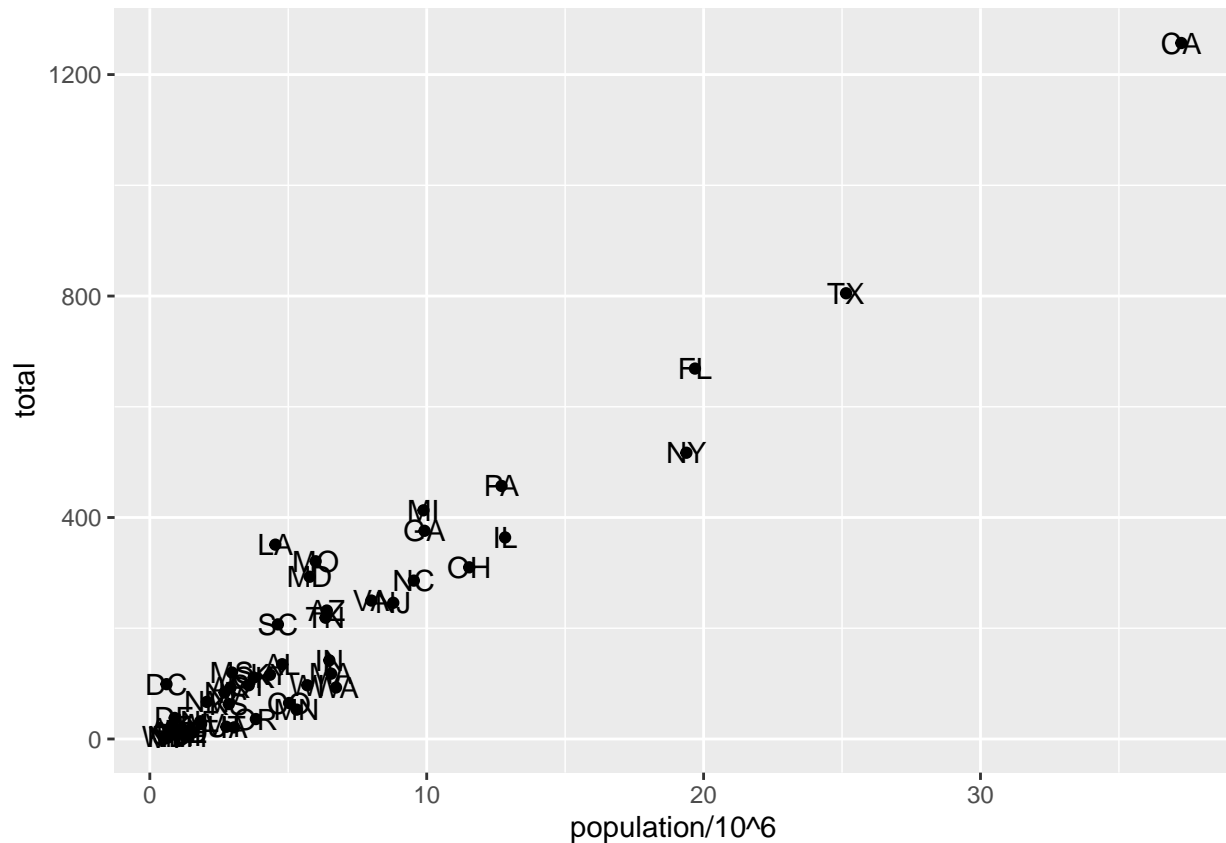
```
murders %>% ggplot() +  
  geom_point(aes(x = population/106, y = total))
```



```
# add points layer to predefined ggplot object  
p <- ggplot(data = murders)  
p + geom_point(aes(population/106, total))
```



```
# 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

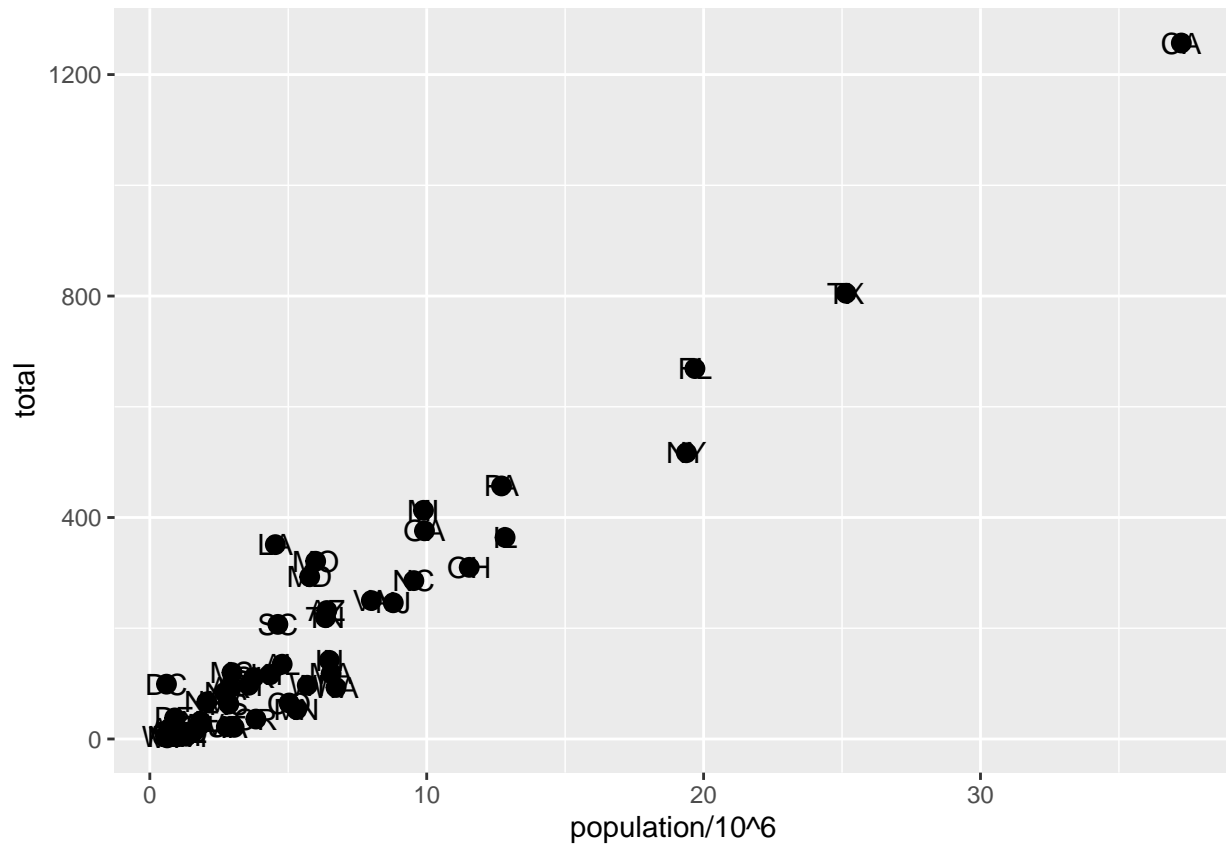
The textbook for this section is available [here](#) and [here](#)

### Key points

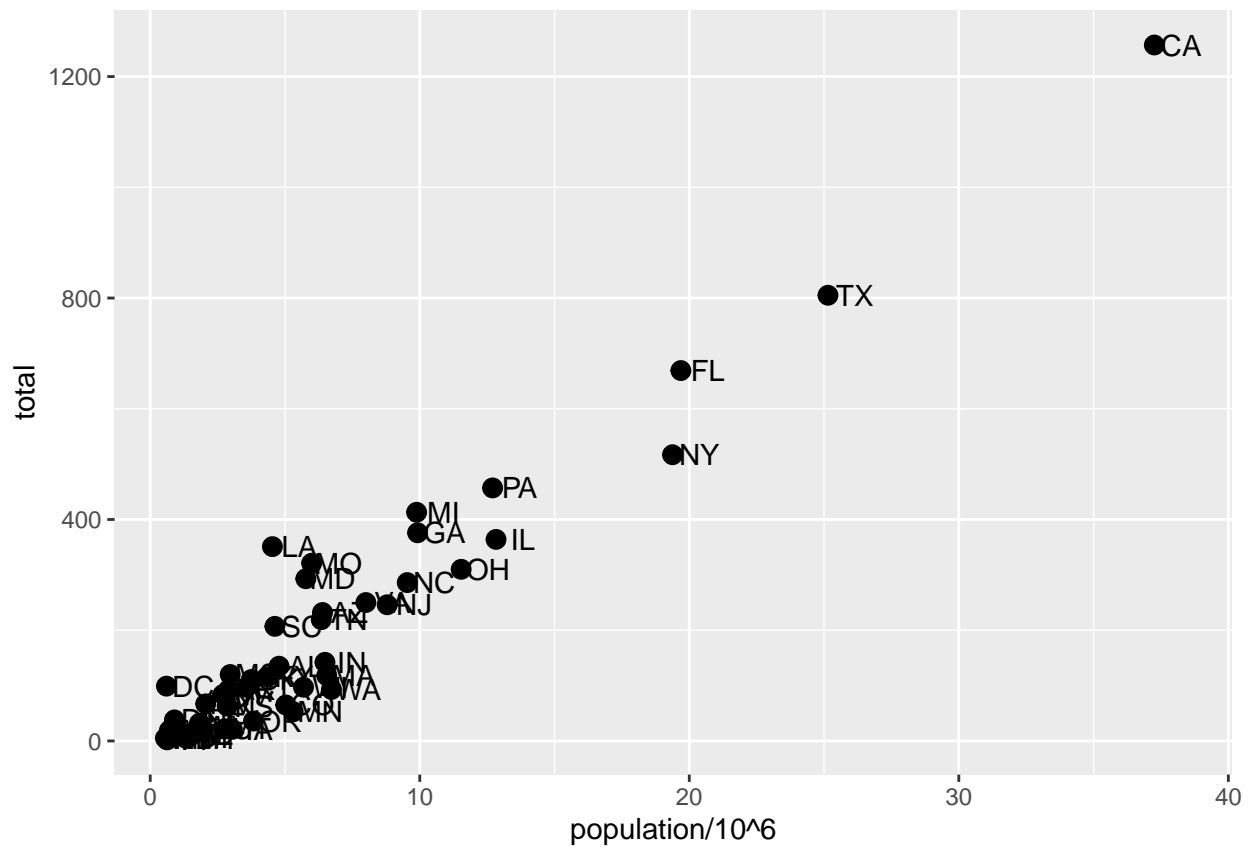
- You can modify arguments to geometry functions other than *aes* and the data. Additional arguments can be found in the documentation for each geometry.
- These arguments are not aesthetic mappings: they affect all data points the same way.
- *Global aesthetic mappings* apply to all geometries and can 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

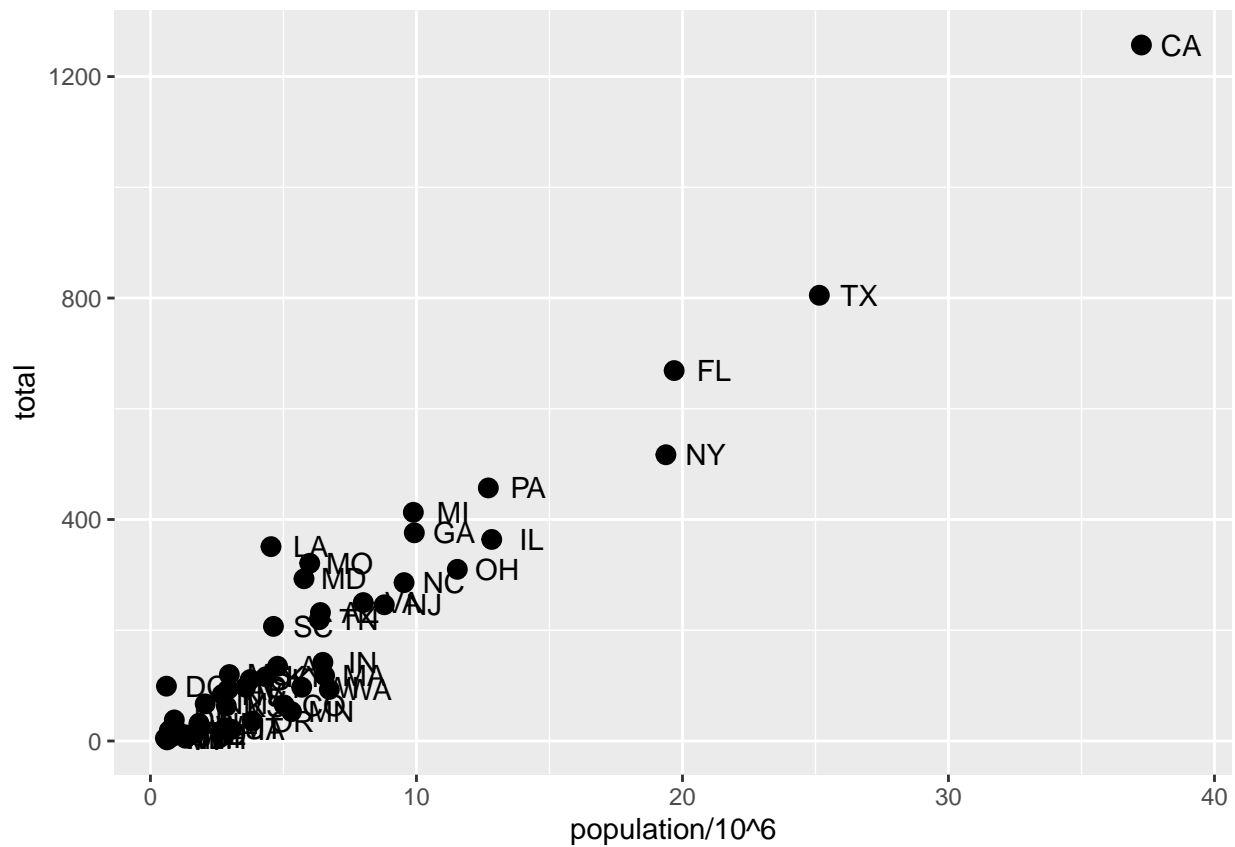
```
# change the size of the points
p + geom_point(aes(population/10^6, total), size = 3) +
  geom_text(aes(population/10^6, total, label = abb))
```



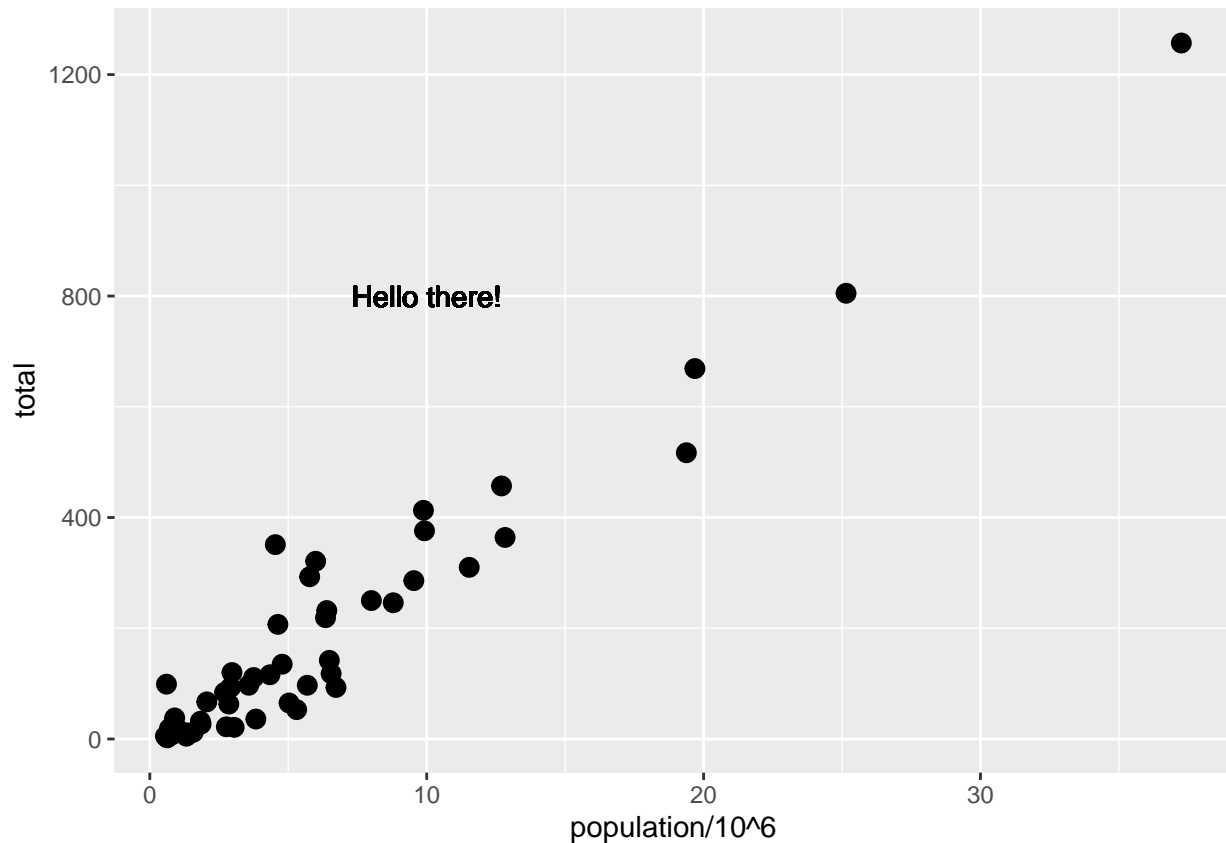
```
# move text labels slightly to the right
p + geom_point(aes(population/10^6, total), size = 3) +
  geom_text(aes(population/10^6, total, label = abb), nudge_x = 1)
```



```
# simplify code by adding global aesthetic
p <- murders %>% ggplot(aes(population/106, 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

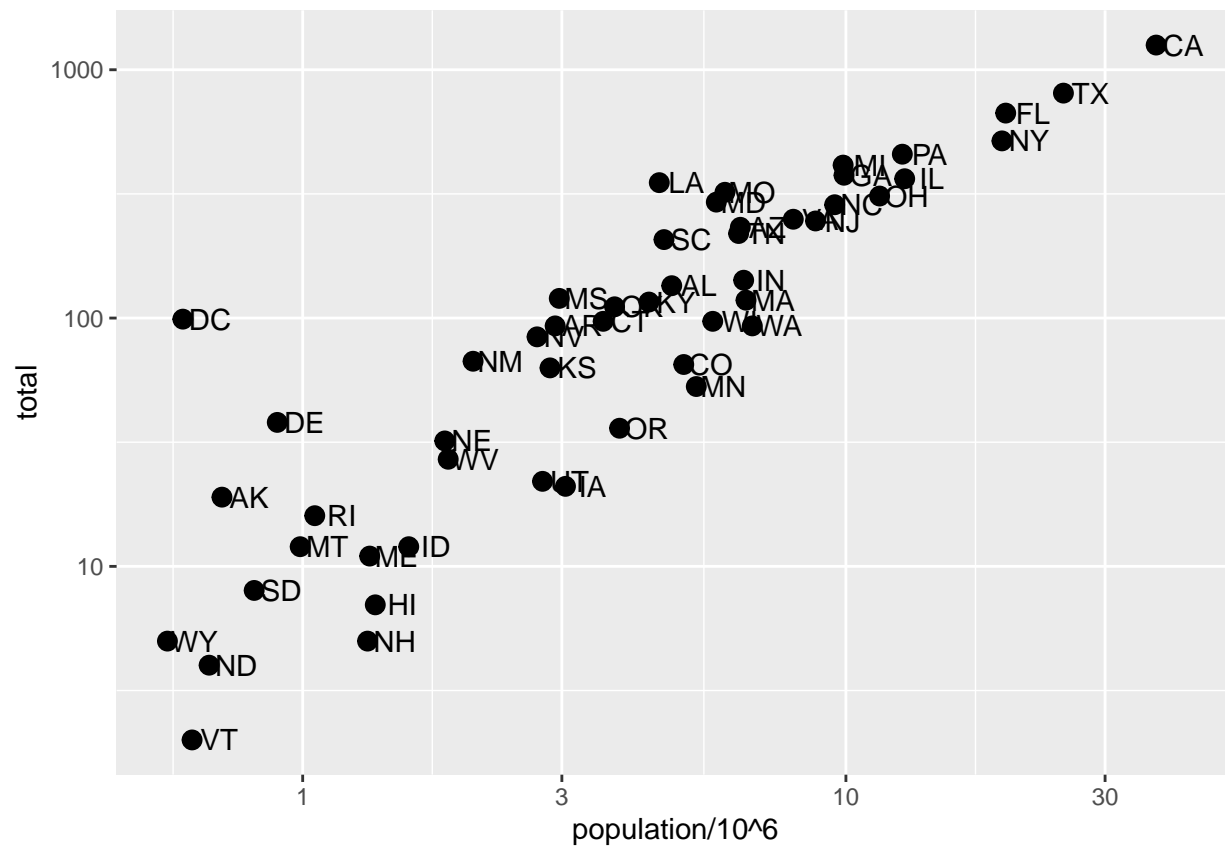
### Key points

- 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*

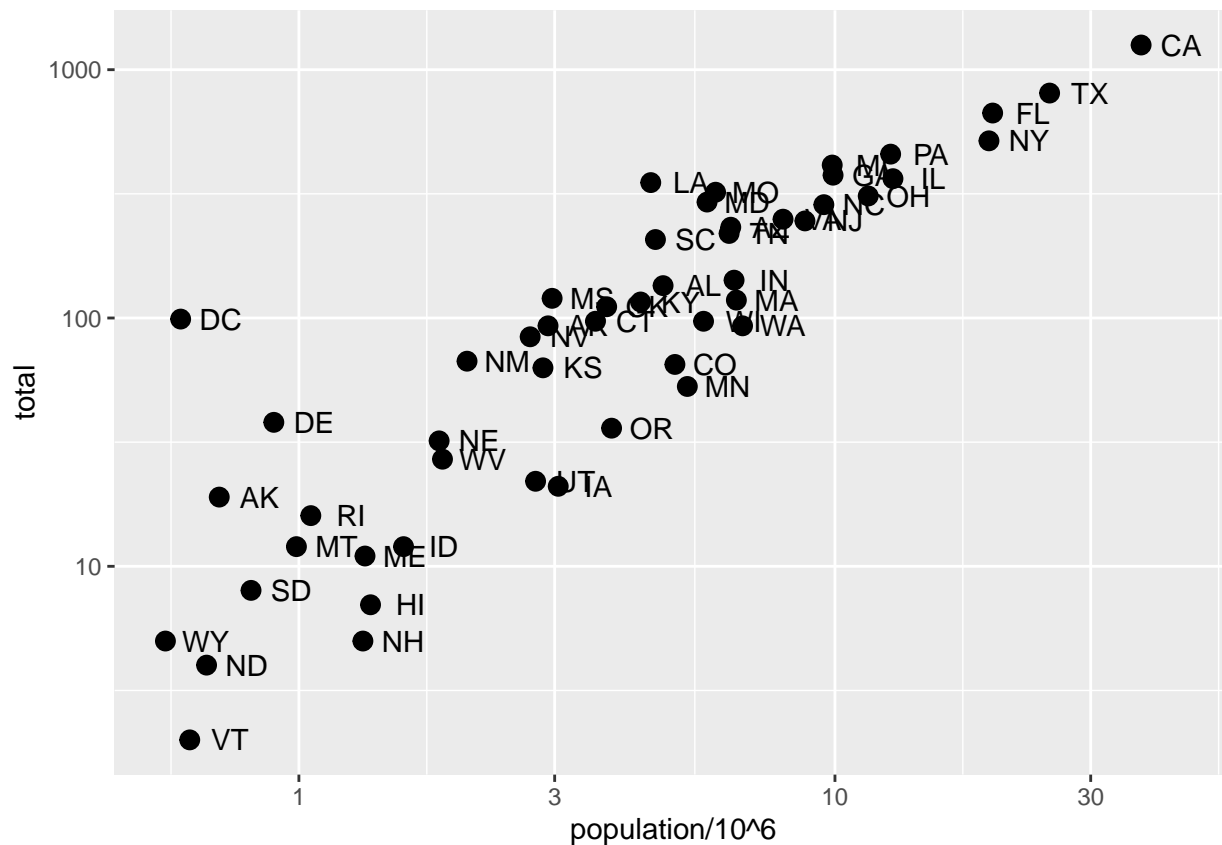
```
# define p
p <- murders %>% ggplot(aes(population/10^6, total, label = abb))

# 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")
```



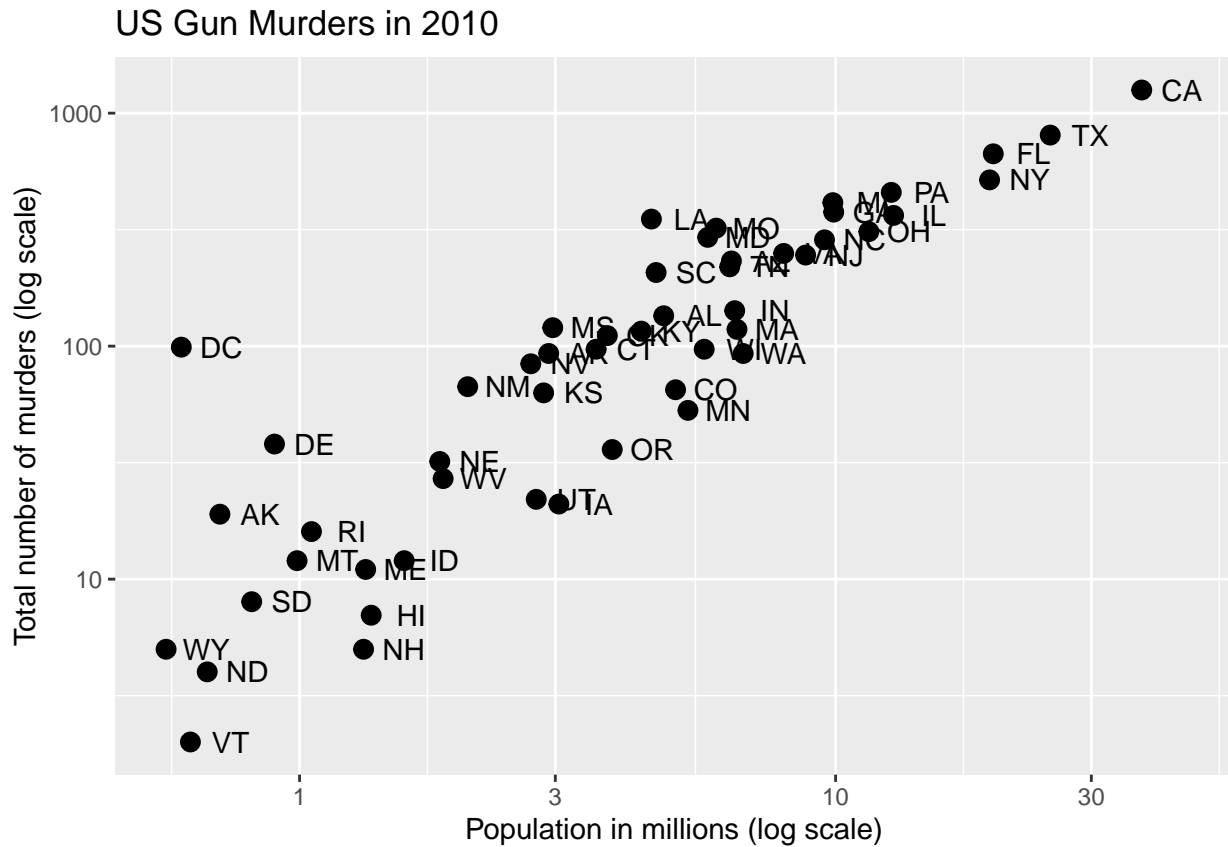
```
# efficient log scaling of the axes
p + geom_point(size = 3) +
  geom_text(nudge_x = 0.075) +
  scale_x_log10() +
  scale_y_log10()
```





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")
```

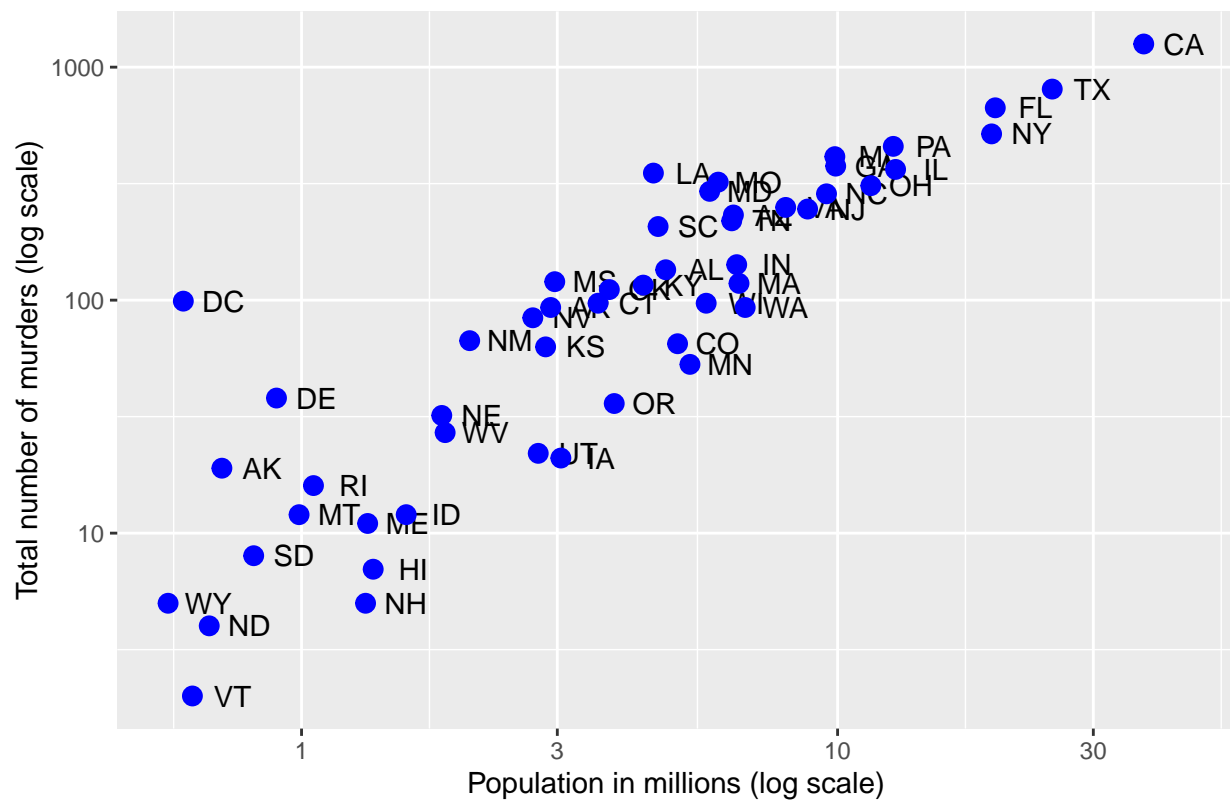


Code: Change color of the points

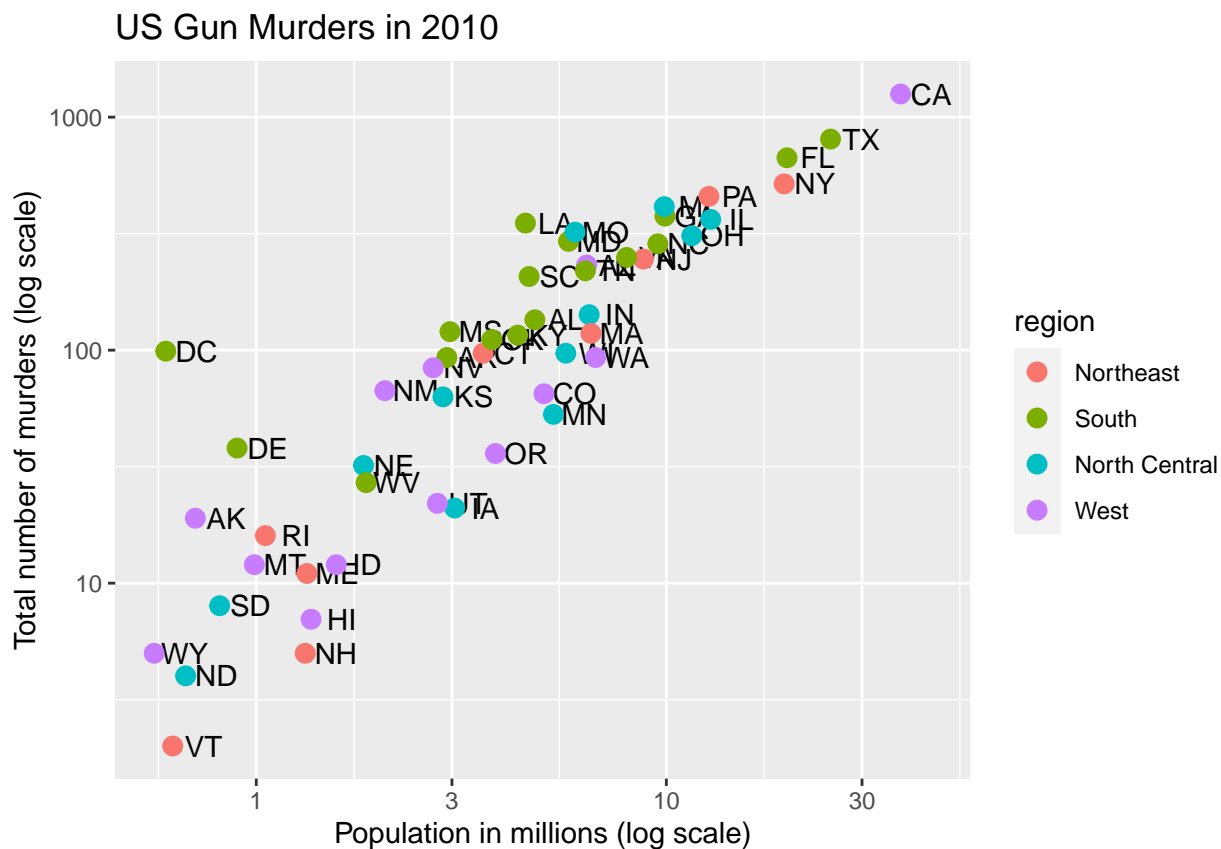
```
# redefine p to be everything except the points layer
p <- murders %>%
  ggplot(aes(population/106, total, label = abb)) +
  geom_text(nudge_x = 0.075) +
  scale_x_log10() +
  scale_y_log10() +
  xlab("Population in millions (log scale)") +
  ylab("Total number of murders (log scale)") +
  ggtitle("US Gun Murders in 2010")

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

US Gun Murders in 2010



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



Code: Add a line with average murder rate

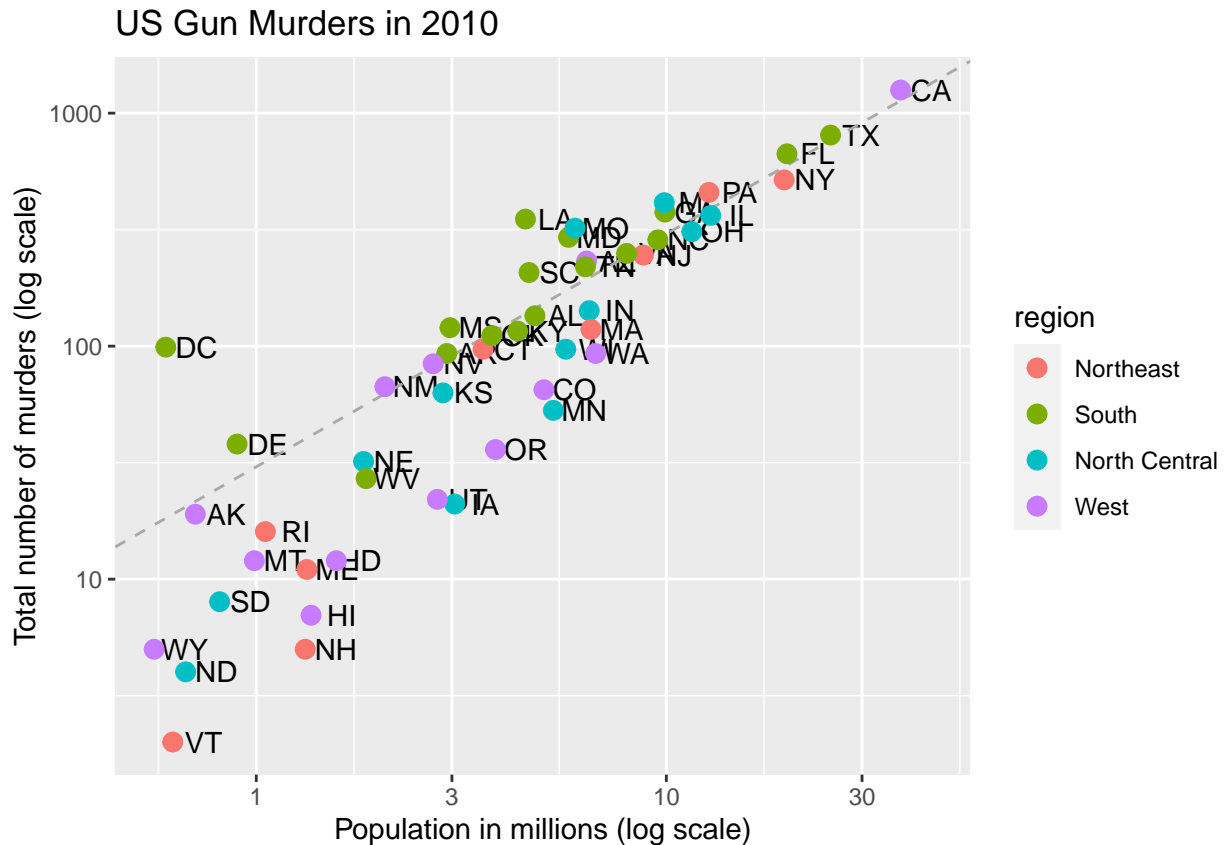
```
# define average murder rate
r <- murders %>%
  summarize(rate = sum(total) / sum(population) * 10^6) %>%
  pull(rate)

# 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
```

Scatter plot showing the relationship between the number of days in the year with a temperature above 50°F (X-axis) and the number of days with a temperature below 32°F (Y-axis) for various US states. The data points are colored by region: West (purple), Midwest (green), Northeast (red), and South (cyan). A positive linear trend line is shown.

State	Days Above 50°F (X)	Days Below 32°F (Y)	Region
CA	280	10	West
TX	250	20	Midwest
FL	240	30	Midwest
NY	230	40	Northeast
PA	220	50	Northeast
IL	210	60	Midwest
OH	200	70	Midwest
IN	190	80	Midwest
MA	180	90	Northeast
VA	170	100	South
WA	160	110	West
CO	150	120	West
MN	140	130	Midwest
OR	130	140	West
UT	120	150	West
NE	110	160	Midwest
WV	100	170	Midwest
AK	90	180	West
RI	80	190	Northeast
MT	70	200	West
ME	60	210	Northeast
SD	50	220	Midwest
HI	40	230	Midwest
NH	30	240	Northeast
VT	20	250	Northeast
WY	10	260	West
ND	0	270	Midwest
DC	20	280	Midwest
DE	50	290	Midwest
LA	100	300	Midwest
MO	110	310	Midwest
MS	120	320	Midwest
AL	130	330	Midwest
SC	140	340	Midwest
GA	150	350	Midwest
NC	160	360	Midwest
MD	170	370	Midwest
CT	180	380	Midwest
NY	190	390	Northeast
PA	200	400	Northeast
OH	210	410	Midwest
IN	220	420	Midwest
MA	230	430	Northeast
VA	240	440	South
WA	250	450	West
CO	260	460	West
MN	270	470	Midwest
OR	280	480	West
UT	290	490	West
NE	300	500	Midwest
WV	310	510	Midwest
AK	320	520	West
RI	330	530	Northeast
MT	340	540	West
ME	350	550	Northeast
SD	360	560	Midwest
HI	370	570	Midwest
NH	380	580	Northeast
VT	390	590	Northeast
WY	400	600	West
ND	410	610	Midwest
DC	420	620	Midwest
DE	430	630	Midwest
LA	440	640	Midwest
MO	450	650	Midwest
MS	460	660	Midwest
AL	470	670	Midwest
SC	480	680	Midwest
GA	490	690	Midwest
NC	500	700	Midwest
MD	510	710	Midwest
CT	520	720	Midwest
NY	530	730	Northeast
PA	540	740	Northeast
OH	550	750	Midwest
IN	560	760	Midwest
MA	570	770	Northeast
VA	580	780	South
WA	590	790	West
CO	600	800	West
MN	610	810	Midwest
OR	620	820	West
UT	630	830	West
NE	640	840	Midwest
WV	650	850	Midwest
AK	660	860	West
RI	670	870	Northeast
MT	680	880	West
ME	690	890	Northeast
SD	700	900	Midwest
HI	710	910	Midwest
NH	720	920	Northeast
VT	730	930	Northeast
WY	740	940	West
ND	750	950	Midwest
DC	760	960	Midwest
DE	770	970	Midwest
LA	780	980	Midwest
MO	790	990	Midwest
MS	800	1000	Midwest
AL	810	1010	Midwest
SC	820	1020	Midwest
GA	830	1030	Midwest
NC	840	1040	Midwest
MD	850	1050	Midwest
CT	860	1060	Midwest
NY	870	1070	Northeast
PA	880	1080	Northeast
OH	890	1090	Midwest
IN	900	1100	Midwest
MA	910	1110	Northeast
VA	920	1120	South
WA	930	1130	West
CO	940		

```
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

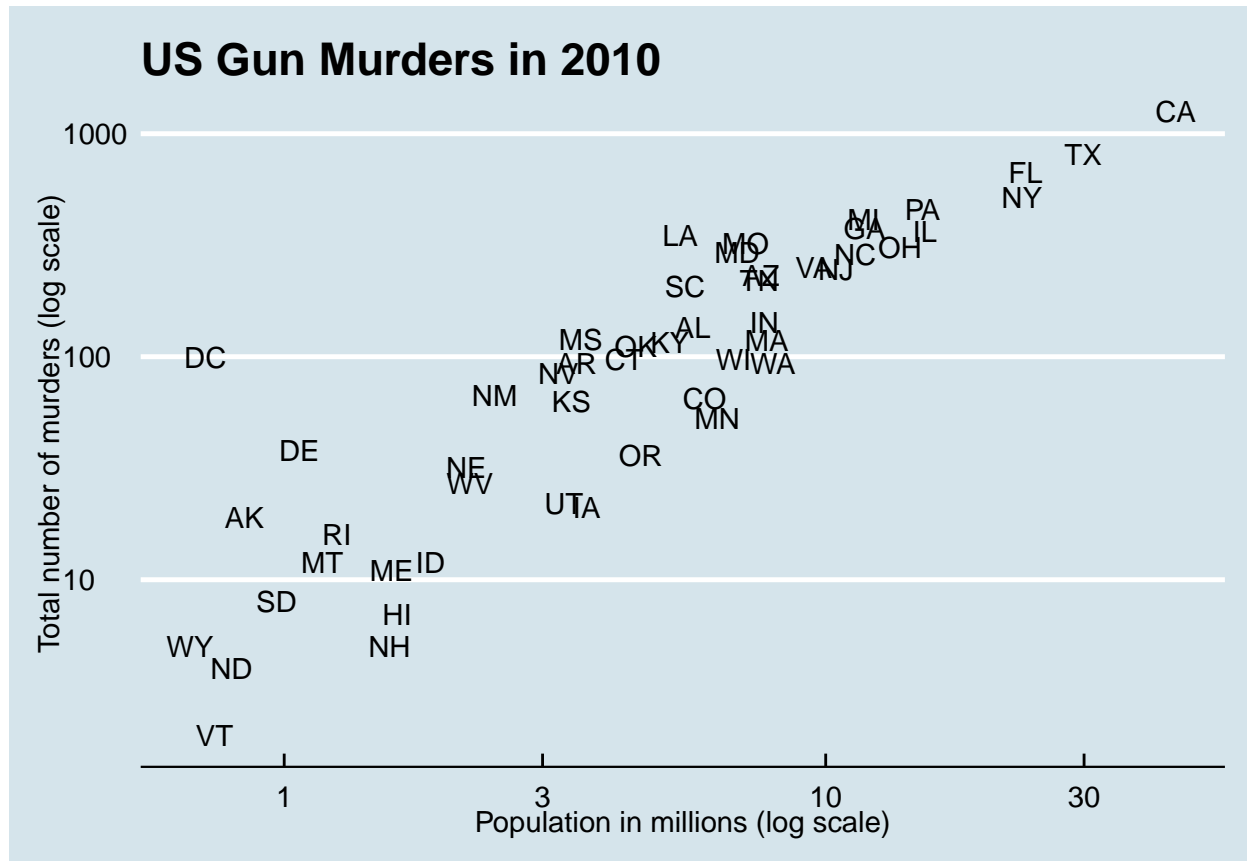
```
if(!require(ggthemes)) install.packages("ggthemes")
```

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

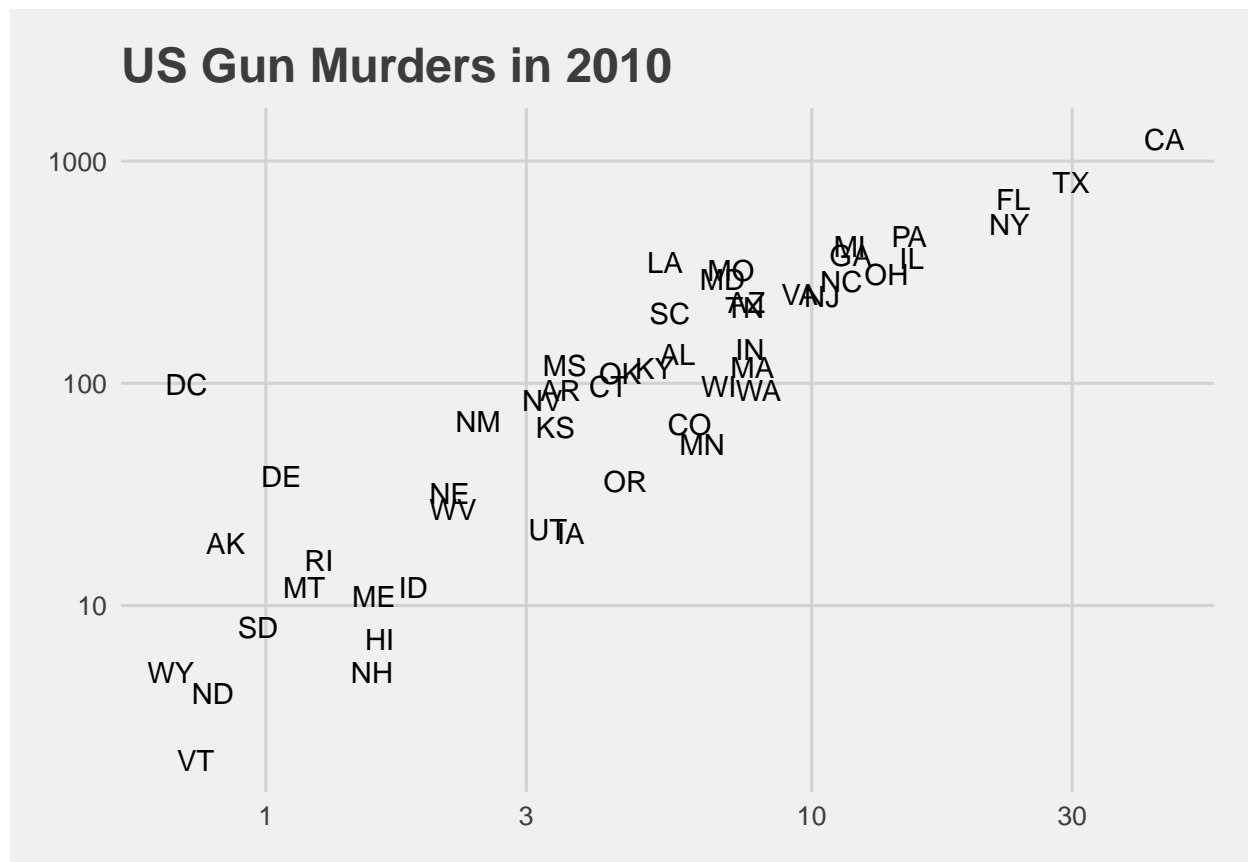
```
## Warning: package 'ggthemes' was built under R version 4.0.2
```

```
# 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
```



```
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(ggplot2)
```

```
# 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() +
```

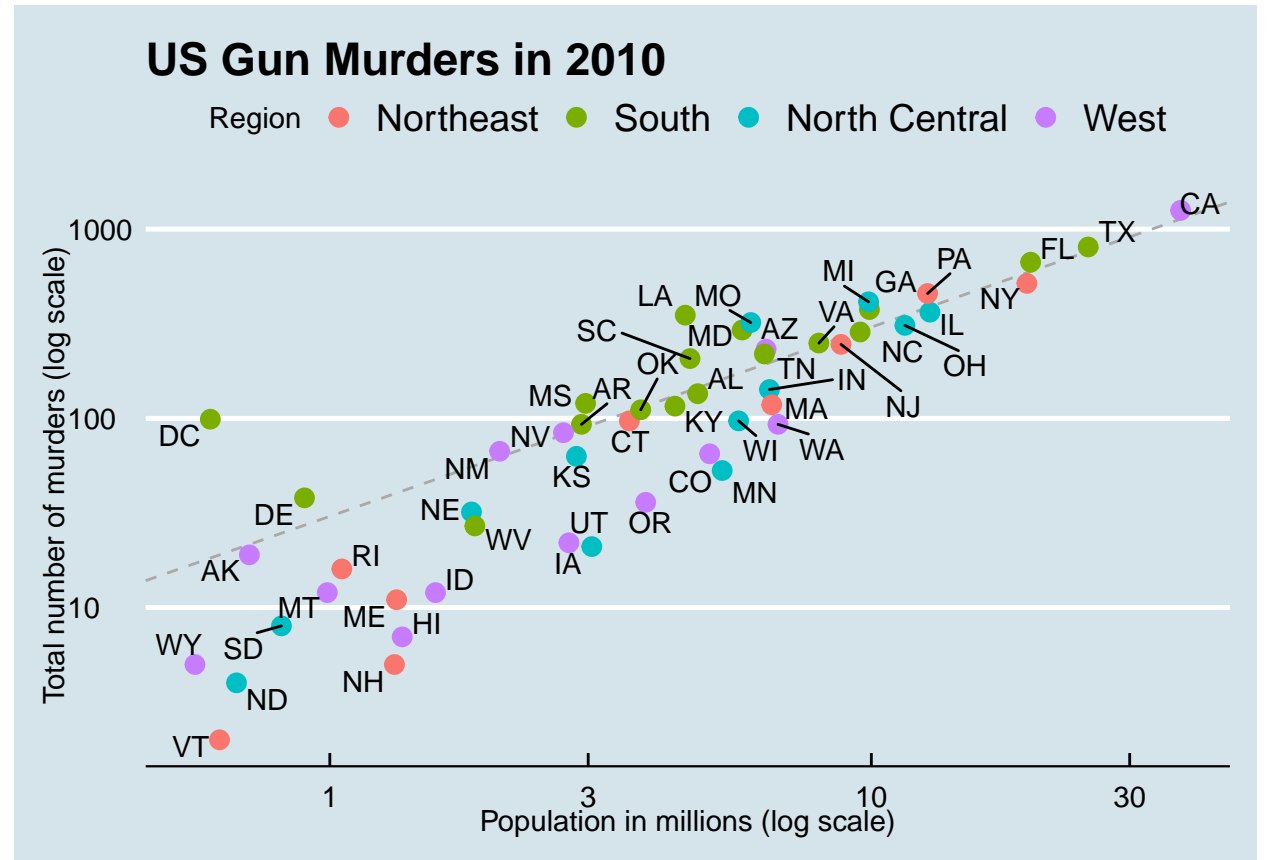
```
  scale_x_log10() +
```

```
  scale_y_log10() +
```

```
  xlab("Population in millions (log scale)") +
```



```
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.

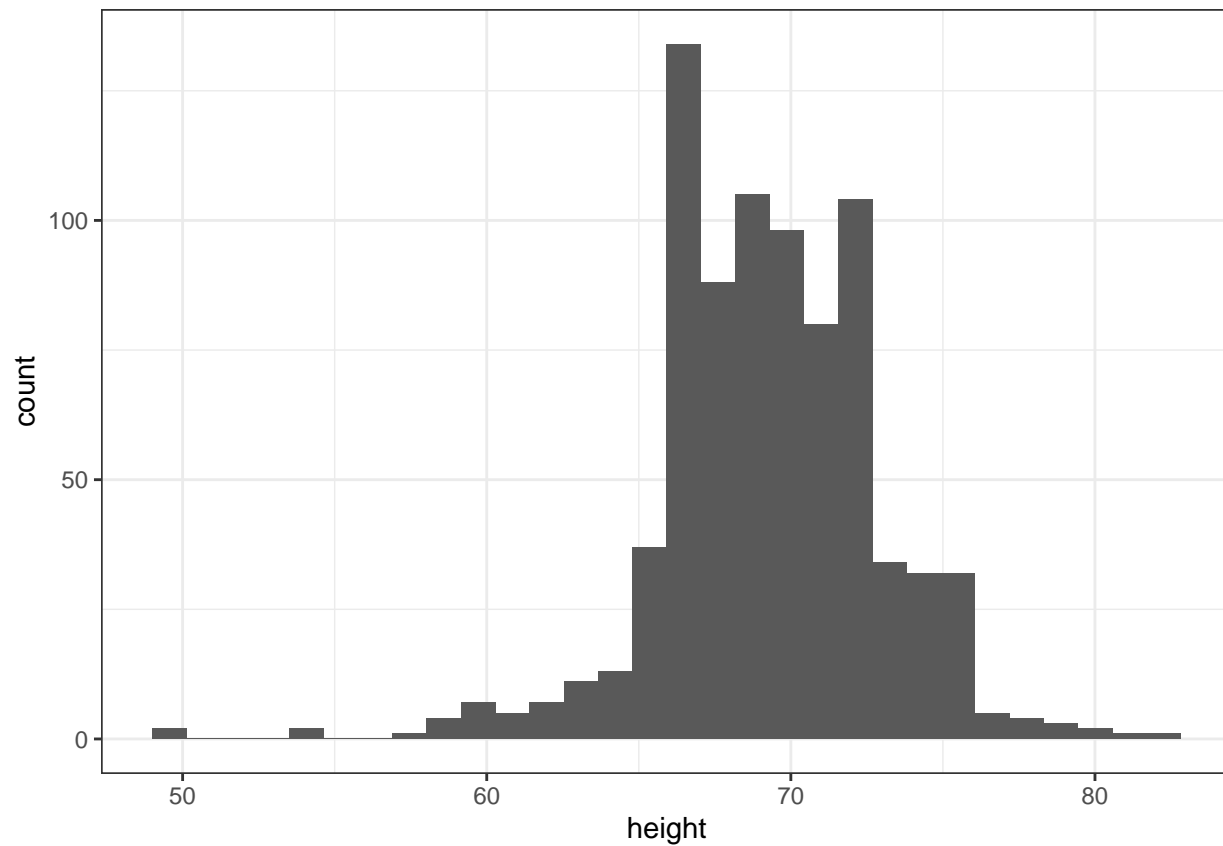
- Plots can be arranged adjacent to each other using the `grid.arrange` function from the `gridExtra` package. First, create the plots and save them to objects (`p1`, `p2`, ...). Then pass the plot objects to `grid.arrange`.

Code: Histograms in ggplot2

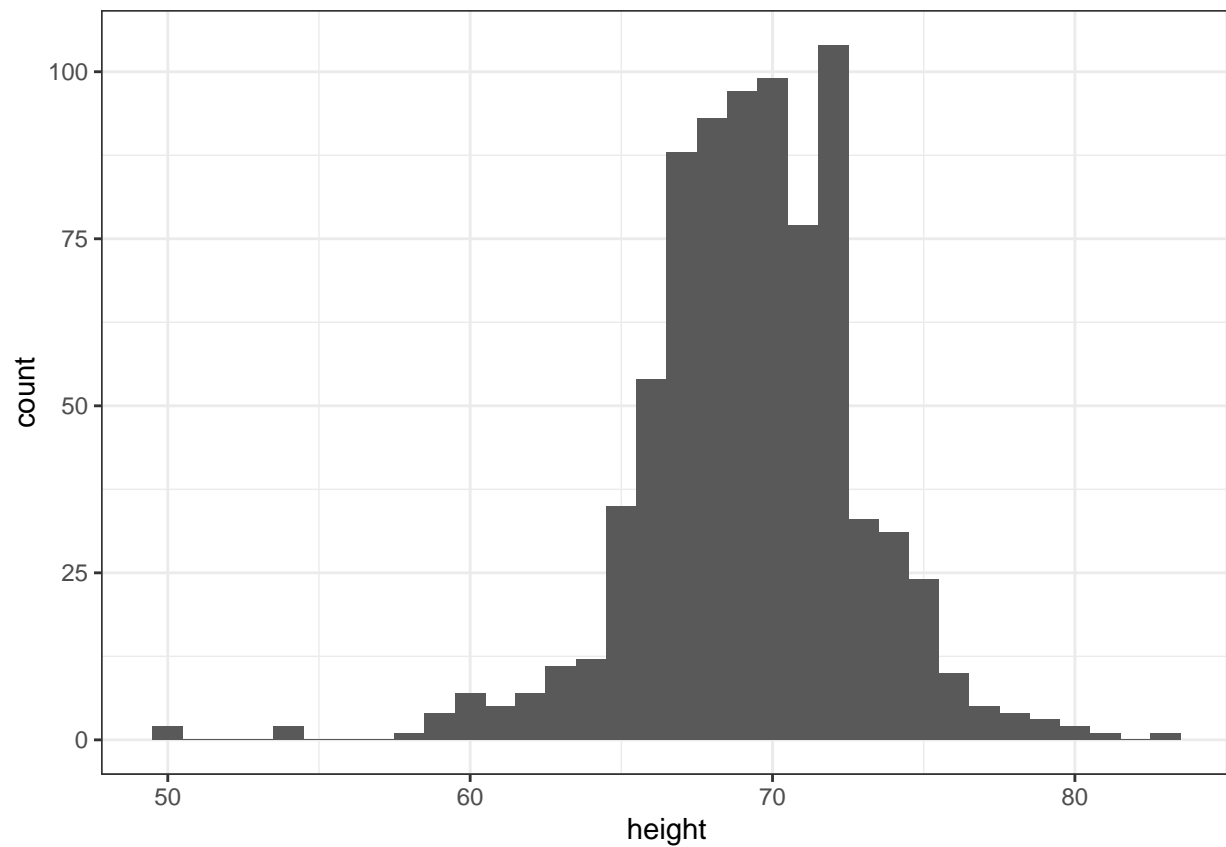
```
# define p
p <- heights %>%
  filter(sex == "Male") %>%
  ggplot(aes(x = height))

# basic histograms
p + geom_histogram()
```

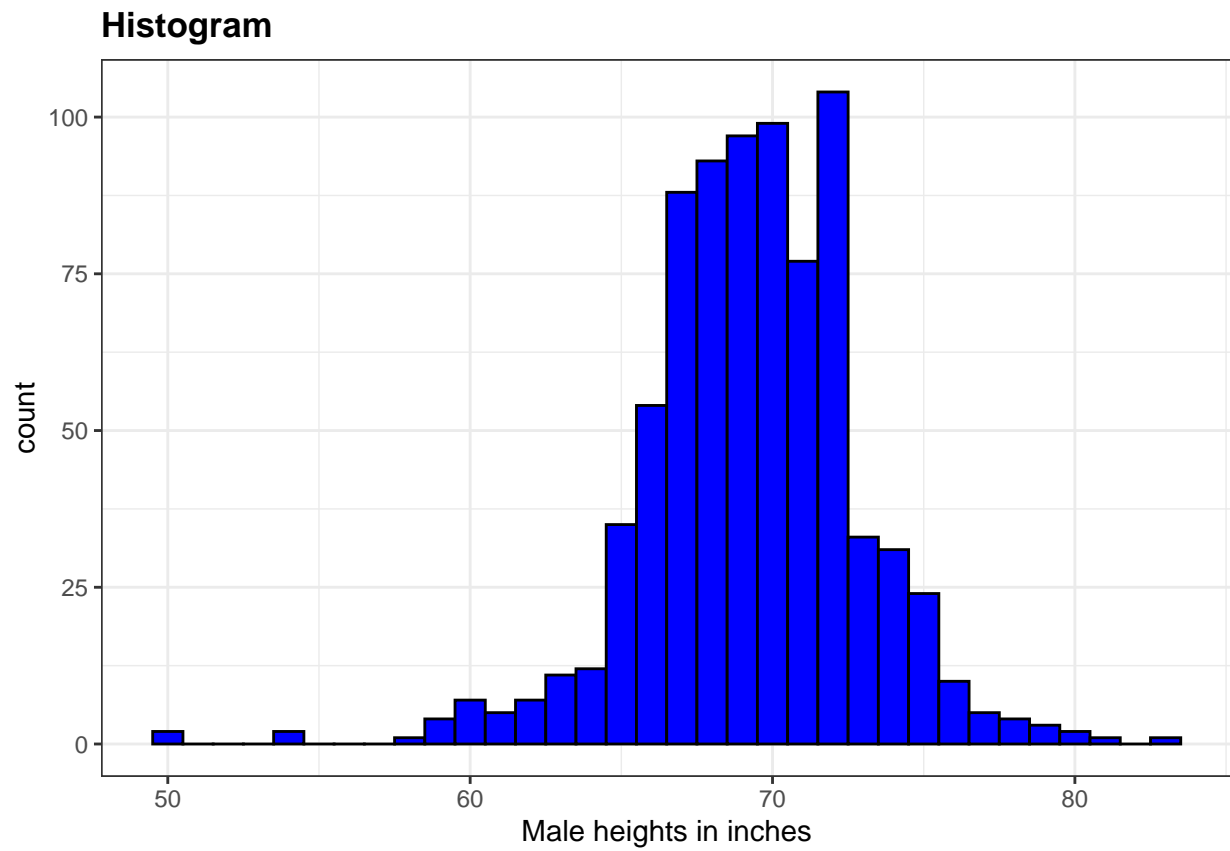
## ``stat_bin()`` using ``bins = 30``. Pick better value with ``binwidth``.



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

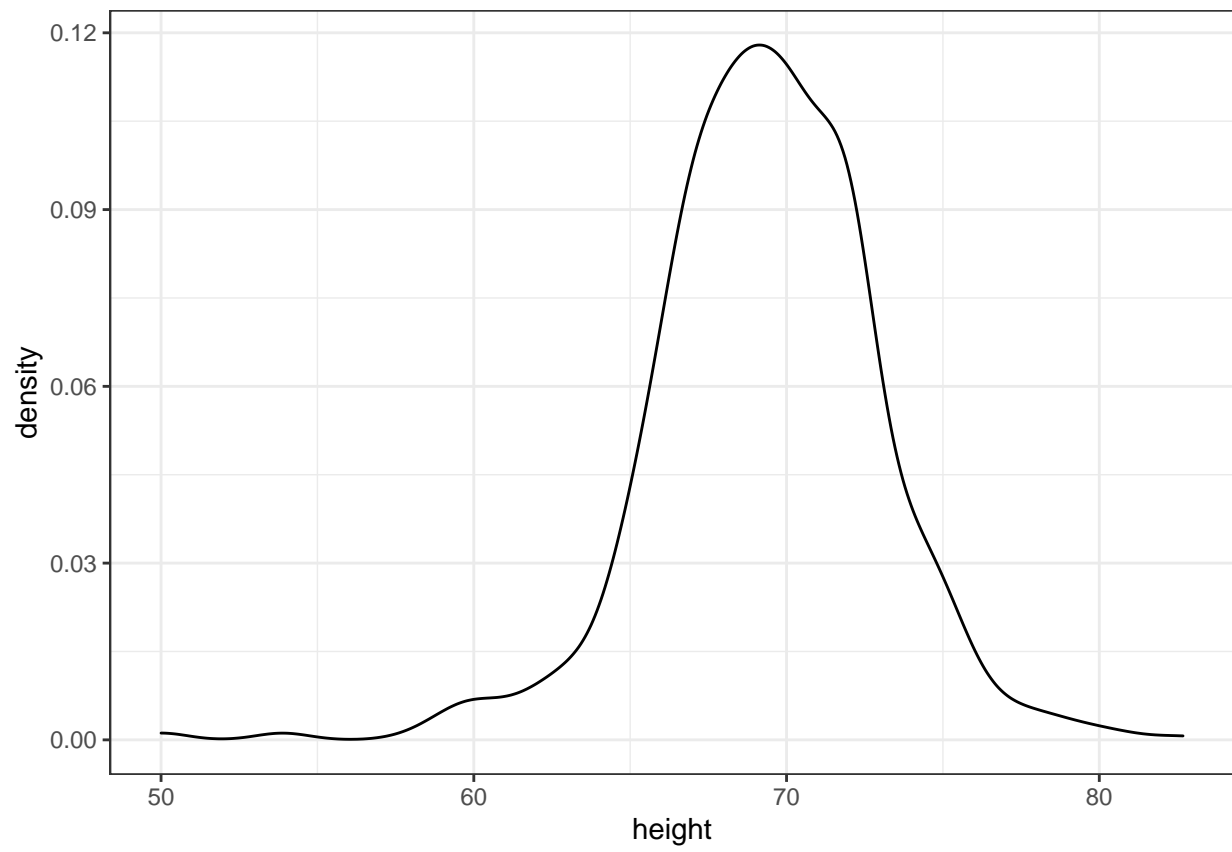


```
# histogram with blue fill, black outline, labels and title
p + geom_histogram(binwidth = 1, fill = "blue", col = "black") +
  xlab("Male heights in inches") +
  ggtitle("Histogram")
```

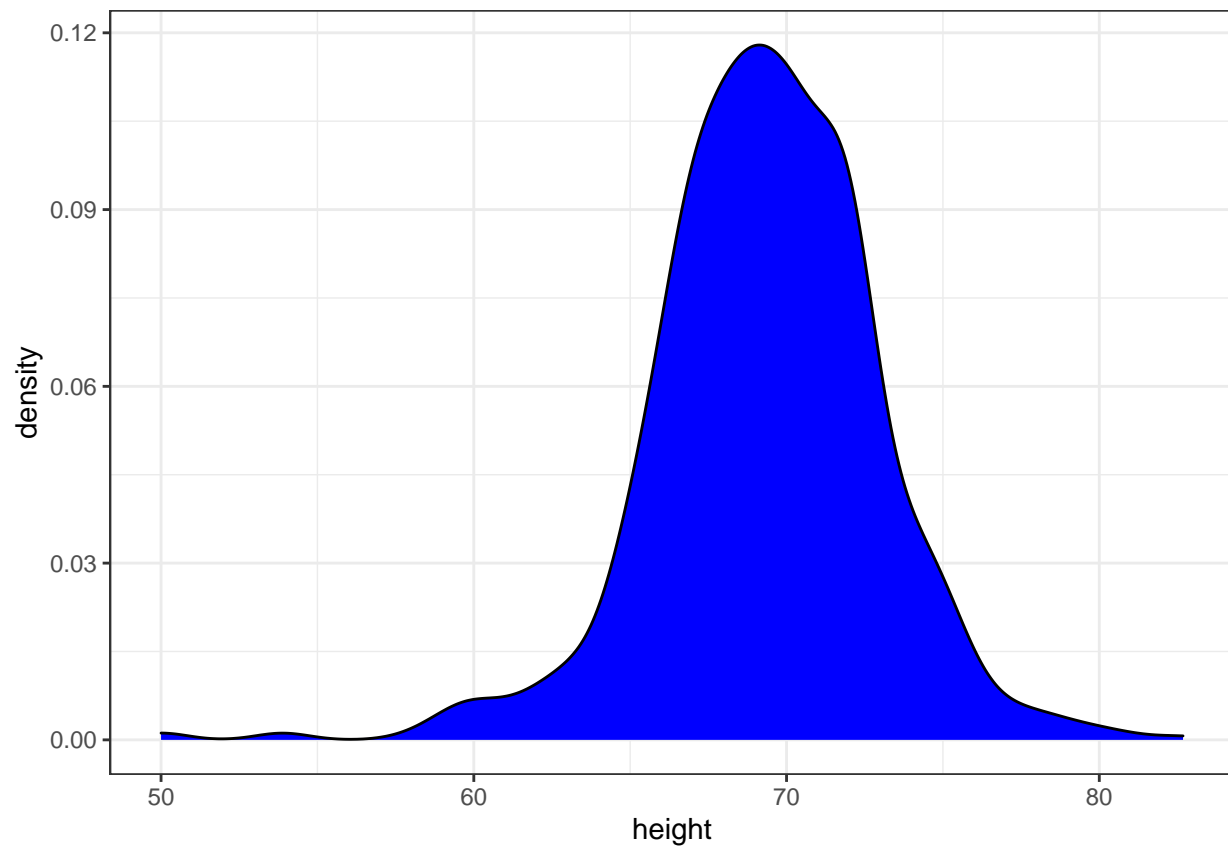


Code: Smooth density plots in ggplot2

```
p + geom_density()
```

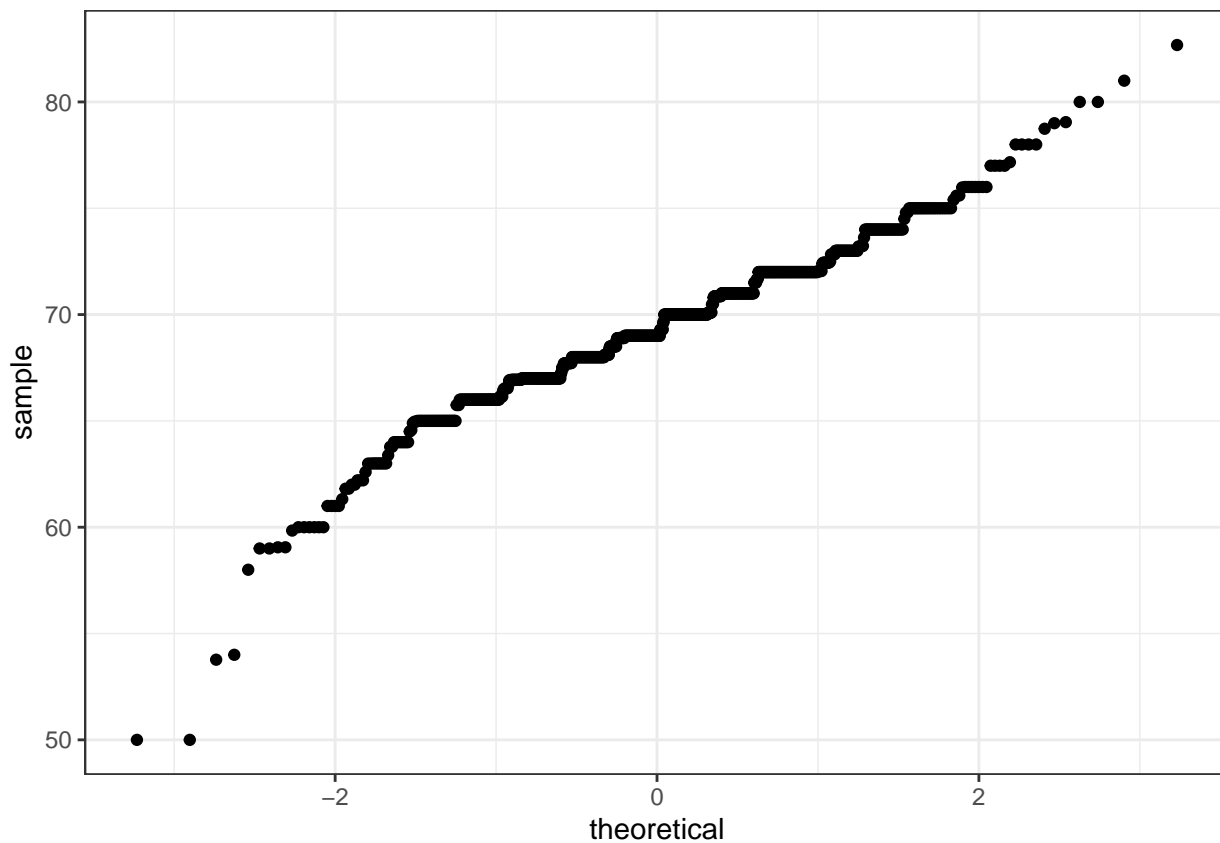


```
p + geom_density(fill = "blue")
```

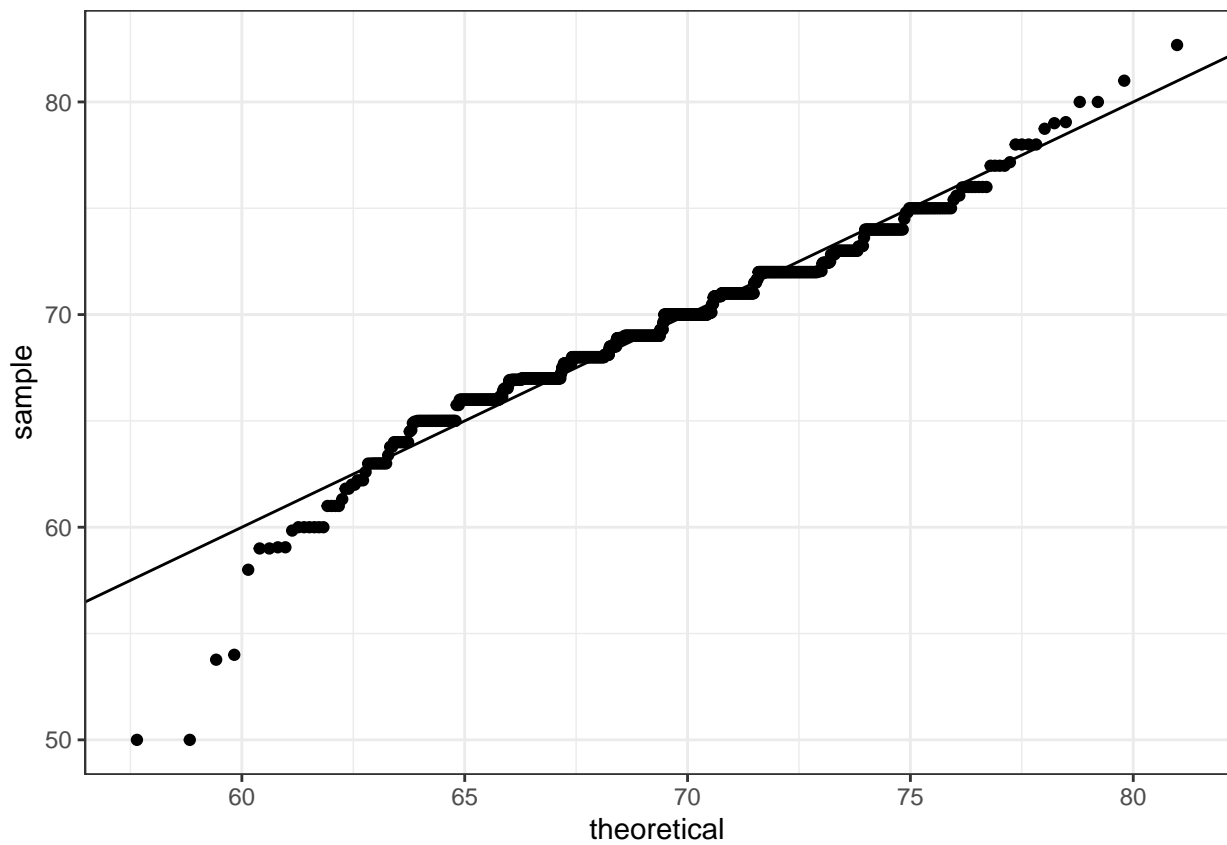


*Code: Quantile-quantile plots in ggplot2*

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

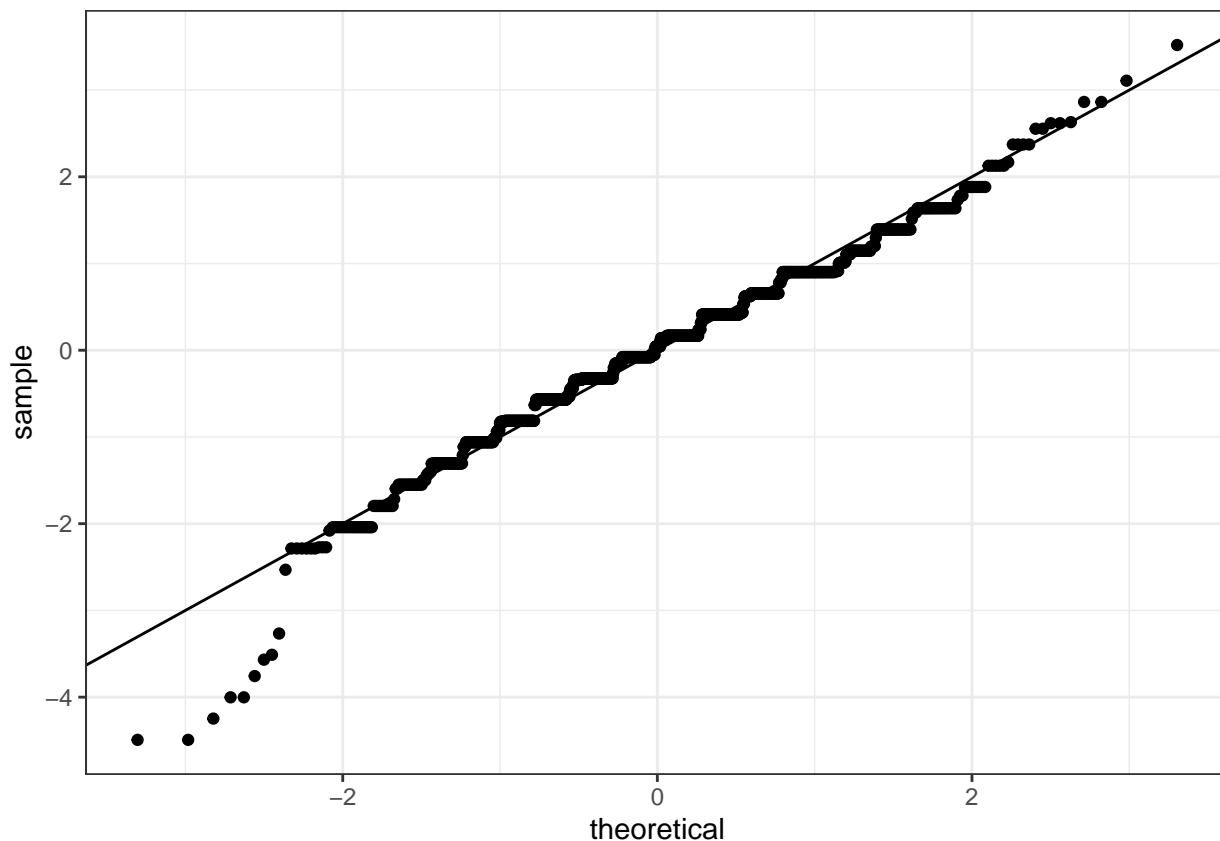


```
# QQ-plot against a normal distribution with same mean/sd as data
params <- heights %>%
  filter(sex == "Male") %>%
  summarize(mean = mean(height), sd = sd(height))
p + geom_qq(dparams = params) +
  geom_abline()
```



```
# 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")
```

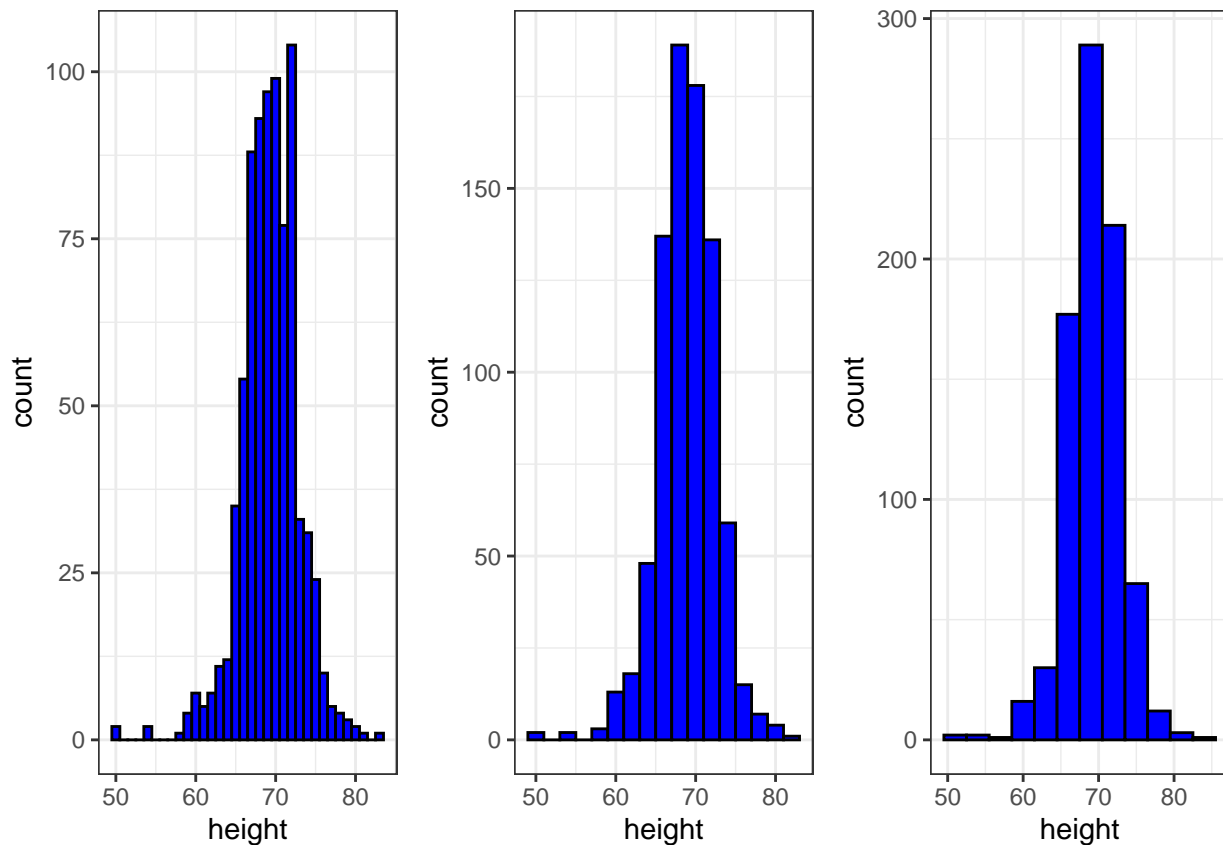
```
p2 <- p + geom_histogram(binwidth = 2, fill = "blue", col = "black")
```

```
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 6 (ggplot2)

Start by loading the dplyr and ggplot2 library as well as the murders and heights data.

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

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

### 1. ggplot2 basics

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

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

or, if we load dplyr, we can also 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 %>%. What is class of the object p?

```
class(p)
```

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

## 2. Printing

Remember that to print an object you can use the command print or simply type the object. For example

```
x <- 2
x
```

```
## [1] 2
```

```
print(x)
```

```
## [1] 2
```

Print the object p defined in exercise one and describe what you see. - [ ] A. Nothing happens. - [X] B. A blank slate plot. - [ ] C. A scatter plot. - [ ] D. A histogram.

## 3. Pipes

Using the pipe %>%, create an object p but this time associated with the heights dataset instead of the murders dataset.

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

```
# What is the class of the object p you have just created?
class(p)
```

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

## 4. Layers

Now we are going to add a layers and the corresponding aesthetic mappings. For the murders data we plotted total murders versus population sizes. Explore the murders data frame to remind yourself what are the names for these two variables and select the correct answer. Hint: Look at ?murders.

- ☐ A. state and abb.
- ☐ B. total\_murders and population\_size.
- ☒ C. total and population.
- ☐ D. murders and size.