Scatterplots and Barplots

INFO 526 Data Analysis and Visualization

Adriana Picoral

In this tutorial, we will focus on two types of visualizations: bar plots, and scatterplots. Like all the other tutorials in this course, this document is accompany by code-along videos, and the source code (to be found on D2L).

The learning outcomes of this tutorial include:

- Produce effective bar charts and scatterplots
- Identify visualization errors and pitfalls

Scatterplots

For our first scatterplot, we will be using the `openintro` library which accompanies Çetinkaya-Rundel and Hardin's (2021) textbook (first chapter can be found to download on D2L, full book can be accessed at https://openintro-ims.netlify.app/). We will also load `scales` for some plot formatting, and `tidyverse` for the `ggplot` function.

```
library(openintro)
library(scales)
library(tidyverse)
```

Many packages come with data in addition to functions. The package `openintro` contains all datasets used in Introduction to Modern Statistics. We will be working with the `county`data, so let's load that using the `data()` function.

```
# load data from openintro packages
data("county")
```

We will now recreate Figure 1.2 in Introduction to Modern Statistics. You can see the actual code that was used to render this figure in the textbook's GitHub repository. For color names, I use this guide.

For scatterplots, you usually plot two continuous numeric variables. In this case we are plotting 1) the percent of housing units that are multi-unit structures in different counties in the United States between 2006 and 2010, and 2) homeownership percentage for the same counties and time frame. We will use `geom_point` to represent each data point with a point for the x and y values. We use `anotate` and `geom_text` to replicate the original annotation in red (to highlight a specific data point). Finally we do some plot formatting using `theme_`, `labs`, and `scale_`.

```
# recreate Figure 1.2
county %>%
  ggplot(aes(x = multi_unit,
             y = homeownership)) +
  geom_point(color = "royalblue4",
             fill = "dodgerblue3",
             shape = 21,
             alpha = .3) +
  annotate("segment",
           x = 39.4, xend = 39.4,
           y = 0, yend = 31.3,
           color = "red",
           linetype = "dashed") +
  annotate("segment",
           x = 0, xend = 39.4,
           y = 31.3, yend = 31.3,
           color = "red",
           linetype = "dashed") +
  geom_point(x = 39.4, y = 31.3,
             shape = 21, color = "red") +
  geom_text(label = "Chattahoochee County", fontface = "italic",
            x = 55, y = 30, color = "red") +
  theme_minimal() +
  labs(x = "Percent of housing units that are muti-unit structures",
       y = "Homeownership rate") +
  scale_x_continuous(labels = percent_format(scale = 1)) +
  scale_y_continuous(labels = percent_format(scale = 1))
```

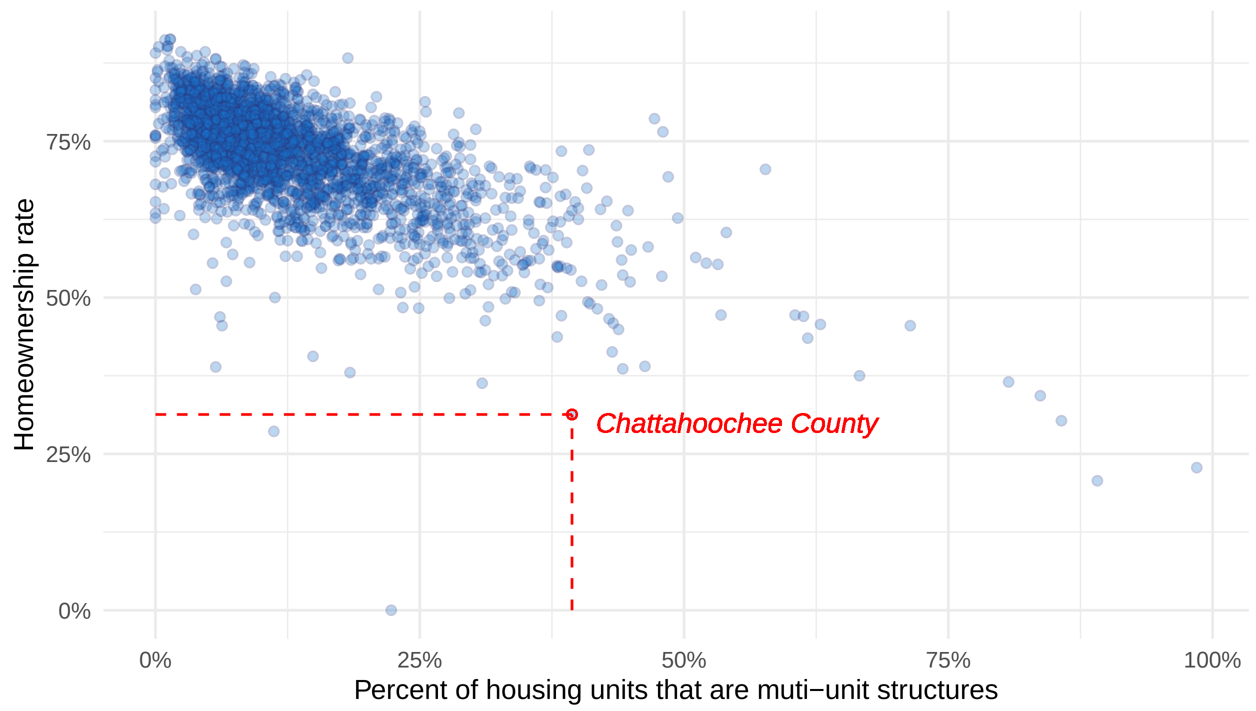


Figure 1: A scatterplot of homeownership versus the percent of housing units that are in multi-unit structures for US counties.

Association does not equal causation

Nicolas Cage's movies

Unintentional Drowning Deaths in the United States, 1999–2010

See more Spurious correlations

```
# read data in
drowning_deaths_nic_cage_movies <- read_csv("data/drowning_deaths_nic_cage_movies.csv")
```

Calculate correlation

```
drowning_deaths_nic_cage_movies %>%
  summarize(correlation = cor(cage_movie_count, deaths))
```

```
## # A tibble: 1 x 1
##   correlation
##         <dbl>
## 1       0.113
```

```
cor.test(drowning_deaths_nic_cage_movies$cage_movie_count,
    drowning_deaths_nic_cage_movies$deaths)
```

```
##
##  Pearson's product-moment correlation
##
## data:  drowning_deaths_nic_cage_movies$cage_movie_count and drowning_deaths_nic_cage_movies$deaths
## t = 0.36023, df = 10, p-value = 0.7262
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4927230  0.6451773
## sample estimates:
##       cor
## 0.1131842
```

Try to recreate the original plot.

```
# plot it
drowning_deaths_nic_cage_movies %>%
  ggplot(aes(x = year)) +
  geom_point(aes(y = cage_movie_count)) +
  geom_point(aes(y = deaths/1600),
             color = "blue") +
  scale_y_continuous(name = "Nicholas Cage movie count",
                     sec.axis = sec_axis(~.*1600, name="deaths by drowning in the US"),
                     limits = c(0, 6)) +
  scale_x_continuous(breaks = c(1999:2010)) +
  theme_linedraw()
```

Bar plots

Barplots are often used to visualize the frequency distribution of a categorical variable. The idea is that we count different categories (i.e., levels) within a categorical variable.

There are geometrics functions within `ggplot` that allow you to draw up bar plots:

- `geom_bar()` counts frequencies by levels within a categorical variable and plot those frequencies. You need to map the categorical variable to one of the axes only (the other axis is unmapped, since it will take the frequency values automatically)
- `geom_col()` requires the two axes to be mapped, one axis is mapped to the categoric variable the other is mapped to the count variable.

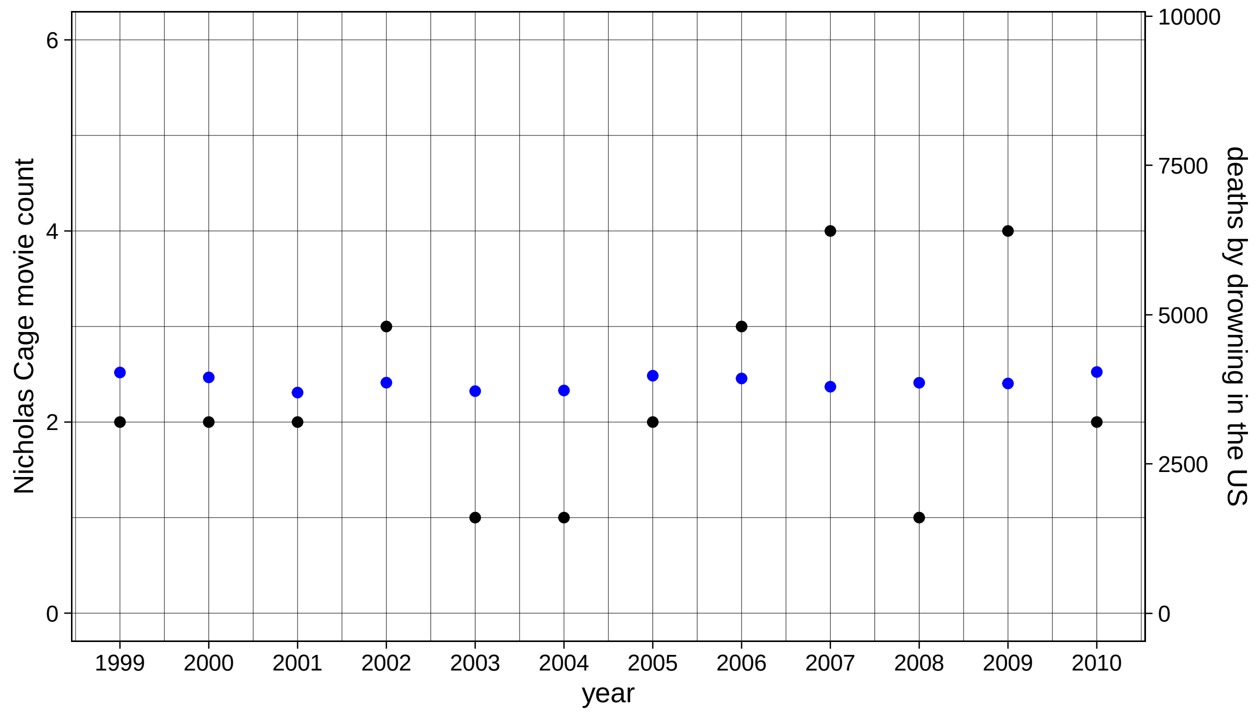


Figure 2: Number of Nicholas Cage movies and number of deaths by drowning in the US.

Using geom_bar()

For this short demonstration, we will be using the `penguins` dataset again. Let's load `tidyverse` and `palmerpenguins`.

```
library(palmerpenguins)
```

As mentioned, you need to map one of the axes to the categorical variable for which you want to visualize the frequency distribution. Let's visualize the frequency of each species in the `penguins` dataset, mapping `species` to `x` first and then adding `geom_col()` to the code block.

```
penguins %>%
  ggplot(aes(x = species)) +
  geom_bar()
```

Note that we can map the categorical variable to `y` instead.

```
penguins %>%
  ggplot(aes(y = species)) +
  geom_bar()
```

Note that the counts calculated in `geom_col()` are the same as if we were to calculate the counts ourselves.

```
penguins %>%
  count(species)
```

```
## # A tibble: 3 x 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
```

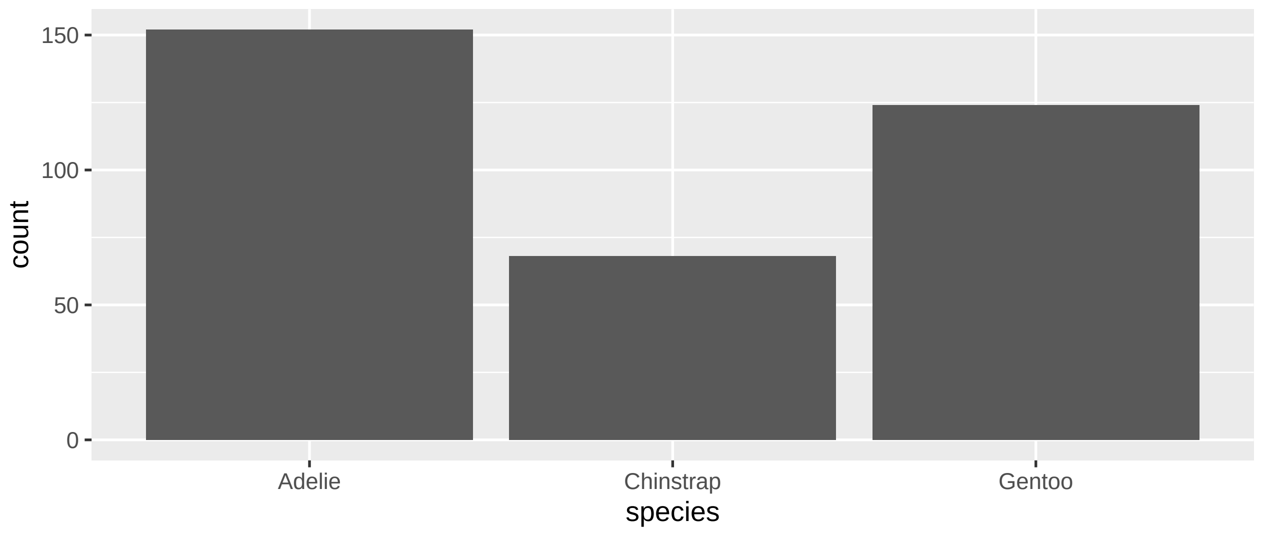


Figure 3: Straight-forward plot of penguin specie count.

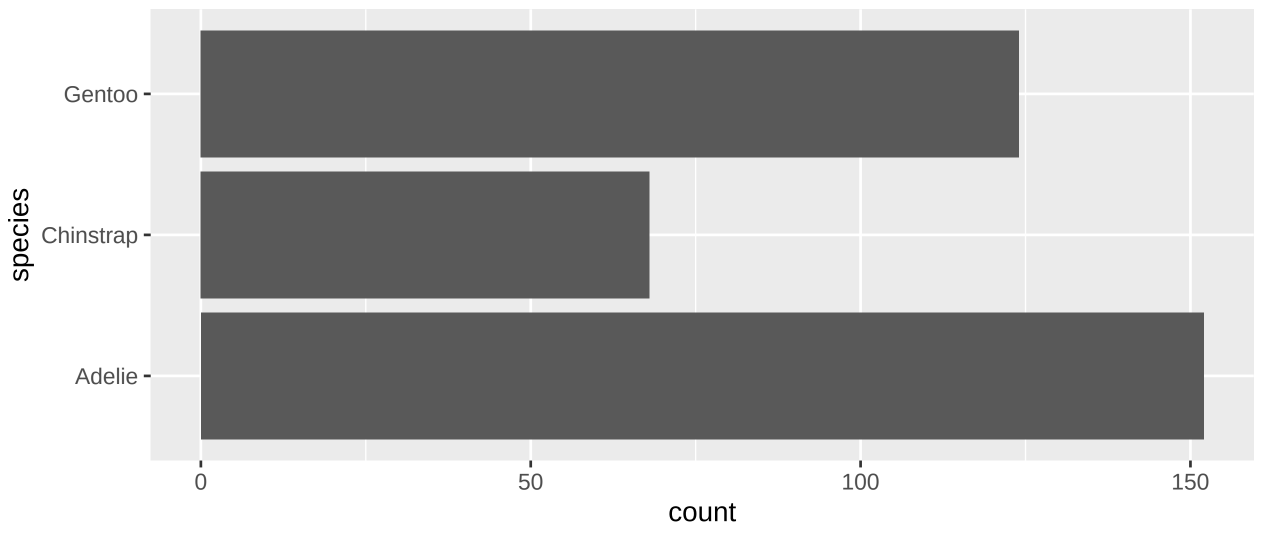


Figure 4: Horizontal bar plot of penguin specie count.

```
## 3 Gentoo      124
```

Using geom_col()

For `geom_col()` you need to have your data pre-counted. Let's create a new data frame with counts per `species` in the `penguins` dataset first.

```
# create new data frame with species counts
species_count_data <- penguins %>%
  count(species)

# inspect new data frame
species_count_data
```

```
## # A tibble: 3 x 2
##   species       n
##   <fct>     <int>
## 1 Adelie      152
## 2 Chinstrap    68
## 3 Gentoo      124
```

We now have a small data frame with two columns: `species` and `n` – a categorical and a numeric variables. We can now use `geom_col()` by first mapping one of the axes to the categorical variable (i.e., `species`) and the other to the numeric variable (i.e., `n`).

```
species_count_data %>%
  ggplot(aes(x = species, y = n)) +
  geom_col()
```

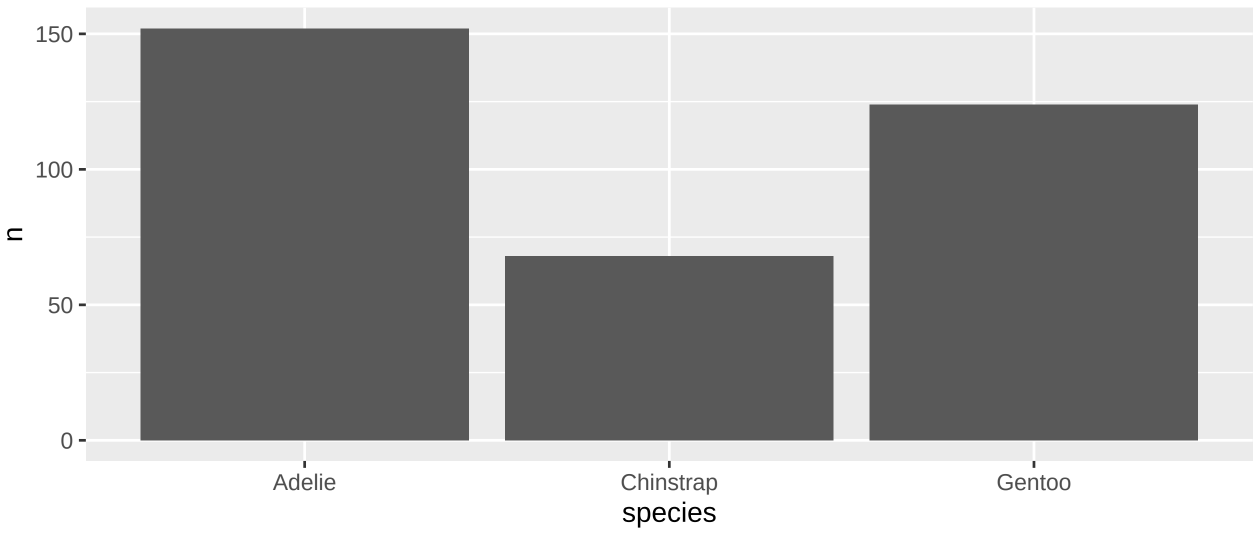
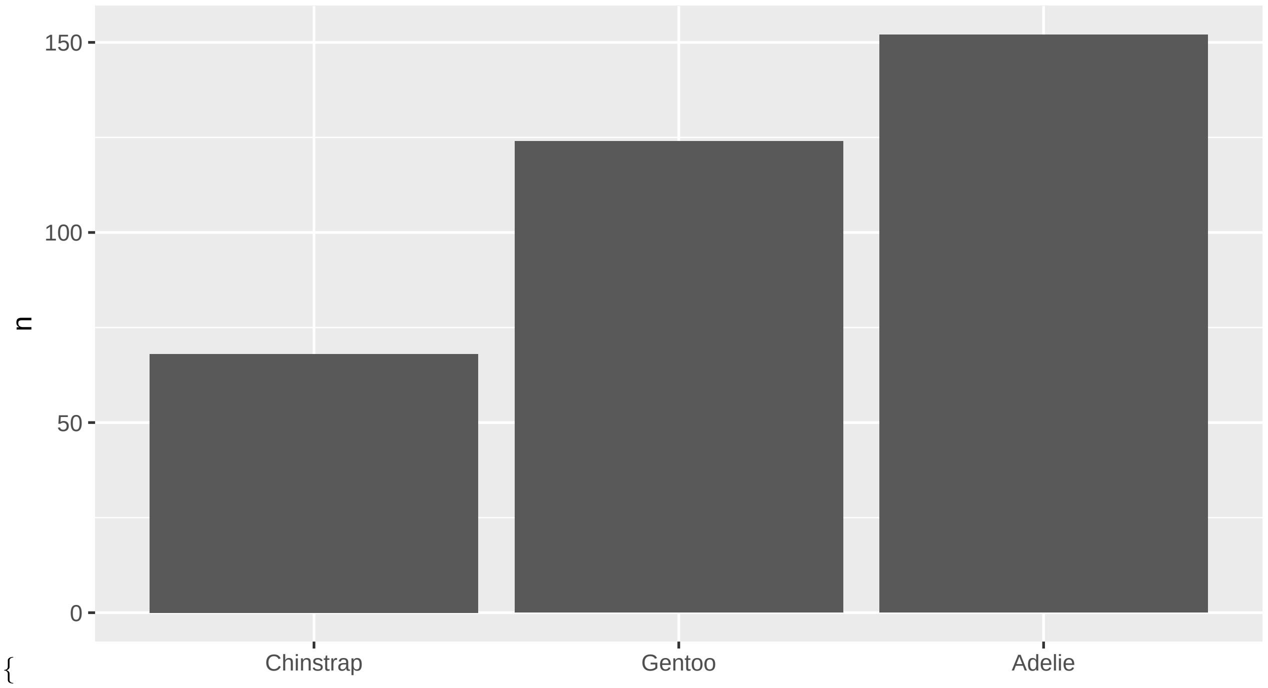


Figure 5: Bar plot of penguin count mapping both axis.

Most of the time we will summarize the data in some way first, and then use `geom_col()` to create barplots, since having both the categorical and the numeric variable explicitly represented in our dataset allows for reordering the bars in an easier more straightforward way.

```
species_count_data %>%
  ggplot(aes(x = reorder(species, n),
             y = n)) +
  geom_col() +
  labs(x = "")
```

\begin{figure}



{

}

\caption{Bar plot of penguin count mapping both axis with geom_col with species reordered by count.} \end{figure}

Dealing with ordered categorical variables

Data from tidytuesday

```
starbucks <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2021/2
```

```
starbucks %>%
  group_by(size) %>%
  summarize(mean_calories = mean(calories),
            n = n()) %>%
  slice_max(order_by = n, n = 4) %>%
  mutate(size = factor(size,
                       levels = c("short",
                                  "tall",
                                  "grande",
                                  "venti"))) %>%
  ggplot(aes(x = size,
             y = mean_calories)) +
  geom_col()
```

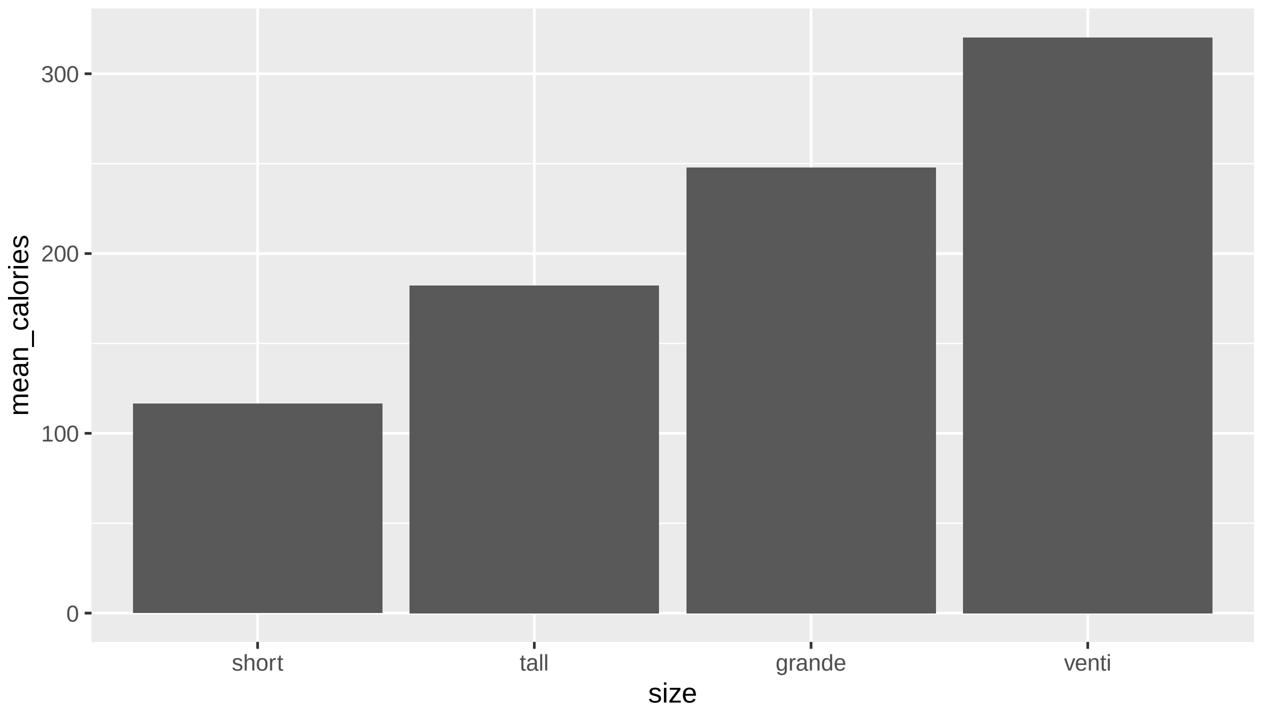


Figure 6: Mean calories across different starbucks drink sizes.