[Why Use xeringan? 2](#_Toc129864014)

[How xaringan Works 3](#_Toc129864015)

[Creating a New Slide 7](#_Toc129864016)

[Adjusting the Size of Figures 10](#_Toc129864017)

[Incrementally Revealing Content 10](#_Toc129864018)

[Aligning Content with Content Classes 12](#_Toc129864019)

[Adding Background Images to Slides 17](#_Toc129864020)

[Applying CSS to Slides 17](#_Toc129864021)

[Custom CSS 18](#_Toc129864022)

[Themes 19](#_Toc129864023)

[The xaringanthemer Package 21](#_Toc129864024)

[Conclusion 22](#_Toc129864025)

Making Slideshow Presentations with xaringan

You’re now able to generate reports using a single tool, R Markdown. But what if you need to make a slideshow presentation, like those you might create in PowerPoint? Luckily, R has robust presentation-making capabilities. In this chapter, you’ll learn how to make presentations in R using xaringan. This package, which uses R Markdown, is the most widely used tool for creating slideshow presentations in R.

We’ll use xerigan to turn the penguin report from Chapter 6 into a slideshow. You’ll learn how to create new slides, selectively reveal content, adjust the alignment of text and images, and style your presentation with CSS.

Why Use xeringan?

In R Studio, you might have noticed that the Presentation option you see when creating a new R Markdown document provides several options for making slides, such as knitting an R Markdown document to PowerPoint. However, using the xarigan package provides advantages over these options.

, taught extensively. She argues that the package’s . For instance, b without needing themthem Instead, you can send someone the presentation by jing a We’ll discuss ways to publish your presentations online in Chapter XX.

A secondHTML documents are easy to manipulate, giving viewers their appearanceFor example, pcanallow them to view the content, such as byinginglets

How xaringan Works

To get started with xaringan, install the package by running install.packages("xaringan") in R Studio. Next, navigate to File4New File4R Markdown to create a new project. Choose the From Template tab and select the template called Ninja Presentation, then press OK.

You should get an R Markdown document containing some default content. Delete this and add your own. Here,I’ve used penguin report, except I’ve changedoutput format in the to xaringan::moon\_reader:

---

title: "Penguins Report"

author: "David Keyes"

date: "2024-01-12"

output: xaringan::moon\_reader

---

```{r setup, include=FALSE}

knitr::opts\_chunk$set(include = TRUE,

echo = FALSE,

message = FALSE,

warning = FALSE)

```

```{r}

library(tidyverse)

```

```{r}

penguins <- read\_csv("https://raw.githubusercontent.com/rfortherestofus/r-without-statistics/main/data/penguins-2008.csv")

```

# Introduction

We are writing a report about the \*\*Palmer Penguins\*\*. These penguins are \*really\* amazing. There are three species:

- Adelie

- Gentoo

- Chinstrap

## Bill Length

We can make a histogram to see the distribution of bill lengths.

```{r}

penguins %>%

ggplot(aes(x = bill\_length\_mm)) +

geom\_histogram() +

theme\_minimal()

```

```{r}

average\_bill\_length <- penguins %>%

summarize(avg\_bill\_length = mean(bill\_length\_mm,

na.rm = TRUE)) %>%

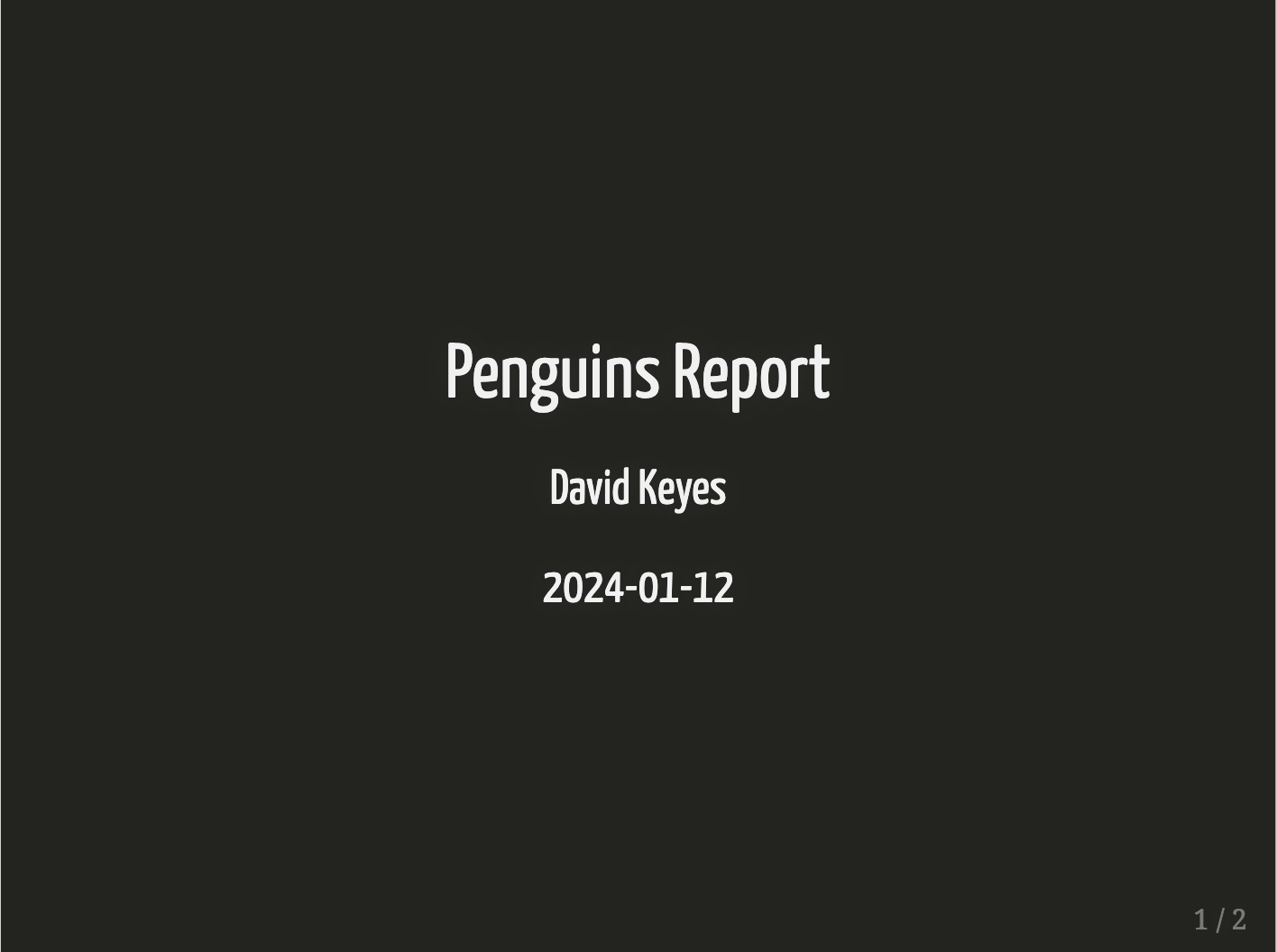
pull(avg\_bill\_length)

```

The chart shows the distribution of bill lengths. The average bill length is `r average\_bill\_length` millimeters.

Using this output format, we’ll now get slides rather than a Word document. XXXX Try hitting the Knit button to see what it looks like. You should receive an HTML file with the same name as R Markdown document (in my case, xaringan-example.html), shown in Figure 8-2.

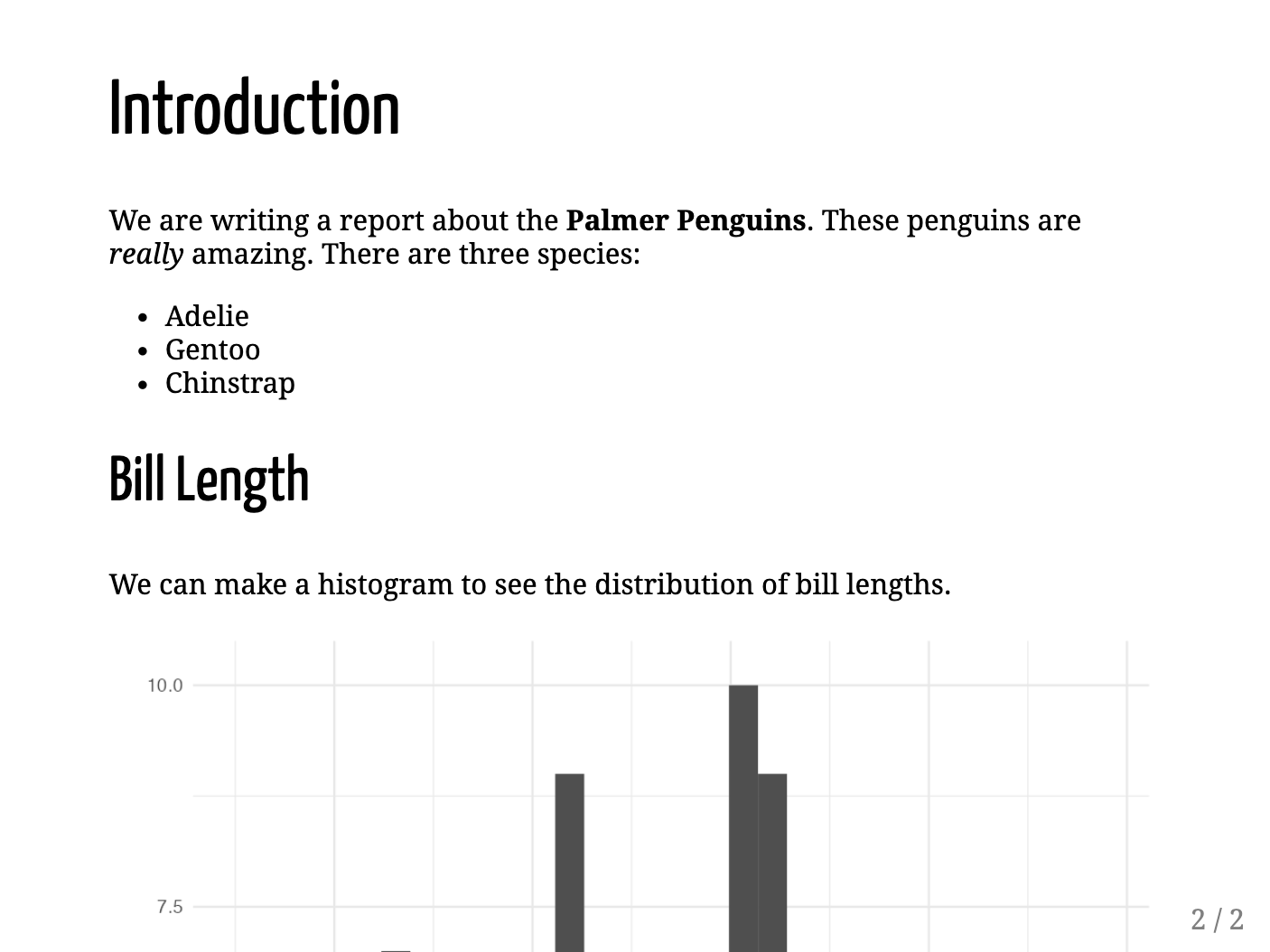
[F08002.png]



* + - * 1. The first slide of my presentation

If you scroll to the next slide with the right arrow key, you should see familiar content. Figure 8-3 shows this slide, which has the same text as the report from Chapter 6 and a cut-off version of its histogram.

[F08003.png]



* + - * 1. The second slide of my presentation

Although the syntax for making slides with xaringan is nearly identical to that used to make reports with R Markdown, we need to make a few tweaks so that our content can fit on the slides. When we’re working in a document that will be knitted to Word, its length doesn’t matter, because reports can be one page or 100 pages. Working with xaringan, however, requires considering how much content can fit on a single slide. Our cut-off histogram shows us what happens if we don’t do this. Let’s fix it.

Creating a New Slide

Let’s make our histogram fully visible by putting it in its own slide. To make a new slide, add three dashes (---) where you’d like the slide to begin. I’ve added these dashes before the histogram code:

---

## Bill Length

We can make a histogram to see the distribution of bill lengths.

```{r}

penguins %>%

ggplot(aes(x = bill\_length\_mm)) +

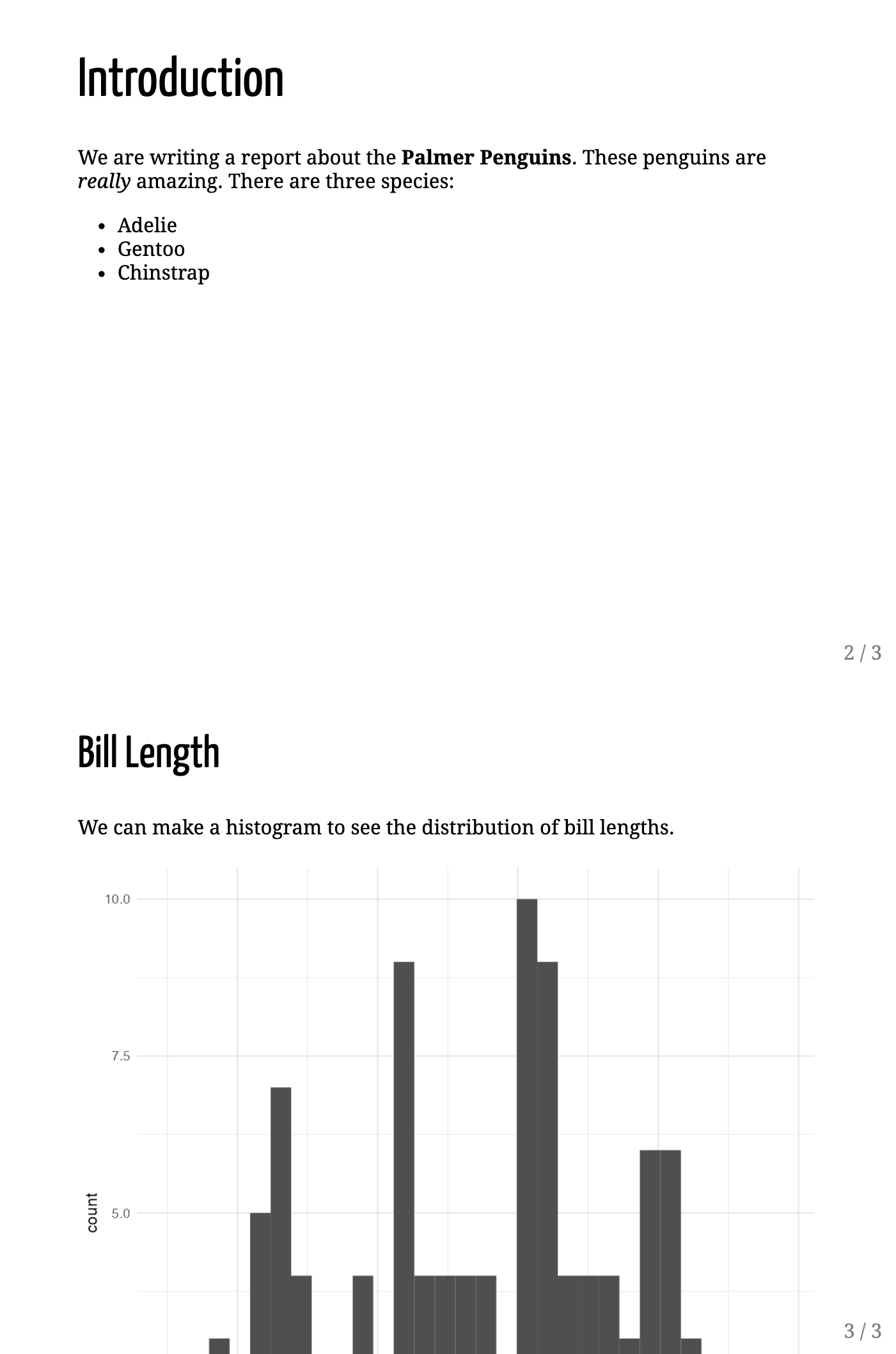
geom\_histogram() +

theme\_minimal()

```

If you knit the document again, what was one slide should now be broken into two: an Introduction slide and a Bill Length slide. You can see both in Figure 8-4.

[F08004.png]



* + - * 1. The presentation broken into two slides

If you look closely, you’ll notice that the bottom of histogram is still slightly cut off. To correct this, we’ll change its size.

Adjusting the Size of Figures

Adjust the size of the histogram using the code chunk option fig.height:

---

## Bill Length

We can make a histogram to see the distribution of bill lengths.

```{r fig.height = 4}

penguins %>%

ggplot(aes(x = bill\_length\_mm)) +

geom\_histogram() +

theme\_minimal()

```

Doing this makes the histogram fit on the slide and also reveals the text that was hidden below. XXXX

Incrementally Revealing Content

When presenting, it’s often useful to show only a portion of the content on each slide at a time. Let’s say, for example, that when we’re presenting the first slide, we want to talk a bit about each penguin species. Rather than show all three species when we open this slide, it would be nice to have the names come up one at a time.

We can do this using what xaringan calls incremental reveal. To use this feature, place two dashes (--) between any content you want to display incrementally. This code, for example, will let us show Adelie on the screen, then Adelie and Gentoo, then Adelie, Gentoo, and Chinstrap:

# Introduction

We are writing a report about the \*\*Palmer Penguins\*\*. These penguins are \*really\* amazing. There are three species:

- Adelie

--

- Gentoo

--

- Chinstrap

When presenting your slides, you’ll use the right arrow to incrementally reveal the species.

Aligning Content with Content Classes

When designing your presentation, you’ll also likely want to control the alignment of content. We can do this by adding what are known as content classes. Surround any content with the classes .left[], right[], and center[] to align them. For example, let’s use .center[] to align the code chunk that makes the histogram:

.center[

```{r fig.height = 4}

penguins %>%

ggplot(aes(x = bill\_length\_mm)) +

geom\_histogram() +

theme\_minimal()

```

]

Doing this would center the chart on the slide.

Other built-in options can make two-column layouts. Adding pull-left[] and pull-right[] will make two equally spaced columns. Let’s use these to display the histogram on the left side of the slide and the accompanying text on the right:

.pull-left[

```{r fig.height = 4}

penguins %>%

ggplot(aes(x = bill\_length\_mm)) +

geom\_histogram() +

theme\_minimal()

```

]

.pull-right[

```{r}

average\_bill\_length <- penguins %>%

summarize(avg\_bill\_length = mean(bill\_length\_mm,

na.rm = TRUE)) %>%

pull(avg\_bill\_length)

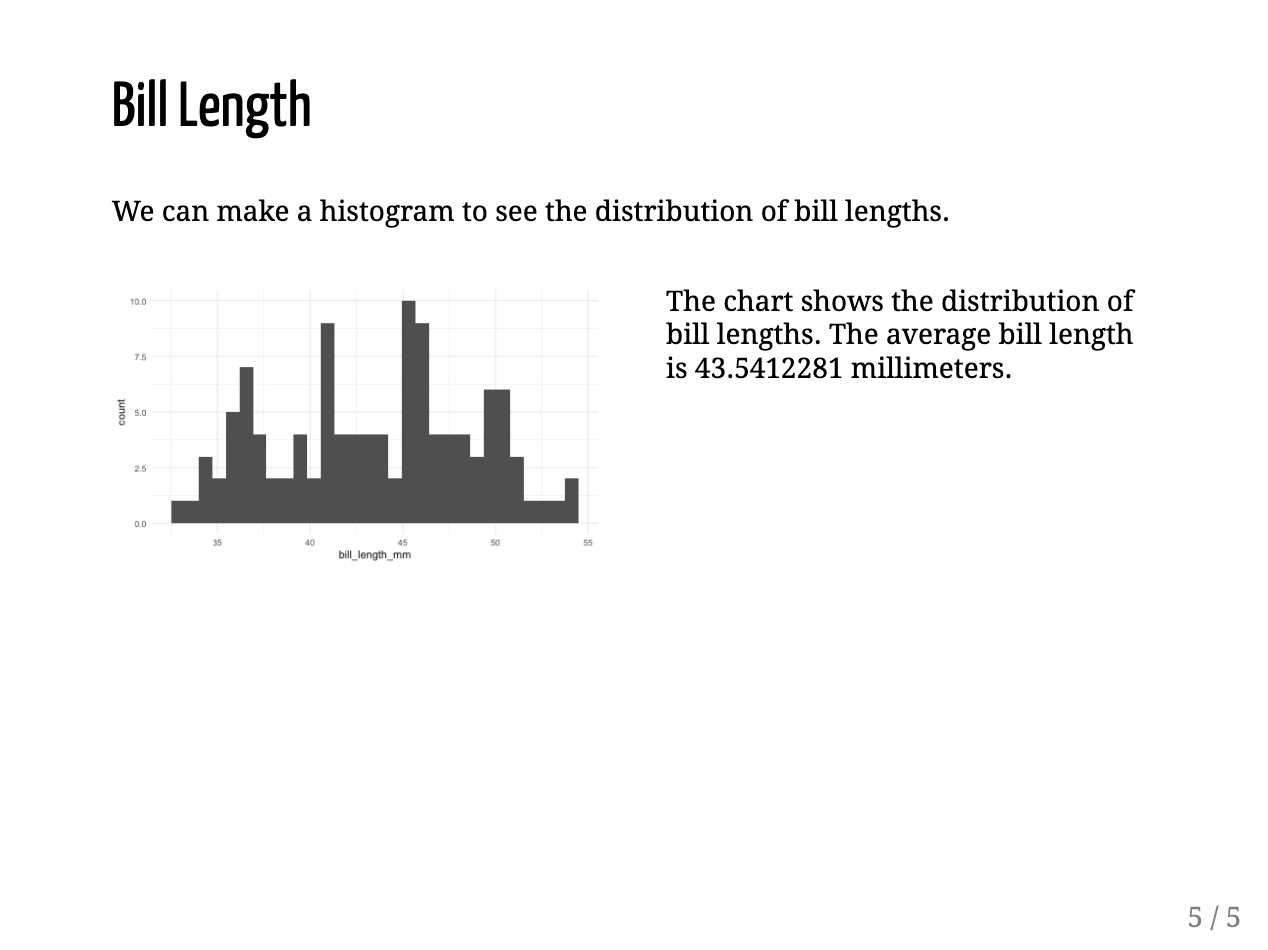
```

The chart shows the distribution of bill lengths. The average bill length is `r average\_bill\_length` millimeters.

]

You can see what this looks like in Figure 8-7.

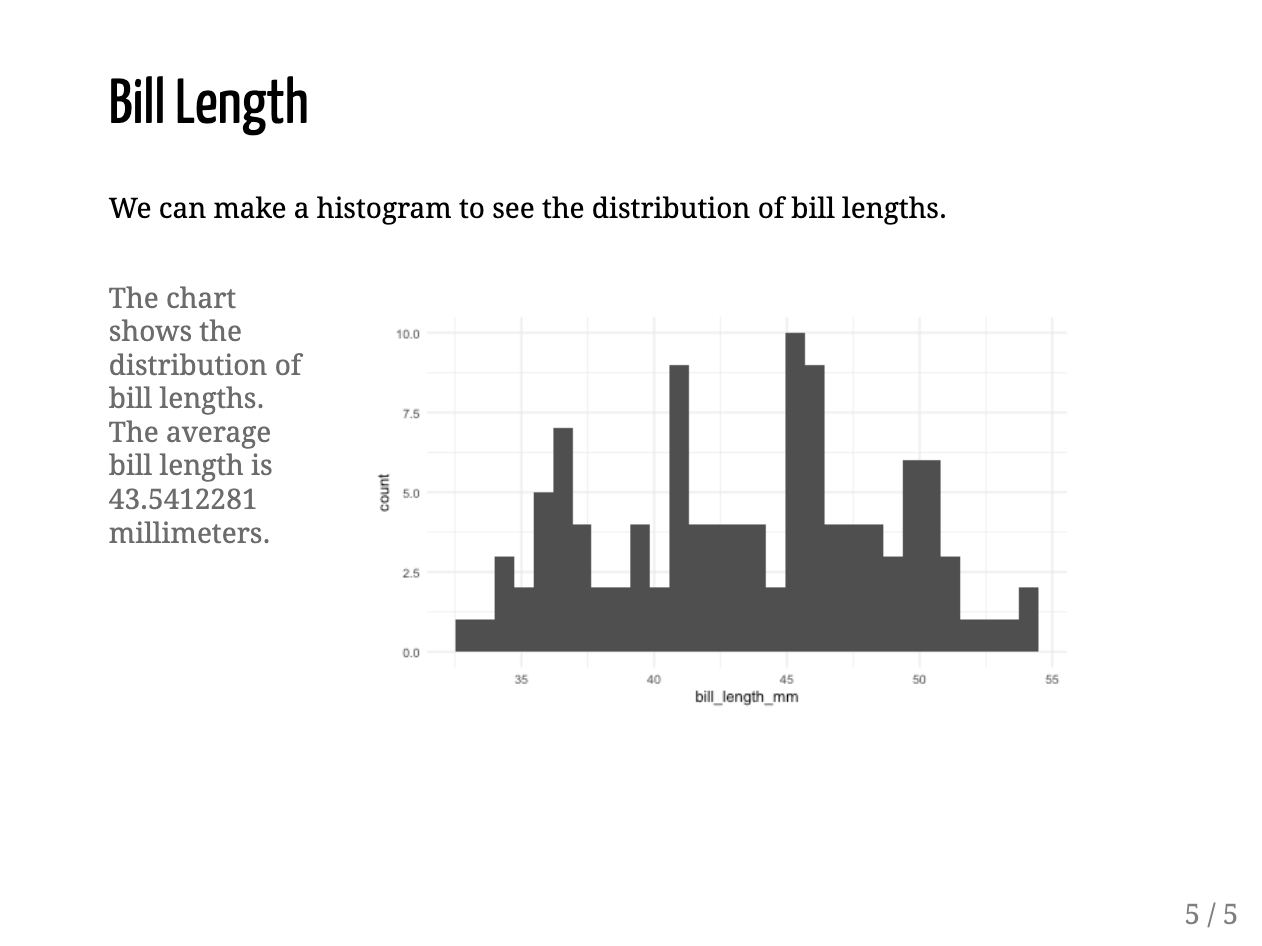
[F08007.png]



* + - * 1. A slide with two columns

To make a narrow left column and wide right column, use the content classes .left-column[] and .right-column[]. Figure 8-8 shows what the slide would look like with the text on the left and the histogram on the right.

[F08008.png]



* + - * 1. A slide with a smaller left column and a larger right column

XXXX

In addition to aligning particular pieces of content on slides, we can also horizontally align the entire content using the .left, right, and center classes. To do this, specify the class right after the three dashes that indicate a new slide, but before any content:

---

class: center

## Bill Length

We can make a histogram to see the distribution of bill lengths.

```{r fig.height = 4}

penguins %>%

ggplot(aes(x = bill\_length\_mm)) +

geom\_histogram() +

theme\_minimal()

```

Doing this would give us a fully horizontally centered slide. To vertically center the slide, you can use the class .middle, as you’ll see in the next section.

Adding Background Images to Slides

The syntax we just used to center the entire slide can also enable us to add a background image. Lets create a new slide, use the classes center and middle to horizontally and vertically align the content, and add a background image by surrounding the path to the image with url().

class: center, middle

background-image: url("penguins.jpg")

## Penguins

Doing this produces a slide with a picture of penguins in the background and the text Penguins in front of it, as shown in Figure 8-10.

[F08010.png]



* + - * 1. Figure 8-10: A slide that uses a background image

Now let’s add custom CSS to improve this new slide.

Applying CSS to Slides

One issue with the slide we just made is that the word Penguins is hard to read. It would probably be best if we could make the text bigger and a different color. To do this, we need to use CSS, the language used to style HTML documents. If you’re thinking, “I’m reading this book to learn R, not CSS,” don’t worry. You’ll need only a bit of CSS to make tweaks to your slides.

To apply CSS to your slides, you can write your own custom code, use a CSS theme, or combine the two approaches using the xaringanthemer package.

Custom CSS

To add custom CSS, create a new code chunk, placing css between the curly brackets:

XXXXX

In this code chunk, we tell R Markdown to make the h2 (the second-level header) 150px and white. We must also add the .remark-slide-content before the h2 to make sure we target the specific element in our presentation. The term remark comes from remark.js, a JavaScript library for making presentations that xaringan uses under the hood. You can see the new slide in Figure 8-11.

[F08011.png]



* + - * 1. The title slide with changes to the text to make it more visible

If you wanted to change the font in addition to the text’s size and color, you could do so with additional CSS:

XXXXXXXXX

The first line makes a font called Inter available to our slides. We do this because some people might not have this font on their computers. The next two lines apply Inter to the h2 and make it bold. You can see the slide with bold Inter font in Figure 8-12.

[F08012.png]



* + - * 1. The title slide with changes to the font to make the text more visible

Because xaringan slides are built as HTML documents, the sky is the limit in terms of customizing them with CSS. Table 8-1 includes other CSS you could use to apply custom styles.

Custom CSS Tweaks

|  |  |
| --- | --- |
| Tweak | CSS |
|  |  |
|  |  |

Themes

You may not care to in two ways without writing any CSS yourself The first way is to apply xaringan themes created by other R users. Run this code to get a list of all available themes:

names(xaringan:::list\_css())

The output should look something like this:

#> [1] "chocolate-fonts" "chocolate"

#> [3] "default-fonts" "default"

#> [5] "duke-blue" "fc-fonts"

#> [7] "fc" "glasgow\_template"

#> [9] "hygge-duke" "hygge"

#> [11] "ki-fonts" "ki"

#> [13] "kunoichi" "lucy-fonts"

#> [15] "lucy" "metropolis-fonts"

#> [17] "metropolis" "middlebury-fonts"

#> [19] "middlebury" "nhsr-fonts"

#> [21] "nhsr" "ninjutsu"

#> [23] "rladies-fonts" "rladies"

#> [25] "robot-fonts" "robot"

#> [27] "rutgers-fonts" "rutgers"

#> [29] "shinobi" "tamu-fonts"

#> [31] "tamu" "uio-fonts"

#> [33] "uio" "uo-fonts"

#> [35] "uo" "uol-fonts"

#> [37] "uol" "useR-fonts"

#> [39] "useR" "uwm-fonts"

#> [41] "uwm" "wic-fonts"

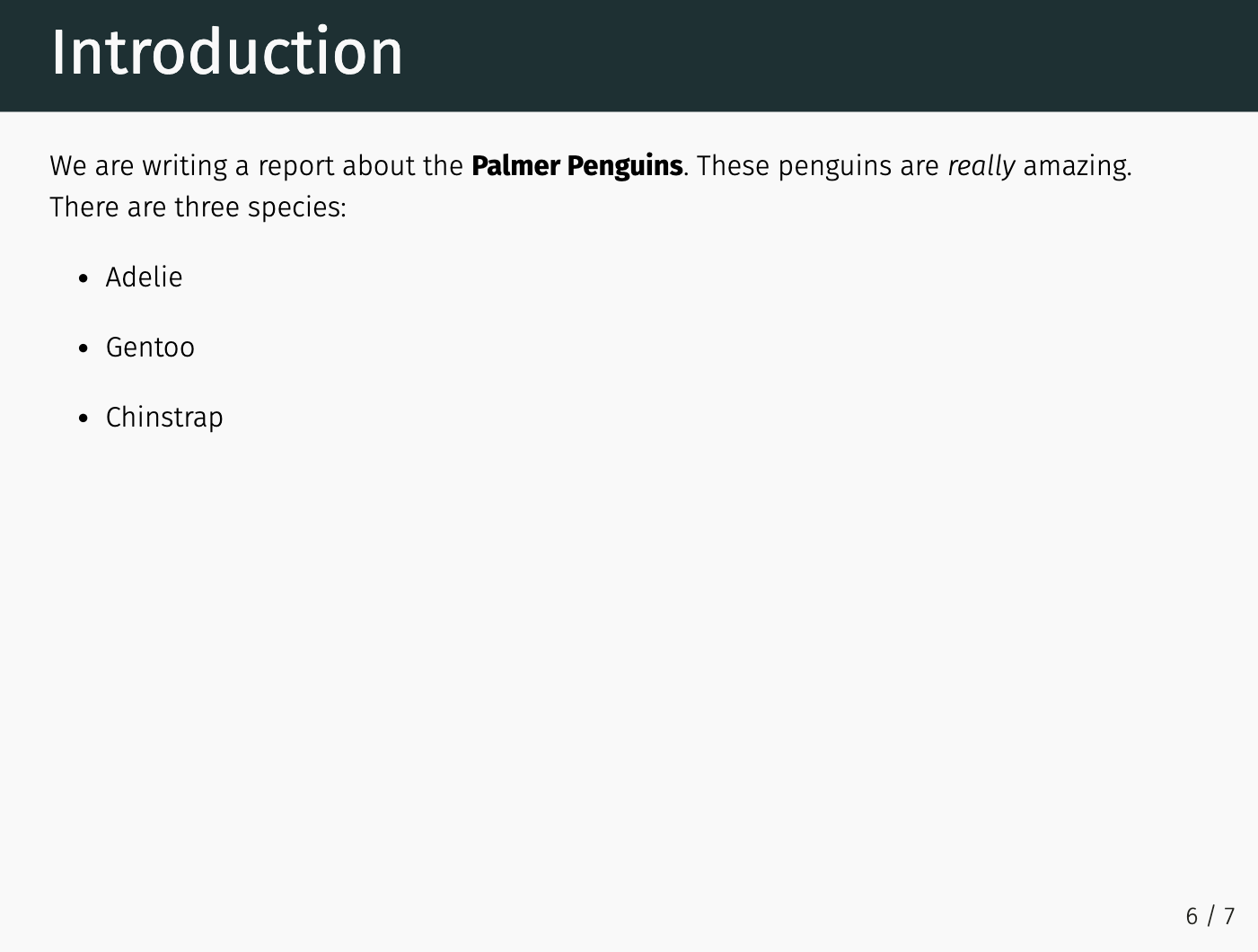
#> [43] "wic"

XXXX To use a particular theme, adjust your YAML as follows:

XXXXX

This code tells xaringan to use the default CSS, as well as customizations made in the metropolis and metropolis-fonts CSS themes. These come bundled with xaringan, so you don’t need to do anything beyond installing the package to access them. You can see how the theme changes the look-and-feel of our slides in Figure 8-13.

[F08013.png]



* + - * 1. A slide using the metropolis theme

If writing custom CSS is the totally flexible but more challenging option to tweaking your xaringan slides, then using a custom theme is way simpler but a lot less flexible. XXXXX

The xaringanthemer Package

A nice middle ground between writing custom CSS and applying someone else’s theme is to use the xaringanthemer package by Garrick Aden-Buie. This package includes several built-in themes but also allows you to easily create your own custom theme. After installing the package, adjust the css line in your YAML to use the xaringan-themer.css file:

XXXX

Now you can customize your slides by using the style\_xaringan() function. This function has over 60 arguments, allowing you to tweak nearly any part of your xaringan slides. To replicate the custom CSS we wrote earlier in this chapter using xaringanthemer, let’s use just a few of the arguments:

XXXX

XXXXX

One particularly nice thing about the xaringanthemer package is that you can use any font available on Google Fonts by simply adding its name to header\_font\_family or any other similar argument. You won’t have to run the line that made the Inter font available to us.

Conclusion