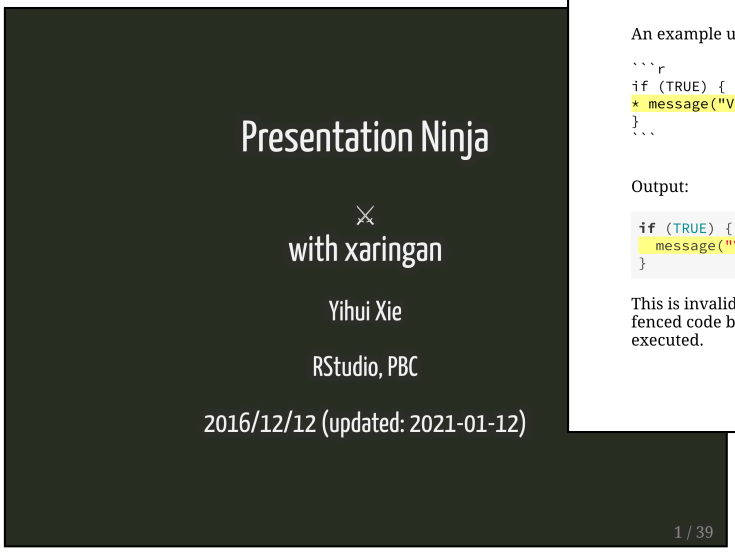
Slides with Xaringan

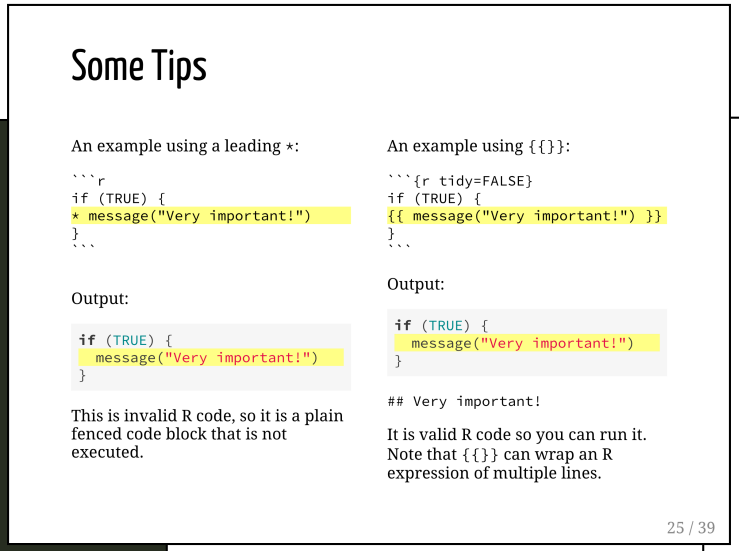
Andrew Zieffler

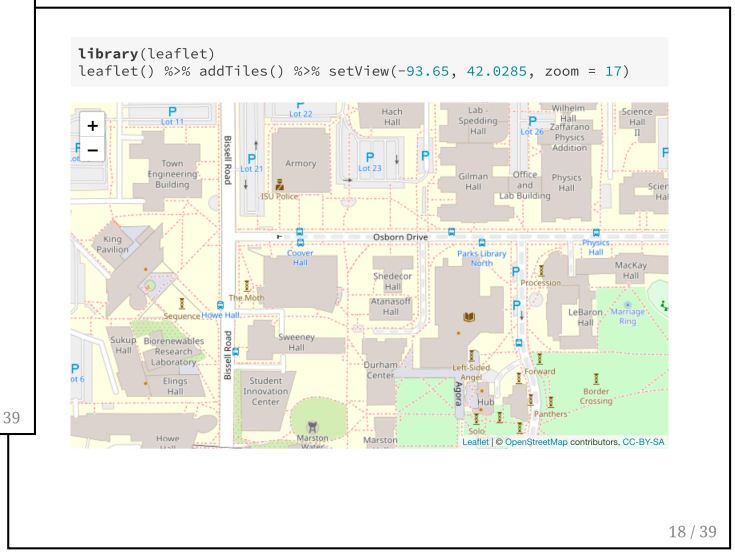
Much of this content came from Hill, A. (2019). Meet xaringan: Making slides in R Markdown.
Used with permission.

xaringan: Creating an HTML5 Slide Deck

The `{xaringan}` package, pronounced Share-ing-gan (/ʃæ.'riŋ.gæn/), allows us to use R Markdown to create a, HTML5 slide deck using Remark.js (a set of javascript-powered processors to covert markdown documents to a browser-based slide deck).



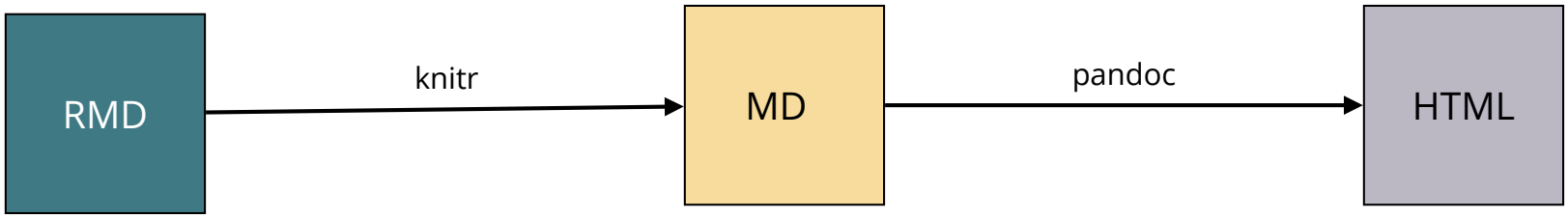




xaringan: Under the Hood

The `{xaringan}` package, pronounced Share-ing-gan (/ʃæ.'riŋ.gæn/), allows us to use R Markdown to create a, HTML5 slide deck using Remark.js (a set of javascript-powered processors to covert markdown documents to a browser-based slide deck).

Recall that the process we used to create HTML documents was:



The `{xaringan}` package uses remark.js a tool called *Moon Reader* (rather than pandoc) to produce the HTML file.

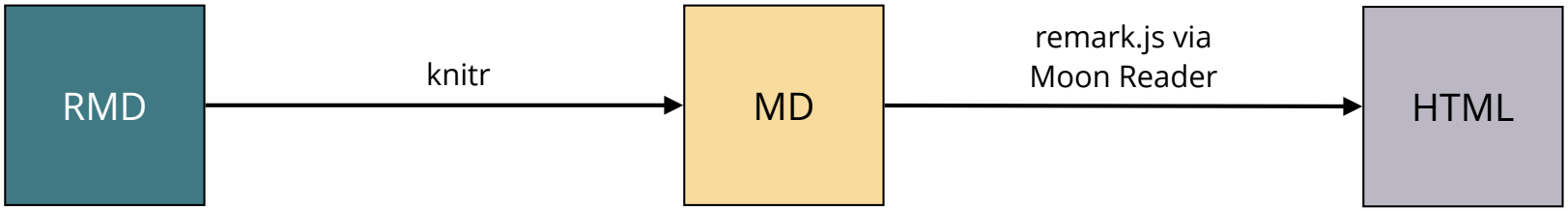


Image credit: Hill, A. (2019) Meet xaringan: Making slides in R Markdown. Workshop presented at RStudio Conf 2019.

Getting Started

Install the xaringan Package

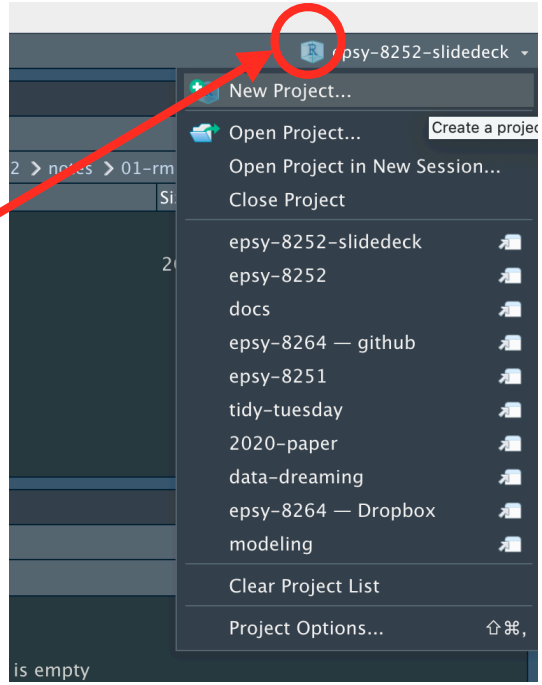
We need to install the `{xaringan}` package. To do this you will use the `install_github()` function from the `{remotes}` package. Note that this package is not available on CRAN so you cannot just click the "Install Packages" tab in RStudio.

```
# Install the xaringan package from github
remotes::install_github('yihui/xaringan')
```

If you get an error saying that remotes was not found you will need to install the `{remotes}` package prior to running the syntax above.

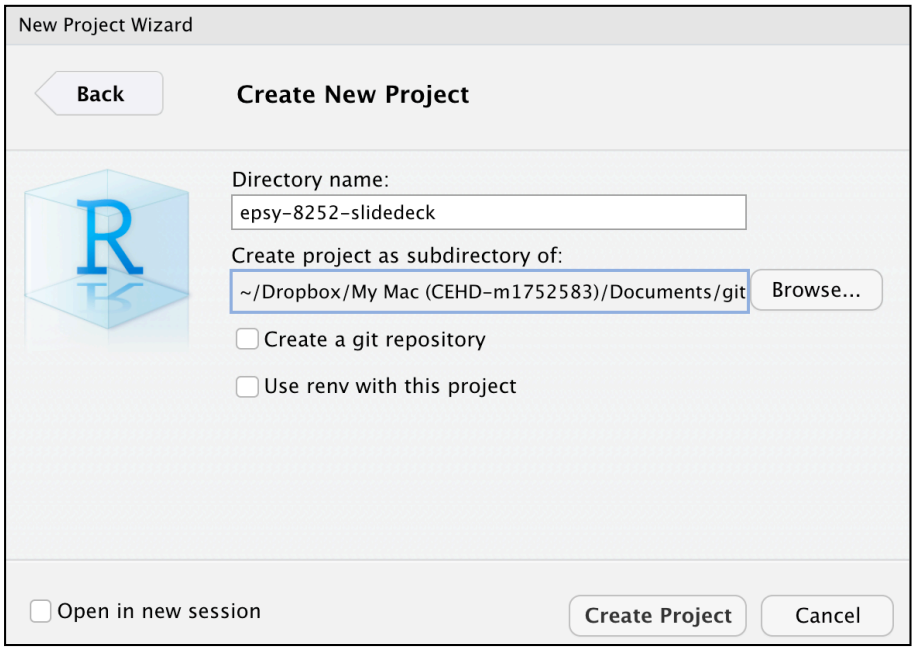
Create a New R Project

Click the project icon in RStudio (it might say No Project or something like that) and select `New Project…`



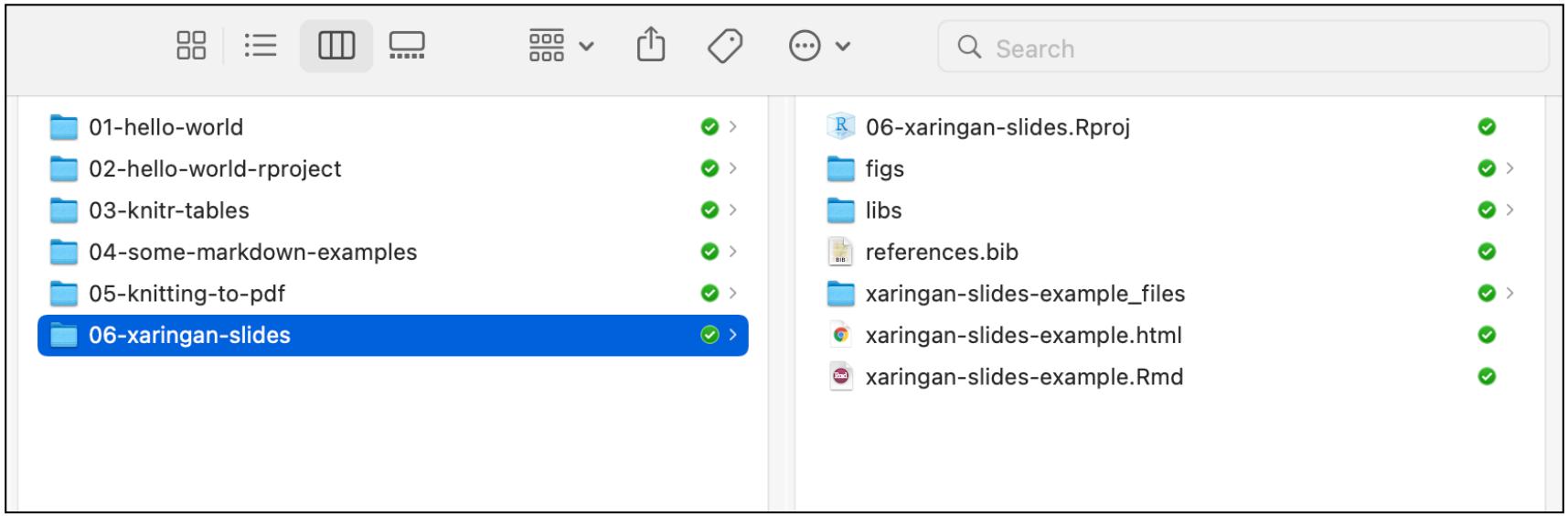
In the project wizard, select:

- `New Directory`, and then
- `New Project`



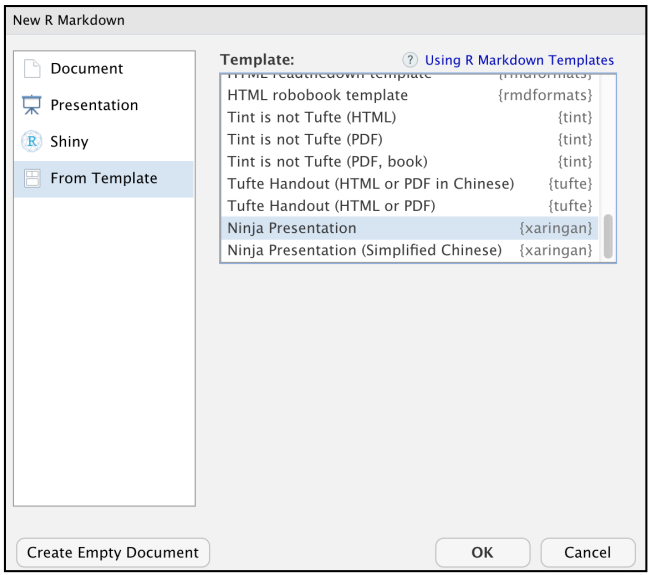
Give a directory name (this will be a folder that is created on your computer) and if you want to put that folder inside one you already have (e.g., in your EPsy 8252 folder), do so by clicking the `Browse…` button and navigating to where you want it to be.

You should now have a directory with whatever name you gave it (I called mine `06-xaringan-slides`). Inside that directory is an R project with the same name as your directory and the file suffix `.Rproj`.



Create New xaringan Slide Deck

Create a new R Markdown document (`File > New File > R Markdown...`).
Select `From Template` and choose `Ninja Presentation`.

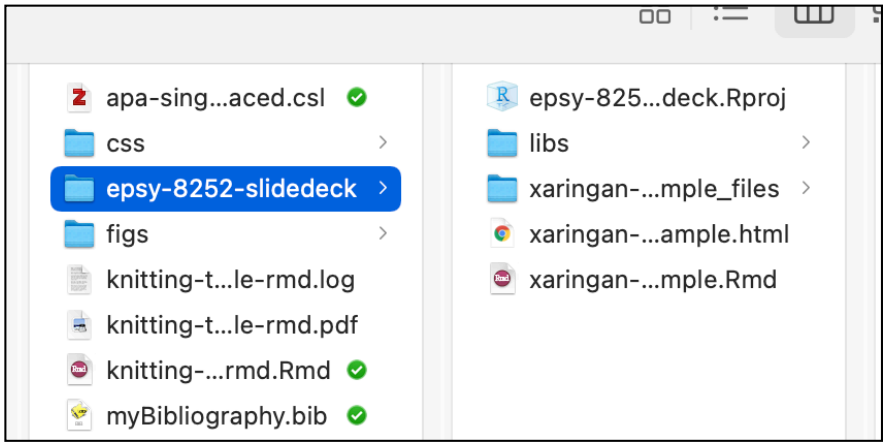


You may need to install the `{leaflet}` package to get the default template to knit. RStudio should warn you about this.

Knit the document.

When it asks you to save the RMD file, be sure it is saved in the same directory as you `.Rproj` file.

You may need to install the `{leaflet}` package to get the default template to knit. RStudio should warn you about this.



Knitting the default template will produce a set of HTML slides (in the `.html` file) that can be viewed and navigated through in any browser. It will also produce a few more files in your project directory.

Changing the Metadata in the YAML

YAML

Many of the xaringan YAML keys are similar to other R Markdown YAML keys.

```
---
title: "Example xaringan Slides"
subtitle: "Fun with R Markdown"
author: "Andrew Zieffler"
institute: "University of Minnesota"
date: "`r Sys.Date()`"
output: xaringan::moon_reader
---
```

Note in the `date:` key that we are using an inline code chunk to compute the date!

The `ouput:` key uses `moon_reader` from the `{xaringan}` package. This is a required key to produce the slides! Here I have put it on a single line because I removed the other keys from the default template.

There are other keys in the default template, but we will not need those now. They add additional functionality to the slide deck. If you are interested, you can see the xaringan documentation.

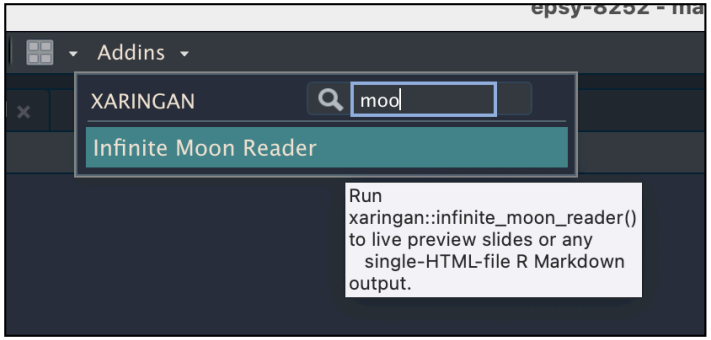
Your Turn

- Update the information in the `title:`, `subtitle:`, `author:`, and `institution:` keys to reflect you.
- Re-knit the document

Infinite Moon Reader

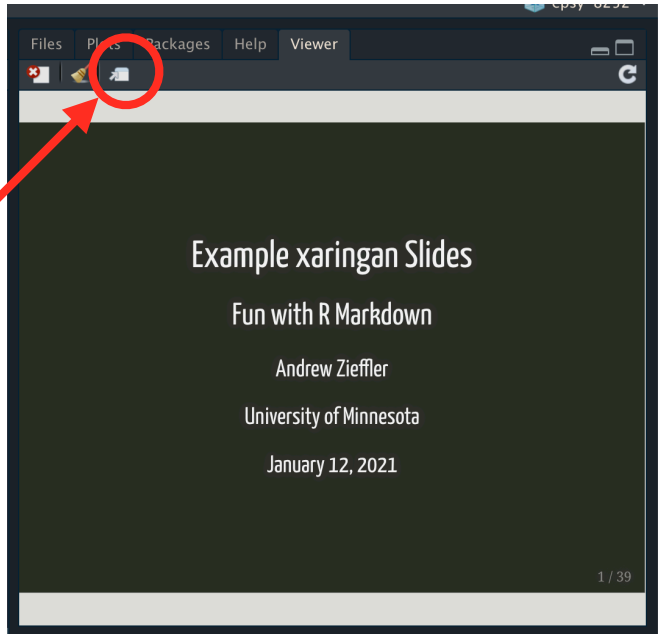
Infinite Moon Reader will allow you to see a live preview of your slides in the RStudio viewer every time you save the RMD file...without knitting.

- Click the `Addins` button in RStudio.
- Select `Infinite Moon Reader` (under the xaringan heading)



You should only need to run this once per session.

You can also preview the slides in your browser by clicking the Show in new window button in the viewer pane. Now saving the RMD file will update the slides in your browser!



Protip

You can use the Infinite Moon Reader to preview *any* HTML document you are creating in R Markdown.

YOUR TURN

- Change the information in the `date:` key to: `"`r format(Sys.Date(), '%B %d, %Y')`"`
- Save the RMD document

Depending on how many slides you have, the preview may take a second or two to update.

Adding Content

Creating Slides

- Slide 1 starts where YAML ends
- Three dashes in a row `---` (on a line by itself) is a new slide

The three question marks, `???`, indicates presenter notes. The text on the slide that appears after the question marks will not display unless you are in presenter mode.

To enter presenter mode, click on the slides in your browser window (or viewer pane) and click `p`.

```
---
title: "Example xaringan Slides"
subtitle: "Fun with R Markdown"
author: "Andrew Zieffler"
institute: "University of Minnesota"
date: "`r Sys.Date()`"
output: xaringan::moon_reader
---

# About Me

My name is Andy and I have two rescue dogs.

---

# Sadie

Sadie is a 10-year old sprollie; a springer spaniel/
border collie mix.

???

Sadie is white with black spots.

---

# Hank

Hank is a three-year old chi-weenie; a chihuahua/long-
haired dachshund mix.

???

Hank is brown with tan feet and chin.
```

Content Alignment

You can align content on a slide by specifying the horizontal and vertical alignment in `class:` at the beginning of the slide. (This is CSS, which if you recall from the previous notes, formats HTML.)

The Sadie slide will now be horizontally and vertically centered.

Horizontal	**Vertical**
`left`	`top`
`center`	`middle`
`right`	`bottom`

```
---

class: center, middle

# Sadie

Sadie is a 10-year old sprollie; a springer spaniel/
border collie mix.

???

Sadie is white with black spots.

---
```

Horizontally Aligning Only Some Content

Specifying `class:` aligns all the content on the slide. We can also select only some of the content to be aligned horizontally.

The text “Hello Hank!” in the Hank slide will be right aligned.

```
---

# Hank

Hank is a three-year old chi-weenie; a chihuahua/long-
haired dachshund mix.

.right[Hello Hank!]

???

Hank is brown with tan feet and chin.
```

To do this we use one of three particular CSS classes:

`.left[`*content*`]`
`.center[`*content*`]`
`.right[`*content*`]`

Add an Image

We can add an image by including the markdown syntax for including an image in a document.

Here we add a PNG image of Hank.

I have uploaded this image to the web and used the UMN URL shortner page to get a zlink.

```
---

# Hank

Hank is a three-year old chi-weenie; a chihuahua/long-haired dachshund mix.

![Image of Hank](https://z.umn.edu/hank)

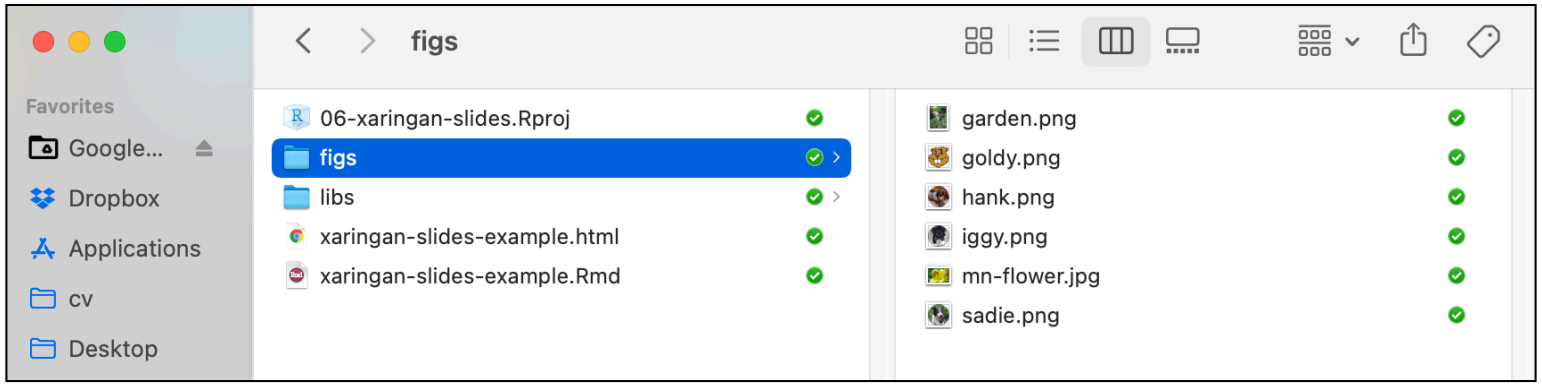
.right[Hello Hank!]

???

Hank is brown with tan feet and chin.
```

Local Images

If the image file is on your local computer, replace the URL with the file's pathname in quotation marks. I created a directory called `figs` in my project directory and that is where I store all image files.



To find the image we give the pathname relative to where the RMD file is. The path `figs/hank.png` tells us we need to go into the `figs` directory (`figs/`) and then get `hank.png`.

```
45
46 ---
47
48 # Iggy
49
50 Ignatius Fauci Fortenberry (Iggy) is a 7-month old schnauzer/lab mix.
51
52 ![Image of Iggy](figs/iggy.png)
53
54 ???
55
56 Iggy is black with white feet, a white chest, and a white goatee.
57
58
```

You have more control if you use HTML syntax to include the image, rather than the Markdown syntax. This is because you can include additional attributes (e.g., `width=`) and even CSS.

Here we add a PNG image of Sadie using HTML syntax that is 40% of the slide's width. The `style=` attribute includes CSS to center the image on the slide.

The `<br />` HTML tag adds a linebreak between the description of Hank and the image of Hank.

```
---

class: middle, center

# Sadie

Sadie is a 10-year old sprollie; a springer spaniel/border collie mix.

<br />

<img src="figs/sadie.png" alt="Image of Sadie" width="40%" style="display:block; margin-left:auto;
margin-right:auto;" />


???

Sadie is white with black spots.

```

protip: Include the `alt=` attribute every time you use an `<img />` tag. If there is a problem displaying your image, the alt text will display. It also helps with search engine optimization (SEO) if you put your slides online, since search engines cannot read images. Lastly, and perhaps most importantly, when visually-impaired or blind users view your slides, their screen readers will read the alt text out loud so they understand any images included in the slides.

Incremental Reveals

We can use two dashes, `--`, within a slide to incrementally reveal content on the slide.

Here we incrementally reveal:

1. The description of Hank;
2. The image of Hank;
3. The text “Hello Hank!”

```
---

# Hank

Hank is a three-year old chi-weenie; a chihuahua/long-
haired dachshund mix.

--

<br />

<img src=“figs/hank.png” alt="Image of Hank" width="40%"
style="display:block; margin-left:auto; margin-
right:auto;" />

--

.right[Hello Hank!]

???

Hank is brown with tan feet and chin.
```

Two Columns

We can use `.pull-left[]` and `.pull-right[]` to include content in two columns on the slide.

Here I create a column on the left-side of the slide that includes some text and a column on the right-side of the slide that includes an image.

- On a full slide, `.pull-left[]` and `.pull-right[]` are each 47% of the slide width.
- You can also similarly use `.left-column[]` and `.right-column[]`. The sizes are then 20% and 75%, respectively.

```
---

# Gardening

.pull-left[
I like to garden. I have replaced most of the grass in my
yard with paths and gardens.
]


.pull-right[
![Andy's Garden](figs/andy-garden.jpg)
]

???

This is on the west side of our house.
```

Your Turn

Create some slides in your deck. Consider including:

- Add presenter notes with `???`
- Incremental reveals with `--`
- Align all content with the class slide property
- Align some text on a slide with `.left` / `.center` / `.right`
- Create two-column slides with `.pull-left` / `.pull-right` OR `.left-column` / `.right-column`

Syntax, Figures, and Tables

Adding Code Chunks

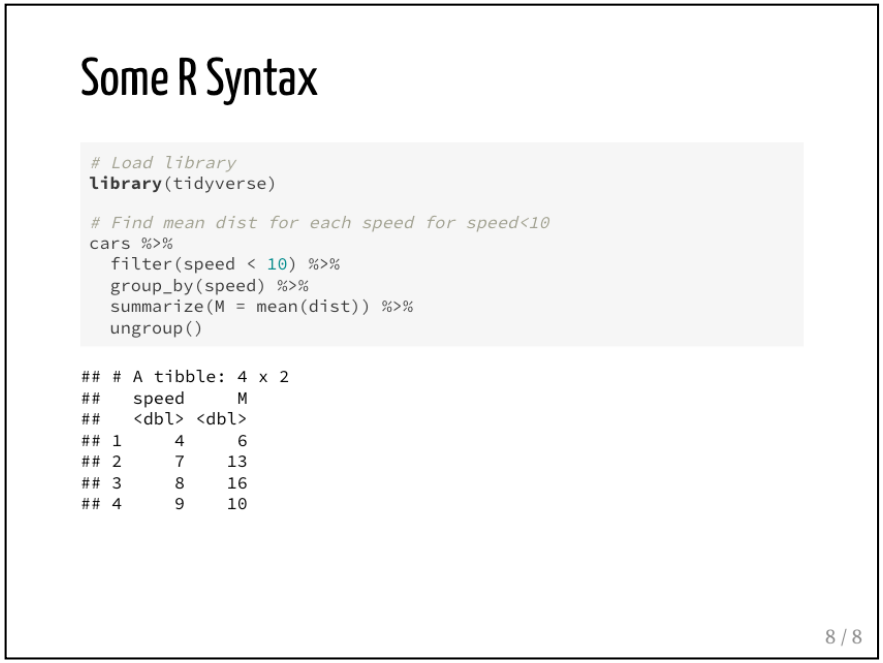
We can include code chunks in the slides as well. We do this the same way we did in any RMD document.

````
---

# Some R Syntax

```{r message=FALSE}
# Load library
library(tidyverse)

# Find mean dist for each speed for speed<10
cars %>%
  filter(speed < 10) %>%
  group_by(speed) %>%
  summarize(M = mean(dist)) %>%
  ungroup()
```
````



Highlighting Code

The extra YAML included in the default xaringan RMD template allows code highlighting.

```
---
title: "Example xaringan Slides"
subtitle: "Fun with R Markdown"
author: "Andrew Zieffler"
institute: "University of Minnesota"
date: "`r format(Sys.Date(), '%B %d, %Y')`"
output:
  xaringan::moon_reader:
    nature:
      highlightStyle: github
      highlightLines: true
---
```

The `nature:` key adds code highlighting.

The highlight style options are: arta, ascetic, dark, default, far, github, googlecode, idea, ir-black, magula, monokai, rainbow, solarized-dark, solarized-light, sunburst, tomorrow, tomorrow-night-blue, tomorrow-night-bright, tomorrow-night, tomorrow-night-eighties, vs, zenburn.

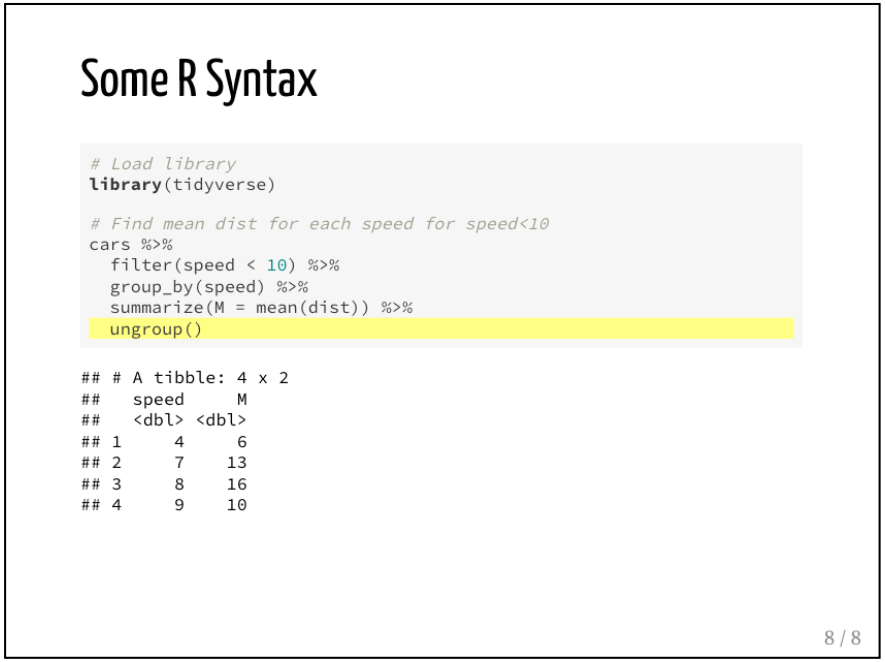
To actually highlight code we include `#<<` at the end of any syntax we want highlighted. Here we highlight the line `ungroup()` in the syntax.

````
---

# Some R Syntax

```{r message=FALSE}
# Load library
library(tidyverse)

# Find mean dist for each speed for speed<10
cars %>%
  filter(speed < 10) %>%
  group_by(speed) %>%
  summarize(M = mean(dist)) %>%
  ungroup() #<<
```
````



Note that to highlight a line of plain markdown text, use a `*` (from remark docs)

Highlighting Output

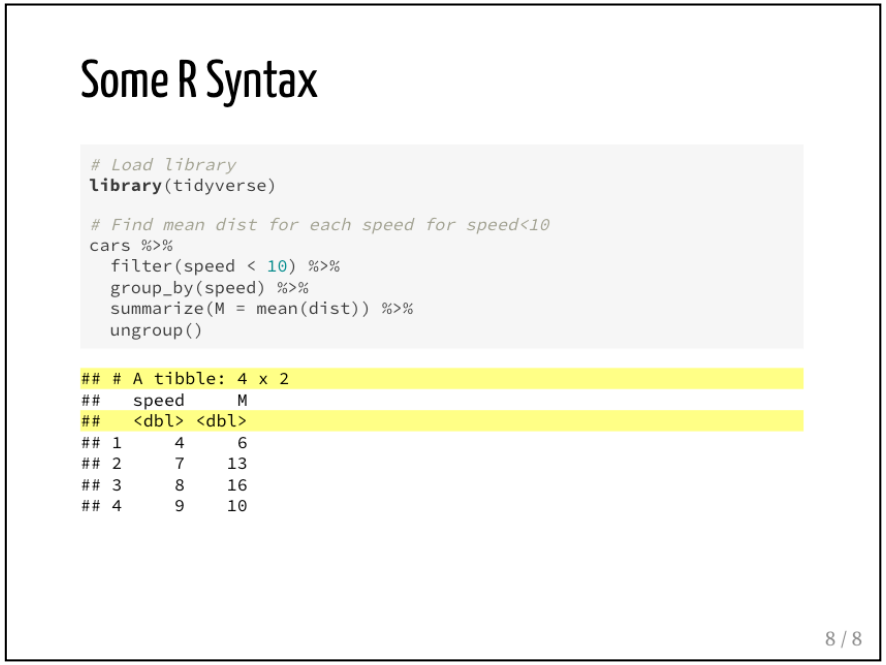
You can also highlight code output, by including `highlight.output=` in the options for the code chunk. Include a vector of the output lines you want to highlight. Here we highlight Lines 1 and 3 in the output.

````
---

# Some R Syntax

```{r message=FALSE, highlight.output=c(1,3)}
# Load library
library(tidyverse)

# Find mean dist for each speed for speed<10
cars %>%
  filter(speed < 10) %>%
  group_by(speed) %>%
  summarize(M = mean(dist)) %>%
  ungroup()
```
````



Some R Syntax

```
# Load library
library(tidyverse)

# Find mean dist for each speed for speed<10
cars %>%
  filter(speed < 10) %>%
  group_by(speed) %>%
  summarize(M = mean(dist)) %>%
  ungroup()
```

```
## # A tibble: 4 x 2
##   speed     M
##   <dbl> <dbl>
## 1     4     6
## 2     7    13
## 3     8    16
## 4     9    10
```

8 / 8

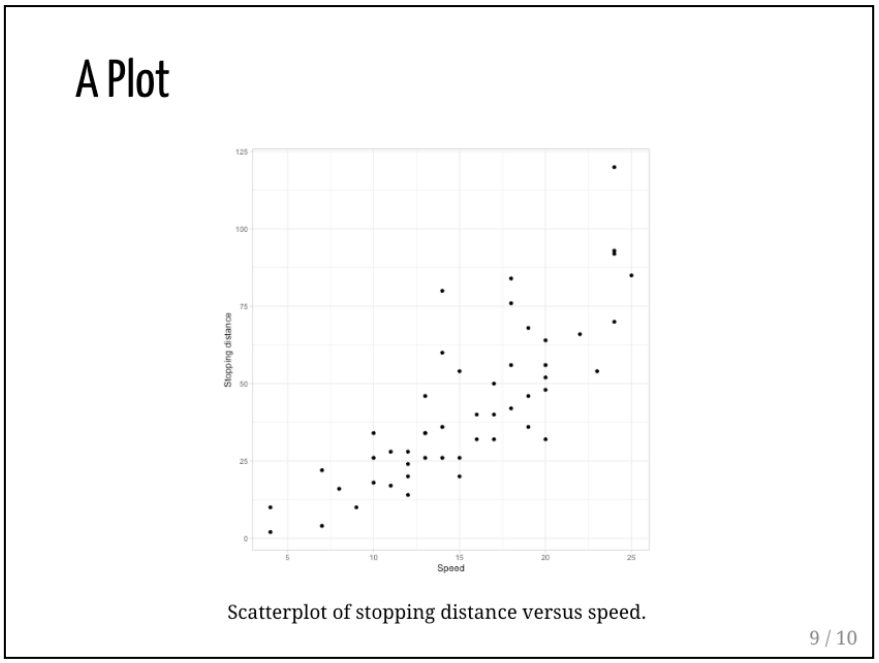
Plots

You can include plots in your slide decks by using code chunks. You may need to adjust the `out.width=` option in the code chunk to get the caption to display on the slide.

````
---

# A Plot

```{r echo=FALSE, fig.align='center',
fig.cap="Scatterplot of stopping distance versus speed.",
out.width="60%"}
# Create plot
ggplot(data = cars, aes(x = speed, y = dist)) +
  geom_point() +
  xlab("Speed") +
  ylab("Stopping distance") +
  theme_light()
```
````



The resolution of the plot is not high. The overall resolution (in dots per inch; DPI) of a plot produced for an HTML document by knitr is a function of: (1) the DPI for the bitmap graphics device, and (2) the retina display ratio.

$$\text{Overall DPI} = \text{Bitmap Device DPI} \times \text{Retina Display Ratio}$$

Using the default knitr settings for a plot produced in an HTML document, the overall resolution for our plot is 144 DPI.

$$\text{Overall DPI} = 72 \times 2 = 144$$

To increase the resolution of our plot we need to adjust either the `dpi=` or the `fig.retina=` options in the code chunk.

Here we change the `fig.retina=` option to 3. This produces an overall resolution of 216 DPI. This looks pretty good when I examine the slide in the browser. Some print journals will have higher standards for figure resolutions (often 300 DPI).

````
---

# A Plot

```{r echo=FALSE, fig.align='center',
fig.cap="Scatterplot of stopping distance versus speed.",
out.width="60%", fig.retina=3}
# Create plot
ggplot(data = cars, aes(x = speed, y = dist)) +
  geom_point() +
  xlab("Speed") +
  ylab("Stopping distance") +
  theme_light()
```
````

To get the figure to look good (i.e., have reasonable aspect ratio and resolution, you may need to play around with the values of some of the chunk options (e.g., `out.width=`, `fig.retina=`). This is because changing one has an effect on the others:

$$\texttt{out.width} = \frac{\texttt{fig.width} \times \texttt{dpi}}{\texttt{fig.retina}}$$

For more info about figures in R Markdown, see: ZevRoss (2017). Tips and tricks for working with images and figures in R Markdown documents.

Side-by-Side Code and Plot

If you want your code and plot to appear side-by-side (in two columns), we use the `.pull-left` and `.pull-right` classes. The clunky way to do this is to include a code chunk with the plot syntax and `eval=FALSE` in one of these classes (this will show your syntax but not produce the plot), and the same syntax but with `echo=FALSE` in the other (this will produce the plot but hide the syntax).

That methodology works, but it has repeated syntax, which violates the adage in computing: *Never repeat yourself*. The following code avoids this repetition. The key is we are labelling the code chunk that includes the syntax to create the plot, and then referencing it in the second code chunk using the `ref.label=` option.

````
---

# Code First; Plot Second

.pull-left[

```{r plot-last, fig.show = 'hide'}
  code goes here
```
]

.pull-right[
```{r ref.label = 'plot-last', echo = FALSE}
```
]
````

````
---

# Plot First; Code Second

.pull-left[

```{r plot-first, echo = FALSE}
  code goes here
```
]

.pull-right[
```{r ref.label = 'plot-first', eval = FALSE}
```
]
````

Code First; Plot Second

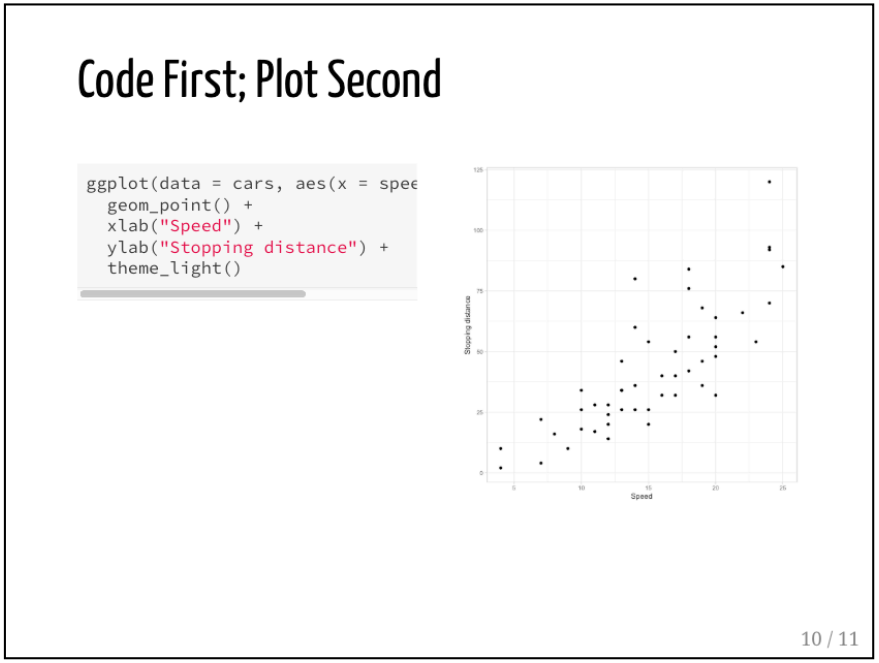
```
---

# Code First; Plot Second

.pull-left[

```{r plot-last, fig.show = 'hide'}
# Create plot
ggplot(data = cars, aes(x = speed, y = dist)) +
  geom_point() +
  xlab("Speed") +
  ylab("Stopping distance") +
  theme_light()
```
]

.pull-right[
```{r ref.label = 'plot-last', echo = FALSE}
```
]
```



Code First; Plot Second

````
---

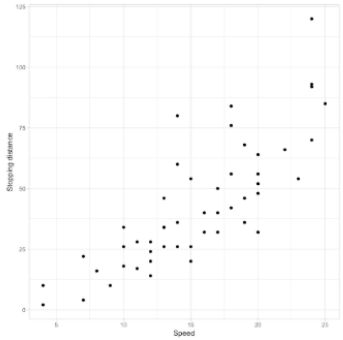
# Plot First; Code Second

.pull-left[

```{r plot-first, echo = FALSE}
  code goes here
```
]

.pull-right[
```{r ref.label = 'plot-first', eval = FALSE}
```
]
````

Plot First; Code Second



```
# Create plot
ggplot(data = cars, aes(x = spee
  geom_point() +
  xlab("Speed") +
  ylab("Stopping distance") +
  theme_light()
```

Tables

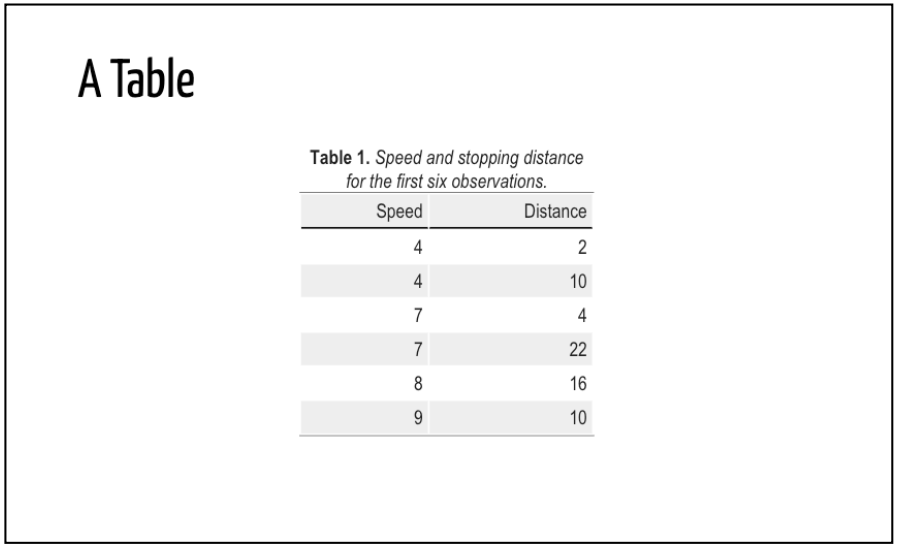
The `kable()` function and functions from the `{kableExtra}` package can be used to create a table.

````
---

# A Table

```{r message=FALSE, echo=FALSE}
# Load libraries
library(knitr)
library(kableExtra)

# Create table
cars %>%
  head() %>%
  kable(
    format = "html",
    col.names = c("Speed", "Distance"),
    caption = "<b>Table 1.</b>  <i>Speed and stopping
distance for the first six observations.</i>",
    table.attr = "style='width:40%;'"
) %>%
  kable_classic()
```
````



A Table

Table 1. *Speed and stopping distance for the first six observations.*

Speed	Distance
4	2
4	10
7	4
7	22
8	16
9	10

Note that to format text (e.g., in the `caption=` argument) we need to use HTML syntax rather than Markdown syntax. As an example, we use `<b>Text</b>` rather than `**Text**` to make the text bold.

Your Turn

Create some more slides in your deck.

- Create a slide that includes a plot (with hidden syntax).
- Create a slide that includes that has some highlighted R syntax.
- Create a slide that includes that some highlighted R output.
- Create a slide that includes side-by-side code and the resulting plot.
- Create a slide that includes a table.

Background Images, Title Slides, and Logos

Background Images

We can use the `background-image:` class at the beginning of a slide to set the location of the file to use in the background image. Along with that, we can also use `background-position:` and `background-size:`.

```
---

background-image: url(figs/mn-flower.jpg)
background-position: center
background-size: cover
class: center, middle

# Background Image

Background Size = cover
```

- `background-size:`
  - `cover` rescales and crops with no empty space
  - `contain` rescales only

- `background-position:` Play with this (see here for options)

Image Credit: Stanley-Erickson, C. **Minnesota**. Flickr.

Title Slides

````
---
title: “Markdown - Slides"
author: "Andrew Zieffler"
output:
  xaringan::moon_reader:
    seal: false
    nature:
      highlightStyle: github
      highlightLines: true
---



```{r setup, include=FALSE}
options(htmltools.dir.version = FALSE)
```

class: inverse, center, middle
background-image: url('figs/title-slide-bg.png')
background-size: cover
background-position: left top


# More R Markdown

<img src="figs/andy-avatar.png" width=“150px" />

### CREATING SLIDES WITH XARINGAN

Andrew Zieffler | EPsy 8252 | Spring 2021

???

Image credit: Hex stickers from https://github.com/
rstudio/hex-stickers
````

The best way to build a title slide is to customize one yourself. In the YAML we will include `seal: false`. This ignores all the metadata included in the YAML (although you still have to include the `title:` key.). Then our first slide we can customize as our title slide.

Logos

If you want to add a logo to every slide, we set the property `layout: true` in the first content slide, and then include a `background-image:`. We should also set the background-size: and background-position: properties.

```
---

layout: true

background-image: url('figs/goldy.png')
background-position: 95% 95%
background-size: 10%

---

  First slide where you want the logo to appear
```

- `background-size:` When given as a percentage, sets the size of the logo relative to its actual size. Here Goldy is set to 10% of the original image size.

- `background-position:` When percentages are provided, the first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%.

The image will appear at the same location on all remaining slides. To cancel it, insert a blank slide with `layout: false`.

```
---

layout: false

---

  No logo from this slide on
```

You can also include multiple logos at various locations. To do this we separate each of the background properties with a comma.

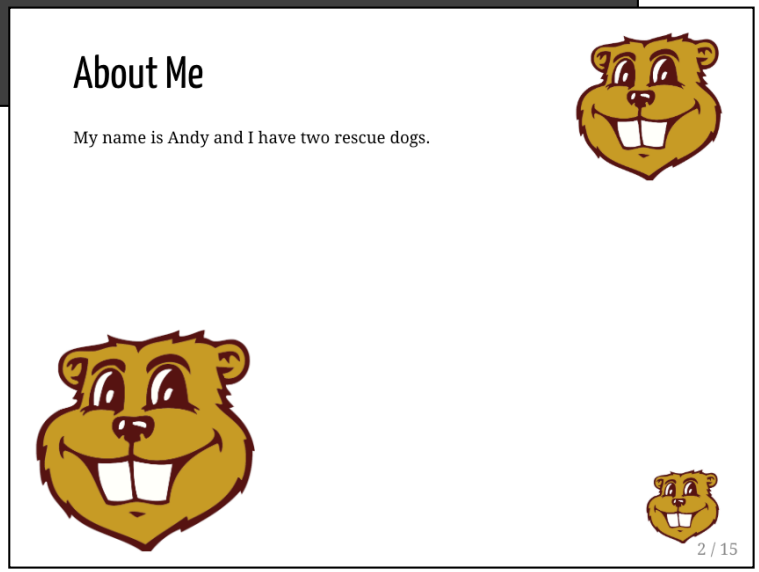
```
---

layout: true

background-image: url(figs/goldy.png), url(figs/goldy.png), url(figs/goldy.png)
background-position: 95% 95%, 95% 5%, 5% 95%
background-size: 10%, 20%, 30%

---

  First slide where you want the logos to appear
```



Your Turn

Create some more slides in your deck.

- Create a slide with a background image.
- Create a fancy title slide.
- Add one or more logos to some of the slides. Use your lab or department logo.

Final Remarks and Learning More

Labeling Slides

You can label slide with the `name:` property. You can give this any name you want, but no spaces or symbols.

The nice thing about naming each slide is that you can immediately go to

```
---
 YAML
---

name: about_me

# About Me

My name is Andy and I have two rescue dogs.

---

name: sadie

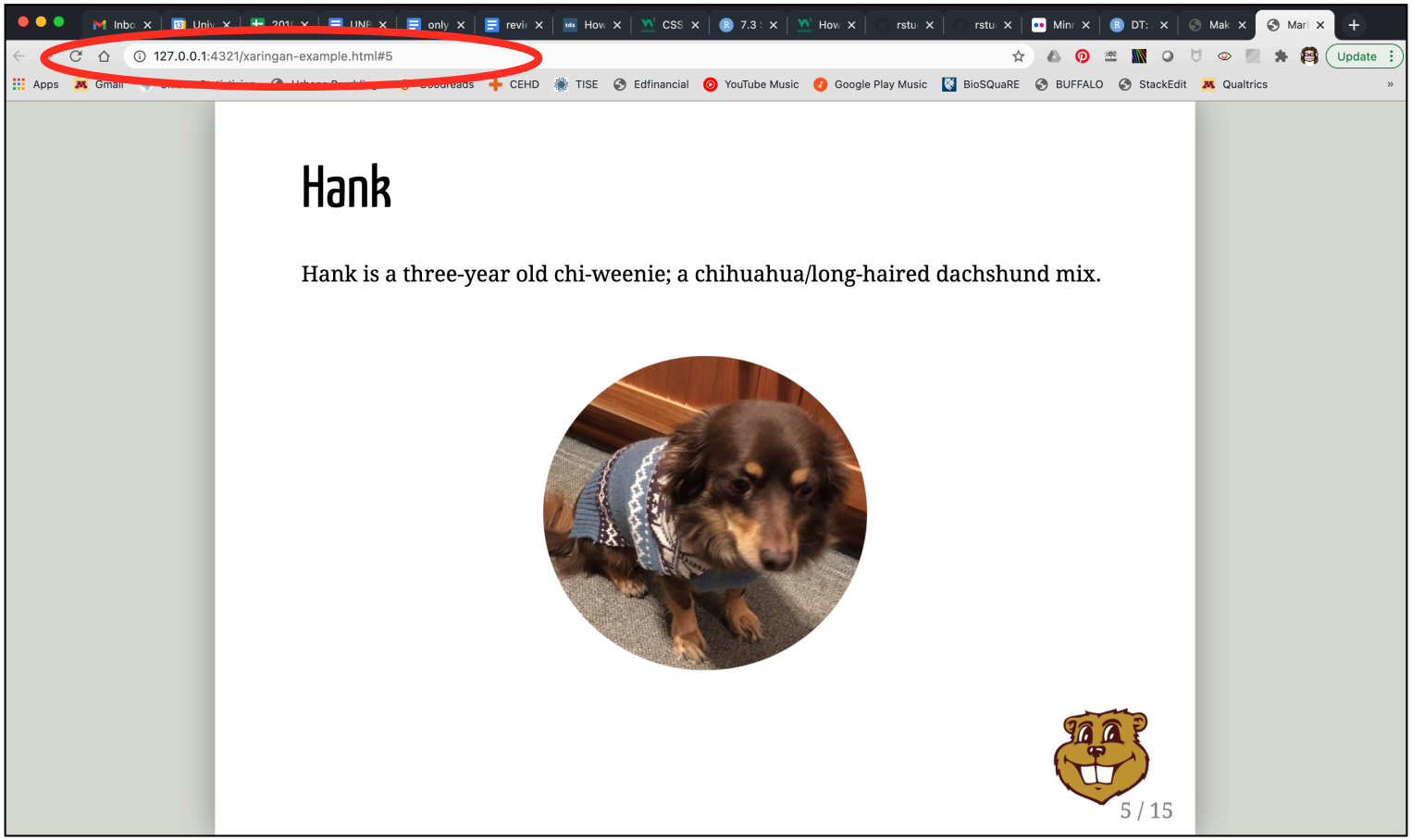
# Sadie

Sadie is a 10-year old sprollie; a springer spaniel/
border collie mix.

???

Sadie is white with black spots.
```

The nice thing about naming each slide is that you can immediately go to a slide in the browser by entering the name after the # in the URL.



For example, change the URL from `http://127.0.0.1:4321/xaringan-example.html#5` to `http://127.0.0.1:4321/xaringan-example.html#about_me`. That will bring you to the “About Me” slide.

Presenting Your Slides

You have probably seen that you can navigate through your slides with the arrow keys (⬅️ and ➡️).
Press h or ? to see a list of all the remark.js keyboard shortcuts.

Alison Hill (@apreshill) suggests this useful workflow for presenting; if you have two screens:

- Turn off display mirroring
- Press c to clone:

On the presentation screen
Move the cloned window to this screen;
Then press f for full-screen mode.

On your laptop screen
Press p for presenter mode on laptop.

Learning More

There are many more things you can do with xaringan. Here are a few resources to get you going on your learning journey:

- Hill, A. (2019). Meet xaringan: Making slides in R Markdown. Workshop presented at RStudio Conf.
- Chapter 7 of the R Markdown book introduces xaringan.
- Check out the xaringan slides that demonstrate the package features.