Git Refresher + Distill Blog Deployment

Maithreyi Gopalan

Week 7

Data viz in the wild

Songyi

Maithreyi Gopalan on Deck

Agenda

- Git Refresher - Using R-Studio and Terminal instead of GitKraken or other clients
- Websites w/Distill
  - Same sort of thing, but also w/deployment
- Wrap up loose ends with Flexdashboards
  - We'll create together, but also skim some slides
- Some customization w/CSS (including fonts)
  - I think there's a good chance we won't get to this today
  - If not, we'll come back to this on Week 9

Gitttttttt

Git vs. Github vs.GitKraken (or other clients)

https://docs.github.com/en/get-started/start-your-journey/about-github-and-git

Is Git insalled?

Go to the shell and check and request the path to your Git executable by typing

which git

/usr/bin/git

and git --version to see its version

If you are successful, that's great! You have Git already. No need to install!

If, instead, you see something more like git: command not found, then you need to install git

Introduce yourself to Git

- You can use the terminal directly by
  - git config --global user.name "Jane Doe"
  - git config --global user.email "jane@example.com"
  - git config --global --list
- Or you can use R-Studio and the usethis

Optional - Install a Git Client like GitKraken - I will not be showing this!!!

- https://happygitwithr.com/git-client

PAT for Https

- Generate youe PAT if you haven't done that
  - usethis::create_github_token()
  - or go to https://github.com/settings/tokens and click "Generate token"
  - Provide this PAT next time a Git operation asks for your password
- My recommendation - Get out ahead of this and store the PAT explicitly right now
  - In R, call gitcreds::gitcreds_set(), to get a prompt where you can paste your PAT
  - Paste the PAT in response to the dialogue in the console
  - You should be able to work with GitHub now, i.e. push and pull.

Connect to Github

- Create a new repo online in Github
- Clone that repo to your local computer
  - git clone https://github.com/maithgopalan/myrepo.git
- Make it your working directory
  - cd myrepo
- Make a minor change to your readme.md file and push it to your remote
- More info here

Websites w/Distill

Sub-agenda

- Introduce Distill
- Deployment

Learning objectives

- Get at least a basic site deployed

By the end of the day! You will have a site!

Distill

https://rstudio.github.io/distill/

Please follow along

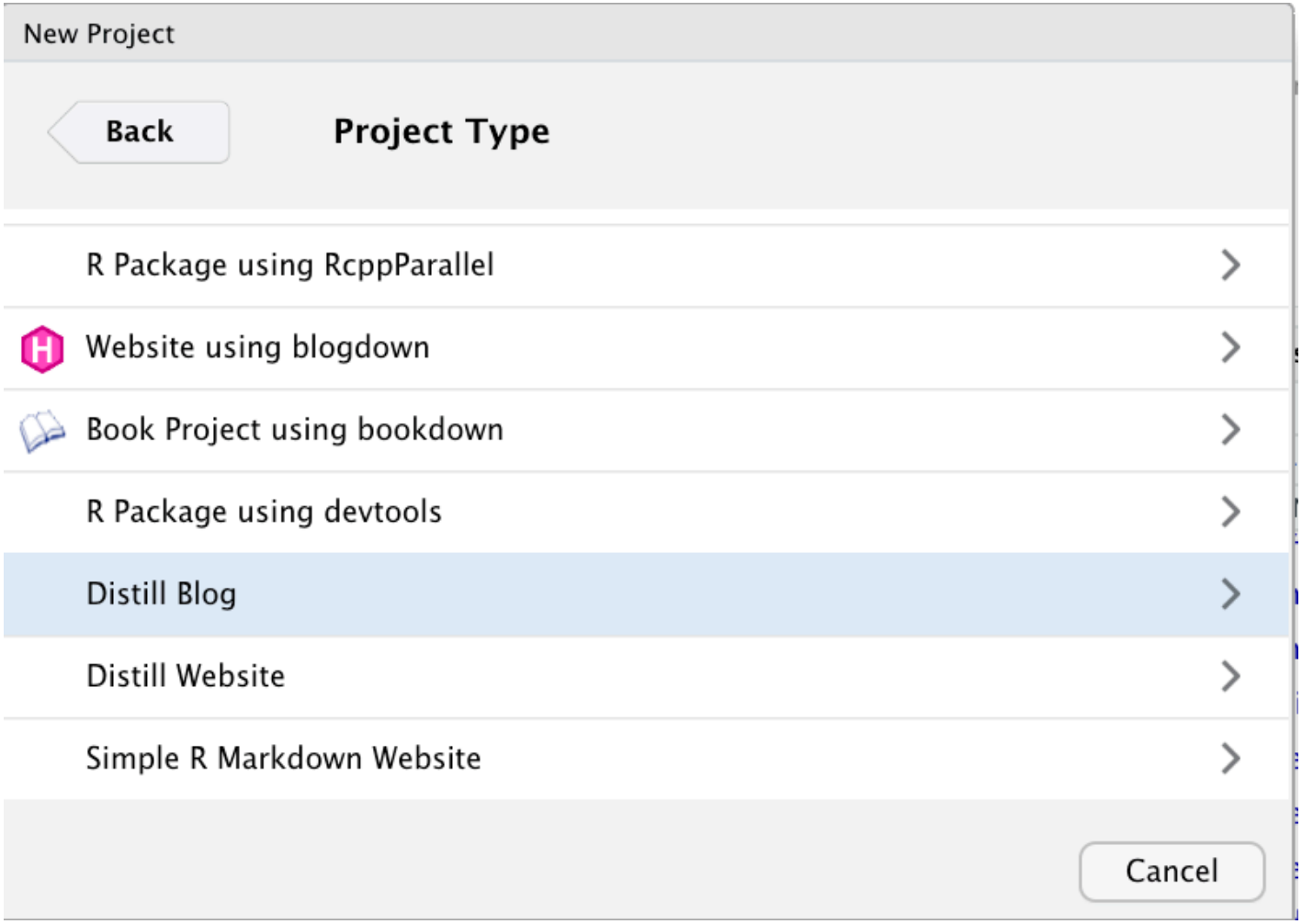
```r
install.packages("distill")

# or

#remotes::install_github("rstudio/distill")
```

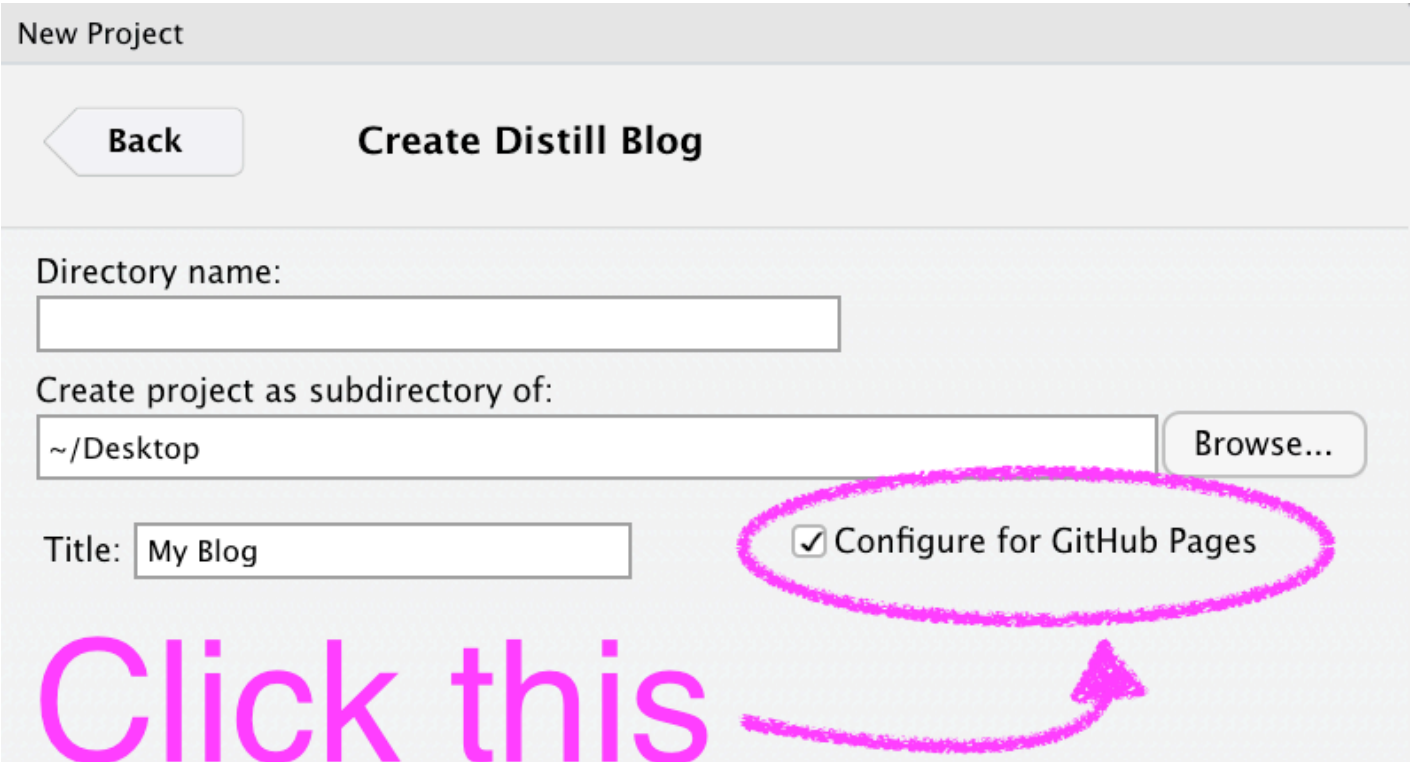
Back to RStudio

Create new project



The steps

- Create a new RStudio Project
- Select distill blog
- **Make sure** to Select "Configure for GitHub Pages"



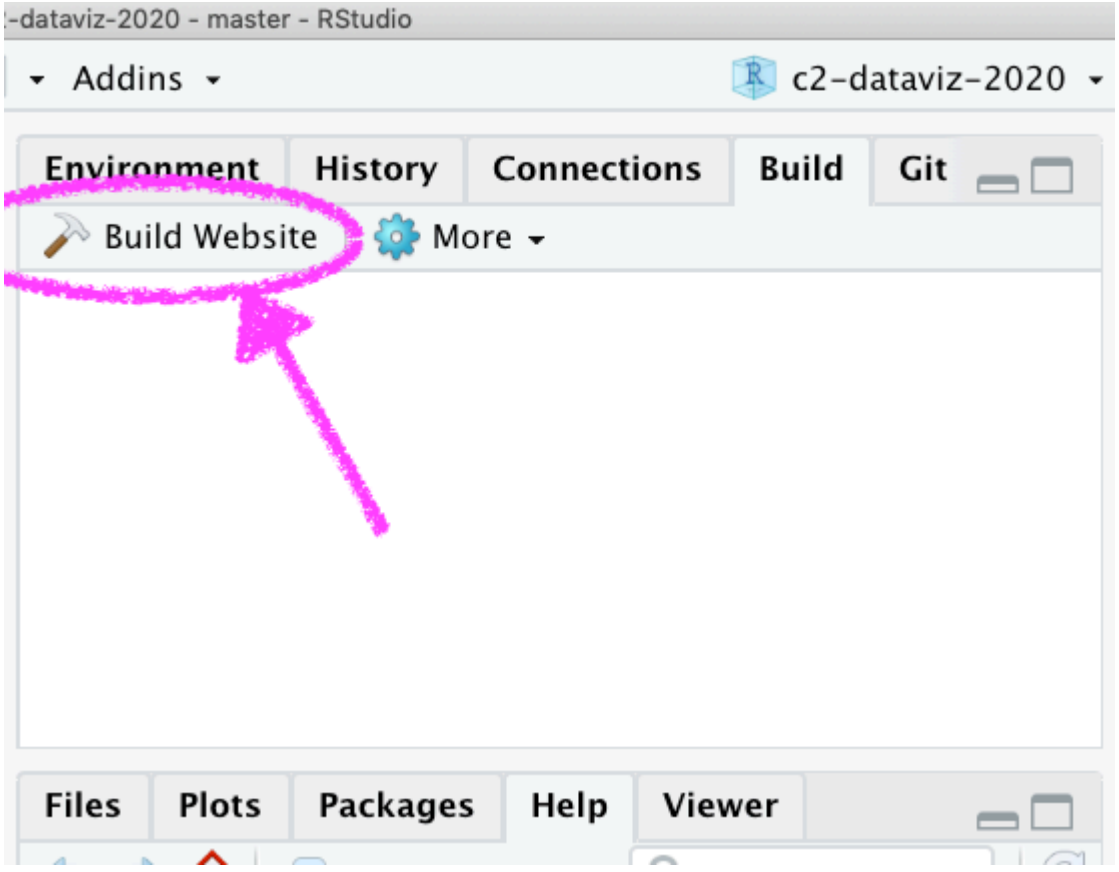
Customize

- Make changes to welcome.rmd
- Knit!!

Author a new article

- `distill::create_post()`
- Create another one!

Build your website

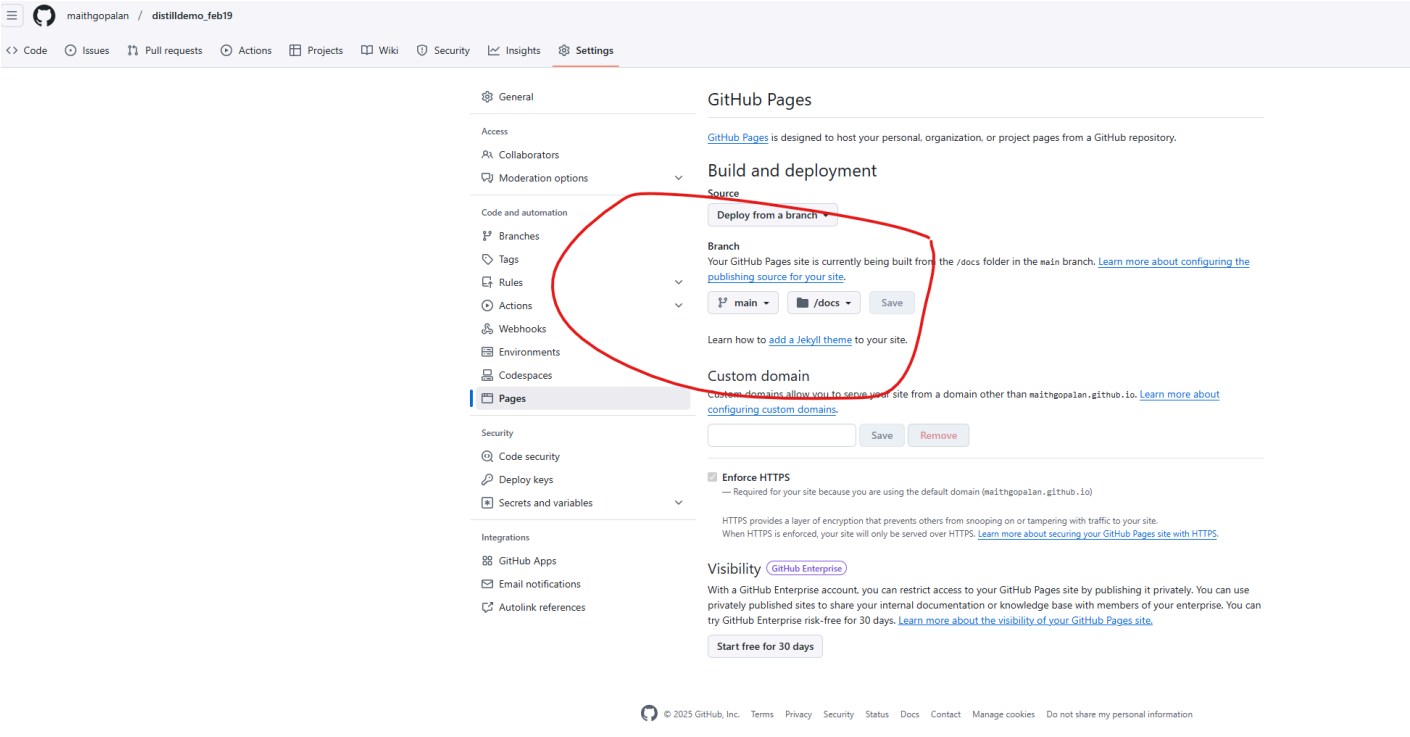


Connect to GitHub

Use the project-first workflow and publish the docs folder

[Demo]

Deplyment of ghpages



That's basically it!

Base URL

Once your site is deployed (or you know the link it will be deployed to), change the `base_url` in the `_site.yml`

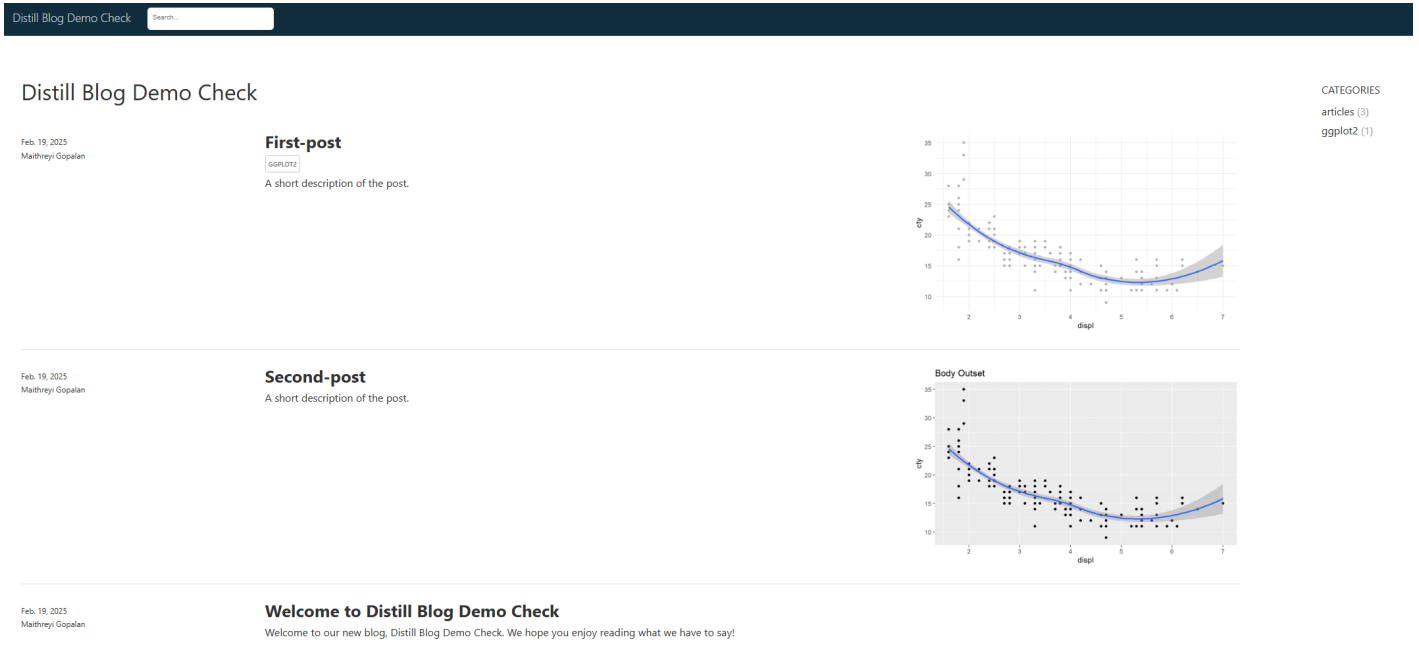
- Gives some nice sharing features (twitter cards)
- Allows you to use citations

A few additional features

Categories

- You make up the category names. Tag posts with those categories, and they will be linkable

```
---
categories:
  - ggplot2
---
```



Navigation

All controlled with `_site.yml`

- Let's add a github logo that links to our repo `

---

name: "distilldemo_feb19" title: "Distill Blog Demo Check" description: | Distill Blog Demo Check output_dir: "docs" base_url: https://maithgopalan.github.io/distilldemo_feb19/ navbar: right:

```
- text: "Home"
  href: index.html
- text: "About"
  href: about.html
- icon: fa fa-github
  href: https://github.com/maithgopalan/distilldemo_feb19
```

output: distill::distill_article

Youc can create drop-down menus - again edit the _site.yml

```
---
navbar:
  left:
    - text: "Labs"
      menu:
        - text: "Getting Started with R"
          href: "lab1.html"
        - text: "Visualizing Distributions"
          href: "lab2.html"
  right:
    - text: "Home"
      href: index.html
    - text: "About"
      href: about.html
    - icon: fa fa-github
      href: https://github.com/maithgopalan/distilldemo_feb19
---
```

Drafts

If you want to work on a post for a while without it being included in your website, use `draft = TRUE`

```
distill::create_post("My new post", draft =
TRUE)
```

```
---
title: "My work on Lab 3"
description: |
  This lab was hard!
draft: true
---
```

Figures

Change figure options with chunk options

- `layout = "l-body"` (default)
- `layout = "l-body-outset"`
- `layout = "l-page"`
- `layout = "l-screen"`
  - `layout = "l-screen-inset"`
  - `layout = "l-screen-inset shaded"`

Try it out!

Additional figure options

- Rather than using `![]()`, you can use `knitr::include_graphics()` to have the same options.
- Use `fig.cap` in chunk options to give nice figure captions.
- Note these options should work for tables as well

Side notes

```
<aside>
This is some text that will appear in the margin - similar to Tufte's style. It is
</aside>
```

You can also use this to show small plots

```
<aside>

ggplot(mtcars, aes(mpg)) +
  geom_histogram() +
  labs(title = "Distribution of Miles Per Gallon")

</aside>
```

Customizing the theme

Use `distill::create_theme("style")`

- Creates a `style.css` file (or whatevs you want to call it in the above)
- Modify `_site.yml` to

```
output:
  distill::distill_article:
    css: style.css
```

- Modify small elements

```
.distill-site-nav {
  color: rgba(255, 255, 255, 0.8);
  background-color: #455a64;
  font-size: 15px;
  font-weight: 300;
}
```

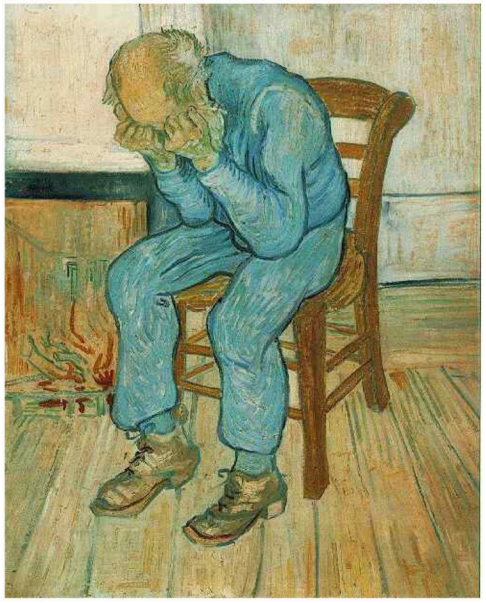
becomes

```
.distill-site-nav {
  color: rgba(255, 255, 255, 0.8);
  background-color: #FF5FDD;
  font-size: 15px;
  font-weight: 300;
}
```

This can be fun!

Just be careful not to go too far: from Yihui

Debugging CSS, van Gogh (1890)



Dashboards!

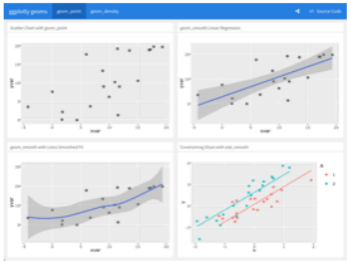
The definitive source!

https://rmarkdown.rstudio.com/flexdashboard/

flexdashboard for R | Home | Using | Shiny | Layouts | Examples

flexdashboard: Easy interactive dashboards for R

- Use R Markdown to publish a group of related data visualizations as a dashboard.
- Support for a wide variety of components including htmlwidgets; base, lattice, and grid graphics; tabular data; gauges and value boxes; and text annotations.
- Flexible and easy to specify row and column-based layouts. Components are intelligently re-sized to fill the browser and adapted for display on mobile devices.
- Storyboard layouts for presenting sequences of visualizations and related commentary.
- Optionally use Shiny to drive visualizations dynamically.



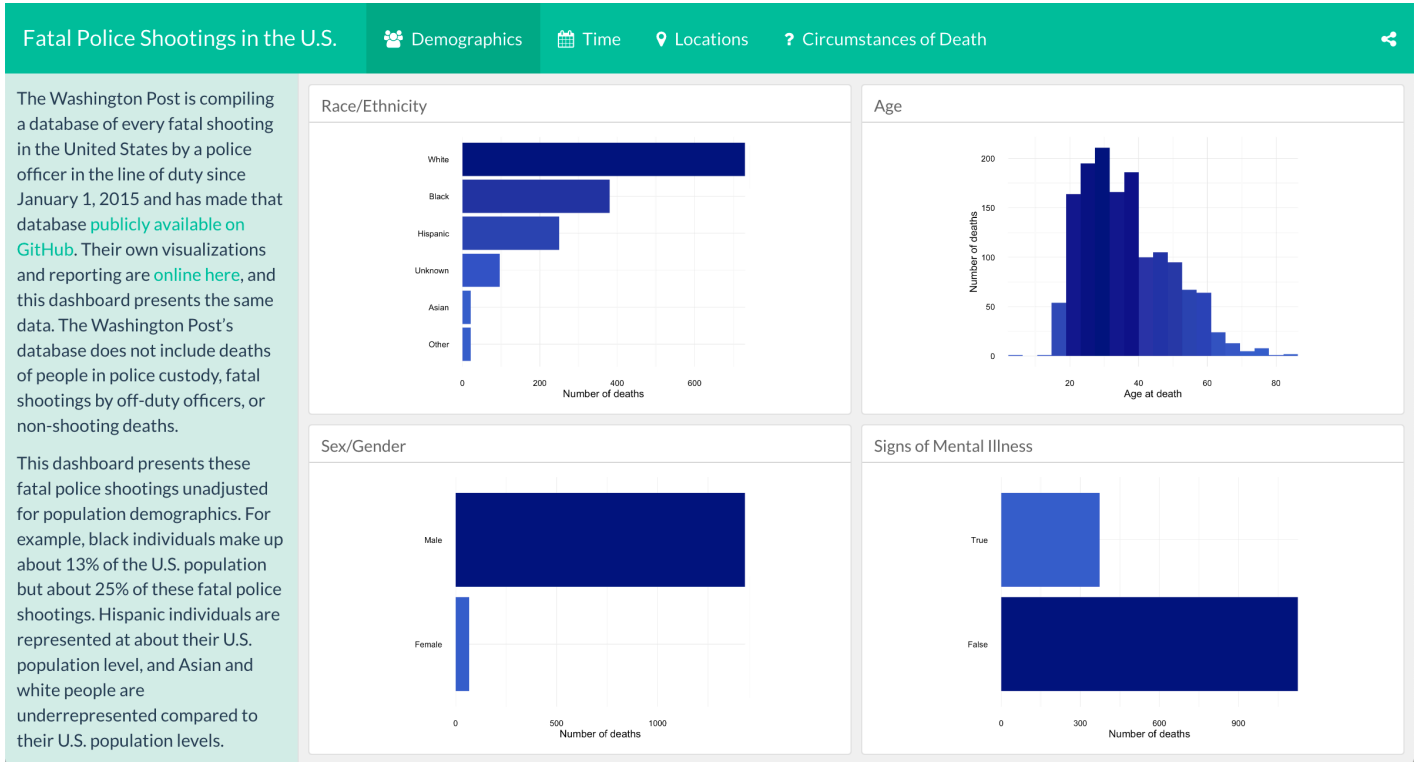
Getting Started

Install the **flexdashboard** package from CRAN as follows:

```
install.packages("flexdashboard")
```

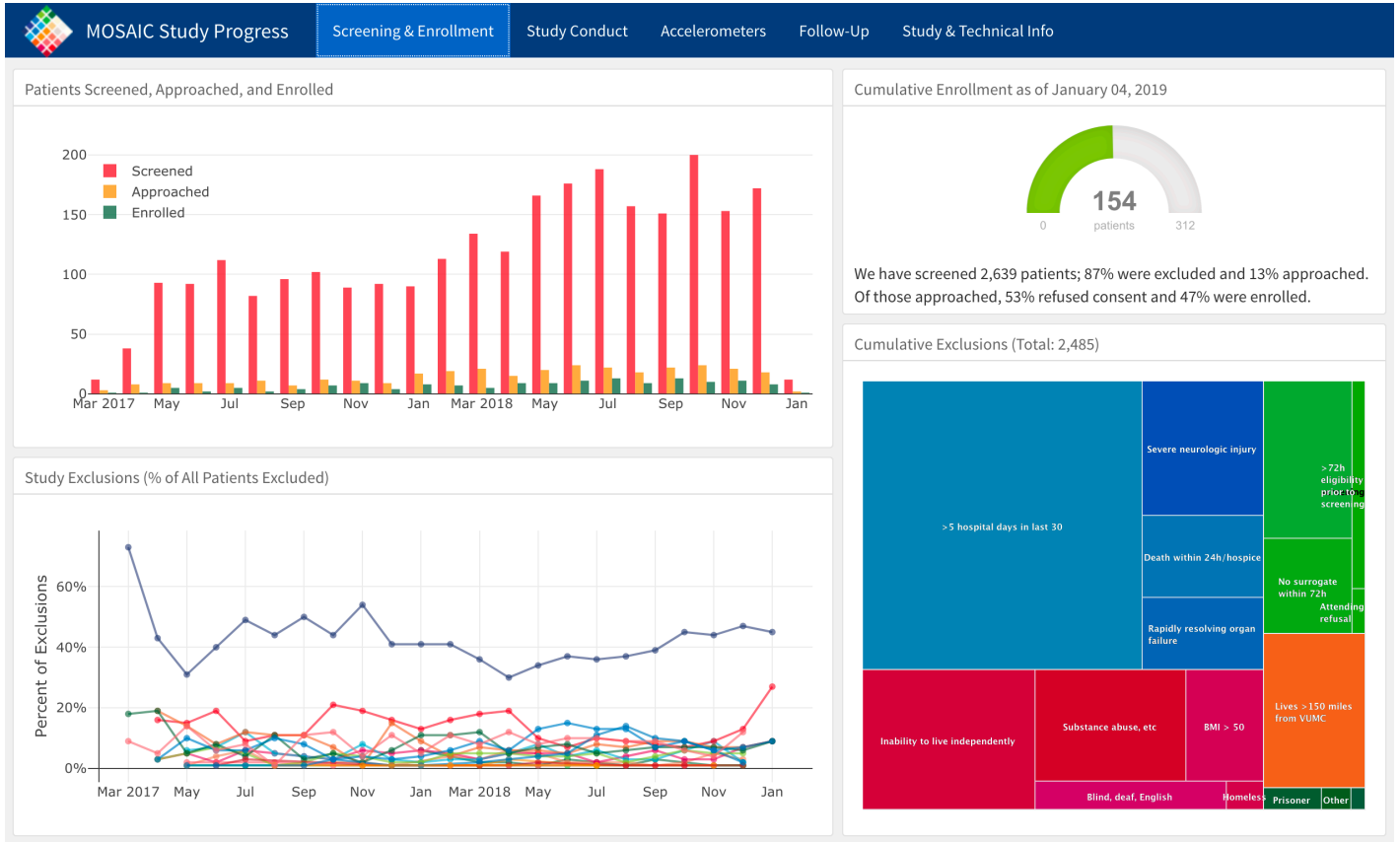
{width="75%"}

Example



By Julia Silge (see the blog post, dashboard, and source code)

Credit: This example from Alison Hill's rstudio::conf(2019L) class



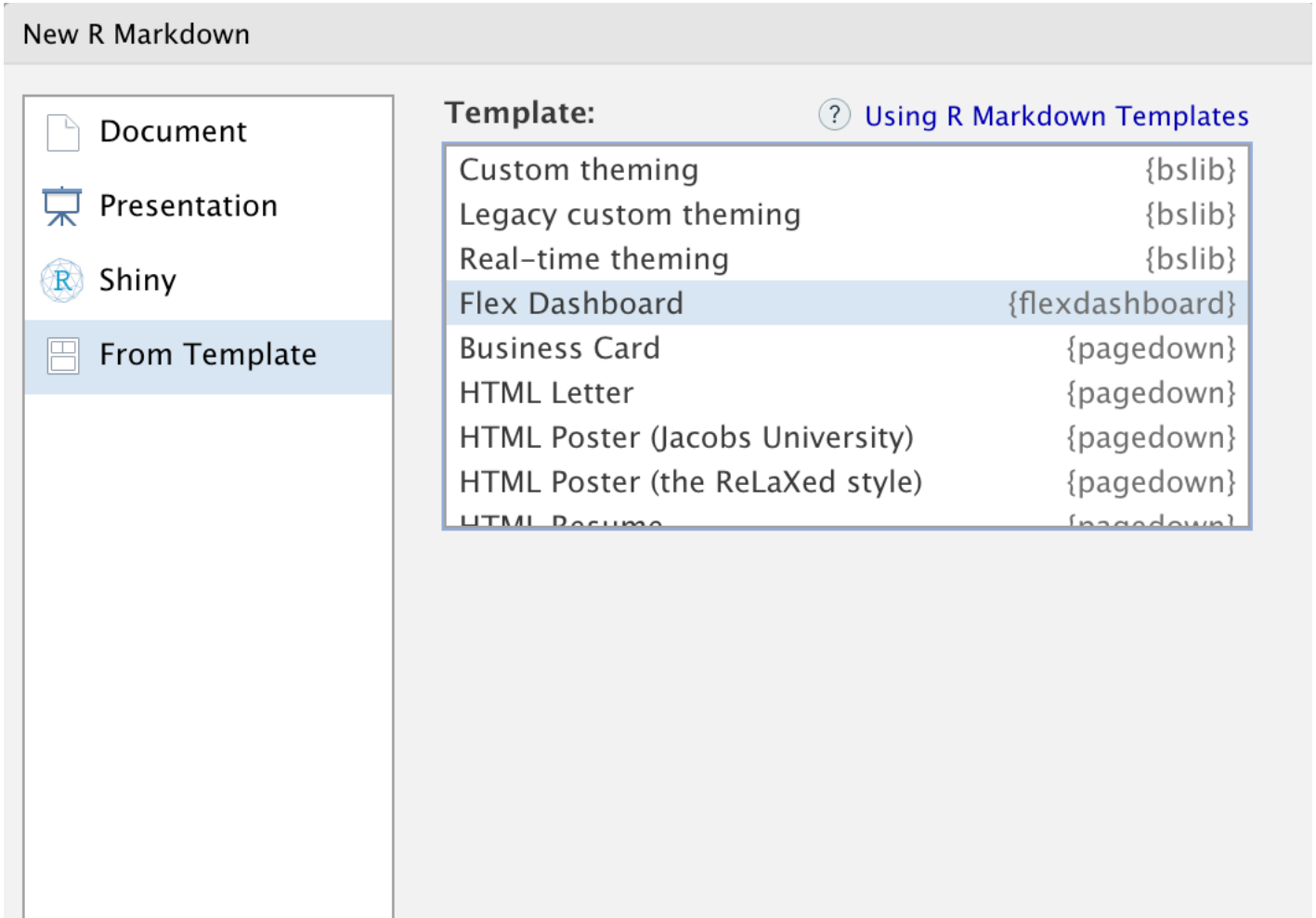
{width="75%"}

By Jennifer Thompson (see the blog post, dashboard, and source code)

Credit: This example from Alison Hill's rstudio::conf(2019L) class

Getting started

```
#install.packages("flexdashboard")
```



Do it

knit right away

Add some plots

Play

05:00

Columns

- Define new column with

```
Column
---------------------------------
```

- Optionally specify the width with `{data-width}`
- Annoyingly, be careful with spacing!

`{data-width=650}` will work

but

`{data-width = 650}` will not work

New squares

Square title< r code chunk >

- Each time you add a square it will split the area evenly among all the squares

Thinking in rows

- Change the YAML to

```
output:
  flexdashboard::flex_dashboard:
    orientation: rows
```

- Each `###` will then create a new column
- Add new rows with

```
Row
-----------------------------------
```

- Modify height with `{data-height=XXX}`

Pages

You can easily specify multiple pages by just specifying a Level 1 Header

```
# Page 1

Column {data-width=650}
----------------------------
### Chart A
< r code>

Column {data-width=350}
----------------------------
### Chart B
< r code>

### Chart C
< r code>

# Page 2
```

A brief aside on interactivity

- Things like `reactable::reactable` and `plotly::ggplotly` can help give your dashboard some nice interactvity.

Steps to interactivity

With multipage layouts

Add `runtime: shiny` to your YAML

```
---
title: "My amazing dashboard"
runtime: shiny
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
---
```

Save your interactive piece into an object, and call the corresponding `render*` fucntion.

```
p <- ggplot(...)
renderPlotly(p)
```

```
tbl <- reactable(...)
renderReactable(tbl)
```

Sidebar

- Julia Silge's had a nice sidebar where she explained things about the flexdashboard... You can have this too!

```
Sidebar Title {.sidebar}
--------

Your text here. You can use markdown syntax, including
[links](http://blah.com), *italics*, **bolding**, etc.
```

- Multiple pages? Just change the separator to keep it there

```
Sidebar {.sidebar}
============

Your text here. You can use markdown syntax, including [links](h
```

Tabsets

This is actually a standard R Markdown feature, but you can use it with flexdashboards as well

```
Column {.tabset}
-----------------------------------------------------------------------

### Chart 1
< r code>

### Chart 2
< r code>

### Data Table
< r code>
```

No comma between multiple column arguments

Good

```
Column {.tabset data-width=650}
```

bad

```
Column {.tabset, data-width=650}
```

Icons

- Probably not the most important thing, but fun
- Use Font awesome!

```
# Years {data-icon="fa-calendar"}
```

HTML Widgets

Add a touch of interactivity

- Plenty of HTML widgets for R out there (see https://www.htmlwidgets.org/showcase_leaflet.html)
- {plotly} is cool

```
#install.packages("plotly")
library(plotly)
p <- ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  geom_smooth()

ggplotly(p)
```

Including Text

- If you want to include text about an overall figure, just put the text in the R Markdown doc like you normally would

```
# Base {data-icon="fa-calendar"}
Here's a description about the plot that follows

### A base R plot
< r code>
```

What if you have tabsets?

- Works great if you want to describe all the plots/tables/content in the tabset
- If you want to provide text for an individual plot, use >

```
Column {.tabset data-width=350}
-------------------

This text will describe the full tabset

###  Chart 1
< r code>

> Here's some text for Chart 1

###  Table 1
< r code>

> Here's some text for Table 1
```

Storyboarding

- A little bit advanced, but pretty cool
- First, change the YAML

```
output:
  flexdashboard::flex_dashboard:
    storyboard: true
```

```
# Method {.storyboard}

### Sample Descriptives {data-commentary-width=400}
< r code>

***

This is some text describing what's going on with the sample, and

### Correlation Matrix {data-commentary-width=200}
< r code>

***

There is less to say here so I made the commentary box smaller

# Results {.storyboard}

### Plot 1 {data-commentary-width=600}
< r code>

***

Lots to say here. There is important

### Plot 2 {data-commentary-width=200}

***

Move along
```

Customization

- Add font-awesome stuff
- Change the theme

```
flexdashboard::flex_dashboard:
  theme: readable
```

CSS

More on this later slides

Change the navigation bar to bright pink with thin blue border

```
.navbar-inverse {
  background-color: #FE08A5;
  border-color: #0822FE;
}
```

Save the previous code in "custom.css" then specify in the YAML

```
flexdashboard::flex_dashboard:
  css: custom.css
```

Making sure "custom.css" is in the same directory as your flexdashboard Rmd.

Add a logo and favicon

```
output:
  flexdashboard::flex_dashboard:
    logo: logo.png
    favicon: favicon.png
```

Fonts

General advice

- Match your plot fonts to your text body font
- Use different fonts to distinguish things
  - Specifically code
  - Consider for different heading levels
- **Always** choose a sans-serif font for code
- Explore and try - it makes a big impact on the overall look/feel
- Try not to get sucked into too deep of a rabbit hole

{ragg}

```
#install.packages("ragg")
```

Alternative device to Cairo, png, etc.

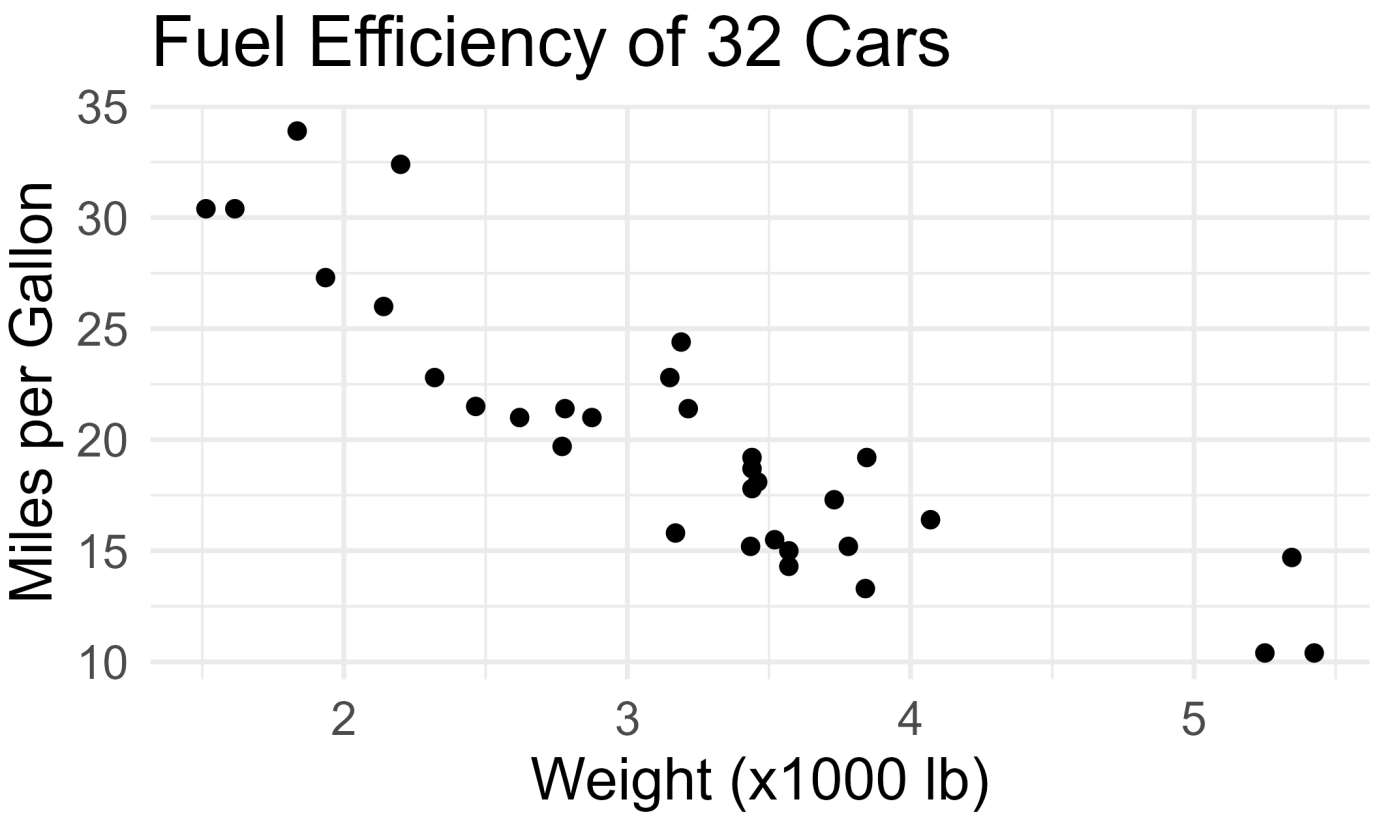
See the announcement here

After install, be sure to set *Global Options* > *General* > *Graphics* to *AGG*

Use with RMarkdown with `knitr::opts_chunk$set(dev = "ragg_png")`

Will automatically detect fonts you have installed on your computer

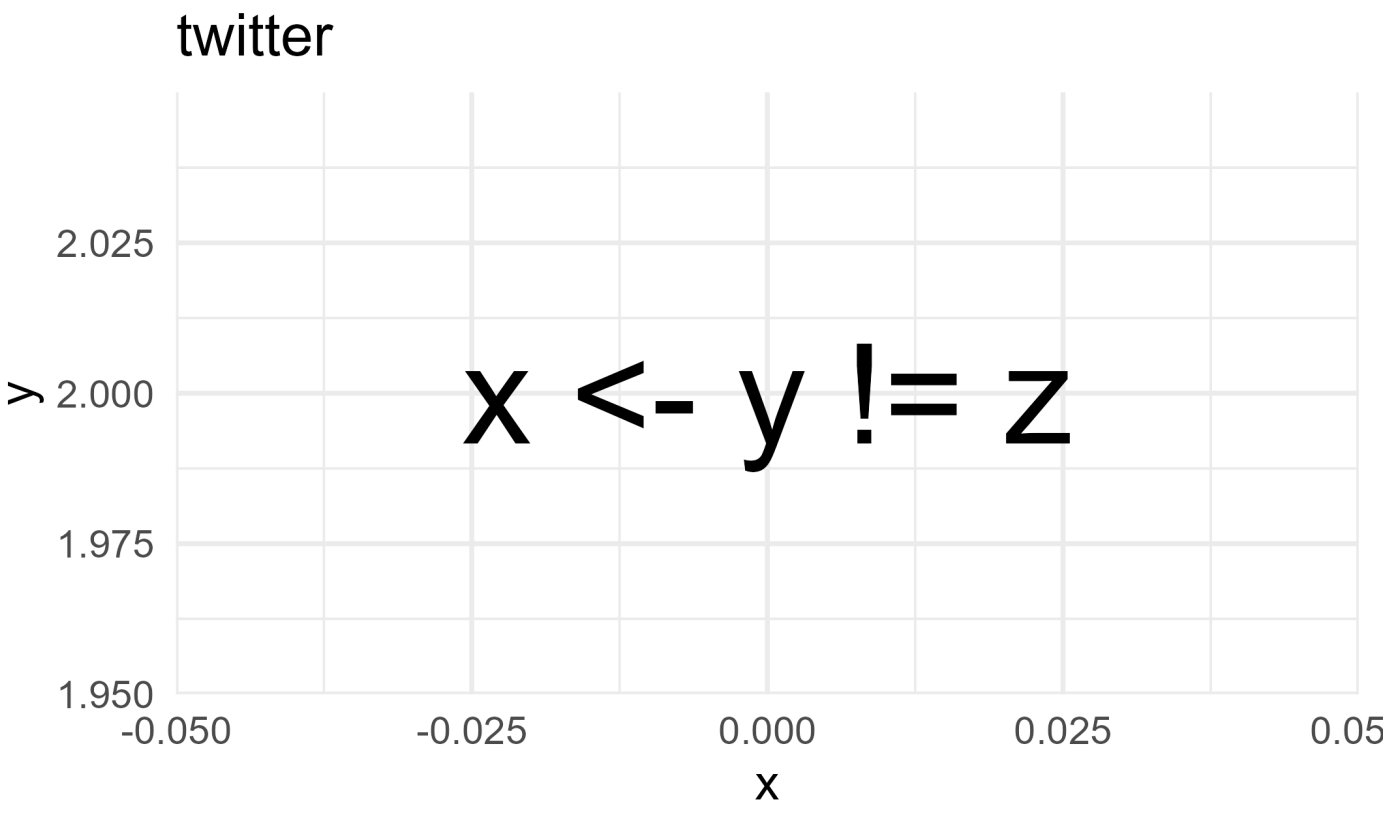
```r
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  labs(title = "Fuel Efficiency of 32 Cars",
       x = "Weight (x1000 lb)",
       y = "Miles per Gallon") +
  theme(text = element_text(family = "Luminari", size = 30))
```



Support for lots of things!

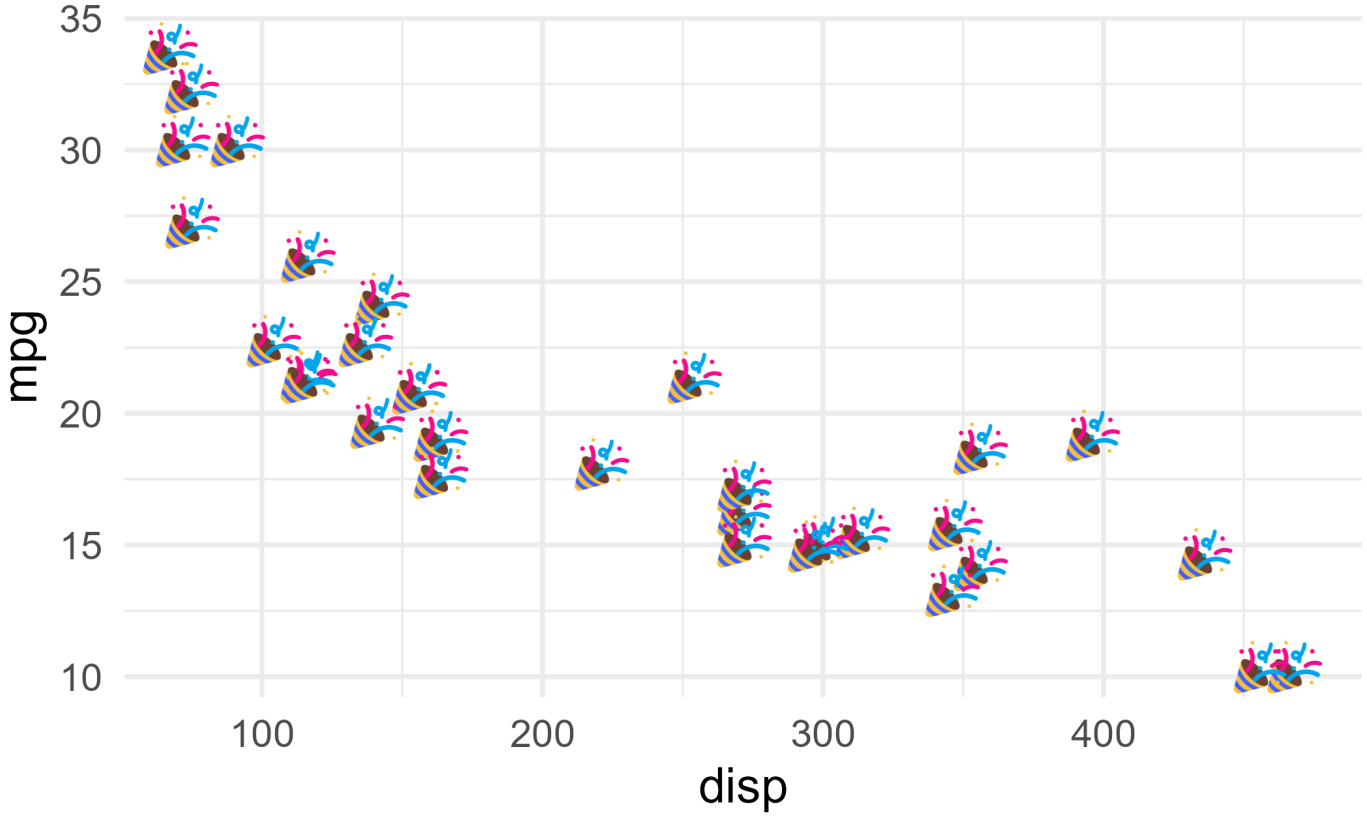
Ligatures and font-awesome icons

```
ggplot() +
  geom_text(
    aes(x = 0, y = 2, label = "x <- y != z"),
    family = "Fira Code"
  ) +
  labs(title = "twitter") +
  theme(
    plot.title = element_text(
      family = "Font Awesome 5 brands"
    )
  )
```


twitter
x <- y != z
2.025
2.000
1.975
1.950
y
-0.050
-0.025
0.000
0.025
0.05
x

emojis

```
ggplot(mtcars, aes(disp, mpg)) +
  geom_text(label = "🎉",
            family = "Apple Color Emoji",
            size = 10)
```



Google fonts

https://fonts.google.com

- Open source, designed for the web
- Good place to explore fonts
- Can be incorporated via the `{showtext}` package!

{showtext} example

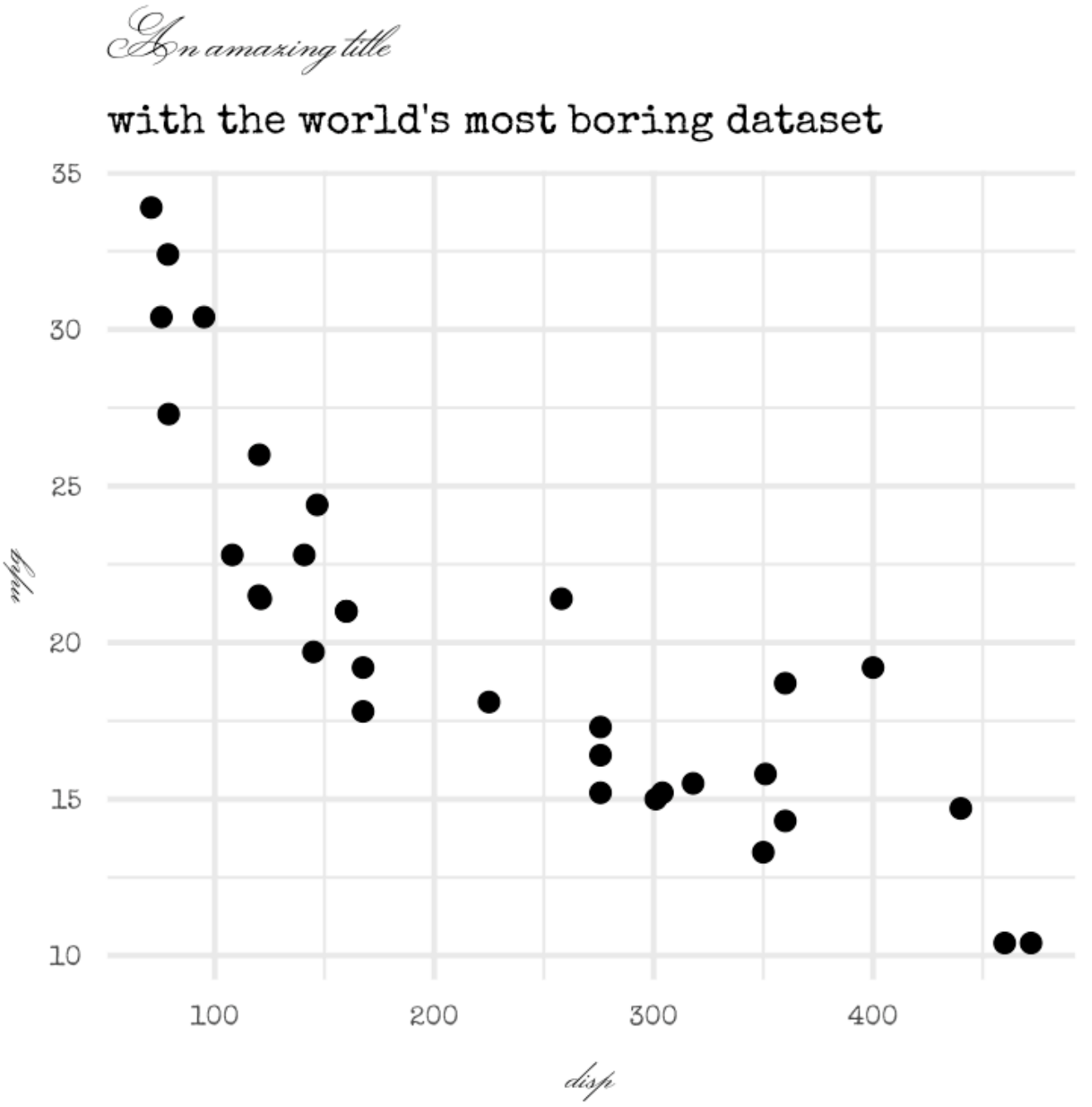
```r
devtools::install_github("yixuan/showtext")

library(showtext)

font_add_google('Monsieur La Doulaise', "mld")
font_add_google('Special Elite', "se")

showtext_auto()

ggplot(mtcars, aes(disp, mpg)) +
  geom_point() +
  labs(title = "An amazing title",
       subtitle = "with the world's most boring dataset") +
  theme(plot.subtitle = element_text(size = 18, family = "se"),
        plot.title = element_text(size = 22, family = "mld"),
        axis.title = element_text(size = 18, family = "mld"),
        axis.text.x = element_text(size = 12, family = "se"),
        axis.text.y = element_text(size = 12, family = "se"))
```


An amazing title
with the world's most boring dataset
35
30
25
20
15
10
mpg
100
200
300
400
disp

Practice

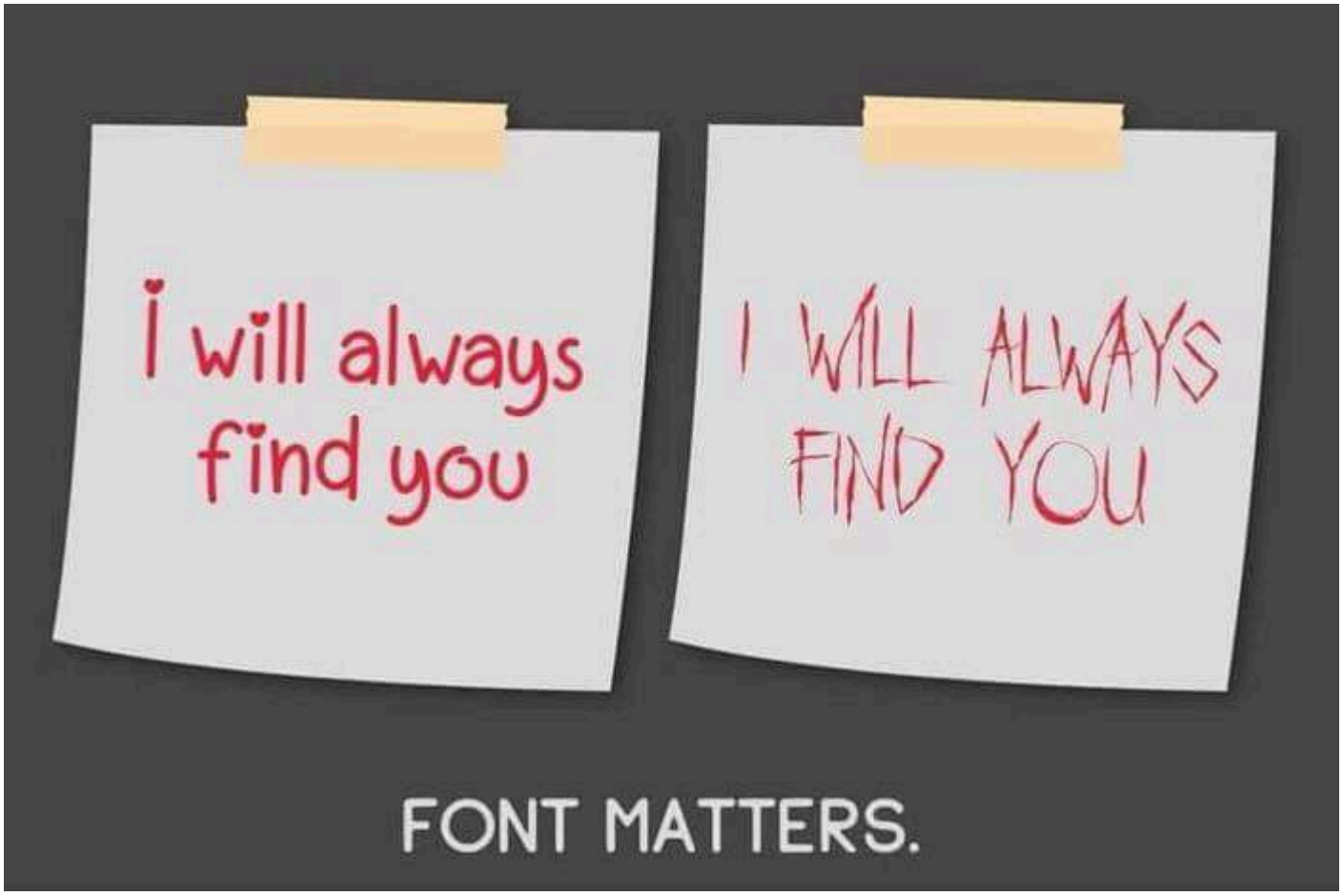
- Create a simple plot
- Change the font to something on your computer (e.g., "Arial")
- Try importing and using a google font with **showtext**
- Try using different fonts for the title and subtitle

05:00

Why fonts matter

A few examples of epic fails

Megaflicks - LOL


i will always find you
I WILL ALWAYS FIND YOU
FONT MATTERS.


You'll Always
Be Mine
YOU'LL
ALWAYS
BE
MINE

Quick aside

Change the font of your R Markdown!

Create a CSS code chunk - write tiny bit of CSS - voila!

```
@import url('https://fonts.googleapis.com/css?family=Akronim&dis

body {
  font-family: 'Akronim', cursive;
}
```

See the CSS slides for more information.

Render!

Untitled

Daniel Anderson

2/18/2020

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For
http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R cod
chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

Aside

I actually did this for the table slides to make them a bit smaller!

```
```{css echo = FALSE}
table {
  font-size: 1rem;
}
```
```

Resource for learning more

- I'm not an expert on fonts. I have mostly just picked what looks nice to me.
- Consider the accesibility of the font (good resource here)

Best I've heard of is practical typography

BUTTERICK'S
PRACTICAL
TYPOGRAPHY

2ND EDITION

Identify fonts

Use others work to help you - I found the font for these slides from Daniel's theme and he used one that he liked.

Use google chrome's developer tools to help! Also consider downloading fonts (from google or wherever) and using them directly.

Check out this great blog post by June Choe.

Next time

Geospatial visualizations

We may come back to more styling of your website/CSS stuff next week also