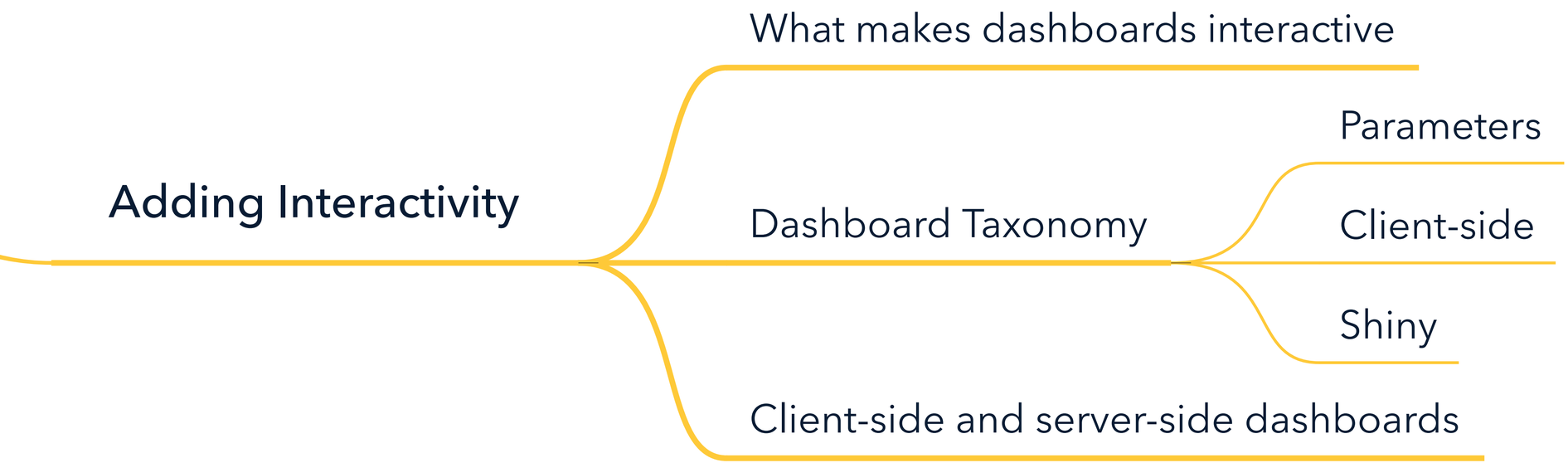
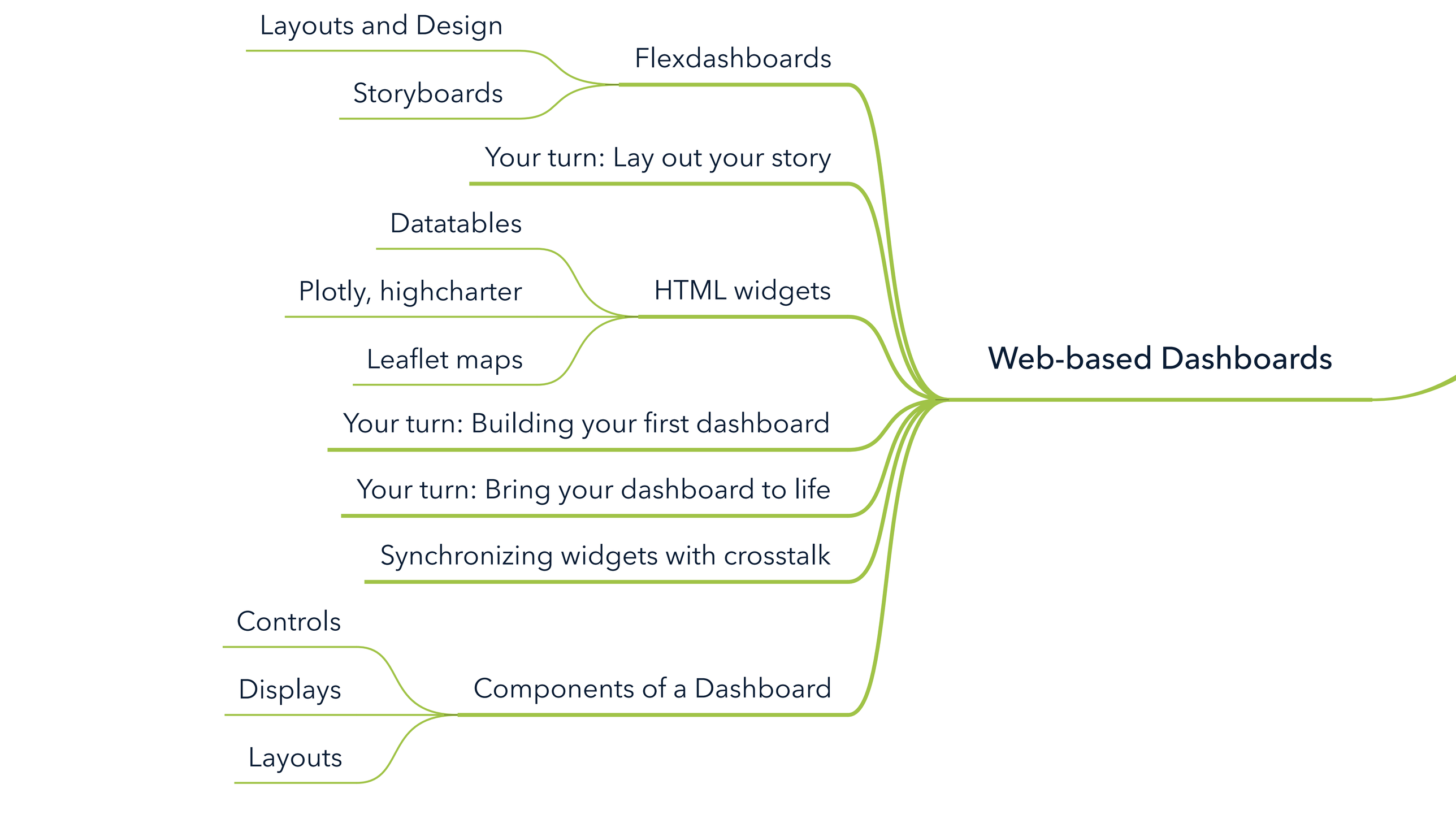
R Markdown and Interactive Dashboards
Web-based Dashboards

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID


Adding Interactivity
What makes dashboards interactive
Dashboard Taxonomy
Parameters
Client-side
Shiny
Client-side and server-side dashboards

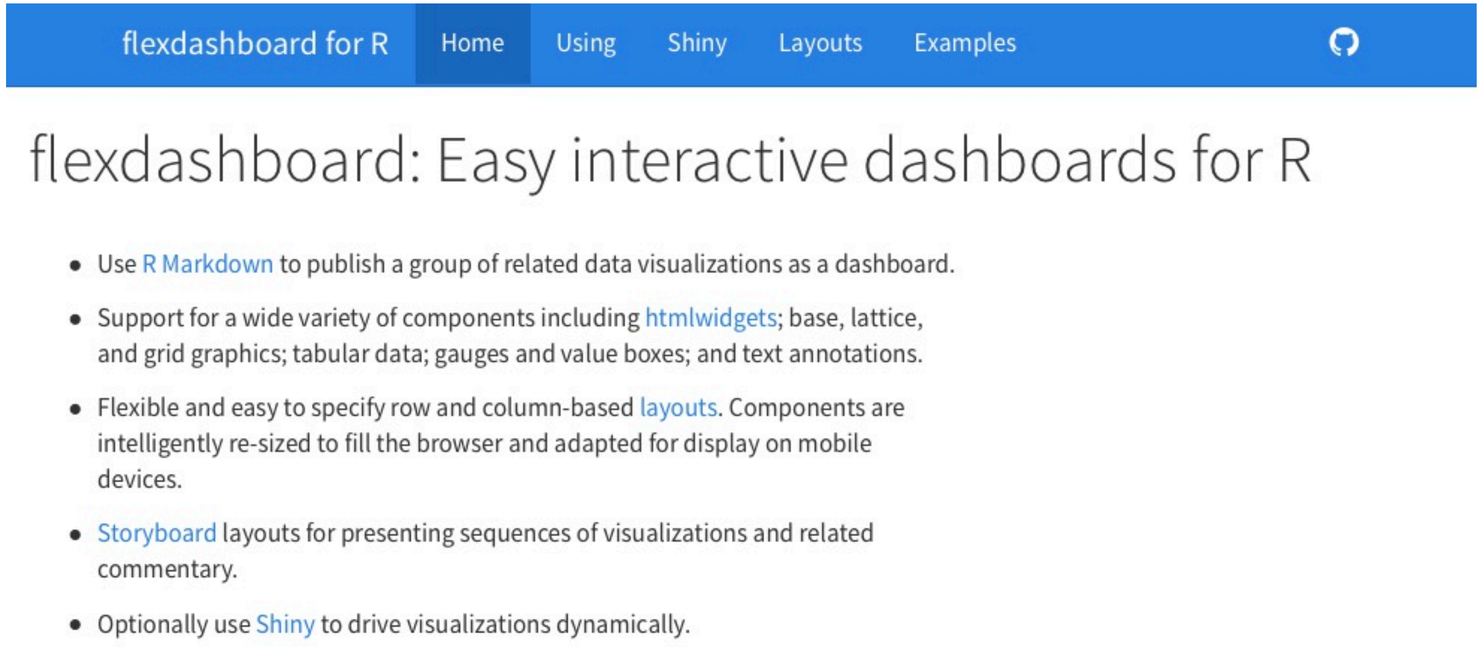

Web-based Dashboards
Flexdashboards
Layouts and Design
Storyboards
Your turn: Lay out your story
HTML widgets
Datatables
Plotly, highcharter
Leaflet maps
Your turn: Building your first dashboard
Your turn: Bring your dashboard to life
Synchronizing widgets with crosstalk
Components of a Dashboard
Controls
Displays
Layouts

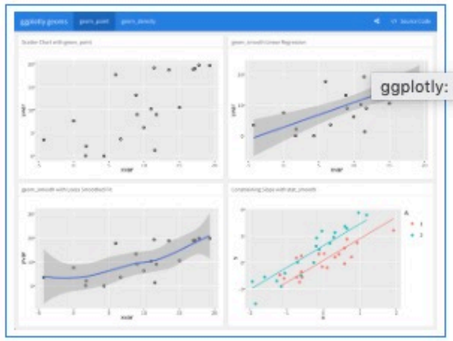
Review Quiz

- What are the 3 types of interactivity?
- Which technologies go with which types?

Flexdashboard Reference

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





Github repo

lio.cloud workspace: rstd.io/RMAID

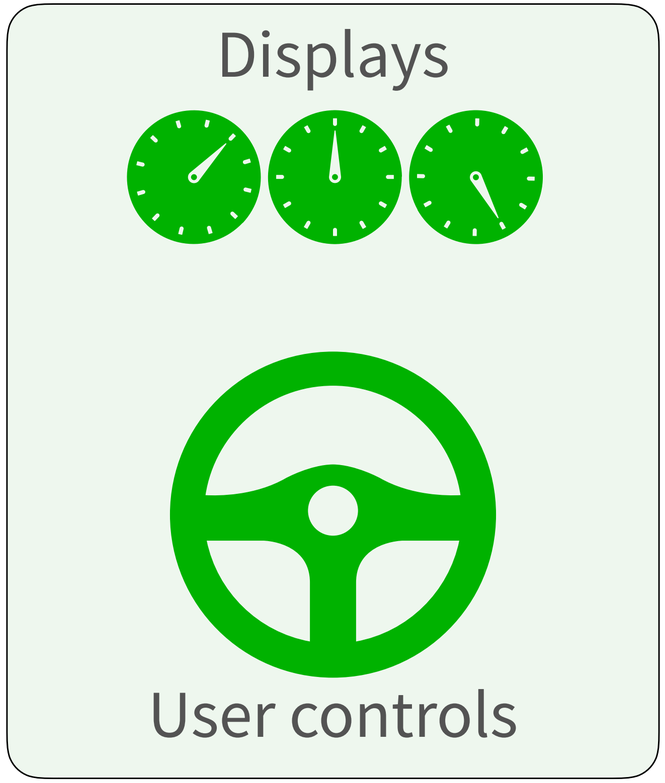
Dashboard Components

What are the common building blocks of a dashboard?

Hint: the fact that these two dashboards look different is also a building block.

05:00

Dashboards Are Composed Of...



Layout

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Dashboards have three building blocks:

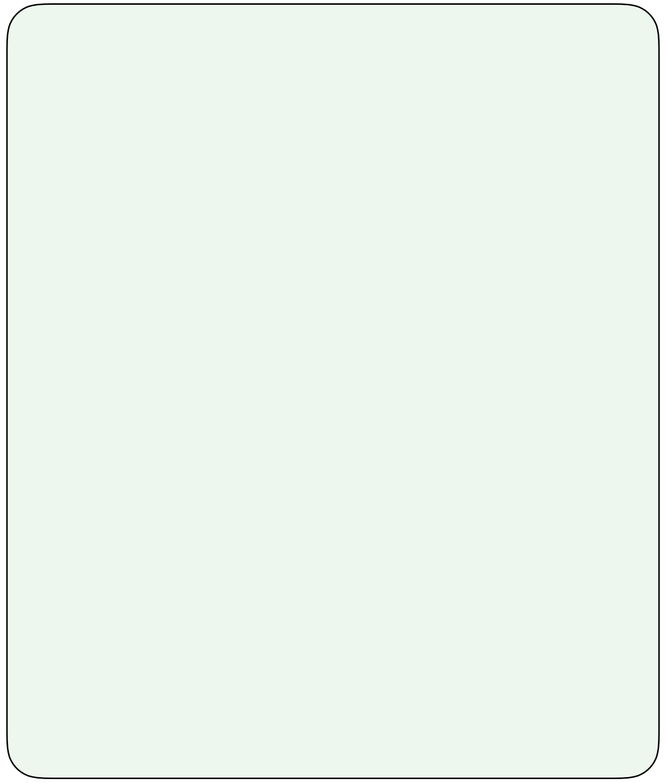
1. Controls

2. Displays

3. Layouts

Let's Start With...

- Allows exploration
- Engages the user
- Increases retention



Layout

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Despite Standardization, Car Dashboard Design Is Essential To The Product

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

R Markdown layouts use header information to mark rows and columns on pages

Layout Interpretations

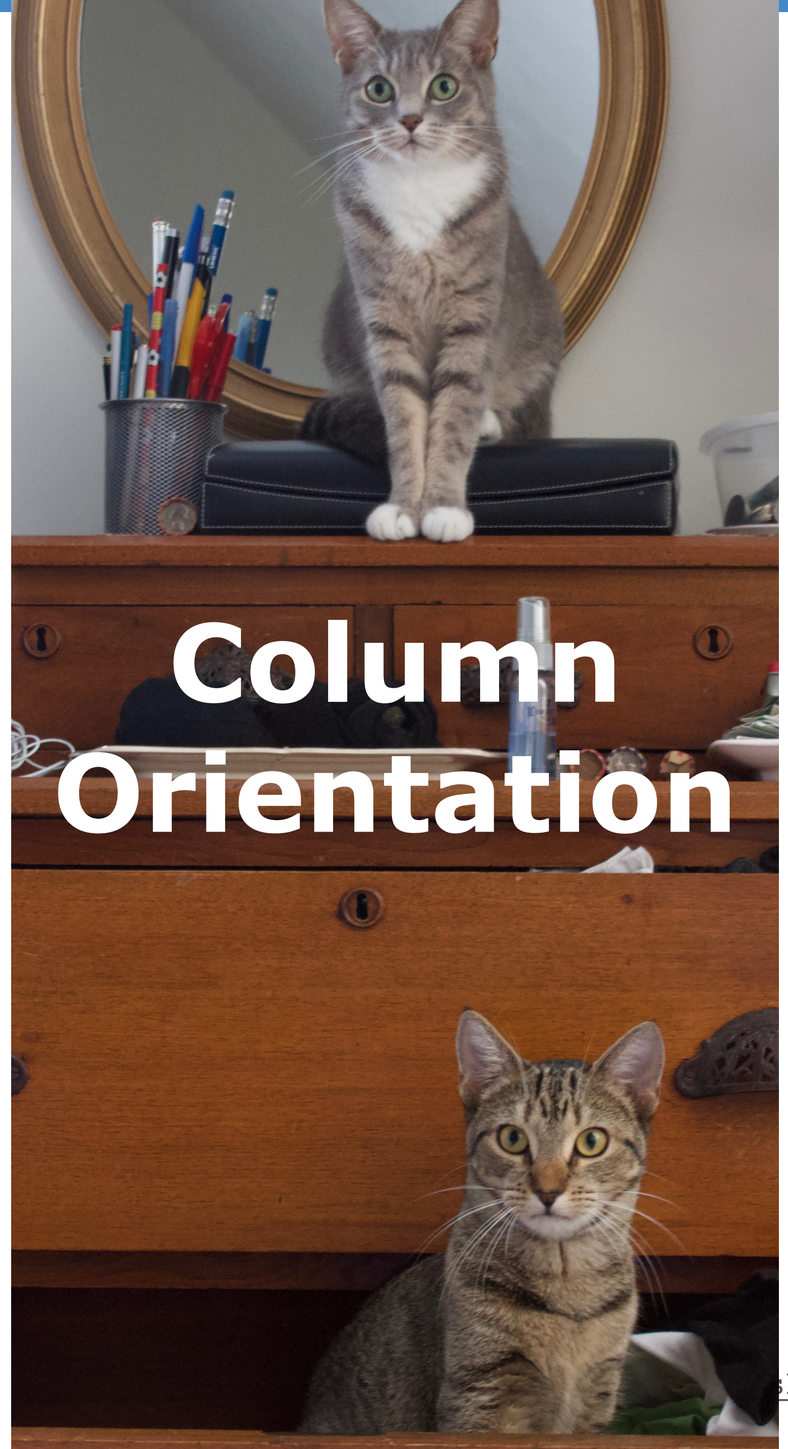
R Markdown Symbol	Alternate Symbol	`flexdashboard orientation: rows`	`flexdashboard orientation: columns`
#	===	Top level page	Top level page
##	----	New row	New column
###		New column in row	New row in column

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Your First Decision: Row Or Column Orientation?

- Our layout maps the one-dimentional R Markdown code onto a grid.
- Your `orientation` YAML parameter determines whether you want your level 2 sections to be laid out on the grid according to columns or rows.



Column Orientation

Section 1

Section 2

Section 3

or

Section 1

Section 3

Section 2

Section 4

Row Orientation

Section 1 | Section 2

Section 3 | Section 4

Section 1

Section 2 | Section 3

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Exercise 41

rstudio::conf

Play With Dashboard Orientation

1.Open the project **04-Web-Based-Dashboards**
2. Click on **`41-layout-column-orientation.Rmd`** in the Files pane to open that file
3. Knit the file and observe the result
4. Change the YAML line **`orientation: columns`** to **`orientation: rows`** and re-knit.
5. Try adding a **`### Text Description`** row at the bottom of column 1 and putting some text below it. Knit the result.
6. Try improving the esthetics of the result by adding **`{row-height=250}`** to **`### Text Description.`** Play with the row-height value for the best look to your eyes.
7. Add the text **`{.tabset .tabset-fade}`** to the end of the text ## Column 2 (which is actually row 2 now). What has changed?

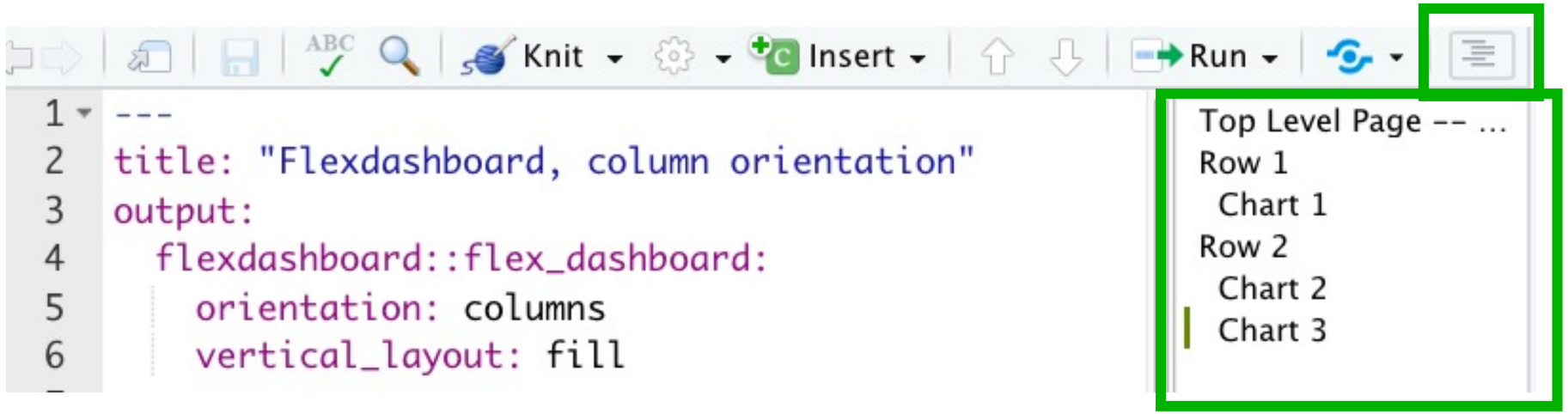
10m 00s

Observations

- Layout documentation is at

https://rmarkdown.rstudio.com/flexdashboard/layouts.html#overview

- You may find the RStudio IDE outline button useful for keeping track of your layout hierarchy.



Observations (continued)

- You can create multiple pages by adding more level 1 headers (`# header`).
- Column and row orientations are local to each page. You change the orientation by appending `{data-orientation=rows}` or `{data-orientation=columns}` to your level 1 header
- Level 4 headers or more don't do anything in dashboards

Storyboards Allow More Text In your Layout

- This layout allows you to tell a story through a sequence of graphs with text descriptions
- The R Markdown syntax doesn't really make sense. You identify Frames with level 3 headers, and then you add commentary after ***
- You can't use Level 1 or Level 2 headers with Storyboards

Try a Storybook Dashboard

Knit the `42-storyboard-layout.Rmd`
How is this layout different from the grid layouts?

3m 00s

Dashboards Are Composed Of...

Displays

HTML widgets create display interactivity using the user's browser

HTML Widgets Create Display Interactivity

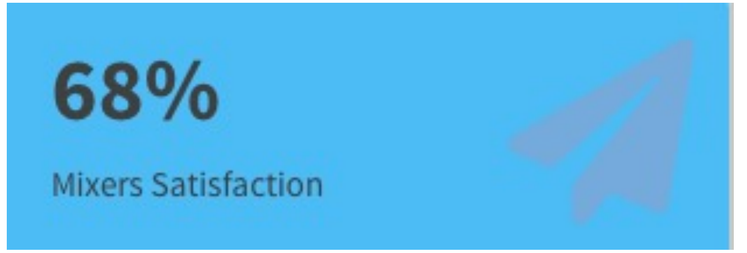
- HTML Widgets create interactivity within a Flexdashboard layout pane.
- See all 106 submitted HTML Widgets at: http://gallery.htmlwidgets.org
- You can develop your own if you know Javascript.
- We'll focus on a few:
  - Valueboxes and Gauges for individual values
  - Plotting with plotly
  - Tables with datatable (DT)
  - Maps with leaflet

Valueboxes Show Single Values

- They dress up single values with an icon and a color

```
valueBox(params$conf_attendees, "Attendees", icon = "fa-users")
```


```
valueBox(paste0(mixers_sat_percent, "%"),
         icon = "fa-paper-plane",
         color = ifelse(mixers_sat_percent >= 50,
                        ifelse(mixers_sat_percent >= 75, params$success_color,
                               params$warning_color),
                               params$danger_color))
```



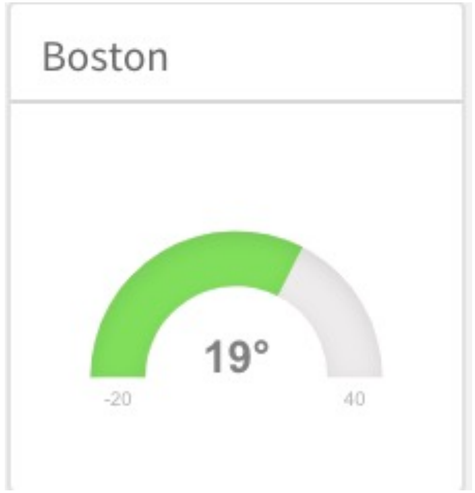
Github repo: rstd.io/conf20-rmd-dash

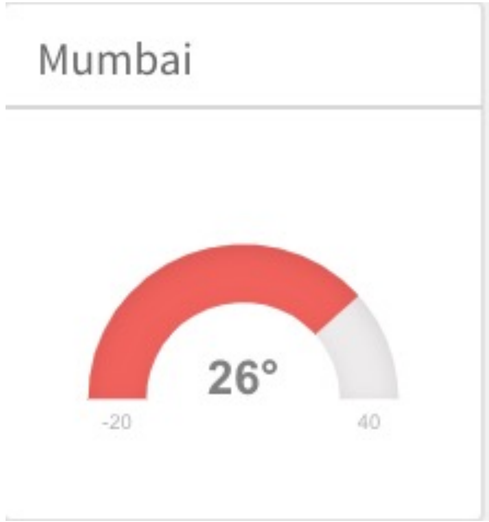
rstudio.cloud workspace: rstd.io/RMAID

Gauges Show Single Values In Context

- They dress up single values with an icon and a color

```
gauge(my_weather$temperature, min = gauge_minimum_temp, max =
gauge_maximum_temp, symbol = "°",
      sectors = gaugeSectors(warning = c(gauge_minimum_temp, gauge_cold_temp),
                             success = c(gauge_cold_temp,gauge_hot_temp),
                             danger = c(gauge_hot_temp,gauge_maximum_temp),
      colors = gauge_colors))
```





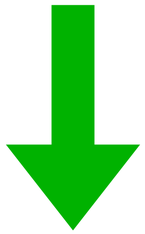
Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Plotting With Plotly

- Plotly allows you to make any ggplot interactive
- Simply assign your ggplot output to a variable and ggplotly that variable

```
ggplot(mtcars, aes(disp, hp, color = cyl)) + geom_point()
```

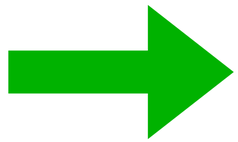


```
g <- ggplot(mtcars, aes(disp, hp, color = cyl)) + geom_point()
ggplotly(g)
```

Tables With Data Table

- Displays tables in a nice paged output
- Allows interactive searching and sorting of a table
- Easy to use: just use DT::datatable(data_frame) to display your table.
- Gotcha: the data table library is called DT, not datatable.

```
DT::datatable(mtcars)
```



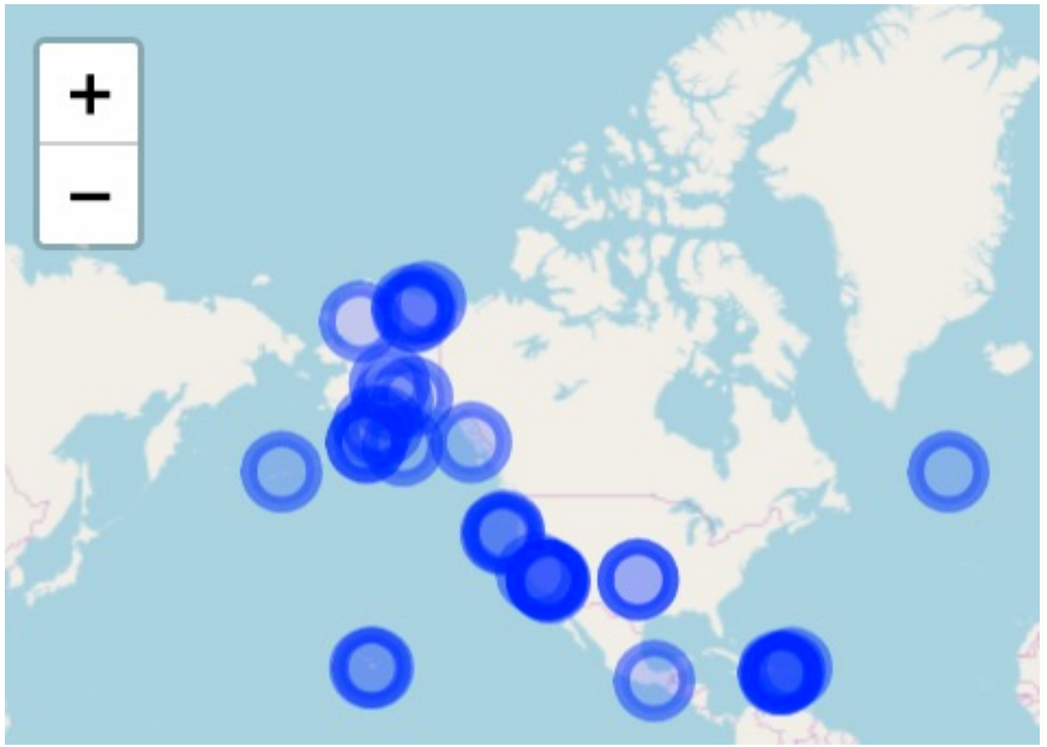
Show 10 entries Search:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
Mazda RX4	21	6	160	110	3.9	2.62	16.46	0	1
Mazda RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0

Maps With Leaflet

- An easy way to create maps with overlays
- All you need is a data.frame or tibble that has columns lat and lng for latitude and longitude
- leaflet() %>% addTiles() gives you a basic map
- Usually you just pipe in new layers of markers

```
leaflet() %>%
  addTiles()  %>%
  addCircleMarkers(data =
     latest_earthquakes)
```



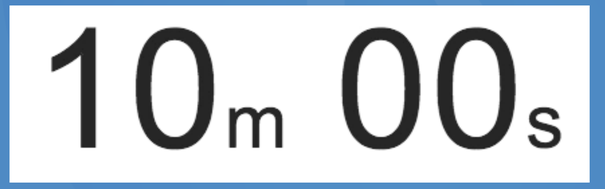
Github repo: rstd.io/conf20-rmd-dash

Exercise 43

Build Your Output Dashboard

1. Click on **`43-add-displays.Rmd`** in the Files pane to open that file
2. Convert all the ggplots to **`plotly`** widgets.
3. Change the **`mtcars`** pane to display a datatable
4. Add layers in the Earthquakes Map pane to display **`latest_earthquakes`**.

Hint: make one change at a time and test each one before knitting.



Can we increase the interactivity without adding controls?

We can create cross-Widget interactivity using `crosstalk`

- Crosstalk is an R package that allows interactions in one widget to affect another
- Documentation is here: https://rstudio.github.io/crosstalk/index.html
- The data used in crosstalk must be a data frame
- Crosstalk only works with HTML widgets designed to support it -- a list of those is here: https://rstudio.github.io/crosstalk/widgets.html

A simple example: 44-simple-crosstalk.Rmd

````
```{r shared_data, warning = FALSE}
shared_df <- SharedData$new(quakes[sample(nrow(quakes), 10),])
```

### Column 1 Map

```{r map}
  leaflet(shared_df) %>% addTiles() %>% addMarkers()
```

### Column 2 Table

```{r table}
  datatable(shared_df)
```
````

Shared data frame shared_df created from regular data frame using S3 function SharedData$new

A simple example: 44-simple-crosstalk.Rmd

````
```{r shared_data, warning = FALSE}
shared_df <- SharedData$new(quakes[sample(nrow(quakes), 10),])
```

### Column 1 Map

```{r map}
leaflet(shared_df) %>% addTiles() %>% addMarkers()
```

### Column 2 Table

```{r table}
datatable(shared_df)
```
````

Leaflet now reads the shared data frame shared_df

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

A simple example: 44-simple-crosstalk.Rmd

````
```{r shared_data, warning = FALSE}
shared_df <- SharedData$new(quakes[sample(nrow(quakes), 10),])
```

### Column 1 Map

```{r map}
  leaflet(shared_df) %>% addTiles() %>% addMarkers()
```

### Column 2 Table

```{r table}
  datatable(shared_df)
```
````

Datatable displays shared_df as a table

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Exercise 44

rstudio::conf

Try a simple crosstalk app

1. Click on **`44-simple-crosstalk.Rmd`** in the Files pane
2. Knit the result
3. Click on the box symbol on the map
4. Draw boxes on the map and see the datatable change
5. Click on specific rows on the table to see that row highlighted

Crosstalk can filter the data frame too

````
```{r shared_data, warning = FALSE}
shared_df <- SharedData$new(quakes[sample(nrow(quakes), 10),])
filter_slider("mag", "Magnitude", shared_df, column=~mag, step=0.1, width=250)
filter_slider("depth", "Depth", shared_df, column=~depth, step=0.1, width=250)
```

### Column 1 Map

```{r map}
  leaflet(shared_df) %>% addTiles() %>% addMarkers()
```

### Column 2 Table

```{r table}
  datatable(shared_df)
```
````

We add two filters that allow us to control the magnitude and depth of earthquakes

Everything else stays the same

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Exercise 45

rstudio::conf

Experiment with filtering

1. Click on **`45-filtered-crosstalk.Rmd`** in the Files pane
2. Knit the result
3. Click on the box symbol on the map
4. Draw boxes on the map and see the datatable change
5. Click on specific rows on the table to see that row highlighted

3m 00s

And crosstalk works with plotly too

```
mtcars$cyl <- as.factor(mtcars$cyl)
shared_mtcars <- SharedData$new(mtcars)
theme_set(theme_minimal())
```

```
# A Cornucopia of Displays

## Column 1

### mtcars Chart 1

```{r plot1}
g <- ggplot(shared_mtcars, aes(disp, hp, color = cyl)) + geom_point()
ggplotly(g) %>% highlight("plotly_selected")
```
```

Our shared data is now shared_mtcars

We apply ggplot to our shared_mtcars and ggplotly the result

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

Exercise 46

rstudio::conf

Experiment with crosstalk and plotly

1. Click on **`46-crosstalk-multiple.Rmd`** in the Files pane
2. Knit the result
3. Click on the Box Select icon in the plotly displays and then brush across points you want to see
4. Watch the other plots and tables change

Your Turn

In your groups, discuss:

1. Was filtering or the plotly document easier to understand and navigate?
2. Can you think of other ways to control the display of information?

Dashboards often have....

User controls

Frankly, we only have 3 control options so far

- Parameterized reports where you edit the parameters and knit the source
- Parameterized reports where you do a Knit with Parameters
- Limited interaction using crosstalk

Github repo: rstd.io/conf20-rmd-dash

rstudio.cloud workspace: rstd.io/RMAID

To build general interactive controls, we need

Summary

- Dashboards consist of 3 components
  - Layouts constructed using R Markdown headers and YAML directives specifying
    - Row orientation
    - Column orientation
    - Storyboard panes
  - Displays built from R code and HTML widgets
    - Valueboxes
    - Plotly
    - Datatable
    - Leaflet
  - Interactive controls which need crosstalk, Shiny, or some other mechanism