Introduction to Dashboards

BMRN CSM: Building dashboards with R markdown

Martin Frigaard

2020-12-07

`flexdashboard` = Dashboards using R Markdown (and Shiny)

Load the packages


```
install.packages(c("tidyverse", "inspectdf",
                   "flexdashboard", "reactable"))
library(tidyverse)
library(inspectdf)
library(flexdashboard)
library(reactable)
```

Outline (1)

Recap `rmarkdown`

What belongs in a dashboard?

Layouts

- *Sidebars, Columns, and Rows*
- *Multiple Pages, Tabs*

Themes

- *Bootstrap themes*

Outline (2)

`inspectdf` package

- *graphs, syntax*

`reactable` package

- *table displays*

Examples with `shiny`

- *shiny reactivity*

Materials

Slides

https://mjfrigaard.github.io/intro-to-dashboards/Index.html

Exercises

RStudio Project

https://rstudio.cloud/project/2000287

`rmarkdown` = `YAML` + Markdown + R (or other languages)

What is RMarkdown?

Three technologies:

1) Markdown is a plain text markup language for capturing *human-readable* prose

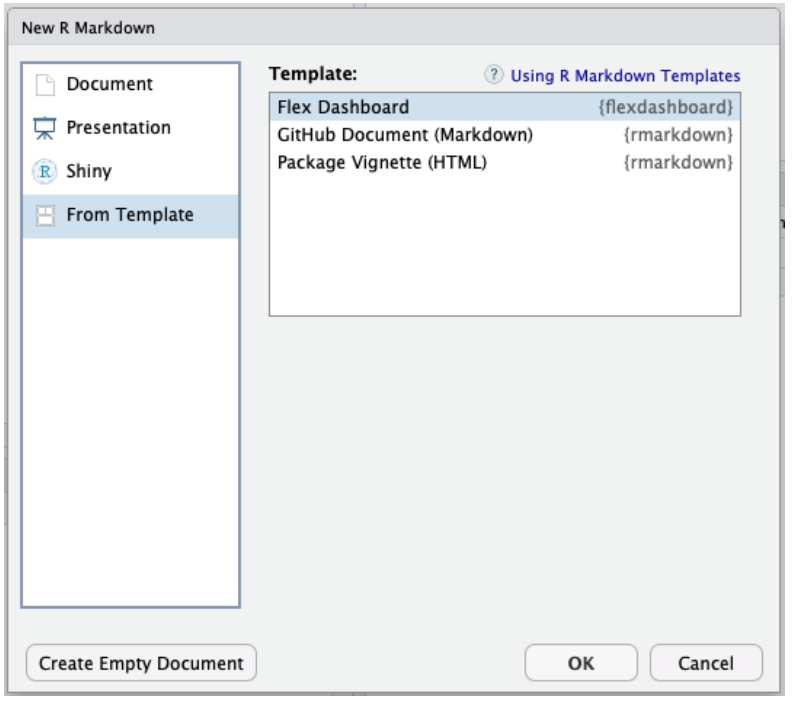
2) Data manipulation/graphing/statistical language engines for computing *machine-readable* code

3) Multiple *output options* for creating PDFs, Word docs, PowerPoints, HTML, etc.

Your Turn 1

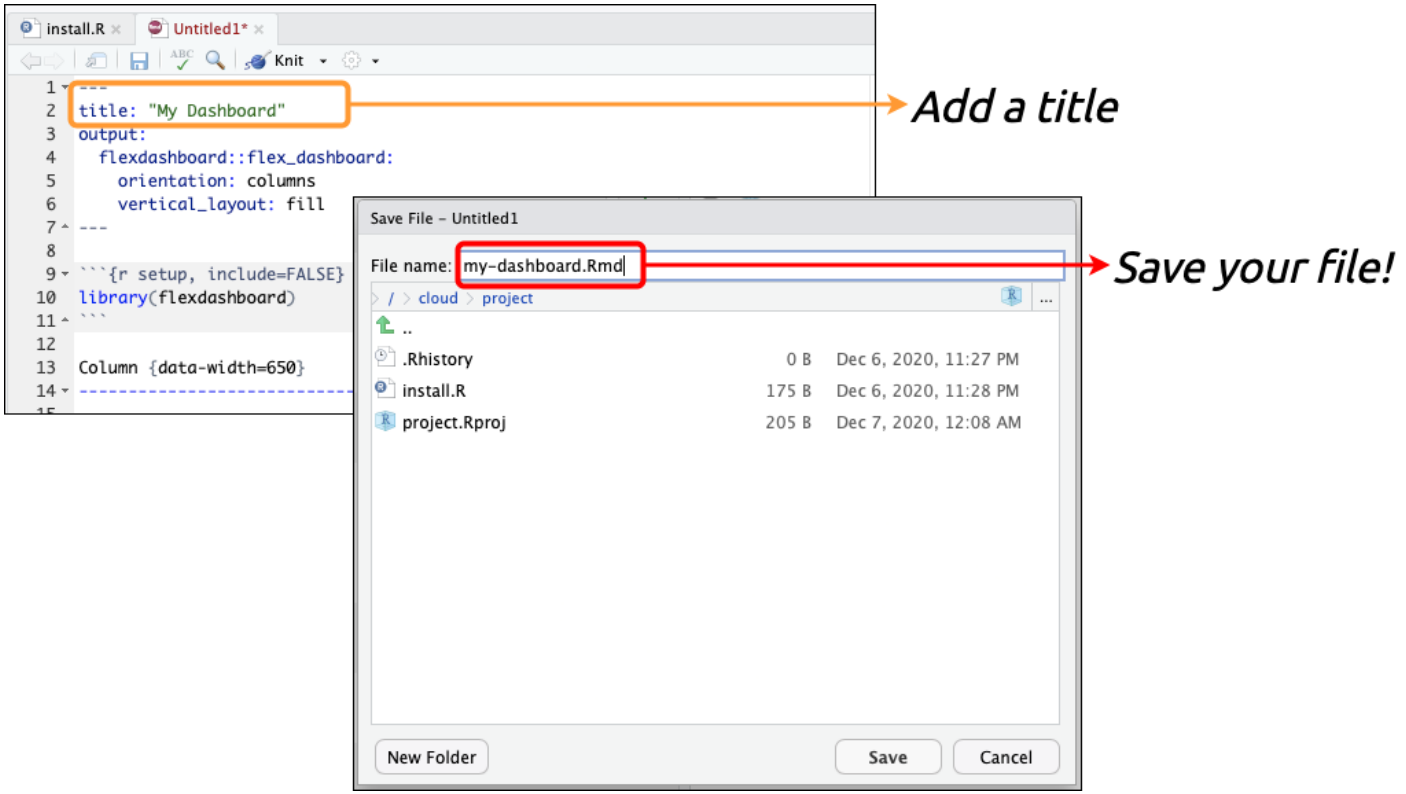
Open a new R Markdown file

file > *New File* > *R Markdown* > *From Template* > `flexdashboard`



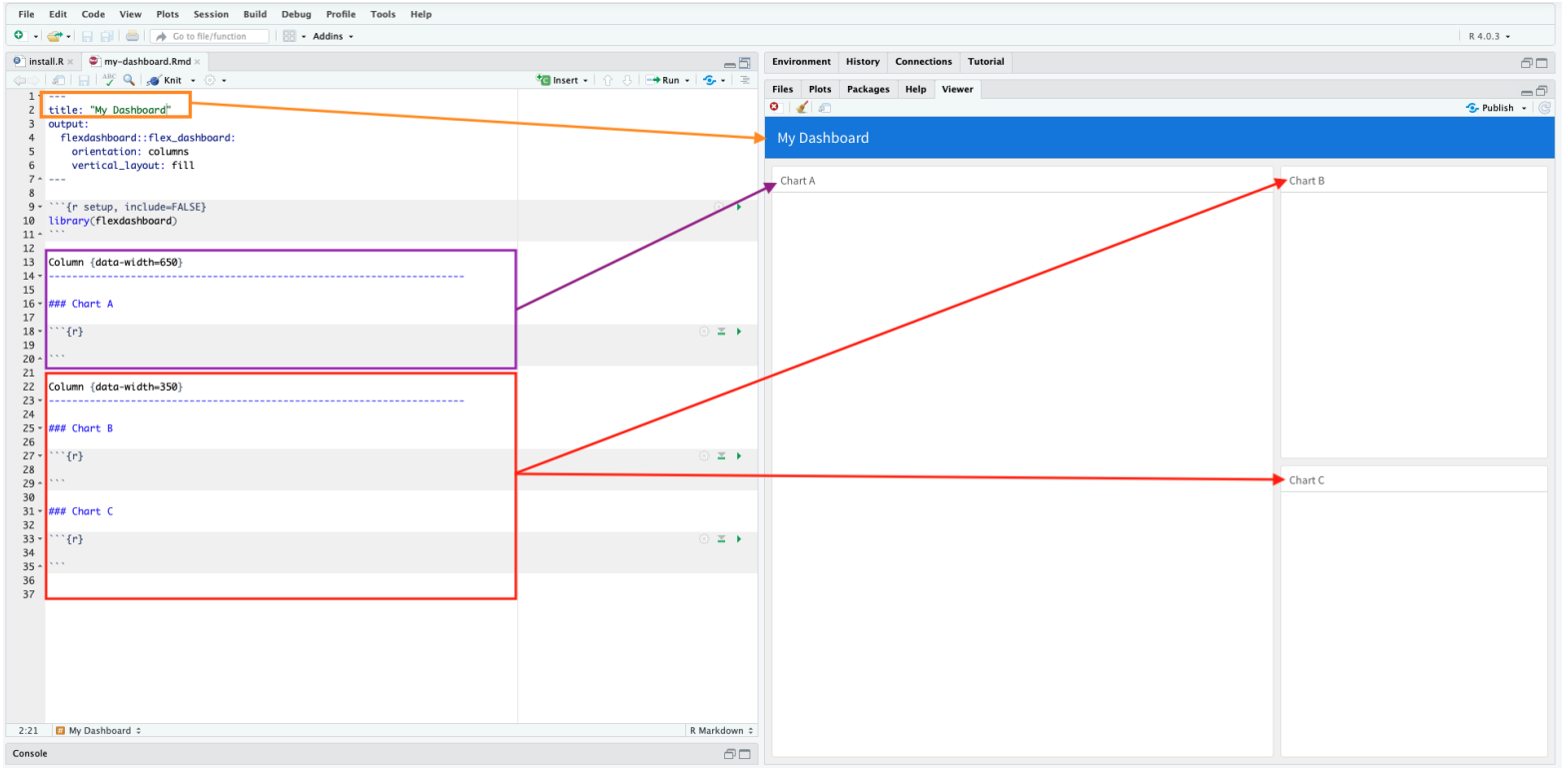
Your Turn 2

Add title and save R Markdown file



Your Turn 3

knit!



What belongs in a dashboard?

> Dashboards are particularly common in **business-style reports**. They can be used to **highlight brief and key summaries of a report**. The layout of a dashboard is often grid-based, with components arranged in boxes of various sizes.

Dashboard Anatomy

The YAML header setting creates the dashboard:

```
output:
  flexdashboard::flex_dashboard:
```

The layout is determined by the **`orientation`** and **`vertical_layout`** options.

```
    orientation: columns
    vertical_layout: fill
```

Column Widths

Column Widths must add up to **1000**

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

### Chart A

```{r}

```

Column {data-width=350}

Chart B

```{r}

```

Chart C

```{r}

```
```

Sidebars

Include a sidebar with `{.sidebar data-width=200}`

````
Inputs {.sidebar data-width=200}
-----------------------------------------------------------------------

```{r}

```
````

Adjust the column widths (set both to `{data-width=400}`)

````
Column {data-width=400}
-----------------------------------------------------------------------

### Chart A

```{r}

```

Column {data-width=400}
-----------------------------------------------------------------------

### Chart B

```{r}

```

### Chart C

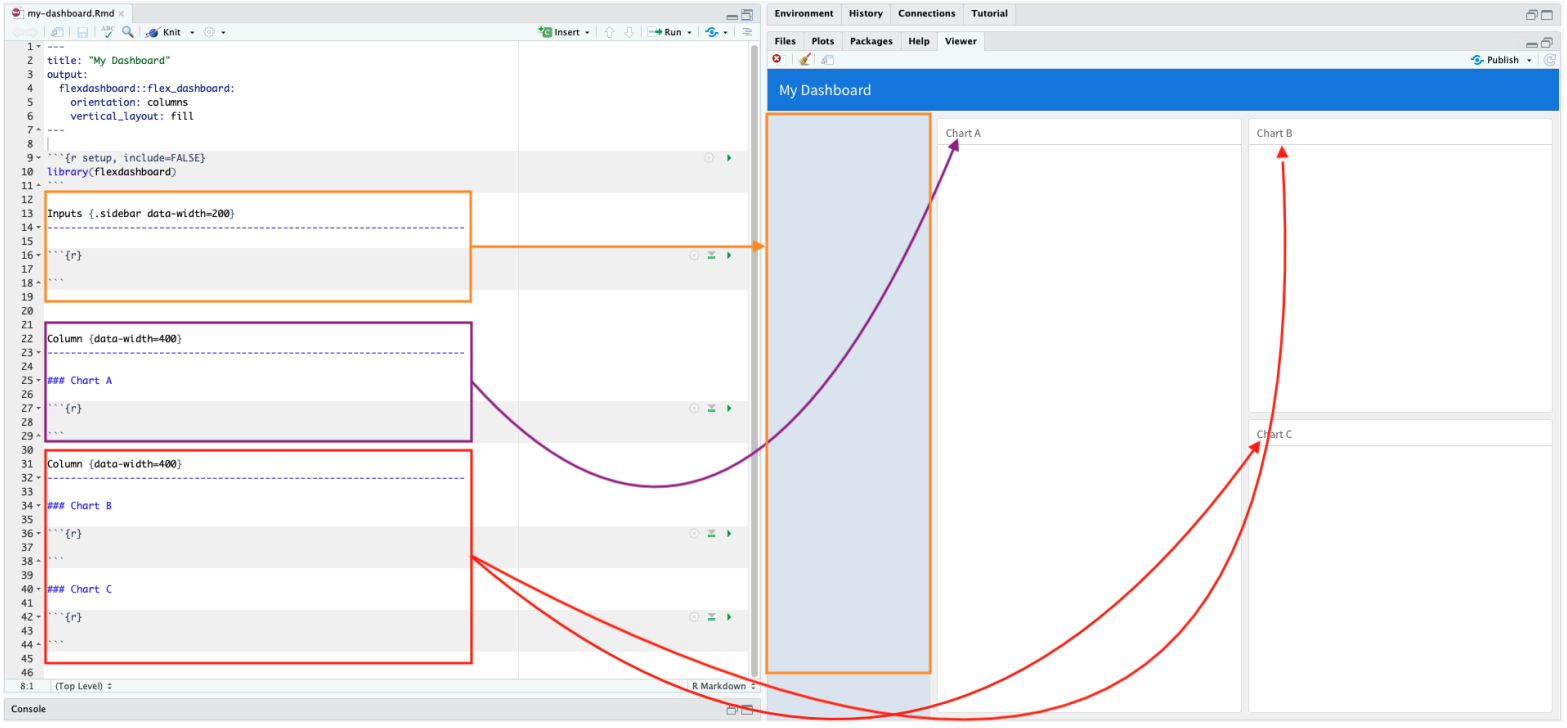
```{r}

```
````

Knit!

Sidebars

Sidebars are typically used for data inputs and user-interface controls



Row Layout

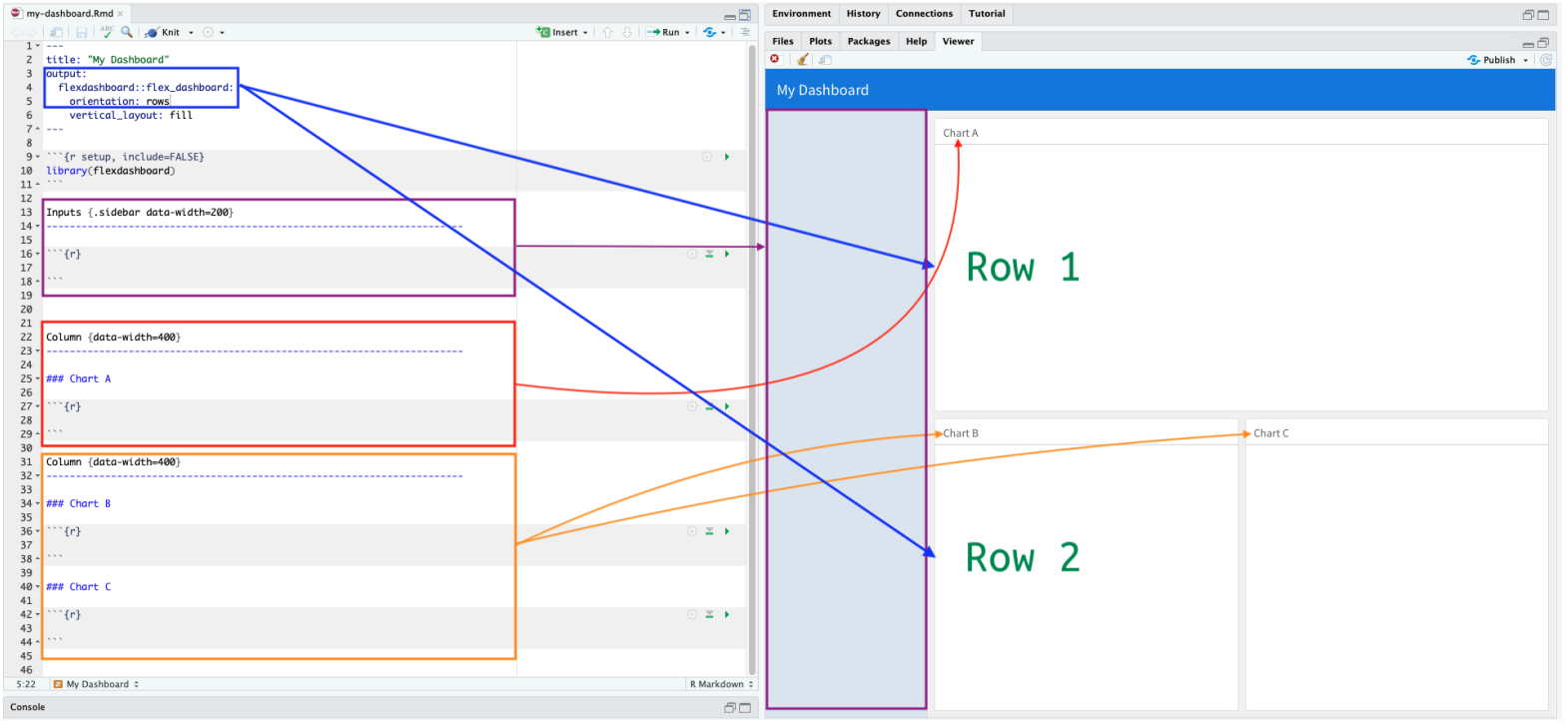
We can also orient by rows

Change the `orientation` of the dashboard

```
output:
  flexdashboard::flex_dashboard:
      orientation: columns
```

Re-knit!

Rows Layout


my-dashboard.Rmd
Knit
Insert
Run
---
title: "My Dashboard"
output:
flexdashboard::flex_dashboard:
orientation: rows
vertical_layout: fill

```{r setup, include=FALSE}
library(flexdashboard)
```
Inputs {.sidebar data-width=200}

```{r}
```
Column {data-width=400}

Chart A
```{r}
```
Column {data-width=400}

Chart B
```{r}
```
Chart C
```{r}
```
5:22
My Dashboard
R Markdown
Console
Environment
History
Connections
Tutorial
Files
Plots
Packages
Help
Viewer
Publish
My Dashboard
Chart A
Row 1
Chart B
Chart C
Row 2

Scrolling

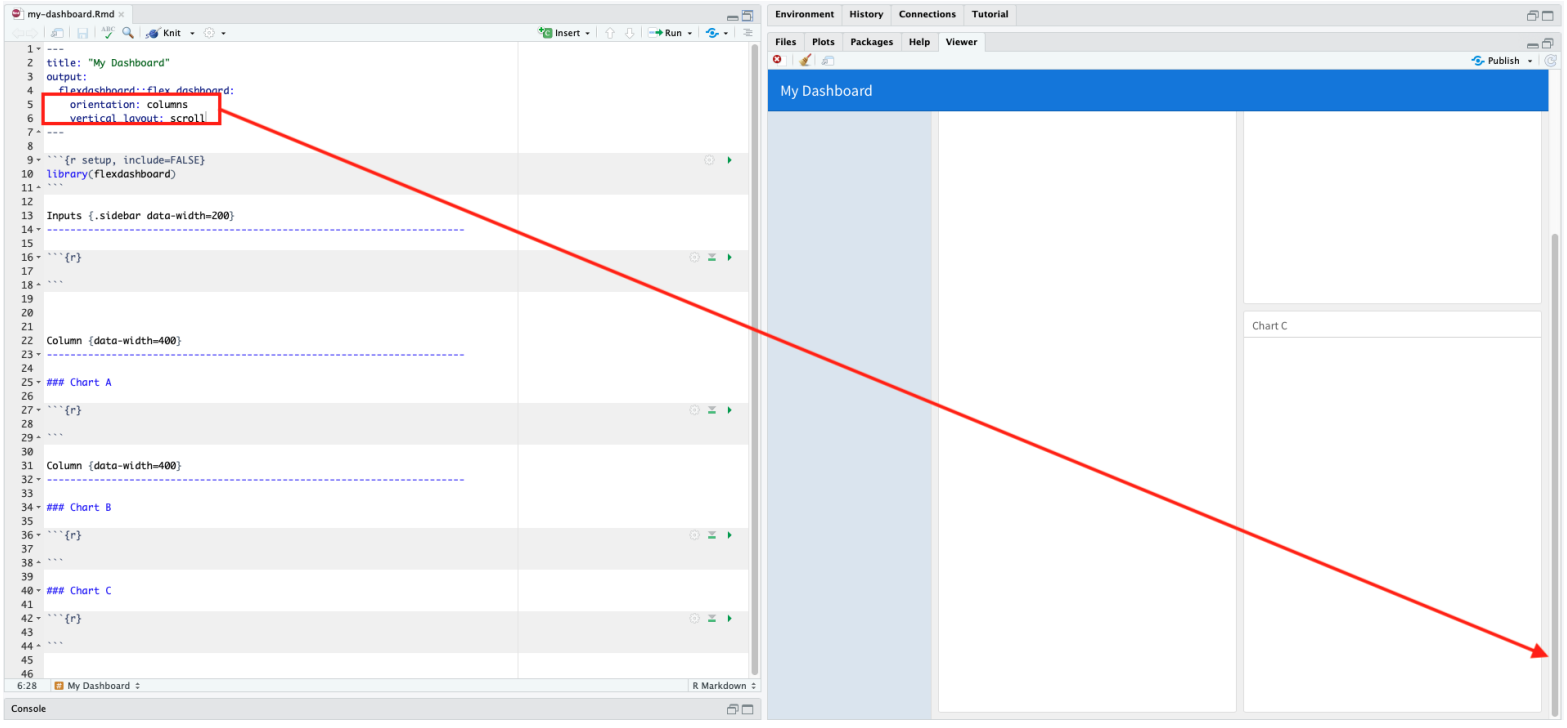
Change the `YAML` header back to `orientation: columns` and `vertical_layout: scroll`

```
    orientation: columns
    vertical_layout: scroll
```

Re-knit!

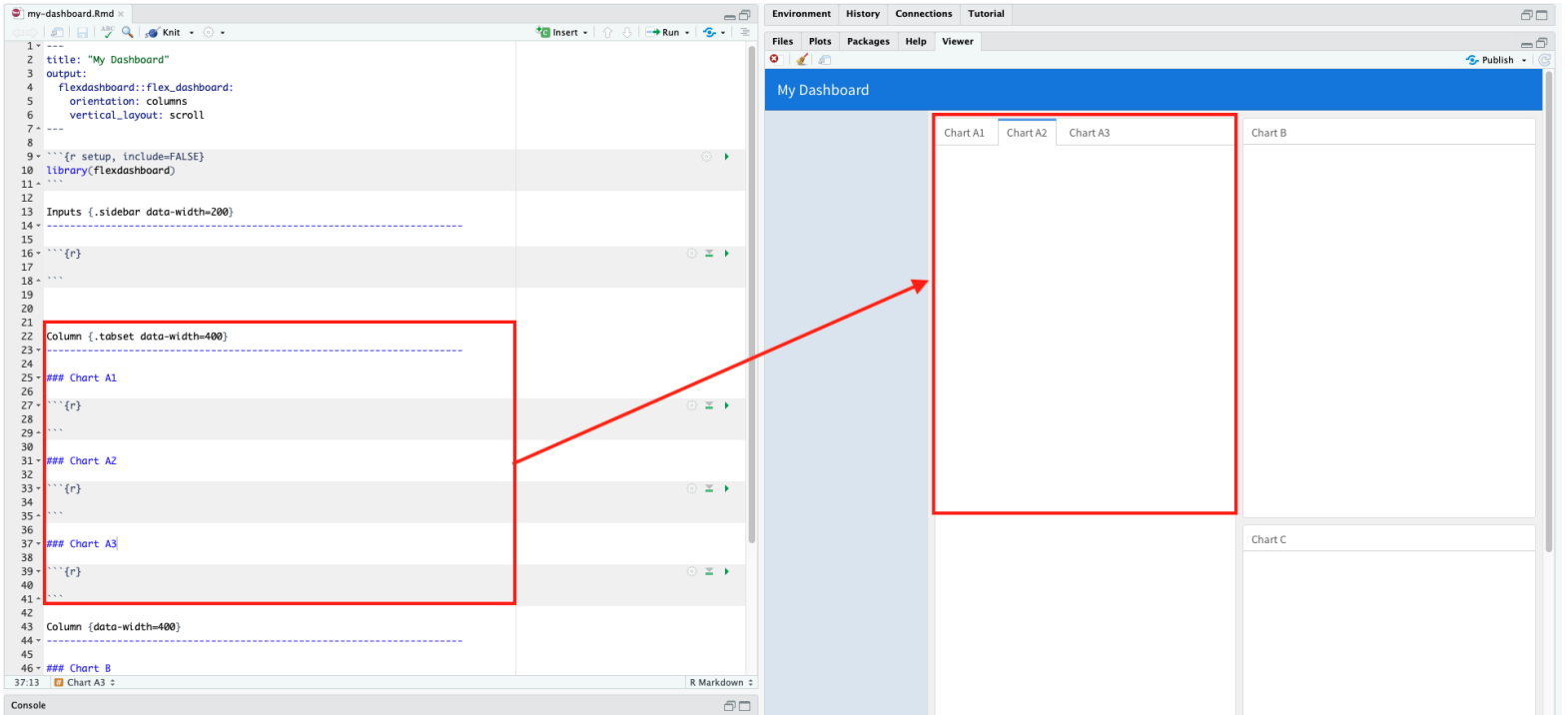
Scrolling

Now we can scroll past the end of the column.



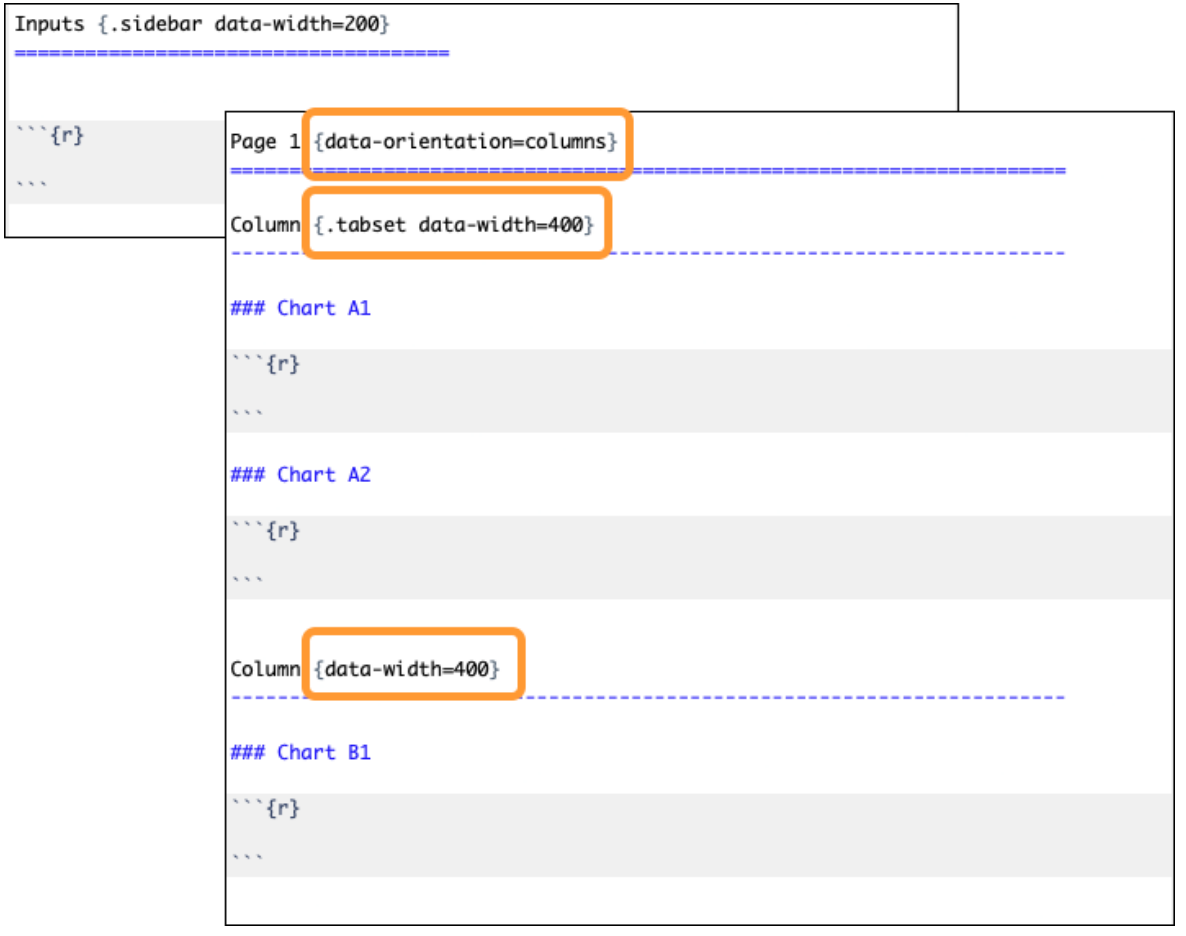
Tabsets

Add tabsets with `{.tabset}`



Global Sidebar and Pages

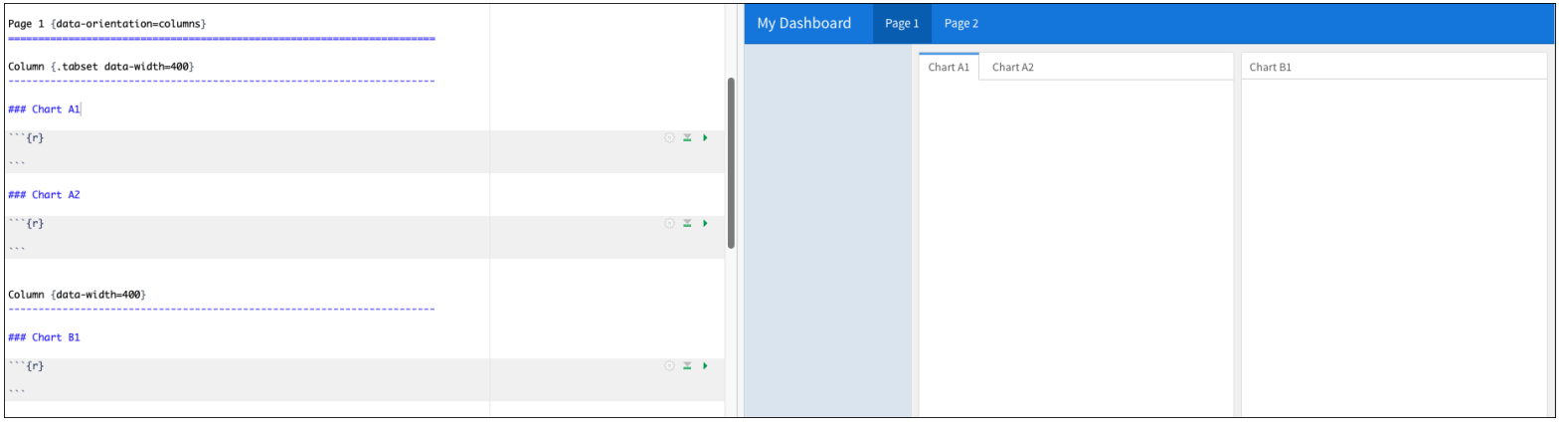
For global settings, we use `==========` instead of `----------`



Global Sidebar and Pages

`data-orientation=columns`

`.tabset`



Global Sidebar and Pages

For global settings, we use `==========` instead of `----------`

```
Page 2 {data-orientation=rows}
==========

Row {.tabset .tabset-fade data-height=600}
----------

### Chart C1

```{r}
```

Chart C2

```{r}
```

Row {.tabset .tabset-fade data-height=400}

Chart D1

```{r}
```

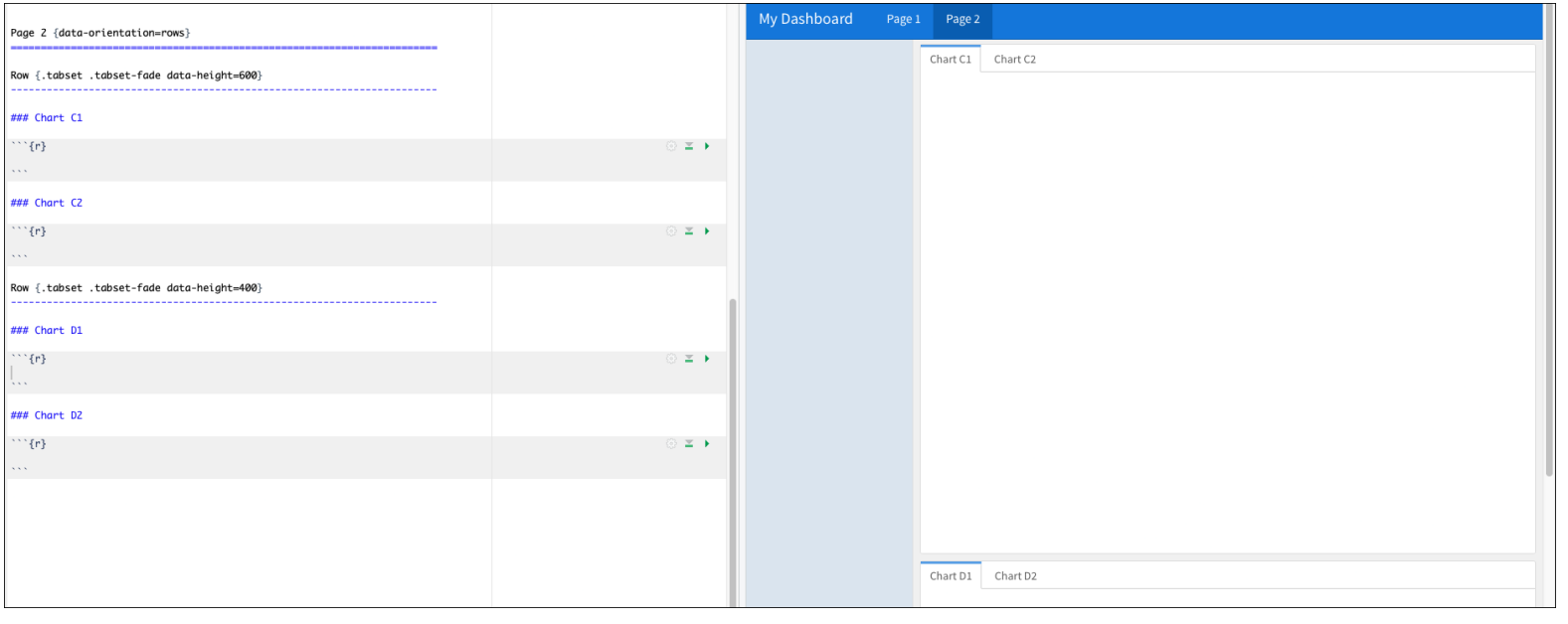
Chart D2

```{r}
```
```

Global Sidebar and Pages

`data-orientation=rows`

`.tabset-fade`



Menus

`data-navmenu=More`

```
Page 3 {data-navmenu='More'}
=====================================================================


Column
---------------------------------------------------------------------

### Chart E

```{r}

```

Page 4 {data-navmenu='More'}
===

Column

Chart F

```{r}

```
```

Menus

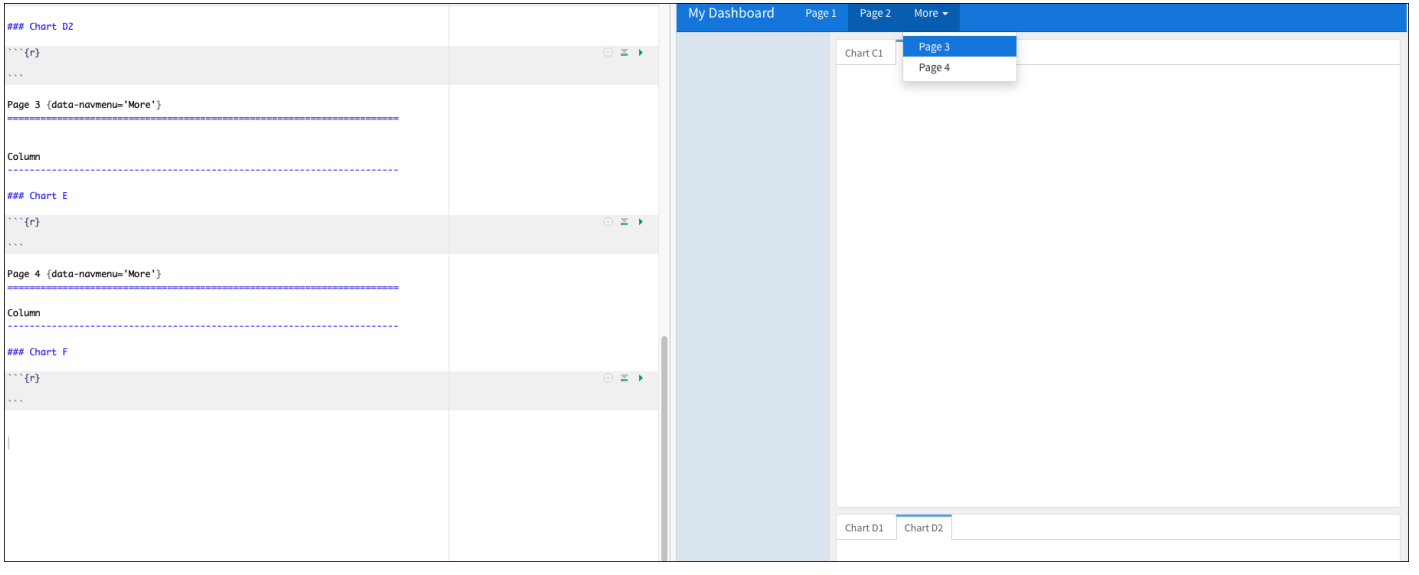


Chart D2
```{r}
```
Page 3 {data-navmenu='More'}
Column
Chart E
```{r}
```
Page 4 {data-navmenu='More'}
Column
Chart F
```{r}
```
My Dashboard
Page 1
Page 2
More
Page 3
Page 4
Chart C1
Chart D1
Chart D2

Themes

Change `themes` (just like `html_document`!)

```
title: "My Dashboard"
output:
  flexdashboard::flex_dashboard:
    theme: spacelab
```

See the website for more information

`inspectdf` = quicky examine datasets

Previous Slides: Apple Mobility Data

https://mjfrigaard.github.io/data-viz-as-comm/Index.html

Import Data

```
AppleMobRaw <- readr::read_csv("https://bit.ly/36tTVpe")
```

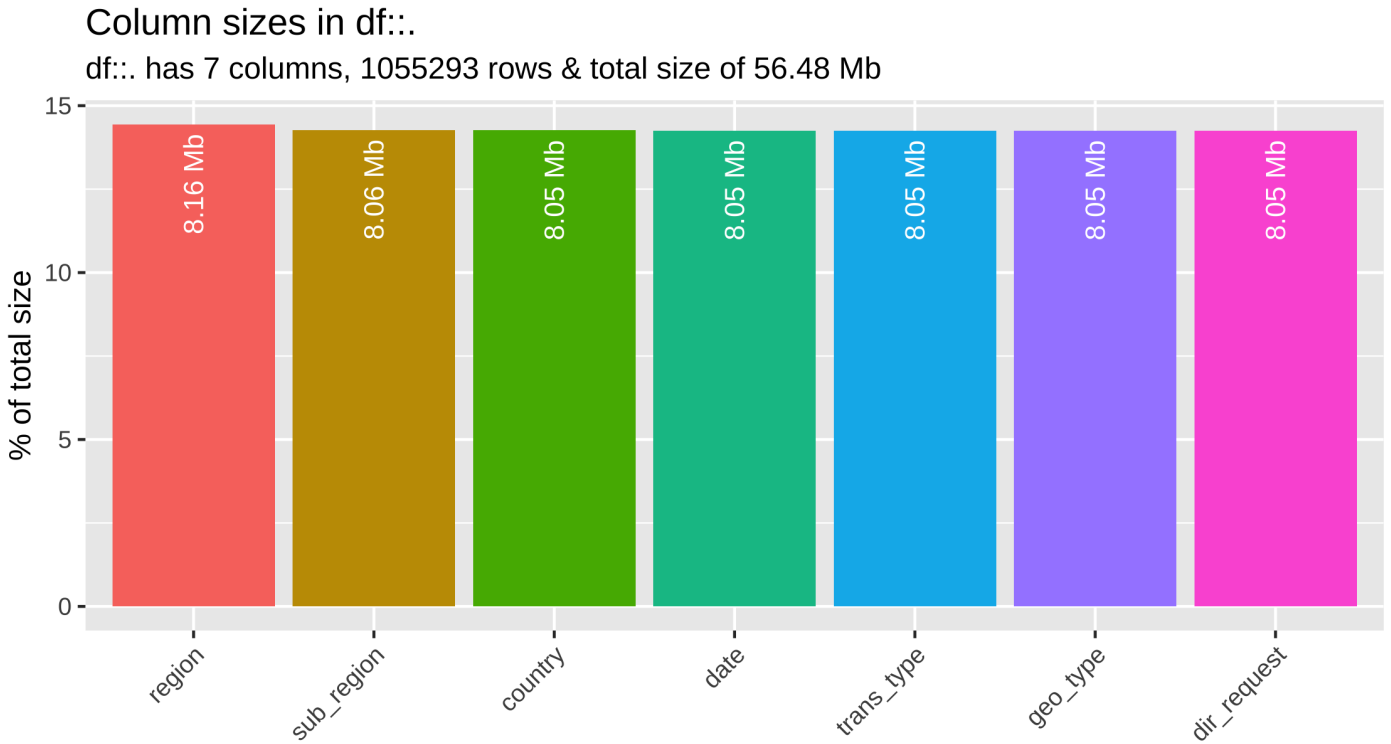
Previous Slides: Apple Mobility Data

Don't Forget Wrangling Steps!

```
AppleMobRaw %>%
  # transpose data
  tidyr::pivot_longer(cols = -c(geo_type:country),
    names_to = "date", values_to = "dir_request") %>%
    # remove missing country data
  dplyr::filter(!is.na(country) & !is.na(`sub-region`)) %>%
  # clean names
  janitor::clean_names() %>%
  # date
  mutate(date = lubridate::ymd(date)) %>%
  # create trans_type
  rename(trans_type = transportation_type) -> TidyApple
```

Dataset size in memory


```
TidyApple %>%
  inspectdf::inspect_mem() %>%
  inspectdf::show_plot(text_labels = TRUE)
```



Your Turn 4

Add the data to the `.sidebar`

Add the import and wrangle code to the sidebar in the dashboard.

```
AppleMobRaw <- readr::read_csv("https://bit.ly/36tTVpe")
AppleMobRaw %>%
  # transpose data
  tidyr::pivot_longer(cols = -c(geo_type:country),
    names_to = "date", values_to = "dir_request") %>%
    # remove missing country data
  dplyr::filter(!is.na(country) & !is.na(`sub-region`)) %>%
  # clean names
  janitor::clean_names() %>%
  # date
  mutate(date = lubridate::ymd(date)) %>%
  # create trans_type
  rename(trans_type = transportation_type) -> TidyApple
```

Your Turn 5

Add the `Memory  Size` graph to the flexdashboard

Add this code to `A1`

```
TidyApple %>%
  inspectdf::inspect_mem() %>%
  inspectdf::show_plot(text_labels = TRUE)
```

Knit--how does it look?

Your Turn 6

Add the `Missing Data` graph to the flexdashboard

Add this code to A2

```
TidyApple %>%
  inspectdf::inspect_na() %>%
  inspectdf::show_plot(text_labels = TRUE)
```

Knit--how does it look?

Your Turn 7

Add the `Categorical Data` Graph

Add this code to `B1`

```
TidyApple %>%
  select_if(is.character) %>%
  inspectdf::inspect_cat() %>%
  inspectdf::show_plot(text_labels = TRUE)
```

Knit--how does it look?

Your Turn 8

Create Another Tab with `.tabset`

Add `Data Imbalances` Graph

Add the code below for page 1, row 2, tab 2.

```
TidyApple %>%
  inspectdf::inspect_imb() %>%
  inspectdf::show_plot(text_labels = TRUE)
```

Knit--how does it look?

Page 2 (tab 1)


```
Page 2 {data-orientation=rows}
======================================
```

Create a `.tabset/.tabset-fade` Row

```
Row {.tabset .tabset-fade data-height=600}
--------------------------------------------
```

Add `Numeric Data` Graph

```
TidyApple %>%
  select_if(is.numeric) %>%
  inspectdf::inspect_num() %>%
  inspectdf::show_plot(text_labels = TRUE)
```

Knit--how does it look?

Page 2 (tab 2)

Add 'Distributions `ggridges`' Graph

```
library(ggridges)
lab_ridges <- labs(x = "Apple directions requests",
                   y = "Transportation Types",
     title = "Direction Requests by Transportation Type",
     subtitle = "source: https://covid19.apple.com/mobility")


TidyApple %>%
  ggplot() +
  geom_density_ridges(aes(x = dir_request,
                          y = trans_type,
    fill = trans_type),
    alpha = 1/5) +
  lab_ridges
```

Knit--how does it look?

Display tables (`rmarkdown`)

Create Another `.tabset/.tabset-fade` Row

```
Row {.tabset .tabset-fade data-height=400}
-------------------------------------
```

In tab 1, add `TopUSCities` as `paged_table`

```
TopUSCities <- TidyApple %>%
  filter(country == "United States" &
           region %in% c("New York City","Los Angeles",
                         "Chicago", "Houston", "Phoenix"))
rmarkdown::paged_table(TopUSCities)
```

Display tables (`reactable`)

In tab 2, add `MaxUSCitiesDriving` as `reactable`

```
TopUSCities %>%
  filter(trans_type == "driving") %>%
  group_by(region) %>%
  slice_max(dir_request) %>%
  ungroup() -> MaxUSCitiesDriving
reactable(MaxUSCitiesDriving,
              resizable = TRUE, showPageSizeOptions = TRUE,
              selection = "multiple", onClick = "select")
```

More Examples

Check out the package website and gallery

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