

# 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.

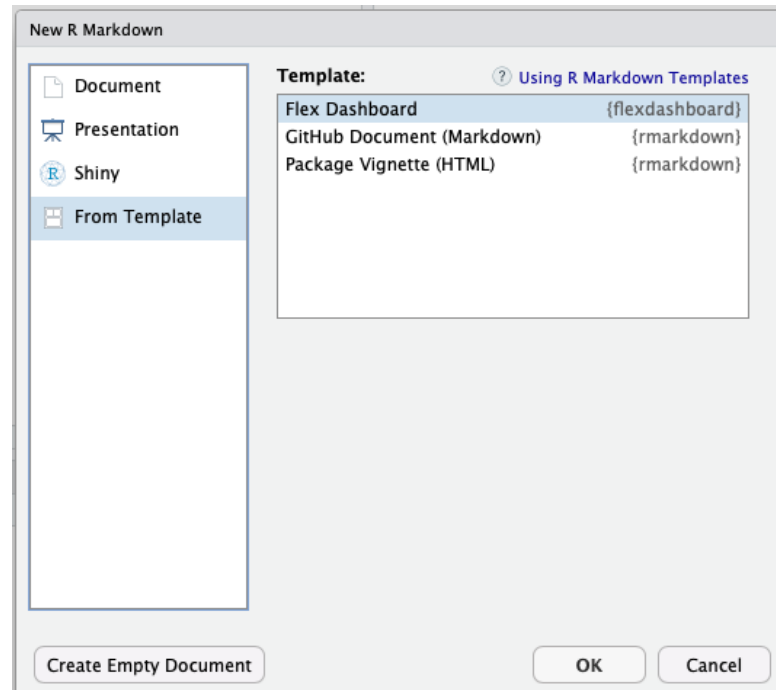


# Your Turn 1



## Open a new R Markdown file

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



# Your Turn 2



## Add title and save R Markdown file

The screenshot shows the RStudio interface with an R Markdown file open. The file content is as follows:

```
1 ---
2 title: "My Dashboard"
3 output:
4   flexdashboard::flex_dashboard:
5     orientation: columns
6     vertical_layout: fill
7 ---
8
9 {r setup, include=FALSE}
10 library(flexdashboard)
11
12
13 Column {data-width=650}
14
15
```

An orange box highlights the title line, with an arrow pointing to the text *Add a title*.

A "Save File - Untitled1" dialog box is open, showing the file name "my-dashboard.Rmd" in the "File name:" field, which is highlighted with a red box. A red arrow points from this box to the text *Save your file!*. The dialog also shows a file explorer view with the following files:

File Name	Size	Modified
..		
.Rhistory	0 B	Dec 6, 2020, 11:27 PM
install.R	175 B	Dec 6, 2020, 11:28 PM
project.Rproj	205 B	Dec 7, 2020, 12:08 AM

The dialog has buttons for "New Folder", "Save", and "Cancel".

# Your Turn 3



knit!

A screenshot of the RStudio interface. The left pane shows the R Markdown source file 'my-dashboard.Rmd'. The right pane shows the rendered output, a dashboard titled 'My Dashboard'. Colored arrows connect code elements in the source to their rendered counterparts: an orange arrow from the title, a purple arrow from 'Chart A', and a red arrow from 'Chart B'.

```
1 ---
2 title: "My Dashboard"
3 output:
4   flexdashboard::flex_dashboard:
5     orientation: columns
6     vertical_layout: fill
7 ---
8
9 ```{r setup, include=FALSE}
10 library(flexdashboard)
11 ```
12
13 Column {data-width=650}
14 -----
15
16 ## Chart A
17 ```{r}
18
19 ```
20
21
22 Column {data-width=350}
23 -----
24
25 ## Chart B
26 ```{r}
27
28 ```
29
30 ## Chart C
31 ```{r}
32
33 ```
34
35
36
37
```

The rendered dashboard on the right has a blue header bar with the title 'My Dashboard'. Below it, there are three chart areas. The first area is labeled 'Chart A' and is connected to the '## Chart A' code block by a purple arrow. The second area is labeled 'Chart B' and is connected to the '## Chart B' code block by a red arrow. The third area is labeled 'Chart C' and is connected to the '## Chart C' code block by a red arrow.

# 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
```

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

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

```
### Chart B
```

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

```
### Chart C
```

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

# Sidebars



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

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

```
```${r}```  
```\n`
```

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

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

```
### Chart A
```

```
```${r}```  
```\n`
```

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

```
### Chart B
```

```
```${r}```  
```\n`
```

```
### Chart C
```

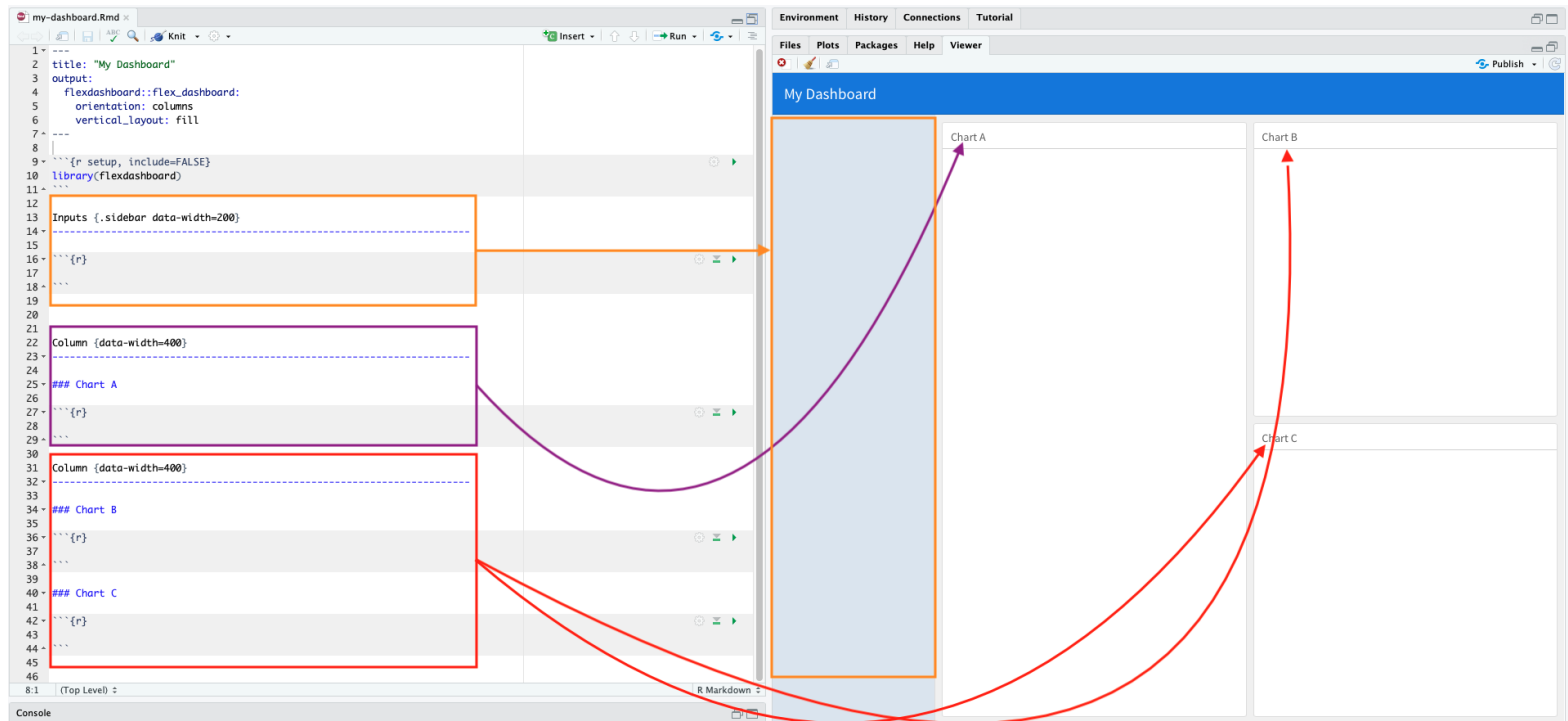
```
```${r}```  
```\n`
```

## 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



The screenshot displays the RStudio interface with an R Markdown file named `my-dashboard.Rmd` open. The code is structured to create a dashboard using the `flexdashboard` package. The `output` section specifies `flexdashboard::flex_dashboard` with `orientation: rows` and `vertical_layout: fill`. The `Inputs` section contains a sidebar with a text input. The `Column` sections are organized into two rows: Row 1 contains 'Chart A', and Row 2 contains 'Chart B' and 'Chart C'. Colored boxes and arrows link the code elements to the rendered dashboard: a blue box highlights the `flexdashboard` configuration, a purple box highlights the `Inputs` section, a red box highlights the first `Column` (Chart A), and an orange box highlights the second `Column` (Charts B and C). The rendered dashboard on the right shows 'My Dashboard' with a blue header, a sidebar input, and three chart areas labeled 'Chart A', 'Chart B', and 'Chart C'. The text 'Row 1' and 'Row 2' is overlaid on the dashboard to indicate the layout structure.

```
1 title: "My Dashboard"
2
3 output:
4   flexdashboard::flex_dashboard:
5     orientation: rows
6     vertical_layout: fill
7
8 ---
9
10 {r setup, include=FALSE}
11 library(flexdashboard)
12
13 Inputs {,sidebar data-width=200}
14
15 {r}
16
17
18
19
20
21
22 Column {data-width=400}
23
24 ## Chart A
25
26 {r}
27
28
29
30
31 Column {data-width=400}
32
33 ## Chart B
34
35 {r}
36
37
38
39 ## Chart C
40
41 {r}
42
43
44
45
46
```

# 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.

The screenshot shows the RStudio interface with a file named "my-dashboard.Rmd" open. The code editor on the left contains R Markdown code for a dashboard. A red box highlights the configuration for the first column: `orientation: columns` and `vertical layout: scroll`. A red arrow points from this box to the right-hand pane, which displays the rendered dashboard. The dashboard has a blue header "My Dashboard" and a sidebar on the left. The main content area is divided into two columns. The right column contains a section titled "Chart C". A vertical scrollbar is visible on the right side of the dashboard, indicating that the content can be scrolled vertically. The status bar at the bottom shows "6.28 My Dashboard" and "R Markdown".

# Tabsets



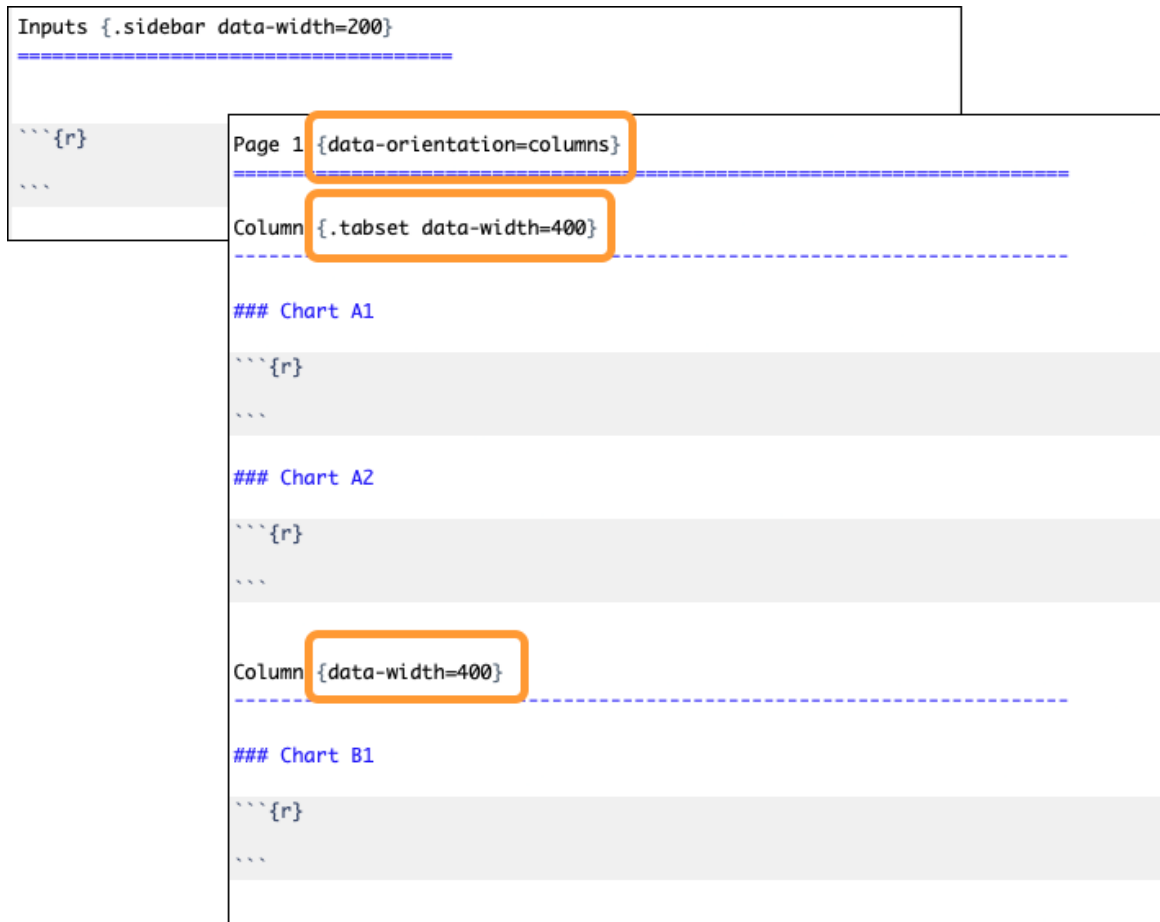
## Add tabsets with `{.tabset}`

The screenshot shows the RStudio interface. On the left, the R Markdown source file 'my-dashboard.Rmd' is open. It contains a title 'My Dashboard', an output section using 'flexdashboard::flex\_dashboard', and two columns of content. The first column is defined with 'Column {.tabset data-width=400}' and contains three charts labeled 'Chart A1', 'Chart A2', and 'Chart A3'. The second column is defined with 'Column {data-width=400}' and contains 'Chart B'. On the right, the rendered dashboard is shown. It has a blue header 'My Dashboard' and a sidebar on the left. The main content area displays the rendered tabset, with 'Chart A1', 'Chart A2', and 'Chart A3' as tabs. 'Chart A2' is currently selected and highlighted with a red box. A red arrow points from the code for 'Chart A2' in the source file to this rendered tab. 'Chart B' is shown to the right of the tabset, and 'Chart C' is shown below it. The RStudio interface includes a top menu bar with 'Environment', 'History', 'Connections', 'Tutorial', 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. The bottom status bar shows '37:13 R Markdown Console'.

# Global Sidebar and Pages



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



# Global Sidebar and Pages



data-orientation=columns

.tabset

The screenshot displays an R Markdown document with a global sidebar and a tabset. The sidebar on the left is titled "Page 1 {data-orientation=columns}" and contains three columns. The first column is titled "Column {.tabset data-width=400}" and contains two charts, "Chart A1" and "Chart A2". The second column is titled "Column {data-width=400}" and contains one chart, "Chart B1". The main content area on the right is titled "My Dashboard" and contains a tabset with three tabs: "Page 1", "Page 2", and "Page 3". The "Page 1" tab is active and shows the content of the first column, "Chart A1". The "Page 2" tab is also visible and shows the content of the second column, "Chart B1". The "Page 3" tab is not visible. The sidebar has a vertical scrollbar on the right side.

# 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

The screenshot displays an R Markdown document with a global sidebar and a tabset. The sidebar on the left is titled "Page 2 (data-orientation=rows)" and contains four rows of content, each with a heading (### Chart C1, ### Chart C2, ### Chart D1, ### Chart D2) and a code block containing a single line of R code: {r}. The main content area on the right is titled "My Dashboard" and features a tabset with two tabs: "Chart C1" and "Chart C2". The "Chart C1" tab is currently selected, showing a large empty box. Below the main content area, there is a footer with two tabs: "Chart D1" and "Chart D2".

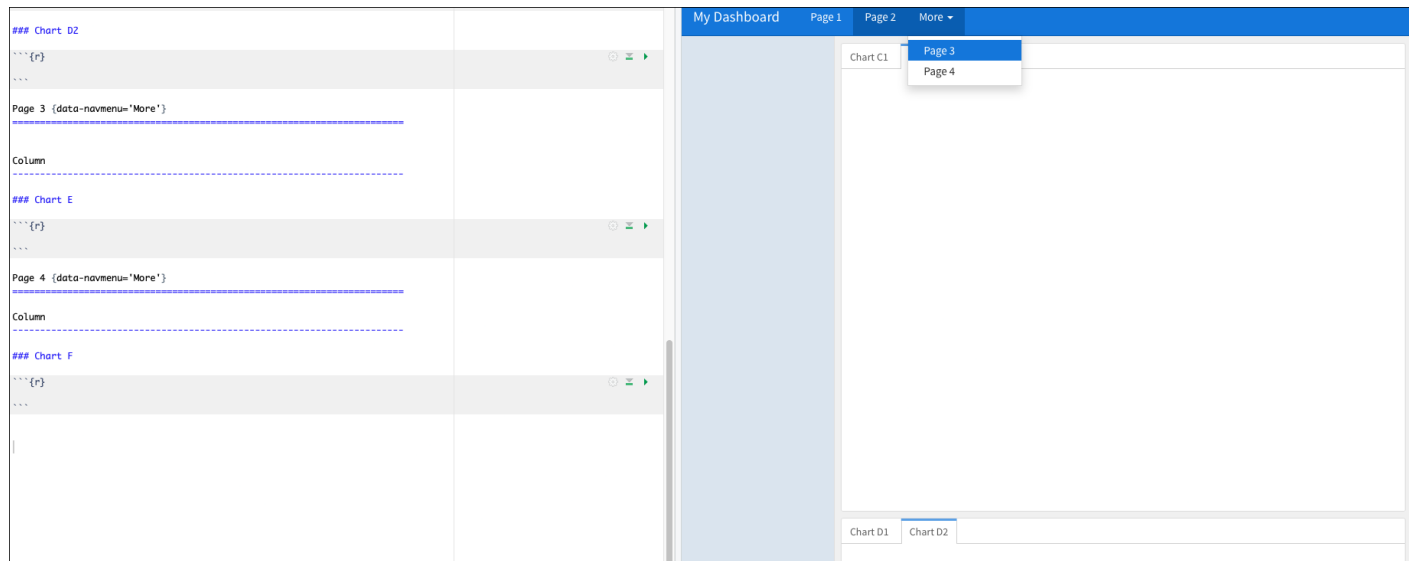
# Menus



data-navmenu=More

|                              |
|------------------------------|
| Page 3 {data-navmenu='More'} |
| Column                       |
| ### Chart E                  |
| ```\${r}```                  |
| ```\n\n                      |
| Page 4 {data-navmenu='More'} |
| Column                       |
|                              |
| ### Chart F                  |
| ```\${r}```                  |
| ```\n\n                      |

# Menus



# Themes



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

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

See the website for more information

**inspectdf** = quickly 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
```

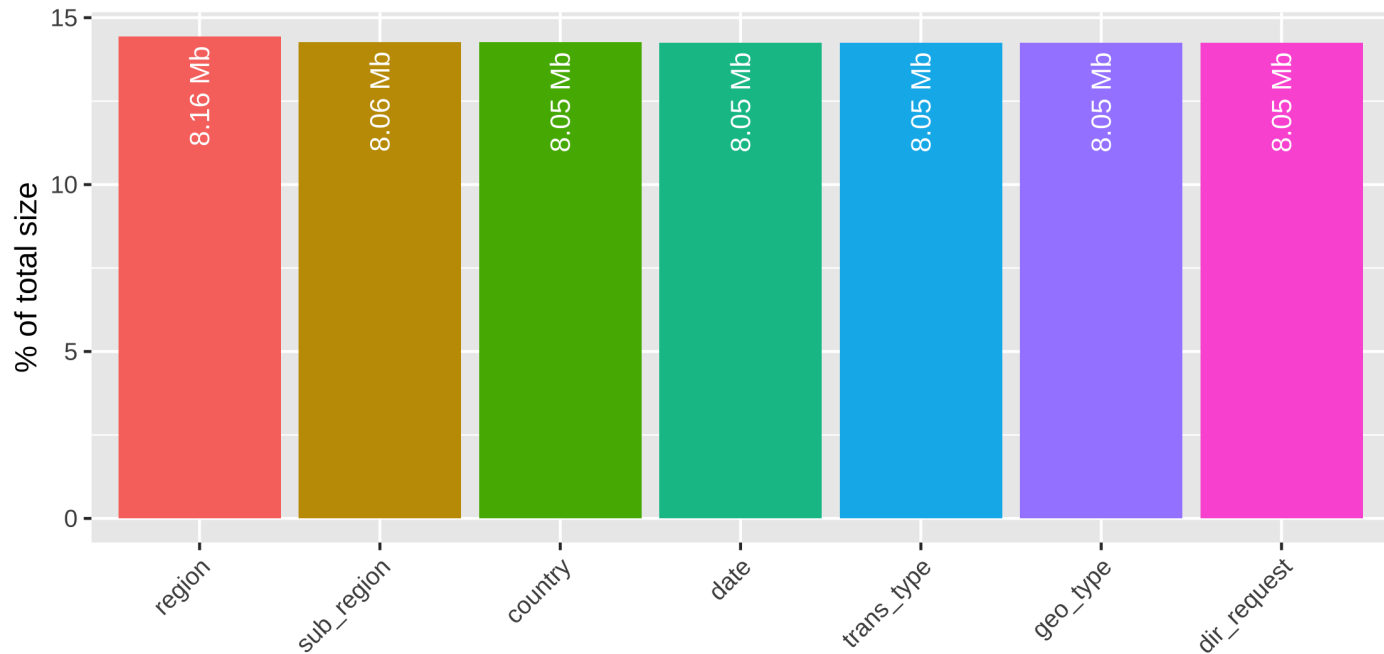
# Dataset size in memory



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

Column sizes in df::.

df::. has 7 columns, 1055293 rows & total size of 56.48 Mb







# 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>