noteMD (how it works) dashboard - dashboard requirements specifications (DRS)

Martin Frigaard

2021-02-12

Contents

Packages

Load the packages below.

```
# app stuff ----
library(shiny)

# data wrangling ----
library(tidyverse)
```

```
library(janitor)



# app downloads -----
library(noteMD)
```

Sidebar

The sidebar contains the two input controls:

- Histogram bins `shiny::sliderInput()`
  - This controls the numnber of `bins` in the `ggplot2::geom_histogram()`
  - accessible in to the `renderPlot({})` function via `input$bins`
- Point size - `shiny::sliderInput()`
  - This controls the numnber of `bins` in the `ggplot2::geom_point()`
  - accessible in to the `renderPlot({})` function via `input$point_size`

Plot 1

```
# draw the histogram with the specified number of bins
carat_df <- diamonds %>% dplyr::select(carat)
carat_df %>% ggplot(aes(x = carat)) + geom_histogram(bins = bins[10])
```

```
#>  Error in layer(data = data, mapping = mapping, stat = stat, geom = GeomBar, : object 'bins' not found
```


Note 1

To get the `noteMD` to render, we need three elements:

1. the `tags$textarea()` includes an `id` that we define with a `markdowninput` prefix (`helpText()` optional)

```
# helpTex = Note...
helpText("Note: make some comments about plot 1...")
# textarea use markdown prefix!
tags$textarea(
  "Please use **markdown** syntax!",
  id    = "markdowninput_plot_1",
  rows  = 3,
  style = "width:100%;"
)
```

1. the `htmlOutput()` gets an `ouputId` with a `htmlmarkdown` prefix (`helpText()` optional)

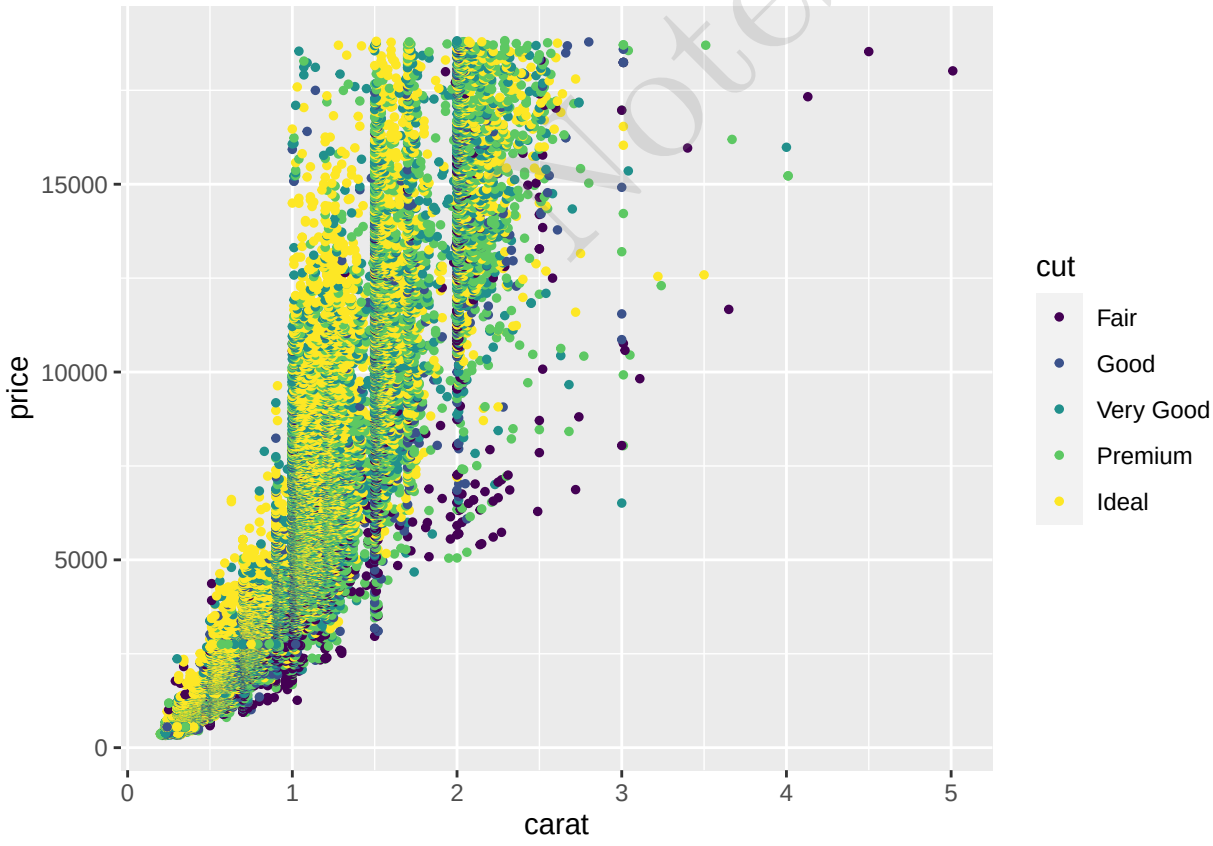
```
# helpText = Preview
helpText("Preview:")
# htmlOutput = use htmlmarkdown_ prefix!
htmlOutput(outputId = "htmlmarkdown_plot_1")
```

1. the `output$htmlmarkdown_plot_1` is from the `htmlOutput(outputId =)` in step 2, and the `reactive({})` contains the `markdowninput_plot_1` from `tags$textarea()` in step 1 (the `noteMD::note_in_html()` helps define the `input` in the reactive)

```
# defines this as a reactive using the `input` value from `textarea`
# (with `markdown_` prefix)and the `outputId` from htmlOutput (with
# `htmlmarkdown_` prefix)
output$htmlmarkdown_plot_1 <- reactive({
  noteMD::note_in_html(input$markdowninput_plot_1)
})
```

Plot 2

```
ggplot2::diamonds %>%
  ggplot2::ggplot(aes(x = carat, y = price, color = cut)) +
    ggplot2::geom_point(size = 1)
```



Note 2

This process is identical to the process above, but we need different names.

1. the `tags$textarea()` includes an `id` that we define with a `markdowninput` prefix, but with the `_plot_2` suffix (`helpText()` optional)

```
# helpTex = Note...
helpText("Note: make some comments about plot 2...")
# textarea use markdown prefix!
tags$textarea(
  "Please use **markdown** syntax!",
  id    = "markdowninput_plot_2",
  rows  = 3,
  style = "width:100%;"
)
```

1. the `htmlOutput()` gets an `ouputId` with a `htmlmarkdown` prefix, but with the `_plot_2` suffix (`helpText()` optional)

```
# helpText = Preview
helpText("Preview:")
# htmlOutput = use htmlmarkdown_ prefix!
htmlOutput("htmlmarkdown_plot_2")
```

1. the `output$htmlmarkdown_plot_2` is from the `htmlOutput(outputId =)` in step 2, and the `reactive({})` contains the `markdowninput_plot_2` from `tags$textarea()` in step 1 (the `noteMD::note_in_html()` helps define the `input` in the reactive)

```
# defines this as a reactive using the `input` value from `textarea`
# (with `markdown_` prefix)and the `outputId` from htmlOutput (with
# `htmlmarkdown_` prefix)
output$htmlmarkdown_plot_2 <- reactive({
  noteMD::note_in_html(input$markdowninput_plot_2)
})
```

Rendering reports

The two .Rmd files contain the following code from the dashboard.

PDF Note 1

```
noteMD::note_in_md_pdf(input$markdowninput_plot_1)
```

PDF Note 2

```
noteMD::note_in_md_pdf(input$markdowninput_plot_2)
```

Word Note 1

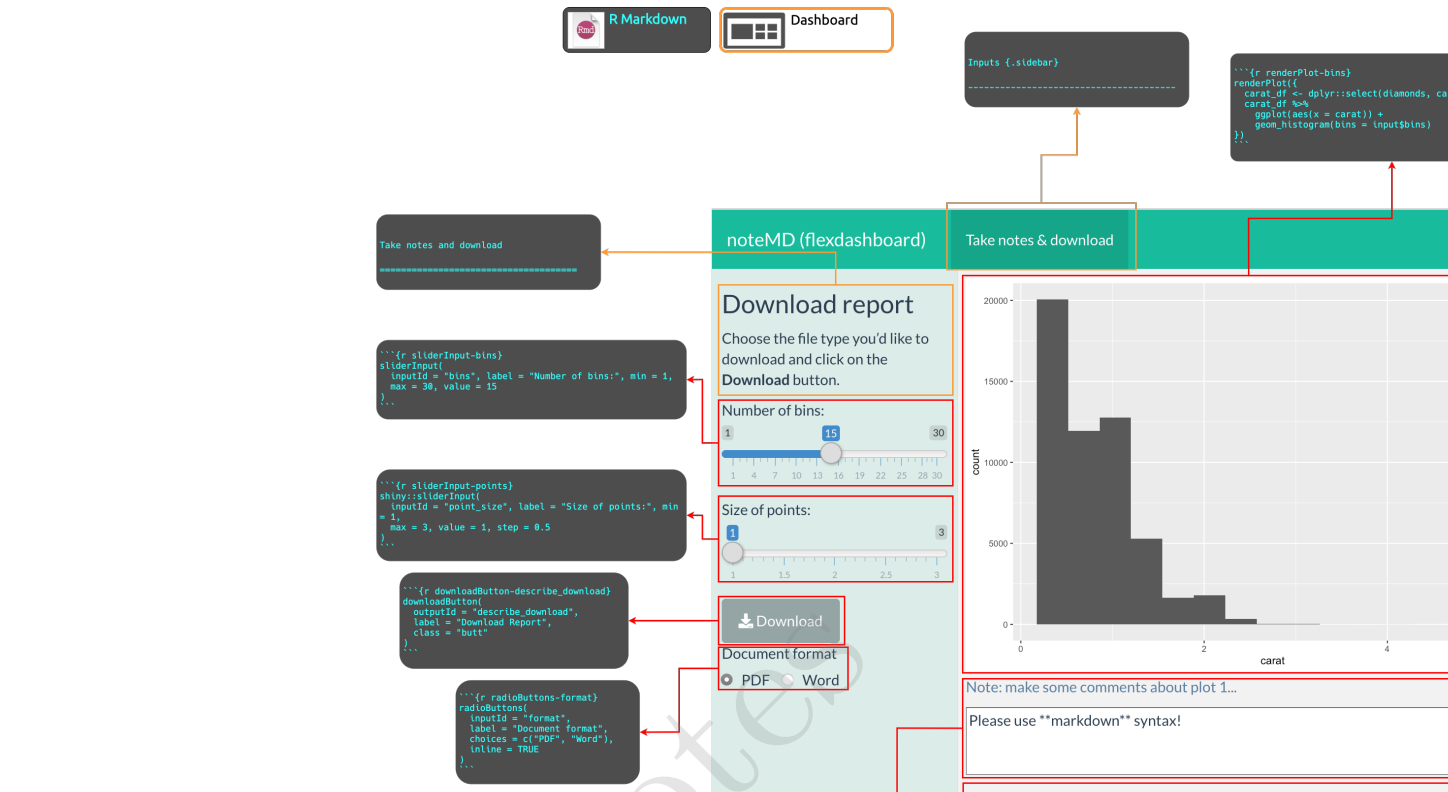
```
noteMD::note_in_md_word(input$markdowninput_plot_1)
```

Word Note 2

```
noteMD::note_in_md_word(input$markdowninput_plot_2)
```

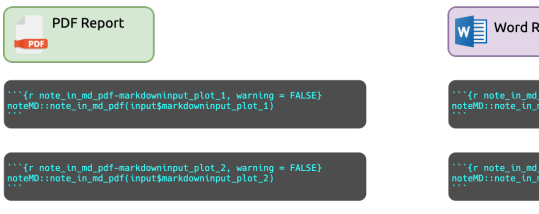
Concept map

The noteMD components in the dashboard are outlined below: Th
and use runtime: shiny in the YAML header



The 'server side' code that renders the reports is presented below:

This code goes into your .Rmd files: *summary_report.Rmd* & *summary_report_word.Rmd*



Martin J Frigaard · Mesa, AZ · 1+503.333.0531 · mjfrigaard@gmail.com