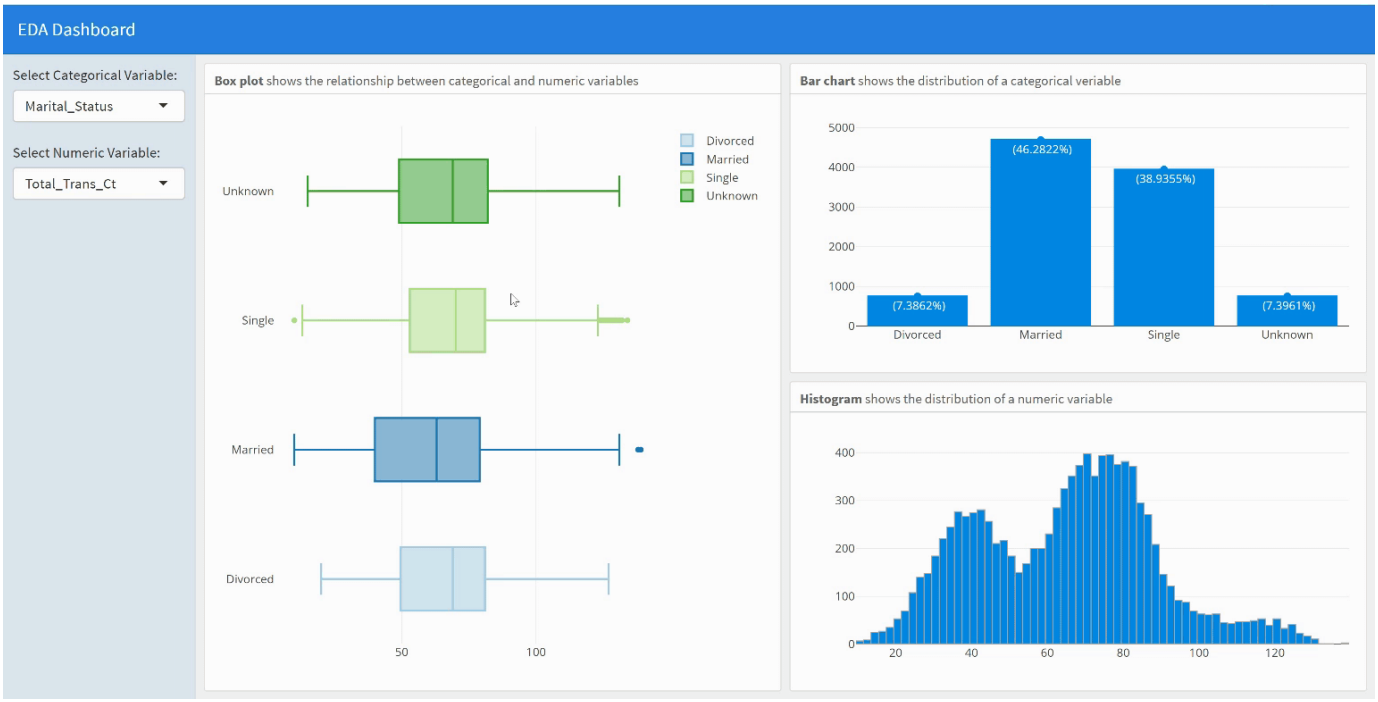
Published in Towards Data Science

Nata Berishvili Follow
Dec 31, 2020 · 4 min read ★ · Listen

Save

Create an Interactive Dashboard with Shiny, Flexdashboard, and Plotly

A step-by-step guide to building your first interactive dashboard



Interactive dashboard

Interactive dashboards empower users to gain valuable insight into key metrics and make data-driven decisions. Interactivity helps optimize the use of dashboard space and updates visualizations automatically as the user changes inputs.

Interactive dashboards with R (Flexdashboard + Shiny)

Flexdashboard is an R markdown file, which can be either static or dynamic. By combining flexdashboard with Shiny, you can write dynamic web applications without any knowledge of HTML, CSS, or JavaScript, using only R and R markdown.

In this article, we walk through the simple steps of building your first flexdashboard Shiny app. We will make this app as simple as possible with two user inputs and three reactive outputs. Once you understand the basics, you can use the same techniques to build more advanced dashboards.

Before we start, make sure you have installed the following libraries: `shiny`, `flexdashboard`, `plotly`, and `dplyr`.

. . .

Step 1. Create Flexdashboard Layout

Initialize a Flexdashboard from R Studio using **File > New File > R markdown > From Template > Flex Dashboard,** save, and knit

```
output:
  flexdashboard::flex_dashboard:
    orientation: columns
    vertical_layout: fill
---

```{r setup, include=FALSE}
library(flexdashboard)
```

Column {data-width=650}

Chart A

```{r}

```

Column {data-width=350}

Chart B

```{r}

```

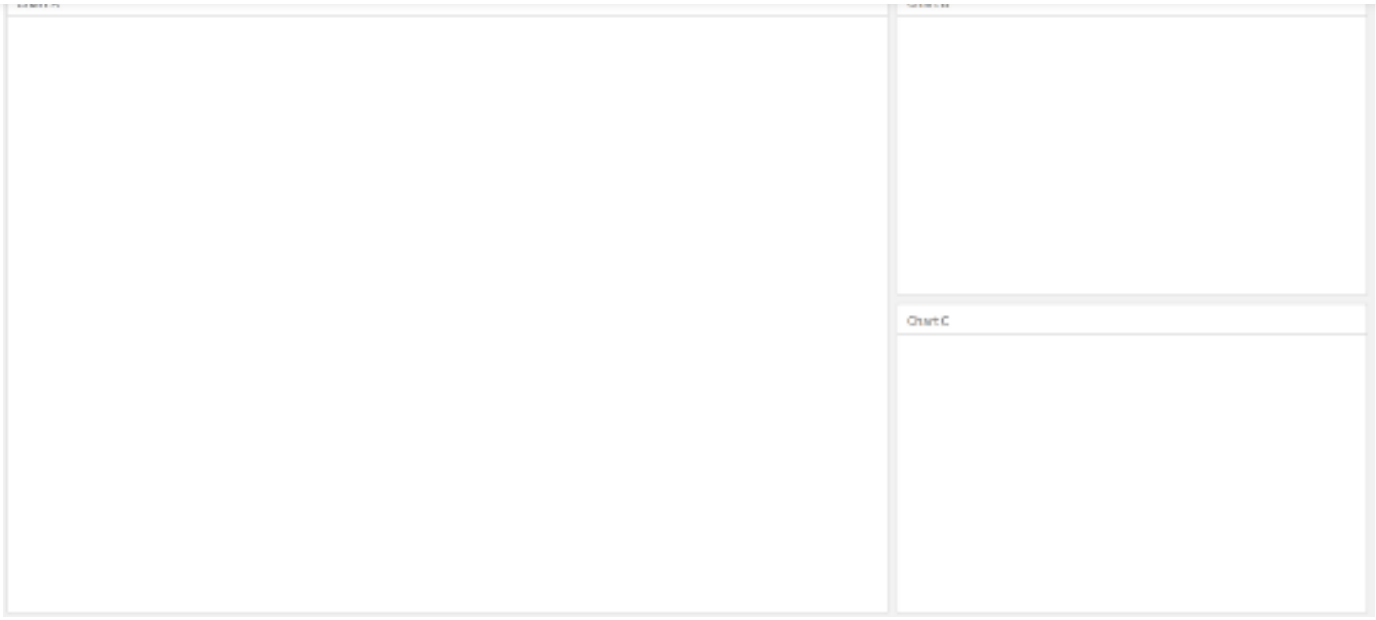
Chart C

```{r}

```
```

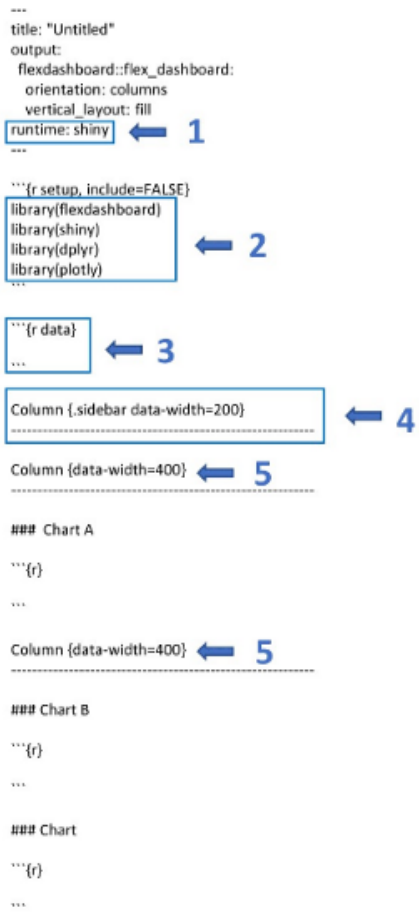
flexdashboard hosted with ❤ by GitHub view raw

This creates a static, two-column dashboard with one chart on the left and two on the right:



Step 1. flexdashboard

Step 2. Customize the Flexdashboard




```
1. Add ***runtime: shiny*** to the YAML header at the top of the document. This specifies that the Shiny package will be used to handle reactive content.

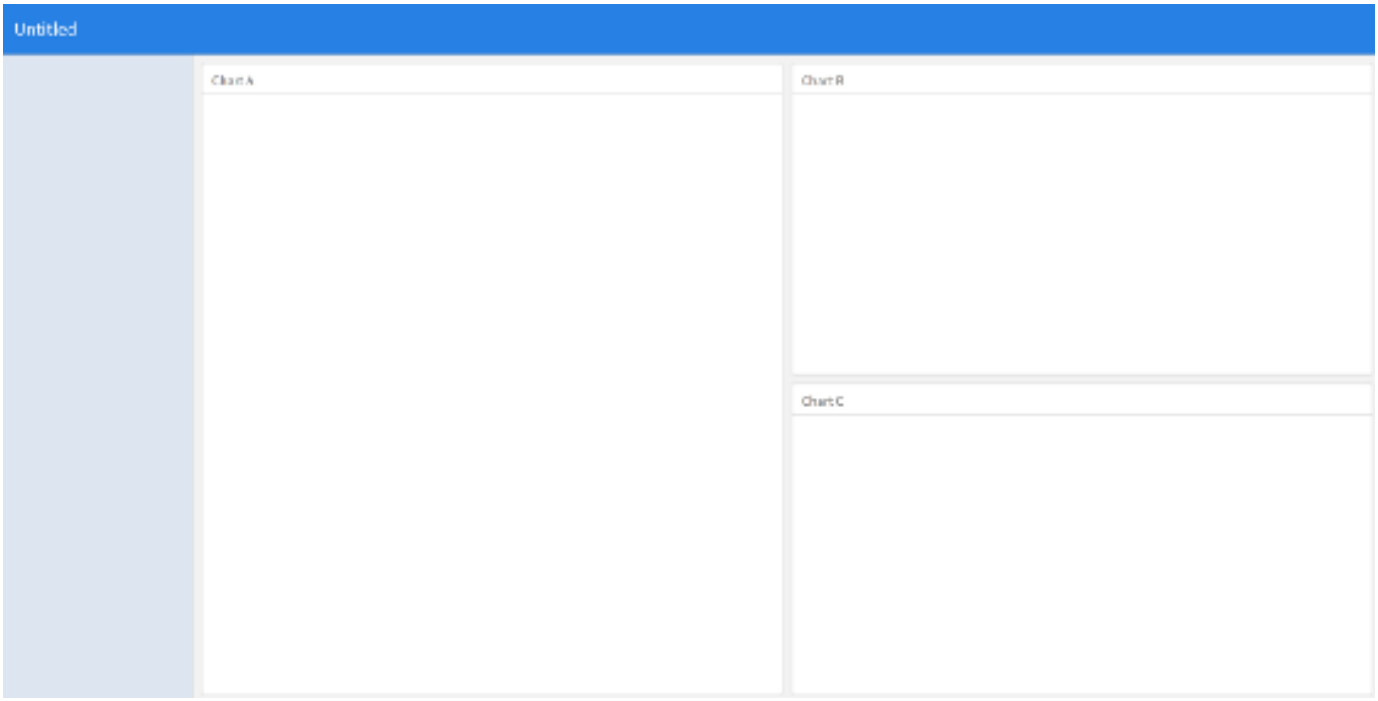
2. Load the libraries ***flexdashboard, shiny, dplyr,*** and ***plotly***.
```

```
4. Add sidebar attribute to the first column of the dashboard. The default width of the sidebar is 250 pixels; let's change it to 200. Later, we will add user input controls to the sidebar.

5. Slightly change the layout dimensions by making each column width equal to 400 pixels.

```

If you did these steps your layout should look like below. Otherwise, you can copy-paste the following template.



Step 2. flexdashboard with sidebar

Step 3. Retrieve and Prepare the Data

For this example, we will use a subset of the Credit Card Customers dataset from Kaggle to explore customer profiles with exploratory data analysis. Let's load and prepare the data under the code chunk of the dashboard.

Remember to store both the dashboard and the data in the same working directory!

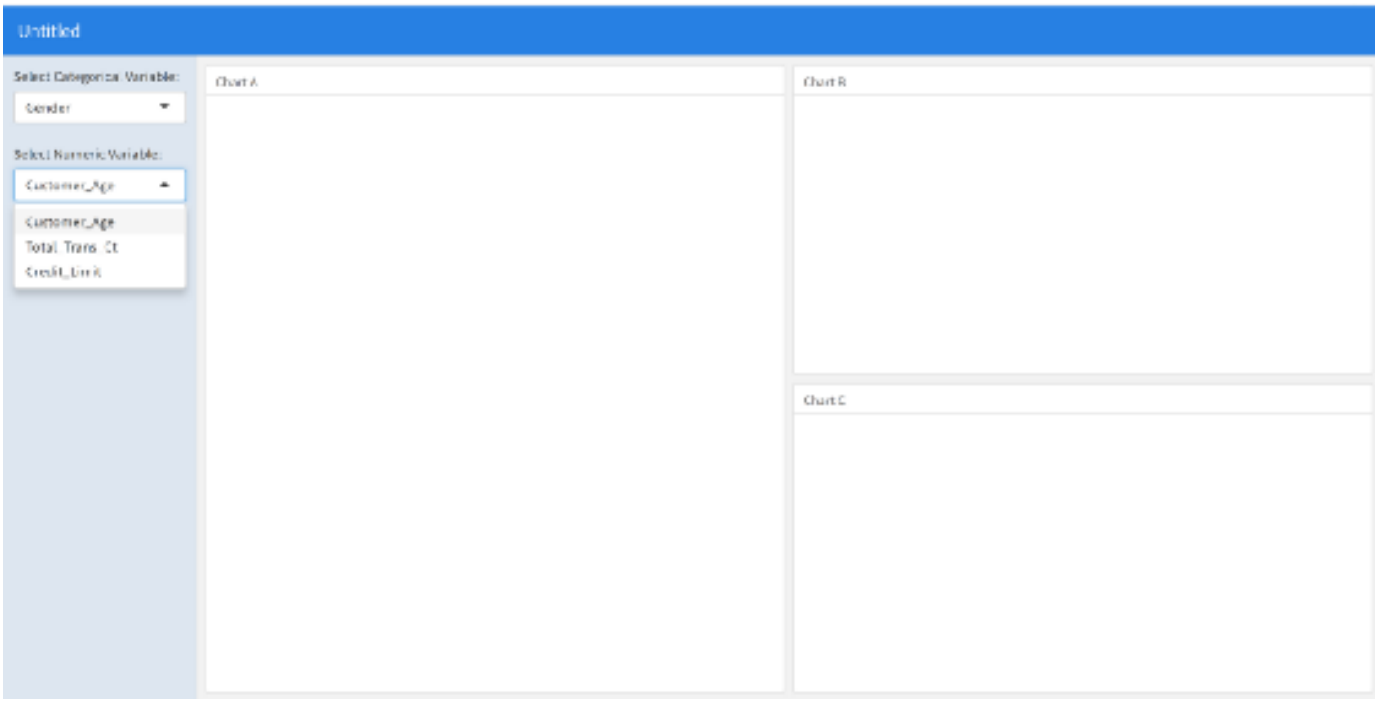
Step 4. Create User Inputs (Widgets)

User inputs are the key components of a dynamic dashboard, driving functionality, user experience, and end results.

SelectInput widget creates a simple dropdown menu. In *SelectInput* widget we specify three arguments: (1) **name**: invisible to user, which we use to access widget's value, (2) **label:** displayed above the dropdown menu, and (3) **choices**: list of values for the user to select.

We will create two ***SelectInput*** widgets in the dashboard's sidebar, allowing the user to select a categorical variable and a numeric variable.

The dashboard should look like this when rendered:



Step 4. flexdashboard with user inputs

Step 5. Create Reactive Outputs — Dynamic Content

Reactivity is what makes Shiny apps responsive, automatically updating whenever the user makes a change. To make an output reactive, we use Shiny's render functions. Changes to inputs automatically render code and update outputs. Shiny offers a wide variety of render functions:

```
renderPlot - renders standard R plots
renderLeaflet — renders leaflet output
renderDT — renders DT DataTables
renderPlotly — renders plotly
```

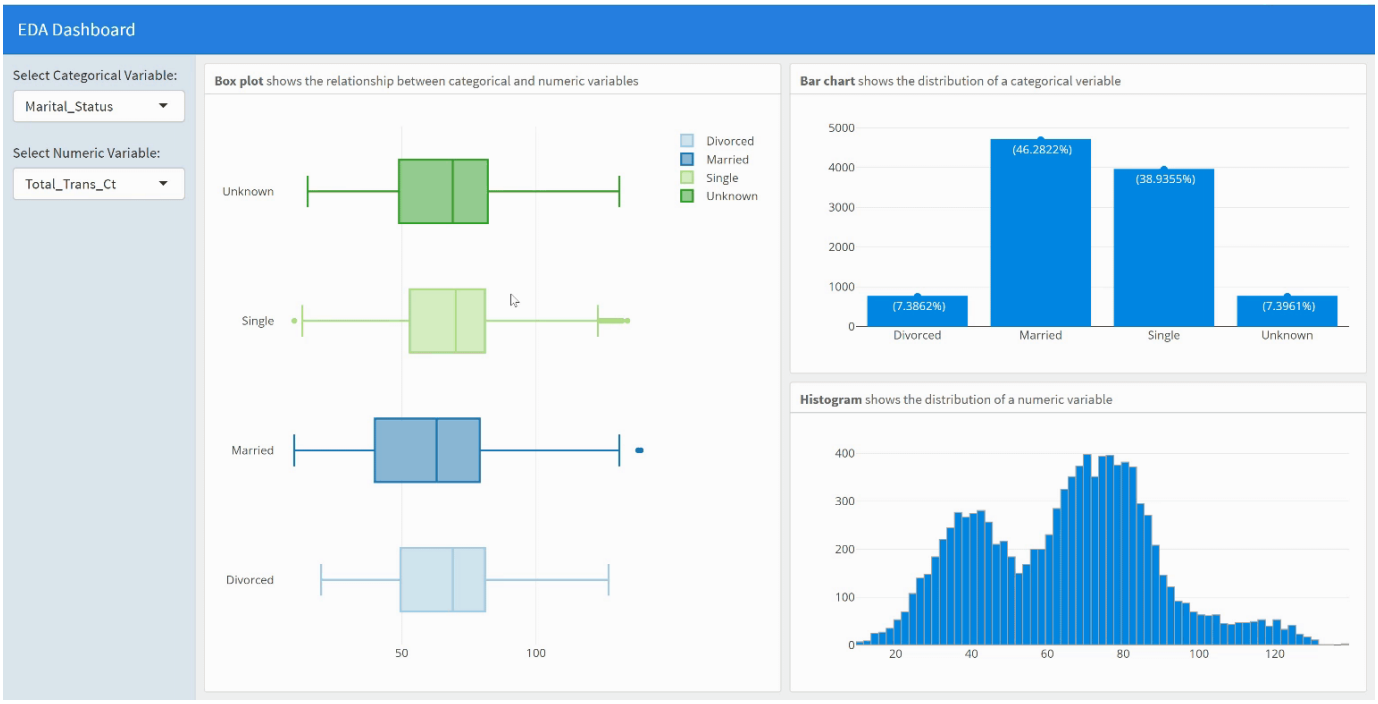
In this project, we will create *Plotly* charts: *(1) boxplot, (2) bar chart, and (3) histogram.* We use *renderPlotly* to insert *Plotly* charts. The charts enclosed in the function renderPlotly will automatically update each time the user changes the corresponding input value.

```
1. The boxplot will react to a change in either widget.
```

*2. The **bar chart** will react to a change in the categorical variable widget.*

*3. The **histogram** will react to a change in the numeric variable widget.*

The code creates the app below:



Step 5. Interactive dashboard

You can find the final code here, and the finished dashboard is available at the following link.

Summary

In this post, we created your first interactive dashboard with flexdashboard and Shiny. Now you have the tools to creatively experiment and design your own dashboard. I hope you enjoyed the post. All suggestions and feedback are welcome and appreciated.

Thank you!

. . .

References

- *Shiny (rstudio.com)*
- *Plotly R Graphing Library | R | Plotly*
- *flexdashboard: Easy interactive dashboards for R (rstudio.com)*
- *Shiny — Introduction to interactive documents (rstudio.com)*