BUILDING INTERACTIVE DASHBOARDS IN R SHINY: AN ANALYSIS OF R'S SHINYDASHBOARD

December 2017

Table of Contents

List of Figures

Executive Summary

For any organization to effectively communicate data and information, others must be able to quickly get the facts they need to make important decisions. One way to do this is to create interactive web applications and present them in a dashboard. Shinydashboard is an R package that allows organizations to develop R Shiny applications, embed them in a dashboard, and publish the dashboard online.

This report discusses the user interface and server components of a shinydashboard, as well as the structure of the dashboard (header, sidebar, and body). Moreover, it examines the flexibility of the dashboard layout and the customization that the Shinydashboard package offers.

An alternative option for creating interactive dashboards, called Flexdashboard, is proposed and compared with Shinydashboard.

The report concludes that Shinydashboard is an excellent software to use to embed R Shiny applications in dashboards since it is user friendly and cost effective, and because it allows individuals to create visually appealing and interactive dashboards that effectively communicate important information.

Introduction

Within any organization, clear and concise communication of data and information is essential for effective strategic planning and informed decision making. With the rise of technology in the workplace, organizations are constantly finding innovative ways to visualize data and convey important information. Building interactive web applications and arranging them into visually appealing dashboards is an effective way to do this as it allows people to quickly see the concepts that are the most important.

The R packages Shiny and Shinydashboard are powerful tools that allow users to interact with and visualize data. Application developers can create graphical output such as charts and tables, as well as text output, and embed it in dashboards for others to explore.

Shiny apps are very customizable, and Shinydashboard allows even more flexibility for the presentation of apps. Applications created in Shiny are reactive to user inputs, which allows users to quickly get the information they need.

This report discusses the structure and components of a Shinydashboard, examines its advantages and limitations, and analyzes the overall effectiveness of it as a tool that helps facilitate strategic planning.

2.0 Analysis

2.1 Background

R is a powerful open source programming language and a statistical computing and graphics environment that is widely used for data analysis and visualization (Venables 2017). It is highly extendable through thousands of packages, and users can define their own functions for even more functionality.

Shiny is an R package that allows people to build interactive applications that tell stories about their data. The apps can be hosted on a webpage or incorporated into documents and dashboards. All Shiny apps consist of a UI (user interface) and a server function; the UI creates the user inputs and controls the appearance and layout of the output and inputs of the app, while the server creates the actual graphical outputs such as plots and tables (Attali 2015). Shiny is easy to learn even without previous web development experience, as there are many learning resources available.

Shinydashboard is an R package that extends the functionality of Shiny; by using Shinydashboard, apps can be displayed in dashboards, with many options to customize the structure and appearance. Like regular Shiny apps, shinydashboards consist of a user interface and server function. What sets shinydashboards apart from regular Shiny apps is the structure; dashboard pages are composed of a header, sidebar, and body, with the content of the body arranged in boxes that can be laid out in rows, columns, or both.

2.2 Shinydashboard Components

Like regular Shiny apps, shinydashboards consist of a user interface (UI) and a server function. The UI and server can be defined in the same app.R file, or they can be separated into ui.R and server.R files, which is more appropriate for larger applications (RStudio 2017). As long as the ui.R and server.R files are placed in the same folder, R will recognize that they are to be run together; clicking 'Run App' in either the ui.R or server.R file will launch the app.

2.2.1 User Interface (UI)

The user interface is the component of the shinydashboard that controls the layout and appearance. This is where the functions `dashboardPage()`, `dashboardHeader()`, `dashboardSidebar()`, and `dashboardBody()` are placed (see section 2.3). The UI is also where the code is written to customize things such as text styling, output size and spacing.

If the app contains user inputs such as checkboxes and dropdown menus, the UI is where the code that generates these is written. While the server function creates the plots, tables or other graphical output, the UI contains the instructions that tell Shiny where to place it.

2.2.2 Server

The server is where the functions that create the output such as tables, plots and graphs are written. Each Shiny output object in the server function is created by adding a ‘render’ command to the usual function name for the desired plot type. For example, if one wanted to create a table of data, they would use `renderDataTable()`. In order to save the output so that it can be called in the user interface, all output in the server must be defined in the form `output$plot_name` where `plot_name` changes for each output object.

The server is also where the reactive data that the functions use to create the output is dynamically updated. By using the `reactive()` function, the original data can be filtered in response to user inputs. This feature makes Shinydashboard an excellent communication tool for organizations since users can quickly interact with plots and graphs based on the data they want to see.

2.3 Shinydashboard Structure

Part of what makes Shinydashboard such a great tool for creating dashboards of Shiny applications is its simple yet flexible layout. Shinydashboards consist of a dashboard page that contains a header, sidebar, and body. As stated in section 2.2.1, the `dashboardPage()`, `dashboardHeader()`, `dashboardSidebar()`, and `dashboardBody()` functions are placed in the user interface file. Here is the minimum code required for a dashboard page's UI:

```
dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
```

The content in each dashboard section can be customized using cascading style sheets (CSS) styling (see section 2.4.1).

2.3.1 Header

In the header, one can include a title, images, and dropdown menus that contain messages, notifications and tasks. For each of these features, there are ways to control the colours, text, and icons. As usual, the dropdown menus in the header can contain dynamic content. Messages, notifications and tasks are useful for dashboards with lots of content; however, a header with just a title and logo is still effective. The following is an example of a header with dropdown menus in the top right corner.

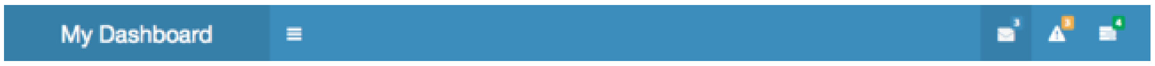


Figure 1: Shinydashboard Header, retrieved from http://rstudio.github.io/shinydashboard/structure.html#header

Headers can be disabled using `dashboardHeader(disable = TRUE)`.

2.3.2 Sidebar

The sidebar of a shinydashboard can contain tabs for menu items, and Shiny user inputs. Sidebars are useful for quickly navigating through the dashboard but are not necessary for smaller applications with just a few tabs since user inputs can be included in the `dashboardBody()` section of the dashboard. If a sidebar is not needed, it can be disabled using `dashboardSidebar(disable = TRUE)`.

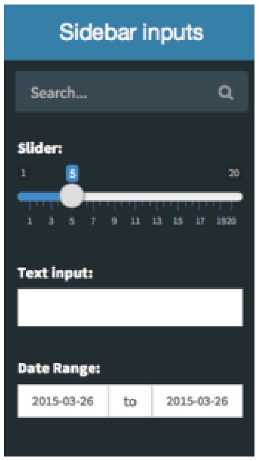


Figure 2: Shinydashboard Sidebar, retrieved from http://rstudio.github.io/shinydashboard/structure.html#sidebar

2.3.3 Body

The body of the shinydashboard contains most of the UI content. The body can include any Shiny content, which is placed in ‘boxes’ that can be laid out in rows and columns. Boxes can have titles, different background and border colours, various heights and widths, and can even be arranged into tabs that users click through.

Depending on the type of layout one wants for the dashboard, boxes are either arranged in rows using `fluidRow()`, columns using `column()` within `fluidRow()`, or a mix of both rows and columns. Rows in the dashboard body have a width of 12, and the default width of a box is 6, so two boxes of width 6 arranged side by side would fill the maximum width of the body. The following is an example of a mixed layout: the two boxes on top are arranged in a row, while the rest are in columns.

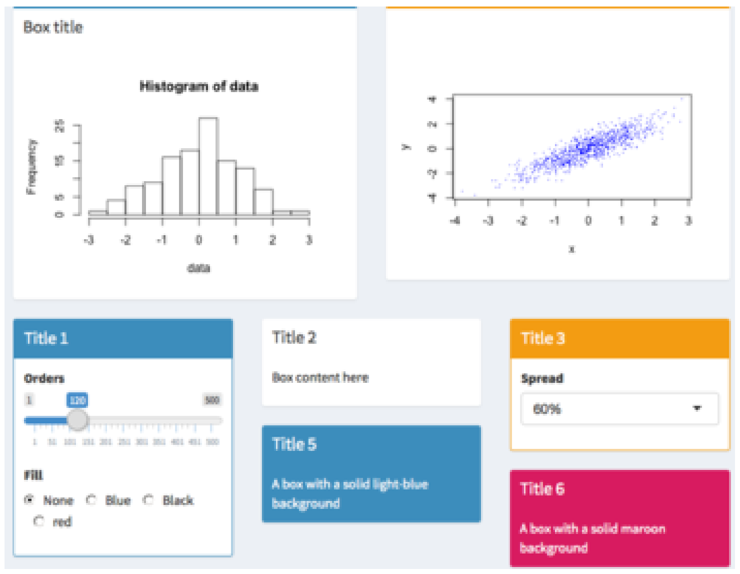


Figure 3: Shinydashboard Row and Column Layout, retrieved from http://rstudio.github.io/shinydashboard/structure.html#body

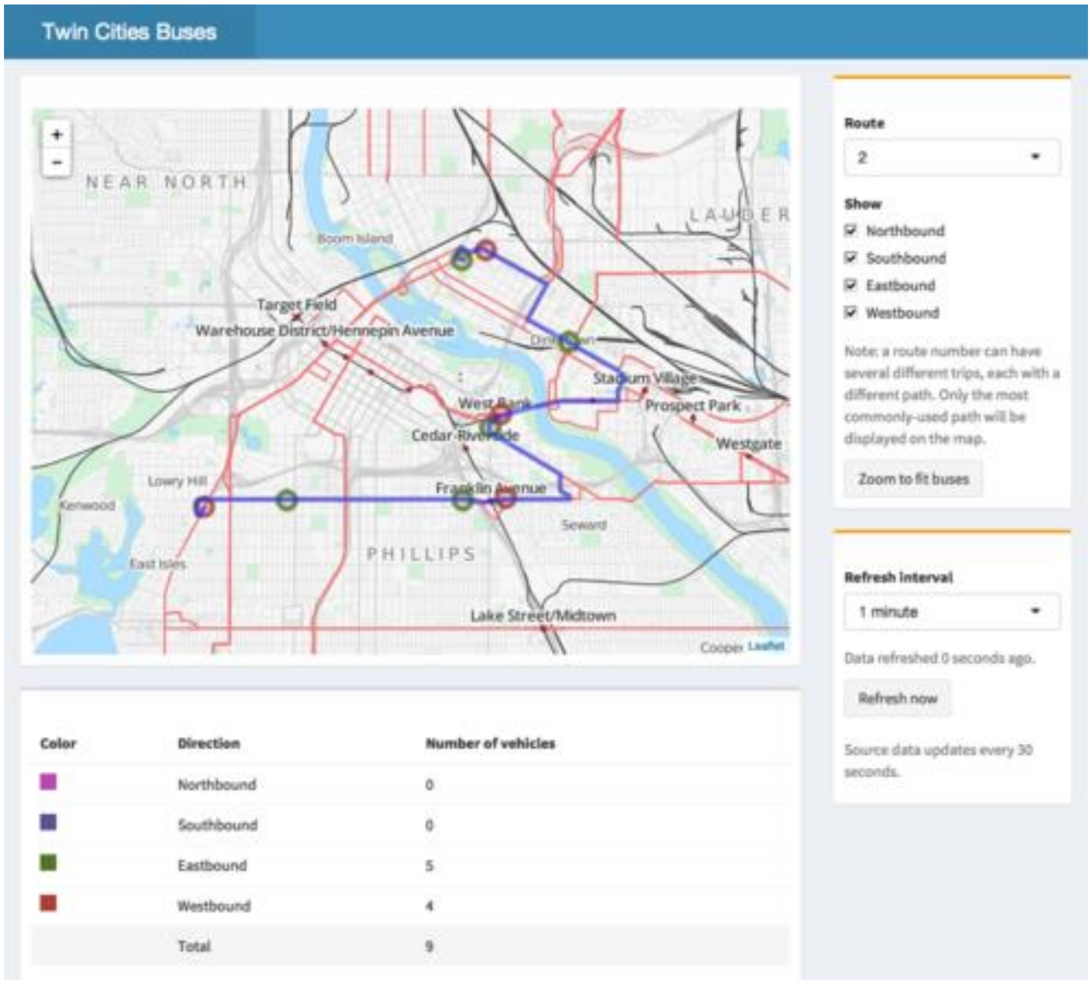
All output that was created in the server function using

`output$plot_name <- renderPlotType()` is called in the UI by

`plotTypeOutput(“plot_name”)` and placed in a box within the dashboard body.

Furthermore, the code for user inputs is written in the `dashboardBody()`.

The following example shows the typical layout of a dashboard.



Route
2

Show
☑ Northbound
☑ Southbound
☑ Eastbound
☑ Westbound

Note: a route number can have several different trips, each with a different path. Only the most commonly-used path will be displayed on the map.

Zoom to fit buses

Refresh interval
1 minute

Data refreshed 0 seconds ago.

Refresh now

Source data updates every 30 seconds.

Color	Direction	Number of vehicles
🟪	Northbound	0
🟦	Southbound	0
🟩	Eastbound	5
🟥	Westbound	4
	Total	9

Figure 4: Dashboard Example, retrieved from http://rstudio.github.io/shinydashboard/examples.html

This example displays the simplicity, yet effectiveness of the shinydashboard package; apps created in shinydashboard are easy to navigate, and there is a graphical output package to fit the data needs of all types of organizations.

The ability to navigate through the dashboard using the sidebar, to mix column and row layouts and to include many different colours, icons, and texts makes it easy for organizations to effectively communicate and share their data with key stakeholders.

2.4 Flexibility and Customization

One of the greatest advantages of Shinydashboard is the ability to customize the appearance of the dashboard and improve the functionality of the apps using cascading style sheets (CSS).

2.4.1 Cascading Style Sheets (CSS)

Most of the CSS code is written in the UI of the app. Using CSS makes it possible to control font type, size and colour, in addition to output sizing and spacing. This greatly improves the appearance and function of the app, especially if plots are resized to fill the dashboard page.

One way of adding CSS code to shinydashboards is to wrap it in calls to `tags$style()` in the UI. For example, the following code will resize the popup labels on a map created using the Leaflet R package, and change the font size of the text so that it fits inside the label.

```
tags$style('.leaflet-popup-content-wrapper {
           font-size: 15px;
           width: 400px;}
           .leaflet-popup-content {
           width: 400px !important;}')
```

Knowing what CSS code to use is not always obvious, but there is usually a solution in documentation online. The ability to style apps using CSS is one of the best features of Shiny and Shinydashboard, and one that gives shinydashboards a professional appearance.

2.5 Learning Resources

Shinydashboard is relatively easy to learn, even if one does not have any previous web development experience. There are many excellent learning resources, such as tutorial videos, articles and online communities. The user-friendliness is another feature of Shinydashboard that makes it so valuable and effective.

2.6 Deployment: shinyapps.io

No matter how beautiful or useful applications and dashboards are, they are ineffective if they cannot be shared with others. Another feature of Shinydashboard is that dashboards can be published online using shinyapps.io, which is a service that runs in the cloud on servers that are operated by R Studio (shinyapps.io 2017). There are five differently priced options based on the available features, but there is a version that allows users to deploy their applications for free.

The combination of a free software and a free deployment service is another benefit of Shinydashboard that makes it desirable for organizations to use.

2.7 Limitations

While Shinydashboard has many advantages, some limitations do exist. One is the maintenance required to keep the packages used to create the output in the dashboard up to date. Since R is open source, developers are constantly improving existing packages and creating new ones. However, installing new or updated packages is very straightforward, so Shinydashboard users can immediately download the updated packages once they are released.

Secondly, fine-tuning aspects of the dashboard such as the placement and sizing of the content takes time, but again, there is usually documentation online that provides a solution.

2.8 Alternative Dashboard Software: Flexdashboard

In addition to the Shinydashboard package, one can build a dashboard in R using Flexdashboard. Flexdashboard is very similar to Shinydashboard, with the main difference being that flexdashboards are created in RMarkdown, an R package that allows users to create HTML or PDF documents and embed R code and output into them (RStudio 2016). Flexdashboards are created in a single RMarkdown file instead of the user interface and server components of a Shinydashboard.

Visually, dashboards created in Flexdashboard look almost identical to shinydashbords. Like Shinydashboard, app content can be laid out in columns and rows, and CSS can be used to style the dashboard content. However, there are a few differences between the two packages that makes creating a dashboard in Shinydashboard more intuitive.

One major difference is that the sizing options for output in Flexdashboard are more restrictive than they are in Shinydashboard. In Shinydashboard, resizing the output so that it fills any browser size, with the option to scroll, has a definitive solution (using CSS). On the other hand, resizing output in Flexdashboard requires more trial and error, such as assigning different heights and widths to output until it fits the dashboard page container, which is time consuming. Furthermore, smaller issues with Flexdashboard such as data tables overlapping with the sidebar when the user scrolls horizontally make Shinydashboard a more suitable option.

From personal experience, the ability to separate Shinydashboard into UI and server functions, and the comprehensiveness of the learning resources online makes programming in Shinydashboard more intuitive and efficient.

3.0 Conclusions

Shinydashboard is an incredibly useful software that has many advantages. It is cost-effective, user friendly, and the dashboards it creates are visually pleasing, customizable, interactive and easy to navigate. Several options exist for organizations to publish their dashboards, including a free version of the shinyapps.io service.

Despite some limitations such as maintenance and time required to fine-tune elements of the dashboard, Shinydashboard proves to be an excellent tool for communicating important information to decision makers within an organization.

References

Attali, D. (2015). Building Shiny apps - an interactive tutorial. Retrieved from https://deanattali.com/blog/building-shiny-apps-tutorial/#1-before-we-begin

RStudio. (2014) shinydashboard. Retrieved from http://rstudio.github.io/shinydashboard/structure.html

RStudio. (2016). R Markdown Quick Tour. Retrieved from http://rmarkdown.rstudio.com/authoring_quick_tour.html#markdown_basics

RStudio. (2017). App formats and launching apps. Retrieved from https://shiny.rstudio.com/articles/app-formats.html

shinyapps.io Team. (2017, December 07). shinyapps.io user guide. Retrieved from http://docs.rstudio.com/shinyapps.io/index.html

Venables, W. N., & Smith, D. M. (2017). *An Introduction to R Notes on R: A Programming Environment for Data Analysis and Graphics* (3.4.3).