PharmaSUG 2021 - Paper EP-122

Interactive Clinical Dashboards Using RStudio®

Syam Chandrala, Allogene Therapeutics.
Chaitanya Chowdagam, Ephicacy Consulting Group, Inc;

ABSTRACT

Data review listings are the standard way of reviewing data by Study Management Team (SMT) involving cross functional groups like clinical science, clinical operations, data managers and biostatisticians. These reports are produced in the form of pdfs, excel or rtf files or sometimes combination all three file formats. Users in SMT team prefers these reports in formats such as excel or pdf. Some user groups don't prefer to excel since they are editable. Data Managers and clinical operations prefer excel since it is easy to identify the data issues.

Statistical programmer has to spend lot of time in generating these reports based on the preference of each user groups, which adds burden on statistical programmer during tight timelines. Flexdashboard package in R is a powerful tool to create these reports in the form of listings and figures and publish them in a webpage using RStudio® Connect. DT and Plotly packages can be used with flexdashboard to make these reports interactive and customizable. In addition to this, other functionalities can be added in such a way that this report can be copied to the clipboard or can be generated in various file formats such as pdf or excel files. This paper will demonstrate how to create and publish interactive reports in the form of a Clinical Dashboard using various R packages.

INTRODUCTION

R is an amazing platform for data analysis and data visualizations. Data visualization is an important tool to understand possible crucial insights from data and it helps in presenting the data effectively to the audience. Dashboards and storyboards are used to communicate large amounts of data visually and quickly. These dashboards can be created using R Markdown with the help of flexdashboard package.

In this paper, we are going to demonstrate step by step process to analyze Clinical data and build a clinical dashboard visualization using various R packages such as haven, DT, ggplot2, flexdashboard. These visualizations can be downloaded in the form of excel, rtf, pdf and can be copied to clipboard.

What we cover in this paper:

1. What is R Markdown?
2. What is Flex dashboard?
3. How to install and load Flexdashboard?
4. R Markdown Template
5. Types of Layouts and examples
6. Components and usage
7. Listings using DT package
8. Example Clinical output
9. How to connect to RStudio Connect® account?
10. How to publish dashboard?

WHAT IS R MARKDOWN?

R Markdown provides a framework to save and execute the code and to generate high quality reports in static and dynamic formats which can be shared.

WHAT IS FLEXDASHBOARD?

Flexdashboard is an R package that allows you to use R Markdown to create a dashboard style output. It handles the task of laying out a dashboard in HTML format. It can be combined with shiny (an R package to create interactive web applications) to build reactive elements. We can use flexdashboard to:

a. publish a group of related data visualizations as a dashboard
b. include wide variety of HTML widgets, R graphics, tabular data, value boxes, text annotations and gauges.
c. specify row and column-based layouts.
d. create storyboards with sequence of visualizations
e. use shiny to create dynamic/interactive visualizations.

HOW TO INSTALL FLEXDASHBOARD IN RSTUDIO

Open RStudio and create a new R script file.

Step 1: Install flexdashboard library using,

install.packages("flexdashboard")

Step 2: Once the package is installed,
Go to File ->New File -> select R Markdown.

Refer Figure 1.

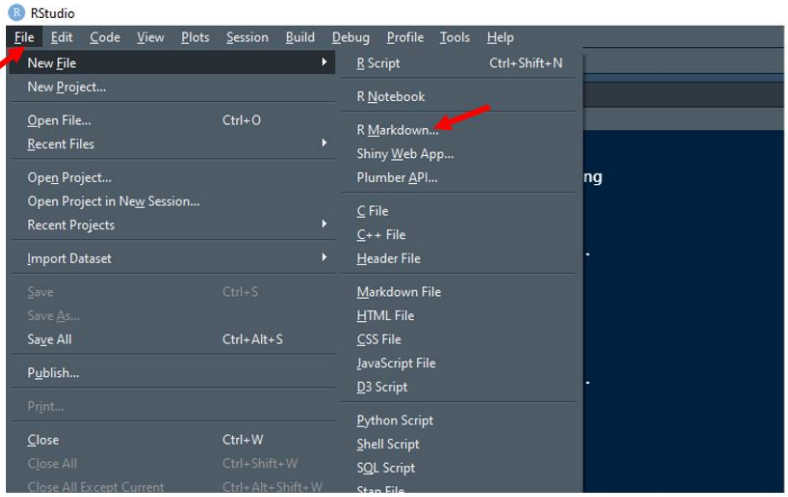


Figure 1.

Step 3: A prompt window will open,
a. From the Left Pane, Select "From Template"
b. Under Template: Select "Flex Dashboard"
c. Click ok

Refer Figure 2

Step 4: A default R Markdown template will be loaded.

Refer Figure 3 for default template.

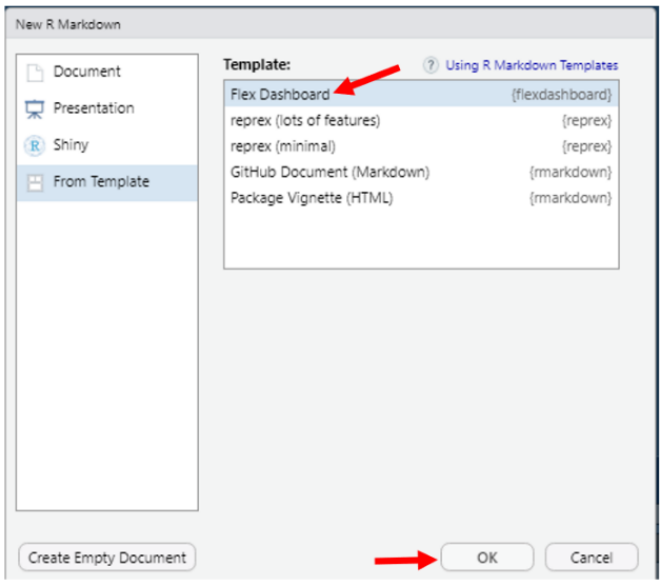


Figure 2.

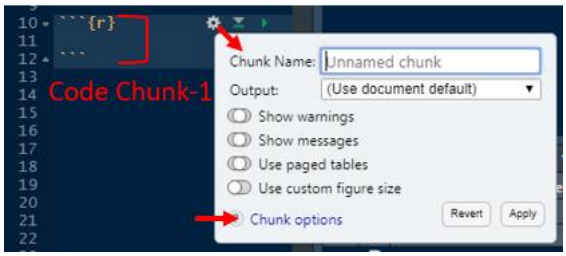
R MARKDOWN TEMPLATE

R Markdown template contains header called YAML header and code chunks.

YAML HEADER section is enclosed by three dashes as shown below. It includes the options to control the dashboard.
By default, YAML header includes Title and output.

```
---
title: "Dashboard 1"
output:
  flexdashboard::flex_dashboard:
    vertical_layout: scroll
    orientation: columns
---
```

CODE CHUNKS starts with ```{r} and ends with 3 back ticks - ```.



If a certain option needs to be frequently set to a value in multiple code chunks, you can consider setting it globally in the first code chunk of your document. To insert a code chunk,
use shortcut key ctrl + alt +i (cmd + option + i on mac OS).
Outputs can be controlled using code chunk options.

To exclude the code chunk in the output, use "include= FALSE" at the beginning of the code chunk.

OUTPUT OF R MARKDOWN TEMPLATE

The default R Markdown template generates HTML output. As shown in screenshot below, the YAML header orientation is set to “columns”. The page is divided into 2 columns based on the template.

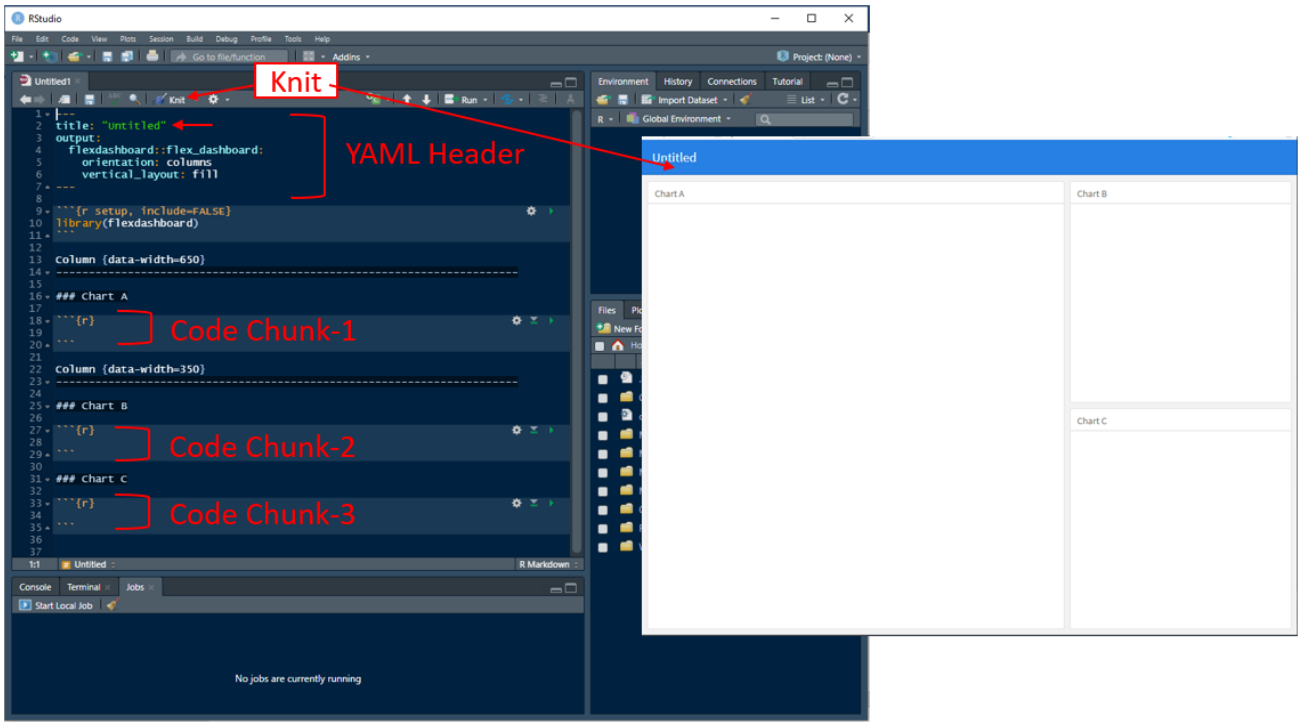


Figure 3. RStudio with R Markdown template (Left) + Knited output of R markdown template (Right)

In order to see the output, we need to knit the file by clicking on Knit icon at the top of the file (or) press ctrl+shift+k.

LAYOUTS AND EXAMPLES

Flexdashboard have different types of Layouts

a. Orientation
    1. Layout by column
    2. Layout by rows
b. Scrolling layout
c. Tabsets

Keeping the scope of this paper in mind we will discuss – (Layout by Column, Tabsets), i.e., Templates 2, 3 shows how to create multiple pages using single hash (#), double dashes(===), or combination of both.

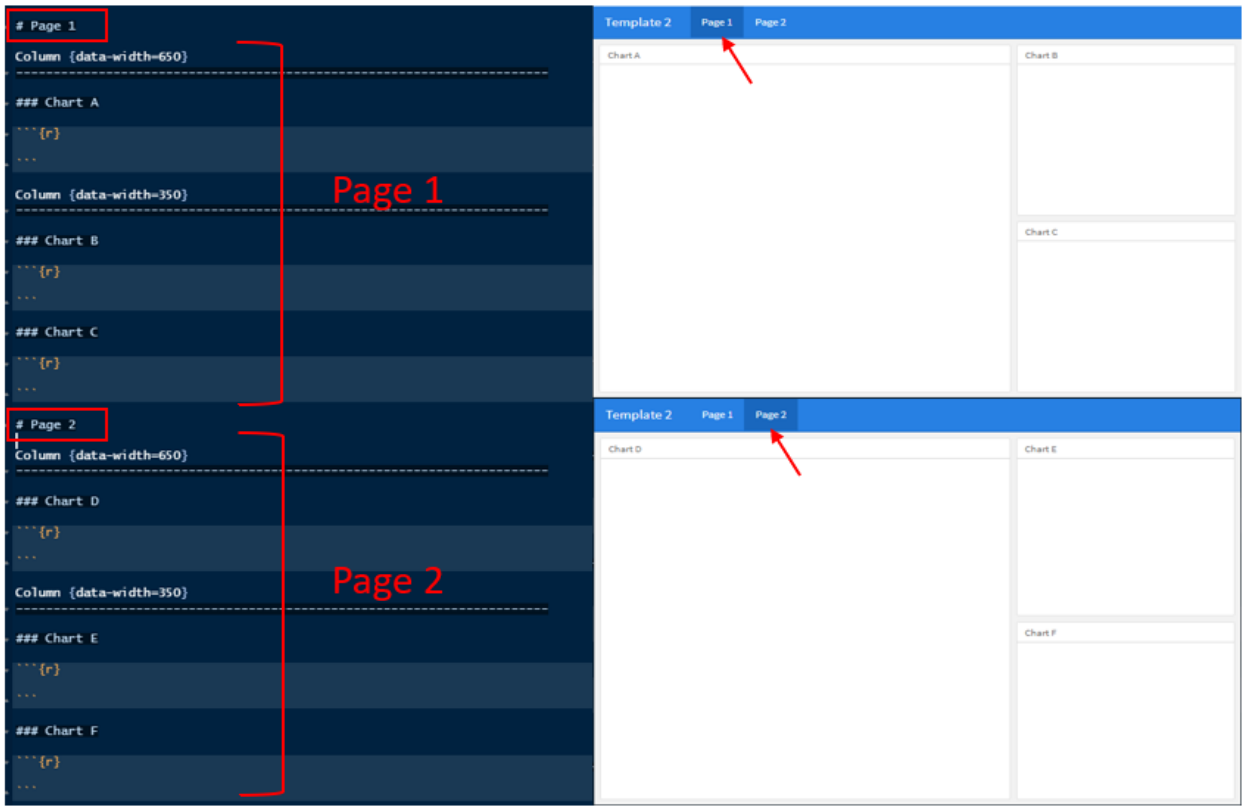


Figure 4. Template 2: Column Layout Dashboard, using Hash, code to left and output to right.

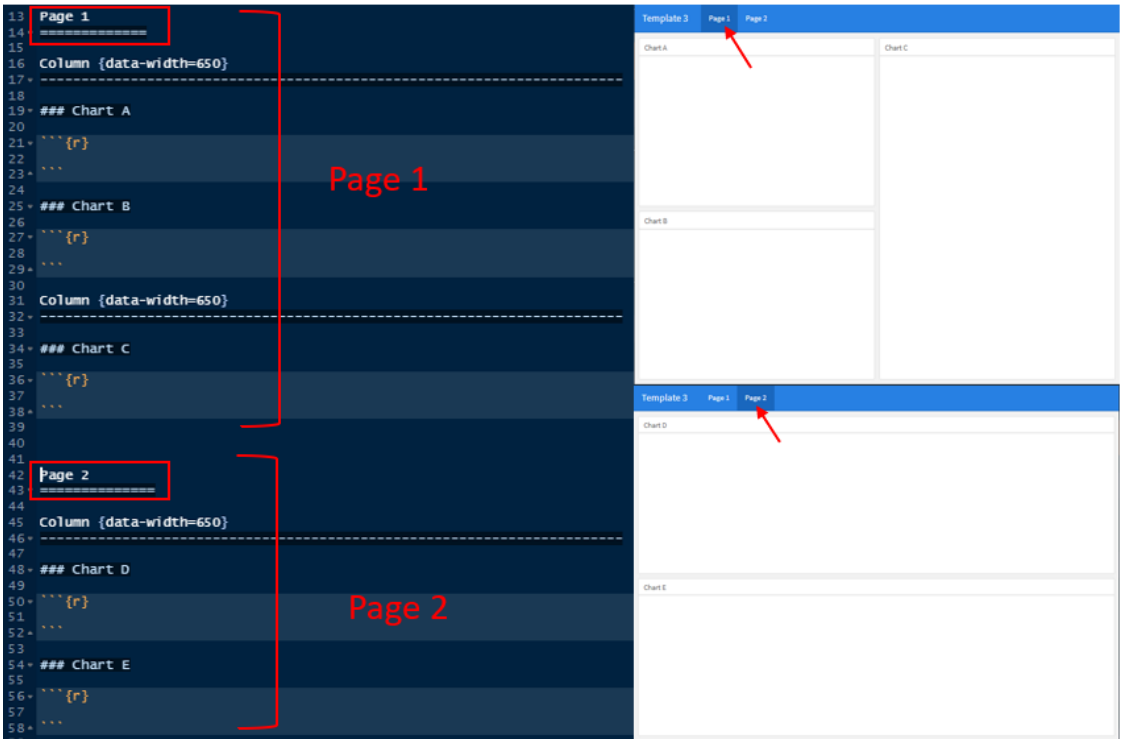


Figure 5. Template 3: Column Layout Dashboard, using double dashes, code to left and output to right.

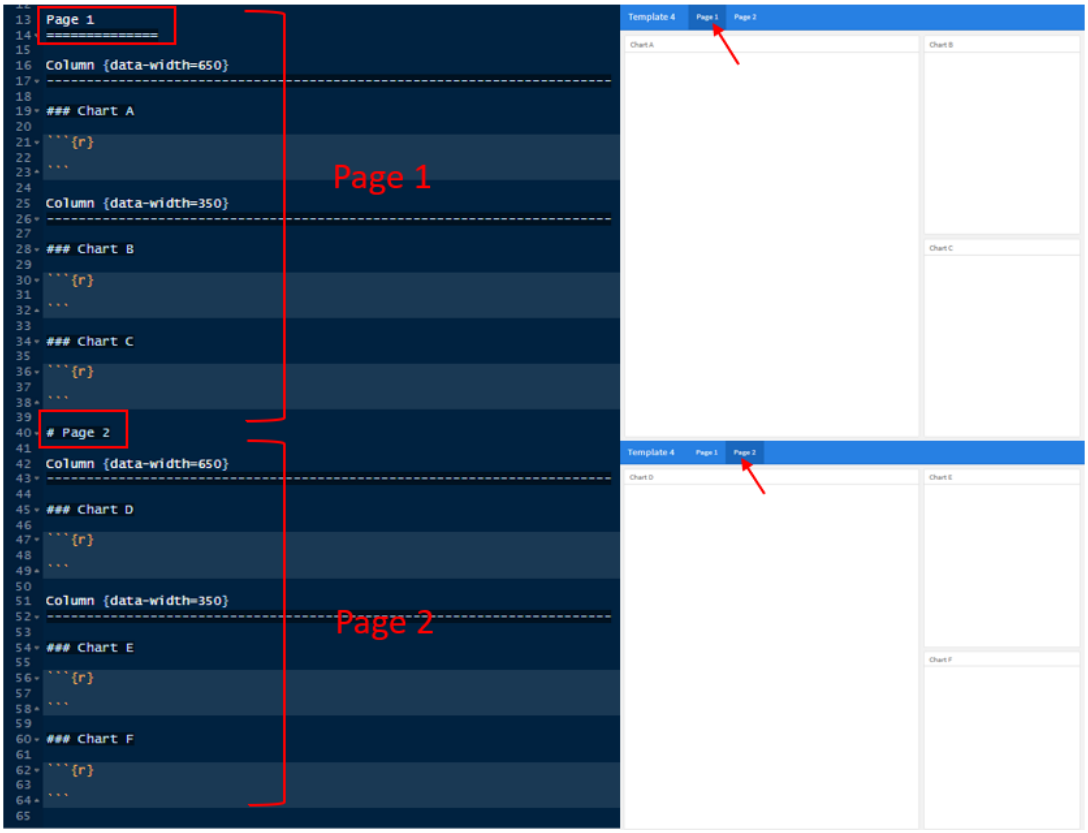


Figure 6. Template 4: Column Layout Dashboard, using combination double dashes and Hash.

Flexdashboard provides flexibility in creating tabs in each page using {.tabset} attribute.

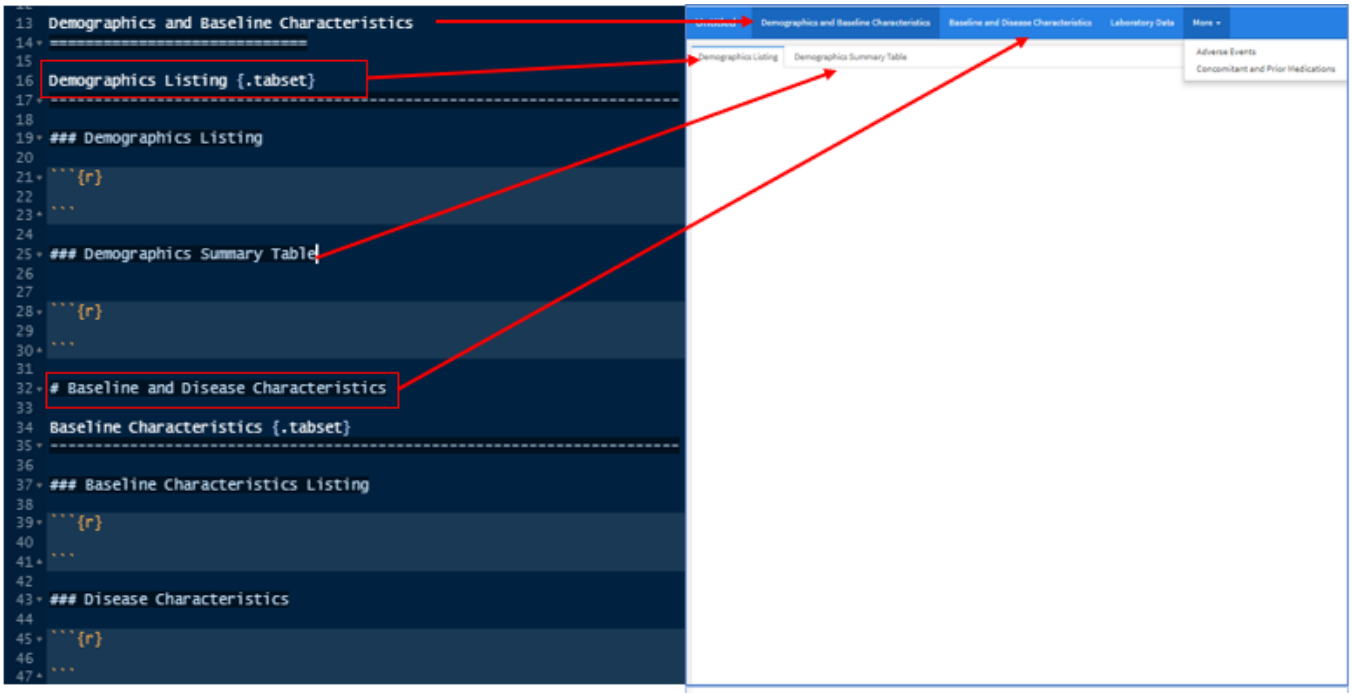


Figure 7. Template 5: Multiple tabs using tabset.

To add more pages with drop down like below screen shot for “Adverse Events” and “Concomitant and Prior Medications” use {data-navmenu=**More**} attribute.

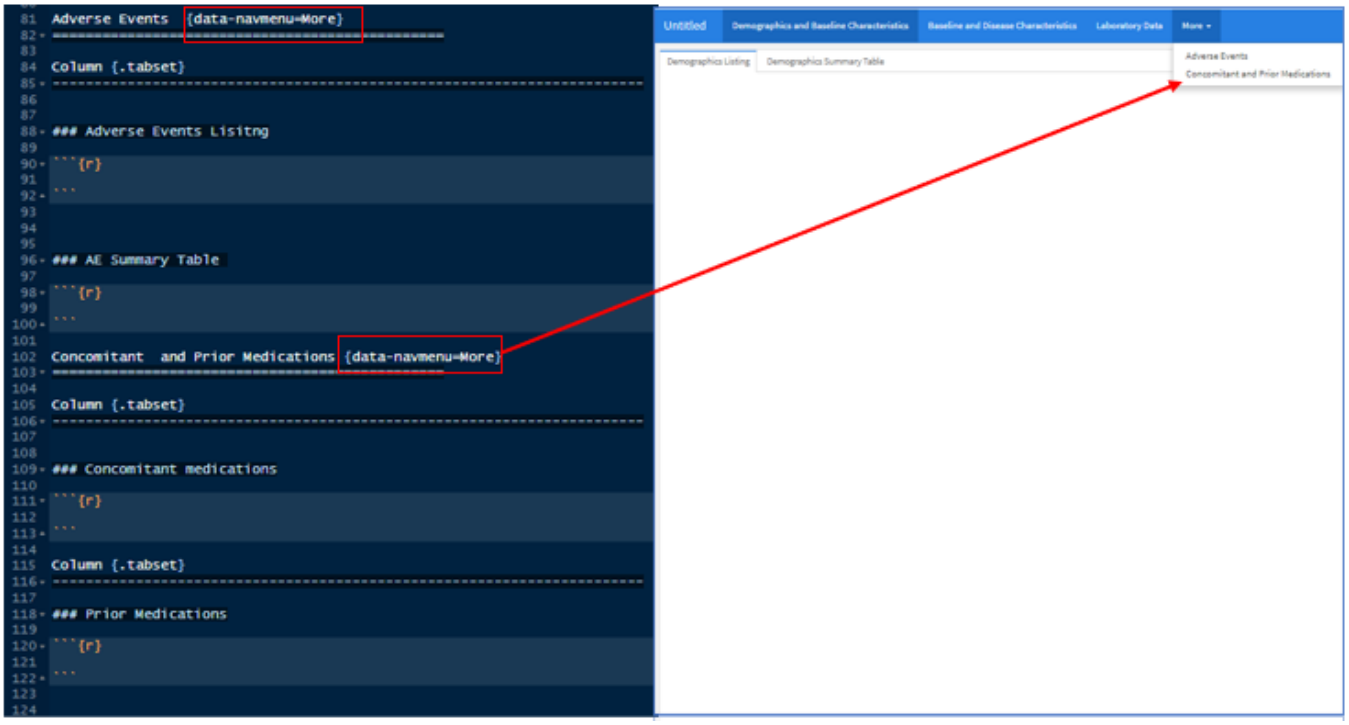


Figure 8. Template 5: Multiple tabs with a drop down using navmenu attribute.

DASHBOARD COMPONENTS

Dashboard components provides the flexibility to control aesthetics of the dashboard like generating interactive data visuals, and provide features like summarizing, sorting, filtering, pagination, and labeling data.

1. HTML widgets
2. R Graphics
3. Tabular Data
4. Gauges
5. Navigation Bar
6. Text Annotations.

HTML WIDGETS

HTML widget is a framework that easily creates binding between Javascript and R.

They include collection of R packages like

1. Leaflet
2. Dygraphs
3. Plotly
4. Rbokeh
5. Highcharter
6. VisNetwork

For Scope of this paper, we have presented examples for all dashboard components except R Graphics (R Shiny).

CLINICAL DASHBOARD EXAMPLE

Now that you have learnt about R Markdown and flexdashboard, layouts and dashboard components it's time to create a clinical dashboard.

Step 1: YAML Header – this header will be loaded on R Markdown template.

Step 2: Code Chunk-1 - Load required R packages to process clinical data, (FLEXDASHBOARD, DT, HAVEN, TIDYVERSE, TPLYR, KNITR, COLLAPSIBLETREE, PLOTLY, RBOKEH) using p_load function from pacman package. Refer figure 9: Code Chunk-1.

Step 3: Read and process SAS datasets into R, using haven package. Refer Figure 9: Code Chunk-2.

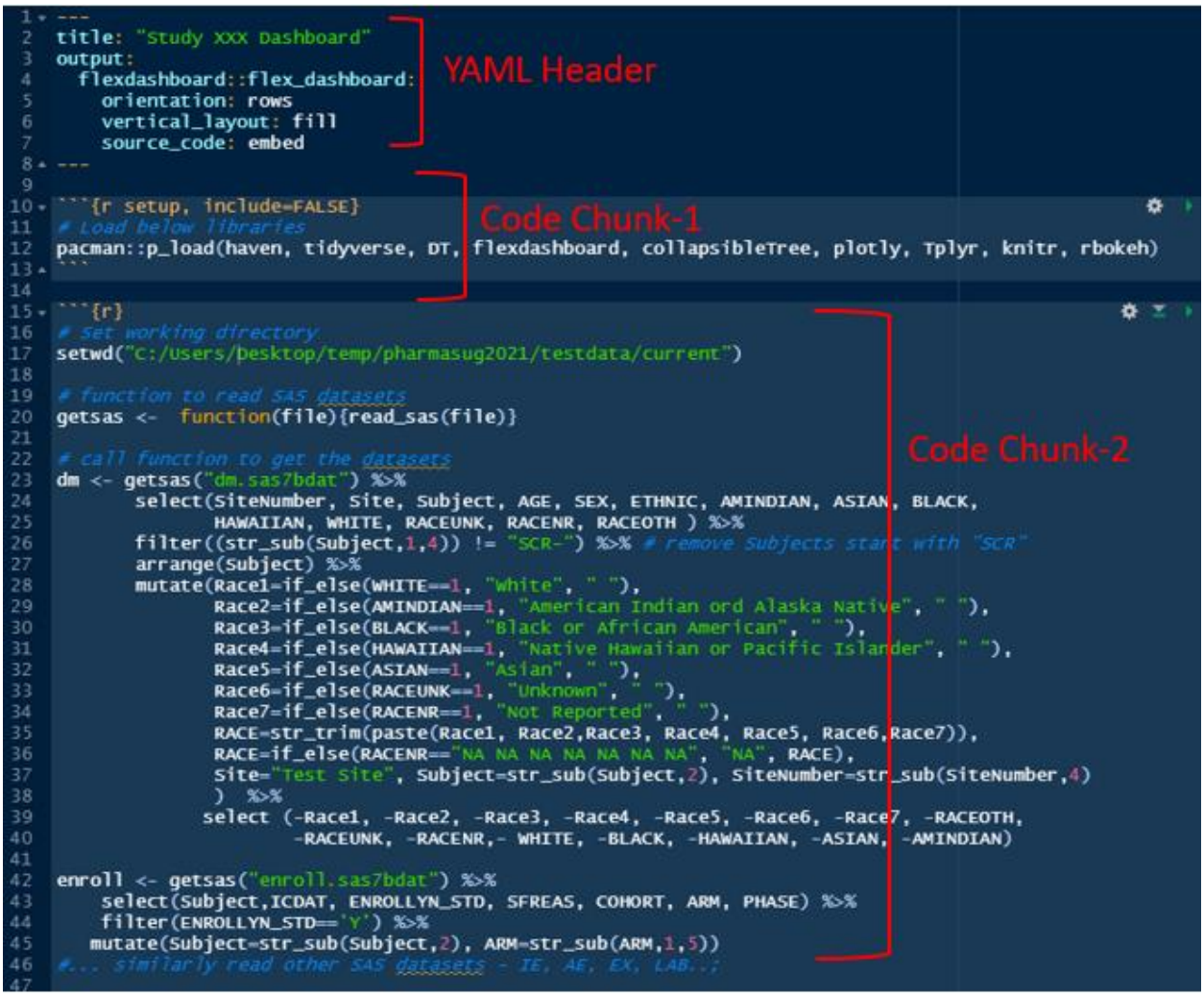


Figure 9.

Step 4: Create printdata function using DT package as shown in the below screen shot. DT package is very flexible and has many options to customize table or print a listing (such as customize table container, table caption, column filter, column names, display rows, styling). For the scope of this paper few options in the below screenshot are used.

```r
```{r}

# create a function to input dataset into DT code for printing interactive table
printdata <- function(dset, trf){
  datatable(dset,  rownames = trf,
            filter = 'top',
            extensions = c('FixedColumns', "Buttons"),
            options = list(dom='Blfrtip',
                           buttons=c('copy', 'csv', 'excel', 'pdf', 'print'),
                           lengthMenu=list(c(10, 50 , 100, -1), c(10, 50, 100, "All")),
                           scrollY="900px",
                           scrollX=TRUE,
                           fixedColumns=list(leftColumns=5, rightColumns=0)
                         ),
          )
}
```
```

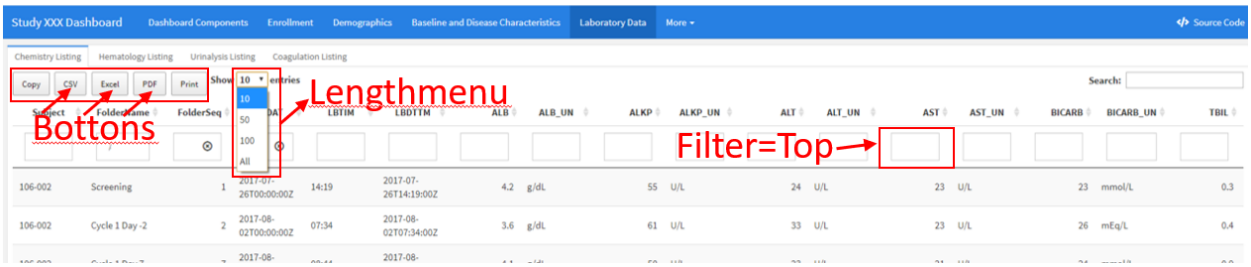
Figure 10. Printdata function code, using Datatable (DT) package.

Datatable (DT package) Options :

Filter=top – Adds search box below each column header where the user can type and filter values in that column.

Length menu – Adds drop down with List of rows per page can be picked from drop down.

Buttons –User can download data to various file formats such as CSV, Excel, PDF or COPY to clipboard.



Step 5: Build the dashboard. In this step, the code creates “Dashboard Components” and “Enrollment Page” (Refer Figure 11 for Code and Figure 12,13 for Output).

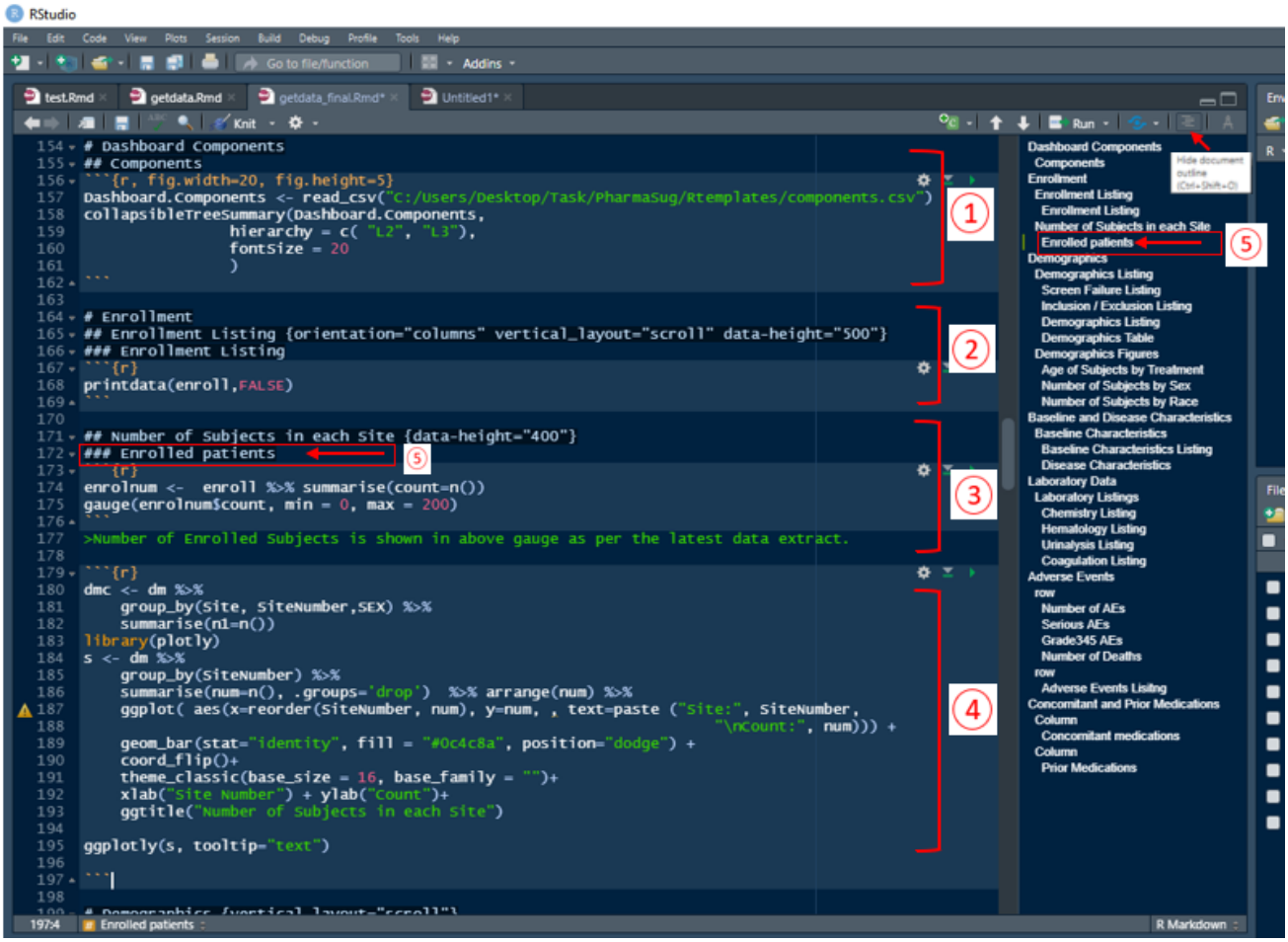


Figure 11. Code to create Enrollment Page.

Figure 11 - Block 1 code: Using Single Hash (#) to create new page “Dashboard Components”. the code reads CSV file using READ_CSV function, and by using collapsibleTree package below graph is generated under “Dashboard Components” page.

Page as shown below shows components of flexdashboard and various packages used in HTML widgets covered in this paper.

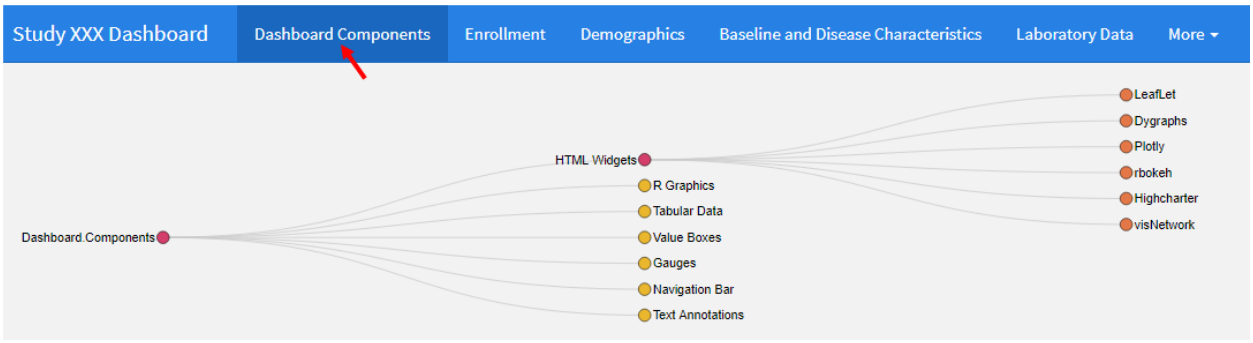


Figure 12. Collapsible tree

Figure 11 - Block 2 code: Using Single Hash (#) create Enrollment page. Three different outputs (1 Listing and 2 Graphs) are created in a single page using column layout. Using Printdata function, Enrollment Listing is printed as shown in Figure 13 (Marked in Red 2).

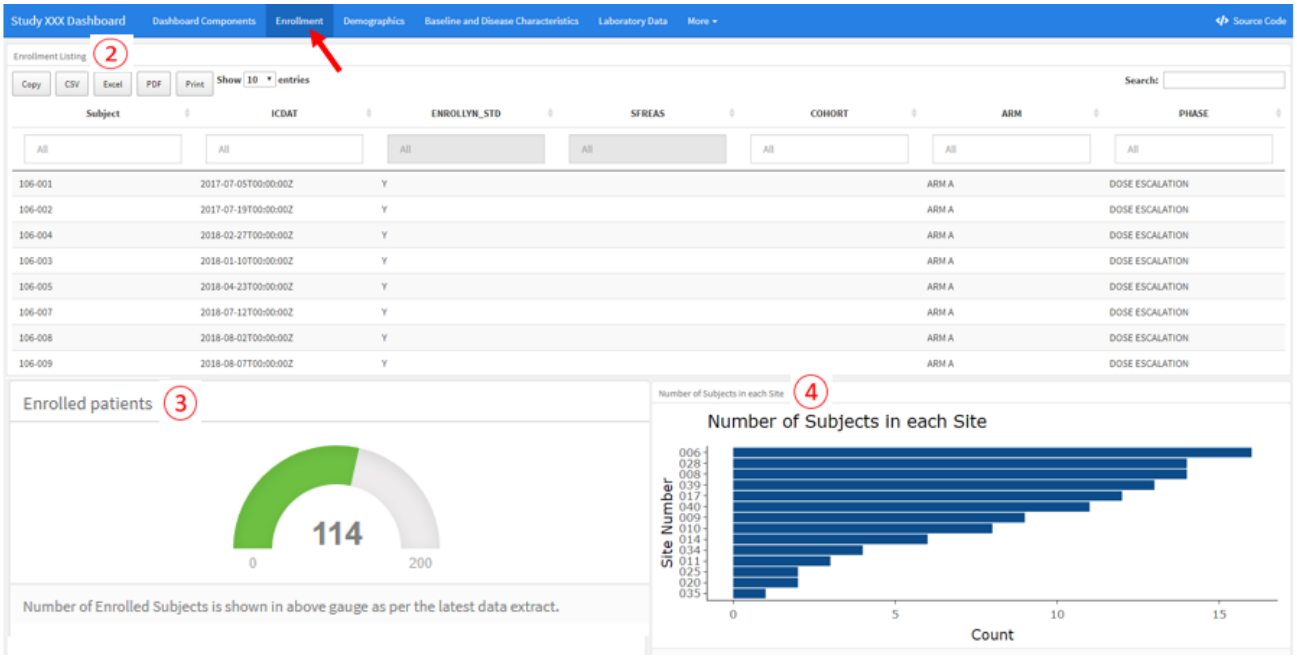


Figure 13. Enrollment page

Figure 11 - Block 3 code: In Enrollment page, Patient numbers are plotted using “Gauge” dashboard component. Footnote is added using “Text Annotation” starting with > symbol in the code.

Figure 11 - Block 4 code: In Enrollment Page, interactive graph “Number of Subjects in each site” was summarized from DM dataset and plotted using ggplotly function from plotly package.

Step 6: This step creates “Demographics Page” using “Tabset” attribute. It creates three demographic listings and one demographic table in multiple tabs.

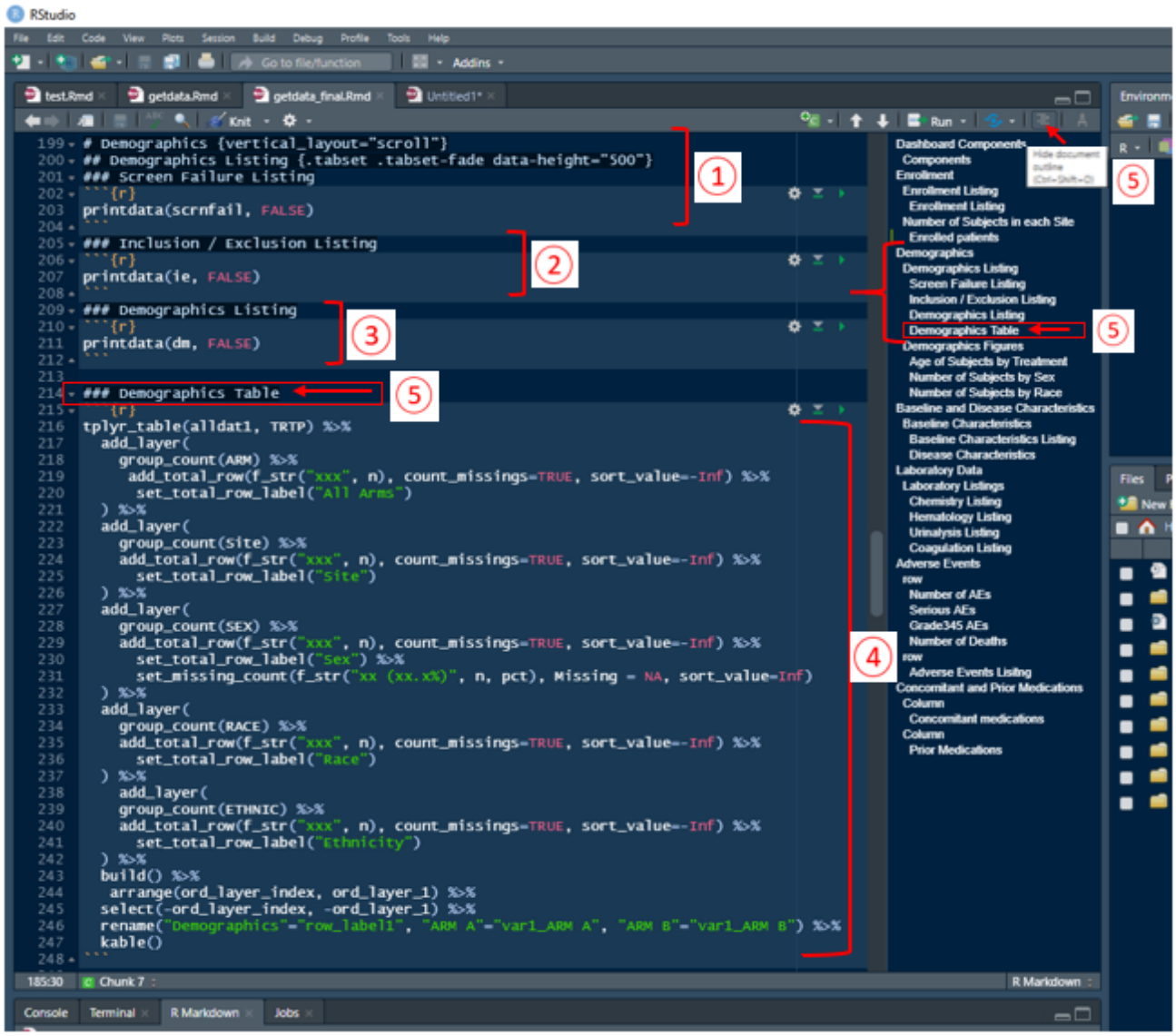


Figure 14. Multiple tabs using {.tabset} attribute.

Figure 14 - Block 1 code: Creates Demographic page using # and vertical scroll option. multiple tabs are created using {tabset} attribute.

Using Printdata function, Screen failure Listing is printed as shown in Figure 14 (Marked in Red 1). Output of this code is shown in Figure 15 (1st tab Marked in Red 1).

Figure 14 - Block 2 code : Using Printdata function, Inclusion / Exclusion Listing is printed as shown in Figure 15 (Marked in Red 2). Output of this code is shown in Figure 15 (2nd TAB Marked Red 2).

Figure 14 -Block 3 code: Using Printdata function, Demographics Listing is printed as shown in Figure 14 (Marked in Red 3). Output of this code is shown in Figure 15 (3rd TAB Marked Red 3)

Figure 14 - Block 4 code: Using tplyr_table function in Tply package, we can summarize counts by groups, add rows, set row label. Output of this code is shown in Figure 15 (4th Tab Marked in Red 4).

Figure 14 - Block 5 code: Shows flexdashboard code view, to expand and collapse R code outline and get high level break down by code chunk, click on the right top corner red arrow as shown in Figure 14. In the right pane you can view "Demographic table" when you click on it, it will take you to the appropriate code chunk.

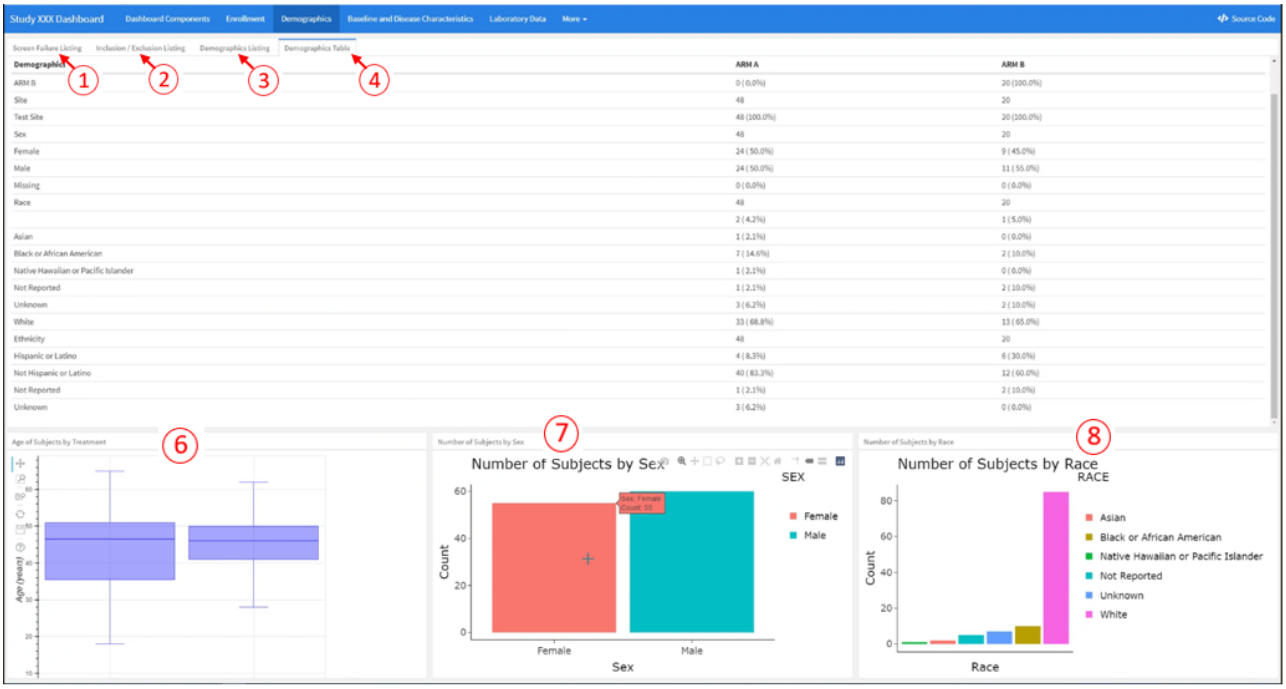


Figure 15. Multiple tabs using {.tabset} attribute.

Figure 16 - Block 6 Code: Is a continuation code of "Demographics" page, under "Demographic table" tab using column layout, one table and three graphs are presented in the dashboard as shown Figure 15 (Marked 6 in red). ly_boxplot function from rbokeh package produces boxplot – Age of subjects by Treatment.

Figure 16 - Block 7 Code: "Number of Subjects by sex" Bar plot using Plotly package. Output of this code is shown in Figure 15 (Marked 7 in Red).

Figure 16 - Block 8 Code: "Number of Subject by Race" Bar plot using Plotly package. Output of this code is shown in Figure 15 (Marked 8 in Red).

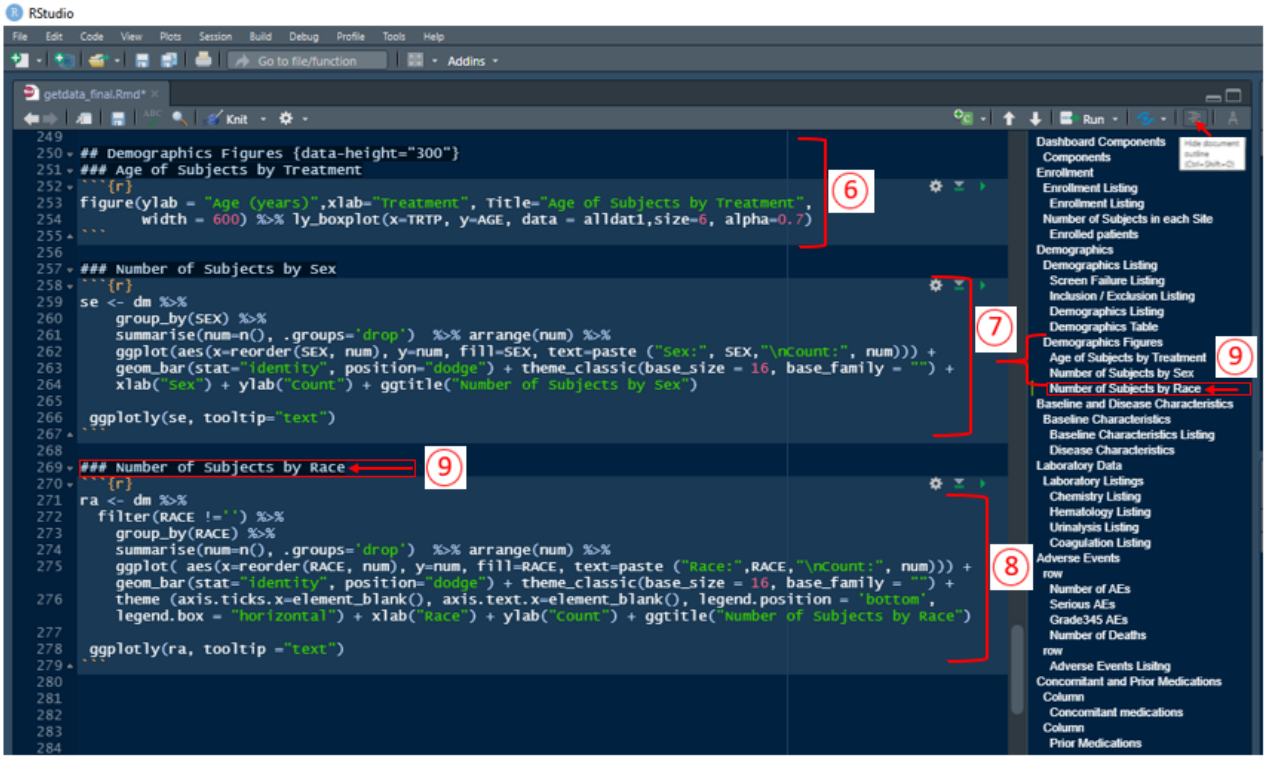


Figure 16. Graphs using rbokeh and plotly packages.

Step 7: One of the dashboard components, “Value boxes” are very useful in presenting critical information about the data. Below Figure 17 has output with valueboxes that gives high level summary of “Number of AEs”, “Serious AEs”, “AE Grade >=3” and “Deaths”.

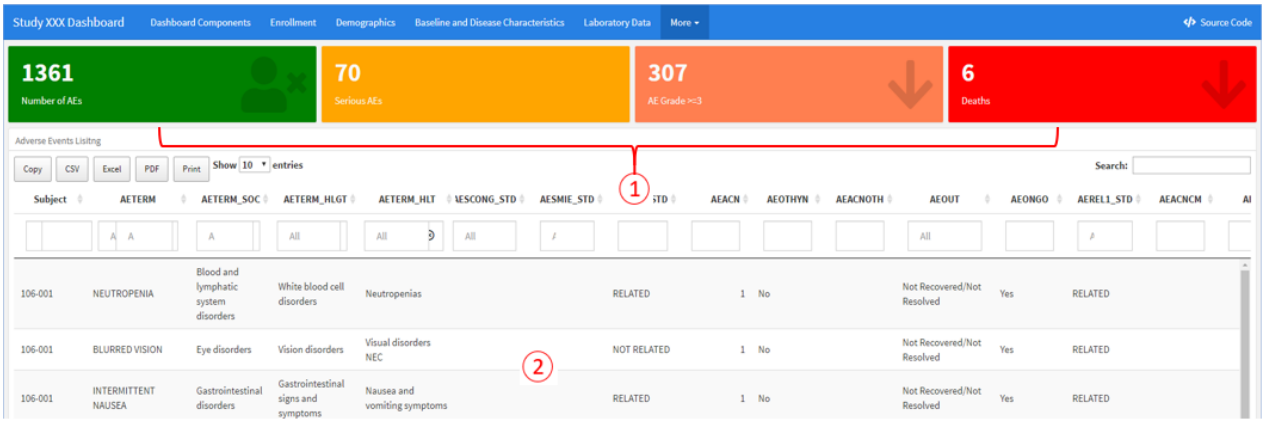


Figure 17. Adverse Events Listing with 4 Value boxes.

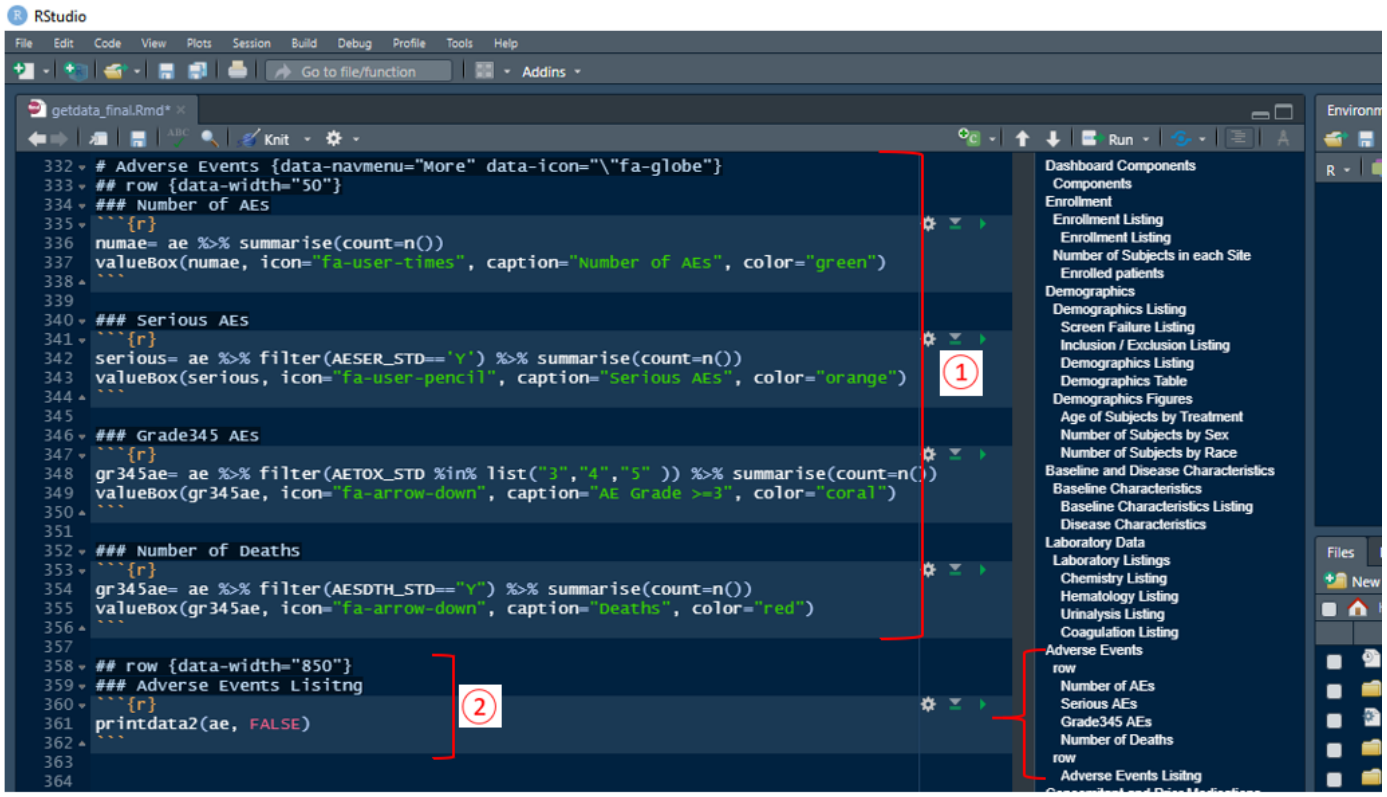


Figure 18. Code of Adverse Events Listings with Value boxes.

Figure 18 - Block 1 code: In Adverse Events page, ae summary counts are added as value boxes using "valueBox" function as shown in Figure 17 (Marked in Red 1).

Figure 18 - Block 2 code: Using Printdata function, Adverse Events Listing is printed as shown in Figure 17 (Marked in Red 2).

Another cool option in R Markdown: In YAML Header using option “source_code: embed”, we can preserve the R code in the output as shown in figure 19.

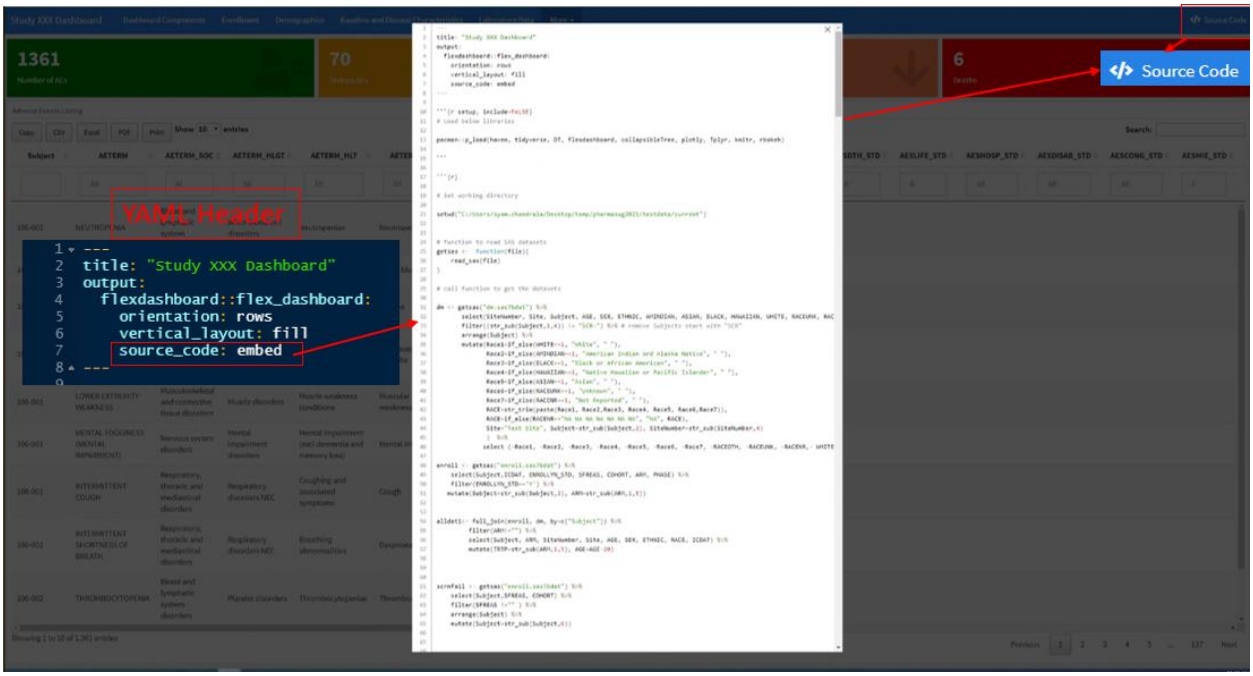


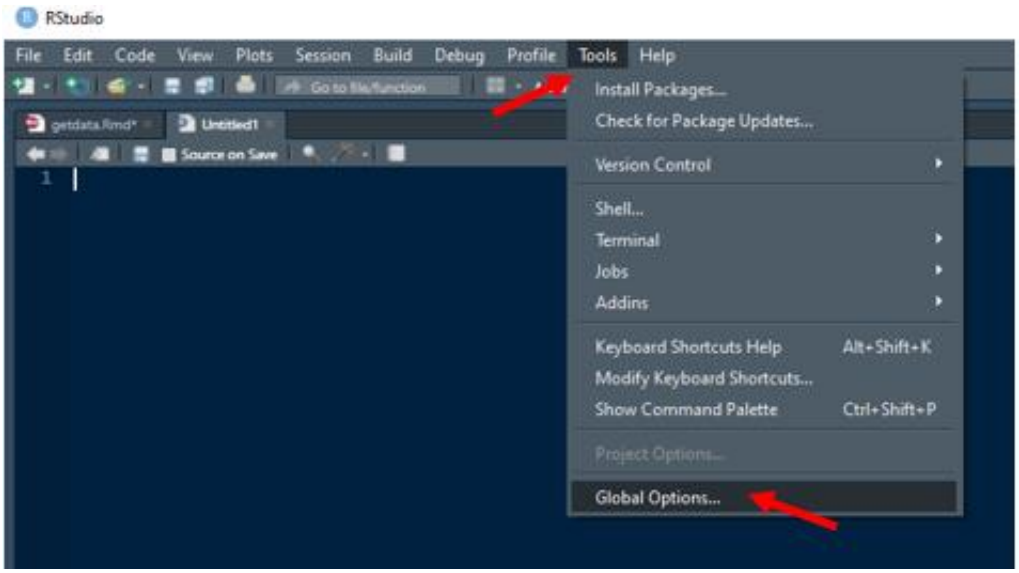
Figure 19. Embed source code in the output

RSTUDIO CONNECT

RStudio Connect® is a standalone publishing platform for the work created in R.

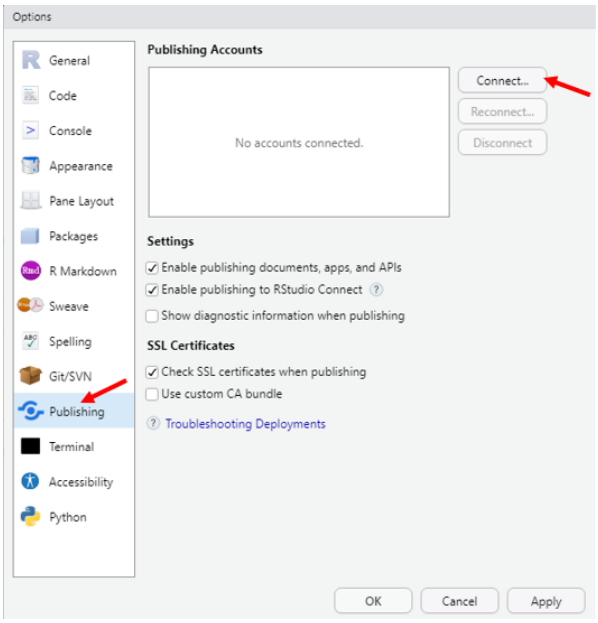
First we have to connect to RStudio Connect® account. Then we publish and share the dashboards, presentations, reports, interactive web applications and documents.

Step 1: Click Tools -> Global Options

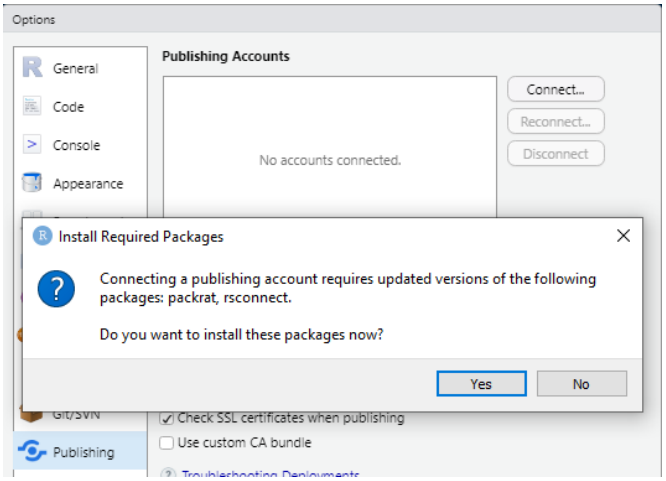


Step 2: A prompt window will open,

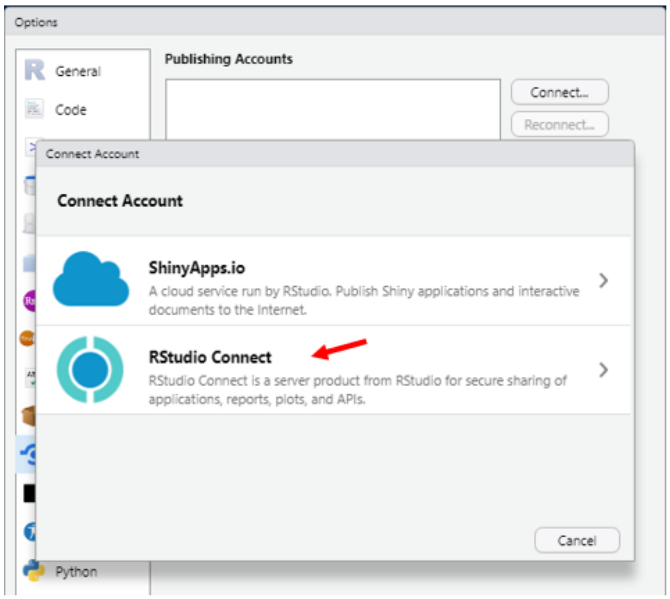
a. From the Left Pane, Select “Publishing” from left pane
b. Click Connect.



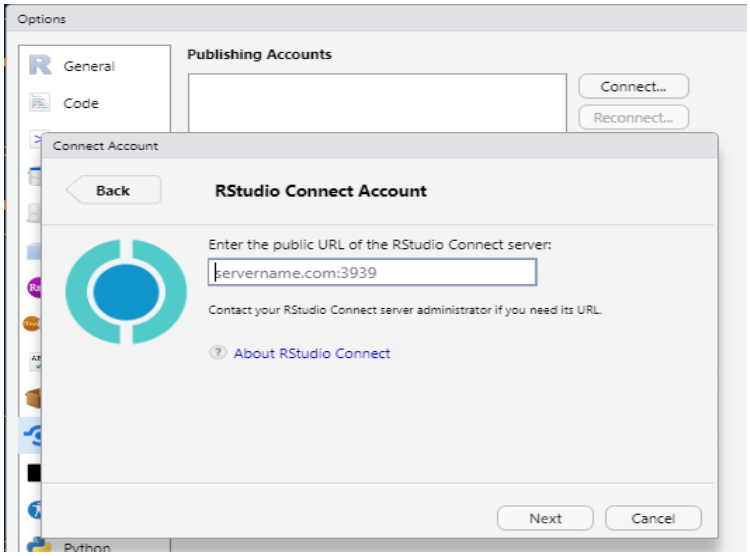
c. If you are publishing for the first time in Your RStudio, you will be prompted to install Packrat and rsconnect packages.



Step 3: Click on RStudio Connect®.



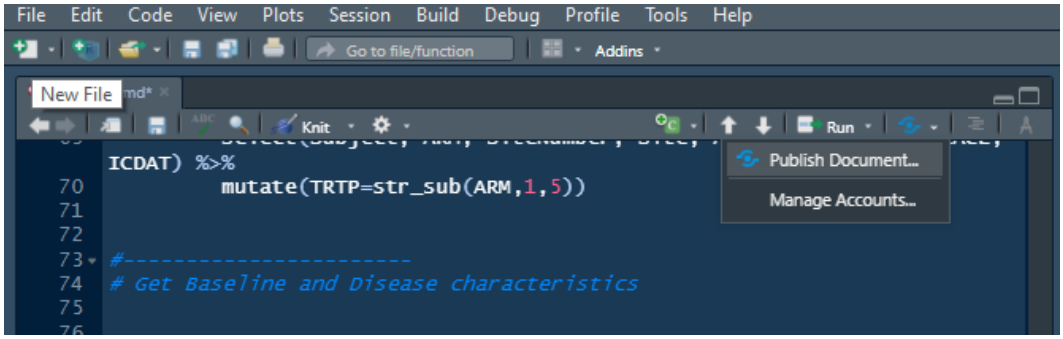
Step 4: Enter the URL of RStudio Connect® server and click Next.



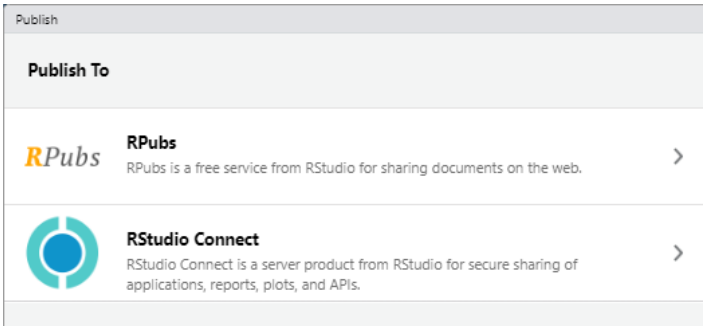
Once your credentials are accepted in RStudio Connect, you can now publish the dashboard as shown in the next session.

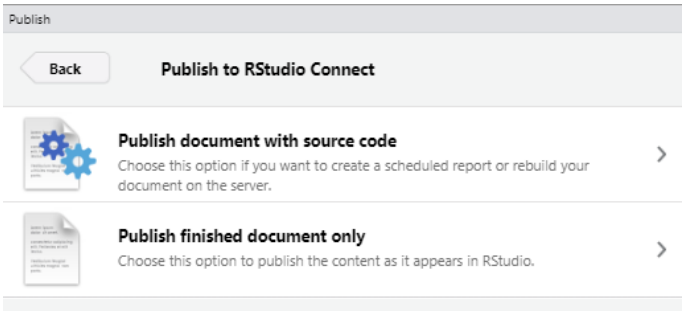
HOW TO PUBLISH THE DASHBOARD

1. Click on blue publishing icon in RStudio IDE and select publish document.

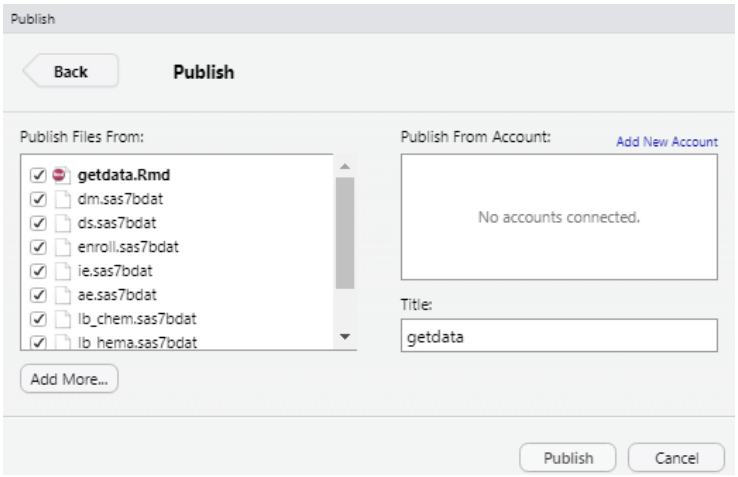


2. Select RStudio Connect®

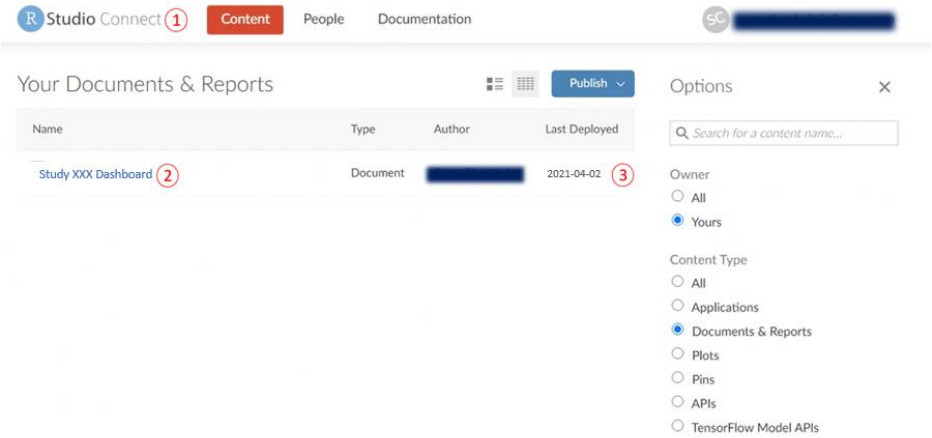




3. To publish the dashboard, we need to connect RStudio to the Connect account. Enter the credentials and click on publish.



4. Published dashboard will appear on RStudio Connect (Marked in red 1), Dashboard name (Marked in red 2) and date published as last deployed (marked in red 3) as shown below.



CONCLUSION

Primary objective of this paper is to share our knowledge about Flexdashboard and R Markdown components, combined with R packages.

We have demonstrated, how to build a dashboard using clinical data, and publish the dashboard to RStudio Connect® in step by step manner. With the knowledge about these packages and options, we found that creating an interactive clinical dashboard using RStudio is a cost effective method, saves lot of time and effort for statistical programmer during critical timelines. Besides, these high quality reports can be useful for cross-functional departments for data review and data cleaning purposes.

It is one time effort to create this dashboard and it can be automated and scheduled as per the requirement. In addition, we can share the code by embedding it in the dashboard and it can be easily replicated for other similar studies. This interactive dashboard can also be used to double check the results that are generated by SAS reports created for internal reviews.

There are many HTML widgets and packages that can generate more advanced visual representation of the data and also build interactive reports. For example, addition of crosstalk package to these dashboards help to link multiple HTML widgets with in an R Markdown page and generates cross-widget interactivity in the dashboard which are more efficient. Adding Shiny components to the flexdashboard would make it even more dynamic.

REFERENCES

1. R-Markdown (RStudio Inc.)
   https://rmarkdown.rstudio.com/
2. Flexdashboard for R
   https://rmarkdown.rstudio.com/flexdashboard/index.html
3. R Studio Webinars
   https://www.rstudio.com/resources/webinars/reproducibility-in-production/
4. DT: An R interface to the DataTables Library
   https://rstudio.github.io/DT/
5. Tidyverse- R packages for Data Science
   https://www.tidyverse.org/
6. Plotly R Open Source Graphing Library
   https://plotly.com/r/

RECOMMENDED READING

1. R Markdown: The Definitive Guide

   https://bookdown.org/yihui/rmarkdown/

2. R for Data Science

   https://r4ds.had.co.nz/index.html

ACKNOWLEDGMENTS

We would like to thank Irina Walsh, Director of Statistical programming at Allogene Therapeutics and Edgardo Gutierrez, R Consultant, Allogene Therapeutics for providing valuable inputs.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Name: Syam Chandrala
Enterprise: Allogene Therapeutics
Address: 210 E Grand Ave.,
South San Francisco, CA 94080
E-mail: Syam.chandrala@gmail.com

Name: Chaitanya Chowdagam
Enterprise: Ephicacy Consulting Group, Inc.
E-mail: chowdagam@gmail.com