Oct 2024

R Shiny Masterclass Series - Introduction

R Shiny UI, navigation and responsive layouts

EPI-interactive

Agenda

- **Session 1** | 30 September | Getting started with Posit Cloud and your first R Shiny app
- **Session 2** | 01 October | R Shiny core concepts and mobile ready layout
- **Session 3** | 03 October | R Shiny user interface components, reactivity and debugging
- **Session 4** | 07 October | Data sources and data processing in R Shiny
- **Session 5** | 08 October | Interactive charts with Plotly: chart types, customising hover boxes and chart styling
- **Session 6** | 10 October | Maps and spatial visualisation with Leaflet: adding map layers, annotations, pins, filters and legend
- **Session 7** | 14 October | Publishing R Shiny apps, design considerations and case study
- **Session 8** | 15 October | Case study, top 10 tips for data visualisation with R Shiny and wrap-up

Today

Recap: Session 1

Have pen and paper ready!

Goals:

- Practice with Shiny layout functions
- Know how to add static content to app
- Create custom layouts with the Shiny Bootstrap grid system

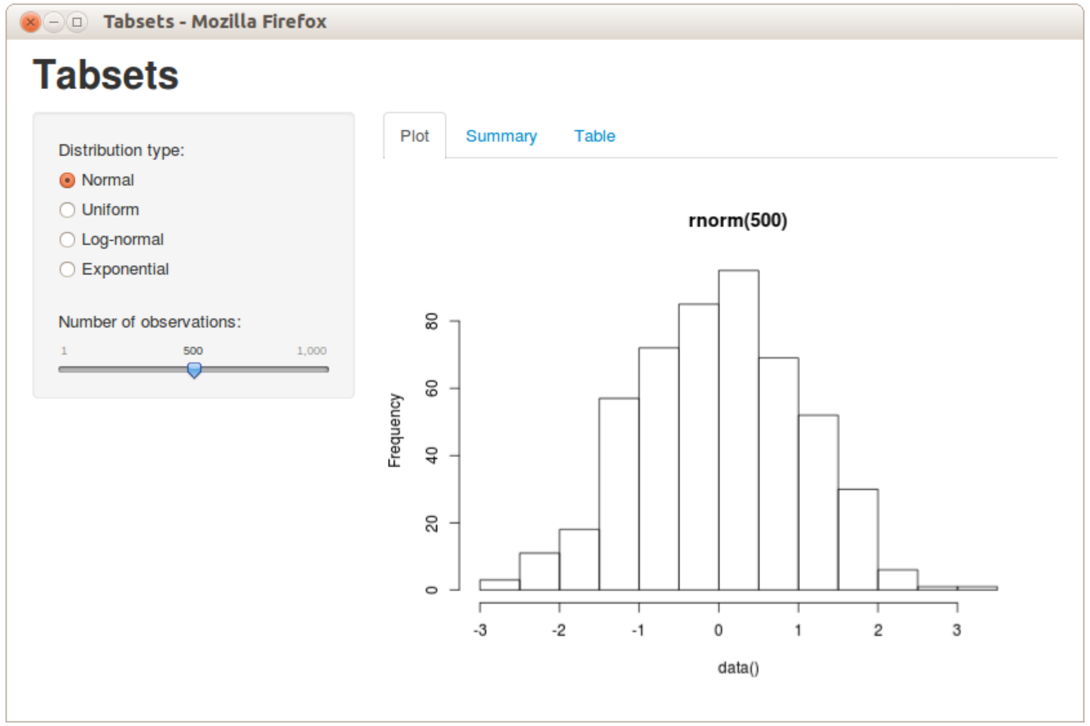
Shiny UI functions

Shiny UI functions

- Shiny gives us access to pre-made UI functions to simplify interface design
- Layout functions
  - **sidebarLayout**
  - splitLayout (horizontal) / verticalLayout (vertical)
  - flowLayout
- Navigation functions
  - **tabsetPanel**
  - **navlistPanel**
  - **navbarPage**
- We can freely combine these together!

Navigation: tabsetPanel

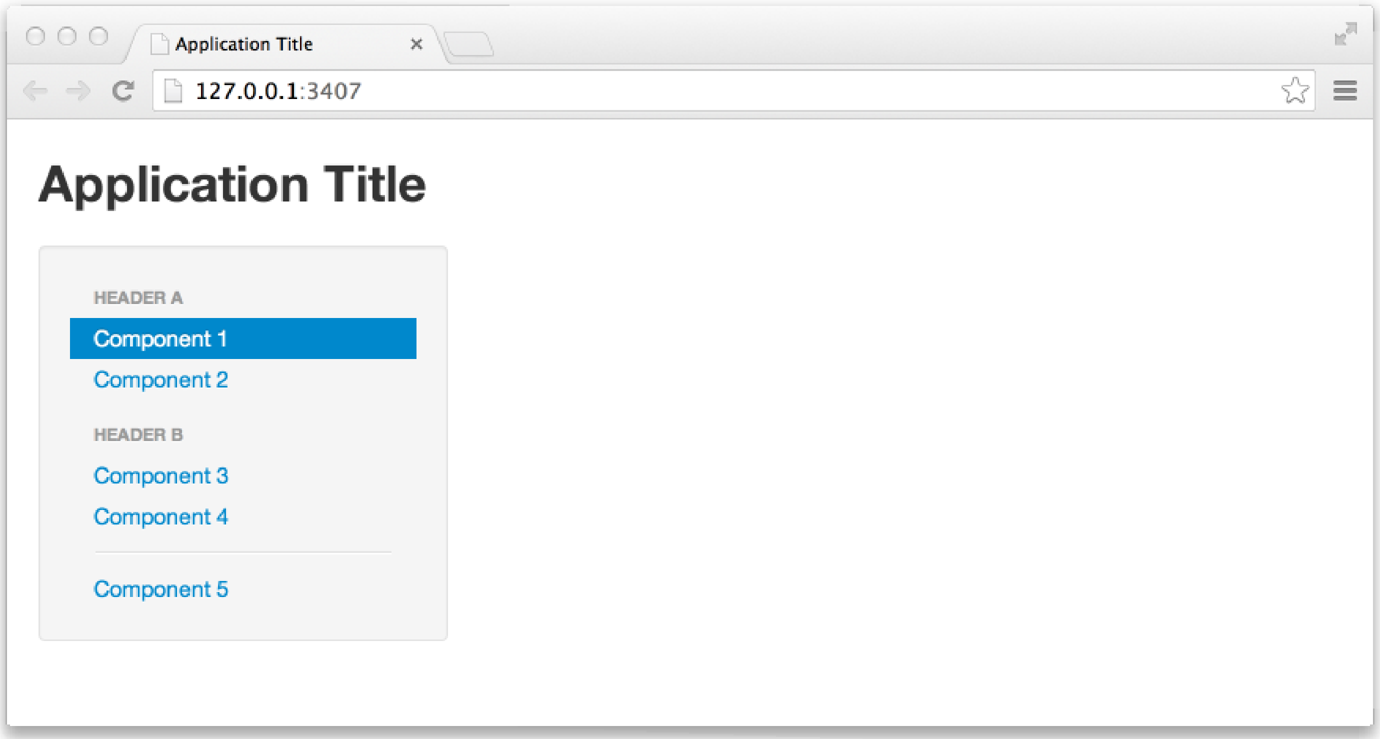
tabsetPanel(..., id = NULL, selected = NULL, type = c("tabs","pills"), header = NULL, footer = NULL)



https://shiny.rstudio.com/articles/layout-guide.html

Navigation: navlistPanel

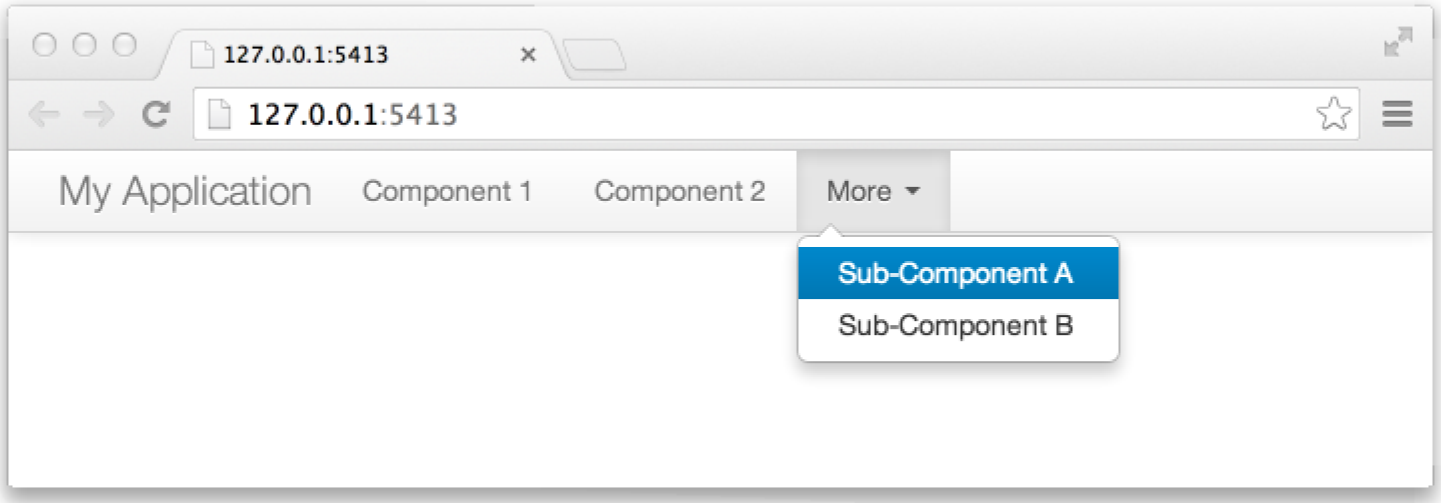
navlistPanel(..., id = NULL, selected = NULL, well = TRUE,
 fluid = TRUE, widths = c(4, 8))



https://shiny.rstudio.com/articles/layout-guide.html

Navigation: navbarPage

- navbarPage(title, ..., id)
- navbarMenu(title, ..., menuName = title)



https://shiny.rstudio.com/articles/layout-guide.html

Navigation: individual tabs

- tabPanel(title, ..., icon = NULL)
- This is where the actual content of the tab itself will go
- Used across all the navigation elements

Navigation: Exercise

- In Posit Cloud, open Session-2, then /stage1

- Add a tabSetPanel to the current app, within the existing mainPanel
- In one tab, add the “distplot” output
- In the other, add the “distPlot2” output

- We have provided “distPlot2”, in server.R, which is the same as “distPlot”, with different coloured bars

Shiny Layout

- Shiny abstracts away a lot of HTML
- E.g. a sidebarPanel(…) is actually
  <div class="col-sm-4"> <form class = “well”> … </form> </div>
- This is helpful for quick development, but it can limit options in future
- Use HTML(…) to wrap pure HTML code
- Use containers like div(), wellPanel(), tagList() to contain multiple elements together

- Note that you are using functions to generate HTML

Shiny HTML

- HTML tags can be generated using identically named functions

Check tags on:
https://shiny.rstudio.com/articles/html-tags.html

- E.g. p("paragraph") becomes <p>Paragraph</p>
- Many tags require the “tags$” format, e.g. tags$table()
- Some commonly use ones can go without, e.g. p()
  (see documentation above)

Adding new components: exercise

Add some new components (HTML tags) to the mainPanel in ui.R

Try out adding different tags in the UI file:

- tags$strong(), tags$i()
- tags$a(href=…)
- HTML() (can be used for direct HTML input as a character string)

Try out tags with children (tags within tags):

- tags$ul(tags$li())

Try adding an image

- tags$img(src=…)

Mobile ready layout & Bootstrap

Bootstrap

What is Bootstrap

- Originally created by a designer / developer at Twitter
- Maintained by an open-source team of contributors under the MIT licence
- Contains most widely used grid system on the web (and more)
- Premade styles and components
- “Mobile first” / responsive layout

How does it work

- The Bootstrap grid is based on **containers**, **rows** and **columns**
- A **container** holds a series of **rows**, and each **row** is divided into 12 units
- Columns can be sized between 1-12 units within a row

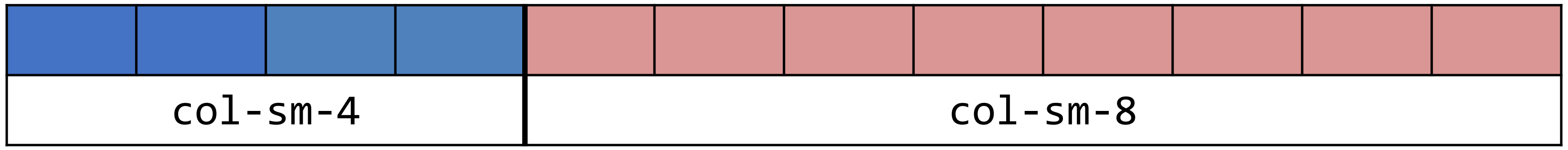
Bootstrap HTML – Breakpoints

Bootstrap target devices:

• Extra small devices, portrait phones (<576px)	“xs”
• Small devices, landscape phones (≥768px)	“sm”
• Medium devices, tablets (≥768px)	“md”
• Large devices, desktops (≥992px)	“lg”
• Extra large devices, desktops (≥1200px)	“xl”

Bootstrap Grid

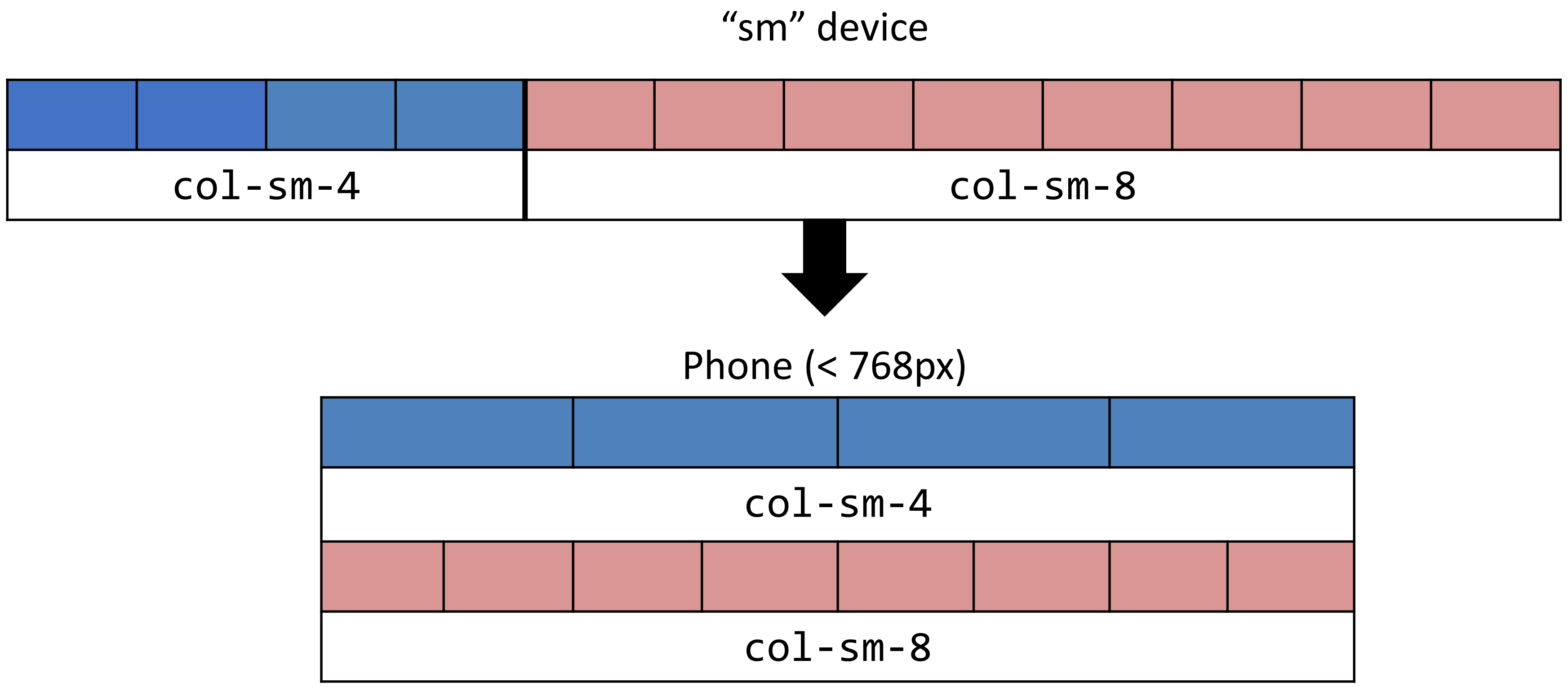
Fixed default sidebarLayout: 4 – 8 column layout



Check HTML layout with Chrome developer tools

Bootstrap Grid

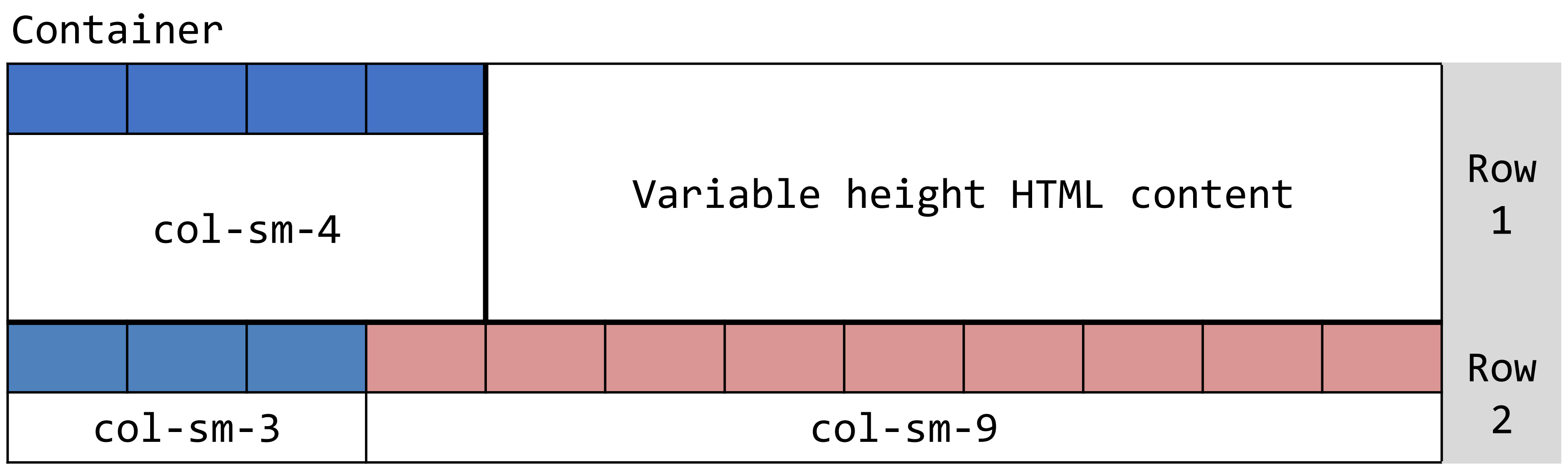
Fixed default sidebarLayout: 4 – 8 column layout



Try it out in your app....

Bootstrap Grid

- Columns have variable height
- Columns don’t need to be specified in full
  E.g. we don’t know how much content we'll have in the top right, but we want the next columns to appear below



EPI-interactive

Bootstrap Grid

- We can guarantee vertical ordering with **rows**

- Wrapper (parent element) can be **container-fluid** or **container:**
    - Both horizontally centre the content
    - Container-fluid always stretches to the edge (Shiny default)

      fluidPage(…)
    - Container applies a fixed width according to breakpoints:

      1200px, 992px, 768px…

      fixedPage(…)

Bootstrap HTML – “Div soup”

Now let’s guess the layout… Draw the grid for a large device

```
<div class="container-fluid">
          <div class="row">
                    <div class="col-sm-6">
                              Content
                    </div>
          </div>
          <div class="row">
                    <div class="col-sm-4">
                              Content2
                    </div>
                    <div class="col-sm-8">
                              Content3
                    </div>
          </div>
</div>
```

EPI-interactive

Bootstrap HTML – “Div soup”

```
<div class="container-fluid">
     <div class="row">
          <div class="col-sm-6 col-md-3">
               Content
          </div>
          <div class="col-sm-6 col-md-9">
               Content2
          </div>
     </div>
</div>
```

Bootstrap HTML – “Div soup”

```
<div class="container-fluid">
     <div class="row">
          <div class="col-sm-3"> Content1 </div>
          <div class="col-sm-3"> Content2 </div>
          <div class="col-sm-2"> Content3 </div>
          <div class="col-sm-2"> Content4 </div>
          <div class="col-sm-2"> Content5 </div>
     </div>
</div>
```

Bootstrap HTML – “Div soup”

```
<div class="container-fluid">
      <div class="row">
            <div class="col-sm-5">
                  Content
            </div>
            <div class="col-sm-6">
                  Content2
            </div>
            <div class="col-sm-4">
                  Content3
            </div>
      </div>
</div>
```

Bootstrap - Exercise

Now let’s convert the Geyser app in /stage2:

```
fluidRow(
    column(6, [add sidePanel elements here]
    ),
    column(6, [add mainPanel elements here]
    )
  )
```

sidebarLayout -> fluidRow
sidebarPanel -> column

mainPanel -> column

Check how the page behaves at different sizes.

EPI-interactive

Next time

- Shiny UI components
- Reactivity
- Debugging

Challenge (Using your stage 2 project):

Modify your application to have the following structure:

- At the top of the page: a fluidRow containing a title element and an image as a logo
- Beneath this, a fluidRow containing either a tabsetPanel, navlistPanel or navbarPage
  - Use your existing tabPanels inside of this navigation element
- Beneath this, a new row for the footer, with 3 evenly sized columns
  - In the first column, add your name. In the second, add a link to an email address. In the third, add a web link that opens in a new tab.

Share the link to your project on the **Session 2 forum**.