**2.基本用户界面**

**2.1简介**

现在，您怀里已经有了一个基本的应用程序，我们可以开始探索让Shiny运作的细节了。正如您在上一章中所看到的，Shiny鼓励将生成用户界面的代码(前端)与驱动应用程序行为的代码(后端)分开。

在本章中，我们将重点介绍前端，让您快速了解Shiny提供的HTML输入和输出。这使您能够掌握多种类型的数据并显示多种类型的R输出。您还没有很多方法将输入和输出缝合在一起，但我们将在第6章中回到这一点。

在这里，我将主要关注Shiny自身内置的输入和输出。然而，有一个丰富而充满活力的扩展包社区，如shinyWidgets、colorpicker和sorttable。您可以在以下位置找到其他软件包的全面、积极维护的列表：https://github.com/nanxstats/awesome-shiny-extensions，由南晓负责维护。

像往常一样，我们将从加载Shiny包开始：

library(shiny)

**2.2输入**

正如我们在上一章中看到的，您可以使用sliderInput()、selectInput()、textInput()和NumericiInput()等函数将输入控件插入到UI定义中。现在，我们将讨论构成所有输入函数基础的通用结构，并快速概览Shiny中内置的输入控件。

**2.2.1共同结构**

所有输入函数都有相同的第一个参数：inputId。这是用于连接前端和后端的标识符：如果您的UI具有ID为“name”的输入，则服务器函数将使用input$name访问它。

inputId有两个约束：

它必须是一个只包含字母、数字和下划线的简单字符串(不允许使用空格、破折号、句点或其他特殊字符！)。命名它就像在R中命名变量一样。

它必须是唯一的。如果它不是唯一的，您将无法在服务器函数中引用此控件！

大多数输入函数都有第二个参数，称为label。这用于为控件创建人类可读的标签。Shiny没有对该字符串设置任何限制，但您需要仔细考虑，以确保您的应用程序可供人类使用！第三个参数通常是值，在可能的情况下，可以设置默认值。其余参数一般是控件独有的。

创建输入时，我建议按位置提供inputId和label参数，并按名称提供所有其他参数：

sliderInput("min", "Limit (minimum)", value = 50, min = 0, max = 100)

以下各节描述了内置在Shiny中的输入，根据它们创建的控件类型松散地分组。目标是让您快速了解您的选项，而不是详尽地描述所有的参数。我将在下面展示每个控件的最重要参数，但您需要阅读文档才能获得全部细节。

**2.2.2自由文本**

使用以下控件收集少量文本，textInput()输入少量文本、passwordInput()[[1]](#footnote-1)输入口令和textAreaInput()输入文本段落。

ui <- fluidPage(

textInput("name", "What's your name?"),

passwordInput("password", "What's your password?"),

textAreaInput("story", "Tell me about yourself", rows = 3)

)



如果您想确保文本具有某些属性，可以使用validate()，我们将在第8章中介绍。

**2.2.3数字输入**

要收集数值，请使用NumericiInput()创建受约束的文本框，或使用sliderInput()来创建滑块。如果为sliderInput()的默认值提供一个长度为2的数字向量，则会得到一个两端为“范围”的滑块。

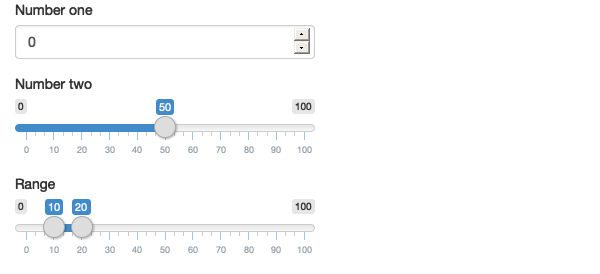
ui <- fluidPage(

numericInput("num", "Number one", value = 0, min = 0, max = 100),

sliderInput("num2", "Number two", value = 50, min = 0, max = 100),

sliderInput("rng", "Range", value = c(10, 20), min = 0, max = 100)

)



通常，我建议仅在小的取值范围内使用滑块，或者在精确值不太重要的情况下使用滑块。试图在一个小滑块上精确地选择一个数字是一个令人沮丧的练习！

滑块是可定制的，有许多方法可以调整其外观。Rstudio 帮助中输入?sliderInputh以及访问https://shiny.rstudio.com/articles/sliders.html以查看更多细节。

**2.2.4日期**

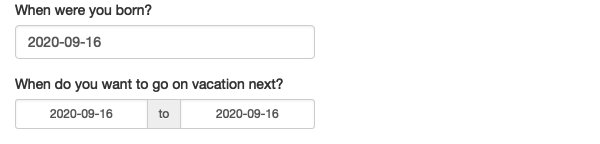
使用dateInput()收集一天，或使用dateRangeInput()采集两天的范围。它们提供了一个方便的日历选择器，datesdisabled和daysofweekdisabled等附加参数允许您限制有效输入集。

ui <- fluidPage(

dateInput("dob", "When were you born?"),

dateRangeInput("holiday", "When do you want to go on vacation next?")

)



日期格式、语言和一周开始的日期默认为美国标准。如果您正在创建一个拥有国际受众的应用程序，请设置format,、language和weekstart参数，以便用户能够自然地选择日期。

**2.2.5有限的选择**

有两种不同的方法允许用户从预先指定的选项集中进行选择：selectInput()和radioButtons()。

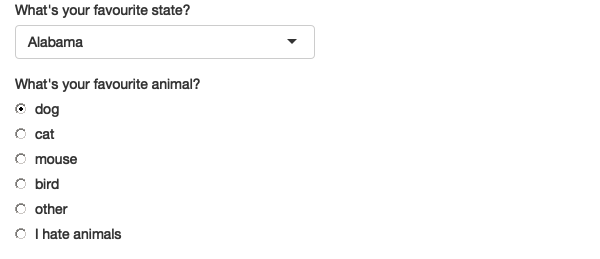
animals <- c("dog", "cat", "mouse", "bird", "other", "I hate animals")

ui <- fluidPage(

selectInput("state", "What's your favourite state?", state.name),

radioButtons("animal", "What's your favourite animal?", animals)

)



单选按钮有两个很好的功能：它们显示所有可能的选项，使其适合于短列表；通过choiceNames/choiceValues参数，它们可以显示纯文本以外的选项。choiceNames确定向用户显示的内容；choiceValues确定服务器函数中返回的内容。

ui <- fluidPage(

radioButtons("rb", "Choose one:",

choiceNames = list(

icon("angry"),

icon("smile"),

icon("sad-tear")

),

choiceValues = list("angry", "happy", "sad")

)

)

#> This Font Awesome icon ('angry') does not exist:

#> \* if providing a custom `html\_dependency` these `name` checks can

#> be deactivated with `verify\_fa = FALSE`

#> This Font Awesome icon ('smile') does not exist:

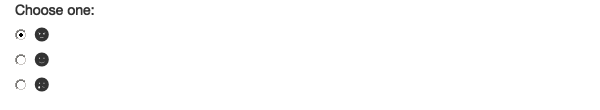
#> \* if providing a custom `html\_dependency` these `name` checks can

#> be deactivated with `verify\_fa = FALSE`

#> This Font Awesome icon ('sad-tear') does not exist:

#> \* if providing a custom `html\_dependency` these `name` checks can

#> be deactivated with `verify\_fa = FALSE`



使用selectInput()创建的下拉列表占用固定的空间，无论选项的数量如何，使它们更适合较长的选项。还可以设置multiple=TRUE以允许用户选择多个元素。

ui <- fluidPage(

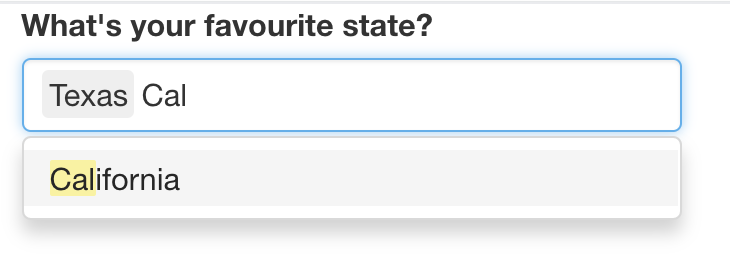
selectInput(

"state", "What's your favourite state?", state.name,

multiple = TRUE

)

)



如果您有一组非常大的可能选项，您可能希望使用“服务器端”selectInput()，这样，完整的可能选项集就不会嵌入到UI中(这会导致加载速度变慢)，而是根据需要由服务器发送。有关此高级主题的更多信息，请访问：https://shiny.rstudio.com/articles/selectize.html#server-side-selectize。

没有办法用单选按钮选择多个值，但有一个概念上类似的替代方法：checkboxGroupInput()。

ui <- fluidPage(

checkboxGroupInput("animal", "What animals do you like?", animals)

)



如果您想为单个是/否问题设置一个复选框，请使用checkboxInput()：

ui <- fluidPage(

checkboxInput("cleanup", "Clean up?", value = TRUE),

checkboxInput("shutdown", "Shutdown?")

)



**2.2.6文件上传**

允许用户使用fileInput()上传文件：

ui <- fluidPage(

fileInput("upload", NULL)

)



fileInput()需要在服务器端进行特殊处理，第9章对此进行了详细讨论。

**2.2.7动作按钮**

让用户使用actionButton()或actionLink()执行操作：

ui <- fluidPage(

actionButton("click", "Click me!"),

actionButton("drink", "Drink me!", icon = icon("cocktail"))

)

#> This Font Awesome icon ('cocktail') does not exist:

#> \* if providing a custom `html\_dependency` these `name` checks can

#> be deactivated with `verify\_fa = FALSE`



动作链接和按钮最自然地与服务器函数中的observeEvent()或eventReactive()配对。您尚未了解这些重要的函数，但我们将在第3.5节中再次讨论它们。

您可以通过使用类参数"btn-primary"、"btn-success"、"btn-info"、"btn-warning"或 "btn-danger"中的一个，自定义按钮的外观。您还可以使用参数"btn-lg"、"btn-sm"、"btn-xs"更改按钮的大小。最后，您可以使用参数“btn-block”使按钮占据其嵌入的元素的整个宽度。

ui <- fluidPage(

fluidRow(

actionButton("click", "Click me!", class = "btn-danger"),

actionButton("drink", "Drink me!", class = "btn-lg btn-success")

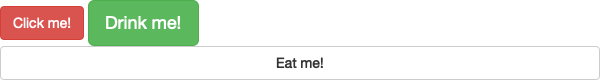
),

fluidRow(

actionButton("eat", "Eat me!", class = "btn-block")

)

)



class参数通过设置底层HTML的class属性来工作，这会影响元素的样式。要查看其他选项，您可以阅读Bootstrap的文档，它是Shiny使用的CSS设计系统：[http://bootstrapdocs.com/v3.3.6/docs/css/#buttons](http://bootstrapdocs.com/v3.3.6/docs/css/" \l "buttons)。

**2.2.8练习**

1、当空间非常有限时，使用文本输入区域内的占位符标记文本框非常有用。如何调用textInput()生成下面的UI？



2、仔细阅读sliderInput()的文档，了解如何创建如下所示的日期滑块。



3、创建一个滑块输入，以选择介于0和100之间的值，其中滑块上每个可选值之间的间隔为5。然后，向滑块输入部件添加动画，以便当用户按下“播放”时，滑块输入自动在取值范围内滑动。

4、如果您在selectInput()中有一个中等长度的列表，则创建将列表拆分为多个部分的子标题非常有用。阅读文档以了解如何操作。(提示：底层HTML标签为＜optgroup＞。)

**2.3输出**

UI中的输出创建占位符，稍后由服务器函数填充。与输入一样，输出采用唯一的ID作为其第一个参数[[2]](#footnote-2)：如果您的UI定义创建了一个ID为“plot”的输出，您将使用output$plot在服务器函数中访问它。

前端的每个输出函数都与后端的一个渲染函数配对。有三种主要的输出类型，对应于一份报告中通常包含的三种内容：文本、表格和绘图。以下各节向您展示了前端输出函数的基础知识，以及后端相应的渲染函数。

**2.3.1文本**

使用textOutput()输出常规文本，使用VerbatimTexOutput()输出格式严格的以及控制台的文本输出。

ui <- fluidPage(

textOutput("text"),

verbatimTextOutput("code")

)

server <- function(input, output, session) {

output$text <- renderText({

"Hello friend!"

})

output$code <- renderPrint({

summary(1:10)

})

}



请注意，只有在需要运行多行代码的情况下，渲染函数中才需要｛｝。正如您将很快了解到的，您应该在渲染函数中进行尽可能少的计算，这意味着您可以经常忽略它们。如果写得更紧凑，上面的服务器函数会是这样的：

server <- function(input, output, session) {

output$text <- renderText("Hello friend!")

output$code <- renderPrint(summary(1:10))

}

请注意，以下两个渲染函数的行为略有不同：

1、renderText()将结果组合成单个字符串，通常与textOutput()配对

2、renderPrint()打印结果，就像您在R控制台中一样，通常与verbatimTextOutput()配对。

我们可以通过一个玩具应用程序看到它们的不同之处：

ui <- fluidPage(

textOutput("text"),

verbatimTextOutput("print")

)

server <- function(input, output, session) {

output$text <- renderText("hello!")

output$print <- renderPrint("hello!")

}



这相当于R控制台中cat()和print()之间的差异。

**2.3.2表格**

把data frame 显示为表格有两个选项：

1、tableOutput()和renderTable()呈现静态数据表格，一次显示所有数据。

2、dataTableOutput()和renderDataTable()呈现一个动态表格，显示固定数量的行以及更改哪些行可见的控件。

tableOutput()最适用于小型固定的汇总表(例如模型系数)；如果要让用户浏览完整的data frame，则dataTableOutput()最合适。如果您想更好地控制dataTableOutput()的输出，我强烈推荐Greg Lin的reactable包。

ui <- fluidPage(

tableOutput("static"),

dataTableOutput("dynamic")

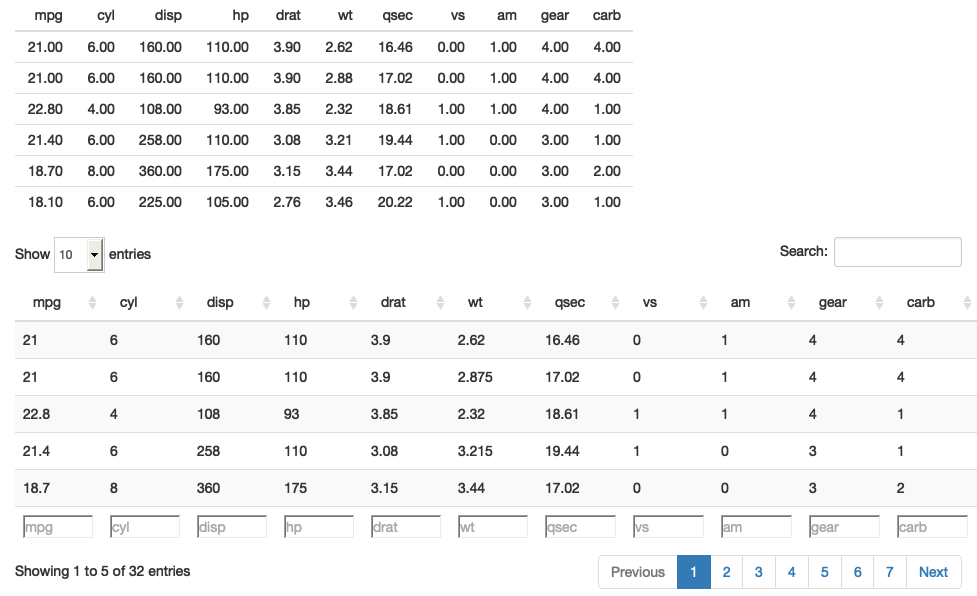
)

server <- function(input, output, session) {

output$static <- renderTable(head(mtcars))

output$dynamic <- renderDataTable(mtcars, options = list(pageLength = 5))

}



**2.3.3绘图**

可以使用plotOutput()和renderPlot()显示任何类型的R图形(base、ggplot2或其他)：

ui <- fluidPage(

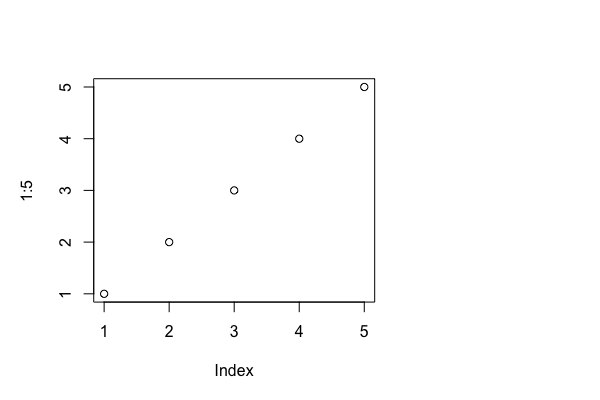
plotOutput("plot", width = "400px")

)

server <- function(input, output, session) {

output$plot <- renderPlot(plot(1:5), res = 96)

}



默认情况下，plotOutput()将占据其容器的整个宽度(稍后将详细介绍)，并且为400像素高。您可以使用height和width参数替代这些默认值。我们建议始终将参数res设置为96，因为这样可以使您的Shiny绘图尽可能接近您在RStudio中所看到的。

绘图是特殊的，因为它们是也可以作为输入的输出。plotOutput()有许多参数，如click、dblclick和hover。如果您给它们传递一个字符串，如click=“plot\_click”，它们将创建一个反应式输入(input$plot\_ click)，您可以使用该输入处理用户在绘图上的交互，例如单击绘图。我们将在第7章中回到Shiny的交互式绘图。

**2.3.4下载**

您可以使用downloadButton()或downloadLink()让用户下载文件。这些都需要在服务器函数中使用新技术，因此我们将在第9章中回到这一点。

**2.3.5练习**

1、以下每个渲染函数应与textOutput()和verbatimTextOutput()中的哪一个配对？

A、renderPrint(summary(mtcars))

B、renderText("Good morning!")

C、renderPrint(t.test(1:5, 2:6))

D、renderText(str(lm(mpg ~ wt, data = mtcars)))

2、从第2.3.3节重新创建Shiny应用程序，这次将高度设置为300px，宽度设置为700px。设置绘图的“alt”文本，以便视力受损的用户可以判断它是由五个随机数组成的散点图。

3、更新下面renderDataTable()调用中的选项，以便显示数据，但所有其他控件都被抑制(即删除搜索、排序和筛选命令)。你大概要在Rstudio帮助中输入？renderDataTable并查看以下链接中的选项：<https://datatables.net/reference/option/>。

ui <- fluidPage(

dataTableOutput("table")

)

server <- function(input, output, session) {

output$table <- renderDataTable(mtcars, options = list(pageLength = 5))

}

4、或者，阅读reactable的文档，并将上面的应用程序转换为使用它。

**2.4小结**

本章向您介绍了构成Shiny应用程序前端的主要输入和输出函数。这是一个巨大的信息转储，所以不要指望在一次阅读后就能记住所有内容。相反，当您寻找特定组件时，请回到本章：您可以快速扫描图形，然后找到所需的代码。

在下一章中，我们将继续讨论一个Shiny应用程序的后端：使您的用户界面栩栩如生的R代码。

1. “passwordInput()”所做的就是隐藏用户正在键入的内容，这样，背后的人就看不清了。

   确保任何密码都不会意外暴露，这取决于您，因此我们不建议您使用密码，除非您接受过安全编程方面的培训。 [↑](#footnote-ref-1)
2. 注意，输入（`inputId')和输出（`outputId')的参数名称不同（参数值相同，参见这些输入输出函数的说明)。

   我不使用第一个参数的名称（直接用值)，因为它非常重要，我希望您记住它的作用，而不需要额外的提示。 [↑](#footnote-ref-2)