Contents

Chapter 1

1.1 添加 UI 控件

- fluidPage() 是一个布局函数，用于设置页面的基本视觉结构。
- selectInput()是一个输入控件，允许用户通过提供值与应用程序交互。
- verbatimTextOutput() 和 tableOutput() 是输出控件，告诉 Shiny将渲染输出放在哪里。verbatimTextOutput()显示代码并 tableOutput()显示表格。

1.2

您可以通过包装一段代码并将 `reactive({...})` 其分配给变量来创建反应式表达式，并且可以通过像函数一样调用它来使用反应式表达式。但是，虽然看起来您正在调用函数，但响应式表达式有一个重要的区别：它仅在第一次调用时运行，然后缓存其结果，直到需要更新为止。

Chapter 2

基础 UI

2.1 输出

请注意，有两个渲染函数的行为略有不同：

- renderText() 将结果组合成一个字符串，并且通常与 textOutput()
- renderPrint() 打印结果，就像您在 R 控制台中一样，并且通常与 verbatimTextOutput()

2.1.1 表格

有两种用于在表中显示数据框的选项：

- tableOutput() 与 renderTable() 渲染一个静态数据表，一次性显示所有数据。
- dataTableOutput() 与 renderDataTable() 呈现一个动态表，显示固定数量的行以及用于更改哪些行可见的控件。

tableOutput() 对于小型、固定的 summary（例如模型系数）最有用；如果您想向用户公开完整的数据框，则 dataTableOutput() 最合适。

2.1.2 绘图

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

2.1.3 下载

您可以让用户使用 downloadButton() 或 downloadLink() 来下载文件。

Chapter 3

响应式基础

3.1 server 函数

3.1.1 input

参数 input 是一个类似列表的对象，其中包含从浏览器发送的所有输入数据，根据输入 ID 命名。与普通的列表不同，input 对象是只读的。如果你尝试在服务函数内的修改输入，你将收到错误。发生此错误是因为 input 反映了浏览器中发生的情况，而浏览器是 Shiny 的“单一事实来源”。如果你可以修改 R 中的值，则可能会导致不一致，即输入滑块在浏览器中表示一件事，而 input$count 在 R 中表示不同的内容。这将使编程变得具有挑战性！稍后，在 **??** 中，你将学习如何使用诸如 updateNumericInput() 修改浏览器中的值之类的功能，然后 input$count 进行相应的更新。

关于 input 更重要的一件事是：它对谁可以阅读是有选择性的。**要读取 input，必须处于由 renderText() 或 reactive() 函数创建的反应式上下文中。**

3.1.2 输出

output 与 input 非常相似：它也是一个根据输出 ID 命名的类似列表的对象。主要区别在于使用它来发送输出而不是接收输入。你总是要把 output 对象与 render 函数结合使用。

渲染函数做了两件事：

- 它设置了一个特殊的反应上下文，可以自动跟踪输出使用的输入。
- 它将 R 代码的输出转换为适合在网页上显示的 HTML。

与 input 一样，output 对如何使用它很挑剔。

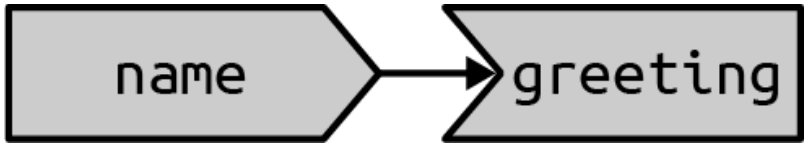

Figure 3.1: 反应图显示了输入和输出的连接方式

3.2 响应式编程

Shiny 的重要思想：你不需要告诉输出何时更新，因为 Shiny 会自动为你计算出来。

3.2.1 命令式编程 imperative programming 与声明式 declarative programming 编程

命令和 recipes 之间的区别是两种重要编程风格之间的主要区别之一：

- 在命令式编程中，你发出特定命令，它会立即执行。这是你在分析脚本中习惯的编程风格：命令 R 加载数据、转换数据、可视化数据，并将结果保存到磁盘。
- 在声明式编程中，你表达更高级别的目标或描述重要的约束，并依靠其他人来决定如何和/或何时将其转化为行动。这是你在 Shiny 中使用的编程风格。

命令式代码是 assertive；声明式代码是 passive-aggressive。

3.2.2 反应图

反应图是了解应用程序工作原理的强大工具。随着你的应用程序变得越来越复杂，制作反应图的快速高级草图通常很有用，以提醒你所有部分如何组合在一起。在本书中，我们将向你展示反应图，以帮助你理解示例的工作原理，稍后在 **??** 中，你将学习如何使用 reactlog 来为你绘制图表。

3.2.3 响应表达式

你将在反应图中看到一个更重要的组件：反应表达式。反应式表达式接受输入并产生输出，因此它们具有结合输入和输出特征的形状。希望这些形状能帮助你记住组件如何组合在一起。

Figure 3.2: 输入和表达式是反应式生产者；表达式和输出是反应式消费者

3.2.4 执行顺序

重要的是要理解代码运行的顺序完全由反应图决定。这与大多数 R 代码不同，大多数 R 代码的执行顺序由行的顺序决定。

3.3 响应表达式

反应式表达式具有输入和输出的风格：

- 与输入一样，您可以在输出中使用反应式表达式的结果。
- 与输出一样，反应式表达式依赖于输入并自动知道何时需要更新。

这种二元性意味着我们需要一些新的词汇：我将使用生产者（producer）来指代反应式输入和表达式，使用消费者（consumer）来指代反应式表达式和输出。