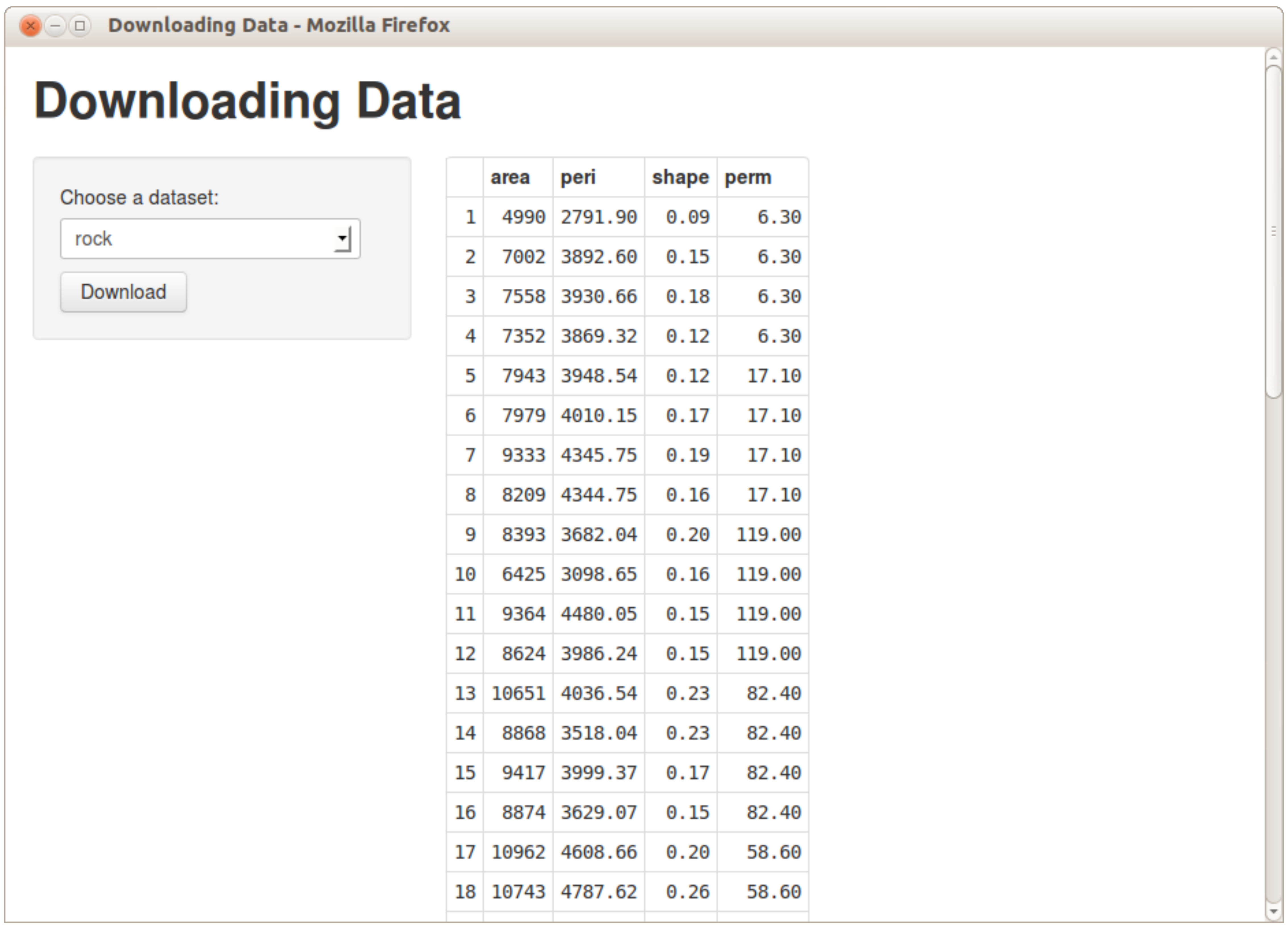
This tutorial is deprecated. Learn more about Shiny at our new location, shiny.rstudio.com.



Downloading Data

Choose a dataset:

rock

Download

	area	peri	shape	perm
1	4990	2791.90	0.09	6.30
2	7002	3892.60	0.15	6.30
3	7558	3930.66	0.18	6.30
4	7352	3869.32	0.12	6.30
5	7943	3948.54	0.12	17.10
6	7979	4010.15	0.17	17.10
7	9333	4345.75	0.19	17.10
8	8209	4344.75	0.16	17.10
9	8393	3682.04	0.20	119.00
10	6425	3098.65	0.16	119.00
11	9364	4480.05	0.15	119.00
12	8624	3986.24	0.15	119.00
13	10651	4036.54	0.23	82.40
14	8868	3518.04	0.23	82.40
15	9417	3999.37	0.17	82.40
16	8874	3629.07	0.15	82.40
17	10962	4608.66	0.20	58.60
18	10743	4787.62	0.26	58.60

The examples so far have demonstrated outputs that appear directly in the page, such as plots, tables, and text boxes. Shiny also has the ability to offer file downloads that are calculated on the fly, which makes it easy to build data exporting features.

To run the example below, type:

```
> library(shiny)
> runExample("10_download")
```

You define a download using the `downloadHandler` function on the server side, and either `downloadButton` or `downloadLink` in the UI:

ui.R

```
shinyUI(pageWithSidebar(
  headerPanel('Download Example'),
  sidebarPanel(
    selectInput("dataset", "Choose a dataset:",
                choices = c("rock", "pressure", "cars")),
    downloadButton('downloadData', 'Download')
  ),
  mainPanel(
    tableOutput('table')
  )
))
```

server.R

```
shinyServer(function(input, output) {
  datasetInput <- reactive({
    switch(input$dataset,
           "rock" = rock,
           "pressure" = pressure,
           "cars" = cars)
  })

  output$table <- renderTable({
    datasetInput()
  })

  output$downloadData <- downloadHandler(
    filename = function() { paste(input$dataset, '.csv', sep='') },
    content = function(file) {
      write.csv(datasetInput(), file)
    }
  )
})
```

As you can see, `downloadHandler` takes a `filename` argument, which tells the web browser what filename to default to when saving. This argument can either be a simple string, or it can be a function that returns a string (as is the case here).

The `content` argument must be a function that takes a single argument, the file name of a non-existent temp file. The `content` function is responsible for writing the contents of the file download into that temp file.

Both the `filename` and `content` arguments can use reactive values and expressions (although in the case of `filename`, be sure your argument is an actual function; `filename = paste(input$dataset, '.csv')` is not going to work the way you want it to, since it is evaluated only once, when the download handler is being defined).

Generally, those are the only two arguments you'll need. There is an optional `contentType` argument; if it is `NA` or `NULL`, Shiny will attempt to guess the appropriate value based on the filename. Provide your own content type string (e.g. `"text/plain"`) if you want to override this behavior.

← Previous   Next →