**Uploading and Downloading Data with Shiny**

* Shiny allows for users uploading their own files into the environment.
* This feature does not work with Internet Explorer 9 and earlier (not even with Shiny Server).
* By default, Shiny limits file uploads to 5MB per file.
* However - You can modify this limit by using the shiny.maxRequestSize option. For example, adding options(shiny.maxRequestSize=30\*1024^2) to the top of server.R would increase the limit to 30MB.
* File upload controls are created by using the fileInput function in your ui.R file.
* You access the uploaded data similarly to other types of input: by referring to input$*inputId*.
* The fileInput function takes a multiple parameter that can be set to TRUE to allow the user to select multiple files, and an accept parameter can be used to give the user clues as to what kind of files the application expects.

|  |
| --- |
| **ui.R**  shinyUI(pageWithSidebar(  headerPanel("CSV Viewer"),  sidebarPanel(  **fileInput('file1', 'Choose CSV File',**  **accept=c('text/csv', 'text/comma-separated-values,text/plain', '.csv')),**  **tags$hr(),**  **checkboxInput('header', 'Header', TRUE),**  radioButtons('sep', 'Separator',  c(Comma=',', Semicolon=';',  Tab='\t'), 'Comma'),  radioButtons('quote', 'Quote', c(None='',  'Double Quote'='"', 'Single Quote'="'"),  'Double Quote')  ),  mainPanel(  **#Contents**  tableOutput('contents')  )  )) |

|  |
| --- |
| shinyServer(function(input, output) {  output$**contents** <- renderTable({    # input$file1 will be NULL initially. After the user selects and uploads a  # file, it will be a data frame with 'name', 'size', 'type', and 'datapath'  # columns. The 'datapath' column will contain the local filenames where the  # data can be found.  **# Some checks**  inFile <- input$file1  if (is.null(inFile))  return(NULL)  **# Read in a CSV !!**  read.csv(inFile$datapath, header=input$header,  sep=input$sep, quote=input$quote)  })  }) |

**Downloads**

Shiny also has the ability to offer file downloads of data that are calculated by the shiny app, which makes it easy to build data exporting features.

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")),  **#HERE**  downloadButton('downloadData', 'Download')  ),  mainPanel(  tableOutput('table')  )  )) |

|  |
| --- |
| shinyServer(function(input, output) {  datasetInput <- reactive({  switch(input$dataset,  "rock" = rock,  "pressure" = pressure,  "cars" = cars)  })    output$table <- renderTable({  datasetInput()  })    output$downloadData <- downloadHandler( **#HERE**  **#Two Arguments – filename and content**  filename = function() { paste(input$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 in the example above).

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.