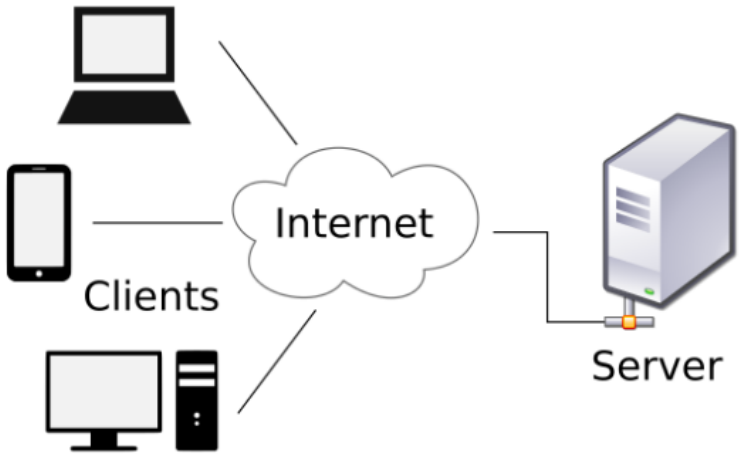
dviz-practice

Shiny review for PDF (`example` code)

TODO Answer conceptual questions

- What is the client/server software model?



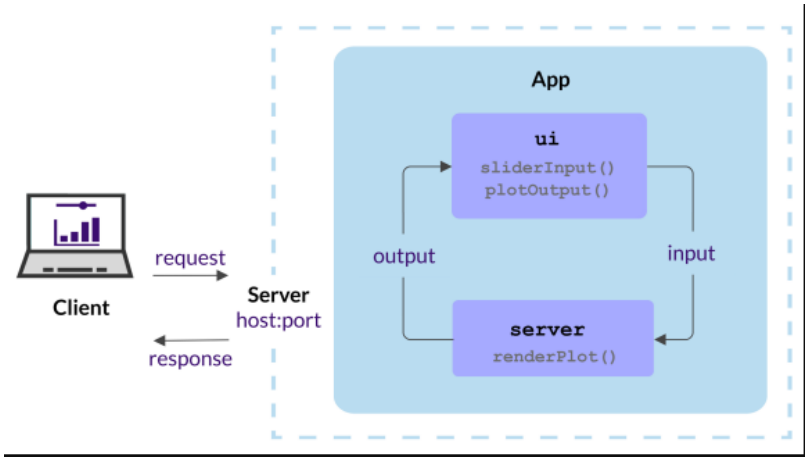
- Why should you learn R Shiny? What's the scenario?

- What exactly do you need to create and run a Shiny app?
    - R
    - Shiny R package
    - Web browser (including client/server)
- What is the order of functions to create a Shiny app?

```
library(shiny)
ui <- fluidPage("hello,world!")
server <- function(input,output,session){
# body of session with input/output commands
}
shinyApp(ui,server)
```

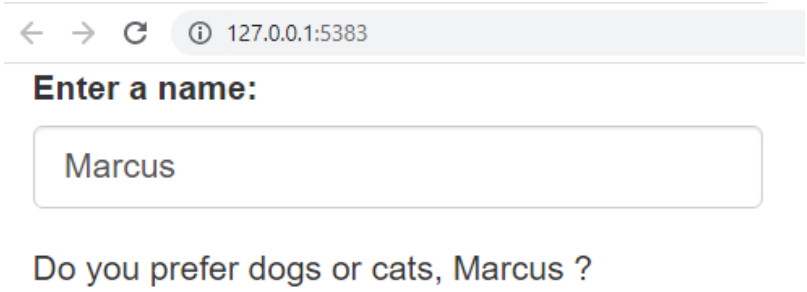
    1. Load package
    2. Define user interface `ui`
    3. Define server function `server`
    4. Call `shinyApp(ui,server)`
- What happens exactly when you call `shinyApp`?
    1. R executes the `shinyApp` function
    2. The `server` builds a web page with `ui <- fluidPage`
    3. The `server` waits for input and generates output

4. I/O is received from/sent to the URL `host:port`



- Where exactly do you see the app when you do this on your PC?

  On your localhost `127.0.0.1` port `5383` (random port)



  Here is a tutorial on how to run Shiny locally (Solymos, 2021).

- What else do you need to build an app for the Internet?
  - A web server application (e.g. Apache - e.g. with XAMPP)
  - A port that's open to the Internet
  - A router that assigns an address to your server

TODO Create a 'hello world!' app

1. Load the Shiny package
2. Create the user interface `ui` as `fluidPage("Hello, world!")`
3. Create the `server` as `function(input, output, session){}`
4. To run, call `shinyApp` on `ui` and `server`
5. To stop the app, enter `C-g` in Emacs

```
library(shiny)
ui <- fluidPage("Hello, world!")
```

```
server <- function(input,output,session){
  ## open server connection and pass string to server
}
shinyApp(ui,server)
```

TODO Break the app

1. After starting the app, open the R console: you can see that the server is listening.
2. Close the browser window that belongs to this process.
3. If you now try to stop the process in the Org-file with `C-g`, the process will not abort: the server is still listening.
4. Change to the R console and shut it down there with `C-c C-c`

TODO Save the app and run it on the shell

1. Move the cursor on the last code block and enter `C-c '`
2. In the new buffer, enter `C-x h M-x write-region` to write the code to a file `app.R` (or copy the code there).
3. Return to the code block with `C-c C-k`
4. Open a terminal (CMD line shell)
5. Navigate to the directory where `app.R` resides
6. Run app.R with the command `Rscript app.R`
7. Open the URL after `Listening on` in a browser
8. In the shell, stop the process with `C-c C-c`

TODO Create an input/output app

1. Load the Shiny package
2. Create the `ui <- ~fluidPage(~textInput(), ~textOutput())`
3. `textInput` should accept your `'name'` and ask `"Enter a name:"`
4. Format `textOutput` should return `'hello'`
5. Create the `server` with `input` and `output` arguments
6. Inside the `server`, assign `renderText()` to `output$hello`
7. Inside `renderText`, enter `{paste()}`
8. Inside `paste()`, enter `"Hello,"` and `input$name`
9. Run the app with `shinyApp`

10. Stop the app in Emacs with `C-g`

```
library(shiny)
ui <- fluidPage(
  textInput('name', "Enter a name:"),
  textOutput('hello')
)
server <- function(input, output) {
  output$hello <- renderText({
    paste("Hello,", input$name)
  })
}
shinyApp(ui,server)
```

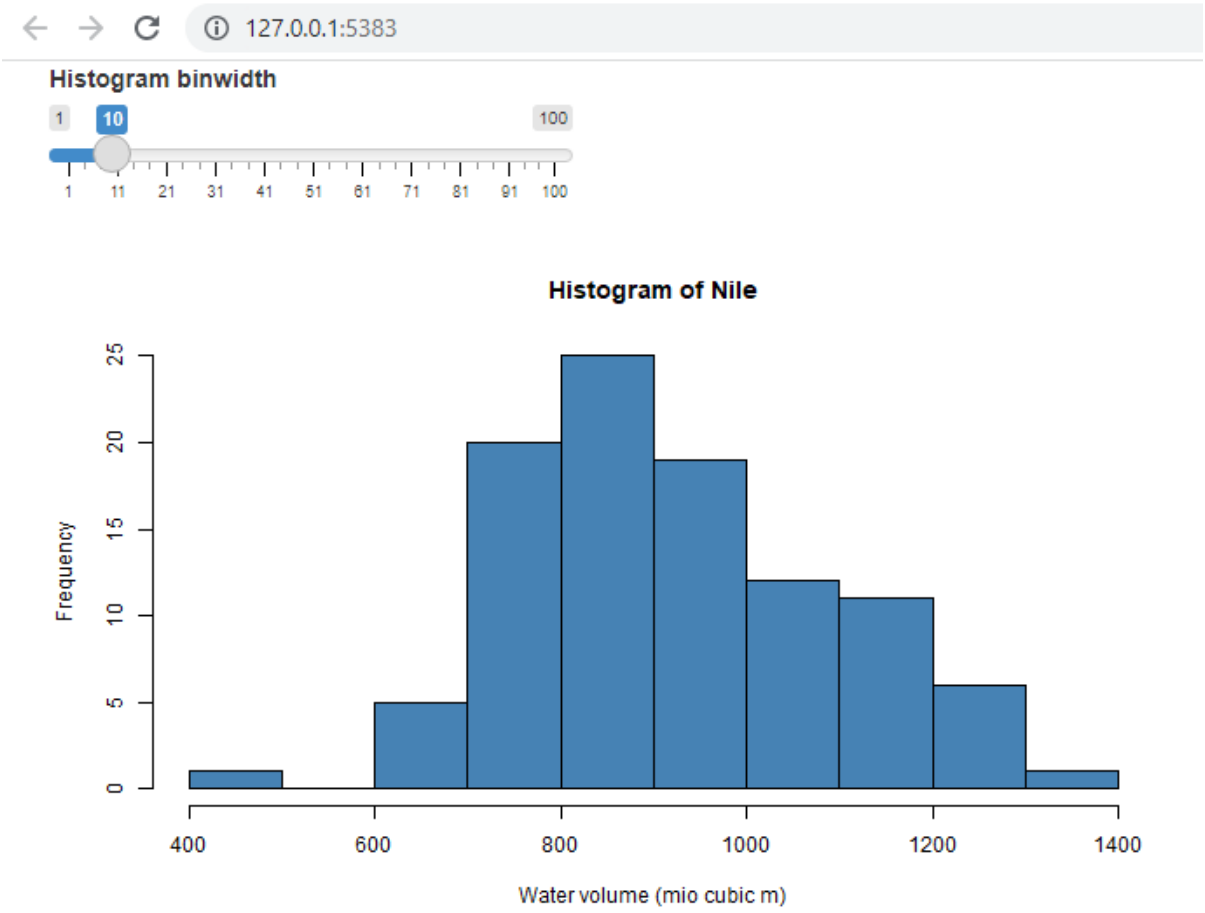
TODO Build an app with a histogram

1. Set up the UI and the server
2. Inside `fluidPage()`, create a `mainPanel()`
3. Inside `mainPanel()`, create a `sliderInput()` and `plotOutput`
4. For `sliderInput()`, define:
    - the `input` variable `'binwidth'`
    - the text `"Histogram binwidth"`
    - the parameters `min=1, max=100, value=10`
5. Inside `plotOutput`, define the `output` variable `'nile'`
6. In the body of the `server` function, assign `renderPlot({p})` to `output$nile`
7. As plot `p`, use `hist(Nile, breaks=input$binwidth)`
8. Customize the inside of `hist()` as you like
9. Call `shinyApp` for `ui` and `server` to open app in browser
10. Close process with `C-g` in the Org-file, and `C-c C-c` in `*R*`

```
library(shiny)

ui <- fluidPage(
  mainPanel(
    sliderInput('binwidth',
                "Histogram binwidth",
                min = 1,
                max = 100,
                value = 10),
    plotOutput("nile")))

server <- function(input, output) {
  output$nile <- renderPlot({
    hist(Nile,
         breaks = input$binwidth,
         main="Histogram of Nile",
         col="steelblue",
         xlab="Water volume (mio cubic m)")
  })
}
shinyApp(ui = ui, server = server)
```



Created: 2022-11-10 Thu 23:00