

Introducing Shiny

Jonas Schöley

jschoeley@health.sdu.dk



Max-Planck Odense Center on the
Biodemography of Aging



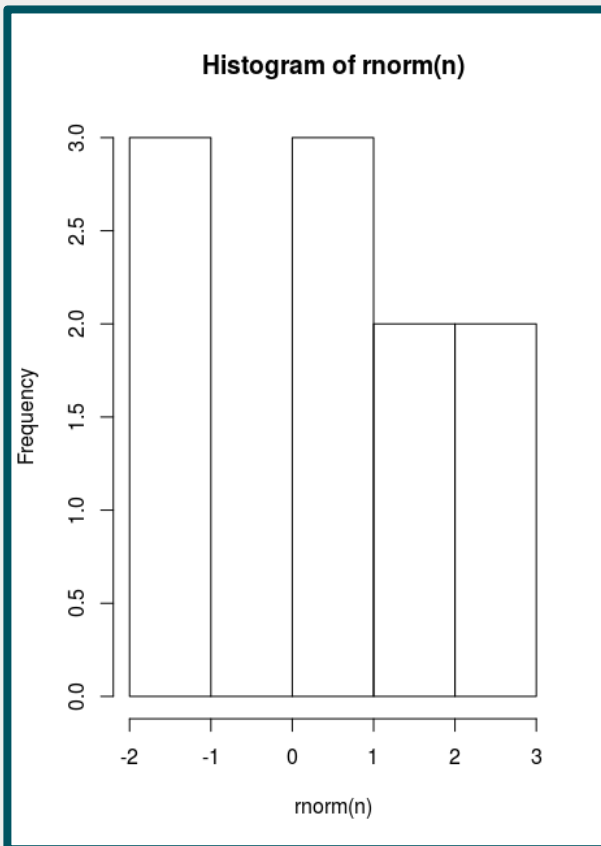
Department of Public Health
University of Southern Denmark

What's “Shiny”?

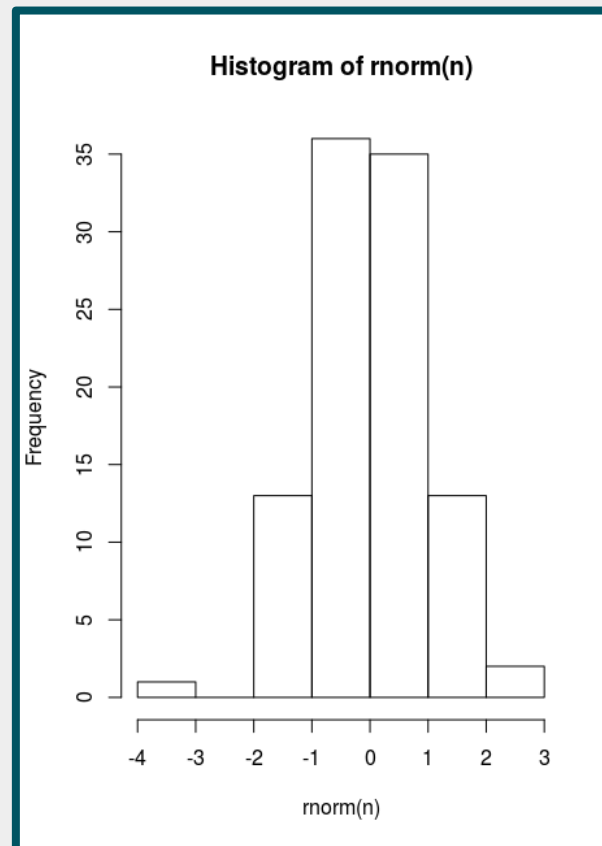
- 1 A graphical user-interface for your R program
- 2 Your R program as a web-page
- 3 A web-application framework for R

A graphical user-interface for your R program

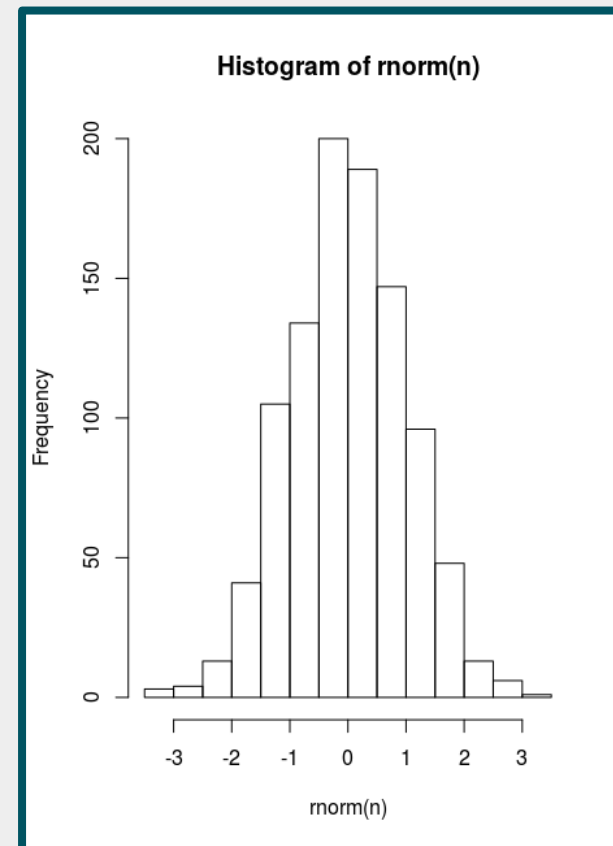
$n = 10$
`hist(rnorm(n))`



$n = 100$
`hist(rnorm(n))`

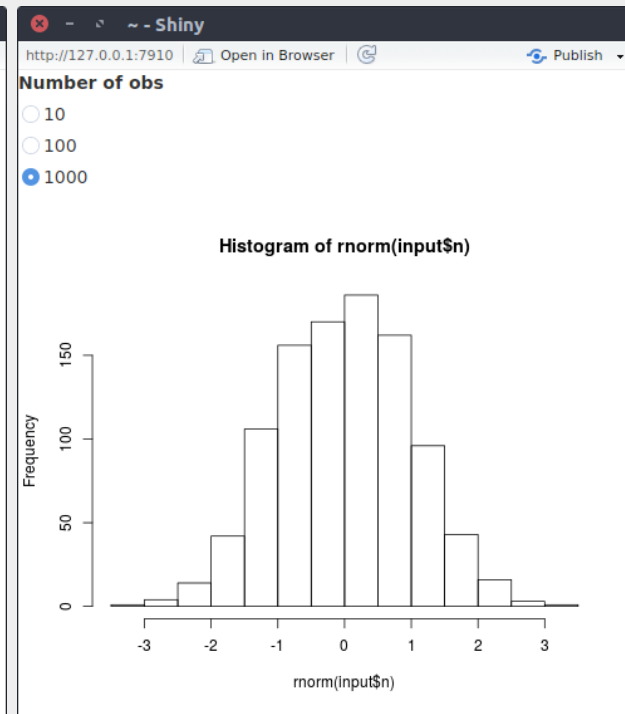
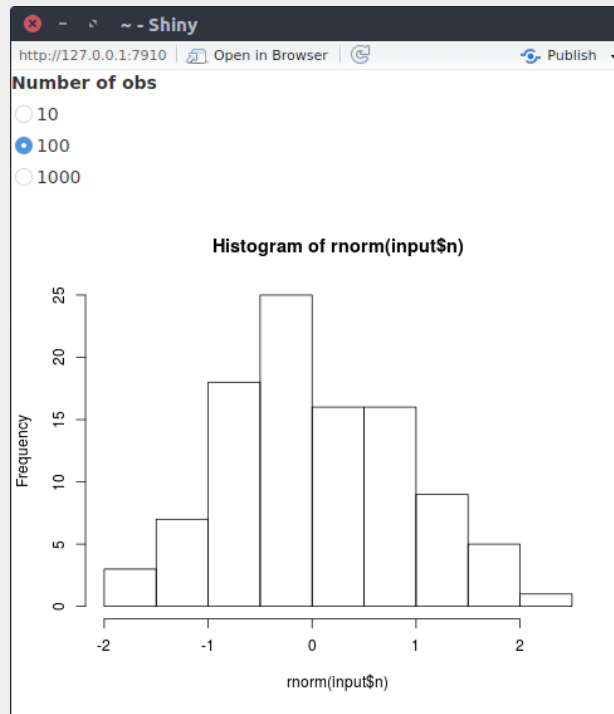
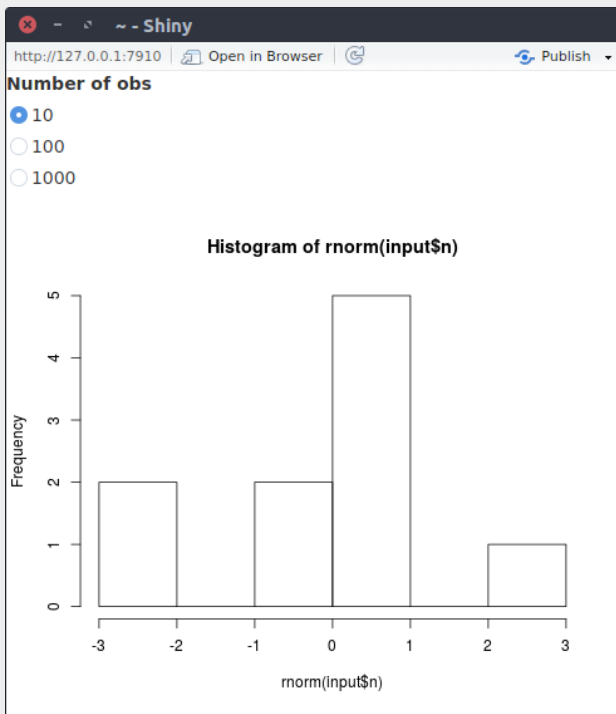


$n = 1000$
`hist(rnorm(n))`



A graphical user-interface for your R program

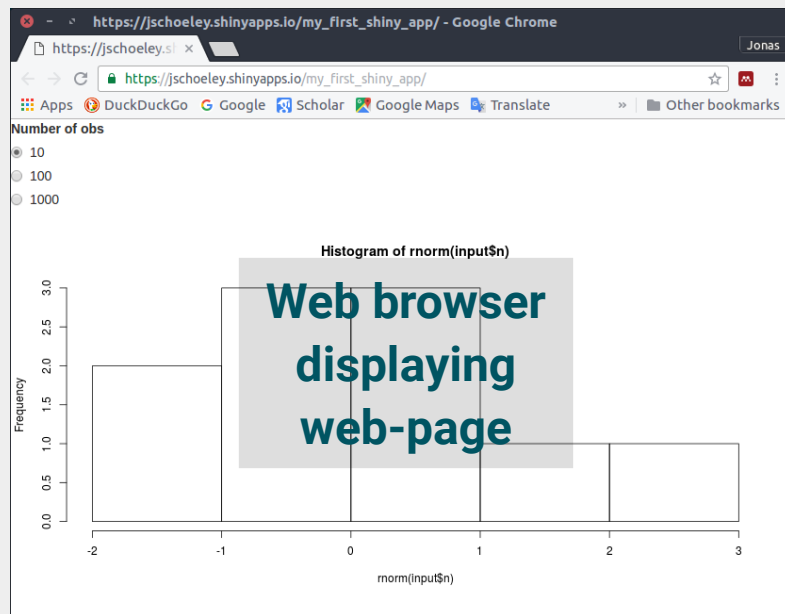
```
library(shiny)
ui <- bootstrapPage(
  radioButtons(inputId = "n", label = "Number of obs", choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)
server <- function(input, output) {output$plot <- renderPlot(hist(rnorm(input$n)))}
shinyApp(ui = ui, server = server)
```



Your R program as a web-page



Shiny: A web-application framework for R



- listens for updates from server
- renders web-page

sends user input
to server

sends UI updates
to user



- listens for user input
- performs R calculations
- builds web-page

- 1 Building a user interface (UI)
- 2 Connecting R code with the UI

Programing Shiny

```
library(shiny)
```

```
ui <- bootstrapPage(  
  radioButtons(inputId = "n",  
               label = "Number of obs",  
               choices = list(10, 100, 1000)),  
  plotOutput(outputId = "plot")  
)
```

```
server <- function(input, output) {  
  output$plot <- renderPlot(  
    hist(rnorm(input$n))  
  )  
}
```

```
shinyApp(ui = ui, server = server)
```


Programing Shiny: Building a User Interface

1 Building a user interface (UI)

```
library(shiny)
```

```
ui <- bootstrapPage(  
  radioButtons(inputId = "n",  
               label = "Number of obs",  
               choices = list(10, 100, 1000)),  
  plotOutput(outputId = "plot")  
)
```

```
server <- function(input, output) {  
  output$plot <- renderPlot(  
    hist(rnorm(input$n))  
  )  
}
```

```
shinyApp(ui = ui, server = server)
```

Programing Shiny: Building a User Interface

Create an empty web-page

learn about more sophisticated layouts at
shiny.rstudio.com/articles/layout-guide.html

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```

Programing Shiny: Building a User Interface

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
    label = "Number of obs",
    choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```



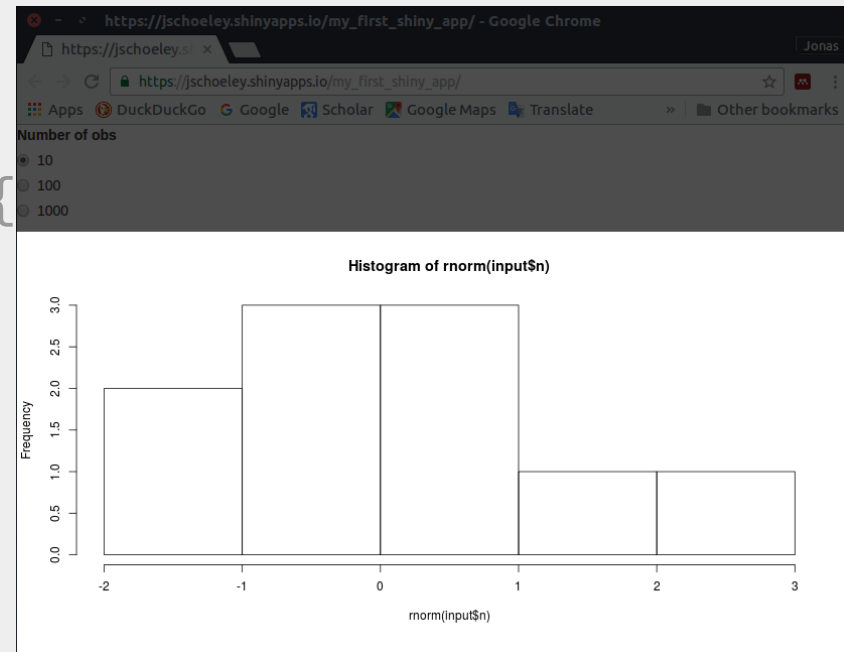
Programing Shiny: Building a User Interface

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```



Programing Shiny: Building a User Interface

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```

Don't forget the comma

the UI specification is one large nested function call with different UI elements as parameters

Programing Shiny: Connecting R to the UI

```
library(shiny)
```

```
ui <- bootstrapPage(  
  radioButtons(inputId = "n",  
               label = "Number of obs",  
               choices = list(10, 100, 1000)),  
  plotOutput(outputId = "plot")  
)
```

```
server <- function(input, output) {  
  output$plot <- renderPlot(  
    hist(rnorm(input$n))  
  )  
}
```

2 Connecting R code with the UI

```
shinyApp(ui = ui, server = server)
```

Programing Shiny: Connecting R to the UI

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```

R communicates with the UI using 2 lists
the **input** list contains the values of all input
elements of the UI, the **output** list contains
the objects which R sends to the UI

Programing Shiny: Connecting R to the UI

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```


Programing Shiny: Connecting R to the UI

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```

Programing Shiny: Connecting R to the UI

The “actual” R code

R calculates new output once new input arrives

```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

shinyApp(ui = ui, server = server)
```

Programing Shiny: Wrapping Things Up

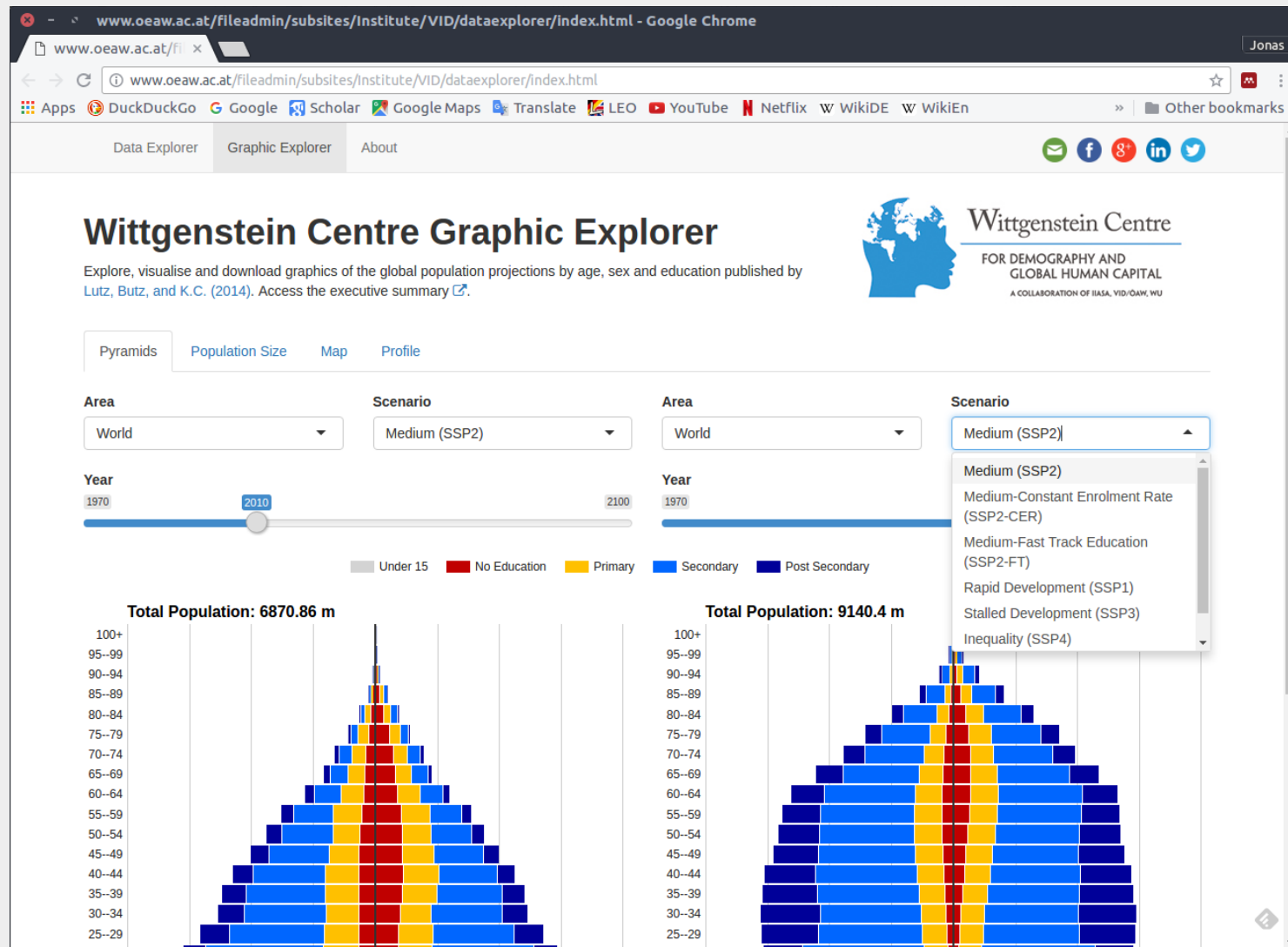
```
library(shiny)

ui <- bootstrapPage(
  radioButtons(inputId = "n",
               label = "Number of obs",
               choices = list(10, 100, 1000)),
  plotOutput(outputId = "plot")
)

server <- function(input, output) {
  output$plot <- renderPlot(
    hist(rnorm(input$n))
  )
}

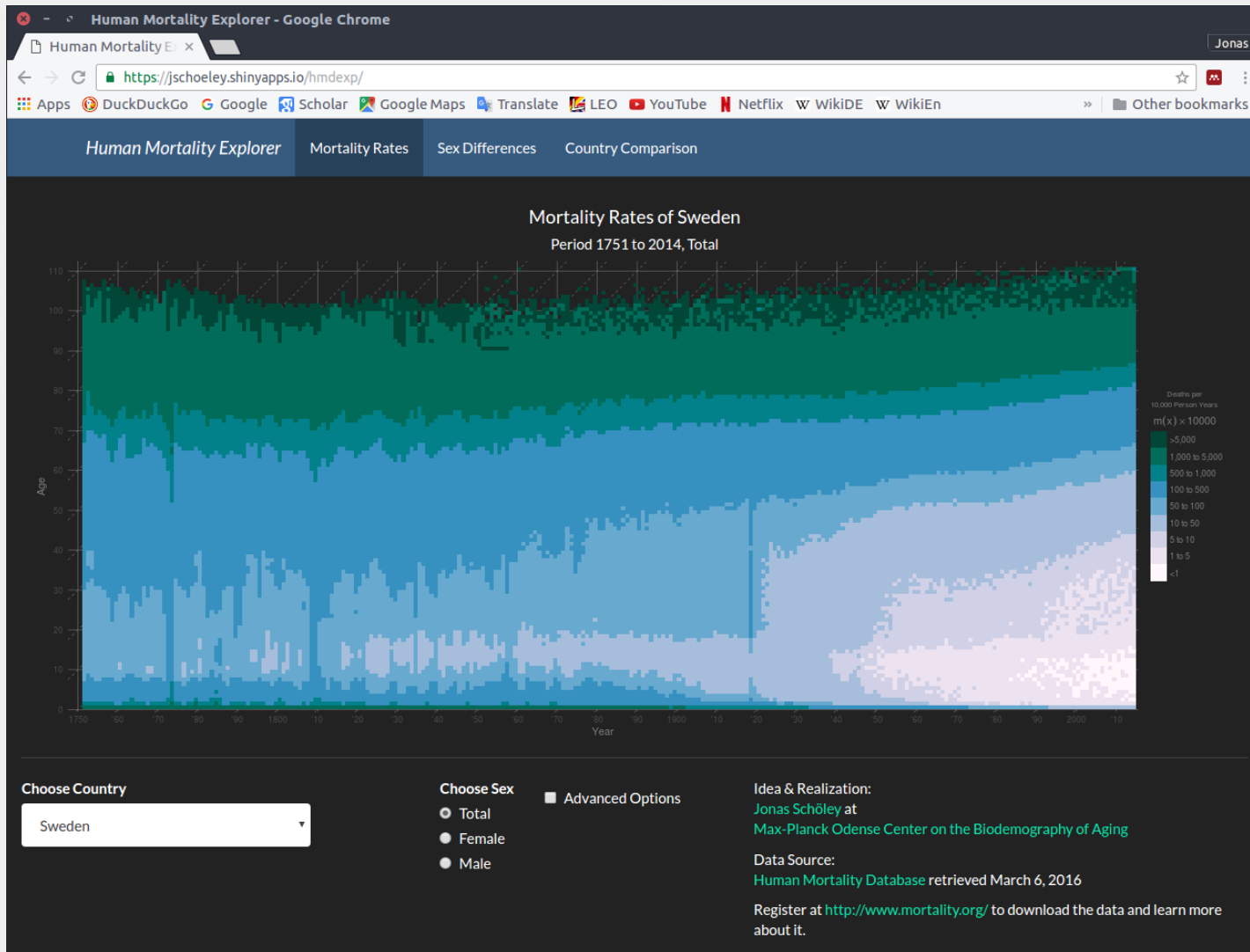
shinyApp(ui = ui, server = server)
```

Who Uses It?



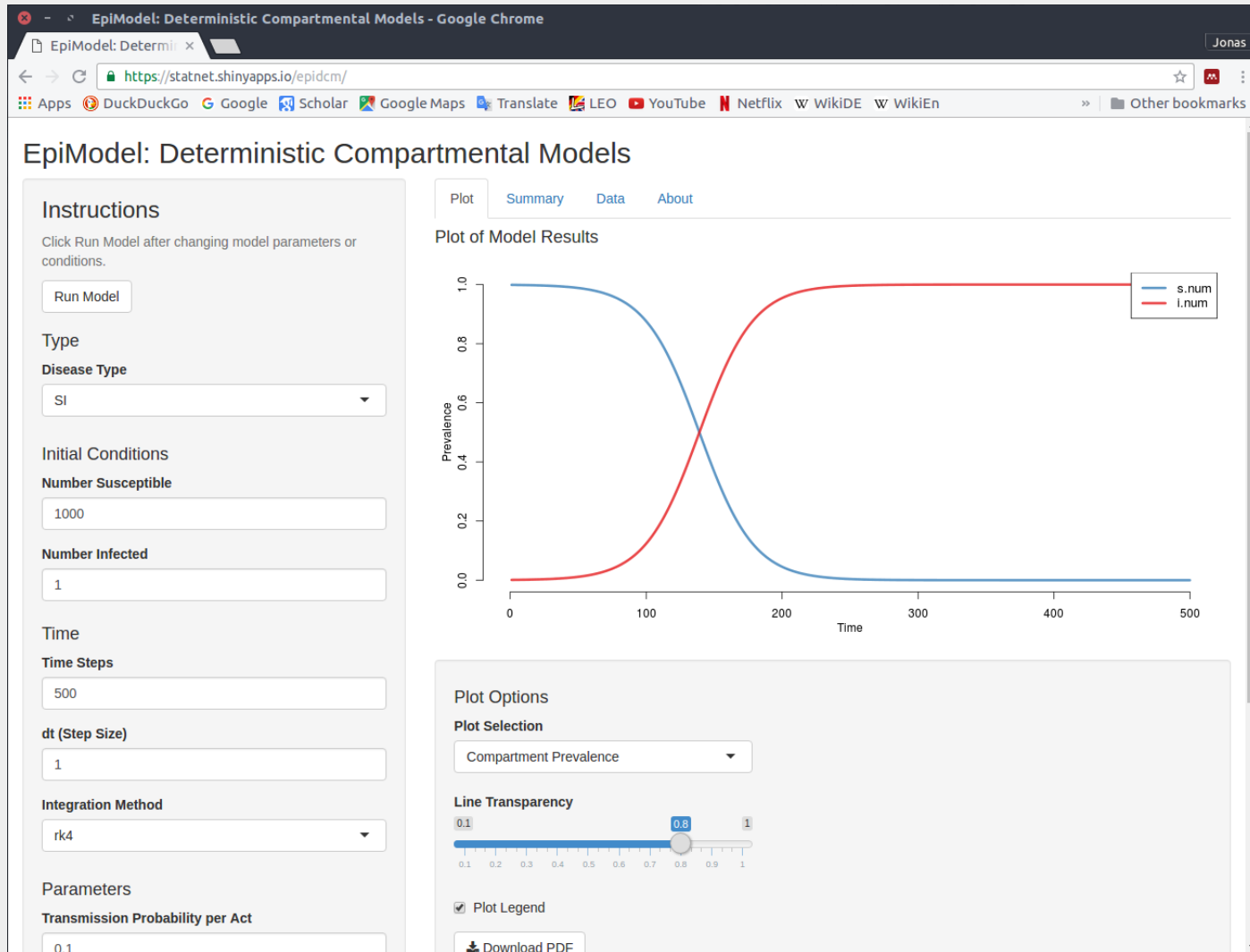
Wittgenstein Centre for Demography and Global Human Capital, (2015). Wittgenstein Centre Data Explorer Version 1.2. Available at: <http://www.wittgensteincentre.org/dataexplorer>

Who Uses It?



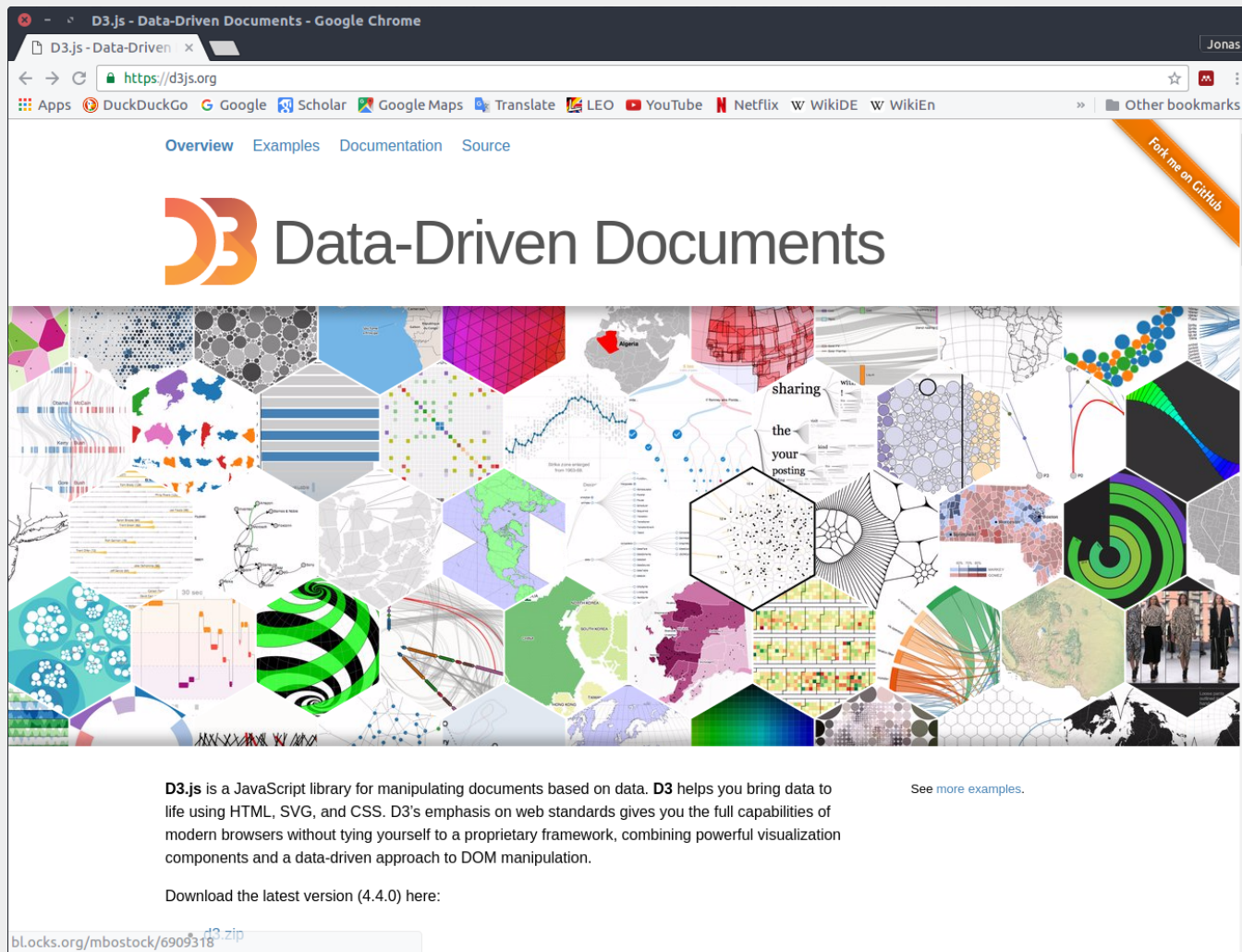
Jonas Schöley (2016): "The Human Mortality Explorer". <https://jschoeley.shinyapps.io/hmdexp/>

Who Uses It?



EpiModel: Deterministic Compartmental Models. <https://statnet.shinyapps.io/epidcm/>

Alternatives: D3.js

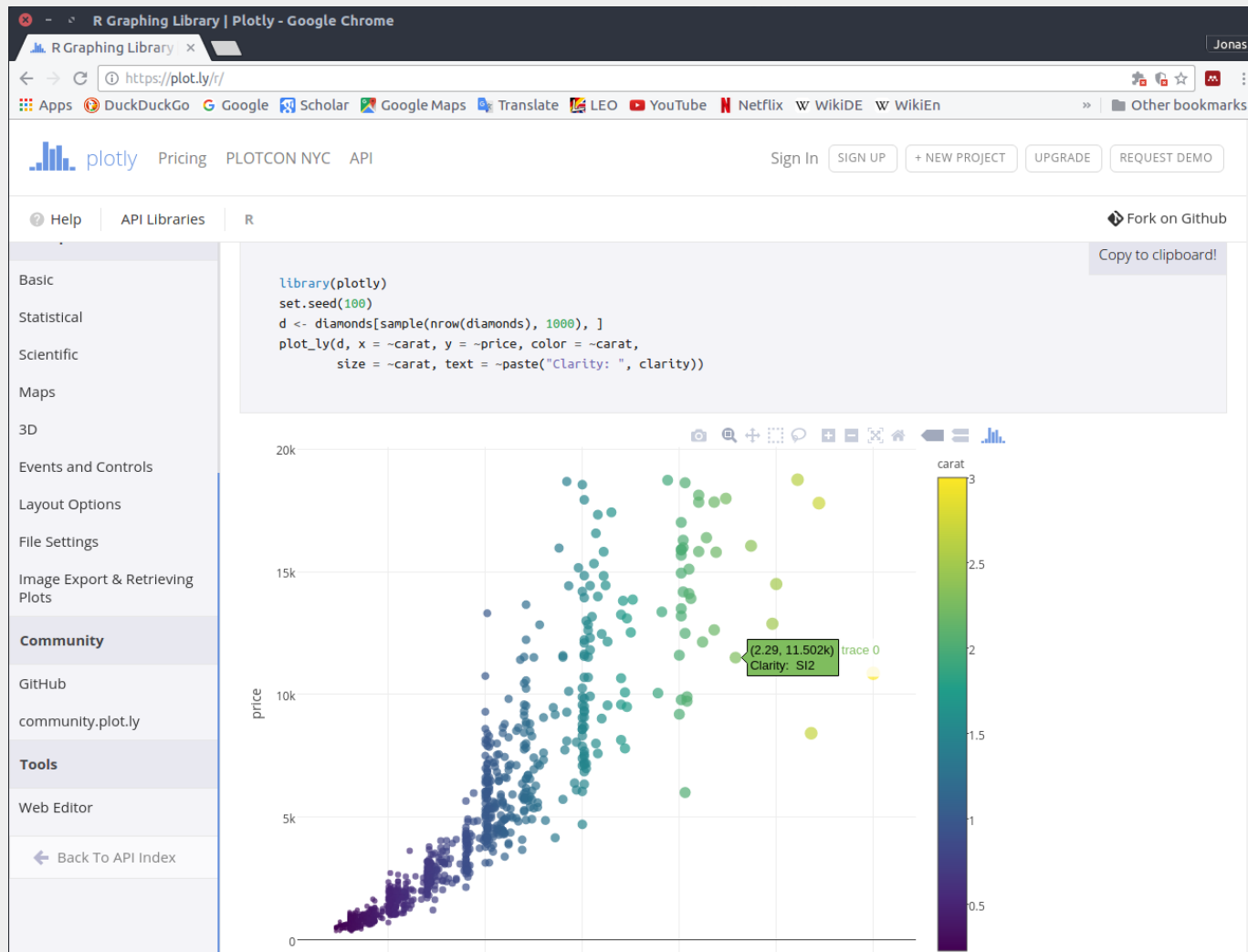


- + fast
- + does not require R/Shiny web server
- + can be easily embedded into an existing web-page
- + large user base
- requires knowledge of JavaScript, HTML, CSS
- does not provide statistical computing features of R

Alternatives: Bokeh

- + Python
- does not provide statistical computing features of R

Alternatives: plotly



- + very user-friendly
- + provides basic interactivity for an existing ggplot without any user-effort (ggplotly)
- web-server offerings are commercial

Slides available at
github.com/jschoeley/idem_viz

Jonas Schöley

jschoeley@health.sdu.dk

Twitter: [@jschoeley](https://twitter.com/jschoeley)