

Workshop Web-Apps mit R-Shiny

Simon Ress

Ruhr-Universität Bochum

15.03.2021

1. Überblick
2. Struktur & Layout
3. Inputs: Widgets
4. Outputs
5. Reactivity
6. Besprechung & Ausblick

Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

Chap. 1 Überblick

- ▶ Framework um Web-Apps in R zu erstellen (lokal o. online)
- ▶ Funktionen:
 - ▶ Reporting: Ermöglicht die Darstellung von Daten
 - ▶ Data collection: Ermöglicht das erfassen von Daten (z.B. Umfragen)
- ▶ Keine Kenntnisse von HTML, CSS o. JavaScript erforderlich für einfache Apps
- ▶ Reactivity: Outputs reagieren live auf veränderte Inputs

Chap. 2 Struktur & Layout

- ▶ Shiny-Apps bestehen aus zwei Komponenten:
 - ▶ ui: Gestaltung des User-Interface (Was sieht User: Struktur, Inhalte & Layout)
 - ▶ server: Back-end Struktur der Datenverarbeitung & Inhaltsgenerierung (Was macht App: z.B. Grafikerstellung)
- ▶ Speicherung der Komponenten in einer Datei (app.r) oder einzelnen Dateien (ui.r & server.r) möglich

▶

ui.r & server.r: Code

```
#ui.r:
# Creating a fluid page layout
# Consists of rows which in turn include columns
# Adjust page automatically to the browser dimensions
fluidPage(
  # Application title
  titlePanel("My first App")
)
```

```
#server.r:
#install & load package
if(!require("shiny")) install.packages("shiny")
library(shiny)

#Define server framework
shinyServer(function(input, output) {
})
```

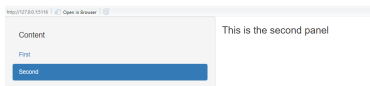
Übung in R: '0 Hallo World'

- ▶ Das User Interface kann auf verschiedensten Ebenen strukturiert werden
- ▶ Diese Ebenen sind ineinander verschachtelt
- ▶ Die Ebenen sind nicht voneinander abhängig und können einzeln & z.T. auch in anderer Schachtelung genutzt werden

1. Ebene: Navigation I

- `navlistPanel()`: Navigationsliste für einzelne Seiten
(→ `tabPanel()`) rechts

```
#ui.r:  
navlistPanel(  
  "Content",  
  tabPanel("First", h3("This is the first panel")),  
  tabPanel("Second", h3("This is the second panel"))  
)
```



Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

1. Ebene: Navigation II

- ▶ `navbarPage()`: Navigationsliste für einzelne Seiten
(→ `tabPanel()`) oben

```
#ui.r:
navbarPage("Content",
  tabPanel("First", h3("This is the first panel")),
  navbarMenu("More",
    tabPanel("Second", h3("This is the second panel")),
    tabPanel("Third", h3("This is the third panel"))
  )
)
```

http://127.0.0.1:5116 |  Open in Browser | 

Content

First

More ▾

Second

Third

This is the th

2. Ebene: Aufteilung der Seite

- ▶ sidebarLayout(): Aufteilung der Seite zwei Spalten

```
#ui.r:  
sidebarLayout(  
  sidebarPanel(  
    h3("This is the sidebar")  
  ),  
  mainPanel(  
    h3("This is the main panel")  
  )  
)
```

http://127.0.0.1:5116 |  Open in Browser | 

This is the sidebar

This is the main panel

Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

3. Ebene: Tab-Struktur auf einer Seite

- ▶ tabsetPanel(): Box die verschiedene Tabs darstellen kann

```
#ui.r:  
tabsetPanel(type = "tabs",  
            tabPanel("First",  
                      h3("This is the first tab")),  
            tabPanel("Second",  
                      h3("This is the second tab")),  
            tabPanel("Third",  
                      h3("This is the third tab"))  
)
```

http://127.0.0.1:5116 |  Open in Browser | 

First

Second

Third

This is the third tab

Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

4. Ebene: Platzierung der Elemente auf Seite

- ▶ `fluidRow()`: Alle Elemente innerhalb, werden in einer Zeile dargestellt
- ▶ `column()`: Definiert die Spalten innerhalb der Zeile

```
#ui.r:
fluidRow(column(2, h3("row:1 / column: 1")),
          column(2, h3("row:1 / column: 2")),
          column(2, h3("row:1 / column: 3"))
),
fluidRow(column(1, h3("row:2 / column: 1")),
          column(1, h3("row:2 / column: 2")),
          column(1, h3("row:2 / column: 3"))
)
```

<http://127.0.0.1:5116>[Open in Browser](#)

row:1 / column: 1

row:1 / column: 2

row:1 / column: 3

row:2 /
column:
1row:2 /
column:
2row:2 /
column:
3

- In Shiny können vereinfacht HTML-Befehle genutzt werden, um Inhalte in bestimmter Form darzustellen

| Shiny | HTML | Content |
|-------------|------------|---------------------------|
| p() | <p> | A paragraph of text |
| h1() - h6() | <h1>- <h6> | Header of different level |
| a() | <a> | Hyperlink |
| img() | | Image |
| br() | | Line break |
| strong() | | Bold text |
| em() | | Italicized text |
| HTML() | | Directly pass HTML code |

Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

Übung in R: '1 Structure'

Chap. 3 Inputs: Widgets

- Select box

A set of navigation icons typically found in Beamer presentations, including symbols for back, forward, search, and other slide controls.

Nutzung von Widgets

- ▶ Widgets werden in der `ui.r` definiert
- ▶ Benötigen verschiedene Inputs für Argumente
 - ▶ check `help(<command>)`
- ▶ Erstes Argument ist meist die `'inputId'` des Widgets
- ▶ Mittels der **'inputId'** kann der Input des Widgets (z.B. eine Zahl) im Server abgerufen werden

```
#ui.r:
fluidPage(
  # Application title
  titlePanel("My first App"),
  #Numeric input
  numericInput("num", label = h3("Numeric input"),
               value = 1),
  #Text input
  textInput("text", label = h3("Text input"),
            value = "Enter text..."),
  ...
)
```

Übung in R: '2 Widgets'

Chap. 4 Outputs

Outputs

- ▶ In server.r & ui.r Änderungen vorgenommen werden
- ▶ server: Hier wird der Output erstellt und gerendert
 - ▶ Output-Erstellung: z.B Produktion einer Grafik mittels ggplot()
 - ▶ Rendering: Transformation des R-Outputs in eine von Browsern lesbare Sprache (JS, HTML, CSS)
- ▶ ui: Hier wird ein Platzhalter für den Output benötigt (passend zum render***-Befehl im Server)

| Server (Render function) | User Interface (Output function) | Content |
|-----------------------------|-------------------------------------|---|
| renderDataTable({}) | dataTableOutput() | DataTable |
| renderPlot({}) | plotOutput() | Plots |
| renderTable({}) | tableOutput() | data frame, matrix, other table like structures |
| renderImage({}) | imageOutput() | images |
| renderText({}) | textOutput() | character strings |

1 Schritt: Platzhalter setzen

```
#ui.r:
fluidPage(
  # Application title
  titlePanel("My first App"),
  #Slider Bar
  sliderInput("obs", "Number of observations:",
              min = 1, max = 50, value = 5)
  #Histogram-Output
  plotOutput("histPlot"),
)
```

```
#server.r:
shinyServer(function(input, output) {
  #Histogram-Creation
  output$histPlot <- renderPlot({
  })
})
```

2 Schritt: Server-Code ausbauen

- ▶ Einsetzen des eigentlichen R-Codes zur Erstellung des Histogramms
- ▶ Hier wird ein Histogramm erzeugt, dass die Werte der Variable *'speed'* des *'cars'* Datensatzes darstellt

```
#server.r:  
shinyServer(function(input, output) {  
  #Histogram-Creation  
  output$histPlot <- renderPlot({  
    hist(cars$speed)  
  })  
})
```

Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

Übung in R: '3 Outputs'

Chap. 5 Reactivity

- ▶ Drei Arten von *reaktiveobjects* in Shiny:
 - ▶ Reactive sources: Ist in der Regel eine Benutzereingabe über eine Browser-Oberfläche (Widgets: `input$*`)
 - ▶ Reactive conductors: Komponente zwischen source & endpoint. Kapselung rechenintensiver o. wiederholter Operationen
 - ▶ Reactive endpoints: Normalerweise etwas, das im Browserfenster des Benutzers angezeigt wird (`output$*`)
- ▶ Reactive conductors & endpoints werden erneut ausgeführt, wenn sich min. eine beinhaltete source ändert
- ▶ Reactive endpoints werden erneut ausgeführt, wenn sich min. ein(e) beinhaltete source o. conductor ändert

Überblick

Struktur & Layout

Inputs: Widgets

Outputs

Reactivity

Besprechung &
Ausblick

Reactivity: Konzepte & Umsetzung in R-Shiny

| Allg. Konzepte | Umsetzung in R Shiny | Eigenschaften |
|-----------------------|-------------------------|---|
| Reactive source | reactive value | <ul style="list-style-type: none">- Reagieren auf User-Inputs- Geben einen Wert aus |
| Reactive conductor | reactive expression | <ul style="list-style-type: none">- Reagieren auf <i>reactive values</i> & andere <i>reactive expressions</i>- Geben einen Wert aus |
| Reactive endpoint | observer | <ul style="list-style-type: none">- Reagieren auf <i>reactive values</i> & <i>reactive expressions</i>- Geben keinen Wert aus- Haben side effects |

Reactivity: Commands

- ▶ Reactive values:
 - ▶ ***Input()**: Evaluiert Code immer wieder, wenn sich einer der beinhalteten reaktive values ändert
 - ▶ **reactiveValues()**: Definiert ein R-Objekt als Speicherplatz für reactive values
- ▶ Reactive expressions:
 - ▶ **reactive()**: Erstellt eine *reactive expression*
- ▶ Observer:
 - ▶ **render*()**: Erzeugt einen observer output\$. Wird bei veränderten zugehörigen *reactive values* & *reactive expressions* erneut ausgeführt
 - ▶ **observeEvent(eventExpr, Expr)**: Ausführung des beinhalteten Codes, nur wenn sich `< eventExpr >` ändert
 - ▶ **observe()**: Wie reactive(), nur das keine Ausgabe erzeugt wird
- ▶ Others:
 - ▶ **isolate()**: Unterrückt die Reaktivität eines R-Objekts

2 Schritt: Server-Code (reaktiv) ausbauen

- ▶ Einsetzen des eigentlichen R-Codes zur Erstellung des Histogramms
- ▶ Das Histogramm reagiert auf den Input mittels 'input\$obs'
 - ▶ Hiermit wird der Wert des sliderInputs abgerufen
 - ▶ Der Datensatz wird auf die jeweilige Fallzahl beschränkt
- ▶ render*()-Funktionen aktualisieren sich automatisch sobald sich der Wert eines enthaltenen Inputs (→ *input\$**) ändert

```
#server.r:  
shinyServer(function(input, output) {  
  #Histogram-Creation  
  output$histPlot <- renderPlot({  
    data <- cars$speed[1:input$obs]  
    hist(data)  
  })  
})
```

Reactivity: Reactive Graph

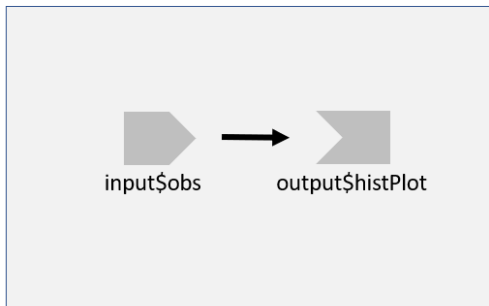
Reactive **source**



Reactive **conductor**



Reactive **endpoint**



Übung in R: '4 Reactivity'

Chap. 6 Besprechung & Ausblick

-

- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ↺ 🔍 ↻

- ▶ R-Shiny Homepage
- ▶ Shiny Themes
- ▶ Shiny tutorial
- ▶ Shiny Layout guide
- ▶ Shiny Build-in Widgets
- ▶ Shiny Function reference
- ▶ Huge collection of exemplary Shiny Apps
- ▶ shinydashboard

Gibt es noch weitere Fragen?