Shiny

Creating interactive web applications in R

Applied R Munich

Session 2

Referenten: Oliver Engl, Renate Moller

Institut für Statistik
LMU

14.12.2015

Aufbau

1. Einführung
2. Aufbau einer Shiny-App
3. Beispiel aus dem Consulting
4. Zusammenfassung
5. Tutorium

Was ist Shiny?

- Ein web application framework für R

Genauer gesagt:

- ein R Paket von RStudio
- mit dessen Hilfe man einfach interaktive Anwendungen für das Internet in R erstellen kann

Warum/Wofür Shiny?

- Erstellen von Web-Anwendungen ohne Web-Development-Kenntnissen
- Einfache und dynamische Präsentation von Ergebnissen
- Ermöglicht auch fachfremden Personen den Umgang mit komplexen Methoden

Shiny – Erste Schritte

```
### Paket installieren und laden
install.packages("shiny")
library(shiny)

### Beispiele fuer Shiny-Apps:
runExample()
# Valid examples are "01_hello", "02_text", "03_reactivity",
# "04_mpg", "05_sliders", "06_tabsets", "07_widgets",
# "08_html", "09_upload", "10_download", "11_timer"
runExample("01_hello")
```

Struktur einer Shiny-App I

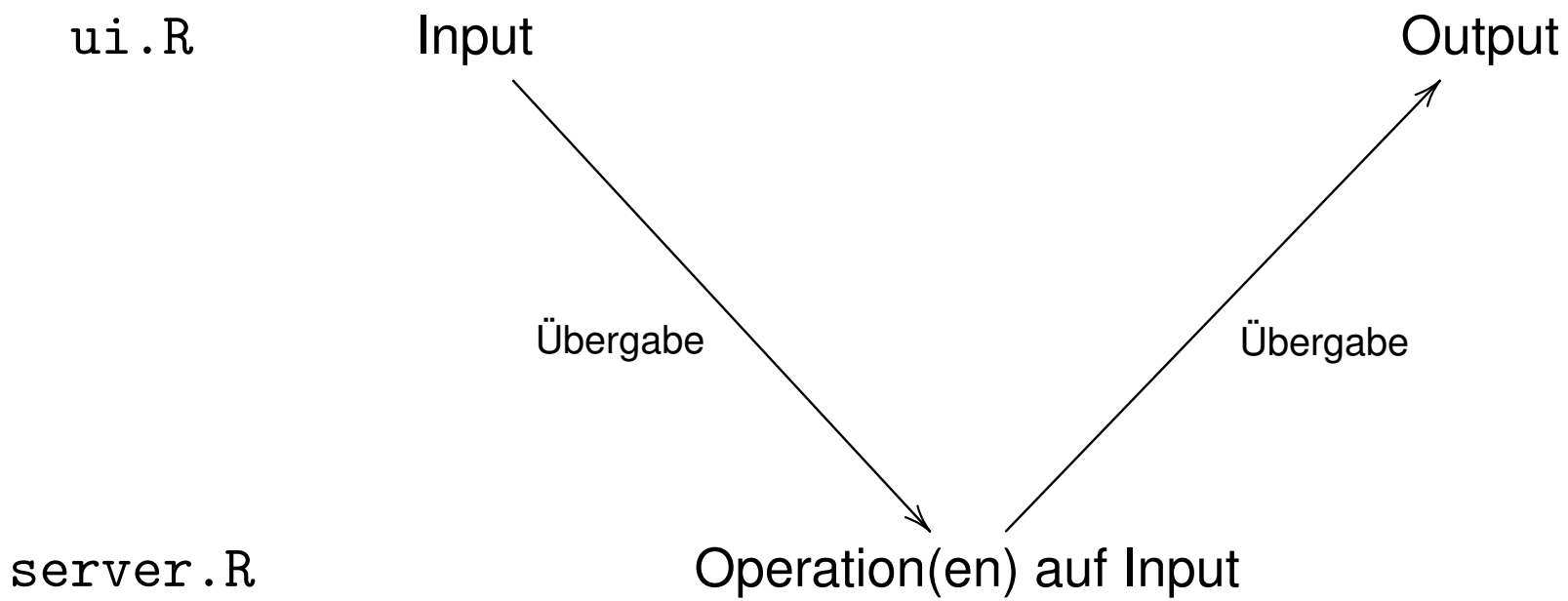
Zweigeteilte Struktur:

- `ui.R`: Layout des User Interfaces
- `server.R`: Berechnungen im Hintergrund (in R)

$\Longrightarrow$ zusammen in einem Ordner (z.B. „Demo_Code“) speichern
$\Longrightarrow$ Befehl: `runApp("Demo_Code")`

Vorsicht: Namen der zwei Dateien so festgelegt!

Struktur einer Shiny-App II



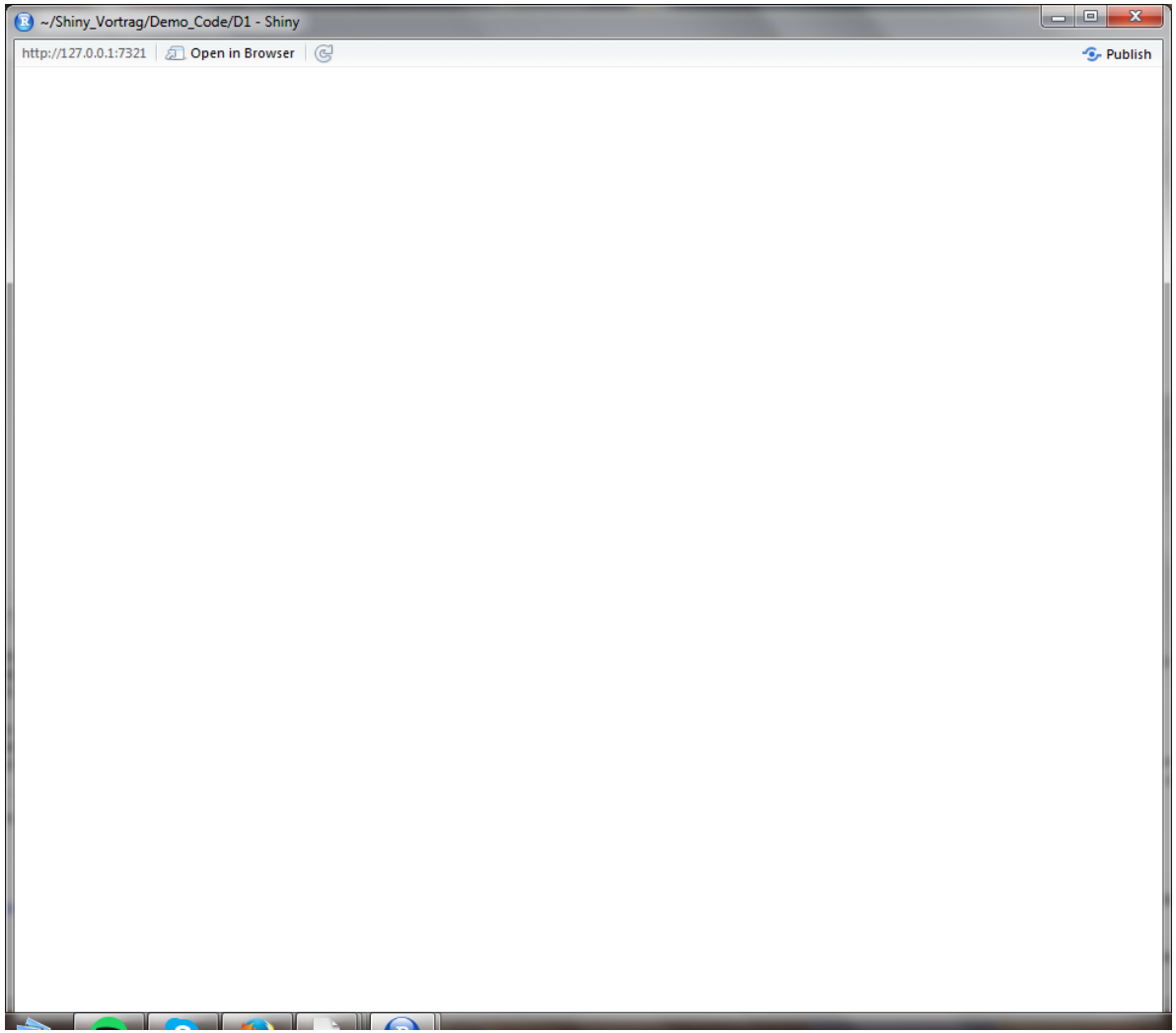
Grundgerüst I

```
1 # ui.R
2
3 shinyUI(fluidPage(
4   ))
```

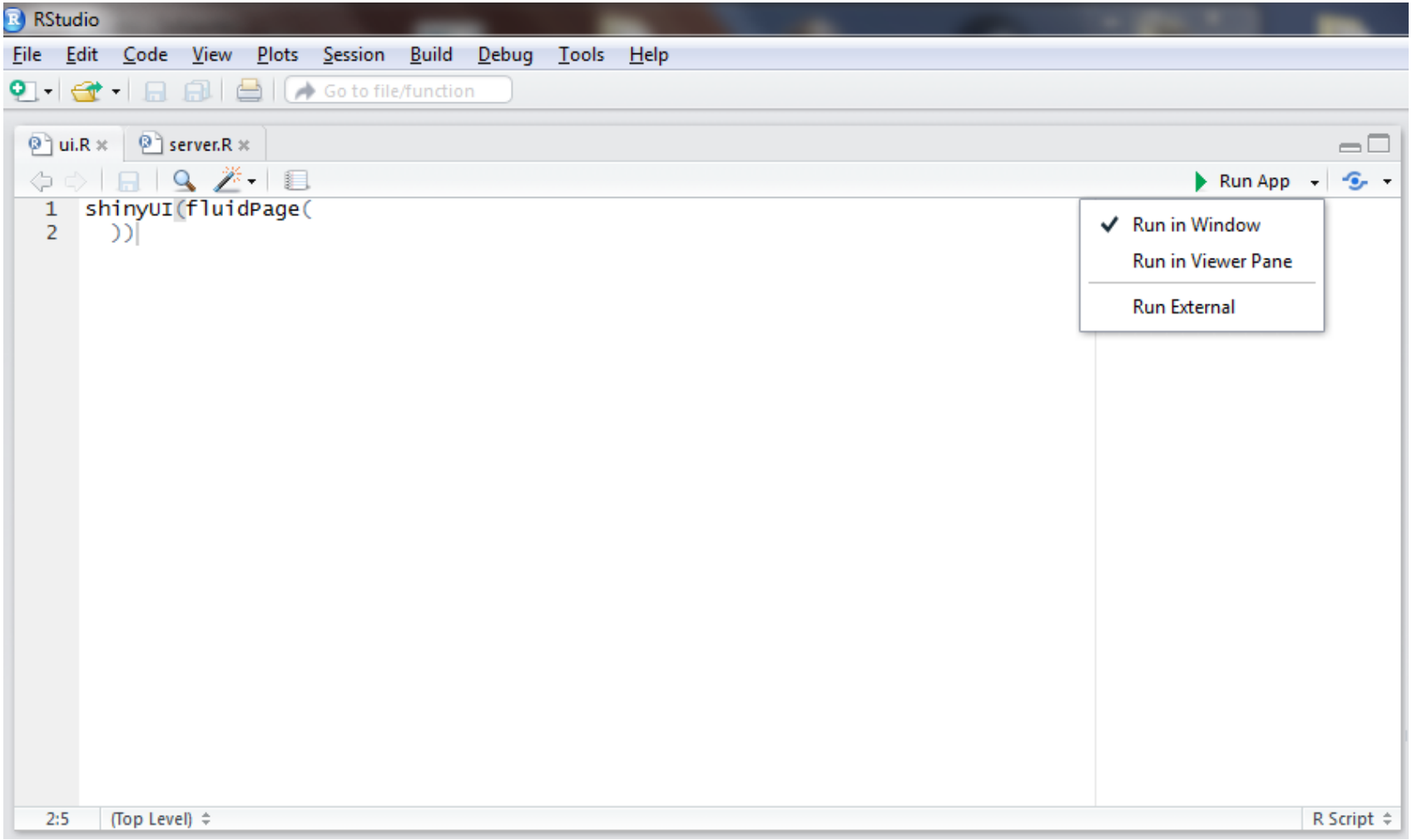
```
1 # server.R
2
3 shinyServer(function(input, output) {
4 })
```

$\Longrightarrow$ leere App

Grundgerüst II



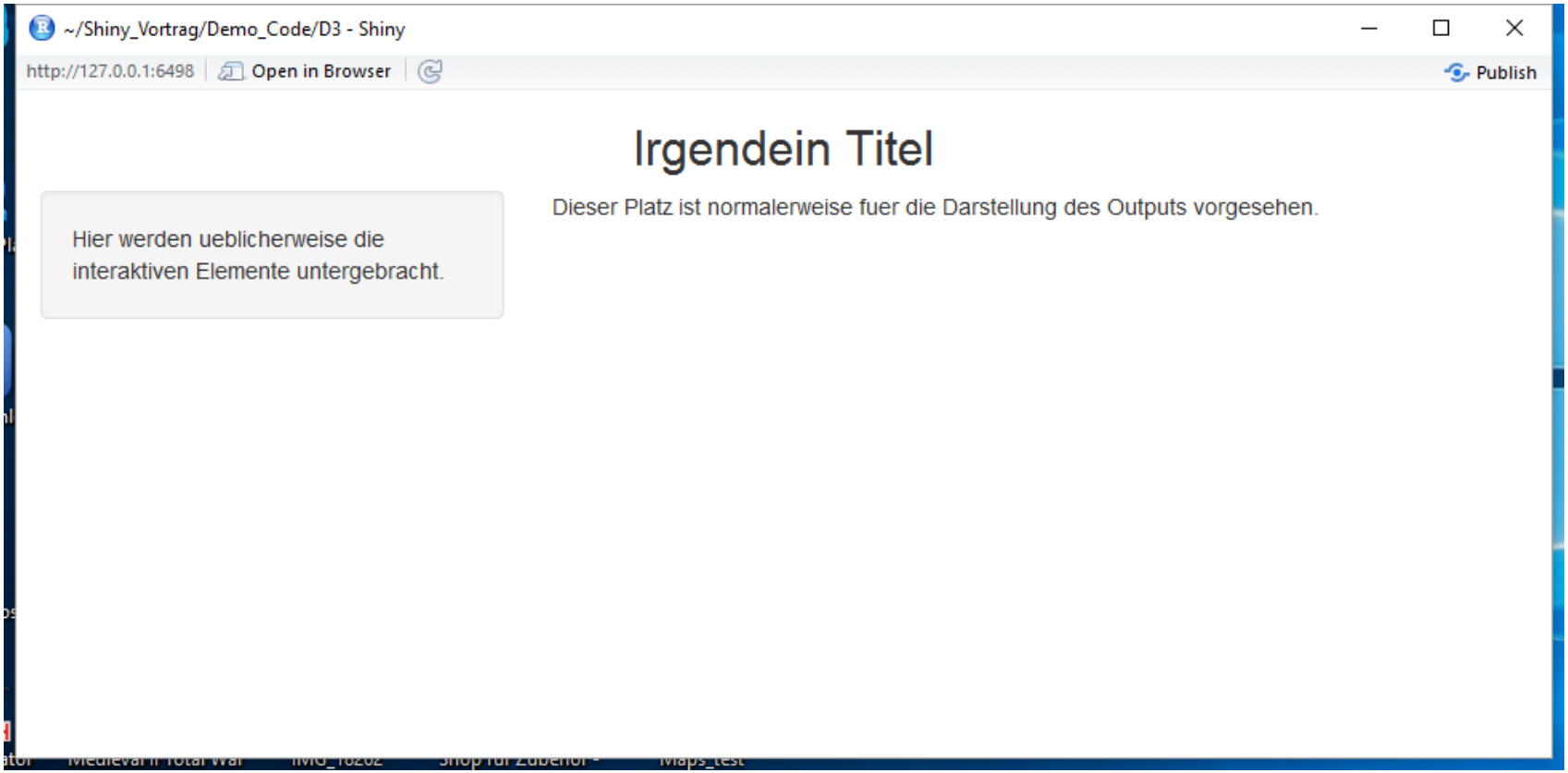
Start-/Ansichtsmöglichkeiten



User Interface bilden mit `ui.R` – Aufbau I

```
1  # ui.R
2
3  shinyUI(fluidPage(
4    titlePanel(p("Irgendein Titel", align = "center")),
5
6    sidebarLayout(
7      # standardmaessig links:
8      sidebarPanel("Hier werden ueblicherweise die
9                    interaktiven Elemente untergebracht."),
10     # standardmaessig rechts:
11     mainPanel("Dieser Platz ist normalerweise fuer die
12                Darstellung des Outputs vorgesehen.")
13   )
14 ))
```

Aufbau II


~/Shiny_Vortrag/Demo_Code/D3 - Shiny
http://127.0.0.1:6498
Open in Browser
Publish
Irgendein Titel
Hier werden ueblicherweise die interaktiven Elemente untergebracht.
Dieser Platz ist normalerweise fuer die Darstellung des Outputs vorgesehen.

Layout der App

$\Longrightarrow$ für fortgeschritteneres Layout: „Application layout guide“ im Abschnitt „Articles“ der Shiny-Website

Textlayout

Textdarstellung und -format erweiterbar durch sog. „HTML tag functions“:

Befehl	**Funktionsweise**
`p(...)`,	bildet einen Text-Paragraphen
`div(...)`,	Textteil wird in einem einheitlichen Stil dargestellt
`h1(...),h2(...),...,h6(...)`,	Überschriften unterschiedlicher Größe
`br()`,	Zeilenumbruch bzw. Leerzeile
`code(...)`,	Text wird als Code formatiert
`strong(...)`,	fettgedruckter Text
`em(...)`,	kursiver Text
`img(...)`,	Bild einfügen

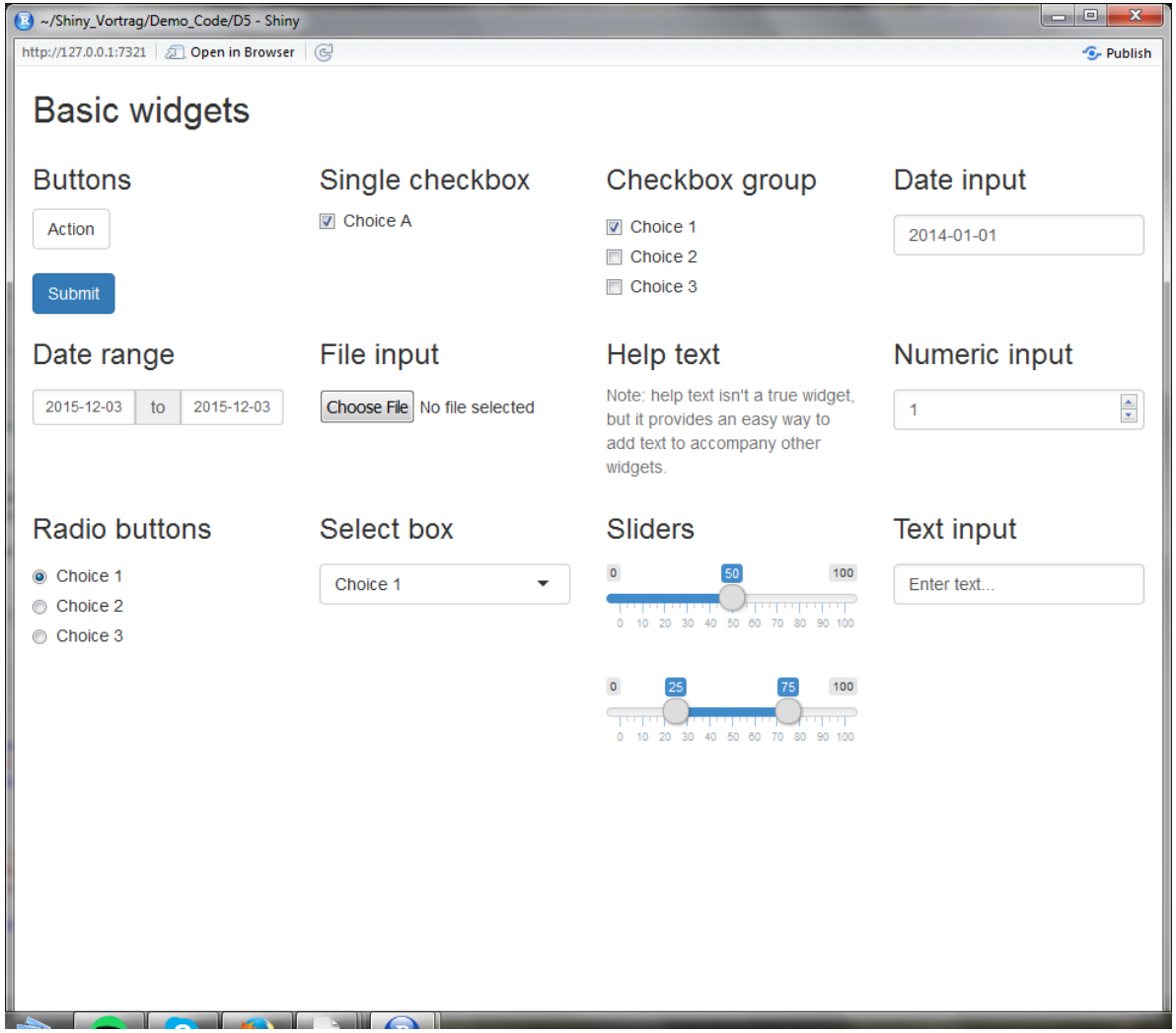
... und mehr

$\Longrightarrow$ jeweils verschiedene Konfigurationen wie z.B. `align` oder `style` möglich

Interaktivität des User Interfaces

- bisher: nur „starres“ Layout des User Interfaces ohne interaktive Elemente
- ändert sich durch Einfügen von „Widgets“
- Widgets empfangen Befehle/Eingaben des Nutzers

Übersicht Widgets I



Übersicht Widgets II

Befehl	**Funktionsweise**
`actionButton(...)`	Zähl-Schalter
`submitButton(...)`	Update-Schalter
`checkboxInput(...)`	einzelne Checkbox
`checkboxGroupInput(...)`	mehrere Checkboxen zur (gleichzeitigen) Auswahl
`dateInput(...)`	Datumsauswahl
`dateRangeInput(...)`	Periodenauswahl
`fileInput(...)`	Dateiauswahl
`helpText(...)`	Hilfetext
`numericInput(...)`	(numerische) Werteingabe
`radioButtons(...)`	mehrere (sich ausschließende) Optionen zum Wählen
`selectInput(...)`	Eingabefeld mit Auswahlmöglichkeiten
`sliderInput(...)`	Schieber zur Werteauswahl
`textInput(...)`	ein Textfeld

Widgets: Konfiguration I

- verschiedene Argumente der jeweiligen Funktionen
- für genaue Beschreibung: Abschnitt „Reference“ der Shiny-Website
- (fast) alle gemeinsam: 1.Stelle – interner Name der Eingabe; 2.Stelle – im User Interface dargestellter Name des Widgets

Widgets: Konfiguration II

```
# ui.R

shinyUI(fluidPage(
  titlePanel("Basic widgets"),

  fluidRow(
    # Zaehl-Schalter und Update-Schalter
    column(3,
           h3("Buttons"),
           actionButton("action", label = "Action"),
           br(),
           br(),
           submitButton("Submit")),
    # einzelne Checkbox
    column(3,
           h3("Single checkbox"),
           checkboxInput("checkbox", label = "Choice A",
                         value = TRUE))
  )
))
```

Von Interaktivität zu Reaktivität I

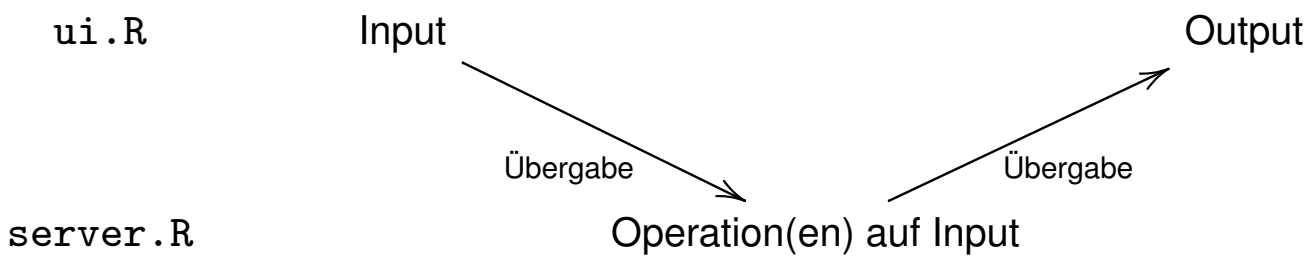
- bis jetzt: interaktiver Input „zeigt ins Leere“
- kein (reaktiver) Output

$\Longrightarrow$ nächster Schritt: `server.R` verwenden, um reaktiven Output zu erzeugen

Von Interaktivität zu Reaktivität II

3 Schritte für **reaktiven** Output:

1. Widgets zu `ui.R` hinzufügen
2. Code in `server.R` schreiben, der auf Eingaben der Widgets zurückgreift und neue Objekte/Werte erzeugt
3. Output-Objekt zu `ui.R` hinzufügen



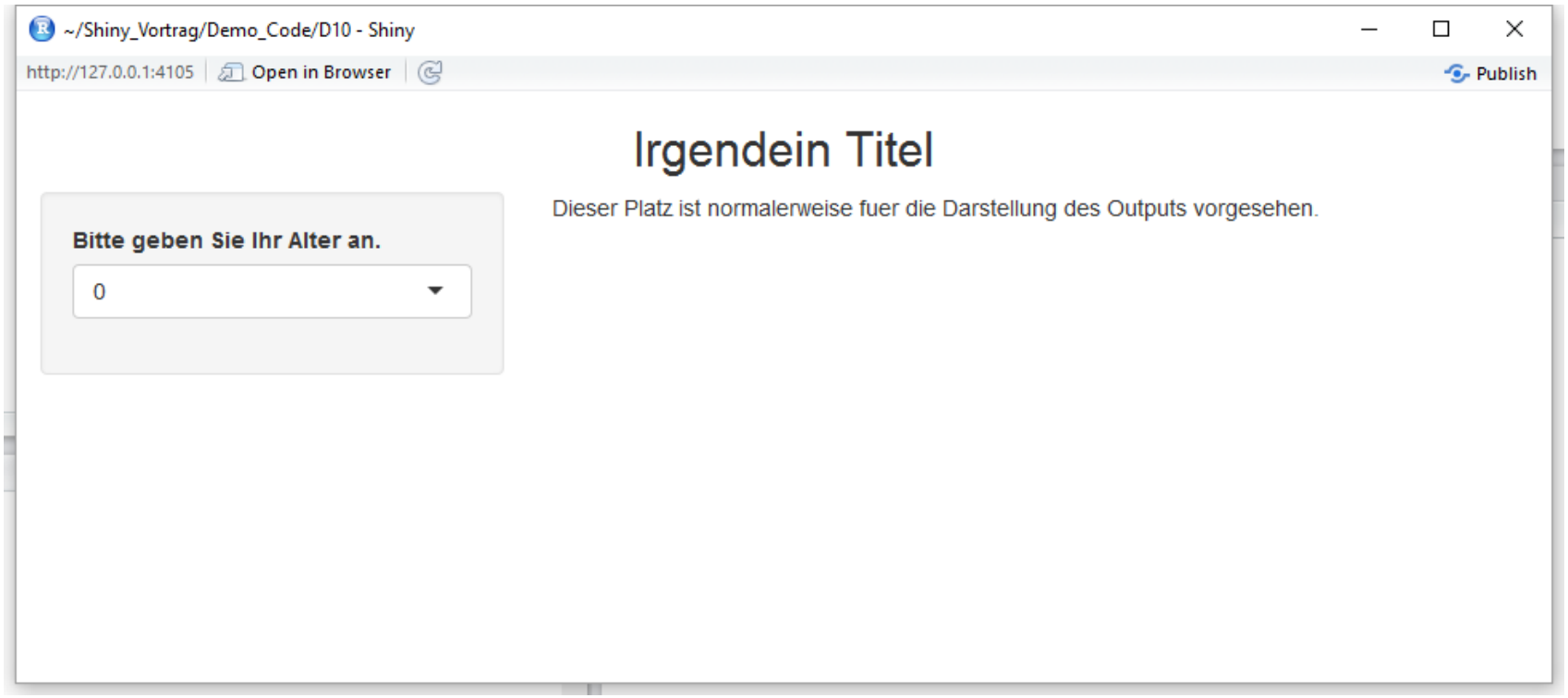
Von Interaktivität zu Reaktivität III

1. Widgets zu `ui.R` hinzufügen: schon bekannt ✓

In unserer Beispiel-App:

```
1  #   ui.R
2
3  shinyUI(fluidPage(
4    titlePanel(p("Irgendein Titel", align = "center")),
5
6    sidebarLayout(
7      # Widget zur Angabe des Alters:
8      sidebarPanel(selectInput("Alter",
9                               label ="Bitte geben Sie Ihr
10                              Alter an.",
11                              choices = 0:150)),
12     mainPanel("Dieser Platz ist normalerweise fuer die
13                Darstellung des Outputs vorgesehen.")
14   )
15 ))
```

Von Interaktivität zu Reaktivität IV



Operationen in `server.R` I

2. Code in `server.R` schreiben:

- Erzeugen eines Output-Objekts in `server.R`:
  `output$beliebiger_Name <- render*({})`
- Form der `render*({})`-Funktion hängt vom gewünschten Output ab:

render-Funktion	**geeignet für**
`renderText({})`	Charakter-Strings
`renderPlot({})`	Plots
`renderTable({})`	Data-Frames und Matrizen
`renderImage({})`	Bilder
`renderPrint({})`	beliebigen gedruckten Output
`renderUI({})`	Shiny-Tag-Funktionen oder HTML

Operationen in `server.R` II

- `render*({})`-Funktionen einmal beim Laden der App berechnet
- danach jedes Mal, wenn sich (Eingabe-)Wert innerhalb von `({...})` ändert
- Einfügen von Eingabewerten in Operation `({...})` mittels `input$interner_Name_des_Widgets`

$\Longrightarrow$ Beispiel-App

$\Longrightarrow$ `function(input,output){}` zentrale Verknüpfung

Operationen in `server.R` III

```
1  # server.R
2
3  shinyServer(
4    # Funktion als Verknuepfung von Input und Output:
5    function(input, output) {
6
7      # Textoutput in Abhaengigkeit von der
8      # Altersangabe erzeugen:
9      output$text <- renderText({
10       paste("Sie sind angeblich", input$Alter,
11             "Jahre alt.")
12     })
13
14   }
15 )
```

Output im User Interface I

3. Output-Objekt zu `ui.R` hinzufügen:

- Ausgabe eines Output-Objekts in `ui.R`: `*Output("Name_des_Objekts")` $\Longrightarrow$ Name muss mit der Bezeichnung des Outputs in `server.R` übereinstimmen
- Form der `*Output()`-Funktion stimmt meist mit Form von `render*({})` überein:

Output-Funktion	**Ausgabe**
`textOutput()`	Text
`plotOutput()`	Plots
`tableOutput()`	Tabellen
`imageOutput()`	Bilder
`verbatimTextOutput()`	Text
`uiOutput()`	HTML
`htmlOutput()`	HTML

Output im User Interface II

Wo soll erzeugtes Objekt platziert werden?

$\Longrightarrow$ `*Output()` an geeigneter Stelle im User Interface hinzufügen

$\Longrightarrow$ Beispiel-App

Output im User Interface III

```
1  #   ui.R
2
3  shinyUI(fluidPage(
4    titlePanel(p("Irgendein Titel", align = "center")),
5
6    sidebarLayout(
7      sidebarPanel(selectInput("Alter",
8                               label = "Bitte geben Sie Ihr
9                               Alter an.",
10                              choices = 0:150)),
11     # Textoutput wird im Hauptfeld des UserInterfaces
12     # angezeigt:
13     mainPanel(textOutput("text"))
14   )
15 ))
```

Erinnerung: server.R

```
1  # server.R
2
3  shinyServer(
4    # Funktion als Verknuepfung von Input und Output:
5    function(input, output) {
6
7      # Textoutput in Abhaengigkeit von der
8      # Altersangabe erzeugen:
9      output$text <- renderText({
10       paste("Sie sind angeblich", input$Alter,
11             "Jahre alt.")
12     })
13
14   }
15 )
```

Eine vollständige App

Vorläufige Zusammenfassung

$\Longrightarrow$ Nötiges Wissen vorhanden, um selbst eine voll funktionsfähige Shiny-App zu erstellen!

Erinnerung – 3 Schritte:

1. Widgets zu `ui.R` hinzufügen
2. Code in `server.R` schreiben, der auf Eingaben der Widgets zurückgreift und neue Objekte/Werte erzeugt
3. Output-Objekt zu `ui.R` hinzufügen

App teilen

Zwei Möglichkeiten:

1. `ui.R` und `server.R` an andere R-User (einschließlich Zusatzmaterial)
2. als Website

App teilen II

1. an andere R-User:

   - Entweder Ordner mit Files (hier: „Demo_Code“) versenden
   - ODER Ordner auf eigener Website bzw. `www.github.com` hosten $\Longrightarrow$ Funktionen `runUrl`, `runGitHub`, `runGist`

App teilen III

2. als Website:

   - Website selber hosten
   - ODER mit Hilfe von RStudio:
     - über `http://www.shinyapps.io/` (Hosting Service von RStudio)
     - über Shiny Server (Web Server bilden, Info: `https://github.com/rstudio/shiny-server/blob/master/README.md`)
     - über Shiny Server Pro $\Longrightarrow$ mehr Möglichkeiten

$\Longrightarrow$ mehr Infos zum App teilen in Shiny Tutorial Lesson 7

Weitere Möglichkeiten des Pakets Shiny

Weitere Möglichkeiten beinhalten:

- Laden von Datensätzen/R-Skripts/Paketen
- Modularisierung des Codes in `server.R`
- Verwendung von `conditionalPanel` in `ui.R`

. . . und mehr (Abschnitt „Articles“ der Shiny-Website)

Laden von Zusatzmaterial

Verstehen einer Shiny-App – Aufbau von `server.R`:

```
1  ### server.R
2
3  # Zone 1  --> Laden von Datensaetzen, R-Skripts,
4  # R-Paketen
5
6  shinyServer(
7    function(input, output) {
8      # Zone 2 --> Session-Information
9      output$text <- renderText({
10       # Zone 3 --> Widget-Input (schon gesehen)
11       paste("Sie sind angeblich", input$Alter,
12             "Jahre alt.")
13     })
14   }
15 )
```

Modularisierung des Codes in `server.R` I

Schlecht programmiert:

```
1  ### Ausschnitt einer server.R-Datei
2
3  # momentan: bei Wechseln zwischen normaler und Log-Skala:
4  # Datensatz wird neu eingelesen --> unnoetig
5  output$plot <- renderPlot({
6    data <- getSymbols(input$symb, src = "yahoo",
7                       from = input$dates[1],
8                       to = input$dates[2],
9                       auto.assign = FALSE)
10
11   chartSeries(data, theme = chartTheme("white"),
12               type = "line", log.scale = input$log,
13               TA = NULL)
14 })
```

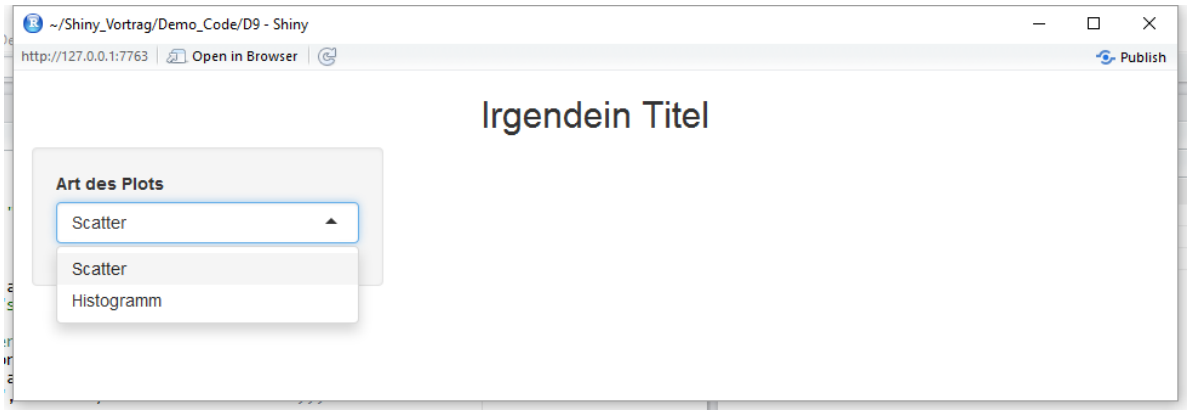
Modularisierung des Codes in `server.R` II

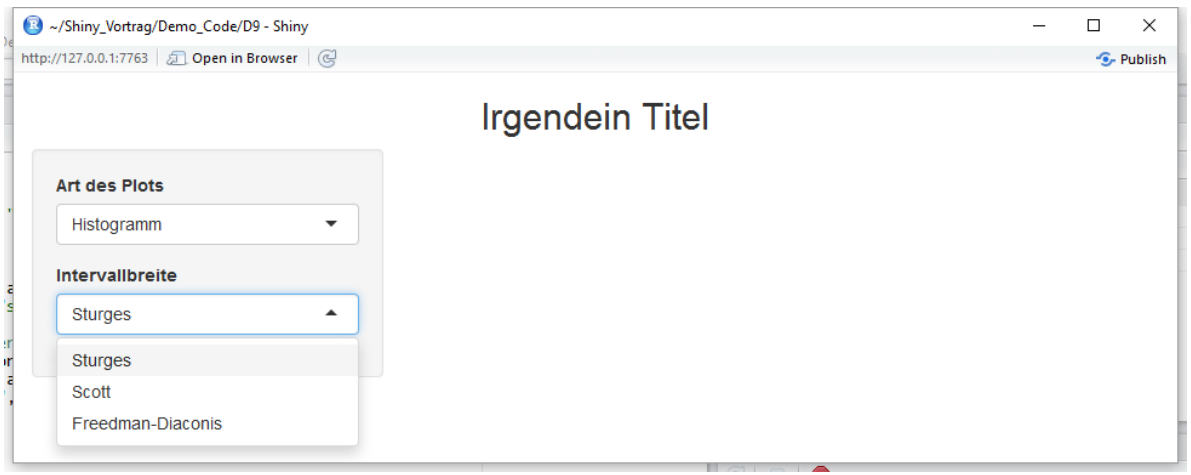
Gut programmiert:

```
### Ausschnitt einer server.R-Datei

# reactive-Teil: Datensatz wird nur neu eingelesen, wenn
# tatsaechlich noetig
dataInput <- reactive({
  getSymbols(input$symb, src = "yahoo",
             from = input$dates[1],
             to = input$dates[2],
             auto.assign = FALSE)
})

# render*-Teil: nur noch Input-Variable input$log enthalten;
# ansprechen des Datensatzes durch dataInput()
output$plot <- renderPlot({
  chartSeries(dataInput(), theme = chartTheme("white"),
              type = "line", log.scale = input$log,
              TA = NULL)
})
```

Verwendung von `conditionalPanel` I

Vorstellung: reaktiver Input





Verwendung von `conditionalPanel` II

entsprechender Code in `ui.R`:

```
1  #  ui.R
2
3  shinyUI(fluidPage(
4    titlePanel(p("Irgendein Titel", align = "center")),
5    sidebarLayout(
6      sidebarPanel(
7        selectInput("plotT", label = "Art des Plots",
8          choices = c(Scatter = "sca", Histogramm = "hist")),
9        # Widget nur anzeigen, wenn Art des Plots = Histogramm
10       # Vorsicht -- Schreibweise: input.plotT
11       conditionalPanel(condition = "input.plotT == 'hist'",
12         selectInput("breaks", label = "Intervallbreite",
13           choices = c("Sturges", "Scott",
14                       "Freedman-Diaconis")))
15     ),
16     mainPanel() # hier wird dann der Output-Plot dargestellt
17   )
18 ))
```

Bei Interesse . . .

. . . finden sich unter `http://shiny.rstudio.com/`

- im Abschnitt „Tutorial“ eine noch ausführlichere Einführung
- im Abschnitt „Articles“ weitere fortgeschrittene Methoden.

Beispiel aus dem Consulting

$\Rightarrow$ Beispiel aus dem Consulting

Zusammenfassung

- Einfache Erstellung von interaktiven Anwendungen
- Keine Web-development-Kenntnisse nötig
- server.R und ui.R Grundlage jeder Shiny-App
- Vorgehen:
  - Widget in ui.R schreiben
  - Code, der auf Eingabe im Widget zurückgreift, in server.R schreiben
  - Output-Objekt zu ui.R hinzufügen

Tutorium: Brummmm, brummmm....

Der Datensatz “mtcars“ mit 32 Beobachtungen und 11 Variablen:

- mpg: Miles/(US) gallon
- cyl: Number of cylinders
- disp: Displacement (cu.in.)
- hp: Gross horsepower
- drat: Rear axle ratio
- wt: Weight (lb/1000)
- qsec: 1/4 mile time
- vs: V/S
- am: Transmission (0 = automatic, 1 = manual)
- gear: Number of forward gears
- carb: Number of carburetors

Tutorium: Auf “los“ geht’s los

Bevor es losgehen kann:

- Package “shiny“ runterladen
- server.R und ui.R erstellen und speichern
- in server.R:
  - library(shiny)
  - library(datasets)
  - attach(mtcars)

Eine kleine Hilfestellung: http://shiny.rstudio.com/tutorial/lesson3/

Tutorium: Unsere kleine App - Teil 1

Nützliche Befehle für Aufgabe 1:

- barplot(table(cyl))
- sidebarLayout()
- pie(table(cyl))
- shinyServer(function(input, output) {...}
- shinyUI(fluidPage(...)
- titlePanel()
- sidebarPanel()
- mainPanel()
- plotOutput()
- output$... ← renderPlot({...})

Tutorium: Unsere kleine App - Teil 2

Nützliche Befehle für Aufgabe 2:

- selectInput()
- plot(hp,mpg)
- conditionalPanel()
- plot(hp)
- plot(mpg)
- if(){}
- plot(hp, mpg, pch=cyl, col=cyl)
- plotOutput()

Quellen

- Shiny.rstudio.com: „Shiny by RStudio“. Abgerufen am 03.12.2015 von `http://shiny.rstudio.com/`.
- Stcorp.nl: „Tutorial: creating webapps with R using Shiny“. Abgerufen am 03.12.2015 von `http://stcorp.nl/R_course/tutorial_shiny.html`.
- R-luminescence.de: „Shiny R.Lum – Interaktive Web Anwendungen für das R Paket Luminescence und ESR“. Abgerufen am 12.12.2015 von `http://www.r-luminescence.de/de/shiny.html`.
- Stat.ethz.ch: „R Documentation: mtcars“. Abgerufen am 12.12.2015 von `https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/mtcars.html`.

Vielen Dank für Ihre Aufmerksamkeit!