Eine kleine Einführung in R Shiny

... und ein paar Worte zu github

Johannes Bracher und Leonhard Held
johannes.bracher@uzh.ch, leonhard.held@uzh.ch

Institut für Epidemiologie, Biostatistik und Prävention, Universität Zürich,
12.11.2018

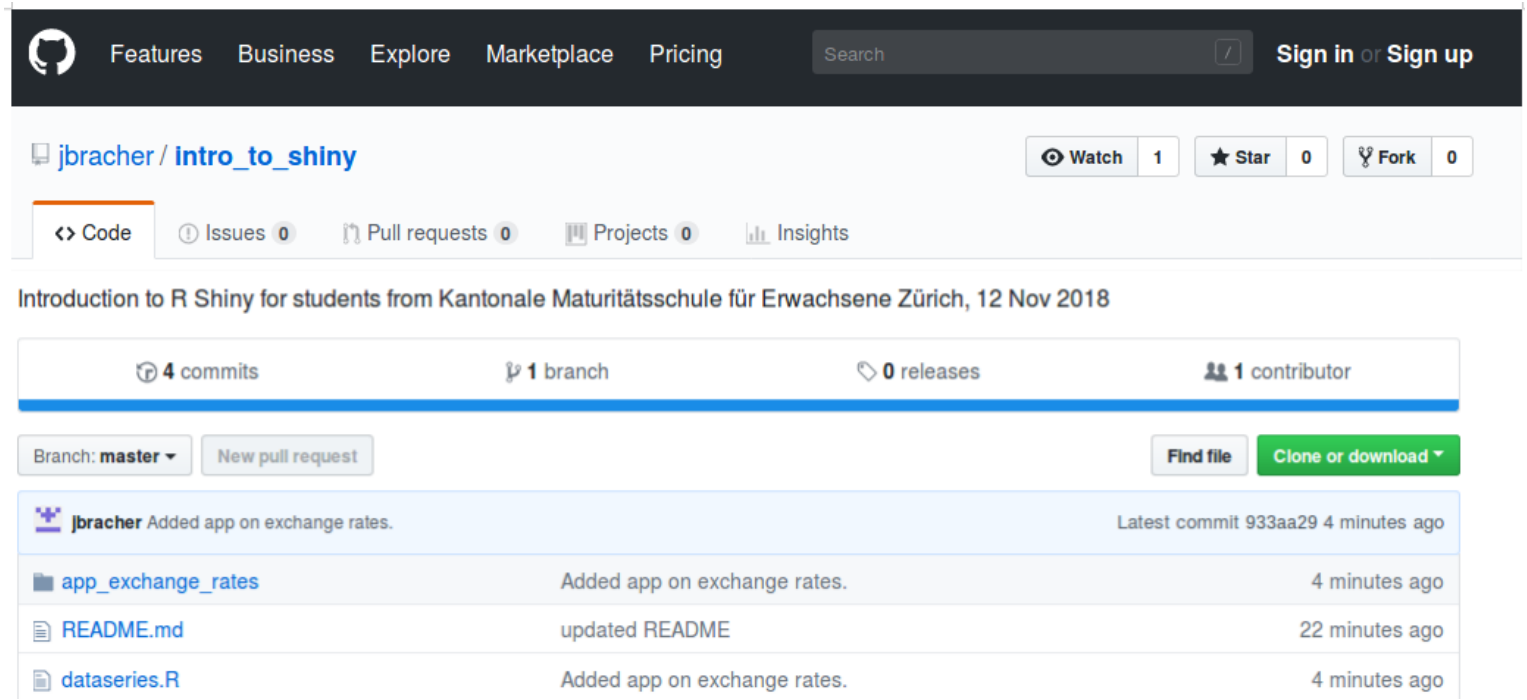
25. Oktober 2018

Vorneweg: `git` und `github`

- `git` ist ein freies System zur Versionskontrolle, d.h. genauen Dokumentierung der Arbeitsschritte in z.B. einem Software-Projekt (gängigste Alternative: svn)
- V.a. bei Projekten mit mehreren Mitarbeitern unerlässlich zu Austausch und Koordination
- `github.com` ist ein kommerzieller Dienst, der `git`-Repositories ($\approx$ Projekte) online hostet (Alternativen: `bitbucket.org`, `gitlab.com`, ...)

Materialien zu diesem Vortrag

- Link: `https://github.com/jbracher/intro_to_shiny`
- I.d.R. wird `git` aus der Kommandozeile bedient.
- Manuelles Herunterladen von Ordnern ebenfalls möglich:



Das R Packaging-System

- Eine der grössten Stärken von R ist das umfangreiche Package-System
- Packages sind von Nutzern beigesteuerte Programmerweitungen, z.B.
  - erweiterte Grafikfunktionen
  - zusätzliche Methoden zur Datenverarbeitung
  - Implementierung spezialisierter statistischer Methoden
  - Datensätze
  - Integration mit anderen Programmiersprachen
  - Integration mit Textverarbeitungsprogrammen
  - ...
- Derzeit 13.346 Packages im *Comprehensive R Archive Network* (CRAN)
- Aus unserem Department z.B. `surveillance`, `partykit`, `pCalibrate`

R Shiny

- Package, das es erlaubt, interaktive Web-Applikationen basierend auf `R` code zu erstellen. Installation:

```
install.packages("shiny")
```

- Programmierung komplett in `R`, kein `JavaScript`, `html` o.ä. nötig
- Vom Unternehmen RStudio entwickelt, jedoch open source (Quellcode unter `https://github.com/rstudio/shiny`)

Einige Beispiele

- Analyse von Infektionskrankheiten: https://jobrac.shinyapps.io/epidemic_modelling/
- Tourismusdaten: https://mbienz.shinyapps.io/tourism_dashboard_prod/
- Statistik-App für Kantonsschulunterricht: http://shiny.math.uzh.ch/user/furrer/shinyas/shiny-lwb/

Weitere Beispiele unter

- https://shiny.rstudio.com/gallery/
- https://www.rstudio.com/products/shiny/shiny-user-showcase/

Listen

```r
# Führe Liste l mit Elementen a = 1 und b = 2 ein:
l <- list(a = 1, b = 2)
# Rufe Element a aus Liste l ab (mittels Dollar-Zeichen)
l$a

## [1] 1

# Erweitere Liste l um Element c mit Wert "Hallo"
l$c <- "Hallo!"
l$c

## [1] "Hallo!"
```

Listen (2)

- Listen sind sehr flexible Objekte und können verschiedenste Objekte enthalten
  - numerische Variablen
  - Textvariablen
  - Funktionen
  - weitere Listen
- Den Elemente werden in der Regel Namen zugewiesen, sodass man leicht auf sie zugreifen kann

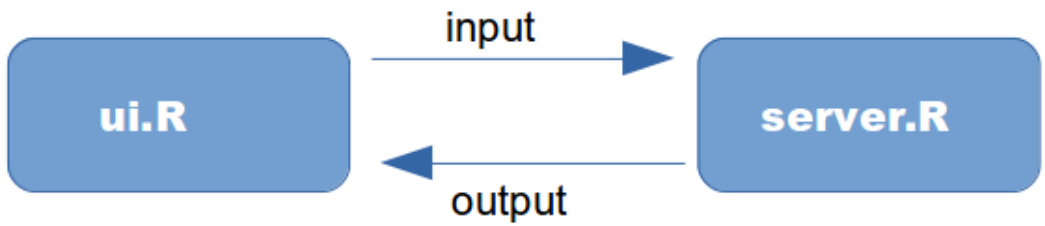
Grundlegende Struktur von Shiny Apps

Eine Shiny App besteht aus zwei Komponenten, die in der Regel in getrennten Dateien liegen:

- `ui.R`: User Interface. Definiert die Darstellung der App und die Eingabemöglichkeiten für den Benutzer. R-Code wird in HTML-Seite konvertiert.
- `server.R`: Definiert die vom Server durchzuführenden Operationen, meist abhängig von Nutzereingaben

Austausch zwischen `ui` und server

Die`ui` und die `server`-Komponenten der App tauschen sich über zwei Listen (bzw. "list-like objects") `input` und `output` aus:



Das Template

In RStudio: Erzeuge Template mit File → New File → Shiny App

ui.R

```
library(shiny) # lade package

# Definiere Benutzeroberfläche
shinyUI(fluidPage(
  # Titel:
  titlePanel("Old Faithful Geyser Data"),
  # Füge "Sidebar" hinzu
  sidebarLayout(
    sidebarPanel(
      # Definiere Slider-Eingabe für input-
      # Objekt "bins"
      sliderInput("bins", # Name der Eingabe
                  "Number of bins:", # Titel
                  min = 1, # Min. Wert
                  max = 50, # Max. Wert
                  value = 30) # Default
    ),
    # Füge ein "Main-Panel" hinzu
    mainPanel(
      plotOutput("distPlot")
      # Hier wird das Output-Objekt
      # "distPlot" von der server-
      # Seite verwendet!
    )
  )
))
```

server.R

```
library(shiny) # lade package

# Definiere Ablauf auf Server-Seite
shinyServer(function(input, output) {
  # Definiere Output-Objekt "distPlot"
  output$distPlot <- renderPlot({
    x    <- faithful[, 2] # lege Daten in x ab
    # lege Grenzen der "Bins" des
    # Histogramms fest:
    bins <- seq(min(x), max(x),
                length.out = input$bins + 1)
    # hier wird der Input von der ui-Seite
    # verwendet!

    # zeichne Histogramm mit gegebener Anzahl
    # von Bins:
    hist(x, breaks = bins)
  })
})



#
```

Einige Ressourcen

- Tutorial von RStudio: https://shiny.rstudio.com/tutorial/
- Tutorial von Dean Attali: https://deanattali.com/blog/building-shiny-apps-tutorial/
- Beispiele für Eingabe-Typen auf der `ui`-Seite: https://shiny.rstudio.com/gallery/widget-gallery.html

Beispiel: Wechselkurse des Schweizer Franken

- Wir wollen die App so modifizieren, dass sie Wechselkurse des Schweizer Franken zeigt
- das Package dataseries (Autor: Christoph Sax) macht Zeitreihendaten der Schweizer Institutionen direkt in R verfügbar

```
install.packages("dataseries")
```

- Suchfunktion und Generierung der Export-Kommandos auf http://www.dataseries.org/

Beispiel-Code

```r
install.packages("dataseries")
library(dataseries)

# Hole Wechselkurse des CHF mit Euro, US-Dollar, Pfund (Codes auf http://www.dataseries.org/)
wechselkurse <- dataseries::ds(c("FXR.M0.USD1", "FXR.M0.EUR1", "FXR.M0.GBP1"))
# Benenne Spalten neu:
colnames(wechselkurse) <- c("Zeit", "USD", "EUR", "GBP")

# Erzeuge Grafik:
plot(wechselkurse$Zeit, wechselkurse$EUR,
     xlab = "Zeit", ylab = "Franken pro Fremdwährung", # Achsenbeschriftungen
     type = "l") # zeichne Linie anstelle von Punkten
```

Überführung in Shiny-App

Modifikation des Templates

ui.R

```
library(shiny)
# UI für Wechselkurse-App
shinyUI(fluidPage(
  # Titel
  titlePanel("Wechselkurse"),
  # Sidebar:
  sidebarLayout(
    sidebarPanel(
      # Checkboxen für verschiedene Währungen
      selectInput("waehrung",
                  label = "Währung",
                  choices = list(
                    "US-Dollar" = "USD",
                    "Euro" = "EUR",
                    "Brit. Pfund" = "GBP"
                    ),
                  selected = 1)
    ),
    # Main Panel mit Plot:
    mainPanel(
      plotOutput("wechselkurse_plot")
    )
  )
))
```

server.R

```
library(shiny)
# Server-Code für Wechselkurse-App
shinyServer(function(input, output) {
  # Hole Wechselkurse des CHF mit Euro, Dollar, Pfund
  # (Codes zu finden auf http://www.dataseries.org/)
  wechselkurse <- dataseries::ds(
    c("FXR.MO.USD1", "FXR.MO.EUR1", "FXR.MO.GBP1"))
  # Benenne Achsen neu:
  colnames(wechselkurse) <-
    c("Zeit", "USD", "EUR", "GBP")
  output$wechselkurse_plot <- renderPlot({
    # Erzeuge Grafik:
    plot(wechselkurse$Zeit,
         wechselkurse[, input$waehrung],
         main = paste("Wechselkurs CHF /",
                      input$waehrung),
         xlab = "Zeit",
         ylab = "Franken pro Fremdwährung",
         type = "l") # zeichne Linie statt Punkte
  })
})
```

Apps veröffentlichen

- R Shiny kann auf den meisten Servern installiert werden.
- Services wie `shinyapps.io` (kommerziell) erlauben es, Apps online zu stellen, ohne selbst einen Server einzurichten.
- Tutorial:

Zurich R User Group

Interesse geweckt? Wir organisieren monatliche Veranstaltungen zu R und verwandten Themen in entspannter Atmosphäre.
Anmeldung über meetup:
https://www.meetup.com/Zurich-R-User-Group/

Nächste Ausgabe: 5.12.18 mit Martin Mächler (ETHZ), einem der Gründerväter von R.

Meetup

5
DEC

Wednesday, December 5, 2018

R programming with Martin Mächler

Hosted by muriel and 2 others
From Zurich R User Group
Public group