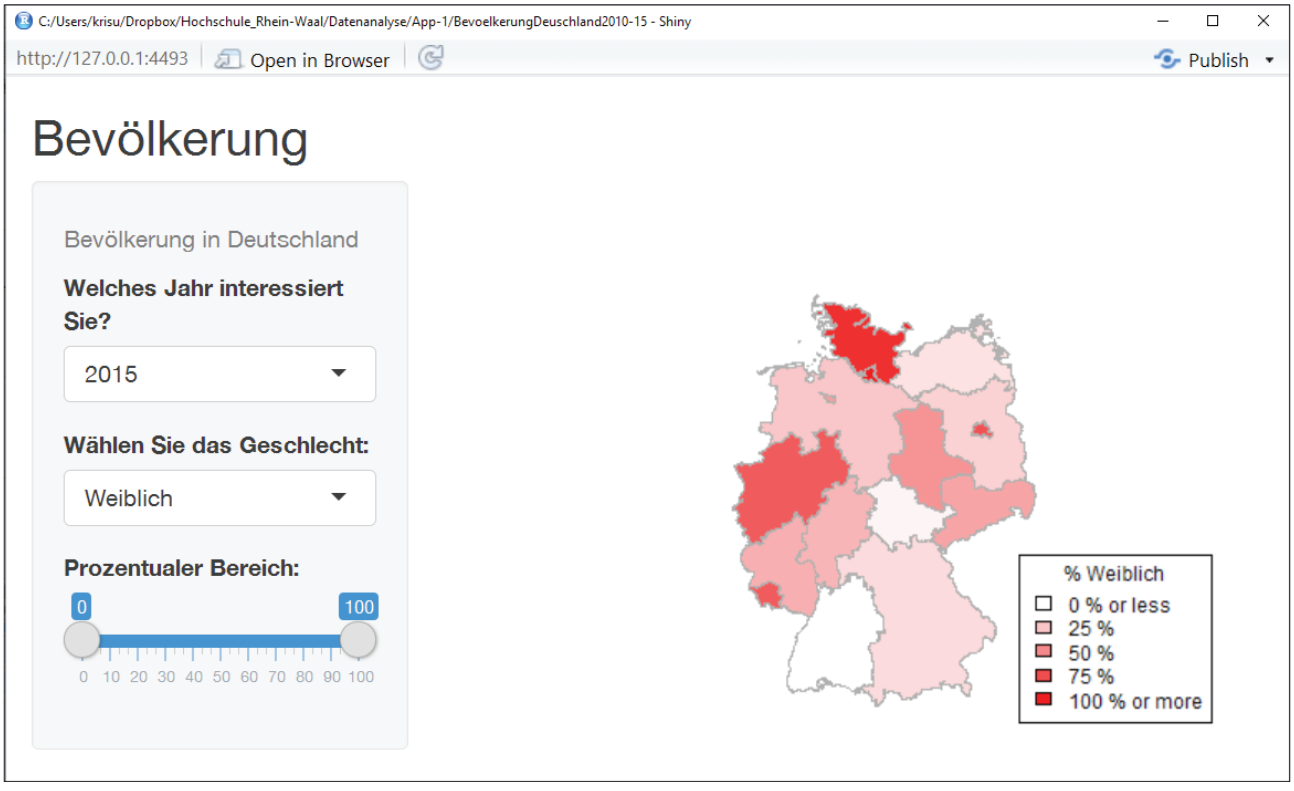
Hochschule Rhein-Waal
Fakultät Kommunikation und Umwelt
Modellierung, Simulation und angewandte Datenanalyse
Dozent: Prof. Dr. Zimmer

Mit Shiny visualisieren:

# Die Bevölkerungsverteilung in Deutschland




Eine Arbeit von:
Kristina-Susann Baudach
Matrikelnummer: 16882
E-Mail: Kristina-Susann.Baudach@hsrw.org

## Abstract

In der Arbeit „Mit Shiny visualisieren – Die Bevölkerungsverteilung in Deutschland“ geht es um die Verwendung des R-Packages Shiny von RStudio als „Web Application Framework“[01] für R, in das eine Einführung gegeben werden soll.

Es wird der Aufbau einer Multiple-File Shiny-App erläutert. Eine solche App besteht grundsätzlich aus den beiden R-Scripten ui.R und server.R Die ui.R ist dabei für die Darstellung der Benutzeroberfläche zuständig, während die server.R die Funktionen der App implementiert[02].

Darüber hinaus wird erläutert, wie weitere R-Scripte und R-Packages, sowie Daten aus Tabellen mit eingebunden werden können. Auch die Besonderheiten der Formatierung von Tabellen werden erklärt. Außerdem müssen diese Daten im Ordner „data“ der Shiny-App gespeichert werden[03]. Auch ist es möglich, Funktionen von dem bekannten Datenvisualisierungswerkzeug „D3“ mit Shiny zu implementieren.

Die Erstellung einer Shiny-App wird an dem Beispiel der Visualisierung der prozentualen Anteile an Männern und Frauen in den einzelnen Bundesländern der Bundesrepublik Deutschland, in den Jahren von 2010 bis 2015, anhand einer Karte noch einmal verdeutlicht.

Außerdem wird erläutert, dass mit Shiny fast alles visualisiert werden kann, aber durchaus die Berechnung von Kurven über Annäherungswerte gegebenenfalls berücksichtigt werden muss.

Ein weiterer Schwachpunkt ist, dass die Webveröffentlichung ohne Einschränkungen, gegebenenfalls kostenintensiv sein kann.

Alles in Allem ist Shiny jedoch ein sehr mächtiges Werkzeug, dass durchaus Potential haben könnte, in Zukunft zu DEM Werkzeug für Datenvisualisierung im Internet zu werden.


01 Vgl. Shiny by RStudio
02 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
03 Shiny by RStudio: LESSON 5 - Use R scripts and data

## Inhaltsverzeichnis

# 1 Einleitung

Shiny von RStudio ist ein „Web Application Framework“[01] für R. In Form eines R-Packages wird es mit dem Programm RStudio verwendet und kann mit sämtlichen R-Packages, sowie eigenen Packages kombiniert werden.

Auf der Webseite von Shiny gibt es eine gute Dokumentation über die Möglichkeiten, die man mit Shiny hat. Diese reichen von der Visualisierung einfacher Diagramme, bis hin zu der Implementierung eines Chatservers[02]. Darüber hinaus gibt es eine verhältnismäßig große Community, die sich gegenseitig bei der Entwicklung von Shiny-Apps unterstützt.

Da Shiny ein „Web Application Framework“[03] ist, ist es natürlich auch möglich eine Shiny-App als Webseite zu veröffentlichen und das ohne besondere Vorkenntnisse in HTML, CSS oder JavaScript[04]. Sinnvoll sind jedoch erste Kenntnisse in der Verwendung von R, da Shiny vollständig darauf basiert.

Die vorliegende Arbeit beschäftigt sich in erster Linie damit, wie Shiny-Apps erstellt werden können und erläutert dies am Beispiel der Visualisierung der männlichen und weiblichen Bevölkerungsverteilung in den einzelnen Bundesländern der Bundesrepublik Deutschland.

Die Visualisierung mit Shiny ist auch für Anfänger möglich, wenn man sich an die in den folgenden Kapiteln vorgestellte Struktur und Hinweise hält. Dann kann nicht nur dieses Beispiel, sondern alles, was mit R möglich ist, auf einer Webseite umgesetzt werden.

Im Übrigen wird diese Arbeit auch kurz auf die Möglichkeiten und Grenzen eingehen, die Shiny zur Zeit besitzt.

Zusammenfassend soll diese Arbeit Shiny von RStudio vorstellen und auch kritisch betrachten.

01 Vgl. Shiny by RStudio
02 Vgl. Shiny by RStudio: Gallery - Advanced Shiny
03 Vgl. Shiny by RStudio
04 Vgl. Shiny by RStudio

# 2 Die ersten Schritte mit Shiny

## 2.1 Die Installation von Shiny

Für die Verwendung von Shiny muss zu erst RStudio (hier Version 1.0.44 – © 2009-2016 RStudio, Inc.) installiert werden, bevor Shiny selbst installiert werden kann. RStudio beinhaltet Funktionen, die gezielt für Shiny erstellt worden sind[01].

Ist RStudio installiert, kann das Package von Shiny installiert und aktiviert werden. Dies geschieht durch folgende, nacheinander ausgeführte Eingaben in der Kommandozeile[02] von RStudio:

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

```
library(shiny)
```

Auf diese Weise kann Shiny, wie jedes R-Package installiert und aktiviert werden. Somit kann Shiny jetzt verwendet werden.

## 2.2 Eine neue Shiny-App erstellen

Da Shiny nun installiert ist, kann mit der Erstellung einer neuen Shiny-App begonnen werden. Dies ist in RStudio mit wenigen Klicks möglich.

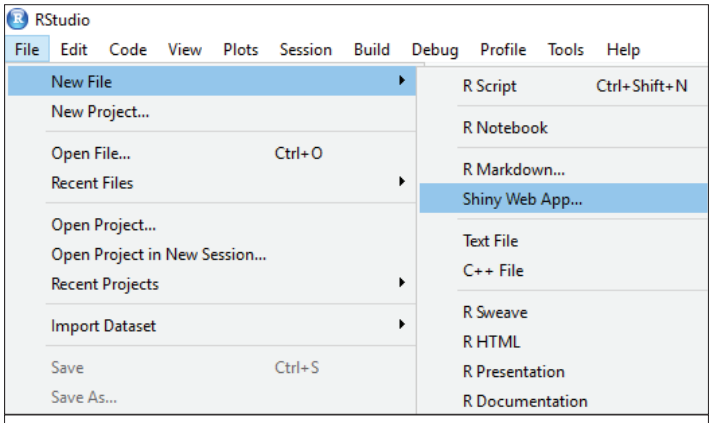



Abb. 1: Neue Shiny-App erstellen

Zu erst muss dazu unter „File“, im Menüpunkt „New File“, „Shiny Web App...“ ausgewählt werden (siehe Abb. 1). Darauf hin öffnet sich ein Fenster (siehe Abb. 2), in dem der Name und der Pfad, auf dem die App auf dem Rechner gespeichert werden soll, eingetragen wird. Dabei ist darauf zu achten, dass Sonderzeichen und deutsche Umlaute nicht verwendet werden können. Außerdem kann hier ausgewählt werden, ob die App aus einer Datei oder aus zwei Dateien bestehen soll. Hier wird empfohlen „Multiple File (ui.R/server.R)“ ausgewählt zu lassen, da dies der am häufigsten gewählte und in Tutorials am meisten verwendete Aufbau ist.

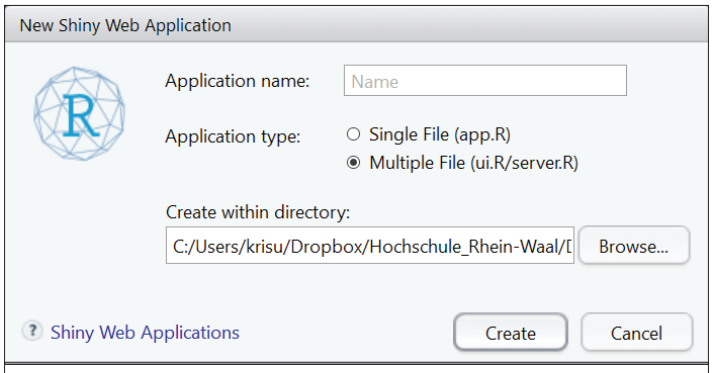



Abb. 2: Fenster zum Einrichten einer neue Shiny-App

Mit einem Klick auf „Create“ wird dann die App erzeugt. Sie beinhaltet dann bereits ein Beispiel, dass mit einem Klick auf „Run App“ in der oberen Leiste gestartet werden kann. Alternativ kann zum Starten der App auch Folgendes in die Kommandozeile eingegeben werden[03]:

```
runApp("my_app")
```

Mit diesen wenigen Einstellungen ist bereits eine erste lauffähige Shiny-App erstellt worden. Diese kann nun verändert und individuell angepasst werden. Dazu mehr in den folgenden Kapiteln.

01 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
02 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
03 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny

# 3 Der Aufbau einer Shiny-App

Eine Shiny-App kann entweder aus einem R-Script, namens app.R, oder aus zwei R-Scripten, ui.R und server.R, bestehen. Besteht die App nur aus einem R-Sript, wird dies als „Single-file“[01] bezeichnet. Dagegen wird ein aus zwei R-Scripten bestehender Aufbau als „Multiple file“ bezeichnet.

Die Single-File Shiny-App enthält sowohl den ui- als auch den server-Teil in einem einzigen Script. Der entscheidende Vorteil der Single-File Shiny-App ist, dass der gesamte Code so problemlos in die Kommandozeile kopierbar und ausführbar ist[02].Diese Form wird jedoch in keinem der Tutorial von Shiny verwendet und auch die Community verwendet bisher eher den Multiple-File-Aufbau.

Anfängern ist aus diesen Gründen empfohlen den Multiple-File-Aufbau zu nutzen und ui.R und server.R zu trennen.

## 3.1 Die ui.R

UI steht für „user-interface“, also Benutzeroberfläche. In der ui.R steht daher alles, was das Aussehen der App betrifft, nicht aber die Funktion der einzelnen Elemente[03].

Grunsätzlich ist die ui.R in Panel untergliedert und kann zum Beispiel, wie hier rechts abgebildet, aufgebaut werden[04]. Die hier gelb markierten Passagen enthalten dann den Code für das künftige Aussehen der App. Es ist möglich hier auch die aus HTML5 bekannten Tags zu verwenden. Welche das sind, kann in der Dokumentation auf der Webseite von Shiny nachgelesen werden. So steht zum Beispiel das p() für das HTML-Tag , dass einen Absatz für Text erstellt. Statt Text zu schreiben, muss die Syntax jedoch dann wie folgt aussehen[05]:

```
shinyUI(fluidPage(
    titlePanel("Titel der Seite"),
    sidebarLayout(
        sidebarPanel(
            # hier kann eigener Code stehen
        ),
        mainPanel(
            # hier kann eigener Code stehen
        )
    )
))
```

```
p("Text")
```

Auch ist die Implementierung von Buttons, Slidern, Texteingabefeldern und so weiter hier möglich. Diese werden „Control widgets“[06] genannt. Ein solches Widget benötigt die Angabe eines Namens für das Widget und ein so genanntes „Label“ als String-Eingabe[07]. Der String des Labels darf jedoch auch leer sein. Ein Action-Button könnte dann zum Beispiel so aussehen:

```
actionButton("action", label = "Action")
```

Die Standard-Widges von Shiny sind dabei Folgende[08]:

| Funktion | Widget |
|---|---|
| actionButton | Aktionsbutton |
| checkboxGroupInput | Gruppe von Check-Boxen |
| checkboxInput | Einzelne Check-Box |
| dateInput | Datumsauswahl über einen Kalender |

01 Chang, Winston: Single-file Shiny apps
02 Vgl. Chang, Winston: Single-file Shiny apps
03 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
04 Vgl. Shiny by RStudio: LESSON 2 - Build a user-interface
05 Vgl. Shiny by RStudio: LESSON 2 - Build a user-interface
06 Shiny by RStudio: LESSON 3 - Add control widgets
07 Vgl. Shiny by RStudio: LESSON 3 - Add control widgets
08 Vgl. Shiny by RStudio: LESSON 3 - Add control widgets

| Funktion | Widget |
|---|---|
| dateRangeInput | Kalenderpaar, um einen Zeitraum auszuwählen |
| fileInput | Datei-Upload |
| helpText | Hilfstext, der zu Input-Möglichkeiten hinzugefügt werden kann |
| numericInput | Zahleneingabefeld |
| radioButtons | Radiobutton |
| selectInput | Auswahlfeld |
| sliderInput | Slider |
| submitButton | Übertragungsbutton |
| textInput | Texteingabefeld |

Auf diese Art kombiniert Shiny R mit HTML5 und erstellt so die Benutzeroberfläche. Hinter diesem Aussehen, auch hinter den Widgets, steht allerdings noch keine richtige Funktion. Wenn man zum Beispiel den Action-Button verwendet, passiert bisher noch nichts. Diese Funktionen werden nur über die server.R implementiert.

### 3.2 Die server.R

Um nun die Funktion der Elemente der ui.R zu implementieren, wird die server.R verwendet[09]. Die server.R arbeitet daher mit den Daten, die die ui.R sammelt, und macht beispielsweise Berechnungen, rendert grafische Elemente und so weiter.

Um die Daten in die server.R zu übermitteln, muss in der ui.R eine der Output-Funktionen verwendet werden[10]. Dazu gibt es folgende Output-Typen:

- htmlOutput
- imageOutput
- plotOutput
- tableOutput
- textOutput
- uiOutput
- verbatimTextOutput

So kann der Output zum Beispiel folgendermaßen definiert werden:

```
plotOutput("Output-Name")
```

Nun kann die server.R die Daten auslesen, wenn in ihr, mit der zugehörigen Funktion, auf diesen Output verwiesen wird. Für den obigen Output wäre dies zum Beispiel die Funktion:

```
output$Output-Name <- renderPlot({ Code })
```

Dabei ist eine Struktur, wie hier rechts zu sehen, grundsätzlich einzuhalten[11]. In dieser Struktur gibt es drei Bereiche, in denen Code eingefügt werden kann[12]. Im ersten Bereich, vor der Funktion „shinyServer“, wird der Code nur beim Starten der App ausgeführt. Alles innerhalb der Funktion „shinyServer“ wird jedes Mal dann ausgeführt, wenn ein Nutzer die App

```
library(shiny)
# Ausführung einmalig, wenn die App gestartet wird.
shinyServer(function(input, output) {
    # Ausführung, wenn ein Nutzer die App besucht.
    output$Output-Name <- renderPlot({
        # Ausführung, wenn die Oberfläche benutzt wird.
    })
})
```

09 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
10 Vgl. Shiny by RStudio: LESSON 4 - Display reactive output
11 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
12 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

verwendet, während der dritte Bereich immer ausgeführt wird, wenn der Nutzer etwas an der Oberfläche verändert, zum Beispiel durch Nutzung der Buttons und Eingabefelder.

Auf diese Weise kann dann in der server.R die gesamte Logik der Shiny-App eingetragen werden, damit die Benutzeroberfläche auch funktionsfähig ist.

### 3.3 Die Einbindung weiterer R-Scripte und R-Packages

Es ist auch möglich, weitere R-Scripte und R-Packages in die Shiny-App einzufügen. Die Einbindung weiterer R-Scripte kann zum Beispiel sinnvoll sein, wenn man einen Teil des Codes ausgliedern möchte, um den Code beispielsweise übersichtlicher zu gestalten oder externe Funktionen aufzurufen. Die Möglichkeit darüber hinaus R-Packages einzubinden erweitert den Funktionsumfang von Shiny, da grundsätzlich jedes R-Package genutzt und somit alle Möglichkeit, die R bietet, genutzt werden können.

Um ein weiteres R-Script einzubinden, wird folgende Zeile[13] in den Teil von server.R, der nur beim Start ausgeführt wird, eingebunden, da das Script nur anfangs einmalig geladen werden muss um dauerhaft darauf zuzugreifen:

```
source("scriptname.R")
```

Wichtig ist dabei, dass die Datei im selben Ordner, wie server.R und ui.R liegen und nicht in einen Unterordner verschoben werden[14]. Ansonsten müsste der entsprechende Pfad angegeben werden.

Nun ist ein weiteres R-Script eingebunden und die darin enthaltenen Funktionen können abgerufen und ausgeführt werden. Auf ähnliche Weise werden R-Packages eingefügt, sofern sie vorher über die Package-Verwaltung installiert worden sind. Dann kann ebenfalls in dem Teil von server.R, der nur beim Start ausgeführt wird, das Package eingebunden werden. Dies geschieht über folgenden Befehl:

```
library(Package-Name)
```

Nun können die eingebundenen R-Scripte und Packages in Shiny verwendet werden.

### 3.4 Die Verwaltung und Einbindung von Daten

Da Shiny ein R-Package zur Visualisierung von Daten ist, ist es natürlich auch möglich, externe Daten einzufügen. So können zum Beispiel Excel- und CSV-Dateien, sowie das R eigene Format RDS ausgelesen, aber auch Bilder verwendet werden.

Bilder werden dazu im Unterordner „www“[15] und Daten-Dateien im Unterordner „data“[16] abgelegt. Beide Ordner werden beim Erstellen der App nicht automatisch angelegt, sondern müssen manuell in der Ordnerstruktur erzeugt werden.

Um Bilder anzuzeigen, werden diese in der ui.R mit dem img-Tag beispielsweise so eingefügt[17]:

```
img(src = "bildname.png", height = 80, width = 80)
```

Auf Datendateien wird in der server.R zugegriffen, indem sie in den Teil, der nur beim Start ausgeführt wird, einer Variable zugewiesen werden, um dann jeder Zeit abrufbar zu sein, zum Beispiel[18]

```
variablenname <- readRDS("data/Dateiname.rds")
```

oder

```
variablenname <- read.csv("data/Dateiname.csv", header = TRUE, sep = ";")
```

13 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
14 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
15 Shiny by RStudio: LESSON 2 - Build a user-interface
16 Shiny by RStudio: LESSON 5 - Use R scripts and data
17 Vgl. Shiny by RStudio: LESSON 2 - Build a user-interface
18 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

Beim späteren Abrufen der Daten ist dann zu bedenken, dass die Daten nur spaltenweise ausgelesen werden können und nicht auf einzelne Zeilen zugegriffen werden kann. Dabei wird der jeweilige Eintrag in der ersten Zeile, also der Spaltenname, zum Abrufen der jeweiligen Spalte verwendet. Ein entsprechender Aufruf beziehungsweise eine Verwendung der Daten erfolgt dann über folgenden Befehl[19]:

```
Variablenname$Spaltenname
```

Verwendet man Daten aus Excel- oder CSV-Dateien, ist außerdem darauf zu achten, dass in Dezimalzahlen der Punkt als Dezimaltrennzeichen verwendet wird. Zum Beispiel darf es nicht 50,5% sondern 50.5% heißen. Dazu muss die Datei auf die englische Schreibweise umgestellt werden. Eine Anleitung, wie dies in Excel funktioniert, ist unter http://www.pctipp.ch/tipps-tricks/kummerkasten/office/artikel/windows-excel-punkt-statt-komma-als-dezimalzeichen-63366/ (Stand: 08.01.2017) zu finden.

Nun können Bilder und Daten aus verschiedenen Formaten eingebunden und verwendet werden.

19 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

# 4 Ein Beispiel einer Multiple-File Shiny-App

## 4.1 Die Funktion der fertigen App

Das vorliegende Beispiel einer Multiple-File Shiny-App namens „BevoelkerungDeutschland2010-15“ ist abgeleitet vom Shiny-Tutorial „Lesson 5 - Use R scripts and data“[01] und soll die prozentualen Anteile an Männern und Frauen in den einzelnen Bundesländern der Bundesrepublik Deutschland in den verschiedenen Jahren von 2010 bis 2015 visualisieren.

In der Beispiel-App kann der Nutzer auswählen, zu welchem Jahr er die Daten visualisiert bekommt und ob er die Anteile der Männer oder der Frauen einsehen möchte. Darüber hinaus kann er den prozentualen Bereich eingrenzen, den er in der Karte angezeigt bekommen möchte.

Die App besteht aus drei R-Scripten, der ui.R, der server.R und der helpers.R und nutzt Daten vom Statistischen Bundesamt (https://www-genesis.destatis.de/) über die Einwohner-Zahlen von Deutschland. Sie kann theoretisch sowohl vom normalen Rechner, als auch von Mobilgeräten verwendet werden (siehe Abb. 3 und 4), sofern sie online gestellt wird.

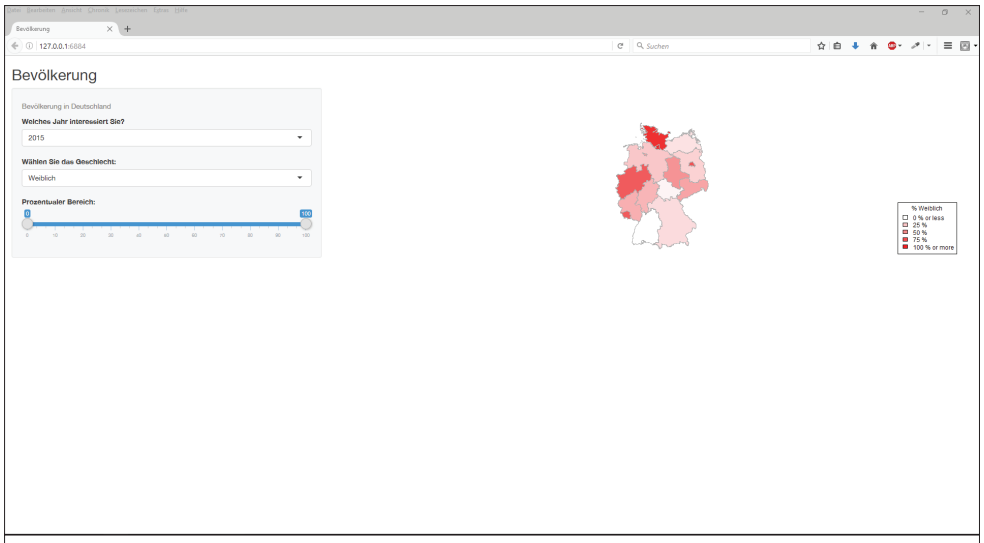



Abb. 3: Browser-Ansicht der Shiny-App

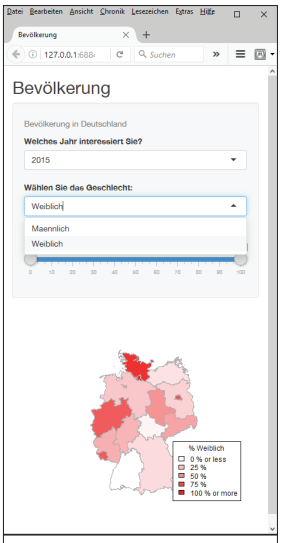



Abb. 4: Mobile Ansicht der Shiny-App

Diese Form der Visualisierung ist auch leicht für andere prozentuale Daten zu den Bundesländern adaptierbar und bietet daher ein gutes Beispiel für die Visualisierung mit Shiny.

## 4.2 Die ui.R

Auch in diesem Beispiel werden in der ui.R die Elemente und ihr Erscheinungsbild festgelegt. Dazu besteht diese App aus drei Paneln, dem „titlePanel“ für die Überschrift, dem „sidebarPanel“ für die Nutzereinstellungen und dem „mainPanel“ zum Plotten der Karte mit den Ergebnissen und der Legende[02].

Der „mainPanel“ steht, wie der „sidebarPanel“, im Bereich des „sidebarLayouts“ und beinhaltet lediglich die Funktion „plotOutput("map")“, um anzugeben, dass in diesem Bereich die Karte mit der Auswertung der Daten stehen soll. Was genau dort stehen soll, gibt die Funktion jedoch nicht an, da sie nur das Ergebnis aus der server.R anfordert[03].

01 Shiny by RStudio: LESSON 5 - Use R scripts and data
02 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
03 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

Der „sidebarPanel“ besteht dagegen aus vier Elementen, dem „helpText“, zwei „selectInput“-Methoden und einer „sliderInput“-Methode. Der „helpText“ ist hier als eine Art Unterüberschrift verwendet, kann theoretisch aber auch normalen Hilfstext für den Nutzer enthalten[04].

Der erste „selectInput“ definiert die Auswahlmöglichkeiten für das Jahr, zu dem sich der Nutzer die Daten anzeigen lassen möchte. Beschriftet wird diese Auswahl mit dem „Label“ „Welches Jahr interessiert Sie?“ und der Nutzer erhält insgesamt fünf Auswahlmöglichkeiten („choices“) für die Jahre 2010 bis 2015. Vorausgewält („selected“) beim Starten der App ist dann das Jahr 2015. Die Bennenug dieses „selectInput“-Elements mit „jahr“ wird dagegen vom Nutzer nicht gesehen. Dies ist aber wichtig, um später die Auswahlwerte in der server.R verwenden zu können.

Auch der zweite „selectInput“, namens „geschlecht“, definiert Auswahlmöglichkeiten für den Nutzer, die mit „Wählen Sie das Geschlecht“ betitelt werden. Hier kann in der App entschieden werden, ob die prozentuale Verteilung der Männer („Maennlich“), oder die der Frauen („Weiblich“) angezeigt werden soll. Wichtig ist, dass die Auswahlmöglichkeiten in ihrer Beschriftung keine deutschen Umlaute oder Sonderzeichen enthalten, da es sonst zu Fehlermeldungen in der App kommt.

Der „sliderInput“, der mit „bereich“ benannt ist, erzeugt eine Auswahlleiste mit den Werten von 0 bis 100. Diese Auswahl kann über zwei Slidern, die mit den Werten („value“) 0 und 100 starten, ausgewählt werden[05]. Diese Bereichsauswahl besitzt den Titel „Prozentualer Bereich“.

08

```
# ui.R

# Code vgl. http://shiny.rstudio.com/tutorial/lesson5/

shinyUI(fluidPage(

    titlePanel(„Bevölkerung“),

    sidebarLayout(

        sidebarPanel(
            helpText(„Bevölkerung in Deutschland“),

            selectInput(„jahr“,
                label = „Welches Jahr interessiert Sie?“,
                choices = c(„2010“, „2011“, „2012“,
                            „2013“, „2014“, „2015“),
                selected = „2015“),

            selectInput(„geschlecht“,
                label = „Wählen Sie das Geschlecht:“,
                choices = c(„Maennlich“, „Weiblich“),
                selected = „Weiblich“),

            sliderInput(„bereich“,
                label = „Prozentualer Bereich:“,
                min = 0, max = 100,
                value = c(0, 100))
        ),

        mainPanel(
            plotOutput(„map“)
        )

    )
))
```

Zusammenfassend erzeugt die ui.R mit ihrem hier rechts abgebildeten Code also einen Haupttitel, zwei Auswahlfelder für Datenkategorien und ein Auswahlfeld für einen Bereich. Außerdem wird die Karte mit den Daten entsprechend der getroffenen Auswahl visualisiert.

04 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
05 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

### 4.3 Die server.R

Die server.R, deren Code hier rechts abgebildet ist, verarbeitet nun die ausgewählten Daten der ui.R und leitet diese Daten an die helpers.R, die die Karte und die Legende der Karte erzeugt, weiter. Diese erzeugte Karte liefert wird dann von der server.R wieder an die ui.R zurück geliefert, damit diese die Karte anzeigen kann.

Damit die server.R dies tuen kann, müssen zu erst die Bibliotheken („library“) „sp“ und „RColorBrewer“ geladen werden, die später in der helpers.R Verwendung finden. Auch die Daten vom Statistischen Bundesamt im CSV-Format müssen Variablen zugewiesen werden, um später verwendet werden zu können[06]. Da es für jedes Jahr (2010-2015) jeweils eine CSV-Datei gibt, werden also sechs Variablen erzeugt und belegt. Außerdem wird hier auch die helpers.R eingebunden, um später aufgerufen werden zu können[07]. Dies alles geschieht am Anfang der server.R und wird daher auch nur ausgeführt, wenn die App gestartet wird, da dies nach dem Laden dauerhaft verfügbar ist, ohne jedes Mal neu erzeugt werden zu müssen.

Die Auswertung der Benutzeroberfläche erfolgt nun in dem Bereich, der jedes Mal lädt, wenn die Benutzeroberfläche verwendet wird. Dazu werden im Output „map“, den die ui.R zum Anzeigen der Karte anfordert, die ausgewählten Einstellungen ausgelesen, Variablen je nach Auswahl mittels „switch-case“-Anweisungen belegt und über diese dann an die helpers.R übergeben, um der ui.R die Karte entsprechend zurückgeben zu können[08].

```
# server.R
# Code vgl. http://shiny.rstudio.com/tutorial/lesson5/
library(sp)
library(RColorBrewer)
ger2010 <- read.csv(„data/Deutschland_2010.csv“,
                    header = TRUE, sep = „;“)
ger2011 <- read.csv(„data/Deutschland_2011.csv“,
                    header = TRUE, sep = „;“)
ger2012 <- read.csv(„data/Deutschland_2012.csv“,
                    header = TRUE, sep = „;“)
ger2013 <- read.csv(„data/Deutschland_2013.csv“,
                    header = TRUE, sep = „;“)
ger2014 <- read.csv(„data/Deutschland_2014.csv“,
                    header = TRUE, sep = „;“)
ger2015 <- read.csv(„data/Deutschland_2015.csv“,
                    header = TRUE, sep = „;“)
source(„helpers.R“)
shinyServer(
  function(input, output) {
    output$map <- renderPlot({
      jahrWahl <- switch(input$jahr,
                „2010“ = ger2010, „2011“ = ger2011,
                „2012“ = ger2012, „2013“ = ger2013,
                „2014“ = ger2014, „2015“ = ger2015)
      dataNew <- switch(input$geschlecht,
                „Maennlich“ = jahrWahl$Prozent.maennlich,
                „Weiblich“ = jahrWahl$Prozent.weiblich)
      color <- switch(input$geschlecht,
                „Maennlich“ = „blue“,
                „Weiblich“ = „red“)
      legend <- switch(input$geschlecht,
                „Maennlich“ = „% Männlich“,
                „Weiblich“ = „% Weiblich“)
      percent_map(var = dataNew,
                color = color, legend.title = legend,
                max = input$bereich[2],
                min = input$bereich[1])
    })}
)
```

Die erste Variable, die erzeugt und über eine „switch-case“-Anweisungen belegt wird, heißt „jahrWahl“. In ihr wird anhand der ausgewählten Jahreszahl die Datei mit den zugehörigen Daten gespeichert.

06 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
07 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
08 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

Die Auswahl des Geschlechts beeinflusst dagegen drei Variablen-Inhalte, den von „dataNew“, „color“ und „legend“. In „dataNew wird nämlich festgelegt, welche Spalte die Daten enthält. Diese Spalte wird durch die Angabe der Variable, die die Datei enthält (hier: „jahrWahl“), und des, mit einem $-Zeichen angehängten, Spaltennamens definiert[09]. Dagegen entscheidet „color“ später darüber, ob die Karte in Rot- oder Blautönen angezeigt wird. „legend“ beinhaltet dagegen die spätere Bezeichnung der Legende der Karte.

Anschließend werden diese Variablen im Aufruf der Funktion „percent_map“ benötigt, die in der helpers.R definiert ist. Darüber hinaus benötigt diese Funktion noch die Werte der beiden Slider von der Bereichsauswahl, um dann die Karte zu erzeugen[10].

Auf diese Art liest die server.R die ausgewählten Werte der Benutzeroberfläche aus, wertet sie aus und gibt am Ende, unter Verwendung der „percent_map“-Funktion der helpers.R, die fertige Karte für die Benutzeroberfläche zurück, damit die ui.R sie anzeigen kann.

### 4.4 Die helpers.R

Die Aufgabe der helpers.R ist es nun, anhand der Daten eine Karte mit entsprechender Färbung, je nach prozentualem Anteil an Männern beziehungsweise Frauen, zu generieren, indem sie die Funktion „percent_map“ implementiert.

Die Funktion „percent_map“ benötigt sechs Variablen, wenn sie aufgerufen wird. Die Variable „var“ beinhaltet die ausgewählten Daten, „color“ enthält den String für den Farbton, „legend.title“ definiert den Titel der Kartenlegende und „min“ und „max“ sind für die minimalen und maximalen Prozente, die dargestellt werden sollen, zuständig. All diese Variablen werden beim Aufruf der Funktion in der server.R durch die Auswertung der jeweiligen Auswahl in der Benutzeroberfläche[11] definiert.

In der Funktion „percent_map“ wird als erstes eine Variable namens „shades“, mit Hilfe von Elementen der Bibliothek „RColorBrewer“, definiert. Die Variable beinhaltet dann Farbwerte von weiß bis zur Volltonfarbe der Farbe, die die Variable „color“ enthält. Diese werden abgestuft in hundert Farbstufen[12].

```
# helpers.R
# Code vgl. http://shiny.rstudio.com/tutorial/lesson5/

percent_map <- function(var, color, legend.title, min = 0, max = 100) {
    shades <- colorRampPalette(c(„white“, color))(100)
    var <- pmax(var, min)
    var <- pmin(var, max)
    percents <- as.integer(cut(var, 100,
        include.lowest = TRUE, ordered = TRUE))
    fills <- shades[percents]
    # vgl Die nächsten beiden Zeilen: https://www.students.
      ncl.ac.uk/keith.newman/r/maps-in-r-using-gadm
    gadm <- readRDS(„data/DEU_adm1.rds“)
        # RDS-Datei von http://www.gadm.org/download
    plot(gadm, col = fills, border = ‚darkgrey‘)
    inc <- (max - min) / 4
    legend.text <- c(paste0(min, „ % or less“),
        paste0(min + inc, „ %“),
        paste0(min + 2 * inc, „ %“),
        paste0(min + 3 * inc, „ %“),
        paste0(max, „ % or more“))
    legend(„bottomright“,
        legend = legend.text,
        fill = shades[c(1, 25, 50, 75, 100)],
        title = legend.title)
}
```

09 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
10 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
11 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
12 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

Im Anschluss daran, werden die Daten so beschränkt, dass sie im ausgewählten Prozentbereich, also zwischen „min“ und „max“ stehen. Das heißt, alles was darunter beziehungsweise darüber liegt, erhält die Werte von „min“ und „max. Danach werden alle Daten-Werte in Integer umgewandelt, um farblich über die hundert Farbstufen von „shades“ in der Karte visualisiert werden zu können. Dazu werden die passenden Farbwerte der Reihe nach in die Variable „fills“ eingeladen[13].

Als nächstes wird zum Plotten der Karte die Bibliothek „sp“ und die Karte „DEU_adm1.rds“ von GADM (http://www.gadm.org/download) benötigt. Hier müssen jedoch die Lizenzbedingungen beachtet werden, da die Datei von GADM, die das Kartenmaterial enthält, nicht für den kommerziellen Gebrauch, sondern nur für den privaten Gebrauch und akademische Veröffentlichungen genutzt werden dürfen[14]. Die Karte wird dann aus dem Kartenmaterial mit der Füllfarbe, die die Variable „fills“ vorgibt, und einer dunkelgrauen („darkgrey“) Kontur erzeugt[15].

Zum Schluss wird noch die Legende zur Karte mit fünf Punkten erzeugt, die die Farben zu 1%, 25%, 50%, 75% und 100% darstellen, um eine Reverenz für die in der Karte dargestellten Daten zu geben[16].

Auf diese Weise wird die Karte mit der Legende von der helpers.R erzeugt und über die server-.R an die ui.R weitergegeben, um in der Benutzeroberfläche angezeigt werden zu können.

### 4.5 Die Verwaltung der abzubildenden Daten und des Kartenmaterials

Um Daten mit Shiny zu visualisieren, müssen dies richtig formatiert werden, da sie sonst nicht richtig ausgelesen werden können. Außerdem benötigt Shiny zum Darstellen der Karte mit den Bundesländern von Deutschland ebenfalls entsprechendes Material.

Die darzustellenden Daten werden für dieses Beispiel als Excel-Tabelle von der Webseite des Statistischen Bundesamt bezogen (siehe Abb. 5).

In dieser Tabelle stehen noch die Daten für alle Jahre von 2010 bis einschließlich 2015 in einer Tabelle. Außerdem stehen in den ersten fünf Zeilen die Spaltenbeschriftung und ein Titel. Auch sind noch keine Prozentsätze in der Tabelle angeben, sondern nur die Gesamtanzahl und die Anzahl der jeweiligen Geschlechter.

Damit Shiny die Daten verwenden kann, müssen zu erst die ersten fünf Zeilen so geändert werden, dass sie, auf eine Zeil ereduziert, nur noch eine Spaltenbeschriftung enthalten, da über diese Beschriftung die Daten der jeweiligen Spalte ausgelesen werden. Wichtig ist dabei, dass diese Spaltenbeschriftung keine deutschen Umlaute oder Sonderzeichen enthalten dürfen.

**Bevölkerung: Bundesländer, Stichtag, Geschlecht**

Fortschreibung des Bevölkerungsstandes
Bevölkerungsstand (Anzahl)

| Bundesländer | Geschlecht | | |
|---|---|---|---|
| | männlich | weiblich | Insgesamt |
| **31.12.2010** | | | |
| Baden-Württemberg | 5296249 | 5457631 | 10753880 |
| Bayern | 6158439 | 6380257 | 12538696 |
| Berlin | 1695438 | 1765287 | 3460725 |
| Brandenburg | 1240553 | 1262720 | 2503273 |
| Bremen | 321940 | 338766 | 660706 |
| Hamburg | 873712 | 912736 | 1786448 |
| Hessen | 2976527 | 3090494 | 6067021 |
| Mecklenburg-Vorpommern | 813283 | 829044 | 1642327 |
| Niedersachsen | 3893761 | 4024532 | 7918293 |
| Nordrhein-Westfalen | 8711858 | 9133296 | 17845154 |
| Rheinland-Pfalz | 1967106 | 2036639 | 4003745 |
| Saarland | 495206 | 522361 | 1017567 |
| Sachsen | 2031630 | 2117847 | 4149477 |
| Sachsen-Anhalt | 1144118 | 1190888 | 2335006 |
| Schleswig-Holstein | 1388912 | 1445347 | 2834259 |
| Thüringen | 1103693 | 1131332 | 2235025 |
| **31.12.2011** | | | |
| Baden-Württemberg | 5151354 | 5361087 | 10512441 |
| Bayern | 6093411 | 6349961 | 12443372 |
| Berlin | 1617768 | 1708234 | 3326002 |
| Brandenburg | 1207189 | 1245991 | 2453180 |
| Bremen | 317060 | 335122 | 652182 |
| Hamburg | 832064 | 886123 | 1718187 |

Abb. 5: Ansicht der ursprünglichen Tabelle vom Statistischen Bundesamt

13 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
14 Vgl. Hijmanns, Robert
15 Vgl. Newman, Keith: Plotting Country maps in R using GADM data
16 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data

Als nächstes muss die Tabelle um zwei Spalten erweitert werden, in denen die Prozentsätze jeweils für den Anteil der Männer und den der Frauen berechnet werden. Dies ist nötig, da in der Karte später die prozentualen Anteile und nicht die realen Werte visualisiert werden sollen. Die Prozentsätze werden nun jedoch noch mit einem Komma als Dezimaltrennzeichen angezeigt, sofern das Excel-Programm beziehungsweise das Betriebssystem des Computer auf deutsch eingestellt ist. Dieses Dezimaltrennzeichen kann Shiny nicht verarbeiten, denn das Dezimaltrennzeichen Punkt wird benötigt. Dazu muss man entweder das Excel-Programm oder das Betriebssystem des Computer auf die englische Sprache umstellen. (Siehe dazu http://www.pctipp.ch/tipps-tricks/kummerkasten/office/artikel/windows-excel-punkt-statt-komma-als-dezimalzeichen-63366/.)

Sind diese Änderungen an der Datei vorgenommen worden, müssen nun die Daten der einzelnen Jahre in voneinander getrennte Dateien kopiert und diese als CSV-Datei mit einem Semikolon als Trennzeichen im Ordner „data“ des Dateisystems von Shiny gespeichert werden, damit alle Jahre einzeln abrufbar sind. Jede diese Dateien sollte dann aussehen, wie in Abb. 6 zu sehen.

| | | Insgesamt | maennlich | weiblich | Prozent-maennlich | Prozent-weiblich |
|---|---|---|---|---|---|---|
| 31.12.2010 | Baden-Wuerttemberg | 10753880 | 5296249 | 5457631 | 49.25 | 50.75 |
| 31.12.2010 | Bayern | 12538696 | 6158439 | 6380257 | 49.12 | 50.88 |
| 31.12.2010 | Berlin | 3460725 | 1695438 | 1765287 | 48.99 | 51.01 |
| 31.12.2010 | Brandenburg | 2503273 | 1240553 | 1262720 | 49.56 | 50.44 |
| 31.12.2010 | Bremen | 660706 | 321940 | 338766 | 48.73 | 51.27 |
| 31.12.2010 | Hamburg | 1786448 | 873712 | 912736 | 48.91 | 51.09 |
| 31.12.2010 | Hessen | 6067021 | 2976527 | 3090494 | 49.06 | 50.94 |
| 31.12.2010 | Mecklenburg-Vorpommern | 1642327 | 813283 | 829044 | 49.52 | 50.48 |
| 31.12.2010 | Niedersachsen | 7918293 | 3893761 | 4024532 | 49.17 | 50.83 |
| 31.12.2010 | Nordrhein-Westfalen | 17845154 | 8711858 | 9133296 | 48.82 | 51.18 |
| 31.12.2010 | Rheinland-Pfalz | 4003745 | 1967106 | 2036639 | 49.13 | 50.87 |
| 31.12.2010 | Saarland | 1017567 | 495206 | 522361 | 48.67 | 51.33 |
| 31.12.2010 | Sachsen | 4149477 | 2031630 | 2117847 | 48.96 | 51.04 |
| 31.12.2010 | Sachsen-Anhalt | 2335006 | 1144118 | 1190888 | 49 | 51 |
| 31.12.2010 | Schleswig-Holstein | 2834259 | 1388912 | 1445347 | 49 | 51 |
| 31.12.2010 | Thueringen | 2235025 | 1103693 | 1131332 | 49.38 | 50.62 |

Abb. 6: Ansicht einer angepassten Tabelle für 2010

Es ist wichtig, dass die Reihenfolge der Bundesländer nicht verändert wird[17], da Shiny die einzelnen Bundesländer in dieser Reihenfolge plottet und ansonsten die Werte in den falschen Bundesländern visualisiert würden.

Das Kartenmaterial für Deutschland kann, wie in Kapitel 4.3 beschrieben, von der Webseite von GDAM heruntergeladen werden. Um die Bundesländer darzustellen wird „Level 1“ von den zur Verfügung stehenden Downloads benötigt[18]. Die heruntergelade Datei, die in diesem Beispiel „DEU_adm1.rds“ heißt, wird nun ebenfalls im Ordner „data“ des Dateisystems von Shiny abgelegt und kann dann von Shiny verwendet werden.

Zusammenfassend ist es also wichtig vorliegende Daten gegebenenfalls ins richtige Format zu bringen und alle im Ordner „data“ des Dateisystems von Shiny zu speichern, damit die Shiny-App darauf zugreifen kann.

17 Vgl. Shiny by RStudio: LESSON 5 - Use R scripts and data
18 Vgl. Hijmanns, Robert

# 5 Die Möglichkeiten und Grenzen von Shiny

Shiny von RStudio ist ein „Web Application Framework“[01] für R. In Form eines R-Package bietet es viele Möglichkeit, von der Datenvisualisierung bis hin zu der Implemtierung eines Chatservers[02].

Auf den ersten Blick hat Shiny keinerlei Grenzen und es ist alles möglich. Shiny kann grundsätzlich alles, was mit R beziehungsweise mit anderen R-Packages, sofern diese eingebunden werden, möglich ist. Darüber hinaus kann Shiny HTML-Elemente verwenden und ist mit eigenen Packages problemlos erweiterbar[03]. So können selbst JavaScript und CSS oder auch Google Analytics in Shiny-Apps verwendet werden.

Auch bei einem Blick in die „Gallery“[04] und in die „Shiny User Showcase“[05] kann man zahlreiche Beispiele zu unterschiedlichsten Themen- und Visualisierungsmöglickeiten finden. Dies zeigt, dass Shiny scheinbar lediglich die Grenzen des Verstandes desjenigen, der die Shiny-App implementiert, gesetzt sind, denn Shiny wird immer weiter entwickelt und in der neusten Version wird aktuell die Anbindung ganzer Datenbanken getestet und entwickelt[06], um auch dies künftig mit Shiny realisieren zu können.

Shiny wird damit beworben, dass es ohne besondere Vorkenntnisse in HTML, CSS oder JavaScript[07] verwendet werden kann. Sinnvoll sind jedoch erste Kenntnisse in der Verwendung von R, da Shiny vollständig darauf basiert. Richtige Programmierkentnisse, wie sie zum Beispiel bei dem vergleichbaren Werkzeug „D3“ benötigt werden, werden hier nicht gebraucht, weswegen es sich auch besonders für Laien und Anfänger eignet. Dennoch ist es möglich in Shiny D3-Funktionen, wie beispielsweise „MetricGraphics“[08], „networkD3“[09], „threejs“[10] oder „d3heatmap“[11] einzubinden.

Es sollte jedoch bedacht werden, dass auch Shiny, zum Beispiel bei der Anzeige von Kurven, genau wie R, nur Näherungswerte anzeigt. Es könnte theoretisch also leichte, minimale Abweichungen von den tatsächlichen Werten geben. Man sollte daher immer daran denken, dass theoretisch kleine Ungenauigkeiten bestehen können. Vermutlich sind diese jedoch mit dem menschlichen Auge nicht zu erkennen. Deshalb ist es für die Visualisierung trotzdem sehr gut einsetzbar. Dies gilt auch für die Visualisierung von Prozentsätzen über Farbtonstufen. Werte die sehr eng beieinander liegen, haben so einen geringeren Farbunterschied, dass der Mensch diesen nicht wahrnehmen kann. Auch hier ist es daher wichtig, sich zu überlegen, wie exakt die Wertunterschiede gezeigt werden sollen und ob es für eine sehr exakte Wertdarstellung die richtige Visualisierungsform ist, oder eine andere Form besser geeignet wäre.

Eine andere Schwierigkeit bei Shiny-Apps stellt die Veröffentlichung dieser da. Dazu gibt es grundsätzlich zwei Möglichkeiten. Man kann entweder aus dem Ordner der App einen Zip-Ordner erstellen und diesen anderen zum Download bereitstellen. Dies hat jedoch den Nachteil, dass die potentiellen Nutzer ebenfalls R auf ihrem Computer installiert haben müssen[12]. Alternativ kann man sie jedoch auch als Webseite veröffentlichen, so dass sie von jedem vollständig gerendert über den

01 Vgl. Shiny by RStudio
02 Vgl. Shiny by RStudio: Gallery - Advanced Shiny
03 Vgl. Shiny by RStudio: Articles - Extend Shiny
04 Shiny by RStudio: Gallery - Advanced Shiny
05 RStudio: Shiny User Showcase
06 Vgl. Shiny by RStudio: Articles - Databases
07 Vgl. Shiny by RStudio
08 Shiny by RStudio: Articles - Extend Shiny
09 Shiny by RStudio: Articles - Extend Shiny
10 Shiny by RStudio: Articles - Extend Shiny
11 Shiny by RStudio: Articles - Extend Shiny
12 Vgl. Shiny by RStudio: LESSON 7 - Share your apps

Browser verwendet werden kann. Dazu muss man jedoch entweder den Hosting Dienst „Shinyapp.io“ von RStudio oder den „Shiny Server“ beziehungsweise „Shiny Server Pro“ von RStudio verwenden[13].

Die Schwierigkeit bei der Veröffentlichung der Shiny-App als Webseite besteht genau in der Verwendung dieser Angebote von Shiny, denn die kostenlosen Versionen haben nur eingeschränkte Funktionen. So bietet der kostenlose „Shiny Server“ zum Beisiel keine passwortgeschützten Zugänge für den Datei-Download und auch keine SSL-Unterstützung. Dies wird nur in der „Pro“-Version für 9995$ pro Jahr angeboten[14]. Auch der Hosting Dienst „Shinyapp.io“ hat Einschränkungen in der kostenlosen Verwendung. So kann die kostenlos gehostete App nur 25 Stunden im Monat aktiv verwendet werden und erhält ein Branding von RStudio. Die günstigste Version ohne Branding mit 100 aktiven Stunden im Monat kostet dann schon 9$ pro Monat (beziehungsweise 100$ pro Jahr), während man einen „Performance Boost“ erst ab 39$ pro Monat (beziehungsweise 440$ pro Jahr) erhält[15]. Dies zeigt, dass die Veröffentlichung einer Shiny-App ziemlich teuer sein kann, auch wenn Shiny selbst kostenlos ist.

Alles in allem bietet Shiny also viele Möglichkeiten zur Visualisierung. Möchte man seine Shiny-App jedoch als Webseite veröffentlichen, muss man bereit sein, entweder die Einschränkungen in Kauf zu nehmen oder entsprechend Geld zu investieren.

13 Vgl. Shiny by RStudio: LESSON 7 - Share your apps
14 RStudio: Pricing
15 shinnyapp.io by RStudio: Pricing

# 6 Abschlussbetrachtung

Das R-Package Shiny von RStudio ist ein „Web Application Framework“[01] für R, dessen Verwendung in den vorangegangenen Kapiteln ausführlich erläutert und erklärt worden ist.

Es gibt zwei Möglichkeiten eine Shiny-App aufzubauen. Einerseits kann sie aus einem R-Script oder den beiden R-Scripten ui.R und server.R bestehen[02]. Diese haben wiederum einen festen internen Aufbau. Dabei ist die ui.R für die Darstellung der Benutzeroberfläche und die server.R für die Funktion der App zuständig[03]. Außerdem können noch weitere Scripte, sowie sämtliche R-Packages und Packages von Shiny selbst verwendet werden, um einen möglichst großen Funktionsumfang zu bieten.

Zur Datenvisualisierung können auch Daten aus Dateien verwendet werden, sofern sie entsprechend formatiert im Ordner „data“ der App gespeichert werden[04]. Dies können zum Beispiel CSV-Dateien sein.

Trotz vieler Möglichkeiten, die Shiny bietet, gibt es natürlich auch Nachteile, wie die Berechnung von Kurven über Näherungswerte, oder dass die Veröffentlichung als Webseite ohne Einschränkungen recht teuer werden kann.

Die vielen Möglichkeiten dieses Packages und die verhältnismäßig einfache Verwendung überwiegen aber bei Weitem die Nachteile, so dass eine Datenvisualisierung mit Shiny nur zu empfehlen ist, zumal Shiny auch noch weiter entwickelt wird. Für professionelle Veröffentlichungen, zum Beispiel von Unternehmen, ist es daher durchaus eine Überlegung wert, gegebenenfalls bei Bedarf die Kosten für die Pro-Version zu investieren.

Alles in Allem ist Shiny ein sehr mächtiges Werkzeug, dass durchaus Potential haben könnte, in Zukunft zu DEM Werkzeug für Datenvisualisierung im Internet zu werden.

01 Vgl. Shiny by RStudio
02 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
03 Vgl. Shiny by RStudio: LESSON 1 - Welcome to Shiny
04 Shiny by RStudio: LESSON 5 - Use R scripts and data

# 7 Literaturverzeichnis

**Chang, Winston (2014):** *Single-file Shiny apps.* http://shiny.rstudio.com/articles/single-file.html (Stand: 08.01.2017)

**Hijmanns, Robert (2009):** http://www.gadm.org/download (Stand: 26.11.2016)

**Newman, Keith (2016):** *Plotting Country maps in R using GADM data.* https://www.students.ncl.ac.uk/keith.newman/r/maps-in-r-using-gadm (Stand: 26.11.2016)

**RStudio (2016):** https://www.rstudio.com/products/shiny/shiny-user-showcase/ (Stand: 15.01.2017)

**Shiny by RStudio (2016):** http://shiny.rstudio.com/ (Stand: 07.01.2017)

**shinnyapp.io by RStudio (2016):** *Pricing.* http://www.shinyapps.io/ (Stand: 15.01.2017)

**Statistisches Bundesamt (2016):** *Bevölkerung: Bundesländer, Stichtag, Geschlecht.* https://www-genesis.destatis.de/genesis/online/data;jsessionid=6308BE3835673F15DC1A00D-126BEBDA3.tomcat_GO_2_2?operation=abruftabelleAbrufen&selectionname=12411-0010&levelindex=1&levelid=1481018500319&index=20 (Stand: 26.11.2016)

# 8 Eigenständigkeitserklärung

Ich Kristina-Susann Baudach versichere eidesstattlich, dass die vorliegende Arbeit mit dem Titel *Mit Shiny visualisieren – Die Bevölkerungsverteilung in Deutschland* von mir selbstständig, ohne Hilfe Dritter und ausschließlich unter Verwendung der angegebenen Quellen angefertigt wurde. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form, auch nicht in Teilen, keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

| 18. Januar 2017 | Kristina-Susann Baudach |
| --- | --- |
| Datum | Unterschrift |