LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
INSTITUT FÜR STATISTIK

Explorative Datenvisualisierung mit *Shiny* in R

BACHELORARBEIT

ZUR ERLANGUNG DES AKADEMISCHEN GRADES
BACHELOR OF SCIENCE (B. SC.)

Author: Cornelius Fritz
Betreuer: Prof. Dr. Christian Heumann
Datum 31. Juli 2016

Abstract

Diese Arbeit beschäftigt sich mit interaktiver explorativer Datenanalyse anhand von *Shiny* in *R*, das ein R-Paket zur Erstellung von Webanwendungen und interaktiven Applikationen ist. Zu Beginn wird auf bereits implementierte Möglichkeiten der interaktiven Datenanalyse eingegangen, um anschließend ein neues Paket namens *interVisu* vorzustellen. Ziel dieses Paketes ist es *Shiny*-Applikationen in Funktionen einzubetten, um explorative Datenanalyse interaktiv gestaltet zu können ohne große R- und Programmierkenntnisse von den Benutzern erwarten zu müssen. Alle implementierten Funktionen bauen auf das R-Paket *Shiny*, somit kann der Quell-Code jeder Anwendung auch leicht den eigenen Präferenzen angepasst werden. Das Paket umfasst acht Funktionen, welche in drei Kategorien eingeteilt werden: zu einen primäre Analyse von metrischen und kategorialen Merkmalen, zum anderen Zeitreihen. Jede Anwendung wird einzeln mit einem zugehörigen Tutorium erklärt.

Inhaltsverzeichnis

1 Einleitung

1.1 Stellenwert der interaktiven Datenanalyse

"*Finding the question is often more important than finding the answer*"[1](Tukey 1980, S. 24) ist die von John W. Tukey vorgeschlagene Maxime der Datenanalyse. Diese ist in zwei sich oft überschneidende Gebiete zu trennen, einmal die explorative und die konformative Datenanalyse, welche heute meist mit Inferenzstatistik gleichgesetzt wird.
Konformative Datenanalyse stellt anfänglich Fragen und Hypothesen und versucht anschließend genug Daten zu sammeln, um mithilfe von Inferenzstatstik auf diese mithilfe von approximativen Wahrscheinlichkeitsmodellen eine Aussage mit einer Fehlerrate von 5% treffen zu können (vgl. Tukey 1988, S. 420). Diese Art der Datenanalyse ist jedoch alleine lückenhaft, da keine Frage nach dem Ursprung der Hypothesen gestellt wird, sondern von diesen ausgegangen wird. Um diese Lücke zu füllen gibt es die explorative Datenanalyse. Während man bei der konformativen Statistik die Daten meist erst nach der Fragestellung erhebt, diese also bezüglich der zu erhebenden Merkmale und Größe der Stichprobe daran anpassen kann, fängt die explorative Analyse erst nach der Erhebung der Daten an (vgl. ebd., S. 420). Nun wird ohne explizite Vermutungen nach latenten Strukturen und Auffälligkeiten in den Daten gesucht. Im Gegensatz zur inferenzstatistischen Analyse der Daten, welche meist auf ein bestimmtes Repertoire an Methoden und Analysen zurückgreift, soll die explorative Analyse sehr viel freier und dadurch flexibler gestaltet werden (vgl. ebd., S. 420). Ganz generell wird anfänglich die Frage gestellt, welche Informationen aus den Daten zu gewinnen sind. Was scheinen die Daten zu bedeuten, ist in diesem Kontext wichtiger als die in der Inferenzstatistik zu beantwortende Frage, welche Charakteristiken der Daten signifikant oder nicht sind. Tukey beschreibt die explorative Datenanalyse als die Einstellung, Daten flexibel und ohne Vorbehalte zu untersuchen (vgl. Tukey 1980, S. 23) .
Für eine schlüssige Datenanalyse ist aber keine dieser beiden Betrachtungen alleine suffizient. Eine Symbiose der beiden Analysen ist von Nöten. Es können mit einer explorative Analyse Fragestellungen oder Hypothesen gewonnen werden, welche darauf in einer inferenzstatistischen Analyse auf Signifikanz überprüft werden (vgl. Tukey 1988, S. 421).
Um eine explorative Datenanalyse praktisch umzusetzen, ist also kein explizites Regelwerk wichtig, sondern die Möglichkeit bereits den Anschein von Strukturen in den Daten erkennen zu können. Ab besten können diese latenten Zusammenhänge durch das menschliche Auge in einer adäquat gewählten graphischen Darstellung identifiziert werden (vgl. Tukey 1969, S. 1). Welcher Plot und wie das zu untersuchende Merkmal transformiert wurde, ist von der Datenlage abhängig. Hier sei auf den Unterschied zwischen der graphischen Darstellung von Daten im Kontext der explorativen und deskriptiven Analyse hingewiesen. In einer deskriptiven Darstellung soll eine

[1] Deutsche Übersetzung: "*Es ist oft wichtiger die Frage als die Antwort zu finden.*"

Aussage anhand einer Graphik visuell dargestellt werden, welche auch ohne Hintergrundinformationen verstanden werden soll. Im Gegensatz dazu ist eine explorative Darstellung eine von vielen Plots, welche meist nur im Zusammenhang mit anderen Darstellungen und Hintergrundwissen eine einzelne Aussage beantwortet (vgl. Theus und Urbanek 2009, S. 5). Da in der Exploration keine generelle Hypothese und somit Richtung der Datenanalyse vorgegeben ist, sind viele unterschiedliche Darstellungen und Blickwinkel auf die Daten notwendig. Sie sollte deswegen auch computergestützt sein. Es ist auch eine interaktive Implementierung hilfreich, in welcher der Benutzer schnell zwischen verschiedenen Graphen wechseln kann.
Heutzutage können auch nur 23% der Daten zu Analysezwecken verwendet werden. Weniger als 3% werden davon nur analysiert (vgl. *Kaum jemand blickt durch im Datenwust* 2011, S. 1). Aus diesem Grund sollte explorative Datenanalyse nicht nur interaktiv sein, sondern auch keine großen Vorkenntnisse von den Benutzern fordern.

1.2 Historische Entwicklung interaktiver und dynamischer explorativer Datenanalyse

Tukey selbst hat mit Fishkeller und Friedmann bereits 1974 an der Fakultät *Graphics Interpretation* des *Linear Accelerator Center* in Stanford einen Computer zur interaktiven Datenvisualisierung entwickelt. Dies war einer der ersten Versuche computaler explorativer Datenanalyse und wurde *PRIM-9* genannt (vgl. Tukey 1988, S. 307 f). Der Name beschreibt in Abkürzungen alle Möglichkeiten des Programms: **P**rojektionen von Daten in den zweidimensionalen Raum - meist Scatterplots oder Streudiagramme genannt- , **R**otationen von zwei Koordinatenachsen, **I**solation von einzelnen Beobachtungen, **M**askieren von Daten und das ganze ist mit bis zu **9** Merkmalen möglich (vgl. ebd., S. 312 ff). Geht man also von der Analyse eines Datensatzes mit m Merkmalen, wobei $(m > 2) \wedge (m < 9)$ gilt, aus, so kann der Anwender immer einen von $\binom{m}{2}$ möglichen Scatterplots betrachten. Um die Koordinaten aus allen möglichen Blickwinkeln zu inspizieren, ist es möglich immer zwei von dem Nutzer bestimmte Koordinatenachsen zu rotieren[2].
1985 implementierte Asimov wieder beim *Stanford Linear Accelerator Center* eine graphische Darstellung namens Grand Tour (vgl. Asimov 1985, S. 1). Es werden Projektionen von Variablen angezeigt, jedoch drehen sich die Achsen stetig. Somit wird eine dynamische Darstellung erzeugt, welche alle möglichen Blickwinkel auf die Daten wie in einem Film darstellt. Da diese Blickwinkel in einem p-dimensionalen Raum schnell mit einer steigenden Zahl an Merkmalen zu einem Fluch der Dimension werden können, wurden hierzu Erweiterungen wie der *Projection Pursiuit*-Ansatz konzipiert. Nun werden alle Blickwinkel anhand eines bestimmten Indexes nach

[2] PRIM-9 wurde auch von Tukey in einem Film vorgeführt: siehe `http://stat-graphics.org/movies/prim9.html`

deren Informationsgehalt eingestuft, und nur ausgewählte Blickwinkel mit hohem Informationsgehalt angezeigt. Für weiterführende Literatur hierzu sei auf Buja und Asimov 1986; Asimov 1985 und Cook u. a. 1995 hingewiesen.
Cleveland und Becker konnten 1987 mithilfe der *AT & T Bell Laboratories*, bei welchen auch bereits Tukey während seiner Karriere als Professor an der Stanford Universität gewirkt hat (vgl. Leonhardt 2000, S. 1), die erste interaktive Scatterplot-Matrix programmieren (vgl. W. Cleveland und Becker 1987, S. 1). Aufgrund der fortgeschrittenen Möglichkeiten der Computer konnten Punkte mithilfe einer Brushing-Interaktion in allen Scatterplots hervorgehoben, gelöscht und skaliert werden[3].
Theus entwickelte 1997 das eigenständige Softwarebundle *Mondrian*, welches exklusiv für interaktive Datenanalyse gedacht ist (vgl. Theus 2002, S. 1). Mithilfe dieses Programms ist es möglich, interaktive Histogramme, Balkendiagramme, Scatterplots, Mosaikplots und Parallel Coordinates Plots interaktiv zu gestalten. Mit logischen Operationen ist es auch möglich, sehr spezifische Mengen an Beobachtungen in allen dargestellten Graphen hervorzuheben und somit explorativ zu erforschen (vgl. ebd., S. 1f).
In R programmierte Theus zusammen mit Urbanek 2006 das erste Programm-Paket für interaktive Graphen namens *iPlots*, inkludierte mögliche Darstellungen sind ähnlich zu *Mondrian* (vgl. Urbanek und Theus 2003, S. 1). Mittels der Konvention die herkömmlichen Namen von Funktionen zum Abrufen von Graphen in R mit einem *i* für *interactive* zu versehen, wurde versucht ‚die Verwendung des Paketes möglichst intuitiv zu strukturieren. *iPlots* wurde 2008 um das Paket *Acinonyx* erweitert, um die Modelldiagnose auch interaktiv durchführen zu können. Genauere Informationen können in Urbanek o.D. und Urbanek und Theus 2003 nachgelesen werden.
In der vorliegenden Arbeit wird nach einer Einführung in das Paket *Shiny*, das von mir entwickelte Paket *interVisu*, welches neue Möglichkeiten der explorativen Datenanalyse eröffnet, vorgestellt. Zum Schluss wird noch ein Ausblick auf mögliche Erweiterung des Paketes gegeben.

2 Implementierung von interaktiver Datenanalyse in R

Neben den bereits genannten Möglichkeiten in R Daten interaktiv zu analysieren - *iPlots* und *Acinonyx*- gibt es auch die Option, selbst interaktive Webanwendungen in R mit dem Paket *Shiny* zu erstellen. Innerhalb dieser Anwendungen kann man damit beliebige interaktive Datenanalysen programmieren.
Shiny steht zum Zeitpunkt dieser Arbeit in der Version 0.13.2 über den CRAN-Server jedem Anwender kostenlos zur Verfügung. Dieses Paket ermöglicht es interaktive Webapplikationen

[3] Eine Präsentation des Programmes kann unter `http://stat-graphics.org/movies/brushing.html` betrachtet werden

relativ einfach und ohne zwingende HTML- und CSS-Kenntnisse gestalten zu können. Entwickelt wurde es von dem Unternehmen RStudio, welches auch den weitverbreiteten R-Editor "*R-Studio*" und viele weitere R-Pakete programmiert hat.

Abbildung 1: Programme und R-Pakete die auch von RStudio veröffentlicht wurden. Quelle: http://www.rstudio.com/wp-content/uploads/2015/01/Untitled.001.jpeg

2.1 Interaktive Webanwendungen mit *Shiny* in R

Zu Beginn soll darauf hingewiesen werden, dass in dieser Arbeit nicht im Detail erklärt wird, wie Web-Applikationen anhand des R-Paketes *Shiny* explizit programmiert oder welche Möglichkeiten dem Anwender durch dieses Paket gegeben werden. Es wird nur auf die Funktionsweise und den Aufbau von *Shiny*-Applikationen eingegangen, soweit sie zum Verständnis der zugrunde liegenden Struktur beitragen.

Um die in dem Paket *interVisu* zur Verfügung gestellten Funktionen nach eigenem Belieben erweitern zu können, sei hier auf zwei kostenlose Online-Kurse hingewiesen. Der erste Kurs wird von den Entwicklern des Paketes auf deren Webseite zum Selbststudium zur Verfügung gestellt. Dieses Tutorium umfasst aber nur grundlegende Möglichkeiten von *Shiny*[4]. Will man weiterführende Informationen erfahren, kann man sich entweder Artikel zu diversen spezifischen Themen auch auf der Webseite der Entwickler[5] oder die von Zev Ross konzipierten und

[4] `http://shiny.rstudio.com/tutorial/`
[5] `http://shiny.rstudio.com/articles/`

angebotenen Kurse durcharbeiten[6].
Mit bereits von *Shiny* implementierten Widgets, kann man interaktive Bestandteile in Anwendungen einbinden. Durch diese werden alle notwendigen und möglichen Input-Werte durch den Anwender festgelegt. Input-Werte können numerische, logische und character Werte sein, je nachdem welches Widget in welcher Wise in der Anwendung eingebunden wurde. Eine vollständige Zusammenfassung aller bereits in *Shiny* zur Verfügung stehenden Widgets findet man auf dem Webauftrit der Entwickler[7]. Mit den notwendigen HTML-, CSS-, und JavaScript-Kenntnissen lassen sich aber auch beliebige weitere Widgets erstellen und anschließend in der Applikation verwenden (*Build custom input objects* 2014).
Grundlegend sind *Shiny*-Applikationen durch ihre Struktur in zwei miteinander interagierende Teile zu trennen. Ein Teil beschreibt den Aufbau der Applikation und wird meist mit UI[8] abgekürzt . In diesem Abschnitt des Codes wird das gesamte Layout der Anwendung mit allen möglichen Interaktionen, also Widgets, definiert. Der zweite Bestandteil ist der Server: Hier werden alle im UI angelegten Outputs mit R-Code, welcher auf alle Input-Werte zurückgreifen kann, definiert.

2.1.1 UI: User-Interface

Man kann sich das User-Interface (UI) eines Programms als die Beschreibung aller visuellen Elemente vorstellen. Dieser Code definiert die Struktur eines digitalen Dokuments und wird in der textbasierten Auszeichnungssprache Hypertext Markup Language[9] verfasst. Kenntnisse in HTML sind jedoch nicht zwingend notwendig, da in *Shiny* Funktionen definiert werden, welche den gewünschten HTML-Code als Rückgabe-Wert liefern. Legt man das gesamte Layout anhand der R-Funktionenfe, kist das Layout eine als Verschachtlung vieler R-Funktionen. Alternativ kann man diese Dokumente aber auch in HTML verfassen (*Build your entire UI with HTML* 2014). Im Folgenden wird das UI jedoch mit R-Funktionen aufgebaut.
Jedes *User Interface* fängt mit einer Hauptfunktion an, welche die aufzubauende Webseite für die Applikation grundlegend definiert. Dies ist eine Art Grundgerüst für jede Anwendung, welche verschiedene Optionen festlegt. Entweder wählt man ein sich fließend erweitern bzw. verkürzendes Fest, falls dieses vergrößert bzw. verkleinert wird, wobei alle voneinander abhängigen Skalierungen der einzelnen Elemente gleich bleiben. Oder ein statisches Layout, welches sich nicht verändert und zuletzt ein Fenster, das immer das gesamte Fenster füllt. Diese drei generellen Arten von Seiten werden `fluidPage, fixedPage` und `fillPage` genannt. In die gewählte Funktionen kann man weitere Layout-Elemente integrieren. Beispielsweise kann man sich für ein vorgefertigtes Layout wie das Sidebar-Layout entschließen, in welchem ein Sidebar-

[6] `http://zevross.com/blog/2016/04/19/`
[7] `http://shiny.rstudio.com/gallery/widget-gallery.html`
[8] für User-Interface
[9] Abgekürzt HTML

und Main-Panel bereits definiert sind. Man kann das Layout jedoch auch nach den fortlaufenden Reihen mit der Funktion `fluidRow` definieren. Ein minimales Layout mit Sidebar-Layout würde folgendermaßen aussehen[10].

```
 fluidPage(
# In diesem Beispiel benutzen wir
# eine sich fließend erweiternde Seite.
 sidebarLayout(
# Nun wird das generelle Layout definiert,
    sidebarPanel(
      # Hier lässt sich definieren, was in dem seitlichen
      # etwas kleineren
      # Panel angezeigt werden soll,
    ),
    mainPanel(
      # In diesem Abschnitt lässt sich definieren, was in dem
      # Haupt-Panel angezeigt werden soll,
    )
  )
 )
```

Der Output dieser Funktion ist nun HTML-Code und wäre bereits ein vollständiges Layout. Für eine genauere Erklärung der möglichen Layoutgestaltungen sei hier auf zwei weitere Artikel verwiesen (Allaire 2014; *Build a dynamic UI that reacts to user input* 2014).

Um jedoch Outputs und Widgets in einer Anwendung zu inkludieren, werden wiederum R-Funktionen die HTML-Code produzieren, verwendet. Führt man die Applikation aus, wird der Inhalt des Plot-Outputs anhand des Server gefüllt. Die meist verwendete Funktion für das Inkludieren von interaktiven Graphiken ist `plotOutput`. Damit der Computer bei dem Ausführen der Anwendung weiß, wo ein im User-Interface angelegter Plot in dem Server-Skript definiert ist, braucht man ein eindeutiges Label zur Referenzierung von jedem Graphen. Optional lassen sich auch Breite (*width*) und Höhe (*height*) definieren.

Ist also beispielsweise ein Sidebar-Layout mit einem Slider-Input im Sidebar-Panel und einer Graphik im Main-Panel gewünscht, produziert folgende Funktion das beabsichtigte HTML-Dokument.

```
fluidPage(
sidebarLayout(
   sidebarPanel(
     # Es wird ein Slider-Input definiert, der numerische
     # Werte von 1 bis 3 annehmen kann
     # und anfänglich den Wert 2 hat.
     # Das Label des Inputs ist slider, was bedeutet,
     # dass man später Werte dieses Inputs mit dem Begriff
     # input$slider im Server Skript referenzieren kann.
```

[10] Weitere *Shiny*-Beispiele befinden sich im Anhang A

```
      sliderInput(inputId="slider",min=1, max=3, value=2)
    ),
    mainPanel(
      # Nun wird ein Plot definiert der das Label "plot" hat
      # und 300 Pixel hoch und breit ist.
      # Will man anschließend genau diesen Output im Server
      # definieren, kann man dies mit output$plot.
      plotOutput(outputId="plot", width=300, height=300)
    )
  )
)
```

2.1.2 Server

Sobald das User-Interface definiert ist, können alle somit angelegten Outputs in dem Server-Skript anhand von herkömmlichem R-Code definiert werden. Konzeptionell ist der Server-Abschnitt einer Applikation eine Funktion, deren Parameter alle möglichen Inputs und die Label der angelegten Outputs sind. Alle durch den Anwender definierten Inputs werden als Listen-Objekt der Server-Funktion übergeben. Hierbei kann man sich jedes einzelne Input-Widget als einen Listeneintrag vorstellen. Der Name des Listen-Eintrags ist der gleiche des bereits im User-Interface definierten Labels. Exemplarisch bedeutet dies, dass man jegliche Input-Werte in der Server-Funktion mit `input$label` referenzieren und verwenden kann.
Aufgebaut ist die Server-Funktion nach den einzeln definierten Outputs, die als Liste abgespeichert werden. Wieder hat nun jeder Output einen Listeneintrag, deren Namen gleich dem im User-Interface definierten Namen ist. Mögliche Outputs sind Plots, Prints, Text und HTML. Diese verschiedenen Typen werden durch eine Listenzuweisung mit den Funktionen `renderPlot`,`renderPrint`, `renderText` oder `renderUI` beschrieben. Indiziert durch deren Namen der Funktionen unterscheiden sich diese grundlegend bezüglich der zu erwartenden Output-Objekte. Legt man in dem Layout einer Applikation beispielsweise ein Plot-Output mit dem Label "plot" an, sähe eine minimalistische Server-Funktion wie folgt aus.

```
shinyServer(function(input, output) {
    output$plot <- renderPlot({
        # Hier kann mit normalem R-Code ein beliebiger plot
        # beschrieben werden.
    })
})
```

Bei jeglicher Alternation der verwendenden Input-Werte, wird der gesamte zu diesem Objekt gehörende R-Code neu durchlaufen.
Unter Umständen verwendet man jedoch in mehreren Graphen die gleichen Daten oder Variablen. Jeder Code-Abschnitt wird jedoch unabhängig voneinander ausgeführt, also können lokale Variablen nur in dem Code-Abschnitt verwendet werden, in welchem sie definiert wurden. Somit

muss man diese Variablen in jedem Code-Abschnitt neu definieren. Das ist jedoch unnötige Arbeit für den Computer, welche durch sogenannte reaktive Objekte erspart werden kann. Man kann ein *reactiveValue* als globale Listen-Variable in dem gesamten Server-Skript mit beliebigen Einträgen betrachten. Definiert werden diese Objekte durch die Funktion `reactiveValues` separat von den einzelnen Outputs in der Server-Funktion. In diesem Beispiel wird ein reaktives Objekt des Namens "*values*" mit dem Anfangswert NULL definiert.

```
values <- reactiveValues(a = NULL)
```

Die einfachste Möglichkeit die Werte der Listeneinträge zu verändern ist mit der Funktion `observeEvent`. Hierbei wird ein R-Code definiert der nur ausgeführt wird, falls ein definiertes Ereignis eines Input-Wertes eintritt.

```
observeEvent(input$slider, {
    values$a <- input$slider
    # Hier kann jedoch auch beliebiger R-Code stehen.
})
```

In diesem Beispiel wird dem Listeneintrag *a* der reaktiven Liste *values* bei jeglicher Aktion des bereits in einem früheren Beispiel definierten Inputs mit dem Label "slider" der Wert des Inputs zugewiesen. In allen Code-Abschnitten der einzelnen Outputs kann man nun mit *values$a* den Wert des Listeneintrags verwenden. Für weiterführende Literatur sei auf die Artikel *Reactivity: An overview* 2014; Grolemund 2015 verwiesen.

2.1.3 Ausführen der Applikation

Wenn das User-Interface und die Server-Funktion definiert wurden, bleibt zu klären wie sich die Applikation ausführen lässt.
Bis zur *Shiny*-Version 0.10.1 war es nötig beide Bestandteile der Applikation in separaten Skripten zu definieren, die mit den Namen `ui.R` und `server.R` in einem bestimmten Ordner abgespeichert werden. Mit der Funktion `runApp` kann man anschließend die Anwendung starten. Ein minimales Beispiel sieht folgendermaßen aus:

```
#ui.R
shinyUI(
 fluidPage(
 # Hier wird das Layout definiert
 )
)
```

```
# server.R
shinyServer(function(input, output) {
    # Falls in ui.R jegliche Outputs angelegt wurden ,werden
    # diese hier definiert
})
runApp("App")  # Der Name des Ordners in welchem sich
```

```
# die beiden Skripte ui.R und server.R befinden muss hier
# genannt werden.  Dieser Ordner wird bei des Ausfuehrung
# als working directory betrachtet.
```

Seit der Version 0.10.2 lassen sich jedoch Applikationen auch als Objekt im *Workspace* von R abspeichern. Somit braucht man nur noch ein Skript zur Beschreibung der Applikation, die sich mit der Funktion *shinyApp* starten lässt. Diese Technik wird von nun an verwendet. Die oben beschriebene Anwendung ließe sich analog anhand folgendes R-Skriptes erzeugen.

```
ui=fluidPage(
# Hier wird das Layout definiert.
)
server=function(input, output) {
# Falls in ui.R jegliche Outputs angelegt
# wurden, werden diese hier definiert
}
shinyApp(ui = ui, server = server)
```

2.2 Erweiterung von *Shiny* durch Funkionen

Die Programmierung von *Shiny*-Applikationen ist zwar relativ leicht zu erlernen, jedoch trotzdem meist zu viel Aufwand für eine explorative Datenanalyse, da Applikation und deren Inputs auf die verwendeten Daten angepasst werden müssen. Deshalb generalisieren wir nun das Konzept von *Shiny*-Applikationen, indem wir beide Bestandteile der Anwendung, also das User-Interface und die Serverfunktion, in eine Funktion einbetten.
Durch diese Art der Erweiterung hat man mehr Flexibilität, da die Applikation nicht mehr für einen spezifischen Datensatz geschrieben sein muss, sondern sich den einzelnen Gegebenheiten anpassen kann. In dem Abruf der Funktion lassen sich so auch optionale User-Interface-Elemente definieren. Durch das Bereitstellen dieser Funktionen können auch Benutzer ohne größere Vorkenntnisse von R und Shiny ihren eigenen Datensatz explorativ und deskriptiv analysieren. Bei gewissen Vorkenntnissen ergibt sich durch diese Anwendungen auch der Vorteil, dass der Quell-Code, wie gerade beschrieben, relativ leicht verständlich ist und daher unproblematisch den spezifischen Präferenzen der Nutzer angepasst werden kann. Insgesamt besteht so bei der interaktiven Datenvisualisierung weit mehr Spielraum für den Anwender.
Leider stehen bis dato keine Funktionen in R-Packages zur Verfügung, welche dem Benutzer ohne jegliche Programmierkenntnisse die Arbeit mit *Shiny*-Applikationen erlauben. Deshalb entstand das R-Package *interVisu*. Ziel ist das Bereitstellen von Funktionen, welche den Nutzern für beliebige Daten eine interaktive explorative Datenvisualisierung mit Shiny in R ermöglichen.

3 *interVisu*: Ein Package zur interaktiven Datenvisualisierung mit *Shiny* in R

Das Paket *intervisu* steht zur Zeit dieser Arbeit in Version 0.1.0 über Github zur Verfügung und kann anhand von folgendem Code in R installiert werden.

```
install.packages("devtools")
library(devtools)
install_github("corneliusfritz/intervisu",force=T)
library(intervisu)
```

Anspruch des Pakets ist es Anwendern ohne große Programmierkenntnisse eine explorative Datenanalyse zu ermöglichen. Weiter soll die Suche einer adäquaten deskriptiven Darstellung von metrischen Variablen interaktiv erleichtert werden. So muss man sich nicht auf standardmäßig implementierte Algorithmen zur Auswahl von Parametern für eine optimale explorative oder deskriptive Darstellung verlassen.
Die Funktionen von *interVisu* werden , wie bereits besprochen, in drei Gruppen zur Visualisierung von verschiedenen Datentypen eingeteilt. Um diese Abgrenzung ziehen zu können, werden nun metrische Merkmale, kategoriale Merkmale und Zeitreihen definiert.

Def. 3.1 Diskretes Merkmal/ Faktor: Unter einem diskreten oder kategorialen Merkmal X versteht man ein Merkmal das abzählbar viele Merkmalsausprägungen besitzt. Die Menge aller möglichen Merkmalsausprägungen ist abzählbar, so lässt sich die Menge aller p möglichen Ausprägungen von X als $\{x_1 \dots x_p\}$ notieren (vgl. Forster 2008, S. 83). Das Merkmal wird auch *Faktor* genannt, daraus folgend alle möglichen Ausprägungen $x_1 \dots x_p$ *Faktorlevel* oder *Faktorstufen* (vgl. Fahrmeir, Künstler u. a. 2011, S. 16f).

Def. 3.2 Stetiges/ metrisches Merkmal: Falls ein Merkmal X eine überabzählbare Menge an möglichen Ausprägungen besitzt, nennt man das Merkmal *stetig* oder *metrisch* (vgl. ebd., S. 16f).

Def. 3.3 Zeitreihe: Falls ein stetiges oder diskretes Merkmal T bei einem Objekt oder einer Person über einen bestimmten Zeitraum m-mal beobachtet wurde, wird die Sequenz von Werten dieser Variablen Zeitreihe genannt (vgl. Diekmann 2012, S. 315).

Generell werden bei allen Anwendung in dem Paket gewisse Konventionen eingehalten, um eine einfache und intuitive Bedienung gewährleisten zu können. So wird bei allen nicht-statischen Funktionen anhand des ersten Parameters `data` der zu analysierende Datensatz übergeben. Auch wird in allen Funktionen, in welchen eine Unterscheidung zwischen metrischen und diskreten Merkmalen getroffen werden muss, ein numerischer Parameter `n` verwendet, um Variablen dadurch kategorisieren zu können. Falls eine Variable mehrere unterschiedliche Ausprägungen als den numerischen Wert `n` besitzt, wird diese Variable als metrisch betrachtet. Auch werden

alle Fenster mit der Funktion `fluidPage` aufgebaut, womit sich die Größe der Anwendung dem Browser- oder Editor-Fenster dynamisch anpasst. Nur Darstellungen, deren explizite Höhe oder Breite im Funktionsaufruf definiert wurden, sind absolut definiert. Die verwendeten Input-Widgets sind auch soweit wie möglich kongruent gestaltet. Jegliche Auswahl von Listen, seien es generelle Optionen oder eine auszuwählende Variable, werden mittels eines Select-Widgets ausgewählt. Klickt der Benutzer auf das Widget, öffnet sich eine Liste aus der man mit einem weiteren Klick einen Eintrag auswählen kann. Alle numerischen Inputs können entweder mit der Tastatur eingegeben werden, oder mithilfe kleiner Pfeiltasten in dem Widget selbst. Zuletzt werden jegliche Eingaben von Daten mithilfe des Date-Input-Widgets abgefragt, wobei man ein Datum mit Jahr, Monat und Tag eingeben kann.
Im Folgenden wird jede Applikation einzeln, nach den Gruppen sortiert, erklärt. Zuerst wird je ein grober Überblick der Anwendung und des Abrufens der zugehörigen Funktion gegeben, danach wird die explorative Idee hinter der Anwendung sowie die jeweils verwendeten Methoden im Detail erklärt[11].
Eine weitere, in englischer Sprache verfasste Erklärung der Interaktionsmöglichkeiten kann in der Dokumentation der Funktionen in R nachgelesen werden. Beispielsweise ließe sich die Hilfeseite zu der Applikation `Stacked_Barplot` anhand des Befehls `help("Stacked_Barplot")` in der in R integrierten Hilfe aufrufen.

3.1 Metrische Variablen

3.1.1 Analyse einer metrischen Variable

Die erste Funktion lässt sich anhand des Befehls `Metric_Single_Variable_Analysis(data, n=10, a=50, width=700, height=700)` aufrufen. Sie zeichnet dem Anwender Boxplots, Histogramme und Dichteschätzer von allen sich in dem übergebenen Datensatz befindenden metrischen Variablen. Mit den Parametern `width` und `height` kann nach der Anzahl an Pixeln die explizite Breite und Höhe des angezeigten Graphen bestimmt werden.
Bei jeder spezifischen Darstellung sind Parameter interaktiv zu wählen. Falls man mit einer Präsentation des Merkmals zufrieden ist, lässt sich der R-Code, welcher zu der interaktiv erarbeiteten Visualisierung führt, in einem Textfeld anzeigen.
Durch diese Anwendung kann der Benutzer alle Entscheidungen bezüglich der Darstellung des metrischen Merkmals selbst interaktiv treffen. Man hat sich daher nicht mehr auf unbekannte Algorithmen zu verlassen, welche automatisiert Parameter auswählen, die für eine graphische Repräsentation nicht objektiv gewählt werden sollten. Speziell bei der explorativen Analyse können solche Mechanismen zu einer verzerrten Darstellung und somit nicht adäquaten Visuali-

[11] In Anhang B ist auch eine Übersicht aller in *interVisu* inkludierten Funktionen mit zugehörigen Analysemethoden dargestellt.

sierung des Merkmals führen. Auch können alle Graphen durch das Anzeigen des Codes leicht reproduziert und nach Belieben angepasst werden. Zudem erlernt man aus pädagogischer Sicht interaktiv die Tragweite von grundsätzlich subjektiv gewählten Parametern und wie diese eine Darstellung modifizieren können.

Methoden

Um den Aufbau und die möglichen Interaktionen verstehen zu können, werden nun alle verwendeten Methoden und graphischen Darstellungen beschrieben.
Die erste mögliche Darstellung ist der Boxplot, welcher als Zusammenfassung diverser Lagemaße einer metrischen Variable zu verstehen ist. Lagemaße sind ganz generell Parameter, die in einer gewissen Weise Aufschluss über die eines Merkmals zugrundeliegenden Wahrscheinlichkeitsverteilung geben.

Def. 3.4 β-Quantil: Das β-Quantil des Merkmals X ist ein Lagemaß mit $\beta \in [0,1]$ und wird als x_β notiert. Mindestens der Anteil β der Daten ist kleiner/gleich x_β und mindestens der Anteil $1-\beta$ ist größer/gleich x_β. Es muss also für x_β des Merkmals X mit den observierten Werten $\{x_1, \ldots x_n\}$

$$\frac{|\{x \mid x \leq x_\beta\}|}{n} \geq \beta \text{ und } \frac{|\{x \mid x \geq x_\beta\}|}{n} \geq 1-\beta$$

gelten. $x_{0.25}$ und $x_{0.75}$ werden *unteres und oberes Quartil* genannt, der *Median* ist $x_{0.5}$ (vgl. Fahrmeir, Künstler u. a. 2011, S. 65f).

In einem Boxplot werden die fünf Lagemaße $x_{min}, x_{0.25}, x_{0.5}, x_{0.75}, x_{max}$ dargestellt. Diese Gruppe an Lagemaßen wird auch oft *Fünf-Punkte-Zusammenfassung* genannt (vgl. ebd., S. 66). x_{min} und x_{max} seien hier als die minimale und maximale Beobachtung des Merkmals X zu betrachten. Es soll hier nur die vertikale Version eines Boxplots beschrieben werden, der horizontale Boxplot ist identisch zu dem vertikalen nur eben um 90° im Uhrzeigersinn gedreht. Auf der Y-Achse wird nun die *Fünf-Punkte-Zusammenfassung* dargestellt. Es wird eine Box mit dem Y-Abschnitt von $x_{0.25}$ bis $x_{0.75}$ und einer nicht informativen Breite gezeichnet. Der *Median* ($x_{0.5}$), wird durch eine horizontale Linie innerhalb der Box indiziert.
Um den Boxplot werden zwei Zäune, im Englischen *whiskers*, gezeichnet. Die Y Koordinaten dieser beiden Zäune seien z_u und z_o, für die gilt:

$$z_u = \begin{cases} x_{min} & \text{, falls } x_{min} \geq x_{0.25} - 1.5(x_{0.75} - x_{0.25}), \\ x_{0.25} - 1.5(x_{0.75} - x_{0.25}) & \text{, sonst} \end{cases}$$

$$z_o = \begin{cases} x_{max} & \text{, falls } x_{max} \leq x_{0.75} - 1.5(x_{0.75} - x_{0.25}), \\ x_{0.75} - 1.5(x_{0.75} - x_{0.25}) & \text{, sonst} \end{cases}$$

Auf der Höhe von z_u und z_o werden die Zäune als zwei horizontale Linien, welche einen Teilintervall des durch die Breite definierten Intervalls der X-Achse abdecken, gezogen. Alle observierten Merkmale, welche kleiner z_u oder größer als z_o sind, werden als Punkte dargestellt, wobei die X-Koordinate durch die Mitte der Breite des Boxplots und die Y-Koordinate durch den Wert der Beobachtung außerhalb von z_u und z_o definiert ist (vgl. ebd., S. 67f).

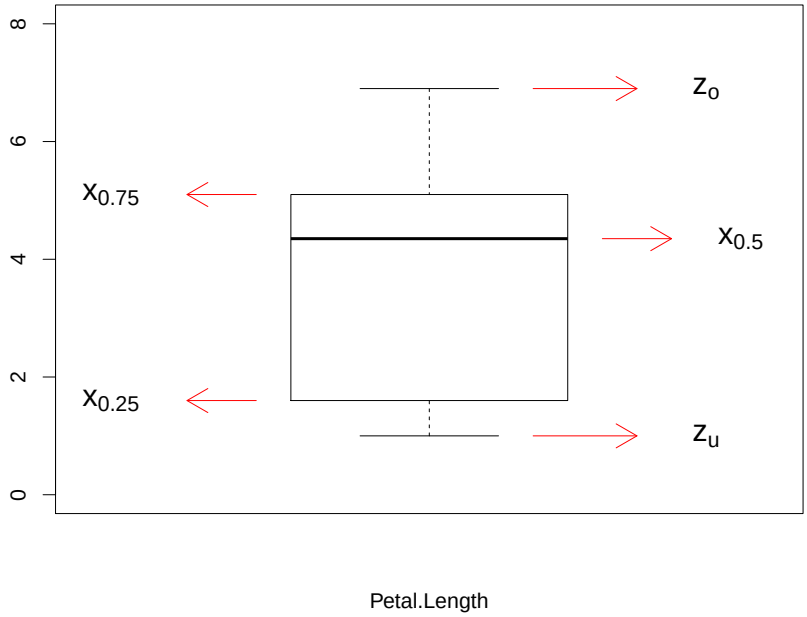

Abbildung 2: Boxplot des Merkmals "petal.length" aus dem Datensatz Iris

In einer weiteren Modifikation des Boxplots kann man alle Punkte - also nicht nur die, welche größer als z_o beziehungsweise kleiner als z_u sind - in den Plot einzeichnen. Um die Darstellung einer großen Anzahl von Beobachtungen weiterhin übersichtlich gestalten zu können, werden die nicht informativen X-Koordinaten der einzelnen Punkte in einer sogenannten *Jitter-Funktion* randomisiert. Die Y-Koordinate der einzuzeichnenden Punkte wird nicht verändert und ist durch die metrische Variable X definiert.

Def. 3.5 *Jitter-Funktion*: Will man in einen Boxplot zur Darstellung des metrischen Merkmals X alle Observierungen $\{x_1, \ldots x_n\}$ eintragen, zeichnet man die Koordinatenpunkte $(x_1, Y_1), \ldots (x_n, Y_n)$ in das Koordinatensystem des normalen Boxplots ein. Y_i ist dann mit $i = 1, \ldots, n$ eine Realisierungen der Zufallsvariable Y für die gilt:

$$Y \sim U(y_u, y_o),$$

Die Werte y_u und y_o sind jedoch unter den Bedingungen, dass $y_u, y_o \in \mathbb{R} \wedge y_u < y_o$ gilt, frei und subjektiv gewählt. Meist werden Werte des Intervalls der Breite der Box verwendet.

Jedem metrischen Merkmal liegt eine latente Dichte, welche die Verteilung der Ausprägungen beschreibt, zugrunde. Diese kann man approximativ schätzen und somit einen Überblick des Merkmals bekommen. Eine Möglichkeit zur Darstellung der unbekannte Dichte ist ein Histogramm. Es wird wieder von einem mindestens ordinal-skalierten metrischen Merkmal X mit der Menge an n observierten Werten $X_{values} = \{x_1, \ldots, x_n\}$ ausgegangen (vgl. Fahrmeir, Künstler u. a. 2011, S. 41). Um ein Histogramm plotten zu können, muss zuerst eine Bandbreite $d \in \mathbb{R}^+$ festgelegt werden. Der Wert kann unter der Bedingung $(d > min\{X_{values}\}) \wedge (d < max\{X_{values}\})$ beliebig gewählt werden. Es existiert nun ein $k = \lfloor \frac{n}{d} \rfloor$, womit man die beobachteten Werte X_{values} in k Subpopulationen gruppieren kann. Diese kleineren Untermengen seien als $X_{values,j}$ notiert und es gilt für $\forall j = 1, \ldots, k$:

$$X_{values,j} = \{x \in X_{values} \mid (x \geq x_{j,u}) \wedge (x < x_{j,o})\}$$

$\alpha \in [-\infty, min\ X_{values}]$ gilt als der Ursprung des Histogramms, und somit auch Startpunkt. $x_{j,u}$ und $x_{j,o}$ ist folgendermaßen darzustellen:

$$x_{j,u} = \alpha + j \cdot d$$
$$x_{j,o} = \alpha + (j+1) \cdot d$$

Letztendlich hat man die eigentlichen Beobachtungen des zu plottenden Merkmals X_{values} in k disjunkte Mengen aufgeteilt. Jede dieser Mengen umfasst einen Intervall mit der gleichen Bandbreite d, angefangen im Ursprung α (vgl. ebd., S. 41).
In einem Histogramm werden nun alle Werte $x_{1,u}, x_{1,o}, \ldots, x_{k,u}, x_{k,o}$ abgetragen und jeder Subpopulation $X_{values,j}$ wird ein Balken mit dem X-Achsenabschnitt von $x_{j,u}$ bis $x_{j,u}$ und dem Y-Achsenabschnitt von 0 bis f_j oder h_j eingezeichnet (Abbildung 3). f_j und h_j bezeichnen in diesem Kontext die relative und absolute Häufigkeit aller Werte in der Menge $X_{values,j}$. Da die Fläche des j-ten Balkens proportional zu der relativen oder absoluten Häufigkeit der sich in $X_{values,j}$ befindenden Werte ist, gilt das *Prinzip der Flächentreue* (vgl. ebd., S. 42).
Die Verwendung von Histogrammen hat jedoch zwei Nachteile. Erstens beeinflussen die subjektiv gewählte Bandbreite d und der Ursprung α den optischen Einruck des Graphen und, falls man eine Dichteschätzung im variablen Punkt x approximieren will, spielt nicht der Abstand der um den Wert $x \in \mathbb{R}$ liegenden observierten Werte X_{values} eine Rolle, sondern nur die Gruppenzugehörigkeit der um x liegenden Punkte. Falls z.B. der Wert x_1 bezüglich der euklidischen Norm näher an x liegt als x_2, gilt einerseits $\| x - x_2 \| > \| x - x_1 \|$, jedoch hat, falls $x_2, x \in X_{values,j}$ und $x_1 \in X_{values,i}$ mit $i \neq j$ gilt, nur x_2 Einfluss auf die Dichteschätzung im Punkt x, obwohl x_1 nach der euklidischen Norm näher an x liegt (vgl. ebd., S. 98f).
Dieser Zusammenhang ist nicht intuitiv zu legitimieren, da eine Dichteapproximation im Punkt x eigentlich ähnlich zu allen beobachteten Werten in der Nähe von x sein sollte. Eine weitere Art der Dichteapproximation - die Kerndichteapproximation- gewichtet bei der Schätzung im Punkt x alle Werte um x mithilfe einer Kernfunktion.

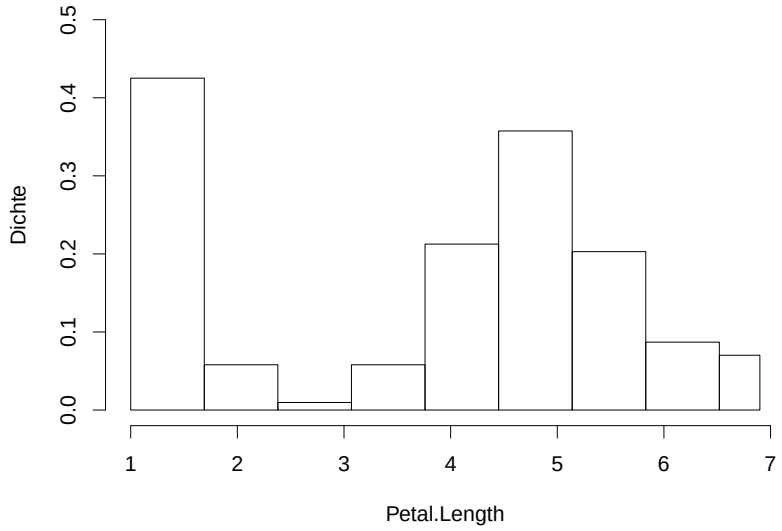

Abbildung 3: Histogramm des Merkmals "petal.length" aus dem Datensatz Iris mit d=0.69 und α=1

Nun wird der Verlauf der metrischen Variable als eine Funktion in Abhängigkeit von x aufgefasst, welche als $f(x)$ notiert wird. Die Funktion $f(x)$ wird durch $\hat{f}(x)$ lokal approximiert, was bedeutet, dass ein Raster an Werten für x angenommen wird, und für jeden Wert in dem Raster die Funktion $\hat{f}(x)$ in dem Punkt x lokal geschätzt wird. $\hat{f}(x)$ ist dabei ein nach einer bestimmten Kern-Funktion gewichtetes Mittel (vgl. ebd., S. 99).

Def. 3.6 Kern-Funktion: Eine Kern-Funktion K mit $K : \mathbb{R} \to \mathbb{R}$ ist eine nicht-negative (3.1) um 0 symmetrische (3.2) Dichtefunktion (3.3) für die somit gilt:

$$K(u) \geq 0 \; \forall u \in \mathbb{R} \tag{3.1}$$

$$K(-u) = K(u) \; \forall u \in \mathbb{R} \tag{3.2}$$

$$\int K(u)du = 1 \tag{3.3}$$

Es können verschiedene Kerne bei der Kerndichteschätzung Verwendung finden, die meist verwendeten sind jedoch:

Gauß-Kern	$K(u) = \frac{1}{\sqrt{2\pi}} exp\Big(-\frac{1}{2}u^2\Big)$
Epanechnikov-Kern	$K(u) = \frac{3}{4}(1-u^2)\mathbb{I}_{\{-1,1\}}(u)$
Rechteck-Kern	$K(u) = \frac{1}{2}\mathbb{I}_{\{-1,1\}}(u)$
Dreieck-Kern	$K(u) = (1-\lvert u\rvert)\mathbb{I}_{\{-1,1\}}(u)$
Bisquare-Kern	$K(u) = \frac{15}{16}(1-u^2)\mathbb{I}_{\{-1,1\}}(u)$

Wobei für $\mathbb{I}_{\{-1,1\}}(u)$ gilt:

$$\mathbb{I}_{\{-1,1\}}(u) = \begin{cases} 1, & ,\text{falls } x \in \{-1,1\}, \\ 0 & ,\text{falls } x \notin \{-1,1\} \end{cases} \tag{3.4}$$

(vgl. Fahrmeir, Künstler u. a. 2011, S. 100)

Def. 3.7 Lokale Schätzung von $\hat{f}$ im Punkt x: Die Schätzung von $\hat{f}$ im Punk x des Merkmals X mit den observierten Werten $\{x_1, \dots x_n\}$,dem Kern K und der gewählten Bandbreite $d \in \mathbb{R}$ ergibt sich nach folgendem gewichteten Mittel über alle Beobachtungen:

$$\hat{f}(x) = \frac{1}{n \cdot d} \sum_{i=1}^{n} K\Big(\frac{x - x_i}{d}\Big)$$

$K\Big(\frac{x-x_i}{h}\Big)$ ist als das Gewicht der i-ten Beobachtung des Merkmals X bei der lokalen Schätzung im Punkt x zu interpretieren. Wie man in Def. 3.7 sehen kann sind die meisten frequent verwendeten Kerndichten nur zwischen -1 und 1 definiert, womit alle Beobachtungen die einen größeren Abstand als 1 von x keine Rolle spielen. Mithilfe der gewählten Bandbreite d kann jedoch dieser Abstand verändert werden. Daher deutet eine kleine Bandbreite d bei der Schätzung von $\hat{f}$ im Punkt x auf die Verwendung von nur relativ wenigen Daten hin, die jedoch sehr nahe bei x liegen. Falls d groß ist spielen relativ viele Daten, nun jedoch auch weiter von x entfernte Observierungen, bei der Berechnung von $\hat{f}(x)$ eine Rolle (vgl. ebd., S. 101).

Trägt man nun alle errechneten Tupel $(x, \hat{f}(x))$ in ein kartesisches Koordinatensystem ein, nennt man diese Abbildung *univariate Kerndichteschätzung* (Abbildung 4).

3.1.2 Analyse von Daten anhand einer Scatterplot-Matrix

Um einen ersten Überblick aller Merkmale in einem Datensatz zu erhalten, ist eine Scatterplot-Matrix sehr hilfreich. In *interVisu* ist eine interaktive Scatterplotmatrix implementiert, welche sich mit dem Befehl `Scatterplot_Matrix(data, metr_data = F, width = c(400, 700, 400), height = c(500, 700, 500))` aufrufen lässt. In dem User-Interface wählt man

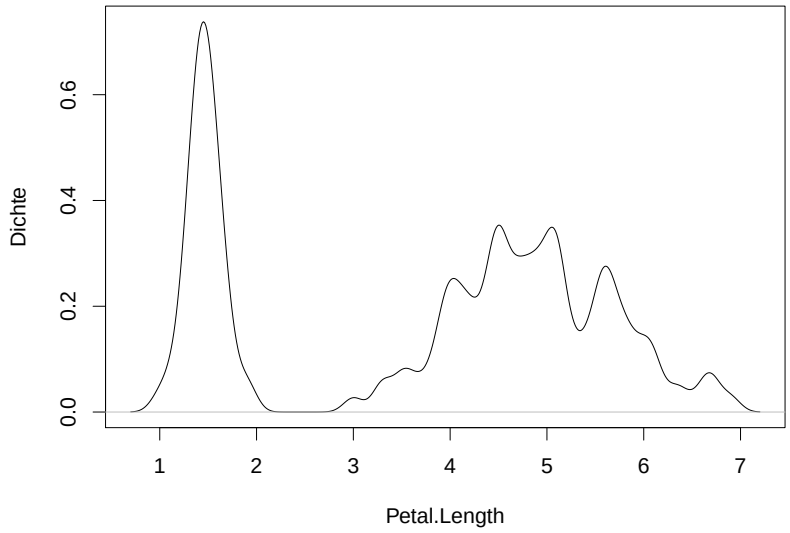

Abbildung 4: Kerndichteschätzer des Merkmals "petal.length" aus dem Datensatz Iris mit d =0.1 und Gauß-Kern

sich zunächst eine Selektion an Merkmalen im Datensatz. Sobald mindestens zwei Variablen ausgewählt wurden, wird eine Scatterplot-Matrix der ausgewählten Variablen geplottet. Zusätzlich zu dieser Darstellung kann der Anwender zwei Scatterplots aus der Matrix gleichzeitig vergrößern. Durch einen einfachen und einen doppelten Mausklick auf eine Merkmalskombination werden zwei einfache Scatterplots links und rechts der Matrix dargestellt.
Kategoriale Variablen in dem Datensatz werden in metrische Variablen umkodiert. Dabei steht der numerische Wert 1 für das erste Faktorlevel et cetera. Der Nutzer kann anhand des logischen Wertes von `metr_data` festlegen , ob die jeweiligen Analogien der metrischen und kategorialen Ausprägungen der metrisierten Merkmale angezeigt werden sollen. Die Größe der drei zu plottenden Graphen wird durch zwei dreidimensionale numerische Vektoren bestimmt. Mit dem Parameter `width` wird die Breite der Graphen von links nach rechts bestimmt, mit `height` die Höhe.
Mithilfe dieser Funktion kann man in einem der einfachen Scatterplots durch die *Brush*-Interaktion bestimmte Punkte-Wolken isolieren. Diese werden in allen Plots rot eingefärbt. Somit kann man im Zusammenspiel mit der Scatterplot-Matrix auch mehrdimensionale Zusammenhänge entdecken. Zusätzlich lassen sich auch all diese exkludierten Punkte in der Datenmatrix einzeln betrachten.
Ebenso gibt es die Möglichkeit in alle Scatterplots lineare oder loess Regressionsgeraden einzuzeichnen. Wie diese Geraden und die angezeigten Plots sich theoretisch ergeben, soll also nun besprochen werden.

Methoden

Ein Scatterplot ist in der deutschsprachigen Literatur auch als Streudiagramm zu finden und die wohl einfachste Darstellung von zwei metrischen Merkmalen. Geht man von zwei Merkmalen X und Y mit der Menge an n beobachteten Ausprägungen $\{x_1, \dots x_n\}$ und $\{y_1, \dots y_n\}$ aus, lassen sich diese Beobachtungen auch als eine Menge von Tupeln $\{(x_1, y_1) \dots (x_n, y_n)\}$ beschreiben. Zeichnet man diese Tupel in ein zweidimensionales kartesisches Koordinatensystem ein, ergibt sich der Scatterplot der beiden Merkmale X und Y (vgl. Fahrmeir, Künstler u. a. 2011, S. 128f).

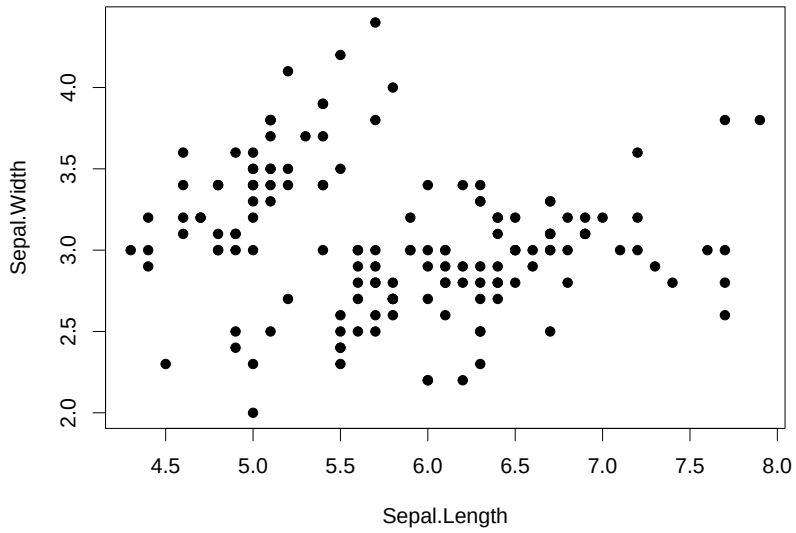

Abbildung 5: Scatterplot der X Variable "Sepal.Length" und der Y Variable "Sepal.Width" aus dem Datensatz Iris

Die Scatterplot-Matrix ist eine Matrix, deren Einträge alle Scatterplots sind, die bei einem Datensatz mit m Merkmalen betrachtet werden können. Falls man also von m metrischen Variablen $X_1, \dots X_m$ und n Beobachtungen ausgeht, ergibt sich die Scatterplot-Matrix S auf folgende Weise:

$$\begin{pmatrix} s_{1,1} & \dots & s_{1,m} \\ \vdots & \ddots & \vdots \\ s_{m,1} & \dots & s_{m,m} \end{pmatrix}$$

Wobei $s_{i,j}$ mit $i, j = 1, \dots, m$ ein einfacher Scatterplot der Merkmale X_i und X_j ist (Abbildung 6). Ein Scatterplot der Variablen X und Y kann Aufschluss über deren Zusammenhänge geben. Es lassen sich auch Linien zur Schätzung und Prognose dieser Abhängigkeiten in den Graphen einzeichnen. Die einfachste Variante ist die Schätzung einer Geraden in das Koordinatensystem mit minimalen Abständen zu den eingezeichneten Werten $(x_1, y_1) \dots (x_n, y_n)$. Dieses Verfahren nennt man *lineare Einfachregression* (vgl. ebd., S. 153f).

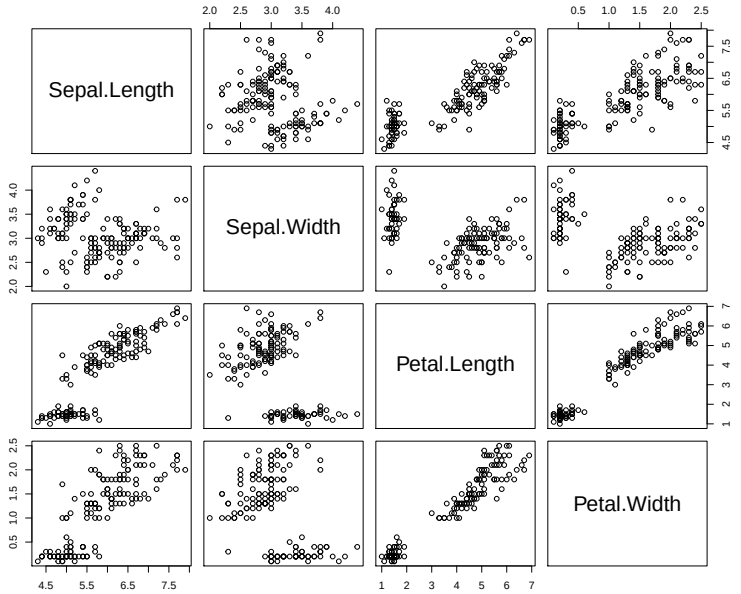

Abbildung 6: Scatterplot-Matrix der X Variable "Sepal.Length" und der Y Variable "Sepal.Width" aus dem Datensatz Iris

Um diese Gerade einzeichnen zu können, wird ein Modell zur Prognose von Y[12] durch X[13] aufgestellt. Man geht von einem linearen Zusammenhang zwischen den beiden Variablen aus, kann die gewünschte Gerade mit y als Beobachtung des Merkmals Y und x des Merkmals X parametrisch dargestellt werden.

$$y = \beta_0 + \beta_1 \cdot x \tag{3.5}$$

Die beiden Koeffizienten β_0 und β_1 sollen nun anhand von n Datenpunkten $\{(x_1, y_1) \dots (x_n, y_n)\}$ geschätzt werden. Interpretatorisch betrachtet kann β_0 als der Y-Achsen-Abschnitt und β_1 als die Steigung der Gerade verstanden werden. Da dieser Zusammenhang in der Realität meist nicht perfekt zu beobachten ist, gilt die Gleichheit von (3.5) nur, wenn wir einen zufälligen Fehlerterm ε zum Ausgleich einführen. Die Gerade lässt sich nun für die i-te Beobachtung als (3.6) darstellen (vgl. Fahrmeir, Kneib und Lang 2007, S. 21 f).

$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i \tag{3.6}$$

Für diese zufällige Fehlerterme ε_i sollen für $i = 1, \dots, n$ folgende Voraussetzung gelten:

$$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2) \tag{3.7}$$

[12] Wird in der Literatur oft abhängige Variable genannt.
[13] Wird in der Literatur oft unabhängige Variable genannt.

Dieser zufällige Anteil des Modells wird nun bei der Schätzung von β_0 und β_1 durch $\hat{\beta}_0$ und $\hat{\beta}_1$ quadratisch minimiert. Es ergeben sich folgende Schätzer[14]:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \cdot \bar{x} \tag{3.8}$$

$$\hat{\beta}_1 = \frac{S_{xy}}{S_y} \tag{3.9}$$

Wobei für die n Ausprägungen von X $\{x_1, \ldots x_n\}$ und Y $\{y_1, \ldots y_n\}$

$$\bar{y} = \sum_{i=1}^{n} y_i,\ \bar{x} = \sum_{i=1}^{n} x_i,\ S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}),\ S_x = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

gilt (vgl. Fahrmeir, Künstler u. a. 2011, S. 159).

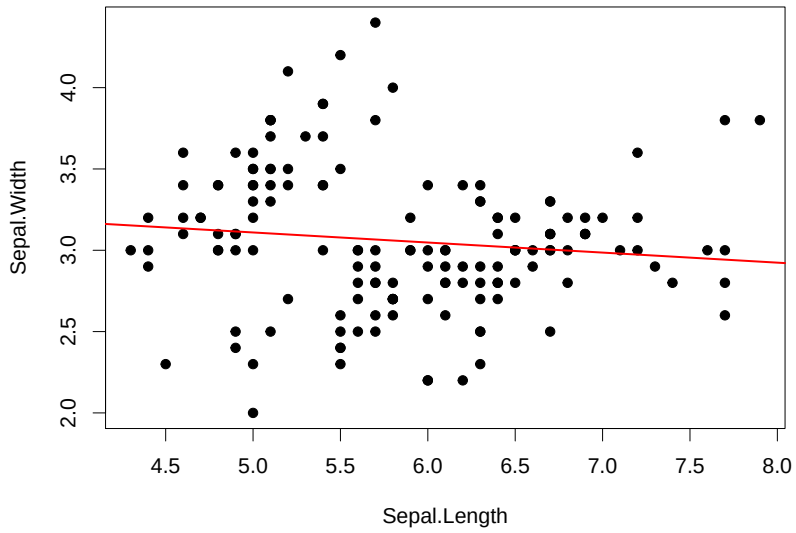

Abbildung 7: Scatterplot der X Variable "Sepal.Length" und der Y Variable "Sepal.Width" aus dem Datensatz Iris mit geschätztem linearem Modell

Der Zusammenhang von zwei metrischen Merkmalen kann jedoch auch nicht linear sein. Ähnlich der bereits in 3.1.1 beschriebenen Kerndichteschätzung lässt sich eine Linie in dem Scatterplot auch lokal schätzen. Dieser Ansatz wird Loess-Regression genannt (vgl. W. Cleveland und Devlin 1988, S. 596). Nun lässt sich für das Modell keine klare Parametrisierung mehr finden, womit die Modellgleichung für die i-te Beobachtung nur mithilfe einer lokal geschätzten Funktion $f(x_i)$ darzustellen ist.

$$y_i = f(x_i) + \varepsilon_i$$

[14] Die detaillierte Herleitung der Schätzer ist im Anhang C 1 zu finden.

Die Annahmen gegenüber des Fehlerterms ε_i sind identisch zur bereits beschriebenen linearen Regression.
Geschätzt wird f durch $\hat{f}$ in einem Raster an vorgegebenen Punkten ähnlich zu Def. 3.8. Im Punkt x wird $\hat{f}(x)$ gleich einem nach einer Kernfunktion gewichteten arithmetischen Mittel der q benachbarten Observationen um den Wert x gesetzt (vgl. ebd., S. 596).

Def. 3.8 q-Umgebung von x: Seien n Ausprägungen $\{x_1, \ldots, x_n\}$ des metrischen Merkmals X gegeben, so ist die Umgebung von x die Untermenge der q Ausprägungen von X, die sich nach der euklidischen Norm am nächsten bei x befinden. Die q-Umgebung des Punktes x bei den Beobachtungen des Merkmals X wird als als $\mathbb{U}_{x,q}(X)$ notiert und es muss $q \in \mathbb{N}$ gelten.

Daher lässt sich die Funktion f durch $\hat{f}$ im Punkt x mit der Gewichtsfunktion K und $x_{(q)}$, als weitest entfernte Beobachtung von x in $\mathbb{U}_{x,q}(X)$, schätzen.

$$\hat{f}(x) = \frac{1}{q} \sum_{x_i \in \mathbb{U}_{x,q}(X)} W\left(\frac{|\, x - x_i \,|}{|\, x - x_{(q)} \,|}\right) \tag{3.10}$$

Als Gewichtsfunktion K wird meist die Trikubische-Funktion verwendet (vgl. ebd., S. 596).

$$K(u) = \begin{cases} (1-u)3 & \text{, falls } 0 \leq u < 1 \\ 0 & \text{, sonst} \end{cases}$$

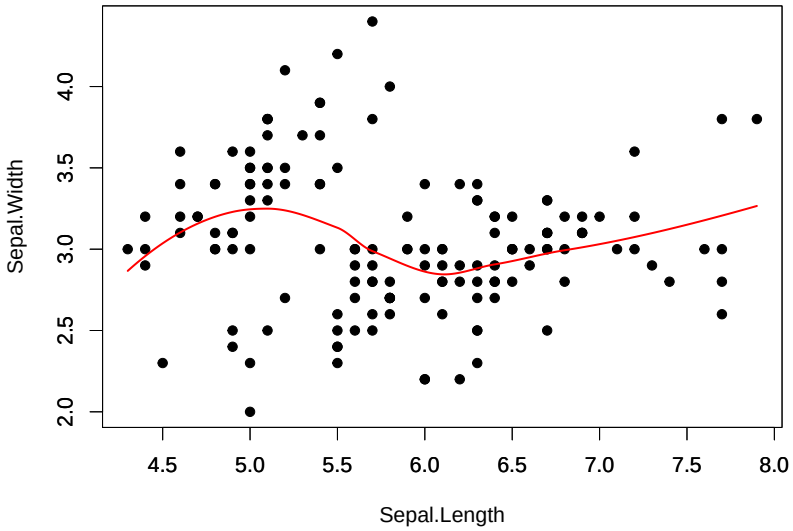

Abbildung 8: Scatterplot der X Variable "Sepal.Length" und der Y Variable " Sepal.Width" aus dem Datensatz Iris mit geschätztem loess Modell

3.1.3 Analyse von Zusammenhängen zweier metrischer Variablen

Zusammenhänge zwischen zwei metrischen Variablen können jedoch auch nicht lokal oder linear geschätzt werden. Um generellere Abhängigkeiten zwischen metrischen Variablen modellieren zu können, gibt es die Applikation `Smoothing_Analysis(data, n = 10, height = 500)`. Hierbei kann ein Anwender eine univariate Regression mit ausgewählten Merkmalen des übergebenen Datensatzes sich interaktiv erarbeiten, und die Prognose und Konfidenzbänder dieser zum Niveau α in einem Scatterplot betrachten. Der Parameter `height` beschreibt die Höhe des Scatterplots in Pixeln.
Kraft dieser Funktion kann man jegliche Parameter, die mit der Modellierung des Zusammenhangs eine Rolle spielen, interaktiv bestimmen. Durch Ausprobieren wird die Suche der besten Modellierung daher immens erleichtert und da auch das Bestimmtheitsmaß R^2 angezeigt wird, können Modelle bezüglich der Güte miteinander verglichen werden. Zuletzt können pädagogisch betrachtet diese Parameter leichter in einem Anwendungsbeispiel verstanden werden und verschiedene nun auch generalisierte Arten der Regression miteinander in Bezug gesetzt werden.

Methoden

Bis jetzt galten im einfachen linearen Modell zur Erklärung des Merkmals Y mit n Realisationen (x_i, y_i) mit $i = 1, \ldots, n$ nach den Annahmen (3.11) und (3.12).

$$\mathbb{E}(y_i|x_i) = \beta_0 + x_i\beta_1 = \eta_i = \mathbb{I}(\eta_i) \tag{3.11}$$

Für die Identität $\mathbb{I}(\eta_i)$ gelte:

$$\mathbb{I}(\eta_i) = \eta_i$$

$$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2) \tag{3.12}$$

Daraus folgt für die Realisation x von X:

$$Y \mid x \sim N(\beta_0 + \beta_1 x, \sigma^2) \tag{3.13}$$

Generell müssen die Annahmen, dass Y bei gegebener Realisation von X normalverteilt ist und, dass sich der lineare Prädiktor η_i über die Identität- also direkt- auf $\mathbb{E}(y_i|x_i)$ auswirkt, nicht gelten. Weicht man diese Annahmen auf, befindet man sich bei generalisierten linearen Modellen, deren linearer Bestandteil immer im linearen Prädiktor η_i dargestellt wird (vgl. Fahrmeir, Kneib und Lang 2007, S. 189 ff).
Um diese unterschiedlichen Annahmen theoretisch beschreiben zu können, werden nun die Begriffe *Link-Funktion* und *Repsonse-Funktion* eingeführt.

Def. 3.9 ***Link- und Repsonse-Funktion*:** In einem generalisierten bivariaten linearen Model mit Zielvariable Y und erklärender Variable X mit n Beobachtungen (x_i, y_i),wobei $i = 1, \ldots, n$, wird

der Zusammenhang zwischen η_i und $\mathbb{E}(y_i|x_i)$ mit der Link-Funktion g und der Responsefunktion h beschrieben. Wobei nun nicht mehr $\eta_i = \beta_0 + \beta_1 x_i$, sondern

$$\mathbb{E}(y_i|x_i) = h(\eta_i) = h(\beta_0 + \beta_1 x_i)$$
$$\eta_i = g(\mathbb{E}(y_i|x_i))$$

gilt (vgl. ebd., S. 221).

Es wird nun also eine Brückenfunktion zwischen dem linearen Teil einer Funktion η_i und $\mathbb{E}(y_i|x_i)$, also dem Erwartungswert für y_i bei gegebenem x_1, als *Link-Funktion* definiert.
Auch $Y \mid x$ muss wie in (3.13) nicht mehr unbedingt normalverteilt sein. Es kann jegliche Verteilung angenommen werden, unter der Bedingung, dass die Verteilung der Exponentialfamilie angehört. Für eine detailierte Beschreibung dieser Verteilungen sei auf Fahrmeir, Kneib und Lang 2007, S. 217 ff verwiesen. Diese Annahme sollte bei der Modellierung von speziellen Zielvariablen passend gewählt werden: Beispielsweise liegt bei Zähldaten die Annahme $Y \mid x \sim Pois(\lambda)$ oder binäre Variablen die Annahme $Y \mid x \sim B(n, \pi)$ nahe.
Betrachten wir nun die Schätzung des Zusammenhangs. Generell lässt sich jede Modellierung der Zielvariable Y durch X mit n Beobachtungen (x_i, y_i), für $i = 1, \ldots, n$ als eine Funktion abhängig von x_i beschreiben.

$$y_i = f(x_i) + \varepsilon_i \tag{3.14}$$

Beschreibt man das in 3.1.2 vorgestellte lineare Modell in diesem Kontext, lässt sich die von x_i abhängige Funktion f auch explizit benennen:

$$f(x_i) = \beta_0 + \beta_1 x_i$$

$f(x_i)$ ist in der linearen einfach Regression ein Polynom ersten Grades. Um mehr Freiheit bei der Erklärung von Y zu erhalten, kann man $f(x_i)$ als Polynom r-ten Grades schätzen. $f(x_i)$ lässt sich weiterhin parametrisch darstellen.

$$f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_r x_i^r + \varepsilon_i = \beta_0 + \sum_{j=1}^{r} \beta_j x_i^j + \varepsilon_i$$

Diese Art der Regression wird Polynomialregression genannt. Im Kontext der generalisierten Regression soll nun mit *Vert* als eine beliebige Verteilung der Exponentialfamilie und einer beliebig gewählten *Link-Funktion h*, die Annahmen (3.15) und (3.16) gelten.

$$\varepsilon_i \overset{iid}{\sim} Vert(\mu = 0, \sigma^2) \tag{3.15}$$

$$Y \mid x \sim Vert(h(f(x_i)), \sigma^2) \tag{3.16}$$

Bei der Schätzung gehen wir jedoch der Einfachheit halber von $h = \mathbb{I}$ aus. Für die explizite Schätzung mit beliebigen *Link-Funktonen* sei auf das Kapitel 4.1.2 in ebd., S. 189 ff verwiesen. Das Modell mit Polynom des Grades r mit beschreibender Variable X kann auch als ein multiples Regressionsmodell mit r Einflussvariablen $X^1, X^2, \ldots, X^r$ betrachtet werden. Daraus ergeben sich $r+1$ zu schätzende Parameter. Das gesamte Modell mit allen n Beobachtungen wird zur besseren Übersichtlichkeit mithilfe einer Matrixmultiplikation dargestellt.

$$f(X) + \varepsilon = \mathbf{X}\beta + \varepsilon = y \tag{3.17}$$

Wobei gilt:

$$f(X) = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_1 & \ldots & x_1^r \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^r \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_r \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Der Schätzer $\hat{\beta}$ ergibt sich wie bei dem linearen Regressionsmodell nach der Kleinsten-Quadrate-Methode mit n Beobachtungen $(x_1, y_1), \ldots, (x_n, y_n)$ nach[15]:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \tag{3.18}$$

Abhängigkeiten zwischen zwei metrischen Variablen können auch durch univariate Glättung mit Polynom-Splines modelliert werden.

Def. 3.10 Polynom-Splines: Eine unbestimmte Funktion f, für die $f : [a,b] \to \mathbb{R}$ gilt, heißt Polynom-Spline vom Grad $l \geq 0$ mit m Knoten $a = \kappa_1 < \ldots < \kappa_m = b$, falls f $(l-1)$-mal stetig differenzierbar ist und f in den Intervallen zwischen den Knoten $[\kappa_j, \kappa_{j+1}) \; \forall j = 1, \ldots, m$ ein Polynom des Grads l ist (vgl. ebd., S. 295).

Nun will man die unbekannte Funktion f durch Polynom-Splines modellieren. Generell stellen wir die Funktionen durch Linearkombinationen von d Basisfunktionen dar, womit sich das Modell wieder für die i-te Beobachtung wie in (3.14) schreiben lässt.

$$y_i = f(x_i) + \varepsilon_i = \sum_{j=1}^{d} \beta_j B_j(x_i) + \varepsilon_i \tag{3.19}$$

Die gewählten Basisfunktionen entscheiden welche Art von Polynom-Splines gefittet werden. Die drei meist verwendeten Basisfunktionen sind die *Basis der trunkierten Potenzen*, die *B-Spline Basis* und die *kubische Basis*.

Def. 3.11 *Basis der trunkierten Potenzen*: Mit den gewählten Knoten $\kappa_1, \ldots, \kappa_m$, $l \geq 0$ und

$$(x - \kappa_j)_+ = \begin{cases} (x - \kappa_j) & , \text{falls} (x - \kappa_j) > 0 \\ 0 & , \text{falls} (x - \kappa_j) \leq 0 \end{cases}$$

[15] Eine detaillierte Herleitung von $\hat{\beta}$ ist im Anhang C 2 zu finden.

sind die Basisfunktionen $B_j(x)$ mit $j = 1,\ldots,d$ auf folgende Art definiert.

$$B_1(x) = 1, B_2(x) = x, \ldots, B_{l+1}(x) = x^l,$$
$$B_{l+2}(x) = (x-\kappa_2)_+^l, \ldots, B_d(x) = (x-\kappa_{m-1})_+^l$$

(vgl. ebd., S. 297)

Diese Basis erweitert das Polynomial-Modell des Grades l um weitere $m-2$ trunkierte Terme.

Def. 3.12 *B-Spline Basis*: Mit den gewählten Knoten $\kappa_1,\ldots,\kappa_m$ und $m+2=d$ wird die *B-Spline Basis* des Grades $l \geq 0$ rekursiv definiert. B_j^l beschreibt nun die j-te Basisfunktion des Grades l und ist folgendermaßen definiert.

$$B_j^1(x) = \frac{x-\kappa_j}{\kappa_{j+1}-\kappa_j}\mathbb{I}_{[\kappa_j,\kappa_{j+1})}(x) + \frac{\kappa_{j+2}-x}{\kappa_{j+2}-\kappa_{j+1}}\mathbb{I}_{[\kappa_{j+1},\kappa_{j+2})}(x)$$
$$B_j^l(x) = \frac{x-\kappa_j}{\kappa_{j+l}-\kappa_j}B_j^{l-1}(x) + \frac{\kappa_{j+l+1}-x}{\kappa_{j+l+1}-\kappa_{j+l}}B_{j+1}^{l-1}(x)$$

(vgl. ebd., S.304 f)

Def. 3.13 *Kubische Basis*: Mit den gewählten Knoten $\kappa_1,\ldots,\kappa_m$, $m+2=d$ und $l=3$ kann die *Kubische-Basis* auf viele verschiedene Arten dargestellt werden, eine mögliche ist:

$$\begin{aligned} B_1(x) =& 1,\ B_2(x) = x \text{ und } B_{j+2} = R(x,\kappa_j) \text{ für } j = 1,\ldots,m \text{ mit} \\ R(x,\kappa_j) =& \left((\kappa_j - \tfrac{1}{2})^2 - \tfrac{1}{12}\right)\left((x-\tfrac{1}{1})^2 - \tfrac{1}{12}\right)/4 \\ & - \left((|x-\kappa_j| - \tfrac{1}{2})^4 - \tfrac{1}{2}(|x-\kappa_i| - \tfrac{1}{2})^2 + \tfrac{7}{240}\right)/24 \end{aligned}$$

(vgl. Woods 2006, S.124 ff)

Für eine Diskussion der spezifischen Vor- und Nachteile der einzelnen Basen sei Kapitel 7 in Fahrmeir, Kneib und Lang 2007, S.291 ff angemerkt.
Unabhängig von den gewählten Basisfunktionen lässt sich die Beschreibung des Merkmals Y durch X mit n Beobachtungen $(x_1,y_1),\ldots,(x_n,y_n)$ und den Knoten $\kappa_1,\ldots,\kappa_m$ durch einen Polynom-Spline des Grades l auch wie (3.17) in Matrixschreibweise wie folgt darstellen.

$$y = \mathbf{B}\beta + \varepsilon \tag{3.20}$$

Nun jedoch gilt:

$$\mathbf{B} = \begin{pmatrix} B_1(x_1) & \ldots & B_d(x_1) \\ \vdots & & \vdots \\ B_1(x_n) & \ldots & B_d(x_n) \end{pmatrix}, \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_d \end{pmatrix}, \quad \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

Die Schätzung $\hat{\beta}$ erhält man analog zu (3.18).

$$\hat{\beta} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T y$$

Unter Umständen führen zu viele Schätzer $\hat{\beta}_1, \ldots, \hat{\beta}_d$ zu einer unruhigen Prognose. Diese kann man jedoch bei der Schätzung der Polynom-Splines durch eine Penalisierung eliminieren (vgl. ebd., S. 306 ff).
Bei der Methode der kleinsten Quadrate wird die Quadratsumme der Residuen nun durch einen additiven von f abhängigen Penalisierungsterm K erweitert. Bei der Schätzung von β wird nicht mehr (3.21) sondern (3.22) mit $\lambda > 0$ minimiert.

$$f(\beta) = \sum_{i=1}^{n} \varepsilon_i^2 \tag{3.21}$$

$$f(\beta) = \sum_{i=1}^{n} \varepsilon_i^2 + \lambda K(f) \tag{3.22}$$

Der verwendete Penalisierungsterm K ist folgendermaßen definiert:

$$\int (f''(z))^2 dz$$

Auf diesem Weg ergibt sich der neue penalisierte Schätzer β mit der Designmatrix $\mathbf{X}$ und Strafmatrix $\mathbf{K}$ nach:

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{K})^{-1}\mathbf{X}^T y \tag{3.23}$$

Für $\mathbf{K}$ bei $K = \int (f''(z))^2 dz$ gilt:

$$\begin{aligned}
\int (f''(z))^2 dz &= \int \big(\sum_{i=1}^{d} \beta_i B_j''(z)\big)^2 dz \\
&= \int \big(\sum_{i=1}^{d}\sum_{j=1}^{d} \beta_i\beta_j B_i''(z)B_j''(z)\big) dz \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d} \beta_i\beta_j \int B_i''(z)B_j''(z) dz \\
&= \beta^T\mathbf{K}\beta
\end{aligned}$$

(vgl. ebd., S. 311 ff)[16]
Zuletzt können bei allen nun vorgestellten Verfahren der Modellierung neben der prognostizierten Funktion $\hat{f}$ auch Konfidenzbänder zu dem Niveau α in den Scatterplot eingezeichnet werden. Interpretieren kann man diese beiden Linien als Bereich, in welchem die echte Gerade oder Linie in $1-\alpha$ Prozent der Wiederholungen von unabhängigen Stichproben der gleichen Grundpopulation liegt.
Für die Herleitung dieser Bänder gehen wir generell von $Y \mid x \sim Vert(\mu = \hat{f}(x), \sigma^2)$ aus, wobei *Vert* Teil der Exponentialfamilie sein muss. Im univariaten Fall gilt nun für jede Beobachtung x_i

[16] Eine detaillierte Herleitung von $\hat{\beta}$ ist im Anhang C 3 zu finden.

mit $i = 1, \ldots, n$:

$$Var(\varepsilon_i) = \sigma^2$$
$$\varepsilon_i \overset{iid}{\sim} Vert(\mu = 0, \sigma^2)$$

$\hat{f}(x)$ kann auch mithilfe einer *Smoother Matrix* $s(x)$ folgendermaßen dargestellt werden:

$$\hat{f}(x) = s(x)^T y$$

Die Form der Matrix $s(x)$ hängt von der gewählten Modellierungsmethode ab. In einem polynomialen Modell gilt für eine explizite Beobachtung x:

$$\hat{f}(x) = \mathbf{X}\beta \overset{(3.18)}{=} \underbrace{x^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}}_{s(x)^T} y$$

Wird die Zielvariable durch Polynom-Splines modelliert ergibt sich für $s(x)^T$ die gleiche Form, nur wird statt $\mathbf{X}$ nach (3.20) $\mathbf{B}$ und statt x nach (3.24) $b(x)$ verwendet. In dem Fall, dass diese auch penalisiert werden, gilt für $s(x)^T$:

$$\hat{f}(x) = \mathbf{B}\beta \overset{(3.23)}{=} \underbrace{b(x)^T(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})^{-1}\mathbf{B}}_{s(x)^T} y$$

Wobei $b(x)$ mit den Basisfunktionen $B_1, \ldots B_d$ folgendermaßen definiert wird (vgl. ebd., S. 340 ff):

$$b(x) = \begin{pmatrix} B_1(x) \\ \vdots \\ B_d(x) \end{pmatrix} \tag{3.24}$$

Mit dieser Notation lässt sich $Var(\hat{f}(x)) = \sigma^2 s(x)^T s(x)$ schreiben. Das Konfidenzintervall von $\hat{f}(x)$ zum Niveau α ist für den gesamten Wertebereich von X definiert durch

$$\left[\hat{f}(x) - z_{1-\frac{\alpha}{2}}\sigma\sqrt{s(x)^T s(x)},\ \hat{f}(x) + z_{1-\frac{\alpha}{2}}\sigma\sqrt{s(x)^T s(x)}\right] \tag{3.25}$$

, wobei $z_{1-\frac{\alpha}{2}}$ das $(1-\frac{\alpha}{2})$-Quantil der Standardnormalverteilung ist (vgl. ebd., S. 342 ff).
Um verschiedene Modelle mit gleichen Einfluss- und Zielvariablen vergleichen zu können wird das Bestimmtheitsmaß verwendet.

Def. 3.14 Bestimmtheitsmaß R^2: Das Bestimmtheitsmaß R^2 eines Modells von der Variable X auf Y gibt den durch durch das Modell erklärten Anteil der Varianz von Y an (vgl. ebd., S. 99).

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Je näher das Bestimmtheitmaß eines Modells an 1 liegt, desto besser ist die Anpassung an die Daten (vgl. Fahrmeir, Kneib und Lang 2007, S. 98).

3.1.4 Analyse von Daten anhand eines dreidimensionalen Scatterplots

Die Präsentation von zwei Merkmalen in einem Koordinatennetz wie in 3.1.2 beschrieben, lässt sich noch um eine Dimension erweitern. Dies wird durch die Tiefe des Netzes erreicht. Diese Darstellung wird dreidimensionaler Scatterplot genannt, womit dreidimensionale Interaktionen von metrischen Merkmalen räumlich zu betrachten sind.
In dem Paket *interVisu* können Daten mithilfe dieser Darstellung in der Funktion `Scatterplot_3d(data, n, height = c(800, 500))` erforscht werden. Zusätzlich wird ein zweidimensionaler Scatterplot der ersten beiden Dimensionen des dreidimensionalem Scatterplots angezeigt, daher muss der Parameter `height` ein zweidimensionaler numerischer Vektor sein.
Alle angezeigten Scatterplots können zusätzlich auf eine selbst gewählte Ausprägung u einer weiteren Variable U bedingt werden, um somit nur eine bestimmte Anzahl an Beobachtungen um diesen Wert u zu plotten. Wählt man diesen Modus der Darstellung, werden nur 30 Beobachtungen (x_i, y_i, z_i) in den Graphen angezeigt, für die mit $i = 1, \ldots, n$ und u_i als Ausprägung des Merkmals U mit Def. 3.8

$$u_i \in \mathbb{U}_{u,q=30}(U)$$

gilt.

Methode

Ausgehend von drei metrischen Merkmalen X, Y und Z mit den Mengen an je n Ausprägungen $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$ und $\{z_1, \ldots, z_n\}$ zeichnet man die Tupel $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$ in ein dreidimensionales Koordinatensystem. Dabei entsteht ein dreidimensionaler Scatterplot, auch dreidimensionales Streudiagramm genannt (Abbildung 9).

3.2 Kategoriale Variablen

3.2.1 Analyse von gruppierten Boxplots

Zur Vorüberlegung oder visuellen Prüfung der Voraussetzungen einer einfaktoriellen Varianzanalyse werden oft nach der Faktorvariablen gruppierte Boxplots der Zielvariable betrachtet. Somit können mögliche Mittelwertunterschiede und die Varianzhomogenität der mit der Faktorvariablen definierten Gruppen graphisch untersucht werden.
Um diese Boxplots von beliebigen metrischen Zielvariablen und Faktorvariablen ohne großen Aufwand plotten zu können, gibt es in *interVisu* die Funktion `Group_Boxplot(data, n = 10, width = 600, height = 600)`. Mittels der Parameter `width` und `height` kann die explizite Breite und Höhe des angezeigten gruppierten Boxplots in der Anzahl an Pixeln angegeben werden.

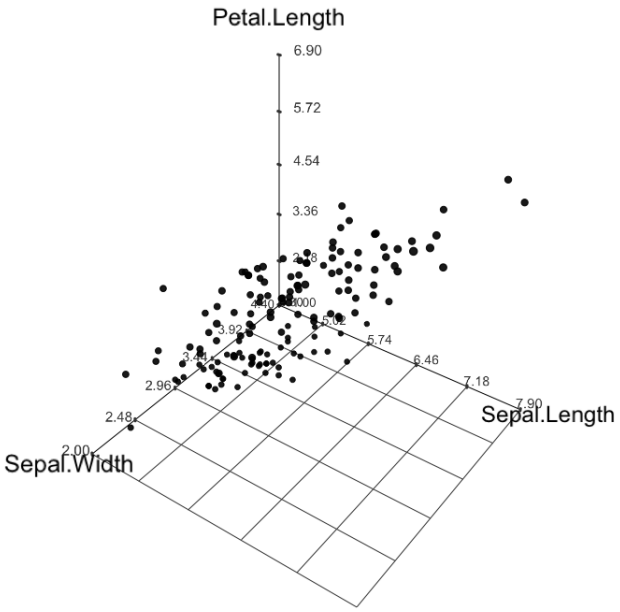

Abbildung 9: Ein dreidimensionaler Scatterplot der X Variable "Sepal.Length", Y Variable "Petal.Length" und der Z Variable "Sepal.Width" aus dem Datensatz Iris

Nachdem die Anwendung alle Merkmale in kategoriale und metrische Merkmale eingeordnet hat, kann man interaktiv auswählen, welche metrische und kategoriale Variable in der Graphik repräsentiert werden soll.
Oft müssen aufgrund von zu wenigen Beobachtungen oder fehlenden Voraussetzungen der Varianzanalyse bestimmte Level zusammengelegt werden. Dieses Zusammenfassen von mehreren Faktorlevel ist in der Applikation möglich.

Methoden

Der nach einer Faktorvariablen gruppierte Boxplot ist eine Erweiterung des bereits in 3.1.1 beschriebenen einfachen Boxplots. Wir gehen von der Präsentation des metrischen Merkmals X gruppiert nach dem kategorialen Merkmal Y mit m Faktorlevel $Y_1,\ldots,Y_m$ und der Menge an n Beobachtungen $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ aus. Die Menge von X, die man mit $X_{values} = \{x_1,\ldots x_n\}$ notiert, wird nun in m Untermengen nach den einzelnen Faktorlevel aufgeteilt. Daher lassen sich die Mengen $X_{values,j}$ an Beobachtungen mit $j = 1,\ldots,m$ folgendermaßen schreiben:

$$X_{values,j} = \{x_i | (x_i \in X_{values}) \wedge (y_i = Y_j)\}$$

Das j-te Faktorlevel wird als Y_j, mit $j \in 1,\ldots,m$ notiert.
Nun werden auf der X-Achse des Koordinatensystems m einzelne Boxplots der beobachteten Wertemengen $X_{values,1}$ bis $X_{values,m}$ nebeneinander eingezeichnet. Die Beschriftung der X-Achse wird gleich der Namen der jeweiligen Faktorlevel gewählt (Abbildung 10).

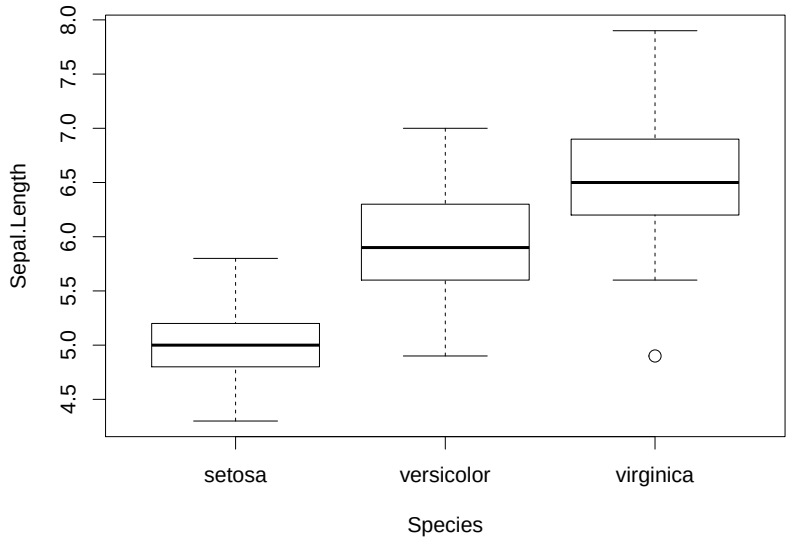

Abbildung 10: Ein gruppierter Boxplot der metrischen Variable "Sepal.Length" und der kategorialen Variable "Species" aus dem Datensatz Iris

3.2.2 Gruppierung kategorialer Variablen mit gestapelten Balkendiagrammen

Die zweite Anwendung, welche sich primär der Analyse kategorialer Variablen widmet, untersucht die Verteilung einer kategorialen Variable bedingt auf die Ausprägung eines weiteren kategorialen Merkmals. Die Variablen werden in einem auf 1 normierten gestapelten Balkendiagramm dargestellt. Mit dem Befehl `Stacked_Barplot(data, n = 10, m = 5, height = c(500, 300))` wird die Funktion abgerufen. Da die bedingten Verteilungen auch als Kontingenztafel dargestellt werden können, gibt es die Möglichkeit, auf Unabhängigkeit der beiden kategorialen Variablen zu testen. Ob ein approximativer oder exakter Test in der Anwendung durchgeführt wird, hängt von den observierten Zellhäufigkeiten in der Kontingenztafel ab. Treten alle möglichen Merkmalskombinationen der beiden Variablen öfter als der numerische Parameter `m` auf, wird ein approximativer χ^2-Test verwendet. Andernfalls wird die Unabhängigkeit mit dem exakten Test nach Fisher geprüft. Neben dem Balkendiagramm kann man auch einen nach der zweiten kategorialen Variable gruppierten Boxplot, konditioniert auf ein Level des ersten Faktors, anzeigen lassen. Daher wird die Höhe der Darstellungen mit einem zweidimensionalen numerischen Vektor `height` angegeben.

Anhand dieser Anwendung können nun auch Zusammenhänge zwischen zwei Faktorvariablen untersucht werden. Diese Analyse beschränkt sich nicht nur auf eine klare und einfache graphische Darstellung in einem Balkendiagramm, sondern inkorporiert sowohl Tests zur Unabhängigkeit als auch die Maßzahl Odds Ratio γ, falls beide Faktoren nur zwei Level aufweisen. Wie in der

vorherigen Applikation kann man die Faktorlevel interaktiv miteinander durch einen einfachen und einen doppelten Klick fusionieren. In Zusammenhang des Boxplots kann man so Daten aus vielen konditionierten Blickwinkeln betrachten, obwohl die Handhabung der Funktion trotzdem leicht verständlich bleibt.

Methoden

Die zentrale graphische Darstellung der Applikation ist ein gestapeltes Balkendiagramm zur Repräsentation der bedingten Verteilung von zwei faktoriellen Variablen. Diese graphische Darstellung ist eine Erweiterung des herkömmlichen Balkendiagramms zur Repräsentation der relativen Häufigkeiten einer Faktorvariable X. Alle Beobachtungen dieses Merkmals können nur Werte eines Levels von X annehmen, die in Menge $X_{Levels} = \{1\ldots,n_x\}$ definiert sein. Wurden nun n Beobachtungen dieses Merkmals erhoben, kennen wir für jedes Faktorlevel j eine absolute Anzahl n_j mit $j = 1,\ldots,n_x$ an Beobachtungen, welche diesen Wert haben. Um mit diesen Informationen nun ein Balkendiagramm zu zeichnen, tragen wir zuerst auf der X-Achse alle Level der Faktorvariable X ab. Gehen wir von einer nominalen Struktur der Level untereinander aus, ist die Reihenfolge egal. Falls X jedoch ordinalskaliert ist, sollte man nach der somit gegebenen Hierarchie die Level auf der X-Achse anordnen. Für jede Stufe des Faktors X wird nun mit $j = 1,\ldots,n_x$ ein Balken mit fester nicht informativer Breite und der Höhe von $h_j = \frac{n_j}{n}$ zur Repräsentation der relativen Häufigkeit in den Graphen gezeichnet.

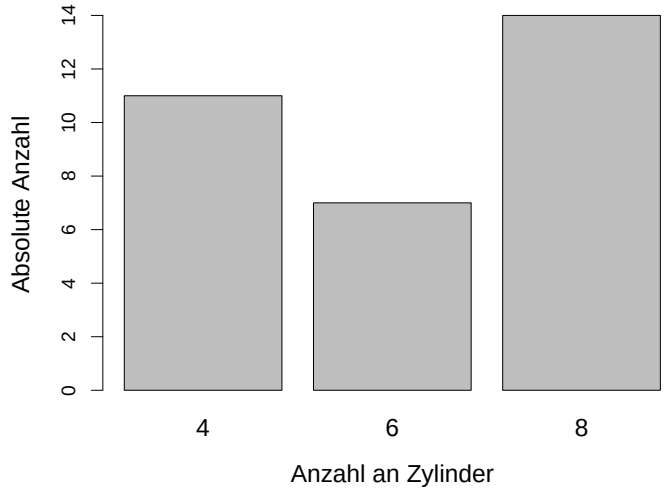

Abbildung 11: Ein Balkendiagramm der Variable "Cyl" aus dem Datensatz mtcars

Um eine weitere faktorielle Variable Y darstellen zu können, erweitert man das einfache zu einem gestapelten Balkendiagramm. Nun sei die Menge aller Stufen der diskreten Variable Y als $Y_{Levels} = \{1,\ldots,n_y\{$ definiert. Falls auch die Variable Y n mal beobachtet wurde, kann nun für

jedes Level der Variable X die Verteilung der Variable Y betrachtet werden. Die Balkenhöhe wird nicht in Bezug zu dem vorherigen Graphen verändert. Einziger Unterschied ist, dass nun jeder j-te Balken für $j = 1,\ldots,n_x$ nach den relativen Anteilen an Beobachtungen der einzelnen Level i mit $i = 1,\ldots,n_y$ des Merkmals Y eingefärbt wird. Falls nur Beobachtungen des j-ten Levels der Variable X betrachtet werden, erhält man für jedes Level des Merkmals Y einen relativen Anteil in den Beobachtungen, die wir nun mit $j = 1,\ldots,n_X$ als $p_j = \{p_{1|j},\ldots,p_{n_y|j}\}$ definieren. Hierbei ist $p_{i|j}$ die relative Häufigkeit von Beobachtungen mit Faktorlevel i der Variable Y bei gegebenem Level j von Variable X. Färbt man nun den j-ten Balken vertikal nach den Wahrscheinlichkeiten p_j mit $j = 1,\ldots,n_x$ ein und schreibt neben den Graphen eine Legende, welche Farbe für welches Level des Merkmals Y steht, resultiert daraus ein gestapeltes Balkendiagramm.

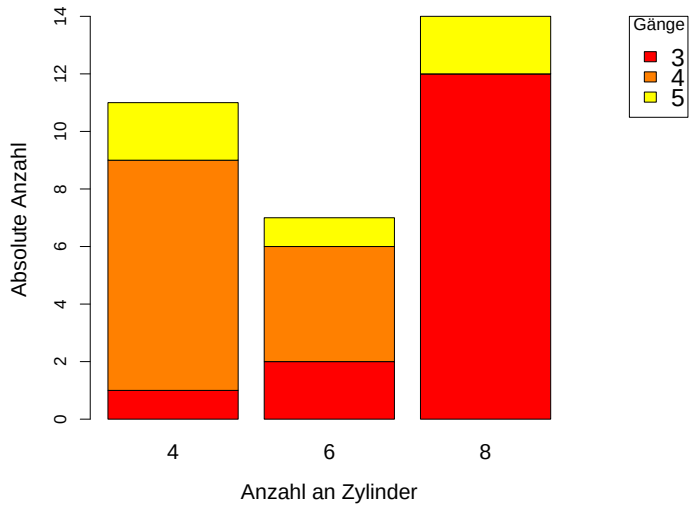

Abbildung 12: Ein gestapeltes Balkendiagramm der Variable "Cyl" nach dem Merkmal "gear" aus dem Datensatz mtcars

Dieses gestapelte Balkendiagramm wird nun auf 1 normiert, um die bedingten Verteilungen $Y \mid x_j$ mit $j = 1,\ldots,n_x$ besser über die verschiedenen Level von X vergleichen zu können. Somit wird nur noch die Verteilung von $Y \mid x_j$ mit $j = 1,\ldots,n_x$ betrachtet (Abbildung 13).
Neben der graphischen Darstellung wird auch der Output eines Tests auf Unabhängigkeit der beiden diskreten Merkmale dargestellt. Da wir zwei Merkmale X und Y mit diskreten Ausprägungen $\{1,\ldots,n_x\}$ von X und $\{1,\ldots,n_y\}$ von Y betrachten, lassen sich die Daten auch als Kontingenztafel darstellen (Tabelle 1).

Abbildung 13: Ein gestapeltes Balkendiagramm auf 1 normiert der Variable "Cyl" nach dem Merkmal "gear" aus dem Datensatz mtcars

	Y= 1	Y= 2	...	Y=n_y	$\sum$
X=1	$h_{1,1}$	$h_{1,2}$	...	h_{1,n_y}	$\sum_{i=1}^{n_y} h_{1,i} = h_{1,\cdot}$
X=2	$h_{2,1}$	$h_{2,2}$	...	h_{2,n_y}	$\sum_{i=1}^{n_y} h_{2,i} = h_{2,\cdot}$
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
X=n_x	$h_{n_x,1}$	$h_{n_x,2}$	...	n_{n_x,n_y}	$\sum_{i=1}^{n_y} h_{n_x,i} = h_{n_x,\cdot}$
$\sum$	$h_{\cdot,1}$	$h_{\cdot,2}$	...	$h_{\cdot,n_y}$	n

Tabelle 1: Kontingenztafel für die beiden diskreten Merkmale X und Y

Die beobachteten Zellhäufigkeiten werden als $h_{i,j}$ mit $i = 1,\ldots,n_x$ und $j = 1,\ldots,n_y$ in der Tabelle notiert. Unter der Nullhypothese sollte $P(X = i, Y = j) = P(X = i) \cdot P(Y = j)$ gelten, was äquivalent zu der Unabhängigkeit zwischen den Variablen ist (vgl. Fahrmeir, Künstler u. a. 2011, S. 240).

$$H_0 : \text{Merkmal } X \text{ und } Y \text{ sind unabhängig} \quad vs. \quad H_1 : \text{Merkmal } X \text{ und } Y \text{sind abhängig}$$

Da die Zahl n an Beobachtungen gegeben ist, kann man so eine unter der Nullhypothese erwartete Kontingenztabelle berechnen. Dabei gilt für die i-te Reihe und j-te Spalte der erwarteten Kontingenztafel mit $i = 1,\ldots,n_x$ und $j = 1,\ldots,n_y$:

$$\hat{\mu}_{i,j} = h_{i,\cdot} \cdot h_{\cdot,j} \tag{3.26}$$

Falls man von unabhängigen Stichproben von den beiden Variablen X und Y ausgeht, ist es möglich sich eine approximativ χ^2-verteilte Teststatistik herzuleiten.

Die Teststatistik ist mit (3.26)

$$\chi^2 = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \frac{(h_{i,j} - \hat{\mu}_{i,j})^2}{\hat{\mu}_{i,j}} \tag{3.27}$$

und approximativ χ^2 verteilt mit $(n_x - 1)(n_y - 1)$ Freiheitsgraden, falls alle observierten Zellhäufigkeiten größer als der numerische Parameter `m` sind (vgl. ebd., S. 467 f). Somit gilt unter der Nullhypothese:

$$\chi^2 \overset{H_0}{\sim} \chi^2(df = (n_x - 1)(n_y - 1))$$

Anhand der n Beobachtungen lässt sich nun χ^2_{Obs}, also die mit den Beobachtungen realisierte Teststatistik, berechnen. Führt man den Test zum Niveau α mit $\alpha \in [0,1]$ durch, muss die Nullhypothese abgelehnt werden, falls $\chi^2_{Obs} > \chi^2_{1-\alpha}(df = (n_x - 1)(n_y - 1))$ ist.
$\chi^2_{1-\alpha}(df = (n_x - 1)(n_y - 1))$ ist das (1-α)-Quantil der χ^2-Verteilung mit $(n_x - 1)(n_y - 1)$ Freiheitsgraden (vgl. Agresti 2002, S. 78 ff).
Unter Umständen stehen einem bei der Datenanalyse nur wenige Beobachtungen zur Verfügung, weswegen manche Zellhäufigkeiten sehr wenig vorkommen. In diesen Fällen kann man nicht mehr von einer approximativen χ^2-Verteilung ausgehen und muss exakte Tests anwenden. Dafür gibt es unterschiedliche Ansätze. Falls jedoch in der Funktion `Stacked_Barplot` geringe Zellenhäufigkeiten auftreten, wird der exakte Fisher-Test angewandt. Er wird nun im 2×2-Fall hergeleitet, für Anwendungen auf $r \times p$- Kontigenztafeln mit $r, p \in \mathbb{N}$ sei auf ebd., S. 97 f verwiesen.

	Y= 1	Y= 2	
X=1	$h_{1,1}$	$h_{1,2}$	$h_{1,1} + h_{1,2} = h_{1,\cdot}$
X=2	$h_{2,1}$	$h_{2,2}$	$h_{2,1} + h_{2,2} = h_{2,\cdot}$
$\sum$	$h_{\cdot,1}$	$h_{\cdot,2}$	n

Tabelle 2: 2×2 Kontingenztafel für die beiden binären Merkmale X und Y

Idee dieses Tests ist das Bedingen auf alle unbekannten Parameter. Daher wird auf die marginale Verteilung der beiden dichotomen Merkmale X und Y bedingt (vgl. ebd., S. 91). De facto nimmt man also unter der Nullhypothese an, dass $h_{\cdot,1}, h_{1,\cdot}$ und n fest ist. Die Zellhäufigkeit $h_{1,1}$, also die Anzahl an Beobachtungen mit $Y = 1$ und $X = 1$, lässt sich unter dieser Bedingung als eine Realisation einer hypergeometrisch verteilten Zufallsvariablen beschreiben (vgl. ebd., S. 91). Wir definieren daher die Zufallsvariable $H_{1,1}$.

$$H_{1,1} \sim Hyp(N = n, M = h_{1,\cdot}, n = h_{\cdot,1})$$

Die Wahrscheinlichkeit, die Zellhäufigkeit $h_{1,1,obs}$ zu beobachten, kann also mit einer Wahrscheinlichkeitsdichte beschrieben werden.

$$f(H_{1,1} = h_{1,1,obs} | n, h_{1,\cdot}, h_{\cdot,1}) = \frac{\binom{h_{1,\cdot}}{h_{1,1,obs}} \binom{n - h_{1,\cdot}}{h_{\cdot,1} - h_{1,1,obs}}}{\binom{n}{h_{\cdot,1}}} \tag{3.28}$$

Zum besseren Verständnis soll das zugrundeliegende Urnenmodell betrachtet werden. Anfänglich geht man von einer Urne mit n Kugeln aus. Jede dieser Kugeln ist entweder schwarz oder weiß, wobei es genau $h_{1,\cdot}$ schwarze und $n - h_{1,\cdot}$ weiße Kugeln gibt. Durch $f(H_{1,1} = h_{1,1,obs} | n, h_{1,\cdot}, h_{\cdot,1})$ wird nun die Wahrscheinlichkeit beschrieben, genau $h_{1,1,obs}$ schwarze und $h_{\cdot,1} - h_{1,1,obs}$ weiße Kugeln bei $h_{\cdot,1}$-maligem Ziehen ohne Zurücklegen zu erhalten (vgl. Fahrmeir, Künstler u. a. 2011, S. 258 f).
Wenden wir dieses Urnenmodell nun auf die 2×2-Kontingenztafel, siehe Tabelle 2, an, so kann die Urne als die Grundpopulation aller Beobachtungen der beiden Merkmale X und Y betrachtet werden. Wegen der festen marginalen Verteilung gibt es nun $h_{1,\cdot}$ Beobachtungen mit $X = 1$ und $h_{2,\cdot}$ Beobachtungen mit $X = 2$ in dieser Grundpopulation. Falls man genauso oft aus der Urne zieht wie $Y = 1$ beobachtet wurde, also $h_{\cdot,1}$ mal, sollte man im Mittel $h_{1,1}$ Beobachtungen mit $X = 1$ und $Y = 1$ und $h_{\cdot,1} - h_{1,1} = h_{2,1}$ mal $Y = 2$ und $X = 1$ ziehen.
Unter der Nullhypothese, also der Unabhängigkeit von X und Y, sollte also
$H_{1,1} \sim Hyp(N = n, M = h_{1,\cdot}, n = h_{\cdot,1})$ gelten. Für eine explizite Realisation von $H_{1,1}$, welche als $h_{1,1,obs}$ notiert wird, kann der P-Wert p über die Verteilungsfunktion der hypergeometrischen Verteilung bestimmt werden (vgl. Agresti 2002, S. 91).

$$\begin{aligned} p = P(H_{1,1} > h_{1,1,obs}) &= 1 - \underbrace{P(H_{1,1} \leq h_{1,1,obs})}_{\text{Verteilungsfunktion von } H_{1,1}} \\ &\overset{\text{mit (3.28)}}{=} 1 - \sum_{i=0}^{h_{1,1,obs}} \frac{\binom{h_{1,\cdot}}{i} \binom{n - h_{1,\cdot}}{h_{\cdot,1} - i}}{\binom{n}{h_{\cdot,1}}} \end{aligned}$$

Falls nun $p < \alpha$ gilt, wobei α das a priori festgelegte Signifikanzniveau ist, wird die Nullhypothese abgelehnt. In diesem Fall würde man von keiner Unabhängigkeit zwischen den Variablen ausgehen. Ist $p \geq \alpha$ kann die Nullhypothese nicht abgelehnt werden, und die Merkmale X und Y können als unabhängig voneinander betrachtet werden.
Anhand dieser Tests kann jedoch nicht der Grad der Assoziation von zwei Variablen untersucht werden, sondern es wird nur grundlegend analysiert, ob von Unabhängigkeit ausgegangen werden kann. Weitere Untersuchungen der beiden Variablen sind notwendig, um auch die Stärke des Zusammenhangs betrachten zu können (vgl. Agresti 2002, S. 84). Eine Möglichkeit diese zwischen diskreten Merkmalen in 2×2-Kontingenztafeln zu analysieren, ist das *Odds Ratio*.

Def. 3.15 *Odds Ratio*: Das *Odds Ratio* wird auch Kreuzproduktverhältnis genannt und ist eine Maßzahl für die Stärke eines Zusammenhangs zwischen zwei dichotomen Merkmalen, hier X und Y.

$$\gamma(X,\overline{X}|Y=2,Y=1) = \frac{\gamma(X,\overline{X}|Y=2)}{\gamma(X,\overline{X}|Y=1)} = dfrac\frac{P(X=2|Y=2)}{P(X=1|Y=2)}\frac{P(X=2|Y=1)}{P(X=1|Y=1)}$$

Die Odds Ratios lassen sich auch aus relativen oder absoluten Zellwerten einer 2×2 - Kontingenztafel hier am Beispiel von Tabelle 2 bestimmen (vgl. Fahrmeir, Künstler u. a. 2011, S. 119 ff).

$$\gamma(X,\overline{X}|Y=2,Y=1) = \frac{h_{1,1}\cdot h_{2,2}}{h_{1,2}\cdot h_{2,1}}$$

Gilt $\gamma(X,\overline{X}|Y=2,Y=1)=1$ geht man von einer Unabhängigkeit zwischen X und Y aus, bei $\gamma(X,\overline{X}|Y=2,Y=1)<1$ ist die Chance, dass auch $Y=1$ gilt, in der Subpopulation mit $X=0$ höher als mit $X=1$. Zuletzt ist es bei $\gamma(X,\overline{X}|Y=1,Y=0)>1$ wahrscheinlicher, dass auch $Y=1$ gilt, in der Subpopulation mit $X=0$ niedriger als mit $X=1$ (vgl. ebd., S. 120).

3.3 Zeitreihen

3.3.1 Season- und Trendbereinigung

Der letzte behandelte Typ von Daten sind Zeitreihen. Man misst über einen bestimmten Zeitraum von t_a bis t_e ein metrisches oder diskretes Merkmal X n-mal an einer Person oder einem Objekt. Dieser Zeitraum $[t_a,t_e]$ lässt sich in Zyklen unterteilen, die davon abhängen, in welchen Abständen die Daten erhoben wurden. Meist dauert ein Zyklus ein Jahr. Hat man also das Merkmal X monatlich erhoben, umfasst ein Zyklus 12 Messungen. Im Gegensatz zu normalen metrischen oder diskreten Daten geht man bei den n Beobachtungen nicht von unabhängigen Stichprobenziehungen aus. Wird beispielsweise das Gewicht monatlich gemessen, hängt eine Messung von der vorherigen ab. Ohne die Unabhängigkeitsannahme stellt sich die Frage nach der adäquaten Analyse von Zeitreihen. In *interVisu* gibt es dafür die Funktion `Timeseries(height = c(800, 400), data, n = 10)`. Einzelne Zeitreihen von bestimmten Personen oder Objekten werden nun jeweils als eine Variable in dem übergebenen Datensatz `data` abgespeichert. Mithilfe der Anwendung kann man Zeitreihen in eine Saison-, Trend- und Zufallskomponente zerlegen. Dargestellt wird diese Aufspaltung in vier untereinander angeordneten Scatterplots. In dem ersten Plot wird die beobachtete Zeitreihe angezeigt, darunter der Saisonanteil der Schätzung, dann der Trend-Anteil und zuletzt ein zufälliger Fehler, der durch die Schätzung der Trend- und Saisonkomponente nicht erklärt werden konnte.

Methoden

Die Zerlegung einer Zeitreihe in Saison-, Trend- und Zufallskomponente kann als ein Regressionsmodell betrachtet werden. Gegeben sei eine Zeitreihe Y mit n Messungen $Y_1,\ldots,Y_n$:

$$Y_i = T_i + S_i + R_i$$

Dafür sei T_i der Trendanteil, S_i die saisonale Erklärung und R_i der zufällige Teil zur Erklärung von Y_i. Wir definieren weiter die Anzahl an Beobachtungen in einem Zyklus als $n_{(s)}$. Es gibt verschiedene Möglichkeiten ,diese einzelnen erklärenden Komponenten zu schätzen. Die in dieser Applikation implementierte Version benutzt eine iterative Loess-Schätzung nach W. Cleveland, R. Cleveland u. a. 1990. Diese Methode wird *STL* genannt und ermöglicht eine leicht zu modifizierende Saisonzerlegung der Zeitreihen, welche auch bei fehlenden Daten anwendbar ist (vgl. ebd., S. 3).
Allgemein lässt sich das Verfahren in eine innere und eine äußere Schleife aufteilen.
Die innere Schleife lässt sich rekursiv beschreiben. Für den k-ten Durchlauf von $n_{(i)}$ Wiederholungen sind dafür $S_i^{(k)}$ und $T_i^{(k)}$ als saisonaler beziehungsweise Trendbestandteil der i-ten Beobachtung am Ende der k-ten Schleife definiert. Im k-ten Durchlauf wird die Saisonkomponente $S_i^{(k)}$ auf $S_i^{(k+1)}$ und $T_i^{(k)}$ auf $T_i^{(k+1)}$ in sechs Schritten aktualisiert (vgl. ebd., S. 7).
Zu Beginn der k-ten Schleife wird die Zeitreihe von $T_i^{(k)}$ bereinigt.

$$Y_{i,bereinigt,T} = Y_i - T_i^{(k)}$$

Da während des ersten Durchlaufs noch keine Trendkomponente $T_i^{(0)}$ geschätzt wurde, wird als Startwert $T_i^{(0)} = 0 \; \forall i = 1, \ldots, n$ angenommen. Nun werden alle Beobachtungen $Y_{i,bereinigt,T}$ in $n_{(s)}$ Gruppen nach den einzelnen Positionen der Beobachtungen in den Zyklen gesplittet. Geht man beispielsweise von monatlichen Messungen aus, fasst man alle Beobachtungen im Januar, alle im Februar et cetera je in einer Gruppe zusammen. In jeder dieser Gruppen wird nun $Y_{bereinigt,T}$ in Abhängigkeit der Zeit mit dem Loess-Verfahren und $q = n_{(s)}$ lokal modelliert. Mit diesen $n_{(s)}$ Modellen werden nun alle Punkte von einer Messung vor der ersten Beobachtung bis zu einer Messung nach der letzten Beobachtung prognostiziert. Diese Schätzungen werden anschließend in saisonalen Serien $C_i^{(k+1)}$ zusammengefasst. Anschließend wird ein sogenannter *Low-Pass Filter* angewandt. Hierfür wird jedes $C_i^{(k+1)}$ zweimal mit einem gleitenden Mittelwert der Länge n_p und anschließend mit dem Wert der Länge 3 gefiltert. Durch dieses Verfahren befinden sich die prognostizierten Beobachtungen wieder zwischen den anfänglichen Start- und Endpunkten. Zuletzt wird in diesem Schritt die resultierende Zeitreihe $L_i^{(k+1)}$ lokal mit $q = n_{(l)}$ geschätzt. Nun ergibt sich die aktualisierte saisonale Komponente $S_i^{(k+1)}$ mit:

$$S_i^{(k+1)} = C_i^{(k+1)} - L_i^{(k+1)}$$

Hinterher wird die Zeitreihe saisonbereinigt.

$$Y_{i,bereinigt,S} = Y_i - S_i^{(k+1)}$$

Zum Ende der k-ten Schleife wird $T_i^{(k+1)}$ lokal mit $q = n_{(t)}$ geschätzt (vgl. ebd., S. 7).
Die äußere Schleife besteht aus $n_{(i)}$ Durchläufen der inneren Schleife und berechnet anschließend robuste Gewichte für jede Beobachtung. Diese Prozedur wird $n_{(o)}$ mal wiederholt mit $n_{(i)}, n_{(o)} \in \mathbb{N}$ (vgl. W. Cleveland, R. Cleveland u. a. 1990, S. 6).

Um die Gewichte ausrechnen zu können, wird im m-ten Durchlauf mit $m \in 1,\ldots,n_{(0)}$ anfänglich der nicht erklärte Rest R_i ausgerechnet.

$$R_i = Y_i - S_i - T_i$$

Nach der Größe dieser Residuen R_i sollen nun die robusten Gewichte ρ_i gewählt werden. Dabei gilt mit $h = 6|R|_{0.5}$ und $|R|_{0.5}$ als der Median aller absoluten Residuen nach Def. 3.4 :

$$\rho_i = B\left(\frac{|R_i|}{h}\right)$$

mit

$$B(u) = \begin{cases} (1-u^2)^2 & ,0 \leqslant u < 1 \\ 0 & , \text{ sonst} \end{cases}$$

Hat man nach $n_{(i)}$-maligem Durchlauf der inneren Schleife auch $\rho_i \; \forall i = 1,\ldots,n$ berechnet, ist ein Durchlauf der äußeren Schleife beendet. In der nächsten Wiederholung der Schleife werden alle in der lokalen Schätzung berechneten Gewichte der inneren Schleife mit ρ_i multipliziert.
Die Wahl der zu verwendeden Parameter $n_{(i)}$,$n_{(o)}$,$n_{(l)}$,$n_{(t)}$ und $n_{(s)}$ wird in Paragraph 3 und 4 bei W. Cleveland, R. Cleveland u. a. 1990 diskutiert.

3.3.2 Season- und Trendbereinigung bei Finanzdaten

Zeitreihen von Finanzdaten können mit der Funktion `get.hist.data` von Yahoo Finance geladen werden. Für die Analyse dieser Daten gibt es eine statische Applikation
`Timeseries_Stat(height = c(800, 400))`. Die Funktionalität der Anwendung ist identisch zu der generellen Applikation, welche bereits in 3.3.1 beschrieben wurde. Einziger Unterschied ist, dass keine zu übergebenen Daten notwendig sind, da diese von
`http://finance.yahoo.com` geladen werden.
Der Nutzer kann in der Anwendung `Timeseries_Stat` zuerst aus einer Liste eine der drei Indizes Dax[17], Nasdaq[18] und Dow Jones auswählen. Weiter kann man sich entweder die Zeitreihe des ausgewählten Indexes oder eines der je 10 stärksten Unternehmen, welche unter den einzelnen Indizes geführt werden, analysieren. Der Anwender kann auch zuletzt die Zeitspanne der Daten wählen.

[17] Deutscher Aktien Index
[18] National Association of Securities Dealers Automated Quotations

4 Tutorium des R Paketes *interVisu*

Im folgenden Abschnitt wird zu jeder Anwendung ein praxisnahes Tutorium beschrieben, in dem interne Beispieldaten von R verwendet werden.

4.1 Metrische Variablen

4.1.1 `Single_Metric_Variable_Analysis`

Wie die in 3.1.1 beschriebene Methodik ist auch die Applikation in die drei verschiedenen Möglichkeiten der Präsentation der metrischen Variable aufgeteilt. Ruft der Anwender die Applikation anfänglich auf, wird ein Boxplot des Merkmals in der ersten Spalte des übergebenen Datensatzes gezeichnet (Abbildung 14). Sobald der Benutzer auf das Feld *Metric Variable* klickt, können auch andere metrische Merkmale in dem Boxplot dargestellt werden (Abbildung 15). Mithilfe des Buttons *Show Points* können alle Datenpunkte angezeigt werden (Abbildung 16). Der Boxplot kann auch horizontal dargestellt werden (Abbildung 17). Für die Reproduzierbarkeit der Graphen lässt man sich den R-Code des gerade dargestellten Graphen anzeigen (Abbildung 18). Will man nur einen bestimmten Wertebereich des Merkmals betrachten oder nur einen ausgewählten Bereich ploten, ist dies mittels der beiden Slider *Use Data From/To* und *Plot Data From/To* möglich (Abbildung 19 und Abbildung 20).
Die zweite implementierte Möglichkeit der deskriptiven Betrachtung metrischer Merkmale ist das Histogramm als Dichteapproximation (Abbildung 21). Der dargestellte Y-Achsenabschnitt scheint größer als nötig zu sein, daher kann über das Feld *UpperY-Axis Limit* das Maximum der dargestellten Y-Achse alterniert werden (Abbildung 22). Auch lässt sich wieder nur ein Ausschnitt der Daten in dem Histogramm darstellen (Abbildung 23). Der Ursprung des Histogramms α ist der Punkt an dem der erste Balken beginnt. Bestimmt wird dieser mit dem Slider *Origin of the Histogram* (Abbildung 24). Zuletzt lässt sich auch die Breite d der Balken verändern (Abbildung 25).
Eine weitere Möglichkeit der Dichteschätzung einer metrischen Variable ist der Kerndichte-Schätzer. Es lassen sich über diese Darstellung verschiedene Kerne verwenden und wie bei allen vorherigen auch der R-Code darstellen (Abbildung 26 und Abbildung 27). Die verwendete Bandbreite kann mithilfe des Sliders *Bandwidth* eingestellt werden (Abbildung 28).

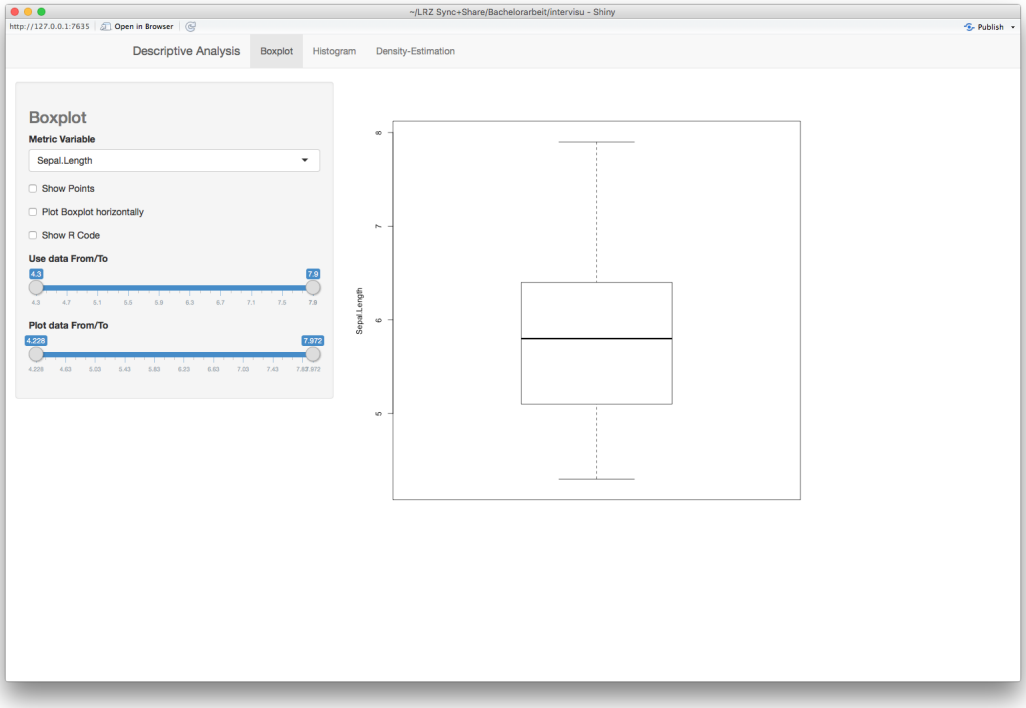

Abbildung 14: `Single_Metric_Variable_Analysis` 1

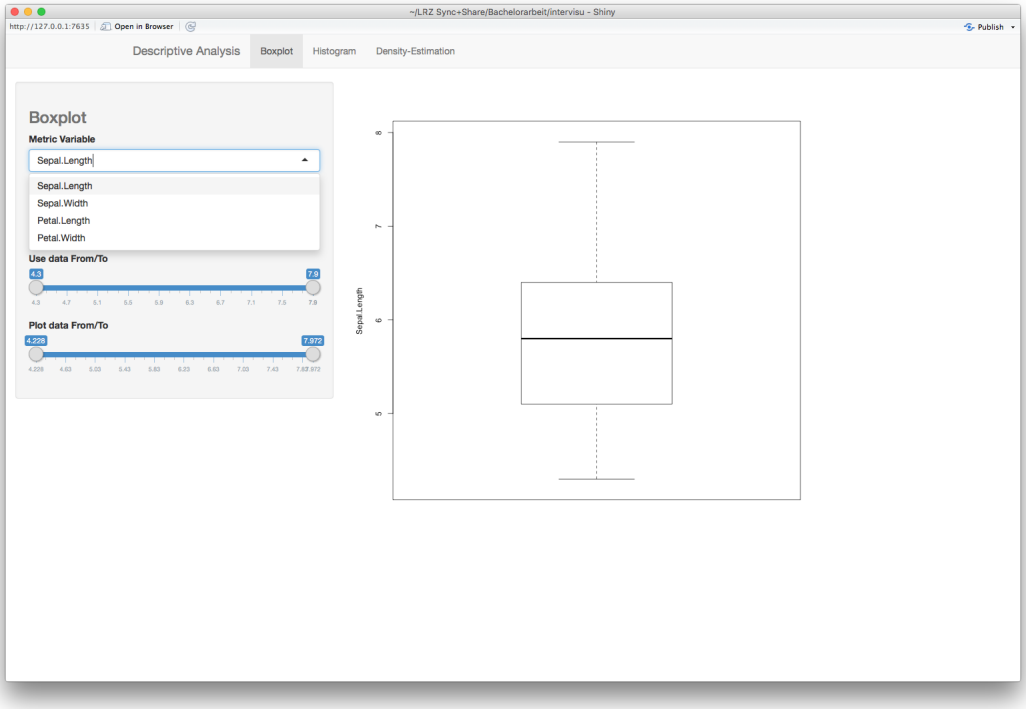

Abbildung 15: `Single_Metric_Variable_Analysis` 2

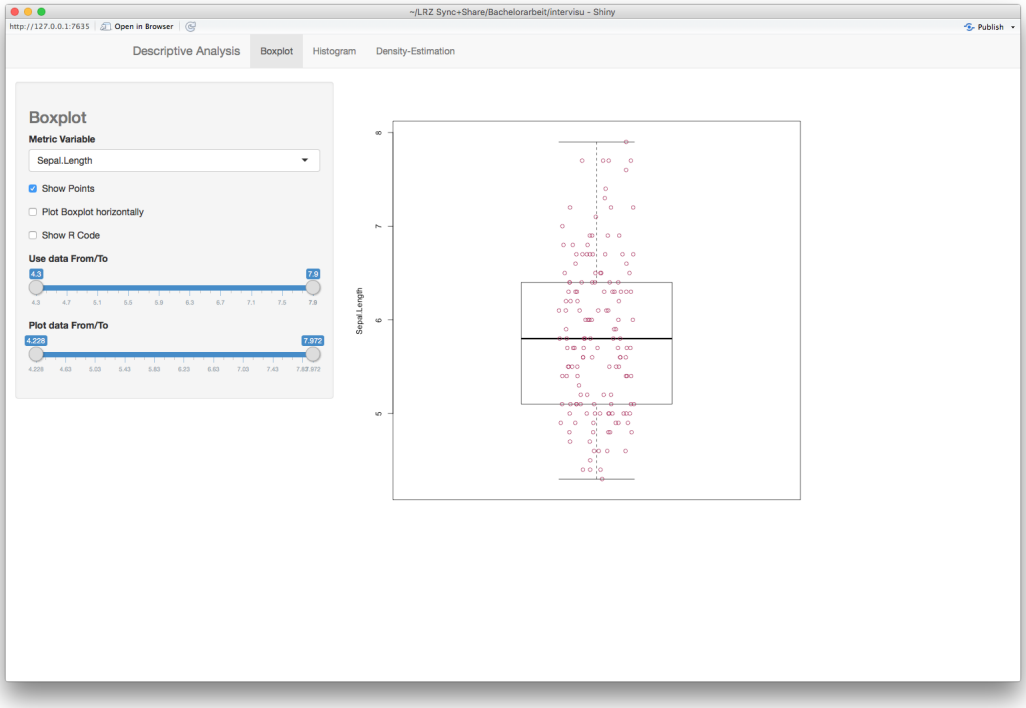

Abbildung 16: `Single_Metric_Variable_Analysis` 3

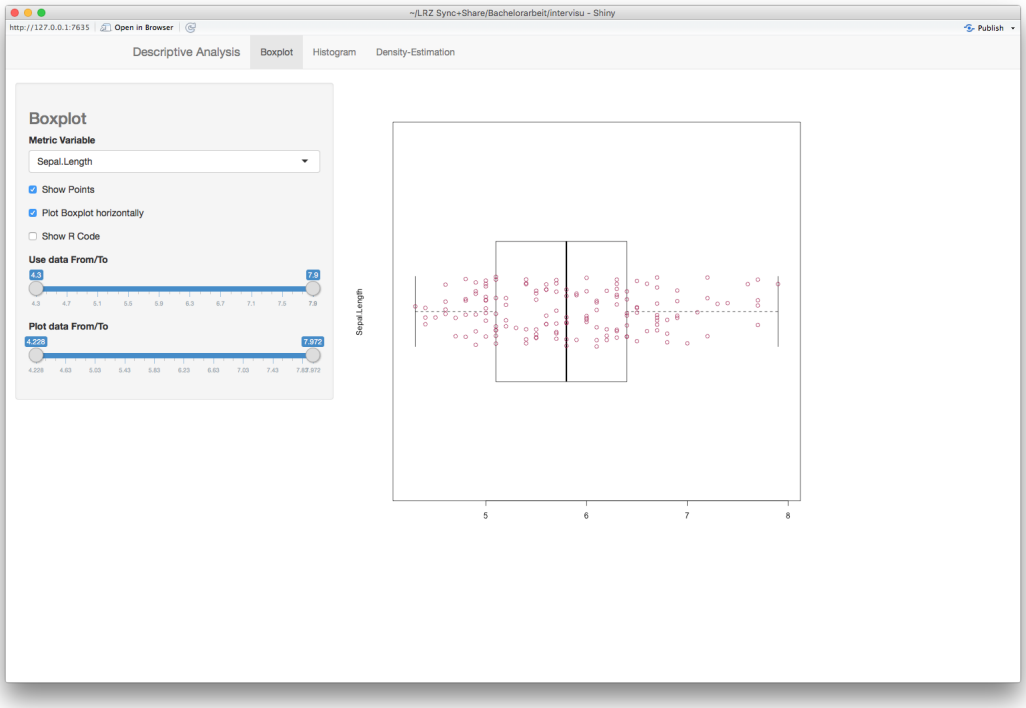

Abbildung 17: `Single_Metric_Variable_Analysis` 4

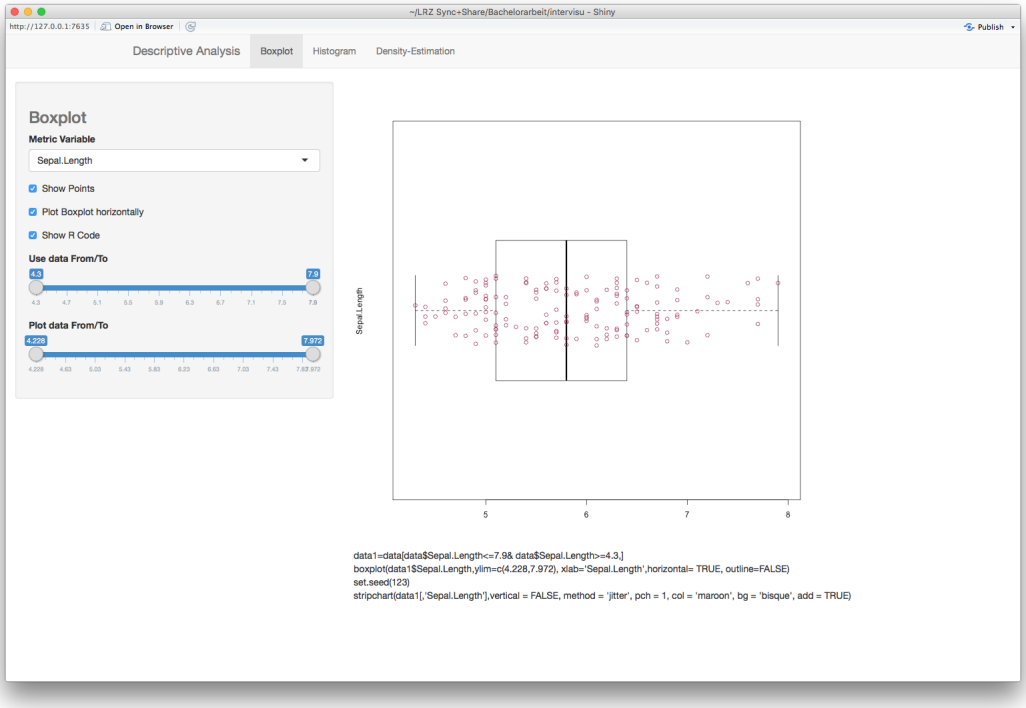

Abbildung 18: `Single_Metric_Variable_Analysis` 5

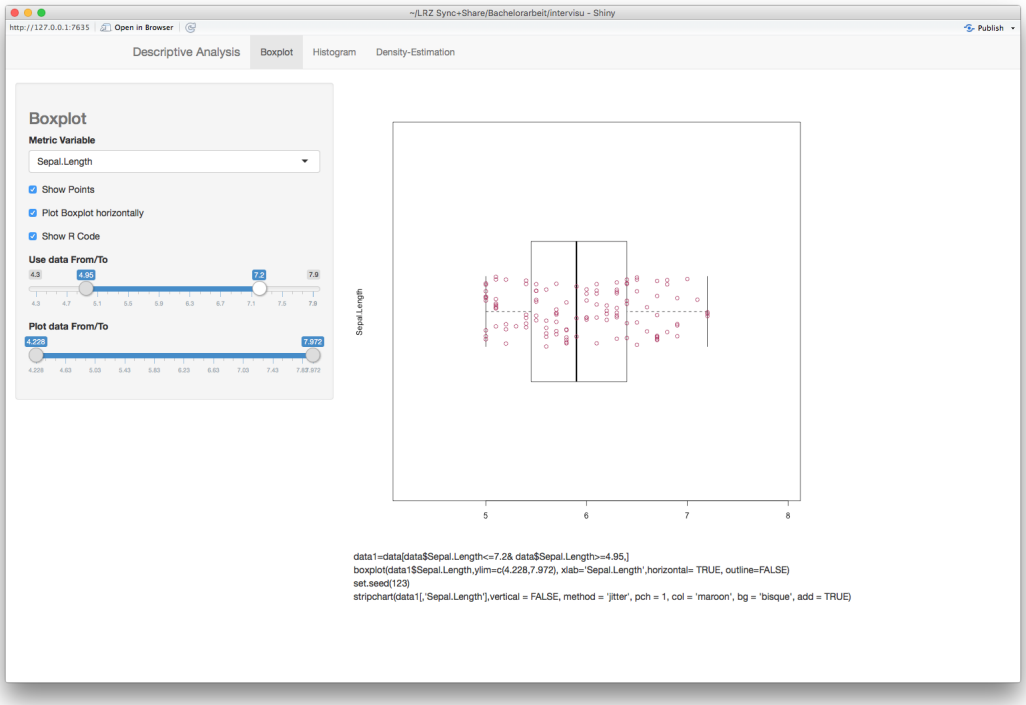

Abbildung 19: `Single_Metric_Variable_Analysis` 6

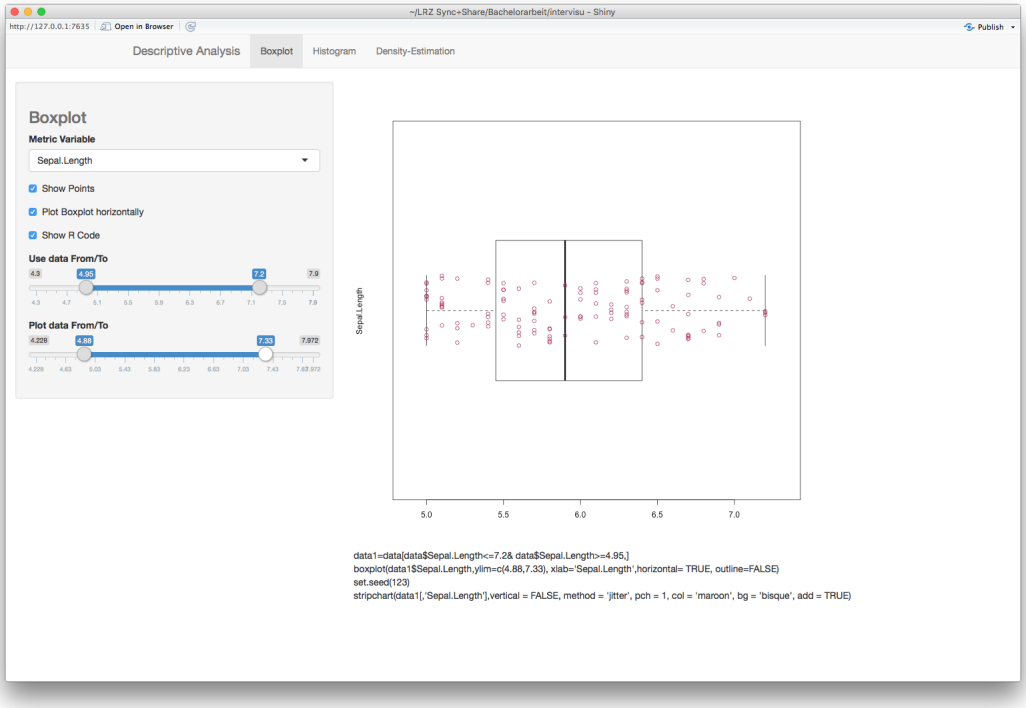

Abbildung 20: `Single_Metric_Variable_Analysis` 7

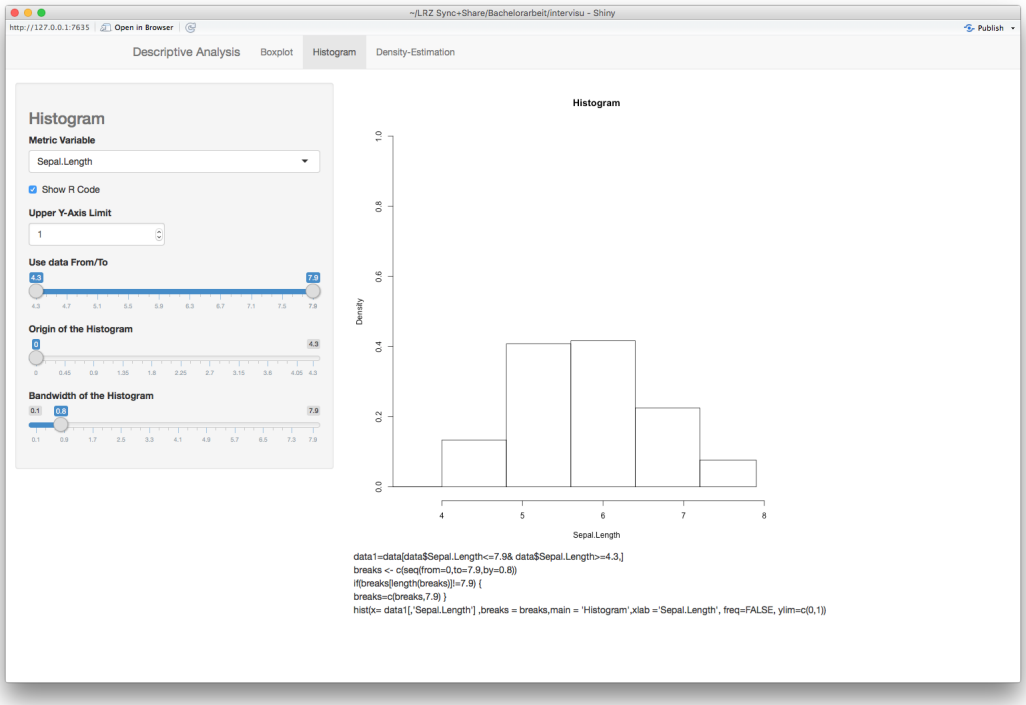

Abbildung 21: `Single_Metric_Variable_Analysis` 8

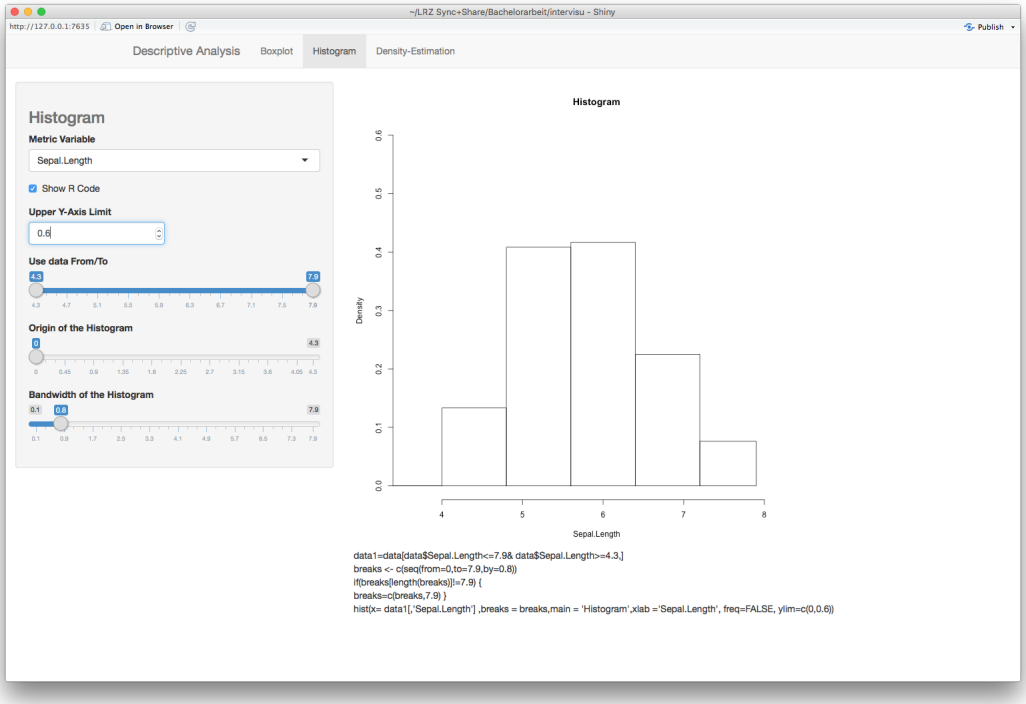

Abbildung 22: `Single_Metric_Variable_Analysis` 9

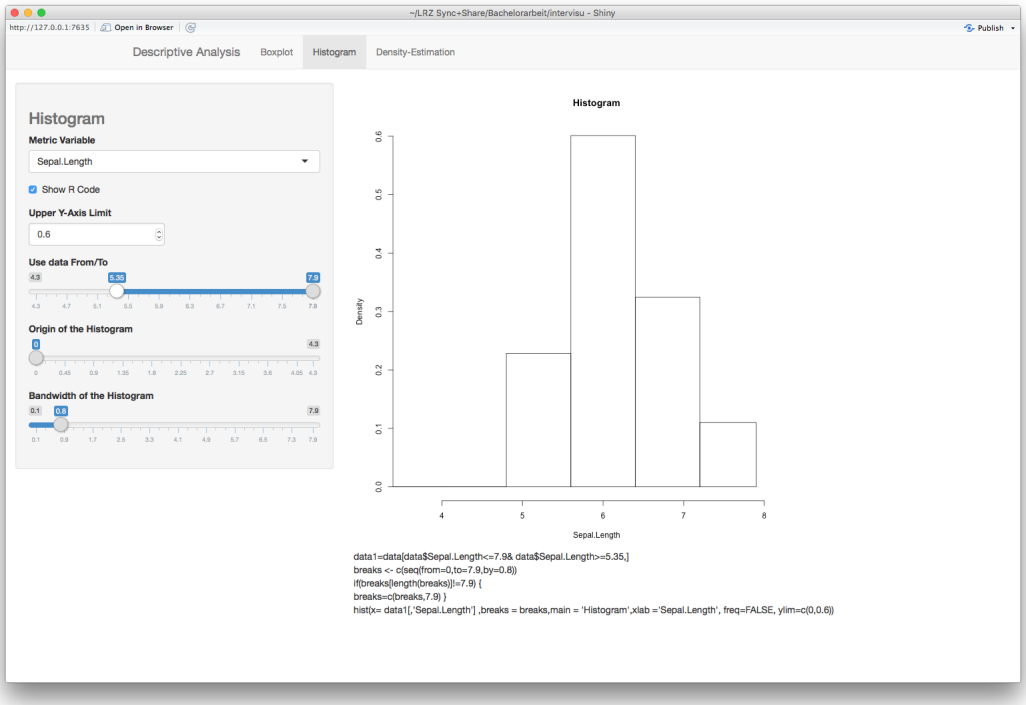

Abbildung 23: `Single_Metric_Variable_Analysis` 10

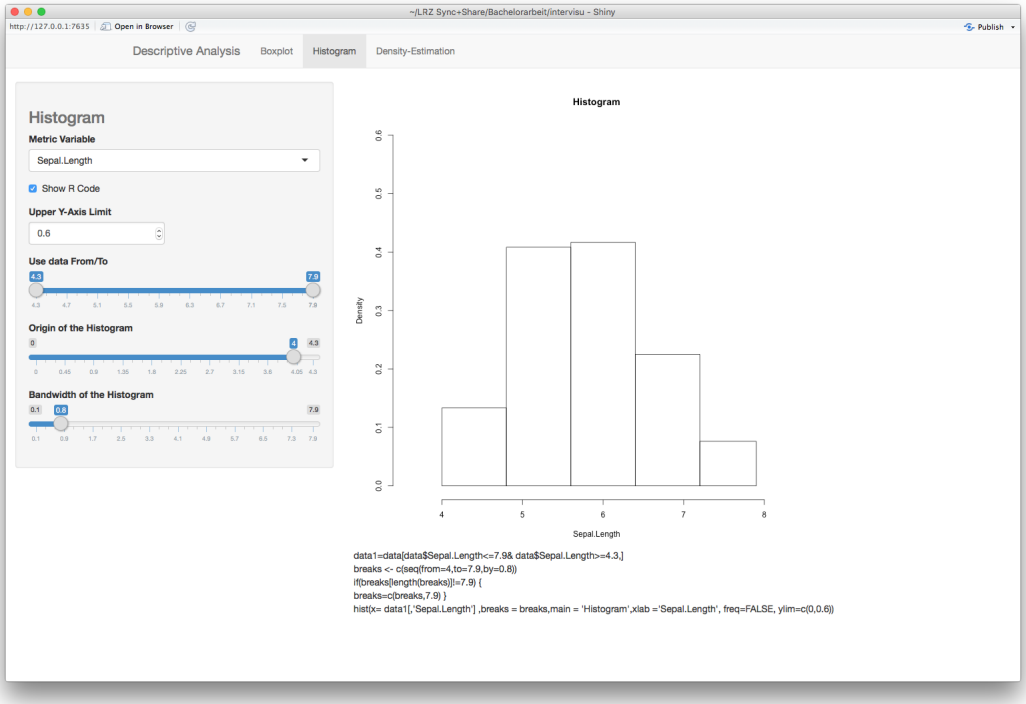

Abbildung 24: `Single_Metric_Variable_Analysis` 11

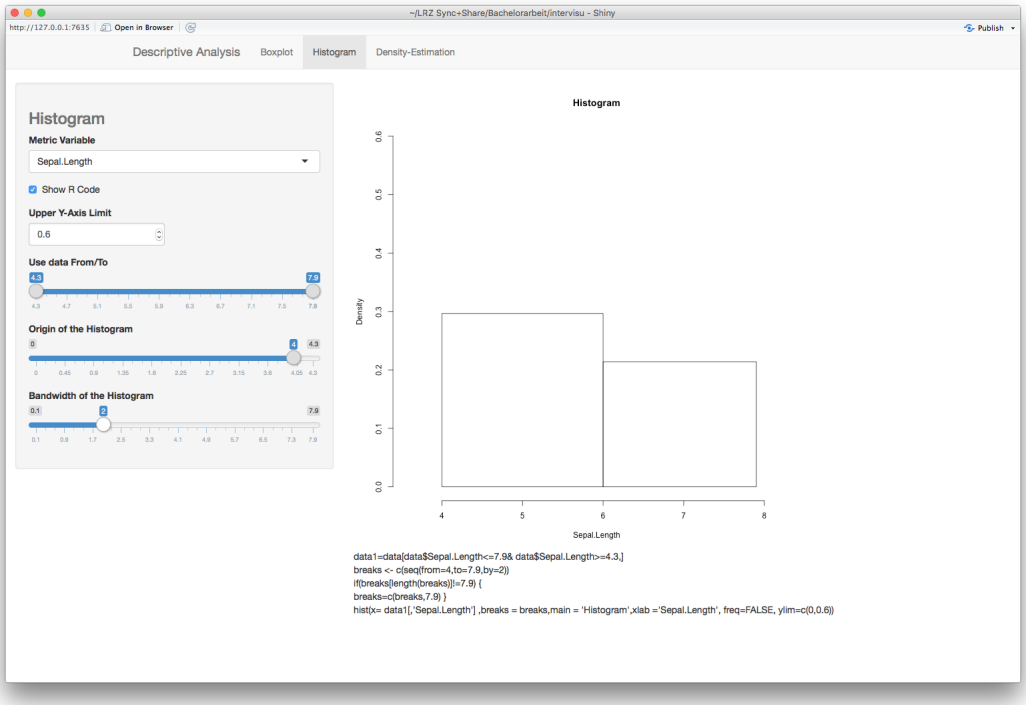

Abbildung 25: `Single_Metric_Variable_Analysis` 12

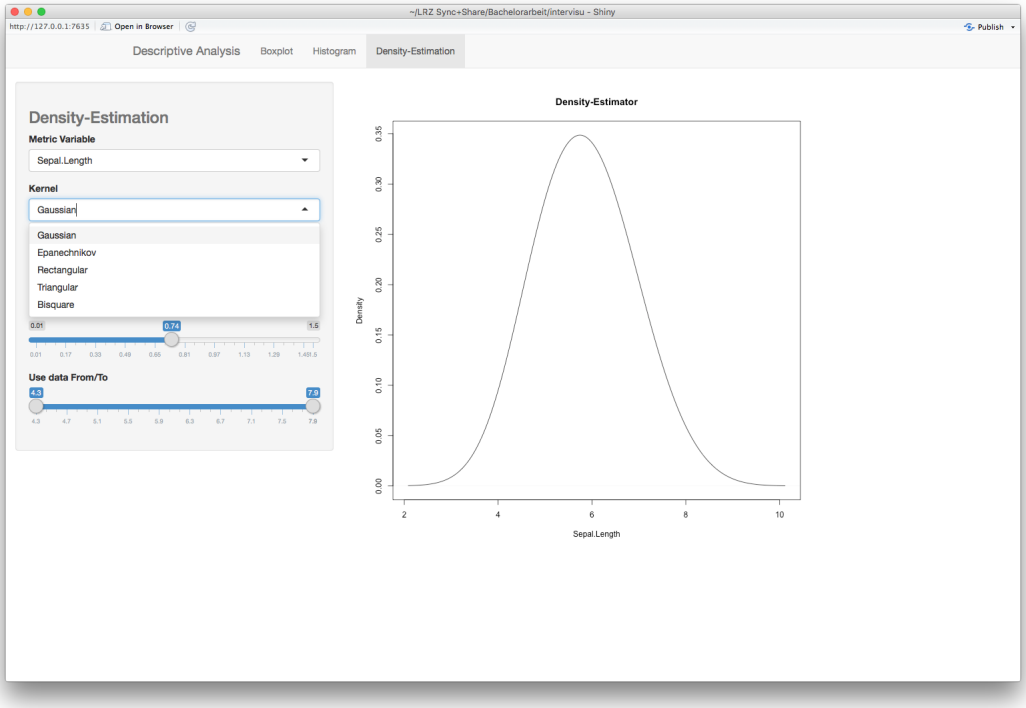

Abbildung 26: `Single_Metric_Variable_Analysis` 13

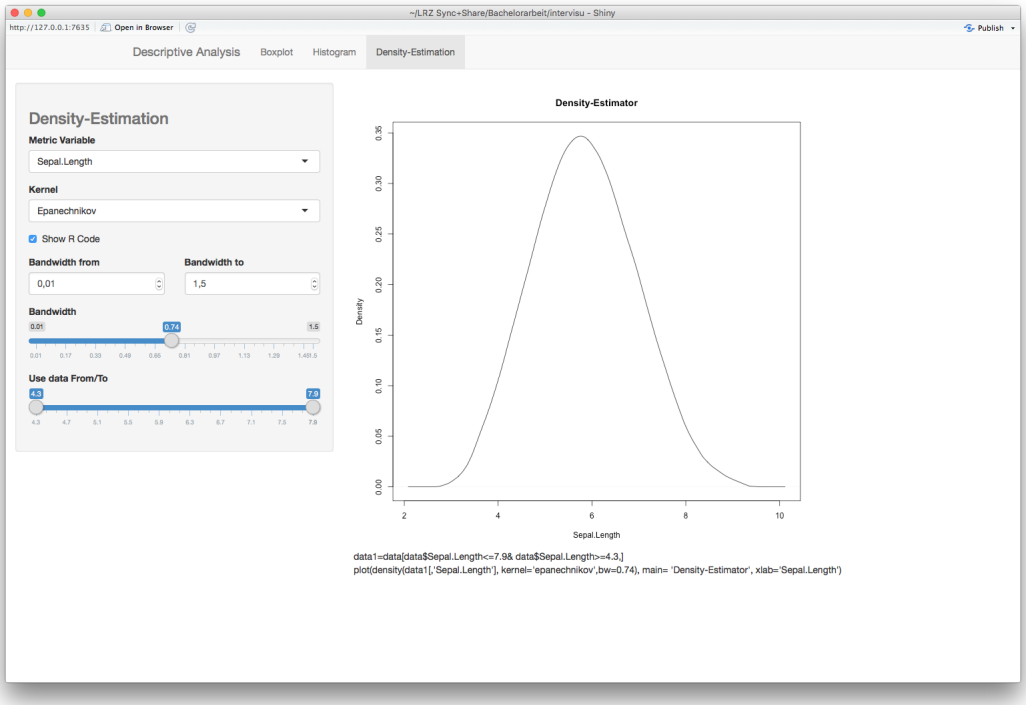

Abbildung 27: `Single_Metric_Variable_Analysis` 14

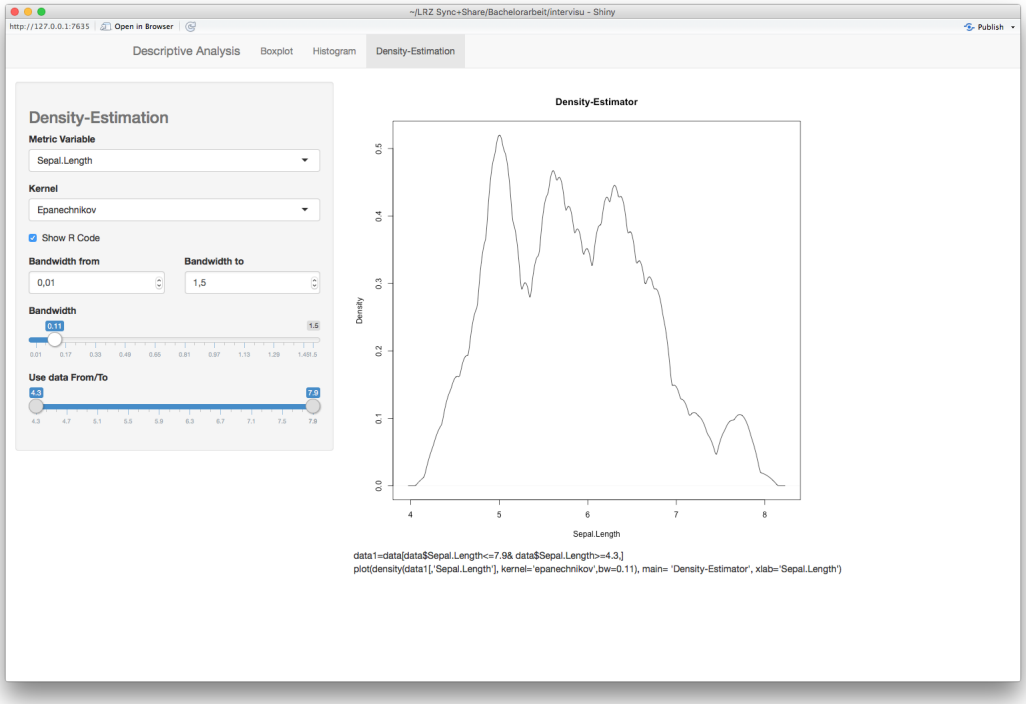

Abbildung 28: `Single_Metric_Variable_Analysis` 15

4.1.2 `Scatterplot_Matrix`

Anfänglich öffnet sich bei der Anwendung ein User-Interface, welches den Nutzer dazu auffordert, mindestens zwei metrische Variablen aus der Liste aller Variablen in dem übergebenen Datensatz auszuwählen (Abbildung 29). Dies kann man machen, indem man das Feld *Variables* anklickt (Abbildung 30). Eine Scatterplot-Matrix wird sobald der Anwender zwei Merkmale ausgewählt hat dargestellt (Abbildung 31). Mit einem Einfachklick auf einen der Scatterplots in der Scatterplot-Matrix öffnet sich ein einfacher größerer Scatterplot auf der linken Seite der Matrix (Abbildung 32). Mittels eines Doppelklicks kann der Anwender analog indizieren, welcher Scatterplot auf der rechten Seite der Matrix angezeigt werden soll (Abbildung 33). In den seitlichen Scatterplots kann man Punkte mithilfe mithilfe der Rechtecks- oder Lassoauswahl bestimmen. Sobald man sich mit dem Cursor auf dem Graphen befindet öffnet sich dazu ein Auswahlmenü (Abbildung 34). Nun werden mehrere Variablen in der Scatterplot-Matrix dargestellt (Abbildung 35). Verwendet man den Button *DrawRegressionline* wird eine lineare Regressionsgerade in alle Plots eingezeichnet (Abbildung 36). Auch Loess-Kurven können anhand des Schalters mit der Aufschrift *DrawLoessline* dargestellt werden (Abbildung 37). Hat man eine der beiden Optionen ausgewählt und Beobachtungen in den seitlichen Scatterplots markiert, steht die Möglichkeit, die Geraden und Kurven nach den so definierten Gruppen zu Berechnen, zur Verfügung (Abbildung 38). Diese Option kann der Anwender mittels des Buttons *ByGroup* benutzen (Abbildung 39). Zuletzt gibt es noch die Möglichkeit sich die markierten Beobachtungen in dem übergebenen Datensatz explizit anzuschauen (Abbildung 40).

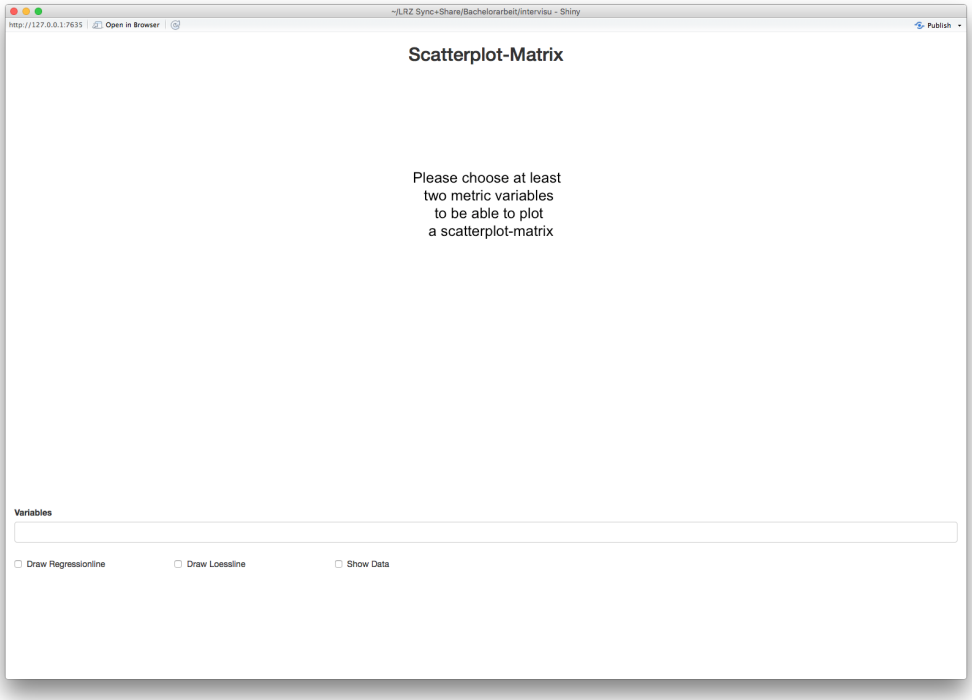

Abbildung 29: `Scatterplot_Matrix` 1

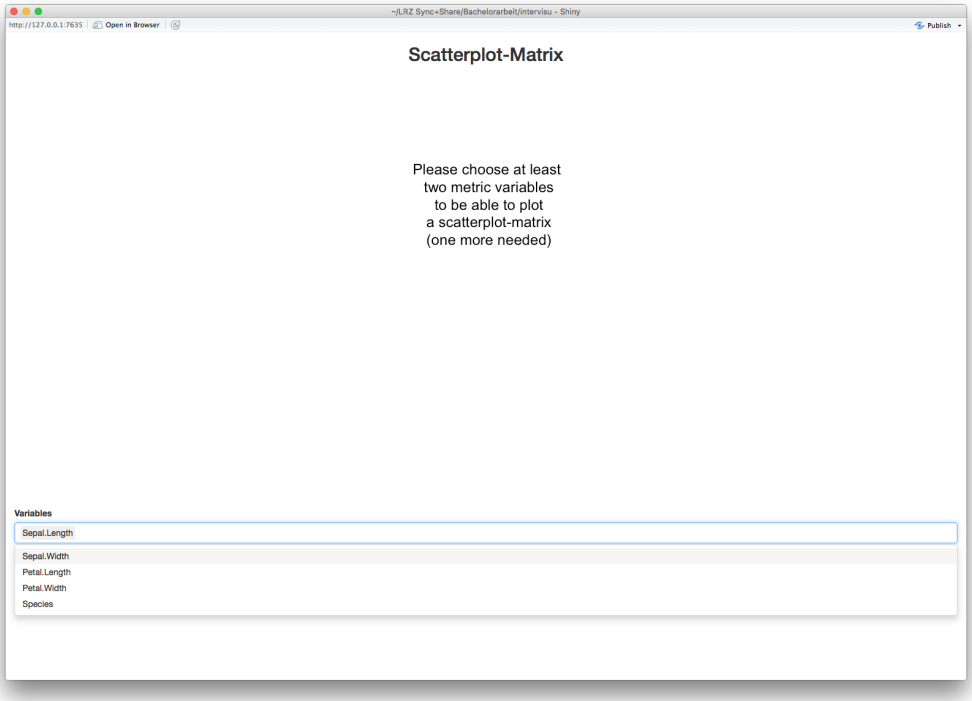

Abbildung 30: `Scatterplot_Matrix` 2

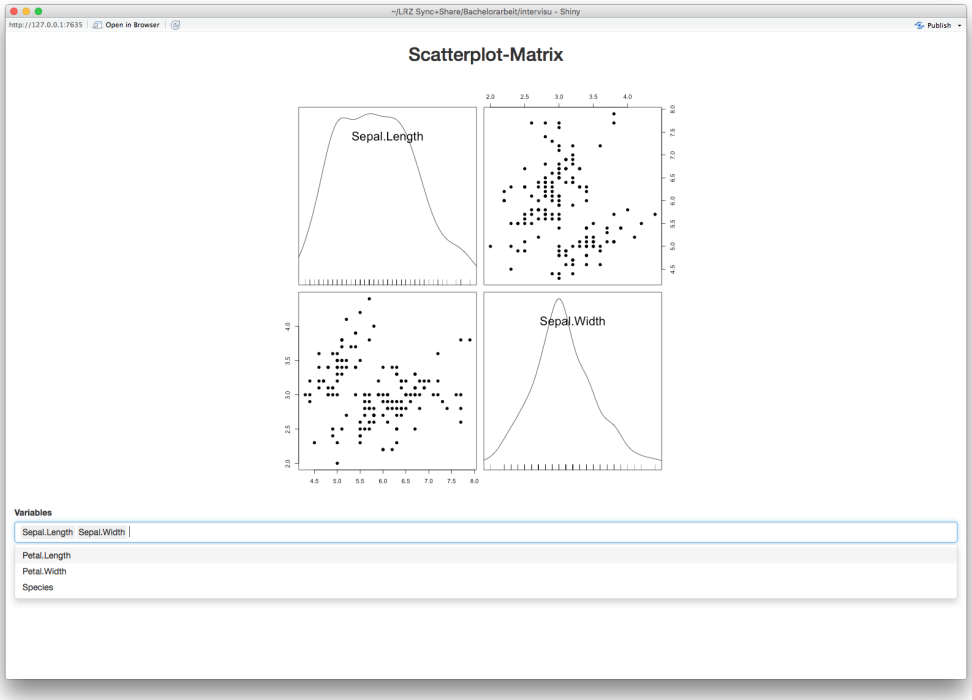

Abbildung 31: `Scatterplot_Matrix` 3

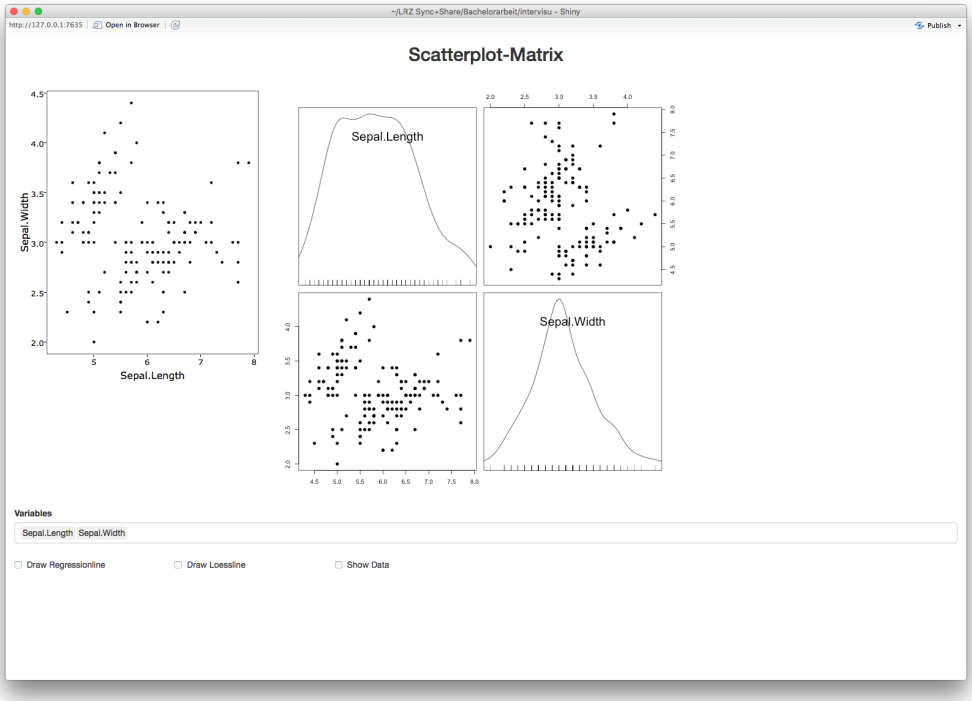

Abbildung 32: `Scatterplot_Matrix` 4

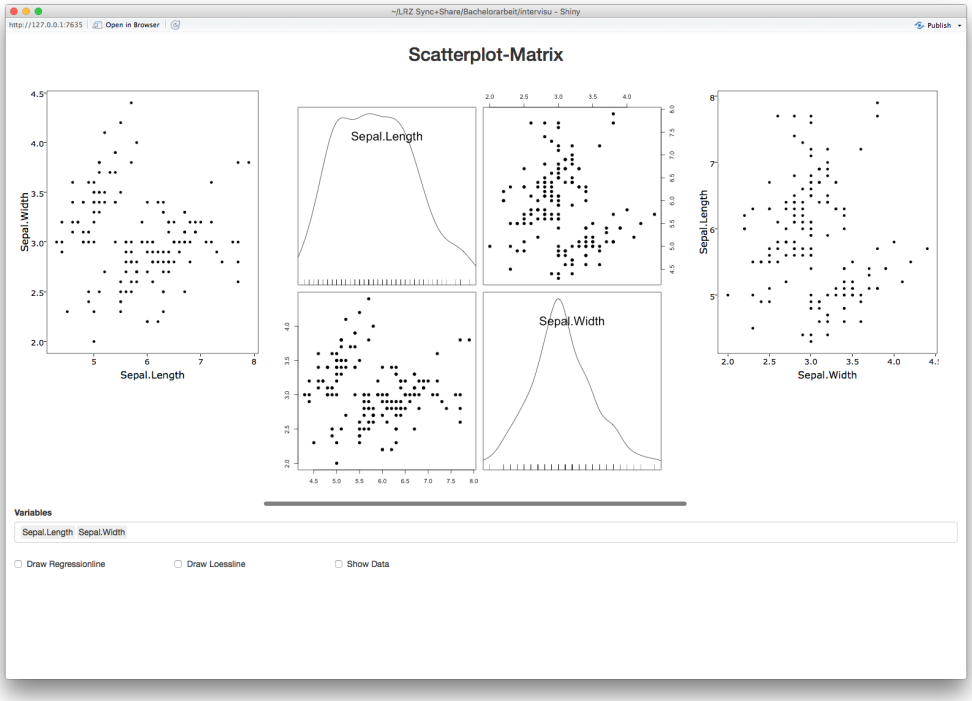

Abbildung 33: `Scatterplot_Matrix` 5

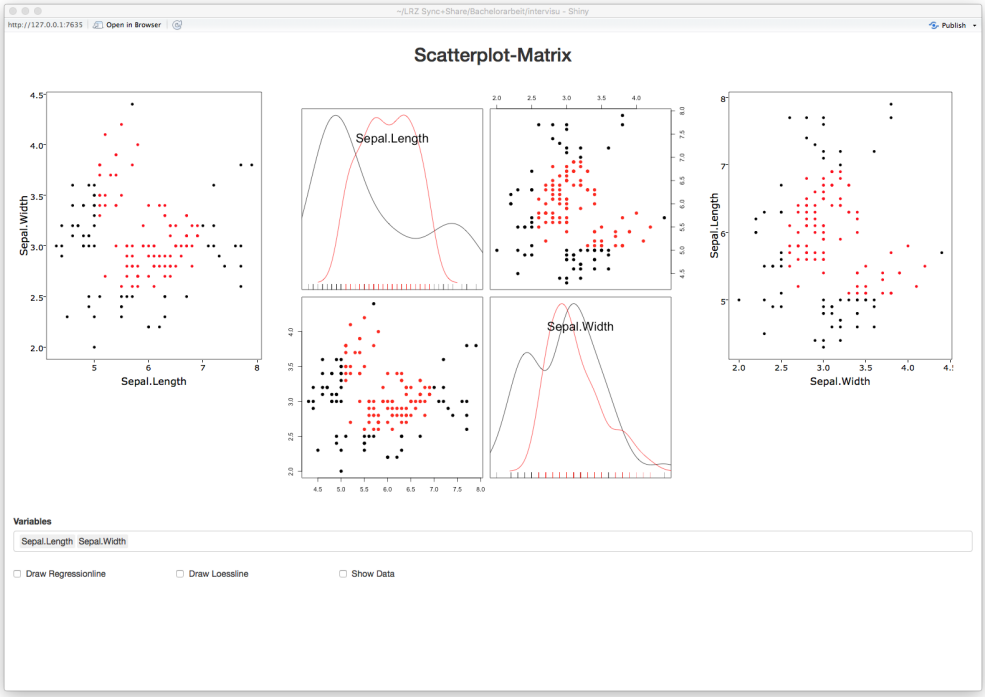

Abbildung 34: `Scatterplot_Matrix` 6

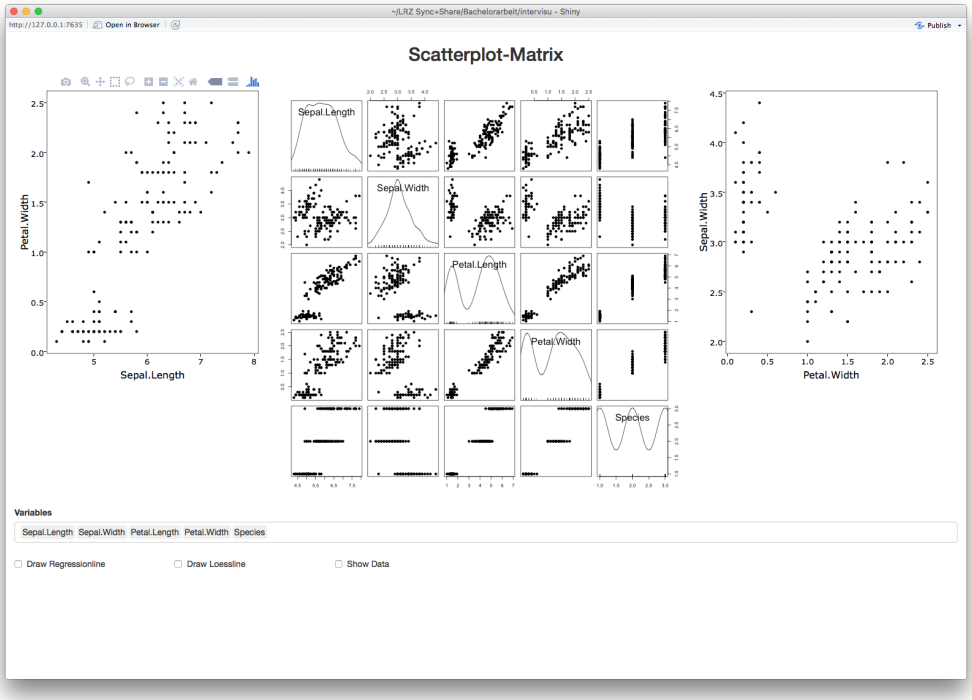

Abbildung 35: `Scatterplot_Matrix` 7

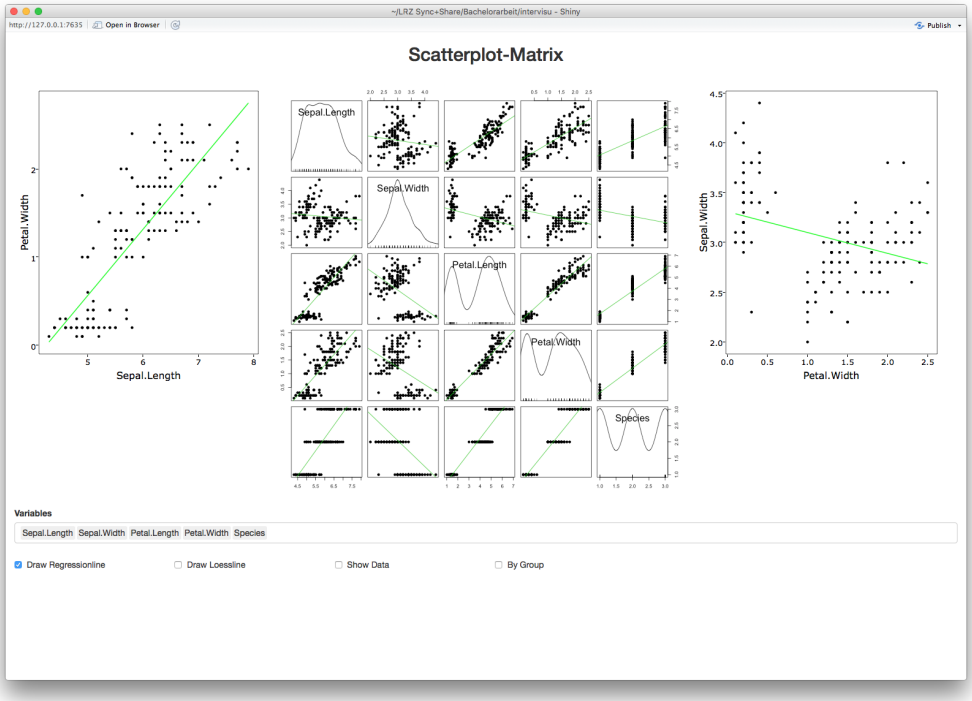

Abbildung 36: `Scatterplot_Matrix` 8

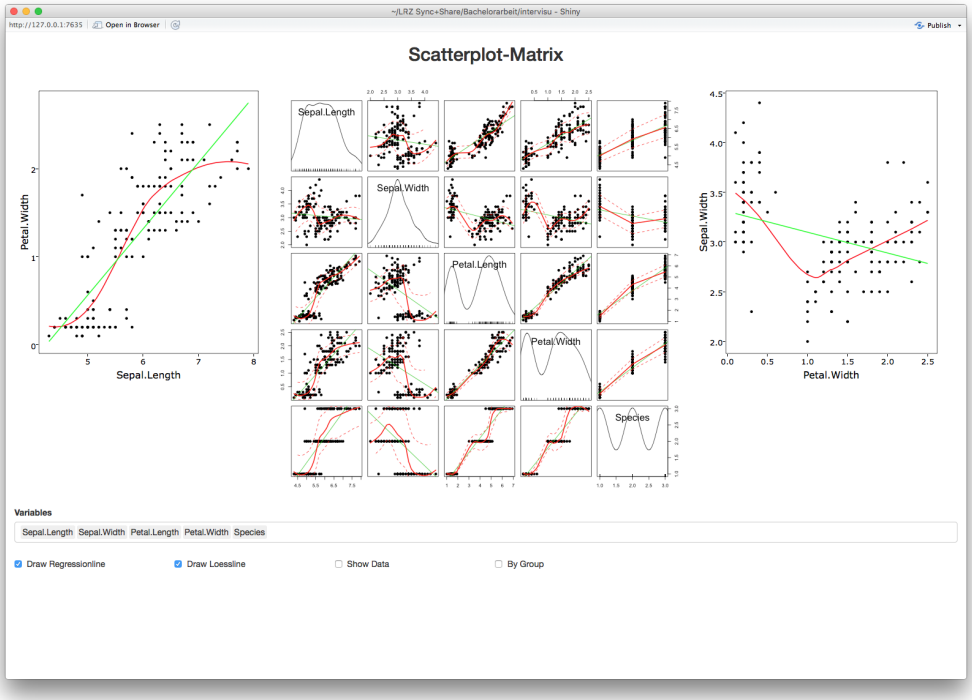

Abbildung 37: `Scatterplot_Matrix` 9

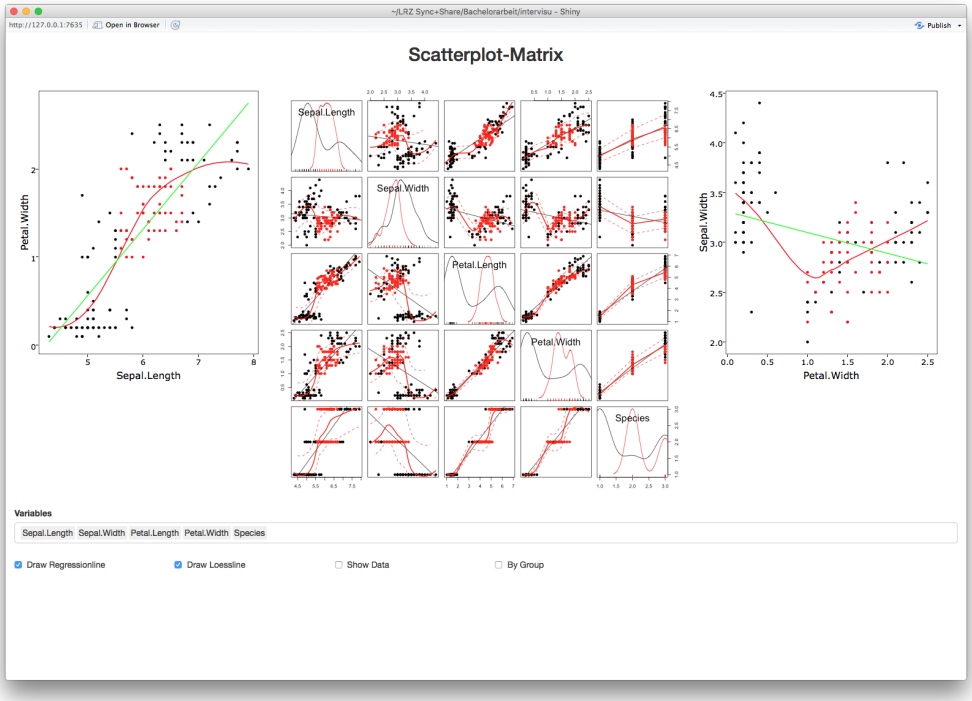

Abbildung 38: `Scatterplot_Matrix` 10

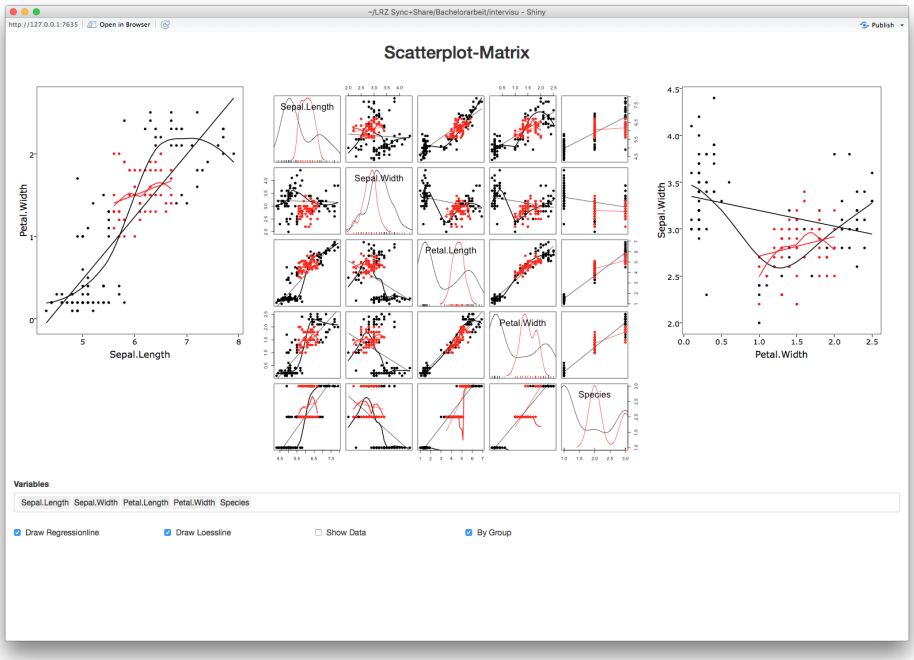

Abbildung 39: `Scatterplot_Matrix` 11

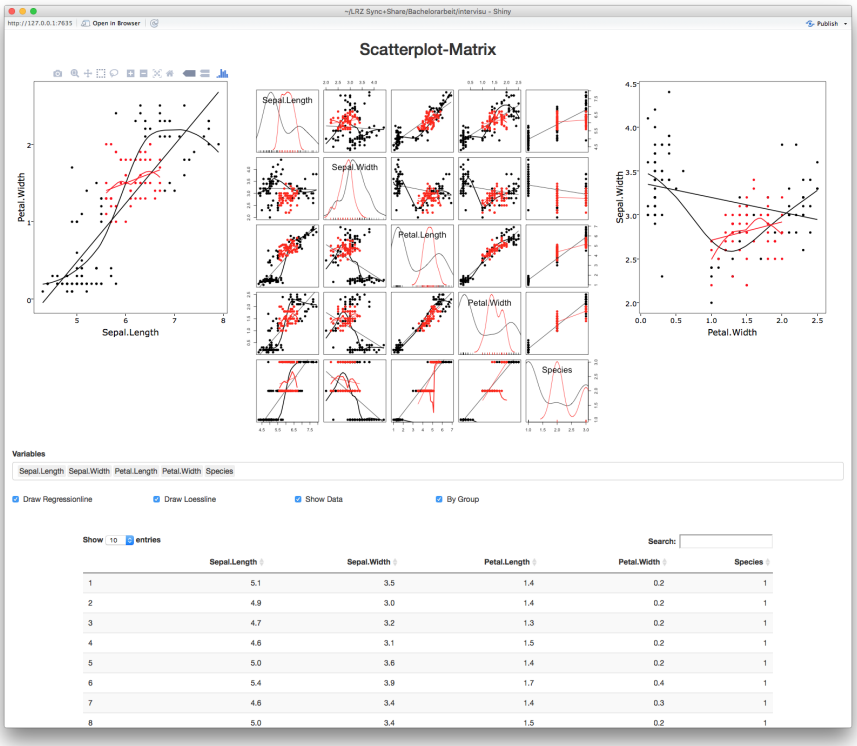

Abbildung 40: `Scatterplot_Matrix` 12

4.1.3 `Smoothing_Analysis`

In der Anwendung lassen sich alle der nun vorgestellten Arten der Modellierung interaktiv verändern. Einzig ausgeschlossener Parameter ist die Anzahl an Knoten bei Polynom-Splines, außer bei nicht penalisierten B-Splines.
Standardmäßig wird zu Beginn eine polynomiale Regression zweiten Grades, also eine lineare Regression, der ersten übergebenen Variable des Datensatzes auf die zweite in einem Scatterplot dargestellt (Abbildung 41). Das α-Niveau der dargestellten Konfidenzbändern kann beliebig in dem Intervall $[0, 0.2]$ gewählt werden (Abbildung 42). Die Einflussvariablen können auch transformiert werden (Abbildung 43). Diese werden für jede Variable einzeln bestimmt. Implementierte Funktionen f sind die Identität ($\mathbb{I}$), die Exponentialfunktion, die Sinus- und Kosinusfunktion und der natürliche Logarithmus, falls Y non-negativ ist.

Def. 4.1 Transformation von Merkmalen: Gegeben sei ein Merkmal X mit den n Beobachtungen $x_1, \ldots, x_n$. Eine Transformation T eines Merkmals ist eine durch f definierte Funktion, für die gilt:

$$T : \mathbb{R}^n \to \mathbb{R}^n$$
$$T\big((x_1, \ldots, x_n)\big) = \big(f(x_1), \ldots, f(x_n)\big)$$

Mögliche grundlegende Arten der Regression sind entweder die polynomiale Regression oder Polynomial-Splines (Abbildung 44).
Ist die polynomiale Regression ausgewählt, lässt sich der Grad r des gefitteten Polynoms $f(x)$ numerisch auswählen (Abbildung 45). Für r hat $r \in \mathbb{N}$ zu gelten.
Entscheidet man sich für Polynomial-Splines, können unterschiedliche Basisfunktion ausgewählt werden (Abbildung 46). Alle im Methodenabschnitt besprochenen Basen wurden implementiert - die *Basis der trunkierten Potenzen* Def.3.12, die *B-Spline Basis* Def.3.13 und die *kubische Basis* Def.3.14.
Falls B-Splines gewünscht sind, kann man einfach P-Splines verwenden, darf diese jedoch nicht penalisieren. In dieser Modifikation kann der Grad l der Polynomial-Splines und die Anzahl an Knoten m gewählt werden (Abbildung 47 und Abbildung 48).
Auch die Linkfunktion und Verteilung von $Y|X$ lässt sich interaktiv bestimmen (Abbildung 49). Sind penalisierte Splines ausgewählt, kann der Anwender auch den Parameter λ bestimmen, um die Tragweite der Penalisierung festzulegen (Abbildung 50).

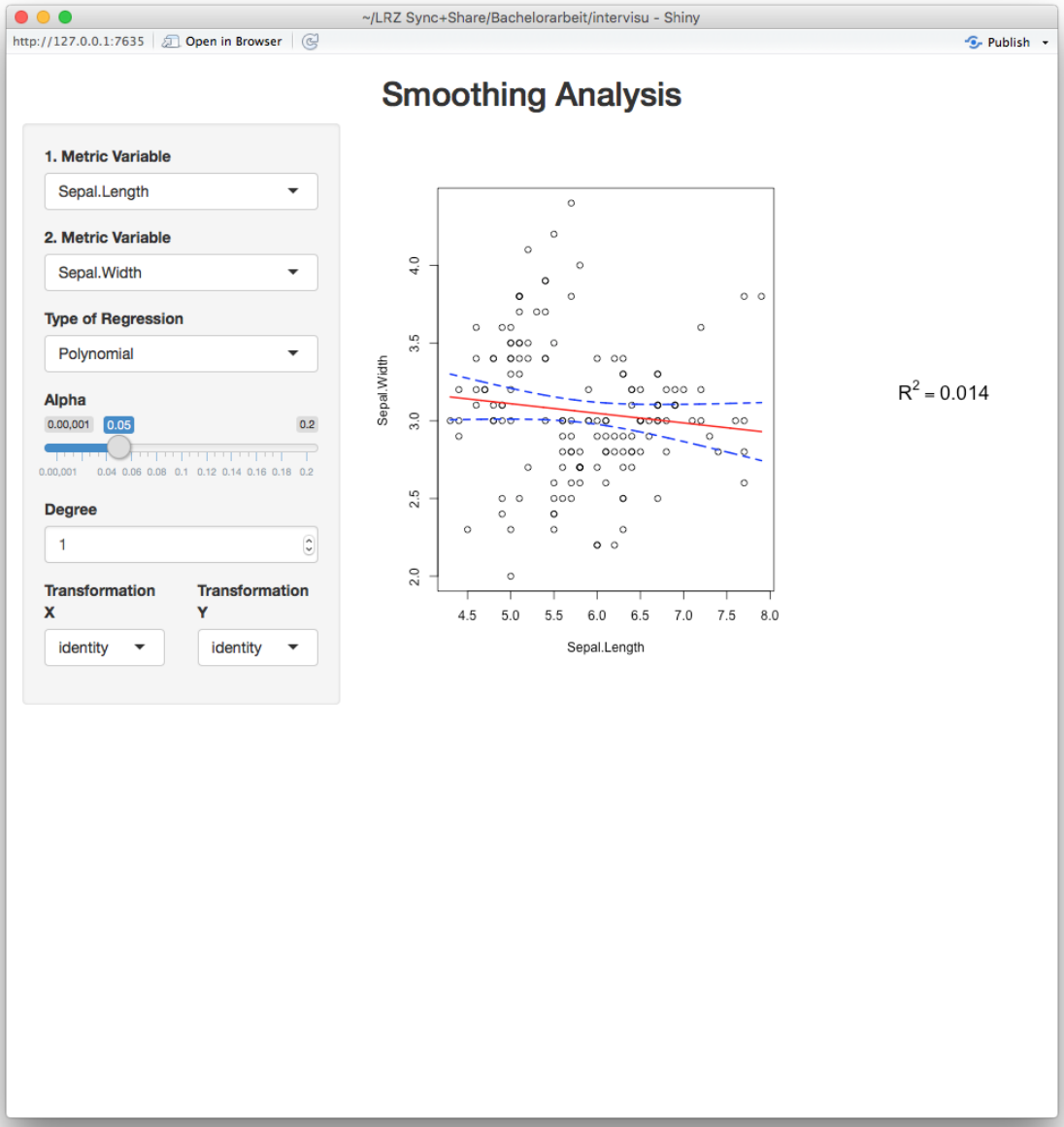

Abbildung 41: `Smoothing_Analysis` 1

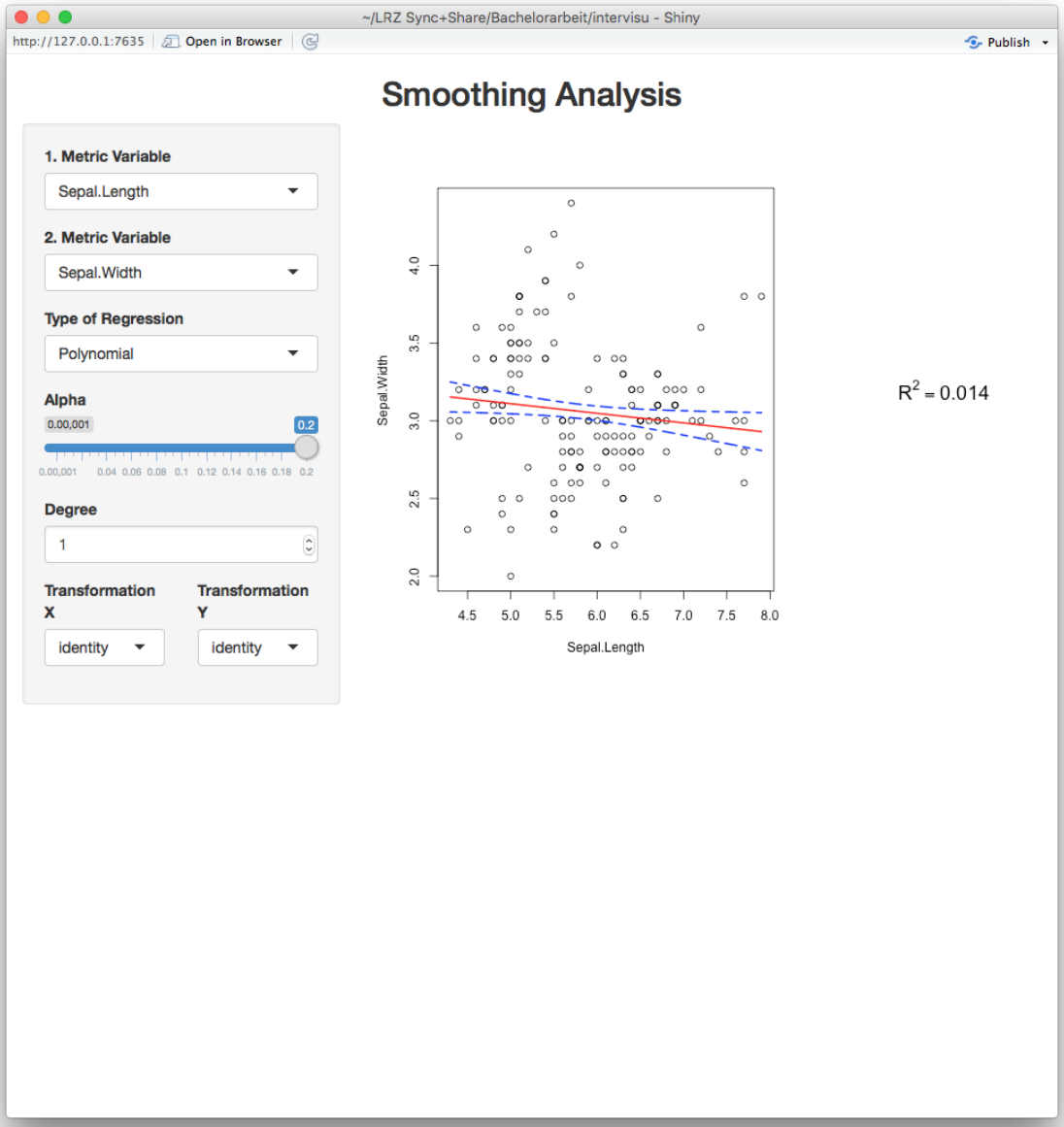

Abbildung 42: `Smoothing_Analysis 2`

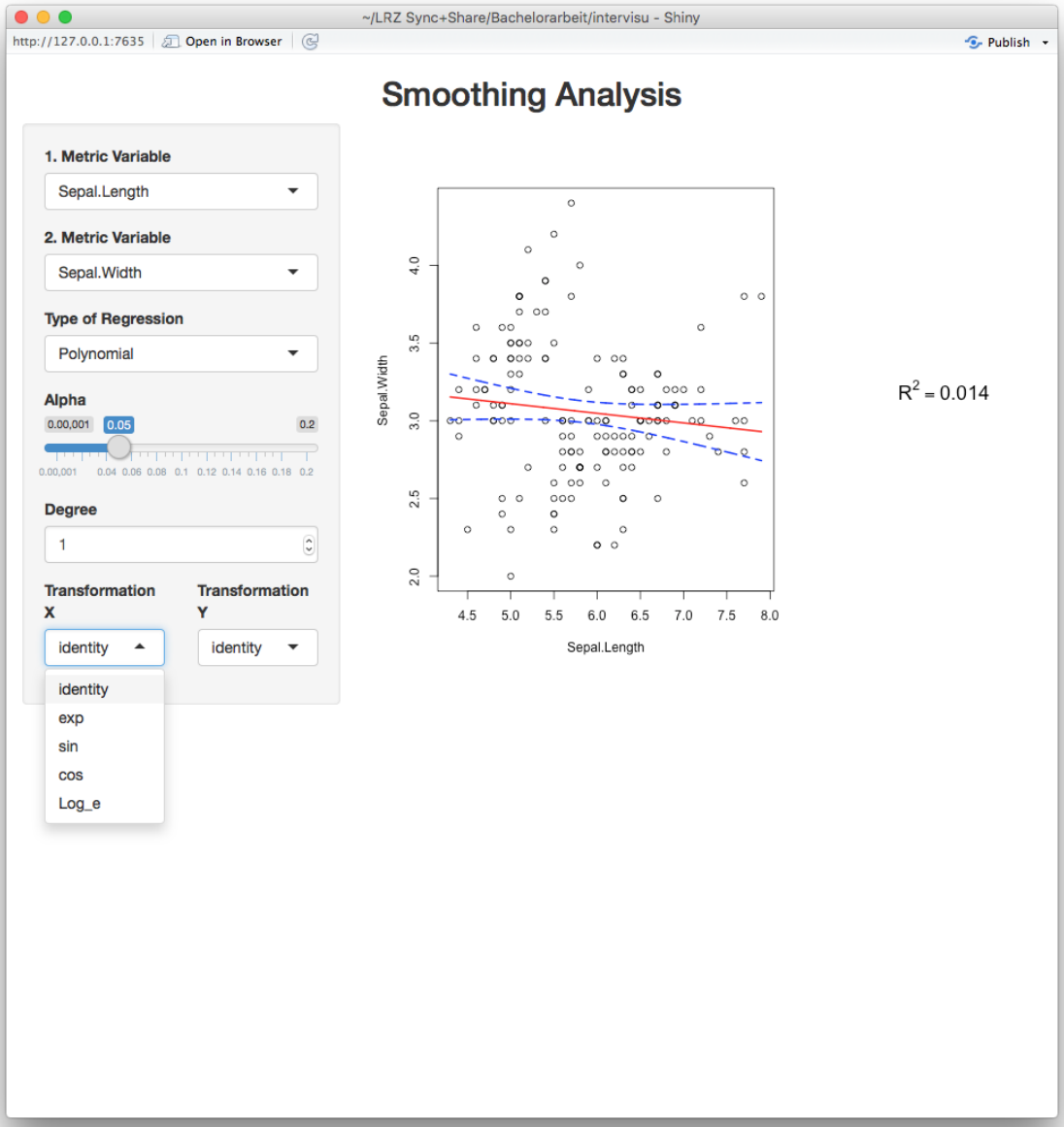

Abbildung 43: `Smoothing_Analysis` 3

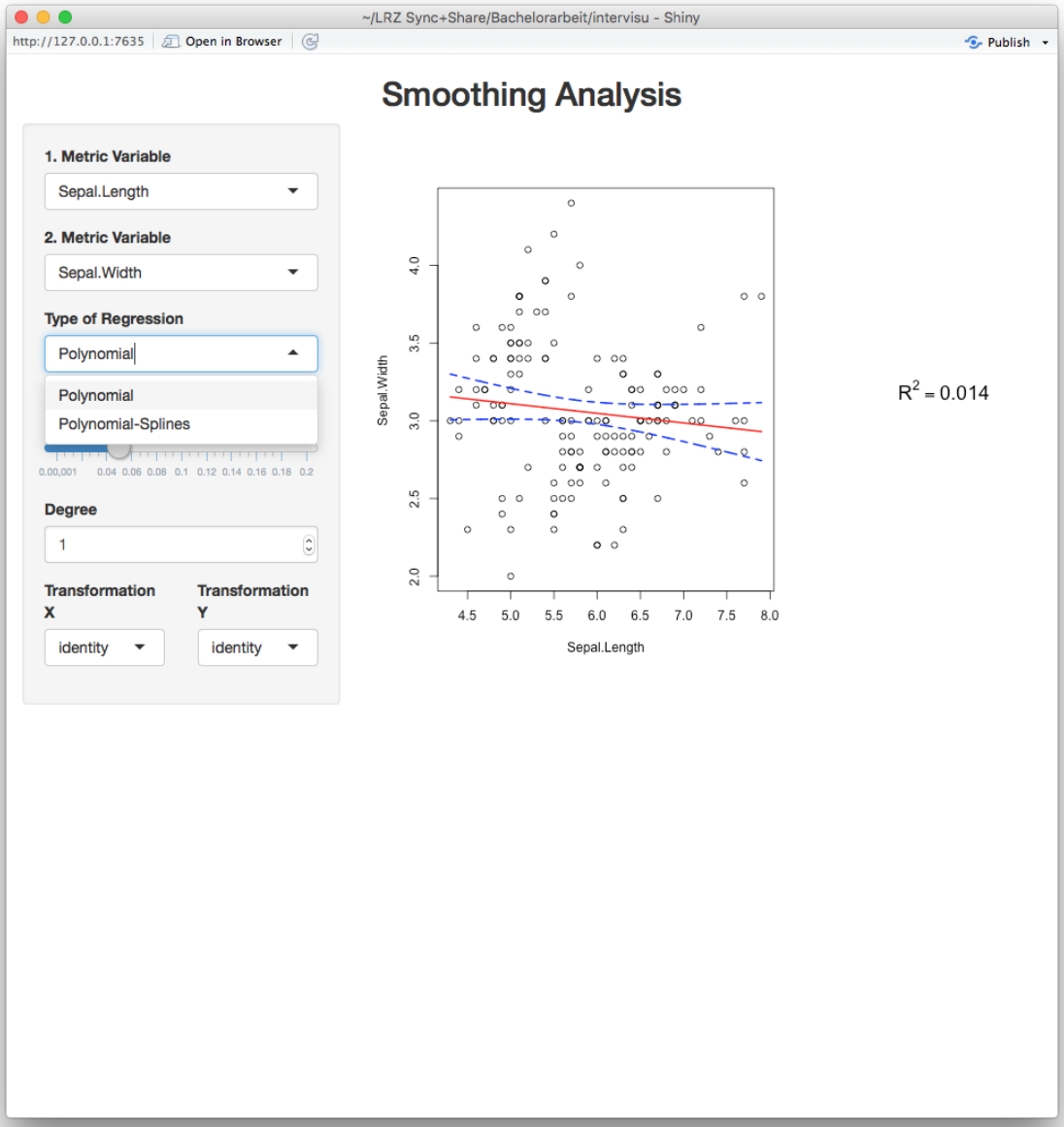

Abbildung 44: `Smoothing_Analysis` 4

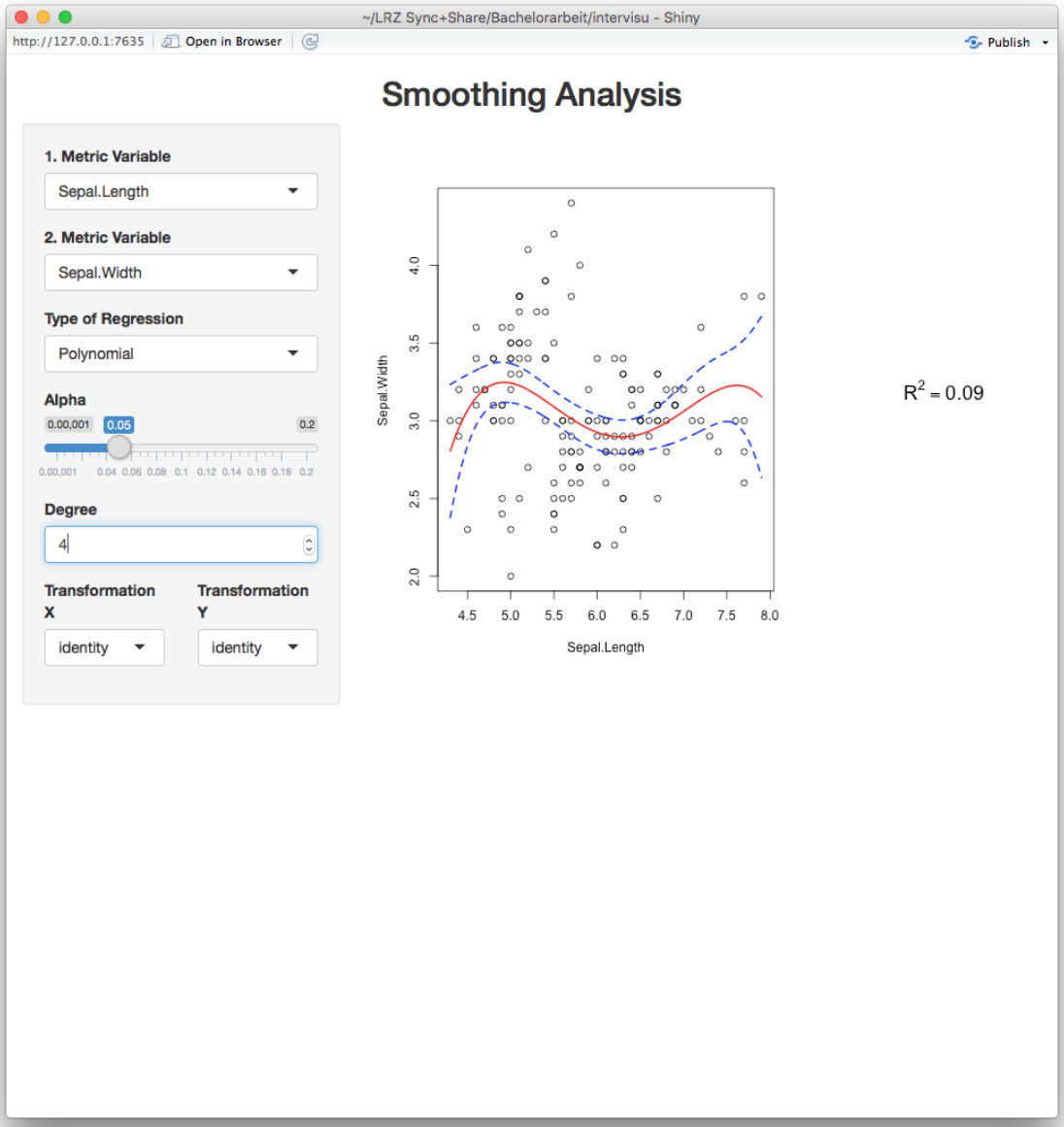

Abbildung 45: `Smoothing_Analysis` 5

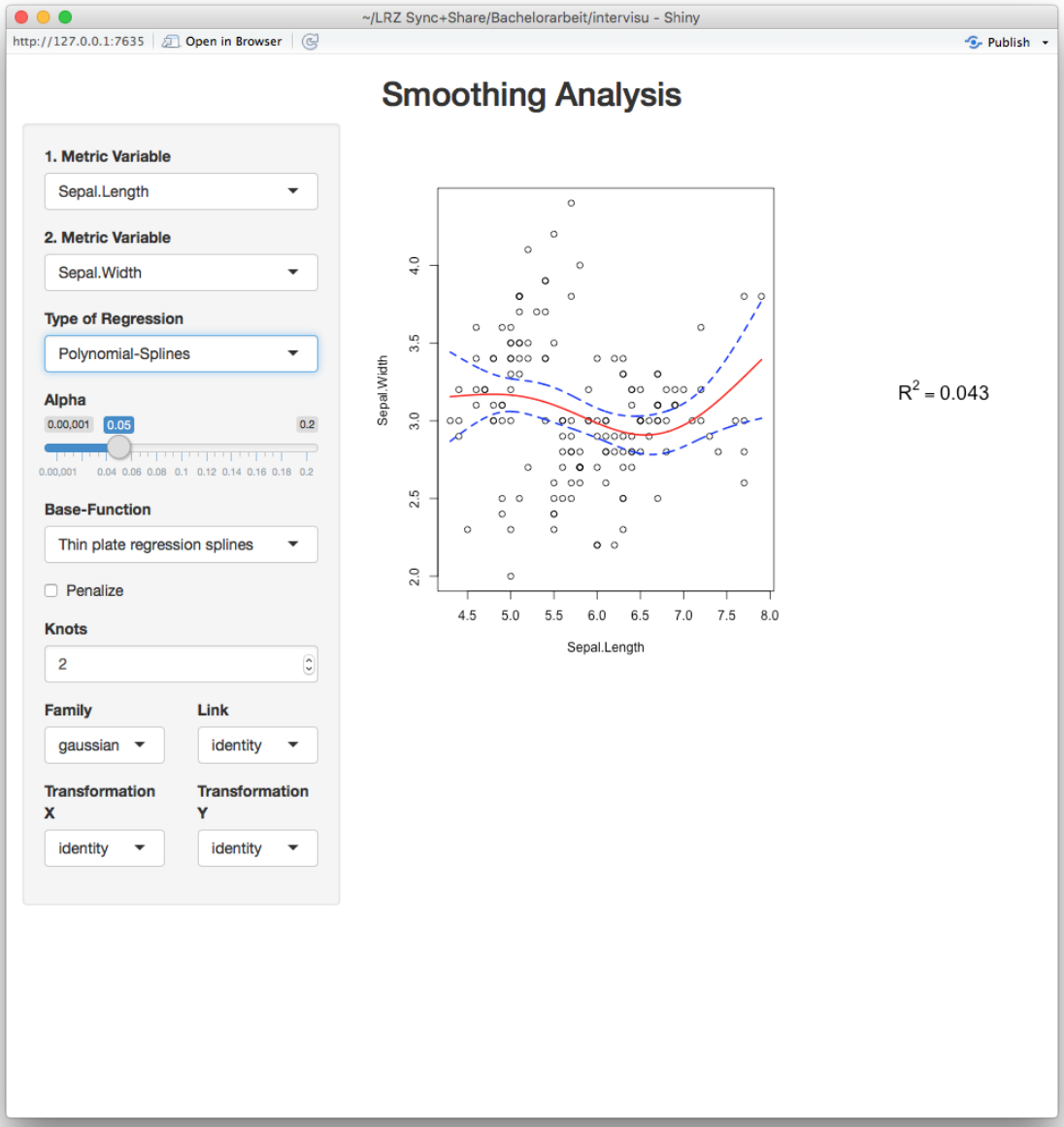

Abbildung 46: `Smoothing_Analysis` 6

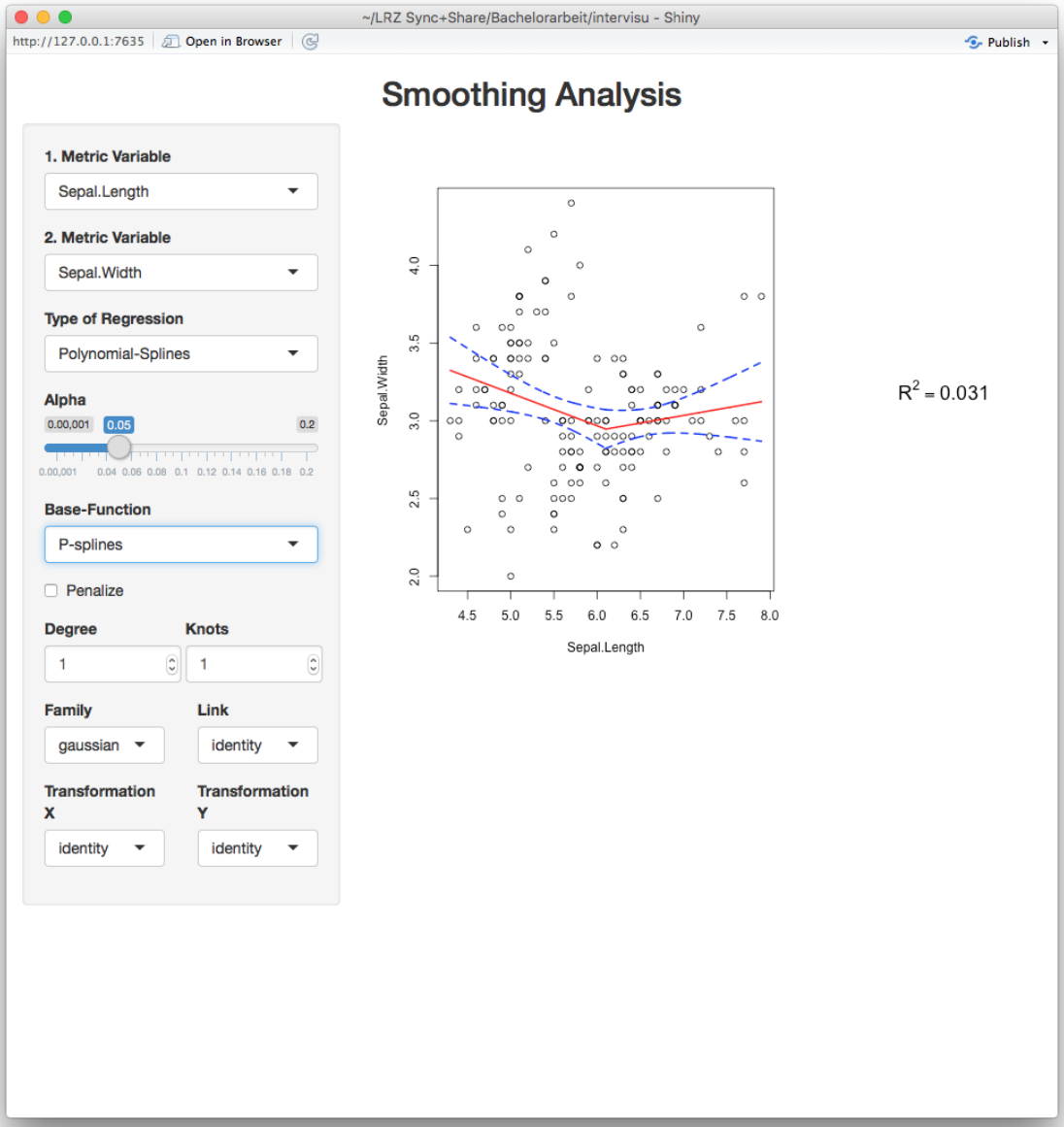

Abbildung 47: `Smoothing_Analysis` 7

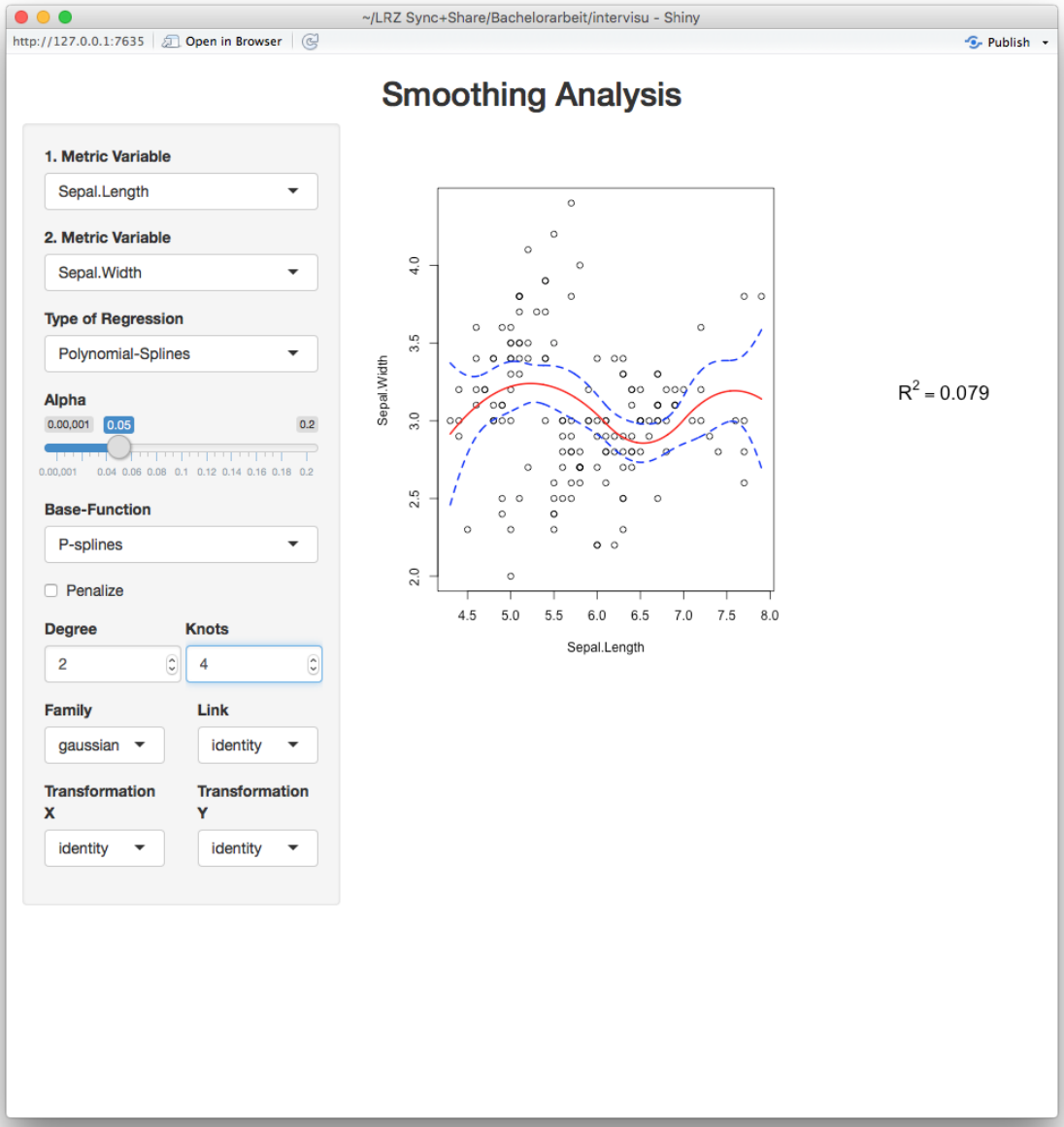

Abbildung 48: `Smoothing_Analysis` 8

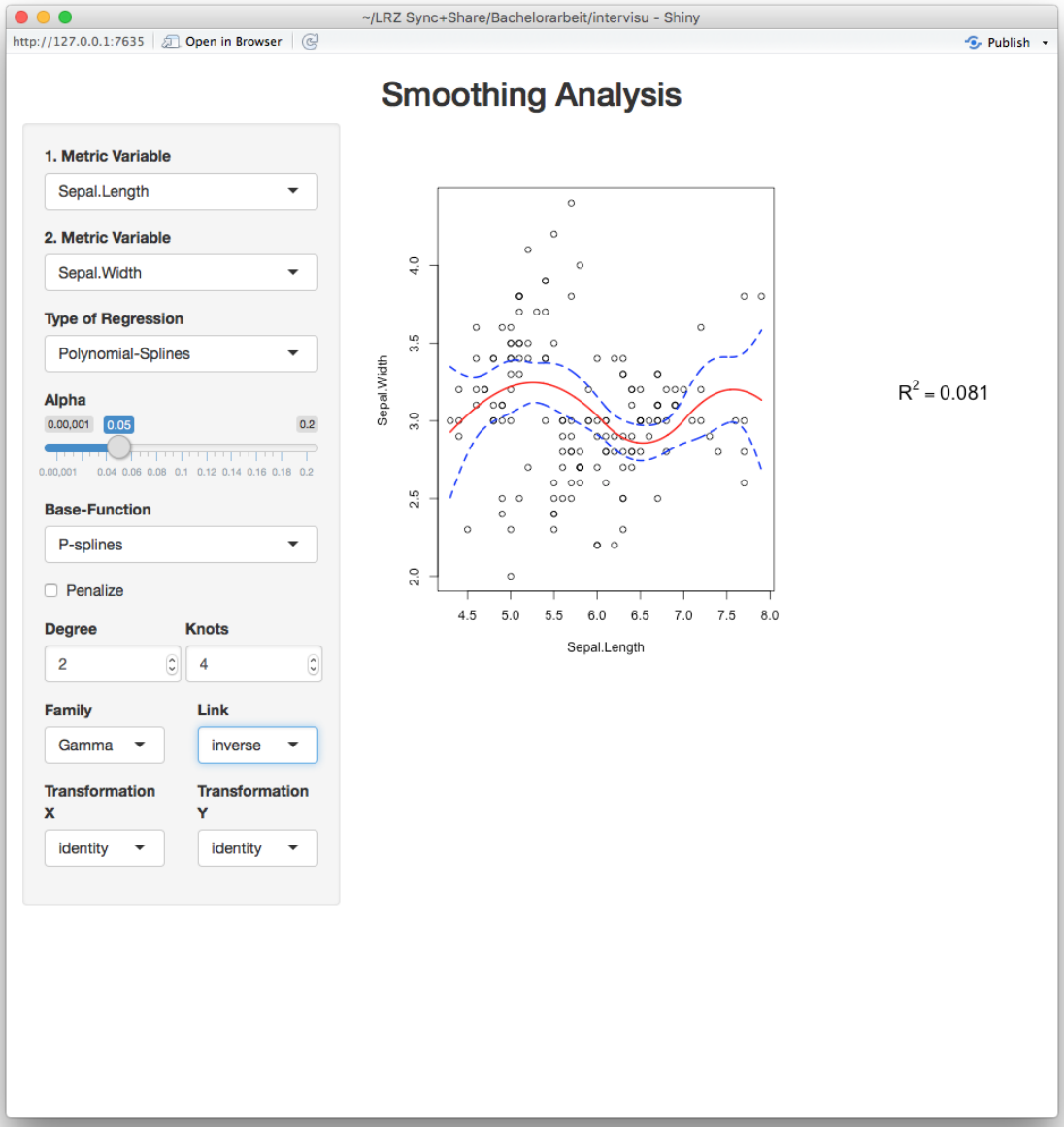

Abbildung 49: `Smoothing_Analysis` 9

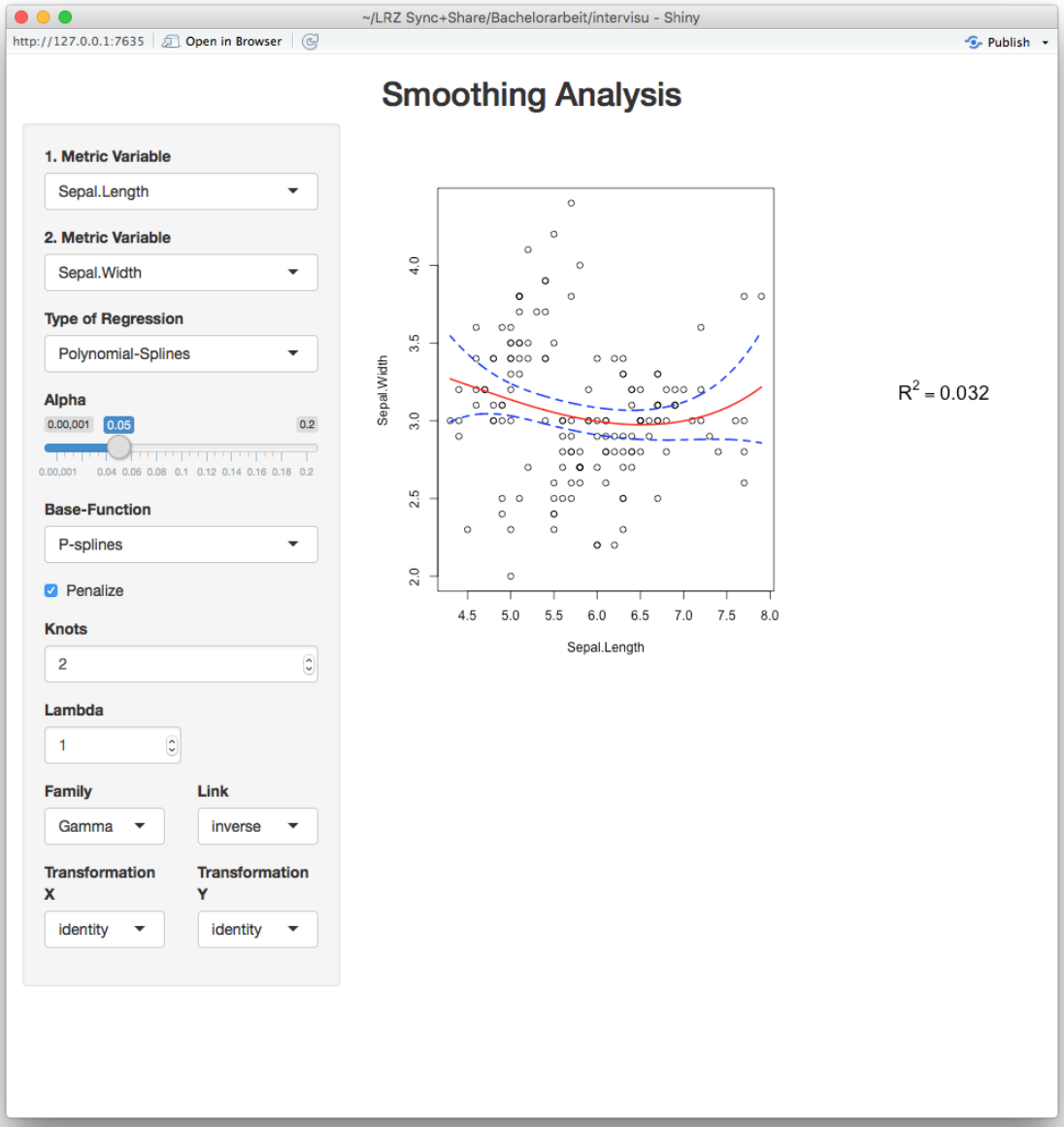

Abbildung 50: `Smoothing_Analysis` 10

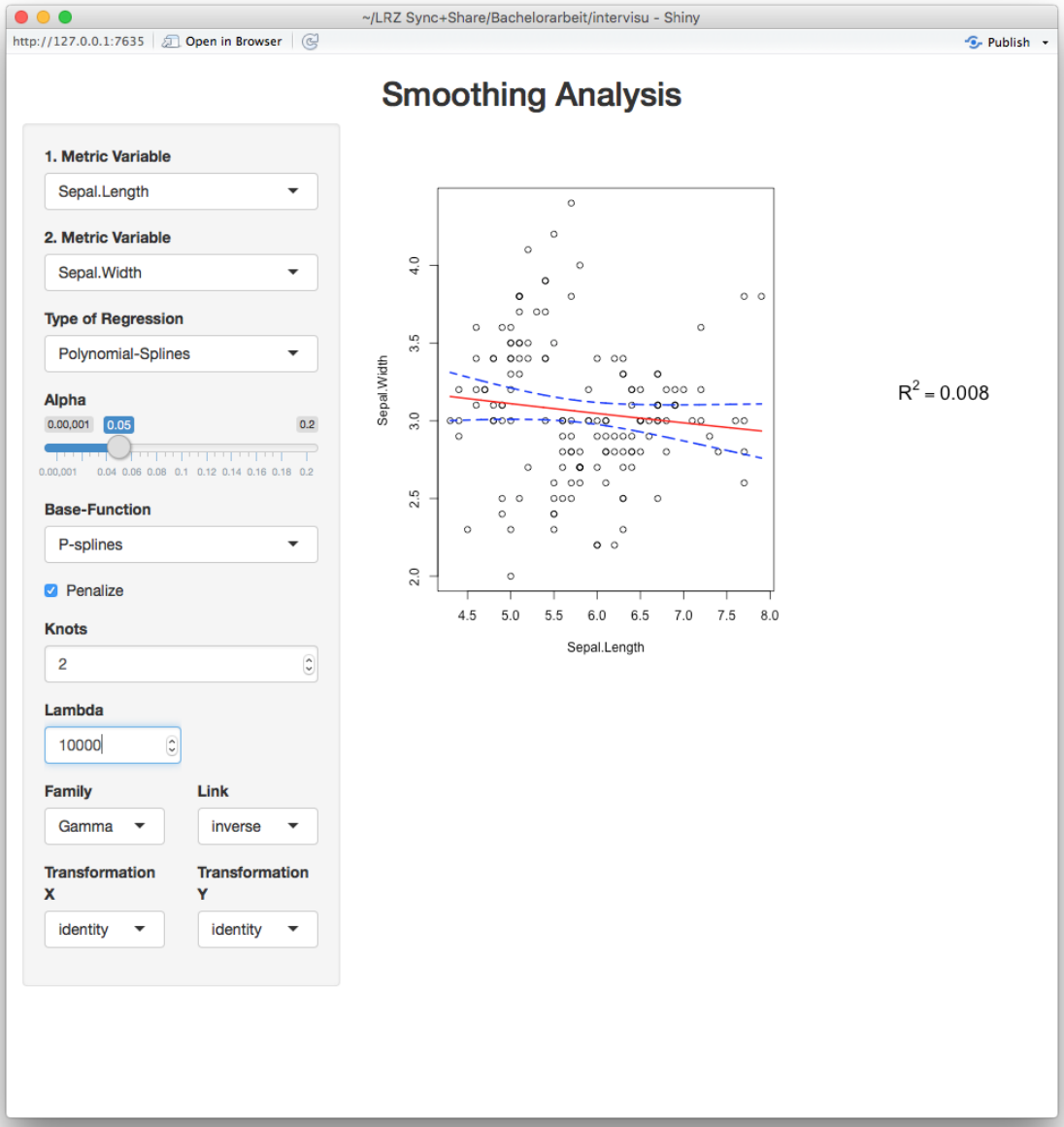

Abbildung 51: `Smoothing_Analysis` 11

4.1.4 `Scatterplot_3d`

Zuerst kann man die drei darzustellenden Variablen auswählen (Abbildung 52). Der Betrachtungswinkel des dreidimensionalen Scatterplots lässt sich interaktiv durch das Bewegen des Cursors bei gedrückter Maustaste auf den Graphen bestimmen. Dabei lässt sich der Winkel horizontal und vertikal verändern, um einen besseren räumlichen Eindruck der dargestellten Daten zu gewinnen.
Benützt der Anwender den Button *Condition on a fourth variable*, wird ein Slider eingeblendet, der einen Wert des vierten Merkmals bestimmt, um welchen 30% der Daten angezeigt werden sollen (Abbildung 53). Interaktiv kann man diesen Wert in dem Wertebereich des vierten Merkmals bewegen (Abbildung 54).
Mithilfe der *brush*-Interaktion lassen sich zuletzt auch bestimmte rechteckige Punktemengen isolieren, welche in beiden Graphen rot eingefärbt werden (Abbildung 55).

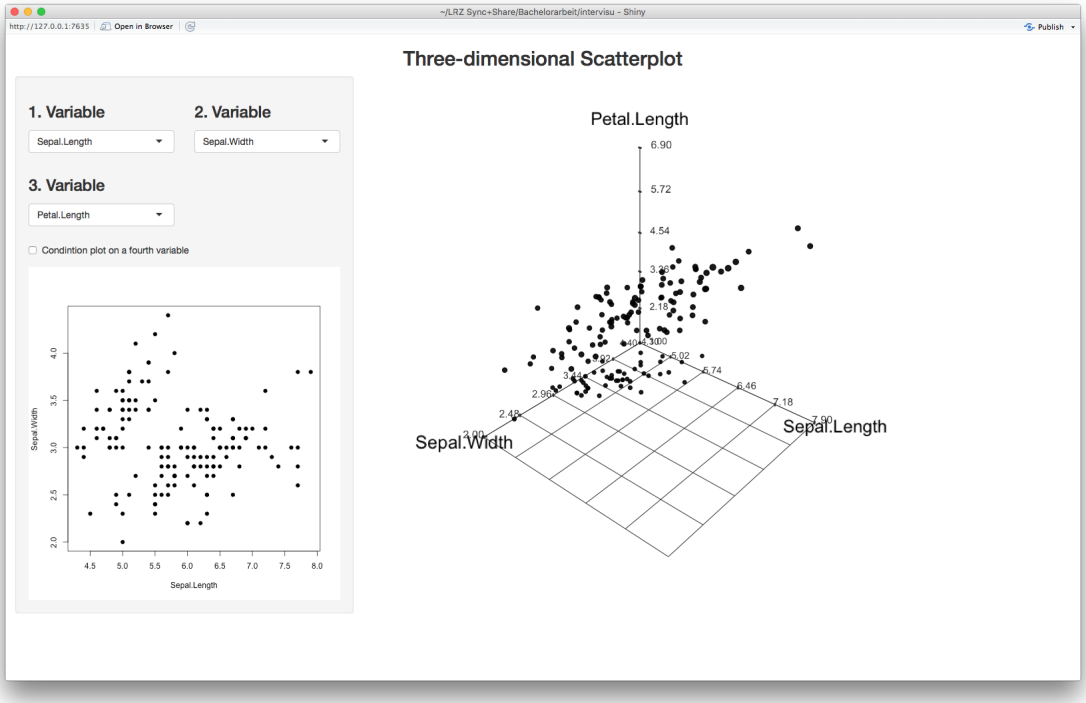

Abbildung 52: `Scatterplot_3d` 1

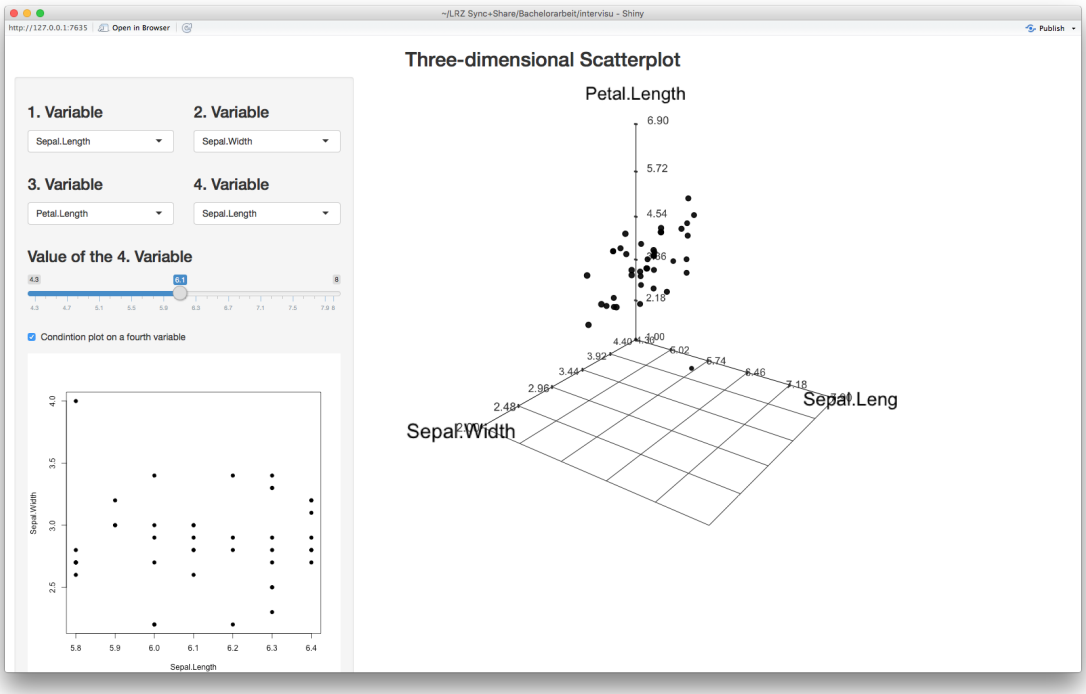

Abbildung 53: `Scatterplot_3d` 2

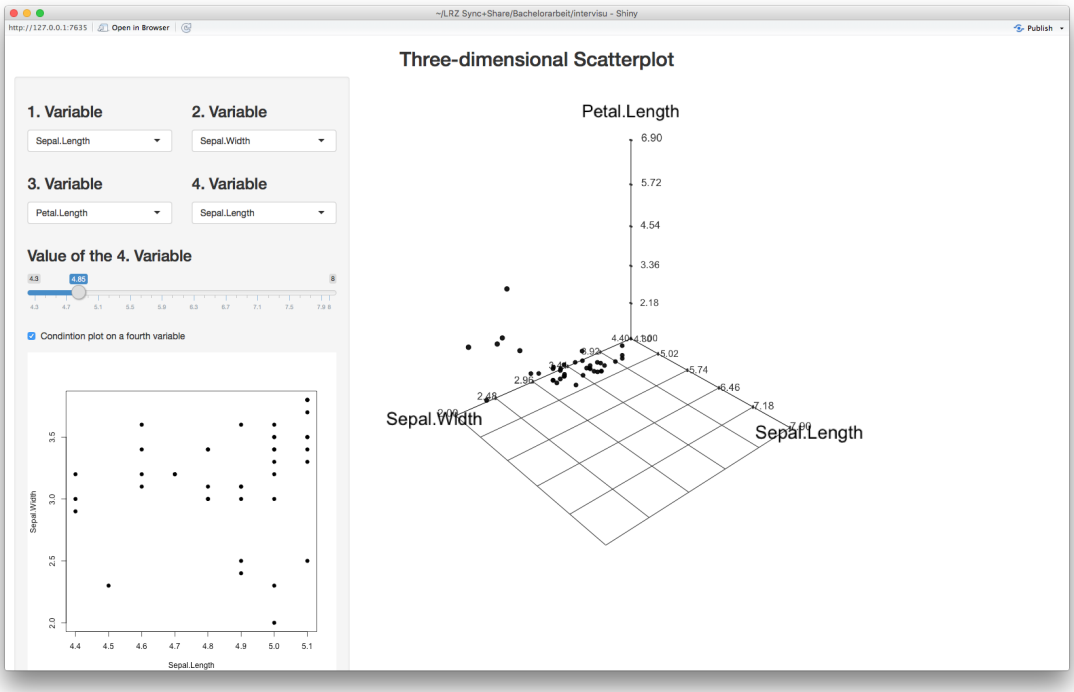

Abbildung 54: `Scatterplot_3d` 3

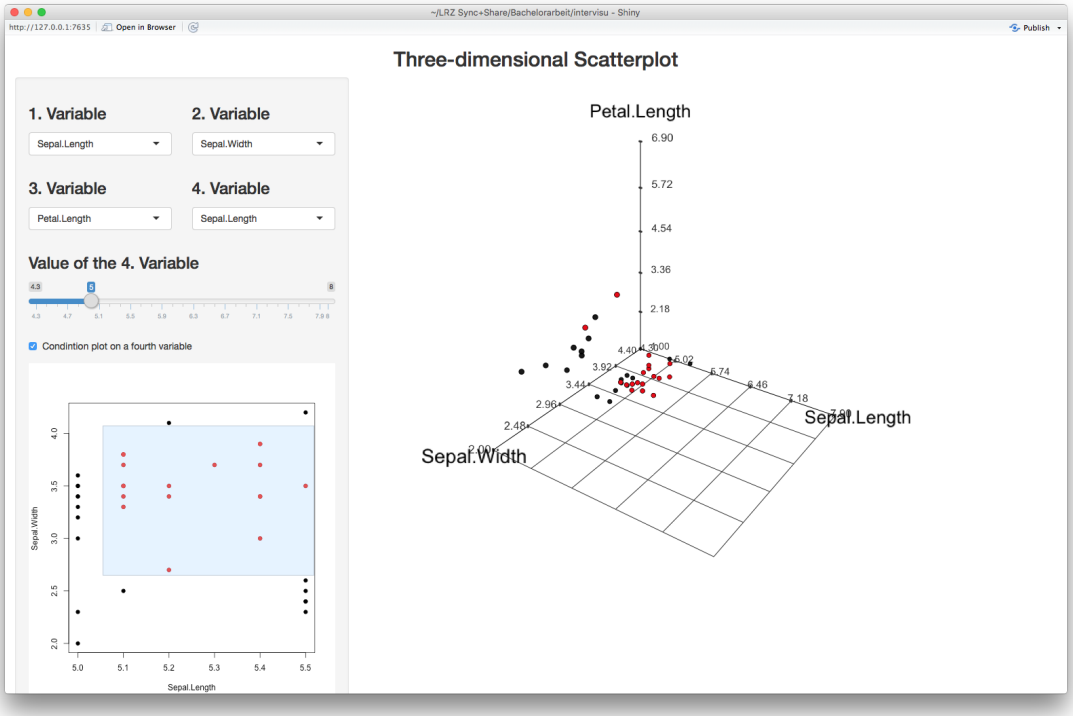

Abbildung 55: `Scatterplot_3d` 4

4.2 Kategoriale Variablen

4.2.1 `Group_Boxplot`

Um diese Funktion verwenden zu können, muss es in dem übergebenem Datensatz mindestens ein metrisches und kategoriales Merkmal geben. Welche der Merkmale als kategoriale beziehungsweise metrische dargestellt werden, wird wie bei den vorherigen Applikationen bestimmt (Abbildung 56).
Um zwei Faktorlevel miteinander fusionieren zu können, muss ein erstes Level durch einen einfachen Klick auf den zugehörigen Boxplot ausgewählt werden, der nun farblich hervorgehoben wird (Abbildung 57). Klickt man hinterher auf einen weiteren Boxplot, werden diese Faktorlevel zu einem verschmolzen. In dem Beispiel werden die Faktorlevel "4" und "8" des Merkmals Zylinder zu dem neuen Faktorlevel "4 & 8 " geformt (Abbildung 58).
Neben der bereits genannten Interaktion um bestimmte Faktorlevel zu fusionieren, gibt es auch die Möglichkeit sich den globalen Mittelwert in den Boxplots über alle Faktorlevel einzeichnen zu lassen (Abbildung 59).

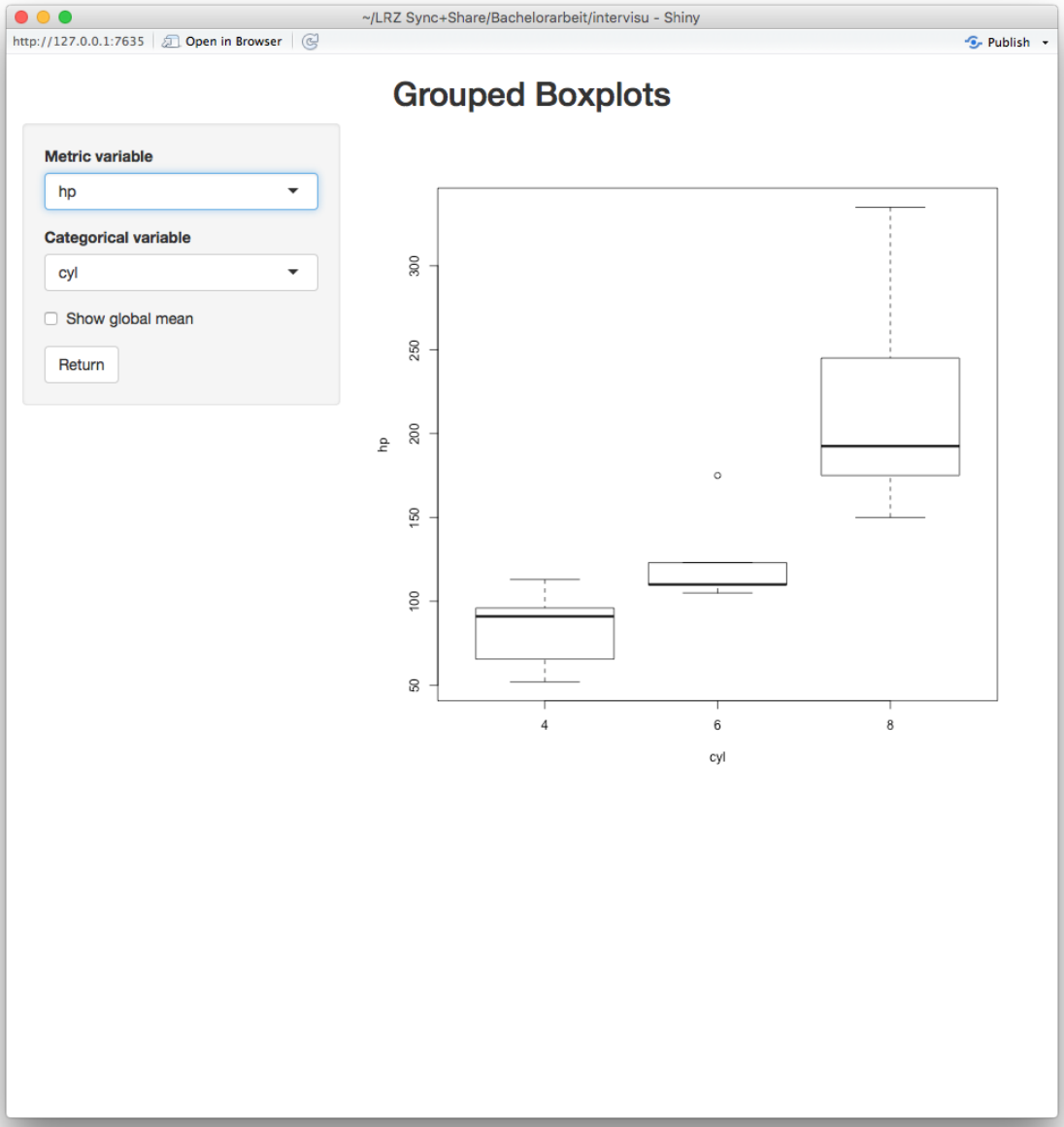

Abbildung 56: `Group_Boxplot` 2

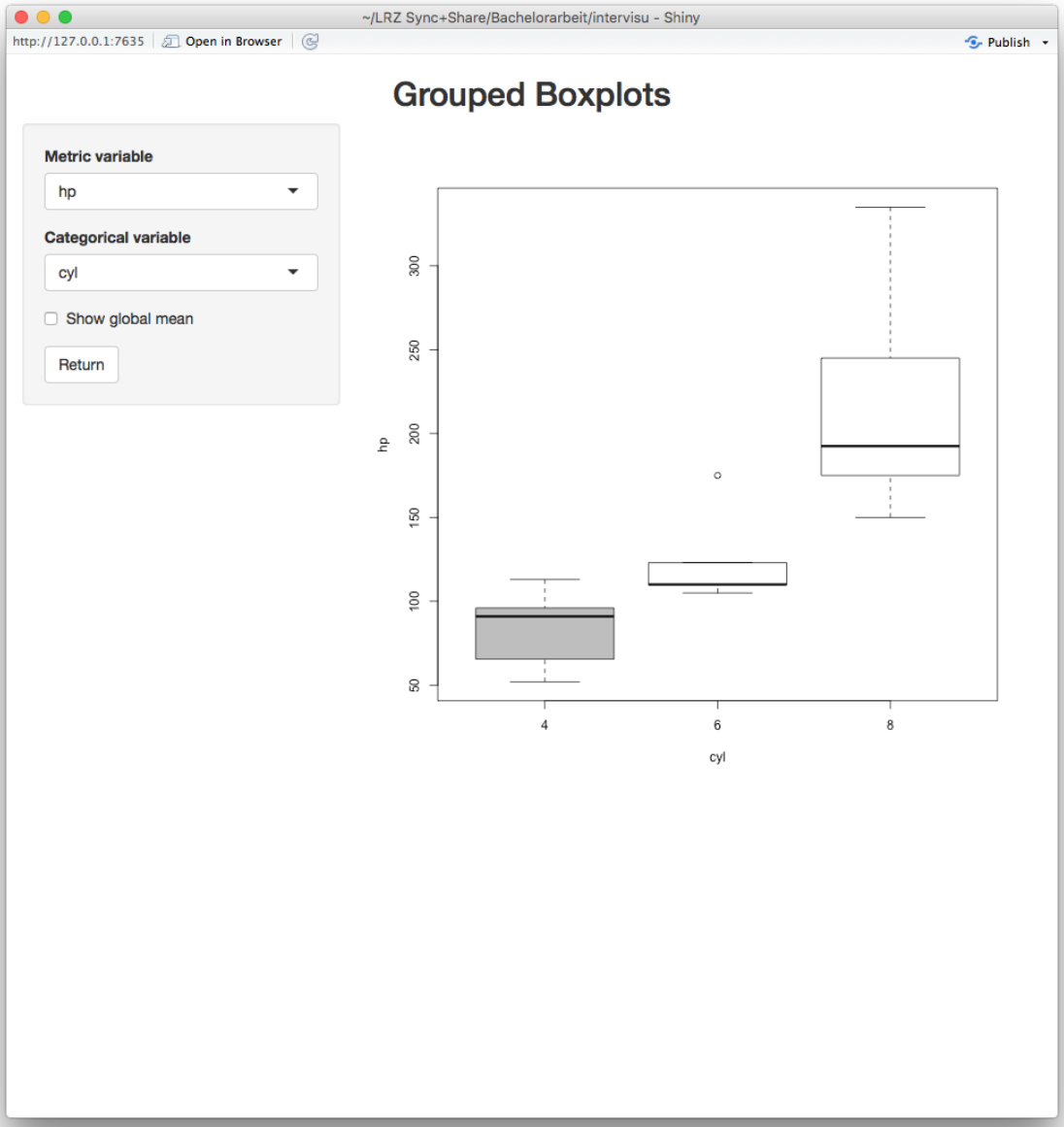

Abbildung 57: `Group_Boxplot` 3

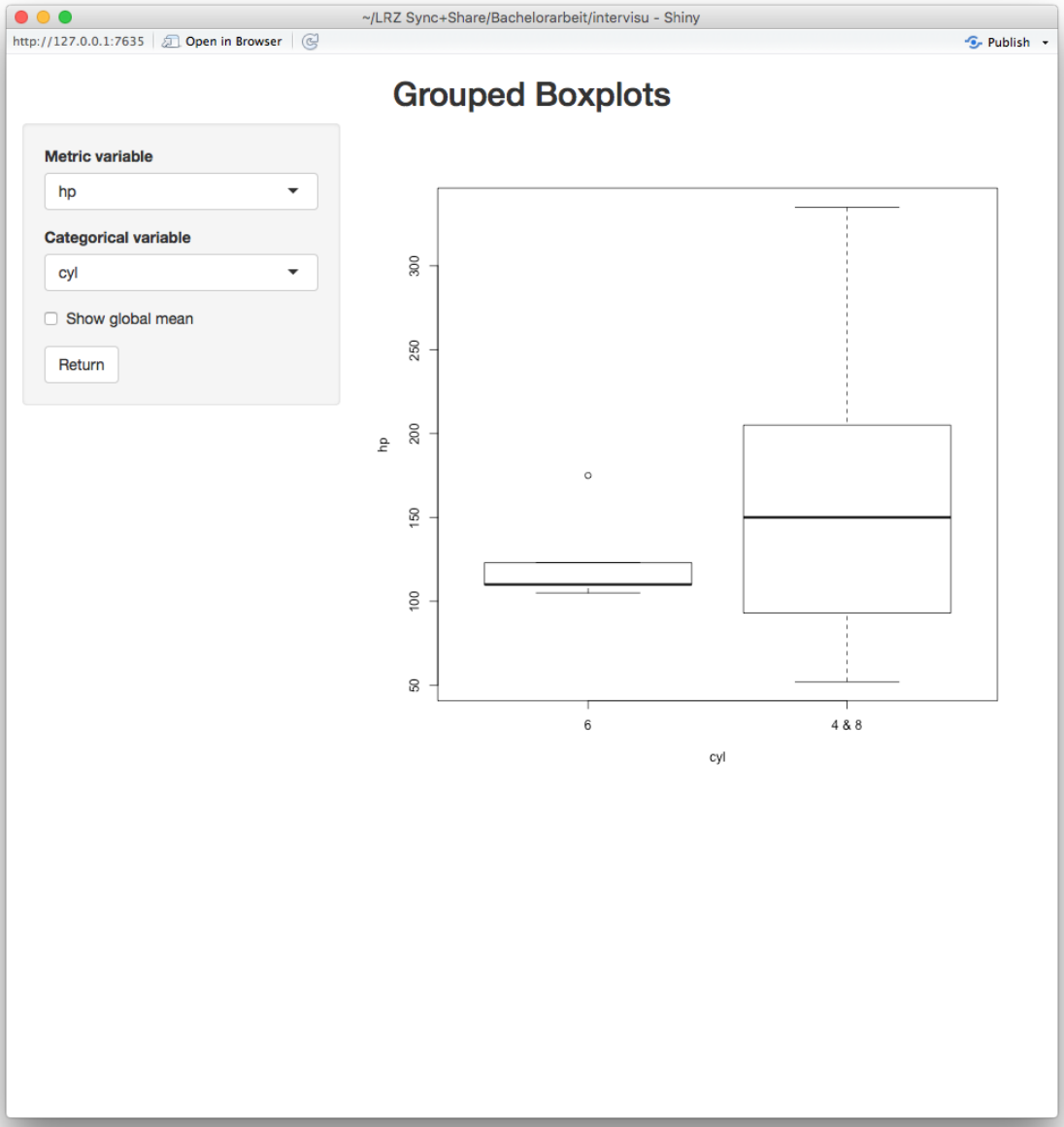

Abbildung 58: `Group_Boxplot` 4

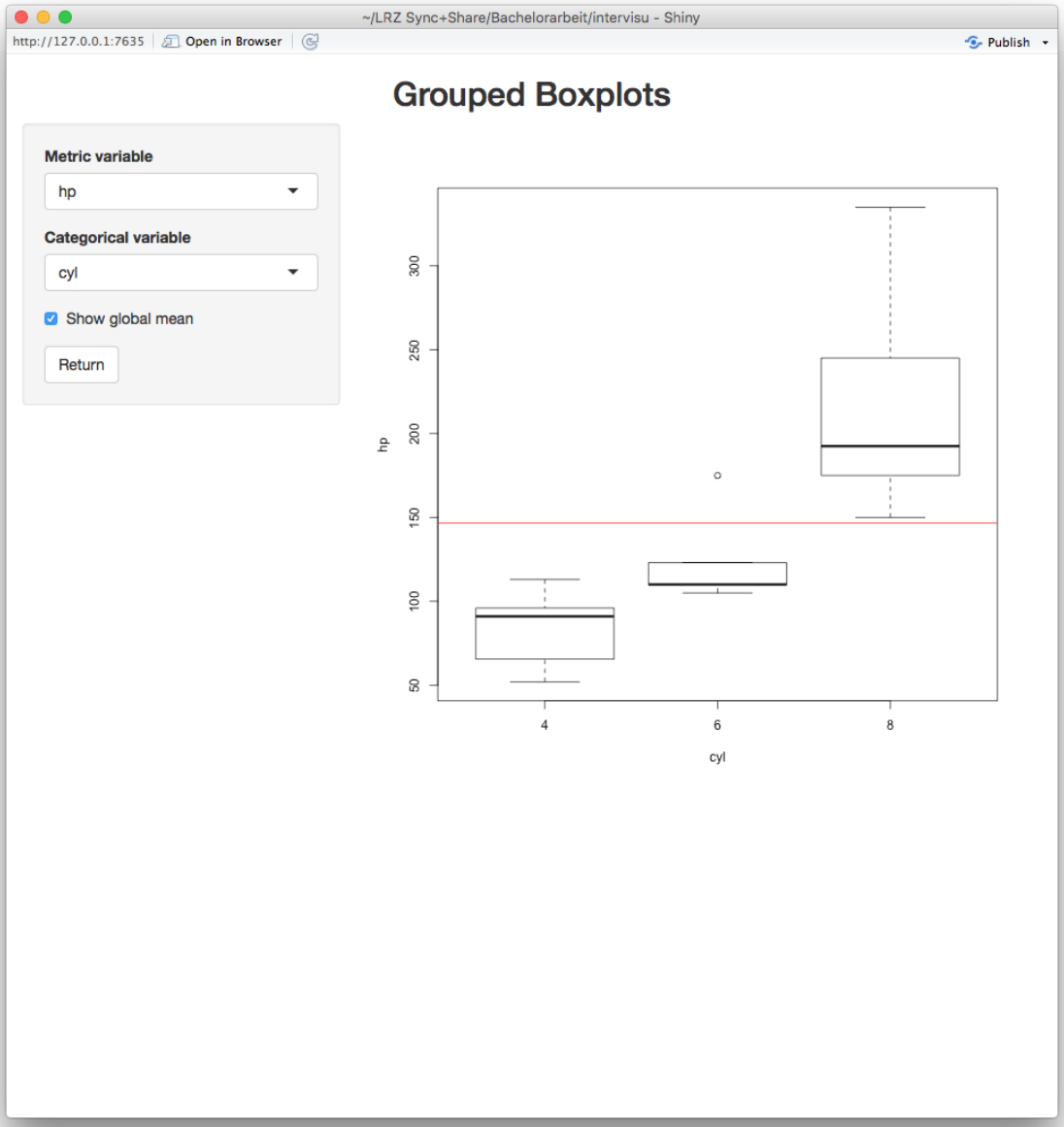

Abbildung 59: `Group_Boxplot` 5

4.2.2 `Stacked_Barplot`

Das Layout führt auf der linken Seite alle in den Darstellungen inkludierten Merkmale auf, welche beliebig interaktiv verändert werden können (Abbildung 60).
Klickt man mit einem Einfachklick auf einen Balken des Diagramms, wird unterhalb ein Boxplot angezeigt, der auf das ausgewählte Level der ersten Faktorvariable konditioniert ist (Abbildung 61). Bei einem Doppelklick auf einen weiteren Balken werden die beiden ausgewählten Faktorstufen analog zur Anwendung `Group_Boxplot` miteinander verbunden (Abbildung 62).
Es wird auch ein Test auf Unabhängigkeit der beiden kategorialen Variablen durchgeführt. Ob ein approximativer oder exakter Test gewählt wird, bestimmt das Programm selbst mit dem übergebenen Parameter `m`. Beispielsweise hat die Kontingenztafel der kategorialen Merkmale *am* und *vs* aus dem Datensatz *mtcars* mehr als die standardmäßig eingestellten $m = 5$ Beobachtungen in allen Zellen, womit ein approximativer Test verwendet werden kann (Abbildung 63).
Falls man verschiedene Möglichkeiten der Gruppierung ausprobieren will, gelangt man mit dem Knopf *Backwards* wieder zu der vorherigen Gruppierung.

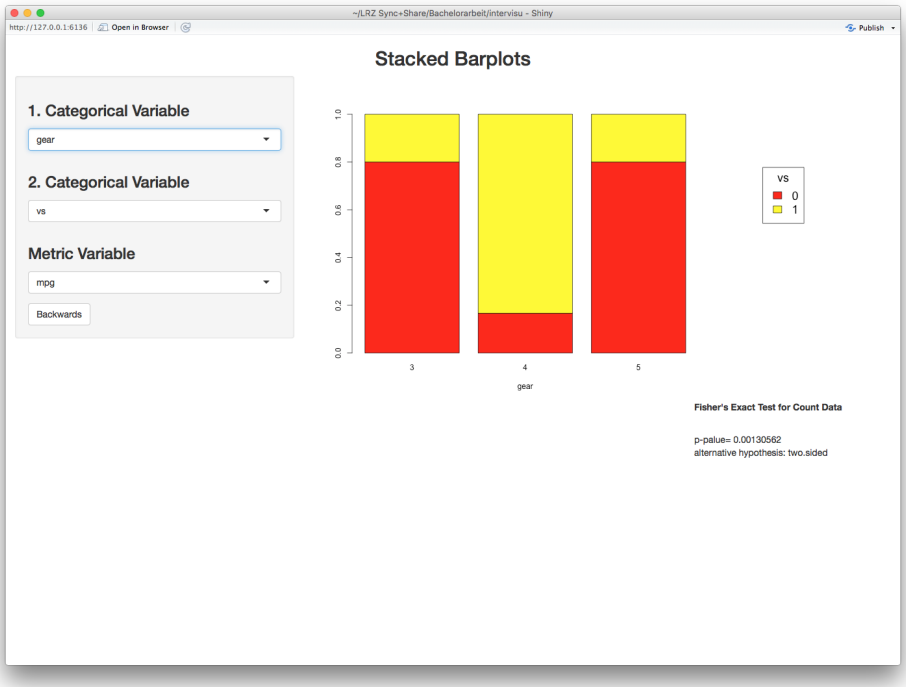

Abbildung 60: `Stacked_Barplot` 3

Abbildung 61: `Stacked_Barplot` 4

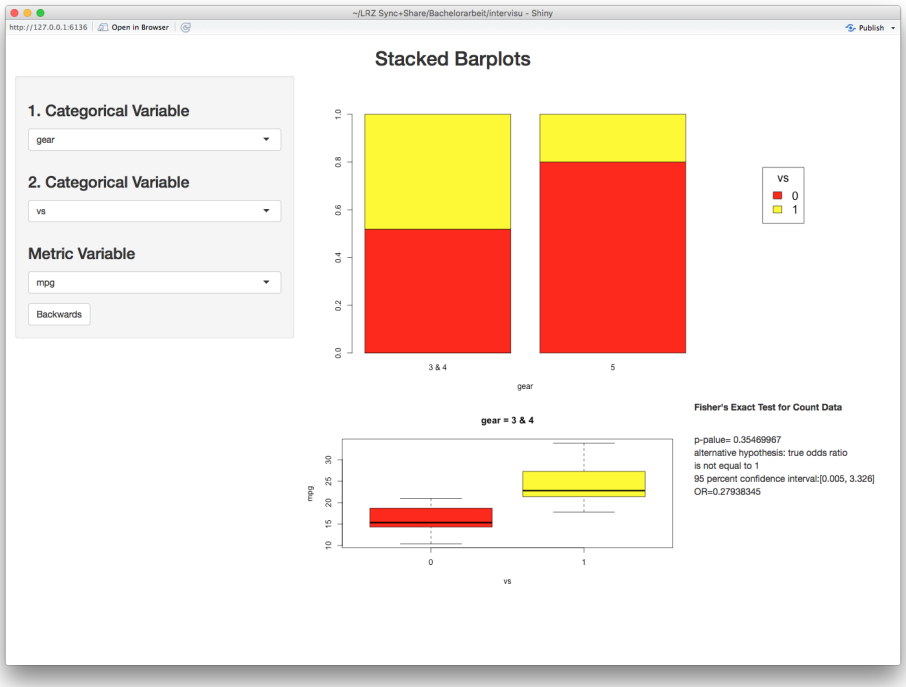

Abbildung 62: `Stacked_Barplot` 6

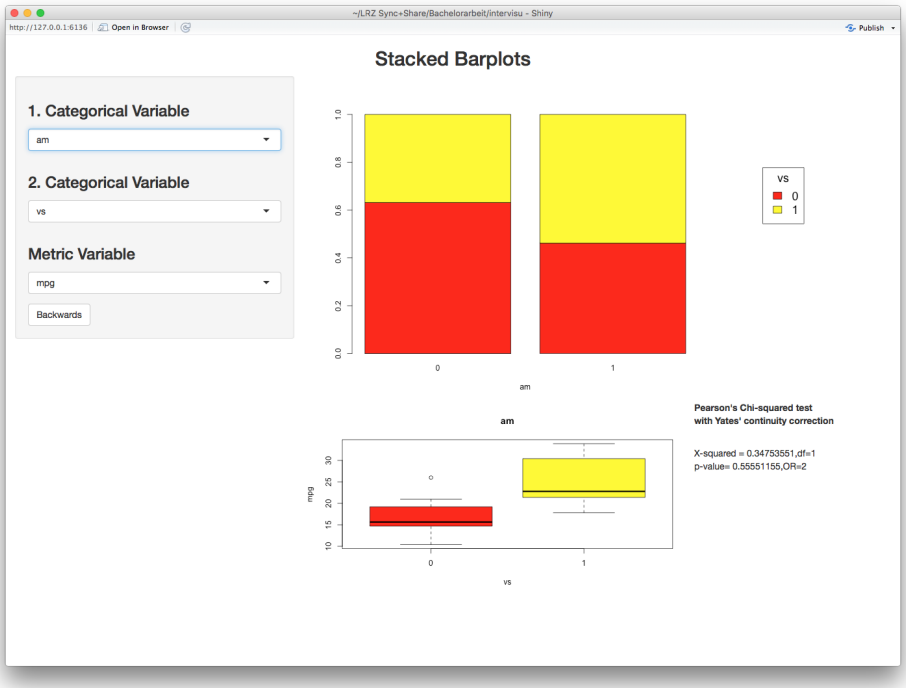

Abbildung 63: `Stacked_Barplot` 7

4.3 Zeitreihen: `Timeseries`

Um eine Zeitreihe in Trend- und Saisonkomponente zerlegen zu können, müssen dem Computer die Daten des Anfangs und Endes der Zeitreihe und die Frequenz der Messungen übergeben werden (Abbildung 64 und Abbildung 65).
Interaktiv können $n_{(o)}$ und $n_{(s)}$ gewählt werden. Dabei wird durch den Button *Robust Fitting* zwischen $n_{(o)} = 0$ und dem Laufen der äußeren Schleife bis zur Konvergenz der Schätzer entschieden (Abbildung 66). Mithilfe des Knopfs *Cycle-Window* kann zwischen den beiden Optionen $n_{(s)}$ gleich der Frequenz oder eigenen numerischen Werten gewählt werden (Abbildung 67).
Soll nur ein bestimmter Ausschnitt einer der Zeitreihenkomponenten dargestellt werden, kann der Anwender den Startpunkt des Abschnitts durch einen Einfachklick indizieren und bei gedrückter Maustaste den Cursor bis zum Endpunkt ziehen. In Abbildung 68 wird in jedem Graphen ein verschiedenes Zeitfenster dargestellt. Will man anschließend wieder die gesamte Zeitspanne betrachten, wird dies durch einen Doppelklick auf den gewünschten Graphen indiziert.

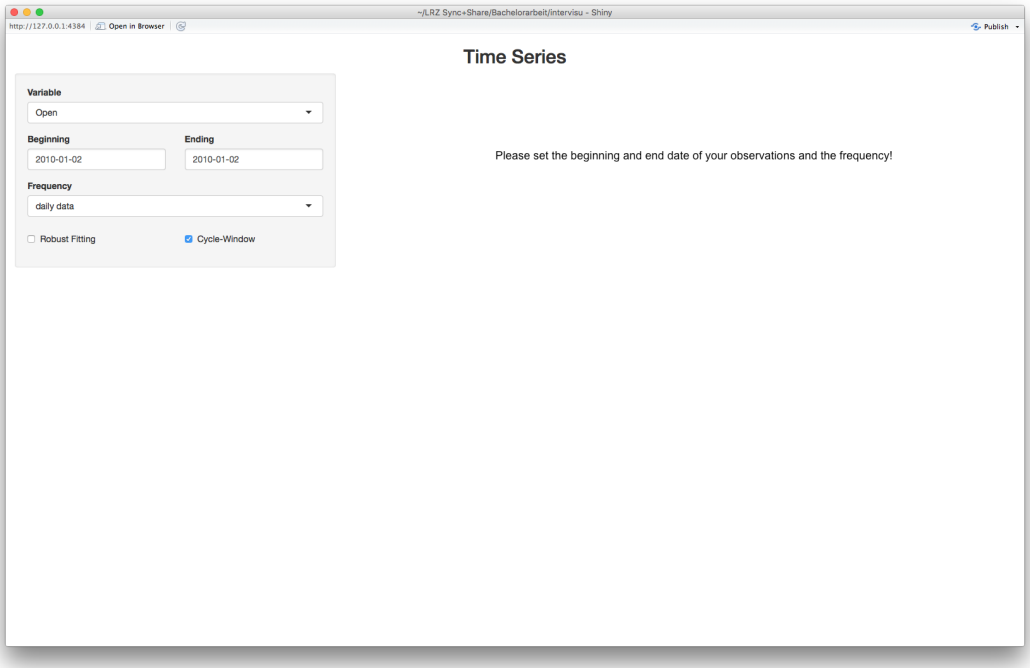

Abbildung 64: `Time_Series` 1

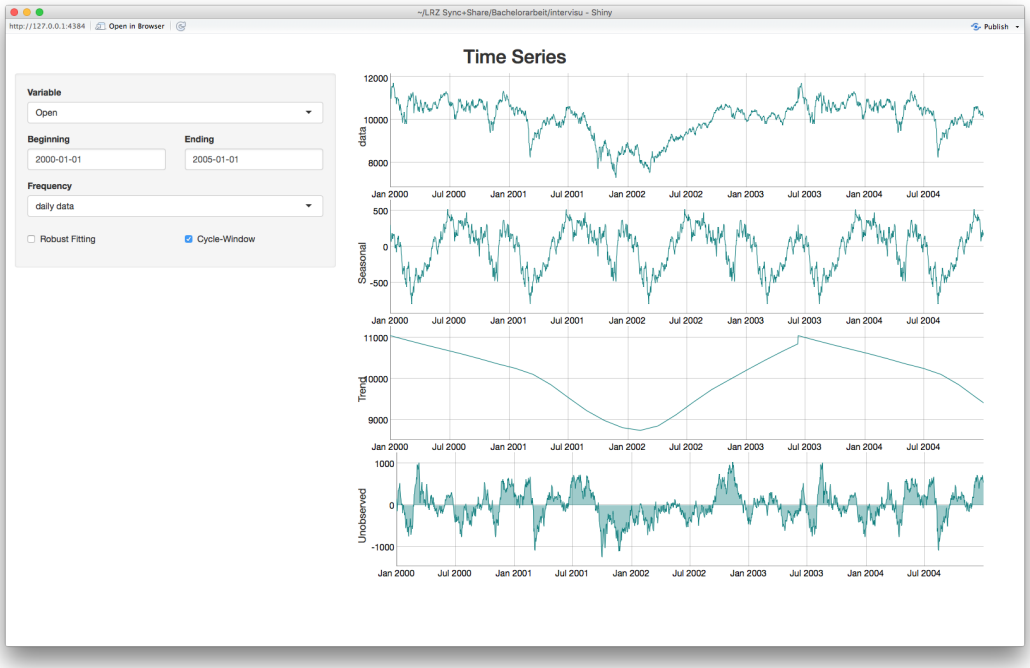

Abbildung 65: `Time_Series` 2

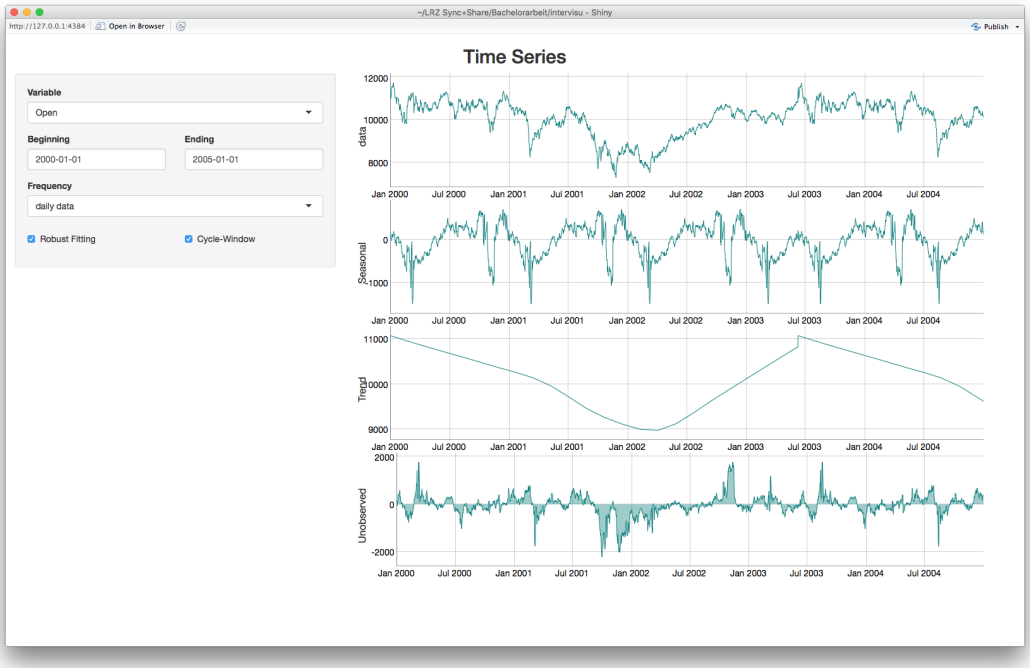

Abbildung 66: Time_Series 3

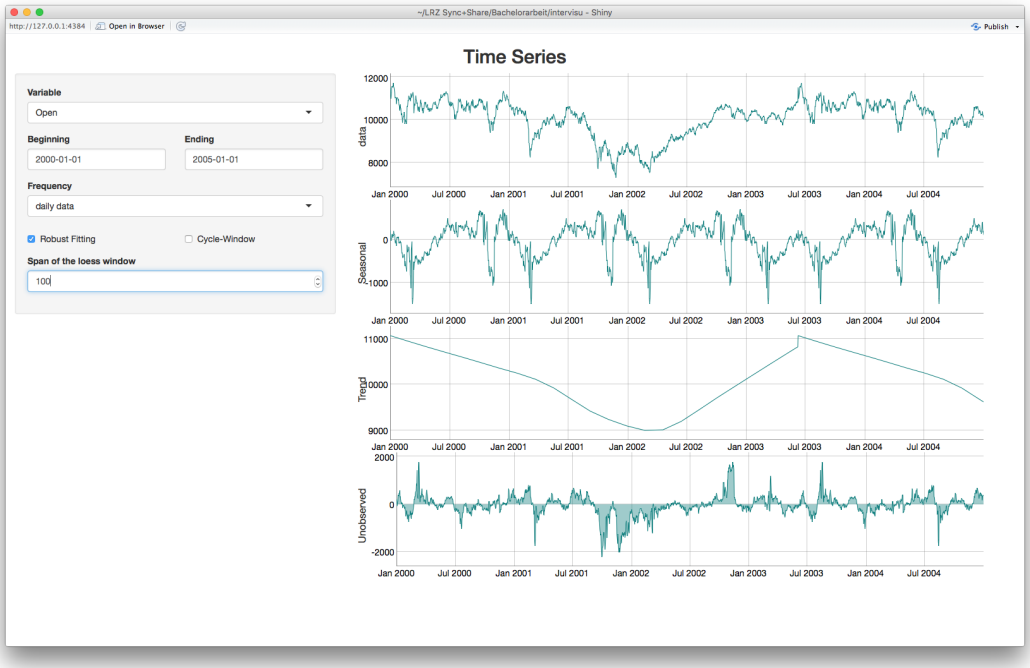

Abbildung 67: Time_Series 4

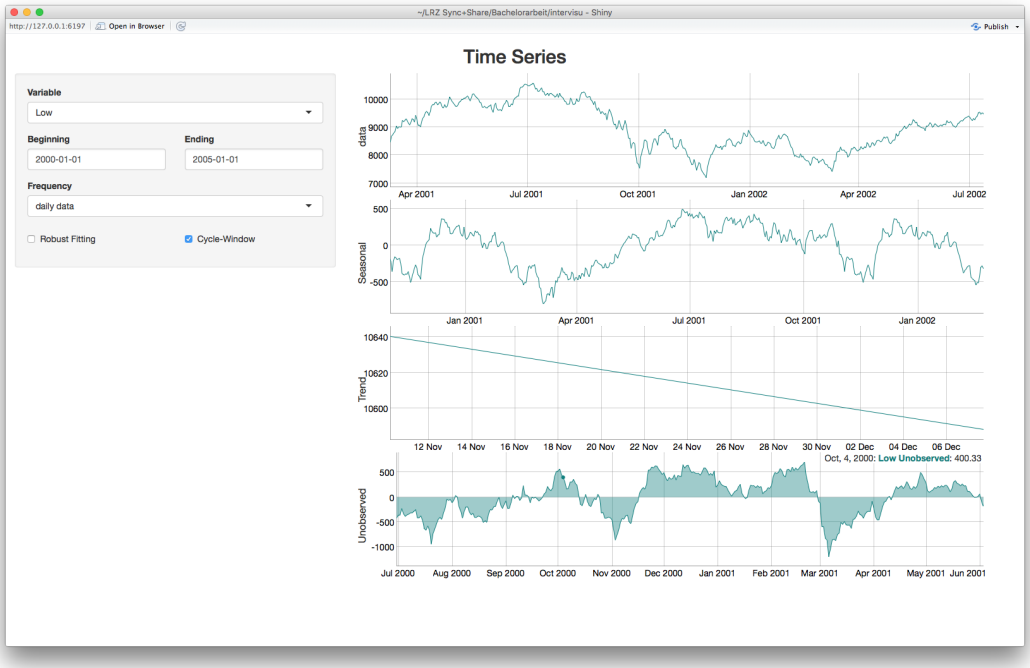

Abbildung 68: `Time_Series` 5

5 Ausblick auf mögliche Erweiterungen

Die Möglichkeiten interaktiver Datenvisualisierung anhand von *Shiny* werden mit diesen Funktionen noch nicht erschöpfend genutzt. Deshalb sollen nun mögliche Erweiterungen der Funktionen von *interVisu* besprochen werden.

Generell soll der Export von PDF- und JPEG-Dateien von explorativen Diagrammen direkt in den einzelnen Funktionen ermöglicht werden. Auch ließen sich Datensätze beispielsweise als TXT- oder CSV-Datei direkt in der Applikation hochladen, um eine noch einfachere und zeitsparendere Analyse zu ermöglichen. In diesem Zusammenhang wäre dem Parameter `data` ein *default Wert* `NULL` zugewiesen, mit dem der Nutzer die zu analysierenden Daten interaktiv zu importieren kann. Falls man jedoch dem Parameter `data` ein`data.frame` Objekt zuordnet, wird die herkömmliche Funktion ohne Datenimport aufgerufen.

Im Hinblick auf Verbesserungen von einzelnen bereits bestehenden Funktionen hat die Funktion `Scatterplot_3d` noch Potenzial bei der Repräsentation von kategorialen Merkmalen in einem dreidimensionalen Scatterplot. So könnte man der Farbe der Punkte und deren Form je ein kategoriales Merkmal zuordnen (vgl. Huber 1987, S. 448). Zum Zeitpunkt dieser Arbeit gibt es bei der für den interaktiven Scatterplot verwendeten Funktion `scatterplot3js` aus dem Paket `threejs` nur die Möglichkeit, Punkte in einen bestehenden Scatterplot einzufügen. Es wäre jedoch eine Linie zur Repräsentation einer bivariaten Regressionsgerade möglich. Um bei dieser Regression mit den Einflussvariablen X_1 und X_2 und Zielgröße X_3 die Residuen auch in schrägen

Betrachtungen erkennen zu können, empfiehlt Huber das Einzeichnen von Verbindungslinien zwischen $(x_{1,i}, x_{2,i}, 0)$ und $(x_{1,i}, x_{2,i}, x_{3,i})$ $\forall i = 1, \ldots, n$ (vgl. ebd., S. 448). Diese Erweiterung eines Scatterplots wird auch *Spike-Plot* genannt.
Ein weiteres Update wäre bei einer Analyse mittels einer Scatterplotmatrix möglich. Da das Auge mit vielen metrischen Einflussvariablen schnell die Übersicht verliert, wurden hierzu von Tukey diverse Verbesserungen, die in der Funktion `Scatterplot_Matrix` realisiert werden könnten, vorgeschlagen. Beispielsweise könnte die Aussagekraft jedes einzelnen Scatterplots anhand eines *projection pursuit clottedness index* beurteilt werden (vgl. Tukey 1988, S. 431). Ein weiterer vorgeschlagener Indikator wäre der Unterschied der einfachen und robusten Korrelationskoeffizienten. Dieser skalare Wert könnte anhand einer Heatmap hinter den einzelnen Scatterplots den Informationsgehalt der Darstellungen indizieren.
Ein weiteres Manko ist, dass die Darstellung von kategorialen Daten in den bis jetzt programmierten Funktionen unterrepräsentiert ist. Die Lücke könnte eine Applikation zur interaktiven Exploration von Baumdiagrammen füllen. Beispielsweise gäbe es die Möglichkeit, die zu verwendenden Variablen und deren hierarchische Reihenfolge zu bestimmen, sich absolute und relative Häufigkeiten der Beobachtungen anzeigen zu lassen oder die *Odds Ratios* einzelner Blöcke von je zwei Merkmalen mit zwei Stufen zu analysieren.
Auch die Clusteranalyse ließe sich interaktiv gestalten, indem man die verwendeten Einflussvariablen und das anzuwendende Verfahren bestimmt. Bei Darstellungen in Scatterplots ist die Zuordnung auf drei metrische Variablen begrenzt.
Um zum Schluss wieder auf Tukeys Theorie der flexiblen explorativen Datenanalyse zu kommen, gäbe es auch die Möglichkeit, mithilfe von Funktionsschleifen, ein frei zu kombinierendes neues Programm zu schreiben. In dieser Applikation müsste man sich nicht mehr mit dem Funktionsabruf für eine spezifische Art der Analyse entscheiden, sondern könnte alles in der Anwendung interaktiv auswählen. Das Fenster kann man hierbei mit beliebigen graphischen Darstellungen füllen und anhand von graphüberschreitendem Hervorheben von Beobachtungen analysieren.
Insgesamt jedoch werden mit den nun implementierten Funktionen in *interVisu* viele explorative Betrachtungen der Daten ermöglicht und erleichtert. Das Ziel des Paketes, speziell wenig erfahrenen Datenanalysten neue explorative Möglichkeiten zu bieten, halte ich für erfüllt. Ohne große Programmierkenntnisse ist es nun möglich, einen Datensatz zumindest bis zu einem gewissem Punkt explorativ analysieren zu können. Es soll darauf hingewiesen werden, dass keine explorative Analyse zu generalisieren ist und die hier vorgestellten Methoden nur einen Überblick über die Zusammenhänge und Verteilungen des observierten Datensatzes geben sollen, ohne ein fixes Regelwerk darstellen zu wollen. Erweiterungen um sicherlich präsente Fehler auszubessern und sinnvolle Ergänzungen einzuarbeiten, werden mit der Zeit noch folgen.

Anhang

A Weitere Shiny Anwendungen

```
 fixedPage(
# in diesem Beispiel verwenden wir
# eine statische Seite.
 sidebarLayout(
    sidebarPanel(
    ),
    mainPanel(

    )
  )
 )
```

Abbildung 69: User-Interface mit `fixedPage`

```
 fillPage(
# in diesem Beispiel verwenden wir
# eine sich immer füllende Seite.
 sidebarLayout(
    sidebarPanel(
    ),
    mainPanel(

    )
  )
 )
```

Abbildung 70: User-Interface mit `fillPage`

```
shinyServer(function(input, output) {
    output$text <- renderText({
        # hier kann mit normalem R-Code ein beliebiger Text
        # beschrieben werden Ausgabewert muss nur ein char Objekt
        # sein.
    })
})
```

Abbildung 71: Server-Funktion zur Definition eines Text-Outputs

```
fluidPage(
  fluidRow(
       colomn(6, "Hier wird ein Eintrag
        der Reihe definiert der den 6/12 einnimmt."),
       colomn(3, "Hier wird ein Eintrag
              der Reihe  definiert der den 3/12 einnimmt."),
       colomn(3, "Alle Spalten einer Reihe sollten immer
              aufaddiert 12 ergeben")
       )

)
```

Abbildung 72: User-Interface mit `fuidPage` und `fuidRow`

```
fluidPage(
sidebarLayout(
   sidebarPanel(
     # Es wird nun ein Select-Input definiert der den Benutzer
     # aus Listennamen einer Liste mit beliebigen Werten aus.
     # Der Rückgabewert des Inputs ist immer der Eintrag
     # des Listeneintrages mit dem ausgewählten Listennamen.
     # Das Label des Inputs ist select, was bedeutet,
     # dass man später Werte dieses Inputs mit dem Begriff
     # input$select referenzieren kann.
     selectInput(inputId="select", choices=list("Hallo" =1,
     "Schönen Tag"=2),selected=1)
   ),
   mainPanel(
     # Nun wird ein Text definiert mit dem Label "Text"
     # angelegt.
     # Will man anschließend genau diesen Output im Server
     # definieren, kann man dies mit output$text.
     textOutput(outputId="text")
   )
 )
)
```

Abbildung 73: User-Interface mit `fuidPage` und `sidebarLayout`, welches ein Select-Gadget benutzt, dass die Optionen "Hallo" und "Schönen Tag" anbietet und bei Auswählen von "Hallo" den numerischen Wert 1 und bei Auswählen von "Schönen Tag" annimmt.

B Funktionen-Überblick des R Paketes *interVisu*

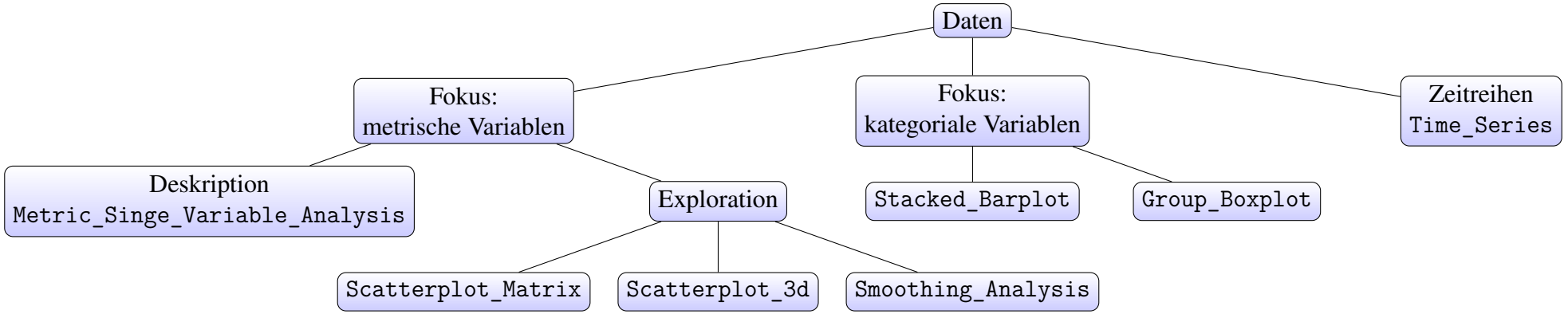

Abbildung 74: Überblick der Funktionen in *interVisu*

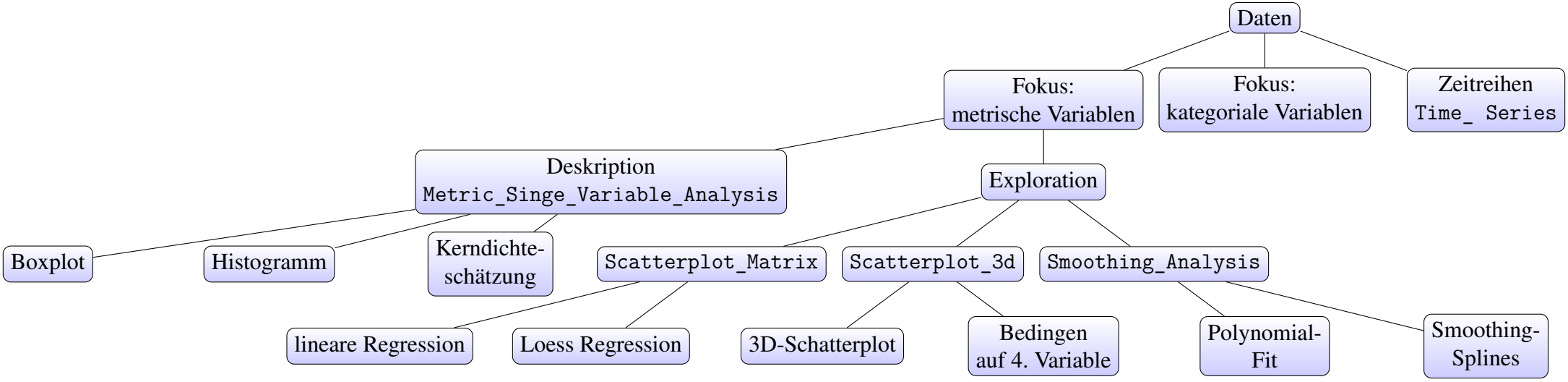

Abbildung 75: Überblick der Möglichkeiten mit den Funktionen in *interVisu* fokussiert auf metrische Variablen

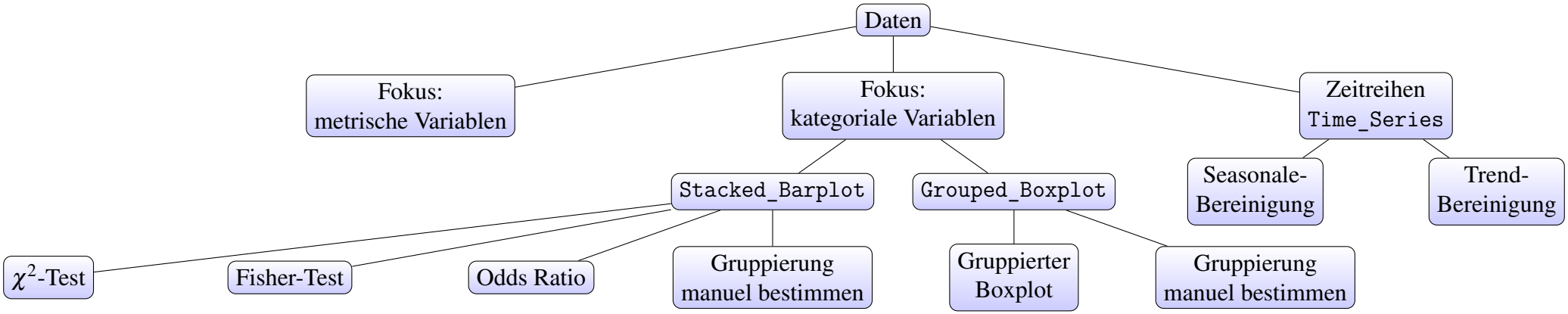

Abbildung 76: Überblick der Möglichkeiten mit den Funktionen in *interVisu* fokussiert auf kategoriale Variablen und Zeitreihen

C Herleitungen

C 1 Herleitung der Schätzer $\hat{\beta}_0$ und $\hat{\beta}_1$ in der linearen Einfachregression

Es soll β_0 und β_1 durch $\hat{\beta}_0$ und $\hat{\beta}_1$ mit den n Beobachtungen (y_i, x_i) mit $i = 1, \ldots, n$ geschätzt werden. Das lineare Modell kann als Gleichung dargestellt werden.

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \varepsilon_i$$

Die Schätzer werden nach der Methode der kleinsten Quadrate gewählt. Es sollen also die quadratischen Residuen ε_i für $i = 1, \ldots, n$ minimiert werden. Seien n Beobachtungen (y_i, x_i) mit $i = 1, \ldots, n$ gegeben, lässt sich dieser zu minimierende Term als Funktion in Abhängigkeit von β_0 und β_1 beschreiben.

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} \varepsilon_i^2 &= \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= f(\beta_0, \beta_1)
\end{aligned}$$

Nun leitet man $f(\beta_0, \beta_1)$ nach β_0 und β_1 ab und setzt die Ableitung gleich 0.

$$\frac{\partial f(\beta_0, \beta_1)}{\partial \beta_0} = -\frac{2}{n}\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)$$

$$\frac{\partial f(\beta_0, \beta_1)}{\partial \beta_1} = -\frac{2}{n}\sum_{i=1}^{n} x_i (y_i - \beta_0 - \beta_1 x_i)$$

Setzte anfänglich $\dfrac{\partial f(\beta_0, \beta_1)}{\partial \beta_0}$ gleich 0.

$$\begin{aligned}
-\frac{2}{n}\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) &\stackrel{!}{=} 0 \\
\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) &= 0 \\
\sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i &= n\beta_0 \\
\hat{\beta}_0 = \frac{\sum_{i=1}^{n} y_i - \beta_1 \sum_{i=1}^{n} x_i}{n} &= \frac{n\bar{y} - n\beta_1 \bar{x}}{n} \\
&= \bar{y} - \beta_1 \bar{x}
\end{aligned}$$

Setzte jetzt $\frac{\partial f(\beta_0,\beta_1)}{\partial \beta_1}$ gleich 0.

$$
\begin{aligned}
-\frac{2}{n}\sum_{i=1}^{n} x_i(y_i-\beta_0-\beta_1 x_i) &\stackrel{!}{=} 0\\
-\frac{2}{n}\sum_{i=1}^{n} x_i y_i - x_i\beta_0 - \beta_1 x_i^2 &= 0\\
\sum_{i=1}^{n} x_i y_i - x_i\beta_0 - \beta_1 x_i^2 &= 0\\
\sum_{i=1}^{n} x_i y_i - x_i(\bar{y}-\beta_1\bar{x}) - \beta_1 x_i^2 &= 0\\
\sum_{i=1}^{n} x_i y_i - \bar{y}\sum_{i=1}^{n} x_i &= \beta_1\sum_{i=1}^{n} x_i^2 - \bar{x}x_i\\
n\overline{yx} - n\bar{y}\bar{x} &= \beta_1(n\overline{x^2} - n\bar{x}^2)\\
\hat{\beta}_1 &= \frac{n\overline{yx}-n\bar{y}\bar{x}}{n\overline{x^2}-n\bar{x}^2} = \frac{S_{xy}}{S_x}
\end{aligned}
$$

(vgl. Fahrmeir, Künstler u. a. 2011, S. 155 f)

C 2 Herleitung des Schätzers $\hat{\beta}$ in einem Polynomial-Modell

Es soll wieder nach der Methode der Kleinsten-Quadrate $\beta = (\beta_0,\ldots,\beta_r)$ mit den Beobachtungen $(x_1,y_1),\ldots,(x_n,y_n)$ in einem polynomialen Modell des Grades r geschätzt werden.Das Modell lässt sich nach (3.17) als

$$y = \mathbf{X}\beta + \varepsilon$$

schreiben und der zu minimierende Ausdruck ist:

$$min\sum_{i=1}^{n}\varepsilon_i^2 = min\ \varepsilon^T\varepsilon$$

Dieser Ausdruck soll in Abhängigkeit des Vektors β minimiert werden.

$$
\begin{aligned}
\varepsilon^T\varepsilon &= (y-\mathbf{X}\beta)^T(y-\mathbf{X}\beta)\\
&= y^Ty - y^T\mathbf{X}\beta - \mathbf{X}^T\beta^T y + (\mathbf{X}\beta)^T\mathbf{X}\beta\\
&= y^Ty - y^T\mathbf{X}\beta - y^T\beta\mathbf{X} + \beta^T\mathbf{X}^T\mathbf{X}\beta = f(\beta)
\end{aligned}
$$

Leitet man nun f nach β ab ergibt sich:

$$
\begin{aligned}
\frac{\partial f(\beta)}{\partial\beta} &= -y^T\mathbf{X} - y^T\mathbf{X} + 2\mathbf{X}^T\mathbf{X}\beta\\
&= -2y^T\mathbf{X} + 2\mathbf{X}^T\mathbf{X}\beta
\end{aligned}
$$

Nun wird $\frac{\partial f(\beta)}{\partial \beta}$ gleich 0 gesetzt.

$$\begin{aligned} -2\mathbf{X}^T y + 2\mathbf{X}^T\mathbf{X}\beta &\stackrel{!}{=} 0 \\ 2\mathbf{X}^T y &= 2\mathbf{X}^T\mathbf{X}\beta \\ \hat{\beta} &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \end{aligned}$$

In dem letzten Schritt wird von der Matrix $\mathbf{X}^T\mathbf{X}$ die Invertierbarkeit vorausgesetzt. Daher muss $\mathbf{X}^T\mathbf{X}$ vollen Rang aufweisen, um zu einer Schätzung von β gelangen zu können.

C 3 Herleitung des Schätzers $\hat{\beta}$ bei einer univariaten Glättung nach dem Verfahren der penalisierten Kleinsten-Quadrate

Der herkömmliche Schätzer $\hat{\beta}$ nach der Methode der Kleinsten-Quadrate minimiert $\varepsilon^T\varepsilon$. Bei einer penalisierten Schätzung soll zusätzlicher ein additiver Penalisierungsterm minimiert werden. Somit gilt es den Term $\varepsilon^T\varepsilon + \lambda\beta^T\mathbf{K}\beta$ zu minimieren. Wir gehen generell von einem Modell von zwei metrischen Variablen X und Y aus, wobei Y durch X mit gewählten Kernfunktionen $B_1,\ldots,B_d$ glatt modelliert werden soll. Es seien n Beobachtungen $(x_1,y_1),\ldots,(x_n,y_n)$ in einer Designmatrix $\mathbf{X}$ gegeben.
Aus $\mathbf{X}$ berechnen wir anfänglich die Werte der einzelnen Kerne $\mathbf{B}$.

$$\mathbf{B} = \begin{pmatrix} B_1(x_1) & \ldots & B_d(x_1) \\ \vdots & & \vdots \\ B_1(x_n) & \ldots & B_d(x_n) \end{pmatrix}, \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_d \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

$\hat{y}$ lässt sich nun als $\mathbf{B}\beta$ schreiben.

$$\begin{aligned} \varepsilon^T\varepsilon &= (y-\mathbf{B}\beta)^T(y-\mathbf{B}\beta) + \lambda\beta^T\mathbf{K}\beta \\ &= y^Ty - y^T\mathbf{B}\beta - \beta^T\mathbf{B}^Ty + \beta^T\mathbf{B}^T\mathbf{B}\beta + \lambda\beta^T\mathbf{K}\beta \\ &= y^Ty - 2\beta^T\mathbf{B}^Ty + \beta^T(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})\beta \\ &= l(\beta) \end{aligned}$$

Nun haben für eine von β abhängige Funktion l, welche abgeleitet und gleich 0 gesetzt werden muss, um das Minimum zu finden.

$$\begin{aligned} \frac{\partial l(\beta)}{\partial \beta} &= -2\mathbf{B}^Ty + 2(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})\beta \\ -2\mathbf{B}^Ty + 2(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})\beta &\stackrel{!}{=} 0 \\ 2(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})\beta &= 2\mathbf{B}^Ty + 2(\mathbf{B}^T\mathbf{B} \\ (\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})\beta &= \mathbf{B}^Ty \\ \hat{\beta} &= (\mathbf{B}^T\mathbf{B} + \lambda\mathbf{K})^{-1}\mathbf{B}^Ty \end{aligned}$$

Literatur

Agresti, A. „Categorical Data Analysis“. In: John Wiley und Sons, Inc., 2002. Kap. Inference for Contingency Tables.

Allaire, J. J. *Application layout guide*. Jan. 2014. URL: http://shiny.rstudio.com/articles/layout-guide.html.

Asimov, D. „The Grand Tour: A Tool for Viewing Multidimensional Data“. In: *SIAM Journal on Scientific and Statistical Computing* 6.1 (1985).

Build a dynamic UI that reacts to user input. Jan. 2014. URL: http://shiny.rstudio.com/articles/dynamic-ui.html.

Build custom input objects. Jan. 2014. URL: http://shiny.rstudio.com/articles/building-inputs.html.

Build your entire UI with HTML. Jan. 2014. URL: http://shiny.rstudio.com/articles/html-ui.html.

Buja, A. und D. Asimov. „Grand Tour Methods: An Outline“. In: *COMPUTER SCIENCE AND STATISTICS: Proceedings of the Seventeenth Symposium on The Interface*. Hrsg. von D. M. Allen. 1986, S. 63–67.

Cleveland, W. und R. Becker. „Brushing Scatterplots“. In: *Technometrics* 29.29 (Mai 1987), S. 127–142.

Cleveland, W., R. Cleveland u. a. „STL: A Seasonal-Trend Decomposition Procedure Based on Loess“. In: *Journal of Official Statistics* 6.1 (1990), S. 3–73.

Cleveland, W. und S. Devlin. „Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting“. In: *Journal of the American Statistical Association* 83.403 (Sep. 1988), S. 596–610.

Cook, D. u. a. „Grand Tour and Projection Pursuit“. In: *Journal of Computational and Graphical Statistics* 4.3 (1995).

Diekmann, A. „Empirische Sozialforschung“. In: Rowohlt Taschenbuch Verlag, 2012. Kap. Querschnitt- und Längsschnitterhebungen.

Fahrmeir, L., T. Kneib und S. Lang. *Regression- Modelle, Methoden und Anwendungen*. Springer, 2007.

Fahrmeir, L., R. Künstler u. a. *Statistik - Der Weg zur Datenanalyse*. Springer, 2011.

Forster, O. „Analysis 1: Differential- und Integralrechnung einer Veränderlichen“. In: Springer, 2008. Kap. Punktmengen, S. 82–93.

Grolemund, G. *Reactivity: An overview*. Mai 2015. URL: http://shiny.rstudio.com/articles/understanding-reactivity.html.

Huber, P. „Experiences with Three-Dimensional Scatterplots“. In: *Journal of the American Statistical Association* 82.398 (Juni 1987), S. 448–453.

Kaum jemand blickt durch im Datenwust. Dez. 2011. URL: http://www.wiwo.de/technologie/digitale-welt/2-800-000-000-000-000-000-000-daten-kaum-jemand-blickt-durch-im-datenwust/7504734.html.

Leonhardt, D. *John Tukey, 85, Statistician; Coined the Word 'Software'*. Juli 2000. URL: http://www.nytimes.com/2000/07/28/us/john-tukey-85-statistician-coined-the-word-software.html.

Reactivity: An overview. Jan. 2014. URL: http://shiny.rstudio.com/articles/reactivity-overview.html.

Theus, M. „Interactive Data Visualisation using Mondrian“. In: *Journal of Statistical Software* 7.11 (2002).

Theus, M. und S. Urbanek. *Interactive Graphics for Data Analysis*. CRC Press, 2009.

Tukey, J. „Analyzing data: Sanctification or detective work?“ In: *American Psychologist* 24.2 (1969).

— *Collected works Vol 5 : Graphics, 1965 - 1985*. Wadsworth Advanced Books und Software, 1988.

— „We Need Booth Exploratory and Confirmatory“. In: *The American Statistician* 34.1 (1980).

Urbanek, S. „iPlots eXtreme - Next-generation Interactive Graphics Design and Implementation of Modern Interactive Graphics“. unpuplished manuscript: http://urbanek.info/pub/dsc09.pdf.

Urbanek, S. und M. Theus. „iPlots: high interaction graphics for R“. In: *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*. 2003.

Woods, S. N. *Generalized Adiitive Models: An Introduction with R*. Chapman & Hall/CRC, 2006.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Hausarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.
Die Stellen der Hausarbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

München, den 1. August 2016

Cornelius Laurin Fritz