Tutorium für die Benutzung von R im Rahmen der Vorlesung Modellgestützte Analyse und Optimierung

Karin Reszka

1 Einführung

R ist eine Programmiersprache für Berechnungen der Statistik und gehört zu den am weitesten Skriptsprachen zur Datenanalyse. Sie beinhaltet eine Großzahl von statistischen sowie graphischen Techniken, sodass mittels R statistische Kennzahlen leicht berechnet werden können sowie Grafiken ohne großen Aufwand erstellt werden können. Daher bietet sich die Benutzung der Programmiersprache auch im Rahmen dieser Vorlesung an. Innerhalb dieses Tutorials werden die Grundlagen von R erklärt. Begonnen wird im Abschnitt 2 mit der Installation der Programmiersprache, nachfolgend wird in Abschnitt 3 erklärt, wie die Benutzeroberfläche von R aufgebaut ist. Anschließend werden drei Aufgaben im Abschnitt 4 gelöst. Das Tutorium schließt mit einer Übersicht über einige relevanten Kommandos in R ab.

2 Installation

Die Sprache R kann kostenlos unter: https://cloud.r-project.org/ heruntergeladen werden. Wir werden im Folgenden RStudio als Entwicklungsoberfläche nutzen, da das Programm eine übersichtliche Benutzteroberfläche bietet. RStudio kann unter: https://rstudio.com/products/rstudio/download/#download heruntergeladen werden. Wir werden die kostenlose Desktop Version benutzen. Wählen Sie Ihr entsprechendes Betriebssytem aus, laden Sie es herunter und installieren Sie das Programm.

3 Erste Schritte mit R

Die Standartansicht von RStudios umfasst drei Fenster: Die Konsole, das Datei-Fenster, sowie das Umgebungsfenster (s. Abb. 1).

In der Konsole werden Sie später Ausgaben Ihres R Skripts finden. Außerdem können Sie die Konsole nutzen um Befehle in R zu schreiben. In dem Umgebungs-Fenster werden die Variablen angezeigt, die Sie angelegt haben und in dem Datei-Fenster können Sie auswählen, welche Dateien Sie in Ihren Code integrieren wollen.

Wählen Sie zunächst unter File„new File“ aus und klicken Sie dann auf „R Script“. So öffnet sich ein neues Fenster, in dem Sie Ihren Code schreiben können (s. Abb. 2).

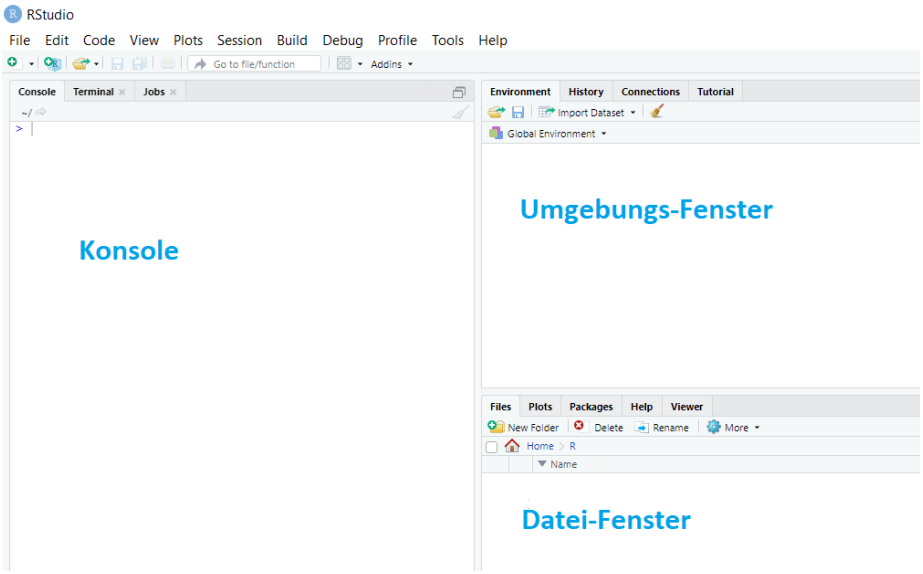

Abbildung 1: Übersicht der Fenster in R

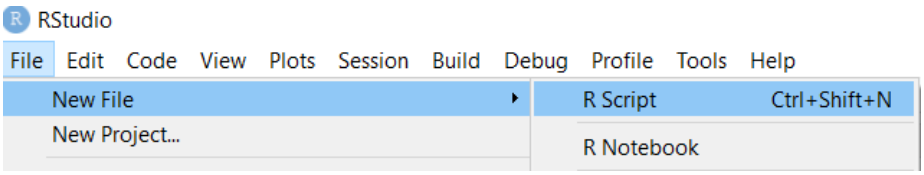

Abbildung 2: R Script erstellen

Speichern Sie Ihre Datei unter „File“, „Save as“ als „Tutorial.R“ ab. In einem R Skript wird durch Anklicken von „Run“ immer nur die gerade ausgewählte Zeile ausgeführt. Wie beispielsweise in Abb. 3 gezeigt wird, wird durch die Auswahl der zweiten Zeile eine Fehlermeldung geworfen, dass die Variable y nicht bekannt ist.

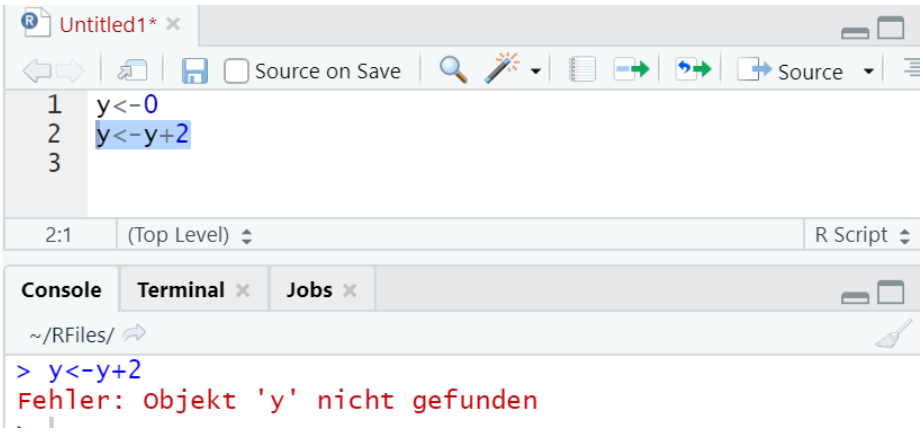

Abbildung 3: Fehlermeldung

Wählt man nun die erste Zeile aus, führt diese durch das Anklicken des Buttons „Run“ aus und wird anschließend die zweite Zeile ausgeführt, verschwindet die Fehleranzeige. Ebenso wird in dem Umgebungsfenster die Variable y angelegt, welcher nun der Wert 2 zugewiesen wurde. Führt man die zweite Zeile erneut aus, wird y wieder inkrementiert. Alternativ kann man durch gedrückt halten der Tastenkombination „Strg“+ „Shift“+ „Enter“ auch das gesamte Skript ausführen.

4 Anwendung von R

4.1 Erste Aufgabe: Erstellung eines Histogramms

Im Folgenden werden wir eine csv-Datei in unser Programm einbinden und ein Histogramm aus den gegebenen Daten erstellen. Laden Sie sich dazu die ZZ.csv Datei herunter, welche Sie auf der Homepage finden können. Dann können Sie die Datei in ihren Home-Ordner verschieben. Es empfiehlt sich einen Ordner für R anzulegen in dem Sie Ihre Dateien speichern können. (s. Abb.4)

Um Zugriff auf die Dateien in R zu bekommen, wählen Sie in dem Files-Fenster den entsprechenden Ordner aus, den Sie benutzen möchten und wählen Sie dann unter „more“ „Set as Working Direction “ aus. Als nächstes schreiben Sie data<-read.csv(“ZZ.csv“), damit speichern Sie die Daten in der Variable data ab, die nun im Umgebungsfenster angezeigt werden sollte. Durch das Anklicken der Variable data wird ein neues Fenster geöffnet, in dem Sie die Datei betrachten können (s.Abb 5).

Um ein Histogramm zu erzeugen, müssen Sie zunächst herausfinden, wie die entsprechende Spalte in der Tabelle benannt wurde. Dazu schauen Sie in der csv-Datei nach, wie diese benannt wurde. Das Histogramm kann anschließend

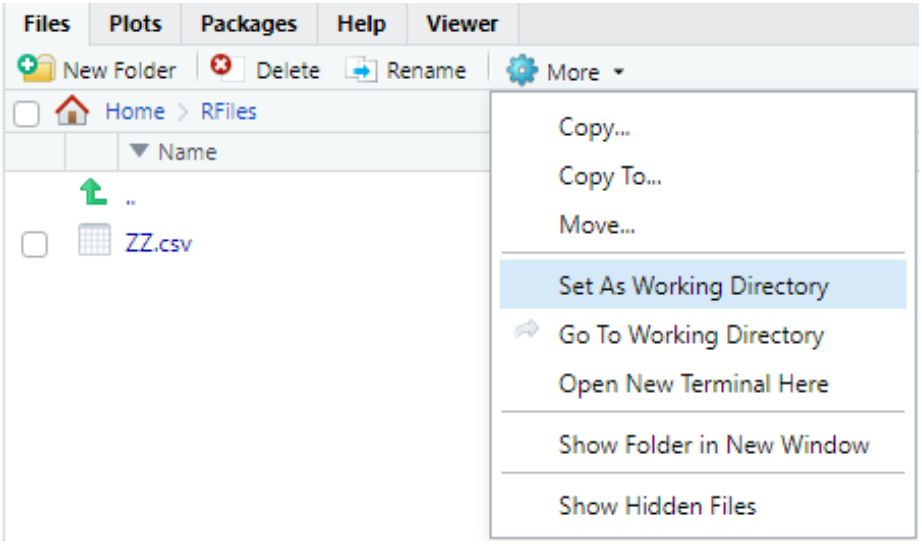

Abbildung 4: Zugriff auf Dateien bekommen

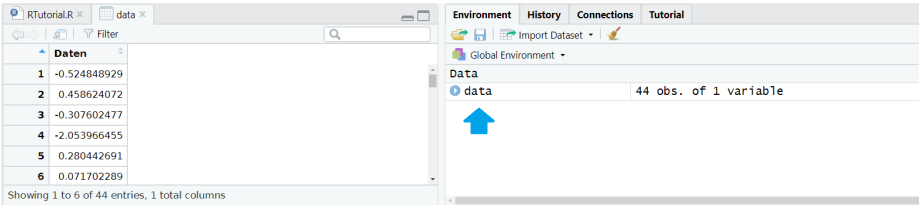

Abbildung 5: Datensatz öffnen

mit dem Befehl hist(data$"Daten",seq(min(data), max(data), length.out = 10)) erstellt werden. Wobei mit seq(min(data), max(data), length.out = n+1) der Datenbereich in n Bereiche eingeteilt werden kann (s. Abb. 6).

4.2 Zweite Aufgabe: Berechnung der Abstandsmaße D und D'

Als nächstes werden wir die Abstandsmaße D und D' berechnen. Basierend auf dem Histogramm nehmen wir als theoretische Verteilung eine Standardnormalverteilung an, diese besitzt die Dichtefunktion: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$. In diesem Tutorial werden die Schätzer nicht berechnet. Der Vollständigkeit halber sei aber erwähnt, dass es genauer wäre die Parameter der Verteilung mit der Momenten- oder Maximum-Likelihood-Methode zu schätzen. Zur Berechnung der Anpassungsgüte D und D' benötigen wir nun also:

- n_i Anzahl Werte Stichprobe im i-ten Intervall und $h_i = \frac{n_i}{n}$
- $p_i = \int_{\triangle} f(x)dx$ wobei $\triangle_i$ das i-te Intervall ist
- Abstandsmaß $D = \sum_i |h_i - p_i|$ und $D' = \sum_i (h_i - p_i)^2$

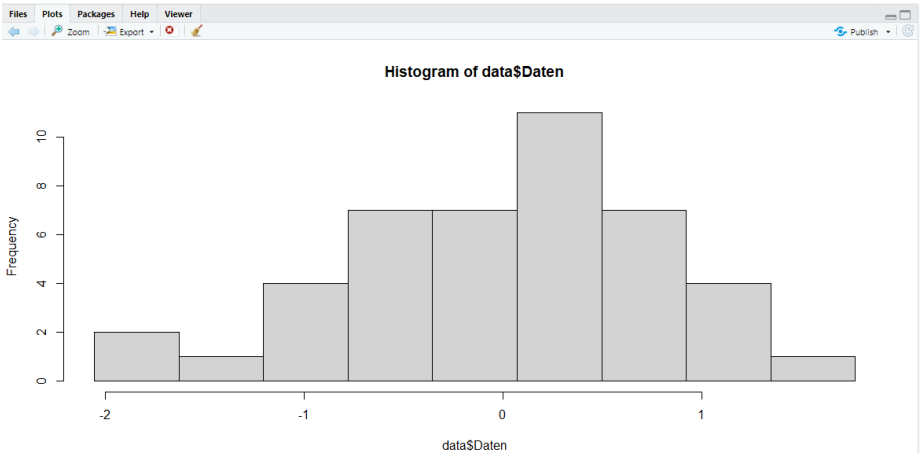

Abbildung 6: Histogramm

Wir bestimmen zunächst die Anzahl der Werte im jeweiligen Intervall, die man aus dem Histogramm ablesen kann. Diese speichern wir in einem Vektor namens vec mit dem Befehl vec<-c(2,0,1,7,10,12,8,3,1) ab. Der Befehl c() dient dazu, die gegebenen Elemente in einem Vektor zusammen zu fügen. Die Anzahl der Stichprobe beträgt 44, diese kann mittels length(data$“Daten“) bestimmt werden. Um h zu bestimmen, teilen wir jeden Eintrag, der in vec liegt durch die Anzahl der Werte. Da in R elementweise gerechnet wird, wird durch h<-vec/44 jeder Wert der in dem Vektor liegt durch 44 geteilt und abgespeichert.
Für den nächsten Schritt der Berechnung legen wir eine Funktion an. Mittels dieser Funktion werden wir die Wahrscheinlichkeit für die Werte innerhalb der Teilintervalle berechnen. Dafür benutzen wir eine For-Schleife, die in R wie folgt angelegt wird:

```
for(i in 1:9)
{
        Code
}
```

Für die Berechnung müssen vor dem Eintreten in die Schleife eine Variable (die hier j genannt wurde) und ein Vektor p initialisiert werden. Die Variable j beschreibt hierbei die Intervallgrenzen der Säulen. Wir schreiben dazu vor Eintritt in die For-Schleife: j<-(-2.5) und in die nächste Zeile p<-vector(“numeric“,length=9) . Mittels des Befehls vector() wird ein Vektor angelegt. Da wir numerische Daten benutzen, haben wir den Parameter “numeric“ und da wir 9 Säulen in dem Histogramm vorliegen haben, besitzt der Vektor eine Länge von 9. Für die Berechnung des Abstandsmaß ist es notwendig, die Differenzen der Verteilungsfunktionen zu ermitteln, deshalb schreiben Sie in die Schleife p[i] <- pnorm(j+0.5) - pnorm(j). Mit pnorm(j) wird die Verteilungsfunktion der Normalverteilung an der Stelle j berechnet. Dabei beschreibt die Variable j die Intervallgrenzen. Für die erste Säule gilt also für j=(-2.5) (wegen der Initialisierung). Ebenso muss die Variable j bei jedem Durchlauf in der

For-Schleife um 0.5 erhöht werden. Dazu schreibt man innerhalb der Schleife: j<-j+0.5.
Nachdem wir p und h ermittelt haben, können wir das Abstandsmaß D berechnen. Mittels abs(h-p) wird die einzelnen Beträge von h und p berechnet und mit sum(abs(h-p)) werden die berechneten Werte aufaddiert. Wir können unser Ergebnis mit D <- sum(abs(h-p)) abspeichern.

```
vec <- c(2,0,1,7,10,12,8,3,1)
h <- vec/44
p<-vector("numeric",length=9)
j<-(-2.5)
for(i in 1:9)
{
        p[i] <- pnorm(j+0.5) - pnorm(j)
        j<-j+0.5
}
 D <- sum(abs(h-p))
```

Um den gesamten Code auszuführen, drücken Sie "Strg"+ "Shift"+ "Enter". Die Berechnung für D' erfolgt analog.

4.3 Dritte Aufgabe: Erstellung eines P-P-Plots

In dieser Aufgabe werden wir einen P-P-Plot zeichnen. Wir verwenden, wie in den bereits vorgestellten Aufgaben, die Daten "ZZ.csv".
Für einen P-P-Plot müssen wir zunächst die Werte der empirischen Verteilungsfunktion sowie der theoretischen Verteilungsfunktion berechnen. Dazu legen wir zwei Vektoren namens Fe und Ft an. Diese werden wieder mittels des Befehls vector() angelegt. Da wir 44 Daten vorliegen haben, besitzen beide Vektoren die Länge 44.
Zunächst beschäftigen wir uns mit den Werten der theoretischen Verteilungsfunktion. Für diese müssen die Daten in sortierter Form vorliegen. Dazu schreiben Sie sortData<-sort(data$"Daten"), um die Zahlen aufsteigend zu sortieren. Im nächsten Schritt wenden wir die Verteilungsfunktion der Normalverteilung auf die einzelnen Daten an. Wie bereits erläutert kann dazu die Methode pnorm() verwendet werden. Diese wird innerhalb einer For-Schleife benutzt, sodass die Methode mit allen vorliegenden Daten gespeist wird. Ihr R Skript sollte nun wie folgt aussehen:

```
sortData<-sort(data$"Daten")
Ft<-vector("numeric",44L)
for(i in 1:44)
{
        Ft[i]<-pnorm(sortData[i])
}
```

Nun widmen wir uns den Werten der empirischen Verteilungsfunktion. Dazu benutzen wir die Indizes der Daten und teilen sie durch die Anzahl der Stichproben. Dies geschieht mittels der bereits erzeugten For-Schleife, in welcher i automatisch inkrementiert wird und in welcher wir den Index durch 44 teilen können. Ihr Code-Abschnitt zur Berechnung der Werte der empirischen Verteilungsfunktion sollte wie folgt aussehen:

```
sortData<-sort(data$"Daten")
Ft<-vector("numeric",44L)
Fe<-vector("numeric",44L)
for(i in 1:44)
{
        Ft[i]<-pnorm(sortData[i])
        Fe[i]<-i /44
}
```

Der P-P-Plot kann mittels der Graph-Funktion plot() erstellt werden. Dazu schreiben Sie plot(Fe,Ft) in Ihr Fenster und führen anschließend Ihr Skript aus.

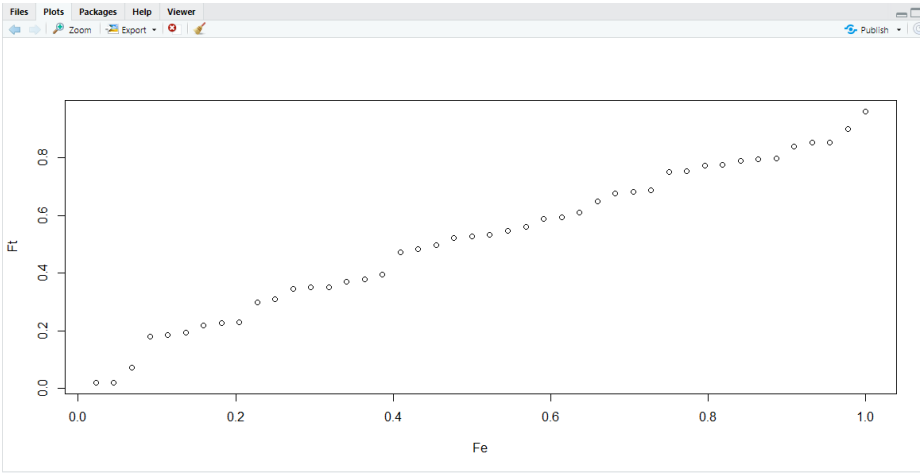

Abbildung 7: P-P-Plot

5 Überblick der Kommandos von R

5.1 Basisbefehle in R

Kommando	Bedeutung
#	Alles was in der Zeile auf das # folgt, wird auskommentiert
a<-12	<- weißt a den Wert 12 zu
a+5	Addition
a-5	Subtraktion
a*5	Multiplikation
a / 5	Division
sqrt(a)	Berechnet die Wurzel von a
a^2	Berechnet das Quadrat von a
c(1,2,3)	Fügt die Elemente zu einem Vektor zusammen
sum(1,2,3)	Berechnet die Summe der Zahlen 1,2 und 3
abs(-2)	Berechnet den Betrag der Zahl -2

5.2 Daten in R

Kommando	Bedeutung
install.packages(“readxl“)	Lädt und installiert das Paket “readxl“, welches für Excel-Daten benötigt wird
library(readxl)	Lädt das Paket in die Session
data<-read_excel(“Daten.xlsx“)	Speichert die Excel-Datei in data ab
data <- read.csv(“Daten.csv“)	Speichert eine csv-Datei in data ab
colnames(data)	Gibt den Spaltennamen aus
head(Daten$Spaltenname, n)	Zeigt die ersten n Daten an

5.3 Statistische Methoden in R

Kommando	Bedeutung
sd(Daten&Spaltenname)	Berechnet die Standartabweichung
var(Daten&Spaltenname)	Berechnet die Varianz
summary(Daten)	Gibt das Minimun, das Maximum den Median, den Mittelwert, 25 % / 75 %-Quantile aus
hist(Daten$Spaltenname,breaks=n,col= “blue“)	Erzeugt ein Histogramm mit n Säulen in der Farbe blau
pnorm(x)	Berechnet die Verteilungsfunktion der Normalverteilung an der Stelle x
pexp(x,rate=y)	Gibt den Wert der Exponentialverteilung mit der Rate y an der Stelle x zurück
plot(x,y)	Zeichnet einen Graphen
qqnorm(Daten$Spaltenname)	Erzeugt einen Q-Q-Plot