Kurze Einführung in R und RStudio I
Deskriptive und explorative Statistik

Christian Reinboth, M.Sc., Dipl.-Wi.-Inf. (FH)
bbgl. Bachelor-Studiengang BWL | Hochschule Harz

1. Einführung

1.1 Was ist R?

Bei R handelt es sich um eine frei nutzbare Programmiersprache, die sich insbesondere für die Durchführung von statistischen Analysen sowie für die Erstellung statistischer Grafiken eignet. Sie wurde Anfang der 1990er Jahre von Ross Ihaka und Robert Gentleman an der Universität von Auckland in Neuseeland entwickelt und wird seitdem von einer globalen Community von Freiwilligen fortlaufend erweitert. An der Weiterentwicklung von R kann sich grundsätzlich jeder beteiligen – insbesondere durch das Schreiben und Veröffentlichen sogenannter „Packages“, mit denen R um neue Funktionen erweitert wird (dazu später mehr).

R kann unter der URL

https://www.r-project.org/

kostenfrei heruntergeladen werden und ist für Windows, MacOS und Linux-Systeme verfügbar.

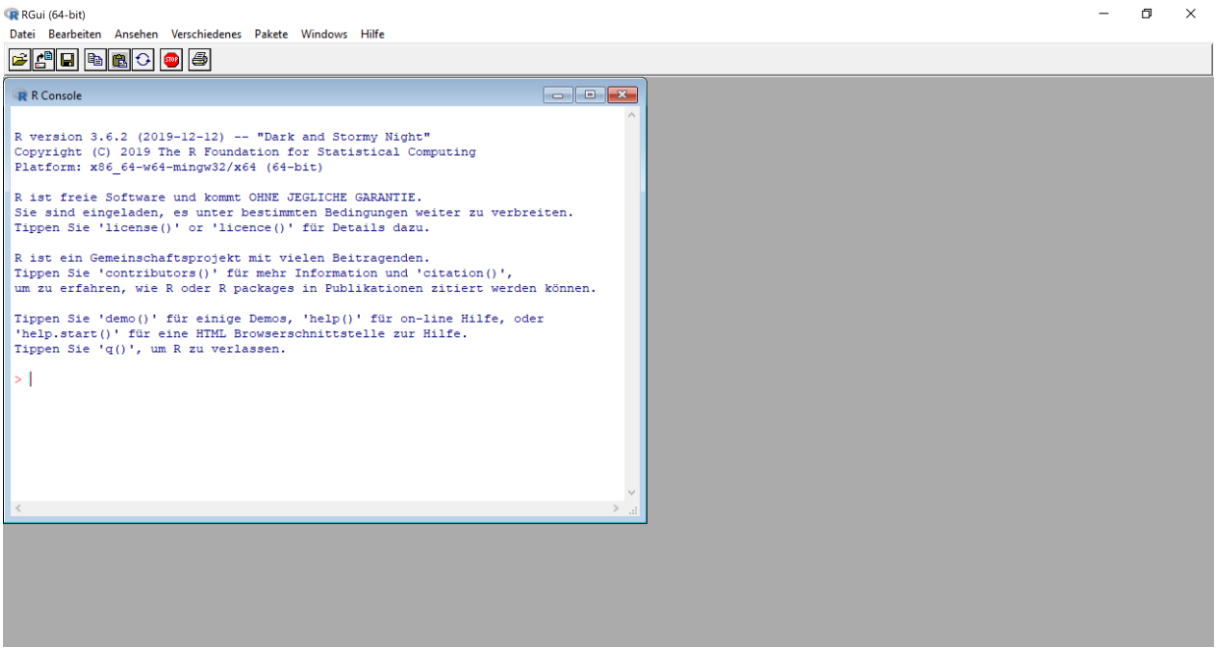

Abbildung 1: Benutzeroberfläche von R

Startet man R, findet man sich auf einer Nutzeroberfläche wieder, die im Wesentlichen aus einem weißen Fenster besteht, in welches man Befehle eintippen kann – die sogenannte Shell. Sämtliche in diesem Skript aufgeführten Übungen lassen sich in dieser Umgebung vollständig durchführen – da wir mit einer anderen Umgebung mit zusätzlichen Features und Anzeigen aber komfortabler arbeiten und lernen können, zeigen alle weiteren Screenshots die Oberfläche von RStudio – einer integrierten Entwicklungsumgebung für R, die umseitig kurz vorgestellt wird.

1.2 Was ist RStudio?

Bei RStudio handelt es sich um eine integrierte Entwicklungsumgebung für R, die von der US-amerikanischen Softwarefirma RStudio Inc. vertrieben wird. Im Vergleich mit der Oberfläche von R bietet RStudio einige zusätzliche Funktionen, die uns das Erlernen von R wesentlich erleichtern werden. Hierzu gehören unter anderem die Anzeige aller derzeit im Hintergrund geladenen Variablen und Datensätze, ein direkt im Programm (statt separat im Browser) abrufbares Benutzerhandbuch, die Auto-Vervollständigung von Befehlen sowie Optionen für den vereinfachten Import von Daten aus anderen Programmen wie z.B. Microsoft Excel.

RStudio kann unter der URL

https://rstudio.com/

kostenfrei heruntergeladen werden.

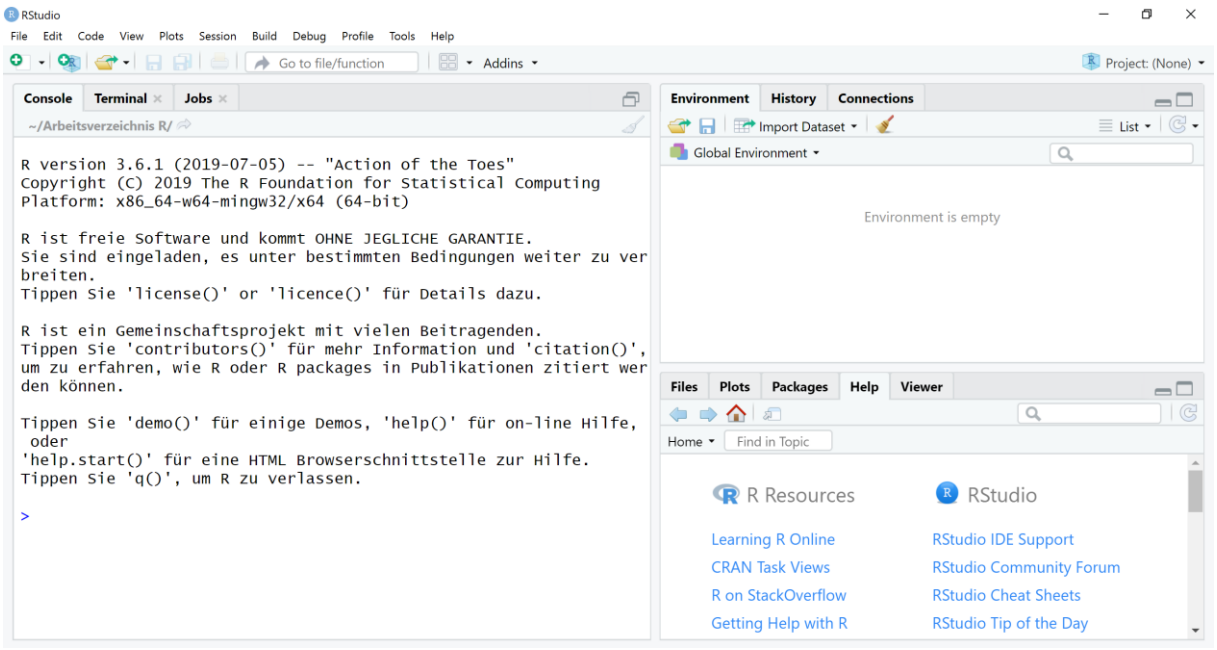

Abbildung 2: Benutzeroberfläche von RStudio

1.3 Was kann man mit R machen?

Im Rahmen dieses Kurses werden wir alle statistischen Inhalte (nicht jedoch die Inhalte aus der Stochastik) mit R umsetzen, die Bestandteil der Statistik I-Vorlesung sind. Dazu gehören u.a.:

- Berechnung von Mittelwerten (z.B. arithmetisches Mittel, Median, Modus)
- Berechnung von Streuungsmaßen (z.B. Spannweite, IQR, Standardabweichung)
- Generierung von Grafiken (insbesondere Box-Plots und Stem-and-Leaf-Diagramme)
- Berechnung von Korrelationskoeffizienten (Bravais-Pearson, Kendall und Spearman)

Wichtiger Hinweis: Die in diesem Skript vorgestellten Inhalte stellen keine vollständige Einführung in R dar, sondern präsentieren lediglich einen höchst unvollständigen Ausschnitt der statistischen Nutzungsmöglichkeiten, mit dem nur die im Rahmen der Statistik I-Vorlesung besprochenen Inhalte abgedeckt werden. Eine umfassendere Einführung befindet sich derzeit in Arbeit.

2. Aufbau von RStudio

2.1 Benutzeroberfläche

Der nachfolgende Screenshot zeigt die Standardkonfiguration von RStudio, wobei sich Position und Größe der Fenster individuell konfigurieren lassen. Der blaue Bereich ist die Shell, die wir bereits von unserem Blick auf die R-Oberfläche kennen. Im roten Bereich finden wir das „Global Environment“, in dem alle jeweils aktuell initialisierten Variablen und Datensätze aufgelistet werden. Der grüne Bereich enthält eine Reihe von Inhalten, die über die Reiter durchgeschaltet werden können, darunter etwa einen Viewer für die mit R generierten Grafiken, eine interaktive Hilfe, eine Übersicht der installierten Packages und einen Explorer für das Öffnen von Datensätzen.

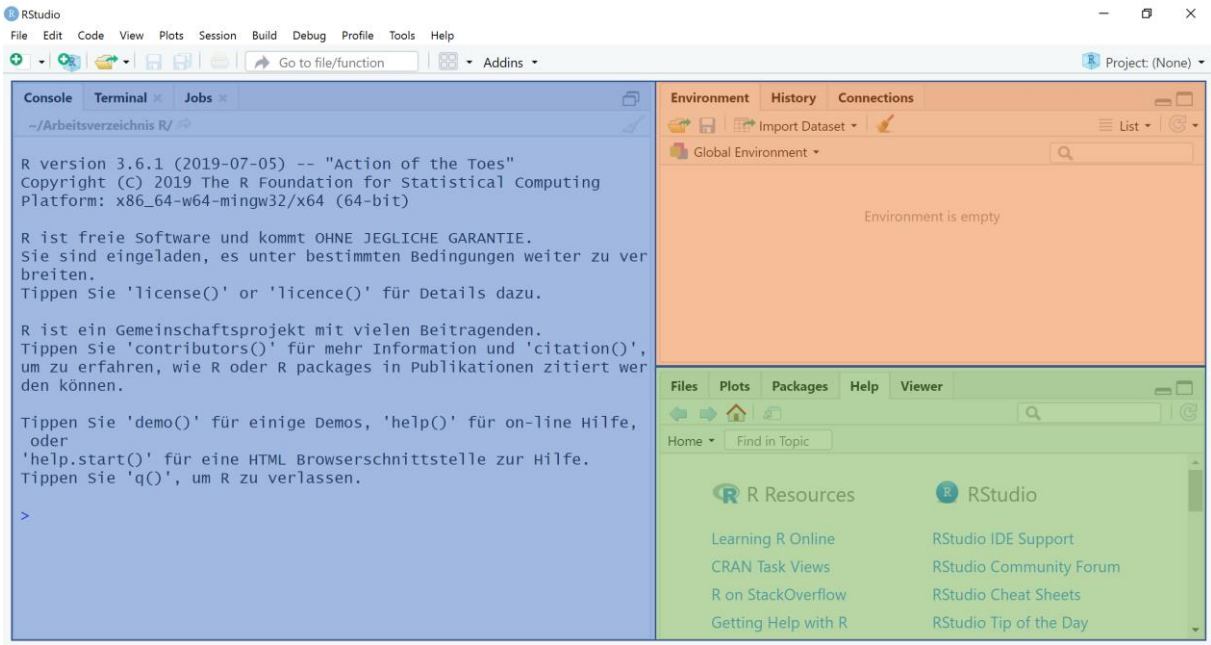

Abbildung 3: Benutzeroberfläche von RStudio

2.2 Nutzung der Hilfe

Die Hilfe steht leider nur in englischer Sprache zur Verfügung, ist aber dennoch äußerst nützlich und sollte daher unbedingt in Anspruch genommen werden. Kennt man eine gesuchte Funktion nicht, so reicht es häufig aus, das gewünschte Ergebnis (in englischer Sprache) in das Suchfeld einzugeben, um die Funktion zu finden. Möchte man beispielsweise ein Stamm-Blatt-Diagramm generieren und kennt den entsprechenden Befehl nicht, führt einen die Eingabe von „Stem-and-Leaf“ zur Hilfeseite für die Funktion „stem“. Kennt man dagegen die Funktion, kann man die Hilfeseite aufrufen, indem man in die Shell ein Fragezeichen und die Funktionsbezeichnung eingibt.

2.3 Installation von Packages

Die Installation von Packages erweitert R um neue Funktionen und Beispieldatensätze. Für die in diesem Skript vorgestellten Aufgaben werden die Packages „main“, „stats“ und „graphics“ benötigt, die alle drei bereits standardmäßig vorinstalliert sind, so dass keine Packages nachinstalliert werden müssen. Für weiterführende Anwendungen sind allerdings ggf. Packages erforderlich, die mit einem Klick auf den jeweiligen Haken installiert werden. In der Hilfe wird zu jeder Funktion oben links auf jeder Hilfeseite stets das Package (in geschweiften Klammern) angegeben, aus dem dieses stammt – so gehört etwa die Funktion „stem“ zum Package „graphics“. Sollte man je eine Funktion benötigen, die man in der Hilfe findet, die man jedoch in R nicht aufrufen kann, so sollte man prüfen, ob ggf. noch das entsprechende Package nachinstalliert werden muss.

3. Erste Schritte in R

3.1 R als Taschenrechner

Um sich mit R vertraut zu machen, kann man einmal ein paar mathematische Operationen in die Shell eingeben und mit Return bestätigen. Testen kann man beispielsweise:

```
1+3
3-5
6*6
3/2
2^2
```

Hier gibt R jeweils das rechnerisch korrekte Ergebnis aus, wobei stets eine [1] vorangestellt wird. Diese [1] ist lediglich für Ausgaben von Relevanz, die so lang sind, dass sie sich über mehrere Zeilen erstrecken. Fängt beispielsweise die zweite Zeile der Ausgabe mit einer [23] an, so bedeutet dies, dass es sich beim ersten Wert dieser Zeile um den 23. Wert der ausgegebenen Datenreihe handelt.

Im Rahmen dieser ersten Kennenlernübung wollen wir uns gleich noch mit Funktionen und deren Argumenten vertraut machen. Das Aufrufen einer Funktion löst eine bestimmte Operation in R aus (wie beispielsweise die Berechnung eines Mittelwerts oder die Generierung eines Histogramms). In den meisten Fällen müssen Funktionen beim Aufruf Argumente übergeben werden, damit sie korrekt ausgeführt werden können (wie beispielsweise die Datenreihe, für welche der Mittelwert berechnet werden soll, oder die Information, dass die Balken des Histogramms hellblau eingefärbt sein sollen).

Sehen wir uns das einmal am Beispiel der Funktion zur Berechnung der Quadratwurzel an. Diese lautet sqrt(), wobei das „sqrt“ als Abkürzung für „Square Root“ (also Quadratwurzel) steht. In den runden Klammern werden die Argumente übergeben, wobei wir für diese Funktion lediglich ein Argument benötigen – die Zahl, aus der R die Quadratwurzel ziehen soll:

```
sqrt(4)
```

Verdeutlichen wir uns diesen prinzipiellen Aufbau einer Funktion noch einmal an einem zweiten Beispiel – der „round“-Funktion, die ihr als Argument übergebene Zahlen auf- bzw. abrundet:

```
round(1.2)
```

Es fällt auf, dass der Wert 1,2 mit einem Punkt statt mit einem Komma geschrieben wird. Das liegt daran, dass das Komma bei der Übergabe mehrerer Argumente als Trennzeichen dient. Wir können uns also merken: Werte mit Nachkommastellen werden in R stets mit einem Punkt erfasst. Und: Bei Funktionen, die mehrere Argumente verarbeiten können, dient das Komma als Trennzeichen.

3.2 Anlegen von Datensätzen

Wie aber bekommen wir nun Daten in R? In vielen Fällen werden wir diese sicherlich aus Excel- oder SPSS-Datensätzen importieren, dennoch soll an dieser Stelle kurz betrachtet werden, wie man Daten direkt in R eingibt. Die Initialisierung einer Variablen ist äußerst simpel:

```
Testvariable = 23
```

Es ist zu beobachten, dass im Fenster „Global Environment“ in RStudio eine neue Variable mit der Bezeichnung „Testvariable“ erscheint, welcher der Wert 23 zugeordnet wird. Diese können wir auch direkt über die Shell abrufen, indem wir die Variablenbezeichnung eingeben:

```
Testvariable
```

Als Ausgabe erhalten wir die 23 sowie die bereits bekannte [1]. Wollen wir einer Variable mehr als nur einen Wert zuordnen, können wir dies über die Funktion „c“ (combine) tun:

```
Testliste<-c(1,2,3,4,5)
```

Bei dem „<-“ handelt es sich übrigens um einen Zuweisungspfeil. Dieser kann in fast allen Fällen auch durch ein Gleichheitszeichen ersetzt werden – da die Pfeile jedoch in den meisten R-Standardwerken Verwendung finden, tun sie es auch in diesem Skript. Wenn wir diese Variable nun durch Eingabe von

```
Testliste
```

aufrufen, gibt R alle zuvor eingegebenen Werte aus. Über die Funktion „length“ können wir in Erfahrung bringen, wie viele Werte eine Variable enthält:

```
length(Testliste)
```

Für alle weiteren Übungen in diesem Skript soll nachfolgend ein fiktiver Datensatz angelegt werden, der aus drei Variablen (Geschlecht, Alter und Einkommen) besteht und der die Daten von insgesamt 30 Probandinnen und Probanden enthält. Hierzu initialisieren wir zunächst jede Variable einzeln[1]:

```
Geschlecht<-c
("m","m","m","w","m","w","w","w","m","m","m","m","m","m","w","m","w","w","w","m","m","m","m","
m","w","m","w","w","w","m","m")

Alter<-
c(21,24,23,26,31,30,36,31,32,29,38,39,43,42,40,46,46,48,45,50,53,58,62,48,50,51,57,48,42,53)

Einkommen<-
c(2300,2100,2300,2100,2600,2800,3000,2700,3100,2700,3300,3600,3500,3500,3600,4000,4200,380
0,3900,4000,4000,4200,4300,4700,4500,4500,5000,4800,5100,5300)
```

Im Global Environment existieren jetzt drei Variablen mit jeweils 30 Werten, die sich über die Eingabe des jeweiligen Variablennamens abrufen lassen. Diese Variablen müssen nun noch in einem Datensatz zusammengeführt werden. Dies geschieht über die Funktion „data.frame“, der wir die drei Variablen als Argumente übergeben müssen:

```
Probanden<-data.frame(Alter,Geschlecht,Einkommen)
```

Im Global Environment sieht man nun einen neuen Bereich („Data“), in dem ein Datensatz mit der Bezeichnung „Probanden“ angezeigt wird. Ein Klick auf den blauen Pfeil links neben der Bezeichnung öffnet eine kurze Übersicht der in diesem Datensatz enthaltenen Variablen. Mit einem Doppelklick auf die Bezeichnung lässt sich der Datensatz in einem Editor oberhalb der Shell öffnen.

[1] Wichtiger Hinweis: Alle Codebeispiele aus diesem Skript lassen sich kopieren und in die Shell einfügen – es ist also nicht erforderlich, jeden Wert einzeln abzutippen.

Wenn wir

```
Probanden
```

in die Shell eingeben, sehen wir erstmals eine R-Ausgabe, die optisch unseren Erwartungen an einen „richtigen“ Datensatz entspricht. Die drei zuvor angelegten Einzelvariablen „Alter“, „Geschlecht“ und „Einkommen“ werden nicht mehr benötigt und können mit der Funktion „rm“ (remove) gelöscht werden. Im Global Environment verbleibt danach ausschließlich der Datensatz „Probanden“.

```
rm(Alter,Einkommen,Geschlecht)
```

3.3 Speichern und Laden von Daten

Würde man RStudio nun schließen, wäre der mühsam eingegebene Probanden-Datensatz wieder verloren. Wie lassen sich also Datensätze in R speichern und laden? Zum Speichern wird zunächst die Funktion „write.table“ benötigt, der wir als Argument den zu speichernden Datensatz sowie den Namen der Datei übergeben müssen, in der die Daten gespeichert werden sollen:

```
write.table(Probanden,"Datensatz.txt")
```

Im Datei-Explorer (zu erreichen über den Reiter „Files“ im unteren rechten RStudio-Fenster) sollte nun im Arbeitsverzeichnis von R die Datei „Datensatz.txt“ zu sehen sein. Um den Datensatz wieder in R zu laden, benötigt man analog zu „write.table“ die Funktion „read.table“. Um das testen zu können, kann man den gespeicherten Datensatz jetzt zunächst bedenkenlos aus R löschen:

```
rm(Probanden)
```

Gibt man nun

```
Probanden
```

ein, erfolgt eine Fehlermeldung – der Datensatz ist also in der Tat nicht mehr verfügbar. Nun wollen wir den Datensatz aus der gespeicherten Datei „Datensatz.txt“ wiederherstellen:

```
Probanden<-read.table("Datensatz.txt")
```

Gibt man nun erneut

```
Probanden
```

ein, wird der Datensatz wieder wie zuvor angezeigt – sowohl das Speichern als auch das Laden haben also funktioniert. Wo aber speichert R also die Dateien ab? Das geschieht im R-Arbeitsverzeichnis, das bei der Installation von R bzw. RStudio festgelegt wird. Über die Funktion „getwd“ (get working directory) lässt sich das aktuelle Arbeitsverzeichnis aber auch direkt abrufen:

```
getwd()
```

Soll das Arbeitsverzeichnis neu gesetzt werden – beispielsweise auf C:\R-Daten – kann dies über die Funktion „setwd“ (set working directory) erfolgen. Das Verzeichnis wird von R nicht neu erzeugt, sondern muss vorab schon angelegt werden:

```
setwd("C:/R-Daten")
```

3.4 Ausgabe von Daten

Die nun im Datensatz „Probanden“ enthaltenen Angaben zum Geschlecht können über die direkte Ansprache der Variablen ausgegeben werden. Hierfür ist die Bezeichnung des Datensatzes gefolgt von einem Dollarzeichen $ und der Bezeichnung der Variablen einzugeben, also:

```
Probanden$Geschlecht
```

Wie man sieht, werden hier einfach die Rohdaten über mehrere Zeilen ausgegeben. Nützlicher ist in den meisten Fällen eine Häufigkeitstabelle, die über die Funktion „table“ erzeugt werden kann:

```
table(Probanden$Geschlecht)
```

Bei längeren Datensätzen ist die Ausgabe aller Daten selten zielführend. Wenn man sich trotzdem einen Eindruck von den in einem Datensatz enthaltenen Variablen und ihren Werten verschaffen möchte, lassen sich über die Funktion „head“ die Tabellenköpfe sowie die ersten sechs Werte jeder Variablen anzeigen:

```
head(Probanden)
```

Enthält der Datensatz fehlende Werte (was bei unserem Beispieldatensatz nicht der Fall ist), kann über das Argument „useNA=“ifany““ festgelegt werden, dass in der Ausgabe auf die Anzahl bzw. auf die Position fehlender Werte im Datensatz hingewiesen wird.

4. Mittelwerte / Lagemaße

4.1 Arithmetisches Mittel

Das arithmetische Mittel kann in R über die Funktion „mean“ bestimmt werden. In diesem und allen weiteren nachfolgenden Beispielen greifen wir dabei jeweils auf den Datensatz „Probanden“ zurück. Um eine Variable in einem Datensatz in R ansprechen zu können, muss die Datensatzbezeichnung über ein Dollarzeichen ($) mit der Variablenbezeichnung kombiniert werden. Die Ausgabe der in „Probanden“ gespeicherten Alterswerte erfolgt also über die Eingabe von

```
Probanden$Alter
```

Diese Nomenklatur ist bei der Berechnung des arithmetischen Mittels via „mean“ zu berücksichtigen:

```
mean(Probanden$Alter)
```

4.2 Getrimmtes arithmetisches Mittel

Das getrimmte arithmetische Mittel kann ebenfalls über die Funktion „mean“ berechnet werden, der über das zusätzliche Argument „trim“ der prozentuale Anteil der Trimmung zu übergeben ist. Soll also etwa das um 10% getrimmte arithmetische Mittel berechnet werden, ist dies wie folgt möglich:

```
mean(Probanden$Alter,trim=0.1)
```

4.3 Median

Der Median kann über die Funktion „median“ bestimmt werden:

```
median(Probanden$Alter)
```

Befinden sich fehlende Werte im Datensatz (was in unserem Beispiel nicht der Fall ist), muss R über das Argument „na.rm = TRUE“ (na = not available / Wert nicht vorhanden) mitgeteilt werden, dass fehlende Werte bei der Berechnung ignoriert werden können:

```
mean(Probanden$Alter,na.rm=TRUE)
```

Befinden sich keine fehlenden Werte im Datensatz, ändert sich durch die Übergabe des Arguments nichts am Ergebnis. Diese Vorgehensweise gilt analog für die Kalkulation anderer Mittelwerte und Streuungsmaße.

4.4 Quartile

Beliebige Perzentilwerte können mit Hilfe der Funktion „quantile“ errechnet werden, welcher als Argumente die entsprechende Variable sowie den prozentualen Anteil derjenigen Werte in der Verteilung übergeben werden müssen, die kleiner oder gleich dem Perzentilwert sein sollen.

Die drei Quartilswerte (25%-, 50%- und 75%-Perzentil) berechnen sich somit wie folgt:

```
quantile(Probanden$Einkommen,0.25)
quantile(Probanden$Einkommen,0.55)
quantile(Probanden$Einkommen,0.75)
```

Über die Funktion „summary“ lässt sich zudem eine Sechs-Werte-Zusammenfassung für eine Verteilung ausgeben, die neben den drei Quartilswerten noch den größten und den kleinsten Wert der Verteilung sowie das arithmetische Mittel umfasst:

```
summary(Probanden$Einkommen)
```

4.5 Modus

Der Modus kann in R standardmäßig nicht direkt ausgegeben werden. Allerdings lässt sich eine Häufigkeitstabelle erzeugen, der man entnehmen kann, ob einzelne Werte besonders häufig im Datensatz auftauchen. Hierfür wird die Funktion „table“ benötigt, der man die darzustellende Variable als Argument übergeben muss:

```
table(Probanden$Geschlecht)
```

Alternativ kann auch eine visuelle Prüfung über ein Balkendiagramm vorgenommen werden (zu Grafiken und deren Customizing später mehr). Diesem lässt sich auch direkt entnehmen, ob es sich um eine unimodale Verteilung handelt.

```
barplot(table(Probanden$Geschlecht))
```

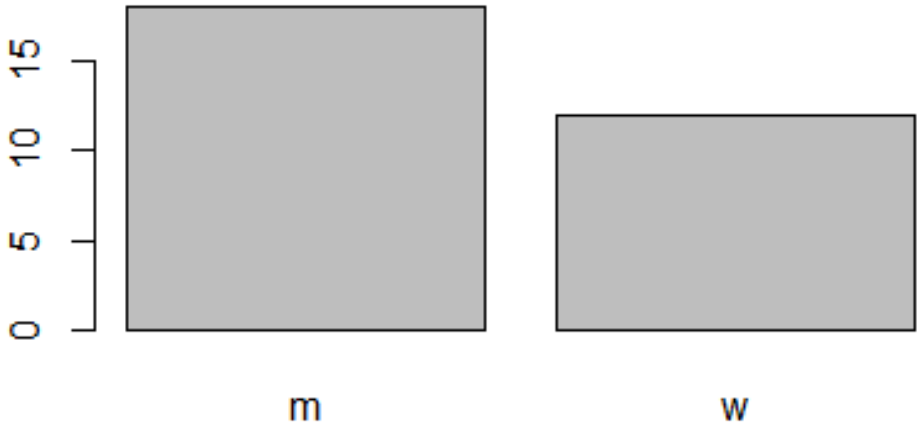

Abbildung 4: Balkendiagramm der Geschlechterverteilung

5. Streuungsmaße / Dispersionsparameter

5.1 Spannweite

Für die Ausgabe der Spannweite existiert in R keine Standardfunktion. Mit Hilfe der beiden Funktionen „max“ und „min“, die jeweils den größten und den kleinsten Wert einer Verteilung zurückliefern, lässt sich dieses Manko jedoch leicht beheben. Liefern uns nämlich

```
max(Probanden$Alter)
```

und

```
min(Probanden$Alter)
```

die jeweils größte und kleinste Altersangabe, so erhalten wir mit

```
max(Probanden$Alter)-min(Probanden$Alter)
```

die Spannweite.

Exkurs: Speichern einer eigenen Funktion

Wenn wir die Berechnung der Spannweite nun dauerhaft als eigene Funktion mit der Bezeichnung „Spannweite“ in R hinterlegen wollen, müssen wir wie folgt vorgehen:

```
Spannweite <- function(Variable) {max(Variable)-min(Variable)}
```

Mit dieser Befehlszeile erzeugen wir eine neue Funktion unter der Bezeichnung „Spannweite“, der ein einziges Argument (die „Variable“, die wir auch anders nennen könnten) zu übergeben ist. Die Funktion berechnet die Spannweite dieser Variable, indem sie die Differenz von Maximal- und Minimalwert bildet und ausgibt. Die neue Funktion ist nun direkt in R verfügbar und kann über

```
Spannweite(Probanden$Alter)
```

getestet werden.

Ein einfacher Weg, solche „selbstgestrickten“ Funktionen dauerhaft zu speichern, ist die Anlage eines kommentierten R-Skripts unter einer passenden Bezeichnung (z.B. „Meine Funktionen“). Hierzu ist in RStudio auf das grüne Plus-Symbol oben links zu klicken und „R Script“ zu selektieren. Es öffnet sich nun ein neues Fenster oberhalb der Shell, in welches das Skript eingegeben werden kann. In diesem Fenster kann man die Zeile

```
Spannweite<-function(Variable){max(Variable)-min(Variable)}
```

hinterlegen und das Skript dann mit einem Klick auf das Diskettensymbol abspeichern. Falls die Funktion noch kommentiert werden soll, kann ein Kommentar mit einem Hashtag (#) hinter der Funktion angelegt werden, also z.B. so:

```
Spannweite<-function(Variable){max(Variable)-min(Variable)} # Diese Funktion berechnet die Spannweite einer ihr übergebenen Variablen.
```

Wenn wir die Funktion nun mittels

```
rm(Spannweite)
```

löschen, können wir sie anschließend wiederherstellen, indem wir auf die entsprechende Zeile im R-Skript klicken und den Button „Run“ betätigen. In „Meine Funktionen“ können wir nun zeilenweise alle von uns neugeschaffenen Funktionen kommentiert hinterlegen und jederzeit in R laden.

5.2 Interquartilsabstand

Im Gegensatz zur Spannweite ist der Interquartilsabstand als Funktion „IQR“ (Inter Quartile Range) wieder standardmäßig in R hinterlegt:

```
IQR(Probanden$Alter)
```

5.3 Varianz und Standardabweichung

Auch Varianz und Standardabweichung sind als Funktionen „var“ (Variance) und „sd“ (Standard Deviation) standardmäßig in R hinterlegt:

```
var(Probanden$Alter)
sd(Probanden$Alter)
```

5.4 Variationskoeffizient

Genau wie die Berechnung der Spannweite ist die Berechnung des Variationskoeffizienten nicht standardmäßig in R enthalten und muss somit aus der Formel (Division der Standardabweichung durch das arithmetische Mittel) hergeleitet werden. Sie kann als zweite eigene Funktion im R-Skript „Meine Funktionen“ hinterlegt werden.

Zunächst einmal liefert uns die Eingabe von

```
sd(Probanden$Alter)/mean(Probanden$Alter)
```

den Variationskoeffizienten der Altersvariablen. Eine neue Funktion können wir – analog zur Erstellung der Spannweitenfunktion – wie folgt generieren:

```
Variationskoeffizient<-function(Variable){sd(Variable)/mean(Variable)}
```

Die neue Funktion kann wie oben beschrieben kommentiert und in das R-Skript „Meine Funktionen“ eingefügt und so dauerhaft gesichert werden.

6. Korrelationskoeffizienten

6.1 Bravais-Pearson-Korrelationskoeffizient

Alle in der Vorlesung besprochenen Korrelationskoeffizienten können in R über die Funktion „cor“ berechnet werden, der als Argumente die beiden zu betrachtenden Variablen sowie die Methode (Bravais-Pearson, Spearman, Kendall) übergeben werden müssen. Die Reihenfolge der Variablen (Welches ist die x- und welches die y-Variable?) spielt dabei – analog zur händischen Rechnung – keine Rolle, da Korrelationskoeffizienten bekanntlich keine Wirkungsrichtung kennen.

Der Bravais-Pearson-Korrelationskoeffizient für die beiden Variablen Alter und Einkommen in unserem Beispieldatensatz berechnet sich somit wie folgt:

```
cor(Probanden$Alter,Probanden$Einkommen,method="pearson")
```

Da der Bravais-Pearson-Korrelationskoeffzient die Default-Methode dieser Funktion ist, kann auf das „method“-Argument in diesem Fall auch verzichtet werden – es wird jedoch empfohlen, es grundsätzlich zu verwenden, um nie aus den Augen verlieren zu können, welcher Koeffizient berechnet wurde.

Im Zusammenhang mit Korrelationskoeffizienten von grundsätzlichem Interesse ist sicherlich die Ausgabe von Streudiagrammen. Diese kann in R über die Funktion „plot“ erfolgen der beide Variablen als Argumente zu übergeben sind. Die Reihenfolge der Variablen wirkt sich auf die Achsenzuweisung aus – die erste übergebene Variable wird an der x-Achse, die zweite übergebene Variable an der y-Achse abgetragen:

```
plot(Probanden$Alter,Probanden$Einkommen)
```

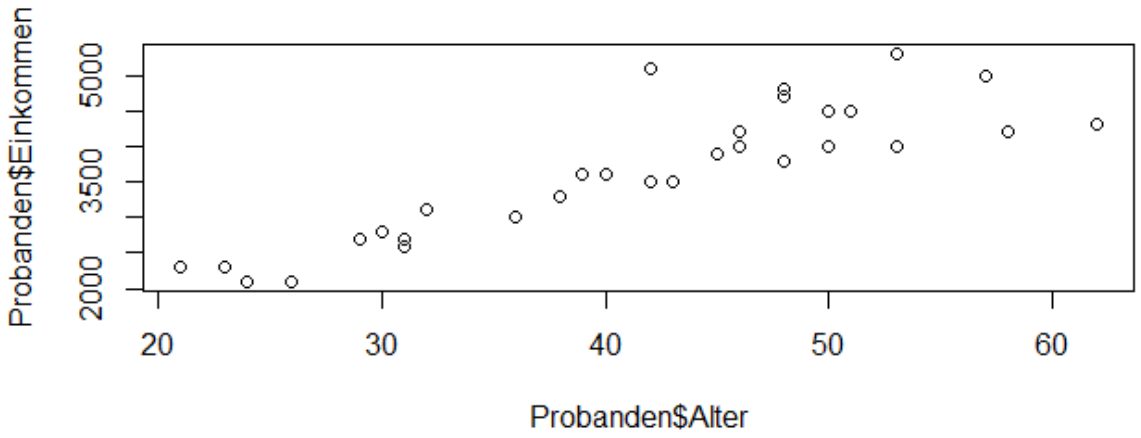

Abbildung 5: Streudiagramm der Alters- und Einkommensverteilung

Bei der Betrachtung des Streudiagramms fällt auf, dass es an der einen oder anderen Stelle noch optimiert werden könnte – störend ist insbesondere die Beschriftung der Achsen mit „Probanden$Alter“ und „Probanden$Einkommen“. Wie dieses Customizing von Grafiken in R funktioniert, wird im nachfolgenden Abschnitt näher erläutert.

6.2 Rangkorrelationskoeffizient nach Spearman

Analog zur Berechnung des Bravais-Pearson-Korrelationskoeffizienten berechnet sich der Rangkorrelationskoeffizient nach Spearman in R über

```
cor(Probanden$Alter,Probanden$Einkommen,method="spearman")
```

6.3 Konkordanzkoeffizient nach Kendall

…sowie der Konkordanzkoeffizient nach Kendall über

```
cor(Probanden$Alter,Probanden$Einkommen,method="kendall")
```

7. Grafische Darstellungsformen

7.1 Balkendiagramme

Balkendiagramme können in R mit Hilfe der Funktion „barplot“ erzeugt werden. Als Argument ist dabei nicht die darzustellende Variable, sondern die Häufigkeitstabelle dieser Variablen zu übergeben, die über die Funktion „table“ generiert werden kann.

Folgende Eingabe generiert eine Häufigkeitstabelle für die Geschlechterverteilung:

```
table(Probanden$Geschlecht)
```

Die Darstellung dieser Häufigkeitsverteilung als Balkendiagramm erfolgt dann über:

```
barplot(table(Probanden$Geschlecht))
```

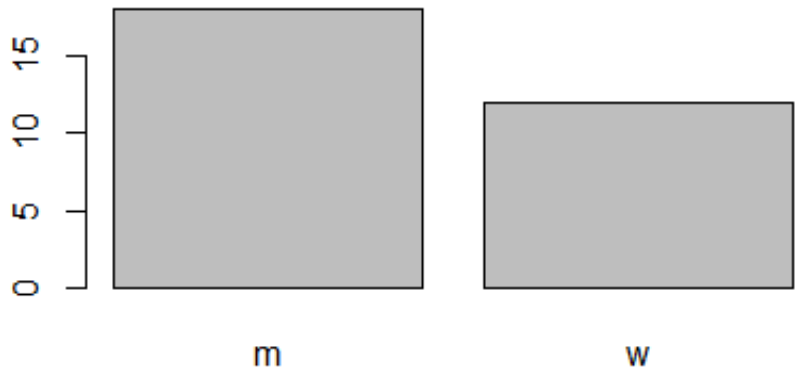

Abbildung 6: Balkendiagramm der Geschlechterverteilung

Wie bereits zuvor festgestellt wurde, lässt diese Grafik optisch noch einiges zu wünschen übrig. Grundsätzlich können Grafiken in R durch die Übergabe zusätzlicher Argumente an die jeweilige Funktion auf vielfältige Art und Weise modifiziert werden. Für die obige Grafik könnten wir uns beispielsweise folgende Modifikationen wünschen:

- Passende Überschrift
- Passende Unterschrift
- Einfärbung der Balken in blau und rot
- Beschriftung der y-Achse mit „Anzahl der Befragten“
- Beschriftung der Balken als „männlich“ und „weiblich“
- Skalierung der y-Achse bis zur 25 zur Entzerrung der Grafik

Die Umsetzung dieser Modifikationen resultiert in diesem überlangen Funktionsaufruf:

```
barplot(table(Probanden$Geschlecht),main="Geschlechterverteilung",sub="Ergebnisse einer repräsentativen Befragung",col=c("lightblue","lightgrey"),ylab="Anzahl der Befragten",names.arg=c("männlich","weiblich"),ylim=c(0,25))
```

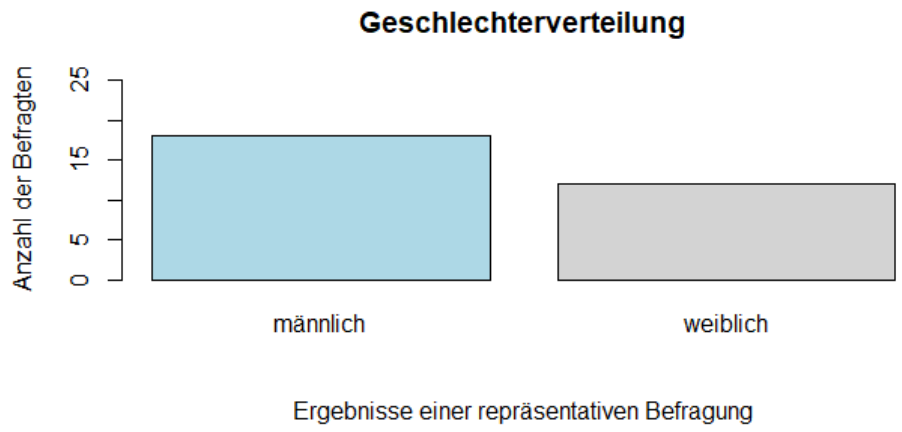

Abbildung 7: Modifiziertes Balkendiagramm der Geschlechterverteilung

Da die Möglichkeiten zum Customizing von Grafiken extrem vielfältig sind, sollen an dieser Stelle nur einige wichtige Argumente aufgelistet werden. Die hier vorgestellten Argumente lassen sich auch bei der Erzeugung anderer Grafikformen als dem Balkendiagramm einsetzen und werden daher bei der Vorstellung dieser Grafikformen nachfolgend nicht wiederholt. Zum besseren Verständnis wird empfohlen, bei Gelegenheit einfach mal ein wenig mit den Argumenten zu experimentieren.

Beschriftungen

main="Überschrift"	-> Überschrift der Grafik
sub="Untertitel"	-> Untertitel der Grafik
ylab="Y-Achsenlabel"	-> Beschriftung der y-Achse
xlab="x-Achsenlabel"	-> Beschriftung der x-Achse
font.main=1	-> Formatierung der Überschrift (z.B. 1,2,3…)
font.sub=1	-> Formatierung des Untertitels (z.B. 1,2,3…)
font.axis=1	-> Formatierung der Achsenbeschriftungen (z.B. 1,2,3…)
names.arg=c(„m","w")	-> Beschriftung der grafischen Elemente (passende Anzahl!)

Aufbau und Skalierung

ylim=c(0,100)	-> Skalierung der y-Achse (hier: 0-100)
xlim=c(0,100)	-> Skalierung der x-Achse (hier: 0-100)
horizontal=TRUE	-> Seitliches Kippen der Grafik

Farben und Aussehen

col="lightblue"	-> Farbe aller Hauptelemente
col=c(„blue","red")	-> Farbe der grafischen Elemente (passende Anzahl!)
border="green"	-> Farbe der Hauptelementlinien
col.axis="blue"	-> Farbe der Achsenbeschriftungen
col.lab="blue"	-> Farbe der Achsenlabels
col.main="blue"	-> Farbe der Überschrift
col.sub="blue"	-> Farbe des Untertitels

Hinweis: Die Funktion „col“ liefert einen Überblick aller in R verfügbaren Farbtöne. Im Internet finden sich zahlreiche Übersichten[2], in denen die Farbtöne gemeinsam mit den jeweiligen Bezeichnungen dargestellt werden, so dass man den gewünschten Farbton leichter selektieren kann.

```
col()
```

In RStudio erzeugte Grafiken lassen sich übrigens ganz einfach speichern oder exportieren, indem man auf den „Export“-Button oberhalb der Grafikausgabe klickt.

7.2 Streudiagramme

Streudiagramme können in R mit der Funktion „plot“ erzeugt werden, der zwei Variablen als Argumente zu übergeben sind. Zu beachten ist, dass sich die Reihenfolge der Übergabe auf die Darstellung auswirkt – die erste übergebene Variable wird an der x-Achse, die zweite übergebene Variable an der y-Achse abgetragen. Ein (unmodifiziertes) Streudiagramm der Altes- und Einkommensverteilung lässt sich somit wie folgt erzeugen:

```
plot(Probanden$Alter,Probanden$Einkommen)
```

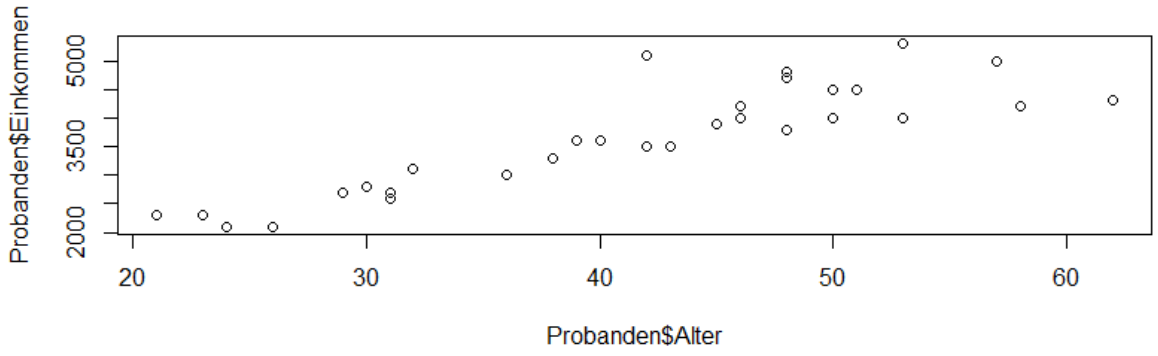

Abbildung 8: Streudiagramm der Alters- und Einkommensverteilung

[2] Eine gute Übersicht gibt es beispielsweise hier: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Ein Attribut, dass bei Balkendiagrammen keine Rolle spielt, bei Streudiagrammen aber schon, ist die Auswahl des verwendeten Punktsymbols, die über „pch“ erfolgt. Der Default ist hier „pch=1“, was der Verwendung des Kreissymbols entspricht. Bei „pch=2“ ergibt sich folgende Grafik:

```
plot(Probanden$Alter,Probanden$Einkommen, pch=2)
```

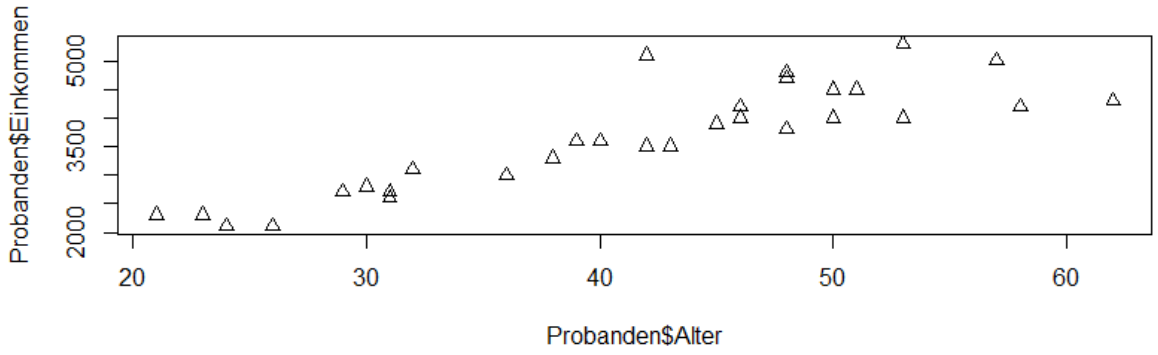

Abbildung 9: Modifiziertes Streudiagramm der Alters- und Einkommensverteilung

Folgende Symbole werden häufig verwendet:

pch=1	-> Kreis
pch=2	-> Dreieck
pch=3	-> Kreuz
pch=4	-> Gedrehtes Kreuz
pch=5	-> Raute
pch=6	-> Gedrehtes Dreieck
pch=7	-> Qudrat mit Kreuz
pch=8	-> Stern
pch=9	-> Raute mit Kreuz
pch=10	-> Kreis mit Kreuz
pch=11	-> Zwei Dreiecke
pch=12	-> Quadrat mit Kreuz
pch=13	-> Kreis mit Stern
pch=14	-> Quadrat und Dreieck
pch=15	-> Ausgefülltes Quadrat
pch=16	-> Ausgefüllter Kreis
pch=17	-> Ausgefülltes Dreieck
pch=18	-> Ausgefüllter Diamant

7.3 Box-Plots

Box-Plots können in R mit Hilfe der Funktion „boxplot“ erzeugt werden. Um die Grafiken besser ins Skript einfügen zu können, wurde über das Attribut „horizontal=TRUE“ eine seitlich gekippte Darstellung gewählt.

```
boxplot(Probanden$Einkommen,horizontal=TRUE)
```

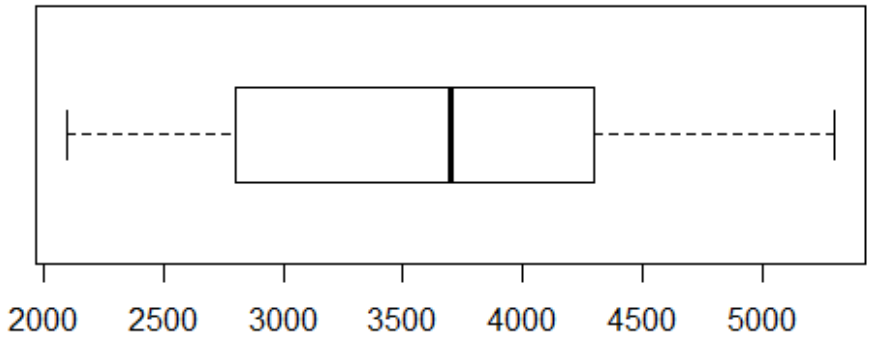

Abbildung 10: Horizontal gekippter Box-Plot

Über die Eingabe von

```
boxplot(Probanden$Einkommen~Probanden$Geschlecht,horizontal=TRUE)
```

erhält man einen gruppierten Box-Plot, der eine Einkommensverteilung pro Geschlecht enthält. Die Gruppierung von Elementen über eine Tilde (~) ist auch bei anderen Grafikformaten möglich.

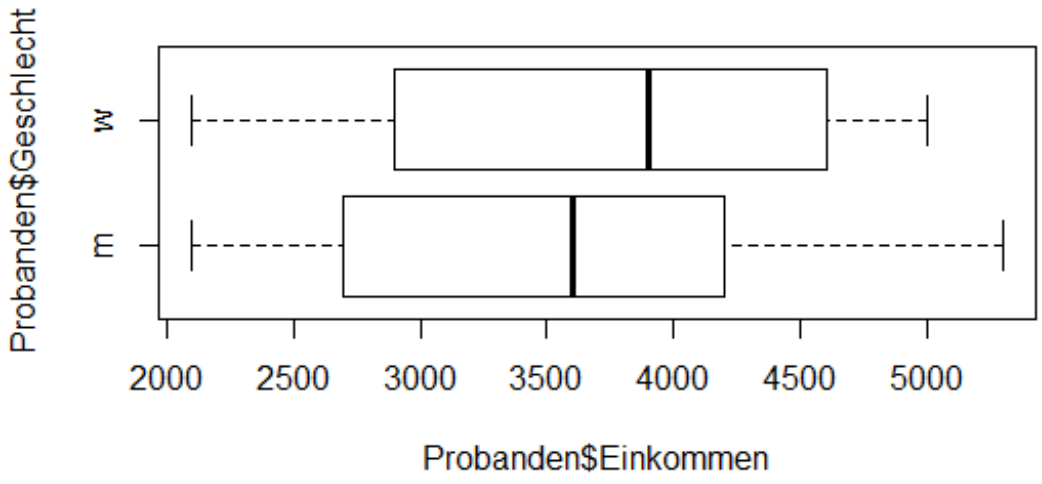

Abbildung 11: Nach Geschlechtern gruppierter Box-Plot der Einkommensverteilung

7.4 Stem-and-Leaf-Plots

Auch Stem-and-Leaf-Plots können in R erzeugt werden, werden jedoch nicht in der Grafikanzeige, sondern direkt in der Shell ausgegeben, da sie lediglich aus Ziffern und Zeichen bestehen. Die für die Erstellung von Stem-and-Leaf-Plots erforderliche Funktion lautet „stem“, als Argument muss ihr mindestens die darzustellende Variable übergeben werden:

```
stem(Probanden$Alter)
```

```
2 | 134
2 | 69
3 | 0112
3 | 689
4 | 0223
4 | 566888
5 | 00133
5 | 78
6 | 2
```

Es fällt auf, dass R automatisch eine Stammbreite von 5 wählt. Soll ist eine größere Stammbreite gewünscht, muss diese über das zusätzliche Argument „stem“ bestimmt werden. Der Default liegt hier bei 1 (von R automatisch gewählte Stammbreite), für eine größere Stammbreite sind kleinere Werte, für eine kleinere Stammbreite größere Werte zu wählen. So lässt sich etwa ein Diagramm mit einer Stammbreite von 10 (doppelte Stammbreite von 5, d.h. 0,5 statt 1) mit folgendem Funktionsaufruf erzeugen:

```
stem(Probanden$Alter,scale=0.5)
```

```
2 | 13469
3 | 0112689
4 | 0223566888
5 | 0013378
6 | 2
```

7.5 Histogramme

Histogramme können in R mit der Funktion „hist“ erzeugt werden, der als Argument mindestens die darzustellende Variable zu übergeben ist. Ein einfaches Histogramm der Einkommensverteilung erhalten wir über:

```
hist(Probanden$Einkommen)
```

Die Anzahl der Klassen wird in diesem Falle durch R selbst festgelegt, alternativ lässt sie sich über das Attribut „breaks“ bestimmen. So erhalten wir etwa mit

```
hist(Probanden$Einkommen,breaks=10)
```

ein Histogramm mit 10 Klassen (soweit möglich), während

```
hist(Probanden$Einkommen,breaks=20)
```

ein Histogramm mit 20 Klassen (soweit möglich) generiert.

Um das Histogramm optisch etwas aufzuwerten, können wir über das Argument „col“ noch zwei Farben übergeben, mit denen die Balken dann wechselseitig eingefärbt werden:

```
hist(Probanden$Einkommen,breaks=10,col=c("lightblue","lightgrey"))
```

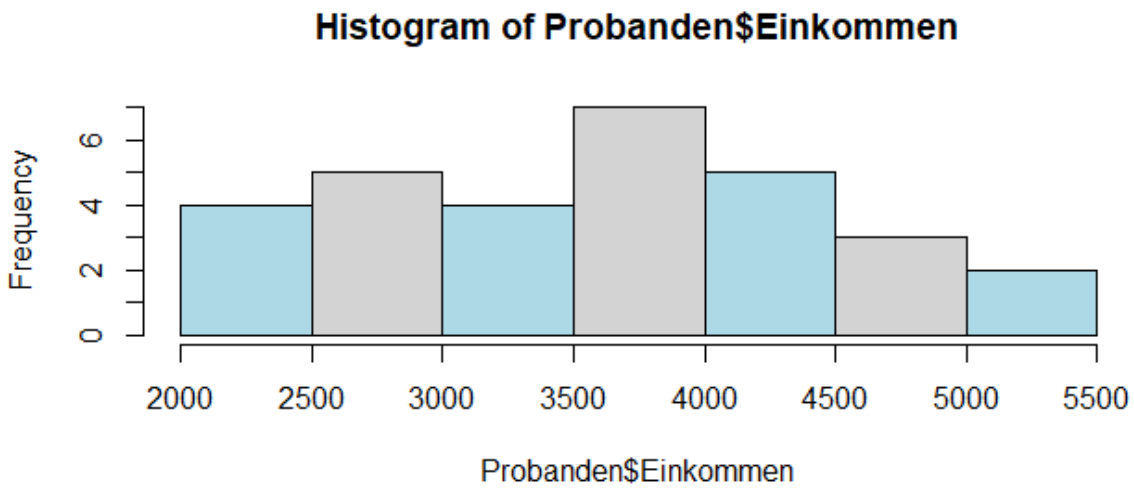

Abbildung 12: Modifiziertes Histogramm der Einkommensverteilung

Zum Abschluss dieses Abschnitts zu grafischen Darstellungen in R sei noch einmal erwähnt, dass dieses Skript zum einen nur eine erste Einführung in einen Bruchteil der Möglichkeiten von R enthält – und dass fast alle vorgestellten Argumente sich mit fast allen grafischen Darstellungsformen kombinieren lassen. Dies soll abschließend durch die Modifikation des obenstehenden Histogramms noch einmal unterstrichen werden.

```
hist(Probanden$Einkommen,breaks=10,col=c("lightblue","lightgrey"),main="Einkommensvert
eilung",xlab="Einkommen in Euro",ylab="Personenzahl",font.main=2,sub="Daten einer
repräsentativen Erhebung",font.sub=3,xlim=c(1100,6000),lwd=2)
```

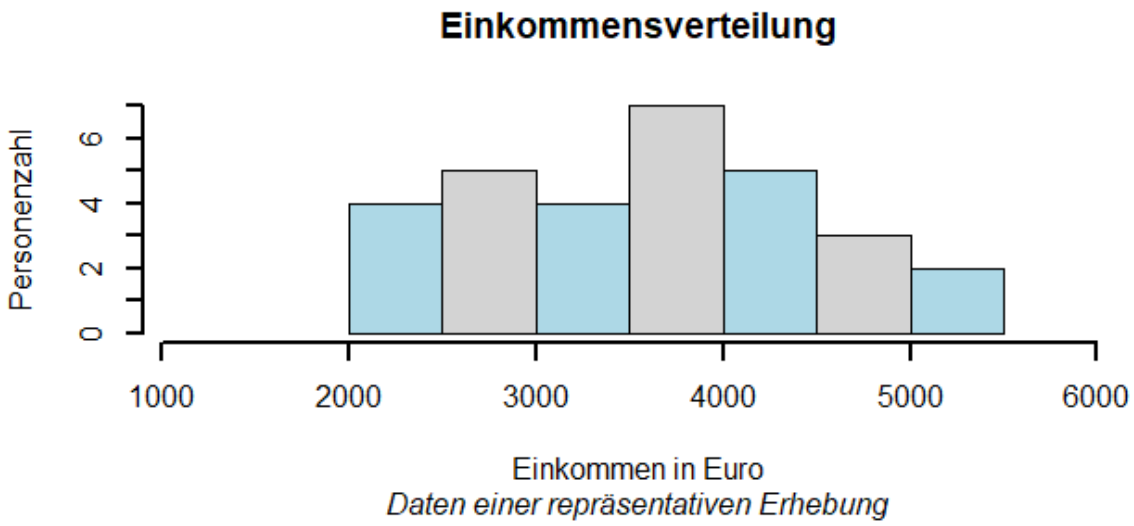

Abbildung 13: Modifiziertes Histogramm der Einkommensverteilung

8. Auswahl von Teilmengen

Gelegentlich wird es erforderlich sein, eine Untermenge von Daten aus einem Datensatz zu selektieren – im Falle unseres Beispieldatensatzes also etwa den Gehaltsmedian ausschließlich für die männlichen Probanden zu berechnen. Hierfür ist die jeweilige Funktion in R um eine Selektion zu ergänzen, die in eckigen Klammern [] ans Ende des Funktionsaufrufs gestellt wird. In dieser Selektion wird festgelegt, welche Eigenschaften Daten erfüllen müssen, um beim Durchlauf der Funktion berücksichtigt zu werden. Sechs Operatoren sind hier für uns von Relevanz:

```
==    -> Auswahl aller Daten, die einen bestimmten Wert / Ausdruck annehmen
!=    -> Auswahl aller Daten, die einen bestimmten Wert / Ausdruck nicht annehmen
>     -> Auswahl aller Daten, die größer als ein bestimmter Wert sind
<     -> Auswahl aller Daten, die kleiner als ein bestimmter Wert sind
>=    -> Auswahl aller Daten, die größer als oder gleich einem bestimmten Wert sind
<=    -> Auswahl aller Daten, die kleiner als oder gleich einem bestimmten Wert sind
```

Die Kombination von Selektionen mit einem „&“ ist möglich. Die nachfolgenden Beispiele sollen den Einsatz von Selektionen verdeutlichen. Berechnet wird jeweils der Einkommensmedian für eine bestimmte Subgruppe an Probandinnen und Probanden.

```
median(Probanden$Einkommen[Probanden$Geschlecht=="m"])
```

-> nur männliche Probanden

```
median(Probanden$Einkommen[Probanden$Geschlecht!="m"])
```

-> nur Probanden, die nicht männlich sind

```
median(Probanden$Einkommen[Probanden$Einkommen>3000])
```

-> nur Probanden mit einem Einkommen größer als 3.000 EUR

```
median(Probanden$Einkommen[Probanden$Einkommen<3000])
```

-> nur Probanden mit einem Einkommen kleiner als 3.000 EUR

```
median(Probanden$Einkommen[Probanden$Einkommen>=3000])
```

-> nur Probanden mit einem Einkommen größer oder gleich 3.000 EUR

```
median(Probanden$Einkommen[Probanden$Einkommen<=3000])
```

-> nur Probanden mit einem Einkommen kleiner oder gleich 3.000 EUR

```
median(Probanden$Einkommen[Probanden$Geschlecht=="w"&Probanden$Alter>40])
```

-> nur weibliche Probanden mit einem Alter von mehr als 40 Jahren

Derartige Selektionen können grundsätzlich mit einer Vielzahl von Funktionen kombiniert werden, so etwa mit der Ausgabe von Tabellen („table“), mit der Berechnung von Kennzahlen (z.B. „mean“, „var“, „cor“) oder mit der Erstellung von Grafiken (z.B. „barplot“, „boxplot“). Bei Funktionen, denen mehrere Variablen als Argumente übergeben werden müssen (z.B. „cor“) ist zu berücksichtigen, dass eine identische Selektion hinter jede übergebene Variable gesetzt werden muss, d.h.

```
cor(Probanden$Einkommen[Probanden$Geschlecht=="m"],Probanden$Alter[Probanden$Geschlecht=="m"])
```

9. Weitere nützliche Funktionen

Abschließend sollen noch einige Funktionen aufgelistet und kurz beschrieben werden, die für den allgemeinen Umgang mit R im Kontext unserer Vorlesung von Nutzen sind.

Erzeugung von Kreuztabellen

Neben der Ausgabe einfacher Häufigkeitstabellen über die Funktion „table" lassen sich mit R auch Kreuztabellen erzeugen. Hierfür ist einfach eine weitere Variable als zweites Argument zu übergeben.

```
table(Probanden$Einkommen,Probanden$Geschlecht)
```

Bearbeitung von Daten

Über die Funktionen „fix" oder „edit" können Datensätze in einem Editor geöffnet und bearbeitet werden (z.B. Hinzufügung zusätzlicher Fälle, Änderung von Werten).

```
fix(Probanden)
```

Erzeugung von Daten

Mit Hilfe der Funktion „sample" lassen sich Zufallsdaten erzeugen. Die Funktion zieht eine zufällige Stichprobe eines bestimmten Umfangs aus einem definierten Zahlenraum. Sowohl der Zahlenraum als auch der Umfang sind als Argumente zu übergeben. Möchte man beispielsweise 100 per Zufall gezogene Zahlen aus dem Zahlenraum von 1 bis 1.000 erzeugen, so geschieht dies wie folgt:

```
sample(1:1000,100)
```

Sollen sich Zahlen auch wiederholen dürfen, ist mit „replace = TRUE" ein weiteres Argument zu übergeben (der Default liegt hier bei „FALSE", d.h. die Wiederholung von Werten ist unzulässig). In diesem Fall könnte man beispielsweise auch 1.000 Zufallszahlen aus dem Zahlenraum von 1 bis 100 erzeugen:

```
sample(1:100,1000,replace=TRUE)
```

Mit der Funktion „rnorm" lassen sich analog normalverteilte Zufallszahlen generieren – hier z.B. 50 normalverteilte Zufallszahlen aus einem Zahlenraum von 1 bis 100:

```
rnorm(1:100,50)
```

Kontakt

Christian Reinboth, M.Sc.
Dipl.-Wirtsch.-Inf. (FH)

Research Funding Manager
Stabsstelle Forschung
Rektorat

Lehrbeauftragter für Statistik
Transferzentrum Harz

Hochschule Harz
Friedrichstr. 57-59
D-38855 Wernigerode
Haus 6 | Raum 6.110a

Tel: 03943 659 896
Fax: 03943 659 5896
Mobil: 0152 0900 6600
E-Mail: creinboth@hs-harz.de
Online: http://www.hs-harz.de/creinboth/

Hinweise und Anregungen zur Verbesserung dieses Dokuments werden jederzeit dankend angenommen.