Kapitel 4
Die Programmiersprache R

In diesem Kapitel führe ich Sie in wichtige Basics und Eigenheiten der Sprache R ein. Sie werden hier die grundlegende Funktionsweise, wichtige Vokabeln und wichtige Grundlagen der Sprache R kennenlernen.

Kapitel 4 soll Ihnen den ersten Zugang zur Programmierung mit R ermöglichen. Es ist hierzu unabdingbar, die grundlegende Syntax, also die Befehlsstruktur der Sprache R kennenzulernen. Wir werden beginnen, in *Objekten*, *Funktionen* und *Paketen* zu denken, und uns ansehen, wie R »gesprochen« wird.

Mit den oben getätigten Berechnungen haben Sie schon die ersten »Sätze« der Sprache R verfasst und bereits mit der Funktionsweise der Sprache R Bekanntschaft gemacht: R ist eine objektorientierte Sprache, in der Objekte und Funktionsaufrufe eine zentrale Rolle spielen. Mehrere Funktionen in einer Sammlung werden zu Paketen (Packages), die Sie wiederum herunterladen und benutzen können. Im Folgenden stelle ich Ihnen diese drei wichtigen Komponenten der Sprache R dar.

4.1 Objekte

Objekte fassen Daten zusammen. Sie sind sozusagen kleine Speicher, die das, was Sie berechnet, aufgelistet oder in einer Tabelle dargestellt haben, unter einem Objektnamen abspeichern. Den Objektnamen vergeben Sie selbst. Zugewiesen werden die Werte dem Objekt mit dem Operator `<-`. Diese(n) dem einmal kreierten Objekt zugewiesenen Wert(e) können Sie sich dann mit einem einfachen Abruf dieses Objektes (Strg + ↵) ausgeben lassen:

```
q <- 3
q
```

Die Ausgabe:

```
[1] 3
```

oder

```
Variable1 <- 3*(4+3)
Variable1
```

Die Ausgabe:

```
[1] 21
```

Mit der hier kreierten Variablen, und das gilt dann auch für alle anderen von Ihnen kreierten Objekte, können Sie dann weiterrechnen:

```
Variable2 <- 9
Variable2
```

Die Ausgabe:

```
[1] 9
```

```
Variable2 <- Variable1/7
Variable2
```

Die Ausgabe:

```
[1] 3
```

Listing 4.1 Erstellung und Abruf von Variablen

Probieren Sie auch das einfach aus!

4.2 Funktionen

Die Vokabeln der Sprache R sind die Funktionen, die Grammatik ist die ihnen zugrunde liegende Syntax, die Funktionsstruktur. Funktionen werden in R in der Regel mit dieser Syntax aufgerufen:

```
Funktionsname(Argumentname1=Argument1,. . .,ArgumentnameN=ArgumentN)
```

Der *Aufruf* einer Funktion beginnt also

1. mit deren Nennung und
2. dem Öffnen einer Klammer.
3. Dann folgt ein Argument oder nicht, dem ein weiteres folgen kann oder nicht, dem ein weiteres folgen kann oder nicht usw.
4. Schließlich muss die geöffnete Klammer wieder geschlossen werden.

Was in der Theorie kompliziert wirken kann, ist veranschaulicht vielleicht besser nachzuvollziehen. Ein Beispiel für eine einfache R-Funktion ist die Funktion `seq()`. Sie gibt Ihnen eine Zahlenreihe aus, die Sie selbst bestimmen können:


```
seq <- seq(1,10)
seq
```

Die Ausgabe:

```
[1]  1  2  3  4  5  6  7  8  9 10
```

`seq(1,10)` ist also der Befehl, dass R die Funktion `seq()`(erstelle mir eine Zahlensequenz) auf die Argumente `(1,10)` »in einer Zahlenreihe von 1 (Anfang) bis 10 (Ende)« anwenden soll. Wenn Sie nun nur jede zweite Zahl in der Zahlenreihe von 1 bis 10 angegeben haben möchten, dann müssen Sie dies R ebenfalls per Argument innerhalb der Funktion mitteilen:

```
seq2 <- seq(1,10,2)
seq2
```

Die Ausgabe:

```
[1] 1 3 5 7 9
```

`seq(1,10,2)` bedeutet also: »Erstelle mir eine Zahlensequenz« auf die Argumente `(1,10,2)` »in einer Zahlenreihe von 1 bis 10 für jede zweite Zahl«. Schauen Sie dazu gern noch mal in die R-interne Hilfe mit `help(seq)` hinein, auch in die Beispiele. Wie oben sind auch die Ergebnisse Ihrer Funktionsaufrufe in den Objekten gespeichert. Sie können sie nun jederzeit wieder abrufen.

```
seq
```

Die Ausgabe:

```
[1]  1  2  3  4  5  6  7  8  9 10
seq2
```

Die Ausgabe:

```
[1] 1 3 5 7 9
```

Listing 4.2 Funktionsaufrufe am Beispiel der Funktion seq()

Beachten Sie, dass Sie den zweiten Funktionsaufruf unter dem Objekt seq2 gespeichert haben.

4.3 Pakete (Packages)

Pakete sind ebenfalls wichtige, grundlegende Bestandteile der Arbeit mit R. Ein Paket in R ist eine Sammlung von (nützlichen) Funktionen, die Ihnen die Arbeit mit R deutlich erleichtern. Häufig werden Pakete auch mit zusätzlich gespeicherten Datensätzen angeliefert. Ein solches Paket ist das Paket *ggplot2*, das hervorragend für die Datenvisualisie-

rung geeignet ist. Da dieses Paket später die Basis für die Visualisierungen in diesem Buch sein soll, laden Sie es am besten gleich hier herunter.

Wie lade ich Pakete (herunter)?

Gehen Sie zum Herunterladen von Paketen auf den Reiter PACKAGES im Teilbildschirm rechts unten in Ihrem RStudio.

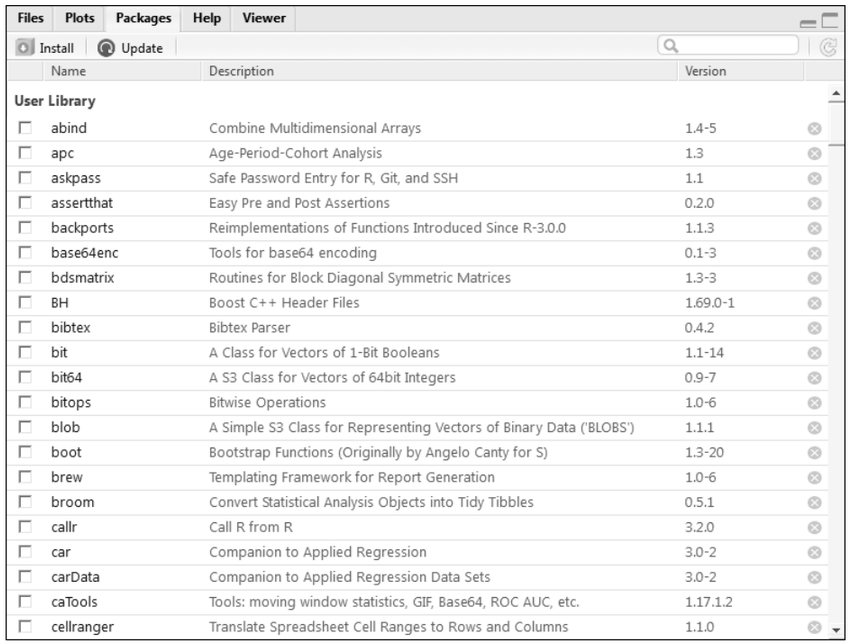

Abbildung 4.1 Pakete installieren, Schritt 1

R zeigt Ihnen an dieser Stelle eine Liste mit allen installierten Paketen, das können vorinstallierte Pakete sein oder Pakete, die Sie selbst bereits installiert haben. Hier können Sie nachprüfen, ob das Paket, das Sie installieren möchten, bereits vorhanden ist oder nicht.

Ist es nicht vorhanden, installieren Sie es. Klicken Sie dazu oben links auf INSTALL. Es öffnet sich ein neues Fenster. Schreiben Sie hier in das mittlere Feld den Namen des Paketes, das Sie installieren wollen, in diesem Fall `ggplot2`. R lädt das Paket voreingestellt von CRAN herunter und speichert es, ebenfalls voreingestellt, in meinem »Documents«-Ordner.

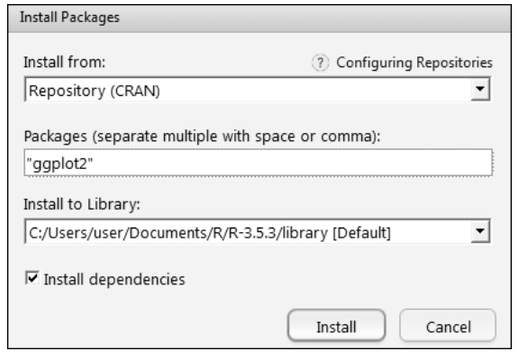

Abbildung 4.2 Pakete installieren, Schritt 2

Tipp

Sie können auch R per Befehl fragen, wo gespeichert wird, nämlich mit dem Befehl

```
getwd()
```

Die Ausgabe:

[1] "C:/Users/user/Documents"

Natürlich ist hier in der Ausgabe mein Dateipfad angegeben. Ihrer ist dann der auf Ihrem Rechner.

Zum Ändern nutzen Sie den Befehl

```
setwd()
```

und schreiben Ihren gewünschten Dateipfad in der gleichen Form wie gerade gesehen zwischen die Klammern.

Wenn Sie nun auf INSTALL klicken, lädt R für Sie das Paket herunter und speichert es für Sie in der »Packages«-Liste aus Abbildung 4.1.

Ein Weg, das Paket zu laden, ist der, die »Packages«-Liste nach dem Paket abzusuchen und links in das Kästchen neben dem Paketnamen einen Haken zu setzen.

Ein zweiter Weg ist das Installieren und Laden der Pakete im Skript selbst. Sie installieren und laden Pakete per Skript wie folgt:

Verwenden Sie die Funktion `install.packages("")`, und schreiben Sie den Namen des Paketes, das Sie herunterladen wollen, in Anführungsstrichen zwischen die Klammern:

```
install.packages("ggplot2")
```

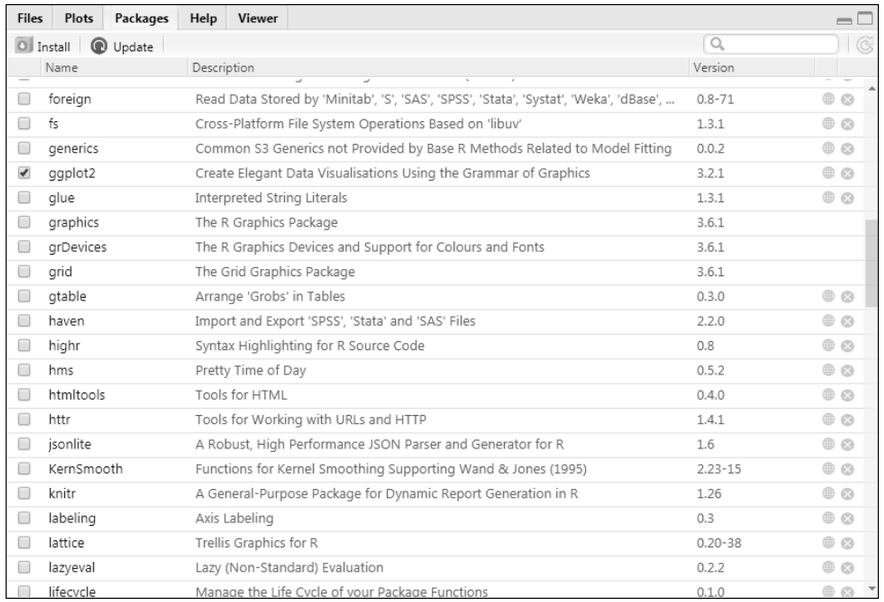

Abbildung 4.3 Händisches Laden von Paketen

Verwenden können Sie die Funktionen und gegebenenfalls die Datensätze der Pakete aber nur, wenn Sie das Paket auch aktivieren. Das machen Sie mit der Funktion

`library()`.

Für das Beispiel ggplot2 schreiben Sie `ggplot2` ohne Anführungszeichen(!) zwischen die Klammern.

```
library(ggplot2)
```

Listing 4.3 Pakete installieren und laden

Achtung

Mit der Funktion `install.packages()` schreiben Sie das Argument immer mit Anführungszeichen(!) und immer mit beiden Anführungszeichen nach oben(!), mit der Funktion `library()` ohne(!). Wenn das Laden also bis hierher nicht funktioniert hat, schauen Sie am besten nach, ob es daran liegt.

Ich lasse häufig alle Pakete, die ich für meine Analyse benötige, am Skriptanfang in einem eigenen Codestück stehen – so muss ich nicht jedes Mal, bevor ich das Skript ausführe, schauen, welche Pakete ich brauchen werde. Ich schreibe auch per *Kommentar*zeile (die beginnt mit #) Stichworte zu den Hauptfunktionen des Pakets hinzu. Ein paar Beispiele finden Sie in Abbildung 4.4.

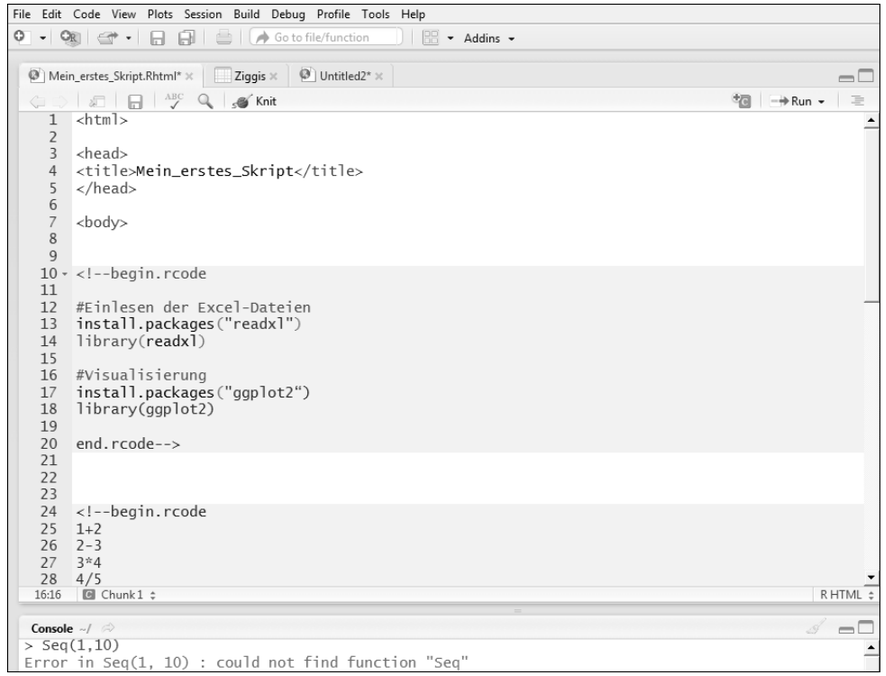

Abbildung 4.4 Pakete installieren und laden per Befehl

Ich kann nun alle Funktionen des heruntergeladenen Paketes und die gegebenenfalls mitgelieferten Datensätze verwenden.

Pakete und ihre Datensätze

Wenn ein Paket einen Datensatz beinhaltet, dann wird der mit dem Laden des Paketes gleich mitgeladen. Datensätze, die mit Paketen angeliefert werden, sind im Prinzip weit verbreitete Übungsdaten, mit denen allerlei Beispiele im Netz gezeigt werden. Sie können sich den Datensatz einfach ansehen, indem Sie ein Objekt kreieren und es dann wieder aufrufen. Ein Datensatz, der mit ggplot2 angeliefert wird, ist der *mpg-Datensatz*. Das finden Sie entweder einfach bei Google heraus (suchen Sie dazu nach »ggplot2

data«, und stöbern Sie ein wenig herum, bis Sie den Datensatz gefunden haben), oder Sie schauen in die Dokumentation des Packages, z. B. unter HELP. Scrollen Sie dann herunter, und suchen Sie den Index des Paketes.

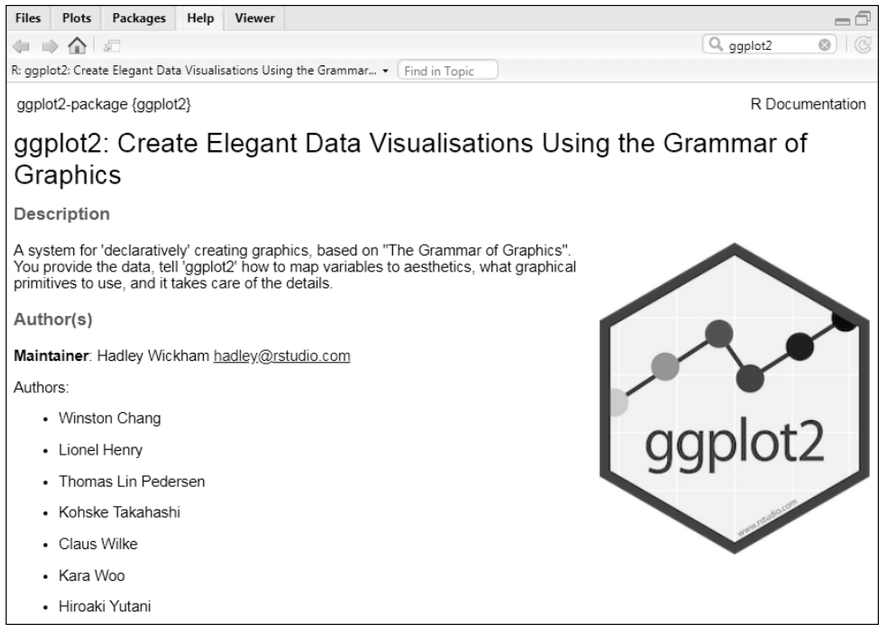

Abbildung 4.5 ggplot2-Dokumentation

Wenn Sie hier nun Zeile für Zeile durchgehen, stoßen Sie auf zwei Datensätze:

1. diamonds (Diamanten), mit der Beschreibung »Prices of 50.000 round cut Diamonds« (Preise von 50.000 rund geschliffenen Diamanten)
2. mpg, mit der Beschreibung »Fuel economy data from 1999 and 2008 for 38 popular models of car« (Benzinverbrauchdaten von 1999 und 2008 für 38 beliebte Automodelle)

Schauen Sie sich den zweiten Datensatz doch mal etwas genauer an. Kreieren Sie dazu ein Objekt, das den Datensatz »mpg« beinhalten soll, und rufen Sie das Objekt dann auf:

```
mpg <- mpg
mpg
```

Listing 4.4 Einladen und Aufrufen des mpg-Datensatzes

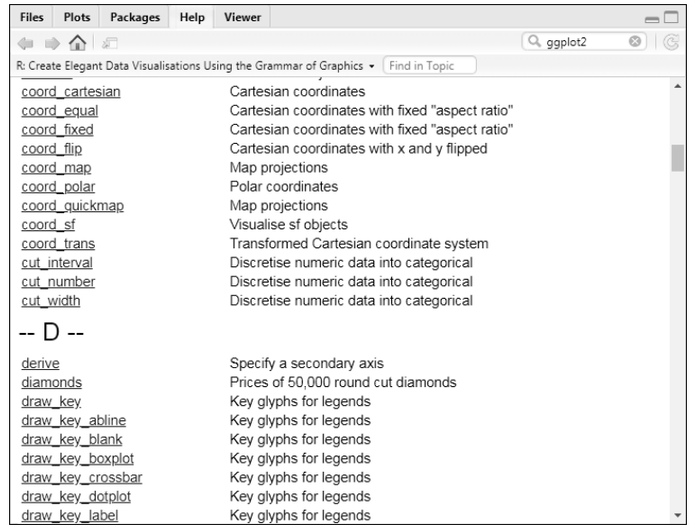

Abbildung 4.6 Diamonds-Datensatz im ggplot2-Paket

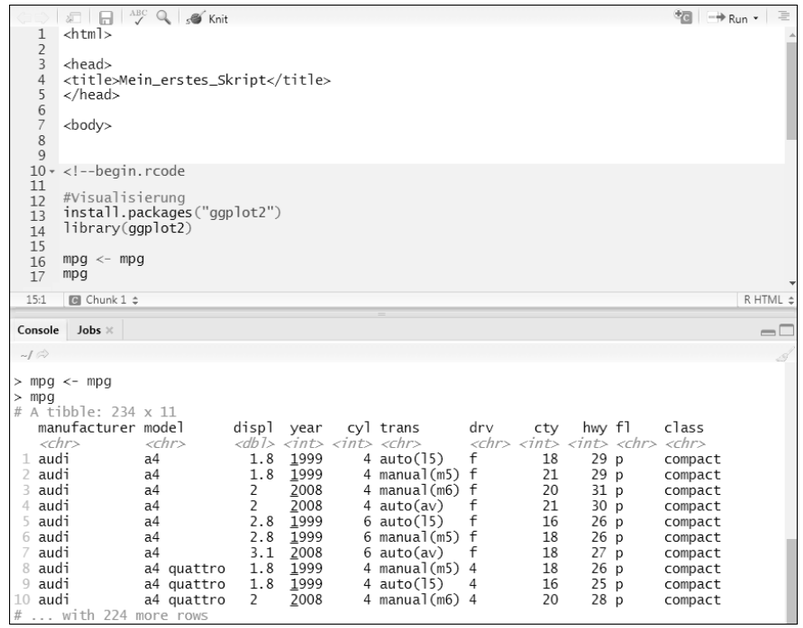

Abbildung 4.7 mpg-Datensatz im ggplot2-Paket

Im Skript sieht das wie in Abbildung 4.8 aus:

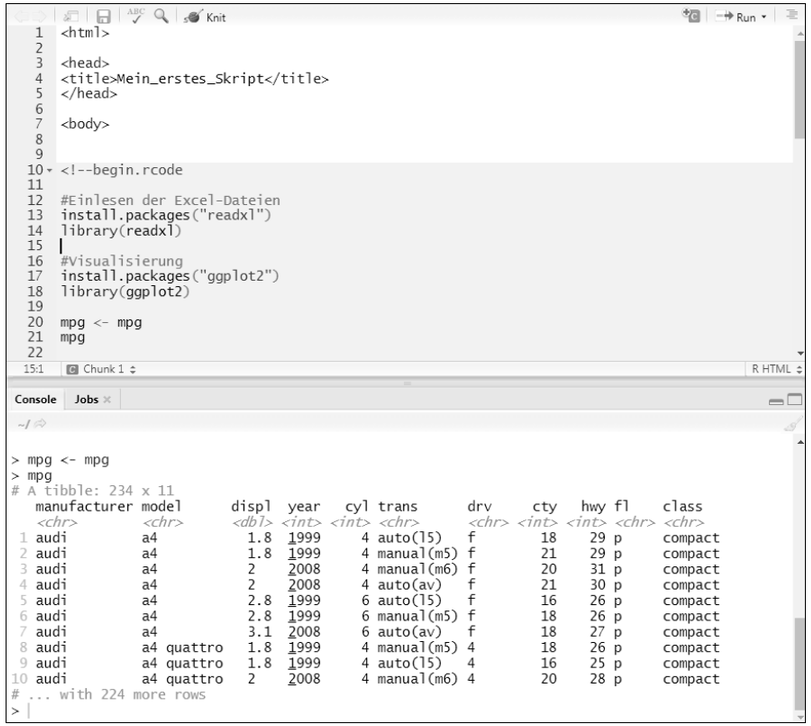

Abbildung 4.8 mpg-Dataset

In der Konsole zeigt Ihnen R die ersten zehn Zeilen des Datensatzes an. Sie können alternativ auch den Befehl `head(name_Ihres_Datensatzes)`, also `head(mpg)`, benutzen. Dass der mpg-Datensatz im *tibble*-Format vorliegt, das 234 Zeilen und 11 Spalten hat, beschreibt R Ihnen gleich in der ersten Zeile »# A tibble 234 x 11«.

Tibble-Datensatz

Ein Datensatz in der tibble-Form ist einfach ein Datensatz, der in einer gut auswertbaren Form (sowohl für R als auch grundsätzlich) daherkommt. Jede Zeile ist hierbei eine Observation, jede Spalte eine Variable. Tibbles sind mit den Daten, die wir hier verwenden, mit der alternativen R-Datensatzform data.table gleichzusetzen. Wir werden in diesem Buch mit der tibble-Datensatzform arbeiten.

Kapitel 6

Daten einlesen und für die Analyse vorbereiten

In diesem Kapitel zeige ich Ihnen, wie Sie den eigentlich wichtigsten Schritt der Datenanalyse mit R gehen können, nämlich das Daten-einlesen. Ich zeige Ihnen hierfür auch ein paar einfache Tricks.

Das (korrekte!) Einlesen der Daten in das Programm, mit dem Sie arbeiten, ist der wichtigste aller Schritte. Ohne Daten gibt es keine Analyse. Dieses Buch beschränkt sich auf den praktischen Umgang mit R. Deswegen ist es von Vorteil, die Daten so zu öffnen, dass Sie sie am besten gleich verwenden können. Ich würde Ihnen deshalb empfehlen, die Daten vor dem Einlesen vorzubereiten – natürlich in einer Kopie der Daten, nicht im Original! Später zeige ich Ihnen natürlich trotzdem, wie Sie Daten auch in R modifizieren können.

Sie können mit R grundsätzlich mittlerweile eine Reihe von Dateiformaten einlesen: Excel-Dateien (*.xlsx*), Textdateien (*.csv*, *.txt*), Shape-Dateien (*.shp*), SPSS-Dateien (*.sav*) etc. Wir beschäftigen uns vor dem Hintergrund der Praktikabilität nur mit dem Einlesen von *.xlsx*- (Excel) und *.csv*-Dateiformaten – den mit Abstand häufigsten Formaten, in denen Daten gesammelt und geliefert werden. RStudio bietet für das Einlesen dieser Dateiformen eine *Einlesehilfe* an, die Ihnen das Einlesen von Dateien erleichtert. Wir gehen die Möglichkeiten dazu nun im Folgenden einzeln durch.

6.1 Daten aus Excel einlesen

Excel-Dateien treten mit zwei Dateiendungen auf: *.xls* und *.xlsx*. Die Dateiendung *.xls* weist auf eine alte Excel-Datei-Version hin, die bereits ab 2007 durch *.xlsx* ersetzt wurde. Sie sollten also grundsätzlich den Nachfolger *.xlsx* verwenden. Sie können die alten Dateien auch einfach in einer neuen Excel-Datei öffnen und im neuen *.xlsx*-Format abspeichern.

Starten Sie zum Einlesen Ihrer Excel-Daten mit Insert Chunk, und kreieren Sie ein neues `Codestück`. Bewegen Sie dann die Maus in Richtung Arbeitsumgebung (Speicher),

und klicken Sie auf IMPORT DATASET. Hier können Sie nun das Dateiformat auswählen, in dem Ihre Daten gespeichert sind. Das ist im Fall des Zigarettendatensatzes eine Excel-Datei im Format *.xlsx*. Wir importieren also FROM EXCEL.

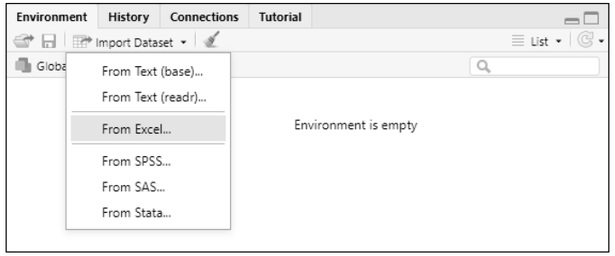

Abbildung 6.1 Datenimport einer Excel-Datei, Schritt 1

Es öffnet sich ein Fenster, in dem oben rechts in der Ecke die Option BROWSE angegeben ist. Klicken Sie zum Suchen und Finden Ihrer Datei darauf. Mit einem Doppelklick auf Ihre Datei, die Sie einlesen wollen, öffnet R Ihnen die ausgewählte Datei in diesem Fenster.

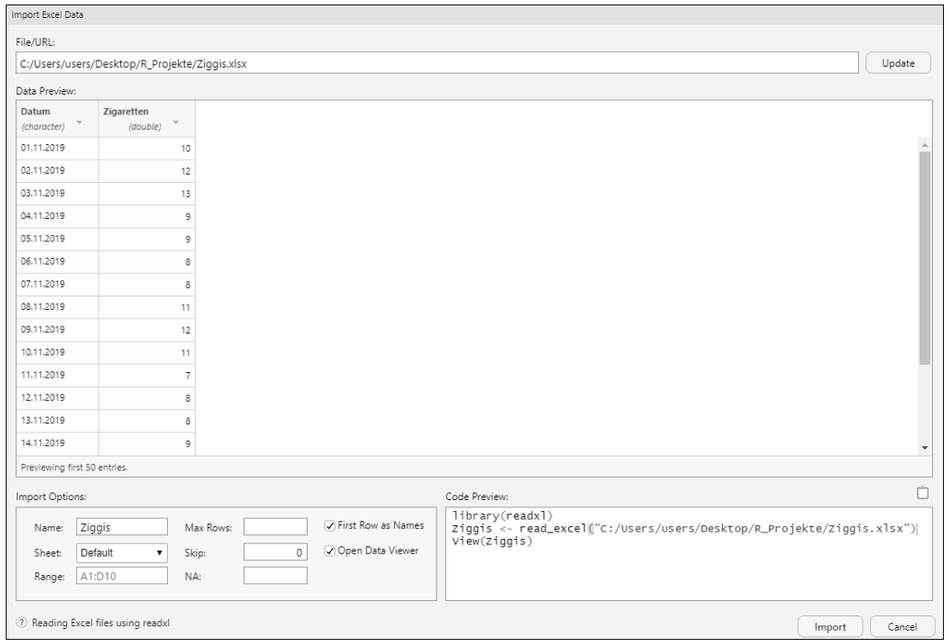

Abbildung 6.2 Datenimport einer Excel-Datei, Schritt 2

Die Hilfe, die R Ihnen hier bietet, liegt in dem Code, den R für Sie unten rechts im Fenster geschrieben hat. Das ist der Code, mit dem Sie die Datei auch im Skript einlesen würden. Diesen im folgenden Bild markierten Code können Sie nun ganz einfach per Rechtsklick auf die markierten Zeilen kopieren, um ihn dann in Ihr Skript einzufügen. Wie Sie sehen, lädt R sogar das Package (`readxl`), das benötigt wird, um Excel-Dateien einzulesen, für Sie gleich mit.

Denken Sie bitte daran, dass Sie das nötige Paket `readxl` erst einmal installieren müssen, wenn Sie das nicht schon gemacht haben. Nur damit kann das Paket auch geladen und die Funktion ausgeführt werden.

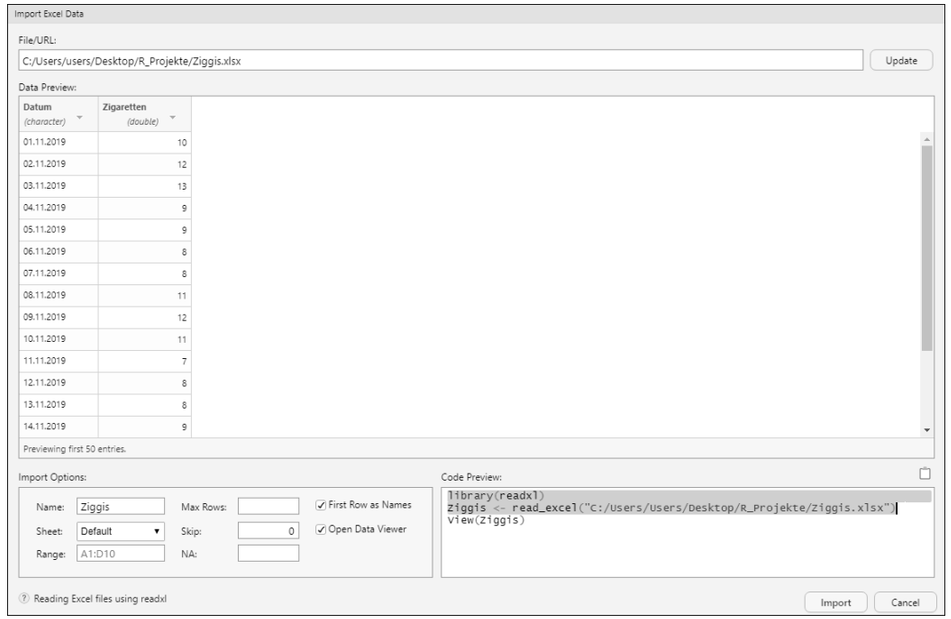

Abbildung 6.3 Datenimport einer Excel-Datei, Schritt 3

Tipp

Wenn Sie direkt unter den Variablennamen schauen, sehen Sie, dass Sie per Klick auf das Dreieck, das nach unten zeigt, das Format der Variable verändern können. Wenn Sie das wollen, z. B. um Speicherplatz zu sparen, um eine `Numeric`-Variable in eine `Integer`-Variable zu verändern, können Sie das hier tun. R schreibt den veränderten Code unten rechts neu und fügt Ihre Änderungen dort ein. Probieren Sie es gern aus.

Kopieren Sie den Code per Rechtsklick und COPY, klicken Sie dann auf CANCEL, und fügen Sie den Code per Rechtsklick und PASTE in Ihr Skript ein. Natürlich funktioniert hier auch das klassische Copy (Strg + C) and Paste (Strg + V).

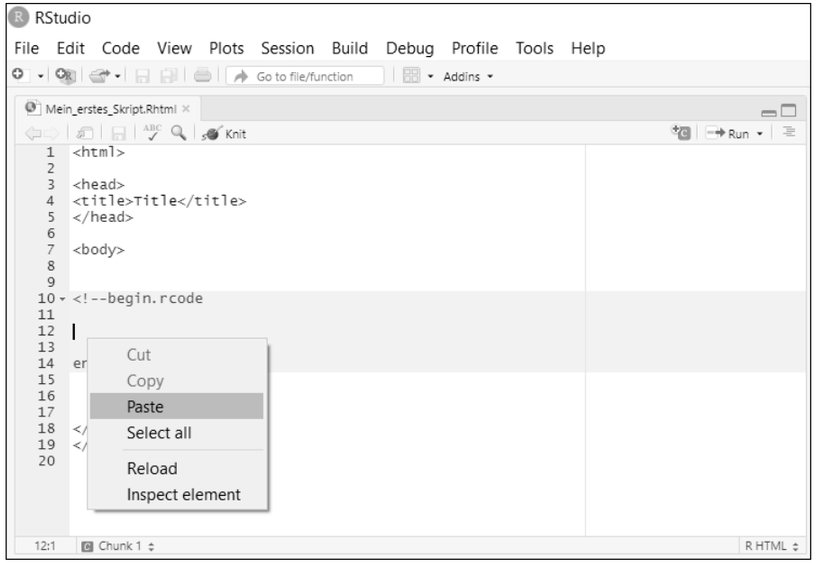

Abbildung 6.4 Datenimport einer Excel-Datei, Schritt 4

Nun lassen Sie den Code laufen. Oben links in der ARBEITSUMGEBUNG sehen Sie dann unter DATA das neue Objekt ZIGGIS, das R für Sie kreiert hat. Darin befinden sich Ihre Daten.

Abbildung 6.5 Datensatz Zigaretten – eingelesen und geladen

Mit einem Klick auf das Objekt ZIGGIS öffnen Sie den Viewer.

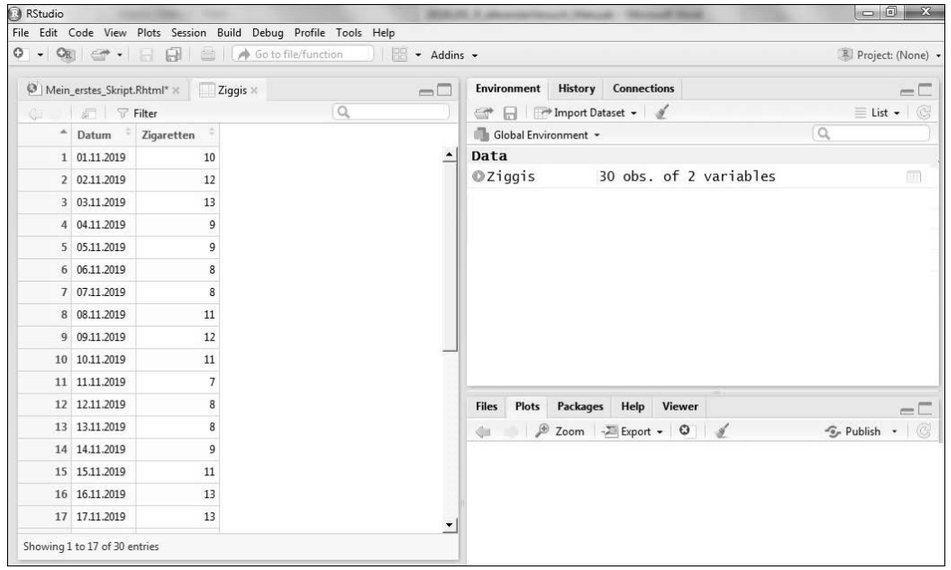

Abbildung 6.6 Daten ansehen

Ihr Skriptstück für das Einlesen sollte nun so aussehen:

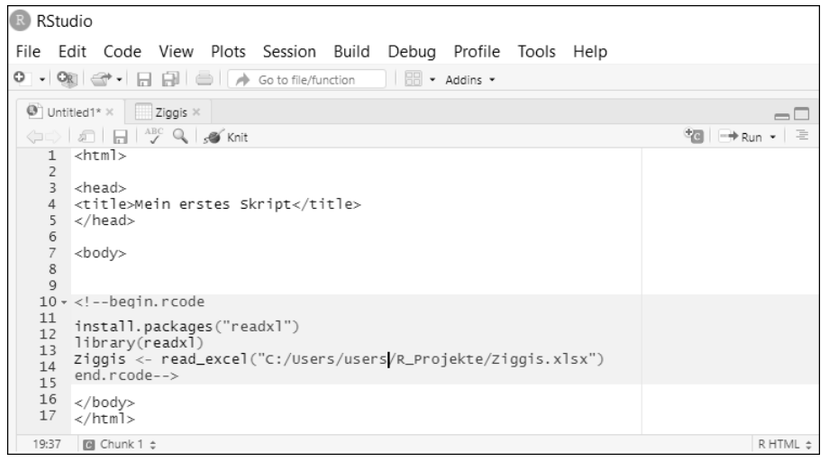

Abbildung 6.7 Das Skript nach dem Einlesen der Excel-Datei

Einzig der Dateipfad *Users/users* sollte sich unterscheiden. Hier müsste Ihrer stehen.

Die deutschen Datumsangaben verursachen hier regelmäßig Probleme. Mal werden sie im falschen Format eingelesen, mal gar nicht. Wie Sie damit umgehen, zeige ich Ihnen in Abschnitt 6.3, »Umgang mit Datumsangaben«.

6.2 Daten im .csv-Format einlesen

Das Einlesen von *.csv*-Dateiformaten (*.csv*-Files) funktioniert beinahe genauso wie das Einlesen von Excel-Dateien. Als Beispiel nehmen wir die oben bereits heruntergeladenen Zensusdaten, die wir aber dieses Mal als *.csv*-Datei herunterladen und abspeichern (*http://r-wrk.de/ZensusDatei*).

6.2.1 Einlesen von .csv-Dateien mit RStudio

Zum Einlesen von *.csv*-Dateien mit RStudio gehen Sie zunächst auf Import Dataset. Dieses Mal wählen Sie From Text (readr) aus, denn *.csv*-(comma-separated values-) Dateien sind eigentlich `Textdateien` (deswegen können Sie diese z. B. auch in einem beliebigen Text-Editor öffnen). Einen Teil der Daten finden Sie im Abschnitt A.2, »Zensus Länder«.

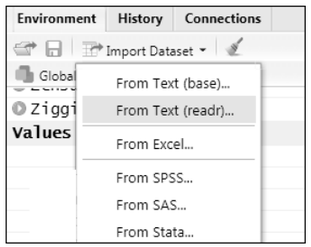

Abbildung 6.8 Importieren einer .csv-Datei, Schritt 1

Dann suchen Sie per Klick auf Browse Ihre Datei, die Sie öffnen wollen.

Jetzt zeigt Ihnen R wieder den ausgewählten Datensatz (bzw. einen Ausschnitt davon) geöffnet an. Oder nicht?

> **Achtung**
>
> Wenn Sie nichts sehen, kann es sein, dass die Texteinträge durch einen anderen Trenner als das Komma ("Comma") voneinander abgegrenzt sind! Das kann ein Semikolon sein oder ein Leerzeichen ("Whitespace"). Wenn das so ist, dann müssen Sie dies R unter Import Options mitteilen.

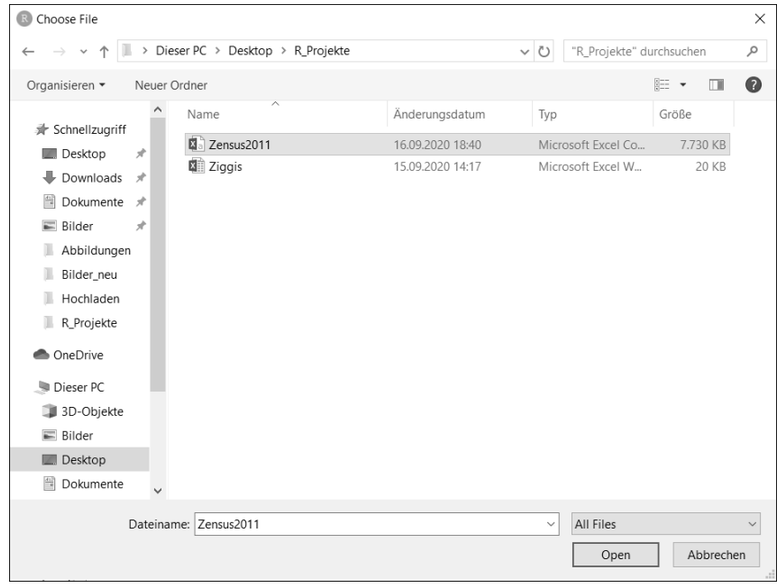

Abbildung 6.9 Importieren einer .csv-Datei, Schritt 2

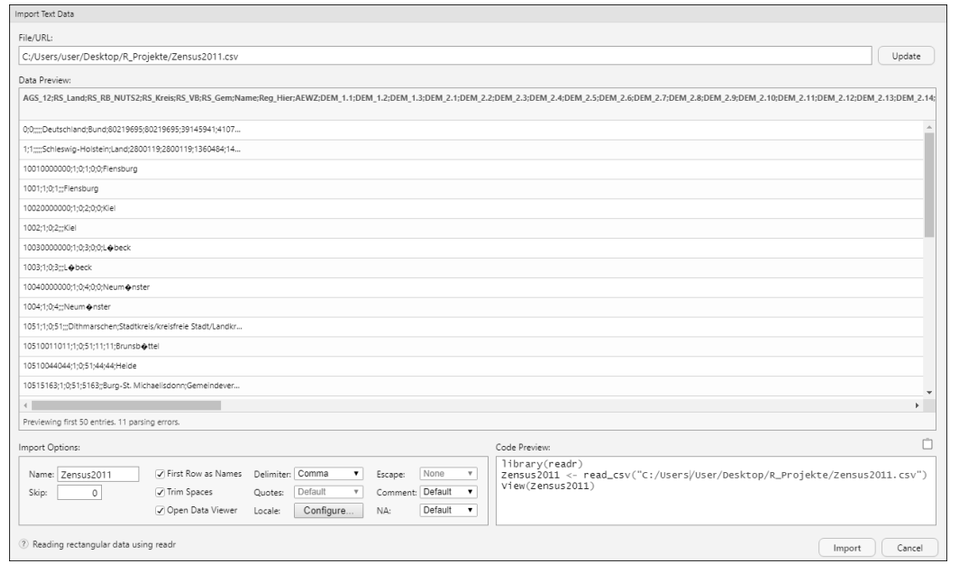

Abbildung 6.10 Importieren einer .csv-Datei, Fehlversuch

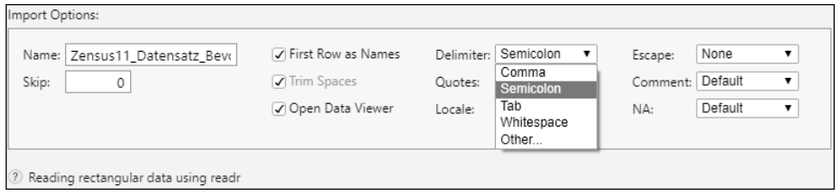

Abbildung 6.11 Import-Option anpassen

R fügt das dann automatisch dem Befehl hinzu, den Sie anschließend ins Skript kopieren. Unten rechts befindet sich der Code, den Sie nun wieder in Ihr Skript kopieren können.

Denken Sie bitte wieder daran, dass Sie das Paket `readr` installiert haben müssen, wenn der Code laufen soll.

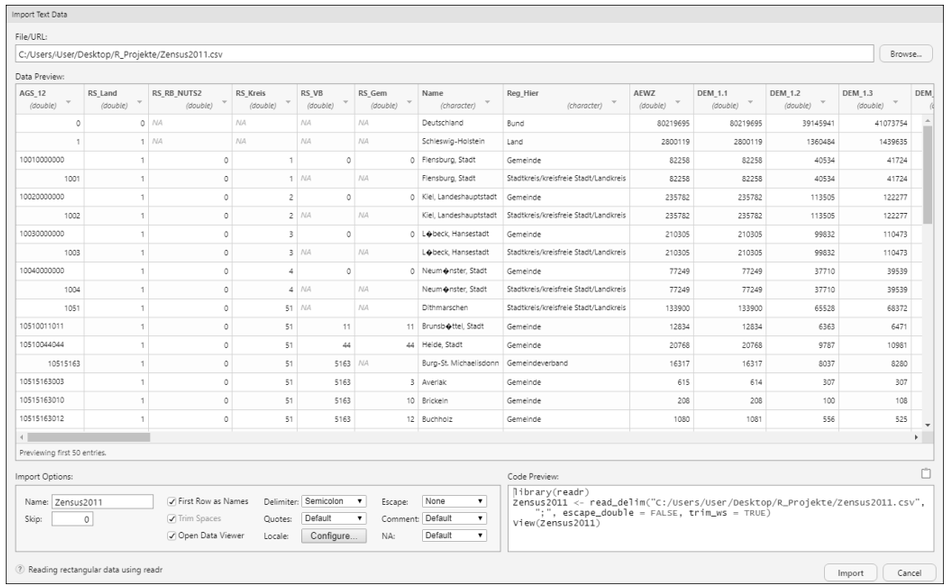

Abbildung 6.12 Importieren einer .csv-Datei, Schritt 3

Machen Sie von hier aus einfach weiter wie bei der Excel-Datei:

Geben Sie per Copy and Paste hinein ins Skript, und führen Sie den Befehl aus.