

R und RStudio auf Ihren Computer holen

In eine R-Session eintauchen

Mit R-Funktionen arbeiten

Mit R-Strukturen arbeiten

Kapitel 1

R: Was R kann und wie R das macht

Sie sind also bereit, die wunderbare Welt von R zu bereisen? R ist von und für Statistiker und Datenwissenschaftler entwickelt und besitzt eine kurze, jedoch glanzvolle Geschichte.

In den 1990er Jahren entwickelten Ross Ihaka und Robert Gentleman R an der University of Auckland, Neuseeland. R wird vom gemeinnützigen Verein »The R Foundation for Statistical Computing« unterstützt und erfreut sich mit jedem Tag größerer Beliebtheit.

R herunterladen

Falls R noch nicht auf Ihrem Computer installiert ist, müssen Sie als Allererstes R herunterladen und installieren.

Sie finden die geeignete Software auf der Website des Comprehensive R Archive Network (CRAN). Falls Sie Windows verwenden, geben in Ihrem Browser diese Adresse ein:

`cran.r-project.org/bin/windows/base`

Falls Sie auf dem Mac arbeiten, ist dies die Downloadseite:

`cran.r-project.org/bin/macosx`

Klicken Sie den Link an, um R herunterzuladen. Hierdurch wird eine ausführbare `.exe`-Datei auf Ihren Windows-PC heruntergeladen beziehungsweise eine `.pkg`-Datei auf Ihren Mac. Führen Sie in beiden Fällen die gewohnten Schritte zur Installation eines Programms

durch. Wenn die Installation fertiggestellt wurde, sehen Windows-Anwender auf ihrem Desktop zwei R-Programmsymbole, eines für 32-Bit-Betriebssysteme und eines für 64-Bit-Betriebssysteme (verwenden Sie das Symbol für Ihre Windows-Variante). Mac-Benutzer finden R im Ordner *Programme*.



Beide URLs enthalten auch nützliche Links zu »Häufig gestellten Fragen, FAQs«. Auf der Downloadseite der Windows-Version finden Sie auch einen Link zu »Installation and other instructions«.

RStudio herunterladen

Die Verwendung von R ist sehr viel einfacher, wenn Sie hierfür eine Anwendung mit dem Namen *RStudio* verwenden. Computerfreaks nennen diese Art von Programm Entwicklungsumgebung oder kurz IDE für den englischen Fachbegriff *Integrated Development Environment*. Hierbei handelt es sich um ein Werkzeug, das Sie beim Erstellen, Bearbeiten, Ausführen und Nachverfolgen Ihres R-Codes unterstützt. Gleichzeitig bietet die Entwicklungsumgebung Zugriff auf hilfreiche Tipps über R.

Hier ist die Webadresse dieses fantastischen Tools:

www.rstudio.com/products/rstudio/download

Klicken Sie den Link des Installationsprogramms für Ihr Betriebssystem – Windows, Mac oder eine der Linux-Varianten – an und führen Sie die üblichen Schritte zur Programminstallation durch.



In diesem Buch verwende ich die R-Version 3.4.3 und die RStudio-Version 1.1.419. Zu dem Zeitpunkt, zu dem Sie diese Zeilen lesen, stehen möglicherweise neuere Versionen zur Verfügung.

Nachdem die Installation abgeschlossen ist, klicken Sie das nagelneue Programmsymbol von RStudio an, damit Sie das Fenster in Abbildung 1.1 sehen.

In der großen Konsole auf der linken Seite wird der R-Code ausgeführt. Eine Möglichkeit, um R-Code auszuführen, besteht darin, ihn direkt in die Konsole einzugeben. Eine andere Variante zeige ich Ihnen in Kürze.

In den beiden Bereichen auf der rechten Seite des Fensters finden Sie hilfreiche Informationen, während Sie mit R arbeiten. Oben rechts finden Sie den Arbeitsbereich *ENVIRONMENT* (Umgebung) und *HISTORY* (Historie). Auf der Registerkarte *ENVIRONMENT* werden die Dinge mitgehalten, die Sie beim Arbeiten mit R erstellen. (Diese werden in R Objekte genannt.) Auf der Registerkarte *HISTORY* wird der R-Code mitgehalten, den Sie eingeben.



Gewöhnen Sie sich schon mal an das Wort *Objekt*. In R ist alles ein Objekt.

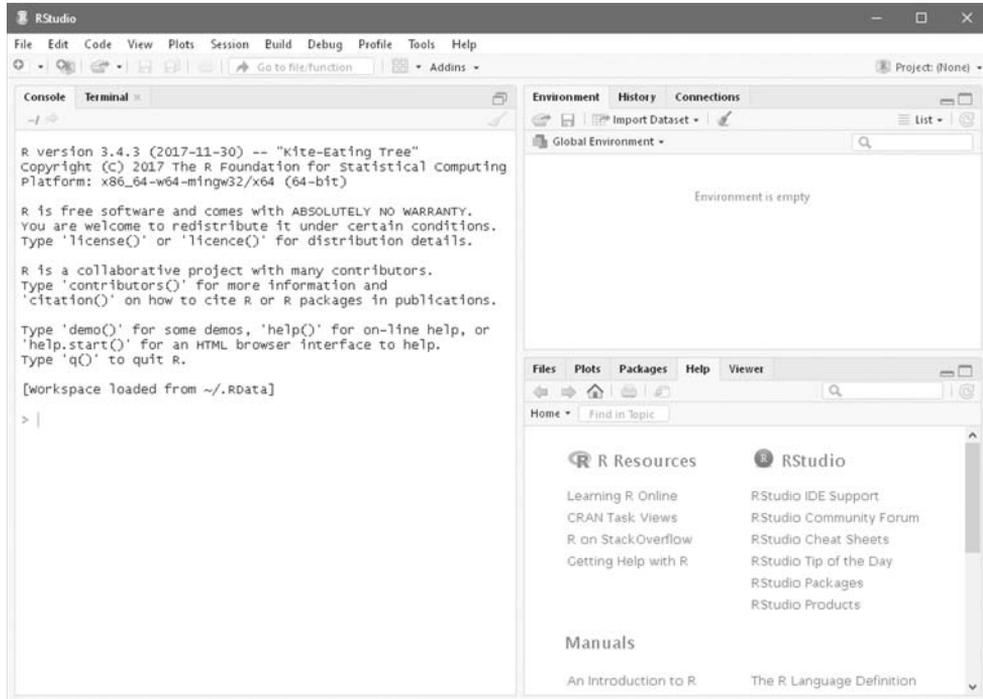


Abbildung 1.1: RStudio, wie es beim ersten Start nach der Installation aussieht

Im Panel unten rechts finden Sie die Registerkarten FILES, PLOTS, PACKAGES und HELP. Auf der Registerkarte FILES werden die von Ihnen erstellten Dateien angezeigt. Die Registerkarte PLOTS enthält die Diagramme, die Sie mit Ihren Daten erstellt haben. Auf der Registerkarte PACKAGES sehen Sie die Add-Ons (die in R Packages, Pakete, genannt werden), die Sie zusammen mit R heruntergeladen haben. Beachten Sie, dass *heruntergeladen* nicht bedeutet, dass die Packages bereits genutzt werden können. Hierfür ist ein weiterer Schritt erforderlich. Und glauben Sie mir, Sie wollen Packages verwenden.

Die Registerkarte PACKAGES sehen Sie in Abbildung 1.2. Packages stelle ich weiter hinten in diesem Kapitel vor.

Die Registerkarte HELP, die Sie in Abbildung 1.3 sehen, enthält Links zu einer Fülle von englischsprachigen Informationen über R und RStudio.

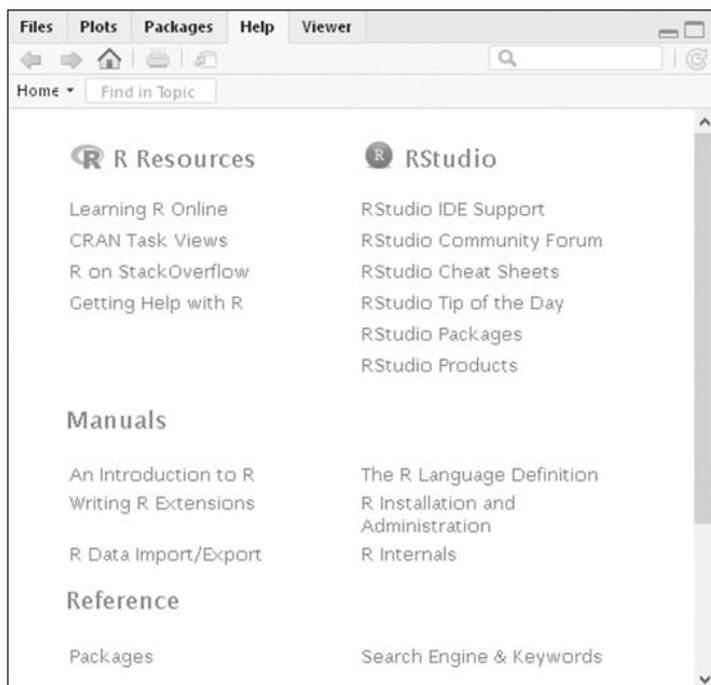
Um alle Möglichkeiten der grafischen Entwicklungsumgebung RStudio zu nutzen, klicken Sie auf das größere der beiden Symbole, die Sie in der oberen rechten Ecke der Konsole sehen. Das Aussehen von RStudio ändert wie in Abbildung 1.4 gezeigt.

Das Konsolenfenster wird verkleinert und befindet sich nun links unten. Darüber sehen Sie ein neues Fenster, das Skriptfenster. Das Skriptfenster ist der Editor für Ihren R-Code. Sie können dort Code eingeben und bearbeiten und dann `Strg + ↵` drücken, um den Code im Konsolenfenster ausführen zu lassen. Auf dem Mac drücken Sie `⌘ + ↵`.



Name	Description	Version
System Library		
<input type="checkbox"/> boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-20
<input type="checkbox"/> class	Functions for Classification	7.3-14
<input type="checkbox"/> cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.6
<input type="checkbox"/> codetools	Code Analysis Tools for R	0.2-15
<input type="checkbox"/> compiler	The R Compiler Package	3.4.3
<input type="checkbox"/> datasets	The R Datasets Package	3.4.3
<input type="checkbox"/> foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-69
<input type="checkbox"/> graphics	The R Graphics Package	3.4.3
<input type="checkbox"/> grDevices	The R Graphics Devices and Support for Colours and Fonts	3.4.3
<input type="checkbox"/> grid	The Grid Graphics Package	3.4.3
<input type="checkbox"/> KernSmooth	Functions for Kernel Smoothing Supporting Wand & Jones (1995)	2.23-15
<input type="checkbox"/> lattice	Trellis Graphics for R	0.20-35
<input type="checkbox"/> MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-47
<input type="checkbox"/> Matrix	Sparse and Dense Matrix Classes and Methods	1.2-12

Abbildung 1.2: Die Registerkarte PACKAGES in RStudio



Home	Find in Topic
<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>R Resources</p> <ul style="list-style-type: none"> Learning R Online CRAN Task Views R on StackOverflow Getting Help with R </div> <div style="width: 45%;"> <p>RStudio</p> <ul style="list-style-type: none"> RStudio IDE Support RStudio Community Forum RStudio Cheat Sheets RStudio Tip of the Day RStudio Packages RStudio Products </div> </div>	
<p>Manuals</p> <ul style="list-style-type: none"> An Introduction to R Writing R Extensions R Data Import/Export The R Language Definition R Installation and Administration R Internals 	
<p>Reference</p> <ul style="list-style-type: none"> Packages Search Engine & Keywords 	

Abbildung 1.3: Die Registerkarte HELP in RStudio



Sie können auch im Skriptfenster die gewünschten Codezeilen markieren und dann im Menü von RStudio den Befehl **CODE | RUN SELECTED LINE(S)** verwenden.

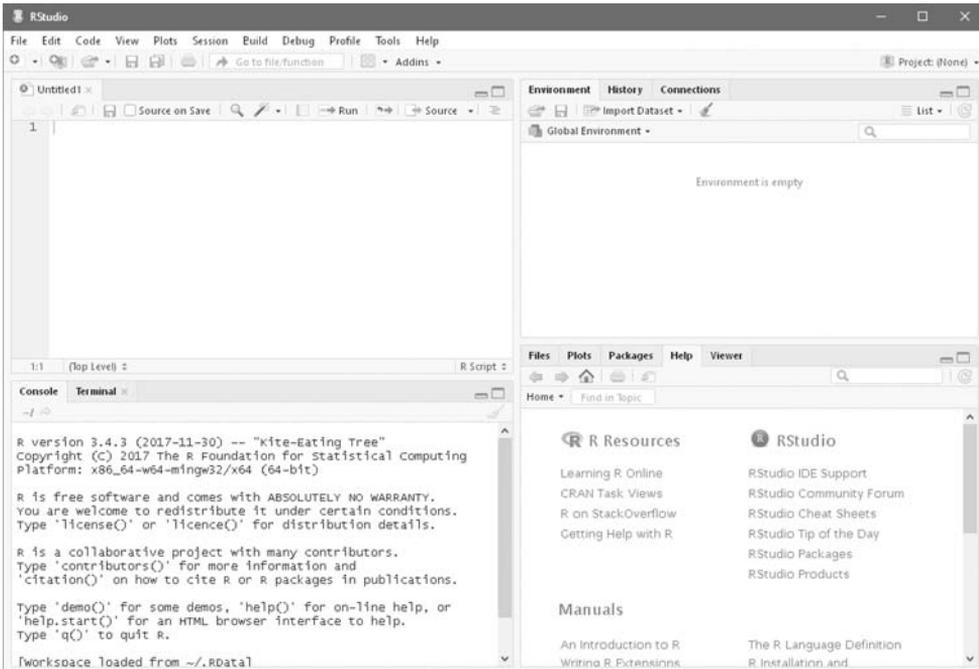


Abbildung 1.4: RStudio, nachdem Sie an der rechten Seite der Titelleiste der Konsole das große Symbol angeklickt haben

Eine Session mit R

Bevor Sie loslegen, wählen Sie den Befehl **FILE | SAVE AS** und verwenden Sie als Dateiname **Meine erste R-Session**. Die Registerkarte im Skriptfenster wird umbenannt und Sie sehen dort den eingegebenen Dateinamen mit der Erweiterung **.R**. Außerdem wird der Dateiname (ebenfalls mit der Erweiterung **.R**) auf der Registerkarte **FILES** angezeigt.

Das Arbeitsverzeichnis

Was genau speichert R und wo wird gespeichert? R speichert den sogenannten **Workspace**; das ist die Umgebung, in der Sie arbeiten. Der **Workspace** wird von R im **Arbeitsverzeichnis** gespeichert. In Windows ist dies das Standardarbeitsverzeichnis:

C:\Benutzer\\Dokumente

Auf dem Mac ist es der folgende Ordner:

/Benutzer/<Benutzername>

Falls Sie jemals den Pfad zu Ihrem Arbeitsverzeichnis vergessen sollten, geben Sie in der Konsole den folgenden Befehl ein. R zeigt dann das Arbeitsverzeichnis auf dem Bildschirm an:

```
> getwd()
```



In der Konsole brauchen Sie das Größerzeichen nicht einzugeben. Es befindet sich bereits am Anfang der Zeile und stellt die Eingabeaufforderung dar.

Mein Arbeitsverzeichnis sieht folgendermaßen aus:

```
> getwd()
[1] "C:/Users/Joseph Schmueller/Documents/R"
```



Der Befehl `getwd()` zeigt den englischen Ordernamen an. In den deutschsprachigen Windows-Versionen kümmert sich das Betriebssystem darum, dass der Ordnername als Benutzer angezeigt wird. Die Pfadangaben `C:\Users` und `C:\Benutzer` verweisen also auf den gleichen Ordner.

Beachten Sie, dass in der Ausgabe von `getwd()` die normalen Schrägstriche und keine Backslashes verwendet werden. Dies ist anders, als Sie es von Windows her gewohnt sind. Der Grund ist, dass R den Backslash (`\`) als Escape-Zeichen verwendet. Zeichen, die nach dem Escape-Zeichen eingegeben werden, erhalten so eine andere Bedeutung. So entspricht `\t` in R dem Tabulatorzeichen.



Wenn Sie lieber den Backslash verwenden, müssen Sie ihn für Pfadnamen in R doppelt eingeben, beispielsweise `C:\\Benutzer\\<Benutzername>\\Dokumente`.

Wenn Sie möchten, können Sie das Arbeitsverzeichnis mit diesem Befehl ändern:

```
> setwd(<Dateipfad>)
```

Eine andere Möglichkeit zum Ändern des Arbeitsverzeichnisses ist der Befehl `SESSION | SET WORKING DIRECTORY | CHOOSE DIRECTORY`.

Jetzt geht es richtig los

Jetzt kümmern wir uns um R. Geben Sie Folgendes in das Skriptfenster ein:

```
x <- c(5,10,15,20,25,30,35,40)
```

Drücken Sie dann `Strg` + `↵`.

Hierdurch wird die folgende Zeile in das Konsolenfenster eingefügt:

```
> x <- c(5,10,15,20,25,30,35,40)
```

Wie ich bereits in einem Tipp weiter vorne erwähnte, ist das Größerzeichen, das Sie in der Konsole sehen, der Prompt, die Eingabeaufforderung in R. Im Skriptfenster ist es nicht zu sehen.

Was hat R gerade gemacht? Die Zeichen `<-` geben an, dass `x` das zugewiesen wird, was auf der rechten Seite von `<-` steht. Sie können sich die Zeichen `<-` als Zuweisungsoperator von R vorstellen. Die Werte 5, 10, 15, 20 ... 40 sind also nun `x` zugewiesen.



In R wird ein Satz von Zahlen *Vektor* genannt. Ich komme hierauf im Abschnitt »R-Strukturen« weiter hinten in diesem Kapitel zurück.

Sie können diese Zeile folgendermaßen lesen: »`x` wird der Vektor 5, 10, 15, 20 usw. zugewiesen«.

Geben Sie im Skriptbereich `x` ein und drücken Sie `Strg` + `↵`. In der Konsole sehen Sie dann die folgende Ausgabe:

```
>x
[1]  5 10 15 20 25 30 35 40
```

Die 1 in den eckigen Klammern ist das Label für den ersten Wert in der Ausgabezeile, in unserem Beispiel also der Wert 5.

In diesem Beispiel gibt es lediglich eine Zeile mit Werten. Was passiert, wenn R mehrere Werte in mehreren Zeilen ausgibt? Jede Zeile beginnt mit einem numerischen Label, das von eckigen Klammern eingeschlossen ist. Die Zahl in den Klammern korrespondiert mit dem Index des ersten Werts der Zeile. Wenn die Ausgabe beispielsweise aus 23 Werten besteht und der 18te Wert der erste Wert in der zweiten Zeile ist, dann beginnt die zweite Zeile mit `[18]`.

Durch die Erstellung des Vektors ändert sich auch der Inhalt der Registerkarte ENVIRONMENT, wie es Abbildung 1.5 zeigt.

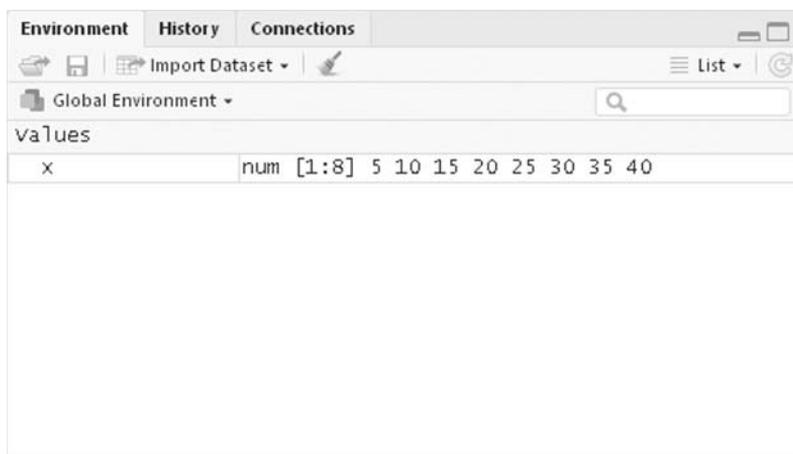


Abbildung 1.5: Die Registerkarte ENVIRONMENT nach der Erzeugung des Vektors `x`



Eine andere Möglichkeit, um sich die Objekte im Environment anzusehen, besteht darin, in das Skriptfenster `ls()` einzugeben und dann `Strg` + `↵` zu drücken. Sie können `ls()` auch direkt in das Konsolenfenster eingeben und `↵` drücken. In beiden Fällen wird im Konsolenfenster folgendes Ergebnis angezeigt:

```
[1] "x"
```

Nun können Sie mit `x` arbeiten. Addieren Sie zuerst alle Zahlen im Vektor. Geben Sie

```
sum(x)
```

im Skriptfenster ein (denken Sie daran, `Strg` + `↵` zu drücken). Hierdurch wird in der Konsole diese Zeile ausgeführt:

```
> sum(x)
[1] 180
```

Wie wäre es mit dem Mittelwert der Zahlen im Vektor `x`? Tippen Sie

```
mean(x)
```

im Skriptfenster ein, drücken Sie `Strg` + `↵` und Sie erhalten in der Konsole diese Ausgabe:

```
> mean(x)
[1] 22.5
```



Während Sie im Skriptfenster oder in der Konsole Anweisungen eingeben, werden hilfreiche Popupfenster angezeigt. Wenn Sie mehr Erfahrung mit RStudio haben, werden Sie lernen, wie Sie diese Informationen nutzen können.

Die Varianz gibt an, wie sehr sich eine Gruppe von Zahlen von deren Mittelwert unterscheidet. Um mit R die Varianz zu berechnen, verwenden Sie die folgende Anweisung:

```
> var(x)
[1] 150
```

Was genau ist Varianz und was bedeutet es? (Achtung: Es folgt ein schamloser Werbehinweis.) Antworten auf diese und viele weitere Fragen über Statistik und Analyse finden Sie in dem Klassiker *Statistik mit R für Dummies* (der ebenfalls von mir verfasst und bei Wiley-VCH veröffentlicht wurde).

Nachdem R all diese Befehle ausgeführt hat, sieht die Registerkarte HISTORY so aus wie in Abbildung 1.6.

Um eine Session zu beenden, wählen Sie FILE | QUIT SESSION oder drücken Sie `Strg` + `Q`. Wie in Abbildung 1.7 zu sehen, wird ein Dialogfeld angezeigt und Sie werden gefragt, welche Elemente der Session gespeichert werden sollen. Wenn Sie die Auswahl unverändert lassen und auf SAVE SELECTED klicken, können Sie beim nächsten Start von RStudio die Session erneut öffnen und an der Stelle weitermachen, an der Sie aufgehört haben. (Beachten Sie, dass die Inhalte der Konsole nicht gespeichert werden.)

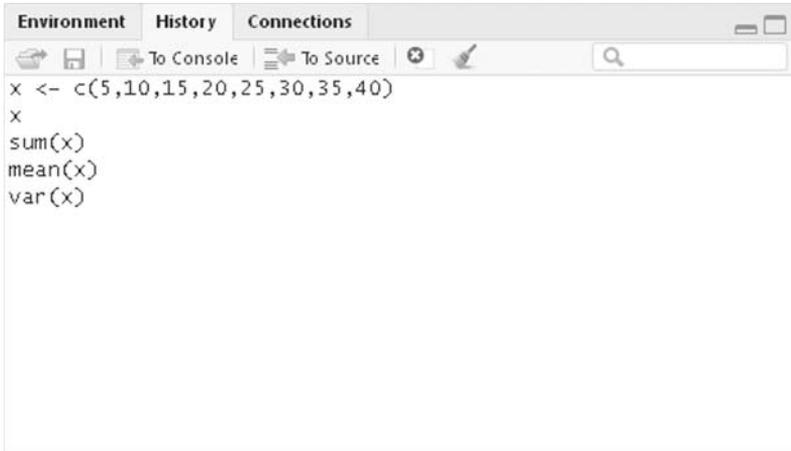


Abbildung 1.6: Die Registerkarte HISTORY nach der Erstellung des Vektors und dessen Verwendung

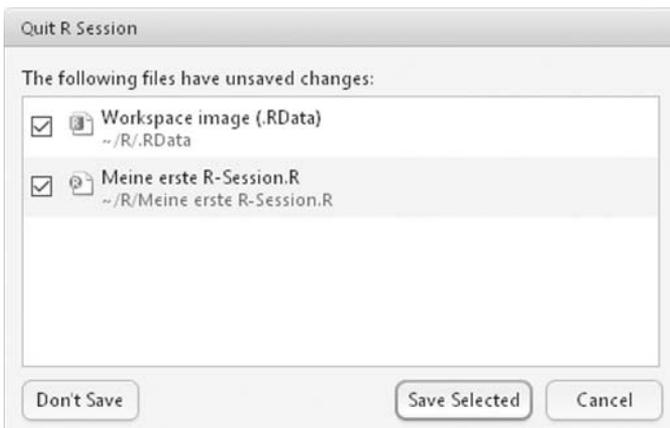


Abbildung 1.7: Das Dialogfeld QUIT R SESSION



Im weiteren Verlauf des Buches werde ich beim Vorstellen eines Beispiels nicht immer schreiben: »Geben Sie diesen R-Code in das Skriptfenster ein und drücken Sie `Strg` + `↵`.« Stattdessen zeige ich Ihnen wie im Beispiel mit `var()` einfach den Code und dessen Ausgabe.



Manchmal ist der Code mit dem Prompt `>` dargestellt und manchmal ohne ihn. Im Prinzip zeige ich Ihnen den Prompt immer dann, wenn ich Ihnen den R-Code mitsamt den Ergebnissen zeigen will. Falls ich lediglich Code darstellen möchte, den ich im Skriptbereich eingegeben habe, lasse ich den Prompt weg.

R-Funktionen

Im vorigen Abschnitt habe ich `c()`, `sum()`, `mean()` und `var()` verwendet. Dies sind Beispiele für die in R eingebauten Funktionen. Die Funktion besteht aus einem Namen, dem ein Paar runde Klammern folgt. Innerhalb der runden Klammern stehen die *Argumente*. In diesem Zusammenhang ist mit »Argument« nicht der Austausch über eine Meinungsverschiedenheit gemeint. Argument, manchmal auch *Parameter* genannt, ist der mathematische Ausdruck für die Daten, die eine Funktion verwendet.



Auch wenn eine Funktion keine Argumente verwendet (wie im Fall der Funktion `ls()`), müssen Sie die runden Klammern angeben.

Die Funktionen in den vorherigen Beispielen sind sehr einfach: Geben Sie ein Argument an und die Funktion liefert ein Ergebnis. Einige der R-Funktionen nehmen jedoch mehr als ein Argument entgegen.

R stellt für den Einsatz von Funktionen, die mehrere Argumente besitzen, verschiedene Varianten zur Verfügung. Eine Möglichkeit besteht darin, die Argumente in der gleichen Reihenfolge anzugeben, die bei der Definition der Funktion verwendet wurde. R nennt dies *Stellungsparameter* (positional matching).

Hier ein Beispiel dafür, was damit gemeint ist. Erinnern Sie sich noch daran, wie ich den Vektor `x` erstellte?

```
x <- c(5, 10, 15, 20, 25, 30, 35, 40)
```

Eine andere Möglichkeit, einen Vektor dieser Zahlen zu erstellen, ist die Verwendung der Funktion `seq()`:

```
> y <- seq(5, 40, 5)
> y
[1] 5 10 15 20 25 30 35 40
```

Mit der Funktion `seq()` können Sie eine Sequenz erstellen. Das erste Argument für `seq()` ist die Zahl, ab der die Sequenz beginnen soll; in unserem Beispiel die Zahl 5. Der Name dieses Arguments lautet *from*. Das zweite Argument ist die Zahl, bis zu der die Sequenz gehen soll; in unserem Beispiel die Zahl 40. Der Name dieses Arguments lautet *to*. Das dritte Argument gibt das Inkrement der Sequenz an, also den Wert, um den sich die Sequenz erhöht (5). Der Name dieses Arguments lautet *by*.

Wenn Sie beim Funktionsaufruf die Namen der Argumente angeben, spielt deren Reihenfolge keine Rolle:

```
> z <- seq(to=40, by=5, from=5)
> z
[1] 5 10 15 20 25 30 35 40
```

Wenn Sie also eine Funktion verwenden, können Sie deren Parameter in beliebiger Reihenfolge angeben, solange Sie die Parameter benennen. Diese Art der Parameterübergabe wird in R *Schlüsselwortparameter* (keyword matching) genannt. Dies ist nützlich, wenn Sie eine R-Funktion mit zahlreichen Argumenten verwenden wollen. Wenn Sie deren Reihenfolge nicht genau kennen, geben Sie einfach die Namen an und die Funktion funktioniert.



Falls Sie jemals Hilfe zu einer bestimmten Funktion benötigen, beispielsweise für `seq()`, geben Sie `?seq` ein und schauen Sie sich auf der Registerkarte `HELP` die nützlichen Informationen an. Während Sie `?seq` eingeben, werden neben dem eingegebenen Text in einem kleinen Popup-Fenster Kurzinfs zu der Funktion angezeigt.

Benutzerdefinierte Funktionen

R erlaubt es Ihnen, eigene Funktionen zu erstellen. Nachfolgend erläutere ich hierzu die Grundlagen.

Eine R-Funktion ist folgendermaßen aufgebaut:

```
meinefunktion <- function(argument1, argument2, ... ){
  Anweisungen
  return(object)
}
```

Hier ist eine Funktion für rechtwinklige Dreiecke. Erinnern Sie sich noch? Ein rechtwinkliges Dreieck hat zwei Seiten, die einen rechten Winkel bilden; die dritte Seite wird Hypotenuse genannt. Vielleicht erinnern Sie sich auch daran, dass ein Typ namens Pythagoras gezeigt hat, dass die Länge der Hypotenuse c folgendermaßen berechnet wird, wenn eine Seite die Länge a und die andere Seite die Länge b besitzt:

$$c = \sqrt{a^2 + b^2}$$

Hier ist eine einfache Funktion mit dem Namen `hypotenuse()`, der zwei Zahlen übergeben werden (die Länge der beiden Seiten des rechtwinkligen Dreiecks) und die c , die Länge der Hypotenuse, zurückgibt:

```
hypotenuse <- function(a,b){
  hyp <- sqrt(a^2+b^2)
  return(hyp)
}
```

Geben Sie diesen Code in das Skriptfenster ein, markieren Sie ihn und drücken Sie `Strg` + `↵`. In der Konsole erscheint die folgende Ausgabe:

```
> hypotenuse <- function(a,b){
+   hyp <- sqrt(a^2+b^2)
+   return(hyp)
+ }
```

Das Pluszeichen ist das Zeilenfortsetzungszeichen. Es gibt an, dass die aktuelle Zeile von der vorherigen Zeile fortgesetzt wird.

Und so können Sie diese Funktion verwenden:

```
> hypotenuse(3,4)
[1] 5
```



Wie Sie zu Recht vermuten, können Sie mit R-Funktionen viel mehr tun als das, was ich kurz angerissen habe. Wenn Sie an weiterführenden Informationen interessiert sind, sollten Sie einen Blick in das Buch *R für Dummies* von Andrie de Vries und Joris Meys werfen.

Kommentare

Kommentare sind Anmerkungen für Ihren Code. Leiten Sie einen Kommentar mit dem Raute-Zeichen ein (#), heute eher als Hashtag bekannt. R ignoriert alles, was rechts vom #, dem Kommentarzeichen, steht.

Kommentare sind für diejenigen nützlich, die den Code, den Sie geschrieben haben, lesen und verstehen wollen. Hier ein Beispiel:

```
hypotenuse <- function(a,b){ # Liste der Argumente
  hyp <- sqrt(a^2+b^2)      # Berechnung durchführen
  return(hyp)              # Den Wert zurückgeben
}
```

Nur zur Info: Im Beispielcode dieses Buchs verwende ich keine Kommentare. Stattdessen stelle ich detaillierte Beschreibungen zur Verfügung. Ich glaube, dass das in einem Buch wie diesem der beste Weg ist, das rüberzubringen, was ich vermitteln möchte.

R-Strukturen

Im Abschnitt »R-Funktionen« weiter vorne in diesem Kapitel habe ich erwähnt, dass eine R-Funktion viele Argumente besitzen kann. Außerdem kann eine R-Funktion auch mehrere Werte, Rückgabewerte genannt, ausgeben. Um die verschiedenen Ausgaben (und auch Eingaben) zu verstehen, müssen Sie die Strukturen verstehen, mit denen R arbeitet.

Vektoren

Der Vektor ist die fundamentale Datenstruktur in R. Vektoren haben Sie bereits in vorherigen Beispielen gesehen. Ein Vektor ist ein eindimensionales Array von Datenelementen, die den gleichen Typ besitzen. Diese Datenelemente werden auch Komponenten genannt.