O'REILLY®

Statistik mit R

EINE PRAXISORIENTIERTE EINFÜHRUNG IN R

Joachim Zuckarelli

Statistik mit R

Eine praxisorientierte Einführung in R

Joachim Zuckarelli

O'REILLY®

Joachim Zuckarelli

Lektorat: Alexandra Follenius
Fachgutachten: Jörg Staudemeyer, Jörg Beyer
Korrektorat: Sibylle Feldmann, www.richtiger-text.de
Satz: III-satz, www.drei-satz.de
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Michael Oréal, www.oreal.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Print 978-3-96009-044-1
PDF 978-3-96010-141-3
ePub 978-3-96010-142-0
mobi 978-3-96010-143-7

Dieses Buch erscheint in Kooperation mit O'Reilly Media, Inc. unter dem Imprint »O'REILLY«.
O'REILLY ist ein Markenzeichen und eine eingetragene Marke von O'Reilly Media, Inc. und wird mit Einwilligung des Eigentümers verwendet.

1. Auflage 2017

5 4 3 2 1 0

Inhalt

KAPITEL 1

Einführung

Die Verbreitung der Statistiksoftware R hat in den vergangenen Jahren deutlich Fahrt aufgenommen. Kein Wunder also, dass es mittlerweile im Internet eine beträchtliche Menge von Tutorials und Foren gibt, die sich dem Open-Source-Programm widmen. Auch an Fachbüchern herrscht sicherlich kein Mangel.

Die meisten dieser Fachbücher führen systematisch in die *Programmiersprache R* ein und beschäftigen sich der Reihe nach ausgiebig mit den Sprachkonzepten, auf denen R aufgebaut ist. Viele der Bücher sind ausgezeichnete Programmierkurse, die dem Leser die Grundlagen der R-Programmierung vermitteln. Vorerfahrung in anderen Programmiersprachen wird vielleicht nicht notwendigerweise vorausgesetzt, ist dem Verständnis der Materie aber ungemein zuträglich. Gleiches gilt für Kenntnisse über statistische Methoden, deren Umsetzung in R – oftmals auch eher knapp – vorgestellt wird. Wenn sich der Leser mehr für die Hintergründe der statistischen Methodik und ihrer Anwendung interessiert, sei er wiederum auf die einschlägigen Statistiklehrbücher verwiesen.

Dieses Buch geht einen vollkommen anderen Weg.

Statistische Methodik und deren Anwendung in R werden Sie hier *gemeinsam* behandelt finden. Das Ziel dabei ist, ein einführendes Verständnis von – der Titel sagt es bereits – *Statistik mit R* zu vermitteln. Dieses Buch ist also nicht einfach ein Buch über R. Auch nicht über Statistik. Es ist ein Buch über Statistik *mit* R. Deren Verbindung steht im Vordergrund dieser Einführung.

Dementsprechend wiederholen wir an den entsprechenden Stellen im Buch zunächst die statistischen Konzepte, bevor wir sie in R umsetzen. Auch der Interpretation der Ergebnisse messen wir einen hohen Stellenwert bei. Denn das Buch soll Ihnen nicht nur helfen, Ihre statistischen Kenntnisse aufzufrischen und sie erfolgreich in R umzusetzen. Sie sollen auch verstehen, was die Ergebnisse, die Sie auf diese Weise produzieren, eigentlich bedeuten.

Weil wir Wert auf Anwendungsorientierung legen, beschäftigen wir uns zum Beispiel auch mit den am häufigsten anzutreffenden Fehlermeldungen, die gerade Anfänger gern an den Rand der Verzweiflung treiben. Dafür sparen wir uns lange theoretische Erörterungen der Sprachkonzepte von R – so elegant und faszinie-

rend sie auch sein mögen (auf die eine oder andere Randbemerkung konnte der Autor aber nicht verzichten). Mit einem echten Datensatz, den Sie unter *http://downloads.oreilly.de/9783960090441* ebenso von der Website zum Buch herunterladen können wie die Beispielskripte, mit denen wir hier arbeiten, steigen Sie direkt in die praktische Arbeit ein.

Das Buch versteht sich im besten Sinne des Wortes als Einführung, es *führt* Sie systematisch in die Statistik mit R *ein*. Als Nachschlagewerk ist es nicht primär gedacht. Viele Abschnitte beginnen aber mit einer Übersicht, der Sie über die Inhalte des folgenden Texts informiert und die R-Anweisungen zusammenfasst, die in diesem Abschnitt behandelt werden. Lesen Sie die Kapitel am besten der Reihe nach, ihre Anordnung orientiert sich an der Struktur des statistischen Arbeitens – vom Vorbereiten der Daten über die eigentliche Analyse bis hin zur Präsentation der Ergebnisse. Zudem werden Sie viele interessante Erkenntnisse über R entlang des Weges entdecken in Abschnitten, die sich primär einem ganz anderen Thema widmen. Denn hier steht die Anwendung von R für statistische Zwecke im Vordergrund, nicht R selbst. Deshalb behandeln wir R-Techniken dort, wo sie für die Anwendung am meisten Sinn ergeben.

Weil *Statistik mit R* eine Einführung ist, können wir natürlich nicht alle denkbaren statistischen Methoden ausführlich diskutieren. Themen wie Panelmodelle oder Zeitreihenanalyse werden Sie hier vergeblich suchen. Dafür setzen wir nur geringe statistische Vorkenntnisse und überhaupt keine Programmiererfahrung voraus. Alles, was Sie zum Verständnis brauchen, werden Sie beim Lesen lernen bzw. wiederholen. Wenn Sie bereits über entsprechende statistische Kenntnisse verfügen, können Sie die überwiegend in separaten Repetitorien organisierte Wiederholung der statistischen Konzepte natürlich überspringen und sich direkt auf die Umsetzung in R konzentrieren.

In diesem ersten Kapitel erfahren Sie, wie dieses Buch aufgebaut ist und wie Sie es am besten nutzen, um einen reibungslosen Einstieg in die Arbeit mit R zu finden. Außerdem werden Sie einiges über R selbst lernen – was R eigentlich ist, was man damit machen kann und was es so besonders attraktiv macht.

Damit Sie startbereit für die nächsten Kapitel sind, in denen wir in die Arbeit mit R einsteigen, erfahren Sie hier außerdem, wie Sie R installieren und wie Sie sich Hilfe zu R beschaffen, wenn Sie einmal nicht weiterkommen.

R ist ein Open-Source-Programm. Viele engagierte Menschen überall auf der Welt arbeiten hart daran, es fortzuentwickeln und seinen Funktionsumfang ständig zu erweitern. Ohne deren großartigen Einsatz gäbe es weder R noch dieses Buch. Ihre Arbeit verdient allerhöchste Anerkennung.

Bedanken möchte ich mich an dieser Stelle auch bei Alexandra Follenius vom O'Reilly Verlag, die den langen und arbeitsreichen Prozess, der schließlich zu diesem Buch geführt hat, mit Rat und Tat begleitet hat, sowie bei den fachlichen Gut-

achtern Jörg Beyer und Jörg Staudemeyer, die viele wertvolle Anregungen geliefert und so erheblich zum Gelingen des Buchs beigetragen haben.

Dank gebührt vor allem aber meiner wunderbaren Frau Anja, ohne deren unermüdliche Unterstützung und immense Geduld dieses Buch und so vieles andere nicht möglich wäre.

Jetzt aber viel Spaß bei den ersten Schritten in die faszinierende Welt von R!

An wen richtet sich dieses Buch?

Statistik mit R wird Sie rasch in die Lage versetzen, selbstständig mit R zu arbeiten. Ganz gleich, ob Sie über einem Seminarpapier sitzen, Ihre Bachelor- oder Master-Thesis anfertigen, oder Ihre Dissertation schreiben; ganz gleich, ob Sie R in der akademischen Welt einsetzen, zum Beispiel im betriebswirtschaftlichen, volkswirtschaftlichen, sozial- oder politikwissenschaftlichen Bereich, oder ob Sie beruflich mit Statistik zu tun haben, weil Sie zum Beispiel Finanzmarktdaten auswerten oder an Marktforschungsstudien arbeiten – dieses Buch bietet Ihnen eine pragmatische und praxisorientierte Einführung in die statistische Arbeit mit R.

Aufbau dieses Buchs

Das Buch ist wie folgt aufgebaut:

- In Kapitel 1 lernen Sie die Grundlagen von R kennen, woher Sie R beziehen können, wie Sie es installieren und wie Sie (abgesehen von diesem Buch) weitere Hilfe, Unterstützung und Informationen zu R erhalten.
- Kapitel 2 widmet sich der oft unterschätzten, tatsächlich für systematisches und fehlerfreies Arbeiten aber sehr wichtigen Frage, wie Sie Ihre Arbeit mit R und um R herum geschickt organisieren, um effizient zu sein, Ordnung in Ihren Daten und Dokumenten zu halten und Datenverlusten vorzubeugen.
- In Kapitel 3 beschäftigen wir uns damit, wie man in R mit Daten arbeitet. Insbesondere lernen Sie hier, wie R Daten speichert und wie Sie Ihre Daten in R einlesen können.
- Kapitel 4 ist ganz der Datenaufbereitung gewidmet, also den Vorbereitungen der eigentlichen statistischen Analysen. Hier sehen Sie unter anderem, wie Sie mit unterschiedlichen Datensätzen arbeiten und wie Sie Daten filtern, sortieren und nach Kriterien selektieren können.
- In Kapitel 5 beginnen wir mit der Datenanalyse, zunächst mit deskriptiven Untersuchungen, um den Datensatz genauer kennenzulernen. Dieses Kapitel behandelt nicht nur die praktische Durchführung deskriptiver Analysen in R, sondern umfasst auch ein Repetitorium zur deskriptiven Statistik.
- Kapitel 6 führt in die lineare Regression ein. Auch hier können Sie, sofern Bedarf besteht, zunächst die statistischen Grundlagen wiederholen, bevor Sie

in die eigentliche Arbeit mit R einsteigen. Neben der Schätzung von Regressionsmodellen behandelt dieses Kapitel auch Hypothesentests sowie den Umgang mit Verletzungen der Annahmen des linearen Regressionsmodells.

- In Kapitel 7 wenden wir uns der Analyse kategorialer Daten zu, also Daten, die nicht jeden beliebigen Wert annehmen können, sondern nur bestimmte, festgelegte Ausprägungen. Die Analyse kategorialer Daten ist in vielfacher Hinsicht eine Erweiterung des linearen Regressionsmodells und schließt insofern an das vorangegangene Kapitel an.
- Kapitel 8 beschäftigt sich damit, wie Sie die Ergebnisse, die mit den in den Kapiteln 5, 6 und 7 behandelten Methoden erzielt wurden, in Tabellen und Grafiken aussagekräftig, übersichtlich und ansprechend präsentieren können.
- Kapitel 9 schließlich widmet sich der Programmierung mit R, also der Frage, wie Sie nicht nur einzelne R-Anweisungen ausführen, sondern viele Anweisungen zu ganzen Programmen zusammensetzen können, zum Beispiel, um wiederkehrende Aufgaben effizient zu automatisieren.

Was ist R?

R – das ist der 18. Buchstabe des Alphabets und zugleich der Anfangsbuchstabe der Vornamen von Ross Ilhaka und Robert Gentlemen. Diese beiden Herren schufen Anfang der Neunzigerjahre des letzten Jahrhunderts an der Universität der neuseeländischen Millionenstadt Auckland auf Basis einer älteren Sprache namens *S* eine neue Programmiersprache, deren Haupteinsatzgebiet die Verarbeitung und Analyse statistischer Daten ist und die man heute, was die beiden damals sicherlich nicht zu träumen wagten, ohne zu übertreiben als Welterfolg bezeichnen kann.

R ist heutzutage aus der akademischen Welt nicht mehr wegzudenken und wird zunehmend auch von Unternehmen eingesetzt. Ersteres zeigt sich nicht zuletzt darin, dass neue statistische Methoden heute oft als Erstes in R programmiert und verbreitet werden. Ein gutes Indiz für die wachsende Popularität von R im geschäftlichen Umfeld ist der Umstand, dass immer mehr Unternehmen R in ihre Produkte integrieren und einige große Player der Softwarebranche, darunter Microsoft und Oracle, das *R Consortium* gegründet haben, um die Entwicklung und Anwendung von R weiter zu fördern.

Seinen Erfolg verdankt R nicht zuletzt der Tatsache, dass es für den Anwender kostenlos ist. Darin unterscheidet es sich von den bekannten kommerziellen Statistiksoftwarepaketen, wie beispielsweise SPSS oder Stata, bei denen man für jährliche Lizenzen selbst der einfachsten Programmeditionen durchaus bereits mehrere Hundert Euro zahlen kann. Wer nicht so viel Geld in die Hand nehmen will oder kann, für den ist R eine günstige Alternative.

R wird unter der sogenannten *GNU General Public License* angeboten, bei der es sich, anders als der ungewöhnliche Name vielleicht vermuten lässt, mitnichten um einen Vertrag über die öffentliche Nutzung afrikanischer Antilopen handelt. Es ist vielmehr eine Softwarelizenz, die dem Nutzer einige Grundfreiheiten garantiert, darunter das Recht, seine in R selbst geschriebenen Programme weiterzuverteilen und sogar R selbst zu verändern. Dieser Umstand führt zum zweiten wesentlichen Erfolgsfaktor von R: Neben der eigentlichen R-Software gibt es buchstäblich Tausende von Erweiterungen, die von Benutzern entwickelt worden sind. Mit diesen Erweiterungspaketen, die Sie sich kostenlos aus dem Internet über das *Comprehensive R Archive Network* (*CRAN*, Website: *https://cran.r-project.org/*) herunterladen können, lässt sich der Funktionsumfang von R beträchtlich erweitern. Kaum ein statistisches Verfahren existiert, für das es in R nicht eine passende Implementierung, das heißt eine Umsetzung in ein R-Programm, gibt. Täglich kommen neue Pakete hinzu. Mit R leben Sie gewissermaßen im Schlaraffenland der computergestützten Statistik!

R wird aber nicht nur durch die aktive R-Community weiterentwickelt, die ständig neue Funktionspakete veröffentlicht, auch der Kern von R wird laufend verbessert und erweitert. Darum kümmert sich die *R Foundation* (Website: *https://www.r-project.org/*), die als Non-Profit-Organisation mit Sitz in Wien mit ihrem *R Development Core Team* die Entwicklungsaktivitäten rund um R koordiniert und die Nutzung von R fördert.

Keine Angst vorm Programmieren!

R unterscheidet sich von den kommerziellen Statistikprogrammen nicht nur dadurch, dass es kostenlos ist. Anders als bei kommerziellen Programmen gibt es standardmäßig in R nur eine sehr rudimentäre Benutzeroberfläche. Wenn Sie durch Ihren Umgang zum Beispiel mit Ihrem Betriebssystem oder einem Office-Paket eine hübsche, übersichtliche Oberfläche gewohnt sind, über die Sie alle Funktionen des Programms bequem per Maus ansteuern und deren Ausführung Sie in übersichtlichen Dialogfenstern genau steuern können, wird R Sie massiv enttäuschen. R hat praktisch keine nennenswerte grafische Benutzeroberfläche. Der Kern von R ist stattdessen die gleichnamige Programmiersprache.

Programmiersprache? Moment mal! Ist Programmieren nicht das, was diese Nerds tun, die ihre Computer mit unendlich langen kryptischen Befehlen füttern? Genau so ist es. Programmieren bedeutet, einem Computer mitzuteilen, was er tun soll. Das geschieht in einer künstlichen Sprache, die meist, und so auch im Fall von R, an das Englische angelehnt ist. Und wie in der natürlichen Sprache gibt es nicht nur Wörter, die zur Sprache gehören, sondern auch eine Grammatik, die sogenannte Syntax, die Sie befolgen müssen, sonst werden Sie von Ihrem Gegenüber nicht verstanden.

Über das Programmieren halten sich hartnäckig einige Vorurteile. Es sei schwierig zu erlernen. Es sei nur etwas für »Techies«. Es sei eine weniger werthaltige, sondern eher ausführende Tätigkeit, während »geistigere«, »strategischere« Tätigkeiten wichtiger, wertschaffender und deshalb überlegen seien. Es sei etwas, mit dem sich nur Männer beschäftigten.

Schwierig zu erlernen? Im Vergleich zu Fremdsprachen wie Englisch, Französisch und Spanisch sind Programmiersprachen wie R erheblich *leichter* zu lernen. Der Wortschatz ist überschaubar, Sie müssen nicht ständig Vokabeln pauken, und auf Ihre Aussprache kommt es auch nicht an. Sprachen übt man ja bekanntlich am besten durch Gespräche mit einem Muttersprachler. Das ist bei Programmiersprachen auch so. Und der Muttersprachler für R steht auf Ihrem Schreibtisch, es ist Ihr Computer. Mithilfe eines geeigneten Sprachkurses, wie ihn dieses Buch darstellt, und durch Üben werden Sie ein gutes Sprachniveau erreichen, das es Ihnen erlaubt, fließend zu sprechen. Das heißt im Fall von Programmiersprachen, ohne große Schwierigkeiten funktionsfähige Programme zu schreiben. Im Übrigen gibt es eine ganze Reihe von Konzepten, die in praktisch allen Programmiersprachen sehr ähnlich sind. Wenn Sie eine Programmiersprache wie R beherrschen, wird Ihnen das Erlernen einer weiteren Sprache sehr viel leichter fallen.

Nur etwas für Techies? Mitnichten! Wenn Sie programmieren können, werden Ihnen Dinge möglich sein, die andere nicht können, Sie werden sich an vielen Stellen das Leben leichter machen können und werden andere mit Ihrem Können zum Staunen bringen. *Programming is the closest we have to a superpower* sagte der Gründer eines amerikanischen Start-ups. Diese Superkräfte helfen jedem, ob Techie oder nicht. Außerdem lernen Sie beim Programmieren, Probleme systematisch zu zerlegen und Lösungen zu entwickeln, die schrittweise die Teilaspekte des Problems adressieren. Dieses systematische Nachdenken über Problemlösungen ist etwas unglaublich Alltagsnützliches, das Ihnen in vielen Situationen, die überhaupt nichts mit Computern und Technologie zu tun haben, erheblich nützen wird.

Weniger wertschaffend? Dieses Vorurteil hört man am häufigsten von denjenigen, die wenig Verständnis von und über Software und Programmierung mitbringen und sich auch lieber gar nicht erst damit befassen wollen. Wir leben in einer Welt, in der Software überall ist. Früher war Software etwas, das in Ihrem Computer steckte. Heute steckt Software in Ihrem Telefon, Ihrem Auto, Ihrer Kaffeemaschine, Ihrer Heizung. Wenn Sie Ihre Wohnung verlassen, werden Sie keine drei Minuten gehen können, ohne dass Ihr Blick irgendetwas trifft, in dem Software steckt. Unsere gesamte Lebens- und Arbeitswelt wird durch die Produkte von Programmierern bestimmt. Software wird immer wichtiger. Etwas plakativ könnte man gar sagen: Es ist nicht mehr die reale, physische Welt, die die Software bestimmt, es ist die Software, die die Welt formt. Werte entstehen heute mehr als je zuvor durch Software selbst. Einige der wertvollsten Unternehmen der Welt, die ihre Branchen tiefgreifend verändert haben, sind gerade nicht von im Grunde tech-

nologieagnostischen Kaufleuten gegründet worden, die Programmierer beschäftigt haben, um ihre genialen Ideen einfach nur noch umsetzen zu lassen, sondern von Programmierern selbst. Wer heutzutage verstehen will, was die Welt im Innersten zusammenhält, muss in Grundzügen verstehen, wie Software funktioniert. Das lernen Sie durch Programmieren.

Nur etwas für Männer? Alles bisher Gesagte trifft auf Frauen und Männer in gleicher Weise zu. Trotzdem scheinen sich Jungs und Männer eher dafür begeistern zu können, mit Programmen zu »spielen« und selbst welche zu entwickeln. Warum auch immer das so *ist*, es gibt keinen offensichtlichen Grund, warum es so sein *muss*. Und es ist beinahe erstaunlich, wenn man bedenkt, welch große Rolle in der Geschichte Frauen für das Programmieren gespielt haben. Man denke in diesem Zusammenhang an die englische Mathematikerin Ada Lovelace (auch bekannt unter ihrem Geburtsnamen Augusta Ada Byron), die im 19. Jahrhundert nicht nur Programme für Charles Babagges Lochkartenmaschine entwickelte und damit als die erste Programmiererin der Welt gelten kann, sondern die auch viele jener Konzepte entwarf, die in allen modernen Programmiersprachen heutzutage absoluter Standard sind. Oder an Margaret Hamilton, die in den Sechzigerjahren des letzten Jahrhunderts für die NASA die Entwicklung des 40.000 Zeilen umfassenden Programms leitete, das die Apollo-11-Mission mit Neil Armstrong und Buzz Aldrin sicher auf den Mond brachte, und die quasi en passant bahnbrechende Konzepte im Bereich der Interaktion von Mensch und Computer entwickelte.

Es gibt wenig gute Gründe, sich vom Programmieren im Allgemeinen und von der Programmiersprache R im Besonderen abschrecken zu lassen. Dieses Buch ist gewissermaßen Ihr Sprachkurs für die Sprache R. Und es wird kein knochentrockener Sprachkurs sein, sondern einer, der relevant für Ihre praktische Arbeit ist. Deshalb arbeiten wir auch mit realen Beispielen und mit realen Daten. Bevor wir aber ans Werk gehen, müssen Sie R zunächst auf Ihrem Computer installieren. Darum geht es im nächsten Abschnitt.

R installieren

R läuft auf Windows-, Mac OS- und Linux-Systemen gleichermaßen. Die Installation auf einem Windows-Computer oder einem Mac ist in der Regel problemlos innerhalb weniger Minuten abgeschlossen.

Gehen Sie zunächst auf die Website des R-Projekts (Website: *https://www.r-project.org/*). Dort sehen Sie oben links unter der Überschrift *Download* den Hyperlink *CRAN*. Wie Sie bereits wissen, ist CRAN das *Comprehensive R Archive Network*, jener Ort, an dem die Tausende von R-Paketen liegen, die andere R-Benutzer zu Erweiterung des Funktionsumfangs von R entwickelt haben und der interessierten Öffentlichkeit kostenlos zur Verfügung stellen. Von CRAN kann

aber auch R selbst heruntergeladen werden. Wenn Sie auf *Download* geklickt haben, gelangen Sie auf eine Seite, auf der Sie aufgefordert werden, einen sogenannten *Mirror* auszuwählen. Das CRAN ist, wie der Name schon andeutet, tatsächlich ein Netzwerk, denn das *Comprehensive R Archive* liegt nicht auf einem einzigen Server, sondern auf einer ganzen Reihe von Servern überall auf der Welt. Alle Server stellen dabei exakt den gleichen Inhalt bereit, sie spiegeln gewissermaßen jeden anderen Server, daher auch der Begriff »Mirror«. Die meisten dieser Mirror-Server werden von Universitäten betrieben. Wählen Sie hier einfach einen der paar deutschen Server (in der Hoffnung, dass die Daten dann nicht um die ganze Welt geschickt werden müssen).

Sie gelangen nun auf die Startseite des ausgewählten Mirror-Servers, von wo aus Sie unter der Überschrift *Download and Install R* zunächst wählen können, für welches Betriebssystem Sie R herunterladen möchten.

Wenn Sie R auf einem Windows-System installieren wollen, klicken Sie auf der folgenden *R for Windows*-Seite auf den Link *Base* oder auf *install R for the first time*. Über beide Links kommen Sie auf eine Seite, auf der Ihnen oben die jeweils aktuellste Version von R zum Download angeboten wird.

Typisch für R ist die aus drei Komponenten bestehende Versionsnummer – zu dem Zeitpunkt, an dem dieses Buch entstand, war das 3.3.1. Die letzte Ziffer wird hochgezählt bei kleineren Updates, die Bugs, also Fehler, korrigieren, die mittlere Ziffer bei »normalen« Änderungen, die nicht in erster Linie fehlergetrieben sind, die erste bei sehr bedeutenden Änderungen. Tatsächlich ist es aber in den meisten Fällen gar nicht so relevant, welche Version genau Sie verwenden. Die Änderungen, die an R vorgenommen werden, finden meist »unter der Haube« statt und bleiben normalen R-Nutzern eher verborgen. Laden Sie daher einfach die aktuellste Version herunter. Auch unter älteren Windows-Betriebssystemen sollte Sie damit keine Probleme bekommen.

Wenn Sie R auf einem Mac-System installieren wollen, werden Sie auf der Seite »R for Mac OS X« feststellen, dass es unterschiedliche R-Versionen je nach Version von Mac OS gibt. Laden Sie, um Probleme zu vermeiden, am besten die für Ihre Version von Max OS gedachte Version von R herunter.

Die Installation läuft unter Windows und auch Mac OS sehr einfach ab: Starten Sie den Installer und folgenden Sie den Anweisungen. Wir benutzen im Buch die englischsprachige Version von R. Das hat den Vorteil, dass man es etwas leichter hat, in Internetforen zum Beispiel nach der Bedeutung von Fehlermeldungen zu suchen, weil es einfach mehr englischsprachige R-Foren gibt und diese mehr Teilnehmer haben als die deutschsprachigen. Sie können aber natürlich ebenso gut während des Installationsprozeses Deutsch als Sprache auswählen.

Nach der Installation können Sie R nun erstmals starten. Ihnen wird sofort die puristisch (und vielleicht auch ein wenig altbacken) anmutende *RGui* (*R Graphi-*

cal User Interface) auffallen – die Standardbenutzeroberfläche von R –, die sie hoffentlich nach dem vorangegangenen Abschnitt nicht mehr verschreckt. Abbildung 1-1 zeigt, wie sich R nach dem ersten Start präsentiert.

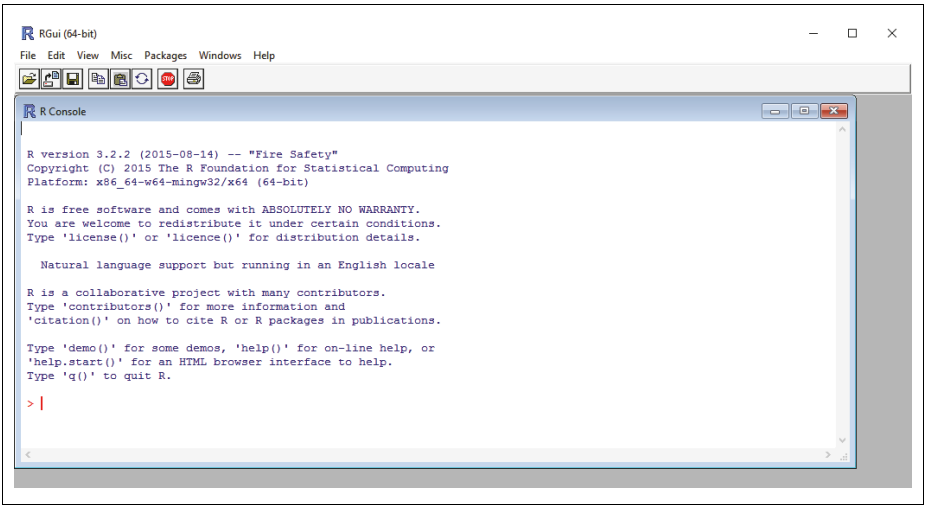


Abbildung 1-1: Die Benutzeroberfläche von R nach dem ersten Start

Das Fenster im Vordergrund ist die sogenannte *R-Konsole*. Hier können nicht nur die R-Befehle eingegeben werden, R zeigt Ihnen in diesem Fenster auch die Ergebnisse der von Ihnen durchgeführten Operationen an. Wenn Sie in der R-Konsole nicht mit dem interaktiven Modus arbeiten, sondern mehrere R-Befehle in einem Skript zusammenfassen möchten, kommt eventuell noch ein weiteres Fenster hinzu, das das aktuell bearbeitete R-Skript anzeigt. (Im interaktiven Modus geben Sie einen R-Befehl ein, R zeigt Ihnen das Ergebnis an, Sie geben den nächsten R-Befehl ein und so fort. Wir schauen uns die beiden Eingabemodi von R im ersten Abschnitt des zweiten Kapitels genauer an.)

Das Menü oben bietet einige rudimentäre Funktionalitäten, zum Beispiel zum Installieren von Packages, also den bereits mehrfach angesprochenen Erweiterungspaketen. Trotz allem ist der Standard-R-Editor RGui eher eine spartanisch ausgestattete Software. Deshalb gibt es eine ganze Reihe von alternativen R-Editoren, mit denen wir uns kurz im nächsten Abschnitt beschäftigen wollen.

Vorher aber, da Sie ja nun R bereits gestartet haben, ein erster kleiner Schritt mit R. Geben Sie in die R-Konsole einmal Folgendes ein:

```
> 2 + 3
[1] 5
```

Sie haben gerade Ihre erste Berechnung mit R ausgeführt! Es ist zugegebenermaßen nicht die anspruchsvollste Berechnung. Aber Sie haben eine Eingabe gemacht, und R zeigt Ihnen das Ergebnis an. R ist natürlich weit mehr als nur der Taschenrechner, als den wir es gerade verwendet haben. Im weiteren Verlauf des Buchs

werden Sie ein wenig mehr von den gigantischen Möglichkeiten kennenlernen, die Sie mit R haben.

Komfortabler arbeiten: R-Editoren

Wer bei der Arbeit mit R etwas mehr Komfort genießen will, als ihn RGui bietet, ist auf alternative Editoren angewiesen. An diesen herrscht allerdings wahrlich kein Mangel.

Ein prominentes Beispiel ist der in Abbildung 1-2 dargestellte R Commander (Website: *http://www.rcommander.com/*). Er selbst kommt als R-Package, also als Erweiterungspaket, daher und kann dementsprechend auch als Package `Rcmdr` von CRAN heruntergeladen werden. Wie das genau geht, erfahren Sie im folgenden Kapitel im Abschnitt »Packages verwenden«. Anders als RGui bietet der R Commander die Möglichkeit, über Menüs und Dialoge bequem viele häufig genutzte statistische Funktionen aufzurufen. Der R Commander übersetzt die Eingaben des Benutzers in R-Code und führt diesen im Hintergrund aus. Dieser Bedienkomfort kommt natürlich nicht im Entferntesten an die kommerziellen Statistikpakete heran, mag aber für jemanden, der sich erst mal vorsichtig herantasten möchte, durchaus sehr nützlich sein.

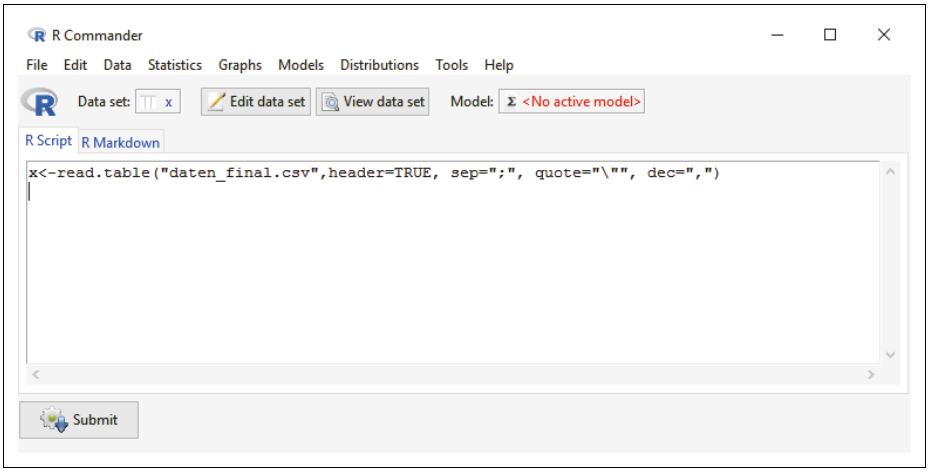


Abbildung 1-2: Der alternative R-Editor R Commander

Ein weiterer alternativer Editor, den ich selbst einsetze, ist *RStudio* (Website: *https://www.rstudio.com/products/rstudio/*). RStudio gibt es in einer kommerziellen und einer kostenfreien Version, die die gleiche Funktionalität bietet wie die kostenpflichtige Variante (die kommerzielle Version bietet darüber hinaus professionellen Support und ermöglicht Unternehmen, R zu nutzen, ohne den strengen Quellcode-Offenlegungspflichten der GNU General Public License genügen zu müssen). RStudio, dessen Oberfläche in Abbildung 1-3 dargestellt ist, unterscheidet sich vom R Commander vor allem dadurch, dass es keinen bequemen Zugriff

auf die statistischen Funktionen von R bietet, sondern stattdessen darauf fokussiert ist, die Entwicklung von R-Skripten, also das Programmieren in R, möglichst komfortabel zu gestalten. Zu den Funktionen von RStudio gehört beispielsweise das Syntax-Highlighting, bei dem bestimmte Schlüsselwörter, Variablennamen, Kommentare und andere Elemente von R-Skripten unterschiedlich gefärbt dargestellt werden, was die Lesbarkeit der Skripte deutlich erhöht. Daneben bietet RStudio einfachen Zugriff auf die aktuell verwendeten Datensätze und Variablen, integriert die Befehlshistorie und die Hilfe sehr schön in die Entwicklungsumgebung und stellt ein praktisches Management für R-Packages zur Verfügung. Auf die Historie, die Hilfe und die Packages kommen wir an späterer Stelle noch detaillierter zu sprechen.

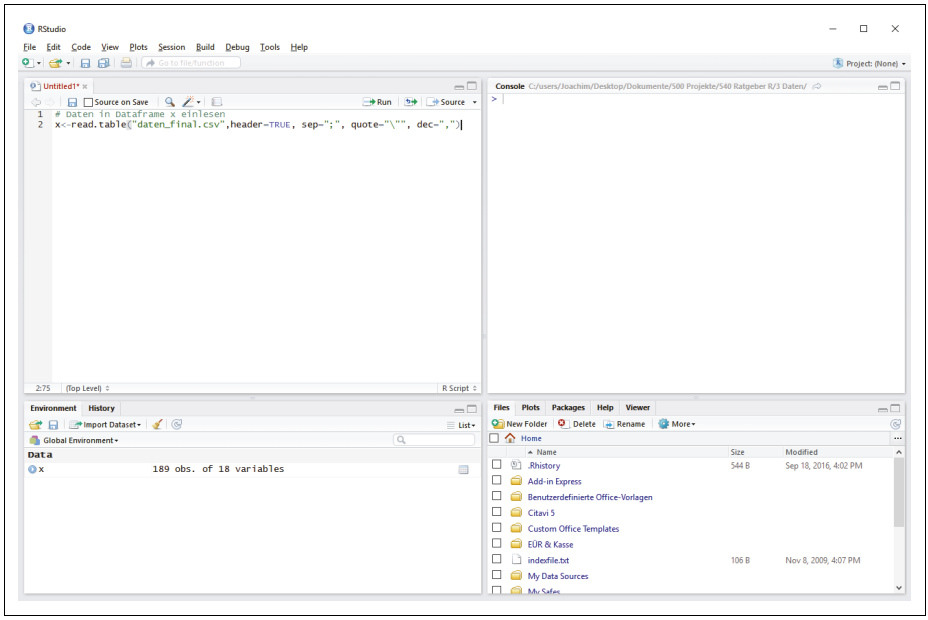


Abbildung 1-3: Der alternative R-Editor RStudio

R Commander und RStudio sind nur zwei Beispiele einer ganzen Reihe von Entwicklungsumgebungen, die die Arbeit mit R erleichtern sollen. Sucht man im Internet zum Beispiel nach »R editors«, stößt man sofort auf unzählige Blogs und Webseiten, die das Thema diskutieren. An den beiden hier kurz vorgestellten Beispielen sehen Sie aber bereits, dass die verschiedenen Editoren unterschiedliche Schwerpunkte setzen.

Auch wenn wir in diesem Buch RStudio einsetzen: Welchen R-Editor Sie verwenden sollten, ist letztlich eine Frage der persönlichen Präferenzen. Probieren Sie ruhig einige Programme aus. Finden Sie heraus, mit welchen Sie gut zurechtkommen und welche die Funktionen bieten, die für Sie in Ihrer täglichen Arbeit am wichtigsten sind.

Hilfe zu R bekommen

In diesem Abschnitt erfahren Sie,

- wie Sie die eingebaute Hilfe von R verwenden,
- welche externen Informationsquellen rund um R Sie einsetzen können.

Folgende R-Funktionen werden in diesem Abschnitt behandelt:

- **`?`** / **`help()`**: Zeigt Hilfeinformationen zu einer R-Funktion an.
- **`??`** / **`help.search()`**: Durchsucht die R-Hilfe nach einem Begriff.
- **`args()`**: Zeigt die Argumente einer Funktion an, das heißt die Werte, die ihr übergeben werden müssen.
- **`example()`**: Zeigt die in der R-Hilfe hinterlegten Beispiele zu einer Funktion an.
- **`vignette()`**: Zeigt eine Vignette – also eine weiterführende Erläuterung im PDF-Format, die typischerweise auch statistische Hintergrundinformationen beinhaltet – zu einer oder mehreren Funktionen oder auch einem ganzen Package an.
- **`vignettes()`**: Listet alle verfügbaren Vignettes in einer Übersicht auf.

Nicht nur als Anfänger braucht man von Zeit zu Zeit Hilfe. Auch wenn man schon eine Weile mit R gearbeitet hat, ergibt sich immer wieder der Bedarf, z. B. die Argumente, die einer statistischen Funktion übergeben werden müssen, nachzuschlagen.

Alle R-Packages, sowohl diejenigen, die bereits bei der Erstinstallation von R mitinstalliert werden und zum Kern von R gehören, als auch die Erweiterungspackages, die Sie sich nach Bedarf von CRAN herunterladen können, beinhalten jeweils Hilfeinformationen. Diese Hilfeinformationen sind zwar – und hier erkennt man die unterschiedliche Herangehensweise der Autoren der Packages – durchaus verschieden in Hinblick auf Umfang und Verständlichkeit, im Allgemeinen aber sehr gut, und sie helfen wirklich weiter. Die Hilfeseiten wirken gerade für Anfänger im statistischen Metier nicht selten recht komplex und sind in Teilen ohne umfangreicheres statistisches Hintergrundwissen einigermaßen unverständlich. Lassen Sie sich nicht davon abschrecken, dass Sie nicht alles verstehen! Die R-Funktionen (also die R-Anweisungen, die wir aufrufen können, um unsere Daten zu bearbeiten und zu analysieren), die wir in diesem Buch behandeln, können in der Regel noch viel mehr, als wir uns hier anschauen wollen. Um sinnvoll mit ihnen zu arbeiten, benötigen Sie aber häufig nur einen Bruchteil der in der Hilfe erläuterten Funktionalität. Möchten Sie dann über die Standardverfahren hinaus etwas fortgeschrittenere statistische Methoden anwenden, werden Sie positiv überrascht sein, wie viel davon die R-Funktionen bereits von Haus aus mitbringen.

Wenn Sie zu einer bestimmten Funktion Hilfe benötigen, geben Sie in Ihren R-Editor einfach den Namen der Funktion mit einem vorangestellten Fragezeichen ein, und schon liefert Ihnen R Informationen zu dieser Funktion. Mit `?median`

rufen Sie die Hilfeinformationen zur Funktion `median` auf, die den Median einer Variablen berechnet. Alternativ können Sie auch `help(median)` eingeben. R zeigt sogleich eine Hilfeseite an. Wenn Sie RGui, den Standardeditor von R, verwenden, ruft R die Hilfeseite in Ihrem Webbrowser auf. Dazu bedarf es jedoch keiner Internetverbindung, denn die Hilfedateien werden beim Installieren der R-Packages mit heruntergeladen. Sie können dieselben Hilfeseiten aber auch im Internet aufrufen, denn CRAN stellt diese für alle Packages öffentlich bereit. Wenn Sie genau wissen, wonach Sie suchen – wie in unserem Beispiel oben –, bietet es sich an, direkt über R zu suchen. Das erspart Ihnen die mitunter mühselige Suche nach dem richtigen Link in den Web-Suchergebnissen. Nutzen Sie RStudio, wird die Hilfeseite direkt in RStudio angezeigt (Abbildung 1-4).

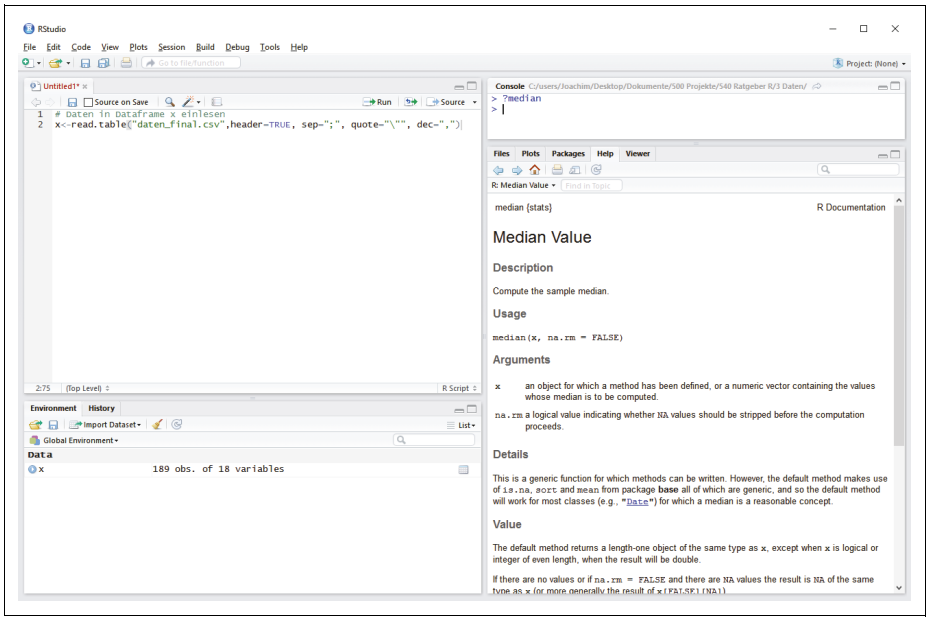


Abbildung 1-4: Anzeige der Hilfeseite für die Funktion median in RStudio

Die Hilfeseiten sind immer gleich aufgebaut: Unter *Description* sehen Sie zunächst einen kurzen Überblick darüber, was die Funktion leistet, oftmals tatsächlich nur ein Satz. Im Abschnitt *Usage* erfahren Sie, wie die Funktion aufgerufen wird, das heißt, welche Argumente ihr beim Aufruf übergeben werden müssen. Im Beispiel unserer Funktion `median` ist das natürlich die Variable, deren Median bestimmt werden soll, und darüber hinaus eine Angabe, die bestimmt, ob leere Datenpunkte, sogenannte *Missings*, ignoriert werden sollen (mit der Rolle von Missings beschäftigen wir uns im Abschnitt »Mit Missings umgehen« auf Seite 46 ausführlicher). Wenn Sie einmal direkt aus R heraus schnell sehen wollen, welche Argumente eine Funktion hat, können Sie, statt die Hilfe aufzurufen, auch einfach die Anweisung `args(funktionsname)` eingeben. Sie erhalten dann die Liste der Argumente der Funktion `funktionsname`, wie im Folgenden für die Funktion `median` gezeigt (ignorieren Sie die Ausgabe `NULL` in der zweiten Zeile, sie hat eher technische Gründe):

```
> args(median)
function (x, na.rm = FALSE)
NULL
```

Aber zurück zur Hilfe. Unter dem Abschnitt mit den Argumenten der Funktion sehen Sie einen Abschnitt *Value*, der beschreibt, wie der Rückgabewert der Funktion zu interpretieren ist. In unserem Beispiel sagt uns »The default method returns a length-one object of the same type as x«, dass wir, wenn wir der Funktion `median` als Argument einen Vektor von Ganzzahlvariablen (zum Beispiel `1,3,7`) übergeben, als Rückgabewert wiederum eine Ganzzahlvariable erwarten dürfen, und zwar von der Länge 1, denn das Ergebnis des Medians unseres Vektors mit drei Elementen ist natürlich nur *eine* Zahl. Weitere Standardabschnitte auf einer R-Hilfeseite (verdeckt in Abbildung 1-4) sind *References*, in dem auf relevante Literatur verwiesen wird (zum Beispiel auf die Monografie oder den Journal-Artikel, in dem die betreffende statistische Methode erstmals vorgeschlagen worden ist), und *Examples*, in dem Sie anhand von Beispielen sehen können, wie man die Funktion verwendet. Diese Beispiele sind sofort in R lauffähig. Sie können sie also in den R-Editor kopieren, dort ausführen und ihre Ergebnisse unmittelbar auswerten. Das Herauskopieren des Beispiels aus der Hilfeseite können Sie sich auch sparen, indem Sie sich mittels der Funktion `example` das Beispiel direkt im R-Editor anzeigen lassen:

```
> example(median)
median> median(1:4)                # = 2.5 [even number]
[1] 2.5
median> median(c(1:3, 100, 1000))  # = 3 [odd, robust]
[1] 3
```

Einige R-Editoren, wie beispielsweise RStudio, geben Ihnen bereits während der Eingabe eine Hilfestellung. In Abbildung 1-5 sehen Sie, wie bereits nach Eingabe weniger Buchstaben eine Einblendung erscheint. Diese zeigt ähnlich wie eine Autovervollständigung mögliche Funktionen an, die Sie aufrufen können, und teilt Ihnen direkt mit, was die Funktion leistet und welche Argumente sie benötigt.

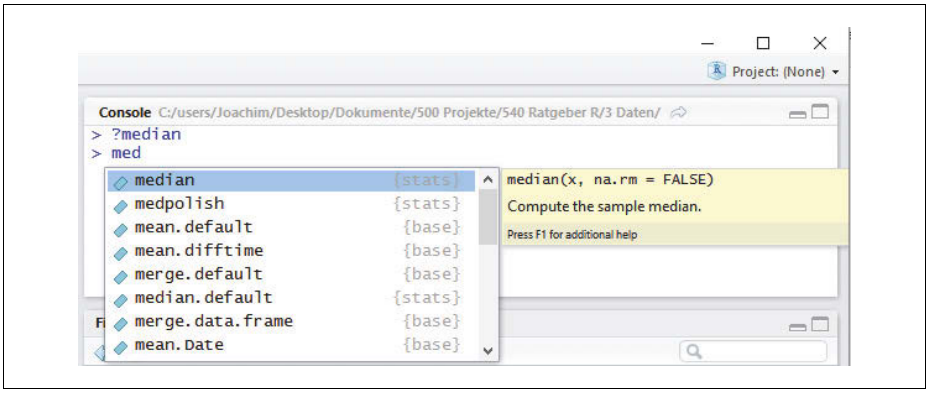


Abbildung 1-5: Kontextsensitive Hilfe in RStudio

Wenn Sie noch nicht genau wissen, nach welcher Funktion Sie suchen wollen, können Sie die Funktion `help.search` verwenden. Wollen Sie beispielsweise erfahren, welche Funktionen zum Thema »Varianz« angeboten werden, rufen Sie einfach `help.search("variance")` oder alternativ auch einfach `??variance` (hier ohne Anführungszeichen!) auf. Das Ergebnis dieser Suche ist in Abbildung 1-6 dargestellt. Links sehen Sie stets eine R-Funktion, rechts eine kurze Beschreibung dieser Funktion. Wenn Sie mit RStudio arbeiten, ist es ebenfalls möglich, im Register *Help* einen Suchbegriff in das Suchfeld rechts oben einzugeben. Sie bekommen dann eine Vorschlagsliste von Sucheinträgen, die mit dem von Ihnen eingegebenen Suchbegriff beginnen.

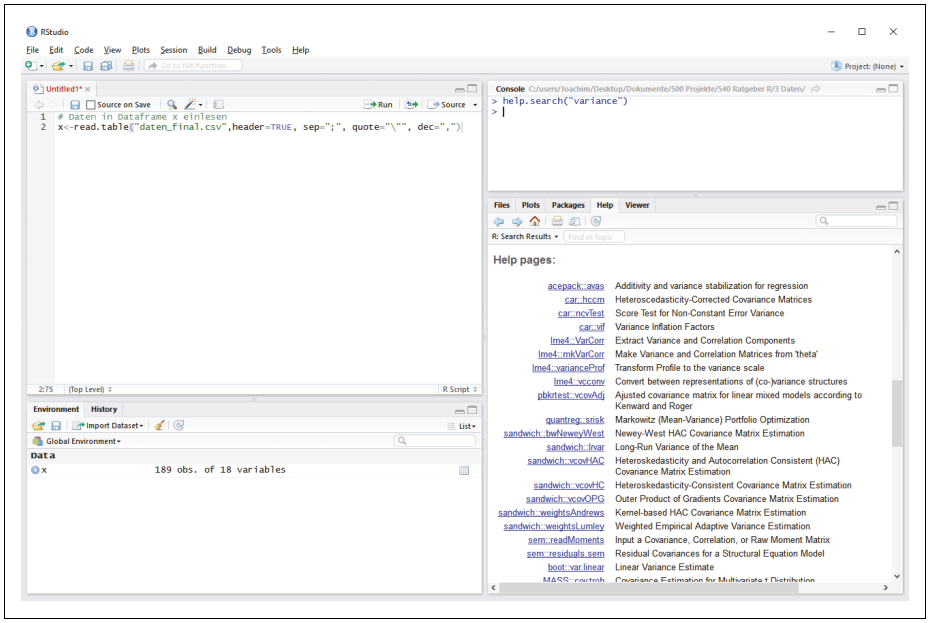


Abbildung 1-6: Ergebnisse der Suche nach variance

Zu vielen R-Packages gibt es neben der eigentlichen Hilfe noch eine weitere Informationsquelle, die bei der Package-Installation frei Haus geliefert wird: die sogenannten *Vignettes*. Diese sind kein obligatorischer Bestandteil der R-Hilfe, das heißt, Package-Autoren können ihr Paket mit einem oder mehreren diese Zusatzdokumente versehen, müssen aber nicht. Bei den Vignettes handelt es sich um PDF-Dokumente, die ein R-Package bzw. eine oder mehrere zusammenhängende Funktionen daraus näher beleuchten und dabei in der Regel nicht nur auf die praktische Verwendung der Funktionen eingehen, sondern auch in Grundzügen ihren statistischen Hintergrund beleuchten sowie die Interpretation ihrer Ergebnisse erläutern. Wenn Sie wissen wollen, welche Vignettes Ihnen durch die aktuell installierten Packages zur Verfügung stehen, können Sie dies mit `vignette(all =TRUE)` leicht überprüfen. Mit `vignette("packagename")` rufen Sie eines dieser Do-

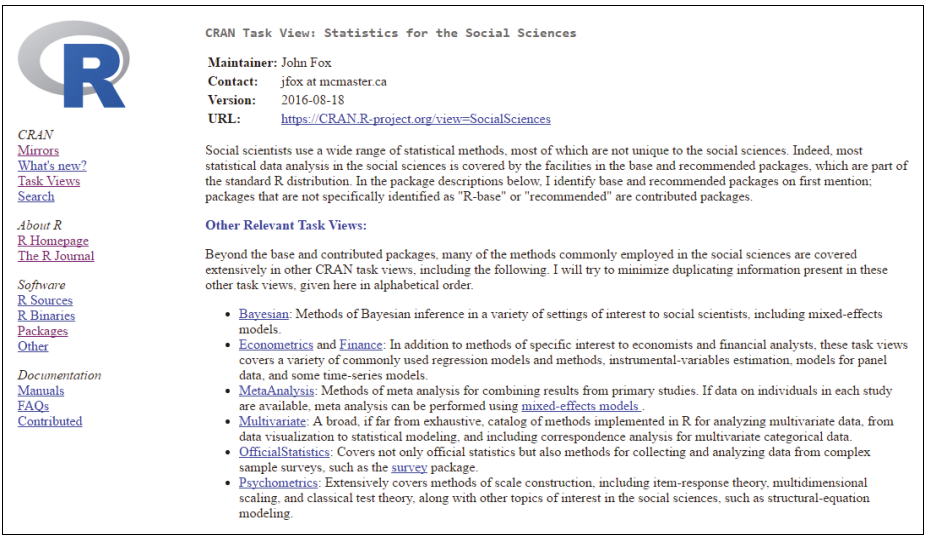
kumente auf, die dann in Ihrem PDF-Reader geöffnet werden; so können Sie sich beispielsweise mit `vignette("lmtest")` die Vignette des Packages `lmtest` anzeigen lassen, das wir später zur Diagnostik von Annahmeverletzungen des klassischen linearen Regressionsmodells benutzen werden. Der Einsatz einer Vignette setzt voraus, dass das Package, zu dem die Vignette gehört, installiert und geladen ist. Die Arbeit mit Packages schauen wir uns im folgenden Kapitel im Abschnitt »Packages verwenden« genauer an.

Viele der Vignettes sind (gekürzte) Versionen von Artikeln, die im *R Journal* (Website: *https://journal.r-project.org/*) erschienen sind. Das *R Journal* ist eine kostenlose, online publizierte Zeitschrift mit *Peer Review*, das heißt einer Qualitätssicherung der veröffentlichten Artikel durch Experten auf dem jeweiligen Gebiet. Neben diversen anderen Informationen zu R beinhaltet das *R Journal* auch regelmäßig Artikel, in denen Autoren ihre neuen Packages vorstellen. Diese Artikel, deren Inhalt sich nicht selten in den Vignettes widerspiegelt, sind insbesondere lesenswert, wenn man entscheiden möchte, ob ein bestimmtes Package für den eigenen Anwendungsfall relevant ist. Auch ordnen die Autoren ihr Package oft in den Kontext bestehender Packages ein. Dieser Vergleich kann wertvolle Hinweise auf andere interessante Packages geben. Neben dem *R Journal* ist auch das *Journal of Statistical Software* (Website: *https://www.jstatsoft.org*) zu empfehlen, das ebenfalls eine Open-Access-Zeitschrift mit Peer Review ist und in dem regelmäßig R-Pakete vorgestellt werden. Das *Journal of Statistical Software* geht zwar generell auf alle Statistikpakete ein, tatsächlich dominiert R aber seinen Inhalt deutlich, wie eine im Journal selbst erschienene Studie eindrucksvoll belegt (Lanage, Fox [2016]: *R and the Journal of Statistical Software*, Journal of Statistical Software 73 [2]).

Eine weitere Informationsquelle, die Sie anzapfen können, um festzustellen, welches Package für Ihre jeweilige Aufgabe besonders gut geeignet ist, sind die sogenannten *Task Views*. Diese finden Sie, ebenso wie das *R Journal*, auf der CRAN-Website (*https://cran.r-project.org/*) im Navigationsbereich auf der linken Seite. Tasks Views sind redaktionell bearbeitete Übersichten über die für ein bestimmtes Themengebiet relevanten R-Pakete (z. B. »Statistics for the Social Sciences«, »Econometrics« oder »Time Series«). Sie beschreiben kurz, übersichtlich und im thematischen Zusammenhang, was die dort aufgenommenen Pakete leisten. Natürlich können die Tasks Views nicht vollständig sein, allein schon weil dies eine ständige Aktualisierung voraussetzen würden, die die ehrenamtlichen Autoren nicht leisten können. Trotzdem können Task Views sehr hilfreich sein – gerade wenn man beginnt, sich mit einem Themenfeld zu befassen. Hat man ein interessant klingendes Package gefunden, kann man sich dann genauer darüber informieren.

Neben all diesen mehr oder weniger »offiziellen« Informationsquellen können Sie natürlich auf der Suche nach hilfreichen Informationen auch eines der vielen Internetforen frequentieren. Empfehlenswert ist hier unter anderem das *stackoverflow-*

Forum (Website: *http://stackoverflow.com/questions/tagged/r*). Hier tummeln sich oft auch die Autoren der R-Packages, um die sich die Diskussionen drehen, und Sie erhalten Informationen und Hilfe »aus erster Hand«. Bevor Sie selbst eine Anfrage zu einem Thema stellen, sollten Sie zunächst durch eine einfache Suche ermitteln, ob es nicht bereits einen Diskussionsthread gibt, der eine Antwort auf Ihre Frage liefern könnte. Verblüffend oft hat sich ein anderer R-Nutzer bereits mit dem gleichen Problem konfrontiert gesehen wie Sie. Weiterhin empfiehlt es sich, einmal einen Blick in die Nutzungsregeln des Forums zu werfen. So hilfsbereit die Foristen im Allgemeinen auch sind, so strikt pochen einige auf die bedingungslose Einhaltung der Regeln, und so harsch kann mitunter die Reaktion ausfallen, wenn Sie es nicht tun.



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

CRAN Task View: Statistics for the Social Sciences

Maintainer: John Fox
Contact: jfox at mcmaster.ca
Version: 2016-08-18
URL: https://CRAN.R-project.org/view=SocialSciences

Social scientists use a wide range of statistical methods, most of which are not unique to the social sciences. Indeed, most statistical data analysis in the social sciences is covered by the facilities in the base and recommended packages, which are part of the standard R distribution. In the package descriptions below, I identify base and recommended packages on first mention; packages that are not specifically identified as "R-base" or "recommended" are contributed packages.

Other Relevant Task Views:

Beyond the base and contributed packages, many of the methods commonly employed in the social sciences are covered extensively in other CRAN task views, including the following. I will try to minimize duplicating information present in these other task views, given here in alphabetical order.

- Bayesian: Methods of Bayesian inference in a variety of settings of interest to social scientists, including mixed-effects models.
- Econometrics and Finance: In addition to methods of specific interest to economists and financial analysts, these task views covers a variety of commonly used regression models and methods, instrumental-variables estimation, models for panel data, and some time-series models.
- MetaAnalysis: Methods of meta analysis for combining results from primary studies. If data on individuals in each study are available, meta analysis can be performed using mixed-effects models.
- Multivariate: A broad, if far from exhaustive, catalog of methods implemented in R for analyzing multivariate data, from data visualization to statistical modeling, and including correspondence analysis for multivariate categorical data.
- OfficialStatistics: Covers not only official statistics but also methods for collecting and analyzing data from complex sample surveys, such as the survey package.
- Psychometrics: Extensively covers methods of scale construction, including item-response theory, multidimensional scaling, and classical test theory, along with other topics of interest in the social sciences, such as structural-equation modeling.

Abbildung 1-7: Auszug aus einem Task View auf CRAN

Beispieldateien zum Download

Im Internet finden Sie unter *http://downloads.oreilly.de/9783960090441* unseren Beispieldatensatz zum Download.

Darüber hinaus können Sie alle Beispiel-R-Skripte, die wir uns im Buch ansehen, herunterladen. So müssen Sie den R-Code nicht selbst abtippen, wenn Sie die Beispiele in R nachvollziehen wollen.

KAPITEL 2

Die eigene Arbeit organisieren

In diesem Kapitel werden Sie die unterschiedlichen Arten, R zu bedienen, kennenlernen. Darüber hinaus beschäftigen wir uns damit, wie Sie Ihre Arbeit in (und außerhalb von) R möglichst effizient organisieren. Dazu zählen insbesondere ein kluger Umgang mit den eigenen R-Dateien, die Nutzung der R-Packages, die, wie Sie im ersten Kapitel bereits gesehen haben, eine große Stärke von R sind, und Vorkehrungen gegen möglichen Datenverlust.

Eingabemodi von R

In diesem Abschnitt erfahren Sie,

- was den interaktiven Modus und den Skriptmodus (auch Batchmodus) auszeichnet – die beiden unterschiedlichen Wege, wie R Befehle von Ihnen entgegennimmt,
- wann es sinnvoll ist, im interaktiven Modus zu arbeiten, und wann sich der Skriptmodus empfiehlt.

Folgende R-Funktion wird in diesem Abschnitt behandelt:

- **`history()`**: Zeigt eine Liste der zuvor ausgeführten R-Befehle an.

Sie haben bereits gesehen, dass Sie in R nicht nur mit dem Standardeditor RGui arbeiten können, der automatisch installiert wird, wenn Sie R aus dem Internet herunterladen, sondern dass Sie ebenso andere Editoren (wie beispielsweise das RStudio) einsetzen können, die typischerweise einen erweiterten Funktionsumfang bieten und komfortabler zu bedienen sind.

Unabhängig davon, ob Sie mit dem Standard-R-Editor oder mit einer anderen R-Umgebung arbeiten, können Sie R in zwei Modi betreiben, dem interaktiven und dem Skriptmodus.

Im *interaktiven* Modus geben Sie eine R-Anweisung in die Konsole ein, bestätigen Ihre Eingabe mit *Enter* und bekommen von R sofort das Ergebnis Ihrer Anweisung

angezeigt. Angenommen, Sie haben eine Variable `x` und wollen den arithmetischen Mittelwert dieser Variablen bestimmen. In R machen Sie das mit der Funktion `mean`. Wir werden später genauer auf `mean` und andere Funktionen der deskriptiven Statistik eingehen. An dieser Stelle benutzen wir `mean` nur, um die Funktionsweise der Eingabemodi von R zu demonstrieren. Die R-Konsole zeigt Ihnen durch ein Größer-Zeichen (>), den sogenannten *Prompt*, die Bereitschaft an, eine Angabe von Ihnen entgegenzunehmen. Wenn Sie früher noch mit kommandozeilenorientierten Betriebssystemen wie MS-DOS gearbeitet haben, kennen Sie die Idee eines Prompts bereits.

Wenn Sie nun `mean(x)` eingeben, teilen Sie R dadurch mit, dass Sie gern den Mittelwert der Variablen `x` sehen möchten. Die Variable `x` ist dabei einfach eine Sammlung einzelner Werte, in unserem Beispiel 1, 2, 3 und 5, vergleichbar mit dem aus der linearen Algebra bekannten Konzept des Vektors, der mehrere Elemente beinhalten kann. Diese Werte/Elemente können wir einfach unter ihrem Namen `x` ansprechen. Der Mittelwert der vier in der Variablen `x` enthaltenen Einzelwerte ist, wie man leicht von Hand nachrechnen kann, 2,75. Und genau dieses Ergebnis zeigt R uns an:

```
> mean(x)
[1] 2.75
```

Zwei Dinge sind an der Ausgabe von R bemerkenswert: Zum einen benutzt R standardmäßig den Punkt als Dezimaltrennzeichen, und zwar unabhängig davon, was Sie in Ihrem Betriebssystems als Standardtrennzeichen eingestellt haben. Wir werden auf diesen Punkt zurückkommen, wenn wir später Daten nach R einlesen. Wichtig wird das auch, wenn Sie einmal Ergebnisse aus R direkt nach Excel kopieren und dort weiterverarbeiten möchten. Zum anderen schreibt R vor seine Ausgabe eine Eins in eckigen Klammern. Diese sagt Ihnen, dass hier das erste Ergebnis kommt. Wir werden später sehen, dass dieser »Ergebniszähler« sehr hilfreich sein kann.

Übrigens: R zeigt Ihre Eingabe (einschließlich des Prompt-Zeichens >) und seine Reaktion darauf in unterschiedlichen Farben an. Das gilt typischerweise auch für R-Editoren von Drittanbietern. Welche Farben das sind, ist der Kreativität des Editoranwenders anheimgestellt. Hier im Buch werden wir statt in unterschiedlichen Farben Ihre Eingaben jeweils in `normaler Schrift` darstellen, die Ausgaben von R in **`fetter Schrift`**.

Den interaktiven Modus kann man sich also beinahe wie ein Gespräch mit R vorstellen. Sie fragen etwas, R antwortet.

Neben dem interaktiven Modus kennt R den Batch- oder Skriptmodus. Im *Skriptmodus* können Sie mehrere R-Befehle hintereinander aufschreiben und diese dann am Stück der Reihe nach ausführen lassen. R wartet also nicht nach jeder Ausführung einer Anweisung auf eine Eingabe von Ihnen, sondern führt einfach die nächste Anweisung aus, die Sie aufgeschrieben haben. Einzelne Anweisungen können Sie durch einen Zeilenumbruch oder – in R eher untypisch – durch ein Semikolon voneinander trennen.

Diese Folgen von Anweisungen, die wir hier im Buch auch als R-Skripte bezeichnen werden, können Sie in einer normalen Textdatei speichern. Diese Textdateien bekommen typischerweise die Dateiendung *.r*, damit man sie als R-Skripte identifizieren kann. Notwendig ist das aber nicht. Wenn Sie Ihre Dateien lieber auf *.rcode* enden lassen, stört R sich nicht daran.

Nun stellt sich natürlich die Frage, mit welchem der beiden Modi Sie arbeiten sollten. Die Frage lässt sich offenbar nicht eindeutig beantworten, denn gäbe es den allein glücklich machenden Eingabemodus, hätte der andere Modus wohl nicht lange überlebt. In der Tat dienen beide Modi unterschiedlichen Zwecken. Der interaktive Modus erlaubt Ihnen, Ihre Daten »direkt zu befragen«. Sie können ad hoc eine Analyse vornehmen und sehen unmittelbar deren Resultat. Sie können ausprobieren, ob eine bestimmte R-Anweisung das Ergebnis hervorruft, das Sie erwarten. Vor allem zu Beginn Ihrer Arbeit, wenn Sie sich erst mal einen Überblick über Ihren Datensatz verschaffen und Ihre Daten richtig kennenlernen wollen, sind Sie im interaktiven Modus sicherlich richtig unterwegs. Gleichwohl hat auch der Skriptmodus unbestreitbare Vorteile. Denn wenn Sie die R-Anweisungen aufschreiben und speichern, können Sie sie später erneut in R ausführen. Damit haben Sie zugleich eine Dokumentation Ihrer Verarbeitungsschritte und Analysen. Um zu verstehen, wie wichtig das ist, kann man entweder tief in die Wissenschaftstheorie greifen und Karl Poppers Forderung nach intersubjektiver Nachvollziehbarkeit wissenschaftlichen Arbeitens bemühen oder sich einfach vor Augen führen, wie viel leichter Sie sich tun werden, wenn Ihr Doktorvater, Thesis-Betreuer oder Auftraggeber danach fragt, wie Sie eigentlich zu Ihren Ergebnissen gekommen sind. Und auch Sie selbst werden irgendwann einmal froh sein, wenn Sie das vor Wochen oder Monaten zustande Gebrachte nachvollziehen können, selbst wenn Sie über ein erheblich besseres Gedächtnis verfügen als ich.

Über den reinen Dokumentationsaspekt hinaus bieten R-Skripte die Möglichkeit, bestimmte Anweisungskonstruktionen zu verwenden, die Ihnen das Leben erheblich vereinfachen können. Dazu gehören beispielweise Schleifenanweisungen und bedingte Anweisungen. Vielleicht kennen Sie diese Konzepte schon aus anderen künstlichen Computersprachen. Falls nicht, macht das überhaupt nichts. Für die »normale« Arbeit mit R benötigen Sie diese Konzepte nicht. Wenn Sie dennoch in die Welt der R-Programmierung einsteigen möchten: Im letzten Kapitel dieses Buchs werden wir uns die wichtigsten Konstrukte anschauen.

Zurück zur Frage der Eingabemodi: In der praktischen Arbeit mit R werden Sie typischerweise mit beiden Modi arbeiten. Eine bewährte Vorgehensweise ist, Anweisungen zunächst im interaktiven Modus auszuprobieren und sie, sobald sie funktionieren und das von Ihnen gewünschte Ergebnis liefern, in Ihr R-Skript zu kopieren und so zu sichern.

Übrigens: Wenn Sie sich im interaktiven Modus vertippen, brauchen Sie nicht die ganze Anweisung noch einmal einzutippen. Dankenswerterweise verfügt R über eine History-Funktion: Drücken Sie auf der Tastatur die Pfeil-Hoch-Cursortaste,

schreibt R Ihre letzte Eingabe in die Prompt-Zeile. Auf diese Weise können Sie nicht nur die letzte Eingabe »zurückholen«, sondern sich auch durch Ihre vorherigen Eingaben bewegen, sobald Sie die Pfeil-Hoch-Taste mehrfach drücken. Ebenso können Sie mit der Pfeil-Runter-Taste in der History wieder zurück in Richtung Ihrer letzten Eingabe navigieren. R-Editoren bieten zudem üblicherweise eine Möglichkeit, die komplette History in einem eigenen Fensterbereich zu sehen. RStudio erlaubt Ihnen, aus dieser History heraus Anweisungen direkt in die Konsole zu übernehmen oder Ihrem R-Skript hinzuzufügen. Im Standard-R-Editor können Sie durch Eingabe der Anweisung `history()` ebenfalls jederzeit eine Liste der zuletzt eingegebenen Befehle aufrufen, wenngleich nicht ganz so komfortabel wie in RStudio.

Packages verwenden

In diesem Abschnitt erfahren Sie,

- wie Sie mithilfe von R-Packages den Funktionsumfang von R erheblich erweitern können,
- wie Sie Packages kostenlos aus dem Internet herunterladen, installieren und laden können.

Folgende R-Funktionen werden in diesem Abschnitt behandelt:

- **`installed.packages()`**: Zeigt die installierten Packages an.
- **`install.packages()`**: Lädt ein Package aus dem Internet herunter und installiert es.
- **`library()`**: Lädt ein bereits installiertes Package, sodass seine Funktionen in R verwendet werden können.

Im ersten Kapitel des Buchs haben Sie bereits gesehen, dass eine große Stärke von R in den Packages liegt, den buchstäblich Tausenden von Erweiterungspaketen, die von R-Anwendern entwickelt worden sind und sich kostenlos aus dem *Comprehensive R Archive Network* (CRAN) herunterladen lassen. So gibt es Packages, mit denen Sie Zugang zu speziellen statistischen Funktionen erhalten (zum Beispiel `tseries` für die Analyse von Zeitreihendaten oder `sandwich` für heteroskedastie- und autokorrelationsrobuste Standardfehler), Packages, mit denen Sie attraktive, publikationsfertige Grafiken erstellen können (zum Beispiel `ggplot2`), oder Packages, die Sie bei der tabellarischen Darstellung Ihrer Ergebnisse unterstützen (zum Beispiel das Package `stargazer`, das angenehm formatierte Ergebnistabellen als Text, LaTex- oder HTML-Code produziert).

Mit Packages können Sie den Funktionsumfang von R drastisch erweitern. Wenn Sie vor einem bestimmten Problem stehen, lohnt es sich daher immer, zu recherchieren, ob es für diesen besonderen Zweck nicht bereits ein R-Package gibt, das Sie verwenden können, anstatt den entsprechenden Code mühsam selbst zu

schreiben (und zu testen). Sie werden überrascht sein, was schon alles gebrauchsfertig für Sie bereitsteht!

Da die R-Packages immer mit dem zugehörigen R-Code geliefert werden, können Sie sogar auf den Funktionen des Packages aufsetzen und diese nach Ihren Bedürfnissen verändern oder erweitern. Das ist im Übrigen vollkommen legal, denn eine der Grundfreiheiten, die Sie bei Verwendung freier Software wie R genießen, ist es, den Quellcode nach eigenen Wünschen modifizieren zu dürfen. Typischerweise werden Sie dabei allerdings Kenntnisse der richtigen R-Programmierung benötigen, das heißt der in R verwendeten Kontrollstrukturen, wie Wenn-Dann-Bedingungen und Wiederholungsschleifen. Diese Kontrollstrukturen schauen wir uns in Kapitel 9 etwas genauer an.

Mit der Erstinstallation von R haben Sie, ohne es zu bemerken, bereits eine Reihe von Packages mitinstalliert, die Grundfunktionalitäten von R sowie häufig verwendete Statistik- und Grafikfunktionen bereitstellen. Mit dem R-Befehl `installed.packages()` können Sie sich eine Übersicht über die aktuell installierten R-Pakete anzeigen lassen. Vergessen Sie bei der Eingabe des Befehls die Klammern nicht! Ohne die Klammern wird Ihnen nicht das Ergebnis der Ausführung des Befehls, hier also die Liste der Packages, angezeigt, sondern der R-Code der Funktion selbst, also derjenige Programmquelltext, der ausgeführt wird, wenn Sie die Funktion aufrufen. Der Output von `installed.packages()` ist eine breite Tabelle, deren Spalten in Ihrer R-Konsole möglicherweise nicht alle nebeneinander dargestellt werden können. Wenn Sie RStudio verwenden (funktioniert in RGui leider nicht), können Sie mit `View(installed.packages())` eine angenehmere Darstellung der Ergebnisse erreichen.

Die Liste der installierten Packages enthält neben dem Namen noch eine Reihe weiterer Informationen zu jedem Paket, unter anderem wo genau das Package physisch auf Ihrem Laufwerk liegt (`LibPath`) und von welchen anderen Paketen dieses Package abhängt (`Depends` und `Imports`; der Unterschied zwischen beiden ist eher technischer Natur und soll uns hier nicht weiter interessieren). Diese Abhängigkeiten entstehen, weil die R-Packages ihrerseits wiederum auf Funktionen aus anderen Paketen zugreifen. Oft werden Sie unter `Depends` eine Angabe wie `R (>= 3.2.0)` sehen. Das bedeutet, dass Sie, um dieses Package nutzen zu können, mindestens Version 3.2.0 von R installiert haben müssen. Dies ist, wie bereits im ersten Kapitel erwähnt, eine der wenigen Gelegenheiten, bei denen es wirklich relevant ist, welche R-Version genau Sie benutzen. Manchmal sind auch andere R-Packages, von denen das installierte Paket abhängt, mit einer Minimum-Version versehen.

Bequemer ist die Anzeige der installierten Packages mit RStudio. Hier bekommen Sie im Register *Packages* (Abbildung 2-1) eine Package-Liste präsentiert, in der Ihnen für jedes Paket anhand des kleinen Häkchens vor seinem Namen auch direkt angezeigt wird, ob das Package aktuell geladen ist oder nicht. Denn Packages sind nach der Installation nicht einfach verfügbar, sondern müssen vor Gebrauch gela-

den werden. Das geschieht in R durch den Befehl `library(packagename)`, wobei `packagename` der Name des betreffenden Pakets ist. In RStudio können Sie auch einfach das kleine Häkchen vor das Paket setzen, um es zu laden. Wenn Sie RGui verwenden, erlaubt Ihnen die Option *Load Package* im Menü *Package*, das zu ladende Paket auch über ein Dialogfenster auszuwählen.

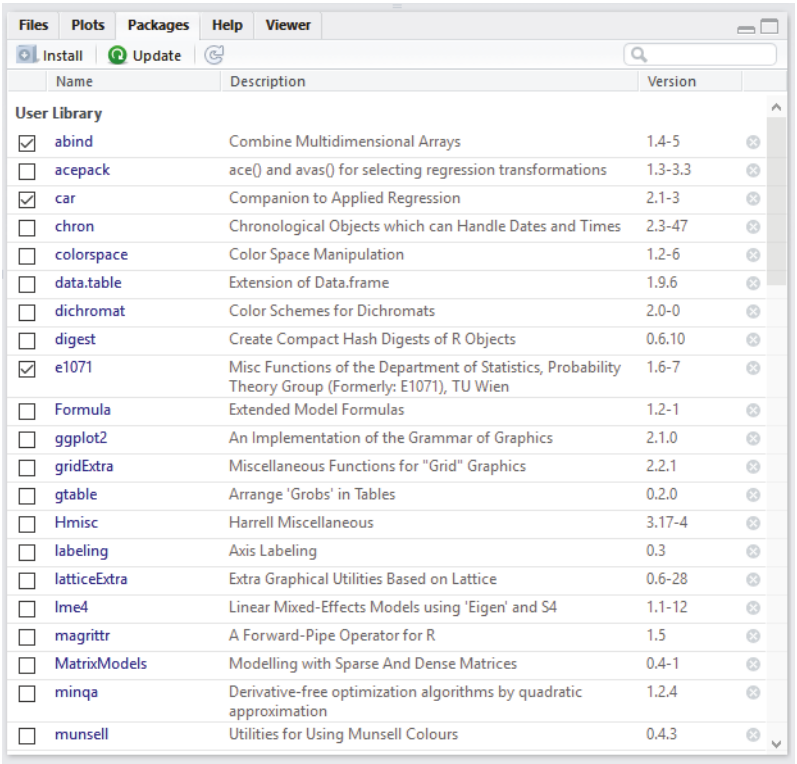


Abbildung 2-1: Liste der Packages in RStudio

Haben Sie ein interessantes Package gefunden, das Sie installieren möchten (zum Beispiel in einem Internetforum, im *R Journal* oder über die *CRAN Task Views*), wählen Sie in RGui aus dem Menü *Packages* die Option *Install package(s)*. Es erscheint ein schlichter Auswahldialog, mit dessen Hilfe Sie ein oder mehrere Packages auswählen und dann installieren können. R lädt das Package selbst sowie darüber hinaus diejenigen Packages, von denen es abhängig ist, selbstständig aus dem Internet herunter und entpackt die komprimierten Dateien. Danach können Sie das Paket mit `library(packagename)` laden.

In RStudio können Sie die Installation mithilfe der Option *Install Packages* im Menü *Tools* einleiten. In dem sich dann öffnenden Dialogfenster können Sie entscheiden, ob Sie das Package von CRAN herunterladen möchten oder ob Sie ein Package installieren möchten, das Sie bereits, zum Beispiel direkt über die CRAN-

Website, auf Ihr Laufwerk heruntergeladen haben (das funktioniert im Übrigen über die Option *Install package(s) from local files* im Menü *Packages* auch in RGui). RStudio lässt Sie außerdem entscheiden, ob Sie die Pakete, von denen Ihr zu installierendes Package abhängt, mitinstallieren wollen, was generell zu empfehlen ist (daher macht RGui das, ohne Sie vorher um Erlaubnis zu bitten). Im RStudio-Dialogfenster müssen Sie den Namen des Packages selbst eingeben, aber es öffnet sich, sobald Sie ein paar Buchstaben eingetippt haben, sofort eine Autovervollständigungsliste, aus der Sie dann Ihr Package auswählen können.

Wenn Ihnen die Auswahl über das Dialogfenster zu mühselig ist, können Sie R genauso gut mit einem Befehl anweisen, Ihr Paket zu installieren: `install.packages("packagename")`. Beachten Sie bitte, dass der Name des Packages dabei in Anführungszeichen stehen muss.

Wir verwenden im Laufe des Buchs verschiedene Packages. In den Überblicksdarstellungen, die Sie am Anfang vieler Abschnitte finden, werden Ihnen unter anderem die in diesem Abschnitt hauptsächlich verwendeten R-Funktionen vorgestellt. Wenn die Funktionen aus Packages stammen, die nicht standardmäßig in R installiert sind und beim Start von R geladen werden, finden Sie jeweils hinter dem Namen der Funktion in geschweiften Klammern den Namen des Packages, zu dem die Funktion gehört. Das sieht dann zum Beispiel so aus: `gqtest()` {Package `lmtest`}.

Noch ein abschließender Hinweis: Manchmal bitten die Entwickler der Packages in der Dokumentation ihres Packages oder durch eine Meldung beim Laden des Packages darum, zitiert zu werden, wenn man das Package verwendet. Das sollten Sie tun! Die Autoren haben viel Zeit und Mühe in die Entwicklung ihrer Packages investiert und Ihnen dadurch bei Ihrer eigenen Arbeit entsprechend Zeit erspart. Sie sollten das honorieren, indem Sie in Ihren Publikationen durch eine Zitation darauf hinweisen, dass Sie das entsprechende Package benutzt haben. Die Package-Entwickler werden es Ihnen danken.

Ein Arbeitsverzeichnis aufbauen

In diesem Abschnitt erfahren Sie,

- wie Sie Ihre Dateien für die Arbeit mit R effektiv organisieren können,
- was das Arbeitsverzeichnis in R ist und wie Sie R mitteilen können, welches Ihr Arbeitsverzeichnis ist.

Folgende R-Funktion wird in diesem Abschnitt behandelt:

- **`setwd`**`()`: Setzt das Arbeitsverzeichnis.

Die eigene Arbeit will gut organisiert sein. Während Ihres Projekts werden sich zahlreiche Dateien ansammeln, ganz gleich ob Sie nun eine Seminararbeit, Ihre

Bachelor- oder Master-Thesis oder gar Ihre Promotion in Angriff nehmen. Datendateien, Literatur, R-Skripte, Notizen, die von Ihnen verfassten Texte – all das sollten Sie gut strukturiert ablegen. Denn auch wenn es zu Beginn Ihrer Arbeit noch nicht danach aussehen mag, unterschätzen Sie nie, wie schnell eine Unmenge unterschiedlicher Dokumente zusammenkommt. Daher lohnt es sich, sich *direkt am Anfang* der eigenen Bemühungen Gedanken darüber zu machen, wie man seine Dateien strukturiert ablegen möchte. Viel schwieriger ist es dagegen, später, wenn man in der riesigen Menge an Dokumenten vollends den Überblick zu verlieren droht, damit zu beginnen, die Unterlagen in eine Struktur zu bringen.

Gute Ordnerstrukturen können sehr unterschiedlich aussehen und natürlich auch stark von der Art und gegebenenfalls sogar vom Fachbereich des Vorhabens abhängen. Den *einen* Königsweg gibt es sicher nicht. Abbildung 2-2 zeigt daher eher exemplarisch, wie eine Verzeichnisstruktur aussehen kann, in der man seine Dateien nicht lange suchen muss.

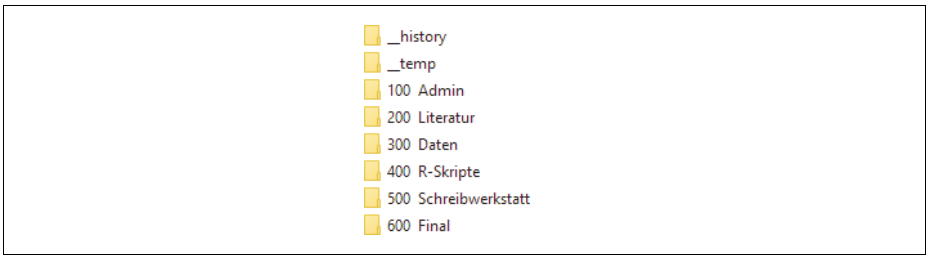


Abbildung 2-2: Beispiel für eine Verzeichnisstruktur

Das Verzeichnis *100 Admin* kann dazu verwendet werden, administrative Dokumente abzulegen, die keinen inhaltlichen Bezug zu Ihrer Arbeit haben (zum Beispiel den Antrag auf Zulassung der Arbeit oder, falls man einen solchen Plan pflegt, den »Projektplan« des Vorhabens). Die Verzeichnisse *200 Literatur*, *300 Daten* und *400 R-Skripte* enthalten, was ihr Name suggeriert, während das Verzeichnis *500 Schreibwerkstatt* dasjenige ist, in dem die eigentlichen Texte der Arbeit entstehen. Im Verzeichnis *600 Final* können schließlich die fertigen Versionen der wesentlichen Dateien abgelegt werden. Das ist sehr hilfreich, wenn man mit einigem zeitlichen Abstand auf seine damaligen Ergebnisse zurückgreifen möchte.

Ihnen ist vielleicht aufgefallen, dass es noch zwei weitere Verzeichnisse gibt, die bezüglich der Namungskonvention ein wenig von den übrigen abweichen: *__history* und *__temp*. Das Verzeichnis *__history* ist dazu gedacht, alte Dateiversionen aufzunehmen, die man zwar eigentlich nicht mehr braucht, aber dennoch nicht wegwerfen möchte. Das kann einerseits interessant sein als eine Art Backup für den Fall, dass an der aktuellen Arbeitsdatei technisch oder inhaltlich etwas »beschädigt« wird und man deshalb auf einen älteren Stand zurückgehen möchte. Oder aber die »alten« Dateien enthalten etwas, das Sie zwar nicht in die finale Version Ihrer Arbeit aufnehmen wollen, das Sie aber behalten möchten, weil es gegebenenfalls für an-

dere Vorhaben noch von Interesse sein könnte. Das History-Verzeichnis wird sich natürlich während Ihrer Arbeit sehr rasch füllen. Deshalb ist es empfehlenswert, ein solches Verzeichnis nicht nur auf der höchsten Ebene Ihrer Verzeichnisstruktur, sondern auch auf darunterliegenden Ebenen anzulegen. So kann es zum Beispiel innerhalb des Verzeichnisses *500 Schreibwerkstatt* ein Unterverzeichnis *__history* geben, das die Zwischenstände Ihres Texts aufnimmt.

Das zweite Verzeichnis, das etwas aus der Reihe fällt, ist *__temp*. Diesen Ordner können Sie dazu benutzen, Dateien vorläufig – eben »temporär« – abzulegen, deren Nutzen Sie noch nicht genau einschätzen können. Sie haben ein Dokument aus dem Internet heruntergeladen, sind sich aber nicht sicher, ob es wirklich für Ihr Thema relevant ist? Sie haben einige Nebenrechnungen in Excel gemacht, die Sie voraussichtlich nicht behalten wollen? Das Verzeichnis *__temp* ist ein guter Ort, solche Dateien abzulegen. Dort sollten die Dateien aber nicht für immer bleiben. Prüfen Sie regelmäßig den Inhalt Ihres Temp-Verzeichnisses und sortieren Sie die Dateien entweder in Ihre Verzeichnisstruktur ein, wenn Sie sie behalten wollen, oder löschen Sie sie, falls das nicht der Fall ist.

Ihnen ist sicher aufgefallen, dass die Namen der meisten Verzeichnisse in der vorgeschlagenen Ordnerstruktur mit einer dreistellen Zahl beginnen. Das hat den Vorteil, dass Sie Ihre Verzeichnisse so in eine logische Reihenfolge bringen können, die von der alphabetischen Reihenfolge abweichen kann. Auch auf Ebene der Unterverzeichnisse können Sie diese Art der logischen Strukturierung fortzusetzen. So könnten Sie zum Beispiel innerhalb des Verzeichnis *300 Daten* weitere Unterordner anlegen wie etwa *310 Rohdaten IWF* und *320 Rohdaten Weltbank*, und können auch diese Verzeichnisse wiederum über die Zehnerstellen der Nummerierung in eine logische Reihenfolge bringen, während durch die führende »3«, die Hunderterstelle, sofort klar ist, dass diese Unterordner zum Verzeichnis *300 Daten* gehören.

Warum sollten Sie sich so intensiv damit beschäftigen, wie Sie Ihre Dateien organisieren? Nun, zum einen ist natürlich eine saubere Ablage von Dateien eine Voraussetzung für eine strukturierte Arbeit. Zum anderen ist es aber auch für R relevant, wo Ihre Dateien liegen.

Ein R-Projekt besteht typischerweise aus mindestens einem R-Skript sowie wenigstens einer Datendatei. Oftmals bleibt es aber nicht dabei. Sie werden vielleicht mehrere Datendateien in R laden und möglicherweise auch Ihr R-Skript auf mehrere Dateien aufteilen. Wie das geht, schauen wir uns im nächsten Abschnitt an.

In all diesen Fällen muss R wissen, wo es nach Ihren Dateien suchen soll. Standardmäßig versucht R, eine Datei, die Sie in ein R-Skript einbinden, im gleichen Verzeichnis zu finden, in dem das R-Skript selbst liegt. Befindet sich die Datei tatsächlich aber in einem anderen Verzeichnis, können Sie beim Laden der Datei deren genauen Pfad angeben. Das ist in vielen Fällen jedoch etwas mühselig. Stellen Sie sich vor, Sie wollten aus Ihrem im Verzeichnis *400 R-Skripte* liegenden R-Skript zehn Dateien laden, die alle im Verzeichnis *300 Daten* liegen. In diesem

Fall ergibt es Sinn, das Datenverzeichnis als *Arbeitsverzeichnis* zu deklarieren. Das geschieht mit dem Befehl `setwd`, der für *set working directory* steht:

```
> setwd("c:/users/mustermann/Masterarbeit/300  Daten")
```

Damit teilen Sie R mit, wo Ihr Arbeitsverzeichnis liegt und wo R zukünftig immer dann, wenn kein anderer Pfad angegeben ist, Ihre Dateien suchen soll.

Beachten Sie die Schreibweise: Der Pfad muss in Anführungszeichen gesetzt werden, damit R ihn als zusammenhängende Zeichenkette erkennt, und Sie müssen den normalen Schrägstrich (`/`) zur Trennung der einzelnen Pfadelemente verwenden. Wenn Sie mit Linux oder MacOS als Betriebssystem arbeiten, kommt das für Sie wenig überraschend, von Windows-Benutzern, die an dieser Stelle den Backslash (`\`) gewohnt sind, wird es aber gern übersehen. Schauen wir einmal, was passiert, wenn wir irrtümlich den Backslash verwenden:

```
> setwd("c:\users\mustermann\Masterarbeit\300  Daten")
Error: '\u' used without hex digits in character string starting ""C:\u"
```

Diese Fehlermeldung wirkt etwas kryptisch. Offenbar glaubt R, Sie würden hier mit hexadezimalen Zahlen arbeiten wollen. Das klingt merkwürdig angesichts der Tatsache, dass Sie ja eigentlich R nur Ihr Arbeitsverzeichnis mitteilen wollten. Warum R Ihre Eingabe so interpretiert, werden wir an etwas späterer Stelle besser verstehen. Hier sei nur gesagt, dass die Fehlermeldung unterschiedlich ausfällt, je nachdem, welcher Buchstabe nach dem ersten `\` folgt.

Typischerweise deklariert man sein Arbeitsverzeichnis zu Beginn eines R-Skripts. Sie können aber, wenn es sinnvoll ist, Ihr Arbeitsverzeichnis auch mehrfach im Skript wechseln. Beachten Sie bitte, dass die Beispielskripte, die Sie von der Website des Buchs herunterladen können, keine `setwd`-Anweisungen enthalten, damit der R-Code auch auf Ihrem Computer direkt lauffähig ist, ohne dass Sie das in `setwd` angegebene Verzeichnis ändern müssen. Das bedeutet also, R sucht alle Dateien im dem Verzeichnis, in dem Sie auch das R-Skript selbst abgelegt haben.

Arbeitsstände sichern und wiederherstellen

In diesem Abschnitt erfahren Sie,

- wie Sie R-Objekte (zum Beispiel Datensätze und einzelne Variablen) dauerhaft speichern und wiederherstellen können, um später mit ihnen weiterzuarbeiten.

Folgende R-Funktionen werden in diesem Abschnitt behandelt:

- **`save`**`()`: Speichert die angegebenen R-Objekte in einer Datei.
- **`save.image`**`()`: Speichert alle aktuell existierenden R-Objekte in einer Datei und erzeugt so eine gespeicherte Abbildung der aktuellen Arbeitsumgebung.

- **`load()`**: Lädt ein oder mehrere zuvor mit `save` oder `save.image` gespeicherte R-Objekte nach R und macht sie so zur Weiterarbeit verfügbar.

Manchmal werden Sie den aktuellen Zustand von R einfach »einfrieren«, R schließen und später an derselben Stelle wieder weiterarbeiten wollen, an der Sie zuletzt aufgehört haben. Oder Sie möchten aus Gründen der Sicherheit oder Bequemlichkeit einmal einige bestimmte oder sogar alle R-Objekte wie etwa Datensätze und Variablen speichern, etwa weil Sie eine komplizierte und langwierige Berechnung durchgeführt haben und das Ergebnis dieser Berechnung beim nächsten Mal einfach direkt laden wollen, anstatt die zeitraubende Berechnung noch einmal durchführen. R enthält Funktionen, um den aktuellen Arbeitsstand ganz oder teilweise zu sichern.

Sie werden an späterer Stelle spezielle Funktionen kennenlernen, mit denen Sie R-Datensätze in Dateien zurückschreiben können. Manche R-Objekte, die wir noch kennenlernen werden, sind aber erheblich komplexer aufgebaut als Datensätze, die ja einfach eine Zeilen-Spalten-Struktur darstellen. Das Ergebnis einer linearen Regression zum Beispiel ist in R ebenfalls ein Objekt. Es beinhaltet die Koeffizienten, deren Standardfehler und p-Werte, darüber hinaus R^2, das Ergebnis des F-Tests auf gemeinsame Signifikanz aller Koeffizienten und eine ganze Reihe weiterer Informationen. Dementsprechend hat dieses Objekt natürlich eine komplexere Struktur und kann nicht mehr als Tabelle dargestellt werden. Um ein solches Objekt dauerhaft zu speichern, sind die in diesem Abschnitt vorgestellten Funktionen unerlässlich.

Mit der Anweisung `save.image(dateiname)` speichert R alle Datensätze und sonstigen Objekte Ihrer aktuellen R-Sitzung (also eine komplette Abbildung, ein *Image*) in die Datei `dateiname`, und zwar in das zuvor mit `setwd` gesetzte Arbeitsverzeichnis, es sei denn, `dateiname` beinhaltet eine komplette Pfadbeschreibung. Damit wird Ihre Arbeitsumgebung, Ihr Workspace, gesichert.

Wenn Sie das Verzeichnis, in das Sie das Image schreiben wollen, explizit in `dateiname` angeben, beachten Sie bitte, dass, wie auch schon zuvor im Fall der Funktion `setwd`, die einzelnen Ordnerhierarchieebenen nicht mit dem unter Windows üblichen Backslash (`\`), sondern mit `/` zu trennen sind, Sie also statt

```
save.image("C:\MeineDaten\sicherung.rdata")
```

schreiben müssen:

```
save.image("C:/MeineDaten/sicherung.rdata").
```

Wie Sie die Datei benennen, ist Ihnen überlassen. Standardmäßig nennt R die Datei, wenn Sie keine Dateinamen angeben und einfach nur `save.image()` aufrufen, `.RData`. Das mutet, zumindest für den Windows-Nutzer, schon ein wenig seltsam an, deshalb vergeben Sie am besten einfach einen aussagekräftigen Dateinamen Ihrer Wahl.