

Erste Einführung in R

Johann Popp

2020-07-28

R¹ ist eine Programmiersprache und -umgebung für statistische Berechnungen. Es ist es wahrscheinlich etwas anders als Computerprogramme, die Sie bisher kennen gelernt haben. Es wirkt sehr technisch und grafisch sehr einfach. Lassen Sie sich davon nicht täuschen. R ist wahnsinnig flexibel und Sie können damit recht einfach genau die statistischen Berechnungen durchführen und grafisch darstellen, die Sie wollen.

Um das richtig zu genießen ...

- fragen Sie nicht “Was kann R?”, sondern “Was möchte ich?” und im Anschluss “Wie kann ich R dazu bringen, das zu tun?”,
- dokumentieren Sie Ihr Vorgehen kleinteilig und
- nutzen Sie RStudio², um mit R zu arbeiten.

R ist sehr leicht erweiterbar. Zusätzlich zur Basis-Version sind mehrere tausend Zusatzpakete für spezielle Anwendungen veröffentlicht. Durch die große Gruppe von Anwendern, die ihre Spezialpakete zur Verfügung stellen, werden auch neue Methoden sehr schnell in R implementiert. Falls noch kein Paket für eine bestimmte Prozedur vorhanden ist, kann man das relativ einfach selbst programmieren.

R und RStudio installieren

Installieren Sie die Programme R und RStudio. Beides ist freie und quelloffene Software, die auf Windows-, Macintosh- und Linux-Betriebssystemen läuft.

- R wird unter <http://cran.r-project.org> bereit gestellt. Laden Sie die Installationsdateien für Ihr Betriebssystem herunter und folgen Sie den entsprechenden Installationsanweisungen.
- RStudio finden Sie unter <http://www.rstudio.com/ide/download/desktop>. Auch hier wählen Sie die für Ihr Betriebssystem passende Version aus. Der Rest sollte selbsterklärend sein.

¹ R Core Team: „R: A Language and Environment for Statistical Computing“ Vienna, Austria: R Foundation for Statistical Computing, 2018, <https://www.R-project.org/>.

² RStudio Team: „RStudio: Integrated Development Environment for R“ Boston, MA: RStudio, Inc., 2012, <http://www.rstudio.com/>.

R: `<http://cran.r-project.org>`

RStudio:
`<http://www.rstudio.com/ide/download/desktop>`

Arbeiten mit R und RStudio

R hat je nach Betriebssystem nur eine sehr eingeschränkte oder sogar gar keine grafische Benutzeroberfläche. Diese Anleitung nutzt die Benutzeroberfläche RStudio, um komfortabel mit R arbeiten zu können. Öffnen Sie also zunächst einmal RStudio.

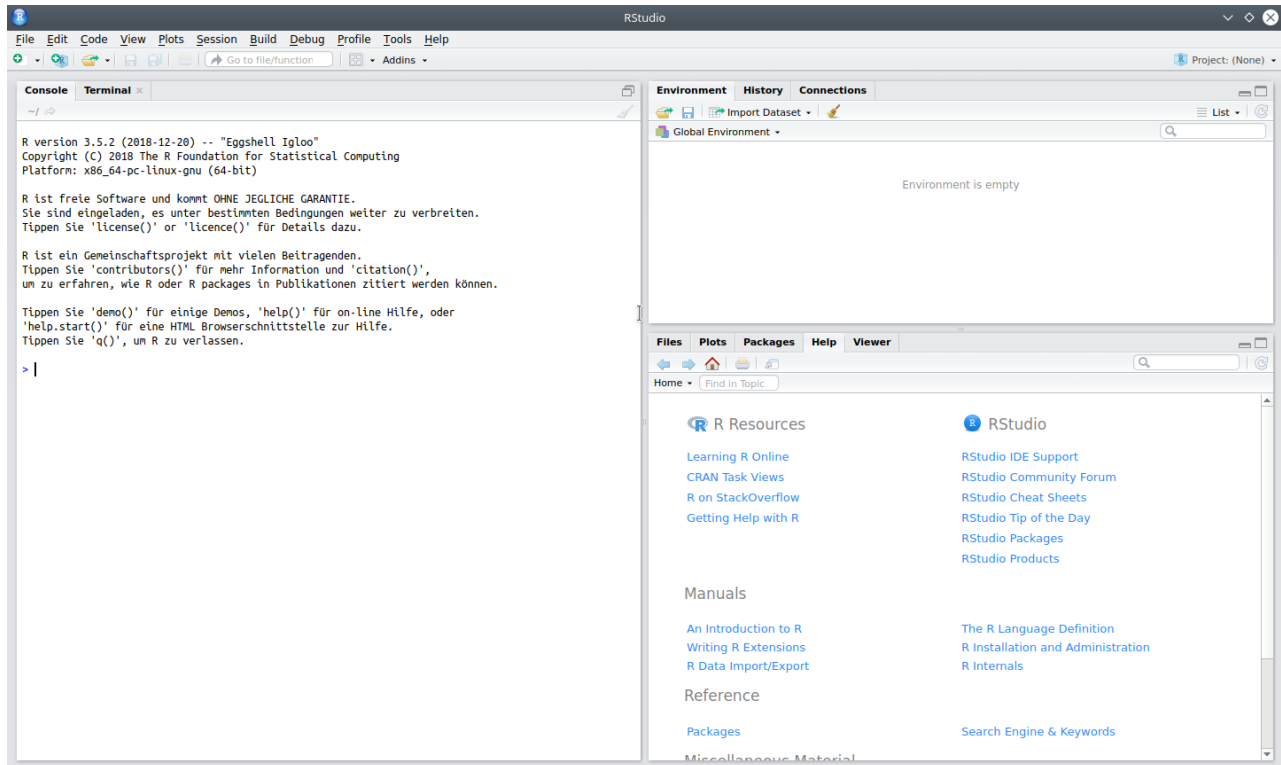


Abbildung 1: RStudio Startbildschirm

Das Programm teilt den Bildschirm in drei (manchmal auch vier) Fenster auf. Das linke (ggf. untere) Fenster ist unter dem Reiter *Console* für R reserviert. Alle anderen Fenster enthalten hilfreiches Zubehör von RStudio.

Zunächst wird Ihnen angezeigt, welche R-Version Sie installiert haben. Für diese Einführung habe ich z.B. die Version 3.5.2 in einer 64 Bit Linux-Version verwendet. Es wird ihnen außerdem mitgeteilt, dass R eine freie Software ist und dass Sie mehr über die Lizenz erfahren können, wenn Sie `licence()` eingeben. Tippen Sie doch einmal `licence()` und drücken Sie dann die Eingabetaste. Sie sollten jetzt die folgende Anzeige ausgegeben bekommen:

```
licence()
```

R läuft links (unten) in der Console

`licence()` erzeugt eine Ausgabe der R-Lizenz

```
##
## This software is distributed under the terms of the GNU General
## Public License, either Version 2, June 1991 or Version 3, June 2007.
## The terms of version 2 of the license are in a file called COPYING
## which you should have received with
## this software and which can be displayed by RShowDoc("COPYING").
## Version 3 of the license can be displayed by RShowDoc("GPL-3").
##
## Copies of both versions 2 and 3 of the license can be found
## at https://www.R-project.org/Licenses/.
##
## A small number of files (the API header files listed in
## R_DOC_DIR/COPYRIGHTS) are distributed under the
## LESSER GNU GENERAL PUBLIC LICENSE, version 2.1 or later.
## This can be displayed by RShowDoc("LGPL-2.1"),
## or obtained at the URI given.
## Version 3 of the license can be displayed by RShowDoc("LGPL-3").
##
## 'Share and Enjoy.'
```

Das ist ein typisches Verhalten von R: Sie geben einen Befehl ein, drücken die Eingabetaste und bekommen eine Ausgabe.

Arbeiten mit Projekten

Bevor Sie sich R genauer ansehen, legen Sie aber zunächst einmal ein *Projekt* an. Das ist wichtig, um später all ihre Daten und Berechnungen einfach wieder zu finden. Klicken Sie im Menü *File* auf *New Project*

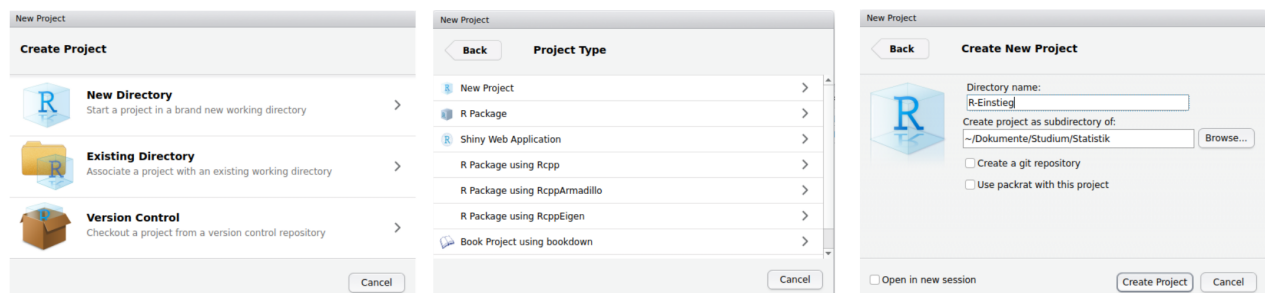


Abbildung 2: Dialogfelder zur Erstellung eines neuen Projektes

Im folgenden Dialogfeld wählen Sie zuerst *New Directory*, dann *New Project* und geben Sie schließlich den Ordner an, in dem Sie Ihr Projekt speichern wollen, z.B. “~/Dokumente/Studium/Statistik/R Ein-

stieg". In diesem Ordner werden nun alle Daten, Skripte etc. abgelegt, die Sie im weiteren Verlauf erstellen.

Einfache Berechnungen mit R

Probieren Sie nun Ihre erste statistische Berechnung in R aus, indem Sie den Mittelwert aus den vier Altersangaben 24, 54, 19 und 30 berechnen:

```
(24 + 54 + 19 + 30) / 4
```

```
## [1] 31.75
```

Die obere Zeile zeigt Ihre Eingabe, die Untere Zeile das Ergebnis (31.75)³.

Um die Daten nicht immer wieder neu einzugeben, kann man sie einem sogenannten *Objekt* zuweisen.

```
alter <- c(24, 54, 19, 30)
```

Mit `c()` haben Sie die Altersangaben aneinander gehängt und mit `<-` (kleiner - minus) dem neu erstellten Objekt `alter` zugewiesen. Zunächst passiert nichts weiter. Jedesmal, wenn Sie jetzt aber `alter` tippen und die Eingabetaste drücken, erhalten Sie den Inhalt dieses Objekts.

```
alter
```

```
## [1] 24 54 19 30
```

Nach dem Muster `Name <- Inhalt` können Sie beliebig viele Objekte definieren und im sogenannten *Workspace* ablegen. Mit dem Befehl `ls()` können Sie sich alle verfügbaren Objekte im Workspace anzeigen lassen.

```
ls()
```

```
## [1] "alter"
```

Außerdem werden im rechten oberen Fenster die verfügbaren Objekte auch unter dem Reiter *Environment* angezeigt. In der linken Spalte steht jeweils der Name des Objektes und in der rechten Spalte Informationen zu deren Inhalt.

R als Taschenrechner

³ Beachten Sie, dass R einen Punkt als Dezimaltrenner benutzt.

Daten mit `c()` zusammenführen und mit `<-` einem *Objekt* zuweisen

`ls()` zeigt verfügbare Objekte an

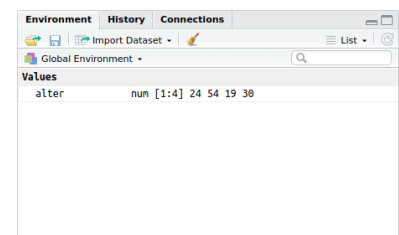


Abbildung 3: Anzeige der verfügbaren Objekte im Environment

AUS DEN VORHANDENEN ALTERSDATEN kann man nun zuerst die Summe bilden, ...

```
Summe <- sum(alter)
```

`sum()` berechnet die Summe

... dann die Stichprobengröße bestimmen ...

```
N <- length(alter)
```

`length()`: Anzahl der Elemente eines Objektes

... und schließlich beides durch einander teilen.

```
Summe / N
```

```
## [1] 31.75
```

Natürlich hat R auch einen eigenen Befehl für den arithmetischen Mittelwert. Er heißt `mean()`.

`mean()` berechnet den arithmetischen Mittelwert

```
mean(alter)
```

```
## [1] 31.75
```

SIE HABEN JETZT SCHON EINIGE BEFEHLE von R kennen gelernt: `licence()`, `c()`, `sum()`, `length()` und `mean()`. All diese Befehle bestehen aus einem Namen, gefolgt von weiteren Angaben in Klammern.

Struktur von Befehlen in R

Befehle werden in R *Funktionen* genannt und die Angaben, die in den Klammern folgen, heißen *Argumente*. Die allermeisten Funktionen beinhalten als erstes Argument die Daten, auf die die Funktion angewendet werden soll. Weitere Argumente können (durch Kommata getrennt) folgen. Jede vorgefertigte Funktion hat eine Hilfeseite, auf der genau erklärt wird, wie sie funktioniert.

Hilfe finden

Zu jeder Funktion gibt es einen Hilfstext, den Sie mit einem voran gestellten `?` aufrufen können. Geben Sie z.B. einmal `?mean` ein.

Im rechten unteren Fenster wird die entsprechende Hilfeseite aufgerufen. Diese Seiten sind immer gleich aufgebaut: Zuerst gibt es unter **Description** eine kurze Beschreibung, was die Funktion macht. Unter **Usage** sehen Sie, welche Eingaben erwartet werden und was

Hilfe mit `?` aufrufen

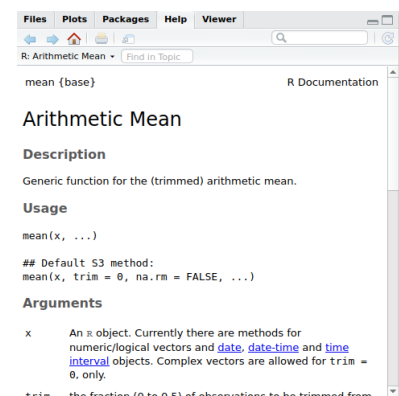


Abbildung 4: Ausschnitt aus der Hilfeseite für die Funktion `mean()`

die Standardeinstellungen sind. Im Abschnitt **Arguments** werden alle Argumente beschrieben, die die Funktion enthält. Diese Argumente werden auf manchen Hilfseiten unter dem Stichwort **Details** ausführlicher beschrieben. Unter **Value** wird beschrieben, welche Ausgaben erzeugt werden. **References** nennt Literaturangaben, auf die sich die Funktion bezieht. Unter **See Also** werden verwandte Funktionen angezeigt. Schließlich sind die Beispiele unter **Examples** oft sehr hilfreich, um zu verstehen, wie die Funktion verwendet wird.

Für die Funktion `mean()` muss man also mindestens angeben, auf welche Daten (`x`) sie angewendet werden soll. Das Argument `trim = 0` legt fest, dass die Daten nicht getrimmt werden. (Es werden keine Extremwerte ausgeschlossen.) Das Argument `na.rm = FALSE` legt fest, dass fehlende Werte (die werden in R als `NA` gekennzeichnet) nicht ausgeschlossen werden, bevor der Mittelwert berechnet wird. Probieren Sie einmal aus, was das bedeutet.

```
mean(c(24, 54, NA, 30))
```

```
## [1] NA
```

R kann nicht berechnen, was der Mittelwert aller vier Werte ist, weil für einen dieser Werte keine Angabe vorliegt. Das Ergebnis ist somit wieder ein fehlender Wert (`NA`).

Wenn man das Argument `na.rm` auf `TRUE` setzt, wird der fehlende Wert vor der Berechnung ausgeschlossen und der Mittelwert aus den verbleibenden drei Werten berechnet.

```
mean(c(24, 54, NA, 30), na.rm = TRUE)
```

```
## [1] 36
```

WENN SIE NICHT MEHR GENAU WISSEN, wie eine Funktion geschrieben wird, können Sie die ersten Buchstaben tippen und sehen, was RStudio Ihnen als Vervollständigung vorschlägt.

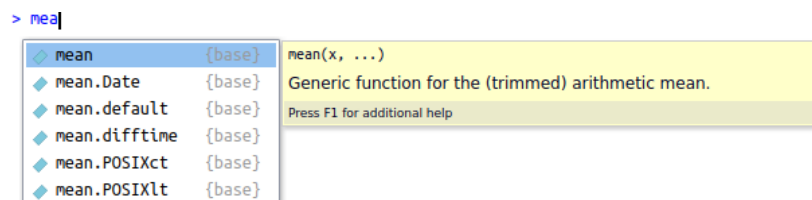


Abbildung 5: Automatische Vervollständigungsfunktion von RStudio

Wenn Sie damit nicht weiter kommen, können Sie gesuchte Inhalte auch mit zwei vorangestellten Fragezeichen eingeben. Es werden dann alle geladenen Hilfe-Seiten nach diesem Begriff durchsucht. Wenn Sie z.B. Hilfe zur Spearman-Korrelation suchen, geben Sie `??Spearman` ein. Es wird Ihnen dann angezeigt, dass es eine Funktion `cor.test()` gibt, die Korrelationen testet - unter anderem auch nach der Spearman-Methode.

Wenn Sie innerhalb von R die benötigte Hilfe nicht bekommen, finden Sie sie bestimmt im Internet. Geben Sie Ihre Frage nach dem Muster “r Inhalt” in Ihre Suchmaschine ein. Ihre Frage wurde bestimmt schon einmal gestellt und kompetent beantwortet.

Daten eingeben

Sie haben bereits eine Methode kennen gelernt, Daten einzugeben nämlich, in dem Sie sie mit `c()` aneinander hängen und mit `<-` einem Objekt zuweisen. Das ist natürlich nur für kleine Datenmengen geeignet.

NORMALERWEISE WERDEN STATISTISCHE DATENSÄTZE in Form einer Datenmatrix angelegt, bei der jede Zeile einer statistischen Einheit (z.B. einer befragten Person) entspricht und jede Spalte einer Variable. In R ist so eine Datenmatrix ein `data.frame()`.

Erstellen Sie erst einmal einen leeren `data.frame()` und weisen Sie ihm dem Objekt `dat` zu, ...

```
dat <- data.frame()
```

... öffnen Sie mit `fix()` die Datenmatrix und geben Sie die Daten wie in Abbildung 6 ein.

```
fix(dat)
```

Schließen Sie nun den Data Editor (mit *Quit*) und betrachten Sie dann das Ergebnis in R:

```
dat
```

```
##      Alter Größe Geschlecht
## 1      24   176           w
## 2      54   168           w
## 3      30   193           m
```

Inhalte mit ?? suchen

Internet: 'r Suchbegriff' in die Suchmaschine eingeben

```
Objektname <- c(Daten)
```

```
data.frame()
```

	Alter	Größe	Geschlecht
1	24	176	w
2	54	168	w
3	30	193	m
4	19	195	m
5	28	157	w
6	27	180	w
7	44	172	m
8	60	195	m
9	18	185	w
10	21	166	m
11			

Abbildung 6: Data Editor zur Eingabe von Daten in die Datenmatrix

```
## 4      19    195      m
## 5      28    157      w
## 6      27    180      w
## 7      44    172      m
## 8      60    195      m
## 9      18    185      w
## 10     21    166      m
```

Um einzelne Variablen in der Datenmatrix anzuwählen, geben Sie den Namen des `data.frame`-Objektes gefolgt von einem `$` und dann dem Variablennamen an, z.B.:

```
dat$Größe
```

```
## [1] 176 168 193 195 157 180 172 195 185 166
```

FÜR GRÖßERE DATENSÄTZE ist es dann aber doch besser, die Daten mit externen Programmen (z.B. EpiData oder EpiInfo) einzugeben und in R zu importieren. Die zuverlässigste Art das zu tun ist es, die Daten in einem Text-Format (z.B. csv-Format) zu speichern und in R dann mit `read.table()` einzulesen. Diese Funktion hat sehr viele Argumente, die es ermöglichen fast jedes Text-Format korrekt zu lesen.

Man kann aber auch andere Datenformate problemlos einlesen. Das Paket *foreign* erlaubt es z.B. Dateien aus anderen Statistikprogrammen wie SAS oder SPSS einzulesen.

Pakete

Es kann sein, dass Sie für eine gewünschte Prozedur zunächst keine Funktion finden. In den meisten Fällen gibt es dann ein Zusatzpaket mit so einer Funktion, das Sie nutzen können. Zur Zeit stehen etwas über 14000 solcher Pakete auf den offiziellen R-Servern zur Verfügung. Sie können Sie sich unter <https://cran.r-project.org> ansehen.

Zuerst müssen Sie ein Paket mit dem Befehl

```
install.packages("Paketname")
```

installieren. Der Befehl lädt das Paket automatisch herunter und installiert es.

Um ein installiertes Paket zu nutzen, muss man es dann bei jeder neuen R-Sitzung erneut laden.

Variablen in *data.frames* auswählen:
`data.frame$Variablenname`

Textdateien einlesen mit
`read.table()`

Mit dem Paket *foreign* Daten aus anderen Statistikprogrammen einlesen

R-Pakete unter <https://cran.r-project.org>

Pakete mit `install.packages()` installieren

Paket mit `library()` laden


```
library("Paketname")
```

R-STUDIO BIETET IHNEN im rechten unteren Fenster unter *Packages* grafische Hilfsmittel zur Paketverwaltung an. Es werden dort alle installierten Pakete angezeigt. Die geladenen Pakete sind mit einem Häkchen markiert. Wenn Sie auf den Namen eines Paketes klicken, kommen Sie zu den zugehörigen Hilfe-Seiten.

Mit  **Install** können Sie neue Pakete installieren.

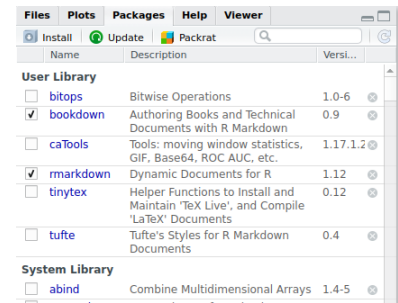


Abbildung 7: RStudio-Fenster Packages

Grafiken

Mit R können Sie sehr einfach statistische Grafiken erstellen. Sie können diese Grafiken bis ins kleinste Detail ganz genau so gestalten, wie Sie das möchten. (Dann wird es allerdings etwas komplizierter). Hier ist erst einmal ein ganz einfaches Beispiel:

```
boxplot(dat$Alter ~ dat$Geschlecht)
```

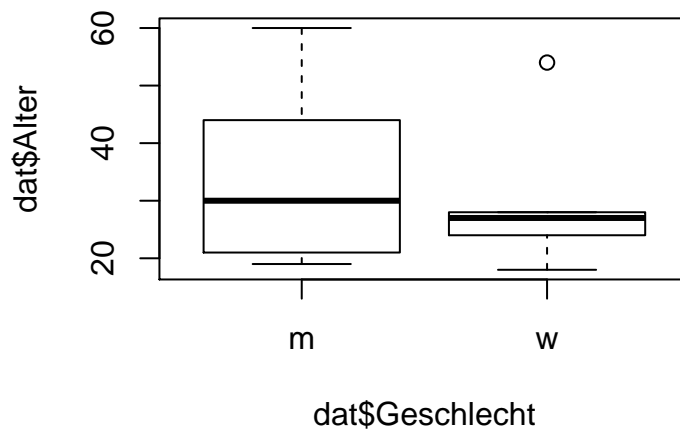


Abbildung 8: Boxplot der Altersangaben, aufgeteilt nach Geschlecht (fiktive Daten).



Mehr Beispiele können Sie sich mit folgender Eingabe ansehen:





```
demo(graphics)
```

Weitere Beispiele gibt es in der R Graph Gallery <https://www.r-graph-gallery.com/> und für interaktive Grafiken unter <https://shiny.rstudio.com/gallery/>.

History und Skript

Meistens wird eine Funktion beim ersten Versuch nicht genau das machen, was Sie sich gedacht haben, sondern Sie müssen sie erst noch ein bisschen weiter bearbeiten, bevor Sie zufrieden sind. Sie müssen dann nicht jedes mal die ganze Funktion neu eintippen, sondern können mit der Pfeil-nach-oben-Taste die zuletzt ausgeführten Funktionen noch einmal aufrufen.

RStudio zeigt Ihnen die zuletzt verwendeten Funktionen auch (rechts oben) im Fenster *History* an (siehe Abb. 9). Sie können dort ausgewählte Funktionen mit  **To Console** in die Konsole kopieren, oder mit  **To Source** in eine Skript-Datei, in der Sie Ihre Arbeitsschritte speichern und protokollieren können.

ES IST UNBEDINGT EMPFEHLENSWERT, mit solchen Skript-Dateien zu arbeiten. Ein neues Skript erstellen Sie, indem Sie links oben  und dann  **R Script** Ctrl+Shift+N klicken. Um einzelne Funktionen aus dem Skript heraus auszuführen, klicken Sie  **Run** oder drücken Sie **STRG-Enter**. Das gesamte Skript können Sie mit  **Source** ausführen, oder Sie können in der Konsole die Funktion `source()` ausführen.

```
source("Name_der_Skriptdatei.R")
```

Kommentieren Sie Ihr Skript mit `#`, damit Sie Ihre Arbeit später noch nachvollziehen können. Im Sinne von Nachvollziehbarkeit und Reproduzierbarkeit sollten die gesamte Datenaufbereitung und Datenanalyse als Skript speichern. Abbildung 10 zeigt ein Beispiel für solch ein Skript.

Ergebnisse Speichern

Sie können das Skript speichern, indem Sie auf das Diskettensymbol am oberen Rand klicken. Standardmäßig wird es in den Ordner gespeichert, in dem Sie ihr Projekt angelegt haben. In der Regel ist das schon ausreichend, wenn Sie nämlich den ganzen Arbeitsablauf vom laden der Daten bis zur Berechnung der Ergebnisse in diesem Skript erledigen. Trotzdem kann es eine gute Idee sein, die Daten noch einmal separat zu speichern.

Sie können den Gesamten *Workspace* - also alle Objekte, die Sie im Fenster *Environment* sehen - mit der Funktion

Mit *Pfeil-nach-oben* die letzten Funktionen wieder aufrufen






Environment	History	Connections
	 To Console	 To Source  
<pre> } Mittelwert(alter) Mittelwert(c(18, 36, 22, 54, 30, 28, 29, 19)) mean(alter) mean(c(24, 54, NA, 30)) mean(c(24, 54, NA, 30), na.rm = TRUE) dat <- data.frame() fix(dat) dat dat\$Größe library("Paketname") plot(dat\$Alter, dat\$Größe) demo(graphics) </pre>		

Abbildung 9: History-Fenster in RStudio

Ein Skript mit `source()` auszuführen.

Kommentieren mit `#`.

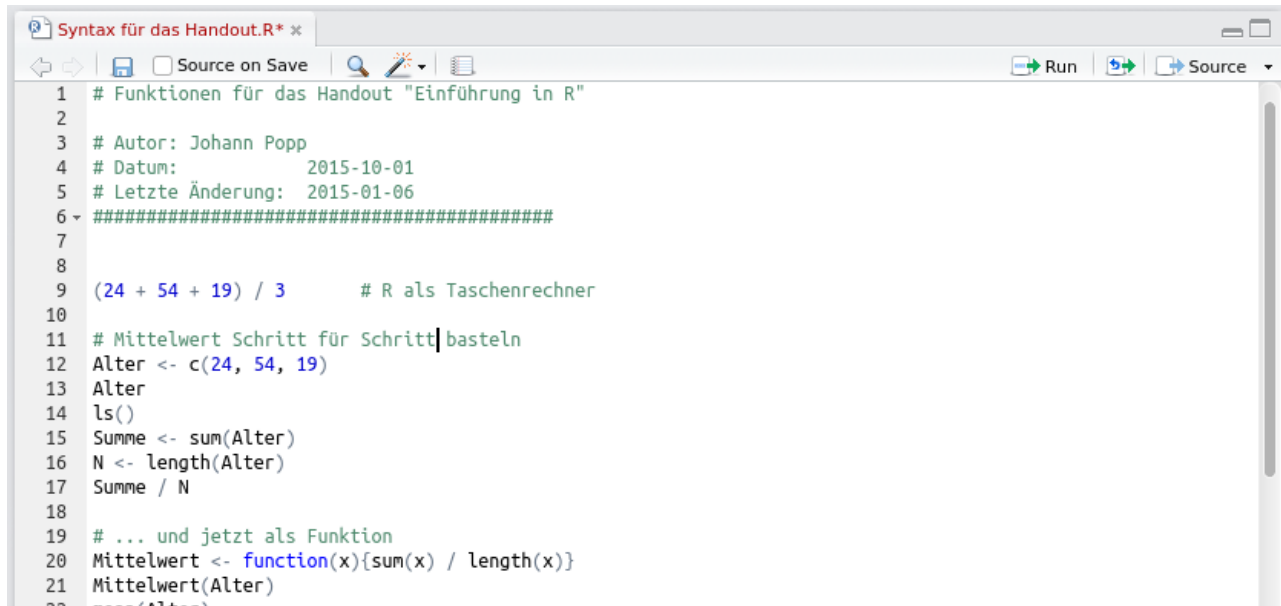


Abbildung 10: Beispiel für ein R-Stkript

Workspace mit `save.image()` speichern

```
save.image("Dateiname.RData")
```

speichern. Die Dateinamensendung ist dabei immer „RData“. Später können Sie ihn dann mit

```
load("Dateiname.RData")
```

wieder laden.

Workspace mit `load()` laden

SIE KÖNNEN DATENSÄTZE AUCH ALS TEXTDATEI speichern. Hier zum Beispiel den oben erzeugten `data.frame()` „dat“, mit dem Dateinamen „Testdaten.txt“ und mit einem Semikolon als Trennzeichen:

```
write.table(dat, "Testdaten.txt", sep = ";")
```

Abbildung 11 zeigt den Inhalt der erzeugten Datei. Sie lässt sich mit `read.table()` später wieder in R einlesen.

```
read.table("Testdaten.txt", header = TRUE, sep = ";")
```

```
##      Alter Größe Geschlecht
## 1      24   176           w
```

Datensätze mit `write.table()` speichern

```
["Alter";"Größe";"Geschlecht"
"1";24;176;"w"
"2";54;168;"w"
"3";30;193;"m"
"4";19;195;"m"
"5";28;157;"w"
"6";27;180;"w"
"7";44;172;"m"
"8";60;195;"m"
"9";18;185;"w"
"10";21;166;"m"]
```

Abbildung 11: Inhalt einer mit `write.table()` erstellten Textdatei

## 2	54	168	w
## 3	30	193	m
## 4	19	195	m
## 5	28	157	w
## 6	27	180	w
## 7	44	172	m
## 8	60	195	m
## 9	18	185	w
## 10	21	166	m

Literatur

R Core Team: „R: A Language and Environment for Statistical Computing“Vienna, Austria: R Foundation for Statistical Computing, 2018, <https://www.R-project.org/>.

RStudio Team: „RStudio: Integrated Development Environment for R“Boston, MA: RStudio, Inc., 2012, <http://www.rstudio.com/>.