

Einführung in R

Inhalt

Installation von R und RStudio	2
Grafische Benutzeroberfläche bzw. Aufbau von RStudio	5
Kurze Einführung in die Bedienung von RStudio	11
Häufige Fehlermeldungen	17

Installation von R und RStudio

Tutorials:

Wie man R und RStudio richtig installiert, wird in folgenden YouTube-Videos sehr gut beschrieben:

Windows:

[R installieren \(Windows\) und RStudio installieren - YouTube](#)

Mac:

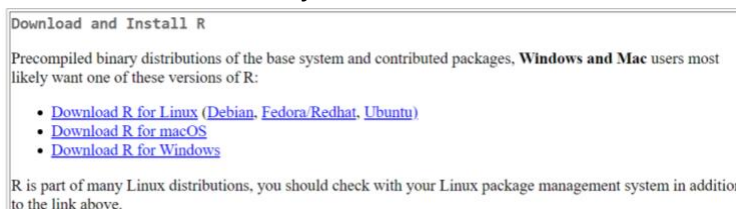
[How to install R & RStudio on Mac in 2021 - step-by-step walkthrough - YouTube](#)

In der folgenden Anleitung sind jeweils nur die ersten grundlegenden Schritte beschrieben. Wer für den Rest oder auch generell eine zusätzliche Hilfestellung benötigt, kann auf die YouTube-Videos zurückgreifen.

Download R


Erste Schritte

1. Klicke auf den folgenden Link: <https://cran.r-project.org/>
2. Wähle dein Betriebssystem aus:



Für Mac-User: Download R for macOS

Klicke auf den gelb markierten Link:



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](#)

R for macOS

This directory contains binaries for a base distribution and packages to run on macOS. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [old](#) directory.

Note: Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Package binaries for R versions older than 3.2.0 are only available from the [CRAN archive](#) so users of such versions should adjust the CRAN mirror setting (<https://cran-archive.r-project.org>) accordingly.

R 4.1.1 "Kick Things" released on 2021/08/10

Please check the SHA1 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`openssl sha1 R-4.1.1.pkg`
in the *Terminal* application to print the SHA1 checksum for the R-4.1.1.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-4.1.1.pkg`

Latest release:

R-4.1.1.pkg (notarized and signed)
SHA1 hash: d8ed71d0733bc80911ac3d16708d41e1399d310e (ca. 86MB)

R 4.1.1 binary for macOS 10.13 (High Sierra) and higher, Intel 64-bit build, signed and notarized package.
Contains R 4.1.1 framework, R.app GUI 1.77 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 6.7. The latter two components are optional and can be omitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

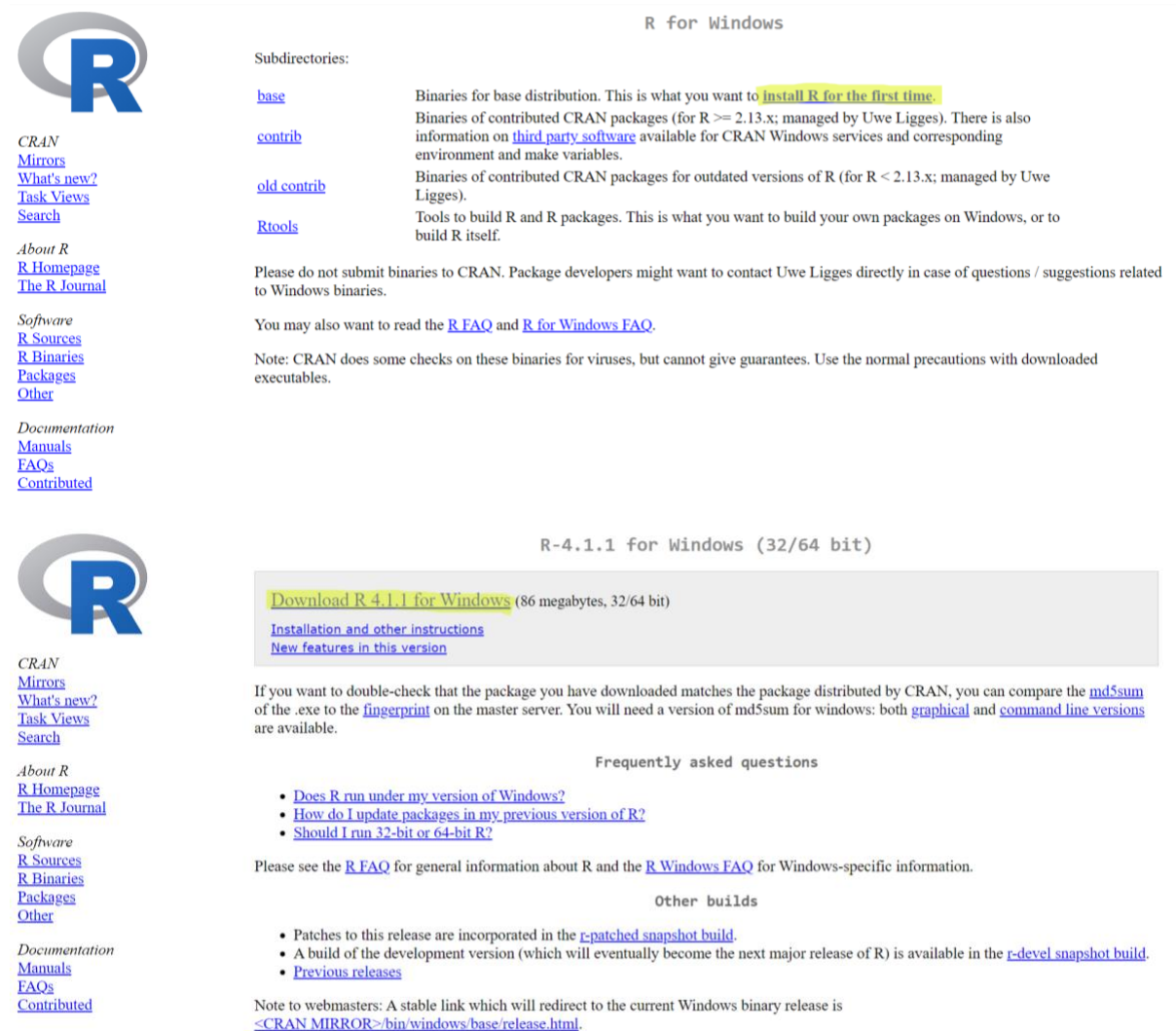
Note: the use of X11 (including `tcltk`) requires [XQuartz](#) to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

<https://cran.r-project.org/bin/macosx/base/R-4.1.1.pkg>

Der Rest ist relativ selbsterklärend. Wer dennoch eine genaue Anleitung braucht, kann sich das oben verlinkte Tutorial (für Mac) auf YouTube ansehen.

Für Windows-User: Download R for Windows

Klicke auf die gelb markierten Links:



R for Windows

Subdirectories:

- [base](#): Binaries for base distribution. This is what you want to [install R for the first time](#).
- [contrib](#): Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.
- [old.contrib](#): Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
- [Rtools](#): Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

R-4.1.1 for Windows (32/64 bit)

[Download R 4.1.1 for Windows](#) (86 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is [<CRAN MIRROR>/bin/windows/base/release.html](#).

Auch hier sind die restlichen Schritte ziemlich selbsterklärend. Wer dennoch Hilfe benötigt, kann sich das oben verlinkte Tutorial (für Windows) auf YouTube ansehen.

Download RStudio

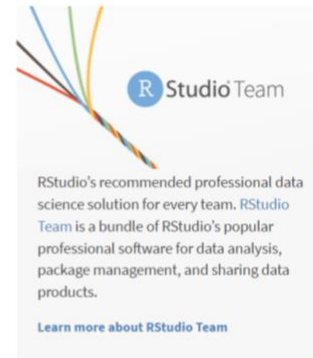
Die grafische Benutzeroberfläche in der Entwicklungsumgebung RStudio ist viel besser ausgebaut als die des Basisprogramms R, weswegen wir diese auch nutzen wollen. RStudio ist eine integrierte Entwicklungsumgebung (integrated development environment, IDE) für die Statistiksoftware R. Diese IDE bietet uns verschiedene Werkzeuge an, die uns den Umgang mit der grundlegenden Software erleichtern.

1. Um RStudio zu installieren, klicke auf den folgenden Link:
<https://www.rstudio.com/products/rstudio/download/#download>
2. Wähle die Open Source Licence, indem du auf folgenden gelb markierten Button klickst:

Choose Your Version

The RStudio IDE is a set of integrated tools designed to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

[LEARN MORE ABOUT THE RSTUDIO IDE](#)



RStudio Desktop	RStudio Desktop Pro	RStudio Server	RStudio Workbench
Open Source License	Commercial License	Open Source License	Commercial License
Free	\$995 /year	Free	\$4,975 /year (5 Named Users)
DOWNLOAD	BUY	DOWNLOAD	BUY

3. Wähle hier den passenden Link für dein Betriebssystem aus:

All Installers

Linux users may need to [import RStudio's public code-signing key](#) prior to installation, depending on the operating system's security policy.

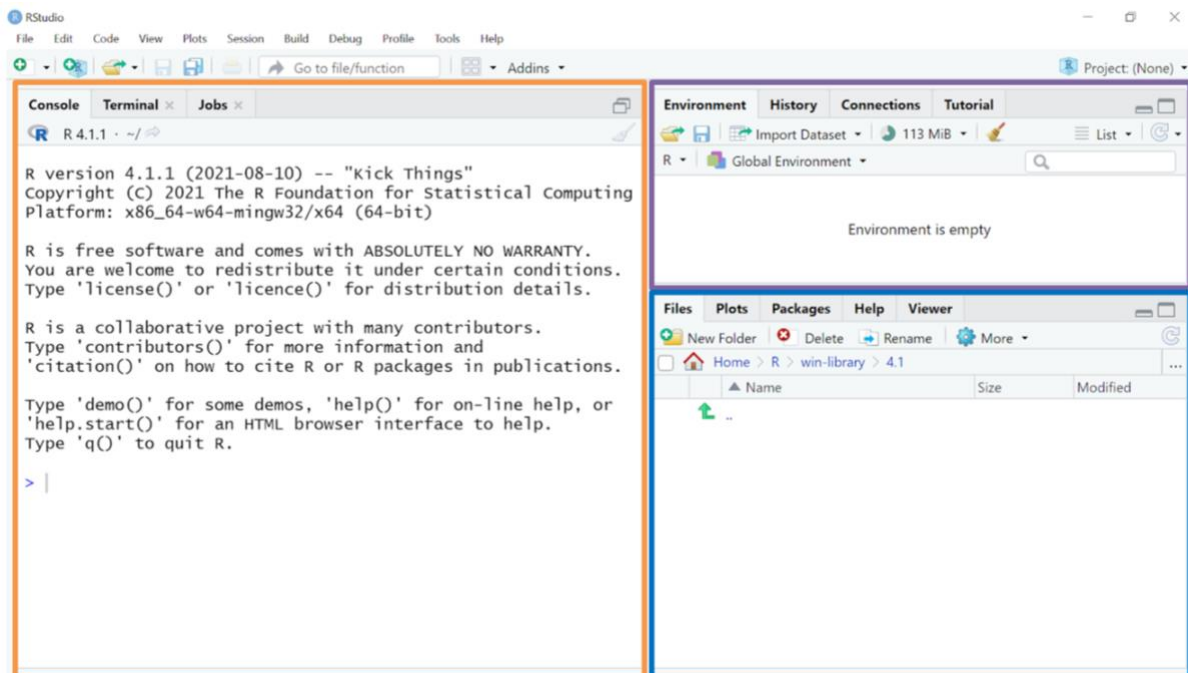
RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an [older version of RStudio](#).

OS	Download	Size	SHA-256
Windows 10	RStudio-2021.09.0-351.exe	156.88 MB	f698d4a2
macOS 10.14+	RStudio-2021.09.0-351.dmg	196.28 MB	f8e97ced
Ubuntu 18/Debian 10	rstudio-2021.09.0-351-amd64.deb	116.53 MB	0d7ef262
Fedora 19/Red Hat 7	rstudio-2021.09.0-351-x86_64.rpm	133.82 MB	3d858521
Fedora 28/Red Hat 8	rstudio-2021.09.0-351-x86_64.rpm	133.84 MB	1043943b
Debian 9	rstudio-2021.09.0-351-amd64.deb	116.79 MB	309b7d7c
OpenSUSE 15	rstudio-2021.09.0-351-x86_64.rpm	119.27 MB	108d4ae4

Wie es danach weitergeht, ist wieder in den oben verlinkten YouTube-Tutorials gut beschrieben.

Grafische Benutzeroberfläche bzw. Aufbau von RStudio

Startet man RStudio zum ersten Mal, werden drei Fenster mit unterschiedlichen Reitern angezeigt:



Konsole

Die Konsole (manchmal auch als Kommandozeile, Befehlszeile, Terminal oder command-line interface, CLI bezeichnet), ist ein Eingabebereich (interface). In die Konsole von RStudio können Variablendefinitionen, Berechnungen/Funktionen, sowie der Aufruf von Hilfsfunktionen direkt eingegeben werden. In der Konsole bekommen wir manchmal auch Fehler- (error) und Warnmeldungen (warning) sowie andere Informationen zu einer Funktion (message) angezeigt.

Der Inhalt der Konsole wird nach jedem Schließen des Programms gelöscht und kann nicht ohne weiteres gespeichert werden. Deswegen ist es wichtig, die eigene Arbeit in Skripten zu speichern (siehe S. 9).


Environment

Im Environment sehen wir alle Objekte, die derzeit in R geladen sind oder welche wir über die Konsole oder ein R Script definiert und ausgeführt haben (z. B. Variablen, Vektoren, Datensätze). Wenn wir (externe) Datensätze einlesen, sehen wir diese auch hier. Außerdem werden gegebenenfalls weitere Informationen zu den Objekten angezeigt.

Bsp.:

Environment	History	Connections
Global Environment		
Data		
df_1	4 obs. of 2 variables	
list_kurs	List of 4	
mat_1	num [1:2, 1:2] 1 3 2 4	
women	15 obs. of 2 variables	
Values		
eins	num [1:4] 1 3 2 1	
nominal_x	Factor w/ 3 levels "1","2","3": 1 3 2 3 2	
nominal_y	Factor w/ 4 levels "b","B","c","f": 4 2 3 1 3	
nums	num [1:5] 4 3 6 2 3	
obj	num [1:3] 1 2 3	
ordinal_x	Ord.factor w/ 3 levels "1"<"2"<"3": 1 3 2 3 2	
ordinal_x.2	Ord.factor w/ 3 levels "3"<"2"<"1": 3 1 2 1 2	
ordinal_y	Ord.factor w/ 4 levels "b"<"B"<"c"<"f": 4 2 3 1 3	
ordinal_y.2	Ord.factor w/ 4 levels "B"<"f"<"b"<"c": 2 1 4 3 4	
vek_1	chr [1:2] "A" "B"	
vek_10	num [1:9] 1 1 1 2 3 2 3 2 3	
vek_2	logi [1:3] FALSE TRUE TRUE	
vek_3	chr [1:2] "1" "3"	
vek_4	num [1:3] 1 2 3	
vek_5	num [1:2] 1.3 4.5	
vek_6	int [1:2] 1 4	
vek_7	int [1:4] 2 3 4 5	
vek_8	int [1:8] 6 7 8 9 1 2 3 4	
vek_9	chr [1:10] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"	
x	num [1:5] 1 3 2 3 2	
y	chr [1:5] "f" "B" "c" "b" "c"	
zwei	chr [1:4] "A" "A" "B" "B"	




Die Objekte werden in zwei Kategorien aufgeteilt:

- Unter **Data** finden wir *Data Frames*, *Listen* und *Matrizen*. Bei allen bekommen wir die Länge der einzelnen Dimensionen angezeigt (**Data Frame**: obs. = Zeilen, variables = Spalten; **Matrix**: [Zeilen, Spalten]; **Liste**: List of ...). Data Frames werden mit einem vorangestellten  markiert. Bei Matrizen bekommen wir zusätzlich die ersten Elemente angezeigt. Durch Klicken können wir uns diese Objekte im Data Viewer anschauen.
- Unter **Values** finden wir *Vektoren* bzw. *Faktoren*. Bei Vektoren bekommen wir folgende Informationen: Datentyp (z. B. num = numeric, chr = character → nähere Informationen zu Datentypen und -strukturen siehe S. 16), Länge sowie die ersten 10 Elemente. Bei Faktoren werden uns die Datenstruktur und die (diskrete) Anzahl an Ausprägungen angezeigt.

History

Environment	History	Connections
To Console To Source		
<pre> 1 + 1 # mathematische Operation vektor <- c(1, 3) # Erstellung eines Objekts mean(vektor) ## Berechnung des Mittelwerts (eines Vektors) mittels einer Funktion </pre>		

In der History sehen wir den zuletzt ausgeführten Code.

Der Vorteil gegenüber der Konsole ist, dass der Inhalt der History *nicht* mit Beenden einer R-Session gelöscht wird, sondern wir auf den Code zugreifen und diesen wiederverwenden können bis dieser explizit gelöscht wird (Löschen erfolgt mit dem ). Mit  bekommen wir den markierten Code in unser Skript; mit  in die Konsole.

Files

Unter Files sehen wir das Arbeitsverzeichnis (Working Directory), also die Ordner(struktur) auf unserem Rechner. Dieser Reiter entspricht im Wesentlichen also einem Dateimanager. Mit dem Working Directory legen wir u.a. fest, wo unser aktuelles R-Skript gespeichert wird und wo andere Objekte, die wir aus R exportieren (z.B. Grafiken), standardmäßig (während der aktuellen Sitzung) gespeichert werden.

Das WD müssen wir (in der Regel) in jeder R-Sitzung erneut festlegen.

Plots

Hier werden (von uns erstellte) Grafiken ausgegeben.

Packages

FilesPlotsPackagesHelpViewer

InstallUpdate

Name

Description

Version

System Library

☐

abind

Combine Multidimensional Arrays

1.4-5

☐

acepack

ACE and AVAS for Selecting Multiple Regression Transformations

1.4.1

☐

afex

Analysis of Factorial Experiments

0.27-2

☐

alr4

Data to Accompany Applied Linear Regression 4th Edition

1.0.6

☐

Amelia

A Program for Missing Data

1.7.6

☐

apaTables

Create American Psychological Association (APA) Style Tables

2.0.5

☐

askpass

Safe Password Entry for R, Git, and SSH

1.1

☐

assertthat

Easy Pre and Post Assertions

0.2.1

☐

backports

Reimplementations of Functions Introduced Since R-3.0.0

1.1.6

☒

base

The R Base Package

3.6.3

☐

base64enc

Tools for base64 encoding

0.1-3

☐

bayestestR

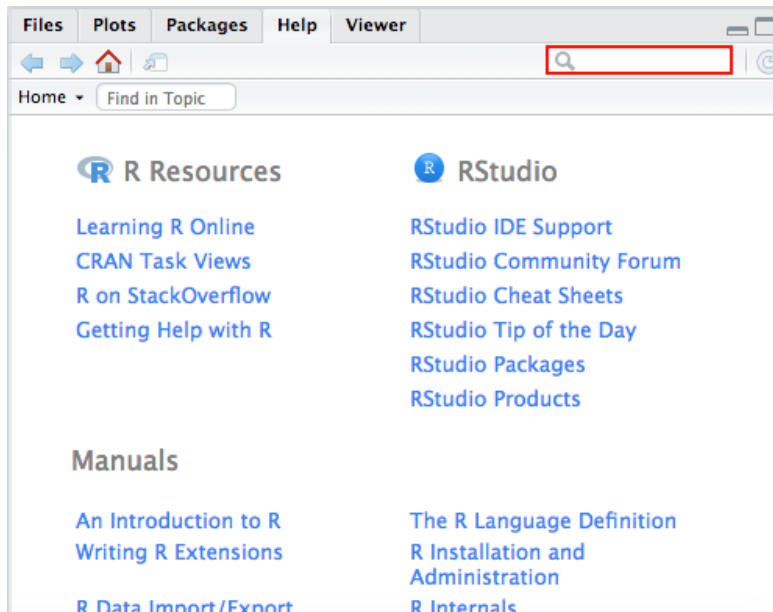
Understand and Describe Bayesian


0.5.3

Eine Übersicht unserer Pakete finden wir unter Packages (nähere Informationen dazu im Kapitel *Installation und Laden von Paketen* auf S. 12-14).

Help

Informationen zu R und zu Funktionen finden wir unter Help. Hier wird uns umfassende Hilfe zum Umgang mit R im Allgemeinen und zu Funktionen angeboten.



Den Namen der Funktion geben wir in das **Suchfeld** ein. Alternativ können wir auch die Funktionen `help(funktion)` oder `?funktion` nutzen (für `funktion` natürlich den konkreten Namen der Funktion einsetzen). Wenn wir auf  klicken, öffnet sich die Dokumentationsseite in einem neuen Fenster, was die Nutzung wesentlich übersichtlicher gestaltet.


Neben der Informationen, aus welchem Paket eine Funktion stammt, finden wir hier zumeist folgende Abschnitte:

- **Description:** Beschreibung, was die Funktion macht
- **Usage:** Funktionsdefinition (Parameter der Funktion und ggf. Defaults)
- **Arguments:** Beschreibung der Parameter und ihrer möglichen Argumente (→ genauere Informationen zu Argumenten siehe S. 14)
- **Details:** detaillierte Beschreibung zur Nutzung der Funktion und etwaigen Sonderfällen
- **See Also:** verwandte Funktionen (meist aus dem gleichen Paket)
- **Examples:** Beispiele zur Nutzung der Funktion

Viewer

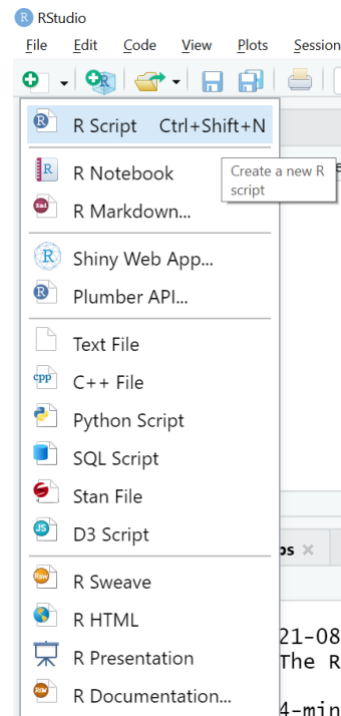
Im Viewer werden (von uns erstellte) Tabellen angezeigt.

Skript

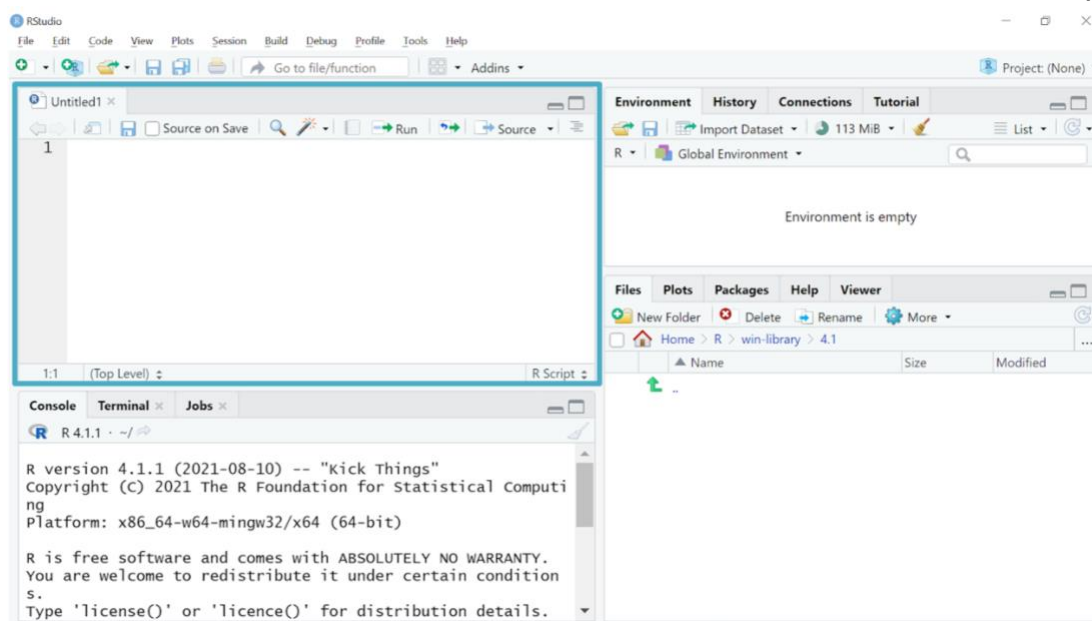
Über *File* → *New File* → *R Script* oder indem man oben in der Leiste ganz links auf  und dann auf *R Script* geht, kann man ein sogenanntes **R Script** erstellen:



oder





Dann öffnet sich zusätzlich zu den drei Fenstern ein viertes Fenster, das Skript:



Ein Skript ist eine einfache Textdatei, in die R-Befehle eingetragen werden können, die dann zur Ausführung an die Konsole geschickt werden.

Hier schreiben wir also unseren Code rein, wenn wir diesen speichern wollen.

Um den Code auszuführen, markieren wir ihn und klicken auf  oder stellen den Cursor in die Zeile und drücken die Tastenkombination .

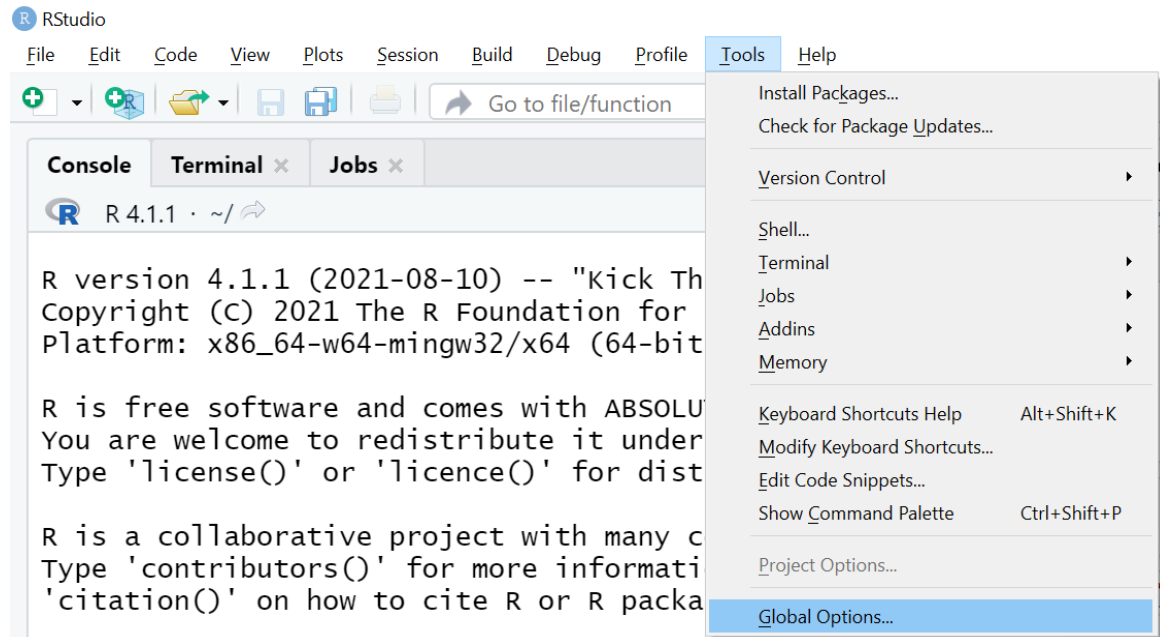
Zusammenfassung

In das **Skript** schreiben wir unseren Code. In der **Konsole** wird dieser ausgeführt und die Ergebnisse angezeigt. Bestehende Objekte sehen wir im **Environment**. Eine Übersicht unserer Pakete finden wir unter **Packages** und Informationen zu R und zu Funktionen finden wir unter **Help**.

Kurze Einführung in die Bedienung von RStudio

Global Options

Über *Tools* → *Global Options* lassen sich Einstellungen vornehmen:

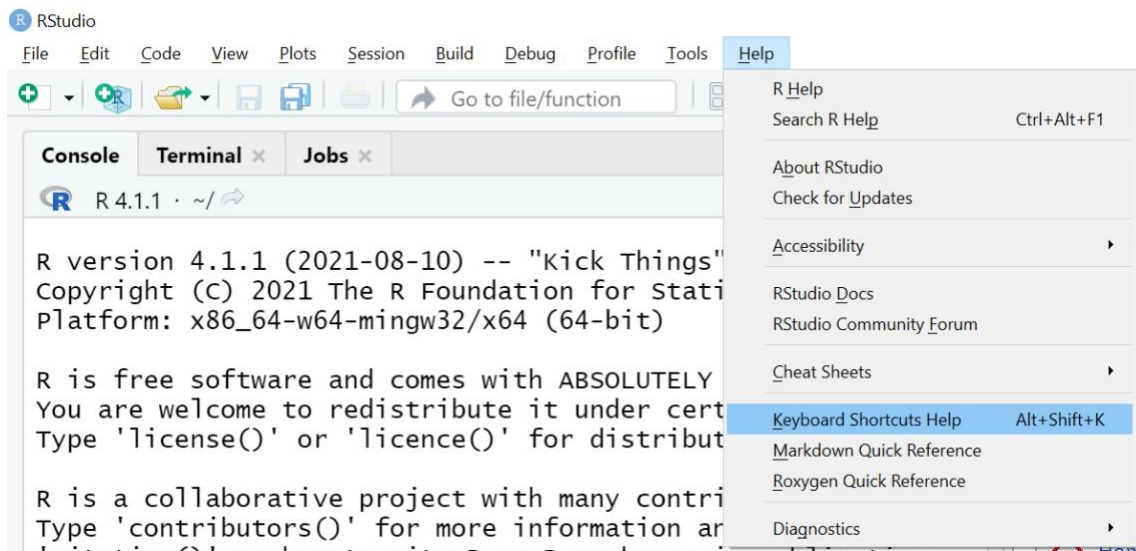


Was sich hier einstellen lässt, wird im folgenden Youtube-Video, ca. ab Minute 03:28, kurz erklärt: [R mit RStudio - eine Einführung in die Bedienung von RStudio - YouTube](#)

Nützliche Kurzbefehle

	Windows	Mac
Code der aktuellen Linie bzw. markierten Code ausführen	Strg + enter	cmd + enter
Code bis zur aktuellen Linie ausführen	alt + Strg + B	alt + cmd + B
Skript speichern	Strg + S	cmd + S
Dateipfad kopieren	shift + Rechtsklick auf Datei dann Als Pfad kopieren	alt + cmd + C

Eine Übersicht weiterer Kurzbefehle für R finden wir in der Leiste ganz oben unter *Help* → *Keyboard Shortcuts Help* (nicht zu verwechseln mit dem Bereich Help, der uns Zugang zur Dokumentation verschafft):



Installation und Laden von Paketen

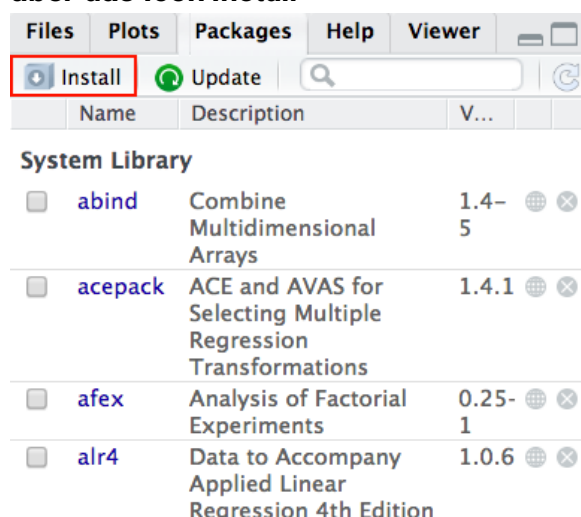
Pakete (Packages) stellen zusätzliche Funktionen (→ nähere Informationen zu Funktionen siehe S. 14) zur Verfügung. Es gibt gewisse Standardpakete, welche in der Basisausstattung von R (base R) enthalten sind. Alle anderen Packages müssen installiert werden. Dies ist auf drei Wegen möglich, welche anhand von *Tidyverse* erklärt werden sollen. Tidyverse ist gleich eine ganze Sammlung von Paketen (z. B. ggplot2, tibble, readr, tidyr).

1. über die Funktion `install.packages()`

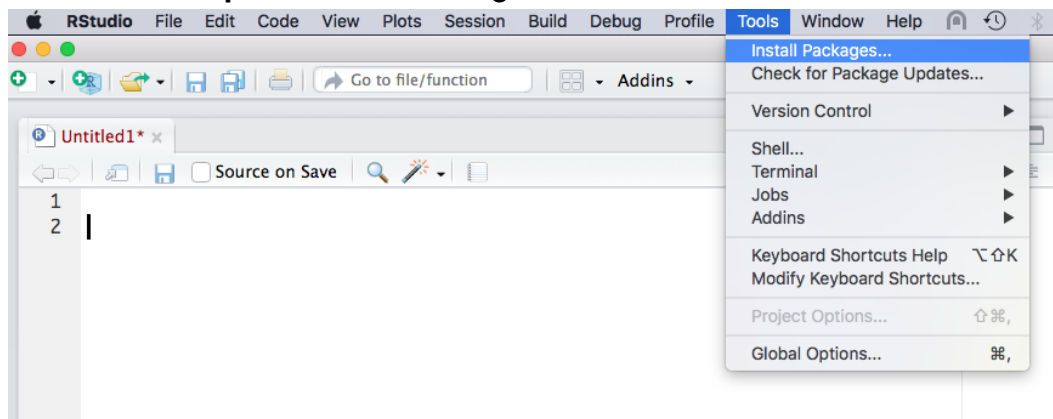
Wir können die Packages, die zu Tidyverse gehören, installieren, indem wir folgenden Code in die Konsole tippen und ausführen:

```
install.packages("tidyverse")
```

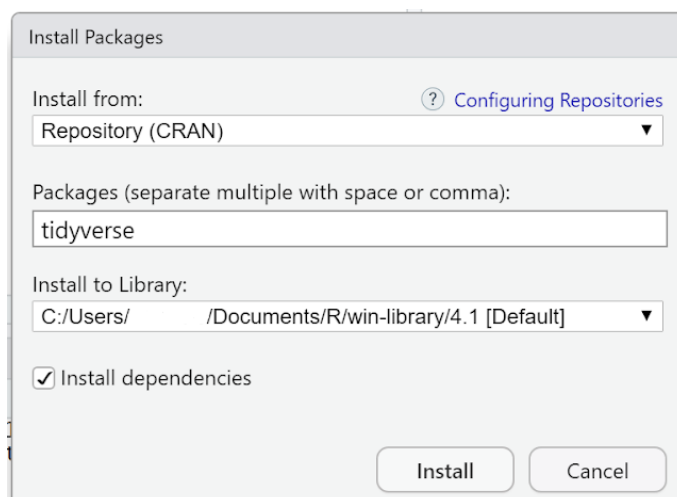
2. über das Icon *Install*



3. über den Menüpunkt *Install Packages*



Sowohl bei Option 2 als auch bei Option 3 öffnet sich anschließend ein neues Fenster, in welchem wir unter dem Reiter **Packages (...)** den Namen des Pakets eingeben können. Anschließend müssen wir noch auf das Icon **Install** klicken:



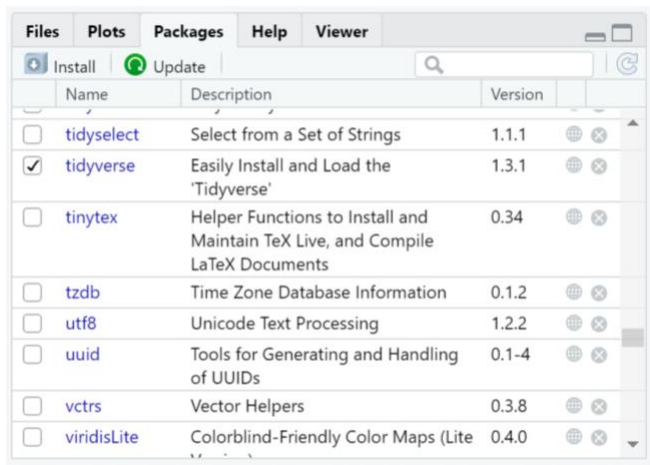
Mit dem Häkchen in dem Kästchen bei **Install Dependencies** werden von dem Zielpaket benötigte oder empfohlene bisher nicht installierte Pakete auch heruntergeladen.

R-Pakete müssen nur einmal installiert werden, bei jedem Start einer neuen R-Session muss jedoch jedes Paket, welches in dieser verwendet werden soll, geladen werden. Das Laden des Pakets kann entweder gleich zu Beginn erfolgen oder auch erst konkret dann, wenn es gebraucht wird. Dabei gibt es wiederum zwei mögliche Wege:

1. Über die Funktion `library()`

Wir können alle Tidyverse-Pakete auf einmal laden, indem wir `library(tidyverse)` in die Konsole tippen und ausführen.

2. Über das Häkchen-Setzen in der System Library:



Funktionen und Argumente

Funktionen sind sowas wie (Unter)Programme, die eine gewisse Funktionalität haben, d.h. eine bestimmte Aufgabe ausführen und das Arbeiten mit R erleichtern. Sie stehen in der Regel vor einer runden Klammer. Wir können Funktionen variierenden Input übergeben. Dieser steht immer in Klammern direkt hinter der Funktion und nennt sich **Argument**. Argumente müssen einen Namen und einen Wert haben und können dann in einer beliebigen Reihenfolge innerhalb der Funktion angegeben werden. Sie werden durch Kommata getrennt. Die schematische Struktur einer Funktion ist also: `Funktion(Argumente)`

Wichtige Operatoren

Logische Operatoren bzw. Vergleichsoperatoren

Operator	Bedeutung
<code>==</code>	gleich
<code>!=</code>	ungleich
<code><</code>	kleiner als
<code><=</code>	kleiner gleich
<code>></code>	größer als
<code>>=</code>	größer gleich

Arithmetische Operatoren

Operator	Bedeutung
+	Plus (Addition)
-	Minus (Subtraktion)
*	Multiplikation
/	Division
^ oder **	Potenz

Verknüpfen von Bedingungen

Operator	Bedeutung
bzw.	oder
& bzw. &&	und

Bemerkung: Das Symbol für oder | erhält man mit folgender Tastenkombination:



Datentypen

Der Datentyp gibt die Art der Daten an, d.h. welche konkreten Werte(bereiche) die Daten annehmen können und welche Operationen darauf anwendbar sind.

Art der Daten	Werte	Operationen	Datentyp in R	
Zeichen(ketten)	z. B. "a" oder "Statistics"	gleich oder ungleich	character	
Wahrheitswerte	TRUE, FALSE	logische Operatoren	logical	
ganze Zahlen	z. B. 2	arithmetische und logische Operatoren	integer	numeric
Kommazahlen	z. B. 2.4		double	

Datenstrukturen

Datenstrukturen können nach Dimensionalität und enthaltenen Datentypen klassifiziert werden. Nachfolgend befindet sich eine Übersicht der in R enthaltenen Datenstrukturen.

		Beinhaltet unterschiedliche Datentypen?	
		nein (homogen)	ja (heterogen)
Anzahl der Dimensionen	1	Vektor	Liste
	2	Matrix	Data Frame
	n	Array	

Häufige Fehlermeldungen

argument ". . ." is missing, with no default

- Man hat vergessen das Argument . . . zu spezifizieren.
- Man hat einen Fehler in der Reihenfolge bzw. Namen der Argumente, sodass R den Wert für . . . einem anderen Argument zugeordnet hat.

cannot open the connection

- Tippfehler im Pfad, z.B. '\' statt '/'
- Die Datei befindet sich nicht dort.
- Instabile Verbindung bei z.B. Netzlaufwerken
- Keine Lese-/Schreibrechte

could not find function

- Tippfehler im Funktionsnamen
- Runde statt eckiger Klammern bei Teilzugriff auf Vektoren, Matrizen, Arrays, Data Frames oder Listen
- Das Paket, in dem sich die Funktion befindet, wurde nicht installiert bzw. geladen. Gegebenenfalls `install.packages("paketname")` und danach einmal `library("paketname")` ausführen.

incorrect number of dimensions

- Zu viele Kommata innerhalb der eckigen Klammern beim Zugriff auf bestimmte Einträge eines Vektors (oder Data Frames, Matrices bzw. Arrays) mittels [...]

object '. . .' not found

- Tippfehler im Objektnamen
- Anführungszeichen vergessen
- Die Datei oder das Paket, in dem sich das gesuchte Objekt befindet, wurde nicht geladen.

object of type '. . .' is not subsettable

- Eckige statt runder Klammern bei einem Funktionsaufruf

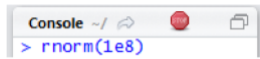
subscript out of bounds

- Zugriff auf nicht vorhandene Zeilen oder Spalten mittels [...].

keine Fehlermeldung

Es gibt keine Fehlermeldung und es passiert vermeintlich nichts. Mögliche Erklärungen:

- Plus in der Konsole: Ein + statt des > in der letzten Zeile der Konsole ist ein Zeichen eines unvollständigen Befehls.
- Stoppschild:



Das Stoppschild rechts von *Console* bedeutet, dass R gerade beschäftigt ist. Mit *Esc* kann man den aktuellen Prozess abbrechen. Mögliche Gründe:

- R hat eine schwierige Aufgabe bekommen
 - R wartet auf eine Eingabe in einem anderen Fenster.
- Der Befehl gibt normalerweise keine Rückmeldung, d.h. es ist alles in Ordnung. Setzt man den kompletten Befehl in Klammern gibt es immer eine Rückmeldung, meistens NULL; so sieht man zumindest, dass der Befehl ausgeführt wurde.
- Bei Grafikfunktionen, die Elemente zu bestehenden Grafiken hinzufügen, kann es sein, dass die hinzugefügten Elemente außerhalb der dargestellten Achsenabschnitte liegen und damit unsichtbar sind. Häufigste Ursache: x- und y-Achse vertauscht!