



# Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2022/23

Belinda Fleischmann

Inhalte basieren auf Programmierung und Deskriptive Statistik von Dirk Ostwald, lizenziert unter CC BY-NC-SA 4.0

## (2) R und RStudio Grundlagen

---

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

---

## **R und RStudio**

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

## Was ist R?

- Eine Programmiersprache und ein Softwarepaket.
- Entwickelt von Ihaka and Gentleman (1996).
- Freier Dialekt der proprietären Software S (Becker, Chambers, and Wilks (1988)).
- Weiterentwickelt und gepflegt durch R Core Team und R Foundation
- Interpretierte imperativ-objektorientierte 4GL Sprache.
- Optimierte und populär für statistische Datenanalysen.
- Große Community mit etwa 20.000 beigetragenen R Paketen (Erweiterungen)
- Evolviert und konservativ im Kern, konsistent und progressiv in R Paketen.

## Wie bekommt man R?

Runterladen (z.B. <https://cran.r-project.org/bin/windows/base/>) und installieren.



CRAN  
Mirrors  
What's new?  
Search  
CRAN Team

About R  
R Homepage  
The R Journal

Software  
R Sources  
R Binaries  
Packages  
Task Views  
Other

Documentation  
Manuals  
FAQs  
Contributed

<p><b>Download and Install R</b></p> <p>Precompiled binary distributions of the base system and contributed packages, <b>Windows and Mac</b> users most likely want one of these versions of R:</p> <ul style="list-style-type: none"><li>• <a href="#">Download R for Linux (Debian, Fedora/Redhat, Ubuntu)</a></li><li>• <a href="#">Download R for macOS</a></li><li>• <a href="#">Download R for Windows</a></li></ul> <p>R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.</p> <p><b>Source Code for all Platforms</b></p> <p>Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!</p> <ul style="list-style-type: none"><li>• The latest release (2022-06-23, Funny-Looking Kid!) <a href="#">R-4.2.1.tar.gz</a>, read <a href="#">what's new</a> in the latest version.</li><li>• Sources of <a href="#">R alpha and beta releases</a> (daily snapshots, created only in time periods before a planned release).</li><li>• Daily snapshots of current patched and development versions are <a href="#">available here</a>. Please read about <a href="#">new features and bug fixes</a> before filing corresponding feature requests or bug reports.</li><li>• Source code of older versions of R is <a href="#">available here</a>.</li><li>• Contributed extension <a href="#">packages</a></li></ul> <p><b>Questions About R</b></p> <ul style="list-style-type: none"><li>• If you have questions about R like how to download and install the software, or what the license terms are, please read our <a href="#">answers to frequently asked questions</a> before you send an email.</li></ul>
--

### What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R project homepage](#) for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN [mirror](#) nearest to you to minimize network load.

### Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the [CRAN Repository Policy](#) and then use the [web form](#).

If this fails, send an email to [CRAN-submissions@R-project.org](mailto:CRAN-submissions@R-project.org) following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

## Was kann man mit R machen?

- Datensätze laden, manipulieren, und speichern.
- Eine Vielzahl von Berechnungen an verschiedenen Datenstrukturen durchführen.
- Eine Vielzahl statistischer Analysemethoden auf Daten anwenden.
- Datenanalyseskripte schreiben und Abbildungen generieren.
- Präsentationen RMarkdown und Bücher RBookdown erstellen.

## Was kann man mit R (bisher) nicht so gut machen?

- In einer ansprechenden Umgebung programmieren ( $\Rightarrow$  RStudio).
- Scientific Computing ( $\Rightarrow$  Python, Matlab, Julia).
- Psychologische Experimente programmieren ( $\Rightarrow$  Python, Matlab)

## Wie bekommt man Hilfe zu R?

- Googlen
- <https://stackoverflow.com/>
- <https://www.r-project.org/help.html>
- Während der Programmierung und bei bekanntem Funktionsnamen über die Kommandozeile:

```
?mean           # Zeigt Hilfe zu der Funktion "mean()"
help(mean)      # Zeigt Hilfe zu der Funktion "mean()"
browseVignettes() # Zeigt Vignetten aller installierten Pakete im Browser
browseVignettes("knitr") # Zeigt Vignetten des Paktes "knitr"
```

- <https://rseek.org/>
- <https://www.rstudio.com/resources/cheatsheets/>
- <https://www.r-bloggers.com/>



## Was ist RStudio?

- Eine Softwareentwicklungsumgebung für R
- Softwareentwicklungsumgebung = Integrated Development Environment
- IDEs sind Programme zum Programmieren mit einer Programmiersprache
- Kommandozeile, Skripteditor, Vielzahl weiterer Tools
- Freemium Produkt von RStudio, Inc. (IDE frei, Server kostenpflichtig)
- Initial Release 2011, Affero General Public License
- Keine Verbindung zu R Core Team oder R Foundation
- RStudio wird im Oktober 2022 zu posit ([posit.co](https://posit.co))

## Wie bekommt man RStudio?

Runterladen (<https://www.rstudio.com/products/rstudio/>) und installieren.

RStudio Desktop 2022.07.2+576 - [Release Notes](#)

1. Install R. RStudio requires R 3.3.0+.
2. Download RStudio Desktop. Find your operating system in the table below.



### 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 10/11	<a href="#">RStudio-2022.07.2-576.exe</a>	190.49 MB	b38bf925
macOS 10.15+	<a href="#">RStudio-2022.07.2-576.dmg</a>	224.49 MB	35828d82
Ubuntu 18+/Debian 10+	<a href="#">rstudio-2022.07.2-576-amd64.deb</a>	133.19 MB	b7d8c386
Ubuntu 22	<a href="#">rstudio-2022.07.2-576-amd64.deb</a>	134.06 MB	e1c51083

## Was kann man mit RStudio machen?

- R Skripte erzeugen, bearbeiten, und laufen lassen
- R Skripte in R Projekten organisieren

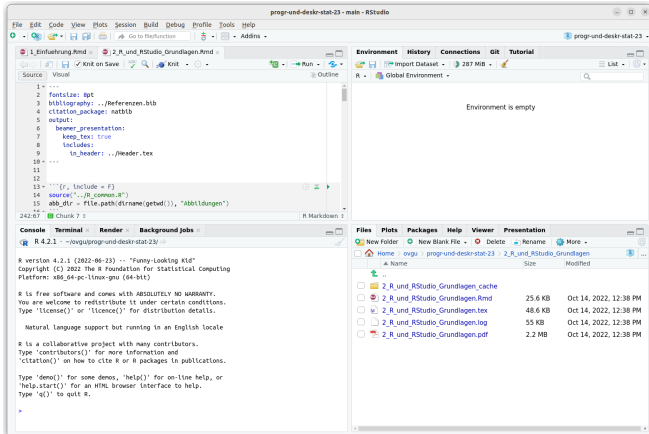
Laut Eigenwerbung:

- Access RStudio locally
- Syntax highlighting, code completion, and smart indentation
- Execute R code directly from the source editor
- Quickly jump to function definitions
- View content changes in real-time with the Visual Markdown Editor
- Easily manage multiple working directories using projects
- Integrated R help and documentation
- Interactive debugger to diagnose and fix errors
- Extensive package development tools

# R und RStudio

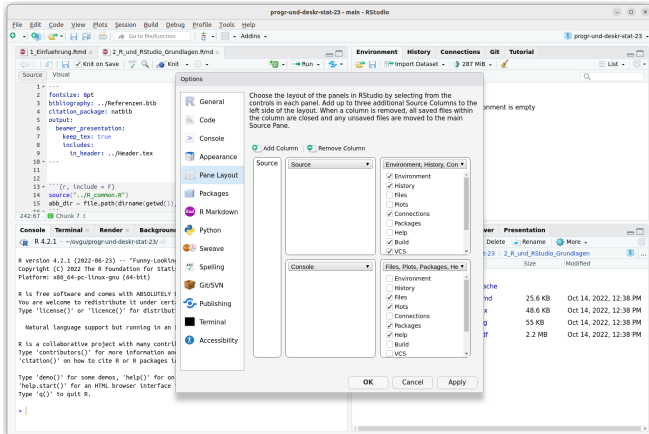
## Was kann man mit RStudio machen?

Custom Layout via Tools → Global Options ...



## Was kann man mit RStudio machen?

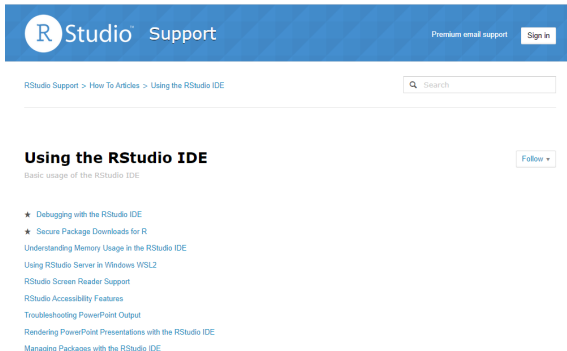
Custom Layout via Tools → Global Options ...



## Wie bekommt man Hilfe zu RStudio?

Googlen

Zur Einführung ⇒ Using the RStudio IDE



The screenshot shows the RStudio Support website. The header is blue with the RStudio logo and the word 'Support'. On the right, there are links for 'Premium email support' and a 'Sign in' button. Below the header, there is a breadcrumb trail: 'RStudio Support > How To Articles > Using the RStudio IDE'. To the right of the breadcrumb is a search bar. The main content area has the title 'Using the RStudio IDE' in bold, with a 'Follow' button to its right. Below the title is the subtitle 'Basic usage of the RStudio IDE'. A list of links follows, each preceded by a star icon: 'Debugging with the RStudio IDE', 'Secure Package Downloads for R', 'Understanding Memory Usage in the RStudio IDE', 'Using RStudio Server in Windows WSL2', 'RStudio Screen Reader Support', 'RStudio Accessibility Features', 'Troubleshooting PowerPoint Output', 'Rendering PowerPoint Presentations with the RStudio IDE', and 'Managing Packages with the RStudio IDE'.

RStudio Support

Premium email support Sign in

RStudio Support > How To Articles > Using the RStudio IDE

Search

### Using the RStudio IDE

Follow

Basic usage of the RStudio IDE

- ★ Debugging with the RStudio IDE
- ★ Secure Package Downloads for R
- Understanding Memory Usage in the RStudio IDE
- Using RStudio Server in Windows WSL2
- RStudio Screen Reader Support
- RStudio Accessibility Features
- Troubleshooting PowerPoint Output
- Rendering PowerPoint Presentations with the RStudio IDE
- Managing Packages with the RStudio IDE

## R Kommandozeile | Working in the Console

Die Basics:

- Eingabe von R Befehlen bei >
- Autocomplete mit `Tab`
- Code ausführen mit `Enter`
- Vorherige Befehle mit Cursor `↑`
- Bereinigen des Konsolenoutputs mit `Ctrl` + `L`
- Code Ausführungsstopp mit `Esc`

Beispiel für einen Befehl:

```
print("Hallo Welt!")
```

```
> [1] "Hallo Welt!"
```

Anmerkung:

**Code-Snippets in diesen Folien immer aktiv in der Konsole nachvollziehen!**

## R Skripte | Executing and Editing Code

### Neue .R Datei erstellen

- GUI: *File* → *New File* → *R Script*
- Keyboard Shortcut: Ctrl + Shift + N

### Bestehende .R Datei öffnen

- GUI: *File* → *Open File*
- Ctrl + O

### Code im .R Skript schreiben (Befehle in Programmiersprache formulieren)

```
print("Hallo Welt!")  # Hinter Hashtags stehen dokumentierende Kommentare
print("Hallo R!")     # Kommentare werden nicht ausgeführt
```

### Befehle eines R Scripts ausführen

- Einzelnen Zeile, auf welcher der Cursor ruht: ⇒ Run oder Ctrl + Enter
- Ausführen aller Zeilen: ⇒ Source oder Ctrl + Shift + Enter oder ⇒ Tickmark bei *Source on Save* setzen und Ctrl + S

Anmerkung:

**Code-Snippets in diesen Folien immer aktiv in einem R Skript dokumentieren!**



## Das R und RStudio Data Science Universum

### Analyse & Explore



The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying philosophy and common APIs.

[Project Site Link >](#)



dplyr is the next iteration of plyr, focusing on only data frames. dplyr is faster and has a more consistent API.

[Project GitHub Link >](#)



ggplot2 is an enhanced data visualization package for R. Create stunning multi-layered graphics with ease.

[Project Site Link >](#)



tidyr makes it easy to "tidy" your data. Tidy data is data that's easy to work with: it's easy to munge (with dplyr), visualise (with ggplot2 or ggridis) and model (with R's hundreds of modelling packages).

[Project Paper Link >](#)

### Model & Predict



TensorFlow™ is an open source software library for Machine Intelligence. The R interface to TensorFlow lets you work productively using the High-level Keras and Estimator APIs and the core TensorFlow API.

[Project Site Link >](#)



The tidymodels framework is a collection of packages for modeling and machine learning using tidyverse principles.

[Project Site Link >](#)



Sparklyr provides bindings to Spark's distributed machine learning library. Together with sparklyr's dplyr interface, you can easily create and tune machine learning workflows on Spark, orchestrated entirely within R.

[Project Site Link >](#)

### Connect & Integrate



Sparklyr is an R interface to Apache Spark, a fast and general engine for big data processing. This package connects to local and remote Apache Spark clusters, a dplyr-compatible back end, and an interface to Spark's ML algorithms.

[Project Site Link >](#)



Plumber enables you to convert your existing R code into web APIs by merely adding a couple of special comments.

[Project Site Link >](#)



The reticulate package provides a comprehensive set of tools for interoperability between Python and R.

[Project Site Link >](#)

### Communicate & Interact



Shiny makes it incredibly easy to build interactive web applications with R. Shiny has automatic "reactive" binding between inputs and outputs and extensive pre-built widgets.

[Project Site Link >](#)



Use R Markdown to develop your code and ideas in a reproducible document. Knit plots, tables, and results together with narrative text, and create analyses ready to be shared.

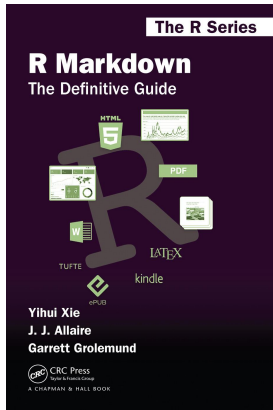
[Project Site Link >](#)



Use flexdashboard to publish groups of related data visualizations on a dashboard.

[Project Site Link >](#)

## Lehrmaterialien mit R und RStudio



---

R und RStudio

## **Arithmetik, Logik und Präzedenz**

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

## R Konsole als Taschenrechner

```
1 + 1
```

```
> [1] 2
```

```
2 * 3
```

```
> [1] 6
```

```
sqrt(2)
```

```
> [1] 1.41
```

```
exp(0)
```

```
> [1] 1
```

```
log(1)
```

```
> [1] 0
```

Anmerkung:

- [1] zeigt das erste und einzige Element des Ausgabevektors an
- Vektoren werden noch im Detail behandelt.

## Arithmetische Operatoren

Operator	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division
^ oder **	Potenz
%*%	Matrixmultiplikation
%/%	Ganzzahlige Teilung ( $5\%/%2 = 2$ )
%%	Modulo ( $5\%/%2 = 1$ )

- Matrixmultiplikation, Modulo, ganzzahlige Teilung benötigen wir zunächst nicht.
- Ganzzahlige Teilung gibt das Resultat der ganzzahligen Teilung an.
- Modulo gibt den ganzzahligen Rest bei ganzzahliger Teilung an.

## Logische Operatoren

- Die Boolesche Algebra und R kennen zwei *logische Werte*: TRUE und FALSE
- Bei Auswertung von Relationsoperatoren ergeben sich logische Werte

Relationsoperator	Bedeutung
==	Gleich
!=	Ungleich
<, >	Kleiner, Größer
<=, >=	Kleiner gleich, Größer gleich
	ODER
&	UND

- <, <=, >, >= werden zumeist auf numerische Werte angewendet.
- ==, != werden zumeist auf beliebige Datenstrukturen angewendet.
- | und & werden zumeist auf logische Werte angewendet.
- | implementiert das inklusive *oder*. Die Funktion xor() implementiert das exklusive ODER.

## Mathematische Funktionen

Aufruf	Bedeutung
<code>abs(x)</code>	Betrag
<code>sqrt(x)</code>	Wurzel
<code>ceiling(x)</code>	Aufrunden ( $\text{ceiling}(2.7) = 3$ )
<code>floor(x)</code>	Abrunden ( $\text{floor}(2.7) = 2$ )
<code>round(x)</code>	Mathematisches Runden ( $\text{round}(2.5) = 2$ )
<code>exp(x)</code>	Exponentialfunktion
<code>log(x)</code>	Logarithmus Funktion

- Hierbei handelt es sich um eine Auswahl. Einen vollständigen Überblick gibt

```
names(methods:::.BasicFunsList)
```

- R unterscheidet formal nicht zwischen Operatoren und Funktionen
- Operatoren können mit der Infix Notation als Funktionen genutzt werden

```
`+`(2,3)           # Infixnotation für 2 + 3
```

```
> [1] 5
```

## Operatorpräzedenz

- Operatorrangfolge
- Regeln der Form “Punktrechnung geht vor Strichrechnung”
- Vordefinierte Operatorpräzedenz kann durch Klammern überschrieben werden

```
2 * 3 + 4
```

```
> [1] 10
```

```
2 * (3 + 4)
```

```
> [1] 14
```

## Generell gilt

- Operatorrangfolge nicht raten oder folgern, sondern nachschauen!
- Lieber Klammern setzen, als keine Klammern setzen!
- Immer nachschauen, ob Berechnungen die erwarteten Ergebnisse liefern!

```
?Syntax
```



## Operatorpräzedenz

### Präzedenz und Ausführungsreihenfolge arithmetischer Operatoren

Operator	Reihenfolge
$^$	Rechts nach links
$-x, +x$	Unitäres Vorzeichen, links nach rechts
$*, /$	Links nach Rechts
$+, -$	Links nach Rechts

### Beispiele

$2^2^3$	# $2^2(2^3) = 2^8 = 256$
$(2^2)^3$	# $(2^2)^3 = 4^3 = 64$
$-1^2$	# $-(1^2) = -1$
$(-1)^2$	# $(-1)^2 = 1$
$2+3/4*5$	# $2+(3/4)*5 = 2+(0.75*5) = 2+3.75 = 5.75$
$2+3/(4*5)$	# $2+3/(4*5) = 2+3/20 = 2+0.15 = 2.15$

## Operatorpräzedenz

### ?Syntax

#### Operator Syntax and Precedence

##### Description

Outlines R syntax and gives the precedence of operators.

##### Details

The following unary and binary operators are defined. They are listed in precedence groups, from highest to lowest.

:: :::	access variables in a namespace
\$ @	component / slot extraction
[ [[	indexing
^	exponentiation (right to left)
- +	unary minus and plus
:	sequence operator
%any%  >	special operators (including %% and %/%)
* /	multiply, divide
+ -	(binary) add, subtract
< > <= >= == !=	ordering and comparison
!	negation
& &&	and
	or
~	as in formulae
-> ->>	rightwards assignment
<- <<-	assignment (right to left)
=	assignment (right to left)
?	help (unary and binary)

Within an expression operators of equal precedence are evaluated from left to right except where indicated. (Note that = is not necessarily an operator.)

---

R und RStudio

Arithmetik, Logik und Präzedenz

**Variablen**

Datenstrukturen

Übungen und Selbstkontrollfragen

## Definition

In der Programmierung ist eine Variable ein abstrakter Behälter für eine Größe, welche im Verlauf eines Rechenprozesses auftritt. Im Normalfall wird eine Variable im Quelltext durch einen Namen bezeichnet und hat eine Adresse im Speicher einer Maschine. Der durch eine Variable repräsentierte Wert kann – im Unterschied zu einer Konstante – zur Laufzeit des Rechenprozesses verändert werden.

*Wikipedia*

## Grundlagen

- Variablen sind vom Programmierenden benannte Platzhalter für Werte
- In 3GL Sprachen wird der Variablentyp durch eine Initialisierungsanweisung festgelegt:

```
VAR A : INTEGER      # A ist eine Variable vom Typ Integer (ganze Zahl)
```

- In 3GL Sprachen wird Variablen durch eine Zuweisungsanweisung ein Wert zugeschrieben:

```
A := 1               # Der Variable A wird der numerische Wert 1 zugewiesen
```

In 4GL Sprachen wie Matlab, Python, R werden Variablen durch Zuweisung initialisiert:

```
a = 1               # a ist eine Variable vom Typ double, ihr Wert ist 1
```

- Der Zuweisungsbefehl in Matlab und Python ist =, der Zuweisungsbefehl in R ist <- oder =.
- Offiziell empfohlen für R ist <-, aus Kohärenzgründen benutzen wir hier =.

# Variablen - Beispiel

Greta geht ins Schreibwarengeschäft und kauft vier Hefte, zwei Stifte und einen Füller. Wie viele analoge Gegenstände kauft Greta insgesamt?

Wir definieren zunächst alle Variablen:

```
hefte = 4      # Definition der Variable 'hefte' und Wertzuweisung 4
stifte = 2     # Definition der Variable 'stifte' und Wertzuweisung 2
fueller = 1    # Definition der Variable 'fuller' und Wertzuweisung 1
```

Nach Zuweisung existieren die Variablen im Arbeitsspeicher, dem sogenannten *Workspace*. Die Variablen können jetzt wie Zahlen in Berechnungen genutzt werden

```
gesamt = hefte + stifte + fueller # Berechnung der Gegenstandsanzahl
print(gesamt)
```

```
> [1] 7
```

Ein Heft kostet einen Euro, ein Stift kostet zwei Euro, und ein Füller kostet 10 Euro. Wie viel Euro muss Greta insgesamt bezahlen?

```
gesamtpreis = hefte*1 + stifte*2 + fueller*10 # Berechnung des Preises
print(gesamtpreis)
```

```
> [1] 18
```

`print()` gibt Variablenwerte in der R Konsole aus.

# Variablen

## Workspace

Nach Zuweisung werden Variablen automatisch im Workspace gespeichert.

- In der RStudio IDE wird das im *Environment* Pane angezeigt.
- `ls()` zeigt die existierenden benutzbaren Variablen im Arbeitsspeicher an

```
ls()                                # Anzeigen aller Variablennamen im Workspace

> [1] "abb_dir"      "error_wrap"  "fueller"     "gesamt"
> [5] "gesamtpreis" "hefte"       "inline_hook" "stifte"
```

## Löschen von Variablen

- `rm()` erlaubt das Löschen von Variablen

```
rm(gesamtpreis)                    # Löschen der Variable Gesamtpreis
ls()

> [1] "abb_dir"      "error_wrap"  "fueller"     "gesamt"
> [5] "hefte"        "inline_hook" "stifte"
```

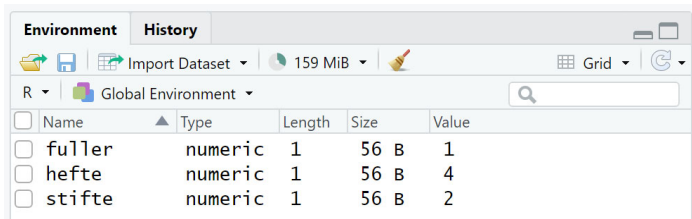
- `rm(list=ls())` löscht alle Variablen

```
rm(list = ls())                    # Löschen aller Variablen
ls()

> character(0)
```

# Variablen

## Workspace



The screenshot shows the R Studio 'Environment' pane. At the top, there are tabs for 'Environment' and 'History'. Below the tabs is a toolbar with icons for file operations and a status bar showing '159 MiB'. The main area displays the 'Global Environment' with a search bar. A table lists the variables in the workspace:

<input type="checkbox"/>	Name	Type	Length	Size	Value
<input type="checkbox"/>	fuller	numeric	1	56 B	1
<input type="checkbox"/>	hefte	numeric	1	56 B	4
<input type="checkbox"/>	stifte	numeric	1	56 B	2



# Variablen

## Variablennamen

### Zulässige Variablennamen

- ... bestehen aus Buchstaben, Zahlen, Punkten (.) und Unterstrichen (\_)
- ... beginnen mit einem Buchstaben oder . nicht gefolgt von einer Zahl
- ... dürfen keine reserved words wie `for`, `if`, `NaN`, usw. sein ( $>?$ reserved)
- ... werden unter `?make.names()` beschrieben

### Sinnvoll Variablennamen

- ... sind kurz ( $\approx$  1 bis 9 Zeichen) und **aussagekräftig**
- ... bestehen nur aus Kleinbuchstaben und Unterstrichen

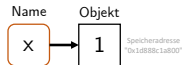
### Anmerkung

- R ist *case-sensitive*. Bsp.  $x \neq X$

# Variablen - Variablenrepräsentation | Binding

`x = 1`

- Intuitiv wird eine Variable genannt mit dem Wert 1 erzeugt.
- De-facto geschehen zwei Dinge:
  1. R erzeugt ein Objekt (Vektor mit Wert 1) mit Speicheradresse `"0x1d888c1a800"`.
  2. R verbindet dieses Objekt mit dem Namen `x`, der das Objekt im Speicher referenziert.



`y = x`

- Intuitiv wird eine Variable genannt `y` mit Wert gleich dem Wert von `x` erzeugt.
- De-facto wird ein neuer Name `y` erzeugt, der dasselbe Objekt referenziert wie `x`.
- Das Objekt (Vektor mit Wert 1) wird nicht kopiert, R spart Arbeitsspeicher.

`y = 1`

- R erzeugt ein Objekt (Vektor mit Wert 1) mit eigener Speicheradresse `"0x1da21b17843"`
- De-facto wird ein neuer Name `y` erzeugt, der ein anderes Objekt referenziert wie `x`.



## Speicheradressen | object address

- Speicheradressen können mit `lobstr::obj_addr()` angezeigt werden

```
library(lobstr)    # Paket `lobstr` laden  
x = 1  
obj_addr(x)
```

```
> [1] "0x557bab769520"
```

- y bekommt die Zuweisung x. De-facto referenziert y dieselbe Speicheradresse wie x

```
y = x  
obj_addr(y)
```

```
> [1] "0x55c5e9b65110"
```

- y bekommt die Zuweisung zu einem neuen Objekt (Vektor mit Wert 1). De-facto referenziert y eine andere Speicheradresse wie x

```
y = 1  
obj_addr(y)
```

```
> [1] "0x55c5e928ba20"
```

### Anmerkungen:

- Ausdrücke der Art `lobstr::obj_addr()` referieren eine Funktion, in diesem Fall `obj_addr()`, bei gleichzeitiger Angabe des Pakets, in diesem Fall `lobstr`.
- Bevor Funktionen eines Pakets verwendet werden können, muss es mit `library()` geladen werden. Ausnahmen sind R Pakete, die per default geladen werden, z.B. `base`.

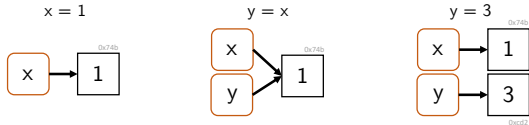
## Variablenrepräsentation | Copy-on-modify

```
x = 1      # Objekt (0x74b) erzeugt, x referenziert Speicheradresse des Objektes
y = x      # y referenziert dieselbe Speicheradresse wie x (0x74b)
y = 3      # y modifiziert, modifizierte Kopie (0xcd2) wird gespeichert
y          # y referenziert jetzt (0xcd2)
```

```
> [1] 3
```

```
x      # x referenziert weiterhin (0x74b)
```

```
> [1] 1
```



R Objekte sind *immutable*, können also nicht verändert werden.

## Variablenrepräsentation | Copy-on-modify

Zur Immutability gibt allerdings zwei Ausnahmen, genannt *Modifications-in-place*

1. Objekte mit nur einem gebundenem Namen werden in-place modifiziert

```
```r
x = c(1,2) # Objekt (0x74b) erzeugt, x referenziert Speicheradresse des Objektes
x[1] = 2   # Objekt (0x74b) veraendert
```
```

- Dieses Verhalten ist allerdings nur in R, nicht innerhalb RStudios reproduzierbar.

2. Environments werden in-place modifiziert (→ Environments und Funktionen).

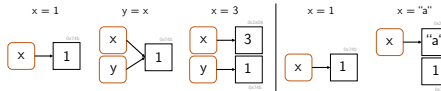
# Variablen

## Variablenrepräsentation | Unbinding und Carbage Collection

Copy-on-modify gilt auch in umgekehrter Reihenfolge

```
x = 1      # Objekt (0x74b) erzeugt, x referenziert Speicheradresse des Objektes
y = x      # y referenziert dieselbe Speicheradresse wie x (0x74b)
x = 3      # Ein neues Objekt (0x2a08) wird erzeugt, x referenziert (0x2a08)
y          # y referenziert weiterhin Objekt (0x74b)
```

> [1] 1



### Unbinding

```
x = 1      # x referenziert Objekt (0x74b)
x = "a"     # x referenziert Objekt (0x2a08), Objekt (0x74b) jetzt ohne Referenz
```

### Carbage collection

- Nicht referenzierte Objekte im Arbeitsspeicher werden automatisch gelöscht.
- Das Löschen geschieht meist erst dann, wenn es wirklich nötig ist.
- Es ist nicht nötig, aktiv die Garbage Collection Funktion gc() zu benutzen.

---

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

**Datenstrukturen**

Übungen und Selbstkontrollfragen

## Klassische Datenstrukturen einer 3GL Programmiersprache

### Fundamentale Datenstrukturen

- Vordefiniert innerhalb der Programmiersprache
- Logische Werte (logical): TRUE, FALSE
- Ganze Zahlen (integer): int8 (-128,...,127), int16 (-32768,..., 32767)
- Gleitkommazahlen (single, double): 1.23456, 12.3456, 123.456, ...
- Zeichen (character): "a", "b", "c", "!"
- Datentyp-spezifische assoziierte Operationen
  - AND, OR (logical) +, - (integer) +,-,\*, / (single), Zeichenkonkatenation (character)

### Zusammengesetzte Datenstrukturen

- Vordefinierte Container zur Zusammenfassung mehrerer Variablen gleichen Datentyps
- Zum Beispiel Vektoren, Listen, Arrays, Matrizen, ...
- Container-spezifische Operationen (Z.B. Vektorindizierung, Matrixmultiplikation, ...)

### Selbstdefinierte Datenstrukturen

- Definition eigener Datenstrukturen aus vordefinierten Datenstrukturen und Containern
- Definition eigener Operationen



## Datenstrukturenkennenlernen beim Erlernen einer Programmiersprache

### Fundamentale Datenstrukturen

- Welche fundamentalen Datenstrukturen bietet die Sprache an?
- Welche Operationen darauf sind bereits definiert?
- Wie lautet die Syntax zur Definition einer Variable eines fundamentalen Datentyps?
- Wie lautet die Syntax, um vordefinierte Operationen aufzurufen?

### Zusammengesetzte Datenstrukturen

- Welche Container und zugehörige Operationen bietet die Programmiersprache?
- Wie lautet die Syntax zum Umgang mit einem Containers?

### Selbstdefinierte Datenstrukturen

- Wie erzeugt man selbstdefinierte Datenstrukturen und zugehörige Operationen?
- Wie lautet die Syntax zum Umgang mit einer selbstdefinierten Datenstruktur?

## Organisation von Daten in R

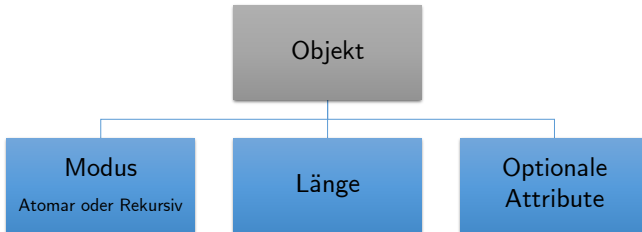
Alles, was in R vorkommt, ist ein **Objekt**

Jedem Objekt kann eindeutig zugeordnet werden

- ein **Modus**
  - Atomar | Komponenten sind vom gleichen Datentyp.
  - Rekursiv | Komponenten können von unterschiedlichem Datentyp sein.
- eine **Länge**
- optional weitere **Attribute**

## Organisation von Daten in R

Alles, was in R vorkommt, ist ein **Objekt**



## Übersicht der R Datentypen

| Datentyp  | Erläuterung   |
|-----------|---|
| logical   | Die beiden logischen Werte TRUE und FALSE                   |
| double    | Gleitkommazahlen  |
| integer   | Ganze Zahlen  |
| complex   | Komplexe Zahlen, hier nicht weiter besprochen               |
| character | Zeichen und Zeichenketten (strings), 'x' oder "Hallo Welt!" |
| raw       | Bytes, hier nicht weiter besprochen                         |

Double und integer werden zusammen auch als numeric bezeichnet.

Viele weitere Typen, hier relevant sind **logical**, **double**, **integer**, **character**.

## Übersicht der R Datentypen

Automatische Festlegung von Datentypen durch Zuweisung

```
b = TRUE           # logical
x = 2.5           # double
y = 1L            # (long) integer
c = 'a'           # character
```

Testen von Datentypen durch `typeof()`

```
typeof(b)
```

```
> [1] "logical"
```

```
typeof(x)
```

```
> [1] "double"
```

```
typeof(y)
```

```
> [1] "integer"
```

```
typeof(c)
```

```
> [1] "character"
```

Testen von Datentypen durch `is.*()`

```
is.logical(x)
```

```
> [1] FALSE
```

```
is.double(x)
```

```
> [1] TRUE
```

## Übersicht atomare Datenstrukturen in R

| Datenstruktur | Erläuterung  |
|---------------|--|
| Vektor        | Container von indizierte Komponenten identischen Typs        |
| Matrix        | Interpretation eines Vektors als zweidimensionaler Container |
| Array         | Interpretation eines Vektors als mehrdimensionaler Container |

⇒ (3) Vektoren, Matrizen, Arrays

## Übersicht rekursive Datenstrukturen in R

| Datenstruktur | Erläuterung   |
|---------------|---|
| Liste         | Container von indizierten Komponenten beliebigen Datentyps<br><br>Insbesondere auch rekursive Struktur, z.B. Liste von Listen |
| Dataframe     | Symbiose aus Liste und Matrix<br><br>Jede Komponente ist Vektor beliebigen Datentyps identischer Länge                        |

⇒ (4) Listen und Dataframes

---

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

**Übungen und Selbstkontrollfragen**



# Übungen und Selbstkontrollfragen

---

1. Installieren Sie R und RStudio auf Ihrem Rechner.
2. Führen Sie die Befehlssequenz auf Folie [R Skripte | Executing and Editing Code](#) aus.
3. Dokumentieren Sie die in dieser Einheit eingeführten R Befehle in einem kommentierten R Skript.
4. Erläutern Sie den Begriff der Operatorpräzedenz.
5. Definieren Sie den Begriff der Variable im Kontext der Programmierung.
6. Erläutern Sie die Begriffe Initialisierungsanweisung und Zuweisungsanweisung für Variablen.
7. Erläutern Sie den Begriff Workspace.
8. Geben Sie jeweils ein Beispiel für einen zulässigen und einen unzulässigen Variablennamen in R.
9. Erläutern Sie die Prozesse, die R im Rahmen einer Zuweisungsanweisung der Form  $x = 1$  durchführt.
10. Erläutern Sie die Begriffe Copy-on-modify und Modify-in-place.
11. Diskutieren Sie die klassischen Datenstrukturen einer 3GL Programmiersprache.
12. Diskutieren Sie die Organisation von Datenstrukturen in R.
13. Wodurch unterscheiden sich eine atomare und ein rekursive Datenstruktur in R?
14. Nennen und erläutern Sie vier zentrale Datentypen in R.
15. Nennen und erläutern Sie vier zentrale atomare Datenstrukturen in R.
16. Nennen und erläutern Sie zwei zentrale rekursive Datenstrukturen in R.

# References

---

- Becker, Richard A., John M. Chambers, and Allen Reeve Wilks. 1988. *The New S Language: A Programming Environment for Data Analysis and Graphics*. Reprint. London: Chapman & Hall.
- Ihaka, Ross, and Robert Gentleman. 1996. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics* 5 (3): 2999–2314.