



# Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

## Modul A2 Einführung in die Forschungsmethoden der Psychologie

- Donnerstags 7.00 - 9.00 Uhr G22A-H2 (Volksbankhörsaal)
- Traditioneller, wenig sinnvoller Kurs
- Sinnvoller, neu überarbeiteter Kurs im Sommersemester 2022

## Modul C2 Computergestützte Datenanalyse

- Donnerstags 15.00 - 17.00 Uhr G40-H231 (hier und jetzt)
- Bearbeitung im WiSe 2021/22 empfohlen

Klausuren A2 und C2 werden sowohl im WiSe 21/22 als auch im SoSe 22 angeboten

Klausurvoraussetzung C2 nach Studienordnung erfolgreicher Abschluss von C1

## (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.

Optimiert 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.

### 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](#).

---

Last change: 2021-08-10

## Was kann man mit R machen?

Datensätze laden, manipulieren, und speichern.

Eine Vielzahl von Berechnungen an verschiedenen Datenstrukturen durchführen.

Eine Vielzahl statistischer Analyse Methoden 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/>

Während der Programmierung und bei bekanntem Funktionsnamen

```
?mean  
help(mean)
```

Für längere Tutorials

```
browseVignettes()
```

<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

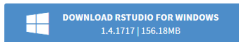
Keine Verbindung zu R Core Team oder R Foundation

## Wie bekommt man RStudio?

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

RStudio Desktop 1.4.1717 - [Release Notes](#)

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:



Requires Windows 10 (64-bit)



## 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](#).

## Was kann man mit RStudio machen?

R Skripte erzeugen, bearbeiten, und laufen lassen

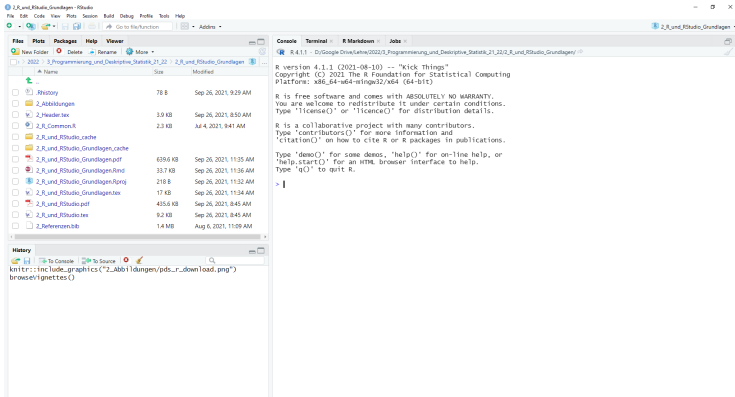
R Skripte in R Projekten organisieren

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

## Was kann man mit RStudio machen?

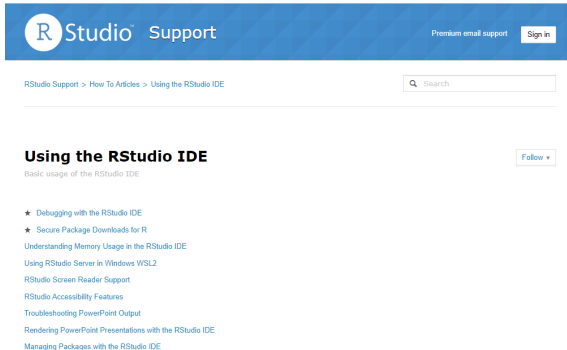
Custom Layout via Tools → Global Options ...



## Wie bekommt man Hilfe zu RStudio?

Googlen

Zur Einführung ⇒ **Using the RStudio IDE**



The screenshot shows the RStudio Support website. At the top is a blue header with the RStudio logo and the word 'Support'. To the right of the header are links for 'Premium email support' and a 'Sign in' button. Below the header is a breadcrumb trail: 'RStudio Support > How To Articles > Using the RStudio IDE'. To the right of the breadcrumb is a search bar with a magnifying glass icon and the word 'Search'. Below the breadcrumb is the title 'Using the RStudio IDE' in bold, followed by a 'Follow' button with a dropdown arrow. Under the title is the subtitle 'Basic usage of the RStudio IDE'. Below the subtitle is a list of links, 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

Eingabe von R Befehlen bei >

Autocomplete mit Tab

Vorherige Befehle mit Cursor ↑

Bereinigen des Konsolenoutputs mit Ctrl + L

Code Ausführungsstopp mit Esc

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

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

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

## R Skripte | Executing and Editing Code

File → New File → R Script oder Ctrl + Shift + N für neue .R Datei

Open File oder Ctrl + O zum Öffnen bestehender .R Datei

Eintippen von

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

Ausführen der 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

**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 merge (with dplyr), visualise (with ggplot2 or ggridges) 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.

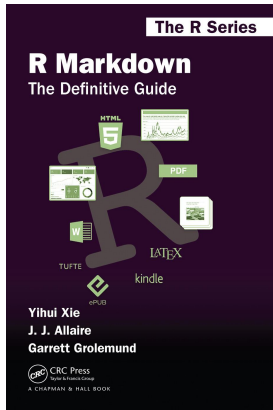
[Project Site Link >](#)



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
```

- `[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.
- 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

Es handelt 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 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

### 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

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

```
hefte = 4      # Definition der Variable 'hefte' und Wertzuweisung 4
stifte = 2     # Definition der Variable 'stifte' und Wertzuweisung 2
fuller = 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 + fuller      # 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 Uta insgesamt bezahlen?

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

```
> [1] 18
```

print() gibt Variablenwerte in der R Konsole aus.

# Variablen

## Workspace

`ls()` zeigt die existierenden benutzbaren Variablen im Arbeitsspeicher an

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

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

`rm()` erlaubt das Löschen von Variablen

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

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

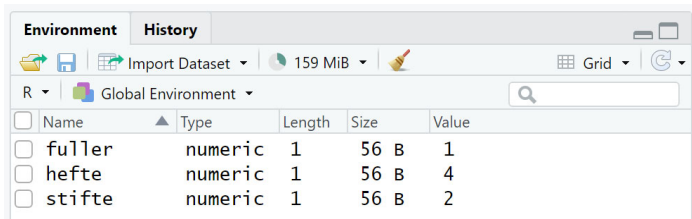
`rm(list=ls())` löscht alle Variablen

```
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



## 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

### Sinnvolle Variablennamen

- ... sind kurz ( $\approx$  1 bis 7 Zeichen) und aussagekräftig
- ... bestehen nur aus Kleinbuchstaben und Unterstrichen

# Variablen

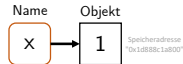
## Variablenrepräsentation | Binding

```
x = 1
```

Intuitiv wird eine Variable genannt  $x$  mit dem Wert 1 erzeugt.

De-facto geschehen zwei Dinge:

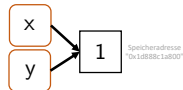
1. R erzeugt ein Objekt (Vektor mit Wert 1) mit Speicheradresse 'lobstr::obj\_addr(x).
2. R verbindet dieses Objekt mit dem Namen  $x$ , der das Objekt im Speicher referenziert.



```
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  $x$ :



Man überzeuge sich mit `lobstr::obj_addr(x)` und `'lobstr::obj_addr(y)`.

Das Objekt (Vektor mit Wert 1) wird nicht kopiert, R spart Arbeitsspeicher.

# Variablen

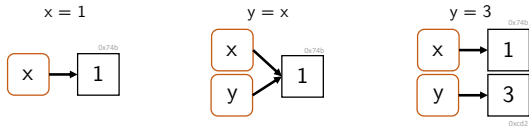
## Variablenrepräsentation | Copy-on-modify

```
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

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

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

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

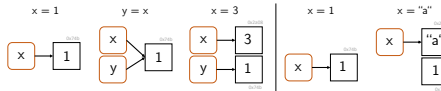
# Variablen

## 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

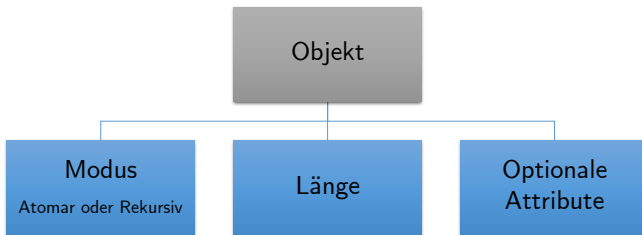
Alles, was in R vorkommt, ist ein **Objekt**

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**.

# Datenstrukturen

## 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 Insbesondere auch rekursive Struktur, z.B. Liste von Listen
Dataframe	Symbiose aus Liste und Matrix Jede Komponente ist Vektor beliebigen Datentyps identischer Länge

⇒ (4) Listen und Dataframes

---

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

**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 den 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.