Computergestützte Datenanalyse

BSc Psychologie SoSe 2021

Prof. Dr. Dirk Ostwald

(2) R und RStudio Grundlagen

Literaturhinweise

Als allgemeine Grundlage dienen Cotton (2013) und Sauer (2019). Die Diskussion der Variablenrepräsentation in R folgt Wickham (2019, Section 2). Die einführende Diskussion der Datenstrukturen in R basiert auf Fußeder (2018).

R und RStudio Grundlagen

- R und RStudio
- Arithmetik, Logik und Präzedenz
- Variablen
- Datenstrukturen
- Übungen und Selbstkontrollfragen

R und RStudio Grundlagen

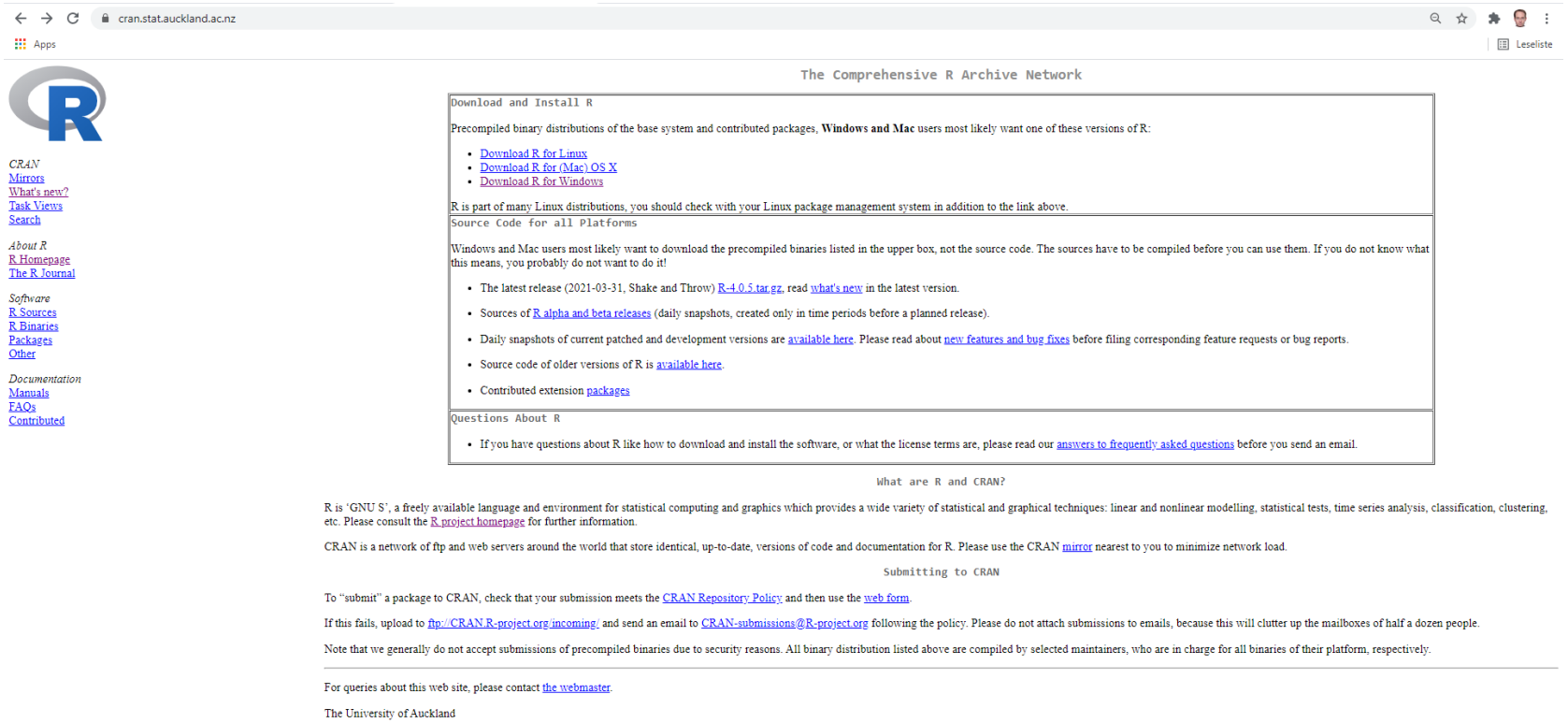
- **R und RStudio**
- Arithmetik, Logik und Präzedenz
- Variablen
- Datenstrukturen
- Übungen und Selbstkontrollfragen

Was ist R?

- Eine Programmiersprache und ein Softwarepaket.
- Entwickelt von Ihaka and Gentleman (1996).
- Freier Dialekt der propietären Software S (Becker et al., 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 20000 beigetragenen R Paketen (Erweiterungen)
- Evolviert und konservativ im Kern, konsistent und progressiv in R Paketen.

Wie bekommt man R?

Runterladen (z.B. `https://cran.stat.auckland.ac.nz/`) und installieren



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.

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

- In einer ansprechenden Umgebung programmieren ($\rightarrow$ RStudio).
- Scientific Computing ($\rightarrow$ Matlab, Julia).
- Psychologische Experimente programmieren ($\rightarrow$ Python, Matlab)
- Apps oder Webseiten programmieren ($\rightarrow$ Java)

Wie bekommt man Hilfe zu R?

- Googlen
- `https://stackoverflow.com/`
- Während der Programmierung und bei bekanntem Funktionsnamen

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

- Für längere Tutorials

```
>browseVignettes()
```

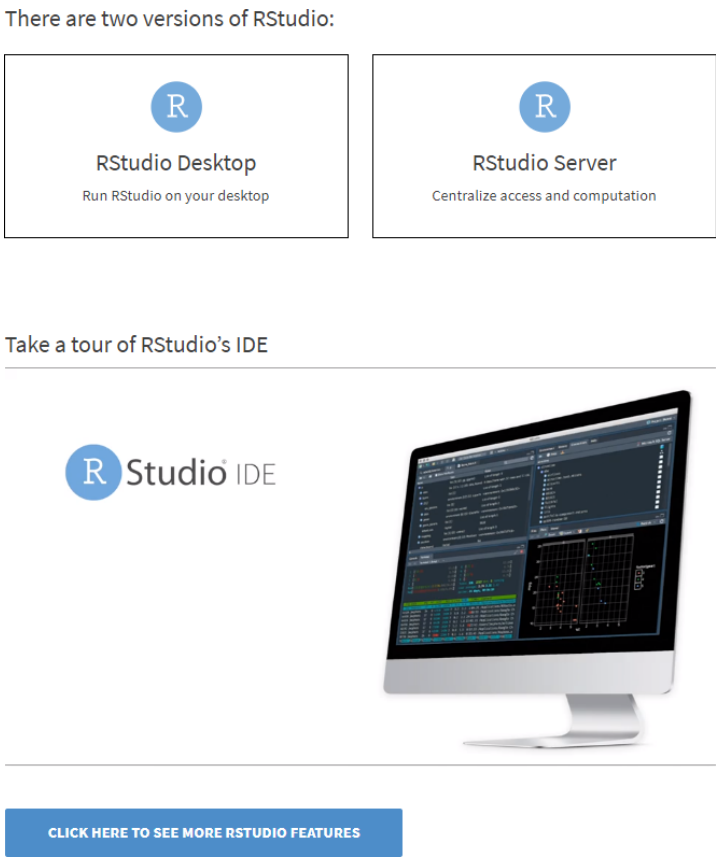
- `https://rseek.org/`
- `https://www.rstudio.com/resources/cheatsheets/`
- `https://www.r-bloggers.com/`

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

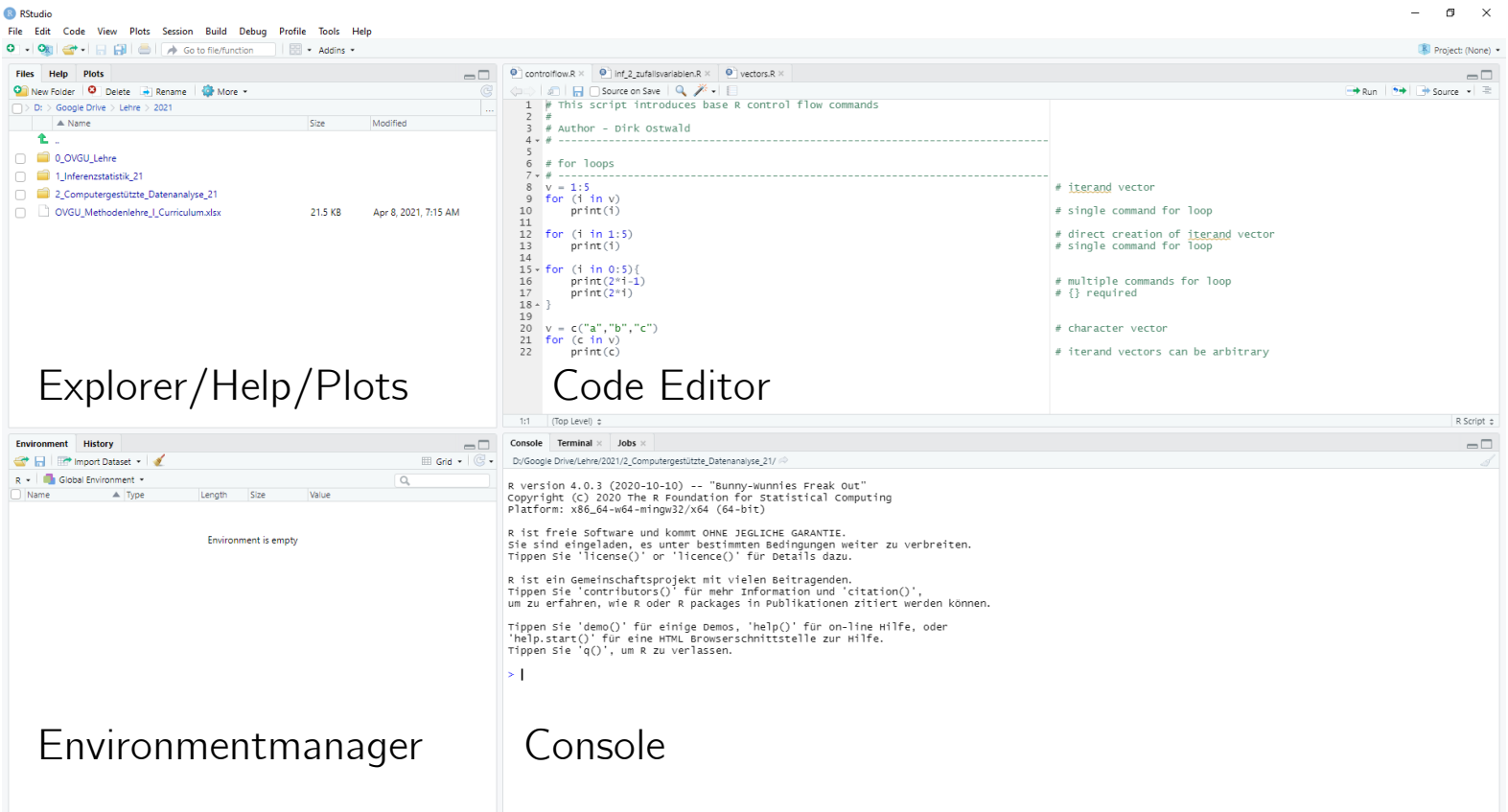


Was kann man mit RStudio machen?

- R Skripte erzeugen, bearbeiten, und laufen lassen
- 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

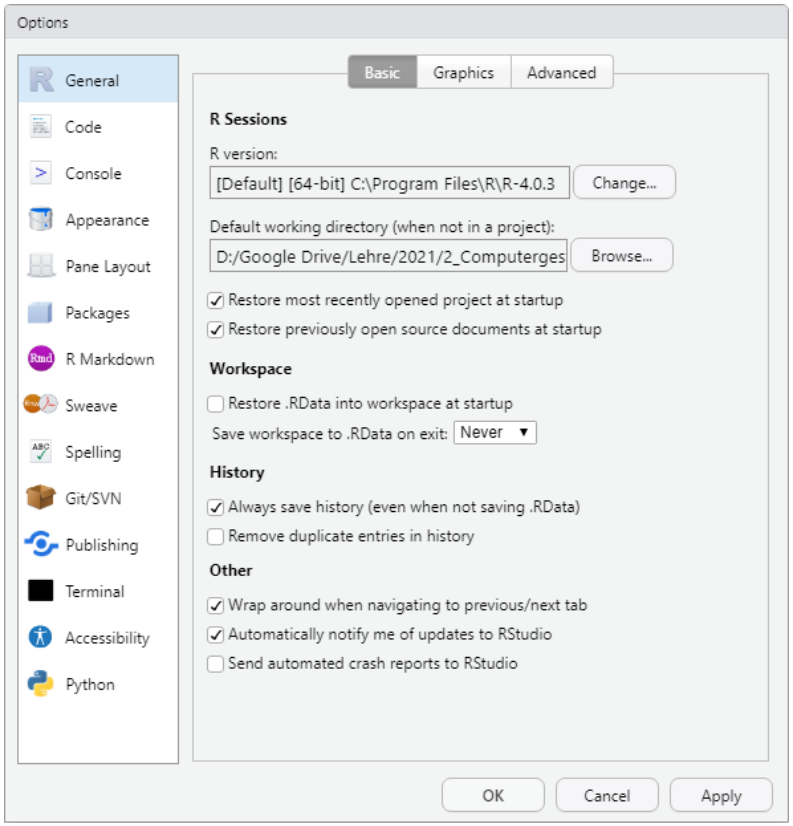
Was kann man mit RStudio machen?

Custom Layout



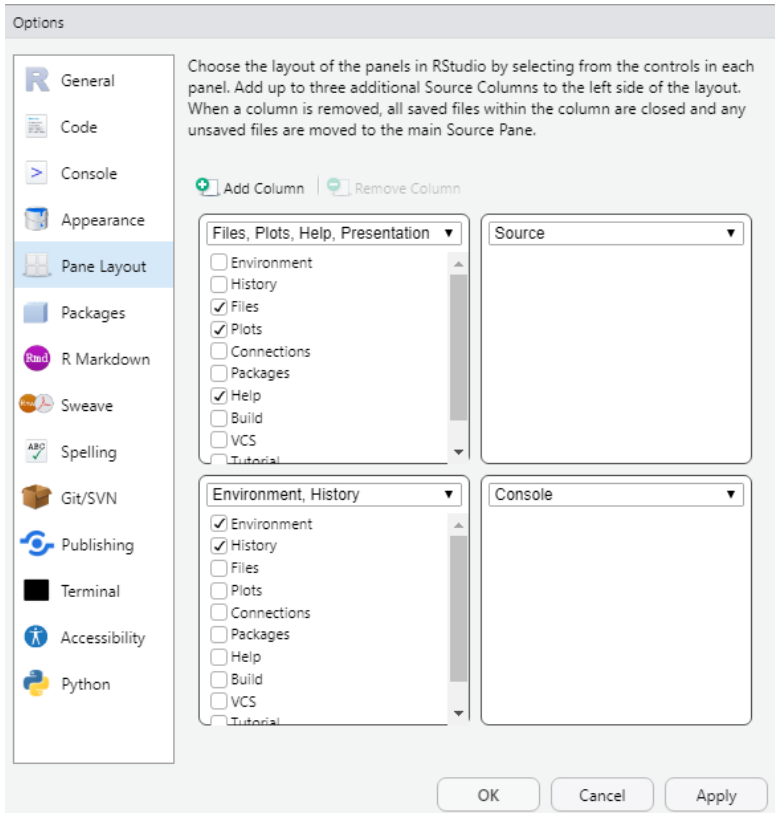
Was kann man mit RStudio machen?

→ Tools → Global Options ...



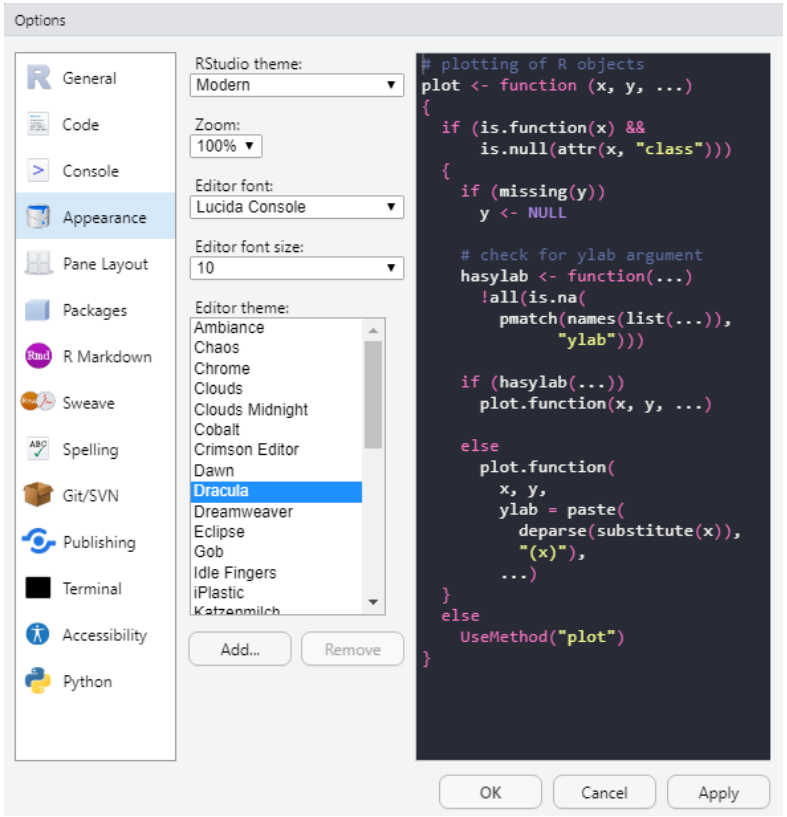
Was kann man mit RStudio machen?

→ Tools → Global Options ...



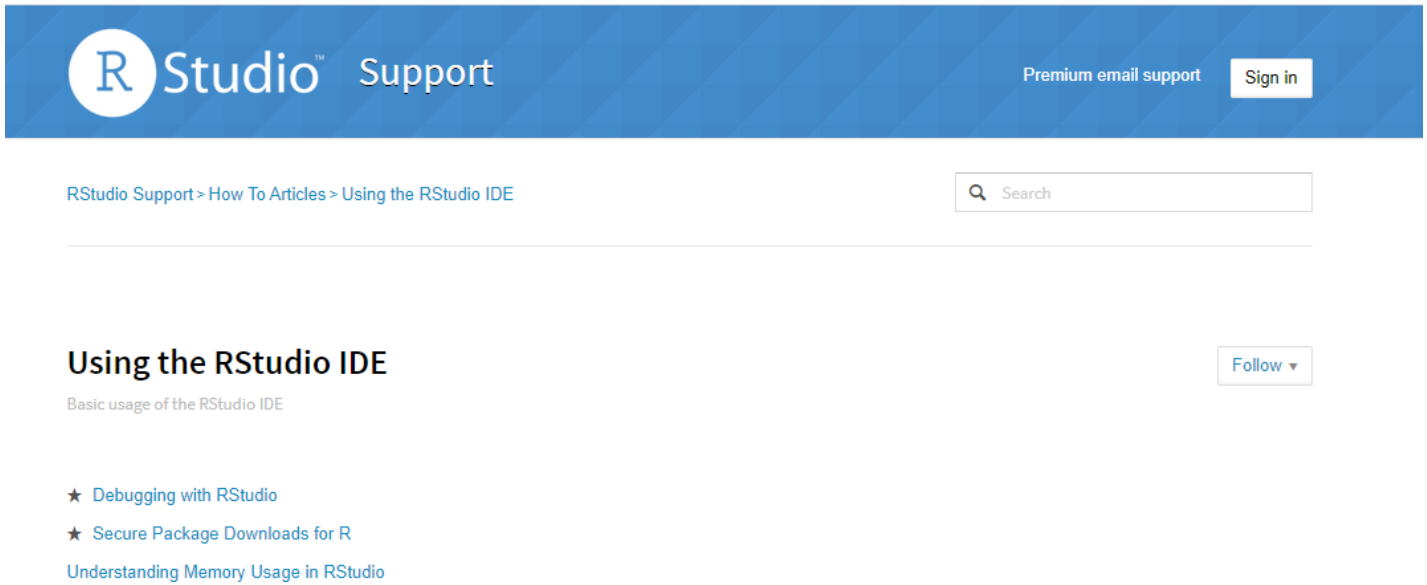
Was kann man mit RStudio machen?

→ Tools → Global Options ...



Wie bekommt man Hilfe zu RStudio?

- Googlen
- `https://support.rstudio.com/hc/en-us/sections/200107586-Using-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
- Code-Snippets auf diesen Folien der Form

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

sollten Sie immer in der Konsole und einem R Skript 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 ausgefuehrt
```

- 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

R und RStudio Grundlagen

- R und RStudio
- **Arithmetik, Logik und Präzedenz**
- Variablen
- Datenstrukturen
- Übungen und Selbstkontrollfragen

R Konsole als Taschenrechner

```
1+1
[1] 2
2*3
[1] 6
6/2
[1] 3
sqrt(2)
[1] 1.414214
exp(0)
[1] 1
log(0)
[1] -Inf
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
$\hat{}$ 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
abs(x)	Betrag
sqrt(x)	Wurzel
ceiling(x)	Aufrunden (ceiling(2.7) = 3)
floor(x)	Abrunden (floor(2.7) = 2)
round(x)	Mathematisches Runden (round(2.5) = 2)
exp(x)	Exponentialfunktion
log(x)	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
`+`(2,3)
```

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!

```
?Syntax
```

  - Lieber Klammern setzen, als keine Klammern setzen!
  - Immer nachschauen, ob Berechnungen die erwarteten Ergebnisse liefern!

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^(2^3)   = 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 Grundlagen

- 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.

(adaptiert von 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 $=$

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
[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 # Berechung des Preises
[1] 18
```

Workspace

ls() zeigt die existierenden benutzbaren Variablen im Arbeitsspeicher an

```
ls()            # Anzeigen aller Variablennamen im Workspace
[1] "fuller"   "gesamt"   "gesamtpreis"   "hefte"    "stifte"
```

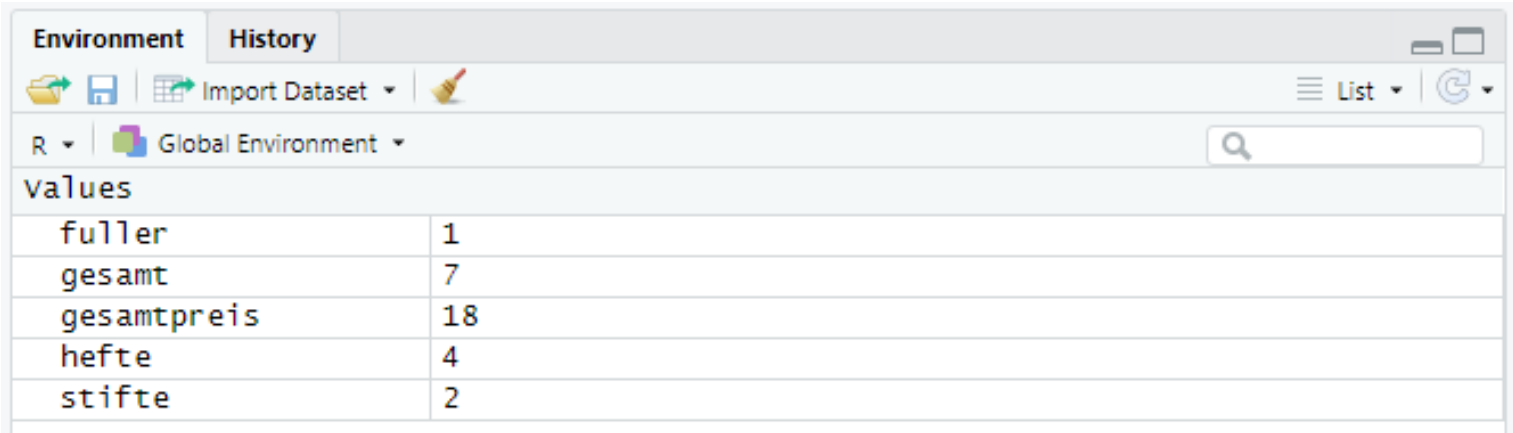
rm() erlaubt das Löschen von Variablen

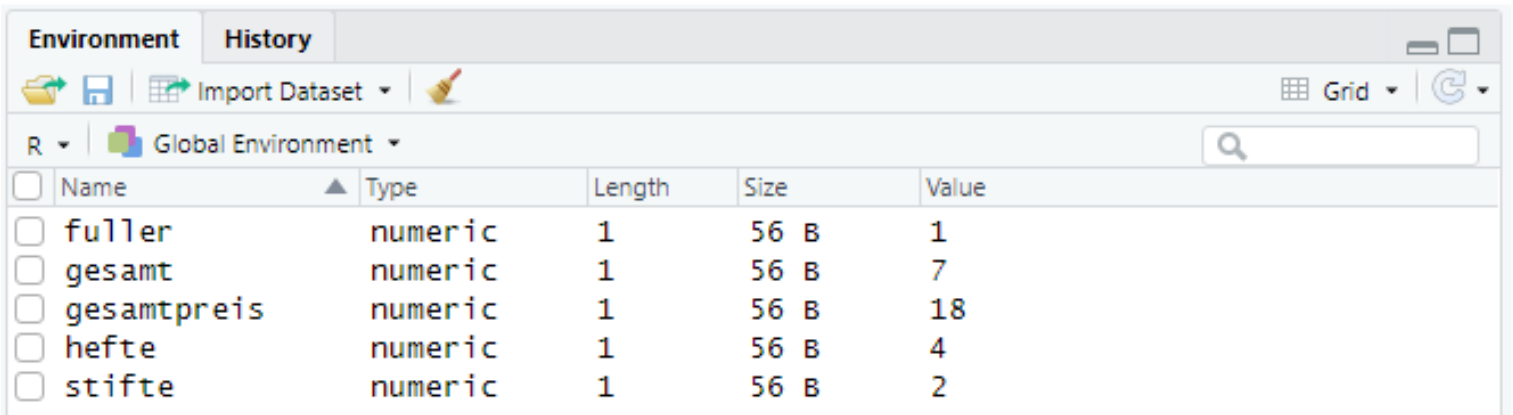
```
rm(gesamtpreis)    # Loeschen der Variable Gesamtpreis
ls()
[1] "fuller"   "gesamt"   "hefte"    "stifte"
```

rm(list=ls()) löscht alle Variablen

```
rm(list = ls())    # Loeschen aller Variablen
ls()
[1] character(0)     # Leerer Workspace
```

Workspace


Environment
History
Import Dataset
List
R
Global Environment
Values
fuller 1
gesamt 7
gesamtpreis 18
hefte 4
stifte 2


Environment
History
Import Dataset
Grid
R
Global Environment
Name
Type
Length
Size
Value
fuller numeric 1 56 B 1
gesamt numeric 1 56 B 7
gesamtpreis numeric 1 56 B 18
hefte numeric 1 56 B 4
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 reserverd 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

Variablenrepräsentation | Binding

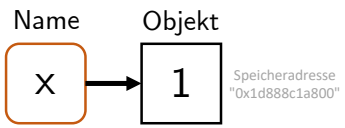
```
x = 1
```

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

De-facto geschehen zwei Dinge:

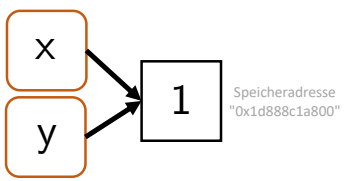
(1) R erzeugt ein Objekt (Vektor mit Wert 1) mit Speicheradresse lobstr::obj_addr(x).

(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:

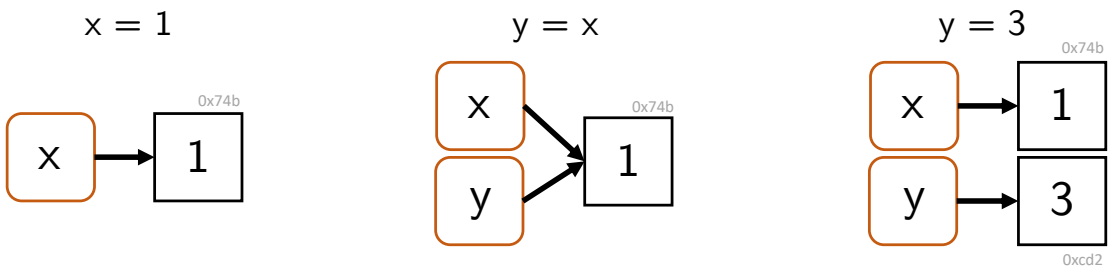


Man überzeuge sich mit lobstr::obj_addr(x) und lobstr::obj_addr(y).

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

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
[1] = 3    # y referenziert jetzt (0xcd2)
x
[1] = 1    # x referenziert weiterhin (0x74b)
```



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

Es gibt allerdings zwei Ausnahmen (*modify-in-place*)

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

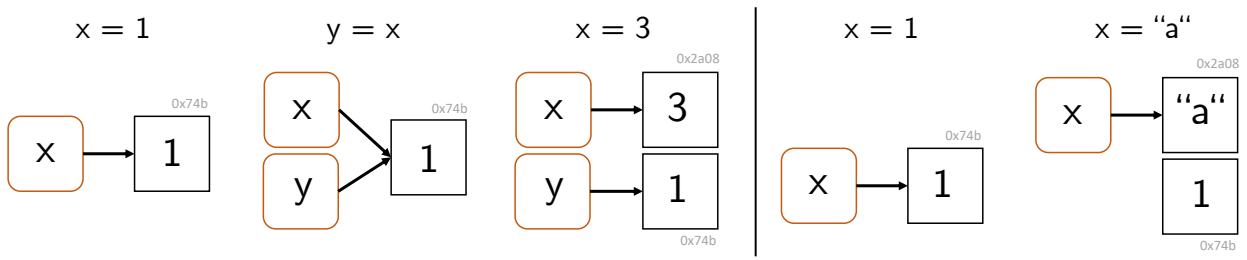
```
x = 1      # Objekt (0x74b) erzeugt, x referenziert Speicheradresse des Objektes
x[1] = 2   # Objekt (0x74b) veraendert
```

- Dieses Verhalten ist allerdings nur in R, nicht innerhalb RStudios reproduzierbar.
- Environments werden in-place modifiziert (→ Environments und Funktionen).

(NEU) 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
[1] 1     # y referenziert weiterhin Objekt (0x74b)
```



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 Grundlagen

- 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
  - Z.B. 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

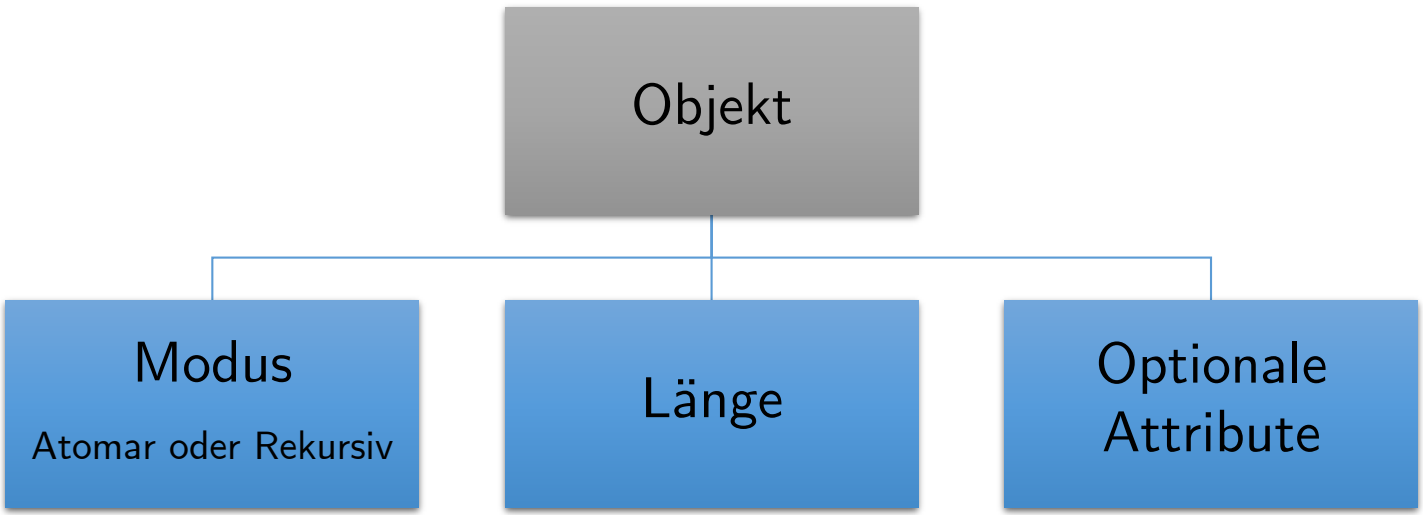
Alles, was in R vorkommt, ist ein **Objekt**

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 der Datenstrukturen 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 Typen in R

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 Datetypen 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
Faktor	Einteilung von Daten in Kategorien

$\Rightarrow$ (3) Vektoren, Matrizen, Arrays, Faktoren

Übersicht rekursive Datenstrukturen in R

Datenstruktur	Erläuterung
Liste	Container von indizierten Komponenten beliebigen Typs
	Insbesondere auch rekursive Struktur, z.B. Liste von Listen
Dataframe	Symbiose aus Liste und Matrix
	Jede Komponente ist Vektor beliebigen Typs identischer Länge
Tibble	Optimierter Dataframe

$\Rightarrow$ (4) Listen, Dataframes, Tibbles

R und RStudio Grundlagen

- 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 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.

Literatur

Becker, R. A., Chambers, J. M., and Wilks, A. R. (1988). *The New S Language: A Programming Environment for Data Analysis and Graphics*. Chapman & Hall, London, reprint edition.

Cotton, R. (2013). *Learning R*. O'Reilly, Beijing ; Sebastopol, CA, first edition edition.

Fußeder, W. (2018). übersicht über Datentypen in R.

Ihaka, R. and Gentleman, R. (1996). R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5(3):2999–314.

Sauer, S. (2019). *Moderne Datenanalyse mit R: Daten einlesen, aufbereiten, visualisieren, modellieren und kommunizieren*. FOM-Edition. Springer Fachmedien Wiesbaden, Wiesbaden.

Wickham, H. (2019). *Advanced R, Second Edition*. CRC Press.