Management of Information Systems

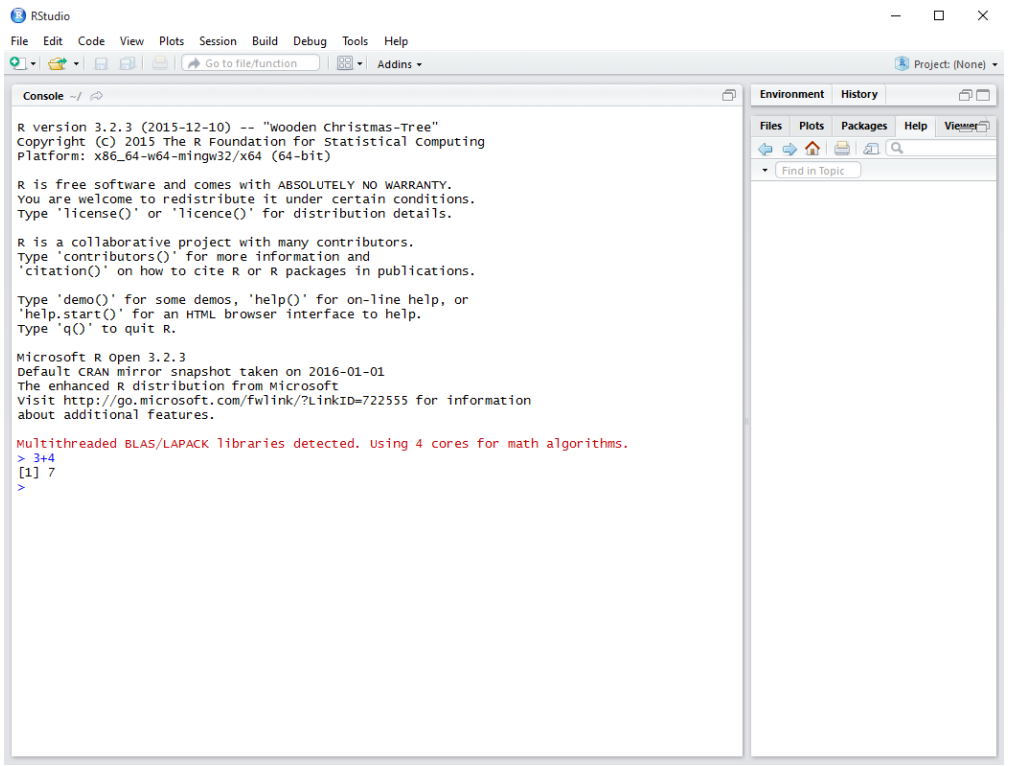
Session 1: Einführung in R

Gliederung

Was ist R?

- Programmiersprache für statistische Berechnungen und Visualisierungen
- Open Source
- Unterstützt von vielen Betriebssystemen (Linux, Mac OS X, Windows)
- Oft benutzt in Psychologie, Medizin, Statistik, Ökonometrie und empirischer Forschung

Download unter: `http://www.r-project.org`



RStudio als Editor

- Entwicklungsumgebung mit grafischer Oberfläche
- Ermöglicht das Speichern, Editieren und Exportieren von Code
- Interaktive Hilfefunktionen

Download von RStudio Desktop unter:
`https://www.rstudio.com/products/rstudio/download/`

Gliederung

Arithmetische Operationen

- Arithmetischen Operatoren +, -, *, /, ^ führen entsprechende Rechnung aus

```
1 + 2 * 3
```

```
## [1] 7
```

```
3 / 4 + 2
```

```
## [1] 2.75
```

```
2 * pi - pi
```

```
## [1] 3.141593
```

```
0 / 0
```

```
## [1] NaN
```

Logische Operatoren

- Logischen Operatoren <, <=, ==, !=, >=, > geben Boolsche Werte zurück (d.h. `TRUE` oder `FALSE`)

```
3 > 4
```

```
## [1] FALSE
```

```
3 <= 4
```

```
## [1] TRUE
```

```
4 == 4
```

```
## [1] TRUE
```

```
3 != 4
```

```
## [1] TRUE
```

Klammern, Kommentare und Dezimalpunkte

- Klammern werden zuerst ausgewertet

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

```
## [1] 18
```

- Dezimalzahlen werden mit einem Punkt geschrieben (kein Komma!)

```
3.1415
```

```
## [1] 3.1415
```

- Kommentare kann man mit # einfügen, diese werden von R ignoriert

```
3 + 4 # Ich bin ein Kommentar.
```

```
## [1] 7
```

Mathematische Funktionen

- Quadratwurzel

```
sqrt(1 + 1)
```

```
## [1] 1.414214
```

- Logarithmus zur Basis 10

```
log10(10 * 10 * 10)
```

```
## [1] 3
```

- Logarithmus zur Basis e

```
log(9)
```

```
## [1] 2.197225
```

Mathematische Funktionen

- Sinus (genauso `cos()`, `tan()`)

```
sin(pi)
```

```
## [1] 1.224606e-16
```

- Runden auf die nächste Ganzzahl

```
round(sin(pi))
```

```
## [1] 0
```

- Absolutbetrag

```
abs(3 - 4)
```

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

Übung: Operationen und Funktionen

Was ist der Rückgabewert von . . . ?

```
a) sin(pi) == 0
b) log(10)
c) sin(pi) / round(sin(pi))
d) cos(1,5 - 0,5 * 3)
```

Variablen

- Speichern Werte, um diese später wiederverwenden zu können
- Rechter Wert wird linker Variable mit <- zugeordnet
- Der Wert rechts von <- wird berechnet, aber nicht am Bildschirm angezeigt
- Erlaubte Zeichen für Variablennamen sind `a-z`, `A-Z`, `_` und `.`
- Achtung: R ist "case-sensitive", d.h. Groß-/Kleinschreibung wird unterschieden

```
x <- 2
x
```

```
## [1] 2
```

Variablen

Beispiel

```
x <- 2
x
```

```
## [1] 2
```

```
x + 3
```

```
## [1] 5
```

```
x
```

```
## [1] 2
```

```
x <- x + 4
x
```

```
## [1] 6
```

Übung: Variablen

Was ist der Wert von `x`?

```
a) 13 = x
b) x = 17
c) x == x
d) x <- x-5
```

Übung: Variablen

Was ist der Wert von z?

```
x <- 2
x <- x + 1
y <- 4
z <- x + y
x <- x + 1
z <- z + x
```

Strings

- Zeichenketten (sogenannte “strings”“) werden in doppelten Anführungszeichen geschrieben

```
"Text"
```

```
## [1] "Text"
```

```
"3.14"
```

```
## [1] "3.14"
```

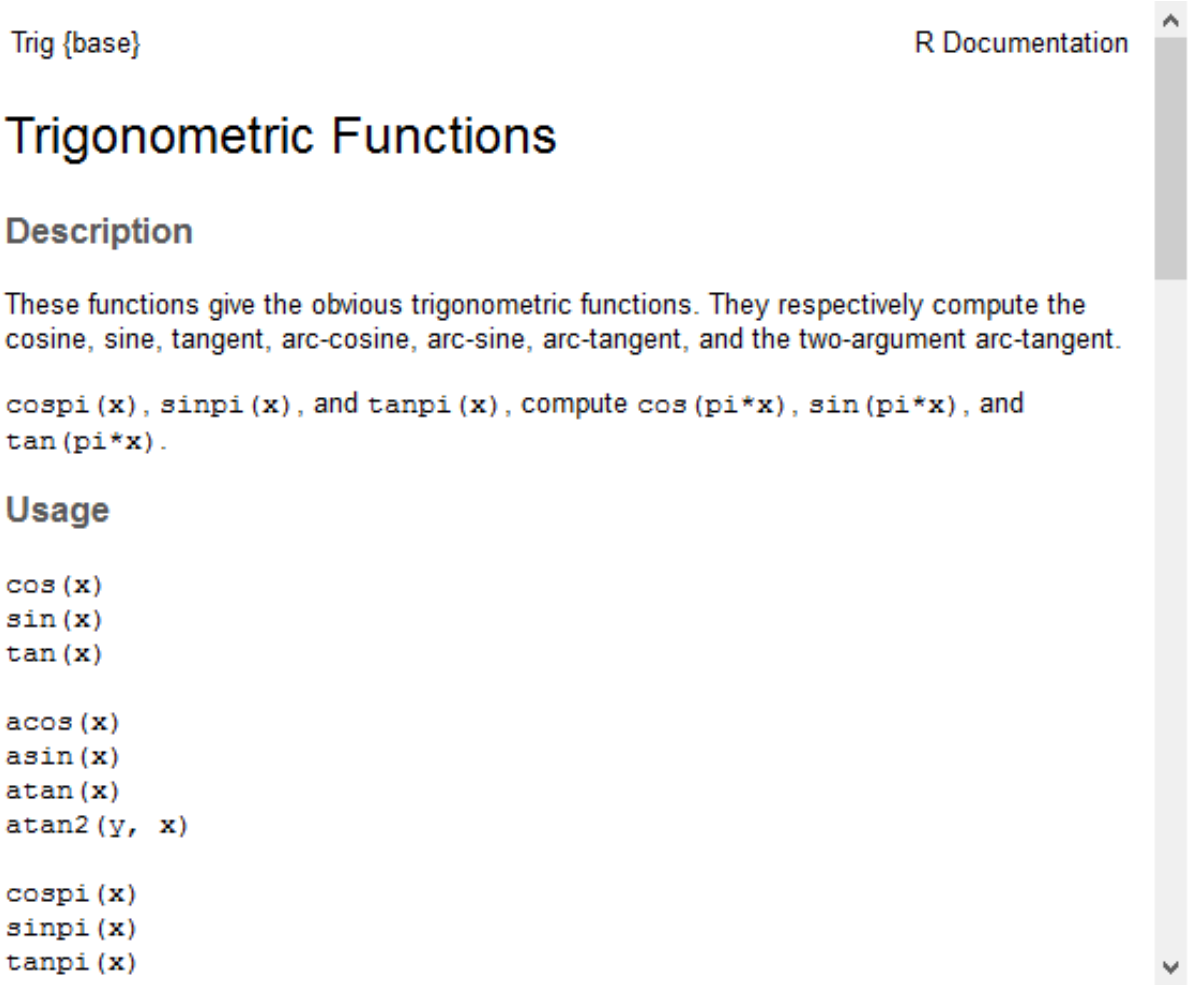
```
"3.14" + 1 # Erzeugt eine Fehlermeldung
```

```
## Error in "3.14" + 1: non-numeric argument to binary operator
```

Hilfe

- Mit `help(func)` kann man für jede Funktion eine Hilfsseite aufrufen

```
help(sin)
```



Trig {base} R Documentation

Trigonometric Functions

Description

These functions give the obvious trigonometric functions. They respectively compute the cosine, sine, tangent, arc-cosine, arc-sine, arc-tangent, and the two-argument arc-tangent.

`cospi(x)`, `sinpi(x)`, and `tanpi(x)`, compute `cos(pi*x)`, `sin(pi*x)`, and `tan(pi*x)`.

Usage

```
cos(x)
sin(x)
tan(x)

acos(x)
asin(x)
atan(x)
atan2(y, x)

cospi(x)
sinpi(x)
tanpi(x)
```

Gliederung

Vektoren

- Vektoren werden durch `c(...)` erzeugt

```
x <- c(4, 0, 6)
x
```

```
## [1] 4 0 6
```

- Auf einzelne Komponenten kann man mit `[...]` zugreifen

```
x[1]
```

```
## [1] 4
```

Vektoren

- `rep(val, count)` erzeugt Vektoren mit sich wiederholenden Elementen

```
rep(1, 5)
```

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

```
rep(c(1, 2), 3)
```

```
## [1] 1 2 1 2 1 2
```

- Auf mehrere Komponenten zugreifen

```
x <- c(4, 0, 6)
x[c(2, 3)]
```

```
## [1] 0 6
```

Vektoren

- Ausschließen bestimmter Komponenten mit -

```
x[-1]
```

```
## [1] 0 6
```

- Nullvektor der Länge `n` mit `numeric(n)`

```
numeric(4)
```

```
## [1] 0 0 0 0
```

Vektoren

- Dimension eines Vektors mit `length()`

```
length(x)
```

```
## [1] 3
```

- Einzelne Komponenten von Vektoren ändern

```
x <- c(4, 0, 6)
x[1] <- 2
x
```

```
## [1] 2 0 6
```

Vektoren

- Vektoren verlängern

```
y <- c(x, 8)
y
```

```
## [1] 2 0 6 8
```

- Vektoren verketten

```
z <- c(x, y)
z
```

```
## [1] 2 0 6 2 0 6 8
```

Funktionen auf Vektoren

- Summe aller Komponenten

```
sum(x)
```

```
## [1] 8
```

- Mittelwert $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$

```
x <- c(1, 3, 0)
mean(x)
```

```
## [1] 1.333333
```

- Korrigierte Stichprobenvarianz $s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$

```
var(x)
```

```
## [1] 2.333333
```

Übung: Vektoren

Wie berechnet man die Standardabweichung s von $x = (1, 4, 9)^T$?

```
a) sqr(var(x))
b) sqrt(var(x))
c) sd(x)
```

Vektoroperationen

- Operationen werden komponentenweise ausgeführt

```
x <- c(1, 2)
y <- c(5, 6)
10 * x
```

```
## [1] 10 20
```

```
x + y
```

```
## [1] 6 8
```

```
10 + x
```

```
## [1] 11 12
```

Folgen

- Folgen ganzer Zahlen

```
1:4
```

```
## [1] 1 2 3 4
```

```
4:1
```

```
## [1] 4 3 2 1
```

- Beliebige Folgen

```
(1:10) / 10
```

```
##  [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

```
seq(4, 5, 0.1) # Notation: Start, Ende, Schrittgröße
```

```
##  [1] 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.0
```

Übung: Vektoren

Wie berechnet man $\sum_{i=1}^{100} i$?

```
a) sum(1:100)
b) sum(1, 100)
c) sum(1 - 100)
```

Gliederung

Matrizen und Datensätze

- Matrizen werden mit `matrix(...)` konstruiert

```
A <- matrix(c(1, 0, 0, 0, 1, 0, 0, 0, 1),
            nrow=3, ncol=3, byrow=TRUE)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

Matrizen und Datensätze

- Alternativ kann man Vektoren mit `cbind(...)` zu einer Matrix kombinieren
- Um verschiedene, nicht-numerischen Datentypen zu speichern, braucht man `as.data.frame(...)`

```
height <- c(163, 186, 172)
name <- c("Anne", "Leyla", "Elias")
M <- as.data.frame(cbind(height, name))
M
```

```
##   height  name
## 1    163  Anne
## 2    186 Leyla
## 3    172 Elias
```

Comma Seperated Values (CSV)

- Das Format von Excel wird nicht vollständig unterstützt
- Stattdessen nutzt man in emprischen Arbeiten bevorzugt CSV
- Export als Comma Seperated Values (CSV)

  Beispieldatei: `Session1_persons.csv`

```
name,height,shoesize
Anne,163,38
Leyla,186,41
Elias,172,41
```

Comma Seperated Values (CSV)

- Import via `read.csv(filename,...)`
  - Falls Spaltennamen vorhanden, werden diese mit `header = TRUE` importiert
  - `sep=","` spezifiziert, welches Zeichen Spalten trennt

```
d <- read.csv("Session1_persons.csv", header=TRUE, sep=",")
d
```

```
##    name height shoesize
## 1  Anne    163       38
## 2 Leyla    186       41
## 3 Elias    172       41
## 4 Marie    166       37
## 5   Ali    180       44
```

- Anstelle den Pfad zur Datei einzugeben, kann dieser auch interaktiv ausgewählt werden

```
d <- read.csv(file.choose(), header=TRUE, sep=",")
```

Matrizen und Datensätze

- Nur die ersten 6 Zeilen ausgeben

```
head(d)
```

```
##    name height shoesize
## 1  Anne    163       38
## 2 Leyla    186       41
## 3 Elias    172       41
## 4 Marie    166       37
## 5   Ali    180       44
```

- Details zum Aufbau und den Spaltennamen einer Matrix anzeigen

```
str(d)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ name    : Factor w/ 5 levels "Ali","Anne","Elias",..: 2 4
##  $ height  : int  163 186 172 166 180
##  $ shoesize: int  38 41 41 37 44
```

Matrizen und Datensätze

- Dimension der Matrix

```
dim(d)
```

```
## [1] 5 3
```

```
nrow(d) #Anzahl der Zeilen
```

```
## [1] 5
```

```
ncol(d) #Anzahl der Spalten
```

```
## [1] 3
```

Matrizen und Datensätze

- Spalten mit ihrem Namen auswählen

```
d$height
```

```
## [1] 163 186 172 166 180
```

```
d[["height"]]
```

```
## [1] 163 186 172 166 180
```

- Auf einen einzelnen Eintrag in einer Matrix zugreifen

```
d[1, 2]
```

```
## [1] 163
```

Matrizen und Datensätze

- Mit einer Bedingung eine Teilmenge der Zeilen aussuchen

```
d[d$height > 170, ]
```

```
##    name height shoesize
## 2 Leyla    186       41
## 3 Elias    172       41
## 5   Ali    180       44
```

- Mehrere Bedingungen verknüpfen (& heißt "und", | heißt "oder")

```
d[d$height > 170 & d$shoesize <= 41, ]
```

```
##    name height shoesize
## 2 Leyla    186       41
## 3 Elias    172       41
```

Matrizen und Datensätze

- Spalten hinzufügen

```
d[["heightInInch"]] <- d$height / 2.51
d$heightInInch
```

```
## [1] 64.94024 74.10359 68.52590 66.13546 71.71315
```

- Spaltennamen mit `colnames(...)` auslesen

```
colnames(d)
```

```
## [1] "name"         "height"       "shoesize"
## [4] "heightInInch"
```

- Spaltennamen ändern

```
colnames(d) <- c("name", "waist", "weight", "shoes")
colnames(d)
```

```
## [1] "name"   "waist"  "weight" "shoes"
```

Übung: Matrizen und Datensätze

Du hast den folgenden Datensatz `mensa`.

1. Füg eine Spalte `preis_mit` hinzu, die den Preis für Mitarbeiter*Innen anzeigt. Es handelt sich dabei um 2,80€ für den Schnellen Teller und 3,80€ für Essen 1 und 2.
2. Lass dir alle Gerichte ausgeben, die dich weniger als 2€ kosten.

```
mensa
```

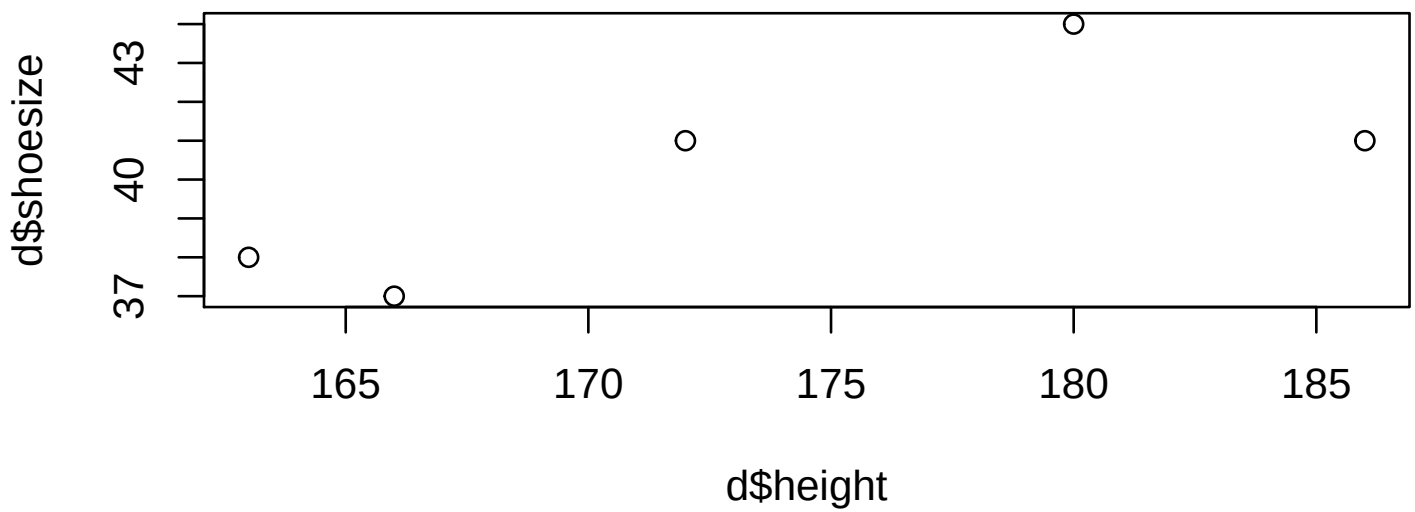
```
##                name   gericht preis_stud
## 1          Essen 1    Gnocci        2.8
## 2          Essen 2 MSC Fisch        2.8
## 3 Schneller Teller Milchreis       1.65
```

Gliederung

Liniendiagramme

- Generieren von einfachen Diagrammen mit `plot(...)`

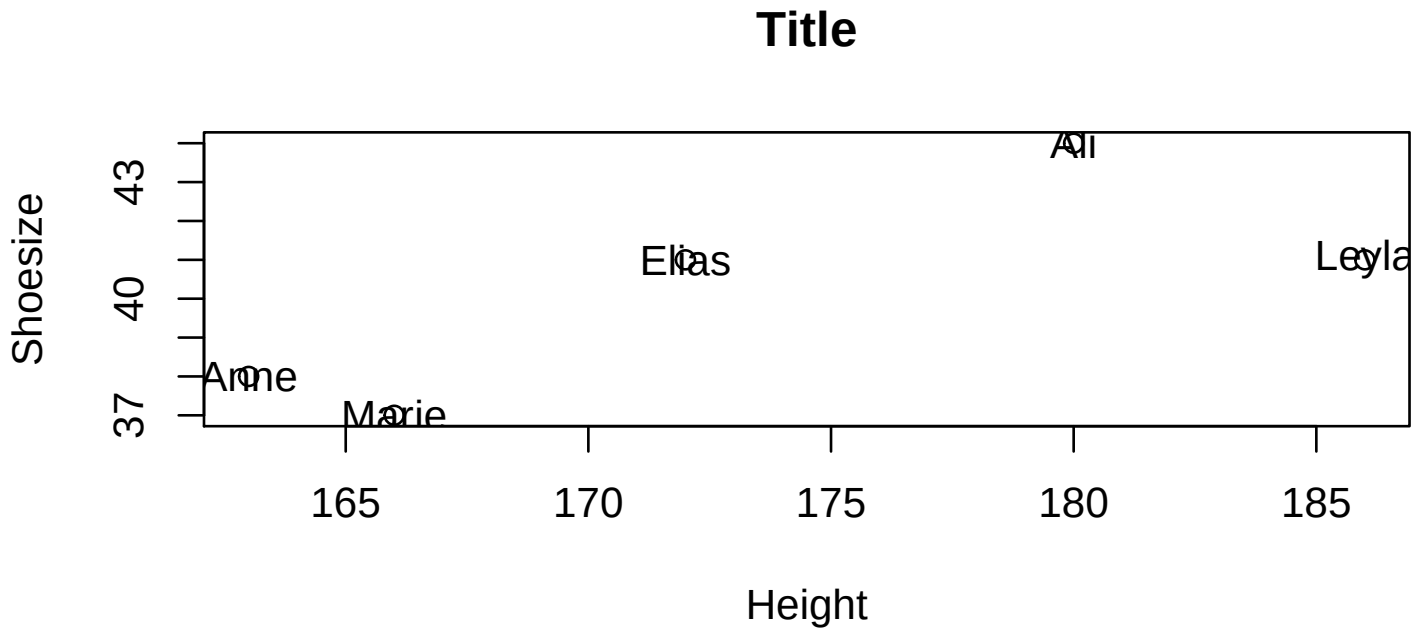
```
d <- read.csv("Session1_persons.csv", header=TRUE, sep=",")
plot(d$height, d$shoesize)
```



Liniendiagramme

- Aussehen kann durch das Setzen verschiedener Parameter beeinflusst werden, wie Titel und Axenbeschriftungen

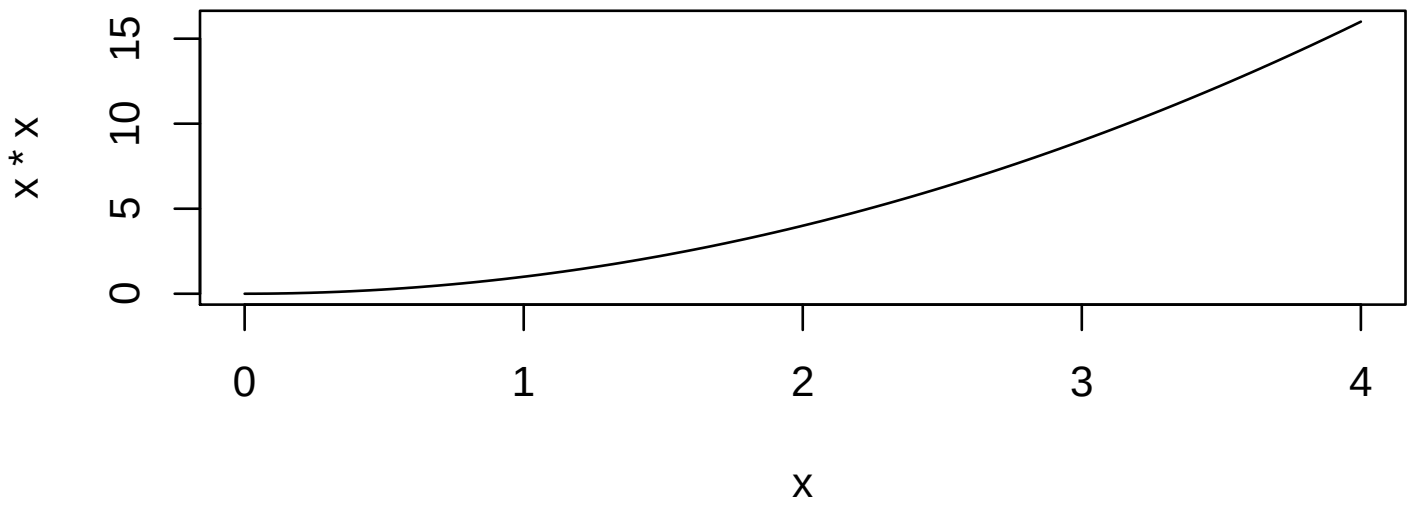
```
plot(d$height, d$shoesize, main="Title", # Titel Plot
     xlab="Height", ylab="Shoesize",     # Titel Achsen
text(d$height, d$shoesize, d$name))      # d$name sind Label
```



Liniendiagramme

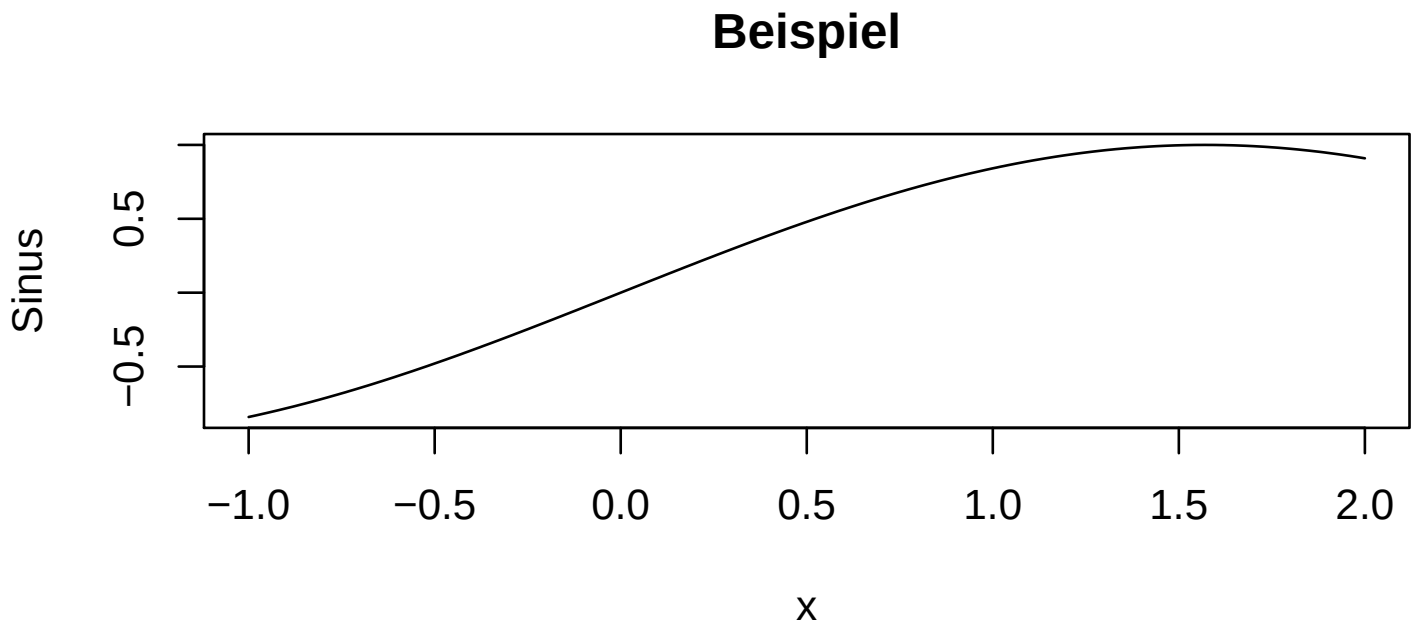
- Punkte können via `type="l"` zu einer durchgezogenen Linie verbunden werden

```
x <- seq(0, 4, 0.01)
plot(x, x * x, type="l")
```



Übung: Visualisierung

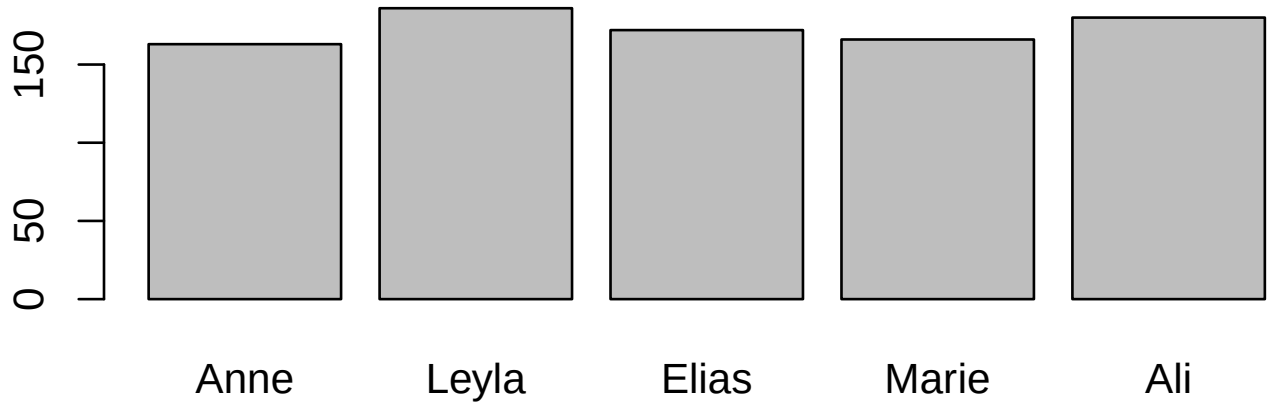
Wie erzeugt man die angegebene Grafik?



Balkendiagramme

- Balkendiagramme via `barplot(...)`

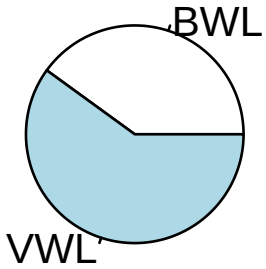
```
d <- read.csv("Session1_persons.csv", header=TRUE, sep=",")
barplot(d$height, names.arg=d$name)
```



Kuchendiagramme

- Kuchendiagramm via `pie(...)`

```
study <- c("VWL", "BWL", "BWL", "VWL", "VWL")
d <- cbind(d, study)
abs.freq <- table(d$study) # abs. Häufigkeit mit table(...)
pie(abs.freq)
```

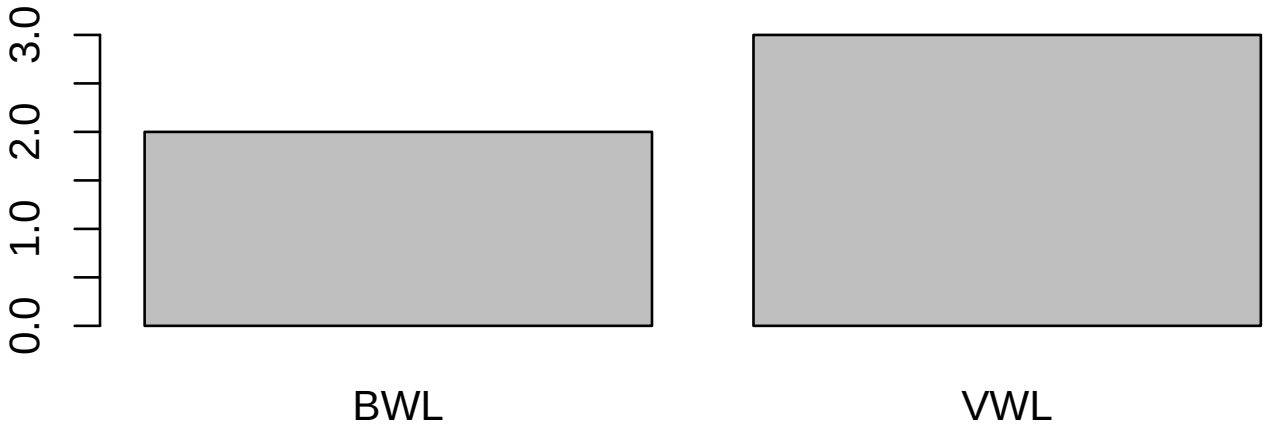


Übung: Visualisierung

Wie erzeugt man die angegebene Grafik?

```
d
```

```
##    name height shoesize study
## 1  Anne    163       38   VWL
## 2 Leyla    186       41   BWL
## 3 Elias    172       41   BWL
## 4 Marie    166       37   VWL
## 5   Ali    180       44   VWL
```



Gliederung

Zusammenfassung

- Zuweisung und Vergleich

```
x <- 4
x == 4
```

```
## [1] TRUE
```

- Mathematische Funktionen (nicht immer genau!)

```
sin(x)
abs(x)
round(sin(x))
```

- Strings und Vektoren

```
y <- c("hello", 11, "world")
y[c(1,3)]
```

```
## [1] "hello" "world"
```

Zusammenfassung

- Komponentenweise Ausführung von Operationen

```
x <- c(1,2,3)
y <- x
x*y
```

```
## [1] 1 4 9
```

- Matrizen und Datensätze

```
S1 <- c("x", "y")
S2 <- c(1, 2)
A <- as.data.frame(cbind(S1,S2))
A[A$S1 == "x",]
```

```
##   S1 S2
## 1  x  1
```

Zusammenfassung

- Einfaches Liniendiagramm

```
a1 <- c(1:100)
a2 <- a1*a1
plot(a1,a2, type="l")
```