Etwas R am Abend

Standort Köln – 21.11.2016

Norman Markgraf

Was ist eigentlich R?

Programmiersprache S:

- Von Bell Labs für Statistik, Simulation, Grafik entwickelt (Becker and Chambers; 1984)
- kommerzielle Implementation: **S-PLUS**

Programmiersprache R:

- Implementation unter GPL (GNU General Public License), offener Quellcode
- **Vorteile:**
  - interpretierter Programmcode, objektorientiert
  - leicht erweiterbar durch eigene Routinen, Pakete, DLLs
  - viele Grafikmöglichkeiten
  - standardisiertes, einfach handhabbares Datenformat (data.frame)
  - gut durchdachtes Format zur Anpassung von (Regressions-)Modellen

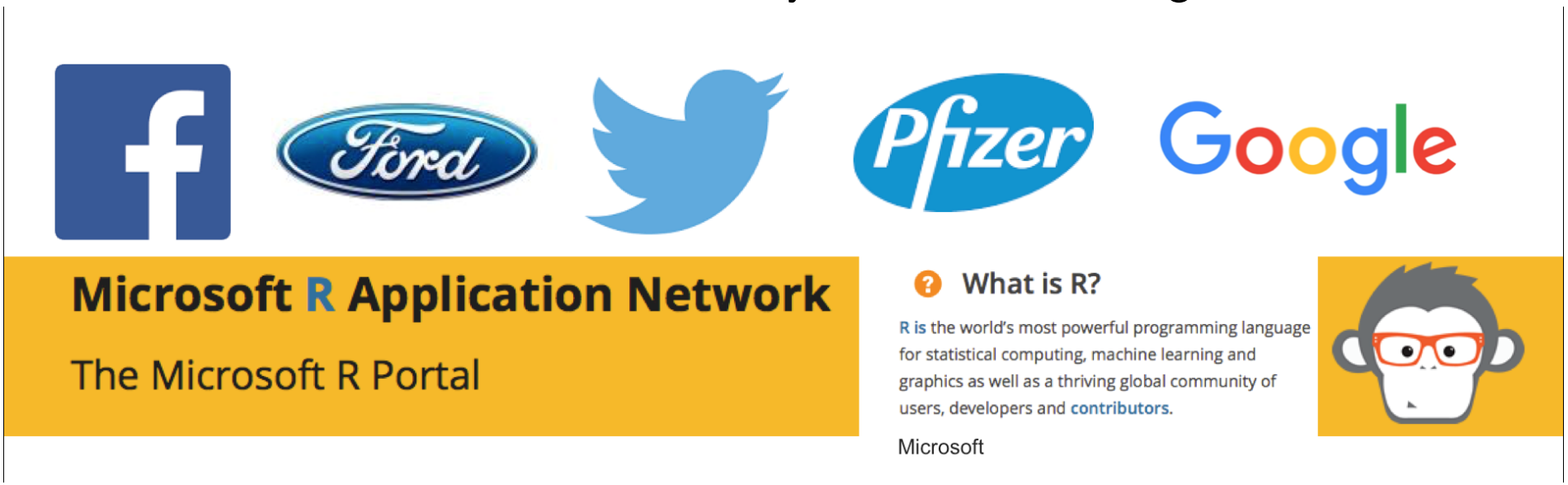
Was ist eigentlich R?

Programmiersprache R:

- **Vorteile (Forts.):**
  - aktive Entwicklergruppe, hilfreiche Mailingliste
  - modularer Aufbau mit mehr als 8000 Erweiterungspaketen
  - man kann ansprechende Diagramme und interaktive Apps entwickeln (z.B. plotly, shiny).
  - führende Plattform für statistische Analysen
- **Nachteile:**
  - bisher keine echte "Standard"-GUI (aber es gibt ja RStudio)
  - verfügbare Routinen/Pakete manchmal unübersichtlich

Wer nutzt R im echten Leben?

Unternehmen, die „ernsthaft“ Daten analysieren, setzen häufig auf R.



Quelle: `http://www.revolutionanalytics.com/companies-using-r`
Falls Sie gerne **Werbevideos** ansehen, hier ein Link
`https://www.youtube.com/watch?v=TR2bHSJ_eck`

Einfach nur R oder darf es etwas mehr sein?

R ist ein eine *komandozeilenorientierte*-Sprache!

```
1+1
```

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

```
1+2*3^4
```

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

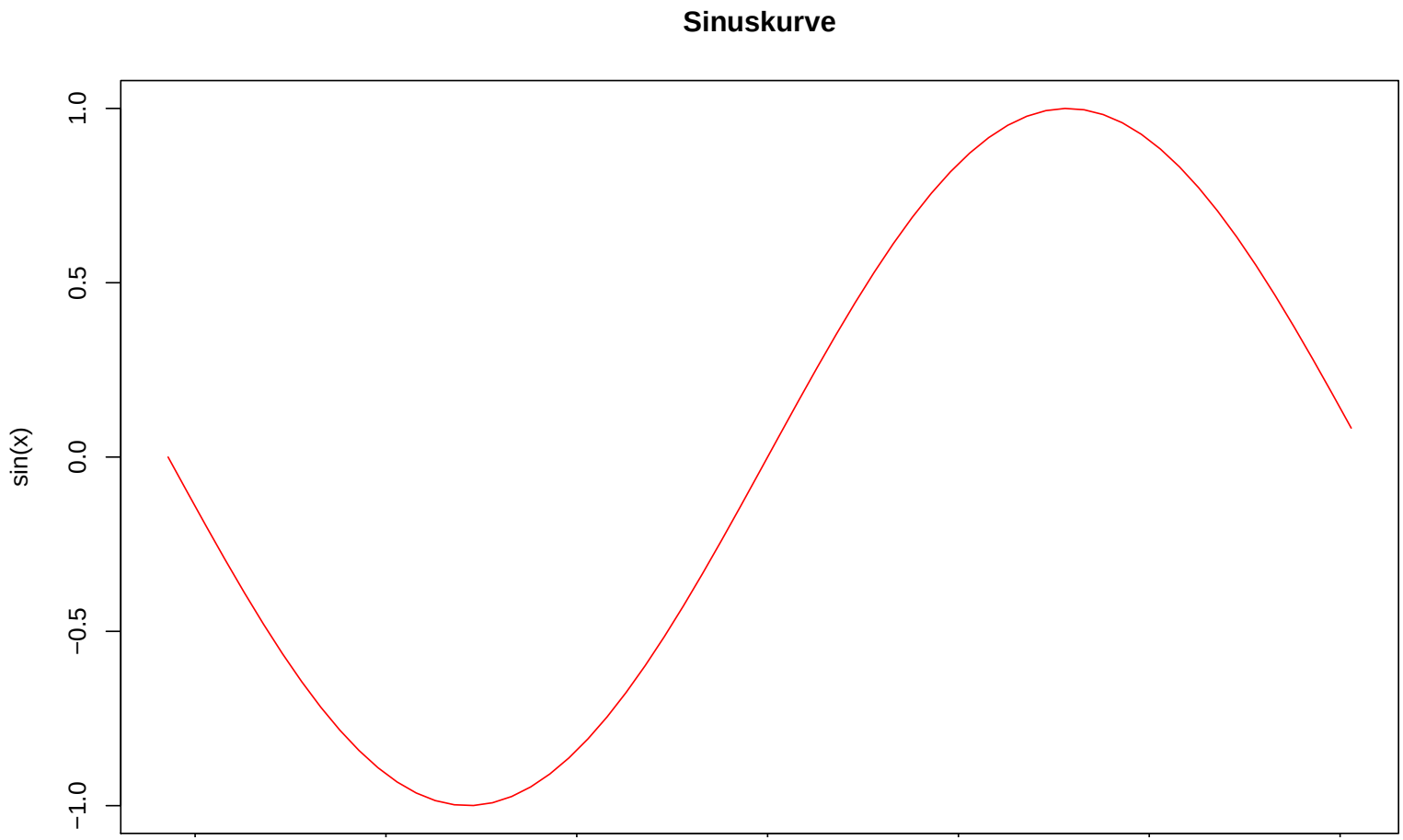
```
x <- 1; y <- 2
x+y
```

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

Einfach nur R oder darf es etwas mehr sein?

Die mit unter recht schnell schöne Ergebnisse produzieren kann:

```
x <- seq(-pi,pi,by=0.1)
plot(x, sin(x),  type="l", col="red",
     main="Sinuskurve")
```



Einfach nur R oder darf es etwas mehr sein?

Natürlich können Sie **R** als Programmiersprache direkt von der Konsole aus füttern. Besser ist es aber seine Skripte vorab mit Hilfe eines Texteditors zu schreiben und R dieses ausführen zu lassen.
Noch besser ist die Nutzung von Integrierten Entwicklungsumgebenung (*IDE*), wie z.B.

- RStudio
- Rcmdr
- StatET for R
  eine auf Eclipse basierende IDE für R
- ESS ein add-on package für GNU Emacs und XEmacs

Was bekommt man wo und wie?

Empfehlung:

- **R** (3.3.2) , RStudio (0.99.903 oder neuer als 1.0.44) (vergessen Sie lieber den **R-Cmdr**)
  - **R** finden Sie hier: `https://cran.rstudio.com` oder `https://www.r-project.org`
  - Aktuell ist die Version 3.3.2
  - **Achtung MAC-Nuzter!!!**: Sie benötigen zusätlich erst noch XQuartz.
    - XQuartz finden Sie hier: `https://www.xquartz.org`
- **RStudio** (Desktop-Version) (1.0.44, leider!)
  - Die aktuelle Version finden Sie hier: `https://www.rstudio.com/products/rstudio/download/`
  - **Besser** ist aber die alte Version 0.99.903 von hier: `https://support.rstudio.com/hc/en-us/articles/206569407-Older-Versions-of-RStudio`
  - Oder, für mutige, die tagesaktuelle Entwicklerversion von hier: `https://dailies.rstudio.com`

Die Installation

- Die wichtigsten Schritte bei der Installation:
  - Abwarten und bestätigen ;-)

Die wichtigsten Pakete und wie man sie installiert

Im Allgemeinen instaliert man ein Paket durch den Befehl:

```
install.packages("<blubber>", dependencies=TRUE)
```

Für einen guten Start sollte man folgende Pakete installieren:

- **tidyverse**
  tidyverse ist eine Sammlung von Paketen, die einem den Umgang mit **R** und *Grafik* erleichtern.
  Das sind u.a. die Pakete:
  - ggplot2 # DAS Grafikpaket von *R*
  - dplyr # Das Paket zur Daten manipulation
  - readr # Das Paket zum Einlesen von Daten

Die wichtigsten Pakete und wie man sie installiert

- **mosaic**
  Mehr Informationen zu **mosaic** finden Sie hier:
  - Project MOSAIC
  - Less Volume, More Creativity – Getting Started with the mosaic Package
- *Daten* zum Experimentieren und Spielen

Die Daten als Rundlage zu den Beispielen hier, finden Sie unter:
`https://github.com/sebastiansauer/Daten_Unterricht`
Entzippen Sie den herunter geladenen Ordner; dort finden Sie die hier verwendeten Daten.

Die wichtigsten Pakete und wie man diese installiert

Die ersten Befehle sollten wie folgt lauten:

```
# Laden von tidyverse Paketen:
install.packages("tidyverse", dependencies=TRUE)
# Laden des mosaic Pakets:
install.packages("mosaic", dependencies=TRUE)
```

Ach ja, mit '#' leitet man einen Kommentar ein. Sie müssen das also nicht mit den Kommentaren eintippen, es reicht:

```
install.packages("tidyverse", dependencies=TRUE)
install.packages("mosaic", dependencies=TRUE)
```

Bitte bestätigen Sie alle Anfragen und haben Sie etwas Gedult. Es wird eine Menge nachgeladen. Aber nur einmal. Also keine Sorge!

Die ersten Schritte

R als Taschenrechner:

Bemerkung	Umsetzung in **R**
Grundrechenarten	$+$ - * $/$
Potenzieren	^
logische Operatoren	$==$!= $<$ $>$ $<=$ $>=$
Funktionen	$\cos$ $\sin$ $\tan$
$\exp$ $\sqrt{}$	
Konstante	pi
Kommentare	$\#$
Dezimalzeichen	.

Die ersten Schritte

```
# R beherscht Punkt vor Strichrechnung
2 * 3 + 2 - 25/5 + 2^3
```

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

```
# Trigometrische Funktionen
cos(pi/2)^2 + sin(pi/2)^2
```

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

```
# Logarithmen und Exponetialfunktion
log(exp(3))
```

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

```
# Unendlich
1/0
```

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

Die ersten Schritte

```
# Not a Number (keine Zahl)
0/0
```

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

```
# Not Available; ein fehleder Wert
NA
```

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

```
# Vektoren (combine)
c(1, 4:8)
```

```
## [1] 1 4 5 6 7 8
```

```
# Vektor/Liste ohne Inhalt
c()
```

```
## NULL
```

Variablen

- Variablen in **R** können Skalare, Vektoren, Matrizen oder Objekte beliebiger anderer Klassen sein.
- Man **erzeugt** eine Variable in dem man ihr mit Hilfe von "<-" oder "=" etwas **zuweist**.
- **Variablennamen** können Kombinationen aus Buchstaben, Ziffern, Punkt und Unterstrich sein. Aber **keine Ziffern vorne**!
- **R** ist **case-sensetiv**, es unterscheidet zwischen Groß- und Kleinschreibung!

```
a <- c("FOM", "und", "R", "sind", "SUPER")
A <- 42
a
```

```
## [1] "FOM"   "und"   "R"     "sind"  "SUPER"
```

```
A
```

```
## [1] 42
```

Datentypen

In **R** gibt es die Datentypen

- **numeric** - ganzzahlige (*integer*) oder reelle (*double*) Zahlen
- **character** - Zeichenketten
- **logic** - die logischen Operatoren **TRUE** und **FALSE**
- **list** - Liste von Objekten jeder Art (die wiederum Listen beinhalten können!)

Befehle zum überprüfen der Datentypen:

```
mode(a)
```

```
## [1] "character"
```

```
str(a)
```

```
##  chr [1:5] "FOM" "und" "R" "sind" "SUPER"
```

```
typeof(a)
```

```
## [1] "character"
```

Vektoren

Ein Vektor wird mit dem Befehl **c()** (für *combine*) erzeugt:

```
a <- 5
vektorMitBeliebigemNamen <- c(log(1), a, sqrt(16), 3^2)
vektorMitBeliebigemNamen
```

```
## [1] 0 5 4 9
```

R kann (Rechen-)Operationen auf ganzen Vektoren (elementweise) durchführen:

```
vektorMitBeliebigemNamen * 2
```

```
## [1]  0 10  8 18
```

```
vektorMitBeliebigemNamen + 1
```

```
## [1]  1  6  5 10
```

Sequenzen

Zahlensequenzen werden mit dem Befehl **seq()** erzeugt. Dem Befehl können verschiedene Argumente Übergeben werden:

```
seq(from = 2, to = 9)
```

```
## [1] 2 3 4 5 6 7 8 9
```

```
seq(from = 2, to = 8, by=3)
```

```
## [1] 2 5 8
```

```
seq(from = 2, by = 0.5, length.out = 10)
```

```
##  [1] 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5
```

```
vektor <- 1:4 # 'n:m' entspricht seq(from=n, to=m, by=1)
```

Sequenzen

Werte können mit **rep()** wiederholt werden:

```
rep("X", times = 5) # wiederholt 'X' 5-mal
```

```
## [1] "X" "X" "X" "X" "X"
```

```
zahlen1 <- c(2, 4)
zahlen1
```

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

```
zahlen2 <- rep(zahlen1, times = 2)
zahlen2
```

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

```
rep(zahlen1, each = 2)
```

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

Logische Abfragen

```
people <- c("Klaus", "Max", "Jens", "Dieter")
people
```

```
## [1] "Klaus"  "Max"    "Jens"   "Dieter"
```

```
people == "Max"
```

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

```
vektorMitBeliebigemNamen != 0
```

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

```
logischerVektor <- vektorMitBeliebigemNamen <= 3
logischerVektor
```

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

Eigenschaften von Vektoren

names(a) gibt die Namen der Einträge des Vaktors a zurück:

```
weight <- c(67, 80, 72, 90)
names(weight)
```

```
## NULL
```

```
names(weight) <- people
weight
```

```
##  Klaus    Max   Jens Dieter
##     67     80     72     90
```

Rechnen mit Vektoren

- **Wichtige** Befehle für Vektoren sind $mean$, sd, var, min, max, $length$, sum, $median$, IQR, $summary$
- **Zugriff** auf das i-te Element eines Vektors a mit ai.

```
aVec <- c(1, 2, 4, 9)
mean(aVec)
```

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

```
sd(aVec)
```

```
## [1] 3.559026
```

```
var(aVec)
```

```
## [1] 12.66667
```

```
min(aVec)
```

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

Rechnen mit Vektoren

```
length(aVec)
```

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

```
sum(aVec)
```

```
## [1] 16
```

```
median(aVec)
```

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

```
IQR(aVec)
```

```
## [1] 3.5
```

```
summary(aVec)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.75    3.00    4.00    5.25    9.00
```

Varianzen

R berechnent die Varianz von Daten mit Hilfe der Formel

$$\frac{1}{n-1} \cdot \sum x - \bar{x}^2,$$

wie man leicht nachrechnen kann:

```
var(aVec)
```

```
## [1] 12.66667
```

```
# Ist das selbe wie
1/(length(aVec)-1) * sum( (aVec-mean(aVec))^2 )
```

```
## [1] 12.66667
```

```
# Dagegen ist
1/length(aVec) * sum( (aVec-mean(aVec))^2 )
```

```
## [1] 9.5
```

Standardabweichung

Die Standardabweichung ist die Quadratwurzel der Varianz:

```
sd(aVec)
```

```
## [1] 3.559026
```

```
sqrt(var(aVec))
```

```
## [1] 3.559026
```

Varianz

Will man die Varianz und die Standardabweichung mit Hilfe der Formel

$$\frac{1}{n} \cdot \sum x - \bar{x}^2,$$

berechnen, so muss man in **R** etwas tun:

```
factor <-(length(aVec)-1)/(length(aVec))
# Wert
var(aVec)
```

```
## [1] 12.66667
```

```
# Korrigierter Wert
factor*var(aVec)
```

```
## [1] 9.5
```

```
# Zur Probe:
1/length(aVec) * sum( (aVec-mean(aVec))^2 )
```

```
## [1] 9.5
```

Standardabweichung

```
factorSD <-sqrt((length(aVec)-1)/(length(aVec)))
# Wert von R:
sd(aVec)
```

```
## [1] 3.559026
```

```
# Korrigierter Wert
factor*sd(aVec)
```

```
## [1] 2.66927
```

```
# Zur Probe
sqrt(1/length(aVec) * sum( (aVec-mean(aVec))^2 ))
```

```
## [1] 3.082207
```

Rechnen mit Vektoren

```
aVec2 <- rep(2, 4)
aVec
```

```
## [1] 1 2 4 9
```

```
aVec2
```

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

```
aVec %*% aVec2
```

```
##      [,1]
## [1,]   32
```

```
aVec * aVec2
```

```
## [1]  2  4  8 18
```

```
aVec3 <- aVec
aVec3[3]
```

Literaturtip

Wenn Sie sich mehr für **R** interssieren. Ein erster Anlaufpunkt wäre z.B. das Skript von Christian Heuman, Institut für Statistik, Ludwig-Maximilians-Universität München:
`http://www.statistik.lmu.de/~chris/rkurs/rkurs.html`

Wir laden ein paar Daten

- Via **RStudio**: Gehen Sie auf das recht obere Fenster und klicken Sie **Import Dataset**, danach **From Web URL** und geben Sie als URL bitte *`http://www.statistik.lmu.de/service/datenarchiv/miete/miete03.asc`* ein. Beachten
- Via **R** direkt: Man kann auch direkt aus **R** mittels ein paar Zeilen die Daten laden! Laden wir ein paar Demodaten aus dem Netz:

```
miete03 <- read.table(
   file = paste0("http://www.statistik.lmu.de/service/",
                 "datenarchiv/miete/miete03.asc"),
   header = TRUE)
```

Die ersten Zeilen der Tabelle ansehen

Mit dem Befehl **head()** schaut man sich die ersten Zeilen (im Bsp. die ersten 4 Zeilen) eines *Dataframes* an:

```
head(miete03, 4)
```

```
##        nm  nmqm wfl rooms   bj bez wohngut wohnbest ww0 zh0 badkac
## 1 741.39 10.90  68     2 1918   2       1         0   0   0
## 2 715.82 11.01  65     2 1995   2       1         0   0   0
## 3 528.25  8.38  63     3 1918   2       1         0   0   0
## 4 553.99  8.52  65     3 1983  16       0         0   0   0
##   badextra kueche
## 1        0      0
## 2        0      0
## 3        0      0
## 4        1      0
```

Die letzten Zeilen der Tabelle ansehen

Mit dem Befehl **tail()** schaut man sich die ersten Zeilen (im Bsp. die letzen 3 Zeilen) eines *Dataframes* an:

```
tail(miete03, 3)
```

```
##           nm nmqm wfl rooms   bj bez wohngut wohnbest ww0 zh0 badk
## 2051 567.54 8.11  70     3 1973  16       0         0   0   0
## 2052 323.42 9.24  35     1 1970  21       0         0   0   0
## 2053 506.19 7.79  65     3 1966   7       0         0   0   0
##      badextra kueche
## 2051        0      0
## 2052        0      0
## 2053        0      0
```

Häufigkeitstabelle und Balkendiagramme

Mit dem Befehl **table** können wir eine *Häufigkeitstabelle* erstellen:

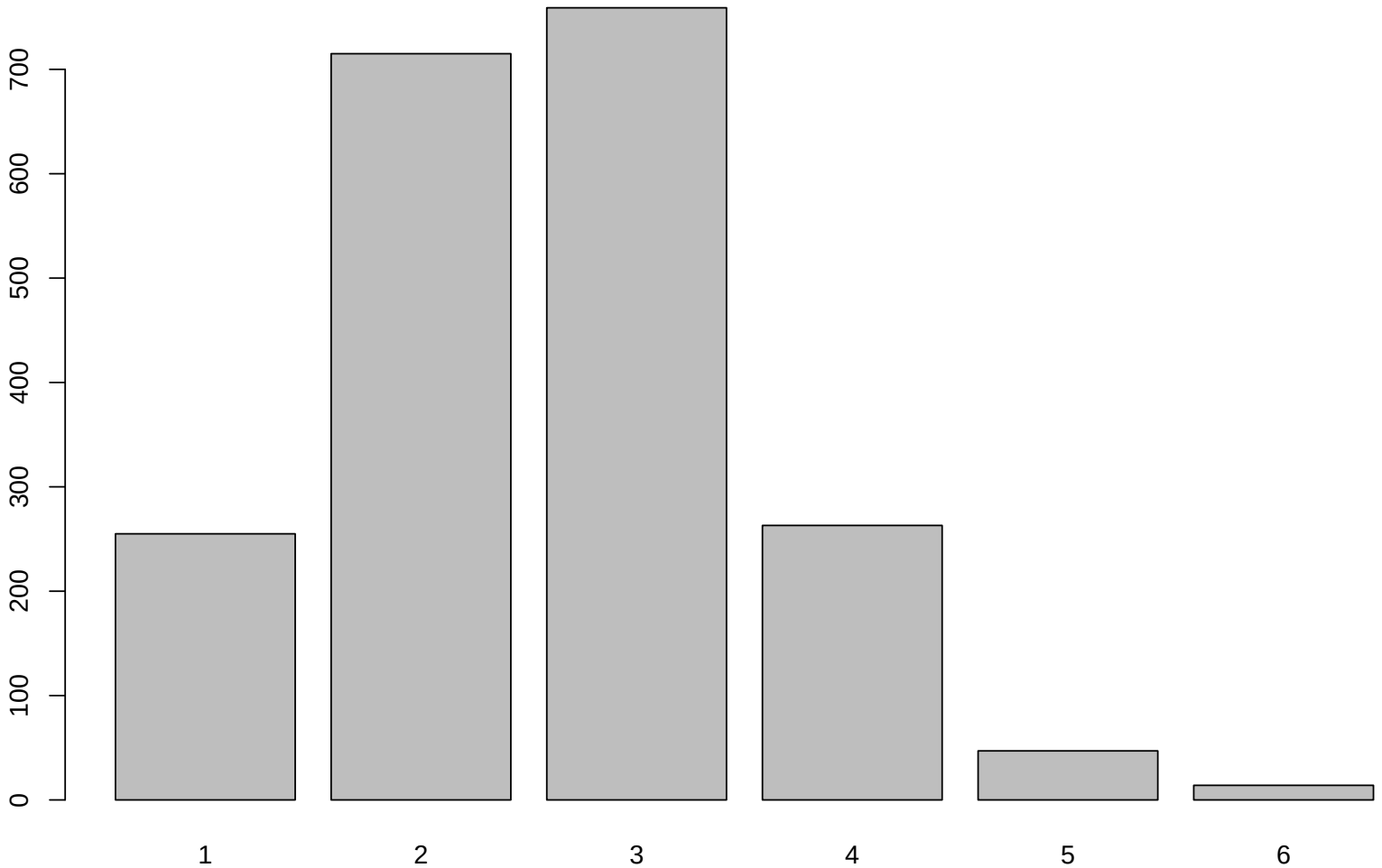
```
table(miete03$rooms)
```

```
##
##   1   2   3   4   5   6
## 255 715 759 263  47  14
```

Häufigkeitstabelle und Balkendiagramme

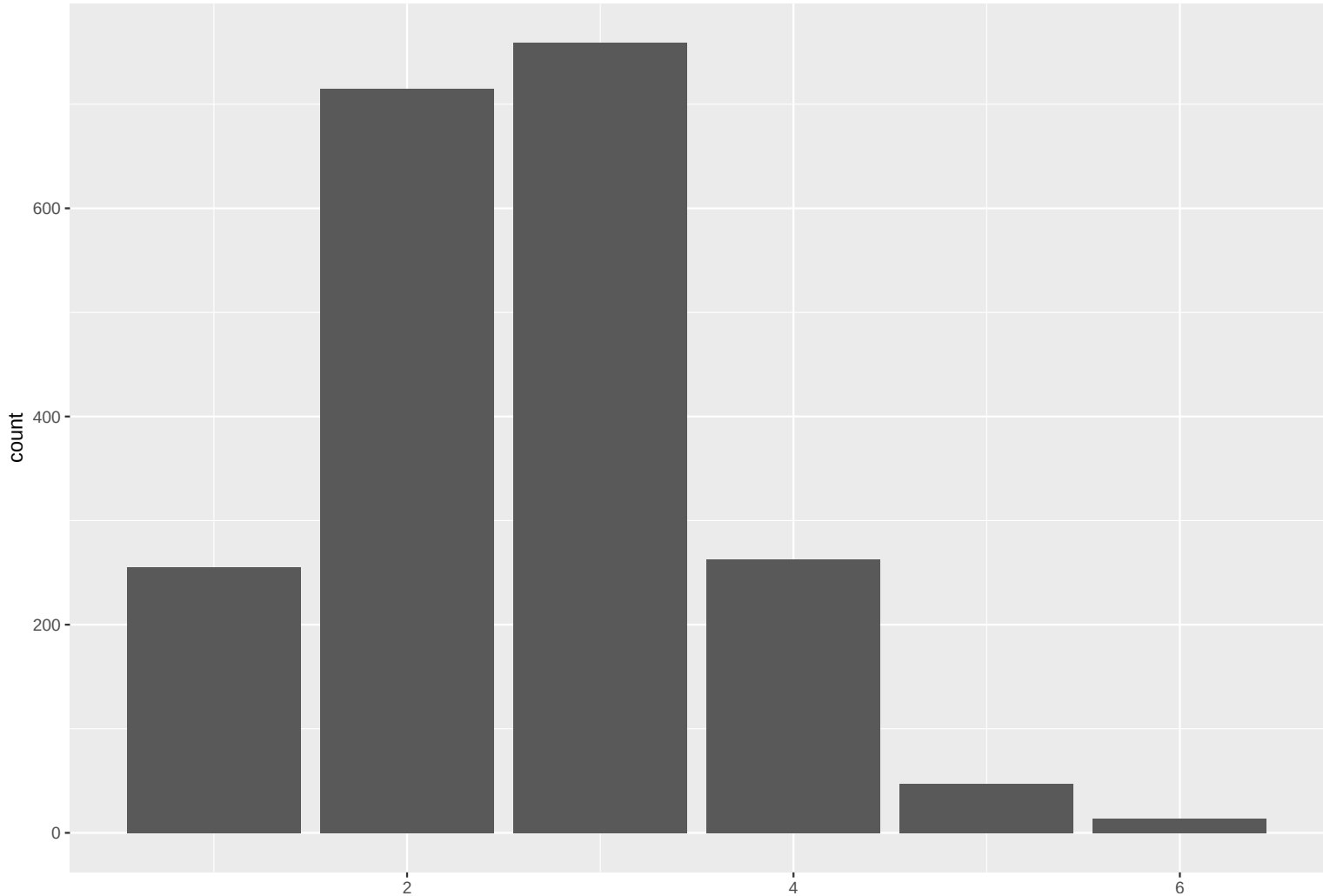
Und mit dem Befehl **barplot()** erstellen wir ein Balkendiagramm daraus:

```
barplot( table(miete03$rooms) )
```



Schönere Diagramme mit **ggplot2**

```
library(ggplot2) # Bibliothek laden!
ggplot(miete03, aes( x = rooms)) + geom_bar()
```



Mehr Informationen zu **ggplot2**

Wie man mit **ggplot2** noch mehr und noch schönere Grafiken erstellt, können Sie finden bei:

- `http://ggplot2.org`
- `http://docs.ggplot2.org/current/index.html`
- `http://www.cookbook-r.com/Graphs/`
- `https://www.datacamp.com/courses/data-visualization-with-ggplot2-1`
- `http://r4ds.had.co.nz`

Mehr Informationen zu **R**

Hier finden Sie Videos, die einige Schritte der Datenaufbereitung und deskriptiver/ explorativer Datenanalyse erläutern (zumeist mit R-Commander):

- boxplots erstellen `https://www.youtube.com/watch?v=9XBjOmA7sNs`
- Textdatei öffnen `https://www.youtube.com/watch?v=QnM9HBe23Y8`
- Öffnen der Datei Polizeistudie `https://www.youtube.com/watch?v=SDOoKuj5_7o`
- SPSS Datei importieren `https://www.youtube.com/watch?v=HS8H_n7Vrm0`
- Deskriptive Statistik erstellen `https://www.youtube.com/watch?v=qrMpgk-7Wus`
- Variablen in Faktoren umwandeln und Balkendiagramm `https://www.youtube.com/watch?v=PRR-3kblt8k`
- Streudiagramm `https://www.youtube.com/watch?v=brE72_0st00`
- Korrelationsmatrix `https://www.youtube.com/watch?v=pl92q_S-r8E`
- Datenmatrix erstellen `https://youtu.be/-EaeBL9J4IE`

Die Videos wurden von Frau Prof. Ferreira erstellt.