R - Eine kurze Einführung

- Was ist R?
- Installation
- R als Tischrechner
- Laden von Datensätzen und Einlesen von Daten
- Hilfe und Dokumentation
- Einfaches Datenmanagement
- Univariate explorative Analyse
- Bivariate explorative Analyse
- Lineare Regression
- Zeitreihen

1.1 Was ist R?

Vorgeschichte: **S** ist eine Sprache für Datenanalyse und Graphik, entwickelt von John Chambers und Kollegen in den Bell Laboratories, New Jersey, seit den späten 70er Jahren.

Ziele:

- interaktive Datenanalyse
- Graphik für explorative Datenanalyse
- vollwertige Programmiersprache

→ "Programmieren mit Daten"

R ist ein Dialekt der Sprache **S**, anfangs entwickelt von Ross Ihaka und Robert Gentleman an der University of Auckland, Neuseeland, seit Mitte der 90er Jahre von einem internationalen Entwickler-Kollektiv (R Development Core Team, "R Core").

R ist ein Open-Source-Softwareprojekt, frei erhältlich unter

`http://www.R-project.org`

1.1 Was ist R?

- Formeller Eigentümer des Copyrights ist die "R Foundation for Statistical Computing" mit Sitz in Wien.

  Damit ist R nicht in der "public domain", aber jeder Nutzer bekommt eine Lizenz (GPL), mit der R benutzt, kopiert, verteilt oder auch verändert werden darf.

  Details siehe `license()`.

- Open Source bedeutet insb., dass der gesamte *Quellcode* frei zugänglich ist – niemand muss der Dokumentation vertrauen, jede(r) kann (im Prinzip) selbst herausfinden, was die Software wirklich tut!

- Alle können diese Software verwenden, damit wird Forschung reproduzierbar.

1.2 Installation

Fertige R Distributionen sind für viele Betriebssysteme (Linux, Unix, Windows, MacOS, ...) erhältlich unter

`http://CRAN.R-project.org`

Installation der Windows-Version: Herunterladen der Datei `R-2.4.1-win32.exe`, dann ausführen.

Diese "Basisversion" von R enthält bereits einige Zusatzpakete (d.h. Sammlungen von Funktionen, Daten etc. für spezielle Fragestellungen), z.B. `KernSmooth` für nichtparametrische Regression und Dichteschätzung.

Daneben gibt es mehr als 900 (!) sog. "contributed packages", die weitere Funktionen zur Verfügung stellen, sowie Tochterprojekte wie Bioconductor ($\rightarrow$ Bioinformatik).

Zusatzpakete können – Internet-Verbindung und Schreibberechtigung auf das entsprechend Verzeichnis vorausgesetzt – online installiert werden über

```
R> install.packages("lmtest")
```

Dabei ist `lmtest` ein Paket mit diagnostischen Test für (lineare) Regressionsmodelle (Tests auf Heteroskedastie, Autokorrelation, etc.).

```
R> 1 + 1
```

```
[1] 2
```

```
R> a <- sqrt(pi)
R> b <- a^2
R> b
```

```
[1] 3.142
```

Alle übliche mathematischen Funktionen sind verfügbar.
Beispiel:

```
R> exp(sin(1)^2 + cos(1)^2)
```

```
[1] 2.718
```

1.4 Einlesen von Daten

- In R(-Paketen) bereits enthaltene Datensätze werden zugänglich gemacht über

  ```
  R> data("LifeCycleSavings")
  ```

- Liegen Daten in einer R-Binärdatei vor (Endungen `.rda` oder `.RData`), wird diese geladen über

  ```
  R> load("BIPSchweiz.rda")
  ```

- Liegen Daten in einem fremden Binärformat vor (z. B: SPSS-Dateien), lassen sich diese mit Hilfe der Funktionen aus dem Paket `foreign` importieren über

  ```
  R> library("foreign")
  R> read.spss("myfile.sav", to.data.frame = TRUE)
  ```

- ASCII-Dateien werden eingelesen über

  ```
  R> mydata <- read.table("myfile", header = FALSE)
  ```

  Das neu erzeugte Objekt `mydata` ist ein sogenannter "data frame".

1.5 Hilfe und Dokumentation

Hilfe zu Syntax etc. einer bekannten Funktion, z.B. `read.table()`, erhält man über

```
R> ?read.table
```

oder

```
R> help(read.table)
```

Weiss man nicht genau, wie das Gesuchte heisst, gibt es

```
R> help.search("suchbegriff")
```

Ausserdem sind ausführbare (!) Beispiele ein integraler Bestandteil der Dokumentation, so liefert

```
R> example(lm)
```

einige Beispiele zur Funktion `lm()` (für linear model), mit der man lineare Regressionsmodelle mit der KQ-Methode anpassen kann.

1.5 Hilfe und Dokumentation

Darüber hinaus existieren diverse Handbücher, u.a.

- An Introduction to **R**
- **R** Data Import/Export

Beide sind Bestandteil einer **R** Distribution (siehe Menü Hilfe) und auch über die **R**-Seite `www.R-project.org` erhältlich.
Weiter gibt es diverse R-Einführungen (in unterschiedlicher Qualität) unter CRAN -> Documentation -> Contributed. Nützlich ist u.a.

- The R Guide (version 2.2) by Jason Owen

Eine "R reference card" mit einigen wichtigen Befehlen gibt es unter

- R reference card by Jonathan Baron (Umfang 1 S.)
- R reference card by Tom Short (Umfang 4 S.)

1.6 Einfaches Datenmanagement

Als Beispiel der schon erwähnte Datensatz `LifeCycleSavings`, der Bestandteil von R ist.

Laden der Daten:

```
R> data("LifeCycleSavings")
R> class(LifeCycleSavings)
```

```
[1] "data.frame"
```

Wie sieht der Datensatz aus?

```
R> head(LifeCycleSavings, n = 7)
```

```
             sr pop15 pop75    dpi ddpi
Australia 11.43 29.35  2.87 2329.7 2.87
Austria   12.07 23.32  4.41 1508.0 3.93
Belgium   13.17 23.80  4.43 2108.5 3.82
Bolivia    5.75 41.89  1.67  189.1 0.22
Brazil    12.88 42.19  0.83  728.5 4.56
Canada     8.79 31.72  2.85 2982.9 2.43
Chile      0.60 39.74  1.34  662.9 2.67
```

Data frames sind matrixartige Objekte und haben eine Dimension:

```
R> dim(LifeCycleSavings)
```

```
[1] 50  5
```

Dieser `data.frame` hat damit 50 Zeilen (= Beobachtungen, hier Länder) zu jeweils 5 Variablen.
Die Variablennamen erhält man über

```
R> names(LifeCycleSavings)
```

```
[1] "sr"    "pop15" "pop75" "dpi"   "ddpi"
```

Selektion von Beobachtungen aus Datensätzen erfolgt mit der Funktion `subset()`.

Erzeugen wir einen neuen Datensatz, der nur die Länder mit einem Bevölkerungsanteil der unter 15-Jährigen von unter 30% enthält:

```
R> small <- subset(LifeCycleSavings, pop15 < 30)
```

Wie gross ist unser Teildatensatz?

```
R> dim(small)
```

```
[1] 20  5
```

Da `data.frames` matrixartige Objekte sind, können sie auch wie Matrizen angesprochen und behandelt werden. Sie sind indiziert mit `[i, j]` (Element aus `i`-ter Zeile, `j`-ter Spalte).

Beispiele:

```
R> small[3, 5]
R> small[1:10, ]
R> small[, 2:4]
R> small[1:10, c(1, 3)]
R> small[1:10, -c(2, 4)]
```

Da `data.frames` Spaltennamen haben, können auch diese Namen statt Nummern verwendet werden:

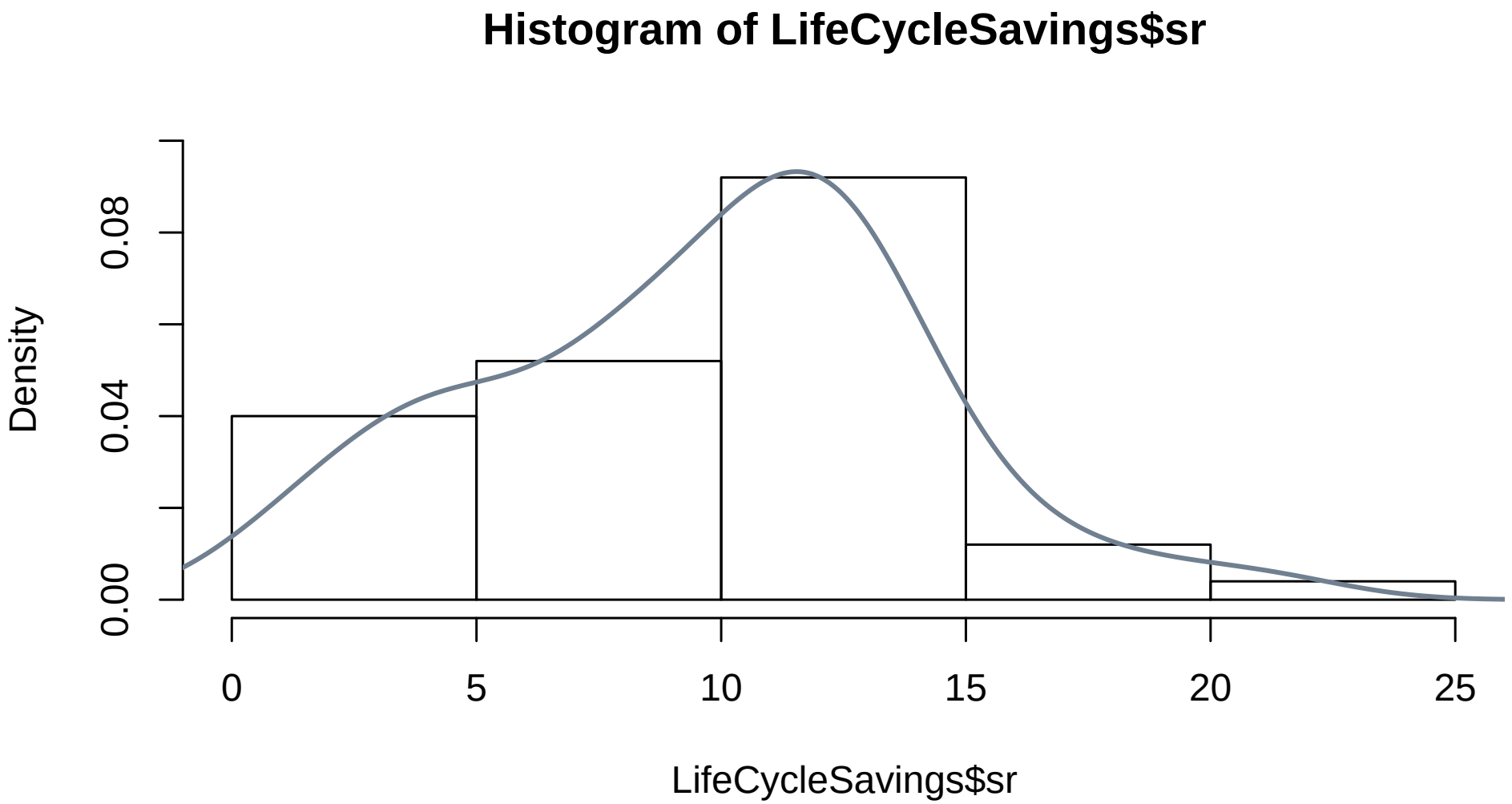
```
R> LifeCycleSavings[, c("pop75", "dpi")]
```

```
R> class(LifeCycleSavings$sr)

[1] "numeric"

R> mean(LifeCycleSavings$sr)

[1] 9.671

R> var(LifeCycleSavings$sr)

[1] 20.07

R> min(LifeCycleSavings$sr)

[1] 0.6

R> max(LifeCycleSavings$sr)

[1] 21.1
```

```
R> summary(LifeCycleSavings$sr)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.60    6.97   10.50    9.67   12.60   21.10
```

Einfache explorative Graphiken z.B. mit:

```
R> hist(LifeCycleSavings$sr, freq = FALSE, ylim = c(0, 0.1))
R> lines(density(LifeCycleSavings$sr), col = "slategrey",
+     lwd = 2)
```


Histogram of LifeCycleSavings$sr
Density
0.00
0.04
0.08
0
5
10
15
20
25
LifeCycleSavings$sr

1.8 Bivariate explorative Analyse

Zwei numerische Variablen:

```
R> cor(LifeCycleSavings$sr, LifeCycleSavings$pop15)
```

```
[1] -0.4555
```

Streudiagramm (beachte Formelschreibweise!):

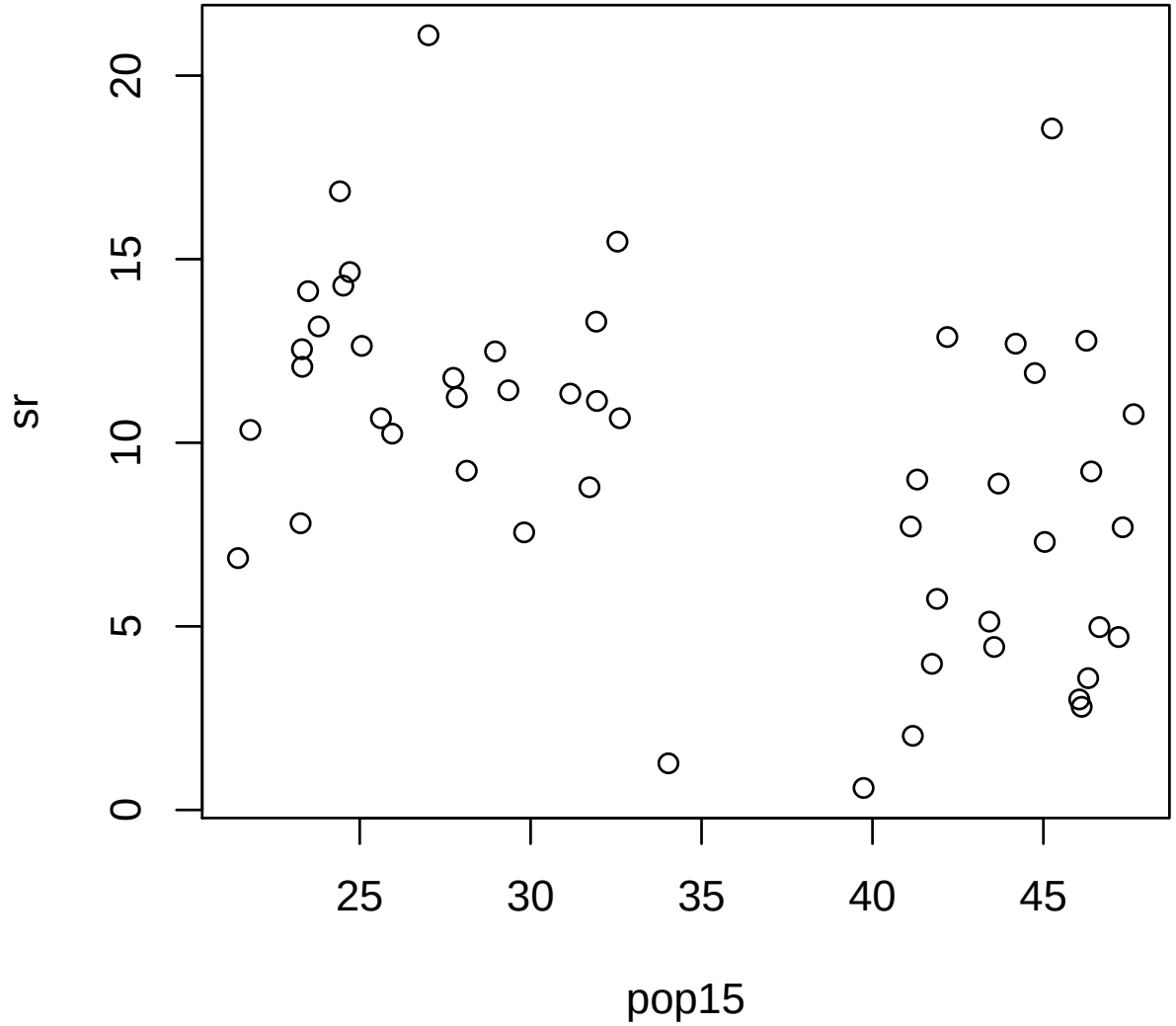
```
R> plot(sr ~ pop15, data = LifeCycleSavings)
```

Hier äquivalent, aber ohne Formel:

```
R> plot(LifeCycleSavings$pop15, LifeCycleSavings$sr)
```

R hat ein sehr umfangreiches Angebot an Graphiken. Empfehlung:

```
R> demo(graphics)
```


sr
0
5
10
15
20
25
30
35
40
45
pop15

```
R> fm1 <- lm(sr ~ pop15, data = LifeCycleSavings)
R> summary(fm1)

Call:
lm(formula = sr ~ pop15, data = LifeCycleSavings)

Residuals:
   Min     1Q Median     3Q    Max
-8.637 -2.374  0.349  2.022 11.155

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  17.4966     2.2797    7.67  6.8e-10
pop15        -0.2230     0.0629   -3.55  0.00089

Residual standard error: 4.03 on 48 degrees of freedom
Multiple R-Squared: 0.208,        Adjusted R-squared: 0.191
F-statistic: 12.6 on 1 and 48 DF,  p-value: 0.000887
```

```
R> library("tseries")
R> data("bev")
R> class(bev)

[1] "ts"

R> bev[1:10]

 [1] 17 19 20 15 13 14 14 14 14 11

R> tsp(bev)

[1] 1500 1869    1

R> lag(bev, -1)[1:10]

 [1] 17 19 20 15 13 14 14 14 14 11

R> tsp(lag(bev, -1))

[1] 1501 1870    1
```

Graphik:

```
R> plot(bev)
```

Weitere nützliche Funktionen sind:

`acf()`, `pacf()`, `arima.sim()`, `arima()`, `ARMAacf()`, `ar()`, `tsdiag()`, ...

Zum Ausprobieren:

```
R> set.seed(4051)
R> arima.sim(100, model = list(ar = c(0.5, -0.4)))
R> ARMAacf(c(0.5, -0.4), lag.max = 10)
R> plot(0:10, ARMAacf(ar = c(0.5, -0.4), lag.max = 10), type = "h",
+     ylim = c(-1, 1), ylab = "ACF")
R> abline(h = 0, col = "grey")
```