WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Einführung in die Datenanalyse mit R

Bettina Grün

`http://statmath.wu.ac.at/courses/gwa`

Einführung

- R ist ein Software-Paket für Statistisches Rechnen und Graphik.
- Es ist die Open Source Implementierung der Sprache S, die von John Chambers und Kollegen in den Bell Laboratories in den 1970ern entwickelt worden ist.
- Eine weitere, jedoch kommerzielle Implementierung von S ist S-PLUS.

Einführung / 2

- R ist ursprünglich von Ross Ihaka und Robert Gentleman an der Universität von Auckland Mitte 1990er entwickelt worden.
- Derzeit Wartung und Weiterentwicklung durch eine weltweite Gruppe von Freiwilligen aus Forschung und Wirtschaft.
- Primäre Verbreitung über Webseiten (`www.r-project.org`) und Archive (`cran.r-project.org`).
- R kann über Packages erweitert werden. Einige Packages sind in der R Distribution enthalten, weitere befinden sich auf CRAN (derzeit 2566 Pakete).

Vor- und Nachteile von R

Vorteile:

- Open source
  - keine „black box“
  - Skalierbarkeit, Adaptierbarkeit
- Erweiterbarkeit durch einfaches Programmieren
  - preisgekrönte Sprache
  - Automatisierung sich wiederholender Abläufe, Simulationen
  - Reproduzierbarkeit, Sweave
- Support durch Mailinglisten, usw.

Nachteile:

- keine komfortable graphische Benutzeroberfläche, sondern Command Line Interface
- keine interaktive Grafik

Wie komme ich zu R

- Die allgemeine Informationswebseite ist `http://www.r-project.org/`.
- CRAN: the Comprehensive R Archive Network
  - Die Hauptseite ist `http://cran.r-project.org/`.
  - Mirror Seiten sind in vielen Länder verfügbar, z.B. `http://cran.us.r-project.org/`.
- Auf den CRAN Seiten gibt es Binärdistributionen für Windows, Mac und mehrere Linux Distributionen.

Installieren von R

Auf Windows:

- Lade den Installer `R-2.11.1-win32.exe` von CRAN herunter und führe die Datei aus.
- Setzen des persönlichen Arbeitsverzeichnis:
  - rechte Maustaste bei Start-Icon: `Eigenschaften` → `Ausführen in`
  - Setzen im GUI über Menü: `File` → `Change dir ...`
  - Setzen über die Command Line mit `setwd()`

  Kontrolle mit `getwd()`
- Starten von R:
  - über den Start-Icon
  - über die Datei `Rgui.exe`

Informationsquellen über R

- Die Webseite `http://www.r-project.org/` und CRAN
- Die frequently asked questions (FAQ) Listen auf CRAN.
- Manuals: z.B.
  - An Introduction to R
  - R Data Import/Export

  Die Manuals sind Teil der R Installation bzw. befinden sich auf CRAN. Unter Windows können diese im Menü unter `Help` → `Manuals` gefunden werden.

Pakete

- Standardisierte Form der Erweiterung von R
  - “Verpacken” von Funktionen, Sourcen, Datensätze und Dokumentation
  - Validierungsmöglichkeiten
  - einfache Verteilung an Dritte, z.B. über CRAN
- Installieren mit `install.packages(`*Paketname*`)`
- Laden mit `library(`*Paketname*`)`

Sweave

- dynamische Dokumente durch Verbindung von Text und Code: LaTeX mit R
- Beginn von Code Chunk mit `<<>>=`, von Dokumentation Chunk mit `@`
- Skalare im Text mit `\Sexpr{}`
- `Sweave()` erzeugt `tex`-Datei mit eingebundenem R-Output
- `Stangle()` extrahiert den R Code
- Manual unter `http://www.statistik.lmu.de/~leisch/Sweave`
- Beispiele als Vignetten in Source Paketen:
  - `vignette(`*Vignettenname*`)`
  - `edit(`*Vignettenobjekt*`)`

  Z.B. im Paket **grid** die Vignette `grid`.

R Basics

Wenn der Funktionsname und anschließend `[RET]` getippt wird, wird die Funktion angezeigt.

Um eine Funktion aufzurufen, muss der Funktionsname zusammen mit der Argumentenliste innerhalb von `()` eingetippt werden, auch wenn diese leer ist. D.h. `q()[RET]` anstatt `q[RET]` ist notwendig, um R zu verlassen.

R Basics / 2

Hilfe:

```
help(topic)
?topic
help.search("topic")
help.start()
```

Operatoren:

```
+, -, *, /, ^
&, &&, |, ||
```

see `help("+")`

Schleifen:

```
for, while, if
```

Namen:

case sensitive,
Buchstabe zu Beginn

Zuweisung:

```
x = 5
x <- 5
5 -> x
```

Vergleiche:

```
==, !=, >, >=, <, <=
```

see `help("==")`

Kommentare:

alles nach #

R Basics / 3

Elementare Datentypen:

- **Logical**: `TRUE, FALSE, T, F`
- **Integer**: `1, 100, 326, ...`
- **Double**: `1.0, 3.1415297, 2.718282, NaN, Inf, -Inf`
- **Complex**: `1+0i, 1i, 3+5i`
- **Character**: `"Hello"`, `"How are you?"`
- **Missing Values**: `NA`

R Basics / 4

Zusammengesetzte Datentypen:

- `vector`: von Elementen desselben elementaren Typs
- `array`: Vektor mit Dimensionsattribut (beliebige Anzahl an Dimensionen erlaubt).
- `matrix`: 2-dimensionaler Array. Spezialfall, um die üblichen Matrixoperationen wie z.B. Matrixmultiplikation durchführen zu können.
- `factor`: Spezieller Vektor für das Kodieren von Klassen.
- `list`: Liste von Elementen von verschiedenen Datentypen (sowohl elementar als auch zusammengesetzt)
- `data.frame`: "Mischung" aus Matrix und Liste

Zugriff über `[...]`, Dimensionen durch `,` getrennt.

Funktionen

```
> moment <- function(x, n=2) {
+   sum(x^n)/length(x)
+ }
```

- Es wird nur ein Wert zurückgegeben (letztes Statement oder Argument von `return`). Wenn mehrere Werte zurückgegeben werden sollen, dann können diese in einer Liste zusammengefasst werden.
- Argumente mit Namen: Bei Aufruf einer Funktion können die Argumente in einer beliebigen Reihenfolge übergeben werden (z.B. `moment(n=3, x=x)`).
- Default Werte: Argumente mit Default Werten können beim Aufruf der Funktion weggelassen werden.

Funktionen / 2

```
> set.seed(20100322)
> y <- rnorm(100)
> y[1:4]
[1] -0.8607870 -0.2587355  1.3743010 -2.9897422
> length(y)
[1] 100
> moment(y)
[1] 0.9785426
> moment(y, 1)
[1] -0.2197408
> moment(n = 1, x = y)
[1] -0.2197408
> mean(y)
[1] -0.2197408
```

Wichtige R Funktionen

- **q()**: Beende die Session. Es wird gefragt, ob der Workspace (im File `.RData`) gespeichert werden soll.
- **help()**: Bekomme Hilfeseite für eine Funktion oder ein Objekt.
- **help.start()**: Verwende einen Webbrowser für das Lesen der Hilfe.
- **ls()**: Liste die Objekte im Workspace auf.
- **rm()**: Lösche Objekte im Workspace.
- **save.image()**: Speichern des Workspace-Inhalts in einem File.
- **save()**: Speichern eines Objekts in einem File.
- **load()**: Laden von mit R gespeicherten Daten.
- **example()**: Führe das Beispiel auf der Hilfeseite aus.
- **str()**: Anzeigen der Struktur eines Objekts.
- **summary()**: Fasse die Information über ein Objekt kurz zusammen.
- **plot()**: Erstelle eine Graphik für ein Objekt.
- **library()**: Laden von zusätzlichen (schon installierten) Packages.

Einlesen von Daten

In R gibt es die Möglichkeit, die Daten aus den verschiedensten Formaten einzulesen. Eine nähere Beschreibung befindet sich in *R Data Import/Export*.

Häufig benötigte Befehle sind:

- **read.table()**: Einlesen von Daten in Tabellen-Format aus einer Datei und Erzeugen eines Data Frames
  Wichtige Parameter sind z.B. `header`, `sep`.
  read.csv() und **read.csv2()** sind spezielle Funktionen für Dateien im CSV-Format, in dem man z.B. Excel-Dateien abspeichern kann. Bei **read.csv2()** sind die Einträge durch Strichpunkte getrennt und das Dezimalzeichen ist—wie im Deutschen üblich—ein Komma.
- **load()**: Laden von Daten (und auch Objekten), die in R mit `save()` gespeichert worden sind

Erstellen von Skripts

Um Reproduzierbarkeit der R Sessions zu erreichen, können die einzelnen Befehle in ein File geschrieben werden und daraus nach R geladen werden.

Verwende einen externen Editor zum Schreiben eines Skripts: Notepad, Tinn-R, Emacs. . .

Einlesen des Skripts in R:

```
> source("skript.R")
```

Wenn die Befehle und Ergebnisse am Bildschirm mitverfolgt werden sollen:

```
> source("skript.R", echo = TRUE)
```

Wenn nur die Ergebnisse am Bildschirm mitverfolgt werden sollen:

```
> source("skript.R", print.eval = TRUE)
```

Erstellen von Skripts / 2

Mithilfe von `sink` kann die Ausgabe vom Bildschirm in eine Datei umgeleitet werden:

```
> sink("sink.txt")
> source("skript.R", echo = TRUE)
> sink()
```

Beispiel

```
> birthwt <- read.csv2("birthwt.csv")
> summary(birthwt)
```

```
      low              age             lwt
 Min.   :0.0000   Min.   :14.00   Min.   : 80.0
 1st Qu.:0.0000   1st Qu.:19.00   1st Qu.:110.0
 Median :0.0000   Median :23.00   Median :121.0
 Mean   :0.3122   Mean   :23.24   Mean   :129.8
 3rd Qu.:1.0000   3rd Qu.:26.00   3rd Qu.:140.0
 Max.   :1.0000   Max.   :45.00   Max.   :250.0
      race           smoke             ptl
 Min.   :1.000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :1.000   Median :0.0000   Median :0.0000
 Mean   :1.847   Mean   :0.3915   Mean   :0.1958
 3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:0.0000
 Max.   :3.000   Max.   :1.0000   Max.   :3.0000
       ht                ui              ftv
 Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
```

Beispiel / 2

```
 Median :0.00000   Median :0.0000   Median :0.0000
 Mean   :0.06349   Mean   :0.1481   Mean   :0.7937
 3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:1.0000
 Max.   :1.00000   Max.   :1.0000   Max.   :6.0000
      bwt
 Min.   : 709
 1st Qu.:2414
 Median :2977
 Mean   :2945
 3rd Qu.:3487
 Max.   :4990
```

Beispiel / 3

```
> attach(birthwt)
> BWT <- data.frame(
+   race = factor(race, 1:3, c("white", "black", "other")),
+   smoke = smoke > 0,
+   age, lwt, bwt)
> detach(birthwt)
```

Beispiel / 4

```
> summary(BWT)
```

```
   race      smoke              age
 white:96   Mode :logical   Min.   :14.00
 black:26   FALSE:115       1st Qu.:19.00
 other:67   TRUE :74        Median :23.00
            NA's :0         Mean   :23.24
                            3rd Qu.:26.00
                            Max.   :45.00
     lwt             bwt
 Min.   : 80.0   Min.   : 709
 1st Qu.:110.0   1st Qu.:2414
 Median :121.0   Median :2977
 Mean   :129.8   Mean   :2945
 3rd Qu.:140.0   3rd Qu.:3487
 Max.   :250.0   Max.   :4990
```

Uni- und bivariate Analysen

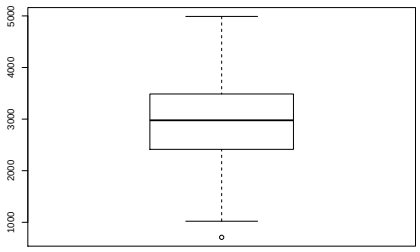
- 1 metrische Variable
- 1 kategoriale Variable
- 2 metrische Variablen
- 2 kategoriale Variablen
- 1 metrische und 1 kategoriale Variable

1 metrische Variable

```
> attach(BWT)
> summary(bwt)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    709    2414    2977    2945    3487    4990
> t.test(bwt, mu = 3000)

One Sample t-test

data:  bwt
t = -1.0447, df = 188, p-value = 0.2975
alternative hypothesis: true mean is not equal to 3000
95 percent confidence interval:
 2839.952 3049.222
sample estimates:
mean of x
 2944.587
```

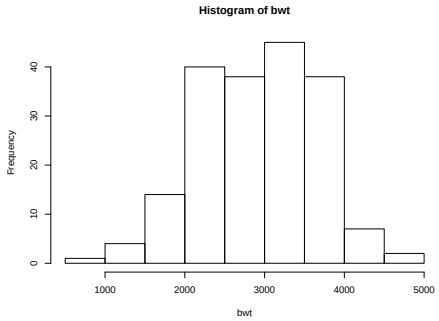
1 metrische Variable / 2

```
> boxplot(bwt)
```



1 metrische Variable / 3

```
> hist(bwt)
```

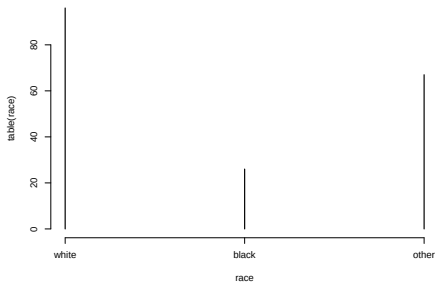


1 kategorielle Variable

```
> table(race)

race
white black other
   96    26    67
> prop.table(table(race))

race
    white     black     other
0.5079365 0.1375661 0.3544974
```

1 kategorielle Variable / 2

```
> plot(table(race))
```



2 metrische Variablen

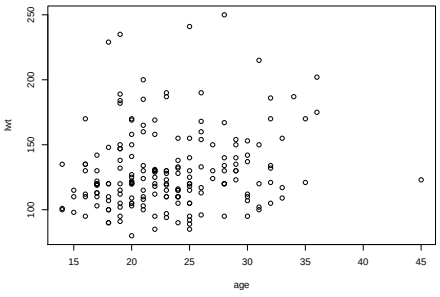
```
> sapply(BWT[, c("lwt", "age")], summary)
           lwt   age
Min.     80.0 14.00
1st Qu. 110.0 19.00
Median  121.0 23.00
Mean    129.8 23.24
3rd Qu. 140.0 26.00
Max.    250.0 45.00
> cor(lwt, age)

[1] 0.1800732
```

2 metrische Variablen / 2

```
> cor.test(lwt, age)

Pearson's product-moment correlation

data:  lwt and age
t = 2.5034, df = 187, p-value = 0.01316
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.03832798 0.31471467
sample estimates:
      cor
0.1800732
```

2 metrische Variablen / 3

```
> plot(lwt ~ age, data = BWT)
```



2 kategorielle Variablen

```
> table(race, smoke)
```

```
       smoke
race    FALSE TRUE
  white    44   52
  black    16   10
  other    55   12
```

```
> prop.table(table(race, smoke), 2)
```

```
       smoke
race        FALSE      TRUE
  white 0.3826087 0.7027027
  black 0.1391304 0.1351351
  other 0.4782609 0.1621622
```

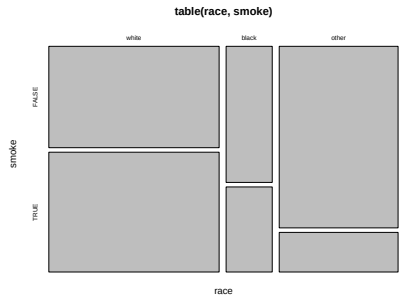
2 kategorielle Variablen / 2

```
> chisq.test(table(race, smoke))
```

```
Pearson's Chi-squared test

data:  table(race, smoke)
X-squared = 21.779, df = 2, p-value = 1.865e-05
```

2 kategorielle Variablen / 3

```
> plot(table(race, smoke))
```



1 metrische und 1 kategorielle Variable

```
> tapply(bwt, smoke, summary)
```

```
$`FALSE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1021    2509    3100    3056    3622    4990

$`TRUE`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    709    2370    2776    2772    3246    4238
```

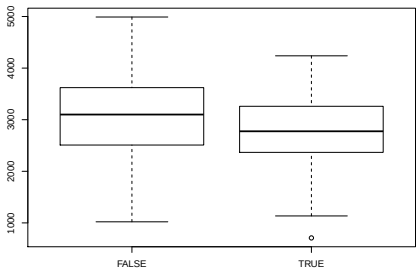
1 metrische und 1 kategorielle Variable / 2

```
> t.test(bwt ~ smoke, data = BWT)
```

```
Welch Two Sample t-test

data:  bwt by smoke
t = 2.7299, df = 170.1, p-value = 0.007003
alternative hypothesis: true difference in means is not equal to
95 percent confidence interval:
  78.57486 488.97860
sample estimates:
mean in group FALSE  mean in group TRUE
           3055.696            2771.919
```

1 metrische und 1 kategorielle Variable / 3

```
> boxplot(bwt ~ smoke, data = BWT)
```



Lineare Regression

Bei der linearen Regression gilt:

$$y_i \quad = \quad x_i^\top \beta + \varepsilon_i, \qquad (i = 1, \ldots, n).$$

- y_i — abhängige Variable.
- x_i — Regressorvektor der Länge k.
- β — Vektor der k unbekannten Regressionskoeffizienten.
- ε_i — Fehlerterm.

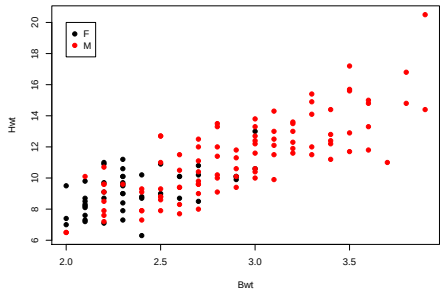
Lineare Regression / 2

```
> data("cats", package="MASS")
> summary(cats)
```

```
 Sex        Bwt             Hwt
 F:47   Min.   :2.000   Min.   : 6.30
 M:97   1st Qu.:2.300   1st Qu.: 8.95
        Median :2.700   Median :10.10
        Mean   :2.724   Mean   :10.63
        3rd Qu.:3.025   3rd Qu.:12.12
        Max.   :3.900   Max.   :20.50
```

Lineare Regression / 3

```
> plot(Hwt ~ Bwt, col = as.integer(cats$Sex),
+      data = cats, pch = 19)
> legend(2, 20, legend = levels(cats$Sex), col = 1:2, pch = 19)
```

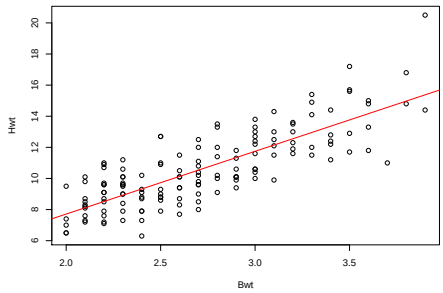


Lineare Regression / 4

```
> cats_lm1 <- lm(Hwt ~ Bwt, data = cats)
> summary(cats_lm1)

Call:
lm(formula = Hwt ~ Bwt, data = cats)

Residuals:
     Min       1Q   Median       3Q      Max
-3.56937 -0.96341 -0.09212  1.04255  5.12382

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.3567     0.6923  -0.515    0.607
Bwt           4.0341     0.2503  16.119   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.452 on 142 degrees of freedom
Multiple R-squared: 0.6466, Adjusted R-squared: 0.6441
F-statistic: 259.8 on 1 and 142 DF,  p-value: < 2.2e-16
```

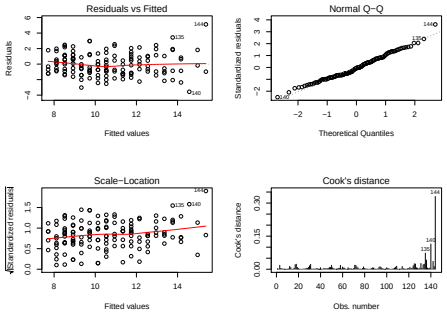
Lineare Regression / 5

```
> plot(Hwt ~ Bwt, data = cats)
> abline(cats_lm1, col = 2)
```



Lineare Regression / 6

```
> par(mfrow=c(2,2))
> plot(cats_lm1, which = 1:4)
```



Lineare Regression / 7

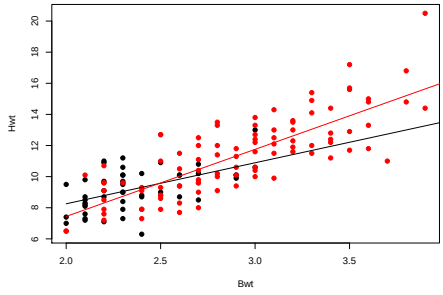
```
> cats_lm2 <- lm(Hwt ~ Bwt * Sex, data = cats)
> summary(cats_lm2)

Call:
lm(formula = Hwt ~ Bwt * Sex, data = cats)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7728 -1.0118 -0.1196  0.9272  4.8646

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.9813     1.8428   1.618 0.107960
Bwt           2.6364     0.7759   3.398 0.000885 ***
SexM         -4.1654     2.0618  -2.020 0.045258 *
Bwt:SexM      1.6763     0.8373   2.002 0.047225 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.442 on 140 degrees of freedom
```

Lineare Regression / 8

```
Multiple R-squared: 0.6566, Adjusted R-squared: 0.6493
F-statistic: 89.24 on 3 and 140 DF,  p-value: < 2.2e-16
```

```
> plot(Hwt ~ Bwt, col = as.integer(cats$Sex),
+      data = cats, pch = 19)
> BwtN <- seq(2, 4, by = 0.1)
> lines(BwtN, predict(cats_lm2,
+       data.frame(Bwt = BwtN, Sex = "F")))
> lines(BwtN, predict(cats_lm2,
+       data.frame(Bwt = BwtN, Sex = "M")), col = 2)
```

Lineare Regression / 9



Lineare Regression / 10

```
> anova(cats_lm1, cats_lm2)

Analysis of Variance Table

Model 1: Hwt ~ Bwt
Model 2: Hwt ~ Bwt * Sex
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    142 299.53
2    140 291.05  2    8.4865 2.0411 0.1337
```