# Programmieren in Anwendungen

Annette Bieniusa

Technische Universität Kaiserslautern

*bieniusa@cs.uni-kl.de*

05.06.2014

# Überblick

# Literatur


- Johannes Hain: Statistik mit R - Grundlagen der Datenanalyse, RRZN-Handbuch, 1. Auflage, Mai 2011.
- Paul Teetor: The R Cookbook. O'Reilly Media, März 2011.
- Drew Conway, John Myles White: Machine Learning for Hackers - Case Studies and Algorithms to Get You Started. O'Reilly Media, Februar 2012.
- Allen B. Downey: Think Stats - Probability and Statistics for Programmers. O'Reilly Media, Juli 2011.

# Software / Online-Ressourcen

- R Software `http://www.r-project.org`
- Entwicklungsumgebung: RStudio `http://www.rstudio.com`
- Interessante Bibliotheken, Tutorials, Trends: `http://www.r-bloggers.com`

# Grundbegriffe der Statistik

# Population und Stichprobe

**Grundgesamtheit/Population** Menge aller Objekte, die untersucht werden soll

**Stichprobe** Ausgewählte Untermenge der Population (meist zufällig ausgewählt); sollte repräsentativ sein

- Eine *statistische Kenngröße* beschreibt ein Merkmal einer Stichprobe.
- Ein *Parameter* beschreibt ein Merkmal einer Population.

# Daten

- Daten sind (Mess-)Werte, die für bestimmte Charakteristiken der Individuen (Variablen) der Population in Untersuchungen erhoben wurden.
- Klassifizierung von Daten
  - Qualitative Daten
  - Quantitative Daten (diskret vs. kontinuierlich)

# Beispiel: Wahlumfragen

- Die Population bei einer Wahlumfrage ist die Menge der Wahlberechtigten.
- Um Trends und Vorhersagen zu treffen, wird kleine Gruppe der Wahlberechtigen (Stichprobe) telefonisch befragt.
- Statistische Kenngrößen bzw. Paramter sind hierbei das Alter, Geschlecht oder auch Bundesland, in dem die Personen leben.
- Bei der Befragung werden verschiedene Daten erhoben, beispielsweise zur Person, Wahlentscheidung, Zufriedenheit mit akuteller Politik, Bekanntheitsgrad von Politikern, etc.
- Qualitative Daten, die erhoben werden, sind dabei z.B. die Zufriedenheit ("wenig zufrieden", "absolut zufrieden", "keine Aussage").
- Quantitative Daten sind hingegen das Einkommen (kontinuierlich).

# Verfahren in der Statistik

**Deskriptive Statistik** Beschreibung von meist umfangreichen Datensätzen durch informative Kenngrößen; graphische Darstellung der Information

**Explorative Statistik** Erkennen von Strukturen und Zusammenhängen, sowie Inkonsistenzen in den Daten; Erstellen von Hypothesen

**Induktive Statistik** Testen von Hypothesen; Verfassen von allgemeinen Aussagen durch Schätzen von Kenngrößen

# Deskriptive Statistik numerischer Daten: Mittelwerte

- Arithmetisches Mittel:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

- Median/Zentralwert $\tilde{x}$: Mindestens die Hälfte der Beobachtungen in der Stichprobe hat einen Wert $\leq \tilde{x}$ und mindestens die Hälfte hat einen Wert $\geq \tilde{x}$.
  Für eine *geordnete* Stichprobe $x_1, \ldots, x_n$ von $n$ Messwerten:

$$\tilde{x} = \begin{cases} x_{\frac{n+1}{2}} & \text{n ungerade} \\ \frac{1}{2}(x_{\frac{n}{2}} + x_{\frac{n+1}{2}}) & \text{n gerade} \end{cases}$$

- p-Quantil: $p \cdot 100\%$ der Daten liegen links vom Quantil und $(1-p) \cdot 100\%$ der Daten rechts vom Quantil

# Deskriptive Statistik numerischer Daten: Stichprobenvarianz

- Korrigierte Stichprobenvarianz:

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

- Unkorrigierte Stichprobenvarianz:

$$s'^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \tilde{x})$$

- Welche der Formeln benutzt werden sollte, hängt von den Eigenschaften der Stichprobe ab (Zufallsstichprobe vs. Vollerhebung).

# Fallstudie 1: Geburtenstatistik in den USA

Adaptiert von: Allen B. Downey: Think Stats - Probability and Statistics for Programmers. O'Reilly Media, Juli 2011.

# The National Survey of Family Growth

- Das National Center for Health Statistics (NCHS) der USA sammelt in regelmäßigen Abständen auf nationaler Ebene Daten zu Schwangerschaften, Geburten, Adoptionen und Familienstatus
- 6. Zyklus der Befragung im Jahr 2002
- 12.571 Befragte im Alter von 15-44 Jahren, davon 7.643 Frauen und 4.928 Männer
- Direkte Interviews, jeweils 60-80 Minuten Dauer
- Website: `http://www.icpsr.umich.edu/nsfg6/`

# Einlesen der Daten

- Daten sind im *fixed width format*
- Sie können unter Angabe der Breite identifiziert werden
  - Spalten 1-12: Fallnummer
  - Die nächsten 262 Spalten werden ignoriert
  - Spalten 275-276: Dauer der Schwangerschaft in Wochen
  - Spalte 277: Ergebnis der Schwangerschaft (1 = Lebendgeburt)
  - Spalten 278-279: Reihenfolge bei Geschwistern

```
> preg <- read.fwf(file="2002FemPreg.dat",
          header=FALSE, width=c(12,-262,2,1,2))
> colnames(preg) <- c("ID","duration",
          "outcome","birthorder")
```

# Dauer der Schwangerschaften

```
> summary(preg$duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   13.00   39.00   29.53   39.00   50.00
```

`summary` liefert einen ersten Überblick:

- Minimum und Maximum
- Median (50% Percentil bzw. 0.5-Quantil) und arithmetischer Mittelwert
- 25% und 75% Percentil (bzw. 0.25-Quantil und 0.75-Quantil)

# Dauer der Schwangerschaften

Alternativen zu `summary`:

```
> min(preg$duration)
[1] 0
> max(preg$duration)
[1] 50
> quantile(preg$duration)
  0%  25%  50%  75% 100%
   0   13   39   39   50
> quantile(preg$duration, probs=seq(0,1,by = 0.20))
  0%  20%  40%  60%  80% 100%
   0   11   36   39   39   50
```

# Visualisierung: Histogramm

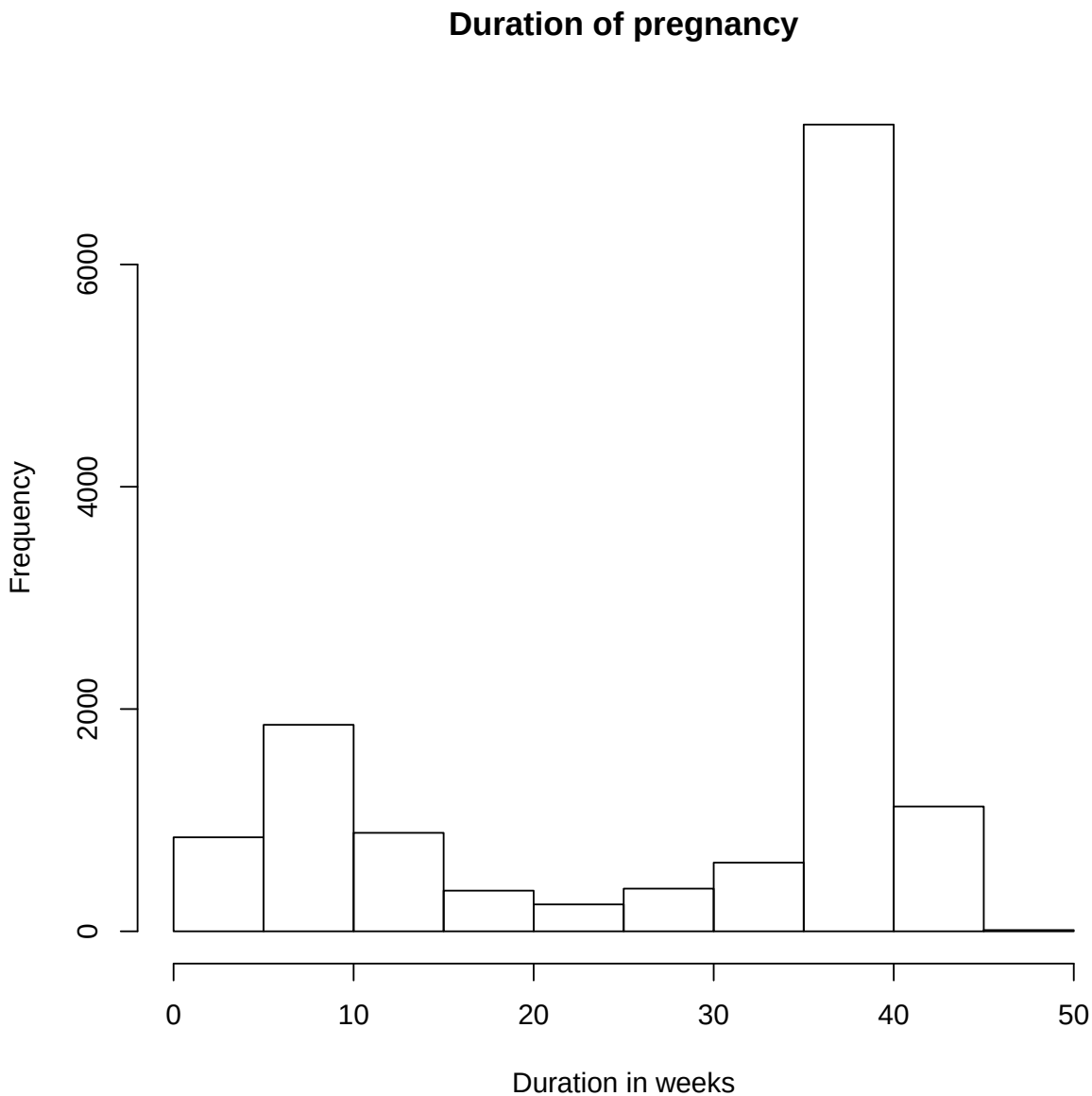
- Direkte Ausgabe

```
> hist(preg$duration,main="Duration of pregnancy",
      xlab="Duration in weeks")
```

- Ausgabe als pdf-Datei

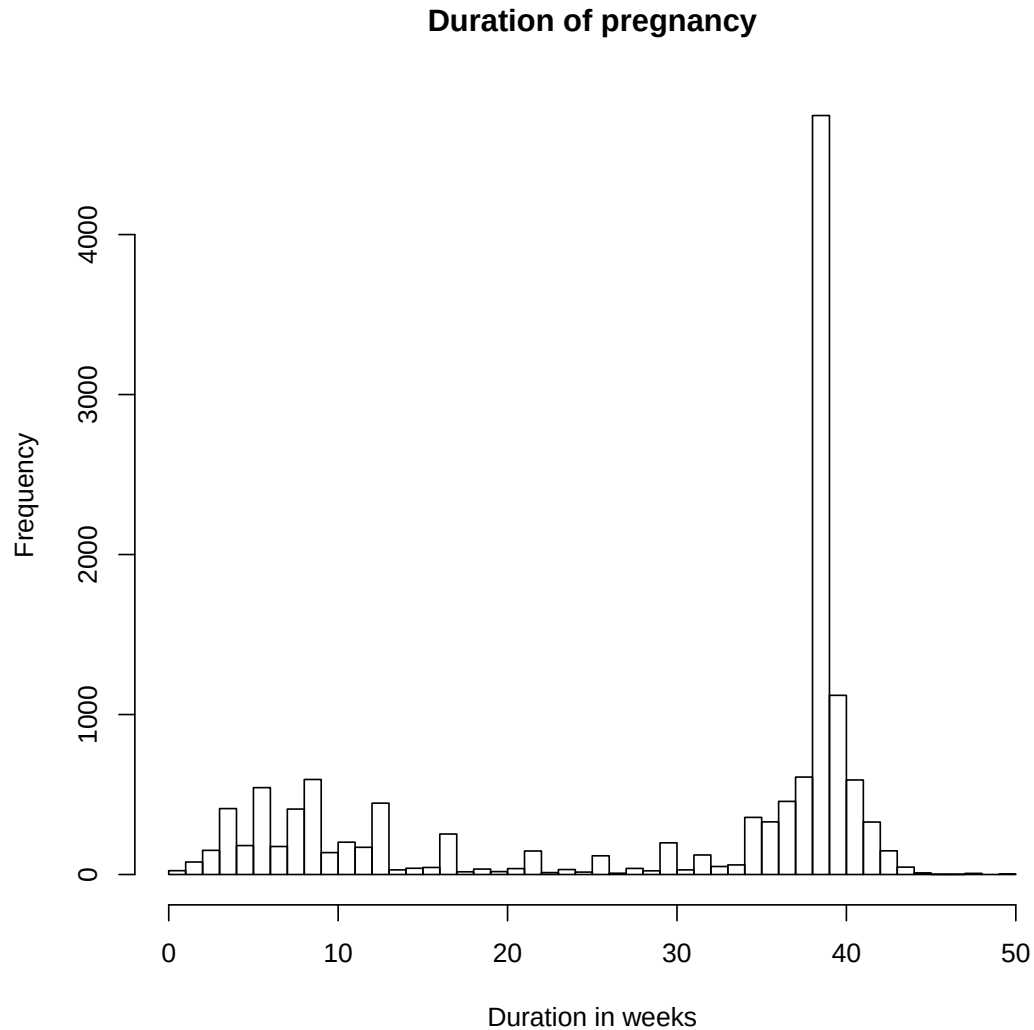
```
> pdf("duration.pdf")
> hist(preg$duration,main="Duration of pregnancy",
      xlab="Duration in weeks")
> dev.off()
```

# Visualisierung: Histogramm



# Visualisierung: Histogramm

```
> hist(preg$duration,main="Duration of pregnancy",
      xlab="Duration in weeks",breaks=50)
```



# Kommen Erstgeborene früher oder später als erwartet?

- Auswahl der Datensätze, die sich auf Erstgeborene und später geborene Geschwister beziehen

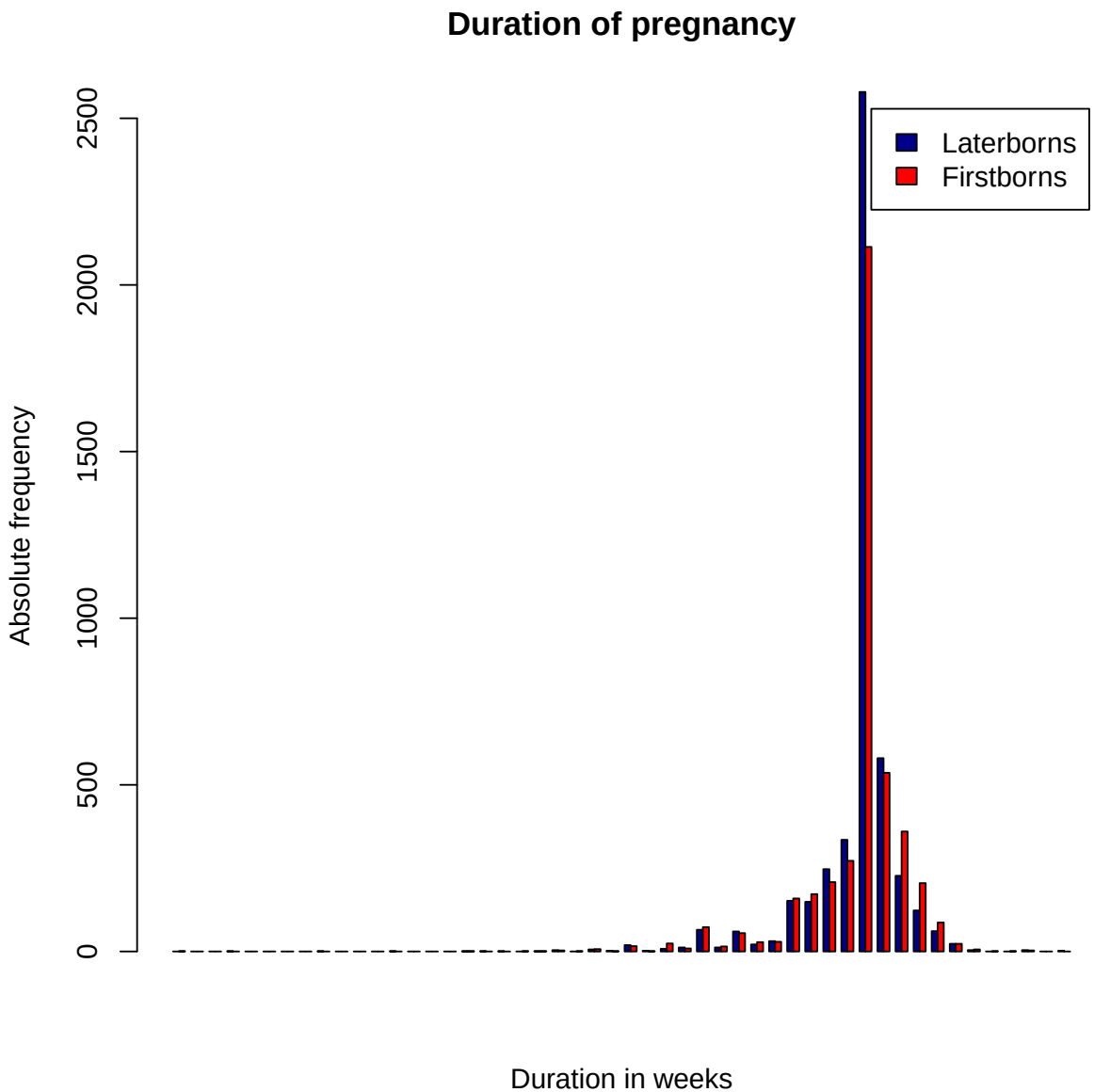
```
> firstborns <- subset(preg,preg$birthorder==1)
> summary(firstborns$duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    39.0    39.0    38.6    40.0    48.0

> laterborns <- subset(preg,preg$birthorder>1)
> summary(laterborns$duration)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   4.00   39.00   39.00   38.52   39.00   50.00
```

# Erstellen der Histogramme sowie einer vergleichenden Grafik

```
> bins=seq(0,50)
> D1 <- hist(laterborns$duration,plot=FALSE,
     breaks=bins)$counts
> D2 <- hist(firstborns$duration,plot=FALSE,
     breaks=bins)$counts
> dat <- rbind(D1,D2)
> barplot(dat, beside=TRUE,
    main="Duration of pregnancy",
    xlab="Duration in weeks",
    ylab="Absolute frequency",
    legend = c("Laterborns","Firstborns"),
    col=c("darkblue","red"))
```

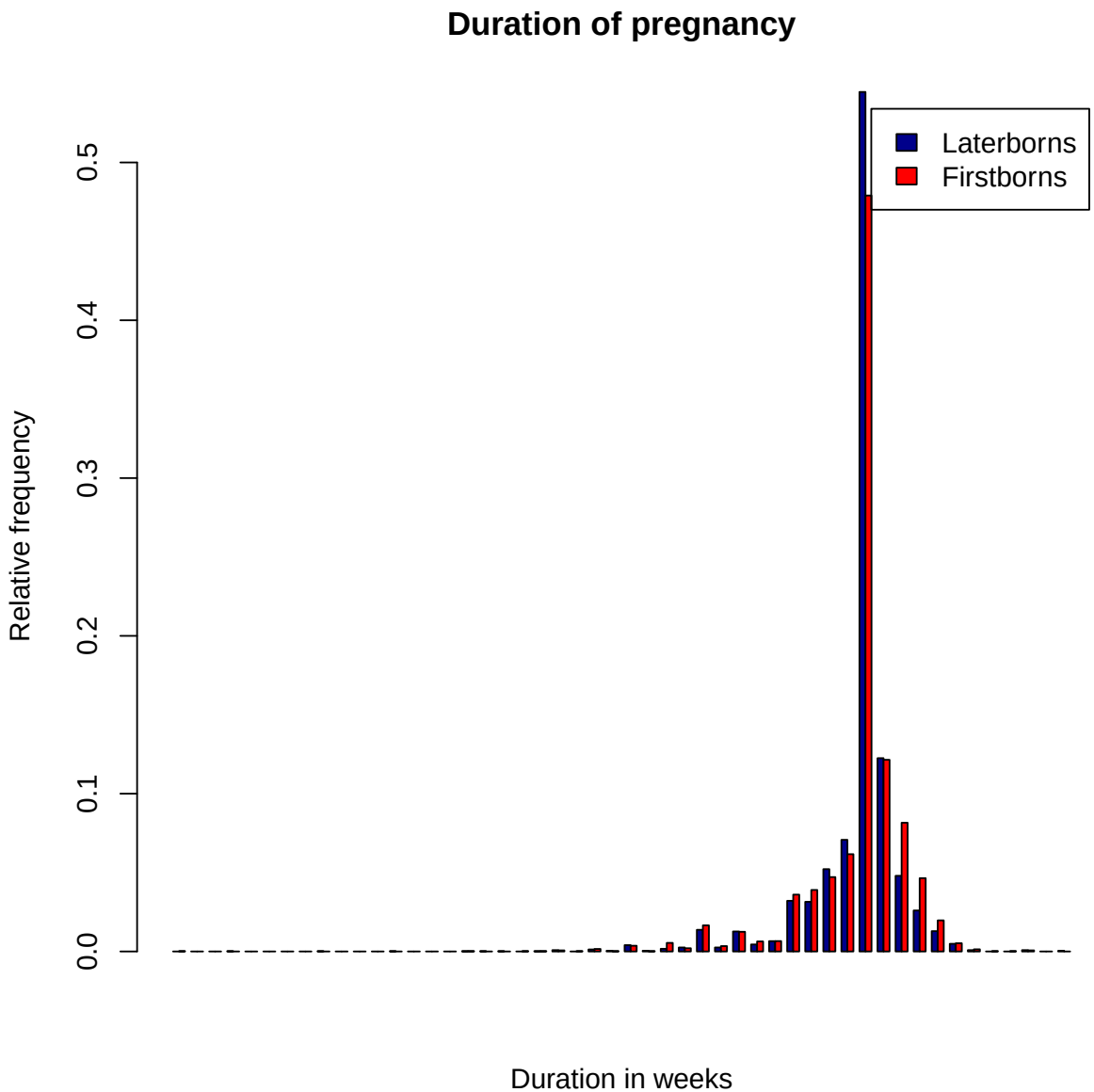
# Visualisierung: Barplot



# Ermitteln der relativen Häufigkeiten

```
> D1_rel <- D1 / nrow(laterborns)
> D2_rel <- D2 / nrow(firstborns)
> dat_rel <- rbind(D1_rel, D2_rel)
```

# Visualisierung



# Offene Fragen

- Sind die Daten repräsentative? (Stichwort: Oversampling von Minderheiten)
- Sind die Daten konsistent? Haben wir die richtigen Filter gewählt? Betrachten wir tatsächlich nur die Lebendgeburten?
- Sind die Unterschiede nur zufällig oder statistisch signifikant?

# Zum Umgang mit R

# Was ist R?

- R bezeichnet eine Programmiersprache sowie ein nichtkommerzielles Programm zur Datenanalyse und Erstellung von Grafiken
- Informationen, Download, Dokumentation: `www.r-project.org`
- Verwendung durch Eingabe von Befehlen über die R-Konsole

# Arbeiten mit R

- Der *Workspace* enthält alle Objekte der aktuellen R-Sitzung.
- Speicher und Laden des Workspace:

```
> save.image(file = "Dateiname")
> load(file = "Dateiname")
```

- Auflisten aller vorhandenen Objekte: `ls()`
- Löschen eines Objekts: `rm(obj)`

# Skripte speichern und ausführen

- Skripte sind Textdateien, die eine Sammlung von R-Befehlen enthalten
- Sie können im Skriptfenster oder anderen Texteditoren bearbeitet werden
- Typische Dateiendung für R-Scripte ist `.R`, z.B. `mein_script.R`
- Laden und Ausführen eines Skripts:

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

# Hilfe zu einzelnen Befehlen

- Integriertes Hilfesystem ermöglicht schnelles Nachschlagen von Syntax und Parametern einzelner Funktionen

```
> ?mean
> help(mean)
> help("mean")
> example(mean)  # Anwendungsbeispiel
```

- Suchen in der integrierten Hilfe

```
> ??mean
> help.search("mean")
> apropos("mean")
```

# Programmieren in R

# Basisdatentypen

| Datentyp | | Beispiel |
|---|---|---|
| `logical` | bool'sche Werte | TRUE bzw. FALSE |
| `numeric` | ganze und reelle Zahlen | 4, 5.23, -67 |
| complex | komplexe Zahlen | 2.3 + 5i |
| character | Zeichenketten | "Text" |

- Jeder Wert eines Datentyps kann durch die in der Tabelle darunter aufgeführten Datentypen dargestellt werden.
- `TRUE` wird dabei durch 1, `FALSE` durch 0 repräsentiert.
- Die Funktion `mode()` ermittelt den Datentyp von Variablen und Konstanten.
- Konvertieren von Werten: as.complex(), as.character(), ...
- Überprüfung des Datentyps: `is`.numeric(), is.character(),...

# Spezielle Werte

- `NaN` (*Not a Number*), z.B. bei der Berechnung von `sqrt(-3)`
- `Inf` und `-Inf`(*infinity*), z.B. bei der Berechnung von `1/0`
- `NA` (*Not Available*) bezeichnet fehlende Datensätze

# Vektoren

- Vektoren sind Datenstrukturen, die aus einer geordneten Menge von Elementen bestehen.
- Erzeugen von Vektoren

```
> a <- c(3,5,6) # explizit definierte Eintraege
> b <- 5:10     # aufsteigende Reihe von 5 bis 10
> c <- 4:-4     # absteigende Reihe von 4 bis -4
> e <- seq(-1,1,by = 0.1)
                # Folge von -1 bis 1 mit Abstand 0.1
> f <- rep(2,3) # 3fache Wiederholung des Werts 2
> g <- rep(c(1,2),3)
                # 3fache Wiederholung des Vektors
```

# Rechnen mit Vektoren

- Operationen auf Vektoren werden in der Regel komponentenweise durchgeführt, z.B. Addition oder Multiplikation mit Skalaren:

```
> c(1,2,3) + 2
[1] 3 4 5
> c(1,2,3) * -1
[1] -1 -2 -3
```

- Auswertung von Funktionen erfolgt häufig auch komponentenweise:

```
> round(c(0.1,2.3,5.6))
[1] 0 2 6
```

# Wichtige Funktionen auf Vektoren

- Länge eines Vektors: `length()`
- Arithmetisches Mittel der Einträge: `mean()`
- Summe der Einträge: `sum()`
- Kumulierte Summe der Einträge: `cumsum()`
- Produkt der Einträge: `prod()`
- Konkatenation von Vektoren aus Zeichenfolgen: `paste()`

# Matrizen

- Matrizen sind zweidimensionale Objekte.
- Datensätze liegen häufig in der Form von Matrizen vor.
- Matrizen können einfach aus Vektoren erzeugt werden.

```
> matrix(1:10, nrow=2)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> matrix(1:10, ncol=2)
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
```

# Wichtige Funktionen auf Matrizen

- Dimensionen einer Matrix: `dim()`
- Operationen wie Addition von zwei Matrizen sind nur möglich, wenn die Dimensionen der Matrizen übereinstimmen.
- Zeilen- bzw. spaltenweises Kombinieren von Matrizen und Vektoren: `cbind()` bzw. `rbind()`

# Datensätze

- Data frames können Elemente verschiedener Datensätze enthalten.

```
a <- c(10,20,15)                    # numerisch
b <- factor(c("m", "w", "m")) # Faktor Geschlecht
c <- c(2,5,8)                       # numerisch
myframe <- data.frame(a,b,c)
```

# Arrays und Listen

- Arrays verallgemeinern Matrizen auf höherdimensionale Objekte.
- Listen können im Kontrast zu Vekoren Elemente verschiedener Datentypen enthalten.

# Indizierung von Vektoren und Matrizen

```
> d <- -10:10
> d[6]
[1] -5
> d[10:15]
[1] -1  0  1  2  3  4
> d[c(2,5,9)]
[1] -9 -6 -2
> m <- matrix(1:10, ncol = 5)
> m[2,5]
[1] 10
> m[1,1]
[1] 1
> m[2,1:5]
[1]  2  4  6  8 10
```

# Indizierung von Vektoren und Matrizen

- Auswahl von Elementen mittels Filter `which()`

```
> which(d>3)
[1] 15 16 17 18 19 20 21
> d[which(d>3)]
[1]  4  5  6  7  8  9 10
```

- Bei negativen Indizes werden die entsprechenden Einträge gelöscht

```
> d <- 1:5
> d[-2]
[1] 1 3 4 5
```

# Die Funktion apply

- Iteratives Programmieren mit Schleifen sind untypisch und ineffizient in R.
- Stattdessen können mittels `apply` Funktionen zeilen- bzw. spaltenweise auf Matrizen anwenden.

```
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
> apply(m,1,max)  # 1 = spaltenweise
[1]  9 10
> apply(m,2,max)  # 2 = zeilenweise
[1]  2  4  6  8 10
```

# Zuweisungen

```
a <- sqrt(2)
```

- Bezeichner dürfen nur aus Buchstaben, Ziffern und Punkten (.) bestehen
- R ist case-sensitive, d.h. R unterscheidet zwischen Groß- und Kleinbuchstaben

# Funktionen

- Es gibt zahlreiche vordefinierte Funktionen: sqrt(), abs(), round(), log(), sin(), ...
- Definition von eigenen Funktionen

```
Funktionsname <- function(Argumente) {
    Definition der Funktion
}
```

- Beispiel:

```
standardabweichung <- function(y) {
    n <- length(y)
    sqrt((1/(n-1)) * sum ((y - mean(y))^2))
}
```

- Aufruf von Funktionen: Funktionsname(Argumente)

# Datenimport und -export

# Manuelle Dateneingabe

- Mit der `scan()` Funktion können Vektoren direkt auf der Kommandozeile manuell eingegeben werden.

```
> x <- scan()
1:
```

- Über den Dateneditor in R:

```
> neue.daten <- data.frame() # leerer Datensatz
> fix (neue.daten)
```

# Einlesen von externen Datensätzen: Hinweise

- Typische Rohdatenformate: .xls, .txt, .csv
- Daten aus Excel (.xls) sollten am besten als .csv-Datei abgespeichert und dann importiert werden (siehe auch Pakete *RODBC* oder *xlsReadWrite*)
- Daten einer Beobachtungseinheit/Messungen pro Zeile
- Variablennamen in der ersten Zeile, am besten ohne Sonder- und Leerzeichen
- Messwerte möglichst ohne Maßeinheit
- Einheitliches Dezimaltrennzeichen (“.” oder “,”)
- Fehlende Werte entweder durch `NA` markieren oder leer lassen

# Einlesen von externen Datensätzen

```
> messungen <- read.table(file = "datei.txt",
                          header = TRUE, sep =";", dec = ".")
```

- Der Dateiname sollte möglichst den ganzen Pfad samt Endung enthalten.
- Bei `header = FALSE` enthält die Datei keine Variablennamen.
- Der Separator `sep` trennt die einzelnen Spalten.
  Bei Tabulatortrennzeichen: `sep = "\t"`.
- Das Dezimaltrennzeichen `dec` sollte sich vom Separator unterscheiden.
- Alternativen: `read.csv()` und `read.csv2()`

# Export von Vektoren

```
> write(x,"datei.txt.",ncolumns = 1)
        # Export von Vektor
> write.table(messungen,"datei.txt")
        # Export von Datensaetzen
> write.csv2(messungen,"datei.csv")
```

- Der Dateiname sollte möglichst den ganzen Pfad samt Endung enthalten.
- Bei `header = FALSE` enthält die Datei keine Variablennamen.
- Der Separator `sep` trennt die einzelnen Spalten.
  Bei Tabulatortrennzeichen: `sep = "\t"`.
- Das Dezimaltrennzeichen `dec` sollte sich vom Separator unterscheiden.
- Alternativen: `read.csv()` und `read.csv2()`

# Visualisierung

# Grafiken in R

- Einfache Grafikerstellung durch vordefinierte R Befehle
- Verschiedene Darstellungsmöglichkeiten
- Export in diverse Datenformate (.jpg, .pdf, etc.)
- Nächste Sitzung: Grafiksystem `ggplot2`

# Einfache Grafik

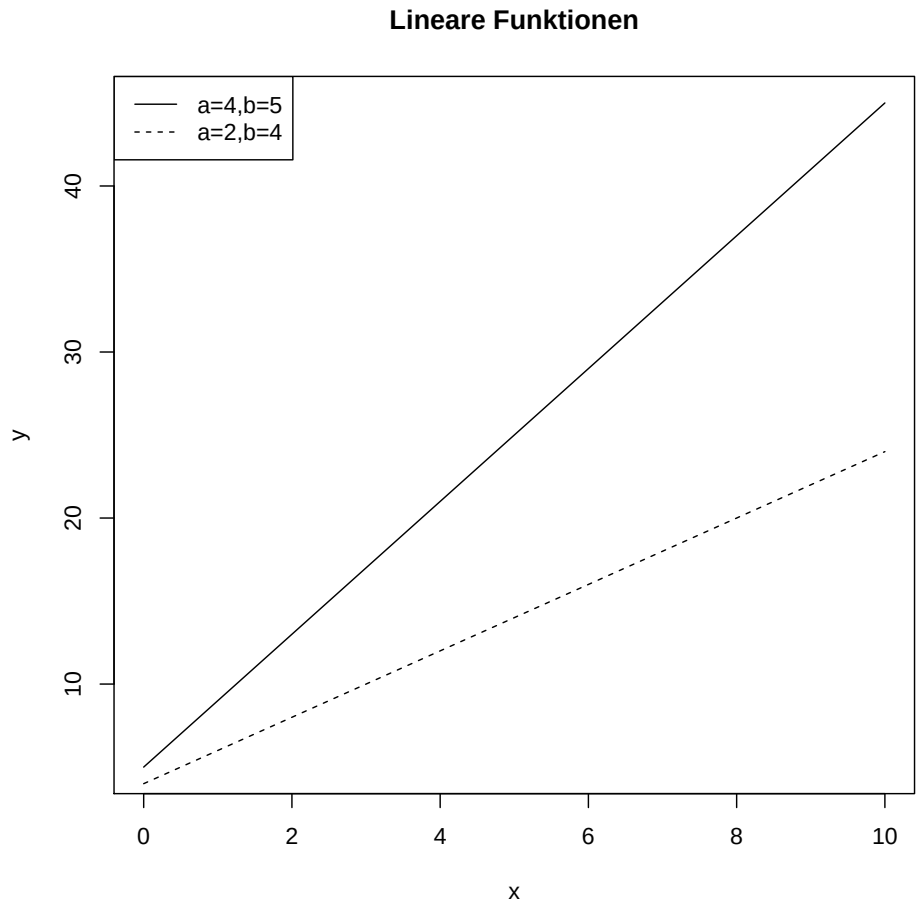
- Gegeben je ein numerischer Vektor mit x-Werten und y-Werten

```
> linear <- function(x,a,b) {a + x*b}
> x <- seq(0,10)
> y <- linear(x,6, 7)
> plot(x,y)
```

- Weitere Parameter:
  - Titel `main = "..."`
  - Achsenbeschriftung `xlab = "..."` und `ylab = "..."`
  - Art der Darstellung `type = "..."` (`l` Linie,`p` Punkte,...)
  - Linienbreite `lwd = ...` (1,2, ...)
  - Weitere Optionen:
    `http://de.wikibooks.org/wiki/GNU_R:_plot`

# Kombination von Grafikelementen

```
> plot(x,linear(x,4,5),type="l",lty=1,xlab="x",ylab="y,
      main="Lineare Funktionen")
> lines(x,linear(x,2,4),lty=2)
> legend("topleft",c("a=4,b=5","a=2,b=4"),lty=c(1,2))
```



# Weitere Arten von Grafik

| | |
|---|---|
| `barplot()` | Balkendiagramm |
| `pie()` | Kreisdiagramm |
| `boxplot()` | Boxplot |
| `curve()` | Funktionen |
| `hist()` | Histogramm |
| `pairs()` | Scatterplot |
| `mosaicplot()` | Mosaikplot |

# Exportieren von Diagrammen

- Angabe des Ausgabeformats und -ortes, z.B. `jpeg("dateiname.jpg")`, `pdf("dateiname.pdf")`
- Plotten der Grafikelemente
- Schließen der Ausgabe mittels `dev.off()`

```
> pdf("linear.pdf")
> plot(x,linear(x,4,5),type="l",lty=1,xlab="x",ylab="y"
      ,main="Lineare Funktionen")
> lines(x,linear(x,2,4),lty=2)
> legend("topleft",c("a=4,b=5","a=2,b=4"),lty=c(1,2))
> dev.off()
```

# Datenmanagement

# Fallstudie 2: UFO-Sichtungen

Aus: Drew Conway, John Myles White: Machine Learning for Hackers - Case Studies and Algorithms to Get You Started. O'Reilly Media, Februar 2012.

# Daten zu UFO-Sichtungen

- Rund 60.000 Datensätze gesammelt durch das National UFO Reporting Center in den letzten Jahrhunderten
- Aufbau der Daten

| | |
|---|---|
| DateOccured | Zeitpunkt der Beobachtung |
| DateReported | Zeitpunkt der Meldung |
| Location | Ort |
| ShortDescription | Kurzbeschreibung |
| Duration | Dauer |
| LongDescription | Ausführliche Beschreibung |

# Import des Datensatzes

```
ufo <- read.delim(file="ufo_awesome.tsv",sep="\t",
       na.strings="", header=FALSE,
       stringsAsFactors=FALSE)
```

Daten-Download: `https://github.com/johnmyleswhite/ML_for_Hackers/tree/master/01-Introduction`

# Struktur von Datensätzen

- Mit der Funktion `str()` erhält man einen Überblick zur Struktur eines Datensatzes.

```
> str(ufo)
'data.frame': 61870 obs. of  6 variables:
 $ V1: chr  "19951009" "19951010" "19950101" "19950510" ...
 $ V2: chr  "19951009" "19951011" "19950103" "19950510" ...
 $ V3: chr  " Iowa City, IA" " Milwaukee, WI" " Shelton, WA" ...
 $ V4: chr  NA NA NA NA ...
 $ V5: chr  NA "2 min." NA "2 min." ...
 $ V6: chr  "Man repts. witnessing flash ...
```

# Schritt 1: Benennung der Spalten

- Benannte Spalten vereinfachen den Umgang mit den Daten, da weniger Verwechslungen

```
names(ufo) <- c("DateOccurred","DateReported",
"Location","ShortDescr", "Duration","LongDescr")
```

# Schritt 2: Datumskonvertierung

- Datumsformat scheint "YYYYMMDD" (year-month-day)

```
> ufo$DateOccurred <- as.Date(ufo$DateOccurred,
                              format="%Y%m%d")
Fehler in strptime(x, format, tz = "GMT") :
  Eingabe-Zeichenkette ist zu lang
```

- Offensichtlich haben einige Eingaben das falsche Format!
- Ausweg: Ausfiltern der fehlerhaften Datensätze oder Ausbessern von Hand mit `fix()` (häufig nicht möglich)

```
good.rows <- ifelse(nchar(ufo$DateOccurred) == 8 &
                    nchar(ufo$DateReported) == 8,
                    TRUE,FALSE)
ufo_clean <- ufo[good.row,]
```

- Dann Datumseinträge konvertieren mit `as.Date()`

# Filtern von Datensätzen

- Einfache Filterkriterien

| | |
|---|---|
| $==$ | gleich |
| $!=$ | ungleich |
| $>, >=$ | größer (gleich) |
| $<, <=$ | kleiner (gleich) |

- Bei mehreren Möglichkeiten durch `%in%`

```
haar %in% c("blond","braun")
```

- Kombination von Filterkriterien

| | |
|---|---|
| & | Und |
| \| | Oder |
| ! | Negation |

# Schritt 3: Konvertieren der Ortsinformation

- Ortsformat ist scheinbar “Stadt, Staat”

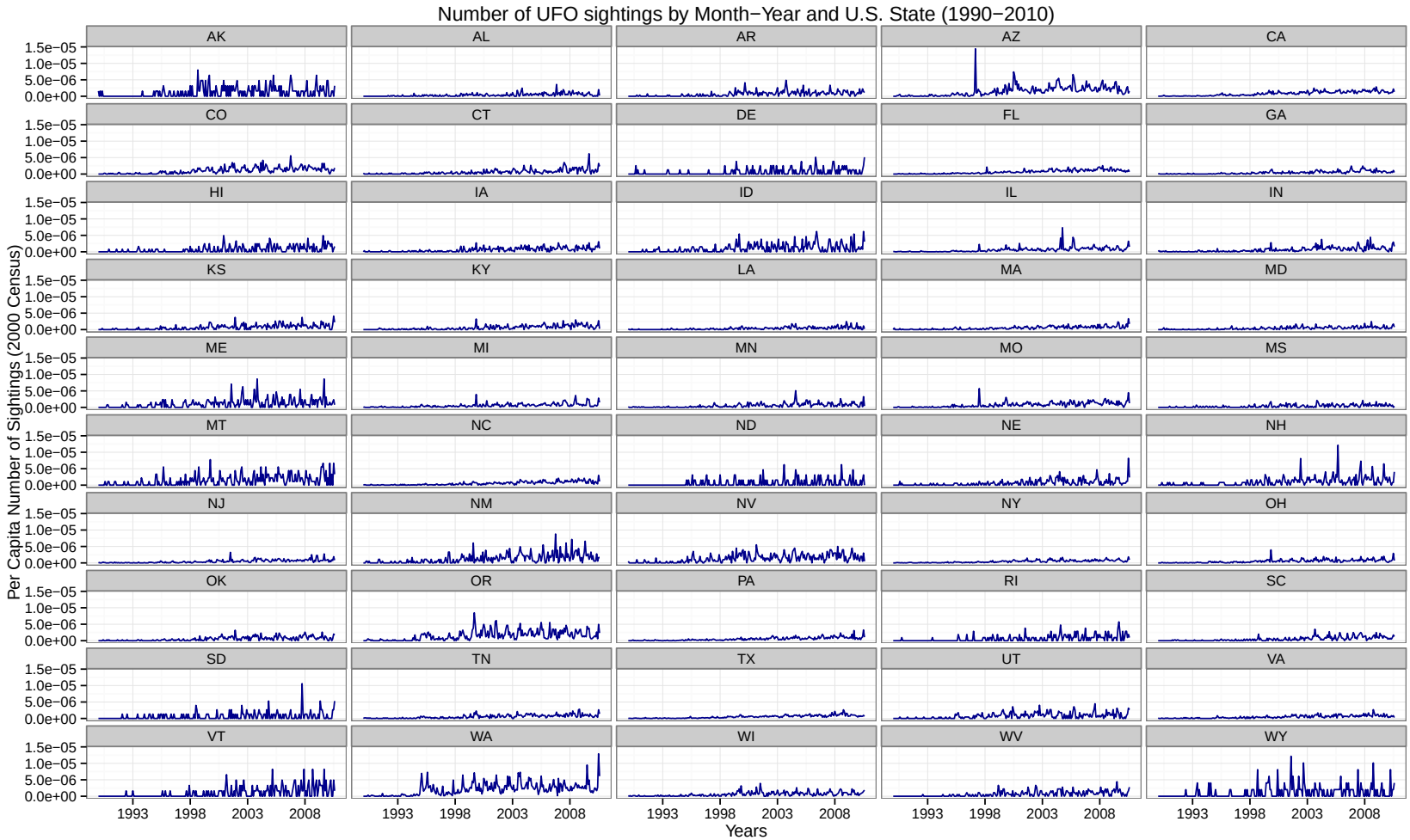
```
get.location <- function(l) {
  split.location <- tryCatch(strsplit(l, ",")[[1]],
                             error = function(e) return(c(NA, NA)))
  clean.location <- gsub("^ ","",split.location)
  if (length(clean.location) > 2) {
    return(c(NA,NA))
  } else {
    return(clean.location)
  }
}
```

- `strsplit(x,",")`
  trennt die Zeichenkette bei Komma, das folgende [[1]] unquoted den Ergebnisvektor
- `gsub("^ ","",x)`
  entfernt alle Leerzeichen am Anfang der Zeichenkette x

# Weitere Schritte

- Filtern der Datensätze, die nicht zu einem US Staat gehören
- Gruppieren der Daten per US-Staat
- Reduzieren der Zeitskala

# UFO-Sichtungen in den einzelnen US-Staaten (1990-2010)



# Weitere Operationen auf Datensätzen

- Sortieren durch `sort()`
- Sortieren nach mehreren Kriterien mit `order()`:

  ```
  teilnehmer[order(teilnehmer$geschlecht,teilnehmer$alter),]
  ```

- Löschen von Variablen durch negative Auswahl oder Zuweisung von NULL

  ```
  teilnehmer$alter <- NULL
  ```

# Umkodieren von Variablen

- Gegeben ein Datensatz mit den Teilnehmern einer Veranstaltung
- Spalte `teilnehmer$alter` bezeichnet das Alter
- Gewünschte neue Variable im Datensatz:
  Welcher der Teilnehmer ist minderjährig?

```
status <- (teilnehmer$alter < 18) * 1
          + (teilnehmer$alter >= 18) * 2
```

- Umwandlung von numerischem Wert in Faktor

```
status_factor <- factor(status,
    labels = c("minderjaehrig","volljaehrig"))
```

- Häufigkeit der einzelnen Faktoren/Kategorien

```
table(status_factor)
```