Einführung in R

Wissensentdeckung in Datenbanken SS 2009

17. April 2009

R und Editoren für R: Auf den Poolrechnern der Fakultät Statistik (M/711 und M/U18, Mathe-Tower, Campus Nord) ist R installiert. R und zusätzliche Pakete für diverse Betriebssysteme können frei unter `http://CRAN.R-project.org` heruntergeladen werden.

Wenn längere Programme geschrieben werden, ist es sinnvoll einen Editor zu benutzen. Auf den Statistik Pool-Rechnern ist `WinEdt` mit der Erweiterung `R-WinEdt` installiert. Frei erhältlich sind z.B.

- `Emacs` mit Erweiterung `ESS` (`http://ess.r-project.org/`),
- `Tinn-R` (nur für Windows) (`http://sourceforge.net/projects/tinn-r`).

Rechnen mit R:

```
1 + 3 * 5
# [1] 16

exp(1)      # Kommentar
# [1] 2.718282

exp(0)
# [1] 1

exp
# function (x)  .Primitive("exp")

pi          # Kreiszahl pi
# [1] 3.141593
```

Zuweisungen:

```
x <- 5      # die 5 wird x zugewiesen

x           # x ausgeben
print(x)
# [1] 5
```

Vektoren erzeugen und indizieren:

```
c(1,3,5)        # Vektor der Länge 3, c steht für concatenate oder combine
# [1] 1 3 5

rep(3,5)        # 5-maliges Wiederholen der 3
# [1] 3 3 3 3 3

1:5             # ganze Zahlen von 1 bis 5
# [1] 1 2 3 4 5

x <- seq(1,5,2) # jede zweite ganze Zahl von 1 bis 5
x
#[1] 1 3 5

length(x)       # Wie lang ist x?
# [1] 3

x[1]            # erstes Element von x
# [1] 1

x[c(1,2)]       # die ersten beiden Elemente von x
# [1] 1 3

x[x > 3]        # alle Elemente von x, die größer als 3 sind
# [1] 5

x[x == 3]       # alle Elemente von x, die gleich 3 sind
# [1] 3
```

Ein paar nützliche Funktionen:

Funktion	Beschreibung
`max()`, `min()`	Maximum und Minimum
`abs()`	Betrag
`sqrt()`	Wurzel
`round()`, `floor()`, `ceiling()`	runden
`sum()`, `prod()`	Summe, Produkt
`log()`, `exp()`	(natürlicher) Logarithmus, e-Funktion

Das Hilfesystem:

```
help.search("Fourier")          # Was gibt es zum Thema Fourier?

help("sum")                     # Dokumentation der Funktion sum aufrufen
?sum

# in der Konsole: Hilfe -> HTML Hilfe   # starten der Hilfe im HTML Format
help.start()
```

Datentypen und Datenstrukturen:

```
# numeric
x <- seq(1,5,2)

class(x)            # Informationen über Datentyp und Datenstruktur von x
# [1] "numeric"
mode(x)
# [1] "numeric"
str(x)
#  num [1:3] 1 3 5

is.numeric(x)       # Ist x vom Typ numeric?
# [1] TRUE

numeric(5)          # Nullvektor der Länge 5
# [1] 0 0 0 0 0


# character             # Buchstaben und Zeichenfolgen
x <- c("w", "m", "w")

class(x)
# [1] "character"
```

```
mode(x)
# [1] "character"
str(x)
# chr [1:3] "w" "m" "w"

is.character(x)     # Ist x vom Typ character?
# [1] TRUE


# factor            # qualitative Variablen
x <- factor(x)

class(x)
# [1] "factor"
mode(x)
# [1] "numeric"
str(x)
#  Factor w/ 2 levels "m","w": 2 1 2


# logical           # logische Werte
x <- 1:3
x > 2


# Vektoren
c(TRUE, "w", 1:5)       # wild gemischte Datentypen

# Matrizen
x <- matrix(1:9,3,3)    # 3*3-Matrix mit Einträgen 1 bis 9
#      [,1] [,2] [,3]
# [1,]    1    4    7
# [2,]    2    5    8
# [3,]    3    6    9

class(x)
# [1] "matrix"
str(x)
# int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
mode(x)
# [1] "numeric"

x[1,2]                  # Eintrag in Zeile 1 und Spalte 2
# [1] 4
```

```
x[x > 5]                # Einträge von x, die größer als 5 sind
# [1] 6 7 8 9

# Data Frames
Daten <- data.frame(Name = c("X","Y","Z"),
                    Groesse = c(171, 180, 159),
                    Gewicht = c(60, 65, 70),
                    Geschlecht = c("w", "m", "w"))

Daten
  Name Groesse Gewicht Geschlecht
1    X     171      60          w
2    Y     180      65          m
3    Z     159      70          w

class(Daten)
[1] "data.frame"

str(Daten)
'data.frame':   3 obs. of  4 variables:
 $ Name      : Factor w/ 3 levels "X","Y","Z": 1 2 3
 $ Groesse   : num  171 180 159
 $ Gewicht   : num  60 65 70
 $ Geschlecht: Factor w/ 2 levels "m","w": 2 1 2

mode(Daten)
[1] "list"

Daten$Name
# [1] X Y Z
# Levels: X Y Z

Daten[,1]
# [1] X Y Z
# Levels: X Y Z
```

Fehlende Werte:

```
x <- c(4, NA, 19)   # NA: Not Available
x
# [1]  4 NA 19

x == NA
# [1] NA NA NA

is.na(x)
# [1] FALSE  TRUE FALSE
```

Daten einlesen:

```
daten <- read.table("c:/.../Daten.txt", header = TRUE)

str(daten)
# 'data.frame':   3 obs. of  4 variables:
#  $ Name      : Factor w/ 3 levels "X","Y","Z": 1 2 3
#  $ Groesse   : int  171 180 159
#  $ Gewicht   : int  60 65 70
#  $ Geschlecht: Factor w/ 2 levels "m","w": 2 1 2
```

Statistik:

```
mean(daten$Groesse)             # arithmetisches Mittel
median(daten$Gewicht)           # Median
quantile(daten$Groesse,0.75)    # 0.75-Quantil
summary(daten)                  # Zusammenfassung verschiedener Kennzahlen

var(daten$Groesse)              # Varianz
sd(daten$Groesse)               # Standardabweichung

range(daten$Groesse)            # Vektor aus Minimum und Maximum

cov(daten$Groesse, daten$ Gewicht)  # Kovarianz von Groesse und Gewicht
cor(daten$Groesse, daten$ Gewicht)  # Korrelation

cov(daten[,2:3])                # Kovarianzmatrix von Groesse und Gewicht
#         Groesse Gewicht
# Groesse     111     -30
# Gewicht     -30      25
```

Grafik:

```
?plot

data(iris)
plot(iris$Sepal.Length, iris$Sepal.Width, col = as.numeric(iris$Species))

pairs(iris, col = as.numeric(iris$Species))
```

Übersicht Klassifikationsverfahren: (siehe Ligges (2005), Tabelle 7.6, S. 144)

Funktion	Paket	Beschreibung
`knn()`	class	k-nearest-neighbor
`lda()`	MASS	Lineare Diskriminanzanalyse
`naiveBayes()`	e1071	Naive Bayes Klassifikation
`NaiveBayes()`	klaR	Erweiterung der Naive Bayes Klassifikation
`nnet()`	nnet	Neuronale Netze mit einer versteckten Schicht
`qda()`	MASS	Quadratische Diskriminanzanalyse
`rda()`	klaR	Regularisierte Diskriminanzanalyse
`mda()`	mda	Mixture Discriminant Analysis
`rpart()`	rpart	Klassifikations- und Regressionsbäume
`svm()`	e1071	Support Vector Machine
`svmlight()`	klaR	Support Vector Machine
`errorest()`	ipred	Schätzung des Vorhersagefehlers (z.B. Fehlklassifikationsrate) per Kreuzvalidierung oder Bootstrap
`predict()`		Vorhersage neuer Beobachtungen mit Hilfe gelernter Klassifikationsregeln

Pakete laden:

```
?lda            # Hilfe zur linearen Diskriminazanalyse

# No documentation for 'lda' in specified packages and libraries:
# you could try '??lda'

??lda           # wie help.search("lda")

library(MASS)
?lda            # jetzt klappt es
```

Zufallszahlen:

```
## diskret:
# 2 Zufallszahlen aus der diskreten Gleichverteilung auf {1,...,5}:
sample(x = 1:5, size = 2, replace = TRUE)

# 2 Zufallszahlen aus der Binomialverteilung mit Parametern n = 10, p = 0.2:
rbinom(n = 2, size = 10, prob = 0.2)

## stetig:
# 2 Zufallszahlen aus der stetigen Gleichverteilung auf (0,1):
runif(n = 2, min=0, max=1)

# 2 Zufallszahlen aus der Standardnormalverteilung:
rnorm(n = 2, mean = 0, sd = 1)
```

Eigene Funktionen schreiben:

```
MeineFunktion <- function(Argumente){
    Anweisungen
}

f <- function(x, y = 3){
  x + y
}

f(2,5)
# [1] 7
```

Schleifen (siehe Ligges (2005), Tabelle 2.5, S. 51)

Schleife	Beschreibung
`repeat{Anweisungen}`	Wiederholung der Anweisungen
`while(Bedingung){Anweisungen}`	Wiederholung, solange Bedingung erfüllt ist
`for(i in M){Anweisungen}`	Wiederhole Anweisungen für jedes $i \in M$
`next`	Sprung in den nächsten Iterationsschritt
`break`	Sofortiges Verlassen der Schleife

```
for(i in 1:5){
  print(f(i,5))
}
# [1] 6
# [1] 7
# [1] 8
# [1] 9
# [1] 10

sapply(1:5, f)
# [1] 4 5 6 7 8
```

Literatur

Ligges, U. (2005): *Programmieren mit R.* Springer, Berlin.