Einführung in R

Klaus Schliep

16. Februar 2004

Allgemeines

R besteht ausschließlich aus Objekten. Die meisten Objekte sind entweder Daten oder Funktionen. Alle Funktionen werden mit runden Klammern geschrieben, auch wenn sie keine Argumente aufnehmen (z.B. `q()` für quit). Mit `ls()` werden alle Objekte im Workspace angezeigt. Wird ein Objekt ohne Klammern aufgerufen, wird es angezeigt. Dies kann bei großen Matrizen sehr lange dauern. Mit der `Esc`–Taste oder (in Windows) der `Stop`-Taste kann eine Berechnung unterbrochen werden. Für viele Objekte stehen die generischen Funktionen `summary`, `print` und `plot` zur Verfügung.

Für R existiert eine Vielzahl Bibliotheken, sog. Packages. Mit dem Befehl `library(vsn)` wird beispielsweise die Bibiliothek `vsn` eingebunden.

R stellt viele Hilfefunktionen bereit. Mit `help(lm)` oder `?lm` erhält man Informationen zur Funktion `lm`. `example(lm)` führt das Beispiel aus der Hilfe `help(lm)` aus und `library(help=vsn)` gibt eine Übersicht über Funktionen im Package `vsn`. Mit `help.start()` öffnet man einen Webbrowser und hat damit einen Startpunkt für weitere Recherchen.

Wir starten nun unsere erste R–Sitzung:

1 Skalare, Vektoren und Matrizen

Wir weisen als erstes einem Vektor `x` die Werte 1 bis 10 zu. Dazu haben wir mehrere Möglichkeiten:

```
> x <- 1:10
> x <- seq(1, 10, length = 10)
> x = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Das Ergebnis für `x` ist aber jeweils

```
> x

 [1]  1  2  3  4  5  6  7  8  9 10
```

Wir erzeugen noch einige weitere Variablen, Vektoren und Matrizen.

```
> a <- 1.5
> y <- rnorm(10)
> Z <- matrix(rnorm(25), nrow = 5, ncol = 5)
> X <- matrix(x, 5, 2)
> abc <- c("a", "b", "c")
```

Indizierung von Arrays

Oft will man nur auf bestimmte Werte oder Bereiche eines Arrays zugreifen. Die Indizierung erfolgt mit eckigen Klammern `[]` und beginnt im Gegensatz zu C oder JAVA bei 1.

```
> x <- c(5, 2, 7, 6, 3)
> y <- c(7, 3, 4, 3, 8)

> x[3]

[1] 7

> x[-4]

[1] 5 2 7 3

> Z[2, ]

[1] -0.2367032  0.4108771 -1.4426040 -0.9678205 -0.8924664

> Z[c(1, 4), 1:3]

           [,1]       [,2]       [,3]
[1,] -0.8169302 -0.2842375 -0.9628498
[2,]  0.4703822  0.9545462  0.7152760

> y > 5

[1]  TRUE FALSE FALSE FALSE  TRUE

> y[y > 5]

[1] 7 8
```

Wir wollen uns jetzt diese Objekte noch etwas genauer anschauen.

```
> ls()

[1] "a"   "abc" "x"   "X"   "y"   "Z"

> length(x)
```

```
[1] 5
```

```
> summary(x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    2.0     3.0     5.0     4.6     6.0     7.0
```

```
> dim(Z)
```

```
[1] 5 5
```

```
> mode(X)
```

```
[1] "numeric"
```

```
> mode(abc)
```

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

```
> rm(abc)
```

Natürlich stehen in R auch die üblichen arithmetischen und statistischen Funktionen (`'+'`, `'-'`, `'/'`, `'*'`,`'%%'` (modulo), `sin()`, `cos()`, `exp()`, `log()`, `log10()`, `abs()`, `sqrt()`, `mean()`, `sd()`, `var()`, etc.) zur Verfügung.

```
> sin(x)
```

```
[1] -0.9589243  0.9092974  0.6569866 -0.2794155  0.1411200
```

```
> exp(x)
```

```
[1]  148.413159    7.389056 1096.633158  403.428793   20.085537
```

```
> sum(y)/length(y)
```

```
[1] 5
```

```
> x^2 + y^2
```

```
[1] 74 13 65 45 73
```

Diese Funktionen werden jeweils elementweise ausgeführt, es existiert aber auch Matrix-Arithmetik. Probieren Sie dazu folgende Funktionen aus: `Z %*% X`, `X %*% Z`, `t(X)`, `diag(Z)`, `diag(5)`, `diag(1:5)` und `solve(Z)`. Was bewirken sie?

```
> plot(x = x, y = y, main = "Unsere erste Graphik")
```

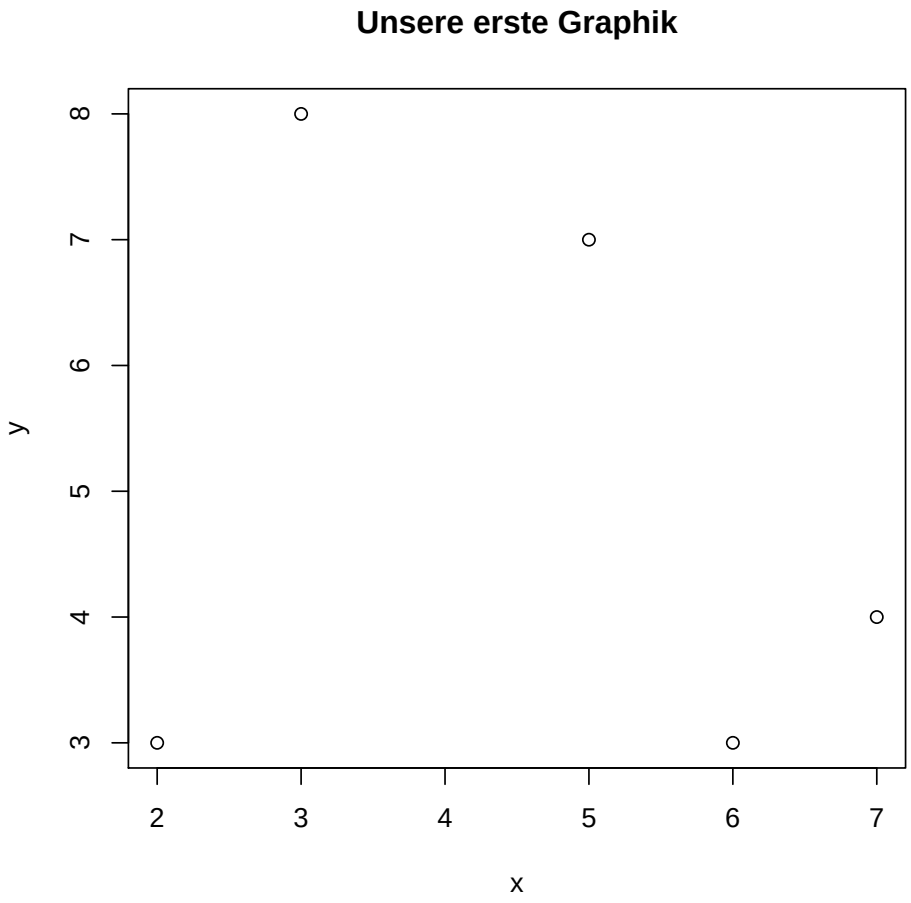

Abbildung 1: Ein paar Punkte in einem Scatterplot.

2 Listen

R kennt wie jede Programmiersprache verschiedene Datentypen. Die wichtigsten sind `numeric, character` und `logical`. R unterscheidet Zahlen aber nicht weiter in integer, double etc. In Vektoren und Matrizen oder noch allgemeiner in Arrays können jeweils nur Daten von einem Datentyp enthalten sein. In R gibt es deshalb allgemeinere Objekte, die Komponenten verschiedener Datenstrukturen aufnehmen können. Die wichtigsten hierbei sind Listen (`list`) und Dataframes (`data.frame`).

```
> daten <- list(werte = matrix(rnorm(50), 10, 5), typ = c("a",
+     "a", "b", "a", "b"), ind = factor(rbinom(10, 1, 0.6)))
> summary(daten)
```

```
      Length Class  Mode
werte 50     -none- numeric
typ    5     -none- character
ind   10     factor numeric
```

```
> daten[2]
```

```
$typ
[1] "a" "a" "b" "a" "b"
```

```
> daten$typ
```

```
[1] "a" "a" "b" "a" "b"
```

```
> daten[[2]]
```

```
[1] "a" "a" "b" "a" "b"
```

Was ergibt daten[[3]][2:4]? Greifen sie auf die 2. Spalte der Wertematrix zu!

3 Programmieren in R: Schleifen und Funktionen

```
> X <- matrix(rnorm(50), 10, 5)
```

Wir haben uns eine 10x5 Matrix mit normalverteilten Zufallszahlen erzeugt. Wir wollen nun über alle Spalten summieren und das Ergebnis in einem Vektor speichern:

```
> result <- numeric(10)
> for (i in 1:10) result[i] <- sum(X[i, ])
```

In R sind Schleifen manchmal etwas langsam, und es ist ratsam, mit der Funktion `apply` bestimmte Operationen (hier: `sum`) direkt auf Matrizen anzuwenden.

```
> result2 <- apply(X, 1, sum)
```

Was ergibt `apply(X,2,sum)`?

Zum Abschluß wollen wir diese Schleifen in eine Funktion einbinden. Die allgemeine Syntax für eine Funktion ist:

```
my.function <- function(a,b,c,...){
   < definition of function >
return( result ) }
```

Damit ergibt sich für unsere Funktion

```
> Zeilensumme <- function(x) {
+     result <- numeric(10)
+     for (i in 1:dim(x)[1]) {
+         result[i] <- sum(x[i, ])
+     }
+     return(result)
+ }
> Zeilensumme(X)

 [1] -0.69010741  0.06473175  0.07573893 -1.41619116 -3.12582602 -1.78779539
 [7]  0.60174536  2.09846241 -0.34855572  4.26211371
```

Schreiben Sie die Funktion um und verwenden Sie dabei die `apply`–Funktion.

Viel Spaß!

Literatur

- Dalgaard, P. (2002): *Introductory Statistics with R*, Springer
- Venables, W. and Ripley, B. (2002): *Modern Applied Statistics with S-PLUS*, Springer
- Venables, W. and Ripley, B. (2000): *S Programming*, Springer
- Chambers, J. (1998): *Programming With Data*, Springer

Einige nützliche Links

www.r-project.org
www.cran.r-project.org
www.bioconductor.org
www.stat.uni-muenchen.de/∼strimmer/rexpress.html