Einführung in das Statistikprogramm
R

Seminar Informatik für Biologen
Wintersemester 2004/2005
Universität Ulm

Karin Wessel

Inhaltsverzeichnis:

1. Allgemeines 3
2. Eigenschaften von R 3
3. Arbeiten mit R 4
   3.1 Interaktive Shell 4
   3.2 Funktionen 4
   3.3 Hilfe 4
   3.4 Variablen 4
   3.5 Statistikanalysen und Grafikerstellung 6
   3.6 Regression 8
   3.7 Verlassen des Programms 9
4. Erweiterungen 10
   4.1 Bioconductor 10
   4.2 S-PLUS 10
5. Quellen 10

1. Allgemeines

R ist eine Software für Statistikanalysen, die u. a. auch zur Erstellungen von Grafiken verwendet werden kann. Sie kann verstanden werden als ein „Dialekt“ der Programmiersprache S, die von den AT&T Bell Laboratories erschaffen wurde.
R wird vom *R Development Core Team* entwickelt und unter den Bedingungen der *GNU General Public Licence* als freie Software vertrieben (http://www.gnu.org).
R ist in mehreren Formen verfügbar. So sind die Quellcodes, die zumeist in C geschrieben sind, im Wesentlichen für UNIX und Linux. Es gibt R aber auch für Windows und Macintosh.

2. Eigenschaften von R

R ist keine Compiler-Sprache, sondern eine Interpreter-Sprache. Das heißt, alle Kommandos, die eingetippt werden, werden sofort ausgeführt, ohne dass ein komplettes Programm benötigt wird, wie es in dem meisten Programmiersprachen der Fall ist (vgl. C).

R ist eine objektorientierte Sprache, das bedeutet, dass Variablen, Daten, Funktionen und Ergebnisse in Form von Objekten, die einen Namen tragen, im Speicher gehalten werden. Der Anwender kann mit diesen Objekten mithilfe von Operatoren (arithmetisch, logisch, vergleichend) oder Funktionen (die selbst wieder Objekte sind) arbeiten. Alle Aktionen, die von R durchgeführt werden, werden an den Objekten im Speicher durchgeführt. Es werden keine temporären Dateien verwendet.

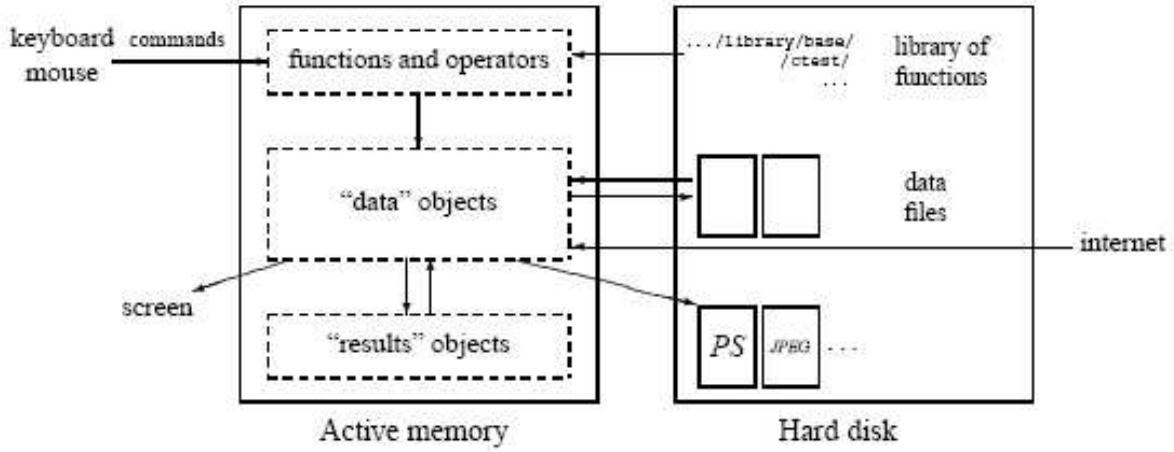

Abb. 1: Schematische Darstellung, wie R arbeitet

Der Anwender führt die Funktionen via Kommandos in einer interaktiven Shell aus. Die Ergebnisse werden direkt dargestellt, in einem Objekt gespeichert oder auf der Festplatte gespeichert (siehe Abbildung 1).
Letzteres gilt speziell für Grafiken, denn Grafiken werden in einem eigenen Fenster erstellt. Dieses ist außerhalb der interaktiven Shell und kann in verschiedenen Formaten abgespeichert werden. Aber auch die Ergebnisse und Zwischenschritte einer Statistikanalyse mit R, können als Datei gespeichert werden.

Wie im folgenden zu sehen ist, ist die Syntax der Sprache R recht einfach und intuitiv.

3. Arbeiten mit R

3.1 Interaktive Shell

Sobald das Programm R gestartet wird, erscheint die interaktive Shell. Der Prompt “>“ bedeutet, dass R auf die Kommandos des Anwenders wartet.

3.2 Funktionen

Funktionen benötigen Klammern, um ausgeführt zu werden. Wird eine Funktion ohne Klammern als Kommando eingetippt, so erscheint eine kurze Beschreibung der Funktion.

3.3 Hilfe

Selbstverständlich bietet R auch Hilfen zu verschiedenen Problemen an:
Sucht der Anwender eine bestimmte Funktion, deren Kommando er nicht kennt, so kann er mit dem Kommando `> help.search(“Suchbegriff“)` nach einer entsprechenden Funktion suchen.

<u>Beispiel:</u>
Der Anwender möchte die Standardabweichung (engl. standard deviation) berechnen. Er kennt die zugehörige Funktion allerdings nicht und gibt deshalb folgendes Kommando an:

```
> help.search(“standard deviation“)
```

In einem neuen Fenster erhält er die gesuchte Funktion: `sd()`.

Kennt der Anwender die Funktion schon, hat allerdings ein Problem mit der Syntax des Kommandos, so erhält er mit folgendem Kommando eine Hilfestellung bezüglich der angegebenen Funktion: `> ?Funktion`

<u>Beispiel:</u>
Der Anwender möchte die Standardabweichung berechnen. Er kennt die zugehörige Funktion, weiß aber nicht, wie er sie anzuwenden hat und gibt deshalb folgendes Kommando an:

```
> ?sd
```

In einem neuen Fenster erhält er u. a. eine kurze Beschreibung, Beispiele, die Syntax und weitere Details der angegebenen Funktion.

3.4 Variablen

In R kann man mit mehreren verschiedenen Variablenarten arbeiten. So kann zum Beispiel eine Variable erzeugt werden, der ein einziger Wert zugewiesen wird. Die Zuweisung erfolgt mit dem Operator `<-`, im Gegensatz zu vielen anderen Programmiersprachen, die hierfür die Operatoren `:=` oder `=` verwenden.

<u>Beispiel:</u>

```
> x<-3
```

Dieses Kommando erzeugt die Variable x, der der Wert 3 zugewiesen wird.

Hierbei muss eine Besonderheit von R erwähnt werden:
Sämtliche Variablen müssen nicht – wie es in anderen Programmiersprachen wie z. B. C der Fall ist - deklariert werden, wenn sie erzeugt werden. Stattdessen erkennt R den Typ einer erzeugten Variablen selbst.
Um den Typ eines Objekts/ einer Variablen anzuzeigen, kann die Funktion `mode()` verwendet werden.

Beispiel:

```
> x<-3
> x
[1]  3
> mode(x)
[1]  "numeric"
> A<-"Caenorhabditis"
> A
[1]  "Caenorhabditis"
> mode(A)
[1]  "character"
```

Des weiteren kann man mit Vektoren arbeiten, die wie im folgenden Beispiel mit der Funktion `c()` erzeugt werden können.

Beispiel:

```
> x<-c(1,2,5,8)
> x
[1]  1 2 5 8
```

Will man mit einer Matrix arbeiten, kann diese wie folgt erzeugt werden:

```
> m<-matrix(data=c(1,2,4,6,7,9), nr=2, nc=3)
> m
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    6    9
```

Die Funktion `matrix()` erzeugt dabei eine Matrix, mit den in Klammern hinter `data` als Vektor angegebenen Werte, sowie den angegebenen Zeilenanzahlen (`nr` von engl. number of rows) bzw. Spaltenanzahlen (`nc` von engl. number of columns).

Stellen die zu verarbeitenden Daten eine Tabelle dar, so kann diese als Matrix verstanden werden und wie oben gezeigt zur Variablen geformt werden. Eine weitere Möglichkeit zur Eingabe solcher Daten, ist der Data Editor. Dieser stellt eine excelähnliche Oberfläche zur Verfügung, wobei die Variable mit der Funktion `data.frame()` zuerst erzeugt werden muss, bevor ihr Werte zugewiesen werden können. Die dadurch erzeugte Variable ist vom Typ `"list".` Die einzelnen Tabellenspalten erhalten ebenfalls Namen und mit dem Operator `$` kann direkt auf sie zugegriffen werden.

Beispiel:
Der Anwender möchte die Daten in der zweiten Spalte (`var2`) seiner Tabelle mit dem Namen `tabelle` verarbeiten, beispielsweise indem er die Standardabweichung der darin enthaltenen Werte berechnen lässt. Das zugehörige Kommando lautet:

```
> sd(tabelle$var2)
```

Außerdem können mit der Funktion `read.table()` Daten aus Textdateien importiert werden.

Beispiel:
```
> daten<-read.table("c:/temp/daten1.txt")
```

Entsprechend können in den Analysen gewonnene Ergebnisse mit der Funktion `write.table()` in Form einer Textdatei gespeichert werden.

3.5 Statistikanalysen und Grafikerstellung

Im folgenden sind mehrere Funktionen zur Statistikanalyse, bzw. zur Erstellung von Grafiken aufgeführt:

Minimum:	`> min()`
Maximum:	`> max()`
Mittelwert:	`> mean()`
Standardabweichung	`> sd()`
Zusammenfassung statistischer Daten	`> summary()`
(gibt Minimum, Maximum, Mittelwert, Median, 1. und 3. Quantil an)	
Häufigkeitsverteilung, bzw. Histogramm	`> hist()`

Beispiel: Statistikanalyse
Es wird zuerst eine Tabelle mit dem Namen `tabelle` erzeugt:
```
> tabelle<-data.frame()
```
Im Data Editor werden in diese Tabelle die zu analysierenden Daten eingefügt. Die Tabelle soll in der Shell angezeigt werden:
```
> tabelle
      V1    V2
1 1120.8 634.1
2 1494.4 528.4
3  907.3 634.0
4 1008.0 634.0
5  378.1 739.8
6 1638.0 739.7
7 1335.8 634.0
8 1008.0 739.7
```
(Die Tabelle heißt `tabelle` und besitzt die beiden Tabellenspalten `V1` und `V2`.)
```
> min(tabelle)
[1] 378.1
> min(tabelle$V2)
[1] 528.4
> max(tabelle$V1)
[1] 1638
> mean(tabelle$V1)
[1] 1111.3
> sd(tabelle$V1)
[1] 391.92
> summary(tabelle$V1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  378.1   982.8  1064.0  1111.0  1375.0  1638.0
```

Beispiel: Histogramm/Häufigkeitsverteilung

```
> tabelle<-read.table("c:/temp/daten.txt")
> hist(tabelle§V2)
```

In einem neuen Fenster wird die zugehörige Grafik dargestellt (siehe Abb. 2).

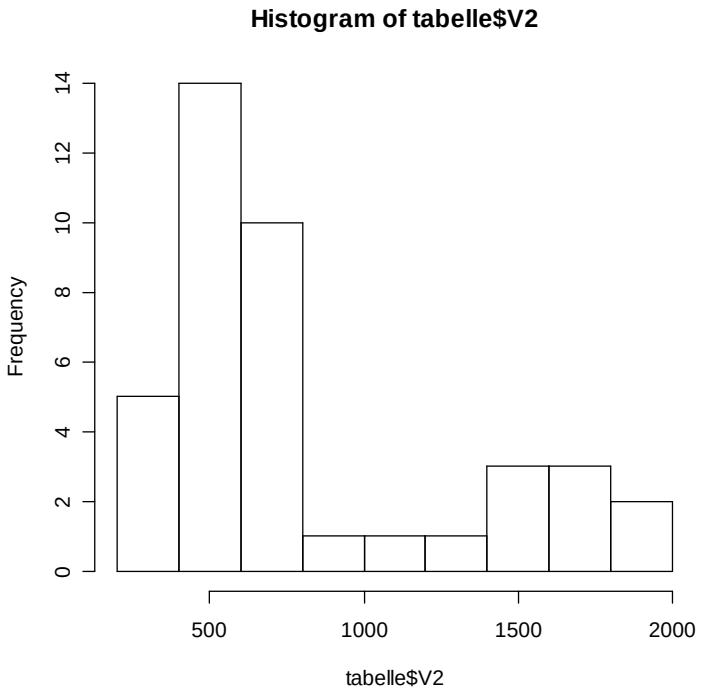

Abbildung 2: Histogramm der zweiten Spalte der Tabelle

Beispiel: Balkendiagramm mit Standardabweichungen

```
> tabelle<-read.table("c:/temp/daten.txt")
```

Zuerst werden Mittelwert und Standardabweichung berechnet:

```
> m<-mean(tabelle)
> s<-sd(tabelle)
```

Nun kann mit der Funktion `barplot()` ein Balkendiagramm mit den gewünschten Eigenschaften erstellt werden (siehe Abb. 3). Zuerst muss die Höhe der Balken angegeben werden(Mittelwert `m`). Außerdem kann die Breite der Balken (`width`), ihr Abstand voneinander (`space`) und der anzuzeigende Achsenbereich (`ylim`) bestimmt werden.

```
> barplot(m, width=c(2,2,2), space=0.5, ylim=c(0,1500))
```

Dieses Balkendiagramm wird mit der Funktion `arrows()` ausgebaut. Es müssen die Anfangs- und Endpunkte der Pfeile angegeben werden, sowie die Winkel und Längen an den Pfeilspitzen. Wird für den y-Wert des Anfangspunktes der Mittelwert `m` und für den y-Wert des Endpunktes die Summe aus Mittelwert `m` und Standardabweichung `s` gewählt, so entstehen Pfeile auf den Balken, die die jeweiligen Standardabweichungen angeben.

```
> arrows(x0=c(2,5,8),y0=m, x1=c(2,5,8), y1=(m+s),angle=90,
  length=.5 )
```

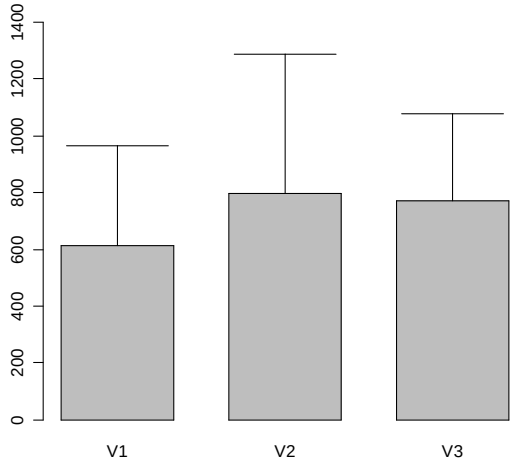

Abbildung 3: Balkendiagramm mit Standardabweichungen

3.6 Regression

Mit R können auch lineare Regressionen durchgeführt werden. Aus den angegebenen Werten werden Trendlinien bestimmt. Die zugehörige Funktion lautet `lm()`. Das Ergebnis dieser Funktion muss einer Variablen zugewiesen werden, die dann die Koeffizienten dieser Trendlinie enthält.

<u>Beispiel: lineare Regression</u>
Zuerst werden die x- und y-Werte angegeben.

```
> werte<-read.table("g:/regression.txt")
> x<-werte$V1
> y<-werte$V2
```

Diese werden in einem Koordinatensystem gegeneinander aufgetragen (siehe Abb. 4).

```
> plot(x,y)
```

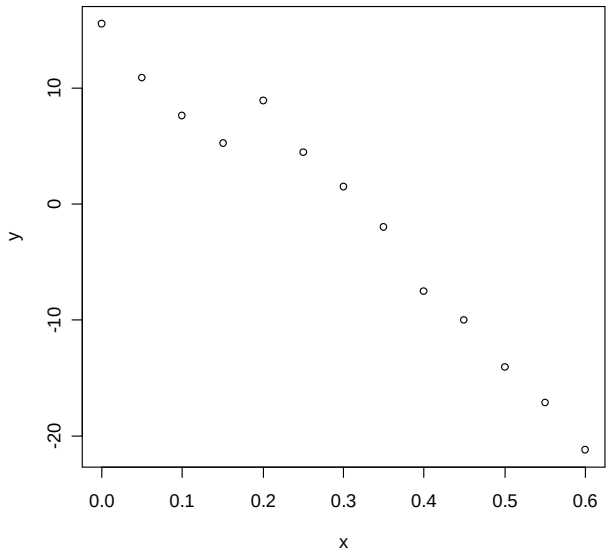

Abbildung 4: lineare Regression

Nun folgt die lineare Regression, wobei das Argument der Funktion lm, den Zusammenhang zwischen den x- und y-Werten angibt. In diesem Fall besteht ein linearer Zusammenhang (x~y).

```
> g<-coef(lm(formula=d$V2~d$V1))
> g
(Intercept)        d$V1
   16.22352   -58.52198
```

Die Variable `g` beinhaltet die Koeffizienten der ermittelten Trendlinie. Der erste Koeffizient ist der y-Achsenabschnitt der Geraden (16.22352). Der zweite Koeffizient gibt die Steigung der Geraden an (-58.52198).

Mit der Funktion `lines()` kann die Grafik aus Abb. 4 um die Trendlinie erweitert werden (siehe Abb. 5). Hierfür müssen zwei Punkte dieser Geraden angegeben werden, die sich aus den Koeffizienten der Regression ergeben. Mit dem Parameter `lwd` kann die Stärke der Trendlinie festgelegt werden.

```
> lines(c(0,0.6),c(g[1], g[2]*0.6 + g[1]), lwd=2)
```

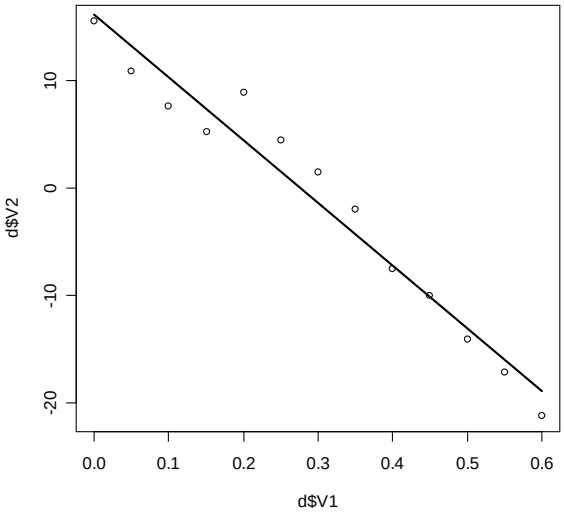

Abbildung 5: lineare Regression mit Trendlinie

3.7 Verlassen des Programms

Das Programm kann mit dem Kommando `q()` verlassen werden.

4. Erweiterungen

4.1 Bioconductor

Bioconductor ist eine freie Software, die verschiedene Werkzeuge zur Analyse und Interpretation genomischer Daten bereithält. Sie wurde von Wissenschaftlern an der Harvard Medical School/Harvard School of Public Health und Mitgliedern verschiedener US-amerikanischer und internationaler Institutionen auf der Grundlage von R zur Anwendung in der Bioinformatik erschaffen.
Weitere Informationen hierzu gibt es unter http://www.bioconductor.org/.

4.2 S-PLUS

S-PLUS ist eine kommerzielle Implementierung der Sprache S mit einer einfacheren Bedienung und vielfältigeren Möglichkeiten für den Benutzer. S-PLUS ist beispielsweise weitgehend per Maus bedienbar. Es stellt damit eine Alternative zu R dar.

5. Quellen

- "R for beginners" von E. Paradis – http://cran.r-project.org/doc/contrib/rdebuts_en.pdf
- "An Introduction to R" von W.N. Venables, D.M. Smith und R Development Core Team – http://cran.r-project.org/doc/manuals/R-intro.pdf
- "Statistical Programming with R: Part 1" von D. Mertz, B. Hutting – http://www-106.ibm.com/developerworks/linux/library/l-r1/
- R – http://www.r-project.org/