S / S-PLUS 3.4
Unterlagen zur UNIX Version

Markus Hürzeler, Martin Mächler, Andreas Ruckstuhl, Caterina Savi
Prisca Risold , Christian Keller und Roberto Invernizzi
Seminar für Statistik, ETH Zürich

Update: Okt. 1998 — 29. Oktober 1999, Date: 1999/04/12 14:00:15

Inhaltsverzeichnis

1 Benutzeroberfläche

1.1 Common Desktop Environment (CDE)

Um die Maschinen im HG E22, E23, resp. E27 zu benutzen, braucht man ein Konto mit zugehörigem Passwort. Die Benutzervorschriften sind strikte einzuhalten. Nach dem Einloggen wird automatisch die Benutzeroberfläche "Desktop" (Common Desktop Environment, "CDE", "dt") gestartet. Unten haben Sie eine Art "Steuerpult" vor sich, woraus Dateimanager, E-mail und vieles anderes gestartet werden kann.

Durch Anklicken des Hintergrundes mit der rechten Maustaste kommt man auch direkt in ein Menu (Workspace), wo man unter anderem folgende Möglichkeiten findet: Ausloggen (`exit`) und unter `Programs...` kann man eine `Console` oder einen `Text Editor` öffnen.

Die Fensteroperationen erhält man durch Anklicken des oberen Randes mit der rechten Maustaste, mit der linken Taste kann man Fenster verschieben, vergrössern, etc.
Damit ein Fenster aktiv wird, muss man mit dem Cursor aufs Fenster fahren und anklicken.

Passwort ändern: Es ist sinnvoll, bei einer der ersten Sitzungen das Passwort mit dem UNIX Kommando `yppasswd` zu ändern.

1.2 Unix

In UNIX (und daher auch S-plus) werden Gross– und Klein–Buchstaben immer unterschieden, z.B. ist eine Variable `x` nicht dasselbe wie `X`. Üblicherweise schreibt man (fast) alles klein.
Um die Output-Dateien anzuschauen, und auf den Drucker zu schicken, sind einige wenige UNIX-Befehle nötig, bzw. nützlich. Wie erwähnt, können im "Desktop" auch viele Dinge direkt via Datei–Manager und Menu–System ausgeführt werden.

Befehl	Erklärung	Beispiel
`apropos` *keyword*	gibt Hilfe zu einem Stichwort (d.h. eine Liste von Befehlen)	`apropos printer`
`man` *command*	'on-line help': Gibt 'MANual' zu best. Befehl	`man more; man ls; man man`
`cat` *filename*	Bildschirm-Ausgabe von Files	`cat login_msg`
`more` *filename*	Bildschirm-Ausgabe *seitenweise*; "?" gibt Liste der Möglichkeiten	(auch von `man` verwendet)
`ls` *[filenames]*	'Verzeichnis' (Datei-Verzeichnis) auflisten	`ls -F po*`
`cp` *fil1 fil2*	Files kopieren ('fil2 := fil1')	`cp this this.save`
`mkdir` *newd* `cd` *newd*	Directories ("Verzeichnisse") kreiieren und "betreten"	`mkdir outputs`
`mv` *fil1 fil2*	Files umbenennen ('**m**o**v**e')	`mv oldname newname`
`rm` *files*	Löschen von Files ('**rem**ove')	`rm out*` (löscht alle files, die mit 'out' beginnen)
`textedit` *file*	Editieren mit Menu, Maus, etc.	
`xemacs` *file*	Editieren mit '(x)emacs' (`e` ist `alias` von `emacs`); **empfohlen**	
`alias`	Zur Definition und Abfrage von Abkürzungen	`alias dir 'ls -sF'`
`lpr` *file*	Ausdrucken von Dateien	`lpr -P e22 .login #--?`
`S`	Aufruf des Statistik- und Graphikpaketes S-plus (*nicht* standard UNIX!)	`S`
`logout` oder `exit`	UNIX verlassen: "ausloggen"	`logout`

2 S-plus–Benützung

> S-plus *is a powerful tool for data analysis, providing you with convenient features for exploratory data analysis, modern statistical techniques, and creating your own* S-plus *programs.*

2.1 Aufruf / On-line-Hilfe / Ausstieg

Mit den Befehlen `S` oder `Splus` wird S-plus aus dem UNIX (bzw. mit `M-x S` aus dem Emacs-Editor) gestartet.

Der Befehl `help.start()` startet die Window-Menu-Version der On-line-Hilfe. Da diese sehr wertvoll ist, wird sie auf den Übungskonten automatisch atkiviert. Daher benötigt das obige Aufstarten eine gewisse Zeit.

$>$ `help()` oder `?` liefert Dokumentation über die On-line-Hilfe von S-plus.
$>$ `help(lm)` oder `?lm` liefert die Dokumentation des wichtigen Befehls `lm`.

Die Beispiele in den Help-Files sind i.A. sehr gut, es lohnt sich, sie genau anzuschauen.

Bei Problemen:
`CTR-C` [1] stoppt die Ausführung von S-plus-Funktionen, `CTR-Z` unterbricht "temporär".

$>$ `q()` **Ausstieg** aus S-plus.

Hinweis: Alle Anweisungen werden in einem Audit-file `.Data/.Audit` aufgezeichnet, ausser dies wird explizit verhindert.

2.2 Allgemeines

Gross- und Kleinschreibung wird unterschieden — wie IMMER im UNIX!

Objektnamen müssen mit einem Buchstaben beginnen, Zahlen und Punkte innerhalb der Namen sind erlaubt (aber keine _, vgl. 4.1). Namen, die schon von S-plus-Funktionen getragen werden, sollten im eigenen Interesse vermieden werden (insbesondere z.B. `c, t, F, max`). S-plus ist eines der wenigen Programme, die es erlauben, eine Programm-Funktion lokal abzuändern.

Es gibt diverse S-plus-Objekt-Verzeichnisse (Objekt–Datenbanken). Eine Liste der aktuellen Verzeichnisse ergibt $>$ `search()` .

Alle eigenen Objekte werden im (Unter-)Verzeichnis `.Data` permanent (**aufräumen!**) gespeichert. Eine Übersicht über die Objekte im Arbeits-Verzeichnis erhält man mit:
$>$ `ls()` oder (etwas allgemeiner) `objects()`

Objekte löschen: $>$ `rm(x,mat2)` .

Objekte anschauen: $>$ `fuel.frame` (*funktioniert, da* `print(...)` *die "Default"-Funktion ist*)

Ort des Objektes: Beispiel $>$ `find(plot)` .

UNIX-Befehle können mit `!` benutzt werden. Beispiel: $>$ `!date` .
Mit der Funktion `unix()` kann auch Input und Output gesteuert werden.

[1] "`CTR-C`" heisst, dass die Taste <u>`Control`</u> gedrückt gehalten und dann dazu <u>`c`</u> gedrückt wird.

2.3 Command-Line-Editor

Es besteht die Möglichkeit, **frühere Eingabezeilen zurückzuholen** und zu editieren, bevor man diese ausführen lässt. Dies spart viel Tipparbeit. Die zur Verfügung stehenden Möglichkeiten sind im Anhang A (Kolonne emacs) ersichtlich.

Es lohnt sich auch, in einem Editor die auszuführenden Befehle zu schreiben und mit Copy-Paste ins S-plus zu kopieren. Dadurch hat man am Ende einer Session ein Log-File, das zu einem späteren Zeitpunkt als Gedächnisstütze dienen kann.

3 Daten einlesen und anschauen

Falls die Daten als Rechtecksmatrix (ASCII-Files) vorliegen und in der ersten Zeile des Files die Kolonnennamen stehen, können Sie auch den folgenden Befehl benützen :

```
oxygen ← read.table(file='~sfsadm/ueb.datasets/oxygen.dat'
, header=T)
```

In komplizierteren Fällen sind die Funktionen `scan`, `make.fields`, und `count.fields` hilfsreich.

Daten auf ein File schreiben: `write`, `sink`, `cat`.

4 S-plus–Objekte

Probieren Sie im folgenden die nach “>   ” angegebenen Befehle aus!

4.1 Zuweisungen

`> x ← 5` oder kürzer
`> x _ 5`
`> 5 -> x` bewirken dasselbe.

Normalerweise wird ← benutzt.

4.2 Datentypen

- **Vektoren:**

```
> y ← c(11,2,15,31)                    # Zuweisung
> y                                    # Object y anschauen
> Monat ← c('jan', 'feb', 'mai')
> y[3]; Monat[1]                       # Indizieren
```

  Bemerkung: `c` steht für **c**ombine.

- **Matrizen:**

```
> mat ← cbind( c(1,3,7), c(10,3,1) ); mat       # Spaltenweise
> mat2 ← rbind( c(1,3,7), c(10,3,1) ); mat2     # Zeilenweise
> mat[1,2]; mat2[1,2]          # Das Element in der 1. Zeile und 2. Spalte
```



```
> mat3 ← matrix(1:15, ncol=3); mat3
> mat3[2,]                             # die zweite Zeile
> mat3[,3]                             # die dritte Spalte
```

- **Arrays:**

```
> ar ← array(1:24, dim=c(3,4,2))       # mehrdimensionale "Matrix"
> ar[2,2,1]                            # ein Element daraus
```

- **Listen:**

  Diese können verschiedene Datentypen enthalten:

```
> l1 ← list(5, 'text', mat)
> kurs ← list(Vorname=c('Rolf','Anna','Urs'), Kurs=1:5, Noten=mat)
```

  Der Zugriff erfolgt mit doppelten eckigen Klammern oder mit Komponenten-Namen:

```
> l1[[2]]
[1] 'text'
> kurs$Vorname
[1] 'Rolf' 'Anna' 'Urs'
```

- **Data-Frames:**

  sind ähnlich wie Matrizen, können aber numerische und `character` Kolonnen enthalten. Dieser Datentyp ist speziell für die Modellierung geeignet. Beispiel:

  `> fuel.frame`

  Umwandlung einer Matrix in ein Data-Frame:
  `> mat.df ← data.frame(mat) ,` oder umgekehrt:
  `> Mat ← data.matrix(mat.df)`

  Bemerkung:
  `> attach(fuel.frame)` ‘hängt’ den Datensatz an, d.h. die Kolonnen können wie Vektoren mit ihren Namen aufgerufen und benutzt werden.

  Für einen ersten Überblick dienen die Funktionen `summary`, `stem`, `plot` oder `pairs`.

  `> detach('fuel.frame')` ‘hängt’ den Datensatz wieder ab.

Missing Values:

`NA`: steht für ‘**N**ot **A**vailable’ (fehlender Wert, “missing value”),
`NULL`: bezeichnet ein leeres Objekt

4.3 Operatoren

Werden diese (und weitere) Grundoperationen auf Vektoren (Matrizen, usw.) angewandt, so wird **komponentenweise** gerechnet.

- $+, -, *, /$ wie üblich,
  ˆ für Exponenten, z.B. `5^2` ist ‘5 hoch 2’ ($= 5^2 = 25$).
- Vergleichende Operatoren:
  $>, >=, <, <=,$ $==$ (gleich), $!=$ ($\neq$, ungleich)

- Logische Operatoren:
  & | ! für 'und', 'oder', 'nicht'

: *from*:*to* ergibt eine ("arithmetische") Folge von Zahlen (s. auch `?seq`).

```
> x ← 5:8; x
  [1] 5 6 7 8
> x+y
  [1] 16 8 22 39
> x>=6
  [1] F T T T                                  # T=true, F=false
> (0:9)^3
  [1]   0   1   8  27  64 125 216 343 512 729
```

4.4 Ausdrücke

```
> 5*(x-7)^3
  [1] -40 -5 0 5                               # Komponenteweise!!
> z ← log(x - 4); z
  [1] 0.000 0.693 1.098 1.386
> sum(x*z)        # multipliziert die Vektoren komponentenweise und summiert diese Werte
  [1] 22.939
```

4.5 Indizieren

```
> y[1:3]                                       # y1, y2, y3
> x[-1]                                        # alle ohne erstes Element
> x[x<7]
  [1] 5 6
> mat[c(1,3),2]                                # 1. und 3. Element der 2. Spalte
[1] 10 3
```

4.6 Eingebaute S-plus-Funktionen

Eine Funktion ist ein S-plus-Ausdruck, der einen Wert zurückliefert. Üblicherweise geschieht dies nach der Ausführung von einer oder mehreren Operationen mit einem oder mehreren Argumenten. S-plus enthält über 2000 verschiedene Funktionen (Übersicht im `help.start` Fenster), die in verschiedenen Verzeichnissen gespeichert sind (Liste mit `search()`).

\> `ls` (ohne Klammern !!) zeigt die Funktion 'ls'
\> `ls()` führt die Funktion aus.
\> `args(ls)` liefert die Argumente von 'ls'
\> `args(lm)` liefert die Argumente von 'lm'

Funktionen können zwei Arten von Argumenten haben: verlangte und freiwillige; für die letzteren ist jeweils ein Weglass-Wert (default) vorhanden. Die Übergabe von Argumenten kann durch die Position oder durch Schlüsselwörter erfolgen. Die Funktion `args` liefert die Argumente von Funktionen.

Beispiel:
\> `win.graph()` *# Grafik-Treiber starten*
\> `plot(x,y)` *# zeichnet die Argumente x und y gegeneinander auf*
\> `lang<- 1:100; breit= sqrt(lang)`
\> `plot(x=lang,y=breit, col=2)` zeichnet die Variablen `lang` und `breit` mit der Farbe 2 gegeneinander auf. Falls ein Argument an der richtigen Position steht, muss der Name nicht ausgeschrieben werden (oben `plot(lang,breit, col=2)`).
\> `args(mean)` liefert alle Argumente von `mean`.

Üblicherweise liefern die Funktionen ein Resultat. Es ist oft sehr wichtig, das Resultat abzuspeichern, d.h. einem neuen Objekt zuzuordnen:
\> `x.log ← log(x)` .

Eine **Liste** von sehr nützlichen und häufig gebrauchten Funktionen ist im Anhang B dieser Anleitung angegeben.

Initialisierungsfunktion:
Wenn eine Funktion `.First` existiert, wird sie beim Beginn der S-plus-Session ausgeführt; man kann damit z.B. bequem Optionen setzen
(vgl. `help(options)`).

5 Programmieren in S-plus

5.1 Schleifen etc.

Es gibt die Möglichkeit, Schleifen mit `for`, `while`, `repeat` zu programmieren; in den meisten Fällen können sie mit einer matrix-orientierten Programmierweise vermieden werden. Schleifen sind unter Umständen sehr langsam!

\> `for(i in` *was*`)` *function*`(w)`

Bedingungen:
\> `if (`*Bedingung*`)` *Ausdruck1* `else` *Ausdruck2*

5.2 Eigene Funktionen schreiben

Für die Erstellung von eigenen `functions` — sehr empfohlen zur Abkürzung mehrerer oder längerer Befehle — empfiehlt sich das Arbeiten mit einem Editor für S-plus-Objekte: Wenn Sie, z.B. `e(Gsqrt)` tippen, können Sie die Definition der Funktion `Gsqrt`() schreiben und verändern. Etwas mühsamer ist es mit den eingebauten S-Funktionen `ed`(*function*) bzw. `vi`(*function*): Sie haben als Resultat die neue edierte *function*. Daher müssen sie *immer* etwas zugewiesen werden, z.B. `sim ← e(sim)`. Auch nach einem Fehler (nach Editieren): `sim ← e()`, sonst sind alle Änderungen verloren!

Die Syntax einer S-Funktion ist:
\> `myfun ← function(` argumente `) {`
  ... Ausdrücke ...
}

Die Beispiele der Help-Files können manchmal als Vorlage für eigene Funktionen dienen. Das Beispiel kann mit der Maus 'ausgeschnitten' und in die Notiz-Datei oder in die S-plus-Session hineinkopiert werden.

Ordnung:
Eine Möglichkeit ist, die Daten, neue Variablen und die eigenen Funktionen in separaten Verzeichnissen abzulegen. Z.B.:
UNIX: `mkdir SFunkt`
S-plus: `> attach('SFunkt')` usw.

Liste der 'angehängten' Verzeichnisse: `> search()`

Arbeitsablauf:
Interaktiv arbeiten im Arbeitsverzeichnis `.Data`, bis eine Funktion oder Auswertung grob erarbeitet worden ist. Diese wird dann in `SFunkt` gespeichert und das Arbeitsverzeichnis `.Data` kann wieder aufgeräumt werden.

5.3 Kleinere Simulationen

Für (einfache und kleinere(!)) ***Simulationen*** mit S-plus sei folgendes Vorgehen empfohlen:

i) Schreibe eine `function`, z.B. `'sim'`, welche als Resultat die Statistik(en) *einer* Simulation ergibt.
ii) Initialisiere einen Resultat-Vektor, z.B. mit `resv <- 1:300`.
iii) Simulation: `for (i in 1:300) { resv[i] <- sim(...) ; ...}`
iv) Verarbeitung der Resultate, z.B. `stem(resv)`.

6 Graphik

6.1 Graphik-Treiber

Bevor Graphiken gezeichnet werden können, muss der richtige Graphik-Treiber gestartet werden. Für den Sun-Monitor geschieht dies mit den Funktionen `motif()`, `openlook()` oder `X11()`. Hardcopies (Ausdruck auf Papier) kann man dann mit dem "Print" Knopf oben im Graphik-Fenster erzeugen. Für (qualitativ besseren) Druck ist `postscript` empfohlen, damit wird eine "encapsulated PostScript" Datei erzeugt, welche dann weiterverarbeitet, z.B. ausgedruckt werden kann.

Es können mehrere Treiber gleichzeitig aktiv sein.
`dev.list()` gibt eine Übersicht der aktiven Treiber.
`dev.set` schaltet auf den gewünschten Treiber um, `dev.off()` schaltet aus.

Beispiel für `postscript(.)`:

```
> postscript('my.ps')
> plot(x,y)
...
> dev.off()  #-- schliesst PostScript-Graphik ab; jetzt kann my.ps ausgedruckt werden.
```

6.2 Graphiken zeichnen

`plot(x,y)` zeichnet die Vektoren x und y gegeneinander auf. Mit dem Argument `type='l'` werden die Punkte durch Linienzüge verbunden. Man kann auch Titel, z.B. `main='Dies ist ein Titel'` und andere Beschriftungen anbringen.


```
> u <- 0 : 5; w <- u*u
> plot(u, w, main='Beispiel')
> plot(u, w, type='l')
```

Für schönere Details gibt es noch viele weitere Möglichkeiten, z.B. wird der Linientyp durch `lty` bestimmt, die Strichdicke durch `lwd`, die Grösse der Zeichen durch `cex`. Default ist jeweils 1. Die Graphikparameter wie Grösse, Schrift, Anordnung,... werden mit der Funktion `par` gesetzt. Sie bieten eine weitgehende Kontrolle über die Graphiken.

Zusätzliche Punkte oder Linien zeichnet man mit den Funktionen `points(xneu, yneu)` oder `lines(xneu, yneu)` in eine bestehende Graphik ein.

```
> plot(u, w, type='n')                # zeichnet nur die Achsen
> lines(u, w,lty=3, lwd=4)
> points(1,4,cex=2)                   # macht grössere Symbole
```

- 6 Bilder auf ein Blatt (3 x 2):
  `> par(mfrow=c(3,2))` *#– siehe auch* `?par`, *auch für weitere Varianten.*
- Geraden werden durch
  `abline(h=1, v=c(3,5))` eingezeichnet,
  die Regressionsgerade durch
  `abline(coef(lm.out))` oder einfach `abline(lm.out)`,
  wobei `lm.out` das Resultat-Objekt der Regression ist.
- Mit der Funktion
  `text(x,y,mytext)`
  wird 'mytext' an den Koordinaten x, y aufgezeichnet.
- Punkte können mit
  `identify(x,y)`
  interaktiv mit der Maus identifiziert werden. Die linke Mausetaste identifiziert den nächstgelegenen Punkt, die mittlere Taste schliesst `identify` wieder ab.
- Koordinaten im Plot können mit
  `locator()`
  und Klicken mit der linken Maustaste abgefragt werden. Mit der mittleren Taste werden die Koordinaten der gewählten Punkte zurückgegeben.

Für spezielle oder zusätzliche (Unter-)Titel verwendet man die Funktion `mtext`.
`brush` und `spin` sind Funktionen für die Visualisierung höherdimensionaler Daten.
Weitere univariate Graphiken: `boxplot`, `hist`, `qnorm`
Multivariate Plots: `pairs`, `matplot`, `stars`, `contour`, `persp`, `image`

7 Lineare Regression

Eine Formel (`"formula"`) ist ein S-PLUS Ausdruck, welcher die Form des statistischen Modells angibt. Sie wird folgendermassen geschrieben

$$\texttt{y} \sim \texttt{x1 + x2}$$

und sie bedeutet

$$y = \alpha + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + E\ .$$

Auf der linken Seite des Operators '$\sim$' (Tilde) steht die Zielvariable y, rechts stehen die erklärenden Variablen $x_1, x_2, \ldots$. Die einzelnen erklärenden Variablen werden durch ein '$+$' voneinander getrennt. Eine Formel in der obigen Form schliesst immer automatisch eine Konstante α ("Intercept") mit ein. Möchte man aber die Konstante explizit ausschliessen, so gibt man das mit dem Term `-1` an: `y ~ -1 + x1 + x2`.

Lineare Regressionsmodelle kann man in S-PLUS mit der Funktion

```
lm(formula, data)
```

anpassen. `data` bezeichnet das Data Frame, in dem die Variablen gegeben sind. Mit `formula` (vgl. oben) wird das Modell beschrieben. Das S-Plus-Objekt `lm` enthält alle notwendige Informationen um eine Regressionsanalyse durchzuführen. Deshalb ist es nützlich diese Informationen in eine Variable (z.B. `fit`) abzuspeichern.

Beispiel

```
> fuel.fit <- lm(Fuel ~ Weight + Disp. , data=fuel.frame)
> coef(fuel.fit)                          # Koeffizienten
> summary(fuel.fit)                       # Konfidenzintervalle, Tests, ...
> summary.aov(fuel.fit)                   # ANOVA-Tabelle
> fitted(fuel.fit)                        # angepasste Werten
> resid(fuel.fit)                         # Residuen
> par(mfrow=c(2,3))                       # Mehrere Graphiken ins gleiche Fenster
> plot(fuel.fit)                          # Residuenplots
> new.data <- data.frame(Weight=3000, Disp.=300)
> predict(fuel.fit, new.data)    # Vorhersage an der Stelle Weight=3000, Disp.=300
```

Modellwahl: `stepwise`

7.1 Weitere Modelle

Varianzanalyse: analog, mit `aov` statt `lm`.

Generalized Linear Models: `glm`.

Nichtparametrische Glättungsmethoden: `smooth.spline`,
multiple: "Generalized Additive Models" : `gam` und "Local Regression Models" `loess`, z.B. `loess(Fuel ~ Weight + Disp.)`, oder `gam(y ~ x + s(x2) + lo(x3))`

Zeitreihen, Survival Analysis, Multivariate Statistik, u.v.a.m.

8 Dokumentation

User Group

Es gibt eine E-mail Liste für Leute, die S-plus benutzen. Für eine Einschreibung schickt man seine E-mail-Adresse an:

`S-news-request@utstat.toronto.edu`

Man kann eigene Beiträge schicken an:

`S-news@utstat.toronto.edu`

Fragen, die immer wieder gestellt werden, wurden gesammelt, beantwortet und via "World Wide

Web" (WWW) veröffentlicht:

`http://www.stat.math.ethz.ch/S-FAQ`

Es gibt ein grosses Archiv "Statlib" für nützliche Funktionen (oft für modernste statistische Verfahren), die von Benutzern geschrieben worden sind. Ein E-mail-Programm schickt die angeforderten Funktionen direkt zu. Für ein Verzeichnis aller vorhandenen Funktionen schickt man die folgende Zeile

`Subject: send index from S`

an:

`statlib@lib.stat.cmu.edu`

Normalerweis einfacher ist der Zugriff auf die "Statlib" im "World Wide Web" (WWW), via den Locator `http://lib.stat.cmu.edu/`. Es gibt dann u.a. Einträge **S** und **s-news**.

"Kostenlose" Unterlagen

"Introductory Guide to S-Plus" von Brian Ripley und Bill Venables kann man (für Unterrichtszwecke) gratis beziehen als PostScript-File (via Statlib (s.o.) "`send sguide from S`"). 68 Seiten, Konzepte mit Beispielen, auch graphischen.

An der epfl wurde eine 51-seitige Einführung in S-plus in Französisch geschrieben. Diese kann ebenfalls als PostScript-File (via Statlib (s.o.) "`send french.intro from S`") bezogen werden.

8.1 Bücher

** *W. Venables und B. Ripley (1997): "Modern Applied Statistics with S-Plus", Springer-Verlag, 548 S.,* (ETH-BIB 772 112) führen in S-plus ein und zugleich in eine zeitgemässe Datenanalyse mit modernen statistischen Methoden.

** Eine detaillierte Einführung gibt *Phil Spector (1994): "An introduction to S and S-plus", Duxbury Press, 286 S.* (ETH-BIB 763 237)

* Mit weniger Details gibt *Brian S. Everitt (1994): "A Handbook of Statistical Analyses using S-Plus", Chapman and Hall, 143 S., ISBN 0-412-56310-X,* eine knappe Einführung und dann 11 schöne Beispiele von Datenanalysen mit S-plus.

* Eine detaillierte Beschreibung von S-plus findet sich in den nachfolgenden Büchern, deren Autoren sich massgeblich an der Entwicklung von S-plus beteiligt haben.
Eine Einführung in S-plus bietet: *Becker A., Chambers J., Wilks A. (1988): "The New S Language", Wadsworth & Brooks Pacific Grove California* (ETH-BIB 749 663).
In *Chambers J., Hastie T. (1991): "Statistical Models in S", Wadsworth & Brooks Pacific Grove California* (ETH-BIB 755 142), werden die Möglichkeiten der statistischen Modellierung mit S-plus aufgezeigt.

– Eine sehr (zu) detaillierte Einführung gibt *Benno Süselbeck (1993): "S und S-Plus; Eine Einführung in Programmierung und Anwendung", Gustav Fischer Verlag, Stuttgart* (ETH-BIB 757 866).

A Command-Line-Editor

Command.edit	Command Line Editing in S-PLUS	**Command.edit**

DESCRIPTION

Command line editing allows you to recall and edit previously issued S-PLUS commands.

USAGE `Splus -e`

DETAILS The editor can do either emacs- or vi-style editing. To determine the editing style, it uses the first valid value in the following list of environment variables, where valid means the value ends with "vi" or with "macs":

```
$S_CLEDITOR
$VISUAL
$EDITOR
```

The default, if none is valid, is vi style.
The most useful editing commands are summarized in the following table:

COMMAND	**EMACS**	VI
backward character	**Ctrl-B**	Esc, h
forward character	**Ctrl-F**	Esc, l
previous line	**Ctrl-P**	Esc, k
next line	**Ctrl-N**	Esc, j
beginning of line	**Ctrl-A**	Esc,^(Shift-6)
end of line	**Ctrl-E**	Esc, $ (Shift-4)
forward word	**Esc, f**	Esc, w
backward word	**Esc, b**	Esc, b
kill char	**Ctrl-D**	Esc, x
kill line	**Ctrl-K**	Esc, Shift-d
delete word	**Esc, d**	Esc, dw
search backward	**Ctrl-R**	Esc, ?
yank	**Ctrl-Y**	Esc, Shift-y
transpose chars	**Ctrl-T**	Esc, xp

REFERENCES

See the section "Command Line Editing" in the chapter *"Getting Started"* in the *S-PLUS User's Manual*.

SEE ALSO `history`, `vi`.

B Grundlegende S-plus Funktionen

Befehl	Erklärung	Beispiel
`help.start()` `help(`*function*`)` `?`*function*	Hilfe über die angegebene `function`	`?plot` `help(Syntax) #` S-plus-*Syntax*
`args(`*function*`)`	Nur Liste der ARGumente	`args(matrix)`
`objects(`*where = 1, frame = NULL, pattern*`)`	Liste von (erzeugten) Var. und Funkt.	`objects() #` *Alle Objekte*
`find(`*object*`)`	findet Objekt in der Suchliste	`find(plot)`
`search()`	gibt die Suchliste an	`search()`
`rm(`*var1,var2,...*`)`	Löschen von Benützer-Objekten	`rm(i1,i2,gugus)`
`c(`*vec1,vec2,...*`)`	Zusammenhängen von Vektoren	`c(-1,0:4,10)`
`rep(`*x,times,length*`)`	Werte von x 'repetieren'	`rep(1:2,5)`
`seq(`*from,to,by,length*`)` $from:to$	arithm. Folge; Verallg. von ':'	`seq(-pi,pi,length=100)` `-2:3 == -2 -1 0 1 2 3`
`rev(`*vec*`)`	Vektor umkehren: `rev(2:4) == 4:2`	`rev(1:4)`
`length(`*vec*`)` `dim(`*mat*`)`	Länge / Dimension von *vec / mat*	`length(3:7) == 5`
`matrix(`*data*`=NA,`*nrow,ncol, byrow*`=F,` *dimnames*`)`	Matrix erzeugen, default: spaltenweise erzeugt	`matrix(1:2,2,3)` `#` 2×3 *(1 2 1, 2 1 2)*
`min(...)`, `max(...)`, `range(...)`	Minimum, Maximum und beide	`range(3:10) == c(3,10)`
`gamma(`*x*`)`, `exp(`*x*`)`, `sqrt(`*x*`)`	$\Gamma(x)$, e^x und $\sqrt{x}$	`gamma(1:10) == 1 1 2 6 ..`
`log(`*x,base*`=exp(1))`	log(x) zur Basis $base$	`log(exp(x)) == x`
`cos(`*x*`)`, `sin`, `tan`, `acos`, ..., `cosh`, `atanh`,...	Die üblichen elementaren Funktionen	`sin(pi/2 * -8:8)`
`sum(`*x*`)`, `prod(`*x*`)`	Summe, bzw. Produkt aller `x[i]`	`prod(1:10) #` *ist 10! (10 Fakultät)*
`any(...)`, `all(...)`	Logische "Summe" und "Product"	`if (any(x<=0)) ...`
`cumsum(`*x*`)`	KUMulative Summe: `cumsum(x)[n] =` $\sum_{i=1}^{n} x[i]$	`cumsum( 1/(1:20)^2 )`
`median(`*x*`)`	Median von `x`	`median( 2^(0:10) )`
`mean(`*x, trim=0*`)`	(Gestutztes) arithm. Mittel, `trim=` 0.5 ist Median	`mean(c(-1000,0:20),trim=0.1)`
`var(`*x,y*`)`, `cor(`*x,y, trim=0*`)`	Varianz- und Korrelations (-Matrix)	`var(1:10)`
`sort(`*data*`)`	Sortiert $data$ aufsteigend	`sort(rexp(30))`
`runif(`*n, min=0, max=1*`)`	n uniform-verteilte Pseudo-Zufallszahlen	`runif(10,max=7) #` *auf [0,7]*
`rexp(`*n*`)`	Vektor von n $exp(1)$-verteilten ZV.	`rexp(20); rexp`
`cut(`*x,breaks,labels*`)`	Kategorische Var. aus 'stetigem' x	`cut(rexp(100),c(0,2^0:4))`
`category(`*x,levels=..,labels=..*`)`	Erzeugt 'Kategorien' Variablen für `table`!	`category(rpoisson(25,la=4)`
`split(`*data, group*`)`	Datenvektor in Gruppen unterteilen	`split(larven[,3],larven[,6`
`table(`*categ1 [,categ2]*`)`	Kontingenz-Tabelle	siehe `help(table)`
`stem(`*x,nl,scale,..., depth=F*`)`	'Stem-and-Leaf' 'Histogramm' (ASCII)	`stem(runif(50))`
`sink(`*file*`)`	Schickt allen S-plus-Output auf $file$, `sink()` macht wieder rückgängig	`sink("out.1")`
'!' `unix(`*command,input,output*`=T)`	UNIX Befehle von S-plus aus	`!ls out*`
`q()`	Verlassen von S-plus	`q()`

Graphiken, etc.		
Befehl	Erklärung	Beispiel
`motif()` `win.graph()`	(sehr viele!) Graphik-Treiber	
`postscript(`*file*`)`	PostScript Ausgabe der Graphik auf Datei *file*; mit UNIX `psvpp` ausdrucken	`postscript() #` *`geht direkt zum Printer`*
`dev.off()`	Graphik-Fenster schliessen	`dev.off()`
`plot(`*x, y, type*`="p",` *log*`=" ",` *main*`= "..",` *xlab*`= ,` *ylab*`= )`	Streu- oder Linien-Diagramm der Punkte (x[i],y[i])	`plot( sin(1:30/pi) ) #` *`x weggelassen`*
`hist(`*x,nclass,breaks,...*`)`	Histogramm von x mit $nclass$ Klassen	`hist(runif(150),nc=12)`
`pairs(`*mat*`)`	"Scatterplot Matrix" der Spalten von `mat`	`pairs(d.auto)`
`mtext(`*text, side=3, line=0*`)`	Text in den Rändern der Figur	`mtext("x-Achse",1,3)`
`text(`*x,y, text,..*`)`	Buchstaben und div. Symbole plotten	`plot(x,y,type='n');text(x,y)`
`par(...)`	Setzen und Abfragen von Graphik-Parametern	`par("usr") #` *`Koordinaten der Eckpunkte`*
`boxplot(`*x,range,width,..*`)`	'Kisten-Diagramm' von x	`boxplot(rexp(50))`
`brush(`*mat,..*`)`	wie `pairs(.)` + dynamisches Markieren	`brush(d.auto)`
`image(`*x,y,z, add*`=F,..)`	3. Dimension mittels "Schattierung"	siehe `?image`