

Statistik mit R

Erwin Grüner

FB Psychologie Uni Marburg

16.02.2006

- Dateisystemfunktionen
- Funktionen für Datum und Zeit
- Systemfunktionen
- Daten und Objekte
- Pakete (Packages)
- Der R Interpreter
- Behandlung von fehlenden Werten in R

Dateisystemfunktionen

Funktionen für Datum und
Zeit

Systemfunktionen

Daten und Objekte

Packages

Der R Interpreter

Fehlende Werte

Funktion	Wirkung
<code>dir.create</code>	Verzeichnis kreieren
<code>file.create</code>	Datei(en) kreieren
<code>file.exists</code>	Test, ob Datei(en) existieren
<code>file.info</code>	Dateiinformationen
<code>file.access</code>	Informationen über Zugriffsrechte
<code>list.files</code> , <code>dir</code> <code>system.file</code>	Liste der Dateien in einem Verzeichnis vollständiger Pfad eines Packages bzw. Files
<code>file.rename</code>	Datei umbenennen
<code>file.append</code>	Datei(en) anhängen
<code>file.copy</code>	Datei(en) kopieren
<code>file.symlink</code>	symbolischen Link erzeugen
<code>file.remove</code>	Datei(en) löschen

Aufruf und weitere Informationen: siehe Online-Hilfe.

Funktion	Wirkung
<code>Sys.time</code>	Datum und Uhrzeit incl. Zeitzone
<code>Sys.timezone</code>	Zeitzone
<code>date</code>	Datum
<code>system.time</code>	Zeitbedarf für die Auswertung eines Ausdrucks
<code>proc.time</code>	Zeitverbrauch für den laufenden R-Prozess

Aufruf und weitere Informationen: siehe Online-Hilfe.

Funktion	Wirkung
<code>system</code>	Aufruf eines Betriebssystemkommandos
<code>shell</code>	Kommando unter einer Shell ausführen
<code>shell.exec</code>	Übergabe einer Datei an die mit ihr verknüpfte Applikation

Aufruf und weitere Informationen: siehe Online-Hilfe.

Verfügbare Datensätze

- Eingebaute Datensätze
- Datensätze in Packages

Zur Zeit werden 4 Datenformate unterstützt:

- Dateien mit der Extension `.R` oder `.r` können mit der Funktion `source()` eingelesen und interpretiert werden; das Arbeitsverzeichnis wechselt vorübergehend in das Verzeichnis dieser Datei.
- Dateien mit der Extension `.RData` oder `.rda` werden mit der Funktion `load()` eingelesen.
- Dateien mit der Extension `.tab`, `.txt` oder `.TXT` werden mit Hilfe der Funktion `read.table()` eingelesen; das Ergebnis ist ein Dataframe.
- Dateien mit der Extension `.csv` oder `.CSV` werden ebenfalls mit Hilfe der Funktion `read.table()` eingelesen und zwar mit folgendem Aufruf:
`read.table(...,header=TRUE,sep=";")`.

- Das Paket `foreign` enthält Funktionen, um Datendateien von anderen Statistiksystem (S, SAS, SPSS, Stata, Systat, Minitab, dBase, u.a.) zu lesen .
- Andere Pakete erlauben es z.B., Daten aus SQL- bzw. ODBC-Datenbanken einzulesen.

Funktion	Wirkung
ls, objects ls.str	Objekte einer Umgebung auflisten detaillierte Informationen über Objekte einer Umgebung
apropos	Objekte auflisten, deren Namen ein an- gegebenes Muster enthalten
find	Pakete auflisten, die angegebene Objekte enthalten
rm, remove	Objekte löschen

Im Zusammenhang mit der Beschreibung der Arbeitsweise des Interpreters wird auf das Thema Umgebung näher eingegangen werden.

Funktion	Wirkung
<code>search</code>	Liste der Objekte und Packages im Suchpfad
<code>searchpaths</code>	Liste der Objekte und Packages (mit Pfad) im Suchpfad
<code>attach</code>	Dataframe in den Suchpfad aufnehmen
<code>detach</code>	Dataframe aus dem Suchpfad entfernen

Zweck: Objekte einer Umgebung auflisten

Aufruf der Funktion (mit Voreinstellungen)

```
ls(name,  
    pos = -1,  
    envir = as.environment(pos),  
    all.names = FALSE,  
    pattern)
```

Parameter	Bedeutung
<code>name</code>	String: Name der Umgebung, deren Objekte aufgelistet werden sollen
<code>pos</code>	Alternative zu <code>name</code> : Position(en) in der Suchliste (-1: die aktuelle Umgebung)
<code>envir</code>	Umgebung: Alternative zu <code>name</code>
<code>all.names</code>	T: auch Namen, die mit einem Punkt beginnen, auflisten
<code>pattern</code>	regulärer Ausdruck: nur Namen auflisten, die damit übereinstimmen

Hinweise

- Beim Parameter `name` muss der Paketname in voller Länge angegeben werden (z.B. `"package:ctest"`).
- `pos` kann auch ein Integervektor sein.
- Mit leerem Argument listet `ls()` den Inhalt der globalen Umgebung auf.
- Die Funktion `objects()` hat die gleichen Parameter wie `ls()` und die gleiche Wirkung.

Zweck: Objekte auflisten, deren Namen ein angegebenes Muster enthalten

Aufruf der Funktion (mit Voreinstellungen)

```
apropos(what,  
where = FALSE,  
mode = "any")
```

Parameter	Bedeutung
<code>what</code>	Name eines Objekts oder regulärer Ausdruck
<code>where</code>	T: auch Position in der Suchliste ausgeben
<code>mode</code>	Typ ('mode') der Objekte, die ausgegeben werden sollen

Zweck: Pakete auflisten, die angegebene Objekte enthalten

Aufruf der Funktion (mit Voreinstellungen)

```
find(what,  
mode = "any",  
numeric. = FALSE,  
simple.words = TRUE)
```

Parameter	Bedeutung
<code>what</code>	Name eines Objekts oder regulärer Ausdruck
<code>mode</code>	Typ ('mode') der Objekte, die ausgegeben werden sollen
<code>numeric.</code>	T: auch Position in der Suchliste ausgeben
<code>simple.words</code>	T: <code>what</code> ist direkt der Name des zu suchenden Objekts

Die Funktion `help.search()`

Zweck: Suche im Online-Hilfesystem

Aufruf der Funktion (mit Voreinstellungen)

```
help.search(pattern,  
fields = c("alias", "concept",  
"title"),  
apropos,  
keyword,  
whatis,  
ignore.case = TRUE,  
package = NULL,  
lib.loc = NULL,  
help.db = getOption("help.db"),  
verbose = getOption("verbose"),  
rebuild = FALSE,  
agrep = NULL)
```

Parameter	Bedeutung
pattern fields	Muster für die Suche nach Informationen Angabe, in welchen Felder gesucht werden soll
apropos	Muster für die Suche bei Topics und Namen
keyword	Muster für die Suche bei den Stichwörtern
whatis	Muster für die Suche bei Topics
ignore.case	T: Groß-/Kleinschreibung ignorieren
package	Pakete, in denen gesucht werden soll
lib.loc	Verzeichnisbaum, in dem gesucht werden soll
help.db	Verzeichnispfad für die zu verwendende Hilfe-Datenbank
verbose	T: Informationen über den Suchprozess
rebuild	T: Hilfe-Datenbank neu aufbauen
agrep	NULL: Fuzzy-Suche

Zweck: Objekte löschen

Aufruf der Funktion (mit Voreinstellungen)

```
remove(...,  
list = character(0),  
pos = -1,  
envir = as.environment(pos),  
inherits = FALSE)
```

Parameter	Bedeutung
...	Objekte, die gelöscht werden sollen
list	Stringvektor mit den Namen der zu löschen- den Objekte
pos	Umgebung, in der Objekte gelöscht werden sollen (-1: die aktuelle Umgebung)
envir	Umgebung, die verwendet werden soll
inherits	T: übergeordnete Umgebungen ebenfalls durchsuchen

Anstatt `remove` kann man die Funktionsnamen mit `rm` abkürzen.

Zweck: Datensätze laden bzw. verfügbare Datensätze auflisten

Aufruf der Funktion (mit Voreinstellungen)

```
data(...,  
list = character(0),  
package = .packages(),  
lib.loc = NULL,  
verbose = getOption("verbose"),  
envir = .GlobalEnv)
```

Parameter	Bedeutung
...	Daten, die geladen werden sollen (Folge von Namen oder Strings)
list	Daten, die geladen werden sollen (Stringvektor)
package	Paket, in dem gesucht werden soll
lib.loc	Verzeichnisbaum, in dem gesucht werden soll
verbose	T: Informationen über den Suchprozess
envir	Umgebung, in der die Daten geladen werden sollen

Hinweise

- Sind keine Datensätze spezifiziert, so werden die verfügbaren Datensätze aufgelistet.
- Ist `lib.loc` nicht angegeben, so wird in den geladenen Paketen gesucht.
- Mit `data(package=NULL)` wird das Data-Verzeichnis des aktuellen Abrietsverzeichnis verwendet.

Die Funktion `attach()`

Zweck: Dataframe in den Suchpfad aufnehmen

Aufruf der Funktion (mit Voreinstellungen)

```
attach(what,  
       pos = 2,  
       name = deparse(substitute(what)))
```

Parameter	Bedeutung
<code>what</code>	Dataframe, Liste oder R-Datendatei (kriert mit <code>save()</code>)
<code>pos</code>	Zielposition im Suchpfad
<code>name</code>	Dateiname (alternative Angabe zu <code>what</code>)

Hinweis

- Der Dataframe wird in den Suchpfad eingefügt. Damit können seine Variablen direkt mit ihrem Namen (ohne Nennung des Dataframes) angesprochen werden.

Die Funktion detach()

Zweck: Dataframe aus dem Suchpfad entfernen

Aufruf der Funktion (mit Voreinstellungen)

```
detach(name,  
pos = 2,  
version)
```

Parameter	Bedeutung
name	Objekt (Name, String oder Position), das entfernt werden soll
pos	Position im Suchpfad
version	Version

Hinweis

- Man kann weder das Arbeitsverzeichnis (Position 1) noch das Basispaket aus dem Suchpfad entfernen.

- Ein großer Teil von R liegt in Form von sog. Paketen vor.
- Einige Pakete gehören zur Basisinstallation.
- Im Internet (z.B. CRAN-Archiv) steht eine große Zahl zusätzlicher Pakete zum Download bereit; diese können auf einfachem Wege nachinstalliert werden.
- Ein Paket kann enthalten:
 - Funktionen, die in R geschrieben sind,
 - DLLs (dynamische Bibliotheken) von kompiliertem Code¹,
 - Datensätze

¹Funktionen, die meist in C oder FORTRAN geschrieben sind.

Funktion	Wirkung
<code>library</code>	Paket laden bzw. Informationen zu einem Paket anfordern
<code>require</code>	Paket innerhalb einer Funktion laden
<code>.packages</code>	Namen der geladenen bzw. installierten Pakete auflisten
<code>.libPaths</code>	Wurzelverzeichnispfad von Paketen
...	...

Hinweis

- Neben den aufgeführten Funktionen existiert noch die Stringvariable `.Library`: in ihr ist das voreingestellte Library-Wurzelverzeichnis, d.h. das 'library'-Unterverzeichnis des Programmverzeichnisses von R, gespeichert.

Zweck: Paket laden bzw. Informationen zu einem Paket anfordern

Aufruf der Funktion (mit Voreinstellungen)

```
library(package,  
help,  
pos = 2,  
lib.loc = NULL,  
character.only = FALSE,  
logical.return = FALSE,  
warn.conflicts = TRUE,  
keep.source = getOption("keep.source.pkgs"),  
verbose = getOption("verbose"),  
version)
```

Parameter	Bedeutung
<code>package</code>	Name des Pakets, das geladen werden soll
<code>help</code>	Name des Pakets, zu dem Informationen ausgegeben werden sollen
<code>pos</code>	Einfügeposition im Suchpfad
<code>lib.loc</code>	Dateiverzeichnis, das durchsucht werden soll (Library-Suchpfad)
<code>character.only</code>	T: <code>package</code> bzw. <code>help</code> sind Strings
<code>logical.return</code>	T: Rückgabe von TRUE/FALSE bei Erfolg bzw. Misserfolg
<code>warn.conflicts</code>	T: Warnung bei Konflikten (z.B. wenn Paket schon geladen ist)
<code>keep.source</code>	Funktionen mit gesamtem Quellcode (incl. Kommentare) laden
<code>verbose</code>	T: zusätzliche Meldungen bei Problemen
<code>version</code>	Version, die geladen werden soll

Hinweise

- Geladene Pakete sind nicht Teil des Arbeitsverzeichnisses ('user workspace').
- Ein bereits geladenes Paket wird nicht zum zweiten Mal geladen.
- Bei großen Pakten empfiehlt es sich, `keep.source` auf `FALSE` zu setzen, um Speicherplatz zu sparen.
- Ein geladenes Paket wird mit der Funktion `detach()` wieder entladen. Als Argument ist dabei der volle Name als String² anzugeben. Beispiel:

```
detach("package:foreign")
```

²so wie er bei `search()` ausgegeben wird

Die Funktion `.packages()`

Zweck: Namen der geladenen bzw. installierten Pakete auflisten

Aufruf der Funktion (mit Voreinstellungen)

```
.packages(all.available = FALSE,,  
lib.loc = NULL)
```

Parameter	Bedeutung
<code>all.available</code>	F: die Namen der geladenen Pakete werden (unsichtbar!) zurückgeben T: die Namen aller bei <code>lib.loc</code> gefundenen Pakete werden zurückgegeben
<code>lib.loc</code>	Dateiverzeichnis, das durchsucht werden soll (Library-Suchpfad)

Zweck: Library-Suchpfad, d.h. Wurzelverzeichnispfad der Pakete

Aufruf der Funktion (mit Voreinstellungen)

```
.libPaths(new)
```

Parameter	Bedeutung
<code>new</code>	Stringvektor mit neuen Library-Suchpfaden

Hinweis

- Wird die Funktion mit leerem Argument aufgerufen, so werden die zur Zeit bekannten Library-Suchpfade zurückgegeben.

- R ist eine Interpretersprache.
- Als solche bietet es dem Benutzer eine Reihe von Vorteilen bei der Evaluierung von Ausdrücken³.
- Bei der Evaluierung eines Ausdrucks wird die sog. Umgebung nach den Variablen abgesucht, die in dem Ausdruck auftauchen, um deren Wert zu ermitteln.
- Zusätzlich verwendet R noch den sog. Suchpfad, um Variablen bzw. Funktionen zu finden.

³Beispielsweise können Anweisungen dynamisch zur Laufzeit erzeugt und ausgeführt werden.

- Ein Frame ist die Menge der lokalen Variablen, die beim Aufruf einer Funktion kreiert werden und beim Verlassen der Funktion wieder verschwinden.
- Eine Umgebung ist eine Verschachtelung von Frames bzw. der innerste Frame incl. seiner umschließenden Umgebung.
- Die globale Umgebung ist das Benutzer-Arbeitsverzeichnis. Bei Wertzuweisungen an der Kommandozeile werden Objekte der globalen Umgebung angesprochen.
- Die Umgebung einer Funktion ist jene Umgebung, die zur Zeit der Definition der Funktion aktiv war⁴.
- Umgebungen kann man auch Variablen zuweisen.
- Eine Umgebung kann durch eine Wertezuweisung manipuliert werden.

⁴Man spricht hier von 'lexikalischem Scope'. Daneben gibt es auch 'dynamischen Scope'.

- Der Suchpfad umfasst an erster Stelle die globale Umgebung (`".GlobalEnv"`), dann die geladenen Daten und Pakete und an letzter Stelle das Basispaket (`"package:base"`).
- Die Reihenfolge der einzelnen Einträge im Suchpfad bestimmt die Reihenfolge, in der sie vom Interpreter durchsucht werden.
- Umgebungen werden mit Hilfe der Funktionen `attach()` oder `library()` in den Suchpfad eingefügt.
- Mit Hilfe der Funktion `search()` kann der aktuelle Suchpfad ausgegeben werden.

Funktion	Wirkung
expression eval environment	Objekte vom Typ <code>expression</code> kreieren Ausdruck in einer Umgebung evaluieren Umgebung abfragen, setzen oder kreieren
parse deparse quote	String in einen Ausdruck umwandeln Ausdruck in einen String umwandeln Argument zurückgeben, ohne es zu evaluieren
substitute ...	Substitution von Variablen in einem Ausdruck ...

Zweck: Objekte vom Typ `expression` kreieren

Aufruf der Funktion (mit Voreinstellungen)

```
expression(...)
```

Parameter	Bedeutung
...	syntaktisch korrekte R Ausdrücke

Hinweise

- Die Funktion gibt einen Vektor vom Typ `expression` zurück.
- Der Ausdruck wird nicht evaluiert.
- Die Funktion `expression()` kann auch bei Textausgaben in einer Graphik (Parameter `text`, `legend`, usw.) verwendet werden, um beispielsweise griechische Symbole od.ä. auszugeben.

Zweck: Ausdruck in einer Umgebung evaluieren

Aufruf der Funktion (mit Voreinstellungen)

```
eval(expr,  
      envir = parent.frame(),  
      enclos = if(is.list(envir) || is.pairlist(envir))  
                parent.frame())
```

Parameter	Bedeutung
expr	Ausdruck, der ausgewertet werden soll
envir	Umgebung für die Evaluation
enclos	'enclosure' (weitere Umgebung, wo Objekte gesucht werden sollen)

Hinweise

- Der Ausdruck muss vom Typ `expression` sein.
- Besteht der Ausdruck aus mehreren Teilen⁵, so werden sie der Reihenfolge nach ausgewertet⁶.
- Zurückgegeben wird der Wert des letzten Teilausdrucks.

⁵Die Teilausdrücke müssen wie üblich mit Semikolon getrennt und das Ganze in geschweifte Klammern gesetzt werden.

⁶Beispiel: `eval(xx<-12; xx*2)` ergibt 24.

Die Funktion `parse()`

Zweck: String in einen Ausdruck umwandeln

Aufruf der Funktion (mit Voreinstellungen)

```
parse(file = "",  
      n = NULL,  
      text = NULL,  
      prompt = "?")
```

Parameter	Bedeutung
<code>file</code>	Eingabedatei (Eingabestrom)
<code>n</code>	Anzahl der Anweisungen, die verarbeitet werden sollen (-1: alle)
<code>text</code>	Stringvektor (jedes Element wird als eine Eingabezeile interpretiert)
<code>prompt</code>	Prompt, wenn die zu parsenden Anweisungen von der Konsole kommen sollen

Funktion	Wirkung
<code>is.na</code>	auf NA testen bzw. NA ersetzen
<code>complete.cases</code>	Fälle auf fehlende Werte prüfen
<code>missing</code>	Prüfung, ob ein Funktionsargument beim Aufruf fehlte
<code>na.action</code>	Maßnahme für die Behandlung von fehlenden Werten; dafür gibt es eine Reihe von Funktionen: <code>na.fail()</code> <code>na.omit()</code> <code>na.exclude()</code> <code>na.pass()</code>
...	...

Zweck: Prüfung auf NA, Ersetzen von NAs

Aufruf der Funktion (2 Formen)

```
is.na(x)
```

```
is.na(x) <- value
```

Parameter	Bedeutung
x	Objekt
value	Wert

Hinweise

- In der ersten Form wird das Objekt auf NA getestet.
- Die Funktion `is.na()` liefert auch TRUE, wenn das Argument NaN (Not-a-Number) ist, d.h. die Funktion `is.na()` differenziert nicht zwischen NA und NaN; für die Prüfung auf NaN existiert die Funktion `is.nan()`.
- In der zweiten Form wird beim Objekt `x` ein NA durch den angegebenen Wert ersetzt.

Zweck: Fälle auf fehlende Werte prüfen

Aufruf der Funktion

```
complete.cases(...)
```

Parameter	Bedeutung
...	Eine Folge von Vektoren, Matrizen oder Dataframes

Hinweis

- Die Argumente müssen jeweils die gleiche Länge haben.
- Die Funktion gibt einen logischen Vektor zurück, der angibt, welche Fälle vollständige Daten enthalten.

Zweck: Maßnahme für die Behandlung von fehlenden Werten

Aufruf der Funktion (mit Voreinstellungen)

```
na.action(object,  
...)
```

Parameter	Bedeutung
object	beliebiges Objekt
...	eventuell weitere Argumente für spezielle Methoden

Hinweis

- Die Funktion sollte auf das Objekt immer dann angewandt werden, wenn keine NAs gewünscht sind.

Die Funktionen `na.fail()`, `na.omit()`,
`na.exclude()`, `na.pass()` ...

Zweck: Behandlung von fehlenden Werten, z.B. in einem
Dataframe

Aufruf der Funktionen

```
na.fail(object, ...)
```

```
na.omit(object, ...)
```

```
na.exclude(object, ...)
```

```
na.pass(object, ...)
```

Parameter	Bedeutung
<code>object</code>	Beliebiges Objekt (Vektor, Matrix oder Dataframe)
<code>...</code>	Eventuell weitere Argumente für spezielle Methoden

Hinweise

- `na.fail`: haben alle Fälle vollständige Daten, wird das Objekt zurückgegeben, sonst ein Fehler gemeldet.
- `na.omit`: Fälle mit fehlenden Werten werden entfernt.
- `na.pass`: das Objekt wird unverändert zurückgegeben.
- `na.exclude`: Fälle mit fehlenden Werten werden entfernt; bei `naresid` und `napredict` werden für die entfernten Fälle NAs ausgegeben, um die korrekte Länge zu erhalten.

Zweck: Berücksichtigung von Missing Data bei der Berechnung von Residuen bzw. vorhergesagten Werten

Aufruf der Funktionen

```
naresid(omit, x, ...)
```

```
napredict(omit, x, ...)
```

Parameter	Bedeutung
<code>omit</code>	Objekt, das durch eine <code>na.action</code> -Funktion erzeugt wurde
<code>x</code>	Vektor, Matrix oder Dataframe
<code>...</code>	Weitere Argumente von anderen bzw. für andere Methoden

Hinweis

- Diese Funktionen werden verwendet, um bei der Schätzung von Residuen und angepassten bzw. vorhergesagten Werte (Methoden `'predict'` und `'resid'`) innerhalb von Modellanpassungsfunktionen (`lm()`, `glm()`, ...) das Entfernen von fehlenden Werten zu kompensieren.
- Ist die Aktion `na.exclude`, so werden an den korrekten Positionen jeweils NA's eingefügt, damit man die gleiche Anzahl von Zeilen erhält wie im Original-Dataframe.