Package ‘utils’

February 20, 2017

Version 3.3.2

Priority base

Title The R Utils Package

Author R Core Team and contributors worldwide

Maintainer R Core Team <R-core@r-project.org>

Description R utility functions.

License Part of R 3.3.2

Suggests methods, XML

NeedsCompilation yes

R topics documented:

utils-package . . . 4
adist . . . 4
alarm . . . 6
apropos . . . 6
aregexec . . . 8
aspell . . . 9
aspell-utils . . . 11
available.packages . . . 13
BATCH . . . 15
bibentry . . . 16
browseEnv . . . 20
browseURL . . . 21
browseVignettes . . . 22
bug.report . . . 23
capture.output . . . 26
changedFiles . . . 27
chooseBioCmirror . . . 29
chooseCRANmirror . . . 30
citation . . . 31
cite . . . 33
citEntry . . . 35
close.socket . . . 36
combn . . . 37
compareVersion . . . 38
COMPILE . . . 39

contrib.url . . . 40
count.fields . . . 41
create.post . . . 42
data . . . 43
dataentry . . . 45
debugger . . . 48
demo . . . 50
download.file . . . 51
download.packages . . . 54
edit . . . 56
edit.data.frame . . . 57
example . . . 59
file.edit . . . 61
file_test . . . 62
findLineNum . . . 63
fix . . . 65
flush.console . . . 66
format . . . 66
getAnywhere . . . 67
getFromNamespace . . . 68
getParseData . . . 70
getS3method . . . 72
glob2rx . . . 73
globalVariables . . . 74
head . . . 76
help . . . 77
help.request . . . 80
help.search . . . 81
help.start . . . 84
hsearch-utils . . . 85
INSTALL . . . 86
install.packages . . . 88
installed.packages . . . 93
isS3method . . . 94
LINK . . . 95
localeToCharset . . . 96
ls.str . . . 97
maintainer . . . 99
make.packages.html . . . 100
make.socket . . . 101
memory.size . . . 102
menu . . . 103
methods . . . 104
mirrorAdmin . . . 106
modifyList . . . 106
news . . . 107
nsl . . . 109
object.size . . . 110
package.skeleton . . . 111
packageDescription . . . 113
packageName . . . 114
packageStatus . . . 115

page . . . 117
person . . . 117
PkgUtils . . . 120
process.events . . . 121
prompt . . . 122
promptData . . . 124
promptPackage . . . 125
Question . . . 126
rcompgen . . . 128
read.DIF . . . 133
read.fortran . . . 135
read.fwf . . . 136
read.socket . . . 138
read.table . . . 139
recover . . . 144
relist . . . 146
REMOVE . . . 148
remove.packages . . . 148
removeSource . . . 149
RHOME . . . 150
roman . . . 150
Rprof . . . 151
Rprofmem . . . 153
Rscript . . . 154
RShowDoc . . . 155
RSiteSearch . . . 156
rtags . . . 158
Rtangle . . . 159
RweaveLatex . . . 161
savehistory . . . 165
select.list . . . 166
sessionInfo . . . 167
setRepositories . . . 169
SHLIB . . . 170
sourceutils . . . 171
stack . . . 172
str . . . 174
summaryRprof . . . 177
Sweave . . . 179
SweaveSyntConv . . . 181
tar . . . 182
toLatex . . . 184
txtProgressBar . . . 185
type.convert . . . 186
untar . . . 187
unzip . . . 189
update.packages . . . 191
url.show . . . 193
URLencode . . . 194
utils-deprecated . . . 195
View . . . 195
vignette . . . 196

write.table 197
zip 200

Index **202**

`utils-package` *R の Utils パッケージ*

Description

R のユーティリティ関数

Details

このパッケージはユーティリティ関数の集まりを含む.

完全なリストについては `library(help = "utils")` を使う.

Author(s)

R Core Team と全世界の貢献者

Maintainer: R Core Team <R-core@r-project.org>

`adist` 近似文字列距離

Description

文字列ベクトル間の近似文字列距離を計算する．距離は一般化 Levenshtein (編集) 距離で，一つの文字列を別の文字列に変換するのに必要な最少の(もしかすると重み付きの)挿入，削除，代入の回数を与える.

Usage

```
adist(x, y = NULL, costs = NULL, counts = FALSE, fixed = TRUE,
      partial = !fixed, ignore.case = FALSE, useBytes = FALSE)
```

Arguments

`x`	文字列ベクトル．Long vectors はサポートされていない.
`y`	文字列ベクトル，または `x` として `y` を取ることを指示する `NULL` (既定値).
`costs`	Levenshtein 距離を計算するためのそれぞれの重みを与える ‘`insertions`’, ‘`deletions`’ そして ‘`substitutions`’ に部分的にマッチする名前を持つ数値ベクトルかリスト，もしくは全ての三つの変換に対して単位コストを与える `NULL` (既定値).
`counts`	オプションで返り値の `"counts"` 属性として変換回数 (挿入，削除そして代入の数)を返すべきかどうかを指示する論理値.

fixed	論理値．もし TRUE (既定値)ならば，要素 x はそのままの文字列として使われる．さもなければ，それらは正規表現とされそして partial = TRUE が含意される (fixed = FALSE の agrep で使われる近似文字列距離に対応する).
partial	変換された x 要素が正確に全ての y 要素にマッチすべきか，または単にそれらの部分文字列にマッチすべきかを指示する．後者は agrep (既定値)により使われる近似文字列距離に対応する．
ignore.case	論理値．もし TRUE なら，距離の計算に対して大文字と小文字を区別しない．
useBytes	論理値．もし TRUE なら距離計算は文字列毎ではなくバイト毎に行われる．

Details

二つの文字列 s と t の間の (一般化) Levenshtein (または編集)距離はもしかすると重み付きの s を t に挿入，消去そして代入で変換する (従って変換は正確に t にマッチする)に必要な最少距離である．この距離は partial = FALSE に対して計算され，現在動的計画法 (例えば https://en.wikipedia.org/wiki/Levenshtein_distance を見よ)で計算され，スペースと時間の計算量的複雑さは $O(mn)$ である，ここで m と n はそれぞれ s と t の長さである．更に変換列とカウントの計算は $O(\max(m,n))$ である．

一般化 Levenshtein 距離はまた近似(ファジー)文字列マッチングにも使われ，この場合パターン s (これは正規表現で良く，その場合原則的に最も左側で最長のマッチが適用される) への最少距離を持つ t の部分文字列を探す，例えば https://en.wikipedia.org/wiki/Approximate_string_matching を見よ．この距離は partial = TRUE に対して Ville Laurikari (http://laurikari.net/tre/) による 'tre' を使って計算され，そして agrep で使われる距離に対応する．この場合与えられたコスト値は整数に強制変換される．

挿入と削除に対するコストは異なって良く，その場合 s と t の間の距離は t と s の間の距離と異なる可能性がある．

Value

x と y の要素に対する近似文字列距離を持つ行列で，行と列はそれぞれ x と y に対応する．

もし counts が TRUE ならば，変換カウントはこの行列の "counts" 属性として，次元がそれぞれ x の要素，y の要素，そして変換のタイプ (挿入，消去，代入)に対応する次元を持つ次元配列として返される．加えて，もし partial = FALSE ならば，変換系列が返り値の "trafos" 属性として返される．これはマッチ，挿入，消去そして代入を指示する要素 'M', 'I', 'D' そして 'S' を持つ文字列である．もし partial = TRUE ならば，マッチした部分文字列のオフセット(最初と最後の要素の位置)が返り値の "offsets" 属性として返される (マッチしなければどちらのオフセットも -1).

See Also

Levenshtein 距離を用いた近似的な文字列マッチング(ファジーマッチング) については agrep.

Examples

```
## Cf. https://en.wikipedia.org/wiki/Levenshtein_distance
adist("kitten", "sitting")
## Levenshtein 距離に対する変換カウントを見るには：
drop(attr(adist("kitten", "sitting", counts = TRUE), "counts"))
```

```
## 変換系列を見るには：
attr(adist(c("kitten", "sitting"), counts = TRUE), "trafos")

## Cf. agrep に対する例：
adist("lasy", "1 lazy 2")
## "部分的近似マッチ" (agrep に対して使われる)に対しては：
adist("lasy", "1 lazy 2", partial = TRUE)
```

---

alarm　　ユーザに警告

Description

ユーザに音か視覚のシグナルを与える.

Usage

```
alarm()
```

Details

alarm() はコンソールに "\a" 文字を送ることで動作する. 殆どのコンソールでこれはユーザにベル音, ビープ音, または他のシグナルを与える (標準出力がリダイレクトされていない限り).

これはコンソールをフラッシュしようとする(flush.console を見よ).

Value

有用な値は何も返されない.

Examples

```
alarm()
```

---

apropos　　オブジェクトを(部分)名前で見つける

Description

apropos() は what にマッチする検索リスト中の全てのオブジェクトの名前を与える文字列ベクトルを返す.

find() は同じことを行う異なったユーザ・インタフェイスである.

Usage

```
apropos(what, where = FALSE, ignore.case = TRUE, mode = "any")

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

Arguments

what	オブジェクトの名前を持つ文字列，またはより一般的にマッチされる regular expression.
where, numeric	検索リスト中の位置もまた返されるべきかどうかを指示する論理値.
ignore.case	検索が大・小文字を区別して行われるべきかどうかを指示する論理値で，既定では TRUE. R のバージョン 2.5.0 より前では，既定値は暗黙のうちに ignore.case = FALSE であったことを注意する.
mode	文字列；もし "any" でなければ，その mode が mode に等しい物だけが検索される.
simple.words	論理値；もし TRUE ならば，what 引数は単語全体としてだけ検索される.

Details

もし mode != "any" ならばモードが mode のオブジェクトだけが考慮される．もし where が TRUE ならば，検索リスト中の位置が名前属性として返される.

find は apropos と同じ作業を行う異なったユーザ・インタフェイスである．しかしながら，既定 (simple.words == TRUE) では， grep(fixed = TRUE) を用いて完全な単語が検索される.

Value

apropos に対しては名前でソートされた文字列ベクトルで，もしかすると検索パス上の(数値)位置の名前を持つかもしれない.

find に対しては環境名の文字列ベクトルか，または numeric = TRUE に対しては検索パス上の位置の数値ベクトルで，対応する環境の名前を与える名前を持つ.

Author(s)

Kurt Hornik と Martin Maechler (May 1997).

See Also

ワイルドカードパターンを正規表現に変換するには glob2rx.

一つの場所からのオブジェクトのリスティングには objects，ヘルプシステムを探索するには help.search，検索パスに対しては search.

Examples

```
require(stats)


## Not run: apropos("lm")
apropos("GLM")                      # 1ダース以上
## もし methods が付加されていると
## '.__C__' で始まる内部オブジェクトを含むかもしれない
apropos("GLM", ignore.case = FALSE) # 一つではない
apropos("lq")

cor <- 1:pi
find("cor")                         #> ".GlobalEnv"   "package:stats"
find("cor", numeric = TRUE)                     # これらの名前を持つ数
```

```
find("cor", numeric = TRUE, mode = "function")  # 二番目だけ
rm(cor)

## Not run: apropos(".", mode="list")  # 長いリスト

# 二重のバックスラッシュ '\\' が必要(最早それが見えない場合)
apropos("\\[")

# 全て% not diff-able
length(apropos("."))

# 'pr' で始まるもの
apropos("^pr")

# 1 文字の場合
apropos("^.$")
# 1 か 2 文字の例
apropos("^..?$")
# 2 から 4 文字の例
apropos("^.{2,4}$")

# 8 とそれ以上の文字の場合
apropos("^.{8,}$")
table(nchar(apropos("^.{8,}$")))
```

---

aregexec 近似的な文字列マッチ位置

Description

近似的な文字列マッチ位置を決定する.

Usage

```
aregexec(pattern, text, max.distance = 0.1, costs = NULL,
         ignore.case = FALSE, fixed = FALSE, useBytes = FALSE)
```

Arguments

pattern	空でない文字列か. マッチされる正規表現 (fixed = FALSE に対して)を含む文字列. もし可能なら文字列に as.character で強制変換される.
text	マッチを探索する場所の文字列ベクトル. もし可能なら as.character により文字列ベクトルに強制変換される.
max.distance	マッチに許される最大距離. agrep を見よ.
costs	変換のコスト. agrep を見よ.
ignore.case	論理値. もし TRUE なら, 距離の計算では大・小文字は区別され無い.
fixed	もし TRUE なら, パターンは字句通り (as is)にマッチされる. さもなければ(既定) それは正規表現としてマッチされる.
useBytes	論理値. もし TRUE ならば, 比較は文字単位ではなくバイト単位で行われる.

Details

aregexec は近似文字列マッチングへの agrep とは異なるインタフェイスである (regexec と grep が提供する正確な文字列マッチングへのインタフェイスのやり方で).

既定では agrep は字句通りのマッチを実行する一方， aregexec は正規表現に依るマッチングを実行することを注意する.

近似的文字列マッチングと距離についてのより詳細は agrep と adist を見よ.

もし pattern または text の任意の要素が "bytes" とマークされていれば比較はバイト毎である.

Value

text と同じ長さのリストで，その各要素はマッチしたものがなければ −1 か，マッチの開始位置と pattern の括弧に入った部分表現式に対応する全ての部分文字列で，マッチの長さを与える整数ベクトルの属性 "match.length" を持つ (またはマッチがなければ −1).

See Also

マッチした部分文字列を取り出すには regmatches.

Examples

```
## Cf. agrep に対する例.
x <- c("1 lazy", "1", "1 LAZY")
aregexec("laysy", x, max.distance = 2)
aregexec("(lay)(sy)", x, max.distance = 2)
aregexec("(lay)(sy)", x, max.distance = 2, ignore.case = TRUE)
m <- aregexec("(lay)(sy)", x, max.distance = 2)
regmatches(x, m)
```

---

aspell　　スペルチェック用のインタフェイス

Description

Aspell, Hunspell または Ispell を使って与えられたファイルのスペルチェックをする.

Usage

```
aspell(files, filter, control = list(), encoding = "unknown",
       program = NULL, dictionaries = character())
```

Arguments

files	チェックされるファイルの名前の文字列ベクトル.
filter	スペルチェックの前にファイルを処理するオプションのフィルターで，関数(形式的引数 ifile と encoding を持つ)，又は組み込みのフィルターを指定する文字列，又は組み込みフィルター名とそれに渡される追加引数を持つリスト，で与えられる．利用可能なフィルターについては詳細節を見よ．もし欠損か NULL なら如何なるフィルタリングも行われない.

control　スペルチェッカーに対する制御オプションのリストか文字列ベクトル．

encoding　ファイルのエンコーディング．必要に応じてリサイクルされる．

program　使われるスペルチェックプログラムの名前 (もしシステムパス上にあれば)または完全なパス，又は NULL (既定値)．既定では，システムパスで aspell, hunspell そして ispell (この順で)が探され，最初に見つかったものが使われる．

dictionaries　使用される R レベルの追加の辞書の名前の文字列ベクトルかファイルパス．パス分離子が無い要素は R のシステム辞書を指定する (R のホームディレクトリのサブディレクトリ 'share/dictionaries' 中)．ファイル拡張子(現在 '.rds' だけ)は省略可能である．

Details

使用されるスペルチェックプログラムはコマンドラインオプション '-a' により活性化されるいわゆる Ispell パイプインタフェイスをサポートしなければならない．プログラムに加えて適当な辞書が利用可能である必要がある． Aspell, Hunspell そして(インターナショナル化) Ispell 辞書を入手するにはそれぞれ http://aspell.net, http://hunspell.sourceforge.net/ そして http://lasr.cs.ucla.edu/geoff/ispell.html を見よ．

現在利用可能な組み込みフィルターは "Rd" (RdTextFilter に対応)，"Sweave" (SweaveTeXFilter に対応)，"R", "pot" そして "dcf" である．

フィルター "R" は R コード用で message, warning, stop, packageStartupMessage, gettext, gettextf そして ngettext (最初の五つに対しては名前無しの文字列定数，そして最後の二つに対してはそれぞれ fmt と msg1/msg2 文字列定数) 中のメッセージ文字列定数を取り出す．フィルター "pot" はメッセージ文字列カタログ '.pot' ファイル用である．両者はスペルチェックに対して無視されるメッセージ文字列部分に対する正規表現を許す ignore を持つ：例えば， "[\t]'[^']*'[\t[:punct:]]" の使用は一重引用符内の全てのテキストを無視する．

フィルター "dcf" は Debian 制御ファイル書式中のファイル用である．保持する欄は引数 keep (対応する欄名を持つ文字列ベクトル)で制御できる．既定では 'Title' と 'Description' 欄が保持される．

aspell が返すオブジェクトに対するプリントメソッドはもしかするとミススペルかもしれない単語の位置のインデントを制御する indent 引数を持つ．既定値は 2 である；Emacs ユーザはインデント量 0 を使い grep-mode で出力を閲覧するのが便利かもしれない．これは又 verbose 引数を持つ：もしこれが真ならば，置き換えに対する示唆が同様に提示される．

追加の R レベル辞書を利用することが可能である．現在これらは saveRDS を用いて単語リストの文字列ベクトルをシリアル化することで得られる拡張子 '.rds' を持つファイルである．もしそうした辞書が使われると，それらは単一の単語リストファイルに結合されそれからスペルチェックの個人辞書(オプション '-p')として使うことが出来る：従って，この場合既定の個人辞書は使用されない．

Value

aspell (これは有用なプリントメソッドを持つ)を継承するデータフレームで，可能なミススペル語に関する情報を持つ．

References

Kurt Hornik and Duncan Murdoch (2011), Watch your spelling! *The R Journal* **3**(2), 22–28. http://journal.r-project.org/archive/2011-2/RJournal_2011-2_Hornik+Murdoch.pdf.

See Also

スペルチェック用のパッケージについては aspell-utils.

Aspell ライブラリの精選された R インタフェイスについては Omegahat (http://www.omegahat.net/Aspell) にあるパッケージ **Aspell**.

Examples

```
## Not run:
## ディレクトリ中の Rd ファイル全てをチェックするために,
## (追加で) \references セクションをスキップする.
files <- Sys.glob("*.Rd")
aspell(files, filter = list("Rd", drop = "\\references"))

## 全ての Sweave ファイルをチェックする
files <- Sys.glob(c("*.Rnw", "*.Snw", "*.rnw", "*.snw"))
aspell(files, filter = "Sweave", control = "-t")

## 全ての Texinfo ファイルをチェックする(Aspell だけ)
files <- Sys.glob("*.texi")
aspell(files, control = "--mode=texinfo")

## End(Not run)

## 利用可能な R システムディレクトリをリストする
Sys.glob(file.path(R.home("share"), "dictionaries", "*.rds"))
```

---

aspell-utils　　スペルチェック用のユーティリティ

Description

Aspell, Hunspell または Ispell によるスペルチェックのためのユーティリティ.

Usage

```
aspell_package_Rd_files(dir, drop = c("\\author", "\\references"),
                        control = list(), program = NULL,
                        dictionaries = character())
aspell_package_vignettes(dir,
                         control = list(), program = NULL,
                         dictionaries = character())
aspell_package_R_files(dir, ignore = character(), control = list(),
                       program = NULL, dictionaries = character())
aspell_package_C_files(dir, ignore = character(), control = list(),
                       program = NULL, dictionaries = character())

aspell_write_personal_dictionary_file(x, out, language = "en",
                                      program = NULL)
```

Arguments

dir	パッケージのルートディレクトリへのパスを指定する文字列.
drop	テキストを RdTextFilter 経由で選択した時に除く追加の Rd セクションの名前の文字列ベクトル.
control	スペルチェッカーに対する制御オプションの文字列ベクトルのリスト.
program	使われるスペルチェックプログラムの名前 (システムパス上にある場合)または完全なパスを与える文字列，または NULL (既定値). 既定では，システムパス上で aspell, hunspell そして ispell (この順)で検索し，最初に見つかったものが使われる.
dictionaries	使用される追加の R レベルの辞書の名前かファイルパスの文字列. aspell を見よ.
ignore	メッセージ文字列をフィルタリングする際に空白で置き換えられる正規表現を持つ文字列ベクトル.
x	文字列ベクトル，または aspell() の呼び出しの結果.
out	書き込まれる個人辞書の名前の文字列.
language	Aspell で使われる言語を指示する文字列.

Details

関数 aspell_package_Rd_files, aspell_package_vignettes, aspell_package_R_files そして aspell_package_C_files はルートディレクトリ dir を持つパッケージの Rd ファイル，ビニエット，R ファイル，そして C レベルのメッセージに対するスペルチェックを実行する．それらは対応するファイルを決定し，適当なフィルターを適用し，そしてスペルチェッカーを実行する.

フィルターに関する詳細は aspell を見よ.

C レベルのメッセージ文字列は 'po/*PACKAGE*.pot' メッセージカタログファイルから得られる，ここで *PACKAGE* は dir の基本名である．より多くの情報は "Writing R Extensions" 中の "C-level messages" に関する節を見よ.

Aspell を使う際は，ビニエットのチェックはパラメータと/そして命令 \Sexpr, \citep, \code, \pkg, \proglang そして \samp に対するオプションをスキップする．追加の命令は control 引数に --add-tex-command オプションを加えることでスキップされる．例えば，\mycmd のオプションとパラメータの双方をスキップするには --add-tex-command='mycmd op' を加える.

control, program, dictionaries, drop そして ignore に対する適切な値をまたパッケージの既定ファイルで指定出来る．これらは dir のサブディレクトリ '.aspell' 中の 'defaults.R' であるべきで，適当な名前付きリストの付値により既定値を与える，例えば (Aspell 使用中は)ビニエットに対しては

```
vignettes <- list(control = "--add-tex-command='mycmd op'")
```

そして同様に Rd ファイル，R ファイル，そして C レベルのメッセージの既定値は Rd_files, R_files そして C_files に付値する.

英語と米語双方を使うパッケージの保守管理者は制御オプション '--master=en_US' と '--add-extra-dicts=en_GB' を Aspell に，そして制御オプション '-d en_US,en_GB' を Hunspell に渡すのが便利である(対応する辞書がインストールされている限り).

R のより古いバージョンは R レベルの辞書に対するサポートを持たず，従って受け入れられた単語からの R レベルの (スペルチェック)プログラム固有の個人辞書を作る関数

aspell_write_personal_dictionary_file を提供していた．新しい機構は R レベルの辞書，つまり saveRDS を使いそうした単語の文字列ベクトルをシリアル化した '.rds' ファイルを使う．パッケージの既定の機構を使い指定されたそうした辞書に対しては，パス分離子を持たない要素は R のシステム辞書または '.aspell' サブディレクトリ中の辞書であり得る．

See Also

aspell

---

available.packages　*CRAN* 類似のレポジトリで利用できるパッケージをリストする

Description

available.packages は一つ又はそれ以上のレポジトリで現在利用可能なパッケージに対応する詳細の行列を返す．パッケージの現在のリストがインターネット越しにダウンロードされる(又は局所的なミラーからコピーされる)．

Usage

```
available.packages(contriburl = contrib.url(repos, type), method,
                   fields = NULL, type = getOption("pkgType"),
                   filters = NULL, repos = getOption("repos"))
```

Arguments

contriburl	レポジトリの 'contrib' セクションをの URL．この引数はもし自分のレポジトリのミラーが不完全な時だけ指定する，例えば CD に 'contrib' セクションだけを焼き付けている時．
method	ダウンロード法，download.file を見よ．
type	文字列で，どのようなタイプのパッケージかを指示する：install.packages を見よ． もし type = "both" ならばこれはソースレポジトリを使う．
fields	既定のものに加えて 'PACKAGES' ファイルから取り出す欄を与える文字列ベクトル，又は NULL (既定)．利用できない欄は NA 値になる．
filters	文字列ベクトル，リスト又は NULL (既定値)．'詳細'節を見よ．
repos	文字列ベクトル，使用するレポジトリの基本 URL．

Details

パッケージのリストは局所的なミラー ('file://' URI で指定される) からコピーされるかダウンロードされる．もしダウンロードされるならば，リストは tempdir() 中のレポジトリ別の

```
repos_http%3a%2f%2fcran.r-project.org%2fsrc%2fcontrib.rds
```

の様な名前のファイル中に R セッションに対してキャッシュされる．

既定では，返り値はそのバージョンと OS 要求が実行中の R バージョンに合致するパッケージだけを含み，最新のパッケージに関する情報だけを与える．

引数 filters はレポジトリのどのパッケージを報告するのか選択するのに使うことが出来る．それは install.packages の様な関数により既定値(NULL)で呼び出される：この値は getOption("available_packages_filters") と，もしこれが未設定か NULL に設定されていれば c("R_version", "OS_type", "subarch", "duplicates") に対応する．

組み込みのフィルターは

"R_version" その R バージョン要求が満たされないパッケージを除外する．

"OS_type" その OS 要求がこのバージョンの R と両立しないパッケージを除外する：これは Unix 風プラットフォームでは Windows 専用パッケージを除外し，逆も又真である．

"subarch" バイナリーパッケージに対しては，現在のサブアーキテクチャに対してそのコンパイル済みコードが利用不可の物を除外する，例えば 64 ビット Windows R では 32 ビット Windows 用だけにコンパイルされた物をR では除外する．

"duplicates" 複数のバージョンが利用可能なら最新のバージョンだけを報告し，もしそれが複数のレポジトリ中にあれば最新のバージョンを持つ (contriburl 中の)最初に名前のあるレポジトリだけを報告する．

"license/FOSS" 利用可能なライセンス指定を用いてフリーかオープンソースソフトウェア(FOSS，つまり https://en.wikipedia.org/wiki/FOSS)として検証可能なパッケージだけに基づいてインストールが進行可能なパッケージだけを含む．このようにしてパッケージとロードがそれらに依存するパッケージが共に FOSS であることが既知であることが必要になる．

これはレポジトリがライセンス情報を提供することに依存することを注意する．

"license/restricts_use" そのインストールを制限的使用で無いことが既知のパッケージだけに基づいて進行できるパッケージだけを含む．

"CRAN" 他のレポジトリからのバージョンに優先して CRAN バージョンを使う(例えそれらがより高いバージョン数を持っていても)．これは既定の "duplicates" フィルターの前に適用される必要があり，従って add = TRUE と一緒には使えない．

もし全てのフィルターがこのセットからのものならば，それらは文字列ベクトルとして指定できる；さもなければ filters はその要素が文字列，ユーザ定義の関数又は add = TRUE (下を見よ)であるリストでなければならない．

ユーザ定義のフィルターは available.packages が返すような形式の行列を単一の引数として取る関数で，そして引数の行のサブセットからなる行列を返す．

特殊な‘フィルター’ add = TRUE は既定のフィルターにフィルターリストの他の要素を追加する．

Value

パッケージ毎に一行ッケージで列名は "Package", "Version", "Priority", "Depends", "Imports", "LinkingTo", "Suggests", "Enhances", "File" そして "Repository" を含む．追加の列は fields 引数を用いて指定できる．

レポジトリが提供する場合は，欄 "OS_type", "License", "License_is_FOSS", "License_restricts_use", "Archs", "MD5sum" そして "NeedsCompilation" が，フィルターと install.packages を含むパッケージ管理ツールで使用するために報告される．

See Also

install.packages, download.packages, contrib.url.

レポジトリをどのように設定するかについてはマニュアル 'R Installation and Administration'.

Examples

```
## Not run:
## install.packages() (等)を
## FOSS であることが知られているパッケージに制限する
options(available_packages_filters =
  c("R_version", "OS_type", "subarch", "duplicates", "license/FOSS"))
## 又は
options(available_packages_filters = list(add = TRUE, "license/FOSS"))

## Omegahat, R-Forge 等の開発バージョンではなく, CRAN の
## リリースバージョンを優先する
options(available_packages_filters =
     c("R_version", "OS_type", "subarch", "CRAN", "duplicates"))

## End(Not run)
```

BATCH　　*R* のバッチ実行

Description

R を非対話的に実行し infile からの入力 (stdout/stderr) を別のファイルに送る.

Usage

```
R CMD BATCH [options] infile [outfile]
```

Arguments

infile	実行される R コードを持つファイルの名前.
options	R のコマンドラインオプションのリスト, 例えば利用可能なメモリー量の設定や, ロード/セーブのプロセスを制御する. もし infile が '-' で始まるときは, '--' を最後のオプションとして使う. 既定のオプションは '--restore --save --no-readline'. (Windows では '--no-readline' 無し.)
outfile	出力をそこへ書き込むファイル名. もし与えられないと, 使われる名前は infile のそれで, もしかすると拡張子 '.R' が取り除かれ, '.Rout' が加えられる.

Details

使い方を思い出すには R CMD BATCH --help を使う.

既定では，入力命令が出力と共にプリントされる．この挙動を抑制するには，入力 infile の先頭に options(echo = FALSE) を加えるか，オプション '--slave' を使う.

infile は LF や CRLF でマークされる行末を持つことが出来 (しかし CR だけでは無い)，不完全な最終行を持つファイル (行末 (EOL) マークを欠く)は正しく処理される.

後者が q(runLast = FALSE) を呼び出すか中断されない限り，最終表現式 'proc.time()' が入力スクリプトの後で実行される．これはオプション '--no-timing' で抑制することが出来る.

追加のオプションは環境変数 R_BATCH_OPTIONS で設定できる：これらは既定オプション (options 引数の説明を見よ)の後かコマンドラインで与えられた任意のオプションの前に来る.

Note

Unix 風システム上での Splus BATCH とは異なり，これは R をバックグラウンドでは実行しない．殆どのシェルでは

```
R CMD BATCH [options] infile [outfile] &
```

がこれを行う.

---

bibentry 文献エントリー

Description

拡張 BibTeX スタイルでの文献情報の表示と操作のための機能.

Usage

```
bibentry(bibtype, textVersion = NULL, header = NULL, footer = NULL,
         key = NULL, ..., other = list(),
         mheader = NULL, mfooter = NULL)
## S3 method for class 'bibentry'
print(x, style = "text", .bibstyle, ...)
```

Arguments

bibtype	BibTeX エントリータイプを持つ文字列．詳細は**エントリータイプ**節を見よ.
textVersion	オプションでプリント時に使われる参照のテキスト表現を持つ文字列.
header	オプションのヘッダーテキストを持つ文字列.
footer	オプションのフッターテキストを持つ文字列.
key	エントリーに対する引用キーを与える文字列.

...	bibentry 用：エントリーの欄を与える形式 *tag*=*value* の引数で， *tag* と *value* はそれぞれ欄の名前と値．空の値の欄は捨て去られる．詳細はエントリー欄節を見よ． print メソッドに対しては，表示に渡される追加のパラメータ．
other	... 中に於けるような引数のリスト (特に bibentry の形式的引数と同じ名前の欄に対して有用).
mheader	オプションの“外部”ヘッダーテキストを持つ文字列.
mfooter	オプションの“外部”フッターテキストを持つ文字列.
x	クラス "bibentry" を継承するオブジェクト.
style	プリントスタイルを指定するオプションの文字列．もし存在すれば，利用可能なスタイルのユニークな省略形 (大・小文字は無視して)でなければならない， 詳細節を見よ．
.bibstyle	参考文献のスタイルを命名する文字列.

Details

bibentry で作られた bibentry オブジェクトは任意の正の参考文献番号を表すことが出来る． c() を使って bibentry オブジェクトを結合でき，そして従って特に単一の参考文献から複数の参照オブジェクトを構築できる．もしくは，与えられた引数を“ベクトル化”することで複数の参照オブジェクトを直接作るのに bibentry を使うことが出来る，つまり，文字列の代わりに文字列ベクトルを使う．

bibentry オブジェクトに対する print メソッドは7種類の異なったスタイルからの選択を提供する：平文テキスト(スタイル "text")， BibTeX ("Bibtex"), 伝統的に引用に対して使われるような平文テキストと BibTeX の混合("citation")， HTML ("html"), LaTeX ("latex")， R コード ("R")，そして textVersion 要素の単純なコピー (スタイル "textVersion")． "text", "html" そして "latex" スタイルは bibstyle 関数を使い .bibstyle 引数を利用する． bibentry オブジェクトを citation スタイルでプリントする際は，各項目に対する header/footer を全ての参照のベクトルに対する mheader/mfooter と並んで表示できる．

print メソッドは同じスタイルを提供する format メソッドに基づいており，そして R コードとしての書式化に対しては各 bibentry (‘CITATION’ フィルで通常使われるような) に対して一つの bibentry() を持つ文字列で与えるか，またはもし複数の bibentry があれば個々の呼び出しを c() で連結して得られる文字列から選択する．これはオプション collapse をそれぞれ FALSE (既定値)か TRUE に設定することで制御できる． (R スタイルでのプリントは常に単一の呼び出しに縮小される．) 更に "citation" スタイルに対しては format() のオプション引数 citation.bibtex.max (既定では getOption("citation.bibtex.max") で既定値は 1) が最大幾つの引用 bibentry に対してテキストスタイルが bibtex と共に表示されるかを自動的に決定する．

bibentry オブジェクトをそれらのキーを使って添え字操作することが出来る (もし名前が NULL ならば文字列添字に対して使用できる).

BibTeX への直接な変換に対する toBibtex メソッドもある．

Value

bibentry はクラス "bibentry" のオブジェクトを作る．

エントリータイプ

bibentry は "bibentry" オブジェクトを作る，これは BibTeX エントリーに倣ってモデル化されている．エントリーは適正な BibTeX エントリータイプであるべきである，つまり，

Article: ジャーナルまたはマガジンからの論文.

Book: 出版社が明示された本.

InBook: 本の一部分で，章(又は節等) と/又はページ範囲.

InCollection: それ自体のタイトルを持つ本の一部.

InProceedings: 集会プロシーディング中の論文.

Manual: ソフトウェアマニュアル等の技術ドキュメント.

MastersThesis: 修士論文.

Misc: 他に何も適当なものがなければこのタイプを使う.

PhdThesis: 学位論文.

Proceedings: 集会プロシーディング.

TechReport: 学校や他の機関が出版した報告書で，普通シリーズ番号を持つ.

Unpublished: 著者とタイトルを持つドキュメントだが，公式な出版物ではない.

エントリー欄

`bibentry` の `...` 引数は以下を含む任意個数の BibTeX 欄でよい，

address: 出版社又は他の機関.

author: 著者の名前， LaTeX 本で説明されているような書式での文字列か，又は `person` オブジェクト.

booktitle: その一部分が引用されている本のタイトル.

chapter: 章(又は節等)の番号.

doi: 参照のための DOI (`https://en.wikipedia.org/wiki/Digital_Object_Identifier`).

editor: 編者の名前，`author` と同じ書式.

institution: テクニカルレポートの発行機関.

journal: ジャーナル名.

note: 読者の助けになる任意の追加情報. 最初の単語は大文字化されるべきである.

number: ジャーナル，マガジン，テクニカルレポート，又はシリーズ物の仕事の番号.

pages: 一つ以上のページ数か，その範囲.

publisher: 出版社名.

school: 学位論文が書かれた学校の名前.

series: 本のシリーズやセットの名前.

title: 作品の名前.

url: 参照のための URL. (もし URL が拡大された DOI ならば，代わりに拡大 DOI を持つ '`doi`' 欄を勧める.)

volume: ジャーナル又は複数巻の本の巻数.

year: 出版年.

See Also

`person`

Examples

```
## R の参照
rref <- bibentry(
   bibtype = "Manual",
   title = "R: A Language and Environment for Statistical Computing",
   author = person("R Core Team"),
   organization = "R Foundation for Statistical Computing",
   address = "Vienna, Austria",
   year = 2014,
   url = "https://www.R-project.org/")

## 異なったプリントスタイル
print(rref)
print(rref, style = "Bibtex")
print(rref, style = "citation")
print(rref, style = "html")
print(rref, style = "latex")
print(rref, style = "R")

## パッケージと関連本双方の参照
bref <- c(
   bibentry(
     bibtype = "Manual",
     title = "boot: Bootstrap R (S-PLUS) Functions",
     author = c(
       person("Angelo", "Canty", role = "aut",
         comment = "S original"),
       person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
         comment = "R port, author of parallel support",
         email = "ripley@stats.ox.ac.uk")
     ),
     year = "2012",
     note = "R package version 1.3-4",
     url = "https://CRAN.R-project.org/package=boot",
     key = "boot-package"
   ),

   bibentry(
     bibtype = "Book",
     title = "Bootstrap Methods and Their Applications",
     author = as.person("Anthony C. Davison [aut], David V. Hinkley [aut]"),
     year = "1997",
     publisher = "Cambridge University Press",
     address = "Cambridge",
     isbn = "0-521-57391-2",
     url = "http://statwww.epfl.ch/davison/BMA/",
     key = "boot-book"
   )
)

## 連結と部分抽出
c(rref, bref)
bref[2]
bref["boot-book"]

## 欄の取り出し
```

```
bref$author
bref[1]$author
bref[1]$author[2]$email

## BibTeX に変換
toBibtex(bref)

## R スタイルの書式
## 各 bibentry に対して一つの bibentry() 呼び出し：
writeLines(paste(format(bref, "R"), collapse = "\n\n"))
## 一つの連結呼び出し：
writeLines(format(bref, "R", collapse = TRUE))
```

---

browseEnv 環境中のオブジェクトをブラウザで見る

Description

browseEnv 関数は現在 sys.frame() 環境中にあるオブジェクトのリストに対してブラウザを開く.

Usage

```
browseEnv(envir = .GlobalEnv, pattern,
          excludepatt = "^last\\.warning",
          html = .Platform$GUI != "AQUA",
          expanded = TRUE, properties = NULL,
          main = NULL, debugMe = FALSE)
```

Arguments

envir	そこのオブジェクトをブラウズすべき environment.
pattern	オブジェクトの部分選択に対する regular expression は内部的な ls() 呼び出しに渡される.
excludepatt	マッチした名前を持つ 除外オブジェクトに対する正規表現.
html	好みのブラウザ中の HTML ページ上で作業空間を表示するのに使われる． OS X 上の R.app から実行している時を除き既定値.
expanded	一段階の再帰のレベルを表示するかどうか．もし作業空間が巨大な時それを FALSE に切り替えるのが有用になり得る．このオプションはもし html が FALSE に設定されると無視される.
properties	ブラウザ中で表示される (選択されたオブジェクトの)大局的性質の名前付きリスト； NULL の時(既定値であるように)は，ユーザ，日付，そしてマシーン情報が使われる.
main	ブラウザ中でで使われるタイトル文字列； NULL の時(既定であるように)はあるタイトルが構成される.
debugMe	論理値スイッチ；もし真ならば，ある種の診断出力が作られる.

Details

非常に実験的なコード：OS X の R.app を除き全てのプラットフォーム上で静的な HTML ページを表示する．

オブジェクトの構造の一つの再帰レベルのみを許す．

これは一般化出来る．詳細はソースを見よ．最も高い可能性として，これはむしろ **tkWidget** パッケージ (https://www.bioconductor.org から)を使って動作すべきである．

See Also

str, ls.

Examples

```
if(interactive()) {
   ## ある面白いオブジェクトを作る：
   ofa <- ordered(4:1)
   ex1 <- expression(1+ 0:9)
   ex3 <- expression(u, v, 1+ 0:9)
   example(factor, echo = FALSE)
   example(table, echo = FALSE)
   example(ftable, echo = FALSE)
   example(lm, echo = FALSE, ask = FALSE)
   example(str, echo = FALSE)

   ## そしてそれらをブラウザーで見る：
   browseEnv()

   ## (簡単な)関数の環境：
   af12 <- approxfun(1:2, 1:2, method = "const")
   browseEnv(envir = environment(af12))
 }
```

---

browseURL　　*URL を HTML ブラウザ中にロード*

Description

与えられた URL を HTML ブラウザ中にロードする．

Usage

```
browseURL(url, browser = getOption("browser"),
          encodeIfNeeded = FALSE)
```

Arguments

url	ロードされる URL を与える空でない文字列．
browser	HTML ブラウザとして使われるプログラムの名前を与える空でない文字列．これは PATH 中にあるか完全なパスを与えるべきである．もしくはブラウザを起動するために呼びだされる R 関数． Windows の下では NULL も許され (そして既定値である)，そしてファイルに関連した機構が使われることを意味する．

encodeIfNeeded URL はブラウザに渡される前に URLencode でエンコードされるべきか? もし browser プログラム/関数がそれ自体エンコーディングを行うならばこれは不要(で有害かもしれない)，そしてある種のシステム上では 'file://' URL に対して，またある種の CGI アプリケーションに渡された 'http://' URL 対して有害かもしれない．幸運にもほとんどの URL はエンコーディングを必要としない．

Details

既定のブラウザはオプション "browser" で設定され，そしてそれは既定で R がコンフィギュアされた時に手動か自動で選択されるファイル '*R_HOME*/etc/Renviron' 中で既定で設定される環境変数 R_BROWSER で設定される．(既定値をどこで書き換えるについては Startup を見よ．) URL の表示を一切抑制するには値 "false" を使う．

多くのプラットフォームではオプション "browser" を本来のプログラム/スクリプトに設定し，そしてユーザが選択したブラウザをそれで起動するのが最良である．例えば OS X では open を使い，そして多くの他の Unix 風システムでは xdg-open を使う．

もし browser が遠隔制御をサポートしそして R がそれをどのように実行するかを知っていれば， URL は任意のすでに稼働中のブラウザや必要なら新しいそれで開かれる．この機構は現在 "-remote openURL(...)" インタフェイスをサポートするブラウザ(Mozilla と Opera が該当する)に対して利用でき， Galeon, KDE konqueror (kfmclient 経由で)そして GNOME は Mozilla へのインタフェイスを持つ．(Firefox はサポートを停止したが，すでに実行中のブラウザの使用が既定である．) ブラウザのタイプはその名前から決定されることを注意する．従ってこの機構はもしブラウザがその標準的な名前でインストールされている時だけ使える．

"-remote" は X サーバ上で表示される任意のブラウザを使うため (それがどのようなマシン上で動いていようと)，遠隔制御機構はもし DISPLAY がローカルホストを指している時だけ使われる．これはリモートホストからの複数の URL を一度に表示できないかもしれない．

もし必要なら url をエンコードするのは呼び出し側の責任である(URLencode を見よ)．これはファイル URL に対してはトリッキーになり得る，そこでは受け入れられる書式はブラウザと OS 双方に依存する．

URL の表示を一切抑制するには browser = "false" と設定する．

URL でない引数 url に対する挙動はプラットフォーム依存である．ある種のプラットフォームは絶対ファイルパスを受け入れる：相対ファイルパスを受け入れるものは少ない．

Examples

```
## Not run: ## 新しいタブでファイルを開きたいKDEユーザ用
options(browser = "kfmclient newTab")
browseURL("https://www.r-project.org")

## End(Not run)
```

---

browseVignettes *HTML* ブラウザでビニエットを表示する

Description

PDF，LaTeX/noweb ソース，そして(tangle された) R コード (もし利用できれば)にリンクした HTML ブラウザに利用可能なビニエットをリストする．

Usage

```
browseVignettes(package = NULL, lib.loc = NULL, all = TRUE)

## S3 method for class 'browseVignettes'
print(x, ...)
```

Arguments

package	探索されるパッケージの名前を持つ文字列ベクトル，又は全てのパッケージを探す NULL (引数 all で定義されたような).
lib.loc	R ライブラリのディレクトリ名の文字列ベクトル，又は NULL．既定値の NULL は現在知られている全てのライブラリに対応する.
all	論理値；もし TRUE なら lib.loc で指定されるライブラリツリー中の全ての利用可能なパッケージを探す．そしてもし FALSE ならば付加されたパッケージだけを探す.
x	クラス browseVignettes のオブジェクト.
...	追加の引数，print メソッドでは無視される.

Details

関数 browseVignettes は同じクラスのオブジェクトを返す；プリントメソッドはそれをブラウザ中の HTML ページとして表示する (browseURL を使う).

See Also

browseURL, vignette

Examples

```
## 付加された全てのパッケージからのビニエットをリストする
browseVignettes(all = FALSE)

## 指定したパッケージからのビニエットをリストする
## List vignettes from a specific package
browseVignettes("grid")
```

---

bug.report　　バグ報告を送る

Description

バグ報告を書くためエディタか電子メイルプログラムを起動するか，バグ投稿のためのウェッブページを開く．現在の R のバージョンとコンフィギュレーションに関するある種の標準的情報が自動的に含められる.

Usage

```
bug.report(subject = "",  address,
           file = "R.bug.report", package = NULL, lib.loc = NULL,
           ...)
```

Arguments

subject	電子メイルの主題.
address	利用可能なら受け取り手の電子メイルアドレス：電子メイルで送られたパッケージのバグ報告に対しては，これは既定でパッケージの保守管理者のアドレスになる (複数いれば最初の人).
file	電子メイルの設定に対して使われるファイル名(もし必要なら).
package	バグ報告の主題である単一のパッケージの名前を与えるオプションの文字列ベクトル.
lib.loc	その中でパッケージを探す R のライブラリツリーの位置を記述する文字列ベクトル，又は NULL．既定値の NULL は現在既知の全てのライブラリに対応する.
...	method と ccaddress の様な create.post に渡される追加の名前付き引数.

Details

もし package が NULL か基本パッケージならば，これは https://bugs.r-project.org/ にある R のバグ追跡機能を開く.

もし package が指定されると，これはバグ報告がそのパッケージに対するものと仮定され，その 'DESCRIPTION' ファイルの一部が標準情報に加えられる．もしパッケージが 'DESCRIPTION' ファイル中に BugReports 欄を持てば，その URL が browseURL を使い開かれる，さもなければパッケージの保守管理者に宛てた電子メイルが create.post を使って作られる.

Value

有用なものは何もない.

バグはいつあるか?

もし R が不正な指示を実行するか，又はプログラム中の問題を指示するオペレティングシステムのエラーメッセージと共に停止すれば ("disk full" といった何かとは異なり)，それは確かにバグである.

命令がいつまでも完了しないことはバグの可能性があるが，それが本当に R の責任かどうか確認する必要がある．ある種の命令は単に長時間を要する．もし入力が素早く処理されるべきであったことを知っていればバグを報告する．もし命令が長い時間を必要とするかどうかを知らなければ，マニュアルを熟読するか助けを求めよ.

もし良く知っている命令がその通常の定義が合理的であるべきケースで R のエラーメッセージを惹き起こすならば，それは恐らくバグである．もし命令が間違ったことを実行すれば，それはバグである．しかしながらなされるべきであったことを確実に知っていることを確認せよ．もしその命令に不慣れか，命令がどのように動作するはずかを知らなければ，それは実際は正しく動作していたかもしれない．結論に飛びつく前に良く知っている人に問題を見せよう.

最後に，命令の想定する定義が統計解析に対して最良ではないかもしれない．これは非常に重要な類の問題であるが，しかしまた判断の問題でもある．又，ある種の既存の特性の無知からそうした結論に至ることは容易である．ドキュメントを通常の仕方でチェックし，自分がそれを理解していることと，自分が必要とすることが利用できないことに確信が持てるまでは，そうした問題に不平を述べないことが恐らくベストである．この種のことの議論に対してはメイリングリスト r-devel@r-project.org はバグリストよりも良い場所である.

もしマニュアルを注意深く読んだ後で命令が何をしようとするのか得心できなければ，これはマニュアル中のバグである．マニュアルの仕事は全てを明確にすることにある．マニュアルのドキュメントのバグを報告することはプログラムのバグのそれと同じぐらい重要である．

もし関数のオンライン引数リストがマニュアルと一致しなければ，それらの一つはバグに違いなく，従ってバグを報告する．

どのようにバグを報告するか

バグがあると決定した時，それを報告しそしてそれを役に立つ仕方で報告することが重要である．最も有用なことはどの命令をタイプしたのかの正確な説明，問題が起きるまでにいつ R を開始したのかである．常に使っている R のバージョン，マシーン，そしてオペレーティング・システムを含めること；これをプリントするには R で `version` とタイプする．どのバグがフィックスされているかそしてどれが依然オープンかを追跡するのを助けるためどうか各バグ毎に別個の報告を送って欲しい．

バグの報告で最も重要な原則は事実の報告であり，仮説や分類ではない．事実を報告することは常により簡単であるが，人は仮説的説明に固執しそれらを報告することを好む傾向がある．もし説明が R がどのように実装されているかに関する推測に基づいているならば，それらは役に立たないであろう；我々はどのような事実がそうした推測に導いたはずかを理解することを試みなければならない．これはしばしば不可能である．しかしながらいかなる場合も，それは我々に取って不要な作業である．

例えば，極めて大きなデータセットに関して命令 `data.frame(x, y, z, monday, tuesday)` が決して終了しないと仮定しよう． `data.frame()` が大きなデータセットに対して失敗することを報告しない．おそらくそれは変数名が曜日の時に失敗する．もしそうであるならば，我々があなたの報告を受け取った時 `data.frame()` 命令を大きなデータセットに対して試みるであろうが，恐らく曜日の変数名を使わず，どのような問題も発見しない．曜日変数名で試みるべきであることを我々が推測出来るはずだったと知る方法は全く無い．

又は命令が失敗するのは，使用した直近の命令が R の内部データ構造を毀損し `data.frame()` 命令がそれ以降失敗するバグを持つ [メソッドだったからである．これがなぜ我々があなたがタイプした (またはあなたのスタートアップファイルから)他の命令を知る必要があるかの理由である．

見かけ上同じバグを生成する簡単な例を試み見つけることは非常に有用であり，時としてバグを起こすはずだが実際は起こさない簡単な例を見つけることが有用である．もしあなたが問題をデバッグしそれがなぜ起きるかを正確に見い出せば，それは素晴らしいことである．あなたは依然として説明や解決策と共に事実を報告すべきである．

R をオプション '`--vanilla`' で起動すればバグを特定するのに役立つかもしれない．これはサイトのプロファイルと保存データファイルを読み込まないことを保証する．

バグ報告は関数 `bug.report()` を用いて作ることが出来る． R に関する報告に対してはこれはウェッブページ https://bugs.r-project.org/ を開く：貢献パッケージに対してはこれはパッケージのバグトラッカーのウェッブページを開くか又は保守管理者宛の電子メイルを書くのを助ける．

contributed packages に関するバグ報告は R のバグトラッカーに送られるべきではない：むしろ `package` 引数を利用しよう．

Author(s)

このヘルプページは Emacs マニュアルと R FAQ から転用された

See Also

bug.report 前に恐らく試みるべき help.request.

create.post は電子メイル報告を処理する.

バグ報告にそこから付け加えるかもしれない R FAQ, 又 sessionInfo().

---

capture.output 出力を文字列かファイルに送る

Description

出力を文字列としてかファイルに送ることでその引数を評価する. with が attach に関連するのと同じ仕方で sink に関連する.

Usage

```
capture.output(..., file = NULL, append = FALSE,
type = c("output", "message"), split = FALSE)
```

Arguments

...	評価される表現式.
file	ファイル名又は connection, または出力を文字列として返すには NULL. もしコネクションが開かれていなければ, 最初にそれが開かれそして終了時に閉じられる.
append	論理値. もし file がファイル名か開かれていないコネクションならば追記するか上書きするか?
type, split	sink() に渡される, そこを見よ.

Details

もし表現式を評価する際にエラーがあれば出力を可能な限り file に書き出す試みが行われるが, file = NULL に対しては全ての出力は失われる.

stderr() に送られるメッセージ (message, warning そして stop からのものを含む)が type = "message" により捕捉される. これは "安全でない"可能性があり注意深く使われるべきであることを注意する.

Value

文字列(もし file = NULL なら), または不可視の NULL.

See Also

sink, textConnection

Examples

```
require(stats)
glmout <- capture.output(summary(glm(case ~ spontaneous+induced,
                                     data = infert, family = binomial())))
glmout[1:5]
capture.output(1+1, 2+2)
capture.output({1+1; 2+2})

## Not run: ## a2ps が利用できるUnixシステムで
op <- options(useFancyQuotes=FALSE)
pdf <- pipe("a2ps -o - | ps2pdf - tempout.pdf", "w")
capture.output(example(glm), file = pdf)
close(pdf); options(op) ; system("evince tempout.pdf &")

## End(Not run)
```

---

changedFiles　どのファイルが変更されたかを検出する

Description

fileSnapshot は各々の要約情報を記録したファイルの選択のスナップショットを取る．changedFiles は二つのスナップショットを比較するか，一つのスナップショットを現在のファイルシステムの状態と比較する．スナップショットは同じディレクトリである必要はない；これは二つのディレクトリの比較に使えるかもしれない．

Usage

```
fileSnapshot(path = ".", file.info = TRUE, timestamp = NULL,
    md5sum = FALSE, digest = NULL, full.names = length(path) > 1,
    ...)

changedFiles(before, after, path = before$path, timestamp = before$timestamp,
    check.file.info = c("size", "isdir", "mode", "mtime"),
    md5sum = before$md5sum, digest = before$digest,
    full.names = before$full.names, ...)

## S3 method for class 'fileSnapshot'
print(x, verbose = FALSE, ...)

## S3 method for class 'changedFiles'
print(x, verbose = FALSE, ...)
```

Arguments

path	文字列ベクトル；記録へのパス．
file.info	論理値；各ファイルに対する file.info 値を記録するか．
timestamp	文字列か NULL；スナップショットが取られる際の書き込みファイルの名前．これは変更に対する簡単なテストを与えるが，信用できないかもしれない；詳細を見よ．

md5sum	論理値；各ファイルの MD5 要約がスナップショットの一部として使われるべきであるか.
digest	関数または NULL；ヘッダー function(filename) を持つ関数でファイル名のベクトルを取り，同じ長さの値のベクトル，またはその数の行を持つ行列を生成する.
full.names	論理値；完全な名前 (list.files 中に於けるような)を記録すべきか. もし length(path) > 1 ならば TRUE でなければならない.
...	スナップショット中のファイルのセットを制御するために list.files に渡される追加のパラメータ.
before, after	fileSnapshot により作られるオブジェクト；比較される二つのスナップショット. もし after が欠損していれば，現在のファイルシステムの新しいスナップショットが，既定として before 中に記録された引数を用い，比較のために生成される.
check.file.info	文字列ベクトル；file.info からのどの列が比較されるべきか.
x	プリントするオブジェクト.
verbose	論理値；プリント時に全てのデータをリストすべきか.

Details

fileSnapshot 関数はファイルのリストを得るのに list.files を使い，そして file.info, md5sum そして digest 引数に依存して各ファイルに関する情報を記録する.

changedFiles 関数は二つのスナップショットを比較する. もし fileSnapshot への timestamp 引数が長さ1ならば，その名前のファイルが作られる. もしそれが changedFiles 中で長さが1ならば， file_test 関数がそれへの before と after の双方に共通の全てのファイルの作成日時を比較するのに使われる. このテストは信用できないかもしれない：それは after ファイルの現在の変更時間をタイムスタンプと比較する；これは after スナップショットが取られた時の変更時間とは異なるかもしれない. これは又もしタイムスタンプを保持するファイルシステムの時計がスナップショットのファイルを保持するそれと異なるなら不正な結果を与えるかもしれない.

もし check.file.info 引数が空でない文字列ベクトルを含めば，file.info への呼び出しの結果からの指示された列が比較される.

もし md5sum が TRUE ならば，fileSnapshot は各ファイルの32バイトのMD5チェックサムを記録するために tools::md5sum 関数を呼び出し，そして changedFiles は値を比較する. digest 引数はユーザが自分自身のダイジェスト関数を提供することを許す.

Value

fileSnapshot はクラス "fileSnapshot" のオブジェクトを作る. これは次の欄を含むリストである

info	その行名がファイル名で，列が要求されたスナップショットデータを含むデータフレーム
path	呼び出しからの正規化された path
timestamp, file.info, md5sum, digest, full.names	呼び出しからの他の引数の記録
args	... により list.files に渡される他の引数.

changedFiles はクラス "changedFiles" のオブジェクト. これは次を含むリストである

added, deleted, changed, unchanged
: 自明の意味を持つ，スナップショットの前と後からのファイル名の文字列ベクトル

changes
: 各共通ファイルに対する行と，各比較テストに対する列を持つ論理値行列． TRUE はそのテストに於ける変化を指示する．

print メソッドはこれらのタイプの各々に対して定義される． "fileSnapshot" オブジェクトに対する print メソッドはそれらを生成するのに使われる引数を表示する一方，"changedFiles" に対するそれはもし空でなければ added, deleted そして changed 欄，そして全ての TRUE 値を含む changes 行列の副行列を表示する．

Author(s)

Duncan Murdoch， Karl Millar 他の示唆を使う．

See Also

file.info, file_test, md5sum.

Examples

```
# 一時ディレクトリ中に幾つかのファイルを作る
dir <- tempfile()
dir.create(dir)
writeBin(1L, file.path(dir, "file1"))
writeBin(2L, file.path(dir, "file2"))
dir.create(file.path(dir, "dir"))

# スナップショットを取る
snapshot <- fileSnapshot(dir, timestamp = tempfile("timestamp"), md5sum=TRUE)

# ファイルの一つを変更する．
writeBin(3L:4L, file.path(dir, "file2"))

# 検出された変更を表示． mtime の変化を見ることが出来るかどうか...
changedFiles(snapshot)
changedFiles(snapshot)$changes
```

---

chooseBioCmirror *Bioconductor* のミラーを選択する

Description

Bioconductor のミラーを対話的に選択する．

Usage

```
chooseBioCmirror(graphics = getOption("menu.graphics"), ind = NULL,
                 useHTTPS = getOption("useHTTPS", TRUE),
                 local.only = FALSE)
```

Arguments

graphics	論理値．もし真ならグラフィカルなリストを使う： Windows や OS X GUI ではリストボックスを使い， Unix 風システムではもしパッケージ **tcltk** と X サーバーが利用できれば Tk ウィジェットを使う．さもなければテキスト menu を使う．
ind	どの項目を選択するかを与えるオプションの数値．
useHTTPS	HTTPS ミラーを優先するかどうか (chooseCRANmirror を見よ)．
local.only	論理値， Bioconductor マスターからの最新のリストを得ることを試みるかまたはローカルなディスクにあるファイルを使うか．

Details

これはオプション "BioC_mirror" を設定する： setRepositories の呼び出し前に使われる．そのオプションのそのままの既定値は NULL であり，ミラー https://bioconductor.org に対応する．

‘(汎世界的な) Bioconductor ’‘ミラー’は信用できる汎世界的なアクセスを提供するミラーのネットワークである；他のミラーは地理的に局所的なスケールでのより速いアクセスを提供する．

ind は ‘*R_HOME*/doc/BioC_mirrors.csv’ 中の行を数で選択する．

Value

無い：この関数は options("BioC_mirror") を更新するその副作用のために起動される．

See Also

setRepositories, chooseCRANmirror.

---

chooseCRANmirror *CRAN* ミラーを選択する

Description

CRAN ミラーを対話的に選択する．

Usage

```
chooseCRANmirror(graphics = getOption("menu.graphics"), ind = NULL,
                 useHTTPS = getOption("useHTTPS", TRUE),
                 local.only = FALSE)

getCRANmirrors(all = FALSE, local.only = FALSE)
```

Arguments

graphics	論理値．もし真ならばグラフィカルなリストを使う： Windows や OS X GUI ではリストボックスを使い， Unix 風システムではもしパッケージ **tcltk** と X サーバーが利用可能なら Tk ウィジェットを使う．さもなければテキスト menu を使う．
ind	どの項目を選ぶかを与えるオプションの数値．
useHTTPS	HTTPS ミラーを優先するか．
all	論理値，全ての既知のミラーを得るか又は OK というフラグが立つものだけにするか．
local.only	論理値， CRAN マスターからの最新のリストを得るか，それとも局所的なディスク上のファイルだけを使うか．

Details

ミラーのリストはファイル '*R_HOME*/doc/CRAN_mirrors.csv' 中に保管されるが，しかし先ず現在のミラーのオンラインリストが検討され，ファイルコピーはオンラインリストがアクセスできない時だけ使われる．

この関数は Windows の GUI メニュー項目と，もしそれが options("repos") の初期ダミー値を見出した時 contrib.url により呼び出される．

useHTTPS 引数の既定値は TRUE である． useHTTPS = TRUE では HTTPS ミラーが HTTP ミラー(これはサブメニュー中にリストされる)に優先して提供される．もし FALSE なら，HTTPS ミラーは提供されない． HTTPS ミラーは選択されたサイトの身元に関するある種の保証を提供するために推薦される．(しかしながら，全てではないがほとんどの R ビルトは HTTPS サイトからのダウンロードをサポートする．)

ind は現在のミラーリスト中の行を数で選択する．これは local.only = TRUE，そして '*R_HOME*/doc/CRAN_mirrors.csv' と共に使われるのが最良である．

Value

chooseCRANmirror() に対しては何もない，この関数は options("repos") を更新するその副作用のため起動される．

getCRANmirrors() はミラー情報を持つデータフレームを返す．

See Also

setRepositories, chooseBioCmirror, contrib.url.

---

citation 出版物中で *R* と *R* パッケージを引用する

Description

出版物中で R と R パッケージをどのように引用するか．

Usage

```
citation(package = "base", lib.loc = NULL, auto = NULL)
readCitationFile(file, meta = NULL)
```

Arguments

package	一つのパッケージの名前を持つ文字列．複数のパッケージの名前を与えるとエラーになる．
lib.loc	R ライブラリのパス名の文字列，または NULL．既定値 NULL は現在既知の全てのライブラリに対応する．もし既定値が使われると，ロードされたパッケージがライブラリの前に検索される．
auto	パッケージの 'DESCRIPTION' メタデータから自動生成された既定の引用を使うべきかどうかを指示する論理値，または NULL (既定)で，もし存在すれば 'CITATION' ファイルを使うか，またはパッケージのメタデータを持つクラス "packageDescription" のオブジェクトを使うかを指示する(下を見よ)．
file	ファイル名．
meta	packageDescription により得られるようなパッケージのメタデータのリスト，または NULL (既知)．

Details

R コア開発チームと非常に活動的なパッケージ作者のコミュニティは現在あるような R を作るために多大な時間と労力を費やしてきている．感謝が相応しい場合は感謝を述べ，そしてデータ解析のためにそれらを使った時は R と R パッケージを引用して欲しい．

基本 R システムを出版物でどのように引用するかの情報は citation() を実行する．もし基本でないパッケージの名前が与えられると，関数はパッケージの 'CITATION' ファイル中に含まれる情報を返すか，または自動生成された引用情報を返す．後者の場合 'DESCRIPTION' ファイルが構文解析され，結果の引用オブジェクトは恣意的な間違いを含むかもしれないが，多くの場合(少なくとも出発点として)非常に有用である．

R >= 2.14.0 では，'DESCRIPTION' 中の 'Authors@R' 欄をパッケージの"作者"の精選された機械可読な (R コードを与える) person オブジェクトを提供するために使うことが出来る(特に彼らの正確な役割を指定する)．作者の役割をもつ者だけが自動生成引用に含められる．

もしただ一つの参照が与えられると，citation() が返すオブジェクトに対するプリントメソッドはテキスト版とそれに対する BibTeX 項目を示す．もしパッケージが複数の参照を持てばテキスト版だけが返される．BibTeX 版は関数 toBibtex() を用いて得ることが出来る(下の例を見よ)．

R パッケージの 'CITATION' ファイルはパッケージソースの 'inst' サブディレクトリ中に置かれるべきである．このファイルは R のソースファイルで条件と計算を含む任意の R 命令を含むことが出来る．関数 readCitationFile() は 'CITATION' ファイル中の情報を取り出すために citation() により使われる．このファイルは R の構文解析器により一時的環境中で source() され，全ての結果の参考文献オブジェクトが収集される (特にクラス "bibentry" のもの)．

伝統的にファイル 'CITATION' は citHeader のゼロ個以上の呼び出し，それから一つ以上の citEntry の呼び出し，そして最後にゼロ個以上の citFooter の呼び出しを含んでいた．ここで実際には citHeader と citFooter は単に paste へのラッパで，それらの ... 引数はそのままで paste に渡される． "bibentry" クラスは参考文献情報の改良された表現と操作に役立つ(実際古い機構は新しい機構を使って実装されている)，そして 'CITATION' ファイルを統一された bibentry インタフェイスを使って書くことが出来る．

citation(auto = meta) を使って 'CITATION' ファイル中に自動生成されたパッケージの引用を含めることが出来る．

readCitationFile はファイルのエンコーディングを決定するために meta の Encoding 要素を使うことが出来る(もしあれば)．

Value

クラス "bibentry" を継承するオブジェクト．．

See Also

bibentry

Examples

```
## 基本的な R 引用
citation()

## パッケージに対する参照 -- これらはインストールされていないかもしれない
if(nchar(system.file(package = "lattice"))) citation("lattice")
if(nchar(system.file(package = "foreign"))) citation("foreign")

## 返り値から bibtex 項目を取り出す
x <- citation()
toBibtex(x)
```

---

cite 参考文献項目を引用する．

Description

テキスト中に bibentry オブジェクトを引用する． cite() 関数はもしあれば既定の bibstyle から，もしなければ citeNatbib() からの cite() 関数を使う． citeNatbib() は LaTeX パッケージ **natbib** が使うものに類似のスタイルを使う．

Usage

```
cite(keys, bib, ...)
citeNatbib(keys, bib, textual = FALSE, before = NULL, after = NULL,
           mode = c("authoryear", "numbers", "super"),
           abbreviate = TRUE, longnamesfirst = TRUE,
           bibpunct = c("(", ")", ";", "a", "", ","), previous)
```

Arguments

keys	引用する項目のキーの文字列ベクトル．一つの項目中に複数のキーをコンマで分離し含むかもしれない．
bib	その中でキーを探すドキュメントのリストを含む "bibentry" オブジェクト．
...	既定のスタイルに対して cite() 関数に渡される追加の引数．
textual	引用の“テキスチャル”なスタイル，つまりどの \citet を LaTeX 中で作り出すか．
before	引用の前に表示されるオプションのテキスト．
after	引用の後に表示されるオプションのテキスト．
mode	引用の “モード”．
abbreviate	長い著者リストを省略するかどうか．

longnamesfirst もし abbreviate == TRUE なら，最初の引用を長いままにするかどうか.

bibpunct 引用中で使われる句読点のベクトルで， **natbib** 中で使われるようなもの．詳細節を見よ.

previous 先に引用されているキーのリストで， abbreviate == TRUE で longnamesfirst == TRUE の時使われる.

Details

引数名は LaTeX の **natbib** パッケージに対するドキュメントに基づいて選ばれる. bibpunct 項目の解釈に対してはドキュメントを見よ.

bibpunct 中の項目は次の通りである：

1. 左の分離子.
2. 右の分離子.
3. 引用中の参考文献間の分離記号.
4. “モード”の指示子：数に対しては "n"，上付き添字に対しては "s"，著者-年度に対するその他.
5. 著者と年度の間に来る句読点.
6. 著者が伏せられた時年度の間に来る句読点.

もし mode が指定されると，それは bibpunct[4] 中のモード指定を上書きすることを注意する． mode に対しては部分マッチングが使われる.

citeNatbib に対する既定値は JSS スタイルにマッチするように選ばれ，そして既定でこれらは cite 中で使われる．異なった既定スタイルをどのように設定するかについては bibstyle を見よ.

Value

引用を含む一つの文字列が返される.

Author(s)

Duncan Murdoch

Examples

```
## R の参照
rref <- bibentry(
   bibtype = "Manual",
   title = "R: A Language and Environment for Statistical Computing",
   author = person("R Core Team"),
   organization = "R Foundation for Statistical Computing",
   address = "Vienna, Austria",
   year = 2013,
   url = "https://www.R-project.org/",
   key = "R")

## boot パッケージと関連書籍に対する参照
bref <- c(
   bibentry(
     bibtype = "Manual",
     title = "boot: Bootstrap R (S-PLUS) Functions",
```

```
      author = c(
        person("Angelo", "Canty", role = "aut",
          comment = "S original"),
        person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
          comment = "R port, author of parallel support",
          email = "ripley@stats.ox.ac.uk")
      ),
      year = "2012",
      note = "R package version 1.3-4",
      url = "https://CRAN.R-project.org/package=boot",
      key = "boot-package"
    ),

    bibentry(
      bibtype = "Book",
      title = "Bootstrap Methods and Their Applications",
      author = as.person("Anthony C. Davison [aut], David V. Hinkley [aut]"),
      year = "1997",
      publisher = "Cambridge University Press",
      address = "Cambridge",
      isbn = "0-521-57391-2",
      url = "http://statwww.epfl.ch/davison/BMA/",
      key = "boot-book"
    )
  )

  ## 連結と引用
  refs <- c(rref, bref)
  cite("R, boot-package", refs)

  ## 数値的引用
  savestyle <- tools::getBibstyle()
  tools::bibstyle("JSSnumbered", .init = TRUE,
           fmtPrefix = function(paper) paste0("[", paper$.index, "]"),
           cite = function(key, bib, ...)
            citeNatbib(key, bib, mode = "numbers",
                bibpunct = c("[", "]", ";", "n", "", ","), ...)
           )
  cite("R, boot-package", refs, textual = TRUE)
  refs

  ## 旧式スタイルを回復
  tools::bibstyle(savestyle, .default = TRUE)
```

---

citEntry 文献項目(古いインタフェイス)

Description

拡張 BibTeX スタイルによる文献情報を指定するための機能.

Usage

```
citEntry(entry, textVersion, header = NULL, footer = NULL, ...)
```

```
citHeader(...)
citFooter(...)
```

Arguments

entry	BibTeX 項目タイプを持つ文字列．詳細は bibentry 中の節 **Entry Types**を見よ．
textVersion	参考文献のテキスト表現を持つ文字列．
header	オプションのヘッダーテキストを持つ文字列．
footer	オプションのフッターテキストを持つ文字列．
...	citEntry に対して，項目の欄を与える形式 *tag=value* の引数で， *tag* と *value* はそれぞれ欄の名前と値．詳細は bibentry 中の節 **Entry Types**を見よ． citHeader と citFooter に対しては文字列．

Value

citEntry はクラス "bibentry" のオブジェクトを作る．

See Also

R と R パッケージの引用に関するこれ以上の情報は citation と 'CITATION' ファイル；文献情報を表現し操作するためのより新しい機能は bibentry．

---

close.socket　　ソケットを閉じる

Description

ソケットを閉じ記述子テーブル中の空きを開放する．ポートはすぐには開放されないかもしれない．

Usage

```
close.socket(socket, ...)
```

Arguments

socket	socket オブジェクト
...	他のメソッドに/から渡される追加引数．

Value

成功か失敗かを指示する論理値

Author(s)

Thomas Lumley

See Also

make.socket, read.socket

ソケットのサポート付きでコンパイルすることは R 3.3.0 以前ではオプションであった：それが利用可能かどうかを見るには capabilities("sockets") を見よ.

---

combn　　*n* 個の要素から一時に *m* 個を取る全ての組み合わせを作る

Description

x の要素から m 個を一時に取ってできる全ての組み合わせを作る．もし x が正整数ならば， seq(x) の要素から m 個を一時に取って出来る全ての組み合わせを返す．もし引数 FUN が NULL でなければ，引数で与えられた関数を各点に適用する．もし simplify が FALS ならば，リストを返す；さもなければ array を返すが，典型的には matrix になる．... は，もし指定されれば，そのまま FUN 関数に渡される．

Usage

```
combn(x, m, FUN = NULL, simplify = TRUE, ...)
```

Arguments

x	組み合わせのソースになるベクトル，または整数 x <- seq_len(n) に対する n.
m	選ばれる要素の数.
FUN	各組み合わせに適用される関数；既定の NULL は恒等関数を意味する，つまり，組み合わせを返す(長さ m のベクトル).
simplify	結果を array (典型的には matrix) に単純化するかを指示する論理値；もし FALSE ならば，関数は list を返す．既定の simplify = TRUE の時は，結果の次元は単に FUN(*1st combination*) から決定される (効率化のため)．これはもし FUN(u) が同じ長さでなければ惨めに失敗する.
...	オプションで，FUN への追加引数.

Details

因子の x は R 3.1.0 から受け入れられる(偶々初期のバージョンでは simplify = FALSE に対して動作した).

Value

list 又は array，上の simplify 引数を見よ．後者の場合恒等式 dim(combn(n, m)) == c(m, choose(n, m)) が成立する.

Author(s)

Scott Chasalow は 1994年に S 用にオリジナルを書いた；R パッケージ **combinat** と Vince Carey <stvjc@channing.harvard.edu> によるドキュメント；僅かな変更が R コアチームにより加えられた，特に simplify = TRUE の全てのケース中で配列を返す，例えば combn(5,5) に対して.

References

Nijenhuis, A. and Wilf, H.S. (1978) *Combinatorial Algorithms for Computers and Calculators*; Academic Press, NY.

See Also

組み合わせの数の高速計算に対しては choose. 因子やベクトルの全ての組み合わせからデータフレームを作るには expand.grid.

Examples

```
combn(letters[1:4], 2)
(m <- combn(10, 5, min))   # 各組み合わせ中の最小値
mm <- combn(15, 6, function(x) matrix(x, 2, 3))
stopifnot(round(choose(10, 5)) == length(m),
          c(2,3, round(choose(15, 6))) == dim(mm))

## 点をエンコードする別の方法：
combn(c(1,1,1,1,2,2,2,3,3,4), 3, tabulate, nbins = 4)

## Compute support points and (scaled) probabilities for a
## 多変量超幾何分布(n = 3, N = c(4,3,2,1)) の確率関数に対する
## サポート点と(スケール化された)確率を計算する：
# table.mat(t(combn(c(1,1,1,1,2,2,2,3,3,4), 3, tabulate, nbins = 4)))

## 恒等性を確認する
for(n in 1:7)
 for(m in 0:n) stopifnot(is.array(cc <- combn(n, m)),
                         dim(cc) == c(m, choose(n, m)))
```

---

compareVersion 二つのパッケージのバージョン番号を比較する

Description

どちらが新しいか二つのパッケージのバージョン番号を比較する.

Usage

```
compareVersion(a, b)
```

Arguments

a, b	パッケージのバージョン番号を表現する文字列.

Details

R のパッケージバージョン数は整数 x, y そしてz に対する形式 x.y-z で， x 以降の成分はオプションで欠損する (その場合バージョン数は成分が存在するものよりも古い).

Value

もし番号が同じなら 0，もし b が最近ならば -1，もし a が最近ならば 1 (C 関数 strcmp と類似).

See Also

package_version, library, packageStatus.

Examples

```
compareVersion("1.0", "1.0-1")
compareVersion("7.2-0","7.1-12")
```

COMPILE　　*R* で使用するためにファイルをコンパイルする

Description

ソースファイルをコンパイルし，それから R CMD SHLIB や R CMD LINK を使う実行プログラムを使い共有オブジェクト中に集めるようにする．

Usage

```
R CMD COMPILE [options] srcfiles
```

Arguments

srcfiles	コンパイルされるソースファイルの名前のリスト．現在 C, C++, Objective C, Objective C++ そして Fortran がサポートされている；対応するファイルは拡張子 '.c', '.cc' (又は '.cpp'), '.m', '.mm' (又は '.M'), '.f' そして '.f90' か '.f95' をそれぞれ持たなければならない．
options	コンパイルに関連する設定，又はユーティリティの用法とバージョンに関する情報を得るためののリスト．

Details

R CMD SHLIB はファイルのコンパイルと共有オブジェクトへのリンクの双方を行える：これは C++, Fortran そして Objective C(++) ソースに渡される時どの実行時ライブラリが必要か知っているため，ソースファイルを R CMD SHLIB に渡すのがより信頼できる．

Ratfor はサポートされていない．もし Ratfor のソースコードがあれば，それを FORTRAN に変換する必要がある．（ある種の Solaris システムでは Ratfor と FORTRAN コードを混在することが出来る．）

Objective C と Objective C++ のサポートはオプションで，そしてもし R のコンフィギュア時に対応するコンパイラが利用できる時だけ動作する：これらの主な使用は OS X 上である．

コンパイルはパスを R のパブリック C/C++ ヘッダに含めるように手配する．

これはコードを共有オブジェクトに含めるのに適するようにコンパイルするので，PIC コードを生成する：これは時として実行プログラムの主コードとしては好ましくないかもしれない．

これは make に基づく機能であるので，もしより新しい対応する '.o' ファイルが存在すればソースファイルをコンパイルしない．

Note

ある種の R にバイナリディストリビューションは COMPILE を別個のバンドルに持つ，例えば R-devel RPM.

これは Windows では利用できない.

See Also

LINK, SHLIB, dyn.load; “R Administration and Installation” 中の節 “Customizing compilation under Unix” (R のソースツリーのサブディレクトリ ‘doc/manual’ を見よ).

---

contrib.url　　*CRAN* 風のレポジトリ中の適当なパスを見つける

Description

contrib.url はレポジトリ内部に適当なタイプ固有のパスを repos 中の各 URL に付け加える.

Usage

```
contrib.url(repos, type = getOption("pkgType"))
```

Arguments

repos	文字列ベクトル，使用するレポジトリの基本 URL.
type	文字列で，パッケージのタイプを指示する： install.packages を見よ.

Details

もし type = "both" ならばこれはソースレポジトリを使う.

Value

repos と同じ長さの文字列.

See Also

引数 repos に対して使われる最も普通の値である options("repos") を設定する setRepositories.

available.packages, download.packages, install.packages.

レポジトリをどのように設定するかについてはマニュアル ‘R Installation and Administration’.

count.fields 行毎の欄の数をカウントする

Description

count.fields は読み込まれた file の各行中の sep で分離された欄の数を数える.

Usage

```
count.fields(file, sep = "", quote = "\"'", skip = 0,
             blank.lines.skip = TRUE, comment.char = "#")
```

Arguments

file	ASCII データファイルの名前の文字列，又は connection で，必要なら開かれ，そしてもしそうなら関数呼び出しの最後に閉じられる.
sep	欄の分離文字．各行の値はこの文字で分離されている．既定では，任意の数の空白が欄を分離できる.
quote	引用文字のセット
skip	データの読み込みの前にスキップするデータファイルの行数.
blank.lines.skip	論理値：もし TRUE なら入力中の空白行は無視される.
comment.char	文字列：単一の文字か空文字列を含む長さ1の文字列ベクトル.

Details

これは read.table により使われていたが．今でもその関数でファイルを読み込む際の問題の発見に有用である.

コメントの処理については scan を見よ.

scan と両立して，count.fields は引用文字列が新行文字を含むことを許す．そうした場合開始行は NA と記録された欄のカウントを持ち，最終行は記録の最初からの全ての欄のカウントを含む.

Value

見つかった欄の数のベクトル.

See Also

read.table

Examples

```
cat("NAME", "1:John", "2:Paul", file = "foo", sep = "\n")
count.fields("foo", sep = ":")
unlink("foo")
```

create.post 電子メイルと投稿を準備する補助関数

Description

パッケージ管理者や R のメイリングリストに投稿するための電子メイルを準備する bug.report と help.request により使われる補助関数.

Usage

```
create.post(instructions = character(), description = "post",
            subject = "",
            method = getOption("mailer"),
            address = "the relevant mailing list",
            ccaddress = getOption("ccaddress", ""),
            filename = "R.post", info = character())
```

Arguments

instructions	テンプレート電子メイルの先頭に置かれる文字列.
description	文字列：メッセージに含まれる記述.
subject	電子メイルの主題. "mailx" メソッドを除きオプション.
method	投稿メソッド, "none", "mailto", "gnudoit", "ess" または(Unix のみ) "mailx". '詳細'節を見よ.
address	適用できる時受け取り手の電子メイルアドレス：電子メイルで送られるパッケージのバグ報告に対してはこれは既定でパッケージの管理者のアドレスになる (複数がリストされていれば最初のもの).
ccaddress	"mailx" と "mailto" メソッドを持つコピーのためのオプションの電子メイルアドレス. コピー無しには ccaddress = "" を使う.
filename	電子メイルを設定する (またはメソッドが "none" かメイルの送付が失敗した時それを保存する)ために使われるファイル名.
info	'please do not edit the information below' 行の下のテンプレート電子メイル中に含める情報の文字列ベクトル.

Details

これが何を行うかは method に依存する. 関数は先ず電子メール本体のテンプレートを作る.

none ファイルエディタ(file.edit を見よ)が指示とテンプレートの電子メイルにより開かれる. これが終わると電子メイルプログラム中に読み込んだりペイストする順位が出来た完全な電子メイルがファイル file 中にある.

mailto これは既定の電子メイルプログラムを編集や送付のためにテンプレート電子メイル(アドレス, Cc: アドレス, そして主題)を開く.

これは既定のメイラーが設定されているところで動作する (OS X と Windows では通例で, そして Unix 風システムでは xdg-open が利用出来てコンフィギュアされている時：もしこれが失敗すると R_BROWSER で設定されたブラウザを試みる).

これは'工場出荷時'既定メソッドである.

mailx (Unix 風システムのみ．) ファイルエディター(file.edit を見よ)が指示とテンプレート電子メイルと共に開かれる．これが終了した時，それは mailx の様な Unix のコマンドラインのメイルユーティリティを用いて与えられた宛先(そしてオプションで Cc: 宛名)に送付される．

gnudoit (X)emacs のメイルバッファが電子メイルの編集と送付のために開かれる：これは gnudoit プログラムが利用できることを必要とする．現在 subject は無視される．

ess テンプレート電子メイルの本体は stdout に送られる．

Value

不可視の NULL.

See Also

bug.report, help.request.

---

data　　データセット

Description

指定されたデータセットをロードする，または利用可能なデータセットをリストする．

Usage

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

Arguments

...	字義とおりの文字列又は名前.
list	文字列.
package	データセットを探すパッケージを与える文字列，又は NULL 既定では検索パス中の全てのパッケージが使われ，それから現在の作業ディレクトリの ‘data’ サブディレクトリ(もしあれば)が使われる.
lib.loc	R ライブラリのディレクトリ名の文字列，又は NULL．既定値 NULL は現在既知の全てのライブラリに対応する.
verbose	論理値．もし TRUE なら，追加の診断がプリントされる.
envir	データがロードされるべき environment.

Details

現在四種類のデータファイルの組織がサポートされている：

1. ‘.R’ 又は ‘.r’ で終わるファイルは source() され， R の作業ディレクトリが一時的にそれぞれのファイルを含むディレクトリに変更される．(data は **utils** パッケージが付加されていることを保証し，その場合それは utils::data 経由で実行される．)
2. ‘.RData’ 又は ‘.rda’ で終わるファイルは load() される.

3. '.tab', '.txt' または '.TXT' で終わるファイルは
   read.table(..., header = TRUE, as.is=FALSE) を用いて読み込まれ，従ってデータフレームになる．
4. '.csv' 又は '.CSV' で終わるファイルは
   read.table(..., header = TRUE, sep = ";", as.is=FALSE) を用いて読み込まれ，そしてやはりデータフレームになる．

もし複数のファイル名がマッチすれば，このリスト上の最初のものが使われる． (拡張子 '.txt', '.tab' 又は '.csv' のファイルは圧縮でき，追加の拡張子 '.gz', '.bz2' または '.xz' を持ったり持たなかったりする．)

ロードされるデータセットは文字列や名前のセットで指定するか，又は文字列ベクトルの list，又は双方で指定できる．

与えられたデータセットの各々に対して，最初の二つのタイプ ('.R' 又は '.r'，そして '.RData' 又は '.rda' ファイル) はロード環境中に幾つかの変数を作ることが出来，それらは全てデータセットとは異なる名前かもしれない．三つ目と四つ目は常にデータセットと同じ名前を持つ (無しの)単一の変数を作る結果になる．

もしデータセットが指定されないと，data は利用可能なデータセットをリストする．これは 'Meta' 中の新しいスタイルのデータ索引を眺め，もしこれが見つからないと，各指定ディレクトリ中の 'data' の中の旧式の '00Index' ファイルを眺め，そしてこれらのファイルをリストのために使う．'data' 領域が在るが索引がなければ，ロードのために利用可能なデータが計算されリストの中に含められ，そして警告が出る：そうしたパッケージは不完全である．利用可能なデータセットに関する情報はクラス "packageIQR" のオブジェクト中に返される．このクラスの構造は実験的である．データセットがそれらを検索するのに使われる引数とは異なる名前を持つ箇所では，索引はデータセット beaver1 は data(beaver) の呼び出しで検索できることを告げる beaver1 (beavers) の様な項目を持つ

もし lib.loc と package が共に NULL (既定値)ならば，データセットは現在ロードされている全てのパッケージ，それから現在の作業ディレクトリの 'data' ディレクトリ(もしあれば)中で探される．

もし lib.loc = NULL であるが package と文字列として指定されると，指定されたパッケージは先ずロードされたパッケージ，次いで既定のライブラリ中で探される(.libPaths を見よ)．

もし lib.loc が指定され(そして NULL でない) ならば，例えそれらがすでに他のライブラリからロードされていても，パッケージは指定されたライブラリ中で探される．

現在の作業ディレクトリの 'data' ディレクトリだけを見るには，package = character(0) と設定する(そして既定である lib.loc = NULL)．

Value

指定された全てのデータセットの文字列，又はもし何も指定されなければクラス "packageIQR" のオブジェクト中の全ての利用可能なデータセットの情報．

良い作法

data() 元々ユーザがパッケージからのデータセットを彼らの例で使うためにロードすることを意図していたので，それはデータセットを作業空間 .GlobalEnv 中にロードしていた．これは未使用の時大きなデータセットがメモリ中にあることを避けていた．この必要性はほとんど完全にデータセットの遅延ロードにより取って代わられた．

データセットを名前 (引用符無しで)指定する機能は便宜性のためである：プログラミング中ではデータセットは文字列(引用符と共に)で指定されるべきである．

data を envir 引数なしで関数中で使うことはほとんど常にオブジェクトをユーザの作業空間に置くという望ましくない副作用を持つ (そして実際，そこにある同名のオブジェクトを置き換えてしまう)．ほとんど常にオブジェクトを data(..., envir = environment()) により現在の評価環境に置くことがより望ましい．しかしながら，二つの別法が普通望ましく，両者はマニュアル 'Writing R Extensions' 中で解説されている．

- データセットに対して，データの遅延ロードを使うようにパッケージを設定する．
- 例えば関数中で計算に使われる参照テーブルであるシステムデータのオブジェクトに対しては，パッケージソース中でファイル 'R/sysdata.rda' を使うか，又はパッケージのインストール時に R コードでオブジェクトを作る．

時折重要な区別は二番目のアプローチはオブジェクトを名前空間に置くが最初のそれは置かないことである．従ってもし関数が mytable をパッケージからのオブジェクトと見做すなら，それはシステムデータであり二番目のアプローチを使うべきである．パッケージがある関数への既定引数として遅延ロードデータセットを使う普通ではないケースでは，それは :: を使って指定される必要がある，例えば survival::survexp.us.

Note

検索順序と '.R' ファイルがディレクトリを変更するという事実を利用できる．もし生のデータが 'mydata.txt' 中に保存されていると，'mydata.txt' を読み込むために 'mydata.R' をセットしそれから例えば transform を使い前処理する．例えば数値ベクトルを適当なラベルを持つ因子に変換できる．このようにして，'.R' ファイルは平文テキストの組織に対するメタデータ指示を効率的に含むことが出来る．

See Also

データセットに関するドキュメントを得る help，第二種のデータ('.rda')を作るためには save，典型的には最も効率的．

パッケージの 'data' ディレクトリ中の準備のためには 'Writing R Extensions'．

Examples

```
require(utils)
data()                          # 全ての利用可能なデータセットをリスト
try(data(package = "rpart") )   # rpart パッケージ中の全ての利用可能なデータセットをリスト
data(USArrests, "VADeaths")     # データセット 'USArrests' と 'VADeaths' をロード
## Not run: ## 又は
ds <- c("USArrests", "VADeaths"); data(list = ds)
## End(Not run)
help(USArrests)                 # データセット 'USArrests' に関する情報を与える
```

---

dataentry　　データ入力のためのスプレッドシートインタフェイス

Description

データ入力または編集のためのスプレッドシートエディター．

Usage

```
data.entry(..., Modes = NULL, Names = NULL)
dataentry(data, modes)
de(..., Modes = list(), Names = NULL)
```

Arguments

...	変数のリスト：現在これらは数値または文字列又はそうしたベクトルを含むリストでなければならない.
Modes	変数に対して使われるモード.
Names	変数に対して使われる名前.
data	数値そして/又は文字列ベクトルのリスト.
modes	変数の(幾つかの)モードを与える長さが最大 data のそれのリスト. list() が許される.

Details

データエントリーエディターはある種のプラットフォームと GUI の下でのみ利用できる．利用できれば，それは注意で解説されるような行列や変数の集まり (データフレームを含む)を視覚的に編集する手段を与える.

data.entry はスプレッドシート中の任意の変更が変数に反映されるという副作用を持つ．関数 de, de.ncols, de.setup そして de.restore はこれらの副作用を達成するのを助けるようにデザインされている．もしユーザが行列，例えば X，を渡せば，行列は dataentry が呼び出される前に列に分解される．それから返されるときには列は集められ一緒に結び付けられそして結果は変数 X に付値される．もしこの挙動が好ましくなければ dataentry を直接使う.

プリミティブ関数は dataentry である．これは可能性として異なった長さとモード (二つ目の引数)を取りそしてスプレッドシートをこれらの変数を列として開く．データエントリーウィンドウの列はスプレッドシートが閉じられる際にリスト中のベクトルとして返される.

de.ncols は data.entry への引数として与えられた列の数を数える．これはリスト，行列そしてベクトル中の列を数えようと試みる． de.setup は関数が返る際に列が再グループ化され正しい名前に再付値出来るように按配する．これは de.restore により処理される.

Value

de と dataentry はそれらの引数の編集された値を返す． data.entry は不可視で変数名のベクトルを返すがその主な価値はユーザの作業空間中でこれらの変数に新しいバージョンを付値する副作用にある.

リソース

データエントリーウィンドウはクラス R_dataentry の X リソースに応答する．リソース foreground, background そして geometry が利用される.

Note

データグリッドへのインタフェイスの詳細はプラットフォームと GUI により異なるかもしれない．以下の説明は Unix での X11 準拠の実装に適用される.

カーソルキーや任意のセル上で(左)マウスボタンをクリックすることでグリッド間を巡回できる．アクティブなセルは境界の矩形を太くすることでハイライトされる．右や下に移動するとグリッドは必要に応じてスクロールする：現在使用中の行や列には制約はない．

キーを使って巡回する別の方法がある． Return と(キーパッドの) Enter と LineFeed は全て下に移動する． Tab は右にそして Shift-Tab は左に移動する．Home はトップの左に移動する．

PageDown や Control-F は一頁下へ，そして PageUp や Control-B は一頁上へ移動する．End は最後に使われた列と最後の幾つかの行(任意の行中)を示す．

他の任意のキーの使用は現在選択されたセル上で編集過程を開始する： selected cell: そのセルから移動すると編集された値が入力され，他方で Esc は編集をキャンセルし先の値を復帰する．編集過程が始まるとセルはクリアされる．数値列(既定)中では適正な数値 (`-.eE` を含む)を作る文字だけが許され，不正な編集値(空白のような)はセル中に `NA` を入力する．直近に入力された値は BackSpace か Del(ete) キーで消去できる．限られた数の文字(現在 29)だけがセル中に入力でき，そしてもし必要なら文字列の最初か終わりだけが表示され，省略は > 又は < で指示される． (最初は編集時を除き表示される．)

セル中に最後に使われたセルよりも下の欄のセルに値を入力すると変数(もしあれば)を `NA` で拡大する(スクリーン上では表示されない)．

列名はそれらの中でクリックすることだけで選択できる．これは列のタイプ(現在実数か文字列)を選択するかそれらを変更するポップメニューを与える．タイプの変更は列の現在の内容を変換する (そして文字列から実数への変換は `NA` を生成するかもしれない．) もし名前の変更が選択されるとヘッダーセルは編集可能になる (そしてクリアされる)．全てのセルと同じく，値は他をクリックするか下への移動のための任意のキー (だけ)によりセルから移動することで入力される．

新しい欄はそれらの中に値を入力することで作られる (そして単に新しい名前を付値することによってではない)．列のモードは最初に入力された値から自動検出される：もしこれが適正な数であれば数値列を与える．未使用の列は無視されるので，

三列のグリッドの `var5` 中にデータを加えることは一つの追加の変数を加え，二つではない．

`Copy` ボタンは現在選択されたセルをコピーする： `paste` は最後にコピーされた値を現在のセルにコピーし，そして右クリックはセルを選択しそして値をその中にコピーする．当初値は空白で，そして空の値をペーストしようとしても何の効果もない．

Control-L は表示をリフレッシュし，現在の項目に欄幅がフィットするように再計算する．

既定モードでは列幅は各列の中身にフィットするように選択され，既定では各空列に対し10文字となる．オプション `de.cellwidth` を必要な固定幅 (文字数で)に設定することで固定幅行を指定できる． (可変幅に復帰するにはそれをゼロに設定する)．任意の欄の表示される幅は 600 ピクセル(そしてウィンドウ幅)に制限される．

See Also

`vi`, `edit`: `edit` はデータフレームの編集に `dataentry` を使う．

Examples

```
# data.entry を変数 x と y で呼び出す
## Not run: data.entry(x, y)
```

debugger 検死デバッグ

Description

評価環境(フレーム)をダンプしそれを調べる関数.

Usage

```
dump.frames(dumpto = "last.dump", to.file = FALSE)
debugger(dump = last.dump)
```

Arguments

dumpto	文字列. オブジェクト名又はダンプ先のファイル.
to.file	論理値. ダンプ先は R オブジェクトであるべきかファイルであるべきか?
dump	dump.frames で作られた R ダンプオブジェクト.

Details

検死デバッグを使うには, オプション error を dump.frames への呼び出しに設定する. 既定ではこれは作業空間中の R オブジェクト last.dump にダンプするが, ファイルにダンプするように設定できる (save への呼び出しにより作られるオブジェクトのダンプ). ダンプされたオブジェクトは geterrmessage が返すような呼び出しスタック, アクティブな環境そして最後のエラーメッセージを含む

ファイルにダンプする際は, dumpto はダンプされたオブジェクトの名前を与えそしてファイル名は '.rda' が付け加えられる.

クラス "dump.frames" のダンプオブジェクトは debugger の呼び出しで調査できる. これはエラーメッセージとそこから繰り返し選択される環境のリストを与える. 環境が選択された時は, それはコピーされコピーの内部から browser が呼び出される. オリジナルのフレーム中の全ての情報が利用可能なわけではないことを注意する, つまりまだ評価されていない予約と任意の ... 引数の中身である.

もし dump.frames がエラーハンドラーとしてインストールされていると, 実行は非対話的なセッション中でも継続する. どのようにダンプしそれから終了するかは例を見よ.

Value

不可視の NULL.

Note

sys.parent の様な関数とそしてクロージャに適用された environment は debugger の内部では正しく動作しない.

もし形式的引数の既定値を計算する際にエラーが起きると, デバッガーはその環境を調べようとする際に "再帰的な既定引数参照"を報告する.

勿論検死デバッグはもし R がダンプを生成し保存出来ない程損傷すると動作しない, 例えば作業スペースの外に出てしまうような場合.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

Browse プロンプトに対して可能な動作については browser.

error オプションの設定については options ; recover は debugger に類似の動作の対話的なデバッガーであるが，エラーが起きた直後用.

Examples

```
## Not run:
options(error = quote(dump.frames("testdump", TRUE)))

f <- function() {
    g <- function() stop("test dump.frames")
    g()
}
f()   # ファイル "testdump.rda" にダンプを生成
options(error = NULL)

## 恐らく別の R セッション中
load("testdump.rda")
debugger(testdump)
Available environments had calls:
1: f()
2: g()
3: stop("test dump.frames")

Enter an environment number, or 0 to exit
Selection: 1
Browsing in the environment with call:
f()
Called from: debugger.look(ind)
Browse[1]> ls()
[1] "g"
Browse[1]> g
function() stop("test dump.frames")
<environment: 759818>
Browse[1]>
Available environments had calls:
1: f()
2: g()
3: stop("test dump.frames")

Enter an environment number, or 0 to exit
Selection: 0

## 非対話的なセッションに対する可能な設定
options(error = quote({dump.frames(to.file = TRUE); q(status = 1)}))

## End(Not run)
```

demo *R* の機能のデモ

Description

demo は幾つかのデモ用の R スクリプトを実行するユーザフレンドリーなインタフェスである．

Usage

```
demo(topic, package = NULL, lib.loc = NULL,
     character.only = FALSE, verbose = getOption("verbose"),
     echo = TRUE, ask = getOption("demo.ask"),
     encoding = getOption("encoding"))
```

Arguments

topic	デモされるべきトピックで， character.only が FALSE (既定)か TRUE に応じて name か文字通りの文字列，又は文字列．もし省略されると，利用可能なトピックが表示される．
package	デモを探すパッケージを与える文字列ベクトル，または NULL．既定では検索パス上の全てのパッケージが使われる．
lib.loc	R ライブラリのディレクトリ名の文字列ベクトル，または NULL．既定値の NULL は現在既知の全てのライブラリに対応する．もし既定値が使われると，ロードされたパッケージがライブラリの前に検索される．
character.only	論理値；もし TRUE なら topic を文字列として使う．
verbose	論理値．もし TRUE ならば追加の診断がプリントされる．
echo	論理値．もし TRUE ならソース時に R 入力を表示する．
ask	論理値(または "default") で devAskNewPage(ask = TRUE) がデモコードからグラフィカルな出力が起こる前に呼び出されるべきかどうかを指示する．値 "default" (工場出荷時既定値)は echo == TRUE かどうか，そしてグラフィカルデバイスが対話的かどうかを問い合わせる．このパラメータは任意の現在開かれているデバイスとでもコードにより開かれる任意のデバイスに適用される．もしこれが TRUE と評価されそしてセッションが interactive なら，ユーザは開始するために RETURN を押すように求められる．
encoding	source を見よ．もしパッケージが宣言されたエンコーディングを持てばそれが優先される．

Details

トピックが与えられないと demo は利用可能なデモをリストする．対応する情報はクラス "packageIQR" のオブジェクト中に返される．

See Also

demo により呼び出される．source と devAskNewPage．

Examples

```
demo() # 付加されたパッケージに対して

## 全ての利用可能なデモ：
demo(package = .packages(all.available = TRUE))


## デモの表示，ページ間でポース
demo(lm.glm, package = "stats", ask = TRUE)

## ポーズ無しで表示
demo(lm.glm, package = "stats", ask = FALSE)


## Not run:
 ch <- "scoping"
 demo(ch, character = TRUE)

## End(Not run)

## デモの位置を見つける
system.file("demo", "lm.glm.R", package = "stats")
```

---

download.file　　インタネットからファイルをダウンロードする

Description

この関数はインタネットからファイルをダウンロードするのに使える．

Usage

```
download.file(url, destfile, method, quiet = FALSE, mode = "w",
              cacheOK = TRUE,
              extra = getOption("download.file.extra"))
```

Arguments

url	ダウンロードすべきリソースの URL の名前の文字列．
destfile	ダウンロードされたファイルをどこに保存するかを示す名前の文字列．チルド展開が実行される．
method	ファイルのダウンロードのために使われるメソッド．現在のダウンロードメソッドは "internal", "wininet" (Windows のみ) "libcurl", "wget" そして "curl" で，そして値 "auto" がある：'詳細'と'注意'節を見よ． メソッドは又オプション "download.file.method" で設定できる：options() を見よ．
quiet	もし TRUE ならステータスメッセージ(もしあれば)，そして進行度バーをを抑制する．
mode	文字列．ファイルを書き込むモード．有用な値は "w", "wb" (バイナリ), "a" (追加)そして "ab"．"internal" メソッドだけに使われる．

cacheOK 論理値．サーバー側のキャッシュ値を受け入れるか?

extra "wget" と "curl" メソッドに対する追加のコマンドライン引数の文字列．

Details

関数 download.file はインターネットから url で記述された単一のファイルをダウンロードしそれを destfile に保存することが出来る． url は 'http://', 'https://', 'ftp://' または 'file://' といったスキームと共に使う必要がある．

もし method = "auto" が選ばれると(既定値)，Unix 風システムではメソッド "libcurl" が 'https://' と 'ftps://' URL に対して選ばれ，そして他のスキームでは "internal" メソッドが選ばれる．

メソッド "libcurl" はその名前のライブラリ (http://curl.haxx.se/libcurl/) を使う．これは 'https://' と(普通) 'ftps://' URL への (非ブロッキング)アクセスを提供する．同時ダウンロードに対するサポートがあるので， url と destfile は一つ以上の同じ長さの文字列ベクトルであって良い．単一の URL と quiet = FALSE に対しては対話的使用では進行度バーが表示される．

メソッド "wget" と "curl" に対しては method が与えるツールへのシステムコールが行われ，対応するプログラムが自分のシステムにインストールされそして実行ファイルに対する検索パス中になければならない．これらは完了するまで R プロセス上の他の全ての活動をブロックする： GUI が応答しなくなるかもしれない．

cacheOK = FALSE は 'http://' と 'https://' URL に対して有用である：これは中間的なキャッシュではなくサイトからコピーを直接得ようと試みる．これは available.packages により使われる．

"libcurl" と "wget" メソッドは 'http://' と 'https://' リダイレクションに従う： "internal" メソッドは違う． (メソッド "curl" に対しては引数 extra = "-L" を使う．wget 中のリダイレクションを無効にするためには extra = "--max-redirect=0" を使う．) ("wininet" メソッドはある種のリダイレクションをサポートするが全てではない．)

'https://' URL は "internal" メソッドではサポートされていないが "libcurl" メソッドと Windows では "wininet" メソッドではサポートされることを注意する．

Windows では 'file://' URL がどのように解釈されるかについては url を見よ． "internal" と "wininet" メソッドは 'file://' URL をパーセントデコードしないが， "libcurl" と "curl" メソッドはする：メソッド "wget" はそれらをサポートしない．

殆どのメソッドは URL 中の空白のような特殊文字をパーセントデコードしないが(URLencode を見よ)，しかし "wininet" メソッドはするように見える．

残りの詳細は "internal", "wininet" そして "libcurl" メソッドだけに適用される．

転送の多くの部分に対するタイムアウトは既定値が 60 である timeout で設定できる．

転送の間に提供される詳細のレベルは quiet 引数と internet.info オプションで設定できる：詳細はプラットフォームとスキームに依存する． "internal" メソッドに対してはオプション internet.info を 0 に設定すると全ての利用可能な詳細をあたえる． 2 (既定値)の使用は深刻なメッセージだけを与え，そして 3 またはそれ以上のメッセージを抑制する． "libcurl" メソッドに対しては 2 未満のオプションの値は冗長な出力を与える．

進行度バーは転送を追跡する．もしファイル長が既知ならば，等号サインは完了した転送の 2% を表す：さもなければドットは 10Kb を表す．

バイナリファイルをダウンロードするために書かれたコードは mode = "wb" を使わなければならないが，テキスト転送により引き起こされる問題は Windows でだけ見られる．

Value

(不可視の)整数コード，0 は成功でゼロでなければ失敗． "wget" と "curl" メソッドに対してはこれは外部プログラムにより返されるステータスコードである． "internal" メソッドは 1 を返すことができるが，ほとんどのケースでエラーを与える．

プロキシの設定

次の二つのパラグラフは内部コードにだけ適用される．

プロキシは環境変数により指定できる． no_proxy を * に設定すると試みられる全てのプロキシを停止する． さもなければ http_proxy または ftp_proxy (またはそれに失敗すると，全ての大文字バージョン) が参考にされ，もし空でなければプロキシサイトとして使われる． FTP 転送に対してはプロキシ上のユーザ名とパスワードは ftp_proxy_user と ftp_proxy_password により指定できる． http_proxy の形式は http://proxy.dom.com/ または http://proxy.dom.com:8080/ でなければならず，ここでポートの既定値は 80 で後続のスラッシュは省略可能である． ftp_proxy に対しては形式 ftp://proxy.dom.com:3128/ を使う，ここで既定ポートは 21 である．これらの環境変数はダウンロードコードが最初に使われる前に設定されなければならない：それらは後で Sys.setenv により変更できない．

ユーザ名とパスワードは HTTP プロキシに対しては環境変数 http_proxy_user により形式 user:passwd で設定できる．別法として， http_proxy は wget との互換性から形式 form http://user:pass@proxy.dom.com:8080/ であって良い． HTTP/1.0 基本認証スキームだけがサポートされている．

ほぼ同じスキームが method = "libcurl" によりサポートされており， no_proxy, http_proxy そして ftp_proxy を含み，そして最後の二つに対しては [user:password@]machine[:port] の内容はオプションである，ここで括弧内の部分はオプションである．詳細は http://curl.haxx.se/libcurl/c/libcurl-tutorial.html を見よ．

安全な URL

'https://' と 'ftps://' URL にアクセスするメソッドはそれらの安全性の検証を試みるべきである．これらは普通 OS によりインストールされたルート認証を使ってなされる (これらが更新されること無く取り除かれている例を見ているものの)．これ以上の情報は http://curl.haxx.se/docs/sslcerts.html を見よ．

これは Windows では method = "libcurl" に対して問題であり，そこでは OS が適当な CA 認証バンドルを提供しないので， Windows では既定では認証は検証されない．検証を有効にするためには，環境変数 CURL_CA_BUNDLE を認証バンドルファイルのパスに設定する．これは普通 'ca-bundle.crt' または 'curl-ca-bundle.crt' という名前である． (これは普通 R のバイナリインストールでは行われ， '*R_HOME*/etc/curl-ca-bundle.crt' をインストールしもし環境変数がすでに設定されていなければそれが CURL_CA_BUNDLE を指すように設定する．) 更新された認証バンドルに対しては， http://curl.haxx.se/docs/sslcerts.html を見よ．現在コピーを https://raw.githubusercontent.com/bagder/ca-bundle/master/ca-bundle.crt からダウンロードでき，そして CURL_CA_BUNDLE をダウンロードされたファイルのフルパスに設定する．

R が用いるルート認証はブラウザが使うものと同じかもしれないし異なるかもしれないことを注意する，そして実際異なったブラウザは異なった認証バンドルを使うかもしれない (典型的にはそれ自体のものかシステムのものを選択するビルドオプションがある)．

FTP サイト

'ftp:' URL は幾つかの変種を持つ FTP プロトコルを用いてアクセスされる． 'active' と '(拡張)パッシブ'モードの間には一つの違いがある：どちらを使うかはクライエントにより選択される． "internal" と "libcurl" メソッドはパッシブモードを使い，これはブラウザによりほとんど普遍的に使われる． R 3.2.3 以前では "wininet" メソッドはアクティブモードを使った：それは最初パッシブそれからアクティブモードを使ったからである．

Note

2GB 以上のファイルは R の 64 ビットビルドでサポートされている； 32 ビットビルドでは打ち切られるかもしれない．

メソッド "wget" と "curl" は主に歴史的な互換性のためにあるが，"libcurl" や "wininet" メソッドでサポートされていない多くの機能を提供する．

メソッド "wget" はもし正しい値が wget にｍ対するコンフィギュレーションファイル中に保管されていればユーザ/パスワード認証を必要とするプロキシファイアウォールと共に使うことが出来る．

wget (http://www.gnu.org/software/wget/) は Unix 風システムでは普通インストールされている(が OS X では無い)． Windows バイナリは Cygwin, gnuwin32 そしてその他から入手できる．

curl (http://curl.haxx.se/) は OS X と Unix 風システムに普通インストールされている． Windows バイナリはその URL で入手できる．

See Also

幾つかのメソッドにより使われるオプション HTTPUserAgent, timeout そして internet.info を設定するには options.

URL からデータを読み込むより精選された方法は url.

アプリケーションに対しては url.show, available.packages, download.packages.

貢献パッケージ **RCurl** は URL からのダウンロードに対するより広範な機能を提供する．

---

download.packages　*CRAN* 風のレポジトリからパッケージをダウンロードする

Description

これらの関数はインストールされたパッケージのバージョン数とレポジトリ上で利用可能な最新のバージョン数を比較し，旧式のパッケージをその場で更新する．

Usage

```
download.packages(pkgs, destdir, available = NULL,
                  repos = getOption("repos"),
                  contriburl = contrib.url(repos, type),
                  method, type = getOption("pkgType"), ...)
```

Arguments

pkgs	その利用可能な最新のバージョンがレポジトリからダウンロードされるべきパッケージ名の文字列ベクトル.
destdir	ダウンロードされたパッケージが保存されるディレクトリ.
available	レポジトリで利用可能なパッケージをリストする available.packages が返すようなオブジェクト，または available.packages への内部呼び出しを行う NULL.
repos	文字列ベクトル，使われるレポジトリの基本 URL，つまり CRAN マスター "https://cran.r-project.org" やその Statlib ミラー "http://lib.stat.cmu.edu/R/CRAN" のような URL.
contriburl	レポジトリの contrib セクションの URL. この引数は使用するレポジトリミラーが不完全な時だけ使う，例えば CD に 'contrib' セクションだけを焼き付けた場合. 引数 repos を上書きする.
method	ダウンロードのメソッド， download.file を見よ.
type	文字列で，パッケージのタイプを指示する： install.packages を見よ.
...	download.file に渡される追加引数.

Details

download.packages はパッケージ名と目標ディレクトリ名のリストを取り，最新バージョンをダウンロードしそれらを destdir に保存する. もし利用可能なパッケージのリストが引数として与えられないと，それはレポジトリから取得される. もしレポジトリがローカルならば，つまり URL が "file:" で始まれば，パッケージはダウンロードされずそのまま使われる. "file:" と "file:///" の双方がファイルパスの接頭辞として許される. 後者は URL に対してだけ使う：それらの解釈については url を見よ. ('file://' URL の他の形式はサポートされていない.)

Value

成功裡にダウンロードされたパッケージの名前と目標ファイル名の二列の行列. もしパッケージが利用できないかダウンロードに問題があれば適当な警告が与えられる.

See Also

available.packages, contrib.url.

主な用途は install.packages が行う.

プロキシの処理とファイル転送をモニターする他のオプションについては download.file.

レポジトリをどのように設定するかはマニュアル 'R Installation and Administration'.

edit テキストエディターを起動する

Description

R オブジェクトに対してテキストエディターを起動する.

Usage

```
## Default S3 method:
edit(name = NULL, file = "", title = NULL,
     editor = getOption("editor"), ...)

vi(name = NULL, file = "")
emacs(name = NULL, file = "")
pico(name = NULL, file = "")
xemacs(name = NULL, file = "")
xedit(name = NULL, file = "")
```

Arguments

name	編集したい名前付きオブジェクト. 名前が欠損していれば file で指定されたファイルが編集のために開かれる.
file	編集済みバージョンを書き込むファイルの名前を与える文字列.
title	編集されるオブジェクトのディスプレイ名.
editor	普通使用したいテキストエディターの名前である文字列. Unix では既定はもしどちらかが設定されていれば環境変数 EDITOR または VISUAL から設定され, さもなければ vi が使われる. Windows では既定値は "internal" で, スクリプトエディター. OS X GUI では引数は無視されドキュメントエディターが常に使われる. editor は又関数でもよく, その場合それは引数 name, file そして title で呼び出される. そうした関数は全ての期待される機能を独立に実装している必要があることを注意する.
...	メソッドに/から渡される追加引数.

Details

edit は editor で指定されたエディターを編集されるべきオブジェクト name で起動する. これは総称的関数で, 現在既定メソッドとデータフレームと行列用のメソッドを持つ.

data.entry はデータの編集に使うことが出来, data.entry が利用可能なシステム上で行列とデータフレームを編集するために edit により使われる.

edit は name と呼ばれるオブジェクトを変更しないことを理解することが重要である. オブジェクト name に変更を適用するには edit の結果を name に付値しなければならない. (もしオブジェクトへの永続的な変更を行いたければ fix を試みよ.)

形式 edit(name) では edit は name を一時ファイルに逆構文解析し, そしてエディター editor をそのファイルに起動する. エディターの終了は file を構文解析しそれを返す. 構文解析中にエラー, 恐らく不正なシンタックスに起因する, が起きれば如何なる値も

返されない．引数無しで edit() を呼びだせば更なる編集のために一時ファイルを再オープンする結果になる．

逆構文解析は完全ではなく，そして編集後に再構成されたオブジェクトは逆構文解析されたそれと微妙な点で異なる可能性がある： dput と .deparseOpts を見よ．(使用される逆構文解析オプションは dump に対する既定値と同じである．) 関数の編集はその環境を保存する．データフレームや行列を編集する際に起こり得る更なる変更については edit.data.frame を見よ．

現在 Windows の内部エディターだけが title オプションを使用する：それは与えられた名前をウィンドウのヘッダーに表示する．

Note

関数 vi, emacs, pico, xemacs, xedit は対応するエディターが利用可能でパス上にあることに依存している．これはシステム依存である．

See Also

edit.data.frame, data.entry, fix.

Examples

```
## Not run:
# 関数 mean にたいして xedit を使い変更を付値する
mean <- edit(mean, editor = "xedit")

# 関数 mean にたいして vi を使い結果をファイル mean.out に書き込む
vi(mean, file = "mean.out")

## End(Not run)
```

---

edit.data.frame　　データフレームと行列を編集する

Description

データフレームか行列の内容にデータエディタを使う．

Usage

```
## S3 method for class 'data.frame'
edit(name, factor.mode = c("character", "numeric"),
     edit.row.names = any(row.names(name) != 1:nrow(name)), ...)

## S3 method for class 'matrix'
edit(name, edit.row.names = !is.null(dn[[1]]), ...)
```

Arguments

name	データフレーム又は(数値，論理値又は文字列)行列.
factor.mode	データフレーム中で因子をどのように扱うか (整数としてか又は文字列水準としてか). 短縮化可能.
edit.row.names	論理値. 行名(もし存在すれば)を別個の編集可能な列として表示するか? これを NULL の行名を持つ行列に使おうとするとエラーになる.
...	メソッドに/から渡される追加の引数.

Details

現在の所これは数値，論理値又は文字列ベクトルを含む単純なデータフレームか数値，論理値又は文字列ベクトル行列に対してだけ動作する．任意の他の行列モードはエラーを与え，そして行列がクラスを持てばエラーになる(それは捨て去られる).

データフレームの列は入力で数値(is.numeric の意味で)，論理値又は因子でない限り *character* に強制変換される．クラスが捨て去られる際は警告が与えられる．特殊文字(タブ，プリントできない ASCII 等)はエスケープ列として表示される.

因子列はスプレッドシート中で数値ベクトル (データエントリーに対してより相応しい)又は文字列ベクトル (ブラウザに対してより良い)として表示される．編集後，ベクトルは同じ長さになるように NA で埋められ，そして因子属性が回復される．因子水準のセットは数値モードで編集することで変更できない；不正な水準は NA に変更されそして警告が出される．もし新しい因子水準が文字列モードで導入されると，それらは水準のリストの最後にそれらが登場した順序で加えられる.

データフレーム中の数理と因子列を交換するために列のモードを選択するのに data-editor の機能を使うことが出来る．数値行列中の任意の列を文字列に変更することは結果を文字列行列に強制変換する．論理値列のモードの変更はサポートされていない.

データフレームに対しては，行名はもし edit.row.names = FALSE で行数が未変更ならばオリジナルのオブジェクトから，そしてもし edit.row.names = TRUE で重複がなければ編集された出力から取られる．(もし row.names 列が不完全ならば，それは row223 の様な項目で拡張される．) 全ての他のケースでは行名は seq(length = nrows) で置き換えられる.

行列に対しては，必要なら列名が付け加えられる(形式 col7 の)．行名はもし edit.row.names = FALSE で行数が未変更ならばオリジナルのオブジェクトから取られ (さもなければ NULL)，そしてもし edit.row.names = TRUE ならば編集されたオブジェクトから取られる．(もし row.names 列が不完全ならば，それは row223 のような項目で拡大される．)

行列やデータフレームの編集は行名と列名を除く全ての属性を失う.

Value

編集されたデータフレームと行列.

Note

fix(dataframe) はこの関数を呼び出すことでその場での編集を行う.

もしデータエディターが利用できなければ，オブジェクトのダンプが既定メソッド edit を用いて編集のために提供される.

現在データエディターは 65535 行に制限されている.

Author(s)

Peter Dalgaard

See Also

data.entry, edit

Examples

```
## Not run:
edit(InsectSprays)
edit(InsectSprays, factor.mode = "numeric")

## End(Not run)
```

---

example　　オンラインヘルプの実例節を実行する

Description

R のオンラインヘルプのトピック topic の **Examples** 部分の全ての R コードを実行するが， dontrun, dontshow そして donttest は除外されるかもしれない，下の‘詳細’節を見よ．

Usage

```
example(topic, package = NULL, lib.loc = NULL,
        character.only = FALSE, give.lines = FALSE, local = FALSE,
        echo = TRUE, verbose = getOption("verbose"),
        setRNG = FALSE, ask = getOption("example.ask"),
        prompt.prefix = abbreviate(topic, 6),
        run.dontrun = FALSE, run.donttest = interactive())
```

Arguments

topic	名前か文字通りの文字列：その実例が実行されるべきオンライン help トピック．
package	トピックを探すパッケージ名を与える文字列ベクトル，又は NULL (既定)で，その時は search パス上の全てのパッケージが使われる．
lib.loc	R ライブラリのディレクトリ名の文字列，または NULL．既定値の NULL は現在既知の全てのライブラリに対応する．もし既定値が使われると，ロードされたパッケージがライブラリの前に検索される．
character.only	topic は文字列であると仮定すべきかどうかを指示する論理値．
give.lines	論理値：もし真なら実例のソースコードの行が文字列として返される．
local	論理値：もし TRUE なら局所的に評価し，もし FALSE ならば作業空間中で評価する．
echo	論理値；もし TRUE ならば，ソース時に R 入力を表示する．
verbose	論理値；もし TRUE なら実例を実行時により多くを表示する．

setRNG 論理値又は表現式；もし FALSE ならば，乱数の状態が保存され，それから指定された状態に初期化され，実例が実行され(保存された)状態が復帰される． setRNG = TRUE はパッケージの実例に対して R CMD check と同じ状態を設定する．これは現在次と同値である
setRNG = {RNGkind("default", "default"); set.seed(1)}

ask グラフィカル出力が実例コードから表示される前に devAskNewPage(ask = TRUE) が呼び出されるべきかどうかを指示する論理値 (または "default")．値 "default" (工場出荷値)はもし echo == TRUE でグラフィカルデバイスが対話的と思われるときは尋ねることを意味する．このパラメータは任意の現在開かれているデバイスと実例コードで開かれる任意のデバイスの双方に適用される．

prompt.prefix 文字列；もし echo = TRUE ならば使われるプロンプトの接頭辞．

run.dontrun \dontrun を無視すべきことを指示する論理値．

run.donttest \donttest を無視すべきことを指示する論理値．

Details

もし lib.loc が指定されていないと，パッケージがすでにロードされたもの，そしてそれから .libPaths() で与えられたライブラリ中で探される．もし lib.loc が指定されると，例えそれらがすでに別のライブラリからロードされていても，パッケージは指定されたライブラリ中だけで探される．捜索はトピックに対するヘルプを持つ最初に見つかったパッケジで停止する．

実例を実行する前にパッケージのロードが試みられるが，これは別の位置からロードされたパッケージを置き換えない．

もし local = TRUE ならばオブジェクトは作業空間中に作られず，従って example が完了した後に吟味することは出来ない：一方それらは作業空間中の同名のオブジェクトを上書きできない．

マニュアル *Writing* R *Extensions* 中で解説されているように，ヘルプページの作者は次のような除外規則で実例の一部をマークアップすることが出来る

dontrun 実行されるべきでは無いコードを囲む．

dontshow ヘルプページで不可視のコードを囲むが，パッケージのチェック用ツールと example() 関数では共に実行される．これは以前は testonly であり，そしてこの形式は依然受け入れられる．

donttest 典型的には実行されるべきであるが，パッケージのチェック中は実行されないコードを囲む．既定の run.donttest = interactive() は \donttest セクションを適当にスキップするため他のヘルプページ中の実例を使う．

Value

give.lines が真でない限り最後に評価された表現式の値，もし真であれば character ベクトルが返される．

Author(s)

Martin Maechler とその他

See Also

demo

Examples

```
example(InsectSprays)
## 標準パッケージ 'stats' に使用を強制：
example("smooth", package = "stats", lib.loc = .Library)

## R CMD check が実行される際のように実例実行前に RNG を設定：

r1 <- example(quantile, setRNG = TRUE)
x1 <- rnorm(1)
u <- runif(1)
## 同値な乱数
r2 <- example(quantile, setRNG = TRUE)
x2 <- rnorm(1)
stopifnot(identical(r1, r2))
## しかし x1 と x2 は example() 前の RNG 状態が異なり
## 回復されるため異なる!
x1; x2

## 実例コードを吟味：
## "lm...()" 関数の実例はどれくらい大きいか?
lmex <- sapply(apropos("^lm", mode = "function"),
               example, character.only = TRUE, give.lines = TRUE)
sapply(lmex, length)
```

---

file.edit 一つ又はそれ以上のファイルを編集する

Description

エディターで一つ又はそれ以上のファイルを編集する.

Usage

```
file.edit(..., title = file, editor = getOption("editor"),
          fileEncoding = "")
```

Arguments

...	表示されるファイルの名前を含む一つまたはそれ以上の文字列ベクトル. チルド展開がされる：path.expand を見よ.
title	エディター中で使われるタイトル；既定値はファイル名.
editor	使われるテキストエディター. '詳細'節を見よ.
fileEncoding	ファイルに対して仮定されるエンコーディング：既定ではネイティブのエンコーディングが仮定される： file に対するヘルプの 'エンコーディング'節を見よ.

Details

この関数の挙動は非常にシステム依存である. Unix では現在ファイルは一時にひとつだけ開くことが出来る； Windows では内部エディターは複数のファイルを開くことを許すが, 同時に 50 の編集ウィンドウの限界がある.

Windows では title 引数はウィンドウのキャプションに使われ，現在他のプラットフォームでは無視される．

ファイルをネイティブのエンコーディングに変換する際のエラーは関数の失敗を招く．

editor に対する既定値はシステム依存である． Windows では既定値はスクリプトエディターの "internal" であり， OS X GUI ではドキュメントエディターが editor の値にかかわらず使われる． Unix では既定値はもし設定されていれば環境変数 EDITOR または VISUAL から設定され，さもなければ vi が使われる．

See Also

files, file.show, edit, fix,

Examples

```
## Not run:
# 二つの R スクリプトを開く
file.edit("script1.R", "script2.R")

## End(Not run)
```

---

file_test　　ファイルに対するシェル風のテスト

Description

シェル風のファイルテストに対するユーティリティ．

Usage

```
file_test(op, x, y)
```

Arguments

op	実行されるテストを指示する文字列． 単項テスト(x だけが使われる)は "-f" (存在とディレクトリではない)， "-d" (存在とディレクトリ)そして "-x" (ファイルとして実行可能またはディレクトリとして検索可能)． 二項テストは "-nt" (厳密により新しい，修正日付を用いる) そして "-ot" (厳密により古い)：どちらの場合もテストは両方のファイルが存在しなければ失敗する．
x, y	ファイルパスを与える文字列ベクトル．

Details

ここで'存在'とはファイルシステム上にありシステムコール stat (または 64 ビット拡張)でアクセスできることを意味する – Unix 風システムではこれはファイルに至る全てのディレクトリに対する実行許可を必要とするが，ファイル自体に対する許可は不要である．

Windows での "-x" の意味は file.access を見よ．

See Also

存在(ある種のシステムでは test -e)テストだけするがディレクトリでは無いことのテストはしない file.exists.

file.path, file.info

Examples

```
dir <- file.path(R.home(), "library", "stats")
file_test("-d", dir)
file_test("-nt", file.path(dir, "R"), file.path(dir, "demo"))
```

---

findLineNum — ソースコードの行の位置を見つける，またはそこにブレーク点を設置する.

Description

これらの関数は特定のソースコード行を含むオブジェクト位置を keep.source = TRUE で構文解析された際に保存された情報を使い見つける.

Usage

```
findLineNum(srcfile, line, nameonly = TRUE,
            envir = parent.frame(), lastenv)

setBreakpoint(srcfile, line, nameonly = TRUE,
              envir = parent.frame(), lastenv, verbose = TRUE,
              tracer, print = FALSE, clear = FALSE, ...)
```

Arguments

srcfile	ソースコードを含むファイル名.
line	ファイル中の行数. これを指定する別の方法は詳細を見よ.
nameonly	もし TRUE (既定値)ならば， basename(srcfile) へのマッチだけを要求し，フルパスではない.
envir	関数オブジェクトに対してどこから探索を始めるか?
lastenv	どこで停止するか? 詳細を見よ.
verbose	どこにブレークポイントが設定されたかについての情報をプリントすべきか?
tracer	trace に渡されるオプションの tracer 関数. 既定では browser の呼び出しが挿入される.
print	trace に渡される print 引数.
clear	もし TRUE なら trace では無く untrace を呼び出す.
...	trace に渡される追加引数.

Details

findLineNum 関数は環境 envir，その親，祖父母環境から lastenv まで全てのオブジェクトを探す．

既定では lastenv はもし envir が指定されていないと大局的環境に，そしてもし envir が指定されるとルート環境になる．(最初の既定値は非常に速い傾向があり，そして普通 S4 メソッド以外の全てのユーザコードを見つける；二番目は全ての付加システムライブラリを検索するため非常に遅い．)

便宜性のため envir は間接的に指定されるかもしれない；もしそれが環境でないと environment(envir) で置き換えられる．

setBreakpoint は単純な trace と untrace のラッパである．それは findLineNum で見つかる位置にブレークポイントを設定したりクリアしたりする．

srcfile は通常文字列として入力されたファイル名であるが，"srcfile" オブジェクトかもしれない，又は "filename.R#nn" の様な接尾辞かもしれず，その場合は数 nn は line に対する既定値として使われる．

trace のマンページの引数 where の解説の様に，R のパッケージシステムはパッケージ中の関数の複数のコピーを含むかもしれない複雑なスキームを使う．ユーザは典型的に検索パス上の公開されたものを見るが，一方ではパッケージ中のコードはパッケージの名前空間中の公的なものを見る．もし envir をパッケージ中の関数の環境に設定すると，既定では findLineNum は両方のバージョンを見つけ，そして setBreakpoint は両者中にブレークポイントを設定する．(これは lastenv を使って制御できる；例えば，envir = environment(foo), lastenv = globalenv() は検索が公的なコピーを見る前に停止するために私的なコピーだけを見つける．)

バージョン 4 の S メソッドは又見つけるのが幾らかトリッキーである．それらは総称的関数と共に保存されており，**base** や他のパッケージ中にあるかもしれないので，普通それらを見つけるためには lastenv = emptyenv() を必要とする．ある場合にはそれらを保管するためには R はある種の変換を行いそして findLineNum はオリジナルのコードを見つけられないかもしれない．プリミティブ総称的関数に対するメソッドのような多くの特殊例は未だサポートされていない．

Value

findLineNum は位置情報を含むオブジェクトのリストを返す．それらに対する print メソッドが定義されている．

setBreakpoint は有用な返り値を持たない；それは trace または untrace を呼び出すという副作用のために呼び出される．

Author(s)

Duncan Murdoch

See Also

trace

Examples

```
## Not run:
# ファイル mysource.R の第100行にどの関数が定義されているかを探す:
findLineNum("mysource.R#100")

# その関数の両方のコピーに中断点を設定する，一つが myfunction と
```

```
# 同じ名前空間中にそしてもう一つが検索パス上にあると仮定する
setBreakpoint("mysource.R#100", envir = myfunction)

## End(Not run)
```

---

fix　　オブジェクトをフィックスする

Description

fix は x に対して edit を起動し，それからユーザの作業関数中に新しい(編集された) x のバージョンを付値する．

Usage

```
fix(x, ...)
```

Arguments

x	R オブジェクトの名前で，名前か文字列として．
...	エディタに渡される引数：edit を見よ．

Details

x で提供される名前は R オブジェクトとして存在する必要はなく，その場合引数無しで空の本体を持つ関数が編集用に提供される．

R オブジェクトの編集は自明ではないような仕方でそれを変更するかもしれない： edit の下のコメントを見よ．データフレームや行列の編集時に起こりえる変更については edit.data.frame を見よ．

See Also

edit, edit.data.frame

Examples

```
## Not run:
 ## 'my.fun' がユーザ定義関数と仮定する：
 fix(my.fun)
 ## 今や my.fun が変更される
 ## 又，
 fix(my.data.frame) # データエディタを呼び出す
 fix(my.data.frame, factor.mode="char") # ... の使用

## End(Not run)
```

flush.console 出力をコンソールにフラッシュする

Description

コンソールベースの R を除いてこれは何もしない． OS X と Windows GUI では，例え出力のバッファリングが実行中でもコンソールの出力表示が最新であることを保証する．

Usage

```
flush.console()
```

format 順序無し又順序付けされたリストをフォーマットする

Description

順序無し(itemize)又順序付けされた(enumerate)リストをフォーマットする．

Usage

```
formatUL(x, label = "*", offset = 0,
         width = 0.9 * getOption("width"))
formatOL(x, type = "arabic", offset = 0, start = 1,
         width = 0.9 * getOption("width"))
```

Arguments

x	リスト項目文字列ベクトル．
label	項目のラベル付けのための文字列．
offset	リストのオフセット(インデント量)を与える非負整数．
width	出力中の行の折り返しに対する目標列を与える正整数．
type	順序付きリスト中のラベルの‘タイプ’を指定する文字列．もし "arabic" (既定)ならアラブ数字が使われる． "Alph" か "alph" に対しては，単一の大文字か小文字が用いられる(この場合，最後の項目の数は 26 を超えてはならない．最後に， "Roman" または "roman" に対してはラベルは大・小文字のローマ数字(最後の項目数は最大 3899). type は上のユニークな短縮形として与えることが出来る，またはそれぞれ HTML スタイルのトークン "1" (アラビア数字)， "A"/"a" (アルファベット)，または "I"/"i" (ローマ数字).
start	順序付きリスト中の最初の項目数を指定する正整数．

Value

フォーマットされた項目を持つ文字列ベクトル．

See Also

description リストに対しては formatDL

Examples

```
## 簡単なレシピ.
x <- c("Mix dry ingredients thoroughly.",
       "Pour in wet ingredients.",
       "Mix for 10 minutes.",
       "Bake for one hour at 300 degrees.")
## 順序無しリストとしてフォーマットし出力する.
writeLines(formatUL(x))
## 順序付きリストとしてフォーマットし出力する.
writeLines(formatOL(x))
## 小文字のローマ数字を使った順序付きリスト.
writeLines(formatOL(x, type = "i"))
## 大文字のローマ文字とある種のオフセットを使った順序付きリスト.
writeLines(formatOL(x, type = "A", offset = 5))
```

---

getAnywhere　　*R* オブジェクトを回収する，名前空間からのものを含む

Description

これらの関数はそれらの引数にマッチする全てのオブジェクトを，検索パス上で可視か，S3 メソッドとして登録されているか，または名前空間中に在るが移出されていない，かにかかわらず特定する． `getAnywhere()` はオブジェクトを返し，そして `argsAnywhere()` は関数である任意のオブジェクトの引数を返す．

Usage

```
getAnywhere(x)
argsAnywhere(x)
```

Arguments

x	文字列又は名前.

Details

これらの関数は，それらが検索パス上のパッケージに関連しているかどうかにかかわらず，全てのロードされた名前空間を調べる．

これらは文字通りには "あらゆる場所"を探索しない：例えば，ロードされていない局所的評価環境と名前空間は調べられない．

関数が登録された S3 メソッドとして見つかる場所では，どの名前空間がそれらを登録しているかを見出すことが試みられる．特に名前空間が登録されていない場合，これは正確ではないかもしれない，

Value

`getAnywhere()` に対してはクラス "getAnywhere" のオブジェクト．これは次の成分を持つリストである

name	探索された名前
objs	見つかったオブジェクトのリスト

where	オブジェクトがどこで見つかったかを説明する文字列
visible	論理値：オブジェクトは可視か
dups	論理値：オブジェクトはリスト中の前のものと同じか.

オブジェクトが同値であるかどうかの計算ではそれらの環境は無視される.

普通 print メソッドでは構造は隠される．見つかった一つまたはそれ以上のオブジェクトを取り出す [メソッドがある.

argsAnywhere() に対しては，一つ又はそれ以上の引数リストが返される.

See Also

使われるメソッドを見つけるには getS3method：アンロードされた名前空間や別の名前で登録されているかもしれないため，これは getAnywhere が返すものではないかもしれない.

get, getFromNamespace, args

Examples

```
getAnywhere("format.dist")
getAnywhere("simpleLoess") # stats からは移入されない
argsAnywhere(format.dist)
```

---

getFromNamespace　　名前空間の開発用のユーティリティ関数

Description

名前空間中の移出されていない関数にアクセスしたり置き換えたりするユーティリティ関数で，名前空間を持つパッケージの開発で使う.

これらは最終コードでは使うべきではない (恐らく assignInMyNamespace を除いて，しかし'注意'節を見よ).

Usage

```
getFromNamespace(x, ns, pos = -1, envir = as.environment(pos))

assignInNamespace(x, value, ns, pos = -1,
                  envir = as.environment(pos))

assignInMyNamespace(x, value)

fixInNamespace(x, ns, pos = -1, envir = as.environment(pos), ...)
```

Arguments

x	オブジェクト名(文字列として与えられた).
value	R オブジェクト.
ns	名前空間，または名前空間を与える文字列.
pos	オブジェクトをどこで見つけるか：get を見よ.
envir	捜索すべき環境を指定する別の方法.
...	エディタに渡される引数：edit を見よ.

Details

assignInMyNamespace はパッケージ内の関数から呼び出されることを意図しており，そしてそれを呼び出す関数の環境として名前空間を選ぶ.

名前空間は幾つかの方法で指定できる．例えば ns = "stats" の使用は最も直接的であるが，ロードされたパッケージは get に対して使われる任意のメソッドにより指定できる：ns は又 <namespace:foo> としてプリントされる環境であっても良い.

getFromNamespace は ::: 演算子に類似(しかし優先する)する：これは名前空間をどの様に指定するかに関してより柔軟である.

fixInNamespace は名前 x のオブジェクトに対して edit を起動し，そしてオリジナルのオブジェクトの代わりに改定されたオブジェクトを付値する．fix との互換性から，x は引用符なしで良い.

Value

getFromNamespace は見つかったオブジェクトを返す (又はエラーを返す).

assignInNamespace, assignInMyNamespace そして fixInNamespace は名前空間中のオブジェクトを変更するという副作用のために起動される.

警告

assignInNamespace は最終コードでは使うべきではなく，将来はパッケージから呼び出されるとエラーになるであろう．すでにある種の利用は許可されなくなっている.

Note

assignInNamespace, assignInMyNamespace そして fixInNamespace は名前空間中のコピーを変更する．特にパッケージ中のその名前のオブジェクト(もしあれば)と名前空間中にすでに移入されている任意のコピーである．これらは実際は名前空間から移出されていないオブジェクトに対してだけ使われることを意図している．これらはもし見つかれば S3 メソッドとして登録されたコピーを変更しようとはしない.

これらは名前空間のオブジェクトの値を変更するためにだけ使うことが出来，新しいオブジェクトを作らない.

See Also

get, fix, getS3method

Examples

```
getFromNamespace("findGeneric", "utils")
## Not run:
fixInNamespace("predict.ppr", "stats")
stats:::predict.ppr
getS3method("predict", "ppr")
## もしくは
fixInNamespace("predict.ppr", pos = 3)
fixInNamespace("predict.ppr", pos = "package:stats")

## End(Not run)
```

getParseData　　オブジェクトからの詳細な構文解析情報を得る．

Description

もしオプション "keep.source" が TRUE なら， R の構文解析器はそれが構文解析したオブジェクトに関する詳細な情報を付加する．これらの関数はその情報を回収する．

Usage

```
getParseData(x, includeText = NA)
getParseText(parseData, id)
```

Arguments

x	parse が返す表現式，又はソース参照情報を持つ関数や他のオブジェクト
includeText	論理値；結果中の構文解析項目のテキストを含めるかどうか
parseData	getParseData が返すデータフレーム
id	そのテキストが回収されるべき項目の識別子のベクトル

Details

バージョン 3.0.0 で R の構文解析器は彼の **parser** パッケージ中で Romain Francois により書かれたコードを含むように修正された．これは構文解析コード中の全てのトークンと高水準構成に関する情報の詳細なテーブルを構築する．このテーブルは構文解析コード中のソース参照に関連した srcfile レコード中に保管され， getParseData 関数により回収される．

Value

getParseData に対しては：
もし構文解析されたデータが存在しないと NULL．さもなければ次の列を含むデータフレーム：

line1	整数．項目が始まる行番号．これは getSrcLocation 中の "parse" と呼ばれる構文解析行番号であり， in getSrcLocation，#line ダイレクティブを無視する．
col1	整数．項目が始まる行番号．最初の文字は column 1．これは getSrcLocation 中の "column" に対応する．
line2	整数．項目が終わる番号．
col2	整数．項目が終わる列番号．
id	整数．この項目に関連する識別子．
parent	整数．この項目の親の id．
token	文字列．トークンのタイプ．
terminal	論理値．トークンが“終端”であるか，つまり構文解析木中の葉．

text　文字列．もし includeText が TRUE ならば，全てのトークンのテキスト；もしそれが NA (既定値)ならば，終端トークンのテキスト．もし includeText == FALSE ならば，この列は含められない．非常に長い文字列(1000 文字かそれ以上のソース)は保存されない；代わりにそれらの長さと分離子が含められる．

データフレームの行名は id 値に等しく，データフレームは使われた srcfile 記録を含む "srcfile" 属性を持つ．行はソースファイル中の開始位置で順序付けられ，親項目は子供の前に現れる．

getParseText に対しては：
関連するテキスト項目を含む id と同じ長さの文字列ベクトル．もしそれらが parseData に含まれないと，それらはオリジナルのファイルから回収される．

Note

getParseData が返す結果はオリジナルの **parser** コード中のそれらと比較して幾つかの違いがある：

- 僅かな列だけが保持される．
- 内部的なトークン番号は返されない．
- col1 は 1 から数え始め 0 からではない．
- id 値は構文解析木の要素に付加されず，それらは getParseData が返すテーブルの中だけに保持される．
- #line ディレクティブは特定されるが，しかし他のコメントマークアップ (例えば **roxygen** コメント)はされない．

Author(s)

Duncan Murdoch

References

Romain Francois (2012). parser: Detailed R source code parser. R package version 0.0-16. https://github.com/halpo/parser.

See Also

parse, srcref

Examples

```
fn <- function(x) {
  x + 1 # コメント，ソースの一部として保持される
}

d <- getParseData(fn)
if (!is.null(d)) {
  plus <- which(d$token == "'+'")
  sum <- d$parent[plus]
  print(d[as.character(sum),])
  print(getParseText(d, sum))
}
```

getS3method *S3 メソッドを得る*

Description

S3 総称的関数に対するメソッドを得る，可能性として名前空間や総称的関数のレジシストリから.

Usage

```
getS3method(f, class, optional = FALSE, envir = parent.frame())
```

Arguments

f	文字列：総称的関数の名前.
class	文字列：クラス名.
optional	論理値：総称的関数やメソッドを見つけることの失敗が許されるべきか?
envir	その中でメソッドとその総称的関数が先ず探される environment.

Details

S3 メソッドは名前空間中に隠されているかもしれず，そして get では見つからないかもしれない：この関数はそうした関数を主にデバッグ目的で回収する.

更に，S3 メソッドは名前空間がロードされた時総称的関数に関して登録することが出来，登録されたメソッドは可視なものがなければ使われる(名前空間のスコープ規則を用いて).

どの S3 メソッドが使われるかは総称的関数がどこから呼び出されるかに依存するかもしれない： getS3method はもし f が同じ環境から呼び出されていた時見つかったメソッドを返す.

Value

見つかった関数，もしくは関数が見つからず optional = TRUE ならば NULL.

See Also

methods, get, getAnywhere

Examples

```
require(stats)
exists("predict.ppr") # 偽
getS3method("predict", "ppr")
```

glob2rx　　ワイルドカードやグロッブパターンを正規表現に変える

Description

globbing パターンとしても知られる *wildcard* を対応する正規表現 (regexp) に変更する.

Usage

```
glob2rx(pattern, trim.head = FALSE, trim.tail = TRUE)
```

Arguments

pattern	文字列ベクトル
trim.head	先頭の "^.*" が結果から取り去られるべきかを指示する論理値.
trim.tail	末尾の "^.*" が結果から取り去られるべきかを指示する論理値.

Details

これはほとんどのシェルが使うようなワイルドカードを取り対応する正規表現を返す. ? は . に(単一の文字にマッチ), * は .* に(任意の文字列にマッチ, 空のものを含む), そしてパターンが固定される (最初から始まり最後で終わらなければならない). オプションで結果の正規表現は単純化される.

現在 (, [そして { すら pattern 中で使えるが, しかし glob2rx() は pattern 中の任意の文字列では正しく動作しないかもしれないことを注意する.

Value

入力 pattern と同じ長さの文字列ベクトルで各ワイルドカードは対応する正規表現に翻訳されている.

Author(s)

Martin Maechler, Unix/sed に基づくバージョン, 1991；現在：2004

See Also

正規表現については regexp, 正規表現を用いた代入については sub 等.

Examples

```
stopifnot(glob2rx("abc.*") == "^abc\\.",
          glob2rx("a?b.*") == "^a.b\\.",
          glob2rx("a?b.*", trim.tail = FALSE) == "^a.b\\..*$",
          glob2rx("*.doc") == "^.*\\.doc$",
          glob2rx("*.doc", trim.head = TRUE) == "\\.doc$",
          glob2rx("*.t*")  == "^.*\\.t",
          glob2rx("*.t??") == "^.*\\.t..$",
          glob2rx("*[*")  == "^.*\\["
)
```

globalVariables　　パッケージのチェックで用いられる宣言

Description

globalVariables に対しては，提供される名前は関数や check ツールがこのパッケージに適用される際に大局的に定義されていると見做されるべき他のオブジェクトのものである． globalVariables の呼び出しはパッケージのソース中に含められるべきである．同じパッケージ中の呼び出しの繰り返しは大局的変数の名前を集積する．

典型的な例は参照クラス中の欄とメソッドであり，これらは codetools に対して大局的オブジェクト.(このケースは setRefClass() とその類似物により自動的に処理され，提供された欄とメソッド名を使う．)

suppressForeignCheck に対しては，提供される名前は checkFF(registration = TRUE) でチェックされるべきではない外部関数中の .NAME として使われる変数のそれである．この宣言無しでは，単純な文字列以外の表現式は登録されたネイティブシンボルオブジェクトとして評価されると仮定される．呼び出しのタイプ(.Call, .External 等)と引数のカウントはチェックされる．この宣言をすると，これらの名前に関するチェックは普通抑制される．(もしコードが実行時にだけ評価されるべき表現式を使うのなら，メッセージはそれを dontCheck 関数呼び出し中にラップするか，またはそれを局所変数に保存しその変数に関するメッセージを抑制することで抑制することが出来る．下の例を見よ．)

Usage

```
globalVariables(names, package, add = TRUE)
suppressForeignCheck(names, package, add = TRUE)
```

Arguments

names	オブジェクト名の文字列ベクトル．もし省略すると，パッケージ中で宣言された大局的変数の現在のリスとは未変更で返される．
package	関連パッケージで，普通パッケージの文字列名であるが，オプションでその対応する名前空間環境． globalVariables 又は suppressForeignCheck の呼び出しがパッケージのソースファイルに由来する場合，引数は下の例のように普通省略される．
add	names の内容を現在の大局的変数に加えるかそれともそれを置き換えるか?

Details

宣言された大局的変数とネイティブなシンボルオブジェクトはパッケージの名前空間中のメタデータオブジェクト中に保管され， globalVariables や suppressForeignCheck の呼び出しがパッケージのソースコード中のトップレベルで生起すると仮定される．

パッケージ tools 中に実装されたようなチェック命令は可能性のある問題のためにパッケージ中の R ソースコードをチェックする前にリストされる必要がある．

globalVariables は R 2.15.1 で導入され，そして suppressForeignCheck は R 3.1.0 で導入されたため両者は条件付きで使われるべきである：例を見よ．

Value

globalVariables は宣言された大局的変数の現在のリストを返し，もしかするとこの呼び出しで修正されているかもしれない.

suppressForeignCheck はチェックされないネイティブなシンボルオブジェクトの現在のリストを返す.

Note

大局的変数のりストは実際は全体としてのパッケージとウよりは制限されたスコープに所属する (例えば，関数又はメソッド定義のグループ). しかしながら，より細かい制御の実装は check と/又は codetools 中の変更を必要とするため，このバージョンでは情報はパッケージのレベルで保管される.

Author(s)

John Chambers と Duncan Murdoch

See Also

dontCheck.

Examples

```
## Not run:
## パッケージが ".obj1" と ".obj2" を付値するあるコードを持つが
## codetools はそれを見つけられないと仮定する.
## 同じソースファイル中(そうしたことを忘れないように)で次を加える：
if(getRversion() >= "2.15.1")  utils::globalVariables(c(".obj1", "obj2"))

## 実行時の計算されたネイティブシンボルに関するメッセージを抑制する
## ために，それを局所変数に保存する.

## トップレベルでこれを置く：
if(getRversion() >= "3.1.0") utils::suppressForeignCheck("localvariable")

##自分の関数内部で，このような呼び出しを行う：
localvariable <- if (condition) entry1 else entry2
.Call(localvariable, 1, 2, 3)

## 別法として，次のようなもの：
if(getRversion() < "3.1.0") dontCheck <- identity
.Call(dontCheck(if (condition) entry1 else entry2), 1, 2, 3)

## しかしながら，完全にチェックできるコードを書くほうがよほど良い,
## 例えば
if(condition) .Call(entry1, 1, 2, 3) else .Call(entry2, 1, 2, 3)

## End(Not run)
```

head　　オブジェクトの最初か最後の部分を返す

Description

ベクトル，行列，テーブル，データフレーム又は関数の最初や最後の部分を返す．head() と tail() は総称的関数であるため，他のクラスに拡張されているかもしれない．

Usage

```
head(x, ...)
## Default S3 method:
head(x, n = 6L, ...)
## S3 method for class 'data.frame'
head(x, n = 6L, ...)
## S3 method for class 'matrix'
head(x, n = 6L, ...)
## S3 method for class 'ftable'
head(x, n = 6L, ...)
## S3 method for class 'table'
head(x, n = 6L, ...)
## S3 method for class 'function'
head(x, n = 6L, ...)

tail(x, ...)
## Default S3 method:
tail(x, n = 6L, ...)
## S3 method for class 'data.frame'
tail(x, n = 6L, ...)
## S3 method for class 'matrix'
tail(x, n = 6L, addrownums = TRUE, ...)
## S3 method for class 'ftable'
tail(x, n = 6L, addrownums = FALSE, ...)
## S3 method for class 'table'
tail(x, n = 6L, addrownums = TRUE, ...)
## S3 method for class 'function'
tail(x, n = 6L, ...)
```

Arguments

x	オブジェクト
n	一つの整数．もし正ならば結果のオブジェクトに対するサイズ：ベクトルの要素数(リストを含む)，行列やデータフレームの行，又は関数の行数．もし負ならば，x の n 個の最後/最初要素を除いた全て．
addrownums	もし行名がなければ，行番号からそれらを作る．
...	他のメソッドへ/から渡される引数．

Details

行列，二次元テーブルそしてデータフレームに対しては head() (tail()) は n > 0 ならば最初 (最後)の n 行，もしくは n < 0 ならば最初(最後)の n 行を除く全て． head.matrix() と tail.matrix() は移出されている．関数に対しては構文解析された行が文字列として返される．

もし行列が行名を持たなければ，tail() は "[n,]" の形の行名を結果に加えるので，プリントされた時の x の最後の行の様に見える． addrownums = FALSE はこの挙動を抑制する．

Value

x と類似(普通)のオブジェクトであるが一般により小さい． ftable オブジェクトの x に対しては変換された format(x)．

Author(s)

Patrick Burns， R-Core により改良と訂正が行われた．負の引数は Vincent Goulet により付け加えられた．

Examples

```
head(letters)
head(letters, n = -6L)

head(freeny.x, n = 10L)
head(freeny.y)

tail(letters)
tail(letters, n = -6L)

tail(freeny.x)
tail(freeny.y)

tail(library)

head(stats::ftable(Titanic))
```

---

help　　ドキュメント

Description

help はヘルプシステムへの第一義的なインタフェイスである．

Usage

```
help(topic, package = NULL, lib.loc = NULL,
     verbose = getOption("verbose"),
     try.all.packages = getOption("help.try.all.packages"),
     help_type = getOption("help_type"))
```

Arguments

`topic` 普通 name 又は文字列でヘルプが求められるトピックを指定する．文字列(明示的に一重又は二重引用符で囲まれる)は常にトピックの名前とされる．

もし `topic` の値が長さ1の文字列ベクトルならばトピックは唯一の要素の値に取られる．さもなければ `topic` は名前か又は reserved 語 (もし構文的に適正なら)か又は文字列でなければならない．

もしこれが省略されると何が起きるかについては'詳細'節を見よ．

`package` 名前又はドキュメントをその中で探す名前か文字列ベクトル，又は `NULL`．既定では名前空間がロードされている全てのパッケージが使われる．構文解析された名前を避けるには例えば `(pkg_ref)` を使う(例を見よ)．

`lib.loc` R ライブラリのディレクトリ名の文字列ベクトル又は `NULL`．既定値 `NULL` は現在既知の全てのライブラリに対応する．もし既定値が使われると，ライブラリの前にパッケージが検索される．これは HTML ヘルプに対しては使われない('詳細'節を見よ)．

`verbose` 論理値；もし `TRUE` ならば，ファイル名が報告される．

`try.all.packages` 論理値；**注意**節を見よ．

`help_type` 文字列：必要なヘルプのタイプ．可能な値は `"text"`, `"html"` そして `"pdf"`．大・小文字の区別は無視され，部分的マッチングが許される．

Details

次のヘルプのタイプが利用できる：

- 平文のテキストヘルプ
- 他のトピックスへのハイパーリンクを持つ HTML ヘルプページで， `browseURL` によるブラウザ中に表示される．(可能な箇所では既存のブラウザが差いるようされる：OS X GUI はそれ自体のブラウザウィンドウを使う．) もしある理由で HTML ヘルプが利用できなければ (`startDynamicHelp` を見よ)．平文のテキストヘルプが代わりに使われる．
- `help` に対してだけ，PDF としてタイプセットする – 'オフラインヘルプ'の節を見よ．

'工場出荷時'既定値はテキストのヘルプで，例外は OS X GUI からのもので，これはそれ自体のブラウザウィンドウ HTML ヘルプ表示を使う．

テキストのヘルプの表現は適当なロケール (UTF-8 と単バイト Windows ロケール)での向き付き引用符を使う：ある場合には使用するフォントがこれらの引用符をサポートせず，これは `options(useFancyQuotes = FALSE)` と設定することでオフに出来る．

`topic` はオプションでは無い：もしそれが省略されると R は以下を与える

- もしパッケージが指定されると．(テキスト又は対話的な使用のみで HTML)パッケージの情報で，適当なヘルプトピックに対するヒント/リンクを含む．
- もし `lib.loc` だけが指定されると，利用できるパッケージの(テキストの)リスト．
- もし最初の三つの引数のどれもが指定されないと `help` 自体に対するヘルプ．

ある種のトピックは引用化される (backtick で)，又は文字列として指定される必要がある．これらは構文的にそれら自体では登場できない単項と二項演算子， `function` そして制御フロー reserved語 (`if`, `else` `for`, `in`, `repeat`, `while`, `break` and `next` を含む)を含む．

他の reserved 語はそれらが名前であるかのように使用できる，例えば TRUE, NA そして Inf.

もし topic にマッチするヘルプファイルが複数あると，対話的な使用ではユーザがどれかを選ぶようにメニューが表示される：バッチ使用では検索パス上の最初のものが使われる． (HTML ヘルプに対してはメニューは HTML ページで，さもなければもし既定の getOption("menu.graphics") が真ならば可能なときはグラフィカルなメニューを使う．)

HTML ヘルプは lib.loc を使わないことを注意する：これは常に最初にロードされたパッケージを探し，そしてそれから .libPaths() に沿って探す．

オフラインのヘルプ

タイプセットされたドキュメントがヘルプページの LaTeX バージョンを pdflatex を通して実行することで生成される：これは PDF ファイルを作る．

出力の見かけは LaTeX の検索パス中のどこかにある 'Rhelp.cfg' ファイルを通じてカスタマイズ出来る：これは LaTeX のスタイルファイルとして Rd.sty の後に入力される．ある種の environment variables が参照される，特に R_PAPERSIZE (getOption("papersize") 経由で) そして R_RD4PDF ('R Installation and Administration Manual' 中の 'Making manuals' を見よ).

もし作業空間か検索パスの更に下に関数 offline_help_helper があればそれがタイプセットのために使われ，さもなければ utils の名前空間中のその名前の関数 (最初のパラグラフがそれに対して適用される)．それは少なくとも二つの引数，タイプセットされる LaTeX ファイルとタイプ (これは今では無視される)．それは三番目の引数 texinputs を受け入れ，これはヘルプドキュメントが図を含む時グラフィックスパスを与え，さもなければ何も提供しない．

Note

lib.loc が明示的に指定されていない限り，ロードされたパッケージが指定されたライブラリの前に検索される．これはもしライブラリが既知のライブラリツリー中にないライブラリからロードされていれば，ロードされたライブラリからのヘルプが使われる．もし lib.loc が明示的に指定されると，ロードされたパッケージは探され無い．

もしこの検索が失敗しそして引数 try.all.packages が TRUE，そして packages も lib.loc も指定されないと，既知のライブラリツリー中の全てのパッケージ中で topic に関するヘルプが探され，そしてヘルプが見つかるかもしれない (任意の)リストが表示される (help_type = "html" に対するハイパーリンクと共に)． **注意：** 全てのパッケージの探索は遅いかもしれない，特に最初は (OS のファイルキャッシュはそれ以降の検索を劇的に迅速にする).

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

ヘルプトピックへのショートカットに対しては ?.

曖昧なトピックのヘルプページを見つけるには help.search() 又は ??；R のヘルプページの HTML バージョンを開く help.start()．利用可能なパッケージとそれらが含むヘルプオブジェクトのリストには library()；利用可能なデータセットをリストにするのは data()；methods().

自分自身のパッケージの help を書くためのプロトタイプを得るには prompt().

Examples

```
help()
help(help)              # 同じ

help(lapply)

help("for")             # 又は ?"for", しかし引用符/バックチックが必要

try({# 動作する TeX のインストールが必要：
 help(dgamma, help_type = "pdf")
 ## -> 見栄え良く書式化された pdf -- 数式を含む -- help(dgamma) 用；
 system2(getOption("pdfviewer"), "dgamma.pdf", wait = FALSE)
})

help(package = "splines") # パッケージがロードされていなくてもヘルプを得る

topi <- "women"
help(topi)

try(help("bs", try.all.packages = FALSE)) # 見つからないと報告(エラー)
help("bs", try.all.packages = TRUE)       # パッケージ 'splines' 中に
                                          # あると報告

## プログラミング用：
topic <- "family"; pkg_ref <- "stats"
help((topic), (pkg_ref))
```

---

help.request *R-help* に投稿する

Description

R-help メイリングリストへの投稿前にそれらに期待されている全てがなされているかチェックすることをユーザに促し，セッション情報を持つ投稿用のテンプレートを提供し，そしてオプションで電子メイルで投稿する(Unix システムでは).

Usage

```
help.request(subject = "",
             address = "r-help@R-project.org",
             file = "R.help.request", ...)
```

Arguments

subject	電子メイルのサブジェクト．サブジェクト中では一重引用符 (')を使わないで欲しい! 複数の問い合わせは別個のヘルプ要請として投稿する.
address	受け取り手の電子メイルアドレス.
file	電子メイルをセットアップするために使われる (もし実用なら)ファイル名.
...	create.post に渡される method と ccaddress のような追加の名前付き引数.

Details

この関数は投稿ガイドを置き換えるものではない． R-help に投稿する前にガイドを読むかこの関数を使って欲しい (https://www.r-project.org/posting-guide.html を見よ).

help.request 関数：

- ユーザが関係するリソースを参照したかを尋ねる，もし否ならば停止し関連する URL を開く．
- 現在の R バージョンを使っているかとアドオンパッケージが更新されているかどうかをチェックし，必要な場合は更新オプションを与える．
- ユーザが投稿中に添える適当 (出来る限り小さく，再生性があり，自己完結的でそしてコメントを持つ)実例コードが用意されているかどうかを尋ねる．

このチェックリストが完了するとセッション情報を含む投稿テンプレートが準備され，そして create.post に渡される．

Value

有用なものは何も無い．

Author(s)

Heather Turner， bug.report() の現在のコードとヘルプページに基づく．

See Also

ポストのガイド(https://www.r-project.org/posting-guide.html)，又そこからヘルプのリクエストを加えられる sessionInfo().

create.post.

---

help.search　　ヘルプシステムを探索する

Description

与えられた (ファイル)名，エイリアス，タイトル，概念，又はキーワード項目 (混たはそれらの任意の組み合わせ)にマッチするドキュメントのヘルプシステムの検索を許し，fuzzy matching または regular expression マッチングのどちらかを使う．マッチしたヘルプ項目の名前とタイトルは見栄え良く整形されて表示される

ビニエット名，タイトル，キーワード，そしてデモ名もまた検索できる．

Usage

```
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, use_UTF8 = FALSE,
            types = getOption("help.search.types"))
??pattern
field??pattern
```

Arguments

pattern	指定された欄にマッチされるべき文字列．もしこれが与えられると apropos, keyword そして whatis は無視される．
fields	検索されるべきヘルプデータベースの欄を指定する文字列．項目は "name", "title", "alias", "concept" そして "keyword" の短縮形でなければならず，ヘルプページの (ファイル)名，そのタイトル，そしてそれに対して提供するドキュメントのトピックと概念，そしてそれが分類されるべきキーワードに対応する．詳細とビニエットとデモがどのように検索されるかについての詳細は下を見よ．
apropos	ヘルプページ中のトピックとタイトルにマッチされるべき文字列．
keyword	ヘルプページ中の‘キーワード’にマッチされるべき文字列：‘キーワード’は実際にはカテゴリーである：標準カテゴリーはファイル ‘R.home("doc")/KEYWORDS’ (例も見よ)中にリストされそしてある種のパッケージ作者はそれら自体の物を定義している．もし keyword が指定されると， agrep の既定値は FALSE になる．
whatis	ヘルプページのトピック中でマッチされるべき文字列．
ignore.case	論理値．もし TRUE ならマッチングの間大・小文字の別は無視される；もし FALSE ならば，パターンマッチングは大・小文字の別を区別する．
package	検索されるパッケージの名前を持つ文字列，又は NULL でその場合は lib.loc で指定される 全ての利用可能なパッケージが探索される．
lib.loc	検索されるべき R ライブラリツリーの位置を記述する文字列，又は NULL．既定値の NULL は現在既知の全てのライブラリに対応する．
help.db	前に構築され保存されたヘルプデータベースを与える文字列，又は NULL．
verbose	論理値；もし TRUE なら，検索過程はトレースされる．整数値が又受け入れられ， TRUE は 2 と同値で，1 はより冗長でない．Windows では進行度バーが再構築中に表示され， Unix では verbose = 1 に対しては心拍マークが，そして verbose >= 2 に対してはパッケージ対パッケージのリストが表示される．
rebuild	ヘルプデータベースを再構築するかどうかを指示する論理値．これは lib.loc 又は検索パスが変更されるか，又は package が使われそして値が見つからなければ自動的に実行される．
agrep	もし NULL (keyword が使われない限り既定値)でマッチされるべき文字列がアルファベット，数字，空白又はダッシュだけから成れば， agrep に依る近似(ファジー)マッチングが文字列が5文字未満で無い限り使われる；さもなければそれは grep によってマッチされるべき regular expression を含むとされる．もし FALSE ならば近似マッチングは使われない．さもなければ，近似マッチングに対する最大距離を指定する数値かリストを与えることが出来る， agrep に対するドキュメント中の引数 max.distance を見よ．
use_UTF8	論理値：結果は UTF-8 エンコーディングで与えられるべきか? 又 agrep 中の regexps を Perl の regexps に変更する．
types	検索するドキュメントのタイプをリストする文字列ベクトル．項目は "vignette" "help" または "demo" の短縮形でなければならない．結果は指定された順序で与えられる．
field	検索されるべき単一の fields 値．

Details

パッケージがインストールされる際，予め構築された help.search 索引が 'Meta' ディレクトリ中の 'hsearch.rds' としてシリアル化される (パッケージがヘルプページを持つ限り)．ビニエットは又 'Meta/vignette.rds' ファイル中に索引化される．これらのファイルは hsearch_db によりヘルプ検索データベースを作るのに使われる．

引数 apropos と whatis は同じ名前の Unix 命令と類似の役目を果たす．

agrep = FALSE による検索は (データベースが一旦構築されると)既定よりも数倍速い．しかしながら，近似検索は十分速いはずである (5000 個のパッケージがインストールされた時約1秒)．

もし可能ならば，ヘルプデータベースはセッション中の引き続く呼び出しで使われるためメモリ中に保存される．

現在マッチしたヘルプファイルのエイリアスは表示されない．

? と同様に，?? ではパターンは検索をそのパッケージに限定するためにパッケージ名の後に :: または ::: が先頭に付くかもしれない．

ヘルプファイルに対しては，R のドキュメントディレクトリ中の 'KEYWORDS' ファイル中にリストされた標準キーワードの中に無い '\keyword' 項目は概念とされる．'internal' と異なる標準キーワード項目に対しては，ファイル 'KEYWORDS' からの対応する解説は概念としても使われる．全ての '\concept' 項目は概念として使われる．

ビニエットは次のように検索される．"name" と "alias" は共にビニエットファイル名の基礎で，そして "concept" 項目は \VignetteKeyword 項目から取られる．ビニエットはヘルプシステムの "keyword" 分類を使って分類されない．デモはビニエットと同様に処理されるが，"concept" 検索はされない．

Value

結果はクラス "hsearch" のオブジェクトのリストで返され，これは問い合わせの結果を見栄え良く書式化するプリントメソッドを持つ．この機構は実験的であり，将来の R バージョンで変更されるかもしれない．

OS X の R.app では，これは選択可能な項目を持つブラウザを表示する．このブラウザの終了時，選択された項目が別のヘルプウィンドウで表示される．

クラスの内部構造はドキュメント化されておらず，変更されるかもしれない．

See Also

使用されるヘルプ検索データベースに関するより多くの情報と，利用可能な概念とキーワードを調べるユーティリティに対しては hsearch_db.

help；R のオンラインドキュメントのハイパーテキスト (現在 HTML)バージョンに対しては help.start，これはよく似た検索機構を与える．

R のソースのオンラインサーチにアクセスするには RSiteSearch.

apropos は正規表現を使い優れた例を持つ．

Examples

```
help.search("linear models")    # 線形モデルの当てはめ方を忘れた場合
help.search("non-existent topic")

??utils::help  # utils パッケージ中の "help" にマッチする全てのトピック
```

```
help.search("print")            # ピックやタイトルが 'print' に
                                # マッチする全てのヘルプページ
help.search(apropos = "print")  # 同じ

help.search(keyword = "hplot")  # 高水準プロットを解説する全てのヘルプページ
file.show(file.path(R.home("doc"), "KEYWORDS"))  # 全てのキーワードを示す

## 'try' で始まるトピックを解説するヘルプページ
help.search("\\btry", fields = "alias")
```

`help.start` ハイパーテキストによるドキュメント

Description

ハイパーテキスト(現在 HTML)バージョンの R のオンラインドキュメントを開始する.

Usage

```
help.start(update = FALSE, gui = "irrelevant",
           browser = getOption("browser"), remote = NULL)
```

Arguments

update	論理値：これはパッケージ索引が現在使用可能なパッケージを反映するように更新することを試みつべきか. (もし remote が NULL でなければ試みられない.)
gui	単に S-PLUS との互換性のため.
browser	ハイパーテキストブラウザとして使われるプログラムの名前. これは PATH 中にあるか, 又はフルパスを指定しなければならない. 別法として, これは URL を唯一つの引数に持ち呼び出される R 関数でも良い. このオプションは Windows では普通未設定で, その時はファイルに関連した機構が使われる.
remote	リモート位置の '*R_HOME*' ディレクトリに対する適正な URL を与える文字列.

Details

remote が指定されていない限り, これは HTTP サーバーが利用できることを必要とする (もし可能ならそれが開始される： startDynamicHelp を見よ).

索引ページのリンクの一つは HTML のパッケージ索引 'R.home("docs")/html/packages.html' であり, これは make.packages.html() で再作成される. 局所的な操作に対しては HTTP サーバーはリンクが最初にクリックされた時このリストの一時的バージョンを再作成し, それ以降毎回更新が必要かどうか (もし .libPaths が変更されたり, 又はディレクトリのどれかが変更されていたりした時). これは遅くなる可能性があり, そして update = TRUE の使用はパッケージリストが索引ページが始まる前に更新されることを保証する.

引数 remote は別の R のインストールが発行した HTML ヘルプを利用することに使える：これは典型的にはそのインストールの主要ライブラリからのパッケージだけを示す.

See Also

他の書式でのオン・オフラインヘルプに対しては help().

ヘルプファイルがどのように表示されるかについては browseURL.

R リソースのオンライン検索にアクセスするには RSiteSearch.

Examples

```
help.start()
## Not run:
## 'remote' 引数は次のようにテストできる
help.start(remote = paste0("file://", R.home()))

## End(Not run)
```

---

hsearch-utils　　ヘルプ検索ユーティリティ

Description

ヘルプシステムの検索のためのユーティリティ.

Usage

```
hsearch_db(package = NULL, lib.loc = NULL,
           types = getOption("help.search.types"),
           verbose = getOption("verbose"),
           rebuild = FALSE, use_UTF8 = FALSE)
hsearch_db_concepts(db = hsearch_db())
hsearch_db_keywords(db = hsearch_db())
```

Arguments

package	検索されるパッケージ名を持つ文字列ベクトル，または NULL でその場合 lib.loc で指定されるライブラリツリー中の 全ての利用可能なパッケージが検索される.
lib.loc	検索される R のライブラリツリーの位置を記述する文字列ベクトル，または NULL. 既定値の NULL は現在既知の全てのライブラリツリーに対応する.
types	検索されるドキュメントのタイプをリストする文字列ベクトル. 詳細は help.search を見よ.
verbose	ヘルプ検索データベースを構築する際の報告冗長度を制御する論理値. 詳細は help.search を見よ.
rebuild	ヘルプ検索データベースを再構築すべきかどうかを指示する論理値. 詳細は help.search を見よ.
use_UTF8	論理値：結果を UTF-8 エンコーディングで与えるべきか?
db	hsearch_db() の呼び出しで得られるようなヘルプ検索データベース.

Details

hsearch_db() は help.search によってそれ以降使われるヘルプ検索データベースを構築しキャッシュする．(実際には強制された時(rebuild = TRUE)か“必要な時” だけ再構築される．)

ヘルプ検索データベースの書式は依然として実験的であり，将来のバージョンで変わるかもしれない．現在，それは四つのテーブルからなる：一つは見つかった全てのドキュメントに関する基本情報を持ち，それらの名前，タイトルそしてユニークな id を含む；他の三つのテーブルは個々のエイリアス，概念そしてキーワードをそれらが属するドキュメントオブジェクトの id と共に含む．後者の三つのテーブルを区別するのは一つのドキュメントオブジェクトがこれらの項目の幾つかを提供するという事実が原因であり，効率的な検索を可能にする．

検索可能な項目がそれらのヘルプタイプに従って如何に解釈されるかについては help.search 中の詳細を見よ．

hsearch_db_concepts() と hsearch_db_keywords() はヘルプデータベースから全ての概念またはキーワードをそれぞれ取り出し，これらをそれらの総頻度とそれらが使われているパッケージの数と共にデータフレームとして返し，項目は総頻度が減少順にソートされている．

Examples

```
db <- hsearch_db()
## ドキュメントオブジェクト，エイリアス，キーワード，そして
## 概念の総数(現在の書式を用いる)：
sapply(db, NROW)
## プリントメソッドからも又得ることが出来る：
db
## 最も頻用な 10 個の概念：
head(hsearch_db_concepts(), 10)
## 最も頻用な 10 個のキーワード：
head(hsearch_db_keywords(), 10)
```

INSTALL　　アドオンパッケージをインストールする

Description

アドオンパッケージをインストールするユーティリティ．

Usage

```
R CMD INSTALL [options] [-l lib] pkgs
```

Arguments

pkgs	インストールされるパッケージのパス名持つ空白で分離されたリスト．‘詳細’を見よ．
lib	インストール先の R ライブラリツリーのパス名．‘--library=lib’ の形式も受け入れられる．空白を含むパスは使用中のシェルに対する慣用法を用い引用化されなければならない．
options	特にヘルプファイルを構築するプロセスが制御できるオプションの空白で分離されたリスト．オプションの現在の完全なリストに対しては R CMD INSTALL --help を使う．

Details

これは最初のエラーで停止するので，もし全ての pkgs を試みたければこれをシェルのループを使い呼び出す．

もし lib を明示的に指定せずに R CMD INSTALL pkgs として使われると，パッケージは現在の環境中で実行される R により使われるライブラリパス中の最初のディレクトリをルートするライブラリツリー中にインストールされる．

ライブラリツリー *lib* 中にインストールするには， R CMD INSTALL -l *lib pkgs* を使う．これはインストールの持続中ライブラリパスの先頭に lib を加え，インストールディレクトリ中に要求されるパッケージが見つけられる (そして他のライブラリ中のそれらに優先して使われる)．

lib と pkgs の要素の双方はディレクトリの絶対または相対パスであって良い． pkgs はまたパッケージのアーカイブファイル名を含むことが出来る．これらは単一のディレクトリを含むターボールであり，オプションで gzip, bzip2, xz または compress で圧縮されている．最後に，バイナリーのパッケージアーカイブファイル (R CMD INSTALL --build で作られたような)を与えることも出来る．

ターボールは既定で内部関数 untar により解凍される：もし必要なら外部の tar 命令を環境変数 R_INSTALL_TAR で指定できる：それがターボールで使われている圧縮のタイプを処理できることを確認して欲しい． (これはある場合には不正かサポートされないセクションを含むターボールに対して必要になり，非常に大きなターボールについて速くなることが可能になる． R_INSTALL_TAR を ‘tar.exe’ に設定することがある種の Windows システムでの許可問題を克服するために必要とされてきた．)

パッケージのソースはインストールに先立って ‘--preclean’ により，または ‘--clean’ により後でクリーン化出来る：クリーン化はもしソースが一つ以上のアーキテクチャやプラットフォームと共に使われるときは本質的である．

ある種のソースは，必要ならば引数や変数をそれぞれオプション ‘--configure-args’ と ‘--configure-vars’で引き渡すことが出来る ‘configure’ スクリプトを含む．後者は特にパッケージに必要なライブラリやヘッダーファイルが非システムディレクトリ中にある時役に立つ．この場合，これらの位置を指定するのにコンフィギュア変数 LIBS と CPPFLAGS を使うことが出来る (そしてこれらを ‘--configure-vars’ を使って設定する)，より詳細は “R Installation and Administration” 中の節 “Configuration variables” を見よ． (もしこれらがコマンドラインで一回以上使われるとそれらは連結される．) コンフィギュア機構はオプション ‘--no-configure’ を使うことで回避できる．

パッケージのインストールの試みが失敗すると，残った余り物は除かれる．もしパッケージがすでにインストールされていると，古いバージョンが回復される．これはもし命令がエラーに出会うか，またはもしインストールがキーボードから中断された時起こる：クリーンアップの後でスクリプトは停止する．

実行されるロックの詳細については， install.packages に対するヘルプ中の節 ‘Locking’ を見よ．

オプション ‘--build’ はインストールされたパッケージを配布用にバイナリパッケージ(OS X で使われるような)としてタール化する (OS X で使われるような)．これは環境変数 R_INSTALL_TAR が設定されない限り utils::tar で行われる．

既定ではパッケージは R がそうされている時そしてその時だけ静的な HTML ヘルプページと共にインストールされる：これを上書きするためにはオプション ‘--html’ と ‘--no-html’ を使う．

パッケージは既定ではソースのフォーマット化を保ってはインストールされない (source への keep.source 引数を見よ)：これはオプション ‘--with-keep.source’ または環境変数 R_KEEP_PKG_SOURCE を yes に設定することで有効にされる．

全ての利用可能なオプションを含む，簡潔な利用情報は R CMD INSTALL --help を使う．

サブアーキテクチャ

R のインストールは一つ以上のサブアーキテクチャをサポートできる：現在これは Windows の 32 ビットと 64 ビットビルドに対して使われる．

そうしたインストールに対しては既定の挙動はソースパッケージを全てのインストールされたサブアーキテクチャに対してインストールを試みることである．例外は，パッケージがコンフィギュア用のスクリプトを持つかまたは ‘src/Makefile’ (または Windows では ‘src/Makefile.win’)を持つ場合で，その時は R CMD INSTALL を実行しているサブアーキテクチャに対するコンパイル済みコードだけがインストールされる．

コンパイル済みコードを持つソースコードを R CMD INSTALL で用いられるサブアーキテクチャに対してだけインストールするには ‘--no-multiarch’ を使う．コンパイル済みコードを別のサブアーキテクチャに対してだけインストールするには ‘--libs-only’ を使う．

全ての利用可能なサブアーキテクチャに対してインストールするには二つの方法がある．もしコンフィギュア用スクリプトが Windows アーキテクチャ双方に対して動作することがわかっているときはフラグ ‘--force-biarch’ を使う (そしてパッケージはこれをそれらの DESCRIPTION ファイル中の ‘Biarch’ 欄を 使って指定できる)．次に単一のターボールは

```
R CMD INSTALL --merge-multiarch mypkg_version.tar.gz
```

を使いインストール出来る．

Note

オプションはコマンドラインで ‘pkgs’ に先立つ必要はないが，もしそうならより信頼がおけるようになる．全てのオプションは任意のパッケージの前に処理され，オプションが相剋する効果を持つところでは最後のものが採用される．

INSTALL の操作のある種の部分は R を実行しているアカウントへの書き込みと実行権限を持つ R の一時ディレクトリに依存する (tempdir を見よ，普通 ‘/tmp’ の下)．これは普通そうなっているが，しかしもし ‘/tmp’ が noexec としてマウントされていると，環境変数 TMPDIR はそこから実行が許可されているディレクトリに設定される必要があるかもしれない．

See Also

REMOVE; 幾つかのライブラリーツリーに対する情報については .libPaths； R-level のパッケージのインストールについては install.packages．インターネットや局所的なレポジトリを使ったパッケージの自動的更新については update.packages．

“R Installation and Administration” 中の “Add-on packages” に関するセクションと “Writing R Extensions” 中の “Creating R packages” に関する章．

RShowDoc そして R のソースツリーの ‘doc/manual’ サブディレクトリ)．

---

install.packages　　レポジトリまたはローカルファイルからパッケージをインストールする

Description

CRAN 風のレポジトリやローカルなファイルからパッケージをインストールする．

Usage

```
install.packages(pkgs, lib, repos = getOption("repos"),
                 contriburl = contrib.url(repos, type),
                 method, available = NULL, destdir = NULL,
                 dependencies = NA, type = getOption("pkgType"),
                 configure.args = getOption("configure.args"),
                 configure.vars = getOption("configure.vars"),
                 clean = FALSE, Ncpus = getOption("Ncpus", 1L),
                 verbose = getOption("verbose"),
                 libs_only = FALSE, INSTALL_opts, quiet = FALSE,
                 keep_outputs = FALSE, ...)
```

Arguments

pkgs
: その現在のバージョンをレポジトリからダウンロードすべきパッケージの名前の文字列ベクトル．

もし repos = NULL ならば，ファイルパスの文字列ベクトル．これらはソースディレクトリかアーカイブかバイナリパッケージのアーカイブファイルで良い(R CMD build --binary で作られるような)．(http://と file:// URL が又受け入れられ，そしてファイルはダウンロードされてローカルなコピーからインストールされる．) OS X に対する R の CRAN ビルドに対してはこれらはバイナリパッケージを含む '.tgz' ファイルで良い．ファイルパスに対してはチルド展開がなされる．

もしこれが欠損しているか長さがゼロの文字列ベクトルならば，対話的 R セッションでは利用可能なパッケージのリストボックスが表示される．

lib
: パッケージがインストールされるライブラリディレクトリを与える文字列ベクトル．必要に応じてリサイクル使用される．もし欠損していれば既定値は .libPaths() の最初の要素．

repos
: 文字列ベクトル，使用するレポジトリの基本 URL，つまり "http://cran.us.r-project.org" の様な CRAN ミラーの URL．サポートされる URL スキームに関する詳細は url を見よ．

ローカルファイル，ディレクトリまたは URL からインストールするためには NULL で良い：もし長さが1ならば拡張子 pkgs により推測される．

contriburl
: レポジトリの貢献セクションの URL．もしレポジトリのミラーが不完全ならばこの引数を使う，例えば CD に 'contrib' セクションだけを焼き付けたり，またはバイナリパッケージだけがある場合．引数 repos を上書きする． type = "both" と両立しない．

method
: ダウンロードの方法，download.file を見よ．もし NULL でない available が提供されると使われない．

available
: available.packages が返すような行列でレポジトリで入手可能なパッケージをリストする，または関数が available.packages への内部呼び出しを行うときは NULL． type = "both" とは両立しない．

destdir
: ダウンロードされたパッケージを保管するディレクトリ．もしこれが NULL (既定)ならばセッションの一時ディレクトリのサブディレクトリ downloaded_packages が使われる (そしてファイルはセッションの終わりに消去される)．

dependencies
: これらのパッケージが依存/リンクする/移入する/提案する (そして再帰的に云々)未インストールのパッケージも又インストールするかどうか

を指示する論理値．もし repos = NULL なら使われない．又 c("Depends", "Imports", "LinkingTo", "Suggests", "Enhances") のサブセットである文字列ベクトルでも良い．

lib が長さ1(欠損)の時だけサポートされるので，従属パッケージをどこにインストールするかは曖昧である．もしそうでなければ警告と共に無視される．

既定の NA は c("Depends", "Imports", "LinkingTo") を意味する．

TRUE は pkgs に対して c("Depends", "Imports", "LinkingTo", "Suggests") を使い，付加された従属性に対しては c("Depends", "Imports", "LinkingTo") を使う：これは pkgs，それらの例，そしてビニエットの実行に必要な全てのパッケージをインストールする (もしパッケージの作者がそれらを正しく指定していれば).

これら全てで， "LinkingTo" はバイナリパッケージに対しては省略される．

type
: 文字列で，ダウンロードとインストールされるパッケージのタイプを指示する． Windows とある種の OS X ビルドを除いて "source" になる：これらに対しては'バイナリパッケージ'節を見よ．

configure.args
: (ソースインストールに対してだけ使われる．) 文字列ベクトル又は名前付きリスト．名前を持たない文字列ベクトルが与えられると，要素は単一の文字列 (空白で分離される)に連結され，そして呼び出し R CMD INSTALL 中の '--configure-args' フラグに対する値として使われる．もし文字列ベクトルが名前を持てばこれらは個々のパッケージに対する '--configure-args' の値を特定すると仮定される．これはどれかのパッケージがインストールされる時に使われるパッケージの全コレクションに対する設定を指定することを許す．(これらの設定は従って再使用され既定設定として動作する．)

名前付きリストは同様に同じ効果のために使うことが出来，そしてそれは '--configure-args' に対する値として使われる単一の文字列に連結される各パッケージに対する複数要素の文字列を許す．

configure.vars
: (ソースインストールに対してだけ使われる．) フラグ '--configure-vars' に対する configure.args に類似し， configure 実行に対する環境変数を設定するために使われる．

clean
: R CMD INSTALL の呼び出しに対し '--clean' フラグを加えるかどうかを指示する論理値．これはパッケージのインストールの終わりに中間ファイルを削除する以外の追加の操作を実行するために時折使われる．

Ncpus
: 複数のソースパッケージの並列インストールに対して使われる並列プロセスの数．一つより大きな値はもし Sys.getenv("MAKE", "make") で指定される make 命令が引数 Sys.getenv("MAKE", "make") -k -j *Ncpus* を受け入れる時サポートされる．

verbose
: ある"進行度報告"を与えるべきかどうかを指示する論理値．

libs_only
: 論理値：ソースのインストールに対する追加のサブアーキテクチャのインストールだけに '--libs-only' オプションを使うべきか? (INSTALL_opts も見よ．) これは Windows ではバイナリファイルから単に DLL をインストールするのに使われる，例えば 32 ビットのインストールに 64 ビットのインストールを加える．

INSTALL_opts
: ソースインストールに対して R CMD INSTALL に渡される追加オプションのオプションの文字列ベクトル．例えば c("--html", "--no-multiarch").

追加オプションとして使われる文字列ベクトルの名前付きリストでも良く，それぞれのパッケージ名で名前が付けられる．

quiet　論理値：もし真ならば出力量を減らす．

keep_outputs　論理値：もし真ならばソースパッケージのインストールのソースパッケージからの出力を現在の作業ディレクトリに保管し，出力のファイル名 '.out' が付加されたパッケージ名．別法として，出力をその中に保存するディレクトリを与える文字列．ローカルなファイルからインストールされる場合は無視される．

...　download.file か OS X と Windows ではバイナリインストールのための関数に渡される引数 (これは引数 "lock" を受け入れる：'ロッキング' の節を見よ).

Details

パッケージをインストールする主要関数である．それは名前のベクトルと目標ライブラリを取り，パッケージをレポジトリからダウンロードしそれらをインストールする．(もしライブラリが省略されると既定値は .libPaths() 中の最初のディレクトリになり，もし複数あればメッセージが出る．) もし lib が省略されるか，長さが1で (グループ)書き込み可能ディレクトリでなければ，対話的な使用ではコードは個人的なライブラリツリーを提供 (Sys.getenv("R_LIBS_USER") の最初の要素)しそこにインストールする．

レポジトリからのインストールに対してはパッケージをそれらの従属性を尊重してインストールする試みがされる．これは lib 中の全ての項目がインストールに対する既定のライブラリパス(環境変数 R_LIBS で設定される) 上にあることを仮定する．

任意のすでにインストールされている従属パッケージが最新のバージョンを持つことを保証するため install.packages の前に update.packages を実行することが勧められる．

Value

不可視の NULL.

バイナリパッケージ

この節はバイナリパッケージが利用可能なプラットフォーム， Windows と OS X に対する CRAN ビルト，だけに適用される．

R パッケージは一義的に ソースパッケージとして配布されるが， バイナリパッケージ (インストールされたパッケージのパッケージング) もサポートされ，そして Windows と OS X に対する CRAN ビルトで最も普通に使われる．この関数はどちらのタイプもインストールでき，ファイルをレポジトリからダウンロードしたりローカルファイルを利用する．

type の可能な値は(現在) "source", "mac.binary", "mac.binary.mavericks" そして "win.binary" である：サポートされている場所での適用なバイナリタイプも "binary" で選択できる．

レポジトリからバイナリインストルに対しては，関数は同じレポジトリからのソースの利用可能性をチェックし，もしソースパッケージがより新しいバージョンを持つか利用可能であるがバイナリバージョンがなければ報告する．このチェックは次を使って抑制でき

```
options(install.packages.check.source = "no")
```

そしてバイナリファイルだけを含む部分的なレポジトリがあればそうすべきである．

別法(そして現在の既定値)は "both" で，'バイナリが利用可能で最新ならばつかい，さもなければソースを試みる'ことを意味する．もし優先すべきソースパッケージが存在するがコンパイルする必要があるコードを含むならば，挙動は

getOption("install.packages.compile.from.source") で制御される．もし contriburl か available が指定されると， type = "both" は黙って "binary" に変更される．

type = "source" を使うパッケージはパッケージがコンパイルが必要な C/C++/Fortran コードを含まなければ常に動作する．さもなければ，OS X では 'Xcode に対するコマンドラインツール'と ('R Installation and Administration Manual' を見よ) そしてパッケージが必要なら Fortran コンパイラがインストールされており自分のパスにそれらがあることが必要である．

ロッキング

ロッキングに対する様々なオプションがある：これらはソースとバイナリとでは異なる．

ソースインストールに対する既定値は，ライブラリディレクトリはその中にディレクトリ '00LOCK' を作ることで 'ロック'される．これは二つの目的を持つ：これは任意の他のプロセスがそのライブラリ中にインストールすることを拒否し，そしてエラー時にパッケージの任意の以前のバージョンを回復するために保存するために使われる．各パッケージごとに別個のロッキングを作るより洗練されたロッキングがオプション '--pkglock' で得られる：これは並列インストールに対する十分な自由度を許す．パッケージ毎のロッキングは単一のパッケージと Ncpus > 1L の時は複数パッケージに対してのインストール時には既定である．最後にロッキング(そしてエラー時の回復)は '--no-lock' で抑制される．

OS X や Windows のバイナリインストールでは，既定ではロッキングは行われない．引数 lock を TRUE に設定する (その既定値は getOption("install.lock", FALSE) の値)とソースインストールに対して解説されたディレクトリ毎のロッキングを使う：もし値が "pkglock" ならばパッケージ毎のロッキングが使われる．

もしパッケージロッキングが Windows 上で libs_only = TRUE と共に使われそしてインストールが失敗すると，パッケージはその以前の状態に回復される．

パッケージのインストールがロックされたディレクトリが取り除けないほどひどく失敗することが起こり得ることを注意する：これはロックディレクトリを手動で取り除くまではライブラリディレクトリ(もしくは --pkglock に対してはパッケージの)へのそれ以上のインストールを禁止する

並列インストール

もし pkgs の長さが1より大きくそして Ncpus > 1 ならば並列インストールが試みられる．これは並列 make を利用するので，R が構築された時 make (既定では make)は make -j *n* をサポート出来なければならない： GNU の make である dmake と pmake はサポートするが，Solaris の make と旧式の FreeBSD の make はしない：もし必要なら適切な make を現在のセッション用に選択するために環境変数 MAKE を設定できる．

install.packages は available からの全ての従属性 pkgs を計算できる必要があり，pkgs の要素の一つが間接的に他に依存するかどうかを含む．これは例えばもしインストールする CRAN パッケージがそれ自体 CRAN パッケージに依存する Bioconductor パッケージに依存すれば，available が CRAN と Bioconductor パッケージの双方をカバーする必要があることを意味する．

Note

ある種の R のバイナリ配布物は別個のバンドル中に INSTALL を持つ，例えば R-devel RPM. install.packages はもしそうしたシステムで type = "source" と共に呼び出されるとエラーを与える．

ある種の R のバイナリ配布物はパッケージのインストールに必要なツールが無いマシーンにインストール出来る：可能な対処法はこれら全てのツールを従属物として持つべき R の完全なインストールを行うことである.

See Also

update.packages, available.packages, download.packages, installed.packages, contrib.url.

プロキシーとファイル転送をモニターする他のオプションをどのように処理するかについては download.file を見よ.

INSTALL, REMOVE, remove.packages, library, .packages, read.dcf

レポジトリの設定についてはマニュアル 'R Installation and Administration'.

Examples

```
## Not run:
## Fedora の udunits2 ヘッダーのレイアウトに対する Linux の例
install.packages(c("ncdf4", "RNetCDF"),
  configure.args = c(RNetCDF = "--with-netcdf-include=/usr/include/udunits2"))

## End(Not run)
```

---

installed.packages　　インストールされたパッケージを見つける

Description

指定されたライブラリにインストールされた全てのパッケージの詳細を見つける(または回収する).

Usage

```
installed.packages(lib.loc = NULL, priority = NULL,
                   noCache = FALSE, fields = NULL,
                   subarch = .Platform$r_arch)
```

Arguments

lib.loc	探索される R のライブラリツリーの位置を記述する文字列ベクトル，または全ての既知のツリーに対しては NULL (.libPaths を見よ).
priority	文字列ベクトル又は NULL (既定値). もしヌルでなければパッケージの選択に用いられる；"high" は c("base", "recommended") と同値である. 与えられた優先度無しに全てのパッケージを選択するには priority = "NA" を使う.
noCache	キャッシュされた情報を使わず，それをキャッシュもしない.
fields	既定のものに加えてパッケージの DESCRIPTION ファイルから取り出す欄を指定する文字列ベクトル，又は NULL (既定). 利用できない欄は NA 値になる.
subarch	文字列または NULL. もしナルでなく空でなければ，そのサブアーキテクチャに対してインストールされるパッケージを選択するのに使われる.

Details

installed.packages は lib.loc に沿って見つかった 'DESCRIPTION' ファイルをスキャンしパッケージ名，ライブラリパス，そしてバージョン数の行列を返す．

見つかった情報は R セッションと fields 引数に対してキャッシュ(ライブラリにより)され，トップレベルのライブラリディレクトリが変更された時だけ更新される，例えばパッケージをインストールしたり削除した時．もしキャッシュされた情報が混乱すると installed.packages(noCache = TRUE) を実行することで最新化出来る．

Value

パッケージ毎に一つの行を持つ行列で，行名はパッケージ名で列名は(現在) "Package", "LibPath", "Version", "Priority", "Depends", "Imports", "LinkingTo", "Suggests", "Enhances", "OS_type", "License" そして "Built" (パッケージが構築された R バージョン)．追加の列は引数 fields を使って指定できる．

Note

これは数千のパッケージがインストールされている時は遅くなる可能性があり，ある名前のパッケージがインストールされているか，(system.file または find.package を使う) あるパッケージが使えるか (require を呼び出しそして返り値をチェックする)，少数のパッケージの詳細を見出す (packageDescription を使う)ために使うべきではない．これはインストールされたパッケージ毎に幾つかのファイルを読み取る必要があり，Windows とある種のネットワークにマウントされたファイルシステムでは遅くなる．

See Also

update.packages, install.packages, INSTALL, REMOVE.

Examples

```
## スピードのために検索を .Library だけに制限
str(ip <- installed.packages(.Library, priority = "high"))
ip[, c(1,3:5)]
plic <- installed.packages(.Library, priority = "high", fields = "License")
## どのようなライセンスがそこにあるか：
table( plic[, "License"] )
```

---

isS3method *'method' は S3 メソッド名か?*

Description

method が適正か/登録された S3 メソッドかどうかチェックする．別法として，f と class が指定された時，f が S3 総称的関数で paste(f, class, sep=".") が適正な S3 メソッドであるかどうかをチェックする．

Usage

```
isS3method(method, f, class, envir = parent.frame())
```

Arguments

method	文字列で，典型的には形式 "<fn>.<class>"．もし省略されると代わりに f と class を指定しなければならない．
f	オプションの文字列で，典型的には S3 総称的関数を指定する．method が指定されない時使われる．
class	オプションの文字列で，典型的には S3 クラス名．method が指定されない時使われる．
envir	メソッドとその総称的関数が先ず探される environment で，getS3method() 中のようなもの．

Value

logical TRUE or FALSE

See Also

methods, getS3method.

Examples

```
isS3method("t")           # FALSE - これは S3 総称的関数
isS3method("t.default")   # TRUE
isS3method("t.ts")        # TRUE
isS3method("t.test")      # FALSE
isS3method("t.data.frame")# TRUE
isS3method("t.lm")        # FALSE - 存在しない
isS3method("t.foo.bar")   # FALSE - 存在しない
```

LINK 実行プログラムを作る

Description

実行プログラムを作るためのフロントエンド．

Usage

```
R CMD LINK [options] linkcmd
```

Arguments

linkcmd	実行プログラムを作るために適当なオブジェクトファイル (ライブラリオブジェクトを含む)を一緒にリンクする命令のリスト．
options	リンクを制御又は用法とバージョンに関する情報を得るための追加オプション，

Details

リンカーフロントエンドは特に R の共有又はスタティックライブラリに対してリンクする際有用である：例を見よ.

実際のリンク命令は 'R_HOME/bin' にインストールされた libtool のバージョンにより構成される.

R CMD LINK --help は用法情報を与える.

Note

ある種の R のバイナリ配布物は LINK を別個のバンドルに持つ，例えば R-devel RPM.

これは Windows では利用できない.

See Also

COMPILE.

Examples

```
## Not run: ## R にリンクされたフロントエンドの例.
## 最初に C プログラム
CC=`R CMD config CC`
R CMD LINK $CC -o foo foo.o `R CMD config --ldflags`

## もし Fortran コードが ForFoo.o にコンパイルされていれば
FLIBS=`R CMD config FLIBS`
R CMD LINK $CC -o foo foo.o ForFoo.o `R CMD config --ldflags` $FLIBS

## そして C++ フロントエンドに対して
CXX=`R CMD config CXX`
R CMD COMPILE foo.cc
R CMD LINK $CXX -o foo foo.o `R CMD config --ldflags`

## End(Not run)
```

---

localeToCharset　ロケール名から適当なエンコーディング名を選択する

Description

この関数は名前が付けられたロケールに対する適当なエンコーディングを見つけることを目的にしている．既定では現在のロケールで，そしてそれが UTF-8 ロケールならば適当な単バイトエンコーディング.

Usage

```
localeToCharset(locale = Sys.getlocale("LC_CTYPE"))
```

Arguments

locale	ロケールの名前である文字列.

Details

捜査は OS により異なる．ロケール名は通常 es_MX.iso88591 の様になる．もし最後の成分がエンコーディングを指示し utf8 でなければ，同値なエンコーディング名を探す必要がある．さもなければ，言語(ここでは es)は主要かフォールバックエンコーディングを選択するのに使われる．

C ロケールでは回答は "ASCII".

Value

エンコーディングともしかするとフォールバック単バイトエンコーディングの名前の文字列ベクトルで，もし未知なら NA.

Note

エンコーディング名は libiconv が使うようなものであり， glibc とでも動作すべきであるが商用 Unix ではそうでないかもしれない．

See Also

Sys.getlocale, iconv.

Examples

```
localeToCharset()
```

---

ls.str　　オブジェクトとその構造をリストする

Description

ls.str と lsf.str は str() を各マッチした名前に適用する ls の変種である：値の節を見よ．

Usage

```
ls.str(pos = -1, name, envir, all.names = FALSE,
       pattern, mode = "any")

lsf.str(pos = -1, envir, ...)

## S3 method for class 'ls_str'
print(x, max.level = 1, give.attr = FALSE, ...,
      digits = max(1, getOption("str")$digits.d))
```

Arguments

pos	search パス位置を指示する整数，又は現在の環境に対しては -1.
name	search パス位置を指示するオプションの名前，現在の環境に対しては -1. ls を見よ．
envir	使用する環境，ls を見よ．

all.names	. で始まる名前は省略するかどうかを指示する論理値；ls を見よ.
pattern	ls に渡される regular expression. pattern にマッチする名前だけが考慮される.
max.level	入れ子構造の表示に適用される入れ子レベルの最大値，例えば副リストを含むリスト．既定値は 1：最初の入れ子レベルだけを表示する.
give.attr	論理値；もし TRUE (既定値)ならば，属性を副構造として表示する.
mode	考慮すべきオブジェクト mode を指定する文字列. exists と get に渡される.
x	クラス "ls_str" のオブジェクト.
...	渡される追加の引数. lsf.str はそれらを ls.str に渡し，それはそれらを ls に渡す. (移出されない)プリントメソッド print.ls_str はそれらを str に渡す.
digits	プリントに使われる有効桁数.

Value

ls.str と lsf.str はクラス "ls_str" のオブジェクトを返し，基本的にはマッチした名前の文字列ベクトルで (lsf.str に対しては関数だけ)，ls に類似し，各オブジェクトに対して str() を呼び出す print() メソッドを持つ.

Author(s)

Martin Maechler

See Also

str, summary, args.

Examples

```
require(stats)

lsf.str()  #- 自分が使っている関数はどのように見えているのか?
ls.str(mode = "list")   #- 定義した構造化オブジェクトは何か?

## 少しのオブジェクトを作る
example(glm, echo = FALSE)
ll <- as.list(LETTERS)
print(ls.str(), max.level = 0)# 詳細は見せない

## どの基本関数がそれらの名前の中に "file" を持つか?
lsf.str(pos = length(search()), pattern = "file")

## ls.str() が関数内部で動作することのデモ
## ["browser/debug mode"]:
tt <- function(x, y = 1) { aa <- 7; r <- x + y; ls.str() }
(nms <- sapply(strsplit(capture.output(tt(2)),"  *: *"), `[`, 1))
stopifnot(nms == c("aa", "r","x","y"))
```

maintainer　　パッケージの管理者を示す

Description

パッケージの管理者の名前と電子メイルアドレスを示す.

Usage

```
maintainer(pkg)
```

Arguments

pkg	文字列. 単一のパッケージの名前.

Details

パッケージの管理者の名前と電子メイルアドレスを返すためにパッケージの記述にアクセスする.

貢献パッケージに関する質問はしばしばパッケージの管理者に送られる；基本パッケージに関する質問は普通 R-help か R-devel メイリングリストに送られる. バグ報告は bug.report 関数を使って投稿されるべきである.

Value

パッケージの管理者の名前と電子メイルアドレスを与える文字列を示す. 又はそうしたパッケージがインストールされていなければ NA.

Author(s)

元々 R-help Charlie Sharpsteen <source@sharpsteen.net> のコードから David Scott <d.scott@auckland.ac.nz>；R-core による複数の修正.

References

https://stat.ethz.ch/pipermail/r-help/2010-February/230027.html

See Also

packageDescription, bug.report

Examples

```
maintainer("MASS")
```

make.packages.html *HTML* パッケージリストを更新する

Description

パッケージの HTML リストを再生する.

Usage

```
make.packages.html(lib.loc = .libPaths(), temp = FALSE,
                   verbose = TRUE, docdir = R.home("doc"))
```

Arguments

lib.loc	文字列ベクトル．含められるライブラリのリスト.
temp	論理値：パッケージの索引は HTTP サーバーによって使用される一時的位置に作られるべきか?
verbose	論理値．もし真なら，メッセージをプリントする.
docdir	もし temp が偽ならば，その 'html' ディレクトリ中のディレクトリに 'packages.html' ファイルが構築/更新される.

Details

これは 'packages.html' ファイルを作る， help.start によって使われる一時的コピーか，'R.home("doc")/html' 中のコピー(それに対して書き込み許可がいる)．全てのライブラリツリー中の全てのパッケージの 'DESCRIPTION' ファイルが読み込まれるので，これは非常に遅い可能性がある.

temp = TRUE に対しては情報のある種のキャッシングがあるので，ファイルは，もし lib.loc かそれがリストするディレクトリのどれかが変更された時にだけ再構築される.

Value

不可視の論理値で，FALSE はファイルの作成の失敗を指示し，恐らく適当な許可の欠如に由来する.

See Also

help.start

Examples

```
## Not run:
make.packages.html()
# これは大量のインストールされたパッケージに対しては遅いかもしれない.

## End(Not run)
```

make.socket　　ソケットコネクションを作る

Description

server = FALSE ならば指定されたポートとホストへのクライエントのソケットを開こうと試みる． server = TRUE では R プロセスは接続に対して指定されたポートを聴取しそれからサーバーのソケットを返す．ソケットが閉じられていることを保証するために on.exit を使うのは良いアイデアである，なぜならそれらは64個しかないからである．

Usage

```
make.socket(host = "localhost", port, fail = TRUE, server = FALSE)
```

Arguments

host	リモートホストの名前
port	接続/聞き取るためのポート
fail	接続の失敗はエラーか?
server	サーバーソケットか?

Value

クラス "socket" のオブジェクトで，次の成分を持つリスト：

socket	ソケット番号．これは内部使用のためである． Unix 風システムではこれはファイル記述子である．
port	コネクションのポート番号．
host	リモート計算機の名前．

警告

server = TRUE が信頼できる時接続したホスト名が返されるかどうか知らない．されないと想像する．

Author(s)

Thomas Lumley

References

Luke Tierney の XLISP-Stat 用のコードから，それは更に Robbins and Robbins “Practical UNIX Programming” からのコードに基づく．

See Also

close.socket, read.socket.

ソケットに対するサポートのコンパイルは R 3.3.0 以前ではオプションであった：これが利用可能かどうかを見るには capabilities("sockets").

Examples

```
daytime <- function(host = "localhost"){
    a <- make.socket(host, 13)
    on.exit(close.socket(a))
    read.socket(a)
}
## US 海洋観測所からの公式時間(UTC)
## Not run: daytime("tick.usno.navy.mil")
```

---

memory.size　　メモリ配置を報告する

Description

memory.size と memory.limit は Windows で総メモリ配分を管理するのに使われる．他のプラットフォームではこれらは警告と共に無限大を報告する切り株である．

Usage

```
memory.size(max = FALSE)

memory.limit(size = NA)
```

Arguments

max	論理値．もし真なら OS から得られるメモリの最大量が報告され，さもなければ現在使われている量．
size	数値．もし NA ならメモリサイズを報告し，さもなければ新しい限界を Mb 単位で要求する．

Details

Unix 風システムでメモリ使用を制限するには R を起動するのに使ったシェルの機能を使う，例えば limit や ulimit．

Value

バイト単位のサイズ：常に Inf．

See Also

他の制限については Memory-limits．

menu　　メニューとやり取りする関数

Description

menu はユーザに 1 から選択肢数までのラベルが付いた選択用のメニューを提示する．項目を選択せずに終了すると '0' を選択したことになる．

Usage

```
menu(choices, graphics = FALSE, title = NULL)
```

Arguments

choices	選択肢の文字列ベクトル
graphics	可能ならグラフィックスのメニューを使うべきかどうかを指示する論理値.
title	メニューのタイトルとして使われる文字列． NULL も受け入れられる．

Details

もし graphics = TRUE でウィンドウシステムが利用できるならば (Windows, OS X または Tcl/Tk を使った X11) リストボックスウィジェットが使われる，さもなければテキストのメニューが使われる． menu を非対話的なセッションで使うのはエラーである．

10 またはそれ以下の項目は一列で与えられる，それより多ければもし可能ならば現在のディスプレイ幅内で複数列で与えられる．

もし title が NULL か "" ならばタイトルは表示されない．

Value

選択項目に対応する数，または選択しなければ 0.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

select.list はグラフィカルなメニューを実装するのに使われる．そして複数の選択を可能にする．

Examples

```
## Not run:
switch(menu(c("List letters", "List LETTERS")) + 1,
       cat("Nothing done\n"), letters, LETTERS)

## End(Not run)
```

methods *S3* 総称的関数やクラスに対するメソッドをリストする

Description

S3 や S4 総称的関数に対する利用可能なメソッド，又は S3 や S4 クラスに対する全てのメソッドをリストする．

Usage

```
methods(generic.function, class)
.S3methods(generic.function, class, envir=parent.frame())

## S3 method for class 'MethodsFunction'
print(x, byclass = attr(x, "byclass"), ...)
```

Arguments

generic.function	総称的関数，または総称的関数の名前である文字列．
class	クラスの名前であるシンボル又は文字列： generic.function が与えられない時だけ使われる．
envir	総称的関数が文字列として渡された時，その中で総称的関数の定義を探す環境．
x	典型的には methods(..) の結果で，クラス "MethodsFunction" のオブジェクト，下の‘値’節を見よ．
byclass	結果をどのようにプリントするかを決める "byclass" 属性を上書きすることを許すオプションの logical．‘詳細’節を見よ．
...	可能性としてメソッドに/から渡される追加引数；現在使われない．

Details

methods() は generic.function 又は class 引数に関連した S3 と S4 メソッドを見つける．メソッドは現在の search() パス上の全てのパッケージ中で見つけられる． .S3methods() は S3 メソッドだけを， .S4methods() は S4 メソッドだけを見つける．

generic.function 引数と共に起動されると， "byclass" 属性(詳細節を見よ)は FALSE で，print メソッドは既定で S3 と S4 メソッドのシグネチャ(完全な名前)を表示する． S3 メソッドは総称的関数とクラスを連結して generic.class の様にプリントされ，‘.’ で分離される S3 メソッド名は，もしその中でメソッドが定義されているパッケージの名前空間から移出されていなければアステリスク * が後に付く． S4 メソッドのシグネチャは generic,class-method の様にプリントされる： S4 は多重の選択適用を許すので，シグネチャ generic,A,B-method 中に幾つかのクラスがあるかもしれない．

class 引数と共に起動されると． "byclass" は TRUE で，print メソッドは既定でクラス generic に関係する総称的関数の名前を表示する．

全ての関数のソースコードが利用できる．名前空間から移出された S3 関数に対しては，コマンドラインに generic.class の様にメソッドを入力する．名前空間から移出されていない S3 関数に対しては，コマンドラインに getAnywhere 又は getS3method を見よ．S4 メソッドに対しては getMethod を見よ．

各総称的関数に加えて各メソッドに対するヘルプが利用できる．対話的なヘルプに対しては，ドキュメントショートカット ? を総称的関数名とタブ補完を使い，ヘルプが要求されるメソッドを選ぶには ?"generic<tab>" を使う．

リストされる S3 関数はメソッドの様に命名されたものであり，実際はメソッドでは無いかもしれない (既知の例外はコード中で捨て去られる).

Value

クラス "MethodsFunction" のオブジェクトで， "byclass" と "info" 属性の持つメソッド名．属性 "byclass" は結果が定義された class 引数と共に得られたかどうかを指示する logical． "info" 属性は次の列を持つデータフレーム：

generic 総称的関数の名前の character ベクトル．

visible logical()，メソッドはそれが定義されたパッケージの名前空間から移出されているか?

isS4 logical()，メソッドが S4 メソッドの時真．

from factor で，メソッドが見つかった位置又はパッケージ名の位置．

Note

オリジナルの methods 関数は Martin Maechler により書かれた．

References

Chambers, J. M. (1992) *Classes and methods: object-oriented programming in S.* Appendix A of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

S3Methods, class, getS3method.

S4 に対しては getMethod, showMethods, Methods.

Examples

```
require(stats)

methods(summary)
methods(class = "aov")    # S3 クラス
## 同じだが，より詳細で可読性が低い：
print(methods(class = "aov"), byclass=FALSE)
methods("[[")             # C-内部選択適用を使う
methods("$")
methods("$<-")            # 置き換え関数
methods("+")              # 二項演算子
methods("Math")           # グループ総称的
require(graphics)
methods("axis")           # 総称的に見えるが，そうではない

if(require(Matrix)) {
print(methods(class = "Matrix"))  # S4 クラス
m <- methods("dim")       # S3 と S4 メソッド
print(m)
print(attr(m, "info"))    # より徹底的な情報
```

```
## --> 関連する例については help(showMethods)
}
```

---

mirrorAdmin　　レポジトリのミラーを管理する

Description

CRAN の管理を助ける関数，それらのあるものは他のレポジトリネットワークの管理に対しても有用かもしれない．

Usage

```
mirror2html(mirrors = NULL, file = "mirrors.html",
  head = "mirrors-head.html", foot = "mirrors-foot.html")
checkCRAN(method)
```

Arguments

mirrors	データフレーム，既定ではミラーの CRAN リストが使われる．
file	connection 又は文字列．
head	オプションのヘッダファイル．
foot	オプションのフッタファイル．
method	ダウンロードのメソッド， download.file を見よ．

Details

mirror2html はミラーの CRAN リストに対する HTML ファイルを作り，不可視で HTML テキストを返す．

checkCRAN は全ての CRAN ミラーに対する健全さをテストする．

---

modifyList　　再帰的にリストの要素を修正する

Description

可能性として入れ子のリストを二番目のリストにマッチする各水準で要素の部分集合を変更することで再帰的に修正する．

Usage

```
modifyList(x, val, keep.null = FALSE)
```

Arguments

x	名前付き list，もしかすると空．
val	x 中の対応する成分を置き換える成分を持つ名前付きリスト．
keep.null	もし TRUE ならば， val 中の NULL 要素は x 中の NULL になる．さもなければ対応する要素はもしあれば x から消去される．

Value

x の修正版で，修正は次のように決定される (ここでリスト要素はそれらの名前で特定される)． val 中の x からは欠損している要素は x に加えられる．双方に共通であるが双方のリスト自体に共通ではない要素に対しては， x 中の成分は置き換えられる (又は可能性として val 中のものにより keep.null) の値に依存して消去される．両方のリスト中にある共通の要素に対しては， x[[name]] は modifyList(x[[name]], val[[name]]) で置き換えられる．

Author(s)

Deepayan Sarkar <Deepayan.Sarkar@R-project.org>

Examples

```
foo <- list(a = 1, b = list(c = "a", d = FALSE))
bar <- modifyList(foo, list(e = 2, b = list(d = TRUE)))
str(foo)
str(bar)
```

---

news　　*R* もしくはパッケージのニュース情報を構築し問い合わせる

Description

R もしくはパッケージのニュースのデータベースを構築し問い合わせる．

Usage

```
news(query, package = "R", lib.loc = NULL, format = NULL,
     reader = NULL, db = NULL)

## S3 method for class 'news_db'
print(x, doBrowse = interactive(),
      browser = getOption("browser"), ...)
```

Arguments

query	ニュースの項目を選択するための表現式
package	インストールされたアドオンパッケージの名前を与える文字列か又は "R"．
lib.loc	R ライブラリのディレクトリ名の文字列，又は NULL．既定値の NULL は現在既知の全てのライブラリ．
format	まだ使われていない．
reader	まだ使われていない．
db, x	news() から得られたニュースのデータベース．
doBrowse	コンソールにプリントする代わりにニュースをブラウザで開く(browseURL を使い， help.start 経由であるかのようにアクセスできる) かどうかを指示する論理値．
browser	使われるブラウザ，browseURL を見よ．
...	print() に渡される可能性のある追加引数．

Details

もし package が "R" (既定)ならば， R の 3.0.0 リリース以来(R のトップレベルの 'NEWS' ファイルに対応する) 新しいデータベースが構築される．さもなければ，もし与えられたアドオンパッケージが与えられたライブラリ中に見つかれば，そのニュースをファイル 'inst/NEWS.Rd', 'NEWS' 又は 'inst/NEWS' (この順序で)から構造化された形式で読み取ろうとする．

ファイル 'inst/NEWS.Rd' は Rd の \itemize リストとして与えられた Rd ファイルでなければならず，名前が適当な接頭辞 (例えば，"Changes in version" の後に空白とバージョン数，そしてオプションで空白と括弧で囲まれた ISO 8601 (%Y-%m-%d， strptime を見よ)書式日付，そして可能性として更に \subsection 要素を用いた名前付きのカテゴリーに従ってグループ化されている) を持つ section 要素でグループ化されている． \section{..} の丁度最後に日付を (%Y-%m-%d, <note>) のようにして指定できる，つまり括弧を含む．

アドオンパッケージ中の平文テキストの 'NEWS' ファイルは様々な異なった書式を使う；既定のリーダーは標準のレポジトリからのパッケージから個別のニュース項目を取り出すことが出来ねばならず，これは次の書式(の僅かな変更)を使う：

- 項目はバージョンに従いグループ化され，行の先頭にバージョンヘッダー "Changes in version" を持ち，バージョン番号とオプションでもしかすると括弧で囲まれた ISO 8601 書式日付が後に続く．
- 項目はカテゴリーに従ってグループ化されるかもしれず，行の先頭から始まるカテゴリーヘッダー(バージョンヘッダーと異なる)を持つ．
- 項目は itemize タイプのリストとして書かれ，項目タグとして 'o', '*', '-' 又は '+' のどれかを使う．項目はインデントされなければならず，そして理想的には項目テキストに対して共通のインデントを使う．

追加の書式とリーダーが将来サポートされるかもしれない．

Package **tools** は平文テキストの 'NEWS' ファイル Rd に変換する (内部)ユーティリティ関数 news2Rd を提供する．既定のリーダーで成功裡に処理できる書式中の 'NEWS' ファイルに対してはパッケージの管理者は tools:::news2Rd(dir, "NEWS.Rd") を使うことが出来，もしかすると追加引数 codify = TRUE を持ち， dir はパッケージのルートディレクトリへのパスを指定する文字列である．成功した時は 'NEWS.Rd' ファイルは更に改良でき，それからパッケージのソースディレクトリ 'inst' サブディレクトリに移される．

構築されたニュースのデータベースは "news_db" を継承する文字列のデータフレームで，変数 Version, Category, Date そして Text を持つ，ここで最後は読み取られた項目テキストを含み，そして他の変数はもしそれらが欠損しているか決定できなければ NA である．

query を使いデータベースからニュース項目を選択できる．もし欠損しているか NULL ならば，データベース全てが返される．さもなければ， query は変数 Version , codeCategory, Date そして Text(の部分集合)を含む表現式でなければならず．そしてデータベース中で評価される時データベース中の項目の長さを持つ論理値ベクトルを返さなければならない．評価が TRUE を与える項目が選ばれる．評価時には Version と Date はそれぞれ numeric_version と Date オブジェクトに強制変換されるので，これらのクラスに対する比較演算子を用いることが出来る．

Value

クラス "news_db" を継承するデータフレームで， attributes "package" (そしてもし query が部分抽出に先だてば "subset") を持つ．

Examples

```
## 全ての R ニュースの項目のデータベースを構築する.
db <- news()

## 3.0.1 中の PR 数のバグフィックス
db3 <- news(Version == "3.0.1" & grepl("^BUG", Category) & grepl("PR#", Text),
            db = db)


## 日にち間のニュース(正規の R インストールでは 'Matrix' がそこにある):
if(length(iM <- find.package("Matrix", quiet=TRUE)) && nzchar(iM)) {
   dM <- news(package="Matrix")
   stopifnot(identical(dM, news(db=dM)))
   dM2014 <- news("2014-01-01" <= Date & Date <= "2014-12-31", db = dM)
   stopifnot(paste0("1.1-", 2:4) %in% dM2014[,"Version"])
}

## どのようなカテゴリーが使われているか?
sort(table(db[, "Category"]), decreasing = TRUE)
## バージョンが >= 3.0.0 の項目("3.0.0 patched" を含む):
table(news(Version >= "3.0.0", db = db)$Version)
```

---

nsl　　ホスト名で *IP* アドレスを探す

Description

gethostbyname へのインタフェイス.

Usage

```
nsl(hostname)
```

Arguments

hostname	ホストの名前.

Details

これはインターネット接続をテストするために含められていたが， R を実行しているノードが接続していなければ失敗した．これは又もしヘッダファイル ‘arpa/inet.h’ を含めて BSD ネットワークがサポートされていないと NULL を返す

この関数は Windows では利用できない.

Value

文字列としての IP アドレス，又はもし呼び出しが失敗すれば NULL.

Examples

```
## Not run: nsl("www.r-project.org")
```

object.size　　オブジェクトに割り当てられた容量を報告する

Description

R オブジェクトの保管に使われているメモリの推定量を提供する.

Usage

```
object.size(x)

## S3 method for class 'object_size'
format(x, units = "b", ...)
## S3 method for class 'object_size'
print(x, quote = FALSE, units = "b", ...)
```

Arguments

x	R オブジェクト.
quote	論理値，結果を引用符で囲んでプリントするかどうかを指示する.
units	サイズのプリントで使われる単位．許される値は "b", "Kb", "Mb", "Gb", "Tb", "Pb", "B", "KB", "MB", "GB", "TB", "PB", "KiB", "MiB", "GiB", "TiB", "PiB", "EiB", "ZiB", "YiB", そして "auto" (‘詳細’節を見よ)．短縮化出来る.
...	他のメソッドに/から渡される引数.

Details

メモリ配置のどの部分がどのオブジェクトに割り当てられているかは正確にははっきりしない．この関数は単に大雑把な見積もりを与える：これは原子的なベクトルに対してはかなり正確であるが，例えばもしリストの要素が共有されているかどうかは検出しない．(文字列ベクトルの要素間の共有は考慮されるが，単一オブジェクト中の文字列ベクトル間ではされない．)

計算はオブジェクトのサイズであり，シンボルテーブル中のその名前の保管に必要なスペースは除外する.

関連するスペース(例えば関数の環境と EXTPTRSXP 中のポインターが何を指しているか)は計算に含められない.

オブジェクトののサイズは64ビットビルドでは 32ビットビルドにおけるよりも大きくなるが，同じワード長とポインターサイズを持つ異なったプラットフォームでは同じになる可能性が高い.

format と print メソッド中の units = "auto" は結果が1またはそれ以上である (丸め前に)最大の単位を選ぶ．キロバイト，メガバイト又はギガバイトの値は最も近い 0.1 に丸められる.

バイナリバイトサイズ単位に対する IEC 標準は KiB 等の記法を使う．我々の使う Kb, Mb 等は又 1024 の倍数(1000 の倍数ではない)を意味し，従って Kb, KB そして KiB による値は皆同じであることを注意する - SI 標準とは異なるが広く使われる伝統に従う.

Value

クラス "object_size" のオブジェクトで長さ1の倍精度実数値を持ち，バイト単位のオブジェクトのメモリー割り当ての推定値.

See Also

オブジェクトのデザイン上の制限については Memory-limits.

Examples

```
object.size(letters)
object.size(ls)
format(object.size(library), units = "auto")

sl <- object.size(rep(letters, 1000))
(fsl <- sapply(c("Kb", "KB", "KiB"),
               function(u) format(sl, units = u)))
stopifnot(identical( ## これら三つが同じことを保証する：
             unique(substr(as.vector(fsl), 1,5)),
             format(round(as.vector(sl)/1024, 1))))

## 基本パッケージ中の最大オブジェクト10個
z <- sapply(ls("package:base"), function(x)
            object.size(get(x, envir = baseenv())))
as.matrix(rev(sort(z))[1:10])
```

---

package.skeleton　新しいソースパッケージの骨格を作る

Description

package.skeleton は新しいソースパッケージのある種のセットアップを自動化する．ディレクトリを作り，関数やデータや R コードファイルを適当な場所に保存し，そしてヘルプファイルの骨格とパッケージングのそれ以上のステップを解説する 'Read-and-delete-me' ファイルを作る.

Usage

```
package.skeleton(name = "anRpackage", list,
                 environment = .GlobalEnv,
                 path = ".", force = FALSE,
                 code_files = character())
```

Arguments

name	文字列：パッケージ名と自分のパッケージのためのディレクトリ名.
list	パッケージ中に置く R オブジェクトの名前の文字列ベクトル．普通少なくとも list, environment 又は code_files を与える．'詳細'節を見よ.
environment	オブジェクトを探す環境．'詳細'節を見よ.
path	パッケージのディレクトリを置くパス.

force	もし FALSE ならば既存のディレクトリを上書きしない.
code_files	パッケージを構築する R コードファイルへのパスの文字列ベクトル. ‘詳細’節を見よ.

Details

引数 list, environment そして code_files はパッケージを初期化する別の方法を提供する. もし code_files が提供されると, そのように命名されたファイルは環境を構成するようにソースされ, それからパッケージの骨格を生成するために使われる. さもなければ list の既定値は environment 中のオブジェクト (名前が . で始まるものを含む) であるが, その環境中のオブジェクトの一部分を選択するように与えることが出来る.

ヘルプファイルの切り株が prompt, promptClass そして promptMethods 関数を用い, 関数, データオブジェクト, そして S4 クラスとメソッドに対して生成される. もし別のパッケージからのオブジェクトを移入することを意図しているならば promptImport 関数がオリジナルのものにリンクする単純なヘルプファイルを生成するために package.skeleton の後に使われるべきである.

パッケージのソースは path の nameサブディレクトリ中に置かれる. もし code_files が提供されると, これらのファイルはコピーされる ; さもなければ, オブジェクトは個別のソースファイルにダンプされる. code_files 中のファイル名は接尾辞 ".R" を持つべきで, そして現在の作業ディレクトリ中でなければならない.

ソースとドキュメントに対して作られるファイル名は全ての R が稼働する OS で適正になるように努められる. 不正な文字は ‘_’ で置き換えられ, 不正な名前は ‘zz’ が先行し, 名前は小文字に変換され (大・小文字を区別しないファイルシステムでケースの衝突が起きるのを防ぐため), そして最後に変換された名前は make.unique(sep = "_") でユニークにされる. これはコードとヘルプファイルに対して行うことができるがデータファイルに対してはそうでない (これは名前で探される). 又コードとヘルプファイルは ASCII 文字か数字で始まる名前を持つべきで, これはチェックされ, そしてもし必要ならば z が先頭に加えられる.

ドットで始まる名前を持つ関数はファイル ‘R/*name*-internal.R’ に置かれる.

作業が終わったらファイル ‘Read-and-delete-me’ を消去する, これは配布されるべきではないから.

Value

その副作用のために使われる.

References

より詳細はマニュアル ‘Writing R Extensions’ を見よ.

一旦ソースパッケージができたらそれをインストールする必要がある : マニュアル ‘R Installation and Administration’, INSTALL そして install.packages を見よ.

See Also

prompt, promptClass そして promptMethods.

Examples

```
require(stats)
## 二つの関数と二つの "データセット":
f <- function(x, y) x+y
g <- function(x, y) x-y
```

```
d <- data.frame(a = 1, b = 2)
e <- rnorm(1000)

package.skeleton(list = c("f","g","d","e"), name = "mypkg")
```

---

packageDescription　　パッケージの記述

Description

パッケージの 'DESCRIPTION' ファイルを構文解析し返す.

Usage

```
packageDescription(pkg, lib.loc = NULL, fields = NULL,
                   drop = TRUE, encoding = "")
packageVersion(pkg, lib.loc = NULL)
```

Arguments

pkg	パッケージ名を持つ文字列.
lib.loc	R ライブラリのディレクトリ名の文字列ベクトルか NULL. 既定値の NULL は現在既知の全てのライブラリに対応する. もし既定値が使われると, ライブラリの前にロードされたパッケージと名前空間が探される.
fields	返される欄のタグを与える文字列ベクトル (もし他の欄がファイル中にあれば無視される).
drop	もし TRUE で fields の長さが 1 ならば, "packageDescription" の代わりにそれぞれの欄の値を持つ単一の文字列が返される.
encoding	もし Encoding 欄があれば, どのエンコーディングの再エンコーディングが試みられるべきか? もし NA なら如何なる再エンコーディングもされない. 他の値は iconv により使われるようなものであり, 既定の "" は現在のロケールのエンコーディングを指示する.

Details

パッケージは適正な Version 欄を持つ 'DESCRIPTION' ファイルを持たない限り'見つから'ない. パッケージディレクトリが見つからない時と, 適当なディレクトリがあるが適正な 'DESCRIPTION' ファイルがないときは異なった警告が出る.

パッケージの様に見える attach された環境 (例えば package:utils2) は無視される.

packageVersion() は便利なショートカットで,
if (packageVersion("MASS") < "7.3") { do.things } の様なものを許す.

Value

もし与えられたパッケージに対する 'DESCRIPTION' ファイルが見つかり成功裡に読み込まれると，packageDescription はクラス "packageDescription" のオブジェクトを返し，これは drop = TRUE で無い限り.(与えられた)欄の値を要素に持ちタグを名前に持つ名前付きリストである.

'DESCRIPTION' ファイルの構文解析が失敗すると，これはもし fields が完全でなければ欄タグを名前に持つ NA の名前付きリストを返し，さもなければ NA を返す.

packageVersion() はクラス "package_version" の(長さが1の) オブジェクトを返す.

See Also

read.dcf

Examples

```
packageDescription("stats")
packageDescription("stats", fields = c("Package", "Version"))

packageDescription("stats", fields = "Version")
packageDescription("stats", fields = "Version", drop = FALSE)

if(packageVersion("MASS") < "7.3.29")
  message("you need to update 'MASS'")
```

---

packageName 環境に関連するパッケージを見つける.

Description

パッケージには多くの環境が関連している：この関数はそのパッケージを決定しようと試みる.

Usage

```
packageName(env = parent.frame())
```

Arguments

env	その名前を探す環境.

Details

Environment 環境 env はもし topenv(env) がそのパッケージの名前空間環境であればパッケージに関連しているとされる．従ってもし env がパッケージ内の関数に関連する環境か，又はその中で定義された局所的関数である時 packageName は普通パッケージ名を返す.

全ての環境がパッケージに関連するわけではない：例えば大局的環境やそこで定義された関数の評価フレーム．この場合 packageName は NULL を返す.

Value

パッケージの名前を含む長さ 1 の文字列ベクトル，もしくは名前がなければ NULL.

See Also

getPackageName はもし何も見つからなければ名前を作り出すより洗練された関数である.

Examples

```
packageName()
packageName(environment(mean))
```

---

packageStatus　　パッケージ管理ツール

Description

インストールされたパッケージや様々なレポジトリで利用可能なパッケージに関する情報を要約し，旧式のパッケージを自動的に更新する.

Usage

```
packageStatus(lib.loc = NULL, repositories = NULL, method,
              type = getOption("pkgType"))

## S3 method for class 'packageStatus'
summary(object, ...)

## S3 method for class 'packageStatus'
update(object, lib.loc = levels(object$inst$LibPath),
       repositories = levels(object$avail$Repository), ...)

## S3 method for class 'packageStatus'
upgrade(object, ask = TRUE, ...)
```

Arguments

lib.loc	探索する R ライブラリツリーの位置を記述する文字列ベクトル，または NULL. 既定値の NULL は現在既知の全てのライブラリに対応する.
repositories	R パッケージのインターネット上かローカルマシーン上の位置の URL を記述する文字列ベクトル.
method	ダウンロードの方法， download.file を見よ.
type	パッケージ配布のタイプ： install.packages を見よ.
object	packageStatus が返すようなクラス "packageStatus" のオブジェクト.
ask	もし TRUE ならば，ユーザはどのパッケージを更新するべきかしないかを促される.
...	現在使われない.

Details

repositories 中の URL はレポジトリの適当な貢献セクションへのフルパスでなければならない．既定値は contrib.url(getOption("repos")).

"packageStatus" オブジェクトに対しては print と summary メソッドがある： print メソッドは簡略なテーブル状の要約でそして summary メソッドは結果をプリントする．

update メソッドは "packageStatus" オブジェクトを更新する． upgrade メソッドは update.packages に類似している：これは現在更新されていないパッケージの現在のバージョンのインストールを提供する．

Value

クラス "packageStatus" のオブジェクト．これは二つの成分を持つリストである

inst	データフレームで，成分 installed.packages が返す行列プラス水準 c("installed", "not installed") を持つ因子である "Status" を持つ．その中で登場した最初のレポジトリ中の各パッケージの最新バージョンだけが報告される．
avail	データフレーム，成分 installed.packages が返す行列プラス水準 c("ok", "upgrade", "unavailable") を持つ因子である "Status" を持つ．

summary メソッドに対しては結果は又クラス "summary.packageStatus" であり追加の成分

Libs	各ライブラリに対して一つの要素を持つリスト
Repos	各レポジトリに対して一つの要素を持つリスト

を持ち，要素は各ステータスに対するパッケージ名の文字列ベクトルのリストである．

See Also

installed.packages, available.packages

Examples

```
## Not run:
x <- packageStatus()
print(x)
summary(x)
upgrade(x)
x <- update(x)
print(x)

## End(Not run)
```

page　　*R* オブジェクトに対しページャを起動する

Description

file.show 経由でページャで名前 x のオブジェクトの表現を表示する.

Usage

```
page(x, method = c("dput", "print"), ...)
```

Arguments

x	R オブジェクト，またはオブジェクトを命名する文字列.
method	既定メソッドは dput を使いオブジェクトをダンプする．別法は print を使いページャ中に表示される出力を捕捉する．短縮化出来る.
...	print または file.show (title の様な)に対する追加の引数.

Details

もし x が長さが1の文字列ベクトルならば， page がそこから呼び出される環境を眺めるためにオブジェクトの名前として使われる.

title の既定値はもし ... 中で提供されていないと. file.show に渡される.

See Also

file.show, edit, fix.

表示時に新しいページに移るには frame を見よ.

Examples

```
## Not run: ## 'page'のコードを眺める四つの方法
page(page)              # オブジェクトとして
page("page")            # 文字列
v <- "page"; page(v)    # 長さ1の文字列ベクトル
page(utils::page)       # 呼び出し

## End(Not run)
```

person　　人物

Description

名前や電子メイルアドレスの様な人物に関する情報を保持するクラスとユーティリティ.

Usage

```
person(given = NULL, family = NULL, middle = NULL,
       email = NULL, role = NULL, comment = NULL,
       first = NULL, last = NULL)
## Default S3 method:
as.person(x)
## S3 method for class 'person'
format(x,
       include = c("given", "family", "email", "role", "comment"),
       braces = list(given = "", family = "", email = c("<", ">"),
                     role = c("[", "]"), comment = c("(", ")")),
       collapse = list(given = " ", family = " ", email = ", ",
                       role = ", ", comment = ", "),
       ...,
       style = c("text", "R")
       )
```

Arguments

given	与えられた名前を持つ文字列ベクトル，又はそれのリスト.
family	家族名を持つ文字列，又はそれのリスト.
middle	縮められたミドルネーム．廃止予定，詳細節を見よ.
email	電子メイルアドレスを与える文字列，又はそのリスト.
role	人物の役割を指定する文字列 (詳細節を見よ)，またはそのリスト.
comment	コメントを提供する文字列，又はそのリスト.
first	個人名を与える文字列．廃止予定，詳細節を見よ.
last	家族名を与える文字列．廃止予定，詳細節を見よ.
x	as.person 既定メソッドに対する文字列；さもなければクラス "person" のオブジェクト.
include	書式化の際に含められる欄を与える文字列ベクトル.
braces	文字のリスト(詳細節を見よ).
collapse	文字のリスト(詳細節を見よ).
...	現在使われない.
style	プリントのスタイルを指定する文字列で， "R" は R コードとしての書式化を生み出す.

Details

クラス "person" のオブジェクトは任意個数の人物に関する情報を保持できる．これらはリスト引数を使った person() への一度の呼び出しか，一人の人物を表すオブジェクトを作りそれらを c() で連結することで得られる.

format() メソッドは人物情報を文字列ベクトル (各個人に対して一つの文字列)に変形する： include 中の欄が選択され， collapse のそれぞれの要素を使い，それから braces のそれぞれの要素を使い “囲み”，そして最後に空白で分離された一つの文字列に変形される．もし braces 且つ/又は collapse が全ての欄に対する文字を指定しなければ，用法節で示された既定値が使われる． print() メソッドは format() メソッドを呼び出しそして結果をプリントする． toBibtex() メソッドは適当な BibTeX 表現を作る.

人物オブジェクトは欄か($ を使う)または位置 ([を使う)により部分抽出出来る.

as.person() は総称的関数である．その既定メソッドは既定の人物書式化を逆転しようとし，そして又コンマか "and" (適当な空白と共に)で繋げられた書式化された人物項目を処理できる．

個人名はかなり面倒である，例えば， https://en.wikipedia.org/wiki/Personal_name.

"person" クラスの現在の実装(R 2.12.0 以来)は 与えられた名前と(ミドルネームを含む)家族名の観念を使い，それぞれ given と family で指定される．以前のスキームは個人名，ミドルネーム，そして家族名に基づいており，個人名が家族名に先立つ殆どの西洋文化に対して妥当であったが，ある種の文化では個人名を家族名の後に置いたり，または家族名を使わないこともあり普遍的ではない．新しいスキームへの以降を円滑にするために，引数 first, middle そして last は依然サポートされるが，それらの使用は廃止予定で対応する新しいスタイルの引数と共に使われるべきではない．本来の個人でない個人(例えば，組織，会社，等)に対しては名前に対して given を使う(しかし family は使わない)のが妥当である，例えば person("R Core Team", role = "aut").

新しいスキームは又 MARC Code List for Relators (https://www.loc.gov/marc/relators/relaterm.html). に基づく役割を指定する可能性を加えている． R パッケージの作者の文脈では人物の役割を与える時は以下の使用が提案されている．

"aut" (Author) パッケージに本質的な貢献をした作者全てに対して使い，パッケージの引用で使われるべきである．

"com" (Compiler) コードを集めたが (可能性として他の言語の)それ以上の本質的貢献をしていない人物に対して使う．

"ctb" (Contributor) 僅かな貢献をしたが (コードのパッチ等の様な)しかしパッケージの引用では示されるべきではない作者に対して使う．

"cph" (Copyright holder) 全ての著作権保持者に対して使う．

"cre" (Creator) パッケージの管理者に対して使う．

"ctr" (Contractor) パッケージ(の一部)を書く契約をしており，従ってそれ自身の知的な財産を所有しない作者に対して使う．

"dtc" (Data contributor) パッケージ中のデータセットに貢献した人物に対して使う．

"ths" (Thesis advisor) もしパッケージが学位論文の一部ならば，学位のアドバイザに対して使う．

"trl" (Translator) もし R コードが別の言語 (典型的には S)からの翻訳ならば，R への翻訳者に対して使う．

古いスキームでは，人物オブジェクトは単一の人物について使われ，これらの集まりに対する対応する生成器 personList() を持つ個別の "personList" クラスであった．新しいスキームは任意個数の人物に関する情報に対して一つのクラスを用い，personList 機構の必要性をなくしている．

Value

person() と as.person() はクラス "person" のオブジェクトを返す．

See Also

citation

Examples

```
## 人物オブジェクトを直接作る ...
p1 <- person("Karl", "Pearson", email = "pearson@stats.heaven")

## ... または文字列を変換する.
p2 <- as.person("Ronald Aylmer Fisher")

## 連結と部分抽出.
p <- c(p1, p2)
p[1]
p[-1]

## 欄を取り出す.
p$family
p$email
p[1]$email

## パッケージ作者を指定する, "boot" からの例:
## AC は S のオリジナルを書いた最初の作者 [aut].
## BR は二番目の作者 [aut] で, コードを R に翻訳した [trl],
## そしてパッケージを管理する [cre].
b <- c(person("Angelo", "Canty", role = "aut", comment =
         "S original, http://statwww.epfl.ch/davison/BMA/library.html"),
       person(c("Brian", "D."), "Ripley", role = c("aut", "trl", "cre"),
              comment = "R port", email = "ripley@stats.ox.ac.uk")
     )
b

## フォーマット化.
format(b)
format(b, include = c("family", "given", "role"),
   braces = list(family = c("", ","), role = c("(Role(s): ", ")")))

## BibTeX の作者欄へ変換する.
paste(format(b, include = c("given", "family")), collapse = " and ")
toBibtex(b)
```

PkgUtils　　アドオンパッケージを構築しチェックするためのユーティリティ

Description

R のアドオンパッケージのソースが正しく動作するかどうかをチェックし, それらからソースパッケージを構築するためのユーティリティ.

Usage

```
R CMD check [options] pkgdirs
R CMD build [options] pkgdirs
```

Arguments

pkgdirs　R のアドオンパッケージを持つディレクトリの名前のリスト． check に対してはこれらはまた拡張子 '.tar.gz', '.tgz', '.tar.bz2' または '.tar.xz' を持つ圧縮 tar アーカイブのファイル名でも良い．

options　処理の制御，またはユーティリティの用法とバージョンに関する情報を得るための追加のオプション．

Details

R CMD check は R のアドオンパッケージをそれらのソースからチェックし，広範囲の診断チェックを実行する．

R CMD build は R のソースのターボールを構築する．パッケージの名前は 'DESCRIPTION' ファイルから取られ，ディレクトリ名からではない．これは完全に提供されたソースディレクトリのコピーの上で行われる．

ユーティリティ *foo* の用法情報を得るには R CMD *foo* --help を使う．

R CMD build へのオプションの幾つかの既定値は環境変数 _R_BUILD_RESAVE_DATA_ と _R_BUILD_COMPACT_VIGNETTES_ で設定できる：'Writing R Extensions' を見よ． R CMD check 中のチェックの多くは環境変数でオン・オフ出来る：マニュアル 'R Internals' の第6章を見よ．

既定では R CMD build はターボールの準備に tar への "internal" オプションを使う．外部 tar プログラムは環境変数 R_BUILD_TAR で指定できる．これは非常に大きなパッケージに対して本質的に速いかもしれず，非常に長いパス名(100バイト以上)や非常に大きなファイル(8GB 以上)に対しては必要かもしれないが，結果のターボールは可搬性が無いかもしれない．

R CMD check は既定ではターボールを内部 untar 関数で開封する：もし必要ならば tar 命令は環境変数 R_INSTALL_TAR で指定することが出来る：それがターボールに対して使われる圧縮タイプを処理できることを確認せよ．(これはある場合には不正かサポートされていないセクションを持つターボールに対して必要で，非常に大きなターボールに対しては速いかもしれない． R_INSTALL_TAR を 'tar.exe' に設定することがある種の Windows システムでのファイルパーミッション問題を克服するのに必要とされてきた．)

See Also

"Writing R Extensions" 中の "Checking and building packages" と "Processing Rd format" に関するセクション (R ソースツリーの 'doc/manual' サブディレクトリを見よ).

---

process.events　イベント処理のトリガー

Description

Windows GUI のような R フロントエンドはキーの押下やマウスクリックを OS が生成する"イベント"により処理する．これらは計算の途中定期的に R により自動的に処理されるが，ある場合には即時のイベント処理を惹き起こすことが望ましいかもしれない．process.events 関数はこれを行う．

Usage

```
process.events()
```

Details

これは C API 関数 R_ProcessEvents に対する簡単なラッパーである．そうしたものとして，もしユーザが計算を中断するシグナルを送っても復帰しない可能性がある．

Value

NULL が不可視で返される．

Author(s)

Duncan Murdoch

See Also

R_ProcessEvents 関数の更なる議論に関しては ‘Writing R Extensions’ と ‘R for Windows FAQ’ を見よ．

---

prompt　　*R* のドキュメントファイルのプロトタイプを作る

Description

R オブジェクトをドキュメント化するファイルの構築を手助けする．

Usage

```
prompt(object, filename = NULL, name = NULL, ...)

## Default S3 method:
prompt(object, filename = NULL, name = NULL,
       force.function = FALSE, ...)

## S3 method for class 'data.frame'
prompt(object, filename = NULL, name = NULL, ...)

promptImport(object, filename = NULL, name = NULL,
importedFrom = NULL, importPage = name, ...)
```

Arguments

object	R オブジェクト，典型的には既定メソッドに対する関数．name が指定されるときは missing で良い．
filename	普通 connection かドキュメント外郭がそこに書き込まれるべきファイルの名前を与える文字列．既定値はその名前が ".Rd" が後に続く name であるファイルに対応する．又 NA でも良い(下を見よ)．
name	オブジェクトの名前を指定する文字列．
force.function	論理値．もし TRUE ならば，いかなる場合も object を関数として扱う．
...	他のメソッドに/から渡される追加引数．
importedFrom	そこから object が移入されたパッケージの名前の文字列．もし object が関数ならば既定値は object の環境．
importPage	そこから object が移入されたパッケージ中のヘルプページの名前の文字列．

Details

filename が NA でない限り． object に対するドキュメント外郭が filename で指定されるファイルに書き込まれ，そしてこれに対するメッセージが与えられる．関数オブジェクトに対しては，この外郭は真正の関数と引数名を含む．このように作られた R ドキュメントファイルは依然として編集されドキュメント化されるべきオブジェクトを含むパッケージの 'man' サブディレクトリ中に移動される必要がある．

もし filename が NA ならば，ドキュメント外郭のリストスタイルの表現が作られ返される．外郭をファイルに書き込むことは cat(unlist(x), file = filename, sep = "\n") に相当し，ここで x はリストスタイルの表現である．

prompt が for ループやスクリプト中で使われる時，明示的な name 指定が役に立つ．

promptImport に対する importPage 引数はオリジナルのヘルプページのヘルプファイルの名前の基礎を与える必要がある．例として，approx 関数は **stats** パッケージ中の 'approxfun.Rd' 中にドキュメント化されているので，もしそれが移入されたり再移出されるとそれは
importPage = "approxfun" を持たなければならない．他のパッケージから移入されたオブジェクトは再移出されない限りドキュメント化されない．

Value

もし filename が NA なら，ドキュメント外郭のリスト風の表現．さもなければ，書き込まれるファイルの名前が不可視で返される．

警告

既定のファイル名は限られたファイルシステムの下 (例えば Windows のそれら)では適正なファイル名ではないかもしれない．

現在，関数でないオブジェクトへの prompt の呼び出しはオブジェクトが実際はデータセットでありドキュメントはそのようなものであると仮定される．これは将来のバージョンの R では変わるかもしれない．データセットに対するドキュメントの骨格を作るには promptData を使う．

Note

prompt.data.frame で作られるドキュメントのファイルは **base** パッケージ中の多くのデータフレームドキュメントファイルと同じ書式を持たない．我々はドキュメントに対する好ましい書式を普及するように務めている．

Author(s)

prompt.data.frame については Douglas Bates

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

promptData, help そして "Writing R Extensions" 中の "Writing R documentation" の章 (R のソースツリーの 'doc/manual' サブディレクトリを見よ).

多くのヘルプページの作成(パッケージ用)については package.skeleton を見よ．

ユーザに入力を促すには readline を見よ．

Examples

```
require(graphics)
prompt(plot.default)
prompt(interactive, force.function = TRUE)
unlink("plot.default.Rd")
unlink("interactive.Rd")

prompt(women) # データフレーム
unlink("women.Rd")

prompt(sunspots) # データフレームでないデータ
unlink("sunspots.Rd")

## Not run:
## .GlobalEnv 中の各関数に対するヘルプファイルを作る：
for(f in ls()) if(is.function(get(f))) prompt(name = f)

## End(Not run)
```

---

promptData　　データセットに対するアウトラインドキュメントを生成する

Description

データセットに対するドキュメントの外郭を生成する.

Usage

```
promptData(object, filename = NULL, name = NULL)
```

Arguments

object	データセットとしてドキュメント化される R オブジェクト.
filename	普通 connection か，ドキュメントの外郭がそこに書き込まれるべきファイルの名前を与える文字列．既定値はその名前が後に ".Rd" が続く name であるファイルに対応する． NA かもしれない(下を見よ).
name	オブジェクトの名前を指定する文字列.

Details

filename が NA でない限り， object に対するドキュメントの外郭が filename で指定されるファイルに書き込まれる.

もし filename が NA なら，ドキュメントの外郭が作られ返される．外郭をファイルに書き込むことは cat(unlist(x), file = filename, sep = "\n") に相当する．ここで x はリストスタイルの表現である.

現在，データフレームだけがコードで明示的に処理される.

Value

もし filename が NA ならば，ドキュメントの外郭のリストスタイルの表現．さもなければ，書き込まれるファイルの名前を不可視で返す.

See Also

prompt

Examples

```
promptData(sunspots)
unlink("sunspots.Rd")
```

---

promptPackage　　パッケージのドキュメントに対する外郭を生成する

Description

インストールされたかソースされたパッケージに対するドキュメントの外郭を生成する.

Usage

```
promptPackage(package, lib.loc = NULL, filename = NULL,
              name = NULL, final = FALSE)
```

Arguments

package	ドキュメント化される インストールされた又はソースパッケージの名前を持つ character.
lib.loc	探される R ライブラリツリーの位置を示す文字列，又は NULL. 既定値の NULL は現在既知の全てのライブラリに対応する. ソースパッケージに対してはこれはパッケージソースの親ディレクトリを指定しなければならない.
filename	普通 connection かドキュメントの外殻がそこに書かれるべきファイルの名前. 既定値はその名前が ".Rd" が後につく name であるファイルに対応する. NA でも良い(下を見よ).
name	ヘルプトピックの名前を指定する文字列で，典型的には形式 '<pkg>-package'.
final	単なる外殻ではなく，ヘルプトピックの利用可能なバージョンの作成を試みるかどうかを指示する論理値.

Details

filename が NA でない限り， package に対するドキュメントの外殻が filename で指定されるファイルに書き込まれ，そしてそれに関するメッセージが与えられる.

もし filename が NA ならドキュメントの外殻のリストスタイルの表現が作られ返される. 外殻をファイルに書き込むことは cat(unlist(x), file = filename, sep = "\n") に相当し，ここで x はリストスタイルの表現である.

もし final が TRUE ならば，生成されたドキュメントはマニュアル編集用の代替スロットを含まず，これはそのままで使用できる. 殆どの場合手動で編集されたファイルが好ましい (しかし final = TRUE はまず使えない).

Value

もし filename が NA ならば，ドキュメントの外殻のリストスタイルの表現．さもなければ，書き込まれるファイルの名前が不可視で返される．

See Also

prompt

Examples

```
filename <- tempfile()
promptPackage("utils", filename = filename)
file.show(filename)
unlink(filename)
```

Question　　ドキュメントへのショートカット

Description

これらの関数はドキュメントへのアクセスを提供する． name という名前のあるトピック (典型的には R オブジェクトかデータセット) に関するドキュメントは help("name") か ?name によって表示できる．

Usage

```
?topic

type?topic
```

Arguments

topic	普通 name か文字列でヘルプを探すトピックを指定する． もしくは対応する S4 メソッドに関するドキュメントを尋ねる関数呼び出し； S4 メソッドドキュメントの節を見よ．呼び出し *pkg*::*topic* と *pkg*:::*topic* は特別に扱われ， *pkg* 中の topic を探す．
type	このトピックに対して使うドキュメントの特別なタイプ；例えば，もしタイプが class ならば，ドキュメントは名前 topic を持つクラスに対して提供される． methods?*function* と method?*call* を含む，形式的メソッドに関するヘルプを得るための type の使用については 'S4 メソッドのドキュメント'節を見よ．

Details

これは help へのショートカットでそのヘルプの既定タイプを使う．

ある種のトピックは引用化される (backtick で)か文字列として与える必要がある．これらは単項，二項演算子のようなシンタックス的にそれら自体では登場できないものを含む．function と制御フローの reserved 語 (if, else, for, in, repeat, while, break そして next) を含む．他の reserved 語はそれらがあたかも名前であるように使うことが出来る，例えば TRUE, NA そして Inf.

S4 メソッドのドキュメント

形式的('S4')メソッドの作者は，関数のメソッドの全般的なドキュメントと並んで，特定のメソッドに関するドキュメントを提供できる． "?" 演算子はこのドキュメントに三つの方法でアクセスすることを許す．

表現式 methods?*f* は関数 *f* に対する全般的なドキュメントメソッドを探す．現在これは別名 *f*-methods を含むドキュメントファイルを意味する．

特定のメソッドに関するドキュメントを探す二つの異なった方法がある．最初は type 引数を省略して，関数呼び出しの形式で topic 引数を与える．効果はもしこの関数呼び出しが実際に評価されたならば使われるメソッドに関するドキュメントが探される．もし関数が総称的でなければ (それに対して S4 メソッドが定義されていない)，ヘルプは関数名に関するドキュメントに戻る．

"?" 演算子は method として与えられた doc_type と共に呼び出すことが出来る；この場合も topic 引数は関数呼び出しであるが，引数は今や引数のクラスを指定すると解釈され，関数の実際の呼び出し中に現れる実際の表現式としてではない．

最初のアプローチはもし実際の呼び出しが複雑な表現式を含むならば面倒になり，もし引数の評価に時間がかかれば遅くなるかもしれない．二番目のアプローチはこれらの問題を避けるが，実際の引数が評価された時どのクラスになるかを知っている必要がある．

どちらのアプローチも任意の継承メソッドを使う；探されるメソッドのシグネチャは selectMethod を用いて見つけられる (getMethod に対する説明を見よ)．

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

help

曖昧なトピックに関するヘルプページを見つけるには ??.

Examples

```
?lapply

?"for"                  # しかし 引用符/バックチック が必要
?`+`

?women                  # データセット "women" に関する情報

## Not run:
require(methods)
## S4 総称関数とあるメソッドを定義
combo <- function(x, y) c(x, y)
setGeneric("combo")
setMethod("combo", c("numeric", "numeric"), function(x, y) x+y)

## combo とそのメソッドに対するあるドキュメントを書いたと仮定 ....

?combo  # 関数ドキュメントを作成

methods?combo  # 全体的なメソッドのドキュメントを探す
```

```
method?combo("numeric", "numeric")  # 上のメソッドに対するドキュメント

?combo(1:10, rnorm(10))  # ... 同じメソッド,
                         # 引数(一つの整数, 他の数値)に従って選択された

?combo(1:10, letters)    # 既定メソッドに対するドキュメント

## End(Not run)
```

---

rcompgen　　*R* に対する補完生成器*R*

Description

このページは部分的に完成された命令行から関連する補完を生成する機構を解説する．これはそれ自体で有用であることを目指しておらず，それを背景で使う他の機構と共に使われる．用法節でリストされた関数は単純な制御と問い合わせ機構を提供する．実際のインタフェイスは更に下で解説される少数の未移出関数からなる．

Usage

```
rc.settings(ops, ns, args, func, ipck, S3, data, help,
            argdb, fuzzy, quotes, files)

rc.status()
rc.getOption(name)
rc.options(...)

.DollarNames(x, pattern)

## Default S3 method:
.DollarNames(x, pattern = "")
## S3 method for class 'list'
.DollarNames(x, pattern = "")
## S3 method for class 'environment'
.DollarNames(x, pattern = "")
```

Arguments

`ops, ns, args, func, ipck, S3, data, help, argdb, fuzzy, quotes, files`
論理値，ある種のオプションの補完機能をオン・オフにする．

`ops`: `$` と `@` 演算子の後での補完を可能にする．

`ns`: 名前空間に関連した補完を制御する．

`args`: 関数引数の補完を有効にする．

`func`: 関数の検出を有効にする．もし有効にすると，カスタム化可能な拡張(既定では "(")が関数名に付加される．可能な補完が関数かどうかを決定する過程は評価を必要とする．これは大きなオブジェクトに対しては望ましくない，なぜなら遅延ロードに関連するオ

ーバーヘッドに加え可能性として無駄なメモリ使用を含むからである．このためこの機能は既定で無効にされている．

S3: args = TRUE の時は，全ての S3 メソッドの引数に関する補完を有効にする (さもなければ単に総称的関数で，これは普通非常に僅かな引数を持つ).

ipck: library と require 内部でインストールされたパッケージ名の補完を有効にする．

data: data の内部でデータセットの補完を有効にする (すでに可視のものを含む).

help: ヘルプ索引ファイルを見ることで，疑問符で始まるヘルプ要求の補完を有効にする．

argdb: args = TRUE の時，関数引数に関する補完が試みられる．一般的には，適正な引数のリストは args の動的な呼び出しで決定される．これは技術的に正しい結果を与えるが，... 引数の使用はしばしばある種の有用な引数を隠蔽する．この点でより柔軟性を与えるために，指定された関数の適正な引数名のオプションのテーブルが内部的に保持される． argdb = TRUE と設定すると，その中に項目を持つ関数に対するこの内部データベース中の優先的検索を有効にする．勿論，これはデータベースが関心のある関数に対する情報を含む時だけ有用になる．ある種の関数はすでに含まれており，そしてそれ以外もユーザが未移出関数 .addFunctionInfo (下を見よ)を使って付け加えることが出来る．

fuzzy: ファジーマッチングを有効にする，もし正確なマッチングが見つからないと，近いが正確ではないマッチ (例えば大・小文字が違う)が考慮される．この特徴は実験的で詳細は変更される可能性がある．

quotes: 引用符内で R コード中の補完を有効にする．これは普通ファイル名の補完を導くが，しかしさもなければ文脈に依存する可能性がある (例えば，開き引用符が code? に先立つとヘルプ補完が起動される)．これを FALSE に設定すると補完を基礎にある補完フロントエンドに付託する，これはそれ自体の処理を行うことが出来る (例えば Unix 風システム上では readline がファイル名の補完を行う).

files: 廃止予定．代わりに quotes を使う．

全ての設定は ipck, func そして fuzzy を除いて既定では有効にされる．もし自分の CPU サイクルが貴重ならより多くをオフにする；検索リスト中のオブジェクト名に関する基本的な補完は残される．追加の詳細に対しては下を見よ．

name, ... ユーザが設定可能なオプション．現在適正な名前は

function.suffix: 既定 "("

funarg.suffix: 既定 " = "

package.suffix 既定 "::"

使用法は options のそれと同様である．

x "$" の後の適正な値が計算され返される R オブジェクト．

pattern 正規表現．マッチした名前だけが返される．

Details

幾つかの補完タイプがある，あるものは rc.settings を使って無効に出来る．一旦補完機能が活性化されると無効に出来ない最も基本的なレベルは検索パス上で可視の名前と

少数のキーワード(例えば TRUE) を補完する．このタイプの補完はもし補完される部分的な‘ワード’(トークンとも呼ばれる) が空なら試みられない (補完が多すぎるかもしれないため)．より高度な補完は以下で解説される．

抽出演算子 $ そして@ の後の補完： ops 設定が有効な時，$ と @ の後の補完が試みられる．これは接頭辞が評価されることを必要とし，それが明示的な関数呼び出しを含まない限り試みられる ([, $ 等を含む暗黙の関数呼び出しは評価を禁止しない)．

$ 抽出演算子の後の適正な補完は総称的関数 .DollarNames で決定される．ある種のメソッドが提供され，より多くをカスタムクラスのために書くことが出来る．

名前空間中の補完： ns の設定が有効にされると，トークンの前に :: か ::: 演算子が先立つ時名前空間内での補完が試みられる．加えて，基本的な補完機構は全てのロードされた名前空間を含むように拡大される，つまりもし "foopkg" がロードされた名前空間ならば foopkg:: は foo の適正な補完になる．

パッケージの名前空間の補完はすでにロードされた名前空間にだけ適用される，つまりもし MASS がロードされていないと MAS は MASS:: に補完されない．しかしながら，明らかな名前空間内部での保管の試みは，もしそれがまだロードされていないと名前空間をロードしようと試みる，つまり MASS::fr を保管しようとすると，もしまだロードされていないと MASS をロードする．

ヘルプ項目に対する補完： help 設定が有効にされていると，トークンに ? が先立つとヘルプトピックに関する補完が試みられる．特殊文字を含む引用化されたヘルプトピックと並んで，接頭辞(class, method の様な)がサポートされている．

関数引数の補完： args 設定が有効にされていると，適当と思われる場合には関数引数の補完が試みられる．使われる機構はもし関連する関数 (補完が要求された箇所で)が前のプロンプトに入力されていれば現在失敗する(これは特に現在の行が普通 + である継続プロンプトに対してタイプされていることを意味する)．新行で分離しても問題ないことを注意する．

生成される可能な引数補完のリストは誤解を与える可能性がある．非総称的関数に対しては何の問題もない (... が補完候補としてリストされることを除けば；これは関数が追加の引数を受け入れることが出来るという事実を知らせるためで意図的である)．しかしながら，総称的関数に対しては引数を評価することなしに信頼できる引数リストを与えることは現実的に不可能であり(ある場合には，そうしたとしても)，これはリスクがあり(コードするのが難しいことに加え，これがそれが試みられさえしていない真の理由である)，特に引数がそれ自体インラインの関数呼び出しの時はそうである．我々の妥協案はその総称的関数の 全ての現在利用可能なメソッドの引数を考慮するということである．これは二つの欠陥を持つ．第一に，リストされる全ての補完が現在構築されている呼び出し中で適切であるとは限らないかもしれない．第二に，多くのメソッドを持つ総称的関数 (print や plot のような)に対しては，多くのマッチが考慮される必要があり，それは感知されるほどの時間を必要とするかもしれない．これらの欠陥にもかかわらず，この挙動は総称的関数の引数だけをリストするという唯一可能な他の実際的な別法よりもより有用であると我々は信じている．

S3 メソッドだけが現在このような仕方で実装されており，これは S3 設定を用いて無効に出来る．

引数は R 関数呼び出しで名前無しで良いため，他のタイプの補完も引数補完が適当である限り妥当である．任意の特定の関数の形式的引数以外に普通非常に多くの可視のオブジェクトがあるので，可能な引数補完はしばしば他の可能性の一群に埋もれてしまう．しかしながら，基本的補完は空白のトークンに対しては抑制されることを思い出そう．これは関数の可能な引数をリストするためには有用であり得る．例えば seq([TAB] と seq(from = 1, [TAB]) を保管しようとすると，両者は seq (またはその任意のメソッド)の引数だけをリストするが，seq(length[TAB] の補完は length.out 引数と length(関数の両者を可能な補完としてリストする．すでに提供

された引数を取り除くことは試みられないことを注意しよう，更なる速度のペナルティーを招くからである．

特殊関数： 少数の特殊関数 (library, data 等)に対しては，普通の補完が抑制され，設定でそれが要求されるとある種の関数固有の補完が有効にされるという意味で，最初の引数は特別に扱われる． library と require がインストールされたパッケージに対して補完するかどうかを制御する ipck 設定は installed.packages への最初の呼び出しが可能性として時間を喰うために既定では無効にされている (例えばパッケージが遠隔ネットワークファイルサーバーにインストールされている場合)．しかしながら， installed.packages の呼び出しはキャッシュされているので次からの呼び出しは普通速いことを注意する，従ってこのオプションを有効にすることはそうした状況ですら特別面倒では無い．

Value

もし rc.settings が引数無しで呼び出されると，これは現在の設定を名前付き論理値ベクトルとして返す．さもなければ NULL を不可視で返す．

rc.status はリストとして直近の補完の試みの結果を記録するのに使われる内部 (未移出)環境を返す．これはデバッグ用に役立つ．そうした使い方に対しては， rc.status 自体の呼び出しをタイプする際に補完を使う誘惑に抵抗しなければならない，それはそうすると呼び出しが実行された直近の試みになるからである．

返り値リスト中の第一に興味のある項目は：

comps	直近の .completeToken の呼び出しで生成された可能な補完，文字列．
token	補完された(またはされるべき)トークンで，直近の .assignToken の呼び出し (可能性として呼び出し .guessTokenFromLine の内部)で設定されるようなもの．
linebuffer	行全体，直近の .assignLinebuffer の呼び出しで設定されるようなもの．
start	行バッファー中のトークンの開始位置，直近の .assignStart の呼び出しにより設定されるようなもの．
end	行バッファー中のトークンの終了位置，直近の .assignStart の呼び出しにより設定されるようなもの．
fileName	論理値，カーソルが現在引用符の内部にあるかどうかを指示する．
fguess	カーソルが現在内部にある関数の名前．
isFirstArg	論理値．もしカーソルが関数内部にあれば，それは最初の引数か?

更に settings と options の成分はそれぞれ設定とオプションの現在の値を与える．

rc.getOption と rc.options はそれぞれ getOption と options の様な挙動を持つ．

移出されていない API

パッケージの中には幾つかの移出されていない関数がある．これらのうち，少数はそれを通じてこのパッケージで提供される機能を他の機構がそれを使う API を提供するために特殊である (これらはユーザにより直接使われることを意図していないために移出されない)．これらの関数の使用法は：

```
.assignToken(text)
.assignLinebuffer(line)
.assignStart(start)
.assignEnd(end)
```

```
.completeToken()
.retrieveCompletions()
.getFileComp()

.guessTokenFromLine()
.win32consoleCompletion(linebuffer, cursorPosition,
                        check.repeat = TRUE,
                        minlength = -1)
.addFunctionInfo(...)
```

最初の四つの関数は補完されるべきトークン(text)を指定することで試みられる補完を設定し，そしてそれまでの全行(line)中でトークンがどこにあるか (start と end，整数でなければならない)を指定する．

トークンの可能性のある補完は .completeToken により生成され，補完は .retrieveCompletions を用いて R の文字列ベクトルとして回収できる．ユーザがこの関数の代わりを rc.options("custom.completer") を設定することで指定できる；もし NULL でなければ，この関数は補完の候補を計算するために呼び出される．この機能は R コードとしての補完が適当でない状況で助けることを意図している．より詳細はソースコードを見よ．

もしカーソルが引用符の内部にあれば，補完は抑制されるかもしれない．関数 .getFileComp は .completeToken の呼び出しの後にこれが該当する (TRUE を返す)かどうか，そして別の補完が有用と思われるかどうかを決定するために使うことが出来る．多くの場合ファイル名の補完は妥当な代替物である．

.guessTokenFromLine 関数はすでに行をトークンに分割していないバックエンドと共に使うように提供されている．これは行バッファと終端点 (カーソル位置)がすでに設定されていることを必要とし，そしてそれ自体トークンと開始位置を設定する．これはトークンを文字列として返す．

.win32consoleCompletion は精神としては類似しているが，より Windows GUI (またはむしろそれ自体の補完機能を持たない任意のフロントエンド)向けに調整されている．これは行バッファとカーソル位置を引数に持つことを必要とし，そして三つの成分 addition, possible そして comps を持つリストを返す．現在の位置に曖昧でない拡張があれば，addition はカーソル位置に挿入されるべき追加のテキストを含む．もし二つ以上の可能性があれば，これらは possible 中の予め書式化された文字列の文字列ベクトルとしてか，または comps 中の単一の文字列として得られる． possible は現在の width オプションを使って書式化された行からなるので，それらをコンソールに一時に一行ずつプリントすることはそれらをリストする申し分のない方法になる． comps は同じ補完の空白で分離 (潰された)リストで，フロントエンドがそれをある種の他の流儀で表示したい時用である．

minlength 引数はトークンが短すぎる時は補完を抑制するために使える (これはもしフロントエンドが全てのキー押下を補完するように設定されているときは役に立つ)．もし check.repeat が TRUE ならば，同じ補完がある行中で二度以上要求されたかどうかを検出し，そして曖昧な補完はこの場合にだけ返される．これは GNU の Readline の挙動を真似ようとしており，そこでは単一の TAB は任意の曖昧な部分を補完し，そして複数の可能性は TAB を続けて押す時だけ報告される．

様々なフロントエンドが絡んでくるため，これらの関数の詳細は同様に変更される可能性がある．

関数 .addFunctionInfo は特殊関数に対する許される引数名に情報を付け加えるために使うことが出来る．複数の名前付き引数がその呼び出し中で許され，ここでタグは関数名で値は適正な引数を表す文字列ベクトルである．設定 argdb が TRUE ならば，これらは関連する関数に対する適正な引数名のソースとして使われる．

Note

もし求められていないコードの欠片の評価が煩わしければ ops = FALSE と設定すべきである．さもなければ， foo@ba を補完しようとすると foo が評価され， foo[i, 1:10]$ba を補完しようとすると foo[i, 1:10] が評価される，云々．明示的な関数呼び出し(括弧を含む)はこのようには評価されないので，これはひどすぎることにはならないはずである．しかしながら予約と遅延ロードされるシンボルに影響する であろう．

Author(s)

Deepayan Sarkar, <deepayan.sarkar@r-project.org>

---

read.DIF　　スプレッドシートからのデータ入力

Description

データ交換書式(DIF)のファイルを読み込みそれからデータフレームを作る． DIF は単一のスプレッドシートの様なデータ行列に対する書式である．

Usage

```
read.DIF(file, header = FALSE,
         dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
         row.names, col.names, as.is = !stringsAsFactors,
         na.strings = "NA", colClasses = NA, nrows = -1,
         skip = 0, check.names = TRUE, blank.lines.skip = TRUE,
         stringsAsFactors = default.stringsAsFactors(),
         transpose = FALSE, fileEncoding = "")
```

Arguments

file	そこからデータを読み取るファイルの名前，もしくは connection か完全な URL. 名前 "clipboard" も Windows では使うことが出来，その場合 read.DIF("clipboard") は Windows のクリップボード中の DIF 書式項目を探す．
header	スプレッドシートが変数名をその最初の行として含むかどうかを指示する論理値．もし欠損していれば，値はファイルの書式から決定される： header は最初の行が文字列だけを含みそしてトップ左のセルが空の時，そしてその時だけ TRUE に設定される．
dec	ファイル中で小数点に対して使われる文字．
numerals	倍精度実数への変換が精度を失う数の変換をどのようにするかを指示する文字列， type.convert を見よ．
row.names	行名のベクトル．これは実際の行名を与えるのベクトル，又は行名を含む列を与える単一の数，又は行名を含むテーブル列の名前を与える文字列，で良い． もしヘッダーがあり最初の行が列数よりも一つ少ない欄を含めば，入力中の最初の列は行名として使われる．さもなければ，もし row.names が欠損していれば行は番号付けられる． row.names = NULL の使用は行番号付けを強制する．

col.names
: 変数に対するオプションの名前のベクトル．既定値は列番号が後に続く "V" を使う．

as.is
: read.DIF の既定の挙動は文字列を因子に変換することである．変数 as.is は colClasses で別様に指定されていない列の変換を制御する．その値は論理値ベクトルであるか(値は必要に応じてリサイクルされる)，又は因子に変換されるべきではない列を指定する数値もしくは文字列添字のベクトルである．
注意：R 2.12.1 以前のリリースでは，文字列タイプであるとマークされたセルは論理値，数値，又は複素数に type.convert を用いて read.table に於けるように変換されていた．
注意：数値列のそれらを含む全ての変換を抑止するには colClasses = "character" と置く．
as.is は列毎に(変数ごとでは無く)指定されるので，行名の列 (もしあれば)と全てのスキップされるべき列を含む．

na.strings
: NA 値と解釈されるべき文字列の文字列ベクトル．空の欄は又論理値，数値，そして複素数値欄の欠損値と見做される．

colClasses
: 文字列．列に対して仮定されるクラスのベクトル．必要に応じてリサイクルされるか，またはもし文字列ベクトルが名前付きならば指定されない値は NA とされる．
可能な値は NA (type.convert が使われた時)， "NULL" (列がスキップされた時)，原子的クラスの一つ (論理値，整数，複素数，文字列，バイト値)，または "factor", "Date" もしくは "POSIXct" である．さもなければ "character" から指定された形式的クラスへの変換に対する指定された形式的クラスに対する as メソッド (パッケージ **methods** から)が必要になる．
colClasses は列毎に指定される (変数毎ではない)ので，行名の列を含む(もしあれば)ことを注意する．

nrows
: 読み込まれる行の最大数．負の値は無視される．

skip
: 読み取り前にスキップされるべきデータファイルの行数．

check.names
: 論理値．もし TRUE ならばデータフレーム中の変数の名前はそれらが構文的に正しい変数名であることを保証するためにチェックされる．必要ならば，それらは適正でありそして又重複が無いように補正(make.names を使い)される．

blank.lines.skip
: 論理値：もし TRUE ならば，入力中の空白行は無視される．

stringsAsFactors
: 論理値：文字列ベクトルは因子に変換されるべきか?

transpose
: 論理値で，行と列の解釈を転置すべきかを指示する． Microsoft の Excel は正しく読み込むためには transpose = TRUE を必要とする (非標準的な) DIF ファイルを生成することが知られている．

fileEncoding
: 文字列：もし空でなければ文字列データが再エンコーディング出来るようにファイルに対して(コネクションやクリップボードではない) 使われるエンコーディングを指定する． file に対する‘エンコーディング’節， ‘R Data Import/Export Manual’ そして‘注意’を見よ．

Value

ファイル中のデータの表現を含むデータフレーム (data.frame)．空の入力は col.names が指定されていない限りエラーになり，その時は 0 行のデータフレームが返される：同様に，もし header = TRUE ならばヘッダー行だけを与えると 0 行のデータフレームが返される．

Note

as.is と colClasses 中で参照される列は行名の列を含む(もしあれば).

もし colClasses が六種類の原子的なベクトルクラスの一つとして指定されるとメモリ使用量が減る.

Author(s)

R Core; transpose オプションは Christoph Buser, ETH Zurich

References

DIF 書式指定は http://www.wotsit.org/ を検索することで見つかる；オプションのヘッダーファイルは無視される． https://en.wikipedia.org/wiki/Data_Interchange_Format も見よ.

用語が誤解を招きやすい： Windows はその WinFX システムの一部として 'Windows Data Interchange Format (DIF) data format' を持ち，これは互換性があるかもしれないし無いかもしれない.

See Also

マニュアル *R Data Import/Export*.

scan, type.convert, 固定長の書式化された入力 (*f*ixed *w*idth *f*ormatted input) に対する読み取りは read.fwf； read.table; data.frame.

Examples

```
## read.DIF() は Excel から移出されたファイルに対しては
## transpose = TRUE を必要とするかもしれない
udir <- system.file("misc", package = "utils")
dd <- read.DIF(file.path(udir, "exDIF.dif"), header = TRUE, transpose = TRUE)
dc <- read.csv(file.path(udir, "exDIF.csv"), header = TRUE)
stopifnot(identical(dd, dc), dim(dd) == c(4,2))
```

---

read.fortran　　*Fortran* 風のスタイルで固定書式データを読み込む

Description

Fortran 風のスタイル指定で固定書式データを読み込む.

Usage

```
read.fortran(file, format, ..., as.is = TRUE, colClasses = NA)
```

Arguments

file	そこから読み取るファイル又は connection.
format	文字列ベクトル又はベクトルのリスト． 下の'詳細'節を見よ.
...	read.fwf に対する他の引数.
as.is	文字列を文字列のままにするか?
colClasses	既定値を上書きする変数クラス． 詳細は read.table を見よ.

Details

欄の書式は以下の形式のどれかである： rFl.d, rDl.d, rXl, rAl, rIl，ここで l は列数，d は少数位の数，そして r は繰り返し数．F と D は数値書式で，A は文字，I は整数，そして X はすスキップされる列数．繰り返しコード r と少数位コード d は常にオプションである．長さコード l は r があるときは X 書式を除いて必要とされる．

単一行の記録に対しては，format は文字列ベクトルでなければならない．複数行記録に対してはこれは各行に対する文字列ベクトルのリストであるべきである．

スキップ(X)される列は read.fwf に渡されないので，read.fwf への colClasses, col.names そして類似の引数はこれらの列を参照すべきではない．

Value

データフレーム

Note

read.fortran は実際の Fortran の入力ルーティンを使わないので，書式は良くても Fortran のそれらの大まかな近似である．特に，F または D 書式中の d > 0 は少数位 d を，それが入力ファイル中で明示的に指定されていても，左にシフトする．

See Also

read.fwf, read.table, read.csv

Examples

```
ff <- tempfile()
cat(file = ff, "123456", "987654", sep = "\n")
read.fortran(ff, c("F2.1","F2.0","I2"))
read.fortran(ff, c("2F1.0","2X","2A1"))
unlink(ff)
cat(file = ff, "123456AB", "987654CD", sep = "\n")
read.fortran(ff, list(c("2F3.1","A2"), c("3I2","2X")))
unlink(ff)
# 最初の数は Fortran がそれを読み込むのとは異なって読み取られることを注意：
cat(file = ff, "12.3456", "1234567", sep = "\n")
read.fortran(ff, "F7.4")
unlink(ff)
```

---

read.fwf 固定幅書式ファイルを読み込む

Description

固定幅書式ファイルを data.frame 中に読み込む．

Usage

```
read.fwf(file, widths, header = FALSE, sep = "\t",
         skip = 0, row.names, col.names, n = -1,
         buffersize = 2000, fileEncoding = "", ...)
```

Arguments

file	そこからデータが読み込まれるファイルの名前. 別の方法として，file は connection でよく，これはもし必要なら開かれ，そしてそうならば関数呼び出しの後に閉じられる.
widths	固定幅欄の幅を与える整数ベクトルか，複数行記録に対する幅を与える整数ベクトルのリスト.
header	ファイルがその第一行として変数名を含むかどうかを指示する論理値. もしあれば，名前は sep で区切られなければならない.
sep	文字列；内部的に使われる分離子；ファイル中(ヘッダーを除く)に登場しない文字でなければならない.
skip	スキップする初期行の数；read.table を見よ.
row.names	read.table を見よ.
col.names	read.table を見よ.
n	読み込まれる記録(行)の最大数，既定では制限はない.
buffersize	一回に読みこまれる行の最大数
fileEncoding	文字列：もし空でなければ文字列データが再エンコーディング出来るようにファイル(コネクションではない)に対して使われるエンコーディングを宣言する． file に対するヘルプの‘エンコーディング’節，‘R Data Import/Export Manual’ そして‘注意’節を見よ.
...	read.table に引き渡される追加引数．有用なそうした引数は as.is, na.strings, colClasses そして strip.white を含む.

Details

複数の記録は処理前に単一の行に連結される． file 中の幅ゼロか行末を完全に超えるものは NA で置き換えられる.

負の幅欄はスキップする列を指示するのに使われる，例えば 5 列をスキップするのに -5 を使う．こうした欄は read.table によって無視されるので col.names や colClasses 引数中 (もしあればヘッダー行中にも)に含まれるべきではない.

buffersize 引数を減らすと長い行を持つ大きなファイルを読み取る際にメモリ使用量を減らす． buffersize の増量は十分なメモリが利用できればより高速な処理をもたらす.

read.fwf (read.table では無く)は提供されたファイルを読み込むので，後者の引数 encoding は有用ではない.

Value

内部的に呼び出される read.table により作成される data.frame.

Author(s)

R バージョンは Brian Ripley：オリジナルは Kurt Hornik により Perl で書かれた.

See Also

scan そして read.table.

別の固定幅ファイルについては read.fortran.

Examples

```
ff <- tempfile()
cat(file = ff, "123456", "987654", sep = "\n")
read.fwf(ff, widths = c(1,2,3))    #> 1 23 456 \ 9 87 654
read.fwf(ff, widths = c(1,-2,3))   #> 1 456 \ 9 654
unlink(ff)
cat(file = ff, "123", "987654", sep = "\n")
read.fwf(ff, widths = c(1,0, 2,3))    #> 1 NA 23 NA \ 9 NA 87 654
unlink(ff)
cat(file = ff, "123456", "987654", sep = "\n")
read.fwf(ff, widths = list(c(1,0, 2,3), c(2,2,2))) #> 1 NA 23 456 98 76 54
unlink(ff)
```

---

read.socket ソケットから読み取る，又は書き込む

Description

read.socket は指定されたソケットから文字列を読み取る． write.socket は指定されたソケットに書き込む．どちらもエラーチェックはほとんどされない．

Usage

```
read.socket(socket, maxlen = 256L, loop = FALSE)
write.socket(socket, string)
```

Arguments

socket	ソケットオブジェクト．
maxlen	読み取る文字列の最大長(バイト単位)．
loop	読み取るものが何もなければずっと待つか?
string	ソケットに書き込む文字列．

Value

read.socket は文字列を長さ1の文字列として返す．

write.socket は書き込まれたバイト数を返す．

Author(s)

Thomas Lumley

See Also

close.socket, make.socket

Examples

```
finger <- function(user, host = "localhost", port = 79, print = TRUE)
{
    if (!is.character(user))
        stop("user name must be a string")
    user <- paste(user,"\r\n")
    socket <- make.socket(host, port)
    on.exit(close.socket(socket))
    write.socket(socket, user)
    output <- character(0)
    repeat{
        ss <- read.socket(socket)
        if (ss == "") break
        output <- paste(output, ss)
    }
    close.socket(socket)
    if (print) cat(output)
    invisible(output)
}
## Not run:
finger("root")  ## もしサイトがフィンガーデーモンを提供する時だけ動作する

## End(Not run)
```

---

read.table　　データ入力

Description

テーブル書式のファイルを読み込み，そしてそれからデータフレームを作る．ケースは行に，ファイル中の変数は欄に対応する．

Usage

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)

read.csv2(file, header = TRUE, sep = ";", quote = "\"",
          dec = ",", fill = TRUE, comment.char = "", ...)

read.delim(file, header = TRUE, sep = "\t", quote = "\"",
```

```
            dec = ".", fill = TRUE, comment.char = "", ...)

read.delim2(file, header = TRUE, sep = "\t", quote = "\"",
            dec = ",", fill = TRUE, comment.char = "", ...)
```

Arguments

`file` データがそこから読み取られるファイルの名前．テーブルの各行はファイルの一つの行として登場する．もしこれが絶対パスを含まなければ，ファイル名は現在の作業ディレクトリ `getwd()` に 相対的である．チルド展開はサポートされている箇所では実行できる．これは圧縮ファイルでも良い(`file` を見よ).

別法としては，`file` は可読なテキストモードの connection でも良い (これは必要ならば読み込み用に開かれ，そしてもしそうなら関数呼び出しの最後に `close` される(そして従って破壊される))．(もし `stdin()` が使われるならば，行に対するプロンプトは幾らか混乱するかもしれない．空白行か EOF シグナル，Unix では `Ctrl-D` そして Windows では `Ctrl-Z` で入力を終了する． `stdin()` に関するプッシュバックは復帰前にクリアされる．)

`file` は又完全な URL でも良い．(サポートされる URL スキームについては， `url` に対するヘルプの 'URL' 節を見よ．)

`header` ファイルがその最初の行として変数名を含むかどうかを指示する論理値．もし欠損していれば値はファイルの形式から決定される： `header` は `TRUE` に最初の行が列数より一つ小さい列を含む時，そしてその時だけ設定される．

`sep` 欄を分離する文字．ファイルの各行の値はこの文字で分離される．もし `sep = ""` (`read.table` に対する既定値)ならば分離記号は'空白'，つまり一つ以上の空白，タブ，新行，またはキャリッジリターンである．

`quote` 引用符のセット．引用化を一切無効にするには `quote = ""` を使う．引用符中に埋め込まれた引用符の挙動については `scan` を見よ．引用化は文字として読みこまれた列に対してのみ考慮され，これは `colClasses` が指定されない限りそれらの全てである．

`dec` ファイル中で小数点として使われる文字．

`numerals` 倍精度実数への変換が精度を失う数をどのように変換するかを指示する文字列， `type.convert` を見よ．短縮化出来る．(複素数値入力にも適用される．)

`row.names` 行名のベクトル．これは実際の行名を与えるベクトル，又行名を含むテーブルの列を与える単一の数，又は行名を含むテーブルの列の名前を与える文字列で良い．

もしヘッダがありそして最初の行が列数よりひとつ少ない欄を含めば，入力中の最初の列は行名として使われる．さもなければ，もし `row.names` が欠損していれば，行は番号付けられる．

`row.names = NULL` の使用は行の番号付けを強制する．欠損か `NULL` の `row.names` は '自動的'と考えられる行名を生成する (そして `as.matrix` により保存されない).

`col.names` 変数に対するオプション名のベクトル．既定は列数が後に付く `"V"` を使う．

`as.is` `read.table` の既定の挙動は文字列 (これは論理値，数値，又は複素数)を因子に変換する．変数 `as.is` は `colClasses` で別途指定されていない列の変換を制御する．その値は論理値のベクトル (必要なら値はリサ

イクルされる)か，又は数値ベクトル，又はその列が因子に変換されるべきではない文字添字である．

注意：数値列のそれらを含む全ての変換を抑制するには colClasses = "character" を設定する．

as.is は列毎に指定され(変数毎では無い)，そして行名の列 (もしあれば)そしてスキップされる任意の列を含むことを注意する．

na.strings
: NA 値と解釈されるべき文字列の文字列ベクトル．空欄は又論理値，整数，数値そして複素数欄の欠損値と見做される．テストは空白が入力から除かれた後に起こるので， na.strings 値は予めそれら自体の空白を取り除く必要があるかもしれない．

colClasses
: 文字列．列に対して仮定されるクラスのベクトル．もし名前無しなら必要に応じてリサイクルされる．もし名前付きなら，名前はマッチされ，指定されていない値は必要に応じてリサイクルされる．NA とされる．

可能な値は NA (type.convert が使われた時の既定値)， "NULL" (列がスキップされる時)，原子的ベクトルクラスの一つ (論理値，整数，数値，複素数，文字列，バイト型)，又は "factor", "Date" もしくは "POSIXct"．さもなければ "character" から指定された形式的クラスへの変換に対する as メソッド (パッケージ **methods** から)が必要になる．

colClasses は列毎 (変数毎ではない)に指定されるので，行名の列を含む(もしあれば)．

nrows
: 整数：読み込まれる行の最大数．負や他の不正な値は無視される．

skip
: 整数：データの読み込み前にスキップされるデータファイル中の行数．

check.names
: 論理値．もし TRUE ならば，データフレーム中の変数の名前はそれらが構文的に適正な変数名であることを保証するためにチェックされる．もし必要ならそれらは (make.names を使って)そうなるように補正され，そして重複が無いことを保証する．

fill
: 論理値．もし TRUE なら，行が等しくない長さを持つ場合には，空の欄が暗黙のうちに加えられる．‘詳細’節を見よ．

strip.white
: 論理値．sep が指定されている時だけ使われ，そして引用化されていない character 欄から先頭と末尾の空白を取り去ることを許す(numeric 欄は常に取り去られる)．より詳細は scan を見よ (‘空白’の正確な意味を含む)，列は行名を含むかもしれないことを思い出そうｄ．

blank.lines.skip
: 論理値：もし TRUE なら入力中の空白行は無視される．

comment.char
: 文字列：単一の文字列か空文字列を含む長さが1の文字列ベクトル．コメントの解釈を一切無効にするには "" を使う．

allowEscapes
: 論理値．‘\n’ の様な C 風のエスケープ列を処理するか，それともそのままで読み取るか(既定)? もし引用符の内部でなければ，これらは分離記号として解釈されるかもしれない(しかしコメント文字としてではない)．より詳細については scan を見よ．

flush
: 論理値：もし TRUE ならば scan は要求された欄の最後を読み込んだ後で行末までフラッシュする．これは最後のコメントを置くことを許す．

stringsAsFactors
: 論理値：文字列ベクトルは因子に変換すべきか? これは as.is と colClasses により上書きされ，そのどちらもより細かい調整を許す．

fileEncoding　文字列：もし空でなければファイル(コネクションではない)に対して使われるエンコーディングを宣言するので，文字列データは再エンコーディングすることが出来る． file に対するヘルプの‘エンコーディング’節，‘R Data Import/Export Manual’ そして ‘注意’節を見よ．

encoding　入力文字列に対して仮定されるエンコーディング．これは文字列が Latin-1 か UTF-8 (Encoding を見よ)にあることが分かっているとマークするのに使われる：これは入力を再エンコーディングするのには使えないが，R がネイティブなエンコーディングでエンコーディングされた文字列を処理することを許す(もしこれら二つのどちらかならば)．‘値’節と ‘注意’節を見よ．

text　文字列：もし file が提供されずこれが提供されると，データは text の値をテキストコネクションのから読み取る．文字通りの文字列は R 中の (小さな)データセットを含めるのに使うことが出来ることを注意する．

skipNul　論理値：ナルはスキップすべきか?

...　read.table に渡される追加の引数．

Details

この関数はテーブル型データを R 中に読み込む主要な手段である．

colClasses が指定されない限り，全ての列は文字列の列として読み込まれ，そしてそれから type.convert を使って論理値，整数，数値，複素数値，又は (as.is に依存して)適当ならば因子に変換される．引用符は(既定では)全ての欄で解釈されるので，"42" の様な値の列は整数の列になる．

欄又は行はもし何も含まなければ (もし分離記号が何も指定されていなければ空白を除いて)コメント文字や欄の終わりや行の前では‘ブランク’である．

もし row.names が指定されずそしてヘッダ行が列数より一つ少ない項目を持てば，最初の列は行名とされる．これはデータフレームをそれらがプリントされる書式で読み込むことを許す．もし row.names が指定され，そして最初の列を参照指定なければ，その列はそうしたファイルから捨てさられる．

データ列の数は入力の最初の5行を眺めることで決定される (又はもしそれが5行より少なければ全入力)，又はもしそれが指定されより長ければ col.names の長さから．これはもし fill や blank.lines.skip が真ならば多分拙いので，もし必要なら col.names を指定する(‘例’中に於けるように)．

read.csv と read.csv2 は既定値を除いて read.table と同じである．これらは ‘comma separated value’ ファイル(‘.csv’)やコンマを小数点にそしてセミコロンを欄分離子に使うその変種 (read.csv2)を読み込むことを意図している．同じように，read.delim と read.delim2 は分離されたファイルの読み込み用であり，既定では分離記号は TAB 文字である．これらの変種中の header = TRUE と fill = TRUE，そしてコメント文字は捨て去られることを注意する．

コメント文字の後の残りはスキップされる；引用符はコメント中では処理されない．完全なコメント行は blank.lines.skip = TRUE である限り許される；しかしながら，ヘッダーの前にあるコメントは最初のブランクでない列中にコメント文字を持たなければならない．

埋め込まれた新行を持つ引用化された欄はコメント文字の後を除いてサポートされる．埋め込みのヌルはサポートされていない：それらのスキップ(skipNul = TRUE で)は動作するかもしれない．

Value

ファイル中のデータの表現を含むデータフレーム(data.frame).

空の入力は col.names が指定されない限りエラーになり，その時は 0 行のデータフレームが返される：同様にもし header = TRUE ならばヘッダー行だけを与えると結果は 0 行のデータフレームになる．どちらの場合も列は colClasses が提供されていなければ論理値になる．

結果中の文字列(因子水準を含む)はもし encoding が "latin1" か "UTF-8" ならば宣言されたエンコーディングを持つ．

メモリ使用

これらの関数は大きなファイルを読み込む時は驚くほどの量のメモリを使う可能性がある． 'R Data Import/Export' マニュアル中に詳細な議論があり，ここでの注意を補足する．

もし colClasses が六種類の atomic ベクトルクラスの一つとして指定されるとメモリ使用が減る．各異なった値を文字列として保管することはそれを整数として保管するよりも最大14倍のメモリを使うため，これは多くの異なった数値を取る列を読み込む際に特にそうである可能性がある．

nrows の使用は，穏やかな過剰推定だとしても，メモリ使用を助ける．

comment.char = "" の使用は read.table の既定よりも明らかに速い．

read.table は大きな行列を読み取る正しいツールではなく，特に多くの列を含むものが該当する：これは非常に異なるクラスの列を持つデータフレームを読み込むためにデザインされている．行列に対しては scan を代わりに使う．

Note

as.is と colClasses 中で参照される列は行名の列を含む(もしあれば).

ローカルなエンコーディング中に無い入力の読み込みに対しては二つのアプローチがある．もし入力が UTF-8 か Latin1 であることがわかっていれば，それを宣言するために encoding 引数を使う．もし入力がある他のエンコーディング中にあれば，それは入力時に変換される． fileEncoding 引数はこれを現在のロケール中に再エンコーディングを行うコネクションを設定することで達成される． Windows や UTF-8 ロケールで実行中の他のシステムではこれは可能ではないかもしれない．

References

Chambers, J. M. (1992) *Data for models.* Chapter 3 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

See Also

マニュアル 'R Data Import/Export'.

scan, type.convert，固定幅書式の入力の読み込みには read.fwf ; write.table; data.frame.

count.fields は不正確な記録長になる問題を決定するために役立つかもしれない(下の'例'を見よ).

CSV ファイルの IANA 定義は https://tools.ietf.org/html/rfc4180 (これは分離記号としてカンマと行末に CRLF を必要とする).

Examples

```
## fill = TRUE の時未知の最大欄数を処理するため count.fields を使う
test1 <- c(1:5, "6,7", "8,9,10")
tf <- tempfile()
writeLines(test1, tf)

read.csv(tf, fill = TRUE) # 1 column
ncol <- max(count.fields(tf, sep = ","))
read.csv(tf, fill = TRUE, header = FALSE,
         col.names = paste0("V", seq_len(ncol)))
unlink(tf)

## "インライン" データセット, text= を使う
## 先頭と末尾の空行は自動的に取り除かれることを注意する

read.table(header = TRUE, text = "
a b
1 2
3 4
")
```

---

recover　　エラー後にブラウズする

Description

この関数はユーザに現在アクティブな任意の呼び出しを直接ブラウズすることを許し，そしてエラーオプションとして適当である．表現式 options(error = recover) はこのエラーオプションを使う．

Usage

```
recover()
```

Details

呼びだされた際，recover は現在の呼び出しのリストをプリントし，そしてユーザにそれらの一つを選ぶように促す．標準の R の browser がそれから対応する環境から起動される；ユーザはその環境中で評価されるように通常の R 言語表現式をタイプ出来る．

この呼び出し中でブラウズを終了すると，ブラウザから recover に戻るには c をタイプする．もう少しをブラウズするには別のフレーム番号をタイプするか， recover を終了するには 0 をタイプする．

エラーを眺めるために真に待機したいので無い限り， recover の使用はエラーオプションとしての dump.frames を大幅に取って代わる．もし recover が非対話的モードで呼び出されると，これは dump.frames の様に動作する．大量のデータを含む計算のためには，recover はその中でブラウズするために全ての環境をコピーする必要がないという利点を持つ．もし対話的なデバッグを中止することを決定すれば，任意のフレーム中でブラウズ中に dump.frames を直接に呼び出す(例を見よ)．

Value

有用なものは何も返されない．しかしながら，例で示されるエラーオプションを使ってでは無く関数から recover を直接に起動出来る．この例では recover を終了するために 0 をタイプした後でも実行は継続する．

互換性の注意

R の recover 関数は同じ名前の S 関数と同じように使うことが出来る；従って表示されるエラーオプションはエラー挙動を指定する互換性のある方法である．しかしながら，実際の関数は本質的に無関係で非常に異なってユーザと交互作用する．移動命令 up と down は R バージョンでは存在しない；代わりにブラウザを終了し別のフレームを選択する．

References

John M. Chambers (1998). *Programming with Data*; Springer.
しかしながら上の互換性の注意を見よ．

See Also

対話的な計算に関する詳細については browser；エラーオプションの設定については options；後のデバッグのために現在の環境を保存する dump.frames.

Examples

```
## Not run:

options(error = recover) # エラーオプションを設定

### 対話の例

> myFit <- lm(y ~ x, data = xy, weights = w)
Error in lm.wfit(x, y, w, offset = offset, ...) :
        missing or negative weights not allowed

Enter a frame number, or 0 to exit
1:lm(y ~ x, data = xy, weights = w)
2:lm.wfit(x, y, w, offset = offset, ...)
Selection: 2
Called from: eval(expr, envir, enclos)
Browse[1]> objects() # このフレーム中の全てのオブジェクト
[1] "method" "n"      "ny"     "offset" "tol"    "w"
[7] "x"      "y"
Browse[1]> w
[1] -0.5013844  1.3112515  0.2939348 -0.8983705 -0.1538642
[6] -0.9772989  0.7888790 -0.1919154 -0.3026882
Browse[1]> dump.frames() # オフラインのデバッグのために保存する
Browse[1]> c # ブラウザから退去

Enter a frame number, or 0 to exit
1:lm(y ~ x, data = xy, weights = w)
2:lm.wfit(x, y, w, offset = offset, ...)
Selection: 0 # recover を終了
>
```

```
## End(Not run)
```

---

relist *unlist()* されたオブジェクトの再リストを許す

Description

relist() は少数のメソッドを持つ S3 総称的関数であり， (S3) クラス "relistable" のオブジェクト obj とともに使われると unlist(obj) の簡単な変換を許す.

Usage

```
relist(flesh, skeleton)
## Default S3 method:
relist(flesh, skeleton = attr(flesh, "skeleton"))
## S3 method for class 'factor'
relist(flesh, skeleton = attr(flesh, "skeleton"))
## S3 method for class 'list'
relist(flesh, skeleton = attr(flesh, "skeleton"))
## S3 method for class 'matrix'
relist(flesh, skeleton = attr(flesh, "skeleton"))

as.relistable(x)
is.relistable(x)

## S3 method for class 'relistable'
unlist(x, recursive = TRUE, use.names = TRUE)
```

Arguments

flesh	再リストされるベクトル
skeleton	リスト，その構造が結果の構造を決定する
x	R オブジェクト，典型的にはリスト(又はベクトル).
recursive	論理値．unlist を x のリスト成分に適用すべきか?
use.names	論理値．名前を保存すべきか?

Details

ある種の関数は多くのパラメータを必要とし，これは最も容易に複雑な構造，つまり入れ子のリスト，により表現される，不幸にも，optim と nlm を含む R の多くの数学関数はその値域がベクトルである関数だけに適用される． R は入れ子のリストオブジェクトをベクトル表現に変換する unlist() を持つ． relist()，そのメソッドとここで述べられた機能はベクトルを定型的な構造表現に変換する逆操作を提供する．これは構造化された関数 (optim() のような)が簡単な数学的インタフェイスを持つことを許す.

例えば，多変量正規モデルに対する尤度関数は分散共分散行列と平均ベクトルを必要とする．それをベクトルと行列を含むリストとして表すのが最も便利である．典型的なパラメータは次のようになる

```
list(mean = c(0, 1), vcov = cbind(c(1, 1), c(1, 0))).
```

しかしながら optim は入力としてリストを取る関数には適用できない；それは数値ベクトルだけを好む．解決法は変換である．要求される確率密度を計算する関数 mvdnorm(x, mean, vcov, log = FALSE) を与え，それから次のようにする

```
ipar <- list(mean = c(0, 1), vcov = c bind(c(1, 1), c(1, 0)))
initial.param <- as.relistable(ipar)

ll <- function(param.vector)
{
   param <- relist(param.vector, skeleton = ipar))
   -sum(mvdnorm(x, mean = param$mean, vcov = param$vcov,
                log = TRUE))
}

optim(unlist(initial.param), ll)
```

relist は二つのパラメータを取る：skeleton と flesh である． skeleton は正しい shape を持つが誤った内容を持つ標本オブジェクトとする． flesh は正しい内容を持つが誤った shape を持つベクトルとする．次のようにすると

```
relist(flesh, skeleton)
```

flesh の内容を skeleton に置く． skeleton が flesh 内部の属性として保存されている限り skeleton を明示的に指定する必要はない．特に，もし flesh があるオブジェクト obj から unlist(as.relistable(obj)) で作られたならば， skeleton 属性は自動的に設定される．(これはここでの例には適用されないことを注意する，なぜなら optim は ll に渡される新しいベクトルを作り，その par 引数に対してでは無いからである．)

skeleton が正しい形状を持つ限り，それは unlist の正確な逆になるべきである．これらの等式が成り立つ：

```
relist(unlist(x), x) == x
unlist(relist(y, skeleton)) == y

x <- as.relistable(x)
relist(unlist(x)) == x
```

Value

(S3) クラス "relistable" (そして "list") のオブジェクト．

Author(s)

R Core，Andrew Clausen の提案したコードに基づく

See Also

unlist

Examples

```
ipar <- list(mean = c(0, 1), vcov = cbind(c(1, 1), c(1, 0)))
initial.param <- as.relistable(ipar)
ul <- unlist(initial.param)
relist(ul)
stopifnot(identical(relist(ul), initial.param))
```

REMOVE アドオンパッケージを取り除く

Description

アドオンパッケージを取り除くユーティリティ

Usage

```
R CMD REMOVE [options] [-l lib] pkgs
```

Arguments

pkgs	空白で分離された削除すべきパッケージの名前のリスト.
lib	そこから削除する R ライブラリツリーのパス名. 絶対パスでも相対パスでも良い. '--library=lib' の形式も受け付ける.
options	ヘルプやバージョンに対する追加のオプション.

Details

明示的に lib を指定することなしに R CMD REMOVE pkgs として使われると, パッケージは現在の環境中で実行されている R により使われているライブラリパス中の最初のディレクトリにルートを持つライブラリツリーから取り除かれる.

既定のものの代わりにライブラリツリー *lib* から取り除くためには, R CMD REMOVE -l lib *pkgs* を使う.

より多くの用法情報は R CMD REMOVE --help を使う.

Note

ある種の R のバイナリ配布物は別個のバンドルに REMOVE を持つ, 例えば R-devel RPM.

See Also

INSTALL, remove.packages

remove.packages インストールされたパッケージを取り除く

Description

インストールされたパッケージ/バンドルを取り除き, そして必要に応じてインデックス情報を更新する.

Usage

```
remove.packages(pkgs, lib)
```

Arguments

pkgs	取り除かれるパッケージの名前を持つ文字列ベクトル.
lib	そこからパッケージを取り除くライブラリディレクトリを与える文字列. もし欠損していれば, 既定値は .libPaths() 中の最初の要素である.

See Also

コマンドラインのバージョンは REMOVE ; インストールされたパッケージに対しては install.packages.

---

removeSource 関数から保管されたソースを取り除く.

Description

options("keep.source") が TRUE の時, 関数のオリジナルのソースコードのコピーがそれと共に保管されている. この関数はそのコピーを取り除く.

Usage

```
removeSource(fn)
```

Arguments

fn	そこからソースを取り除く単一の関数.

Details

これは "srcref" と関連属性を取り除く.

Value

ソースが取り除かれた関数のコピー.

See Also

ソース参照記録の説明は srcref, 関数がどのように逆構文解析されるかについては deparse.

Examples

```
fn <- function(x) {
  x + 1 # コメント, ソースの一部として保持されている
}
fn
fn <- removeSource(fn)
fn
```

RHOME *R* のホームディレクトリ

Description

R のホームディレクトリの位置を返す，これはインストールされた R ツリーのルートである．

Usage

```
R RHOME
```

roman ローマ数字

Description

(小さなセットの)整数のローマ数字としての簡単な操作.

Usage

```
as.roman(x)
.romans
```

Arguments

x	アラブ又はローマ数字の数値もしくは文字列ベクトル.

Details

as.roman は内部的には整数として表現されるクラス "roman" のオブジェクトを作り，プリント，書式化，部分抽出，強制変換等の適当なメソッドを持つ，methods(class = "roman") を見よ.

1 と 3899 の間の数だけがローマ数字としてユニークに表現できる.

.romans は基本辞書であり，名前付き character ベクトル.

References

Wikipedia contributors (2006). Roman numerals. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Roman_numerals&oldid=78252134. Accessed September 29, 2006.

Examples

```
## 最初の五つのローマ '数字'.
(y <- as.roman(1 : 5))
## 中央の物.
y[3]
## ローマ数字としての現在の年.
(y <- as.roman(format(Sys.Date(), "%Y")))
## 今日, そして 10, 20, 30 と 100 年前 ...
y - 10*c(0:3,10)

## アラブ数字とローマ数字のミックス：
as.roman(c(NA, 1:3, "", strrep("I", 1:6))) # + NA で "IIIIII" に対しては警告
cc <- c(NA, 1:3, strrep("I", 0:5))
(rc <- as.roman(cc)) # 二つの NA：0 は "roman" ではない
(ic <- as.integer(rc)) # 自動的に動作 [明示的なメソッド無しで]
## 簡単な一貫性チェック -- 結果が {1,2,..,3899} 中の時の算術:
stopifnot(identical(rc, as.roman(rc)), # as.roman(.) は "idempotent"
          identical(rc + rc + (3*rc), rc*5),
          identical(ic, c(NA, 1:3, NA, 1:5)),
          identical(as.integer(5*rc), 5L*ic),
          identical(as.numeric(rc), as.numeric(ic)),
          identical(as.list(rc), as.list(ic)))
```

Rprof　　*R* の実行プロファイルを可能にする

Description

R 表現式の実行のプロファイルを可能にしたり無効にしたりする.

Usage

```
Rprof(filename = "Rprof.out", append = FALSE, interval = 0.02,
       memory.profiling = FALSE, gc.profiling = FALSE,
       line.profiling = FALSE, numfiles = 100L, bufsize = 10000L)
```

Arguments

filename	プロファイルの結果を記録するために使われるファイル. プロアイルを無効にするためには NULL か "" に設定する.
append	論理値：ファイルは上書きするか追加するか?
interval	実数：サンプル間の時間間隔.
memory.profiling	論理値：ファイルにメモリ使用情報を書くか?
gc.profiling	論理値：GC が実行中かどうかを記録するか?
line.profiling	論理値：ファイルに行位置を書くか?
numfiles, bufsize	整数：行プロファイルのメモリ確保

Details

プロファイルを有効にすると別のまたは同じファイルへの既存のプロファイルを自動的に無効にする.

プロファイルは，指定されたファイルに，コールスタックを interval 秒毎に書き込むことで動作する． summaryRprof 関数かラッパスクリプト R CMD Rprof は使用の要約を生成するために出力ファイルを処理するのに使うことが出来る；使用法については R CMD Rprof --help を使う.

時間がどのように測られるかはプラットフォームで異なる： Unix 風システムではこれは R プロセスの CPU 時間であり，従って例えば R が入力を待っていたり system により実行されるプロセスから復帰する時間を除く.

タイミング間隔は実用的には小さ過ぎてはならないことを注意する：一旦タイマーがオフになると．情報は次回のタイミングクリック (恐らく範囲 1–10msec)まで記録されない.

関数はもしそれらがコンテクストを呼び出しスタック上に置く時だけ記録される (sys.calls を見よ)． ある種の primitive 関数はこれをしない：特に type "special" であるものが該当する (より詳細はマニュアル 'R Internals' を見よ).

個々の実行文はもし line.profiling が TRUE でコードがソース参照付きで構文解析ならばプロファイルログに記録される． ソース参照の議論については parse を見よ．既定では実行文位置は summaryRprof 中に示されないが，表示を可能にするオプションに対するヘルプページを見よ.

Note

プロファイルは全てのプラットフォームで利用できるとは限らない．既定では，もし可能ならプロファイルに対するサポートがコンパイルで導入される – これを変更するためには R を '--disable-R-profiling' でコンフィギュアする.

R のプロファイルは C のプロファイルと同じ機構を使い二つを同時に使うことが出来ないため， Rprof を C レベルのプロファイルに対する実行ビルト中で使わないこと.

Note

プロファイラーは R を非同期で中断し，それは実行時の結果を記録するメモリを確保出来ない．これは未知の数のファイルパス名を保管する必要がある行のプロファイルに影響する． numfiles と bufsize 引数はこれらの結果を保持する予め確保されたバッファのサイズを制御する：前者はパスの最大数をカウントし，後者はそれら中のバイト数をカウントする．もしプロファイラーがメモリーを使い果たすと，それは新しい行に対する行情報の記録をスキップし，プロファイルを終了するために Rprof(NULL) が呼び出される時警告を出す.

See Also

"Writing R Extensions" の章 "Tidying and profiling R code" (R ソースツリーのサブディレクトリ 'doc/manual' を見よ).

出力ファイルを解析する summaryRprof.

メモリ使用を追跡する他の方法は tracemem, Rprofmem.

Examples

```
## Not run: Rprof()
## プロファイルされるあるコード
```

```
Rprof(NULL)
## プロファイルされないあるコード
Rprof(append = TRUE)
## プロファイルされるあるコード
Rprof(NULL)
...
## そして詳細で説明されたように出力を事後処理

## End(Not run)
```

Rprofmem　　*R* のメモリ使用のプロファイルを有効にする

Description

R 中のメモリ配置のレポートを有効にしたり無効にしたりする.

Usage

```
Rprofmem(filename = "Rprofmem.out", append = FALSE, threshold = 0)
```

Arguments

filename	メモリ配置を記録するために使われるファイル. 記録を無効にするためには NULL または "" に設定する.
append	論理値：ファイルは上書きされるか追記されるか?
threshold	数値：このバイト数を超える R の "大きなベクトル" ヒープの配置は報告される.

Details

プロファイルを有効にすると別または同じファイルに対する任意の既存のプロファイリングを自動的に無効にする.

プロファイルは malloc が大きなベクトルオブジェクトや小さなオブジェクトに対するメモリのページを配置するために呼び出される度に呼び出しスタックを指定のファイルに書く. メモリのページのサイズと malloc 以上がベクトルのために使われるサイズはコンパイル時の定数であり, 既定ではそれぞれ 2000 と 128 バイトである.

プロファイラは配置を追跡し, その幾つかは以前に使われたメモリに対するもので, R のトータルのメモリ使用を増加させない.

Value

無い

Note

メモリのプロファイルは例え使われていなくても R を遅くするので, コンパイル時オプションにされている. メモリプロファイラは他の R や C のプロファイルと同時に使うことが出来る.

See Also

R のサンプリングプロファイラ Rprof もまたメモリ情報を収集する．

tracemem は指定オブジェクトの重複をトレースする．

マニュアル "Writing R Extensions" の節 "Tidying and profiling R code".

Examples

```
## Not run:
## R がそれをサポートするようにコンパイルされていなければサポートされない
Rprofmem("Rprofmem.out", threshold = 1000)
example(glm)
Rprofmem(NULL)
noquote(readLines("Rprofmem.out", n = 5))

## End(Not run)
```

Rscript　　*R* に対するスクリプトフロントエンド

Description

これは '#!' スクリプト中で使う別のフロントエンドと他のスクリプトアプリケーションである．

Usage

```
Rscript [options] [-e expr [-e expr2 ...] | file] [args]
```

Arguments

options	オプションのリストで，全て '--' で始まる．これらは標準の R フロントエンドの任意のオプション，また詳細で解説されるもので良い．
expr, expr2	R 表現式で，適切に引用化されている．
file	R 命令を含むファイル名．'-' は 'stdin' を指示する．
args	file 中のスクリプトに渡される引数．

Details

Rscript --help は用法の詳細を与え，そして Rscript --version は Rscript のバージョンを与える．

他の起動は選択されたオプションで R フロントエンドを起動する．このフロントエンドは実行可能であり file を直接引数として取るため '#!' スクリプトを書くために便利である．オプション '--slave --no-restore' は常に提供される：これらは '--no-save' を意味する．(標準の Windows のコマンドラインは '#!' スクリプトの概念を持たないが，Cygwin シェルは持つ．)

一つ以上の '-e' オプションか file の どちらかが提供されるべきである．'-e' オプションを使うときは使われるシェルの引用化規則に注意する：例を見よ．

受け入れられる追加のオプション (file か args の前)は

'--verbose' Rscript が実行することの詳細を与える．同様に R に渡される．

'--default-packages=list' ここで list はコンマで分離されたパッケージ名のリストか NULL．開始時にロードされるパッケージを決める環境変数 R_DEFAULT_PACKAGES を設定する．既定では Rscript はスタートアップ時間の約 60% を占める **methods** を除外する．

expression と file 中に空白があっても良い (しかしもしあれば使用中のシェルから守られなくてはならない，例えば引数を引用符で囲む).

普通 R のバージョンはインストール時に決定されるが，これは環境変数 RHOME の設定で上書きできる．

stdin() は入力ファイルを参照し，そして file("stdin") はプロセスの stdin ファイルストリームを参照する．

Note

Rscript は execv システムコールを持つシステム上でだけサポートされている．

Examples

```
## Not run:
Rscript -e 'date()' -e 'format(Sys.time(), "%a %b %d %X %Y")'

## Unix 風システムでの #! スクリプトの例

#! /path/to/Rscript --vanilla --default-packages=utils
args <- commandArgs(TRUE)
res <- try(install.packages(args))
if(inherits(res, "try-error")) q(status=1) else q()


## End(Not run)
```

RShowDoc　　*R* マニュアルと他のドキュメントを示す

Description

R ドキュメントを見つけ表示するユーティリティ関数.

Usage

```
RShowDoc(what, type = c("pdf", "html", "txt"), package)
```

Arguments

what	文字列：'詳細'節を見よ．
type	好みの書式を与えるオプション文字列．短縮化出来る．
package	ドキュメントをその中で探すパッケージの名前を指定するオプションの文字列.

Details

what は幾つかの異なったドキュメントソースの一つを指定でき， R マニュアル(R-admin, R-data, R-exts, R-intro, R-ints, R-lang), NEWS, COPYING (GPL ライセンス), 'share/licenses' 中の任意のライセンス， FAQ (R-FAQ としても得られる)， そして '*R_HOME*/doc' 中のファイルを含む.

Windows に対してだけ， Windows FAQ に対する R は rw-FAQ で指定される.

もし package が指定されると， ドキュメントは 'doc' とその名前のインストール済みのパッケージのトップレベルのディレクトリ中で探される.

もし what が欠損していれば簡単な使用法メッセージがプリントされる.

ドキュメントのタイプは先ず type 中(または指定されなければ "pdf") で指定されたものから始めて順に試みられる.

Value

見つかったファイルへのパスを与える不可視の文字列.

See Also

通常のヘルプファイルを表示するには help (または ?) そして help.start.

type = "txt" に対しては file.show が使われる． vignette は RShowDoc(*, package= .) で見栄え良く閲覧できる.

Examples

```
RShowDoc("R-lang")
RShowDoc("FAQ", type = "html")
RShowDoc("frame", package = "grid")
RShowDoc("changes.txt", package = "grid")
RShowDoc("NEWS", package = "MASS")
```

RSiteSearch　　ドキュメント中のキーワードや語句を探す

Description

ヘルプページ， ビニエットまたはタスクビュー中のキーワードや語句を， http://search.r-project.org にある検索エンジンを使い探索し， それらをウェッブブラウザ中で眺める.

Usage

```
RSiteSearch(string,
            restrict = c("functions", "vignettes", "views"),
            format = c("normal", "short"),
            sortby = c("score", "date:late", "date:early",
                       "subject", "subject:descending",
                       "from", "from:descending",
                       "size", "size:descending"),
            matchesPerPage = 20)
```

Arguments

string	検索するキーワードや単語や語句を指定する文字列．もし単語が一つの項目として検索されるべきならば全てを括弧で囲む(最初の例を見よ).
restrict	文字列ベクトルで，典型的には長さは1より大きい．検索する可能な分野：ヘルプページに対しては functions，タスクビューに対しては views そしてパッケージのビニエットに対しては vignettes.
format	normal または short (引用なし)；短縮化出来る.
sortby	検索結果をどのようにソートするかを指示する文字列 (短縮化可能)：(score, 直近の結果を最初にする日付でソートするには date:late，最も古いものを最初にするには date:early，主題のアルファベット順には subject，アルファベットの逆順には subject:descending，送り手(もし可能なら)に対しては from か from:descending，サイズに対しては size か size:descending．)
matchesPerPage	ページ毎に幾つの項目を表示するか.

Details

この関数は検索サイト http://search.r-project.org を使って動作するようにようにデザインされており，そしてこのサイトが利用可能であり続けることに依存する (Jonathan Baron と School of Arts and Sciences of the University of Pennsylvania に感謝する).

ユニークな部分マッチは全ての引数に対して動作する．各新しい部分ウィンドウはそれを閉じない限り開き続ける.

Value

(不可視で)ブラウザに渡される完全な URL で，検索文字列を含む.

Author(s)

Andy Liaw と Jonathan Baron

See Also

ローカルな検索に対しては help.search, help.start.

ヘルプファイルがどのように表示されるかについては browseURL.

Examples

```
# インターネット接続が必要
RSiteSearch("{logistic regression}") # 正確な語句にマッチ
Sys.sleep(5) # ブラウザを開くことを許す，一瞥する
## ビニエット中を探しそして検索文字列を保存：
fullquery <- RSiteSearch("lattice", restrict = "vignettes")
fullquery # 約100文字の文字列
```

rtags *R* に対する *Etags* 風のタグ付けユーティリティ

Description

rtags は R コードに対する etag 風の添え字化機能であり， R 自体の構文解析を用いる．

Usage

```
rtags(path = ".", pattern = "\\.[RrSs]$",
      recursive = FALSE,
      src = list.files(path = path, pattern = pattern,
                       full.names = TRUE,
                       recursive = recursive),
      keep.re = NULL,
      ofile = "", append = FALSE,
      verbose = getOption("verbose"))
```

Arguments

path, pattern, recursive
: タグ付けされるファイルを決定する list.files に渡される引数．既定では，これらは現在のディレクトリ中の拡張子 .R, .r, .S そして .s を持つ全てのファイルである．これらの引数はもし src が指定されると無視される．

src
: 索引が作られるファイル名のベクトル．

keep.re
: src (索引が作られるファイル)を更に制限する正規表現．例えば，指定 keep.re = "/R/[^/]*\\.R$" は R という名前のディレクトリ内部の拡張子 .R を持つファイルだけを残す．

ofile
: file 引数として cat に渡される；典型的にはタグが書き込まれる出力ファイル(規約で "TAGS")．既定では，出力は R コンソールに書き込まれる(リダイレクトされていない限り)．

append
: 論理値，出力は既存のファイルを上書きするか，それともそれに付け加えるかを指示する．

verbose
: 論理値．もし TRUE ならば，ファイル名は処理につれて R コンソールにエコーされる．

Details

多くのテキストエディタはタグ付けユーティリティでソースファイル中の位置を素早く容易に見つけられる関数や他の言語オブジェクトの定義を許す．この機能は関連するソースファイルの事前処理を必要とし，名前とそれらの対応する位置を含む索引(タグ)ファイルを作る．複数のタグファイル書式があり，最もポピュラーなものは vi スタイルの ctags 書式と emacs スタイルの etags 書式である．これらの書式のタグファイルは普通それぞれ ctags と etags ユーティリティで作られる．不幸にも，これらのプログラムは R コードの構文を認識しない．これらは正規表現を使った任意の言語ファイルのタグ付けを許すが，これでは不十分である．

rtags 関数は R コードに対するタグ付けユーティリティになることを意図している．これは R コードファイルを(R の構文解析器を用い)構文解析し Emacs の etag 書式でタグを生成する． vi スタイルのタグは現在無いが，付け加えるのは困難ではないはずである．

Author(s)

Deepayan Sarkar

References

https://en.wikipedia.org/wiki/Ctags, https://www.gnu.org/software/emacs/manual/html_node/eintr/etags.html

See Also

list.files, cat

Examples

```
## Not run:
rtags("/path/to/src/repository",
pattern = "[.]*\\.[RrSs]$",
keep.re = "/R/",
verbose = TRUE,
ofile = "TAGS",
append = FALSE,
recursive = TRUE)

## End(Not run)
```

Rtangle　　*Stangle* に対する *R* ドライバー

Description

R コードの塊を抜き出す Stangle に対するドライバー.

Usage

```
Rtangle()
RtangleSetup(file, syntax, output = NULL, annotate = TRUE,
             split = FALSE, quiet = FALSE, ...)
```

Arguments

file	Sweave ソースファイルの名前. Sweave の対応する引数の解説を見よ. .
syntax	クラス SweaveSyntax のオブジェクト.
output	split = TRUE で無い限り使われる出力ファイル名：'詳細'を見よ.
annotate	既定では, コードの塊はコードの塊の名前と数を指定するコメント行で分離される. もし FALSE ならば装飾コメントは省略される.
split	出力を各コードの塊ごとに分割するか?
quiet	もし TRUE ならば全ての進行メッセージは抑制される.
...	追加のオプションに対する既定値を設定する名前付き引数.

Details

split = TRUE で無い限り，出力ファイルの既定名は basename(file) で Sweave のシンタックスに対応する拡張子 (つまり 'Rnw', 'Stex') を 'R' で置き換えたものになる．ファイル名 "stdout" と "stderr" はそれぞれ出力とメッセージコネクションと解釈される．

もし分割が選択されると(ファイル中のオプションにより含む)，各塊は別個のファイルに書き込まれ，拡張子は 'エンジン'名になる(既定では '.R')．

注釈は次の形式の一つであり

```
##################################################
### code chunk number 3: viewport
##################################################

##################################################
### code chunk number 18: grid.Rnw:647-648
##################################################

##################################################
### code chunk number 19: trellisdata (eval = FALSE)
##################################################
```

塊ラベルかファイル名と行番号を使う．

コードの塊は前にある塊を再利用できるので，このドライバは単にコードの塊をそのまま取り出すのではないことを注意する．

サポートされるオプション

Rtangle はコードの塊に対する次のオプションをサポートする (括弧内の値は既定値)：

engine: 文字列("R")． engine が "R" か "S" に等しい塊だけが処理される．

keep.source: 論理値(TRUE)． もし keep.source == TRUE ならばオリジナルのソースがファイルにコピーされる．さもなければ構文解析されたソースが出力される．

eval: 論理値(TRUE)． もし FALSE ならばコードの塊は横切ってコピーされるがコメントアウトされる．

prefix もし split = TRUE ならば使われる． prefix.string を見よ．

prefix.string: 文字列，既定値はソースファイル名(拡張子無し)． もし split = TRUE ならば，塊がラベルを持たないか，それがラベルを持ち prefix = TRUE ならばファイル名に対する接頭辞として使われる．これはファイル名の一部として使われるので可搬性が無ければならないことを注意する．

show.line.nos 論理値(FALSE)． 出力は塊の最初のコード行の行番号を示すコメントで注釈されるべきか?

Author(s)

Friedrich Leisch と R-core.

See Also

utils パッケージ中のビニエット 'Sweave User Manual'.

Sweave, RweaveLatex

RweaveLatex *Sweave* に対する *R/LaTeX* ドライバー

Description

R コードの塊を LaTeX ファイルに翻訳する Sweave に対するドライバー.

Usage

```
RweaveLatex()

RweaveLatexSetup(file, syntax, output = NULL, quiet = FALSE,
                 debug = FALSE, stylepath, ...)
```

Arguments

file	Sweave ソースファイルの名前. Sweave の対応する引数の解説を見よ.
syntax	クラス SweaveSyntax のオブジェクト.
output	出力ファイル名. 既定値は拡張子 '.nw', '.Rnw' または '.Snw' を取り去り拡張子 '.tex' を加える. file 中の任意のディレクトリパスも取り除かれ出力が現在の作業ディレクトリ中に作成されるようにする.
quiet	もし TRUE なら全ての進行メッセージは抑制される.
debug	もし TRUE ならば, 全てのコードの塊の入力と出力はコンソールにコピーされる.
stylepath	'詳細'節を見よ.
...	'サポートされるオプション'節に挙げられたオプションの名前付きリスト.

Details

生成される LaTeX ファイルは '\usepackage{Sweave}' という行を含まねばならず, もしこれが Sweave ソースファイル中に無ければ(コメント中かもしれない), それが RweaveLatex ドライバーで挿入される. もし stylepath = TRUE ならば R のインストール中のファイル 'Sweave.sty' のハードコードされたパスが Sweave の代わりに設定される. ハードコードされたパスは LaTeX ファイルの可搬性を低くするが, 'Sweave.sty' の現在のバージョンを使用中の TeX 入力パス中のどこかにインストールする手間を省く. しかしながら TeX はハードコードされたパスが空白 (Windows ではしばしば起きる)か TeX の特殊文字を含むと処理できないかもしれない.

stylepath に対する既定値は環境変数 SWEAVE_STYLEPATH_DEFAULT から取られるか, またはそれが未設定か空ならば FALSE となる. もし設定されていれば, それは正確に TRUE か FALSE でなければならない : 他の任意の値は FALSE とされる.

頻繁に Sweave を使うユーザに対する 'Sweave.sty' が TeX の入力パスにあることを保証する最も簡単な方法は '*R_HOME*/share/texmf' を 'texmf ツリー'として加えることである ('MiKTeX 設定'ユーティリティの言葉では 'ルートディレクトリ').

既定では, 'Sweave.sty' は全ての含められるグラフィックスの幅を次に設定する : '\setkeys{Gin}{width=0.8\textwidth}'.

この設定は '\includegraphics{}' ディレクティブに渡される幅サイズオプションに影響し, 次いで最終ドキュメント中に現れるプロットファイルのスケーリングに影響する.

このように，例えば，図の塊に width=3 を設定すれば生成されたグラフィックスは幅が3インチになる．しかしながら，最終的なドキュメント中のグラフィックスの幅は '0.8\textwidth' に設定され，高さはそれに従ってスケールされる．フォントとシンボルは最終ドキュメント中で同様にスケール化される．

'.Rnw' ファイル中 '\begin{document}' ディレクティブの後に '\setkeys{Gin}{width=...}' ディレクティブを含めることで既定値を調整でき，標準的な LaTeX の測定値を使って width オプション値を好みに変更できる．

もしこの既定動作を完全に上書きしたければ，プリアンブル中に '\usepackage[nogin]{Sweave}' ディレクティブを加えることが出来る．この場合，如何なるサイズ/スケールオプションも '\includegraphics{}' ディレクティブに加えることは出来ず，そして height と width オプションが最終ドキュメント中の実行時に生成されるグラフィックファイルサイズとグラフィックスのサイズの双方を決定する．

'Sweave.sty' は又 '[noae]' オプションをサポートし，これは 'ae' パッケージを抑制し，その使用はあるエンコーディングとタイプフェイスに影響する．もしある文字セットの表示に問題があれば，このオプションを試みよ．

R 3.1.0 以来これは '[inconsolata]' オプションもサポートし， inconsolata 中の等幅テキストを R のヘルプページで使われるフォントで表示する．

ファンシーな引用符(sQuote を見よ)の使用は R 出力の設定時に問題を惹き起こす可能性がある． options(useFancyQuotes = FALSE) を設定するか LaTeX が使用されるエンコーディング (宣言 '\usepackage[utf8]{inputenc}' により： Rgui.exe からの Sweave の Windows ユーザは 'utf8' を 'cp1252' で置き換える等の必要がある) を理解し，そして向き付きの引用符を含むタイプライタフォントが使われることを保証するように按配する．

ある種の LaTeX グラフィックスドライバは既知の拡張のリスト中に '.png' または '.jpg' を含まない．それらを有効にするためには '\DeclareGraphicsExtensions{.png,.pdf,.jpg}' のようなものをドキュメントのプリアンブルに加えるか，グラフィックスドライバはの挙動をチェックする． pdf と png の双方が TRUE の時は，どちらのファイルも Sweave により生成され， 'DeclareGraphicsExtensions' 中のそれらの順序は pdflatex によりどれが使われるかを決める．

サポートされるオプション

RweaveLatex はコードの塊に対する次のオプションをサポートする (カッコ内の値は既定値を示す)． 文字列値は Sweave から ... 経由で渡されるときは引用化されなければならないがコードの塊のヘッダー中で使われるときは不要である．

engine: 文字列 ("R")． engine が "R" か "S" の塊だけが処理される．

echo: 論理値(TRUE)． 出力ファイル中に R コードを含めるか?

keep.source: 論理値(TRUE). エコー時にもし keep.source == TRUE ならばオリジナルのソースがファイルにコピーされる．さもなければ構文解析されたソースがエコーされる．

eval: 論理値(TRUE)． もし FALSE ならばコードの塊は評価されす，従ってテキストもグラフィカルな出力も生成されない．

results: 文字列("verbatim")． もし "verbatim" ならば R 命令の出力が verbatim 風の 'Soutput' 環境に含められる．もし "tex" ならば出力はすでに適正な LaTeX マークアップとされ，そのように含められる．もし "hide" なら全ての出力は完全に抑制される (しかしコードは weave の間に実行される)． 値は短縮可能である．

print: 論理値(FALSE)． もし TRUE ならばこれは全ての表現式の自動プリントを強制する．

term: 論理値(TRUE). もし TRUE ならば，値の可視性は対話的な R セッションを模擬する：付値の値はプリントされず，単一のオブジェクトの値はプリントされる．もし FALSE ならば出力は明示的な print や同様の実行文からだけ由来する．

split: 論理値(FALSE). もし TRUE ならば，テキスト出力は各コードの塊ごとに別個のファイルに書き出される．

strip.white: 文字列("true"). もし "true" ならば，出力の先頭と末尾の空白行は取り除かれる．もし "all" ならば，全ての空白行が出力から除かれる．もし "false" ならば，空白行は保存される．

'空白行'とは空かホワイトスペース (空白とタブ)だけを含むものを言う．

コードの塊中の空白行は普通空白行の出力の代わりにプロンプトを生み出す．

prefix: 論理値(TRUE). もし TRUE なら生成された図のファイル名と出力は全て prefix.string オプションで与えられる共通の接頭辞を持つ：さもなければラベル無しの塊が接頭辞を使う．

prefix.string: 文字列で，既定値はソースファイル名(拡張子無し). これはファイル名の一部として使われるため，可搬性が必要になることを注意する．

include: 論理値(TRUE)で，テキスト出力(もし split = TRUE なら) に対する '\includegraphics' 文が自動生成されるかどうかを指示する．もし出力がコードの塊以外の場所に現れるべきであれば include = FALSE を使う(入力行を手動で置くことで).

fig: 論理値(FALSE)で，コードの塊がグラフィックカルな出力を生成するかどうかを指示する．コードの塊毎に一つの図だけがこの方法で処理できることを注意する．図の塊に対するラベルはファイル名の一部として使われるので，アルファベットと数字であることが望ましい．

eps: 論理値(FALSE)で， EPS 図を生成すべきかどうかを指示する．もし fig = FALSE なら無視される．

pdf: 論理値(TRUE)で， PDF 図を生成すべきかどうかを指示する．もし fig = FALSE なら無視される．

pdf.version, pdf.encoding, pdf.compress: バージョン，エンコーディングそして圧縮を設定する (またはしない)ために pdf に渡される．既定値は pdf.options() から取られる．

png: 論理値(FALSE)で， PNG 図を生成するかどうかを指示する．もし fig = FALSE なら無視される． R >= 2.13.0 でだけ利用可能．

jpeg: 論理値(FALSE)で， JPEG 図を生成するかどうかを指示する．もし fig = FALSE ならば無視される． R >= 2.13.0 でだけ利用可能．

grdevice: 文字列(NULL)：節 'カスタムグラフィックスデバイス'を見よ．もし fig = FALSE なら無視される． R >= 2.13.0 でだけ利用できる．

width: 数値(6)で，インチ単位の図の幅． '詳細'節を見よ．

height: 数値(6)で，インチ単位の図の高さ． '詳細'節を見よ．

resolution: 数値(300)で，インチ毎のピクセル数の解像度： PNG と JPEG グラフィックスで使われる．既定値はかなり高い値で，高品位のプロットに相応しいことを注意する． 100 程度がパッケージのビニエットに対してはより良い選択である．

concordance: 論理値(FALSE). 入力行番号と出力行番号をリンクする索引ファイルを書く．これは実験的な特徴である；出力書式に対するソースコードを見よ，これは将来のリリースで変更されるかもしれない．

figs.only: 論理値(FALSE). 既定では各図の塊は一回実行され，それから図の各選択タイプに対して再実行される．これは最初の図の塊に対する既定のグラフィックスデバイスを開き，そして全ての引き続く塊の最初の評価に対してそのデバイスを使う．も

しこのオプションが真なら，図の塊はグラフィックスの各選択タイプに対してだけ実行され，それらに対して新しいグラフィックスデバイスが開かれそして閉じられる．

加えて，ユーザは追加のオプションを個々のコードセクションのヘッダー中かドキュメントの '\SweaveOpts{}' 行中で追加のオプションを指定できる．未知のオプションに対しては，それらのタイプは最初の使用時に設定される．

カスタムグラフィックスデバイス

もしオプション grdevice が fig と eval が真のコードの塊に対して与えられると．次の呼び出しが行われる

```
get(options$grdevice, envir = .GlobalEnv)(name=, width=,
                                          height=, options)
```

これはグラフィックスデバイスを開くべきである．塊のコードはそれから評価されて dev.off が呼び出される．普通与えられた名前の関数は Sweave ドキュメントの前の方で定義されている，つまり

```
<<results=hide>>=
my.Swd <- function(name, width, height, ...)
  grDevices::png(filename = paste(name, "png", sep = "."),
                 width = width, height = height, res = 100,
                 units = "in", type = "quartz", bg = "transparent")
@
```

現在各塊に対してただ一つのカスタムデバイスが定義可能であるが，異なった塊に対しては異なったデバイスを使うことが出来る．

dev.off に対する置き換えが接頭辞 .off を持つ関数として提供できる，例えば my.Swd.off().

フック関数

各塊が評価される前に，ゼロかそれ以上のフック関数を実行できる．もし設定 getOption("SweaveHooks") がされていると，それはフック関数の名前付きリストとされる．コードの塊の各 *logical* オプション (echo, print, ...) に対してフックを指定でき，それらは各オプションが TRUE の時そしてその時だけ実行される．フックは getOption("SweaveHooks") が返すリストの名前付き要素でなければならず関数は引数を持たない．例えば，もしオプション "SweaveHooks" が list(fig = foo) として定義され foo が関数ならば，それは各図の塊中のコードに対して実行される．これは特に一連の図の塊中のグラフィカルパラメータの既定値を設定するのに有用である．

ユーザは自由に新しい Sweave の論理値オプションを定義しそれらに任意のフックを関連付けることが出来ることを注意する．例えば，作業空間中の全てのオブジェクトを消去する clean と呼ばれる新しいオプションに対するフック関数を定義できるであろう．そうすると clean = TRUE で指定された全てのコードの塊は空の作業空間上で操作を開始するであろう．

Author(s)

Friedrich Leisch と R-core

See Also

utils パッケージ中のビニエット ‘Sweave User Manual’.

Sweave, Rtangle

---

savehistory　　命令履歴をロード，保存，又は表示する

Description

命令履歴をロード，保存，又は表示する.

Usage

```
loadhistory(file = ".Rhistory")
savehistory(file = ".Rhistory")

history(max.show = 25, reverse = FALSE, pattern, ...)

timestamp(stamp = date(),
          prefix = "##------ ", suffix = " ------##",
          quiet = FALSE)
```

Arguments

file	その中に履歴を保存する，又はそこから履歴をロードするファイルの名前．パスは現在のディレクトリに対して相対的である.
max.show	表示する最大行数． Inf は現在利用できる全ての履歴を与える.
reverse	論理値．もし真ならば，行は逆順に示される．注意：これは継続行がある場合は役に立たない.
pattern	履歴の行に対してマッチされるべき文字列．提供された際は，ユニークにマッチした行だけが表示される.
...	マッチングを行う際に grep に渡される引数.
stamp	履歴中に書き込まれる値もしくは値のベクトル.
prefix	各行に適用される接頭辞.
suffix	各行に適用される接尾辞.
quiet	もし TRUE ならば，コンソールにタイムスタンプのプリントを抑制する.

Details

異なった R コンソールに対して利用できる幾つかの履歴機構があり，同じようだが同一ではない仕方で動作する． Unix と Windows に対するこのヘルプの別個のバージョンがある.

ここで解説される関数は readline コマンド行インタフェイスの下での Unix 風システムで動作するが，さもなければ動作しないかもしれない (例えば，バッチ使用や埋め込みアプリケーション中)． R は readline 無しで構築できることを注意する.

OS X 上のコンソール R.app は別個で相当互換性のない履歴機構を持ち，既定ではファイル '.Rapp.history' を使い最大 250 個の項目を保存する．これらの関数は現在そこでは実装されていない．

readline 履歴機構は二つの環境変数で制御される： R_HISTSIZE は保存される行数 (既定値 512)を制御し，R_HISTFILE (既定 '.Rhistory') はセッションの初め/終わりに要求されると (しかし loadhistory/savehistory に対しては既定ではない) 履歴のロード/保存に使われるファイル名を設定する．セッション中に保存される履歴の行数には制限がないので， R_HISTSIZE を大きな値に設定することは大きなファイルが実際に作られない限りペナルティにはならない．

これらの環境変数は保存時に読み取られるので，セッション中に Sys.setenv を使って変更できる．

readline 履歴ライブラリはファイルをパーミッション 0600 で保存することを注意する，つまりユーザに対しては読み取り/書き込み許可を与え，他のアカウント対しては読み取り許可すら無い．

timestamp 関数は履歴中にタイムスタンプ (又は他のメッセージ)を書きそれをコンソールにエコーする．履歴機構をサポートしないプラットフォームではコンソールメッセージだけがプリントされる．

Note

もし(ほとんど)全ての対話的セッションの最後に履歴を保存したければ (作業空間中に保存したくないものを含め)，呼び出し savehistory() を .Last 中に置くことが出来る．例を見よ．

Examples

```
## Not run:
## 履歴をホームディレクトリに保存：それは(既定では)そこから読み取り
## 現在のディレクトリではないことを注意する
.Last <- function()
    if(interactive()) try(savehistory("~/.Rhistory"))

## End(Not run)
```

---

select.list　　リストから項目を選択する

Description

文字列ベクトルから項目を選択する．

Usage

```
select.list(choices, preselect = NULL, multiple = FALSE,
            title = NULL, graphics = getOption("menu.graphics"))
```

Arguments

choices	項目の文字列ベクトル.
preselect	文字列ベクトル又は NULL. もしヌルでなくそしてもし文字列がリスト中にあれば, その項目が先ず選ばれる.
multiple	論理値：一つより多い項目を選択できるか?
title	ウィンドウに対するオプションの文字列, 又はタイトル無しなら NULL.
graphics	論理値：グラフィカルなウィジェットを使うべきか?

Details

正規の既定は graphics = TRUE である. OS X の GUI の下ではこれは (スクロール可能な)項目リストを持つモーダルな対話ボックスを登場させ, それはマウスで選択できる. Unix 風のプラットフォームではこれは可能ならば Tcl/Tk のリストボックスウィジェットを使う.

もし graphics が FALSE か又はグラフィカルなウィジェットが使えなければ, これはそこから番号で選択できるテキストリストを表示する. multiple = FALSE の場合は menu を使う. 予めの選択は multiple = TRUE に対してだけサポートされ, そこでは項目の前の "+" で指示される.

非対話的なセッションで select.list を使うことはエラーである.

Value

選択された項目の文字列ベクトル. もし multiple が偽でそして何も項目が選択されないと(又は Cancel が使われると), "" が返される. もし multiple が真で何も項目が選択されない (又は Cancel が使われる)と, 長さ 0 の文字列ベクトルが返される.

See Also

menu, Tcl/Tk を使ったグラフィカルなバージョンは tk_select.list.

Examples

```
## Not run:
select.list(sort(.packages(all.available = TRUE)))

## End(Not run)
```

---

sessionInfo 現在の *R* セッションに関する情報を収集する

Description

R, OS そして付加又はロードされたパッケージに関するバージョン情報をプリントする.

Usage

```
sessionInfo(package = NULL)
## S3 method for class 'sessionInfo'
print(x, locale = TRUE, ...)
## S3 method for class 'sessionInfo'
toLatex(object, locale = TRUE, ...)
```

Arguments

package	インストールされたパッケージの名前の文字列ベクトル，又は NULL (既定値)は全ての付加されたパッケージを意味する．
x	クラス "sessionInfo" のオブジェクト．
object	クラス "sessionInfo" のオブジェクト．
locale	ロケール情報を示す?
...	現在使われない．

Value

クラス "sessionInfo" のオブジェクトで， print メソッドを持つ．これは次の成分を持つリスト

R.version	リスト，呼び出し R.Version() の結果．
platform	プラットフォームを記述する文字列．サブアーキテクチャが使用中なら，これは 'platform/sub-arch (nn-bit)' という形式である．
running	その下で実行中の OS とバージョンを記述する文字列 (そのもとでコンパイルされたものとは異なる)．
locale	文字列，呼び出し Sys.getlocale() の結果．
basePkgs	付加された基本パッケージの文字列．
otherPkgs	(いつもあるとは限らない)：付加された他のパッケージの文字列．
loadedOnly	(いつもあるとは限らない)：名前空間がロードされているが付加されていないパッケージに対する packageDescription の呼び出しの結果の名前付きリスト．

Note

'ロードされた'パッケージと名前空間に関する情報はパッケージがそこからロードされた位置でのインストールされた 現在のバージョンである：これは別のプロセスがセッション中にパッケージを変更していれば間違っている可能性がある．

OS がどのように自分自身とバージョンをを特定するかは不可解であり得る：可能な場合には running は可読な形式を使う．

See Also

R.version

Examples

```
sessionInfo()
toLatex(sessionInfo(), locale = FALSE)
```

setRepositories　　パッケージのレポジトリを選択する

Description

使用されるレポジトリをユーザと対話的に選択する．

Usage

```
setRepositories(graphics = getOption("menu.graphics"),
                ind = NULL, addURLs = character())
```

Arguments

graphics	論理値．もし真ならばグラフィカルなリストを使う： Windows や OS X GUI ではリストボックスを使い， Unix 風システムではもし **tcltk** と X サーバーが利用できれば Tk ウィジェットを使う．さもなければテキスト menu を使う．
ind	NULL 又は整数添字のベクトルで， graphics = FALSE に対するプロンプトにそれらが入力されたのと同じ効果を持つ．
addURLs	追加の URL の文字列ベクトル：名前付きベクトルを使うのがしばしば役に立つ．

Details

既知のレポジトリの既定リストはファイル '*R_HOME*/etc/repositories' 中に保管される．このファイルはサイトに対して編集でき，又はユーザは環境変数 R_REPOSITORIES により指示されるファイル中の個人的コピーを持つことが出来，又はもしこれが未設定か存在しなければ '*HOME*/.R/repositories' 中で，これが優先する．

Bioconductor ミラーは設定 options("BioC_mirror") で選択できる，例えば chooseBioCmirror を使う — 既定値は '"https://bioconductor.org"'．

予め選択される項目は現在 options("repos") 中にあるものに加え，既存レポジトリのリスト中で既定としてマークされたものである．

ある種のレポジトリはタイプのサブセットだけを提供するので，提供されるレポジトリのリストはオプション "pkgType" の設定に依存する (例えばソースパッケージだけか OS X バイナリパッケージでないものだけ)．更に，バイナリパッケージに対してはある種のレポジトリ (特に R-Forge)は現在か最近の R に対するパッケージだけを提供する．(タイプ "both" は "source" と同値である．)

レポジトリ 'CRAN' は特別に扱われる：値はもし getOption("repos") の現在の設定が要素 "CRAN" を持てばそれから取られる：これはミラーの選択に固執することを保障する．

この関数は ind が提供されていない限り R セッションが対話的であることを必要とする．

Value

この関数は主に options("repos") を更新するその副作用のために起動される．これは(不可視で)前の repos オプション設定 (成分 repos を持つ list として)，又はもし変更が何もされなければ NULL を返す．

Note

これはスタート時にレポジトリのリストを設定しない：そうするためにはスタートアップファイル中に options(repos =) を設定する (ヘルプトピック Startup を見よ).

See Also

chooseCRANmirror, chooseBioCmirror, install.packages.

Examples

```
## Not run:
setRepositories(addURLs =
                c(CRANxtras = "http://www.stats.ox.ac.uk/pub/RWin"))

## End(Not run)
```

SHLIB 動的ローディングに対する共有オブジェクト/DLL を構築する

Description

与えられたソースファイルをコンパイルし，そしてそれから全ての指定されたオブジェクトファイルを dyn.load または library.dynam を使って R 中にロードできる共有オブジェクト，別名 DLL，をリンクする.

Usage

```
R CMD SHLIB [options] [-o dllname] files
```

Arguments

files	共有オブジェクト/DLL中に含められるオブジェクトファイルを指定するリスト．ソースファイルの名前と (それに対してオブジェクトファイルがそれらのソースから自動的に作られる)ライブラリのリンク命令を含めることが出来る.
dllname	構築される共有オブジェクト/DLL で，拡張子を含む (典型的にはUnixでは '.so' で，そして Windows では '.dll'). もし与えられないと，オブジェクト/DLL の最初のファイルの基礎名から取られる.
options	処理を制御する追加オプション．現在のリストに対しては R CMD SHLIB --help を使う.

Details

R CMD SHLIB はパッケージ中のソースコードをコンパイルするため INSTALL によって使われる機構である．これは C, C++, Objective C(++) そして Fortran ソースに対する適切なコンパイル命令を生成する： Fortran 90/95 ソースも使うことができるが，これらを他の言語と混用することは出来ないかもしれない (殆どのプラットフォームでは C と混用できるが， C++ との混用は滅多に出来ない).

それをどのようにカスタマイズするか (例えば cpp フラグを加えたりリンクステップにライブラリを加える)そしてその幾つかの奇妙な癖の詳細についてはマニュアル 'Writing R Extensions' 中の節 'Creating shared objects' を参照せよ

files 中の拡張子 '.c', '.cpp', '.cc', '.C', '.f', '.f90', '.f95', '.m' (ObjC), '.M' そして '.mm' (ObjC++) を持つ項目はソースファイルと見做される，そして拡張子 '.o' を持つものはオブジェクトファイルである．他の全ての項目はリンカーに渡される．

オブジェクティブ C(++) のサポートは R がコンフィギュアされた時のオプションである：これらの主な使い道は OS X 上である．

適切な実行時ライブラリがもし C++, Fortran またはオブジェクティブ C(++) ソースが与えられるとリンク時に使われるが，これらの言語からのコンパイル済みのオブジェクトに対しては使われない．

オプション '-n' (また '--dry-run' としても知られる) はそれらを実行すること無く実行される命令を表示する．

Note

ある種の R のディストリビューションは SHLIB を別個のバンドル，例えば R-devel RPM，に持つ．

See Also

COMPILE, dyn.load, library.dynam.

マニュアル 'R Installation and Administration' と 'Writing R Extensions'，前者の節 "Customizing compilation" を含む．

Examples

```
## Not run:
# システムのライブラリパス上に無いライブラリに対してリンクする：
R CMD SHLIB -o mylib.so a.f b.f -L/opt/acml3.5.0/gnu64/lib -lacml

## End(Not run)
```

---

sourceutils　　ソース参照ユーティリティ

Description

これらの関数はソース参照からの情報を抜き出す．

Usage

```
getSrcFilename(x, full.names = FALSE, unique = TRUE)
getSrcDirectory(x, unique = TRUE)
getSrcref(x)
getSrcLocation(x, which = c("line", "column", "byte", "parse"),
               first = TRUE)
```

Arguments

x	ソース参照を含むオブジェクト(典型的には関数).
full.names	ファイル名の結果で完全なパスを含むかどうか.
unique	ユニークなファイル名/ディレクトリだけをリストするか.
which	ソース参照のどの部分を取り出すか．短縮化出来る.
first	オブジェクトの最初(又は最後)の位置を示すかどうか.

Details

もし "keep.source" オプションが TRUE ならば関数の各文はそれ自体のソース参照を持つ．これらの関数はそれらの全てを回収する．

成分は以下の通りである：

line オブジェクトが始まるか終わる行番号．

column オブジェクトが始まるか終わる列番号．

byte "column" のようだが，バイトを数え，これは多バイト文字の場合は異なるかもしれない．

parse "line" のようだが，これは #line ダイレクティブを無視する．

Value

getSrcFilename と getSrcDirectory はファイル名/ディレクトリを保持する文字列ベクトルを返す．

getSrcref は "srcref" オブジェクトのリストを返す，又はもしそれがなければ NULL を返す．

getSrcLocation は要求されたタイプの位置の整数ベクトルを返す．

See Also

srcref, getParseData

Examples

```
fn <- function(x) {
  x + 1 # コメント，ソースの一部として保存される
}

# 実例が保存されている一次ファイルディレクトリを示す

getSrcDirectory(fn)
getSrcLocation(fn, "line")
```

---

stack　データフレームかリストからのベクトルをスタック化するか非スタック化する

Description

ベクトルのスタック化は複数ベクトルを単一のベクトルに連結し，各観測値がどこからきたのかを指示する因子を持つ．非スタック化はこの操作の逆を行う．

Usage

```
stack(x, ...)
## Default S3 method:
stack(x, ...)
## S3 method for class 'data.frame'
stack(x, select, ...)

unstack(x, ...)
## Default S3 method:
unstack(x, form, ...)
## S3 method for class 'data.frame'
unstack(x, form, ...)
```

Arguments

x	スタック化又は非スタック化されるリストかデータフレーム.
select	表現式，どの変数をデータフレームから選択するかを指示する.
form	両側公式で，左辺は非スタック化されるベクトルを評価し，そして右側は作られるグループの指示子に評価される．既定値では unstack に対するデータフレームメソッド中の formula(x).
...	他のメソッドに/から渡される追加引数.

Details

stack 関数はデータフレーム又はリスト中の個別の列として得られるデータを，分散分析モデルや他の線形モデルで使える単一の列に変換する．unstack 関数はこの操作を逆転する.

stack はベクトル (is.vector で決定されるような)に適用されることを注意する：非ベクトルの列(例えば因子)は無視される (R 2.15.0 以来警告が出る)．異なったタイプのベクトルが選択された時は，それらは unlist で連結され，そのヘルプページは結果のタイプがどのように選択されるかを説明する.

これらの関数は総称的である：提供されるメソッドはデータフレームと as.list でリストに強制変換可能なオブジェクトを処理する.

Value

unstack は公式 form に従い列のリストを作る．もし全ての列が同じ長さを持てば，結果はデータフレームに強制変換される.

stack は次の二つの列を持つデータフレームを作る：

values	x 中の選択されたベクトルを連結した結果.
ind	x 中のベクトルがどこに由来するかを指示する因子.

Author(s)

Douglas Bates

See Also

lm, reshape

Examples

```
require(stats)
formula(PlantGrowth)          # 既定の公式をチェック
pg <- unstack(PlantGrowth)    # この公式に従い非スタック化
pg
stack(pg)                     # そしてそれを一緒に戻す
stack(pg, select = -ctrl)     # 一つのベクトルを除外する
```

---

str　　任意の *R* オブジェクトの構造を簡潔に表示する

Description

R オブジェクトの内部構造を簡潔に表示する診断関数(そしてある程度 dput の代替物)である．理想的には各'基本的'構造に対しては一行．これは(可能性として入れ子の) リストの(省略された)内容を簡潔に表示するのに特に適している．アイデアは任意の R オブジェクトに対して合理的な出力を与えることにある．これは(プリミティブで無い)関数オブジェクトに対して args を呼び出す．

strOptions() は options(str = .) を設定するための便利関数である，例を見よ．

Usage

```
str(object, ...)

## S3 method for class 'data.frame'
str(object, ...)

## Default S3 method:
str(object, max.level = NA,
    vec.len  = strO$vec.len, digits.d = strO$digits.d,
    nchar.max = 128, give.attr = TRUE,
    give.head = TRUE, give.length = give.head,
    width = getOption("width"), nest.lev = 0,
    indent.str = paste(rep.int(" ", max(0, nest.lev + 1)),
                       collapse = ".."),
    comp.str = "$ ", no.list = FALSE, envir = baseenv(),
    strict.width = strO$strict.width,
    formatNum = strO$formatNum, list.len = 99, ...)

strOptions(strict.width = "no", digits.d = 3, vec.len = 4,
           formatNum = function(x, ...)
                       format(x, trim = TRUE, drop0trailing = TRUE, ...))
```

Arguments

object	ある情報を得たい任意の R オブジェクト．
max.level	入れ子構造の表示に対して適用される入れ子の最大レベル，例えば副リストを含むリスト．既定では NA：全ての入れ子水準を表示する．

vec.len	各ベクトルに対して幾つの‘最初の僅かな’要素を表示するかを指示する数値(>= 0). この数はベクトルの種類により異なった因子 (.5 から 3)倍される. 既定値はオプション "str" の vec.len 成分で既定値は 4 (options を見よ).
digits.d	数値成分に対する桁数 (print に対するようなもの). 既定値はオプション "str" の digits.d 成分で既定値は 3.
nchar.max	character 文字列に対して示される文字数の最大値. 長過ぎる文字列は打ち切られる, 下の例 longch を見よ.
give.attr	論理値；もし TRUE なら(既定値) 副構造として属性を示す.
give.length	論理値；もし TRUE (既定値)ならば長さを示す([1:...] のように).
give.head	論理値；もし TRUE (既定値)ならば (可能性として簡略化された)モード/クラスとそして長さ (<type>[1:...] のような)を示す.
width	使われるページ幅. 既定値は現在有効な options("width")；これは strict.width が "no" で無い限り弱い効果しか持たないことを注意する.
nest.lev	str の再帰的な呼び出し中の現在の入れ子レベル.
indent.str	使用されるインデント文字列.
comp.str	リスト成分の分離に対して使われる文字列.
no.list	論理値；もし真ならば ‘...のリスト’もクラスもプリントされない.
envir	予約オブジェクトに対してだけ使われる環境 (delayedAssign を見よ).
strict.width	width 引数の指定を厳格に守るべきかどうかを指示する文字列で, c("no", "cut", "wrap") のどれかで短縮化可能. 既定値はオプション "str" の strict.width 成分 (options を見よ)で, その既定値は後方互換性のため "no"；"wrap" は strwrap(*, width = width) を使い, "cut" は直接的に width で切り取る. 小さな vec.length は設定 strict.width = "wrap" より好ましいかもしれないことを注意する.
formatNum	数値ベクトルを書式化するための format のような関数. その既定値は "str" の formatNum 成分, 上の strOptions() の“用法”を見よ. これはほとんど R <= 2.7.x と後方互換であるが, しかしながら formatC の使用のほうが少しよいかもしれない.
list.len	数値；レベル中で表示されるリスト要素の最大数.
...	可能性のある追加引数 (メソッド/総称関数のために必要とされる).

Value

効率性のため str は何も返さない. 明らかな副作用は端末への出力である.

Author(s)

1990 以来 Martin Maechler <maechler@stat.math.ethz.ch>

See Also

構造付きのオブジェクトのリストには ls.str； summary, args.

Examples

```
require(stats); require(grDevices); require(graphics)
## 次の例は 'str' の機能の幾つかを示す
str(1:12)
str(ls)
str(args) #- args(args) よりも有用!
str(freeny)
str(str)
str(.Machine, digits.d = 20) # バイナリ数の特定のための余分の桁数
str( lsfit(1:9, 1:9))
str( lsfit(1:9, 1:9), max.level = 1)
str( lsfit(1:9, 1:9), width = 60, strict.width = "cut")
str( lsfit(1:9, 1:9), width = 60, strict.width = "wrap")
op <- options(); str(op)   # 最初を保存；
                           # さもなければ内部的な options() が使われる.
need.dev <-
  !exists(".Device") || is.null(.Device) || .Device == "null device"
{ if(need.dev) postscript()
  str(par())
  if(need.dev) graphics.off()
}
ch <- letters[1:12]; is.na(ch) <- 3:5
str(ch) # 文字列の NA

str(list(a = "A", L = as.list(1:100)), list.len = 9)
nchar(longch <- paste(rep(letters,100), collapse = ""))
str(longch)
str(longch, nchar.max = 52)

str(longch, strict.width = "wrap")

## 狭い表示のための設定：
op <- options(width = 60,
              str = strOptions(strict.width = "wrap"))
str(lsfit(1:9,1:9))
str(options())
## 元に戻す：
options(op)



str(quote( { A+B; list(C, D) } ))



## S4 クラス：
require(stats4)
x <- 0:10; y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)
ll <- function(ymax = 15, xh = 6)
      -sum(dpois(y, lambda=ymax/(1+x/xh), log=TRUE))
fit <- mle(ll)
str(fit)
```

summaryRprof　　*R* のサンプリングプロファイラーの出力を要約する

Description

異なった R 関数により使われる時間量を示す Rprof 関数の出力を要約する.

Usage

```
summaryRprof(filename = "Rprof.out", chunksize = 5000,
             memory = c("none", "both", "tseries", "stats"),
             lines = c("hide", "show", "both"),
             index = 2, diff = TRUE, exclude = NULL,
             basenames = 1)
```

Arguments

filename	Rprof() で作られるファイルの名前.
chunksize	一度に読み込まれる行数.
memory	メモリ情報の要約. 下の'メモリープロファイリング'節を見よ. 短縮化出来る.
lines	行情報の要約. 下の'行プロファイリング'節を見よ. 短縮化出来る.
index	メモリ情報に対するスタックトレースをどのように要約するか. 下の'詳細'節を見よ.
diff	もし TRUE ならばメモリ要約は現在のメモリではなくメモリ中の変化を要約する.
exclude	メモリ要約に対するスタックトレースの要約時に除外する関数.
basenames	表示するファイル名へのパスの成分名.

Details

この関数は R CMD Rprof が使う. Rprof ファイルに対する解析コードを提供する.

プロファイリングの出力ファイルは利用可能なメモリよりも大きくなる可能性があるので, これは chunksize 行のブロック単位で読み込まれる. もし十分なメモリが利用できれば chunksize を増やすと関数はより速く実行される.

Value

もし memory = "none" で lines = "hide" ならば, 次の成分を持つリスト

by.self	'self' 時間でソートされたタイミングのデータフレーム.
by.total	'総'時間でソートされたタイミングのデータフレーム.
sample.interval	サンプリング間隔.
sampling.time	プロファイル実行の総時間.

最初の二つの成分は列 'self.time', 'self.pct', 'total.time' そして 'total.pct' を持ち, それぞれその関数中又はその関数から呼び出された実行コードで費やされる秒単位の時間と総時間とのパーセントを示す.

もし lines = "show" ならばリストに追加の成分が付け加えられる :

by.line　　ソース位置でソートされるタイミングのデータフレーム.

もし memory = "both" ならば同じリストであるが，タイミングと共に Mb 単位のメモリ消費を持つ.

もし memory = "tseries" ならば時間に関するメモリ統計を与えるデータフレーム.

もし memory = "stats" ならば関数によるメモリ統計を与える by オブジェクト.

R 2.15.3 以前はもし如何なるイベントも記録されないとエラーが起きた：今は列数がゼロのデータフレームが返される.

メモリーのプロファイリング

memory = "none" 以外のオプションは Rprof(memory.profiling = TRUE) で作られるファイルにだけ適用される.

memory.profiling = TRUE と共に呼び出されると，プロファイラーはメモリ使用の三つの様相に関する情報を書き出す： R ヒープに関する小さなブロックのベクトルメモリ，大きなブロックのベクトルメモリ (malloc からの)，R ヒープに関するノード単位のメモリ．また時間単位での内部関数 duplicate の呼び出し数も記録する． duplicate は引数がコピーされる必要がある時 C コードにより呼び出される．プロファイラはどの関数が実際にメモリを確保したかについては追跡しないことを注意する.

memory = "both" では総メモリ (ゼロで打ち切られる)の変化がタイミングのデータと共に報告される.

memory = "tseries" 又は memory = "stats" では index 引数はスタックのトレースをどのように要約するかを指定する．正の数はスタックの底からの多くの呼び出しを指定する；負の値はスタックのトップからの呼び出しを指定する． memory = "tseries" では index はラベルの構成に使われ，そしてラベルの複数のセットを与えるベクトルかもしれない． memory = "stats" では index は単一の数でなければならず，そしてデータをメモリ統計量の最大値と平均値にどのように集積するかを指定する． memory = "tseries" と memory = "stats" 双方では引数 diff = TRUE はサンプリング区間でのメモリ使用量の増加の要約を要求し，そして diff = FALSE は区間の最後でのメモリ使用を要求する.

行のプロファイリング

もし実行されるコードがソース参照情報を保持していれば (source 中の keep.source = TRUE かパッケージの 'DESCRIPTION' ファイル中の KeepSource = TRUE かある種の他の方法で)，行の源はプロファイルの過程で記録される．既定ではこれは表示されないが， lines パラメータは表示を可能にすることが出来る.

もし lines = "show" ならば行位置は通常の関数名情報に優先して使われ，そして結果は他の順序に加えて位置により順序付けられて表示される.

もし lines = "both" ならば行位置は結合された表示中で関数名と混ぜ合わされる.

See Also

"Writing R Extensions" 中の章 "Tidying and profiling R code" (R のソースツリーの 'doc/manual' サブディレクトリを見よ).

Rprof

tracemem は C 関数 duplicate を使いオブジェクトのコピーをトレースする.

Rprofmem はサンプリングを行わないメモリ使用のプロファイラである.

https://developer.r-project.org/memory-profiling.html

Examples

```
## Not run:
## Rprof() は全てのプラットフォームで利用可能ではない
Rprof(tmp <- tempfile())
example(glm)
Rprof()
summaryRprof(tmp)
unlink(tmp)

## End(Not run)
```

Sweave 報告の自動生成

Description

Sweave は自動的なレポート作成のためにテキストと R/S コードを混在させる柔軟な枠組みを提供する．基本的なアイデアはコードをその出力で置き換えることで，最終的なドキュメントはテキストと統計解析の出力のみを含むようになる：しかしながらソースコードも含めることが出来る．

Usage

```
Sweave(file, driver = RweaveLatex(),
       syntax = getOption("SweaveSyntax"), encoding = "", ...)

Stangle(file, driver = Rtangle(),
        syntax = getOption("SweaveSyntax"), encoding = "", ...)
```

Arguments

file	Sweave ソースファイルへのパス．これは拡張子なしに提供できるが，関数は基本名が file にマッチするディレクトリ中にもし丁度一つの Sweave ファイルがある時だけ処理を行う．
driver	実働関数，‘詳細’節を見よ．
syntax	NULL またはクラス SweaveSyntax のオブジェクトを，またはその名前の文字列．節‘シンタックス定義’を見よ．
encoding	file に対して仮定される既定エンコーディング．
...	ドライバーのセットアップ関数に渡される追加引数：節‘ドライバー’，RweaveLatex そして Rtangle を見よ．

Details

文章処理マークアップ (latex のような)と S コードを混在させレポートを自動生成する．S コードは最終的なマークアップファイル中でその出力 (テキストまたはグラフ)で置き換えられる．これはもし入力データが変わればレポートを再生成し，そしてすでにレポートを作成した同じファイル中の解析を再作成するコードをドキュメント化することを可能にする．

Sweave はドキュメントとコード塊 (またはそれらの出力)を一つのドキュメントに連結する． Stangle は Sweave ファイルからコードだけを取り出し source を使って実行される S のソースファイルを作る．(\Sexpr{} 文内部のコードは Stangle により無視される．)

Stangle は異なった既定値のドライバーを指定する Sweave への単なるラッパである．別のドライバーを使うことが出来る： CRAN のパッケージ **cacheSweave** と Bioconductor のパッケージ **weaver** は共にコードの塊に対する計算結果をキャッシュするアイデアを取り入れた既定ドライバー RweaveLatex に基づくドライバーを提供する．

環境変数 SWEAVE_OPTIONS はドライバーの初期設定を上書きするのに使うことが出来る：これはドキュメント中の '\SweaveOpts' 文中で使われるような key=value 項目のコンマで分離されたセットであるべきである．

非ASCIIのソースファイルは形式

```
\usepackage[foo]{inputenc}
```

の行を含まねばならず (ここで 'foo' は典型的に 'latin1', 'latin2', 'utf8' または 'cp1252' または 'cp1250')，さもなければエラーになる．再エンコーディングは引数 encoding = "bytes" で完全に無効に出来る．

シンタックス定義

Sweave はドキュメントとテキストの塊を作る柔軟なシンタックスの枠組みを可能にする．既定は noweb スタイルのシンタックスで， latex 風のシンタックスの代替物として使うことが出来る． (より詳細はユーザマニュアルを見よ．)

もし syntax = NULL (既定)ならば，利用可能なシンタックスオブジェクトが順番に検討され，そしてもしそれらの extension 成分が(正規表現として) ファイル名にマッチすれば選択される． オブジェクト SweaveSyntaxNoweb (codeextension = "[.][rsRS]nw$" の時)そして SweaveSyntaxLatex (extension = "[.][rsRS]tex$" の時)が提供されるが，ユーザまたはパッケージはパターン SweaveSyntax.* にマッチする名前を持つ他の物を提供できる．

Author(s)

Friedrich Leisch と R-core.

References

Friedrich Leisch (2002) Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 - Proceedings in Computational Statistics*, pages 575–580. Physika Verlag, Heidelberg, Germany, ISBN 3-7908-1517-9.

See Also

'Sweave User Manual'， **utils** パッケージ中のビニエット．

RweaveLatex, Rtangle.

パッケージ **cacheSweave**, **weaver** (Bioconductor) そして **SweaveListingUtils**.

Sweave の他のドライバーは例えばパッケージ **R2HTML**, **ascii**, **odfWeave** そして **pgfSweave** 中にある．

Sweave を利用しないビニエットは tools::buildVignette を使って作成できる．

Examples

```
testfile <- system.file("Sweave", "Sweave-test-1.Rnw", package = "utils")

## par(ask = FALSE) を強制する
options(device.ask.default = FALSE)
```

```
## LaTeX ファイルを作る
Sweave(testfile)

## これは
## tools::texi2pdf("Sweave-test-1.tex")
## により，または R の外部で次の命令で PDF にコンパイルされる
## R CMD texi2pdf Sweave-test-1.tex
## これは適当な TEXINPUTS パスを設定する.

## コードの塊から R ソースファイルを作る
Stangle(testfile)
## これは例えば次でソース出来る
source("Sweave-test-1.R")
```

SweaveSyntConv *Sweave* シンタックスを変換

Description

この関数は Sweave 書式のファイルのシンタックスを別の Sweave シンタックス定義に変換する.

Usage

```
SweaveSyntConv(file, syntax, output = NULL)
```

Arguments

file	Sweave ソースファイルの名前.
syntax	クラス SweaveSyntax のオブジェクトまたはファイルがそれに変換されるターゲットのシンタックスを与えるその名前の文字列.
output	出力ファイル名，既定値は入力から拡張子を取り除きターゲットの拡張子を加える． file 中の任意のディレクトリ名は取り除かれ，出力は現在の作業ディレクトリ中に作成される.

Author(s)

Friedrich Leisch

See Also

'Sweave User Manual'，パッケージ **utils** 中のビニエット.

RweaveLatex, Rtangle

Examples

```
testfile <- system.file("Sweave", "Sweave-test-1.Rnw", package = "utils")

## ファイルを latex シンタックスに変換
SweaveSyntConv(testfile, SweaveSyntaxLatex)

## そしてそれを Sweave を使って実行
Sweave("Sweave-test-1.Stex")
```

---

tar *Tar* アーカイブを作る

Description

Tar アーカイブを作る.

Usage

```
tar(tarfile, files = NULL,
    compression = c("none", "gzip", "bzip2", "xz"),
    compression_level = 6, tar = Sys.getenv("tar"),
    extra_flags = "")
```

Arguments

tarfile	tar ファイルのパス名：チルド展開(path.expand を見よ)が実行される. 別法としてバイナリ書き込みのために connection を使うことが出来る.
files	アーカイブ先のファイルパスの文字列ベクトル：既定では現在のディレクトリ下に全てのファイルにアーカイブする.
compression	使用される(既定では無し)圧縮タイプを与える文字列. 短縮化出来る.
compression_level	整数：圧縮のレベル：内部メソッドに対してのみ使われる.
tar	文字列：使用される命令へのパス. もし命令自体が空白を含めば引用化される必要がある (例えば shQuote により) – しかし引数 tar は又命令から空白で分離されたフラグを含むことが出来る.
extra_flags	外部の tar に対する追加のフラグ.

Details

これは tar 名に対するラッパか又は R 中の内部実装を使う. 後者はもし tarfile がコネクションか又は引数 tar が "internal" か "" (‘工場出荷時’既定値)の時使われる. R の Unix 風のバージョンは環境変数 TAR を設定し, その値はこの関数に対する既定値ではないことを注意する.

引数 extra_flags は外部 tar に渡されるのでプラットフォーム依存である. 可能性として有用な値は ‘-h’ (シンボリックリンクに従い, ある種のプラットフォームでは又 ‘-L’), ‘--acls’, ‘--exclude-backups’, ‘--exclude-vcs’ (そして類似の物)そして Windows では ‘--force-local’を含む(従ってドライバーはファイルパス中に含めることが出来る：しかしながらこれは Rtools tar に対しては既定値である). GNU tar に対しては,

'--format=ustar' はより可搬性のある書式を強制する． (既定はコンパイル時に設定され，そして tar --help からの出力の最後に示される：バージョン 1.28 '以来' これは '--format=gnu' であるが，マニュアルは GNU が不正確に 'POSIX' を呼び出す '--format=pax' に変更することが意図であると主張する – これは決して tar に対する POSIX 標準の一部ではなく使われるべきではない．) libarchive の bsdtar に対しては，'--format=ustar' が既定よりもより可搬性がある．

Value

system からの復帰コード，又は内部バージョンに対しては 0 で，不可視．

可搬性

'tar' 書式は最早合意された標準を持たない! 'Unix の標準 Tar 'は POSIX 1003.1 の一部であった：しかし 1998 年に pax を優先してとりのぞかれ，そしてあらゆるケースで多くの普通の実装は以前の標準から逸脱してきている．多くの R のプラットフォームは GNU tar のバージョンを使う (Windows の Rtools を含む)が，その挙動は各バージョンで変更されているように見える． OS X >= 10.6 と FreeBSD は 'libarchive' プロジェクトからの bsdttar を使い，商用 Unix はそれら自体のバージョンを使う．

既知の問題は以下の点から生じる

- 100 バイトを超えるファイルパスの処理．これらは tar の初期のバージョンではサポートされておらず， POSIX の tar によりある方法でサポートされ，別に GNU tar と最近の tar プログラムがしばしばサポートしている POSIX pax 命令によるもう一つがある．内部的実装は 100 バイト以上のパスに関して警告し，もし可能なら最大 256 バイトまでサポートする 1998年の POSIX 標準からの 'ustar' の流儀を使う (パスに依存する：特に最後の成分は 100 バイトに制限されている)，さもなければ GNU の流儀を使う (これは広く使われており untar に含まれている)．

  殆どの書式はファイルパスのエンコーディングを記録しない．

- (ファイルの)リンク． tar はハードリンクを使った OS 上で開発され，含められるファイルのリスト中で複数回参照されていた物理的ファイルは一度だけ含められ，残りはリンクとして付け加えられる．後にシンボリックリンクを含める手段が付け加えられた．内部的な実装はシンボリックリンクだけをサポートする (それらをサポートする OS 上で)．勿論，それらをサポートしていない OS 上でリンクをどのように開封するかについての問題が生じる：正規のファイルに対してはファイルのコピーを使うことが出来る．

  'ustar' 書式中のリンクの名前は 100 バイトに制限されている．任意に長いリンク名に対する GNU 拡張があるが， bsdtar はそれを無視する．内部メソッドは GNU を使うが，警告が出る．

- ヘッダファイル，特に欄が一杯でないか使われない時に使われる詰物．POSIX は正しい挙動を定義していないが，普通使われる実装については合意は無かった(そして依然無い)．

- ファイルサイズ． 'ustar'書式は(非圧縮)ファイル毎に 8GB に制限される．

可搬性のため， 100 バイトを超えるファイルパスと全てのリンクを避ける (特にディレクトリに対するハードリンクとシンボリックリンク)．

内部実装は書き込みに 512 バイトのブロックだけを必要とする (末尾のヌルのブロックを含む)が， GNU tar は既定では 20 ブロック(10KB)の倍数になるように 'nul' で埋める．詰物を行う実装はブロック詰物が圧縮の前か後(又は両方)に行われるかによって異なる：詰物は物理的なテープドライバーに関する効率をあげるためにデザインされた．

See Also

https://en.wikipedia.org/wiki/Tar_(file_format)， POSIX ユーティリティ pax が tar 書式を処理する仕方については http://pubs.opengroup.org/onlinepubs/9699919799/utilities/pax.html#tag_20_92_13_06.

https://github.com/libarchive/libarchive/wiki/FormatTar.

untar.

---

toLatex　　*R* オブジェクトを *BibTeX* や *LaTeX* に変換する

Description

これらのメソッドは R オブジェクトを BibTeX か LaTeX のマークアップを持つ文字列ベクトルに変換する.

Usage

```
toBibtex(object, ...)
toLatex(object, ...)
## S3 method for class 'Bibtex'
print(x, prefix = "", ...)
## S3 method for class 'Latex'
print(x, prefix = "", ...)
```

Arguments

object	toBibtex 又は toLatex 用のメソッドが存在するクラスのオブジェクト.
x	クラス "Bibtex" 又は "Latex" のオブジェクト.
prefix	各行の最初にプリントされる文字列，殆どインデント用に空白を挿入するために使われる.
...	プリントメソッドで writeLines に渡される.

Details

クラス "Bibtex" か "Latex" のオブジェクトは単に各要素が対応する BibTeX か LaTeX ファイルに対応する一行を保持する文字列である.

See Also

例えば citEntry と sessionInfo.

txtProgressBar テキストの進行度バー

Description

R コンソール中のテキストによる進行度バー.

Usage

```
txtProgressBar(min = 0, max = 1, initial = 0, char = "=",
               width = NA, title, label, style = 1, file = "")

getTxtProgressBar(pb)
setTxtProgressBar(pb, value, title = NULL, label = NULL)
## S3 method for class 'txtProgressBar'
close(con, ...)
```

Arguments

min, max	進行度バーの極値に対する(有限の)数値. min < max でなければならない.
initial, value	進行度バーに対する初期値又は新しい値. 不正な値に対して何が起きるかは'詳細'節を見よ.
char	進行度バーを作る文字(又は文字列).
width	進行度バーの幅で, char の幅の倍数として. もし既定の NA ならば, 文字の数は getOption("width") にフィットするもの.
style	バーの'スタイル' – '詳細'節を見よ.
file	開かれたコネクションオブジェクトかコンソールを指示する ""; stderr() がここで有用かもしれない.
pb, con	クラス "txtProgressBar" のオブジェクト.
title, label	無視される, 他の進行度バーとの互換性のため.
...	総称的関数との一貫性のため.

Details

txtProgressBar はテキスト形式で R コンソール(又はコネクション)上に進行度バーを表示する.

setTxtProgessBar は値を更新する. 欠損値(NA) と範囲外の value の値は (黙って)無視される. (そうした initial の値は適正な値が設定されるまで進行度バーが表示されないようにする.)

進行度バーはそれが終了した時は close されるべきである : これは最終的な新行文字を出力する.

style = 1 と style = 2 は単に char の行を表示する. これらは style = 2 が毎回線を引き直す点で異なり, これは R コンソールに他のコードが書き込んでいるかもしれないなら有用である. style = 3 は範囲の終わりを | でマークし, そしてバーの右にパーセントを与える.

Value

txtProgressBar に対してはクラス "txtProgressBar" のオブジェクト.

getTxtProgressBar と setTxtProgressBar に対しては直近の値を与える長さ1の数値ベクトル (setTxtProgressBar に対しては不可視).

Note

style の 2 か 3 又は style = 1 で値を減らすと左マージンに復帰するのに '\r' を使う – キャリッジリターンの解釈は R が実行中の端末かコンソールに依存し，これは端末以外のコネクションでは醜い出力を生み出しやすい， stdout() がファイルにリダイレクトされた場合を含む.

See Also

tkProgressBar.

R の Windows バージョンは又 winProgressBar を持つ.

Examples

```
 # 遅い
testit <- function(x = sort(runif(20)), ...)
{
    pb <- txtProgressBar(...)
    for(i in c(0, x, 1)) {Sys.sleep(0.5); setTxtProgressBar(pb, i)}
    Sys.sleep(1)
    close(pb)
}
testit()
testit(runif(10))
testit(style = 3)
```

---

type.convert 文字列変数のタイプ変換

Description

文字列変数を論理値，整数，数値，複素数値，又は適当ならば因子に変換する.

Usage

```
type.convert(x, na.strings = "NA", as.is = FALSE, dec = ".",
             numerals = c("allow.loss", "warn.loss", "no.loss"))
```

Arguments

x	文字列変数.
na.strings	NA 値と解釈されるべき文字列のベクトル．空白の欄は又論理値，整数，数値又は複素数値の欠損値と見做される.
as.is	論理値．'詳細'節を見よ.
dec	少数点に対して仮定される文字.

numerals　倍精度実数への変換で精度が失われる数をどのように変換するかを指示する文字列，典型的には double 中に保存できるよりも多くの数字を持つ場合．短縮化出来る．可能な値は次の通り

numerals = "allow.loss", **default:** 変換は幾らかの精度の喪失で行われる．これは R のバージョン 3.0.3 とそれ以前だけの挙動であった．

numerals = "warn.loss": 精度の損失がシグナルされそして変換が numerals = "allow.loss" で起きると warning が出る．

numerals = "no.loss": x は数に変換され無いが，as.is に応じて factor に変換されるか文字列のままにされる．これは R のバージョン 3.0.3 とそれ以前だけの挙動であった．

Details

これは基本的に read.table に対する手助け関数である．文字列ベクトルをあたえた時，これはそれを論理値，整数，数値，複素数値に変換し，それに失敗すれば as.is = TRUE で無い限りそれを因子に変換する．欠損値でない値を全て受け入れる最初のタイプが選択される．

NA は本来論理値であるため，欠損値だけのベクトルは論理値に変換される．

F, T, FALSE, TRUE そして na.strings からの値だけを含むベクトルは論理値に変換される．R の整数として表現可能な小数点や na.strings からの値が後に続くオプションの空白は整数に変換される．オプションの空白に他の10進数や16進数 (NumericConstants を見よ)，又は NaN, Inf か infinity (大・小文字の別を無視する) 又は na.strings からの値が続くものを含む他のベクトルは数値に変換される．数値や複素数値への入力の変換が精度を失うならば，それらはオプションで文字列(as.is = TRUE に対して)又は因子として返すことが出来る．

これは手助け関数であるため，呼び出し側は常に適当な as.is の値を渡すべきである．

Value

原子的ベクトル，又は因子(as.is = FALSE に対して)．

See Also

read.table

---

untar　　*tar* アーカイブを取り出す又はリストする

Description

tar アーカイブから内容を取り出す又はリストする．

Usage

```
untar(tarfile, files = NULL, list = FALSE, exdir = ".",
      compressed = NA, extras = NULL, verbose = FALSE,
      restore_times =  TRUE, tar = Sys.getenv("TAR"))
```

Arguments

tarfile	tar ファイルのパス名：チルド展開 (path.expand を見よ)が実行される．もしくはバイナリ読み込みに対して使える connection.
files	取り出しが行われる記録されたファイルパスの文字列ベクトル：既定では全てのファイルを取り出す．
list	もし TRUE ならファイルをリストする (tar -tf と同値)．さもなければファイルを取り出す (tar -xf と同値).
exdir	ファイルをそこに取り出すディレクトリ (tar -C と同値)．必要なら作られる．
compressed	論理値又は文字列．値 "gzip", "bzip2" そして "xz" は圧縮のタイプを選択する(そして最初の文字に短縮化可能)． TRUE は gzip 圧縮を指示し，FALSE は既知の圧縮を使わない (しかし外部 tar 命令が自動的に圧縮を検出するかもしれない)，そして NA (既定値)はタイプはファイルヘッダーから推測されることを指示する．
extras	NULL 又は文字列：外部 tar プログラムに渡される '-p' のような追加コマンドライン引数.
verbose	論理値：もし真なら使われた命令をエコーする．
restore_times	論理値．もし真(既定値)ならばファイルの修正時間を復活する．もし偽なら '-m' フラグと同値．ターボール中の時間は UTC であると仮定されるが，ターボールは CRAN に未来かはるか過去の時間を用いて投稿されてきている：この引数はそうした時間を捨て去ることを許す．
tar	文字列：使用される命令へのパス．もし命令自体が空白を含めば引用化される必要がある – しかし tar はまた空白で命令から分離されたフラグを含むことが出来る．

Details

これは tar 命令か又は R で書かれた内部的実装に対するラッパである．後者はもし tarfile がコネクションかもしくは引数 tar が "internal" か "" の時使われる(Windows を除く， tar.exe が最初に試みられた時).

どのオプションがサポートされているかは使用する tar に依存する．最新の GNU 風の tar は圧縮アーカイブをサポートし， 1.15 以来圧縮のタイプを自動的に検出する：バージョン 1.20 は lzma をサポートし，そして xz 圧縮のバージョン 1.22 は LZMA2 を使う．OS X 10.6 とそれ以降(そして FreeBSD とある種の他の OS)は 'libarchive' プロジェクトからの tar を持っており(bsdtar としても知られる)，これは又 gzip と bzip2 圧縮を自動的に検出する．他の風味の tar に対しては，環境変数 R_GZIPCMD は gzip と compress ファイルを解凍する命令を与え，そして R_BZIPCMD は bzip2 ファイルを解凍する命令を与える．

引数 compressed, extras そして verbose は外部 tar が使われる時だけ使われる．

内部実装は Unix 風システムに於けるようにシンボリックリンクをリンクとして回復し，そして Windows ではファイルとしてコピーする (これは既存のファイルに対してだけ動作し，ディレクトリに対しては動作しない)，そしてハードリンクをリンクとして回復する．もしリンク操作が失敗すると (FAT ファイルシステムで起こるかもしれないような)，ファイルのコピーが試みられる．これはファイルの読み込みに gzfile を使うため，関数が処理できる任意のメソッドで圧縮されたファイルを処理できる：少なくとも compress, gzip, bzip2 そして xz 圧縮と，ある種のタイプの lzma 圧縮．これは絶対パスの回復に対してはある種の tar 実装がそうするようには防御されていない．これは必要ならアーカイブ中のファイルに対するディレクトリの親ディレクトリを作る．これは標

準(USTAR/POSIX)，GNU と pax 流の 100 バイトを超えるファイルパス，そして GNU 流の 100 バイトを超えるリンクターゲット処理を処理する．

内部実装からの

```
unsupported entry type 'x'
```

のような警告が出るかもしれない．これはしばしば不正なアーカイブを指示する：エントリータイプ "A-Z" は拡張子として許されるが，他のタイプは予約されている．そうしたアーカイブに対して出来る唯一の事はそれを処理できる tar プログラムを見つけ，そして結果のファイルを慎重に調べることである．又

```
using pax extended headers
```

という警告が出るかもしれない．これは ACL，エンコーディング等の追加の情報が捨て去られたかもしれないことを指示する．

標準は ASCII のファイル名だけをサポートする (実際はアルファニューメリックにピリオッド，アンダースコア，そしてハイフンを加えたもの)． untar はファイル名を現在のシステム上で受け入れられるものにマップする試みを一切せず，そしてアーカイブ中のファイル名を現在のロケール中に再エンコーディングすることなしに適用できるかのように扱う．

Value

もし list = TRUE ならば tar アーカイブ中に含まれるファイルの (相対もしくは絶対)パスの文字列ベクトル．

さもなければ外部 tar を用いた system からの復帰コードか 0L で，不可視．

See Also

tar, unzip.

---

unzip　　*Zip* アーカイブから取り出す又はリストする

Description

zip アーカイブからファイルを取り出すかリストする．

Usage

```
unzip(zipfile, files = NULL, list = FALSE, overwrite = TRUE,
      junkpaths = FALSE, exdir = ".", unzip = "internal",
      setTimes = FALSE)
```

Arguments

zipfile	zip ファイルのパス名：チルド展開が行われる (path.expand を見よ).
files	取り出す記録されたファイルパスの文字列ベクトル：既定では全てのファイルを取り出す.
list	もし TRUE ならファイルをリストし何も取り出さない． unzip -l と同値.
overwrite	もし TRUE ならば既存のファイルを上書きする，さもなければそうしたファイルを無視する． unzip -o と同値.
junkpaths	もし TRUE ならば，取り出し時に保存された基本名だけを使う．unzip -j と同値.
exdir	取り出したファイルを置くディレクトリ (unzip -d と同値)．必要なら作られる.
unzip	使われるメソッド．別法は getOption("unzip") を使うことで，これは Unix 風システムでは unzip プログラムへのパスに設定できる.
setTimes	論理値．内部メソッド専用で，ファイルの時刻は zip ファイル中の時刻に設定されるべきか? (注意：これは含まれるファイルに適用され，ディレクトリにはされない．)

Value

もし list = TRUE ならば，列 Name (c文字列)， Length (非圧縮ファイルのサイズ，数値) そして Date (クラス "POSIXct" の)を持つデータフレーム.

さもなければ "internal" メソッドに対してはそこに取り出されるファイルパスの文字列ベクトルで不可視.

Note

既定の内部メソッドは最少の実装であり，主として Windows のユーザが外部ソフトウェア無しに Windows のバイナリパッケージを開封出来るようにデザインされている．これは(例えば) zip 3.0 で導入された Unicode ファイル名をサポートしていない：そのためには unzip = "unzip" を unzip 6.00 又はそれ以降で使う．これは bzip2 圧縮と > 2GB zip ファイルをサポートしている (しかし zip ファイル中の予め圧縮された >= 4GB のファイルはしない： unzip の多くのビルドと同様にこれはそれらを打ち切るかもしれない， R の場合可能なら警告が出る).

もし unzip がプログラムを指定すると， list = TRUE でリストされた時刻の書式は未知であり(Windows ではこれは現在のロケールにすら依存する可能性がある)，そして返り値は NA や間違ったタイムゾーンで表現されたり誤って解釈されたりする (後者は unzip 6.00 以来滅多にありそうもない).

zip ファイル中のファイル時刻は MS-DOS のスタイルで保管されており，ローカル時間は 2 秒の精度を持つ．これはファイルを計算機同士でやり取りする (大陸間ですら)際は特に有用では無いため我々はそれらを既定で回復しない.

Source

内部 C コードは zlib を使い，そして特に Gilles Vollant による zlib ソース (from http://zlib.net)中の 'minizip' アプリケーションに基づく.

See Also

zip ファイルから単一の成分を取り出すには unz.

zip.

---

update.packages　　インストールされたパッケージを *CRAN* 風のレポジトリと比較する

Description

old.packages はレポジトリに (適当な)新しいバージョンを持つパッケージを指示し，一方 update.packages はそうしたパッケージのダウンロードとインストールを提供する．

new.packages はレポジトリ上の既にインストールされていないパッケージを探し，そしてオプションでそれらのインストールを提供する．

Usage

```
update.packages(lib.loc = NULL, repos = getOption("repos"),
                contriburl = contrib.url(repos, type),
                method, instlib = NULL,
                ask = TRUE, available = NULL,
                oldPkgs = NULL, ..., checkBuilt = FALSE,
                type = getOption("pkgType"))

old.packages(lib.loc = NULL, repos = getOption("repos"),
             contriburl = contrib.url(repos, type),
             instPkgs = installed.packages(lib.loc = lib.loc),
             method, available = NULL, checkBuilt = FALSE,
             type = getOption("pkgType"))

new.packages(lib.loc = NULL, repos = getOption("repos"),
             contriburl = contrib.url(repos, type),
             instPkgs = installed.packages(lib.loc = lib.loc),
             method, available = NULL, ask = FALSE, ...,
             type = getOption("pkgType"))
```

Arguments

lib.loc	探索される(そしてそこでパッケージを更新する) R のライブラリツリーの位置を記述する文字列ベクトル，又は全ての既知のツリーに対しては NULL (.libPaths を見よ).
repos	文字列ベクトル，使用するレポジトリの基本 URL，例えば "http://cran.us.r-project.org" のような CRAN の URL.
contriburl	レポジトリの貢献セクションの URL．もしレポジトリが不完全であればこの引数を使う．引数 repos を上書きする．type = "both" と両立しない．
method	ダウンロードの方法，download.file を見よ．もし NULL でない available が使われると使用されない．

instlib　パッケージをインストールするライブラリのディレクトリを与える文字列.

ask　パッケージをダウンロードしインストールする前にユーザに尋ねるかどうかを指示する論理値，もしくは文字列 "graphics" で，これはユーザが更新可能なパッケージのリストから選択するか否かの選択を許すウィジェットを提示する．(後者の値は select.list の GUI バージョンを持つシステム上でだけ動作し，そしてさもなければ ask = TRUE と同値．)

available　available.packages が返すようなオブジェクトでレポジトリで利用可能なパッケージをリストする，もしくは available.packages の内部呼び出しを行う NULL． type = "both" と両立しない.

checkBuilt　もし TRUE ならば，過去の R のメジャー・マイナーバージョン(例えば 3.1) の下で構築されたパッケージを '旧式'であると考える.

oldPkgs　もし NULL で無いと指定されると．update.packages() はこれらのパッケージだけを更新対象とする．これはパッケージ名の文字列ベクトルか old.packages が返すような行列で良い.

instPkgs　既定で全てのインストールされたパッケージ，installed.packages(lib.loc = lib.loc)．サブセットを指定できる：現在これは installed.packages() が返すのと同じ(文字列行列) 書式でなければならない.

...　install.packages に渡される destdir と dependencies のような引数.

type　文字列，ダウンロードしインストールするパッケージのタイプを指示する． install.packages を見よ.

Details

old.packages は available.packages からの情報を instPkgs (既定で installed.packages により計算される) からのそれと比較し，レポジトリに新しいバージョンがあるインストール済みパッケージを報告する．又はもし checkBuilt = TRUE ならば，それは R の以前のマイナーバージョンの下で構築されたものである (例えば R 3.2.0 が稼働中に 3.1.x の下で構築された)． (バイナリのパッケージタイプに対してはここでレポジトリ上のバージョンが現在の R のマイナーバージョンの下で構築されたかをチェックしないが，しかしそれはこのバージョンに適していると宣伝されている．)

new.packages は同じ比較を行うがレポジトリで利用可能なインストールされていないパッケージを報告する．もし ask != FALSE ならば，これはどのパッケージが lib.loc の最初の要素中にインストールされるべきかを問い合わせる.

セットの主要な関数は update.packages である．先ず lib.loc 中で見つかった全てのパッケージのリストが作られ，そしてレポジトリで利用可能なそれらと比較される．もし ask = TRUE (既定)ならばより新しいバージョンを持つパッケージが報告され，その各々に対してそれを更新するかどうかをユーザが指定できる．もしそうならパッケージがレポジトリからダウンロードされそして対応するライブラリパス(又はもし指定されていれば instlib)中にインストールされる.

適当な利用可能なパッケージのリストがどのように決定されるかについては available.packages を見よ． available = NULL は呼び出し available.packages(contriburl = contriburl, method = method) を行い，従って既定で R バージョン，OS タイプに関して選別しそして重複するものを取り除く.

Value

update.packages は NULL を不可視で返す.

old.packages に対しては NULL か，行列で各パッケージ毎に一つの列で行名はパッケージ名で列名は "Package", "LibPath", "Installed" (バージョン), "Built" (そのもとで構築されたバージョン), "ReposVer" そして "Repository".

new.packages に対しては ask により選択されたものがインストールされた 後のパッケージ名の文字列ベクトル.

警告

dependencies (install.packages に渡される)を update.packages とともに使うには注意がいる，なぜなら新しい従属性をどこにインストールするかが明らかではないからである．現在の実装は，そのライブラリが使われる時，更新される全てのパッケージが単一のライブラリ中に更新されることだけを認めるからである.

See Also

install.packages, available.packages, download.packages, installed.packages, contrib.url.

options の下で install.packages に対してリストされるオプション.

プロキシとファイル転送をモニターする他のオプションを如何に処理するかについては download.file を見よ.

INSTALL, REMOVE, remove.packages, library, .packages, read.dcf

レポジトリをどのように設定するかについてはマニュアル 'R Installation and Administration'.

---

url.show　　テキストの *URL* を表示する

Description

遠隔サーバーからのテキストファイルを表示する file.show の拡張.

Usage

```
url.show(url, title = url, file = tempfile(),
         delete.file = TRUE, method, ...)
```

Arguments

url	そこから読み取る URL.
title	ブラウザに対するタイトル.
file	コピー先のファイル.
delete.file	ファイルを事後に消すか?
method	ファイルの転送メソッド：download.file を見よ.
...	file.show に渡される引数.

Note

これはテキストファイル用なので， Windows では行末を CRLF に変換する.

See Also

url, file.show, download.file

Examples

```
## Not run: url.show("http://www.stats.ox.ac.uk/pub/datasets/csb/ch3a.txt")
```

URLencode (部分的な) *URL* をエンコードまたはデコードする

Description

URL 中の文字列をパーセントエンコードまたはデコードする関数.

Usage

```
URLencode(URL, reserved = FALSE, repeated = FALSE)
URLdecode(URL)
```

Arguments

URL	文字列.
reserved	論理値：'予約'文字をエンコードすべきか? '詳細'節を見よ.
repeated	論理値：見かけ上すでにエンコードされている URL を再びエンコードすべきか?

Details

URL 中の英数字以外の文字と '- _ . ~' は % に二桁の16進数表現でエンコードされるべきで，そして任意の一バイト文字はそのようにエンコードすることが出来る． (多バイト文字はバイト毎にエンコードされる．) 標準ではこれは'パーセントエンコーディング'と呼ばれる.

加えて，'! $ & ' ( ) * + , ; = : / ? @ # [ ]' は予約文字で，そしてそれらのスキーム固有の予約された意味以外で使われない限りエンコードされるべきである． URLencode 中の既定ではそれらをそのままにしておき，これは 'file://' URL に対しては適切であるが，しかし恐らく 'http://' に対してはそうではない.

'見かけ上すでにエンコードされた URL'とは二つの16進数に対して %xx を含むものである.

Value

文字列.

References

インターネット STD 66 (以前の RFC 3986)， https://tools.ietf.org/html/std66

Examples

```
(y <- URLencode("a url with spaces and / and @"))
URLdecode(y)
(y <- URLencode("a url with spaces and / and @", reserved = TRUE))
URLdecode(y)

URLdecode(z <- "ab%20cd")
c(URLencode(z), URLencode(z, repeated = TRUE)) # 最初が普通必要とされる
```

---

utils-deprecated　パッケージ *list("utils")* 中の廃止予定の関数

Description

これらの関数は R のより古いバージョンとの互換性のためだけに提供されており，次のリリースではすぐに廃止されるかもしれない．

See Also

Deprecated, Defunct

View　データビューアを起動する

Description

行列風の R オブジェクトに対して表計算ソフト風のデータビューアを起動する．

Usage

```
View(x, title)
```

Arguments

x	ゼロでない行と列数のデータフレームに強制変換できる R オブジェクト．
title	ビューアウィンドウに対するタイトル．既定では接頭辞 Data: を持つ x の名前．

Details

オブジェクト x は(もし可能なら)データフレームに強制変換され，それから列は format.data.frame を用いて文字列に変換される．オブジェクトはそれから data.entry の読み取り専用のバージョンである表計算ソフト風のデータビューア中で眺められる．

もし 1:nrow でないデータフレームの行名があれば，それらは row.names と呼ばれる別個の最初の列中に表示される．

ゼロの列やゼロの行を持つオブジェクトは受け入れられない．

セルの配列はスクロールバーや Cursor, Home, End, Page Up, Page down キー (X11 でサポートされていれば)，そして Enter や Tab キー経由で巡回できる．

Value

不可視の NULL．関数はウィンドウを開き即座に返る：ウィンドウはその制御やメニューで閉じることが出来る．

See Also

edit.data.frame, data.entry.

---

vignette　　パッケージのビニエットを眺める，リストする又はソースを得る

Description

指定されたパッケージのビニエットを眺める，又は利用可能なものをリストする；ビューアで整形されたそれを表示する，そしてその R ソースファイルを編集する．

Usage

```
vignette(topic, package = NULL, lib.loc = NULL, all = TRUE)

## S3 method for class 'vignette'
print(x, ...)
## S3 method for class 'vignette'
edit(name, ...)
```

Arguments

topic	眺めるビニエットの (基本)名を与える文字列．もし省略されると，全てのインストールされたパッケージからのビニエットがリストされる．
package	探されるパッケージの名前を持つ文字列，又は NULL で‘全て’のパッケージ (引数 all) で定義されたようなものが探される．
lib.loc	R ライブラリのディレクトリ名の文字列ベクトル，又は NULL．既定値の NULL は現在既知の全てのライブラリに対応する．
all	論理値；もし TRUE ならば lib.loc で指定されるライブラリツリー中の全ての利用可能なパッケージが探され，そしてもし FALSE ならば，付加されたパッケージだけが探される．
x, name	クラス vignette のオブジェクト．
...	print メソッドでは無視され，edit メソッドで file.edit に渡される．

Details

関数 vignette は同じクラスのオブジェクトを返し，プリントメソッドはそれに対してビューアを開く． pdfviewer オプションで指定されるプログラムはビニエットの PDF バージョンを眺めるために使われる．もし幾つかのビニエットが topic と同じ名前の PDF/HTML バージョンを持てば，見つかった最初のものが使われる．

もし topic が与えられないと，全ての利用できるビニエットがリストされる．対応する情報はクラス "packageIQR" のオブジェクトとして返される．

See Also

HTML ベースのビニエットブラウザについては browseVignettes；
RShowDoc(<basename>, package = "<pkg>") は "整形された"ビニエット(pdf か html) を表示する.

Examples

```
## 付加されたパッケージからビニエットをリストする
vignette(all = FALSE)

## 全てのインストールされたパッケージからのビニエットをリストする(時間がかかるかもしれない!):
vignette(all = TRUE)

## grid の入門ビニエット -- それを開く
## Not run: vignette("grid") # print() を呼び出し
## 同じ(ビニエットの存在に対する条件付き).
## 多くのインストールされたパッケージの場合 'package = *' は相当早いことを注意：
if(!is.null(v1 <- vignette("grid", package="grid"))) {
## Not run: v1 # calling print(.)
  str(v1)
  ## ここでコードを注意深く眺める
## Not run: edit(v1) # 例えば行を送る ...
}# if( has vignette "installed")
## パッケージは複数のビニエットを持つことが出来る(パッケージ grid が複数持つ)：
vignette(package = "grid")
if(interactive()) {
   ## vignette("rotated")
   ## 同じだがそれを探さない：
   vignette("rotated", package = "grid")
}
```

---

write.table　　データ出力

Description

write.table はその要求された引数 x をファイル又は connection にプリントする (もしそれがデータフレームでも行列でもなければデータフレームに変換した後で).

Usage

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")

write.csv(...)
write.csv2(...)
```

Arguments

x	書き込まれるオブジェクト，行列かデータフレームが好ましい．もしそうでなければ， x をデータフレームに強制変換しようとする．
file	ファイルの名前を与える文字列か書き込みのために開かれた connection． "" はコンソールへの出力を指示する．
append	論理値．file が文字列の時だけ関係する．もし TRUE ならば，出力はファイルに追記される．もし FALSE ならば，その名前の任意の既存のファイルは破棄される．
quote	論理値 (TRUE か FALSE) 又は数値ベクトル．もし TRUE ならば任意の文字列又は因子の列は二重引用符で囲まれる．もし数値ベクトルならば，その要素は引用化される列の添字とされる．どちらの場合も，列名と行名は書き込まれる時引用化される．もし FALSE ならば何も引用化されない．
sep	欄を分離する文字列． x の各行内の値はこの文字列で分離される．
eol	各行の終端にプリントされる文字列．例えば eol = "\r\n" は Unix 風 OS で Windows の行末を作り，そして eol = "\r" は Excel:mac 2004 が期待するようなファイルを作り出す．
na	データ中の欠損値に対して使われる文字列．
dec	数値もしくは複素数値中で小数点に対して使われる文字列：単一文字でなければならない．
row.names	x の行名が x と一緒に書き込まれるべきかどうかを指示する論理値か，書き込まれる行名の文字列．
col.names	x の列名が x と一緒に書き込まれるべきかどうかを指示する論理値か，書き込まれる列名の文字列． col.names = NA の意味については‘CSV ファイル’節を見よ．
qmethod	文字列の引用化の際埋め込みの二重引用符文字をどのように扱うかを指示する文字列．次のどちらかでなければならない， "escape" (write.table に対する既定値)，この場合引用符文字はバックスラッシュで C 風にエスケープされる，又は "double" (write.csv と write.csv2 に対する既定値)，この場合それは二重化される．最初の文字を指定すれば良い．
fileEncoding	文字列：もし空でなければファイル(コネクションでは無い)に対して使われるエンコーディングを宣言するので，文字列データは書き込まれる際再エンコーディングされる． file を見よ．
...	write.table への引数：append, col.names, sep, dec そして qmethod は変更できない．

Details

もしテーブルが列を持たなければ行名は row.names = TRUE の時だけ書き込まれ，そして逆も真である．

実数と複素数は最大可能精度で書き込まれる．

もしデータフレームが行列風の列を持てば，これらは結果中の複数の列に変換され (as.matrix を使い)ので，文字列の col.names 又は数値の quote は入力ではなく結果の列を参照すべきである．そうした行列風の列は既定で引用化されない．

リストやクラスを持つ(例えば日付)データフレーム中の任意の列は適当な as.character メソッドで変換される：そうした列は既定で引用化されない．一方で，行列に対する任

意のクラス情報は捨て去られ，そして原子的で無い(例えばリスト)行列は文字列に強制変換される．

文字列に変換されている列だけがもし quote で指定されていれば引用化される．

dec 引数は文字列への変換がされない列にだけ適用される，なぜならそれらはクラスを持つか行列風の列(又は行列)の一部だからである，特に I() で保護された列に適用される．そうした強制変換を制御するには options("OutDec") を使う．

殆ど全てのケースで数値量への強制変換はオプション "scipen" (options を見よ)で制御されるが，内部的には digits = 15 に同値である．より細かい制御には文字列行列/データフレームを作る format を使い，そしてそれに write.table を呼び出す．

これらの関数は出力の1000行毎にユーザによる中断をチェックする．

もし file が開かれていないコネクションならば，それを開こうとし，そうして使用後はそれを閉じる．

Windows で Unix 風のファイルを書き込むには例えば file = file("filename", "wb") のようなバイナリコネクションを使う，

CSV ファイル

既定では行名の列には列名は無い．もし col.names = NA で row.names = TRUE ならば空白の列名が加えられる，これはスプレッドシートにより読み込まれる CSV ファイルに対して使われる慣例である．そうした CSV ファイルは R 中に次のようにして読み込むことが出来ることを注意する

```
read.csv(file = "<filename>", row.names = 1)
```

write.csv と write.csv2 は CSV ファイルの書き出しに対する便利なラッパである．これらは sep と dec (下を見よ), qmethod = "double" を設定し，そしてもし row.names = TRUE (既定)ならば col.names を NA に，さもなければ TRUE に設定する．

write.csv は小数点として "." を，そして分離記号としてコンマを使う．

write.csv2 はコンマを小数点にそしてセミコロンを分離記号に使う．これはある種の西ヨーロッパロケールでの CSV ファイルに対する Excel の慣例である．

これらのラッパは故意に扱いにくくしてある：これらは適正なファイルを書き出すために正しい慣例を使うことを保証する様にデザインされている． append, col.names, sep, dec 又は qmethod を変更しようとしても無視され，警告が出る．

CSV ファイルはエンコーディングを記録せず．そしてこれはもし ASCII でなければ多くの他のアプリケーションで問題を惹き起こす．Windows Excel 2007/10 はもしそれらが ASCII か UTF-16 (fileEncoding = "UTF-16LE" を使う)，又は恐らく現在の Windows のコードページ(例えば "CP1252") ならばファイルを正しく開く(例えばファイル連携機構により)が，‘Text Import Wizard’ (‘Data’ タブからの)はより多くのエンコーディングの選択を許す． Excel:mac 2004/8 は ‘Macintosh’ (Mac Roman を意味するらしい)，‘Windows’ (恐らく Latin-1) そして ‘PC-8’ ファイルだけを 移入出来る． OpenOffice 3.x はファイルを開く時文字セットを問い合わせる．

CSV ファイルに対する規約 IETF RFC4180 (https://tools.ietf.org/html/rfc4180) があり，これはコンマを分離記号にそして CRLF を行末に指定している．write.csv は Windows 上で規約に沿ったファイルを書き出す：他のプラットフォームでは eol = "\r\n" を使う．

Note

write.table は沢山の (百又はそれ以上)列を持つデータフレームに対しては遅いかもしれない：各列は異なったクラスかもしれず別個に処理されねばならないためこれは避けがたい．もしそれらが全て同じクラスならば，代わりに行列を使うことを考えよう．

See Also

マニュアル 'R Data Import/Export'.

read.table, write.

パッケージ **MASS** 中の write.matrix.

Examples

```
## Not run:
## Excel への入力用に CSV ファイルを書き出すには次が使える
x <- data.frame(a = I("a \" quote"), b = pi)
write.table(x, file = "foo.csv", sep = ",", col.names = NA,
            qmethod = "double")
## そしてこのファイルを R に読み戻すには次が必要
read.table("foo.csv", header = TRUE, sep = ",", row.names = 1)
## 注意：もし qmethod = "double" なら分離子は不要.

### もしくは
write.csv(x, file = "foo.csv")
read.csv("foo.csv", row.names = 1)
## または行名無しで
write.csv(x, file = "foo.csv", row.names = FALSE)
read.csv("foo.csv")

## Mac Excel 2004/8 で使用するためファイルを Mac Roman で書き出すには
write.csv(x, file = "foo.csv", fileEncoding = "macroman")
## または Windows Excel 2007/10 に対しては
write.csv(x, file = "foo.csv", fileEncoding = "UTF-16LE")

## End(Not run)
```

---

zip　　*Zip* アーカイブを作る

Description

zip アーカイブを作る外部 zip 命令に対するラッパ.

Usage

```
zip(zipfile, files, flags = "-r9X", extras = "",
    zip = Sys.getenv("R_ZIPCMD", "zip"))
```

Arguments

zipfile	zip ファイルのパス名；チルド展開 (path.expand を見よ)が実行される.
files	含められるべき記録されたファイルパスの文字列.
flags	命令に渡されるフラグの文字列：'詳細'節を見よ.
extras	オプションの文字列ベクトル：'詳細'節を見よ.
zip	使用される外部命令を指定する文字列.

Details

Unix 風システムでは zip に対する既定値は既定で R_ZIPCMD の値を使う，これはコンフィグレーションの過程で unzip 命令が見つかれば 'etc/Renviron' 中に設定される．Windows では既定値はパス中にある zip プログラムを使う(例えば Rtools からのもの)．

flags に対する既定値はディレクトリツリーを可搬的な仕方で zip 化するのに相応しいものである：他の可能性については zip 命令のシステム固有のヘルプを見よ．

引数 extras は -x や -i の後に除外したり含めたりするファイルパスを指定するのに使うことが出来る．

Value

外部命令が返す状態値で不可視．

See Also

unzip, unz.

Index

∗Topic **IO**
rcompgen, 128
∗Topic **arith**
roman, 150
∗Topic **character**
adist, 4
aregexec, 8
glob2rx, 73
∗Topic **connection**
read.DIF, 133
read.fortran, 135
read.fwf, 136
read.table, 139
∗Topic **datasets**
data, 43
∗Topic **data**
apropos, 6
getAnywhere, 67
getFromNamespace, 68
getS3method, 72
∗Topic **debugging**
findLineNum, 63
recover, 144
∗Topic **documentation**
apropos, 6
bibentry, 16
browseVignettes, 22
cite, 33
data, 43
demo, 50
example, 59
help, 77
help.search, 81
help.start, 84
hsearch-utils, 85
prompt, 122
promptData, 124
promptPackage, 125
Question, 126
RShowDoc, 155
RSiteSearch, 156
str, 174
vignette, 196
∗Topic **environment**
apropos, 6
∗Topic **error**
bug.report, 23
debugger, 48
help.request, 80
∗Topic **file**
browseURL, 21
changedFiles, 27
count.fields, 41
dataentry, 45
file_test, 62
glob2rx, 73
package.skeleton, 111
read.DIF, 133
read.fortran, 135
read.fwf, 136
read.table, 139
tar, 182
untar, 187
unzip, 189
url.show, 193
write.table, 197
zip, 200
∗Topic **interface**
browseEnv, 20
∗Topic **iteration**
combn, 37
∗Topic **list**
relist, 146
∗Topic **manip**
head, 76
relist, 146
stack, 172
type.convert, 186
∗Topic **methods**
getS3method, 72
isS3method, 94
methods, 104
∗Topic **misc**
citation, 31
citEntry, 35
close.socket, 36

make.socket, 101
mirrorAdmin, 106
person, 117
read.socket, 138
sessionInfo, 167
toLatex, 184
url.show, 193
utils-deprecated, 195
*Topic **packages**
globalVariables, 74
*Topic **package**
utils-package, 4
*Topic **print**
format, 66
ls.str, 97
str, 174
write.table, 197
*Topic **programming**
menu, 103
recover, 144
rtags, 158
utils-package, 4
*Topic **utilities**
alarm, 6
aspell, 9
aspell-utils, 11
available.packages, 13
BATCH, 15
bibentry, 16
bug.report, 23
capture.output, 26
changedFiles, 27
chooseBioCmirror, 29
chooseCRANmirror, 30
cite, 33
combn, 37
compareVersion, 38
COMPILE, 39
contrib.url, 40
create.post, 42
dataentry, 45
debugger, 48
demo, 50
download.file, 51
download.packages, 54
edit, 56
edit.data.frame, 57
example, 59
file.edit, 61
fix, 65
flush.console, 66
getParseData, 70
glob2rx, 73
help.request, 80
INSTALL, 86
install.packages, 88
installed.packages, 93
LINK, 95
localeToCharset, 96
ls.str, 97
maintainer, 99
make.packages.html, 100
memory.size, 102
menu, 103
modifyList, 106
nsl, 109
object.size, 110
package.skeleton, 111
packageDescription, 113
packageName, 114
packageStatus, 115
page, 117
PkgUtils, 120
process.events, 121
rcompgen, 128
REMOVE, 148
remove.packages, 148
RHOME, 150
Rprof, 151
Rprofmem, 153
Rscript, 154
RSiteSearch, 156
rtags, 158
Rtangle, 159
RweaveLatex, 161
savehistory, 165
select.list, 166
setRepositories, 169
SHLIB, 170
sourceutils, 171
str, 174
summaryRprof, 177
Sweave, 179
SweaveSyntConv, 181
tar, 182
txtProgressBar, 185
untar, 187
unzip, 189
update.packages, 191
URLencode, 194
View, 195
zip, 200
*Topic **utility**
removeSource, 149

.DollarNames (rcompgen), 128
.GlobalEnv, *44*
.Last, *166*
.S3methods (methods), 104
.deparseOpts, *57*
.libPaths, *44*, *60*, *84*, *88*, *89*, *93*, *149*, *191*
.packages, *93*, *193*
.romans (roman), 150
::, *45*
:::, *69*
?, *79*, *83*, *156*
? (Question), 126
??, *79*, *127*
?? (help.search), 81
[.getAnywhere (getAnywhere), 67
[.person (person), 117
$.person (person), 117

adist, 4, *9*
agrep, *5*, *8*, *9*, *82*
alarm, 6
approx, *123*
apropos, 6, *83*
aregexec, 8
args, *68*, *98*, *129*, *174*, *175*
argsAnywhere (getAnywhere), 67
array, *37*
as.character, *8*
as.character.person (person), 117
as.list, *173*
as.matrix, *140*, *198*
as.person (person), 117
as.personList (person), 117
as.relistable (relist), 146
as.roman (roman), 150
aspell, 9, *12*, *13*
aspell-utils, *11*, 11
aspell_package_C_files (aspell-utils), 11
aspell_package_R_files (aspell-utils), 11
aspell_package_Rd_files (aspell-utils), 11
aspell_package_vignettes (aspell-utils), 11
aspell_write_personal_dictionary_file (aspell-utils), 11
assignInMyNamespace (getFromNamespace), 68
assignInNamespace (getFromNamespace), 68
atomic, *143*
attach, *26*, *113*
attributes, *108*
available.packages, 13, *40*, *52*, *54*, *55*, *89*, *93*, *116*, *192*, *193*

backtick, *78*, *126*
BATCH, 15
bibentry, 16, *32*, *33*, *36*
bibstyle, *17*, *33*, *34*
browseEnv, 20
browser, *48*, *49*, *63*, *144*, *145*
browseURL, 21, *23*, *24*, *78*, *85*, *107*, *157*
browseVignettes, 22, *197*
bug.report, 23, *42*, *43*, *81*, *99*
build (PkgUtils), 120
buildVignette, *180*
by, *178*

c.person (person), 117
capabilities, *37*, *101*
capture.output, 26
cat, *158*, *159*
changedFiles, 27
character, *60*, *105*, *125*, *150*, *175*
check, *60*
check (PkgUtils), 120
checkCRAN (mirrorAdmin), 106
checkFF, *74*
choose, *38*
chooseBioCmirror, 29, *31*, *169*, *170*
chooseCRANmirror, *30*, 30, *170*
CITATION (citation), 31
citation, 31, *36*, *119*
cite, 33
citeNatbib (cite), 33
citEntry, *32*, 35, *184*
citFooter (citEntry), 35
citHeader (citEntry), 35
class, *105*
close, *140*
close.socket, 36, *101*, *138*
close.txtProgressBar (txtProgressBar), 185
combn, 37
compareVersion, 38
COMPILE, 39, *96*, *171*
completion (rcompgen), 128
connection, *26*, *41*, *106*, *122*, *124*, *125*, *133*, *135*, *137*, *140*, *182*, *188*, *197*, *198*
contrib.url, *15*, *31*, 40, *55*, *93*, *193*
count.fields, 41, *143*
create.post, *24*, *26*, 42, *80*, *81*

data, 43, *79*, *129*, *131*
data.entry, *57*, *59*, *195*, *196*

data.entry (dataentry), 45
data.frame, *134–137*, *143*
dataentry, 45
Date, *108*
de (dataentry), 45
debugger, 48
Defunct, *195*
delayedAssign, *175*
demo, 50, *60*
deparse, *149*
Deprecated, *195*
dev.off, *164*
devAskNewPage, *50*, *60*
dontCheck, *74*
double, *187*
download.file, *13*, 51, *55*, *89*, *91*, *93*, *115*, *191*, *193*, *194*
download.packages, *15*, *40*, *54*, 54, *93*, *193*
dput, *57*, *117*, *174*
dump.frames, *144*, *145*
dump.frames (debugger), 48
dyn.load, *40*, *171*

edit, *47*, 56, *59*, *62*, *65*, *68*, *69*, *117*
edit.data.frame, *57*, 57, *65*, *196*
edit.matrix (edit.data.frame), 57
edit.vignette (vignette), 196
emacs (edit), 56
Encoding, *142*
environment, *20*, *43*, *48*, *72*, *95*
environment variables, *79*
example, 59
exists, *98*
expand.grid, *38*

factor, *105*, *187*
file, *61*, *134*, *137*, *140*, *142*, *155*, *198*
file.access, *62*
file.edit, *42*, *43*, 61, *196*
file.exists, *63*
file.info, *27–29*, *63*
file.path, *63*
file.show, *62*, *117*, *156*, *193*, *194*
file_test, *28*, *29*, 62
files, *62*
fileSnapshot (changedFiles), 27
find (apropos), 6
find.package, *94*
findLineNum, 63
fix, *56*, *57*, *62*, 65, *69*, *117*
fixInNamespace (getFromNamespace), 68
flush.console, *6*, 66
for, *123*
format, *17*, 66, *175*, *199*
format.data.frame, *195*
format.object_size (object.size), 110
format.person (person), 117
formatC, *175*
formatDL, *66*
formatOL (format), 66
formatUL (format), 66
formula, *173*
frame, *117*
ftable, *77*
fuzzy matching, *81*

get, *68*, *69*, *72*, *98*
getAnywhere, 67, *72*
getBioCmirrors (chooseBioCmirror), 29
getCRANmirrors (chooseCRANmirror), 30
geterrmessage, *48*
getFromNamespace, *68*, 68
getMethod, *105*, *127*
getOption, *14*, *17*, *79*, *92*, *131*
getPackageName, *115*
getParseData, 70, *172*
getParseText (getParseData), 70
getS3method, *68*, *69*, 72, *95*, *105*
getSrcDirectory (sourceutils), 171
getSrcFilename (sourceutils), 171
getSrcLocation, *70*
getSrcLocation (sourceutils), 171
getSrcref (sourceutils), 171
gettext, *10*
gettextf, *10*
getTxtProgressBar (txtProgressBar), 185
getwd, *140*
glob2rx, *7*, 73
globalVariables, 74
grep, *7*, *9*, *82*, *165*
gzfile, *188*

head, 76
help, *45*, *59*, 77, *83*, *85*, *123*, *126*, *127*, *156*
help.request, *26*, *42*, *43*, 80
help.search, *7*, *79*, 81, *85*, *86*, *157*
help.start, *79*, *83*, 84, *100*, *107*, *156*, *157*
history (savehistory), 165
hsearch-utils, 85
hsearch_db, *83*
hsearch_db (hsearch-utils), 85
hsearch_db_concepts (hsearch-utils), 85
hsearch_db_keywords (hsearch-utils), 85

I, *199*
iconv, *97*, *113*

INSTALL, 86, *93, 94*, *112*, *148*, *170*, *193*
install.packages, *13–15*, *40*, *55*, *87, 88*, 88, *94*, *112*, *115*, *149*, *170*, *192, 193*
installed.packages, *93*, 93, *116*, *131*, *192, 193*
interactive, *50*, *60*
is.relistable *(relist)*, 146
is.vector, *173*
isS3method, 94

library, *39*, *79*, *93*, *129*, *131*, *193*
library.dynam, *171*
limitedLabels *(recover)*, 144
LINK, *40*, 95
list, *37*, *106*, *147*, *169*
list.files, *28*, *158, 159*
lm, *173*
load, *43*
loadhistory *(savehistory)*, 165
localeToCharset, 96
logical, *95*, *104, 105*
Long vectors, *4*
ls, *20, 21*, *97, 98*
ls.str, 97, *175*
lsf.str *(ls.str)*, 97

maintainer, 99
make.names, *134*, *141*
make.packages.html, *84*, 100
make.socket, *37*, 101, *138*
make.unique, *112*
matrix, *37*
md5sum, *28, 29*
Memory-limits, *102*
memory.limit *(memory.size)*, 102
memory.size, 102
menu, *30, 31*, 103, *167*, *169*
message, *10*, *26*
Methods, *105*
methods, *72*, *79*, *95*, 104, *150*
mirror2html *(mirrorAdmin)*, 106
mirrorAdmin, 106
missing, *122*
mode, *7*, *98*
modifyList, 106

NA, *134*, *141*, *185, 186*
name, *50*, *78*, *126*
new.packages *(update.packages)*, 191
news, 107
ngettext, *10*
nlm, *146*
nsl, 109
NULL, *169*
numeric_version, *108*
NumericConstants, *187*

object.size, 110
objects, *7*
old.packages *(update.packages)*, 191
on.exit, *101*
optim, *146, 147*
options, *31*, *40*, *49*, *51*, *54*, *78*, *129*, *131*, *145*, *162*, *169, 170*, *174, 175*, *193*, *199*

package.skeleton, 111, *123*
package_version, *39*, *114*
packageDescription, *32*, *94*, *99*, 113, *168*
packageName, 114
packageStartupMessage, *10*
packageStatus, *39*, 115
packageVersion *(packageDescription)*, 113
page, 117
parse, *70, 71*, *152*
paste, *32*
path.expand, *61*, *182*, *188*, *190*, *200*
pdf, *163*
person, *18*, *32*, 117
personList *(person)*, 117
pico *(edit)*, 56
PkgUtils, 120
POSIXct, *190*
primitive, *152*
print, *17*, *29*, *117*, *163*, *175*
print.bibentry *(bibentry)*, 16
print.Bibtex *(toLatex)*, 184
print.browseVignettes *(browseVignettes)*, 22
print.changedFiles *(changedFiles)*, 27
print.fileSnapshot *(changedFiles)*, 27
print.getAnywhere *(getAnywhere)*, 67
print.hsearch *(help.search)*, 81
print.Latex *(toLatex)*, 184
print.ls_str *(ls.str)*, 97
print.MethodsFunction *(methods)*, 104
print.news_db *(news)*, 107
print.object_size *(object.size)*, 110
print.packageDescription *(packageDescription)*, 113
print.packageIQR *(data)*, 43
print.packageStatus *(packageStatus)*, 115
print.person *(person)*, 117
print.sessionInfo *(sessionInfo)*, 167
print.socket *(make.socket)*, 101
print.vignette *(vignette)*, 196
process.events, 121

prompt, *79*, *112*, 122, *125*, *126*
promptClass, *112*
promptData, *123*, 124
promptImport, *112*
promptImport (prompt), 122
promptMethods, *112*
promptPackage, 125

q, *16*
Question, 126

R.Version, *168*
R.version, *168*
R_HOME, *22*, *30*, *31*, *84*, *156*, *169*
R_INSTALL_TAR (INSTALL), 86
rc.getOption (rcompgen), 128
rc.options (rcompgen), 128
rc.settings (rcompgen), 128
rc.status (rcompgen), 128
rcompgen, 128
RdTextFilter, *10*, *12*
read.csv, *136*
read.csv (read.table), 139
read.csv2 (read.table), 139
read.dcf, *93*, *114*, *193*
read.delim (read.table), 139
read.delim2 (read.table), 139
read.DIF, 133
read.fortran, 135, *137*
read.fwf, *135*, *136*, 136, *143*
read.socket, *37*, *101*, 138
read.table, *41*, *44*, *134–137*, 139, *187*, *200*
readCitationFile (citation), 31
readline, *123*
recover, *49*, 144
regexec, *9*
regexp, *73*
regmatches, *9*
regular expression, *7*, *20*, *81*, *82*, *98*
relist, 146
REMOVE, *88*, *93*, *94*, 148, *149*, *193*
remove.packages, *93*, *148*, 148, *193*
removeSource, 149
require, *94*, *129*, *131*
reserved, *78*, *126*
reshape, *173*
RHOME, 150
roman, 150
Rprof, 151, *154*, *177*, *178*
Rprofmem, *152*, 153, *178*
Rscript, 154
RShowDoc, *88*, 155, *197*
RSiteSearch, *83*, *85*, 156
rtags, 158
Rtangle, 159, *165*, *179–181*
RtangleSetup (Rtangle), 159
RweaveLatex, *160*, 161, *179–181*
RweaveLatexSetup (RweaveLatex), 161

S3Methods, *105*
save, *45*, *48*
savehistory, 165
saveRDS, *10*, *13*
scan, *41*, *135*, *137*, *140*, *141*, *143*
search, *7*, *59*, *97*
select.list, *103*, 166, *192*
seq_len, *37*
sessionInfo, *26*, *81*, 167, *184*
setBreakpoint (findLineNum), 63
setRefClass, *74*
setRepositories, *30*, *31*, *40*, 169
setTxtProgressBar (txtProgressBar), 185
SHLIB, *40*, 170
showMethods, *105*
shQuote, *182*
sink, *26*
source, *43*, *50*, *87*, *178*, *179*
sourceutils, 171
sQuote, *162*
srcfile, *64*, *70*, *71*
srcref, *71*, *149*, *172*
stack, 172
Stangle, *159*
Stangle (Sweave), 179
startDynamicHelp, *78*, *84*
Startup, *22*, *170*
stderr, *26*, *185*
stdin, *140*, *155*
stop, *10*, *26*
str, *21*, *97*, *98*, 174
strOptions (str), 174
strptime, *108*
strwrap, *175*
sub, *73*
summary, *98*, *175*
summary.packageStatus (packageStatus), 115
summaryRprof, *152*, 177
suppressForeignCheck (globalVariables), 74
Sweave, *159–162*, *165*, 179, *181*
SweaveSyntaxLatex (Sweave), 179
SweaveSyntaxNoweb (Sweave), 179
SweaveSyntConv, 181
SweaveTeXFilter, *10*
sys.calls, *152*

Sys.getlocale, *97*, *168*
sys.parent, *48*
Sys.setenv, *53*, *166*
system, *152*, *183*, *189*
system.file, *94*

tail (head), 76
tar, *121*, 182, *189*
tempdir, *13*, *88*
textConnection, *26*
timestamp (savehistory), 165
tk_select.list, *167*
tkProgressBar, *186*
toBibtex, *17*
toBibtex (toLatex), 184
toBibtex.person (person), 117
toLatex, 184
toLatex.sessionInfo (sessionInfo), 167
topenv, *114*
trace, *63, 64*
tracemem, *152*, *154*, *178*
txtProgressBar, 185
type, *152*
type.convert, *133–135*, *140–143*, 186

unlist, *146, 147*, *173*
unlist.relistable (relist), 146
unstack (stack), 172
untar, *87*, *121*, *183, 184*, 187
untrace, *63, 64*
unz, *191*, *201*
unzip, *189*, 189, *201*
update.packages, *88*, *93, 94*, *116*, 191
update.packageStatus (packageStatus), 115
upgrade (packageStatus), 115
url, *52*, *54, 55*, *89*, *140*, *194*
url.show, *54*, 193
URLdecode (URLencode), 194
URLencode, *22*, *52*, 194
utils (utils-package), 4
utils-deprecated, 195
utils-package, 4

vi, *47*
vi (edit), 56
View, 195
vignette, *23*, *156*, 196
vignettes (vignette), 196

warning, *10*, *26*, *187*
with, *26*
write, *200*
write.csv (write.table), 197
write.csv2 (write.table), 197
write.matrix, *200*
write.socket (read.socket), 138
write.table, *143*, 197
writeLines, *184*
wsbrowser (browseEnv), 20

xedit (edit), 56
xemacs (edit), 56

zip, *191*, 200