Package 'icesTAF'

May 24, 2019

Version 3.1-1

Date 2019-05-24

Title Functions to Support the ICES Transparent Assessment Framework

Imports grDevices, lattice, stats, tools, utils, bibtex, remotes

LazyData yes

Description Functions to support the ICES Transparent Assessment Framework <http://taf.ices.dk> to organize data, methods, and results used in ICES assessments. ICES is an organization facilitating international collaboration in marine science.

License GPL (>= 2)

URL http://taf.ices.dk

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation no

Author Arni Magnusson [aut, cre],
Colin Millar [aut],
Alexandros Kokkalis [ctb]

Maintainer Arni Magnusson <arni.magnusson@ices.dk>

Repository CRAN

Date/Publication 2019-05-24 16:11:33 UTC

R topics documented:

icesTAF-package *Functions to Support the ICES Transparent Assessment Framework*

Description

Functions to support the ICES Transparent Assessment Framework, to organize data, methods, and results used in ICES assessments.

Details

Initial TAF steps:

draft.data	draft DATA.bib file
draft.software	draft SOFTWARE.bib file
period	paste period string for DATA.bib
taf.bootstrap	set up data files and software
taf.skeleton	create empty TAF template

Running scripts:

clean	clean TAF directories
make	run R script if needed
makeAll	run all TAF scripts as needed
makeTAF	run TAF script if needed
msg	show message
sourceAll	run all TAF scripts
sourceTAF	run TAF script

File management:

convert.spaces	convert spaces
cp	copy files
mkdir	create directory
os.linux	operating system
os.macos	operating system
os.windows	operating system
read.taf	read TAF table from file
sourceDir	read all *.R files
taf.library	load package from TAF library
taf.unzip	unzip file
write.taf	write TAF table to file

Tables:

div	divide column values
flr2taf	convert FLR to TAF

long2taf	convert long format to TAF
plus	rename plus group column
rnd	round column values
sam2taf	convert SAM to TAF
taf2long	convert TAF to long format
taf2xtab	convert TAF to crosstab
tt	transpose TAF table
xtab2taf	convert crosstab to TAF

Plots:

lim	compute axis limits
taf.colors	predefined colors
taf.png	open PNG graphics device
zoom	change lattice text size

Example tables:

catage.long	long format
catage.taf	TAF format
catage.xtab	crosstab format
summary.taf	summary results

Administrative tools, rarely used in scripts:

deps	list dependencies
dos2unix	convert line endings
download	download file
file.encoding	examine file encoding
latin1.to.utf8	convert file encoding
line.endings	examine line endings
process.bib	read and process metadata
rmdir	remove empty directory
unix2dos	convert line endings
utf8.to.latin1	convert file encoding

Author(s)

Arni Magnusson and Colin Millar.

References

ICES Transparent Assessment Framework: http://taf.ices.dk.

To explore example TAF stock assessments, see the introductory video and tutorial.

`catage.long` *Catch at Age in Long Format*

Description

Small catch-at-age table to describe a long format data frame to store year-age values.

Usage

```
catage.long
```

Format

Data frame containing three columns:

`Year`	year
`Age`	age
`Catch`	catch (millions of individuals)

Details

The data are an excerpt (first years and ages) from the catch-at-age table for North Sea cod from the ICES (2016) assessment.

Source

ICES (2016) Report of the working group on the assessment of demersal stocks in the North Sea and Skagerrak (WGNSSK). *ICES CM 2016/ACOM:14*, p. 656.

See Also

`catage.taf` and `catage.xtab` describe alternative table formats.

`long2taf` converts a long table to TAF format.

`icesTAF-package` gives an overview of the package.

Examples

```
catage.long
long2taf(catage.long)
```

`catage.taf` *Catch at Age in TAF Format*

Description

Small catch-at-age table to describe a TAF format data frame to store year-age values.

Usage

```
catage.taf
```

Format

Data frame containing five columns:

`Year`	year
1	number of one-year-olds in the catch (millions)
2	number of two-year-olds in the catch (millions)
3	number of three-year-olds in the catch (millions)
4	number of four-year-olds in the catch (millions)

Details

The data are an excerpt (first years and ages) from the catch-at-age table for North Sea cod from the ICES (2016) assessment.

Source

ICES (2016) Report of the working group on the assessment of demersal stocks in the North Sea and Skagerrak (WGNSSK). *ICES CM 2016/ACOM:14*, p. 656.

See Also

`catage.long` and `catage.xtab` describe alternative table formats.

`taf2long` and `taf2xtab` convert a TAF table to alternative formats.

`icesTAF-package` gives an overview of the package.

Examples

```
catage.taf
taf2long(catage.taf)
taf2xtab(catage.taf)
```

`catage.xtab` *Catch at Age in Crosstab Format*

Description

Small catch-at-age table to describe a crosstab format data frame to store year-age values.

Usage

```
catage.xtab
```

Format

Data frame with years as row names and containing four columns:

1	number of one-year-olds in the catch (millions)
2	number of two-year-olds in the catch (millions)
3	number of three-year-olds in the catch (millions)
4	number of four-year-olds in the catch (millions)

Details

The data are an excerpt (first years and ages) from the catch-at-age table for North Sea cod from the ICES (2016) assessment.

Source

ICES (2016) Report of the working group on the assessment of demersal stocks in the North Sea and Skagerrak (WGNSSK). *ICES CM 2016/ACOM:14*, p. 656.

See Also

`catage.long` and `catage.taf` describe alternative table formats.

`xtab2taf` converts a crosstab table to TAF format.

`icesTAF-package` gives an overview of the package.

Examples

```
catage.xtab
xtab2taf(catage.xtab)
```

clean *Clean TAF Directories*

Description

Remove TAF directories: data, model, output, report.

Usage

```
clean(dirs = c("data", "model", "output", "report"))
```

Arguments

dirs	directories to delete.

Note

The purpose of removing the directories is to make sure that subsequent TAF scripts start by creating new empty directories.

If any of the `dirs` is `"bootstrap"` it is treated specially. Instead of completely removing the bootstrap directory, only the subdirectories config, data, library, and software are removed. This protects the subdirectory bootstrap/initial and *.bib metadata files from being accidentally deleted.

See Also

`mkdir` and `rmdir` create and remove empty directories.

`sourceTAF` and `sourceAll` run TAF scripts.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:
clean()

## End(Not run)
```

convert.spaces *Convert Spaces*

Description

Convert spaces in filenames.

Usage

```
convert.spaces(file, sep = "_")
```

Arguments

file	filename, e.g. "file name.csv", "*.csv", or "dir/*".
sep	character to use instead of spaces.

Value

TRUE for success, FALSE for failure, invisibly.

See Also

file.rename is the base function to rename files.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
write(pi, "A B.txt")
convert.spaces("A B.txt")

## Many files
convert.spaces("bootstrap/initial/data/*")

## End(Not run)
```

cp *Copy Files*

Description

Copy or move files, overwriting existing files if necessary, and returning the result invisibly.

Usage

```
cp(from, to, move = FALSE)
```

Arguments

from	source filenames, e.g. *.csv.
to	destination filenames, or directory.
move	whether to move instead of copy.

Value

TRUE for success, FALSE for failure, invisibly.

Note

To prevent accidental loss of files, two safeguards are enforced when move = TRUE:

1. When moving files, the to argument must either have a filename extension or be an existing directory.
2. When moving many files to one destination, the to argument must be an existing directory.

If these conditions do not hold, no files are changed and an error is returned.

See Also

file.copy and unlink are the underlying functions used to copy and (if move = TRUE) delete files.

file.rename is the base function to rename files.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
write(pi, "A.txt")
cp("A.txt", "B.txt")
cp("A.txt", "B.txt", move=TRUE)

## Copy directory tree
cp(system.file(package="datasets"), ".")
mkdir("everything")
cp("datasets/*", "everything")
```

```
## End(Not run)
```

---

deps *List Dependencies*

Description

Search R scripts for packages that are required.

Usage

```
deps(path = ".", base = FALSE, installed = TRUE, available = TRUE,
  list = FALSE)
```

Arguments

path	a directory or file containing R scripts.
base	whether to include base packages in the output.
installed	whether to include installed packages in the output.
available	whether to include available packages in the output.
list	whether to return packages in list format, split by script.

Value

Names of packages as a vector, or in list format if `list=TRUE`. If no dependencies are found, the return value is `NULL`.

Note

Package names are matched based on four patterns:

```
library(*)
require(*)
*::object
*:::object
```

The search algorithm may return false-positive dependencies if these patterns occur inside if-clauses, strings, comments, etc.

See Also

`installed.packages`, `available.packages`.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:
dir <- system.file(package="MASS", "scripts")
script <- system.file(package="MASS", "scripts/ch08.R")

deps(script)                    # dependencies
deps(script, base=TRUE)         # including base packages
deps(script, installed=FALSE)  # not (yet) installed

deps(dir)
deps(dir, list=TRUE)

deps(dir, available=FALSE)  # dependencies that might be unavailable

## End(Not run)
```

`div` *Divide Columns*

Description

Divide column values in a data frame with a common number.

Usage

```
div(x, cols, by = 1000, grep = FALSE, ...)
```

Arguments

`x`	a data frame.
`cols`	column names, or column indices.
`by`	a number to divide with.
`grep`	whether `cols` is a regular expression.
`...`	passed to `grep()`.

Value

A data frame similar to `x`, after dividing columns `cols` by the number `by`.

Note

Provides notation that is convenient for modifying many columns at once.

See Also

transform can also be used to recalculate column values, using a more general and verbose syntax.

grep is the underlying function used to match column names if grep is TRUE.

rnd is a similar function that rounds columns.

icesTAF-package gives an overview of the package.

Examples

```
# These are equivalent:

x1 <- div(summary.taf, c("Rec","Rec_lo","Rec_hi",
                         "TSB","TSB_lo","TSB_hi",
                         "SSB","SSB_lo","SSB_hi",
                         "Removals","Removals_lo","Removals_hi"))

x2 <- div(summary.taf, "Rec|TSB|SSB|Removals", grep=TRUE)

x3 <- div(summary.taf, "Year|Fbar", grep=TRUE, invert=TRUE)

# Less reliable in scripts if columns have been added/deleted/reordered:

x4 <- div(summary.taf, 2:13)
```

---

dos2unix *Convert Line Endings*

Description

Convert line endings in a text file between Dos (CRLF) and Unix (LF) format.

Usage

```
dos2unix(file)

unix2dos(file)
```

Arguments

file	a filename.

See Also

line.endings examines line endings.

write.taf uses unix2dos to ensure that the resulting files have Dos line endings.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
file <- "test.txt"
write("123", file)

dos2unix(file)
file.size(file)

unix2dos(file)
file.size(file)

file.remove(file)

## End(Not run)
```

`download` *Download File*

Description

Download a file in binary mode, e.g. a model executable.

Usage

```
download(url, dir = ".", mode = "wb", chmod = file_ext(url) == "",
  destfile = file.path(dir, basename(url)), quiet = TRUE, ...)
```

Arguments

`url`	URL of file to download.
`dir`	directory to download to.
`mode`	download mode, see details.
`chmod`	whether to set execute permission (default is `TRUE` if file has no filename extension).
`destfile`	destination path and filename (optional, overrides `dir`).
`quiet`	whether to suppress messages.
`...`	passed to `download.file`.

Details

With the default mode `"wb"` the file is downloaded in binary mode (see `download.file`), to prevent R from adding ^M at line ends. This is particularly relevant for Windows model executables, while the `chmod` switch is useful when downloading Linux executables.

This function can be convenient for downloading any file, including text files. Data files in CSV or other text format can also be read directly into memory using `read.table`, `read.taf` or similar functions, without writing to the file system.

See Also

read.taf reads a TAF table into a data frame.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
url <- paste0("https://github.com/ices-taf/2015_had-iceg/raw/master/",
              "bootstrap/initial/software/catageysa.exe")
download(url)

## End(Not run)
```

---

draft.data *Draft DATA.bib*

Description

Create an initial draft version of a 'DATA.bib' metadata file.

Usage

```
draft.data(originator = NULL, year = format(Sys.time(), "%Y"),
  title = NULL, period = NULL, source = "file", file = "",
  data.dir = "bootstrap/initial/data", data.files = dir(data.dir,
  recursive = TRUE), append = FALSE)
```

Arguments

originator	who prepared the data, e.g. a working group acronym.
year	year of the analysis when the data were used. The default is the current year.
title	description of the data, including survey names or the like.
period	a string of the form "1990-2000", indicating the first and last year that the data cover, separated by a simple dash. Alternatively, a single number if the data cover only one year. If the data do not cover specific years, this metadata field can be suppressed using period = FALSE.
source	where the data are copied/downloaded from. This can be a URL, filename, or the special value "file".
file	optional filename to save the draft metadata to a file.
data.dir	directory containing data files.
data.files	data filenames. The default is all files inside data.dir.
append	whether to append metadata entries to an existing file.

Details

Typical usage is to specify `originator`, while using the default values for the other arguments. Most data files have the same originator, which can be specified to facilitate completing the entries after creating the initial draft.

The special value source = "file" is described in the `process.bib` help page, along with other metadata information.

The default value `file = ""` prints the initial draft in the console, instead of writing it to a file. The output can then be pasted into a file to edit further, without accidentally overwriting an existing metadata file.

This function is intended to be called from the top directory of a TAF analysis which contains a '`bootstrap/initial/data`' directory, as reflected in the default value of `data.dir`.

Value

Object of class Bibtex.

Note

After creating the initial draft, the user can complete the description of each data file inside the title field and look into each file to specify the period that the data cover.

See Also

`period` pastes two years to form a `period` string.

`draft.software` creates an initial draft version of a '`SOFTWARE.bib`' metadata file.

`process.bib` reads and processes metadata entries. The help page contains example metadata entries and commentary.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:
# Print in console
draft.data("WGEF", 2015)

# Export to file
draft.data("WGEF", 2015, file="bootstrap/DATA.bib")

## End(Not run)
```

`draft.software` *Draft SOFTWARE.bib*

Description

Create an initial draft version of a '`SOFTWARE.bib`' metadata file.

Usage

```
draft.software(package, author = NULL, year = NULL, title = NULL,
  version = NULL, source = NULL, file = "", append = FALSE)
```

Arguments

`package`	name of one or more installed R packages, or files/folders inside '`bootstrap/initial/software`'.
`author`	author(s) of the software.
`year`	year when this version of the software was released, or the publication year of the cited manual/article/etc.
`title`	title or short description of the software.
`version`	string to specify details about the version, e.g. GitHub branch and commit date.
`source`	string to specify where the software are copied/downloaded from. This can be a GitHub reference of the form owner/repo[/subdir]@ref, URL, or a filename.
`file`	optional filename to save the draft metadata to a file.
`append`	whether to append metadata entries to an existing file.

Details

Typical usage is to specify `package`, while using the default values for the other arguments.

With the default version = NULL, the function will automatically suggest an appropriate version entry for CRAN packages, but for GitHub packages it is left to the user to add further information about the GitHub branch (if different from master) and the commit date.

With the default source = NULL, the function will automatically suggest an appropriate source entry for CRAN and GitHub packages, but for other R packages it is left to the user to add information about where the software can be accessed.

The default value `file = ""` prints the initial draft in the console, instead of writing it to a file. The output can then be pasted into a file to edit further, without accidentally overwriting an existing metadata file.

Value

Object of class Bibtex.

Note

After creating the initial draft, the user can complete the version, source, and other fields as required.

This function is especially useful for citing exact versions of R packages on GitHub. To prepare metadata for software other than R packages, see the process.bib help page for an example.

See Also

citation and packageDescription are the underlying functions to access information about installed R packages.

draft.data creates an initial draft version of a 'DATA.bib' metadata file.

process.bib reads and processes metadata entries.

icesTAF-package gives an overview of the package.

Examples

```
# Print in console
draft.software("icesTAF")

## Not run:
# Export to file
draft.software("icesTAF", file="bootstrap/SOFTWARE.bib")

## End(Not run)
```

---

file.encoding *File Encoding*

Description

Examine file encoding.

Usage

```
file.encoding(file)
```

Arguments

file	a filename.

Value

"latin1", "UTF-8", "unknown", or NA.

This function requires the file shell command. If the file utility is not found in the path, this function looks for it inside 'c:/Rtools/bin'. If the required software is not installed, this function returns NA.

Note

The encoding "unknown" indicates that the file is an ASCII text file or a binary file.

In TAF, text files that have non-ASCII characters must be encoded as UTF-8.

See Also

Encoding examines the encoding of a string.

latin1.to.utf8 converts files from latin1 to UTF-8 encoding.

line.endings examines line endings.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
file.base <- system.file(package="base", "DESCRIPTION")
file.nlme <- system.file(package="nlme", "DESCRIPTION")
file.encoding(file.base)  # ASCII
file.encoding(file.nlme)

## End(Not run)
```

---

flr2taf *Convert FLR Table to TAF Format*

Description

Convert a table from FLR format to TAF format.

Usage

```
flr2taf(x, colname = "Value")
```

Arguments

x	a table of class FLQuant.
colname	a column name to use if the FLR table contains only one row.

Value

A data frame in TAF format.

Note

FLR uses the FLQuant class to store tables as 6-dimensional arrays, while TAF tables are stored as data frames with a year column.

See Also

catage.taf describes the TAF format.

as.data.frame is a method provided by the **FLCore** package to convert FLQuant tables to a 7-column long format.

icesTAF-package gives an overview of the package.

Examples

```
x <- array(t(catage.xtab), dim=c(4,8,1,1,1,1))
dimnames(x) <- list(age=1:4, year=1963:1970,
                    unit="unique", season="all", area="unique", iter=1)
flr2taf(x)

x1 <- x[1,,,,,,drop=FALSE]
flr2taf(x1)
flr2taf(x1, "Juveniles")
```

---

latin1.to.utf8 *Convert File Encoding*

Description

Convert file encoding between "latin1" and "UTF-8".

Usage

```
latin1.to.utf8(file, force = FALSE)

utf8.to.latin1(file, force = FALSE)
```

Arguments

file	a filename.
force	whether to perform the conversion even if the current file encoding cannot be verified with file.encoding. Not recommended.

Note

In TAF, text files that have non-ASCII characters must be encoded as UTF-8.

See Also

iconv converts the encoding of a string.

file.encoding examines the encoding of a file.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
utf8.to.latin1("data.txt")
latin1.to.utf8("data.txt")

## End(Not run)
```

`lim` *Axis Limits*

Description

Compute axis limits. The lower limit is 0 and the upper limit is determined by the highest data value, times a multiplier.

Usage

```
lim(x, mult = 1.1)
```

Arguments

`x`	a vector of data values.
`mult`	a number to multiply with the highest data value.

Value

A vector of length two, which can be used as axis limits.

See Also

`icesTAF-package` gives an overview of the package.

Examples

```
plot(precip)
plot(precip, ylim=lim(precip))
plot(precip, ylim=lim(precip), yaxs="i")
```

line.endings *Line Endings*

Description

Examine whether file has Dos or Unix line endings.

Usage

```
line.endings(file)
```

Arguments

file	a filename.

Value

String indicating the line endings: "Dos" or "Unix".

See Also

file.encoding examines the encoding of a file.

dos2unix and unix2dos convert line endings.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
file <- system.file(package="icesTAF", "DESCRIPTION")
line.endings(file)

## End(Not run)
```

long2taf *Convert Long Table to TAF Format*

Description

Convert a table from long format to TAF format.

Usage

```
long2taf(x)
```

Arguments

x	a data frame in long format.

Value

A data frame in TAF format.

Note

TAF stores tables as data frames, usually with a year column as seen in stock assessment reports. The long format is more convenient for analysis and producing plots.

See Also

catage.long and catage.taf describe the long and TAF formats.

taf2long converts a TAF table to long format.

icesTAF-package gives an overview of the package.

Examples

```
long2taf(catage.long)
```

---

make *Run R Script If Needed*

Description

Run an R script if underlying files have changed, otherwise do nothing.

Usage

```
make(recipe, prereq, target, include = TRUE, engine = source,
  debug = FALSE, ...)
```

Arguments

recipe	script filename.
prereq	one or more underlying files, required by the script. For example, data files and/or scripts.
target	one or more output files, produced by the script. Directory names can also be used.
include	whether to automatically include the script itself as a prerequisite file.
engine	function to source the script.
debug	whether to show a diagnostic table of files and time last modified.
...	passed to engine.

Value

TRUE or FALSE, indicating whether the script was run.

Note

This function provides functionality similar to makefile rules, to determine whether a script should be (re)run or not.

If any target is missing or older than any prereq, then the script is run.

References

Stallman, R. M. *et al.* An introduction to makefiles. Chapter 2 in the *GNU Make manual*.

See Also

source runs any R script, sourceTAF is more convenient for running a TAF script, and sourceAll runs all TAF scripts.

make, makeTAF, and makeAll are similar to the source functions, except they avoid repeating tasks that have already been run.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
make("model.R", "data/input.dat", "model/results.dat")

## End(Not run)
```

---

makeAll *Run All TAF Scripts as Needed*

Description

Run core TAF scripts that have changed, or if previous steps were rerun.

Usage

```
makeAll(...)
```

Arguments

...	passed to makeTAF.

Value

Logical vector indicating which scripts were run.

Note

TAF scripts that will be run as needed: data.R, model.R, output.R, and report.R.

See Also

`source` runs any R script, `sourceTAF` is more convenient for running a TAF script, and `sourceAll` runs all TAF scripts.

`make`, `makeTAF`, and `makeAll` are similar to the `source` functions, except they avoid repeating tasks that have already been run.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:
makeAll()

## End(Not run)
```

---

makeTAF *Run TAF Script If Needed*

Description

Run a TAF script if the target directory is either older than the script, or older than the directory of the previous TAF step.

Usage

```
makeTAF(script, ...)
```

Arguments

`script`	TAF script filename.
`...`	passed to `make` and `sourceTAF`.

Value

`TRUE` or `FALSE`, indicating whether the script was run.

Note

Any underlying scripts are automatically included if they share the same filename prefix, followed by an underscore. For example, when determining whether a script data.R should be run, this function checks whether data_foo.R and data_bar.R have been recently modified.

See Also

source runs any R script, sourceTAF is more convenient for running a TAF script, and sourceAll runs all TAF scripts.

make, makeTAF, and makeAll are similar to the source functions, except they avoid repeating tasks that have already been run.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
makeTAF("model.R")

## End(Not run)
```

---

mkdir *Create Directory*

Description

Create directory, including parent directories if necessary, without generating a warning if the directory already exists.

Usage

```
mkdir(path)
```

Arguments

path	a directory name.

Value

TRUE for success, FALSE for failure, invisibly.

See Also

dir.create is the base function to create a new directory.

rmdir removes an empty directory.

clean can be used to remove non-empty directories.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
mkdir("emptydir")
rmdir("emptydir")

mkdir("outer/inner")
rmdir("outer", recursive=TRUE)

## End(Not run)
```

`msg` *Show Message*

Description

Show a message, as well as the current time.

Usage

```
msg(...)
```

Arguments

`...`	passed to `message`.

See Also

`message` is the base function to show messages, without the current time.

`sourceTAF` reports progress using `msg`.

`icesTAF-package` gives an overview of the package.

Examples

```
msg("script.R running...")
```

os *Operating System*

Description

Determine operating system name.

Usage

```
os()

os.linux()

os.macos()

os.windows()

os.unix()
```

Value

`os` returns the name of the operating system, typically `"Linux"`, `"Darwin"`, or `"Windows"`.

`os.linux`, `os.macos`, `os.unix`, and `os.windows` return `TRUE` or `FALSE`.

Note

The macOS operating system identifies itself as `"Darwin"`.

Both Linux and macOS are `os.unix`.

These shorthand functions can be useful when writing workaround solutions in platform-independent scripts.

See Also

`Sys.info` is the underlying function used to extract the operating system name.

`icesTAF-package` gives an overview of the package.

Examples

```
os()
os.linux()
os.macos()
os.unix()
os.windows()
```

period *Period*

Description

Paste two years to form a period string.

Usage

```
period(x, y = NULL)
```

Arguments

x	the first year, vector of years, matrix, or data frame.
y	the last year, if x is only the first year.

Details

If x is a vector or a data frame, then the lowest and highest years are used, and y is ignored.

If x is a matrix or data frame, this function looks for years in the first column. If the values of the first column do not look like years (four digits), then it looks for years in the row names.

Value

A string of the form "1990-2000".

Note

This function can be useful when working with draft.data.

See Also

paste is the underlying function to paste strings.

draft.data has an argument called period.

icesTAF-package gives an overview of the package.

Examples

```
period(1963, 1970)
period(c(1963, 1970))
period(1963:1970)

period(range(catage.taf$Year))
period(catage.taf$Year)
period(catage.taf)
period(catage.xtab)
```

plus *Rename Plus Group Column*

Description

Rename the last column in a data frame, by appending a "+" character. This is useful if the last column is a plus group.

Usage

```
plus(x)
```

Arguments

x	a data frame.

Value

A data frame similar to x, after renaming the last column.

Note

If the last column name already ends with a "+", the original data frame is returned without modifications.

See Also

names is the underlying function to rename columns.

icesTAF-package gives an overview of the package.

Examples

```
catage <- catage.taf

# Rename last column
catage <- plus(catage)

# Shorter and less error-prone than
names(catage)[names(catage)=="4"] <- "4+"
```

process.bib *Process Metadata*

Description

Read and process metadata entries in a *.bib file.

Usage

```
process.bib(bibfile, quiet = FALSE)
```

Arguments

bibfile	metadata file to process, either "DATA.bib" or "SOFTWARE.bib".
quiet	whether to suppress messages reporting progress.

Note

This is a helper function for taf.bootstrap. It is called within the 'bootstrap' directory that contains the metadata file.

A metadata file contains one or more entries that use a general BibTeX format:

```
@Type{key,
  field = {value},
  ...,
}
```

Consider, for example, the following metadata entry from a DATA.bib file:

```
@Misc{PLE7DFleet_2016.txt,
  originator = {WGNSSK},
  year       = {2016},
  title      = {Survey indices: UK_BTS, FR_GFS, IN_YFS},
  period     = {1987-2015},
  source     = {file},
}
```

Here, a data file is described using the @Misc entry type and the string following the entry type is called a *key*. The next fields state that this file was prepared by the North Sea working group in 2016 and it contains survey indices from 1987 to 2015. It is not necessary to specify the stock name, since that will be automatically recorded on the TAF server. The special value source = {file} means that the key, in this case PLE7DFleet_2016.txt, is the name of the file located inside bootstrap/initial/data. This file shorthand notation is equivalent to specifying the relative path: source = {initial/data/PLE7DFleet_2016.txt}.

The *source* field specifies where data or software originate from. There are four types of values that can be used in the source field:

1. GitHub reference of the form owner/repo[/subdir]@ref, identifying a specific version of an R package. A fixed reference such as a tag, release, or SHA-1 hash is recommended. Branch names, such as master, are pointers that are subject to change, and are therefore not reliable as long-term references.
2. URL starting with http or https, identifying a file to download.
3. Relative path starting with '`initial`', identifying the location of a file or directory provided by the user.
4. Special value `file`, indicating that the metadata key points to a file location.
5. Special value `script`, indicating that an R script should be run to fetch data files from a web service. The metadata key is used both to identify the script '`bootstrap/key.R`' and target directory '`bootstrap/data/key`'.

Another example metadata entry is from a SOFTWARE.bib file:

```
@Manual{FLAssess,
  author  = {Laurence T Kell},
  year    = {2018},
  title   = {{FLAssess}: Generic classes and methods for stock assessment
             models},
  version = {2.6.2, released 2018-07-18},
  source  = {flr/FLAssess@v2.6.2},
}
```

This entry describes a specific version of an R package that is required for the TAF analysis. It is similar, but not identical, to the output from the R command citation("FLAssess"). The version field specifies the version number and release date, with a corresponding GitHub reference. When an R package is not an official release but a development version, the version and source may look like this,

```
  version = {2.6.3, committed 2018-10-09},
  source  = {flr/FLAssess@f1e5acb},
```

or this:

```
  version = {0.5.4 components branch, committed 2018-03-12},
  source  = {fishfollower/SAM/stockassessment@25b3591},
```

For development versions like these, the version number itself may not be important or accurate, but the branch name and commit date may be informative. The 7-character SHA reference code is a pointer to the exact version of the package required for the analysis.

As a final metadata example, we look at a software entry that is not an R package. It is made available as a directory '`sole`' containing the model source code (sole.tpl) and executables for different platforms (sole, sole.exe). The model does not have an explicit version number, so the version field contains the year in which the model is used, along with the date when the source code was last modified:

```
@Article{sole,
  author  = {G. Aarts and J.J. Poos},
  year    = {2009},
  title   = {Comprehensive discard reconstruction and abundance estimation
             using flexible selectivity functions},
  journal = {ICES Journal of Marine Science},
  volume  = {66},
  pages   = {763-771},
  doi     = {10.1093/icesjms/fsp033},
  version = {2016, last modified 2016-04-27},
  source  = {initial/software/sole},
}
```

The source field can specify multiple URLs to download, separated by newlines. To shorten the source entries, the *prefix* field can be useful to specify a prefix that is common for all source entries.

The *dir* field is optional and creates a directory to place files in. If the dir field is used, it can only have the value dir = {TRUE} and the resulting directory will be named after the metadata key. The dir field is mainly useful when two or more data files that need to be downloaded have the same name. It is implied and not necessary to set dir = TRUE when source = {script} or when the source field specifies multiple URLs.

In summary, the metadata are similar to bibliographic entries, with the important addition of source directives that guide the bootstrap procedure to set up data files and software.

See Also

`taf.bootstrap` calls `process.bib` to process metadata.

`draft.data` and `draft.software` can be used to create initial draft versions of '`DATA.bib`' and '`SOFTWARE.bib`' metadata files.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:
process.bib("DATA.bib")
process.bib("SOFTWARE.bib")

## End(Not run)
```

`read.taf` *Read TAF Table from File*

Description

Read a TAF table from a file into a data frame.

Usage

```
read.taf(file, check.names = FALSE, stringsAsFactors = FALSE,
  fileEncoding = "UTF-8", ...)
```

Arguments

file	a filename.
check.names	whether to enforce regular column names, e.g. convert column name '"3"' to '"X3"'.
stringsAsFactors	whether to import strings as factors.
fileEncoding	character encoding of input file.
...	passed to read.csv.

Details

Alternatively, file can be a directory or a vector of filenames, to read many tables in one call.

Value

A data frame in TAF format, or a list of data frames if file is a directory or a vector of filenames.

See Also

read.csv is the underlying function used to read a table from a file.

write.taf writes a TAF table to a file.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
write.taf(catage.taf, "catage.csv")
catage <- read.taf("catage.csv")

write.taf(catage)
file.remove("catage.csv")

## End(Not run)
```

rmdir *Remove Empty Directory*

Description

Remove empty directory under any operating system.

Usage

```
rmdir(path, recursive = FALSE)
```

Arguments

path	a directory name.
recursive	whether to remove empty subdirectories as well.

Value

TRUE for success, FALSE for failure, invisibly.

Note

The base function `unlink(dir, recursive=FALSE)` does not remove empty directories in Windows and `unlink(dir, recursive=TRUE)` removes non-empty directories, making it unsuitable for tidying up empty ones.

See Also

unlink with `recursive = TRUE` removes non-empty directories.

mkdir creates a new directory.

clean can be used to remove non-empty directories.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
mkdir("emptydir")
rmdir("emptydir")

mkdir("outer/inner")
rmdir("outer", recursive=TRUE)

## End(Not run)
```

rnd *Round Columns*

Description

Round column values in a data frame.

Usage

```
rnd(x, cols, digits = 0, grep = FALSE, ...)
```

Arguments

x	a data frame.
cols	column names, or column indices.
digits	number of decimal places.
grep	whether cols is a regular expression.
...	passed to grep().

Value

A data frame similar to x, after rounding columns cols to the number of digits.

Note

Provides notation that is convenient for modifying many columns at once.

See Also

round is the underlying function used to round numbers.

grep is the underlying function used to match column names if grep is TRUE.

div is a similar function that divides columns with a common number.

icesTAF-package gives an overview of the package.

The **icesAdvice** package provides the icesRound function to round values for ICES advice sheets.

Examples

```
# With rnd() we no longer need to repeat the column names:

m <- mtcars
m[c("mpg","disp","qsec")] <- round(m[c("mpg","disp","qsec")])
m <- rnd(m, c("mpg","disp","qsec"))

# The x1/x2/x3/x4 approaches are equivalent:

x1 <- rnd(summary.taf, c("Rec","Rec_lo","Rec_hi",
```

```
                          "TSB","TSB_lo","TSB_hi",
                          "SSB","SSB_lo","SSB_hi",
                          "Removals","Removals_lo","Removals_hi"))
x1 <- rnd(x1, c("Fbar","Fbar_lo","Fbar_hi"), 3)

x2 <- rnd(summary.taf, "Rec|TSB|SSB|Removals", grep=TRUE)
x2 <- rnd(x2, "Fbar", 3, grep=TRUE)

x3 <- rnd(summary.taf, "Fbar", grep=TRUE, invert=TRUE)
x3 <- rnd(x3, "Fbar", 3, grep=TRUE)

# Less reliable in scripts if columns have been added/deleted/reordered:

x4 <- rnd(summary.taf, 2:13)
x4 <- rnd(x4, 14:16, 3)
```

`sam2taf` *Convert SAM Table to TAF Format*

Description

Convert a table from SAM format to TAF format.

Usage

```
sam2taf(x, colname = NULL, year = TRUE)
```

Arguments

`x`	a matrix containing columns Estimate, Low, and High.
`colname`	a descriptive column name for the output.
`year`	whether to include a year column.

Details

The default when `colname = NULL` is to try to infer a column name from the `x` argument. For example,

```
sam2taf(ssbtable(fit))
sam2taf(ssb)
sam2taf(SSB)
```

will recognize ssbtable calls and ssb object names, implicitly setting `colname = "SSB"` if the user does not pass an explicit value for `colname`.

Value

A data frame in TAF format.

Note

The **stockassessment** package provides accessor functions that return a matrix with columns Estimate, Low, and High, while TAF tables are stored as data frames with a year column.

See Also

summary.taf describes the TAF format.

catchtable, fbartable, rectable, ssbtable, and tsbtable (in the **stockassessment** package) return matrices with SAM estimates and confidence limits.

The summary method for sam objects produces a summary table with some key quantities of interest, containing duplicated column names (Low, High) and rounded values.

icesTAF-package gives an overview of the package.

Examples

```
## Example objects
x <- as.matrix(summary.taf[grep("SSB", names(summary.taf))])
rec <- as.matrix(summary.taf[grep("Rec", names(summary.taf))])
tsb <- as.matrix(summary.taf[grep("TSB", names(summary.taf))])
dimnames(x) <- list(summary.taf$Year, c("Estimate", "Low", "High"))
dimnames(rec) <- dimnames(tsb) <- dimnames(x)

## One SAM table, arbitrary object name
sam2taf(x)
sam2taf(x, "SSB")
sam2taf(x, "SSB", year=FALSE)

## Many SAM tables, recognized names
sam2taf(rec)
data.frame(sam2taf(rec), sam2taf(tsb, year=FALSE))

## Not run:

## Accessing tables from SAM fit object
data.frame(sam2taf(rectable(fit)), sam2taf(tsbtable(fit), year=FALSE))

## End(Not run)
```

---

sourceAll *Run All TAF Scripts*

Description

Run core TAF scripts in current directory.

Usage

```
sourceAll(...)
```

Arguments

`...`	passed to `sourceTAF`.

Value

Logical vector, indicating which scripts ran without errors.

Note

TAF scripts that will be run if they exist: data.R, model.R, output.R, and report.R.

See Also

`sourceTAF` runs a TAF script.

`makeAll` runs all TAF scripts as needed.

`clean` cleans TAF directories.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:
sourceAll()

## End(Not run)
```

---

sourceDir *Source Directory*

Description

Read all `*.R` files from a directory containing R functions.

Usage

```
sourceDir(dir, pattern = "\\.[r|R]$", all.files = FALSE,
  recursive = FALSE, quiet = TRUE, ...)
```

Arguments

dir	a directory containing R source files.
pattern	passed to dir when selecting files.
all.files	passed to dir when selecting files.
recursive	passed to dir when selecting files.
quiet	whether to suppress showing names of sourced files.
...	passed to source when sourcing files.

Details

The dir argument can also be a vector of filenames, instead of a directory name. This can be useful to specify certain files while avoiding others.

Value

Names of sourced files.

Note

This function is convenient in TAF analyses when many R utility functions are stored in a directory, see example below.

See Also

source is the base function to read R code from a file.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
sourceDir("bootstrap/software/utilities")

## End(Not run)
```

---

sourceTAF *Run TAF Script*

Description

Run a TAF script and return to the original directory.

Usage

```
sourceTAF(script, rm = FALSE, clean = TRUE, quiet = FALSE)
```

Arguments

script	script filename.
rm	whether to remove all objects from the global environment before and after the script is run.
clean	whether to clean the target directory before running the script.
quiet	whether to suppress messages reporting progress.

Details

The default value of rm = FALSE is to protect users from accidental loss of work, but the TAF server always runs with rm = TRUE to make sure that only files, not objects, are carried over between scripts.

Likewise, the TAF server runs with clean = TRUE to make sure that the script starts with a clean directory. The target directory of a TAF script has the same filename prefix as the script: data.R creates 'data' etc.

Value

TRUE or FALSE, indicating whether the script ran without errors.

Note

Commands within a script (such as setwd) may change the working directory, but sourceTAF guarantees that the working directory reported by getwd() is the same before and after running a script.

See Also

source is the base function to run R scripts.

makeTAF runs a TAF script if needed.

sourceAll runs all TAF scripts in a directory.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
write("print(pi)", "script.R")
source("script.R")
sourceTAF("script.R")
file.remove("script.R")

## End(Not run)
```

summary.taf *Summary Results in TAF Format*

Description

Small summary results table to describe a TAF format data frame to store values by year.

Usage

```
summary.taf
```

Format

Data frame containing 16 columns:

Year	year
Rec	recruitment, numbers at age 1 in this year (thousands)
Rec_lo	lower 95% confidence limit
Rec_hi	upper 95% confidence limit
TSB	total stock biomass (tonnes)
TSB_lo	lower 95% confidence limit
TSB_hi	upper 95% confidence limit
SSB	spawning stock biomass (tonnes)
SSB_lo	lower 95% confidence limit
SSB_hi	upper 95% confidence limit
Removals	total removals, including catches due to unaccounted mortality
Removals_lo	lower 95% confidence limit
Removals_hi	upper 95% confidence limit
Fbar	average fishing mortality (ages 2-4)
Fbar_lo	lower 95% confidence limit
Fbar_hi	upper 95% confidence limit

Details

The data are an excerpt (first years) from the summary results table for North Sea cod from the ICES (2016) assessment.

Source

ICES (2016) Report of the working group on the assessment of demersal stocks in the North Sea and Skagerrak (WGNSSK). *ICES CM 2016/ACOM:14*, p. 673.

See Also

div and rnd can modify a large number of columns.

icesTAF-package gives an overview of the package.

Examples

```
summary.taf
x <- div(summary.taf, "Rec|TSB|SSB|Removals", grep=TRUE)
x <- rnd(x, "Rec|TSB|SSB|Removals", grep=TRUE)
x <- rnd(x, "Fbar", 3, grep=TRUE)
```

---

taf.bootstrap *Bootstrap TAF Analysis*

Description

Set up data files and software required for the analysis. Model configuration files are also set up, if found.

Usage

```
taf.bootstrap(clean = TRUE, config = TRUE, data = TRUE,
  software = TRUE, quiet = FALSE)
```

Arguments

clean	whether to clean directories during the bootstrap procedure.
config	whether to process configuration files.
data	whether to process data.
software	whether to process software.
quiet	whether to suppress messages reporting progress.

Note

This function should be called from the top directory of a TAF analysis. It looks for a directory called 'bootstrap' and prepares data files and software according to metadata specifications.

The bootstrap procedure consists of the following steps:

1. If a directory bootstrap/initial/config contains model configuration files, they are copied to bootstrap/config.
2. If a bootstrap/DATA.bib metadata file exists, it is processed with process.bib.
3. If a bootstrap/SOFTWARE.bib metadata file exists, it is processed with process.bib.

After the bootstrap procedure, data and software have been documented and are ready to be used in the subsequent analysis. Specifically, the procedure populates up to four new directories:

- bootstrap/config with model configuration files.
- bootstrap/data with data files.
- bootstrap/library with R packages compiled for the local platform.
- bootstrap/software with software files, such as R packages in tar.gz source code format.

See Also

process.bib is a helper function used to process metadata.

taf.library loads a package from bootstrap/library.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
taf.bootstrap()

## End(Not run)
```

---

taf.colors *TAF Colors*

Description

Predefined colors that can be useful in TAF plots.

Usage

```
taf.green
taf.orange
taf.blue
taf.dark
taf.light
```

See Also

icesTAF-package gives an overview of the package.

Examples

```
taf.green

par(mfrow=c(3,1))

barplot(5:1, main="Five",
        col=c(taf.green, taf.orange, taf.blue, taf.dark, taf.light))

barplot(6:1, main="Six", col=c(taf.green, taf.orange, taf.blue,
                               taf.dark, taf.light, "white"))

barplot(7:1, main="Seven", col=c("black", taf.dark, taf.light,
                                 taf.green, taf.orange, taf.blue, "white"))
```

taf.library *TAF Library*

Description

Load package from local TAF library.

Usage

```
taf.library(package, messages = FALSE, warnings = FALSE)
```

Arguments

package	name of a package found in bootstrap/library.
messages	whether to show messages when package loads.
warnings	whether to show warnings when package loads.

Value

The names of packages currently installed in the TAF library.

Note

The purpose of the TAF library is to retain R packages that are not commonly used (and not on CRAN), to support long-term reproducibility of TAF analyses.

See Also

`library` is the underlying base function to load a package.

`taf.bootstrap` is the procedure to install packages into a local TAF library, via the 'SOFTWARE.bib' metadata file.

`icesTAF-package` gives an overview of the package.

Examples

```
## Not run:

# Show packages in TAF library
taf.library()

# Load packages
taf.library(this)
taf.library(that)

## End(Not run)
```

`taf.png` *PNG Device*

Description

Open PNG graphics device to export a plot into the TAF report folder.

Usage

```
taf.png(filename, width = 1600, height = 1200, pointsize = 32, ...)
```

Arguments

`filename`	plot filename.
`width`	image width.
`height`	image height.
`pointsize`	text size.
`...`	passed to png.

Details

The `filename` can be passed without the preceding `"report/"`, and without the `".png"` filename extension.

Specifically, the function prepends `"report/"` to the filename if (1) the filename does not contain a `"/"` separator, (2) the working directory is not report, and (3) the directory report exists. The function also appends ".png" to the filename if it does not already have that filename extension.

This automatic filename manipulation can be bypassed by using the png function directly.

Note

A simple convenience function to shorten

```
png("report/plot.png", width=1600, height=1200, pointsize=32)
```

to

```
taf.png("plot")
```

The `pointsize` argument only affects base plots. To change the text size of a lattice plot, the `zoom` function can be helpful.

For consistent image width and text size, it can be useful to keep the default `width = 1600` but vary the `height` to get the desired aspect ratio for each plot.

See Also

png is the underlying function used to open a PNG graphics device.

zoom changes text size in a lattice plot.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
taf.png("myplot")
plot(1)
dev.off()

library(lattice)
taf.png("mytrellis")
zoom(xyplot(1~1))
dev.off()

library(ggplot2)
taf.png("myggplot")
qplot(1, 1)
dev.off()

## End(Not run)
```

---

taf.skeleton *TAF Skeleton*

Description

Create initial directories and R scripts for a new TAF analysis.

Usage

```
taf.skeleton(path = ".", force = FALSE)
```

Arguments

path	where to create initial directories and R scripts. The default is the current working directory.
force	whether to overwrite existing scripts.

Value

Full path to analysis directory.

See Also

package.skeleton creates an empty template for a new R package.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
taf.skeleton()

## End(Not run)
```

---

taf.unzip *Unzip File*

Description

Extract files from a zip archive, retaining executable file permissions.

Usage

```
taf.unzip(zipfile, files = NULL, exdir = ".", unzip = NULL, ...)
```

Arguments

zipfile	zip archive filename.
files	files to extract, default is all files.
exdir	directory to extract to, will be created if necessary.
unzip	extraction method to use, see details below.
...	passed to unzip.

Details

The default method unzip = NULL uses the external unzip program in Unix-compatible operating systems, but an internal method in Windows. For additional information, see the unzip help page.

Note

One shortcoming of the base unzip function is that the default "internal" method resets file permissions, so Linux and macOS executables will return a 'Permission denied' error when run.

This function is identical to the base unzip function, except the default value unzip = NULL chooses an appropriate extraction method in all operating systems, making it useful when writing platform-independent scripts.

See Also

unzip is the base function to unzip files.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
exefile <- if(os.unix()) "run" else "run.exe"
taf.unzip("bootstrap/software/archive.zip", files=exefile, exdir="model")

## End(Not run)
```

---

taf2long *Convert TAF Table to Long Format*

Description

Convert a table from TAF format to long format.

Usage

```
taf2long(x, names = c("Year", "Age", "Value"))
```

Arguments

x	a data frame in TAF format.
names	a vector of three column names for the resulting data frame.

Value

A data frame with three columns.

Note

TAF stores tables as data frames, usually with a year column as seen in stock assessment reports. The long format is more convenient for analysis and producing plots.

See Also

catage.long and catage.taf describe the long and TAF formats.

long2taf converts a long table to TAF format.

icesTAF-package gives an overview of the package.

Examples

```
taf2long(catage.taf, names=c("Year","Age","Catch"))
```

taf2xtab *Convert TAF Table to Crosstab Format*

Description

Convert a table from TAF format to crosstab format.

Usage

```
taf2xtab(x)
```

Arguments

x	a data frame in TAF format.

Value

A data frame with years as row names.

Note

TAF stores tables as data frames, usually with a year column as seen in stock assessment reports. The crosstab format can be more convenient for analysis and producing plots.

See Also

catage.taf and catage.xtab describe the TAF and crosstab formats.

tt converts a TAF table to transposed crosstab format.

xtab2taf converts a crosstab table to TAF format.

icesTAF-package gives an overview of the package.

Examples

```
taf2xtab(catage.taf)
```

`tt` *TAF Transpose*

Description

Convert a table from TAF format to transposed crosstab format.

Usage

```
tt(x, column = FALSE)
```

Arguments

`x`	a data frame in TAF format.
`column`	a logical indicating whether the group names should be stored in a column called 'Age' instead of in row names. Alternatively, `column` can be a string supplying another name for that first column.

Value

A data frame with years as column names.

Note

Transposing can be useful when comparing TAF tables to stock assessment reports.

See Also

`t` transposes a matrix.

`catage.taf` describes the TAF format.

`taf2xtab` converts a TAF table to crosstab format, without transposing.

`icesTAF-package` gives an overview of the package.

Examples

```
taf2xtab(catage.taf)
tt(catage.taf)
tt(catage.taf, TRUE)
tt(catage.taf, "Custom")
```

`write.taf` *Write TAF Table to File*

Description

Write a TAF table to a file.

Usage

```
write.taf(x, file = NULL, dir = NULL, quote = FALSE,
  row.names = FALSE, fileEncoding = "UTF-8", underscore = TRUE, ...)
```

Arguments

`x`	a data frame in TAF format.
`file`	a filename.
`dir`	an optional directory name.
`quote`	whether to quote strings.
`row.names`	whether to include row names.
`fileEncoding`	character encoding for output file.
`underscore`	whether automatically generated filenames (when `file = NULL`) should use underscore separators instead of dots.
`...`	passed to `write.csv`.

Details

Alternatively, `x` can be a list of data frames or a string vector of object names, to write many tables in one call. The resulting files are named automatically, similar to `file = NULL`.

The default value `file = NULL` uses the name of `x` as a filename, so a data frame called `survey.uk` will be written to a file called 'survey_uk.csv' (when `underscore = TRUE`) or 'survey.uk.csv' (when `underscore = FALSE`).

The special value `file = ""` prints the data frame in the console, similar to `write.csv`.

Note

The resulting CSV file has Dos line endings, as specified in the RFC 4180 standard (IETF 2005).

This function gives a warning when column names are duplicated, unless the target directory name is report.

References

IETF (2005) Common format and Mime type for Comma-Separated Values (CSV) files. *IETF RFC 4180*.

See Also

write.csv is the underlying function used to write a table to a file.

read.taf reads a TAF table from a file into a data frame.

icesTAF-package gives an overview of the package.

Examples

```
## Not run:
write.taf(catage.taf, "catage.csv")
catage <- read.taf("catage.csv")

write.taf(catage)
file.remove("catage.csv")

## End(Not run)
```

---

xtab2taf *Convert Crosstab Table to TAF Format*

Description

Convert a table from crosstab format to TAF format.

Usage

```
xtab2taf(x, colname = "Year")
```

Arguments

x	a data frame in crosstab format.
colname	name for first column.

Value

A data frame in TAF format.

Note

TAF stores tables as data frames, usually with a year column as seen in stock assessment reports. The crosstab format can be more convenient for analysis and producing plots.

See Also

catage.taf and catage.xtab describe the TAF and crosstab formats.

taf2xtab converts a TAF table to crosstab format.

icesTAF-package gives an overview of the package.

Examples

```
xtab2taf(catage.xtab)
```

---

zoom *Zoom*

Description

Change text size in a lattice plot.

Usage

```
zoom(x, ...)

## S3 method for class 'trellis'
zoom(x, size = 2.7, main = 1.2 * size, lab = size,
  axis = size, strip = size, sub = 0.9 * size, legend = 0.9 * size,
  splom = 0.9 * size, ...)
```

Arguments

x	a lattice plot of class "trellis".
...	further arguments, currently ignored.
size	text size multiplier.
main	size of main title (default is 1.2 * size).
lab	size of axis labels (default is size).
axis	size of tick labels (default is size).
strip	size of strip labels (default is size).
sub	size of subtitle (default is 0.9 * size).
legend	size of legend labels (default is 0.9 * size).
splom	size of scatterplot matrix diagonal labels (default is 0.9 * size).

Details

Pass NULL for any argument to avoid changing the size of that text component.

The legend component of a lattice plot can be somewhat fickle, as the object structure varies between plots. One solution is to pass legend = NULL and tweak the legend before or after calling the zoom function.

Value

The same lattice object, but with altered text size.

Note

The default values result in lattice plots that have similar text size as base plots, when using `taf.png`.

This function ends with a `print` call, to make it easy to export the lattice plot to a file, without the need of an explicit `print`.

See Also

`Lattice` plots are created using `xyplot` or related functions.

`taf.png` opens a PNG graphics device.

`icesTAF-package` gives an overview of the package.

Examples

```
library(lattice)

xyplot(1~1)
zoom(xyplot(1~1))
zoom(xyplot(1~1), size=1, axis=0.8)
zoom(xyplot(1~1), lab=NULL, axis=NULL)

## Not run:
taf.png("myplot")
plot(1)
dev.off()

taf.png("mytrellis")
zoom(xyplot(1~1))
dev.off()

## End(Not run)
```

Index