

Package ‘favir’

February 14, 2012

Type Package

Title Formatted Actuarial Vignettes in R

Version 0.5-1

Date 2011-01-03

Author Benedict Escoto

Maintainer Benedict Escoto <favir@emerose.org>

Depends R (>= 2.7.0), ChainLadder, ggplot2, plyr, reshape, tools

Description FAViR lowers the learning curve of the R environment. It is a series of peer-revied Sweave papers that use a consistent style.

License GPL (>= 2)

LazyLoad no

URL <http://www.favir.net>

Repository CRAN

Date/Publication 2011-01-04 07:28:28

R topics documented:

Assert	2
AssertSingle	3
Caption	4
DefaultFormatter	4
favir.colors	5
FavirDF	6
FavirLoadModule	7
FavirSweave	8
FieldFormatters	9
FieldGroup	10

FieldHeadings	11
FieldSettings	12
formatters	13
GroupFormatter	13
GroupHeading	14
IncludeGraph	15
IncludeGrid	16
IncludeLegal	17
IncludePrelude	18
InitPaper	18
Label	19
Last	20
LookupDF	20
MakeFormatter	21
OnError	22
Orientation	23
PreLatexFDF	23
RBindPreLatex	24
RowColors	25
RowIndicies	26
SummaryRow	26
TextSize	27
%SplitOn%	28

Index 29

Assert	<i>stopifnot() with an error message</i>
--------	--

Description

This is a simple wrapper around `stopifnot`. The main difference is that `Assert` takes an error message that is useful for explaining why the error occurred.

Usage

```
Assert(boolean, err.msg=NULL)
```

Arguments

boolean	Argument to test
err.msg	If this is specified, it will be given as the error message

Details

If `boolean` is `TRUE`, then nothing will happen. Otherwise an error will be raised and some diagnostic information will be printed to standard error.

R has a tendency towards invisible and/or inscrutable errors. `stopifnot` or this `Assert` function, can be used frequently to improve code reliability.

Examples

```
## Not run:
Assert("hello", "Error: hello is not TRUE")

## End(Not run)
Assert(1==1, "No error: TRUE is TRUE")
```

AssertSingle	<i>Raise error if arguments don't have length 1</i>
--------------	---

Description

This is a simple convenience function. Many errors in R are caused when a vector of length > 1 appears when an item of length 1 is expected. This function tests each argument and aborts with an error if any of them have length ≥ 1 .

Usage

```
AssertSingle(..., err.msg=NULL)
```

Arguments

...	Arguments to test
err.msg	If this is specified, it will be given as the error message

Details

AssertSingle prints some helpful information to stderr when it raises an error. Don't use AssertSingle if this behavior isn't intended.

Examples

```
x <- 3
y <- "hello"
z <- 1:5

AssertSingle(x, y)
## Not run:
AssertSingle(x, y, z)

## End(Not run)
```

Caption	<i>Query or set the caption in a favir.data.frame</i>
---------	---

Description

Query or set the latex caption of a favir.data.frame object

Usage

```
Caption(fdf)
Caption(fdf) <- value
```

Arguments

fdf	A favir.data.frame object
value	The caption as a string

Details

Latex will render the caption as text below the table.

See Also

[FavirDF](#), [Label](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
Caption(fdf)
Caption(fdf) <- "Example Table"
fdf
```

DefaultFormatter	<i>Get and set the default formatting operation</i>
------------------	---

Description

Get or set the default formatting operation for a favir.data.frame object

Usage

```
DefaultFormatter(fdf)
DefaultFormatter(fdf) <- value
SetGlobalDefaultFormatter(formatter)
```

Arguments

fdf	A favir.data.frame object
value	A function to set as the default formatter
formatter	A function to set as the global default formatter

Details

When an object of class `favir.data.frame` is converted to latex, the data, which probably has type numeric, must be converted from the internal R representation to a string. The system must decide how many decimal places to show, whether a comma should separate hundreds from thousands, etc. It does this using formatter functions, which are functions of a single argument that return the string representation of the argument.

Assuming no field-level formatting is set, the default formatter is applied to each entry and the result is passed on to latex. A field's `FieldFormatter` will override the default formatter for that particular field.

`SetGlobalDefaultFormatter` sets the global default formatter that will apply to all tables. Using this may break the formatting of other's code which assumes the default formatter.

See Also

[FieldFormatters](#). [FavirDF](#). [MakeFormatter](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
print(fdf)
DefaultFormatter(fdf) <- MakeFormatter(digits=5)
print(fdf)
```

<code>favir.colors</code>	<i>Named vector of default colors</i>
---------------------------	---------------------------------------

Description

`favir.colors` is a vector that has the default graphics colors in web-hexidecimal format (e.g. `#D3BDA8`).

Usage

```
favir.colors
```

Details

The FAViR color scheme is based on three color hues, each with five shade variants, for 15 basic colors total. The main group is abbreviated “M”, the two accent groups are “A” and “B”. Within each group the colors are numbered 1 to 5 in order of decreasing brightness.

By default, “M” is brown, “A” is green, and “B” is blue.

Graphics functions should use colors from `favir.colors` so other users can more easily produce color-consistent graphics.

See Also

[RowColors](#).

Examples

```
favir.colors["M1"] # Return main color, lightest shade
```

FavirDF

Return data frame formatted for latex

Description

Return `favir.data.frame` object, a data frame with extra latex formatting

Usage

```
FavirDF(df, field.formatters=list(),
        field.headings=list(), default.formatter=NULL,
        row.colors=NULL, orientation="normal",
        caption="", label=NULL, text.size="normalsize")
```

Arguments

<code>df</code>	The data frame object to format
<code>field.formatters</code>	See FieldFormatters
<code>field.headings</code>	See FieldHeadings
<code>default.formatter</code>	See DefaultFormatter
<code>row.colors</code>	See RowColors
<code>orientation</code>	See Orientation
<code>caption</code>	See Caption
<code>label</code>	See Label
<code>text.size</code>	See TextSize

Details

This function accepts a normal data frame object as its main input and returns a `favir.data.frame` object, which is a normal data frame with attributes indicating how it should be displayed in latex.

Because a `favir.data.frame` still has class `data.frame`, it can be used wherever normal data frames can be used.

The other arguments control how the table should be formatted. For instance `caption` sets the latex caption displayed below the table. For more information, see the `get/set` functions associated with each argument.

Examples

```
df <- data.frame(a=1:5, b=6:10)
df <- FavirDF(df, caption="Example Table")
print(df)
print(as.data.frame(df))
```

FavirLoadModule	<i>Create and use favir environments</i>
-----------------	--

Description

Create a new environment, source the given file in it, and return the environment.

Usage

```
FavirLoadModule(filename)
```

Arguments

`filename` The filename of an R source file.

Details

If every FAViR paper used the same namespace, the potential for conflicts would rise. `FavirLoadModule` is a simple function that allows a source code file to be loaded in its own environment, where conflicts are not possible. Objects from the environment can be accessed using the `$` operator.

Examples

```
# Create a new file with the single statement "a <- 5" in it
filename <- "example-file.R"

write("a <- 5", filename)

# Load it into the environment called favir.env, and read a back
FavirEnv <- FavirLoadModule(filename)
FavirEnv$a
```

```
with(FavirEnv, ls())

# tidy up
unlink(filename)
```

FavirSweave

Sweave and LaTeX a FAViR paper

Description

This function Sweaves and then LaTeXs each paper. It is a simple wrapper around the Sweave and texi2dvi functions.

Usage

```
FavirSweave(filenamees, change.dir=TRUE, clean=TRUE)
```

Arguments

filenamees	Vector of character filenames of Rnw files
change.dir	Whether to put output files in same directory
clean	TRUE to clean-up auxiliary latex files

Details

FavirSweave is a simple convenience function which runs Sweave and then texi2dvi on each of the filenames given.

If `change.dir` is TRUE, the working directory will be changed before processing each file to the directory that file is in. Afterwards the working directory will be restored. If `change.dir` is FALSE, the working directory will not be changed.

The argument `clean` is passed to `texi2dvi`. If TRUE, latex auxiliary files like `.log` and `.aux` will be removed. This may not work on some platforms.

See Also

[Sweave](#), [texi2dvi](#).

Examples

```
## Not run:
FavirSweave("example-paper")
FavirSweave(c("path1/first.paper", "path2/second.paper"))

## End(Not run)
```

FieldFormatters	<i>Get and set formatters for particular fields</i>
-----------------	---

Description

Get or set the field-level formatting operations for a `favir.data.frame` object

Usage

```
FieldFormatters(fdf)
FieldFormatters(fdf) <- value
```

Arguments

<code>fdf</code>	A <code>favir.data.frame</code> object
<code>value</code>	List of field formatters to use

Details

When an object of class `favir.data.frame` is converted to latex, the data, which probably has type numeric, must be converted from the internal R representation to a string. The system must decide how many decimal places to show, whether a comma should separate hundreds from thousands, etc. It does this using formatter functions, which are functions of a single argument that return the string representation of the argument.

Each column/field in the data frame can have a field formatter set, which controls the formatting for that particular field. The argument `value` to `FieldFormatter<-` should be a list whose keys are the field names in the data frame, and whose values are formatter functions.

See Also

[formatters](#), [DefaultFormatter](#), [FavirDF](#), [MakeFormatter](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
print(fdf)
FieldFormatters(fdf) <- list(a=MakeFormatter(), b=MakeFormatter(digits=3))
print(fdf)
```

FieldGroup	<i>Query and declare a group of fields</i>
------------	--

Description

Query and declare a group of fields of a favir.data.frame object

Usage

```
FieldGroup(fdf, group.name, as.names=TRUE)
FieldGroup(fdf, group.name) <- value
ListFieldGroups(fdf)
ParentFieldGroups(fdf, field)
```

Arguments

fdf	A favir.data.frame object
group.name	The name of a field group
as.names	TRUE to return field names, FALSE to return field indices
value	A vector either of field names or indices
field	The name of a single field

Details

If several fields (i.e. columns) in a favir.data.frame object hold similar items, they may share formatting. For instance, perhaps three fields all contain premium, so all should be comma-separated. These functions allow those fields to be grouped together, so formatting may be specified for all of them at once.

A field group is just a set of fields. Using the these functions don't affect formatting immediately; they just group the field together so formatting may be specified later on the group.

FieldGroup gets and sets a group, specifying which fields are in that group.

ListFieldGroups returns a vector of all existing field group names.

ParentFieldGroups returns all the field group names that contain the given field. As a field may be a member of more than one group, the results of this function may have length > 1.

See Also

[FavirDF.GroupFormatter](#). [GroupHeading](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
fdf
FieldGroup(fdf, "all") <- c("a", "b")
FieldGroup(fdf, "all")

FieldGroup(fdf, "only a") <- 1
ListFieldGroups(fdf)
ParentFieldGroups(fdf, "a")
ParentFieldGroups(fdf, "b")
```

FieldHeadings*Get and set printable field names*

Description

Get or set the printable field names for a `favir.data.frame` object

Usage

```
FieldHeadings(fdf)
FieldHeadings(fdf) <- value
```

Arguments

<code>fdf</code>	A <code>favir.data.frame</code> object
<code>value</code>	List field headings to use

Details

For convenience, data frame field names may be short strings like `olep` instead of descriptive titles like "On-Level Earned Premium". This function allows for the more descriptive titles to be printed.

See Also

[FieldSettings](#), [GroupHeading](#), [FavirDF](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
print(fdf)
FieldHeadings(fdf) <- list(a="Policies In-Force", b="Earned Premium")
FieldHeadings(fdf)
print(fdf)
```

FieldSettings*Get or set attributes for a particular field*

Description

Get or set general formatting attributes for a particular field in a `favir.data.frame` object

Usage

```
FieldSettings(fdf, field)
FieldSettings(fdf, field) <- value
```

Arguments

<code>fdf</code>	A <code>favir.data.frame</code> object
<code>field</code>	The name of a field/column
<code>value</code>	A list of attributes to set (set below)

Details

The heading and formatting for a `favir.data.frame` field can be controlled using the `FieldHeadings` and `FieldFormatters` functions. However, it may be more convenient to specify the heading and/or formatting for a particular field instead of specifying all of them at once.

The `FieldSettings` function allows this. It accepts or returns a list with two possible components:

heading The descriptive title of the field

formatter The formatting function to use on the field's data

See Also

[FieldHeadings](#), [FieldFormatters](#), [FavirDF](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
FieldSettings(fdf, "b")
FieldSettings(fdf, "b") <- list(heading="Example Header", formatter=formatters$percent)
fdf
```

 formatters

Several common formatting functions pre-packaged

Description

formatters is a list of common formatting functions that can be used with [FieldFormatters](#) or [FieldSettings](#).

Usage

```
formatters
```

Details

Each component of the formatters list is a function that formats numbers into strings. The basic prefixes are 'comma', 'flat', 'space', and 'percent'. Each can then have an option digit between 0 and 3 indicating how many decimal places to show. See the examples below.

See Also

[FieldSettings](#), [FieldFormatters](#), [FavirDF](#).

Examples

```
formatters$percent(0.12345)
formatters$percent3(0.12345)
formatters$flat(123456)
formatters$comma(123456)
formatters$space3(123456)
```

 GroupFormatter

Query or set formatter for a group of fields

Description

Query or set formatter for a group of fields in a favir.data.frame object

Usage

```
GroupFormatter(fdf, group.name)
GroupFormatter(fdf, group.name) <- value
GetGlobalGroupFormatter(group.name)
SetGlobalGroupFormatter(group.name, formatter)
```

Arguments

fdf	A favir.data.frame object
group.name	The name of a field group
value	The formatting function to use on that field group
formatter	The global formatting function for that group

Details

See [FieldGroup](#) for background on groups of fields, and [FieldFormatters](#) for formatting functions.

GroupFormatter sets the formatting function for the given group of fields.

This is similar to setting the FieldFormatter for each field in the group. However, there is a hierarchical relationship—if an individual field’s formatter is set, it will override the group formatting.

Global field groups are field groups that span more than one data frame. For instance, a paper may contain many tables that have various types of premium as one of the fields. If a "premium" group were declared, it could set formatting for all those fields at the same time.

GetGlobalGroupFormatter and SetGlobalGroupFormatter query and establish global field groups.

See Also

[FavirDF](#), [FieldGroup](#), [FieldFormatters](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10, c=11:15))
FieldGroup(fdf, "first2") <- 1:2
GroupFormatter(fdf, "first2")
GroupFormatter(fdf, "first2") <- formatters$percent
fdf

SetGlobalGroupFormatter("last1", formatters$comma2)
GetGlobalGroupFormatter("last1")
FieldGroup(fdf, "last1") <- "c"
fdf
```

GroupHeading

Query or set the heading for a field group

Description

Query or set the descriptive heading for a group of fields in a favir.data.frame object

Usage

```
GroupHeading(fdf, group.name)
GroupHeading(fdf, group.name) <- value
```

Arguments

fdf	A favir.data.frame object
group.name	The name of a field group
value	The descriptive heading to set

Details

See [FieldGroup](#) for background on groups of fields, and [FieldHeadings](#) for descriptive field headings.

In latex form, group headings will be centered above the field group and separated by a horizontal line. Fields must be contiguous in order to have a heading set.

See Also

[FavirDF.FieldGroup.FieldHeadings](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10, c=11:15))
FieldGroup(fdf, "first2") <- 1:2
GroupHeading(fdf, "first2") <- "The first two fields"
fdf
```

IncludeGraph	<i>Include a ggplot graph in a Sweave paper</i>
--------------	---

Description

IncludeGraph is a convenient way to place a ggplot graph inside a FAViR latex paper produced with Sweave. It handles several of the details such as size and caption.

Usage

```
IncludeGraph(graph, caption="", label=NULL, width=12.5, height=12.5, filename=NULL)
```

Arguments

graph	The ggplot graph to include
caption	The Latex caption to display under the graph
label	The Latex label that can be used to refer to the graph
width	The width of the graph in centimeters
height	The height of the graph in centimeters
filename	The filename to use for the graph, or NULL

Details

If no filename is provided, IncludeGraph will try to pick a sensible one.

By default graphs are stored in the ‘favir-graphs’ subdirectory of the current directory, so this directory must be writable.

See Also

[IncludeGrid](#), [FavirDF](#).

Examples

```
## Not run:
example.graph <- qplot(1:10, (1:10)^2)
IncludeGraph(example.graph, caption="Example Graph", label="exgraph")

## End(Not run)
```

IncludeGrid

Include a grid of ggplot graphs in a Sweave paper

Description

IncludeGrid is a convenient way to place several ggplot graphs in a uniform grid inside a FAViR latex paper produced with Sweave. It organizes the viewports automatically.

Usage

```
IncludeGrid(graph.list, ...)
```

Arguments

<code>graph.list</code>	The graph specification (see below)
<code>...</code>	caption, label, width, or height (see below)

Details

`graph.list` must be a list of ggplot graphs, where each name is of the form ‘X.Y’ where ‘X’ is the row number and ‘Y’ is the column number. The total dimensions of the grid will be determined from this list. Each graph in the grid will have the same dimensions (height and width).

IncludeGrid takes the same caption, label, width, and height arguments as IncludeGraph.

See Also

[IncludeGraph](#), [FavirDF](#).

Examples

```
## Not run:
example.graph1 <- qplot(1:10, 1:10)
example.graph1 <- qplot(1:10, (1:10)^2)
# Produce two plots side by side
IncludeGraph(list("1.1"=example.graph1, "1.2"=example.graph2), caption="Example Graph", label="exgraph")

## End(Not run)
```

IncludeLegal

Print default legal disclaimer and license info

Description

This function can be called near the end of FAViR paper, written using Sweave. It specifies default licensing for the paper.

Usage

```
IncludeLegal(author, year)
```

Arguments

author	The author(s) of the paper
year	The year(s) to claim copyright for

Details

The license specified is the GNU GPL v2 or later for the source code and the GNU all-permissive license for the resulting paper. Distributing your paper with the output of IncludeLegal licenses the paper under these terms.

See Also

[InitPaper](#), [IncludePrelude](#).

Examples

```
IncludeLegal("John Doe and Jill Smith", 2008)
```

IncludePrelude	<i>Print default latex prelude</i>
----------------	------------------------------------

Description

This function is typically called at the beginning of a FAViR paper, written using Sweave. It writes the default Latex introductory code.

Usage

```
IncludePrelude(title, author, subtitle="", header.lines="", include.logo=TRUE)
```

Arguments

title	The paper's title
author	The author of the paper
subtitle	Paper subtitle, or the empty string if none
header.lines	Additional string to include the LaTeX header
include.logo	Whether or not to include the "FaViR" logo

Details

The latex code printed by this function loads the necessary latex packages as well as defining the default theme. If you need to add your own custom header lines, use the `header.lines` argument.

See Also

[IncludeLegal](#), [InitPaper](#).

Examples

```
IncludePrelude("Paper title", "Random Actuary", "The shocking truth
about <actuarial topic>")
```

InitPaper	<i>Initialize FAViR paper settings</i>
-----------	--

Description

This function is typically called at the beginning of a FAViR paper, written using Sweave. It sets various global or environment settings.

Usage

```
InitPaper()
```

Details

Currently this just modifies error handling and makes sure the graph directory exists.

See Also

[IncludePrelude](#), [FavirLoadModule](#).

Examples

```
## Not run:  
InitPaper()  
  
## End(Not run)
```

Label

Query or set the latex label of a favir.data.frame

Description

Query or set the latex reference label in a favir.data.frame object

Usage

```
Label(fdf)  
Label(fdf) <- value
```

Arguments

fdf	A favir.data.frame object
value	The label as a string

Details

In latex, the label can be used to refer to a figure or table using the ref command, as in `\ref{label}`.

See Also

[FavirDF](#), [Caption](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))  
Label(fdf)  
Label(fdf) <- "latex label"  
fdf
```

Last	<i>Return last n elements of a vector or data frame</i>
------	---

Description

This function is similar to `tail` but does more error checking. In particular, it is an error if the argument is too small.

Usage

```
Last(x, n=1, allow.fewer=FALSE)
```

Arguments

<code>x</code>	A vector or data frame
<code>n</code>	The number of elements to return
<code>allow.fewer</code>	Boolean. If FALSE, raise an error if <code>x</code> has length < <code>n</code> .

See Also

[tail](#).

Examples

```
x <- 1:2
Last(x)
## Not run: Last(x, 3)
tail(x)
```

LookupDF	<i>Select a single row from a data frame</i>
----------	--

Description

With extensive error checking, return a single row from a data frame.

Usage

```
LookupDF(df, ..., allow.NULL=FALSE)
```

Arguments

<code>df</code>	The data frame to look in
<code>...</code>	The fields to look in
<code>allow.NULL</code>	Whether to allow empty searches

Details

This is a safe way to look for a row in a data frame. Usually when querying a data frame, too many or not enough results will be returned or perhaps a field name will be misspelled. With `LookupDF`, that will trigger an error.

Examples

```
df <- data.frame(a=1:5, b=11:15, c=21:25)
result <- LookupDF(df, a=3, b=13)
result

df2 <- data.frame(a=1:5, b=rep(0, 5))
## Not run:
LookupDF(df2, b=0)
LookupDF(df2, a=-1)

## End(Not run)
stopifnot(identical(NULL, LookupDF(df2, a=-1, allow.NULL=TRUE)))
```

MakeFormatter

Create and return a formatting function

Description

Create and return a function to format numbers

Usage

```
MakeFormatter(digits=0, format="f", big.mark=",", math.mode=TRUE, ...)
```

Arguments

<code>digits</code>	Number of displayed digits to right of decimal point
<code>format</code>	Normal vs scientific notation
<code>big.mark</code>	Separator between groups of digits
<code>math.mode</code>	Whether latex math mode is enabled
<code>...</code>	Other arguments to pass through to <code>formatC</code>

Details

MakeFormatter returns a function which accepts a numeric type and returns a string representation of that number. It mostly a wrapper around `formatC` which has sensible defaults for actuarial work. The arguments are summarized briefly below, but see the documentation for `formatC` for more information.

The `format` argument defaults to "f" for traditional (xxx.yyy) notation. "e" is for scientific notation (n.de+n).

`big.mark` separates thousands, millions, etc. Set it to "" for no separator.

If `math.mode` is TRUE, the entry will be displayed in latex math mode. This should probably be set to false for strings.

See Also

[FieldSettings](#), [FieldFormatters](#), [FavirDF](#).

Examples

```
x <- 12345.6789
MakeFormatter()(x)
MakeFormatter(digits=3)(x)
MakeFormatter(digits=1, big.mark="")(x)
```

OnError

Print traceback and dump debugging frames to a file

Description

This may be an appropriate debugging function for non-interactive use.

Usage

```
OnError()
```

Details

Frames will be dumped into a file called "last.dump.rda" in the current directory. This can be read with the `debugger()` function. For interactive use, `options(error=recover)` may be more appropriate.

Examples

```
## Not run:
options(error=OnError)
aoestnuh # Cause an error
debugger(load("last.dump.rda"))

## End(Not run)
```

Orientation	<i>Query or set the orientation of a favir.data.frame</i>
-------------	---

Description

Control whether a favir.data.frame object is displayed in picture or landscape orientation.

Usage

```
Orientation(fdf)
Orientation(fdf) <- value
```

Arguments

fdf	A favir.data.frame object
value	Either “normal” or “sideways”

Details

If a table is too wide to fit on a page, Orientation can be used to turn it sideways. [TextSize](#) may also help here.

See Also

[FavirDF](#), [TextSize](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10, c=11:15))
Orientation(fdf)
Orientation(fdf) <- "sideways"
print(fdf)
```

PreLatexFDF	<i>An object a favir data frame is converted into before printing</i>
-------------	---

Description

When a favir data frame is printed, it is first converted into an object of type favir.prelatex. This intermediate stage is returned by the PreLatexFDF function.

Usage

```
PreLatexFDF(favir.df)
print.favir.prelatex(x, ...)
```

Arguments

<code>favir.df</code>	An object of type <code>favir.data.frame</code>
<code>x</code>	An object of type <code>favir.prelatex</code>
<code>...</code>	Additional arguments supplied to print

Details

A prelatex object still keeps track of which items are in which row and column, but it is otherwise much closer to LaTeX than a `favir.data.frame`. In particular, all data is stored as strings (the type of each column has been lost).

See Also

[FavirDF](#), [RBindPreLatex](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
pl.fdf <- PreLatexFDF(fdf)
print(pl.fdf)
```

RBindPreLatex

Join several favir.prelatex objects into one

Description

When a `favir` data frame is printed, it is first converted into an object of type `favir.prelatex`. This intermediate stage is returned by the `PreLatexFDF` function.

Usage

```
RBindPreLatex(..., include.headings=TRUE)
```

Arguments

<code>...</code>	Two or more <code>favir.prelatex</code> objects
<code>include.headings</code>	Whether to keep the headings of the prelatex objects (other than the first, which is always kept)

Details

`RBindPreLatex` may be useful for joining two tables horizontally even when they don't hold comparable information.

See Also

[PreLatexFDF](#).

Examples

```

fdf1 <- FavirDF(data.frame(a=1:5, b=6:10))
pl.fdf1 <- PreLatexFDF(fdf1)

fdf2 <- FavirDF(data.frame(c=11:15, d=16:20))
pl.fdf2 <- PreLatexFDF(fdf2)

print(RBindPreLatex(pl.fdf1, pl.fdf2))

```

RowColors

Get or set row coloring

Description

Get or set the colors of each row in a `favir.data.frame` object

Usage

```

RowColors(fdf)
RowColors(fdf) <- value
DefaultRowColoring(df, primary="white", secondary="colorM1")

```

Arguments

<code>fdf</code>	A <code>favir.data.frame</code> object
<code>value</code>	Character vector of row colors to use
<code>df</code>	A data frame
<code>primary</code>	The primary row color
<code>secondary</code>	The secondary row color

Details

By default, `favir.data.frame` objects alternate colors for groups of rows to add readability. The exact coloring of each row can be queried and set with `RowColors`.

`DefaultRowColoring` returns the default vector of colors to use for a given data frame.

See Also

[FavirDF](#).

Examples

```

df <- data.frame(a=1:10, b=11:20)
DefaultRowColoring(df)
fdf <- FavirDF(df)
RowColors(fdf)
RowColors(fdf) <- rep("white", nrow(fdf))
RowColors(fdf)

```

RowIndicies*Enumerate the row indicies of a data frame or array*

Description

This is a simple convenience function. For instance, if a data frame has 3 rows, RowIndicies returns 1:3.

Usage

```
RowIndicies(x)
```

Arguments

x A data frame or matrix

Details

This may be useful for iterating over a data frame. It can replace 1:nrow(x), which can lead to buggy code, because the data frame may be empty. Correct is seq(length.out=nrow(x)) but this is more typing.

See Also

[seq](#),

Examples

```
RowIndicies(data.frame(a=1:3))  
RowIndicies(data.frame())
```

SummaryRow*Query or set the summary row of a favir.data.frame*

Description

Query or set the last row of a favir.data.frame object

Usage

```
SummaryRow(fdf)  
SummaryRow(fdf) <- value
```

Arguments

fdf	A favir.data.frame object
value	A list containing the summary row data

Details

It's common to have a printed table with a final row that contains different information than the rest of the table. For instance, each field's last entry may be the sum of the previous data. These functions allow that last row to be highlighted when printed.

See Also

[FavirDF](#),

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10))
SummaryRow(fdf)
SummaryRow(fdf) <- list(a="Total", b=sum(fdf$b))
fdf
```

TextSize

Query or set the text size in a favir.data.frame

Description

Set the latex text size for a favir.data.frame object

Usage

```
TextSize(fdf)
TextSize(fdf) <- value
```

Arguments

fdf	A favir.data.frame object
value	Text size as a string, see below

Details

The text size will be passed through to latex. Thus the following options are available (in order of smallest to biggest): 'tiny', 'scriptsize', 'footnotesize', 'small', 'normalize', 'large', 'Large', 'LARGE', 'huge', 'Huge'.

See Also

[FavirDF](#), [Orientation](#).

Examples

```
fdf <- FavirDF(data.frame(a=1:5, b=6:10, c=11:15))
TextSize(fdf)
TextSize(fdf) <- "small"
fdf
```

%SplitOn%*Split a character type at the given substring*

Description

This function splits a character vector of length 1 into a list of strings.

Usage

```
string %SplitOn% substring
```

Arguments

string	A single string to separate into components
substring	The string that separates the components

Details

substring will not be included in the output. Importantly, the string can be recovered exactly from the output of SplitOn by gluing the pieces together with substring separating.

Examples

```
comps <- "hello there" %SplitOn% "l"
comps
Assert(all(comps == c("he", "o there")))

comps2 <- "hello there!" %SplitOn% "l"
comps2
Assert(all(comps2 == c("he", "", "o there", "")))
Assert(all("hello there!" == paste(comps2, collapse="l")))
```

Index

`$.favir.environment (FavirLoadModule)`, 7
`%SplitOn%`, 28

`Assert`, 2
`AssertSingle`, 3

`Caption`, 4, 6, 19
`Caption<- (Caption)`, 4

`DefaultFormatter`, 4, 6, 9
`DefaultFormatter<- (DefaultFormatter)`, 4
`DefaultRowColoring (RowColors)`, 25

`favir.colors`, 5
`FavirDF`, 4, 5, 6, 9–16, 19, 22–25, 27
`FavirLoadModule`, 7, 19
`FavirSweave`, 8
`FieldFormatters`, 5, 6, 9, 12–14, 22
`FieldFormatters<- (FieldFormatters)`, 9
`FieldGroup`, 10, 14, 15
`FieldGroup<- (FieldGroup)`, 10
`FieldHeadings`, 6, 11, 12, 15
`FieldHeadings<- (FieldHeadings)`, 11
`FieldSettings`, 11, 12, 13, 22
`FieldSettings<- (FieldSettings)`, 12
`formatC`, 22
`formatters`, 9, 13

`GetGlobalGroupFormatter (GroupFormatter)`, 13
`GroupFormatter`, 10, 13
`GroupFormatter<- (GroupFormatter)`, 13
`GroupHeading`, 10, 11, 14
`GroupHeading<- (GroupHeading)`, 14

`IncludeGraph`, 15, 16
`IncludeGrid`, 16, 16
`IncludeLegal`, 17, 18
`IncludePrelude`, 17, 18, 19
`InitPaper`, 17, 18, 18

`Label`, 4, 6, 19
`Label<- (Label)`, 19
`Last`, 20
`ListFieldGroups (FieldGroup)`, 10
`LookupDF`, 20

`MakeFormatter`, 5, 9, 21

`OnError`, 22
`Orientation`, 6, 23, 27
`Orientation<- (Orientation)`, 23

`ParentFieldGroups (FieldGroup)`, 10
`PreLatexFDF`, 23, 24
`print.favir.data.frame (FavirDF)`, 6
`print.favir.prelatex (PreLatexFDF)`, 23

`RBindPreLatex`, 24, 24
`RowColors`, 6, 25
`RowColors<- (RowColors)`, 25
`RowIndices`, 26

`seq`, 26
`SetGlobalDefaultFormatter (DefaultFormatter)`, 4
`SetGlobalGroupFormatter (GroupFormatter)`, 13
`stopifnot`, 2
`SummaryRow`, 26
`SummaryRow<- (SummaryRow)`, 26
`Sweave`, 8

`tail`, 20
`texi2dvi`, 8
`TextSize`, 6, 23, 27
`TextSize<- (TextSize)`, 27

`with.favir.environment (FavirLoadModule)`, 7