Package ‘fmtr’

September 8, 2022

Type Package
Title Easily Apply Formats to Data
Version 1.5.9
Author David Bosak
Maintainer David Bosak <dbosak01@gmail.com>
Description Contains a set of functions that can be used to apply formats to data frames or vectors. The package aims to provide functionality similar to that of SAS® formats. Formats are assigned to the format attribute on data frame columns. Then when the fdata() function is called, a new data frame is created with the column data formatted as specified. The package also contains a value() function to create a user-defined format, similar to a SAS® user-defined format.
License CC0
Encoding UTF-8
URL https://fmtr.r-sassy.org
BugReports https://github.com/dbosak01/fmtr/issues
Depends R (>= 3.6.0), common
Suggests testthat, knitr, markdown, covr, logr, utils
Imports tibble, stats, crayon, Rcpp
RoxygenNote 7.2.0
VignetteBuilder knitr
LinkingTo Rcpp
NeedsCompilation yes
Repository CRAN
Date/Publication 2022-09-08 04:32:54 UTC
**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>as.data.frame.fcat</td>
<td>3</td>
</tr>
<tr>
<td>as.data.frame.fmt</td>
<td>4</td>
</tr>
<tr>
<td>as.data.frame.fmt_lst</td>
<td>5</td>
</tr>
<tr>
<td>as.fcat</td>
<td>6</td>
</tr>
<tr>
<td>as.fcat.data.frame</td>
<td>6</td>
</tr>
<tr>
<td>as.fcat.fmt_lst</td>
<td>8</td>
</tr>
<tr>
<td>as.fcat.list</td>
<td>9</td>
</tr>
<tr>
<td>as.fcat.tbl_df</td>
<td>9</td>
</tr>
<tr>
<td>as.flist</td>
<td>11</td>
</tr>
<tr>
<td>as.flist.data.frame</td>
<td>13</td>
</tr>
<tr>
<td>as.flist.fcat</td>
<td>15</td>
</tr>
<tr>
<td>as.flist.list</td>
<td>17</td>
</tr>
<tr>
<td>as.flist.tbl_df</td>
<td>18</td>
</tr>
<tr>
<td>as.fmt</td>
<td>20</td>
</tr>
<tr>
<td>as.fmt.data.frame</td>
<td>21</td>
</tr>
<tr>
<td>condition</td>
<td>22</td>
</tr>
<tr>
<td>descriptions</td>
<td>24</td>
</tr>
<tr>
<td>fapply</td>
<td>25</td>
</tr>
<tr>
<td>fapply2</td>
<td>28</td>
</tr>
<tr>
<td>fattr</td>
<td>29</td>
</tr>
<tr>
<td>fattr&lt;-</td>
<td>31</td>
</tr>
<tr>
<td>fcat</td>
<td>32</td>
</tr>
<tr>
<td>fdata</td>
<td>33</td>
</tr>
<tr>
<td>flist</td>
<td>35</td>
</tr>
<tr>
<td>fmtr</td>
<td>37</td>
</tr>
<tr>
<td>fmt_cnt_pct</td>
<td>38</td>
</tr>
<tr>
<td>fmt_mean_sd</td>
<td>39</td>
</tr>
<tr>
<td>fmt_median</td>
<td>40</td>
</tr>
<tr>
<td>fmt_n</td>
<td>41</td>
</tr>
<tr>
<td>fmt_quantile_range</td>
<td>42</td>
</tr>
<tr>
<td>fmt_range</td>
<td>43</td>
</tr>
<tr>
<td>formats</td>
<td>44</td>
</tr>
<tr>
<td>FormattingStrings</td>
<td>45</td>
</tr>
<tr>
<td>is.fcat</td>
<td>47</td>
</tr>
<tr>
<td>is.flist</td>
<td>48</td>
</tr>
<tr>
<td>is.format</td>
<td>48</td>
</tr>
<tr>
<td>justification</td>
<td>49</td>
</tr>
<tr>
<td>labels.fmt</td>
<td>51</td>
</tr>
<tr>
<td>print.fcat</td>
<td>52</td>
</tr>
<tr>
<td>print.fmt</td>
<td>53</td>
</tr>
<tr>
<td>print.fmt_lst</td>
<td>53</td>
</tr>
<tr>
<td>read.fcat</td>
<td>54</td>
</tr>
<tr>
<td>value</td>
<td>55</td>
</tr>
<tr>
<td>widths</td>
<td>57</td>
</tr>
<tr>
<td>write.fcat</td>
<td>58</td>
</tr>
</tbody>
</table>

**Index** 60
as.data.frame.fcat  Convert a format catalog to a data frame

Description

This function takes the information stored in a format catalog, and converts it to a data frame. This data frame is useful for storage, editing, saving to a spreadsheet, etc. The data frame shows the name of the formats, their type, and the format expression. For user-defined formats, the data frame populates additional columns for the label and order.

Usage

```r
## S3 method for class 'fcat'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

- `x`: The format catalog to convert.
- `row.names`: Row names of the return data frame. Default is NULL.
- `optional`: TRUE or FALSE value indicating whether converting to syntactic variable names is desired. In the case of formats, the resulting data frame will always be returned with syntactic names, and this parameter is ignored.
- `...`: Any follow-on parameters.

Value

A data frame that contains the values stored in the format catalog.

See Also

Other fcat: `as.fcat.data.frame()`, `as.fcat.fmt_lst()`, `as.fcat.list()`, `as.fcat()`, `fcat()`, `is.fcat()`, `print.fcat()`, `read.fcat()`, `write.fcat()`

Examples

```r
# Create a format catalog
c1 <- fcat(num_fmt = \%.1f",
           label_fmt = value(condition(x == "A", "Label A"),
                              condition(x == "B", "Label B"),
                              condition(TRUE, "Other")),
           date_fmt = \"%d%b%Y\")

# Convert catalog to data frame to view the structure
df <- as.data.frame(c1)
print(df)
```

```
#  Name Type Expression  Label Order
# 1  num_fmt  S  \%.1f     NA
```
as.data.frame.fmt

# Convert data frame back to a format catalog
c2 <- as.fcat(df)

---

as.data.frame.fmt  
Casts a format to a data frame

Description

Cast a format object to a data frame. This function is a class-specific implementation of the the generic as.data.frame method.

Usage

```r
## S3 method for class 'fmt'
as.data.frame(
x,
row.names = NULL,
optional = FALSE,
..., 
name = deparse(substitute(x, env = environment()))
)
```

Arguments

- `x`  
  An object of class "fmt".
- `row.names`  
  Row names of the return data frame. Default is NULL.
- `optional`  
  TRUE or FALSE value indicating whether converting to syntactic variable names is options. In the case of formats, the resulting data frame will always be returned with syntactic names, and this parameter is ignored.
- `...`  
  Any follow-on parameters.
- `name`  
  An optional name for the format. By default, the name of the variable holding the format will be used.

See Also

Other fmt: `as.fmt.data.frame()`, `as.fmt()`, `condition()`, `is.format()`, `labels.fmt()`, `print.fmt()`, `value()`
Description
This function takes the information stored in a formatting list, and converts it to a data frame. The data frame format is useful for storage, editing, saving to a spreadsheet, etc. The data frame shows the name of the formats, their type, and the format expression. For user-defined formats, the data frame populates additional columns for the label and order.

Usage
## S3 method for class 'fmt_lst'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

Arguments

x
The formatting list to convert.

row.names
Row names for the returned data frame. Default is NULL.

optional
TRUE or FALSE value indicating whether converting to syntactic variable names is desired. In the case of formats, the resulting data frame will always be returned with syntactic names, and this parameter is ignored.

...
Any follow-on parameters.

Value
A data frame that contains the values stored in the formatting list.

See Also
Other flist: as.flist.data.frame(), as.flist.fcat(), as.flist.list(), as.flist.tbl_df(), as.flist(), fлист(), is.flist(), print.fmt_lst()

Examples
# Create a formatting list
c1 <- flist(num_fmt = "%.1f",
            label_fmt = value(condition(x == "A", "Label A"),
                                condition(x == "B", "Label B"),
                                condition(TRUE, "Other")),
            date_fmt = "%d%b%Y")

# Convert catalog to data frame to view the structure
df <- as.data.frame(c1)
print(df)

    Name Type Expression    Label Order
    1 num_fmt  S    %.1f       NA
# Convert data frame back to a formatting list
c2 <- as.flist(df)

---

**as.fcat**  
*Generic casting method for format catalogs*

**Description**

A generic method for casting objects to a format catalog. Individual objects will inherit from this function.

**Usage**

```r
as.fcat(x)
```

**Arguments**

- `x`  
The object to cast.

**Value**

A formatting object, created using the information in the input object.

**See Also**

For class-specific methods, see `as.fcat.data.frame`, `as.fcat.list`, and `as.fcat.fmt_lst`.

Other fcat: `as.data.frame.fcat()`, `as.fcat.data.frame()`, `as.fcat.fmt_lst()`, `as.fcat.list()`, `fcat()`, `is.fcat()`, `print.fcat()`, `read.fcat()`, `write.fcat()`

---

**as.fcat.data.frame**  
*Convert a data frame to a format catalog*

**Description**

This function takes a data frame as input and converts it to a format catalog based on the information contained in the data frame. The data frame should have 5 columns: "Name", "Type", "Expression", "Label" and "Order".

**Usage**

```r
## S3 method for class 'data.frame'
as.fcat(x)
```
Arguments

x The data frame to convert.

Details

The `as.fcat.data.frame` converts a data frame to a format catalog. A corresponding conversion for class "tbl_df" converts a tibble.

To understand the structure of the input data frame, create a format and use the `as.data.frame` method to convert the format to a data frame. Then observe the columns and organization of the data.

Value

A format catalog based on the information contained in the input data frame.

Input Data Frame Specifications

The input data frame should contain the following columns:

- **Name**: The name of the format
- **Type**: The type of format. See the type codes below.
- **Expression**: The formatting expression. The expression will hold different types of values depending on the format type.
- **Label**: The label for user-defined, "U" type formats.
- **Order**: The order for user-defined, "U" type formats.

Any additional columns will be ignored. Column names are case-insensitive.

Valid values for the "Type" column are as follows:

- **U**: User Defined List created with the `value` function.
- **S**: A formatting string of formatting codes. See `FormattingStrings`.
- **F**: A vectorized function.
- **V**: A named vector lookup.

The "Label" and "Order" columns are used only for a type "U", user-defined format created with the `value` function.

See Also

Other fcat: `as.data.frame.fcat()`, `as.fcat.fmt_lst()`, `as.fcat.list()`, `as.fcat()`, `fcat()`, `is.fcat()`, `print.fcat()`, `read.fcat()`, `write.fcat()`
Examples

```r
# Create a format catalog
cl <- fcat(num_fmt = "%.1f",
           label_fmt = value(condition(x == "A", "Label A"),
                               condition(x == "B", "Label B"),
                               condition(TRUE, "Other")),
           date_fmt = "%d-%b-%Y")

# Convert catalog to data frame to view the structure
df <- as.data.frame(cl)
print(df)

# Name Type Expression Label Order
# 1 num_fmt S %.1f NA
# 2 label_fmt U x == "A" Label A NA
# 3 label_fmt U x == "B" Label B NA
# 4 label_fmt U TRUE Other NA
# 5 date_fmt S %d-%b-%Y NA

# Convert data frame back to a format catalog
c2 <- as.fcat(df)

# Use re-converted catalog
fapply(123.456, c2$num_fmt)
fapply(c("A", "B", "C", "B"), c2$label_fmt)
fapply(Sys.Date(), c2$date_fmt)
```

---

**as.fcat.fmt_lst**

Convert a formatting list to a format catalog

**Description**

The `as.fcat.list` function converts a formatting list to a format catalog. For additional information on formatting lists, see `flist`.

**Usage**

```r
## S3 method for class 'fmt_lst'
as.fcat(x)
```

**Arguments**

- `x` The formatting list to convert.

**Value**

A format catalog based on the formats contained in the input formatting list.
**as.fcat.list**

Convert a list to a format catalog

Description

The `as.fcat.list` function converts a list of formats to a format catalog. Items in the list must be named.

Usage

```r
## S3 method for class 'list'
as.fcat(x)
```

Arguments

- `x` The list to convert. List must contained named formats.

Value

A format catalog based on the formats contained in the input list.

See Also

Other fcat: `as.data.frame.fcat()`, `as.fcat.data.frame()`, `as.fcat.list()`, `as.fcat()`, `fcat()`, `is.fcat()`, `print.fcat()`, `read.fcat()`, `write.fcat()`

**as.fcat.tbl_df**

Convert a tibble to a format catalog

Description

This function takes a data frame as input and converts it to a format catalog based on the information contained in the data frame. The data frame should have 5 columns: "Name", "Type", "Expression", "Label" and "Order".

Usage

```r
## S3 method for class 'tbl_df'
as.fcat(x)
```
Arguments

x  The data frame to convert.

Details

The \texttt{as.fcat.tbl_df} converts a data frame to a format catalog. A corresponding conversion for class "tbl_df" converts a tibble.

To understand the structure of the input data frame, create a format and use the \texttt{as.data.frame} method to convert the format to a data frame. Then observe the columns and organization of the data.

Value

A format catalog based on the information contained in the input data frame.

Input Data Frame Specifications

The input data frame should contain the following columns:

- \textbf{Name}: The name of the format
- \textbf{Type}: The type of format. See the type codes below.
- \textbf{Expression}: The formatting expression. The expression will hold different types of values depending on the format type.
- \textbf{Label}: The label for user-defined, "U" type formats.
- \textbf{Order}: The order for user-defined, "U" type formats.

Any additional columns will be ignored. Column names are case-insensitive.

Valid values for the "Type" column are as follows:

- \textbf{U}: User Defined List created with the \texttt{value} function.
- \textbf{S}: A formatting string of formatting codes. See \texttt{FormattingStrings}.
- \textbf{F}: A vectorized function.
- \textbf{V}: A named vector lookup.

The "Label" and "Order" columns are used only for a type "U", user-defined format created with the \texttt{value} function.

See Also

Other fcat: \texttt{as.data.frame.fcat()}, \texttt{as.fcat.fmt_lst()}, \texttt{as.fcat.list()}, \texttt{as.fcat()}, \texttt{fcat()}, \texttt{is.fcat()}, \texttt{print.fcat()}, \texttt{read.fcat()}, \texttt{write.fcat()}
Examples

# Create a format catalog
c1 <- fcat(num_fmt = "%.1f",
label_fmt = value(condition(x == "A", "Label A"),
condition(x == "B", "Label B"),
condition(TRUE, "Other")),
date_fmt = "%d-%b-%Y")

# Convert catalog to data frame to view the structure
df <- as.data.frame(c1)
print(df)

# Name Type Expression Label Order
# 1 num_fmt S %.1f NA
# 2 label_fmt U x == "A" Label A NA
# 3 label_fmt U x == "B" Label B NA
# 4 label_fmt U TRUE Other NA
# 5 date_fmt S %d-%b-%Y NA

# Convert data frame back to a format catalog
c2 <- as.fcat(df)

# Use re-converted catalog
fapply(123.456, c2$num_fmt)
fapply(c("A", "B", "C", "B"), c2$label_fmt)
fapply(Sys.Date(), c2$date_fmt)

as.flist

Convert a formatting list

Description

Converts an object to a formatting list. All other parameters are the same as the flist function.

Usage

as.flist(x, type = "column", lookup = NULL, simplify = TRUE)

Arguments

x Object to convert.
type The type of formatting list. Valid values are 'row' or 'column'. The default value is 'column'.
lookup A lookup vector. Used for looking up the format from the formatting list. This parameter is only used for 'row' type formatting lists.
simplify Whether to simplify the results to a vector. Valid values are TRUE or FALSE. Default is TRUE. If the value is set to FALSE, the return type will be a list.
Details

To apply more than one formatting object to a vector, use a formatting list. There are two types of formatting list: column and row. The column type formatting lists applies all formats to all values in the vector. The row type formatting list can apply a different format to each value in the vector.

Further, there are two styles of row type list: ordered and lookup. The ordered style applies each format in the list to the vector values in the order specified. The ordered style will recycle the formats as needed. The lookup style formatting list uses a lookup to determine which format from the list to apply to a particular value of the vector. The lookup column values should correspond to names on the formatting list.

Examples of column type and row type formatting lists are given below.

Value

A formatting list object.

See Also

Other flist: as.data.frame.fmt_lst(), as.flist.data.frame(), as.flist.fcat(), as.flist.list(), as.flist.tbl_df(), flist(), is.flist(), print.fmt_lst()

Examples

## Example 1: Formatting List - Column Type ##
# Set up data
v1 <- c(Sys.Date(), Sys.Date() + 30, Sys.Date() + 60)

# Create formatting list
fl1 <- flist("%B", "The month is: %s")

# Apply formatting list to vector
fapply(v1, fl1)

# [1] "The month is: October" "The month is: November" "The month is: December"

## Example 2: Formatting List - Row Type ordered ##
# Set up data
# Notice each row has a different data type
l1 <- list("A", 1.263, as.Date("2020-07-21"),
            "B", 5.8732, as.Date("2020-10-17"))

# These formats will be recycled in the order specified
fl2 <- flist(type = "row",
c(A = "Label A", B = "Label B"),
"%.1f",
"%d%b%Y")

fapply(l1, fl2)

# [1] "Label A" "1.3" "21Jul2020" "Label B" "5.9" "17Oct2020"
## Example 3: Formatting List - Row Type with lookup ##

# Set up data
# Notice each row has a different data type
l2 <- list(2841.258, "H", as.Date("2020-06-19"),
            "L", as.Date("2020-04-24"), 1382.8865)
v3 <- c("num", "char", "date", "char", "date", "num")

# Create formatting list
fl3 <- flist(type = "row", lookup = v3,
             num = function(x) format(x, digits = 2, nsmall = 1,
                                      big.mark="",
                                      char = value(condition(x == "H", "High"),
                                                    condition(x == "L", "Low"),
                                                    condition(TRUE, "NA")),
             date = "%d%b%Y")

# Apply formatting list to vector, using lookup
fapply(l2, fl3)

# [1] "2,841.3" "High" "19Jun2020" "Low" "24Apr2020" "1,382.9"

---

**as.flist.data.frame**

Convert a data frame to a formatting list

### Description

Converts a data frame to a formatting list. All other parameters are the same as the `flist` function.

### Usage

```
## S3 method for class 'data.frame'
as.flist(x, type = "column", lookup = NULL, simplify = TRUE)
```

### Arguments

- **x**
  Data frame to convert.

- **type**
  The type of formatting list. Valid values are 'row' or 'column'. The default value is 'column'.

- **lookup**
  A lookup vector. Used for looking up the format from the formatting list. This parameter is only used for 'row' type formatting lists.

- **simplify**
  Whether to simplify the results to a vector. Valid values are TRUE or FALSE. Default is TRUE. If the value is set to FALSE, the return type will be a list.
Details

To apply more than one formatting object to a vector, use a formatting list. There are two types of formatting list: column and row. The column type formatting lists applies all formats to all values in the vector. The row type formatting list can apply a different format to each value in the vector.

Further, there are two styles of row type list: ordered and lookup. The ordered style applies each format in the list to the vector values in the order specified. The ordered style will recycle the formats as needed. The lookup style formatting list uses a lookup to determine which format from the list to apply to a particular value of the vector. The lookup column values should correspond to names on the formatting list.

Examples of column type and row type formatting lists are given below.

Value

A formatting list object.

See Also

Other flist: as.data.frame.fmt_lst(), as.flist.fcat(), as.flist.list(), as.flist.tbl_df(), as.flist(), flist(), is.flist(), print.fmt_lst()

Examples

## Example 1: Formatting List - Column Type ##
# Set up data
v1 <- c(Sys.Date(), Sys.Date() + 30, Sys.Date() + 60)

# Create formatting list
fl1 <- flist("%B", "The month is: %s")

# Apply formatting list to vector
fapply(v1, fl1)

# [1] "The month is: October" "The month is: November" "The month is: December"

## Example 2: Formatting List - Row Type ordered ##
# Set up data
# Notice each row has a different data type
l1 <- list("A", 1.263, as.Date("2020-07-21"),
"B", 5.8732, as.Date("2020-10-17"))

# These formats will be recycled in the order specified
fl2 <- flist(type = "row",
c(A = "Label A", B = "Label B"),
"%.1f",
"%d%b%Y")

fapply(l1, fl2)

# [1] "Label A" "1.3" "21Jul2020" "Label B" "5.9" "17Oct2020"
## Example 3: Formatting List - Row Type with lookup ##

Set up data

Notice each row has a different data type

```r
l2 <- list(2841.258, "H", as.Date("2020-06-19"),
            "L", as.Date("2020-04-24"), 1382.8865)
v3 <- c("num", "char", "date", "char", "date", "num")
```

Create formatting list

```r
fl3 <- flist(type = "row", lookup = v3,
              num = function(x) format(x, digits = 2, nsmall = 1,
                          big.mark="","),
              char = value(condition(x == "H", "High"),
                            condition(x == "L", "Low"),
                            condition(TRUE, "NA")),
              date = "%d%b%Y")
```

Apply formatting list to vector, using lookup

```r
fapply(l2, fl3)
```

```
[1] "2,841.3" "High" "19Jun2020" "Low" "24Apr2020" "1,382.9"
```

---

### as.flist.fcat

Convert a format catalog to a formatting list

**Description**

Converts a format catalog to a formatting list. All other parameters are the same as the flist function.

**Usage**

```r
## S3 method for class 'fcat'
as.flist(x, type = "column", lookup = NULL, simplify = TRUE)
```

**Arguments**

- **x**: Format catalog to convert.
- **type**: The type of formatting list. Valid values are 'row' or 'column'. The default value is 'column'.
- **lookup**: A lookup vector. Used for looking up the format from the formatting list. This parameter is only used for 'row' type formatting lists.
- **simplify**: Whether to simplify the results to a vector. Valid values are TRUE or FALSE. Default is TRUE. If the value is set to FALSE, the return type will be a list.
Details

To apply more than one formatting object to a vector, use a formatting list. There are two types of formatting list: column and row. The column type formatting lists applies all formats to all values in the vector. The row type formatting list can apply a different format to each value in the vector.

Further, there are two styles of row type list: ordered and lookup. The ordered style applies each format in the list to the vector values in the order specified. The ordered style will recycle the formats as needed. The lookup style formatting list uses a lookup to determine which format from the list to apply to a particular value of the vector. The lookup column values should correspond to names on the formatting list.

Examples of column type and row type formatting lists are given below.

Value

A formatting list object.

See Also

Other flist: as.data.frame.fmt_lst(), as.flist.data.frame(), as.flist.list(), as.flist.tbl_df(), as.flist(), flist(), is.flist(), print.fmt_lst()

Examples

```r
## Example 1: Formatting List - Column Type ##
# Set up data
v1 <- c(Sys.Date(), Sys.Date() + 30, Sys.Date() + 60)

# Create formatting list
fl1 <- flist("%B", "The month is: %s")

# Apply formatting list to vector
fapply(v1, fl1)

# [1] "The month is: October" "The month is: November" "The month is: December"

## Example 2: Formatting List - Row Type ordered ##
# Set up data
# Notice each row has a different data type
l1 <- list("A", 1.263, as.Date("2020-07-21"),
        "B", 5.8732, as.Date("2020-10-17"))

# These formats will be recycled in the order specified
fl2 <- flist(type = "row",
        c(A = "Label A", B = "Label B"),
        ".1f",
        "%d%b%Y")

fapply(l1, fl2)

# [1] "Label A" "1.3" "21Jul2020" "Label B" "5.9" "17Oct2020"
```
as.flist.list

Convert a list to a formatting list

Description

Converts a normal list to a formatting list. All other parameters are the same as the flist function.

Usage

## S3 method for class 'list'
as.flist(x, type = "column", lookup = NULL, simplify = TRUE)

Arguments

x List to convert.

type The type of formatting list. Valid values are 'row' or 'column'. The default value is 'column'.

lookup A lookup vector. Used for looking up the format from the formatting list. This parameter is only used for 'row' type formatting lists.

simplify Whether to simplify the results to a vector. Valid values are TRUE or FALSE. Default is TRUE. If the value is set to FALSE, the return type will be a list.
Details

To apply more than one formatting object to a vector, use a formatting list. There are two types of formatting list: column and row. The column type formatting lists applies all formats to all values in the vector. The row type formatting list can apply a different format to each value in the vector.

Further, there are two styles of row type list: ordered and lookup. The ordered style applies each format in the list to the vector values in the order specified. The ordered style will recycle the formats as needed. The lookup style formatting list uses a lookup to determine which format from the list to apply to a particular value of the vector. The lookup column values should correspond to names on the formatting list.

Examples of column type and row type formatting lists are given below.

Value

A formatting list object.

See Also

`flist` function documentation for additional details.

Other `flist`: `as.data.frame.fmt_lst()` , `as.flist.data.frame()` , `as.flist.fcat()` , `as.flist.tbl_df()` , `as.flist()`, `flist()`, `is.flist()`, `print.fmt_lst()`

Examples

# Example 1: Create flist from list - column type
lst1 <- list("%d%b%Y", "%.1f")
fl1 <- as.flist(lst1, type = "column")

# Example 2: Create flist from list - row type
lst2 <- list(lkup = c(A = "Label A", B = "Label B"),
             dec1 = "%.1f",
             dt1 = "%d%b%Y")
fl2 <- as.flist(lst2, type = "row")
Arguments

x Tibble to convert.
type The type of formatting list. Valid values are 'row' or 'column'. The default value is 'column'.
lookup A lookup vector. Used for looking up the format from the formatting list. This parameter is only used for 'row' type formatting lists.
simplify Whether to simplify the results to a vector. Valid values are TRUE or FALSE. Default is TRUE. If the value is set to FALSE, the return type will be a list.

Details

To apply more than one formatting object to a vector, use a formatting list. There are two types of formatting list: column and row. The column type formatting lists applies all formats to all values in the vector. The row type formatting list can apply a different format to each value in the vector.

Further, there are two styles of row type list: ordered and lookup. The ordered style applies each format in the list to the vector values in the order specified. The ordered style will recycle the formats as needed. The lookup style formatting list uses a lookup to determine which format from the list to apply to a particular value of the vector. The lookup column values should correspond to names on the formatting list.

Examples of column type and row type formatting lists are given below.

Value

A formatting list object.

See Also

Other flist: `as.data.frame.fmt_lst()`, `as.flist.data.frame()`, `as.flist.fcat()`, `as.flist.list()`, `as.flist()`, `flist()`, `is.flist()`, `print.fmt_lst()`

Examples

```r
## Example 1: Formatting List - Column Type ##
# Set up data
v1 <- c(Sys.Date(), Sys.Date() + 30, Sys.Date() + 60)

# Create formatting list
fl1 <- flist("%B", "The month is: %s")

# Apply formatting list to vector
fapply(v1, fl1)

# [1] "The month is: October" "The month is: November" "The month is: December"

## Example 2: Formatting List - Row Type ordered ##
# Set up data
# Notice each row has a different data type
l1 <- list("A", 1.263, as.Date("2020-07-21"),
           "B", 5.8732, as.Date("2020-10-17"))
```

# These formats will be recycled in the order specified
fl2 <- flist(type = "row",
              c(A = "Label A", B = "Label B"),
              "%.1f",
              "%d%b%Y")

fapply(l1, fl2)

# [1] "Label A" "1.3" "21Jul2020" "Label B" "5.9" "17Oct2020"

## Example 3: Formatting List - Row Type with lookup ##
# Set up data
# Notice each row has a different data type
l2 <- list(2841.258, "H", as.Date("2020-06-19"),
           "L", as.Date("2020-04-24"), 1382.8865)
v3 <- c("num", "char", "date", "char", "date", "num")

# Create formatting list
fl3 <- flist(type = "row", lookup = v3,
              num = function(x) format(x, digits = 2, nsmall = 1,
                                       big.mark="",)
              char = value(condition(x == "H", "High"),
                            condition(x == "L", "Low"),
                            condition(TRUE, "NA")),
              date = "%d%b%Y")

# Apply formatting list to vector, using lookup
fapply(l2, fl3)

# [1] "2,841.3" "High" "19Jun2020" "Low" "24Apr2020" "1,382.9"

---

**as.fmt**  
Generic casting method for formats

**Description**
A generic method for casting objects to a format. Individual objects will inherit from this function.

**Usage**

```r
as.fmt(x)
```

**Arguments**

- `x`  
The object to cast.

**Value**

A formatting object, created using the information in the input object.
as.fmt.data.frame

Convert a data frame to a user-defined format

Description
This function takes a data frame as input and converts it to a user-defined format based on the information contained in the data frame. The data frame should have 5 columns: "Name", "Type", "Expression", "Label" and "Order".

Usage
## S3 method for class 'data.frame'
as.fmt(x)

Arguments
x
The data frame to convert.

Details
The as.fmt.data.frame function converts a data frame to a user-defined format.

To understand the structure of the input data frame, create a user-defined format and use the as.data.frame method to convert the format to a data frame. Then observe the columns and organization of the data.

Value
A format catalog based on the information contained in the input data frame.

Input Data Frame Specifications

The input data frame should contain the following columns:

- **Name**: The name of the format
- **Type**: The type of format. See the type codes below.
- **Expression**: The formatting expression. The expression will hold different types of values depending on the format type. Within the data frame, this expression is stored as a character string.
- **Label**: The label for user-defined, "U" type formats.
- **Order**: The order for user-defined, "U" type formats.

Any additional columns will be ignored. Column names are case-insensitive. Valid values for the "Type" column are as follows:
- **U**: User Defined List created with the `value` function.
- **S**: A formatting string of formatting codes. See `FormattingStrings`.
- **F**: A vectorized function.
- **V**: A named vector lookup.

The "Label" and "Order" columns are used only for a type "U", user-defined format created with the `value` function.

**See Also**

Other fmt: `as.data.frame.fmt()`, `as.fmt()`, `condition()`, `is.format()`, `labels.fmt()`, `print.fmt()`, `value()`

**Examples**

```r
# Create a user-defined format
f1 <- value(condition(x == "A", "Label A"),
             condition(x == "B", "Label B"),
             condition(TRUE, "Other"))

# Convert user-defined format to data frame to view the structure
df <- as.data.frame(f1)
print(df)

# Name Type Expression Label Order
# 1 f1 U x == "A" Label A NA
# 2 f1 U x == "B" Label B NA
# 3 f1 U TRUE Other NA

# Convert data frame back to a user-defined format
f2 <- as.fmt(df)

# Use re-converted format
fapply(c("A", "B", "C", "B"), f2)
# [1] "Label A" "Label B" "Other" "Label B"
```

---

**condition**

`Define a condition for a user-defined format`

**Description**

The condition function creates a condition for a user-defined format. It is typically used in conjunction with the `value` function.

**Usage**

`condition(expr, label, order = NULL)`
Arguments

**expr**
A valid R expression. The value in the expression is identified by the variable 'x', i.e. x == 'A' or x > 3 & x < 6. The expression should not be quoted. The expression parameter will accept equality, relational, and logical operators. It will also accept numeric or string literals. String literals should be quoted. It will not accept functions or any expression that includes a comma. For these more complex operations, it is best to use a vectorized function. See `fapply` for an example of a vectorized function.

**label**
A label to be assigned if the expression is TRUE. The label can any valid literal value. Typically, the label will be a character string. However, the label parameter does not restrict the data type. Meaning, the label could also be a number, date, or other R object type. The label may also be a string format, which allows you to perform conditional formatting.

**order**
An optional integer order number. When used, this parameter will effect the order of the labels returned from the `labels.fmt` function. The purpose of the parameter is to control ordering of the format labels independently of the order they are assigned in the conditions. The order parameter is useful when you are using the format labels to assign ordered levels in a factor.

Details

The `condition` function creates a condition as part of a format definition. The format is defined using the `value` function. The condition is defined as an expression/label pair. The expression parameter can be any valid R expression. The label parameter can be any valid literal. Conditions are evaluated in the order they are assigned. A default condition is created by assigning the expression parameter to TRUE. If your data can contain missing values, it is recommended that you test for those values first. Any data values that do not meet one of the conditions will fall through the format as-is.

The condition object is an S3 class of type "fmt_cond". The condition labels can be extracted from the format using the `labels` function.

The format object may be applied to a vector using the `fapply` function. See `fapply` for further details.

Value

The new condition object.

See Also

`fdata` to apply formatting to a data frame, `value` to define a format, `levels` or `labels.fmt` to access the labels, and `fapply` to apply the format to a vector.

Other fmt: `as.data.frame.fmt()`, `as.fmt.data.frame()`, `as.fmt()`, `is.format()`, `labels.fmt()`, `print.fmt()`, `value()`

Examples

```r
# Set up vector
```
v1 <- c("A", "B", "C", "B")

# Define format
fmt1 <- value(condition(x == "A", "Label A"),
               condition(x == "B", "Label B"),
               condition(TRUE, "Other"))

# Apply format to vector
v2 <- fapply(v1, fmt1)
v2

---

**descriptions**

Get or set descriptions for data frame columns

**Description**

The `descriptions` function extracts all assigned description attributes from a data frame, and returns them in a named list. The function also assigns description attributes from a named list.

**Usage**

```r
descriptions(x)
```

```r
x <- value
```

**Arguments**

- `x` A data frame or tibble
- `value` A named list of description values.

**Details**

If descriptions are assigned to the "description" attributes of the data frame columns, the `descriptions` function will extract those values. The function will return the description values in a named list, where the names correspond to the name of the column that the description was assigned to. If a column does not have a description attribute assigned, that column will not be included in the list.

When used on the receiving side of an assignment, the function will assign descriptions to a data frame. The description values should be in a named list, where each name corresponds to the name of the data frame column to assign values to.

Finally, if you wish to clear out the description attributes, assign a NULL value to the `descriptions` function.

**Value**

A named list of description values.
fapply

See Also

data to display formatted data, value to create user-defined formats, and fapply to apply formatting to a vector.

Examples

# Take subset of data
df1 <- mtcars[1:10, c("mpg", "cyl")]

# Print current state
print(df1)

# Assign descriptions
descriptions(df1) <- list(mpg = "Miles per Gallon", cyl = "Cylinders")

# Display descriptions
descriptions(df1)

# Clear descriptions
descriptions(df1) <- NULL

# Confirm descriptions are cleared
descriptions(df1)

---

fapply

Apply formatting to a vector

Description

The fapply function applies formatting to a vector.

Usage

fapply(x, format = NULL, width = NULL, justify = NULL)

Arguments

x A vector, factor, or list to apply the format to.

format A format to be applied.

width The desired character width of the formatted vector. Default value is NULL, meaning the vector will be variable width.

justify Whether to justify the return vector. Valid values are 'left', 'right', 'center', 'centre', or 'none'.

Details

The `fapply` function accepts several types of formats: formatting strings, named vectors, vectorized functions, or user-defined formats. It also accepts a formatting list, composed of any of the previous types. The function will first determine the type of format, and then apply the format in the appropriate way. Results are returned as a vector.

The function also has parameters for width and justification.

Parameters may also be passed as attributes on the vector. See the `fattr` function for additional information on setting formatting attributes.

Value

A vector of formatted values.

Types of Formats

The `fapply` function will process any of the following types of formats:

- **Formatting string**: A single string will be interpreted as a formatting string. See the `FormattingStrings` documentation for further details.
- **Named vector**: A named vector can serve as a lookup list or decode for a vector. You can use a named vector to perform simple lookups on character vectors.
- **Format object**: A format object may be created using the `value` function. The format object is included in the `fmtr` package, and is specially designed for data categorization.
- **Vectorized formatting function**: A vectorized function provides the most flexibility and power over your formatting. You can use an existing formatting function from any package, or create your own vectorized formatting function using `Vectorize`.

`fapply` will also accept a formatting list, which can contain any number of formats from the above list. To create a formatting list, see the `flist` function.

See Also

`fcat` to create a format catalog, `value` to define a format, `fattr` to easily set the formatting attributes of a vector, and `flist` to define a formatting list. Also see `fdata` to apply formats to an entire data frame, and `FormattingStrings` for how to define a formatting string.

Examples

```
## Example 1: Formatting string ##
v1 <- c(1.235, 8.363, 5.954, 2.465)

# Apply string format.
fapply(v1, "%.1f")
```

```
## Example 2: Named vector ##
# Set up vector
v2 <- c("A", "B", "C", "B")
```

# Set up named vector for formatting
fmt2 <- c(A = "Vector Label A", B = "Vector Label B", C = "Vector Label C")

# Apply format to vector
fapply(v2, fmt2)

## Example 3: User-defined format ##
# Define format
fmt3 <- value(condition(x == "A", "Format Label A"),
                condition(x == "B", "Format Label B"),
                condition(TRUE, "Format Other"))

# Apply format to vector
fapply(v2, fmt3)

## Example 4: Formatting function ##
# Set up vectorized function
fmt4 <- Vectorize(function(x) {
    if (x %in% c("A", "B"))
        ret <- paste("Function Label", x)
    else
        ret <- "Function Other"

    return(ret)
})

# Apply format to vector
fapply(v2, fmt4)

## Example 5: Formatting List - Row Type ##
# Set up data
# Notice each row has a different data type
v3 <- list(2841.258, "H", Sys.Date(),
          "L", Sys.Date() + 60, 1382.8865)
v4 <- c("int", "char", "date", "char", "date", "int")

# Create formatting list
lst <- flist(type = "row", lookup = v4,
            int = function(x) format(x, digits = 2, nsmall = 1,
                                     big.mark="",
                                     char = value(condition(x == "H", "High"),
                                                  condition(x == "L", "Low"),
                                                  condition(TRUE, "NA")),
                                      date = "%d%b%Y")

# Apply formatting list to vector
fapply(v3, lst)
## Example 6: Formatting List - Column Type ##

Set up data

```r
v5 <- c(Sys.Date(), Sys.Date() + 30, Sys.Date() + 60)
```

Create formatting list

```r
lst <- flist("%B", "This month is: %s", type = "column")
```

Apply formatting list to vector

```r
fapply(v5, lst)
```

---

**fapply2**

*Apply formatting to two vectors*

**Description**

The `fapply2` function applies formatting to two different vectors, and combines them into a single vector. This function is useful in cases where your data is in two different variables, and you would like them displayed as a single column for reporting purposes. For example, if you wish to create one column to display mean and standard deviation.

**Usage**

```r
fapply2(
  x1, 
  x2, 
  format1 = NULL, 
  format2 = NULL, 
  sep = " ", 
  width = NULL, 
  justify = NULL
)
```

**Arguments**

- `x1`: A vector, factor, or list to apply the format1 to.
- `x2`: A second vector, factor, or list to which format2 will be applied.
- `format1`: A format to be applied to the first input.
- `format2`: A format to be applied to the second input.
- `sep`: A separator to use between the two formatted values. Default is a single blank space (" ").
- `width`: The desired character width of the formatted vector. Default value is NULL, meaning the vector will be variable width.
- `justify`: Whether to justify the return vector. Valid values are 'left', 'right', 'center', 'centre', or 'none'.

Details

The \texttt{fapply2} function works nearly the same as \texttt{fapply}. The difference is it has parameters for two vectors and formats instead of one. The output of the function is a single vector. The function essentially calls \texttt{fapply} on each vector and pastes them together afterwards.

There is an additional \texttt{sep} parameter so you can define a separator between the two formatted values. The \texttt{width} and \texttt{justify} parameters will apply to the single vector result. The function will also pick up format attributes on the supplied vectors.

The \texttt{fapply2} function accepts any of the format types that \texttt{fapply} accepts. See \texttt{fapply} for additional information on the types of formats that can be applied.

Parameters may also be passed as attributes on the vector. See the \texttt{fattr} function for additional information on setting formatting attributes.

Value

A vector of formatted values.

See Also

\texttt{fapply} to format a single input, \texttt{fcat} to create a format catalog, \texttt{value} to define a format, \texttt{fattr} to easily set the formatting attributes of a vector, and \texttt{flist} to define a formatting list. Also see \texttt{fdata} to apply formats to an entire data frame, and FormattingStrings for how to define a formatting string.

Examples

```r
# Create sample data
dt <- c(2.1, 5, 6, 9, 2, 7, 3)

# Calculate mean and standard deviation
v1 <- mean(dt)
v2 <- sd(dt)

# Apply formats and combine
fapply2(v1, v2, "%.1f", "(%.2f)")
# [1] "4.9 (2.66)"
```

---

\texttt{fattr} \hspace{1cm} \textit{Set formatting attributes}

Description

Assign formatting attributes to a vector.
Usage

fattr(
  x,
  format = NULL,
  width = NULL,
  justify = NULL,
  label = NULL,
  description = NULL,
  keep = TRUE
)

Arguments

x                   The vector or data frame column to assign attributes to.
format             The format to assign to the format attribute. The format can be a formatting
                   string, a named vector decode, a vectorized formatting function, or a formatting
                   list.
width               The desired width of the formatted output.
justify             Justification of the output vector. Valid values are 'none', 'left', 'right', 'center',
                   or 'centre'.
label               A label string to assign to the vector. This parameter was added for convenience,
                   as the label is frequently assigned at the same time the formatting attributes are
                   assigned.
description         A description string to assign to the vector. This parameter was added for conve-
                   nience, as the description is frequently assigned at the same time the formatting
                   attributes are assigned.
keep                Whether to keep any existing formatting attributes and transfer to the new vector.
                   Default value is TRUE.

Details

The `fattr` function is a convenience function for assigning formatting attributes to a vector. The
function accepts parameters for format, width, and justify. Any formatting attributes assigned can
be applied using `fapply` or `fdata`.

Value

The vector with formatting attributes assigned.

See Also

`fdata` to apply formats to a data frame, `fapply` to apply formats to a vector. See `FormattingStrings`
for documentation on formatting strings.
Examples

# Create vector
a <- c(1.3243, 5.9783, 2.3848)

# Assign format attributes
a <- fattr(a, format = "%1f", width = 10, justify = "center")

# Apply format attributes
fapply(a)

# [1] " 1.3 " " 6.0 " " 2.4 "

fattr<-  
Set formatting attributes

Description

Assign formatting attributes to a vector

Usage

fattr(x) <- value

Arguments

x  The vector or data frame column to assign attributes to.
value  A named vector of attribute values.

Details

The fattr function is a convenience function for assigning formatting attributes to a vector. The function accepts a named list of formatting attributes. Valid names are 'format', 'width', 'justify', 'label' and 'description'. See fattr for additional details.

See Also

fdata to apply formats to a data frame, fapply to apply formats to a vector.

Examples

# Create vector
a <- c(1.3243, 5.9783, 2.3848)

# Assign format attributes
fattr(a) <- list(format = "%1f")

# Apply format attributes
fapply(a)
Create a format catalog

Description

A format catalog is a collection of formats. A format collection allows you to manage and store formats as a unit. The \texttt{fcat} function defines the format catalog.

Usage

\texttt{fcat(..., log = TRUE)}

Arguments

\texttt{...} \hspace{1cm} A set of formats. Pass the formats as a name/value pair. Multiple name/value pairs are separated by a comma.

\texttt{log} \hspace{1cm} Whether to log the creation of the format catalog. Default is \texttt{TRUE}. This parameter is used internally.

Details

A format catalog is an S3 object of class "fcat". The purpose of the catalog is to combine related formats, and allow you to manipulate all of them as a single object. The format catalog can be saved to/from a file using the \texttt{write.fcat} and \texttt{read.fcat} functions. A format catalog can also be converted to/from a data frame using the \texttt{as.fcat.data.frame} and \texttt{as.data.frame.fcat} functions. Formats are accessed in the catalog using list syntax.

A format catalog can be used to assign formats to a data frame or tibble using the \texttt{formats} function. Formats may be applied using the \texttt{fdata} and \texttt{fapply} functions.

A format catalog may contain any type of format except a formatting list. Allowed formats include a formatting string, a named vector lookup, a user-defined format, and a vectorized formatting function. A formatting list can be converted to a format catalog and saved independently. See the \texttt{flist} function for more information on formatting lists.

Value

The format catalog object.

See Also

\texttt{formats} function for assigning formats to a data frame, and the \texttt{fdata} and \texttt{fapply} functions for applying formats.

Other fcat: \texttt{as.data.frame.fcat()}, \texttt{as.fcat.data.frame()}, \texttt{as.fcat.fmt_lst()}, \texttt{as.fcat.list()}, \texttt{as.fcat()}, \texttt{is.fcat()}, \texttt{print.fcat()}, \texttt{read.fcat()}, \texttt{write.fcat()}
Examples

# Create format catalog
cl <- fcat(num_fmt = "%.1f",
           label_fmt = value(condition(x == "A", "Label A"),
                              condition(x == "B", "Label B"),
                              condition(TRUE, "Other")),
           date_fmt = "%d%b%Y")

# Use formats in the catalog
fapply(2, cl$num_fmt)
fapply(c("A", "B", "C", "B"), cl$label_fmt)
fapply(Sys.Date(), cl$date_fmt)

fdata

Format a data frame or tibble

Description

The fdata function applies formatting attributes to the entire data frame.

Usage

fdata(x, ...)

Arguments

x         A data frame or tibble to be formatted.
...       Any follow-on parameters to the format function.

Details

If formats are assigned to the "format" attributes of the data frame columns, the fdata function will apply those formats to the specified columns, and return a new, formatted data frame. Formats can be specified as formatting strings, named vectors, user-defined formats, or vectorized formatting functions. The fdata function will apply the format to the associated column data using the fapply function. A format can also be specified as a formatting list of the previous four types. See the fapply function for additional information.

After formatting each column, the fdata function will call the base R format function on the data frame. Any follow on parameters will be sent to the format function.

The fdata function will also apply any width or justify attributes assigned to the data frame columns. These attributes can be controlled at the column level. Using attributes to assign formatting and fdata to apply those attributes gives you a great deal of control over how your data is presented.

Value

A new, formatted data frame or tibble with the formats applied.
See Also

fcat to create a format catalog, fapply to apply a format to a vector, value to define a format object, fatttr to assign formatting specifications to a single column/vector, and the formats, widths, and justification functions to get or set formatting for an entire data frame. Also see FormattingStrings for documentation on formatting strings.

Examples

# Construct data frame from state vectors
df <- data.frame(state = state.abb, area = state.area)[1:10, ]

# Calculate percentages
df$pct <- df$area / sum(state.area) * 100

# Before formatting
df

# state area pct
# 1 AL 51609 1.42629378
# 2 AK 589757 16.29883824
# 3 AZ 113909 3.14804973
# 4 AR 53104 1.46761040
# 5 CA 158693 4.38572418
# 6 CO 104247 2.88102556
# 7 CT 5009 0.13843139
# 8 DE 2057 0.05684835
# 9 FL 58560 1.61839532
# 10 GA 58876 1.62712846

# Create state name lookup list
name_lookup <- state.name
names(name_lookup) <- state.abb

# Assign formats
formats(df) <- list(state = name_lookup,
                     area = function(x) format(x, big.mark = ","),
                     pct = "%.1f%%")

# Apply formats
fdata(df)

# state area pct
# 1 Alabama 51,609 1.4%
# 2 Alaska 589,757 16.3%
# 3 Arizona 113,909 3.1%
# 4 Arkansas 53,104 1.5%
# 5 California 158,693 4.4%
# 6 Colorado 104,247 2.9%
# 7 Connecticut 5,009 0.1%
# 8 Delaware 2,057 0.1%
# 9 Florida 58,560 1.6%
# 10 Georgia 58,876 1.6%
Create a formatting list

**Description**

A formatting list contains more than one formatting object.

**Usage**

```r
flist(..., type = "column", lookup = NULL, simplify = TRUE)
```

**Arguments**

- `...`: A set of formatting objects.
- `type`: The type of formatting list. Valid values are 'row' or 'column'. The default value is 'column'.
- `lookup`: A lookup vector. Used for looking up the format from the formatting list. This parameter is only used for 'row' type formatting lists.
- `simplify`: Whether to simplify the results to a vector. Valid values are TRUE or FALSE. Default is TRUE. If the value is set to FALSE, the return type will be a list.

**Details**

To apply more than one formatting object to a vector, use a formatting list. There are two types of formatting list: column and row. The column type formatting lists applies all formats to all values in the vector. The row type formatting list can apply a different format to each value in the vector.

Further, there are two styles of row type list: ordered and lookup. The ordered style applies each format in the list to the vector values in the order specified. The ordered style will recycle the formats as needed. The lookup style formatting list uses a lookup to determine which format from the list to apply to a particular value of the vector. The lookup column values should correspond to names on the formatting list.

Examples of column type and row type formatting lists are given below.

**Value**

A vector or list of formatted values. The type of return value can be controlled with the `simplify` parameter. The default return type is a vector.

**See Also**

- `fapply` for information on how formats are applied to a vector, `value` for how to create a user-defined format, and `as.flist` to convert an existing list of formats to a formatting list. Also see `FormattingStrings` for details on how to use formatting strings.

Other `flist`: `as.data.frame.fmt_lst()`, `as.flist.data.frame()`, `as.flist.fcat()`, `as.flist.list()`, `as.flist.tbl_df()`, `as.flist()`, `is.flist()`, `print.fmt_lst()`
Examples

## Example 1: Formatting List - Column Type ##

```r
# Set up data
v1 <- c(Sys.Date(), Sys.Date() + 30, Sys.Date() + 60)

# Create formatting list
fl1 <- flist("%B", "The month is: %s")

# Apply formatting list to vector
fapply(v1, fl1)

# [1] "The month is: October" "The month is: November" "The month is: December"
```

## Example 2: Formatting List - Row Type ordered ##

```r
# Set up data
# Notice each row has a different data type
l1 <- list("A", 1.263, as.Date("2020-07-21"),
            "B", 5.8732, as.Date("2020-10-17"))

# These formats will be recycled in the order specified
fl2 <- flist(type = "row",
c(A = "Label A", B = "Label B"),
"%.1f",
"%d%b%Y")
fapply(l1, fl2)

# [1] "Label A" "1.3" "21Jul2020" "Label B" "5.9" "17Oct2020"
```

## Example 3: Formatting List - Row Type with lookup ##

```r
# Set up data
# Notice each row has a different data type
l2 <- list(2841.258, "H", as.Date("2020-06-19"),
            "L", as.Date("2020-04-24"), 1382.8865)
v3 <- c("num", "char", "date", "char", "date", "num")

# Create formatting list
fl3 <- flist(type = "row", lookup = v3,
             num = function(x) format(x, digits = 2, nsmall = 1,
                                      big.mark=""),
             char = value(condition(x == "H", "High"),
                           condition(x == "L", "Low"),
                           condition(TRUE, "NA"),
             date = "%d%b%Y")

# Apply formatting list to vector, using lookup
fapply(l2, fl3)

# [1] "2,841.3" "High" "19Jun2020" "Low" "24Apr2020" "1,382.9"
```
Description

The \texttt{fmtr} package contains functions to format vectors and data frames.

Details

The \texttt{fmtr} package helps format data frames, similar to the way datasets are formatted in SAS®. Formats are assigned to columns using a \texttt{format} attribute. Formats are then applied by calling the \texttt{fdata} function on the data frame. The \texttt{fdata} function returns a new data frame with the formats applied to each column as specified.

All functions in the \texttt{fmtr} package also work with tidyverse tibbles.

The \texttt{fmtr} package also contains a function named \texttt{value} for defining user-defined formats in a manner similar to SAS® formats.

Functions

The main functions included in the \texttt{fmtr} package are as follows:

- \texttt{fdata}: Applies formatting to a dataframe.
- \texttt{fapply}: Applies formatting to a vector.
- \texttt{fcat}: Creates a format catalog.
- \texttt{value}: Creates a new user-defined format.
- \texttt{formats}: Helps assign format attributes to a data frame.
- \texttt{fattr}: Helps assign formatting attributes to a vector.

Available Formats

The formats that can be used with \texttt{fmtr} include the following:

- Formatting strings
- Named vectors
- Vectorized functions
- User-defined formats
- Formatting lists

A formatting string is a compact set of codes typically used for formatting dates and numbers. See \texttt{FormattingStrings} for a glossary of common formatting codes.

Named vectors map one string value to another string value.

Vectorized functions can be those created by the user or by a formatting function like those created with Base R or the \texttt{scales} package.
User-defined functions are those created by the fmtr value function. See value for additional details.

Formatting lists are lists which contain any of the above format types. The formatting lists allow the user to apply one of several formats to the column. Formatting lists may also be used to apply different formats to each item in a vector. See flist for additional details on formatting lists.

See the fapply function documentation for additional details on applying formats.

Formats can be collected and stored as a single file, called a format catalog. This functionality makes it easy to reuse formats in new programs, with new data, or to share them with colleagues. See the fcat function for additional information.

---

fmt_cnt_pct  Formatted count and percent

**Description**

A function to calculate and format a count and percent.

**Usage**

fmt_cnt_pct(x, denom = NULL, format = "%5.1f", na = NULL, zero = NULL)

**Arguments**

- `x`: The input data vector or data frame column.
- `denom`: The denominator to use for the percentage. By default, the parameter is NULL, meaning the function will use the number of non-missing values of the data vector as the denominator. Otherwise, supply the denominator as a numeric value.
- `format`: A formatting string suitable for input into the sprintf function. By default, this format is defined as "%5.1f", which displays the value with one decimal place.
- `na`: The value to return for any NA value encountered in the input vector. Usually this parameter is passed as a string, such as "-", yet any value can be supplied.
- `zero`: The value to return for any zero values encountered in the input vector. Usually this value is supplied as string such as "0 (-)".

**Details**

This function calculates a percent and appends to the provided count. The input vector is assumed to contain the counts. This function will not perform counting. It will calculate percentages and append to the given counts.

The result is then formatted using sprintf. By default, the number of non-missing values in the input data vector is used as the denominator. Alternatively, you may supply the denominator using the `denom` parameter. You may also control the percent format using the `format` parameter. The function will return any NA values in the input data unaltered.

If the calculated percentage is between 0% and 1%, the function will display "(< 1.0%)" as the percentage value. Zero values will be displayed as "(0.0%)"
**fmt_mean_sd**

**Value**

A character vector of formatted counts and percents.

**See Also**

Other helpers: `fmt_mean_sd()`, `fmt_median()`, `fmt_n()`, `fmt_quantile_range()`, `fmt_range()`

**Examples**

```r
v1 <- c(4, 3, 8, 6, 9, NA, 0, 7, 4, 3, 7)

# Format count and percent
fmt_cnt_pct(v1)

# Output
# [1] "4 (36.4%)" "3 (27.3%)" "8 (72.7%)" "6 (54.5%)"
# [5] "9 (81.8%)" "5 (45.5%)" NA "0 (0.0%)"
# [9] "7 (63.6%)" "4 (36.4%)" "3 (27.3%)" "7 (63.6%)"

# Custom values for NA and zero
fmt_cnt_pct(v1, na = "N/A", zero = "0 (-)"

# Custom NA and zero output
# [1] "4 (36.4%)" "3 (27.3%)" "8 (72.7%)" "6 (54.5%)"
# [5] "9 (81.8%)" "5 (45.5%)" "N/A" "0 (-)"
# [9] "7 (63.6%)" "4 (36.4%)" "3 (27.3%)" "7 (63.6%)"
```

---

**fmt_mean_sd**

**Formatted mean and standard deviation**

**Description**

A function to calculate and format a mean and standard deviation.

**Usage**

```r
fmt_mean_sd(x, format = "%.1f", sd_format = NULL)
```

**Arguments**

- `x`: The input data vector or data frame column.
- `format`: A formatting string suitable for input into the `sprintf` function. By default, this format is defined as "%.1f", which displays the mean and standard deviation with one decimal place.
- `sd_format`: An optional format for the standard deviation. If this parameter is not supplied, the standard deviation will be formatted the same as the mean, according to the 'format' parameter.
fmt_median

Details

This function calculates a mean and standard deviation, and formats using `sprintf`. You may control the format using the `format` parameter. Function will ignore NA values in the input data. Results are returned as a character vector.

Value

The formatted mean and standard deviation.

See Also

Other helpers: `fmt_cnt_pct()`, `fmt_median()`, `fmt_n()`, `fmt_quantile_range()`, `fmt_range()`

Examples

```r
v1 <- c(4.3, 3.7, 8.7, 6.1, 9.2, 5.6, NA, 0.7, 7.8, 4.9)
# Format mean and standard deviation
fmt_mean_sd(v1)
# Output
# "5.7 (2.7)"
```

fmt_median

Formatted Median

Description

A function to calculate and format a median.

Usage

```r
fmt_median(x, format = "%.1f")
```

Arguments

- `x`: The input data vector or data frame column.
- `format`: A formatting string suitable for input into the `sprintf` function. By default, this format is defined as "%.1f", which displays the value with one decimal place.

Details

This function calculates a median using the stats package `median` function, and then formats the output using `sprintf`. You may control the format using the `format` parameter. Function will ignore any NA values in the input data. Results are returned as a character vector.

Value

The formatted median value.
See Also

Other helpers: `fmt_cnt_pct()`, `fmt_mean_sd()`, `fmt_n()`, `fmt_quantile_range()`, `fmt_range()`

Examples

```r
v1 <- c(4.3, 3.7, 8.7, 6.1, 9.2, 5.6, NA, 0.7, 7.8, 4.9)

# Format median
fmt_median(v1)

# Output
# "5.6"
```

---

### fmt_n

**Formatted Count**

Description

A function to calculate and format a numeric count.

Usage

```r
fmt_n(x)
```

Arguments

- `x` The input data vector or data frame column.

Details

This function calculates a count using the Base R `sum` function. NA values are not counted. Results are returned as a character vector.

Value

The formatted count value.

See Also

Other helpers: `fmt_cnt_pct()`, `fmt_mean_sd()`, `fmt_median()`, `fmt_quantile_range()`, `fmt_range()`
Examples

```r
# Create example vector
v1 <- c(4.3, 3.7, 8.7, 6.1, 9.2, 5.6, NA, 0.7, 7.8, 4.9)

# Format n
fmt_n(v1)
```

```r
# Output
# "9"
```

fmt_quantile_range  Formatted Quantile Range

Description
A function to calculate and format a quantile range.

Usage
```r
fmt_quantile_range(
  x,  
  format = "%.1f",
  sep = "-",
  lower = 0.25,
  upper = 0.75,
  type = 7
)
```

Arguments
- `x`: The input data vector or data frame column.
- `format`: A formatting string suitable for input into the `sprintf` function. By default, this format is defined as "%.1f", which displays the value with one decimal place.
- `sep`: The character to use as a separator between the two quantiles.
- `lower`: The lower quantile range. Default is .25.
- `upper`: The upper quantile range. Default is .75.
- `type`: An integer between 1 and 9 selecting one of the nine quantile algorithms. The default is 7, which is the standard R default. If you are trying to match SAS results, use type 2. See the `quantile` function documentation for further details.

Details
This function calculates a quantile range using the stats package `quantile` function, and then formats the output using `sprintf`. You may control the format using the `format` parameter. Function will ignore any NA values in the input data. Results are returned as a character vector.

By default, the function calculates the 1st and 3rd quantiles at .25 and .75. The upper and lower quantile ranges may be changed with the `upper` and `lower` parameters.
Value

The formatted quantile range.

See Also

Other helpers: fmt_cnt_pct(), fmt_mean_sd(), fmt_median(), fmt_n(), fmt_range()

Examples

# Create example vector
v1 <- c(4.3, 3.7, 8.7, 6.1, 9.2, 5.6, NA, 0.7, 7.8, 4.9)

# Format Quantiles
fmt_quantile_range(v1)

# Output
# "4.3 - 7.8"

---

fmt_range

Formatted Range

Description

A function to calculate and format a numeric range.

Usage

fmt_range(x, format = "%s", sep = "-")

Arguments

x

The input data vector or data frame column.

format

A formatting string suitable for input into the sprintf function. By default, this format is defined as "%s", which simply converts the value to a string with no specific formatting.

sep

The token used to separate the minimum and maximum range values. Default value is a hyphen ("-").

Details

This function calculates a range using the Base R range function, and then formats the output using sprintf. You may control the format using the format parameter. Any NA values in the input data are ignored. Results are returned as a character vector.

Value

The formatted range values.
See Also
Other helpers: \texttt{fmt\_cnt\_pct()}, \texttt{fmt\_mean\_sd()}, \texttt{fmt\_median()}, \texttt{fmt\_n()}, \texttt{fmt\_quantile\_range()}

Examples

\begin{verbatim}
# Create example vector
v1 <- c(4.3, 3.7, 8.7, 6.1, 9.2, 5.6, NA, 0.7, 7.8, 4.9)

# Format range
fmt_range(v1)

# Output
# "0.7 - 9.2"
\end{verbatim}

\textbf{Description}

The \texttt{formats} function extracts all assigned formats from a data frame, and returns them in a named list. The function also assigns formats from a named list.

\textbf{Usage}

\begin{verbatim}
formats(x)

formats(x) <- value
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{1cm} A data frame or tibble
\item \texttt{value} \hspace{1cm} A named list of formats
\end{itemize}

\textbf{Details}

If formats are assigned to the "format" attributes of the data frame columns, the \texttt{formats} function will extract those formats. The function will return the formats in a named list, where the names correspond to the name of the column that the format was assigned to. If a column does not have a format attribute assigned, that column will not be included in the list.

When used on the receiving side of an assignment, the function will assign formats to a data frame. The formats should be in a named list, where each name corresponds to the data frame column to assign the format to.

The \texttt{formats} function can also accept a format catalog as input. This feature allows you to save formats in metadata, load them into a format catalog, and quickly assign them to a data frame or tibble. See the \texttt{fcat} function for additional information.

Finally, if you wish to clear out format attributes, assign a NULL value to the \texttt{formats} function.
Value

A named list of formats.

See Also

fdata to display formatted data, value to create user-defined formats, fapply to apply formats to a vector, and fcat to create a format catalog. Also see FormattingStrings for documentation on formatting strings.

Examples

# Take subset of data
df1 <- mtcars[1:10, c("mpg", "cyl")]

# Print current state
print(df1)

# Assign formats
attr(df1$mpg, "format") <- value(condition(x >= 20, "High"),
condition(x < 20, "Low"))
attr(df1$cyl, "format") <- function(x) format(x, nsmall = 1)

# Display formatted data
fdata(df1)

# Extract format list
lst <- formats(df1)

# Alter format list and reassign
lst$mpg <- value(condition(x >= 22, "High"),
condition(x < 22, "Low"))
lst$cyl <- function(x) format(x, nsmall = 2)
formats(df1) <- lst

# Display formatted data
fdata(df1)

# Clear formats
formats(df1) <- NULL

# Confirm formats are cleared
formats(df1)

FormattingStrings  Formatting Strings

Description

Formatting codes for formatting strings follow the conventions of the base R strptime and sprintf functions. See below for further details.
Details

The `fmtr` packages accepts single strings as formatting specifications. These formatting strings are interpreted differently depending on the data type of the vector being formatted. For date and datetime vectors, the string will be interpreted as an input to the base R `strptime` function. For all other types of vectors, the formatting string will be interpreted as an input to the `sprintf` function.

The formatting codes for these functions are simple to use. For example, the code `fapply(as.Date("1970-01-01"), "%B %d, %Y")` will produce the output "January 01, 1970". The code `fapply(1.2345, "%.1f")` will produce the output "1.2".

Below are some commonly used formatting codes for dates:

- `%d` = day as a number
- `%a` = abbreviated weekday
- `%A` = unabbreviated weekday
- `%m` = month
- `%b` = abbreviated month
- `%B` = unabbreviated month
- `%y` = 2-digit year
- `%Y` = 4-digit year
- `%H` = hour
- `%M` = minute
- `%S` = second
- `%p` = AM/PM indicator

See the `strptime` function for additional codes and examples of formatting dates and times.

Below are some commonly used formatting codes for other data types:

- `%s` = string
- `%d` = integer
- `%f` = decimal number

See the `sprintf` function for additional codes and examples of formatting other data types.

See Also

`fapply` for formatting vectors, and `fdata` for formatting data frames.

Examples

```r
# Examples for formatting dates and times
set.seed(42)
t <- Sys.time()
fapply(t, "%d/%m/%Y")  # Day/Month/Year
fapply(t, "%d%b%Y")  # Day abbreviated month year
fapply(t, "%y-%m")  # Two digit year - month
fapply(t, "%A, %B %d")  # Weekday, unabbreviated month and date
fapply(t, "%Y-%m-%d %H:%M:%S %p")  # Common timestamp format
```
is.fcat

# Examples for formatting numbers
a <- 1234.56789
fapply(a, "%.1f") # Floating point number
fapply(a, "%8.1f") # Fixed width
fapply(a, "%-8.1f") # Fixed width left justified
fapply(a, "%%8.1f") # Zero padded
fapply(a, "%+.1f") # Forced sign
fapply(-a, "%+.1f") # Negative
fapply(a, "%.1f%%") # Percentage
fapply(a, "$%.2f") # Currency
fapply(a, "The number is %f.") # Interpolation

## is.fcat

Class test for a format catalog

Description

This function tests whether an object is a format catalog. The format catalog has a class of "fcat".

Usage

is.fcat(x)

Arguments

x The object to test.

Value

TRUE or FALSE, depending on whether or not the object is a format catalog.

See Also

Other fcat: as.data.frame.fcat(), as.fcat.data.frame(), as.fcat.fmt_lst(), as.fcat.list(), as.fcat(), fcat(), print.fcat(), read.fcat(), write.fcat()

Examples

# Create format catalog
c1 <- fcat(num_fmt = "%+.1f",
    label_fmt = value(condition(x == "A", "Label A"),
    condition(x == "B", "Label B"),
    condition(TRUE, "Other")),
    date_fmt = "%d%b%Y")

# Test for "fcat" class
is.fcat(c1)
is.fcat(Sys.Date())
is.flist  

*Is object a formatting list*

### Description

Determines if object is a formatting list of class `fmt_lst`.

### Usage

```r
is.flist(x)
```

### Arguments

- `x`  
  Object to test.

### Value

TRUE or FALSE, depending on class of object.

### See Also

Other flist: `as.data.frame.fmt_lst()`, `as.flist.data.frame()`, `as.flist.fcat()`, `as.flist.list()`, `as.flist.tbl_df()`, `as.flist().flist()`, `print.fmt_lst()`

### Examples

```r
# Create flist
flst <- flist("%d%b%Y", "%f")
is.flist(flst)
is.flist("A")
```

is.format  

*Determine whether an object is a user-defined format*

### Description

The `is.format` function can be used to determine if an object is a user-defined format of class `"fmt"`.

### Usage

```r
is.format(x)
```

### Arguments

- `x`  
  A user-defined format of class `"fmt"`.  

Details

The `is.format` function returns TRUE if the object passed is a user-defined format. User-defined formats are defined using the `value` function. See the `value` function help for further details.

Value

A logical value or TRUE or FALSE.

See Also

`value` to define a format, `condition` to define the conditions for a format, and `fapply` to apply the format to a vector.

Other fmt: `as.data.frame.fmt()`, `as.fmt.data.frame()`, `as.fmt()`, `condition()`, `labels.fmt()`, `print.fmt()`, `value()

Examples

```r
# Define format
fmt1 <- value(condition(x == "A", "Label A"),
               condition(x == "B", "Label B"),
               condition(TRUE, "Other"))

# Check for format
is.format(fmt1)
is.format("A")
```

justification

Get or set justification for data frame columns

Description

The `justification` function extracts all assigned justify attributes from a data frame, and returns them in a named list. The function also assigns justify attributes from a named list.

Usage

```r
justification(x)

justification(x) <- value
```

Arguments

- `x` A data frame or tibble
- `value` A named list of justification values.
Details

If justification values are assigned to the "justify" attributes of the data frame columns, the justification function will extract those values. The function will return the justification values in a named list, where the names correspond to the name of the column that the justification was assigned to. If a column does not have a justify attribute assigned, that column will not be included in the list.

When used on the receiving side of an assignment, the function will assign justification to a data frame. The justification values should be in a named list, where each name corresponds to the name of the data frame column to assign values to.

Finally, if you wish to clear out the justification attributes, assign a NULL value to the justification function.

Value

A named list of justification values.

See Also

fdata to display formatted data, value to create user-defined formats, and fapply to apply formatting to a vector.

Examples

# Take subset of data
df1 <- mtcars[1:10, c("mpg", "cyl") ]

# Print current state
print(df1)

# Assign justification
justification(df1) <- list(mpg = "left", cyl = "right")
widths(df1) <- list(mpg = 12, cyl = 10)

fdata(df1)

# Display justification
justification(df1)
widths(df1)

# Clear justification
justification(df1) <- NULL

# Confirm justifications are cleared
justification(df1)
Description
The `labels` function creates a vector of labels associated with a user-defined format.

Usage
```r
## S3 method for class 'fmt'
labels(object, ...)
```

Arguments
- `object` A user-defined format of class "fmt".
- `...` Following arguments.

Details
The `condition` function creates a condition as part of a format definition. Each condition has a label as part of its definition. The `labels` function extracts the labels from the conditions and returns them as a vector. While the labels will typically be of type character, they can be of any data type. See the link(`condition`) function help for further details.

Value
A vector of label values.

See Also
- `value` to define a format, `condition` to define the conditions for a format, and `fapply` to apply the format to a vector.

Other fmt: `as.data.frame.fmt()`, `as.fmt.data.frame()`, `as.fmt()`, `condition()`, `is.format()`, `print.fmt()`, `value()`

Examples
```r
# Define format
fmt1 <- value(condition(x == "A", "Label A"),
               condition(x == "B", "Label B"),
               condition(TRUE, "Other"))

# Extract labels
labels(fmt1)
```
print.fcat  
*Print a format catalog*

**Description**

A class-specific instance of the `print` function for format catalogs. The function prints the format catalog in a tabular manner. Use `verbose = TRUE` to print the catalog as a list.

**Usage**

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

**Arguments**

- `x`  
The format catalog to print.
- `...`  
Any follow-on parameters.
- `verbose`  
Whether or not to print the format catalog in verbose style. By default, the parameter is `FALSE`, meaning to print in tabular style.

**Value**

The object, invisibly.

**See Also**

Other `fcat`: `as.data.frame.fcat()`, `as.fcat.data.frame()`, `as.fcat.fmt_lst()`, `as.fcat.list()`, `as.fcat()`, `fcat()`, `is.fcat()`, `read.fcat()`, `write.fcat()`

**Examples**

```r
# Create format catalog
cl <- fcat(num_fmt = "%.1f",
           label_fmt = value(condition(x == "A", "Label A"),
                              condition(x == "B", "Label B"),
                              condition(TRUE, "Other")),
           date_fmt = "%d/%b/%Y")

# Print the catalog
print(cl)
```
print.fmt  

Print a format

Description

Prints a format object. This function is a class-specific implementation of the the generic print method.

Usage

## S3 method for class 'fmt'
print(
  x,
  ..., 
  name = deparse(substitute(x, env = environment())),
  verbose = FALSE
)

Arguments

x  An object of class "fmt".
...
name  The name of the format to print. By default, the variable name that holds the format will be used.
verbose  Turn on or off verbose printing mode. Verbose mode will print object as a list. Otherwise, the object will be printed as a table.

See Also

Other fmt: as.data.frame.fmt(), as.fmt.data.frame(), as.fmt(), condition(), is.format(), labels.fmt(), value()

print.fmt_lst  

Print a formatting list

Description

Print a formatting list

Usage

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

x  The formatting list to print
...  Follow-on parameters to the print function
verbose  Whether to print in summary or list-style.

See Also

Other fcat: as.data.frame.fmt_lst(), as.flist.data.frame(), as.flist.fcat(), as.flist.list(), as.flist.tbl_df(), as.flist.flist(), is.flist()

read.fcat  Read a format catalog from the file system

Description

The read.fcat function reads a format catalog from the file system. The function accepts a path to the format catalog, reads the catalog, and returns it.

Note that the format catalog is saved as an RDS file. The ".fcat" file extension only serves to distinguish the format catalog from other RDS files.

Usage

read.fcat(file_path)

Arguments

file_path  The path to the format catalog.

Value

The format catalog as an R object.

See Also

Other fcat: as.data.frame.fcat(), as.fcat.data.frame(), as.fcat.fmt_lst(), as.fcat.list(), as.fcat(), fcat(), is.fcat(), print.fcat(), write.fcat()

Examples

# Create format catalog
c1 <- fcat(num_fmt = "%.1f",
           label_fmt = value(condition(x == "A", "Label A"),
                               condition(x == "B", "Label B"),
                               condition(TRUE, "Other")),
           date_fmt = "%d%b%Y")

# Get temp directory
value <- tempdir()

# Save catalog to file system
pth <- write.fcat(c1, dir_path = tmp)

# Read from file system
c2 <- read.fcat(pth)

# Use formats in the catalog
fapply(2, c1$num_fmt)
fapply(c("A", "B", "C", "B"), c1$label_fmt)
fapply(Sys.Date(), c1$date_fmt)

---

value | Create a user-defined format

Description

The value function creates a user-defined format.

Usage

value(..., log = TRUE)

Arguments

... | One or more condition functions.
log | Whether to log the creation of the format. Default is TRUE. This parameter is used internally.

Details

The value function creates a user-defined format object, in a manner similar to a SAS® format. The value function accepts one or more condition arguments that define the format. The conditions map an R expression to a label. When applied, the format will return the label corresponding to the first true expression.

The format object is an S3 class of type "fmt". When the object is created, the levels attribute of the object will be set with a vector of values assigned to the labels property of the condition arguments. These labels may be accessed either from the levels function or the labels function. If no order has been assigned to the conditions, the labels will be returned in the order the conditions were passed to the value function. If an order has been assigned to the conditions, the labels will be returned in the order specified.

The format object may be applied to a vector using the fapply function. See fapply for further details.

Note that the label may also be a string format. That means a user-defined format can be used to apply string formats conditionally. This capability is useful when you want to conditionally format data values.
Value
The new format object.

See Also
condition to define a condition, levels or labels.fmt to access the labels, and fapply to apply the format to a vector.

Other fmt: as.data.frame.fmt(), as.fmt.data.frame(), as.fmt(), condition(), is.format(), labels.fmt(), print.fmt()

Examples

## Example 1: Character to Character Mapping ##
# Set up vector
v1 <- c("A", "B", "C", "B")

# Define format
fmt1 <- value(condition(x == "A", "Label A"),
                condition(x == "B", "Label B"),
                condition(TRUE, "Other"))

# Apply format to vector
fapply(v1, fmt1)

## Example 2: Character to Integer Mapping ##
fmt2 <- value(condition(x == "A", 1),
               condition(x == "B", 2),
               condition(TRUE, 3))

# Apply format to vector
fapply(v1, fmt2)

## Example 3: Categorization of Continuous Variable ##
# Set up vector
v2 <- c(1, 6, 11, 7)

# Define format
fmt3 <- value(condition(x < 5, "Low"),
               condition(x >= 5 & x < 10, "High"),
               condition(TRUE, "Out of range"))

# Apply format to vector
fapply(v2, fmt3)

## Example 4: Conditional formatting
v3 <- c(10.398873, 12.98762, 0.5654, 11.588372)
fmt4 <- value(condition(x < 1, "< 1.0"),
               condition(x >= 10.5, "High"),
               condition(TRUE, "Out of range"))

# Apply format to vector
fapply(v3, fmt4)
widths

condition(TRUE, "%.2f")

fapply(v3, fmt4)
# [1] "10.40" "12.99" "< 1.0" "11.59"

---

**widths**

*Get or set column widths for a data frame*

**Description**

The `widths` function extracts all assigned widths from a data frame, and returns them in a named list. The function also assigns widths from a named list.

**Usage**

```r
widths(x)
widths(x) <- value
```

**Arguments**

- `x` A data frame or tibble
- `value` A named list of widths. The widths must be positive integers greater than zero.

**Details**

If widths are assigned to the "width" attributes of the data frame columns, the `widths` function will extract those widths. The function will return the widths in a named list, where the names correspond to the name of the column that the width was assigned to. If a column does not have a width attribute assigned, that column will not be included in the list.

When used on the receiving side of an assignment, the function will assign widths to a data frame. The widths should be in a named list, where each name corresponds to the data frame column to assign the width to.

Finally, if you wish to clear out the width attributes, assign a NULL value to the `widths` function.

**Value**

A named list of widths. The widths must be positive integers greater than zero.

**See Also**

`fdata` to display formatted data, `value` to create user-defined formats, and `fapply` to apply formats to a vector.
Examples

```r
# Take subset of data
df1 <- mtcars[1:10, c("mpg", "cyl")]

# Print current state
print(df1)

# Assign widths
widths(df1) <- list(mpg = 12, cyl = 10)

# Display formatted data
fdata(df1)

# View assigned widths
widths(df1)

# Clear widths
widths(df1) <- NULL

# Confirm widths are cleared
widths(df1)
```

**Description**

The `write.fcat` function writes a format catalog to the file system. By default, the catalog will be written to the current working directory, using the variable name as the file name. These defaults can be overridden using the appropriate parameters. The catalog will be saved with a file extension of ".fcat".

Note that the format catalog is saved as an RDS file. The ".fcat" file extension only serves to distinguish the format catalog from other RDS files.

**Usage**

```r
write.fcat(x, dir_path = getwd(), file_name = NULL)
```

**Arguments**

- `x` The format catalog to write.
- `dir_path` The directory path to write the catalog to. Default is the current working directory.
- `file_name` The name of the file to save the catalog as. Default is the name of the variable that contains the catalog. The ".fcat" file extension will be added automatically.
Value

The full path of the saved format catalog.

See Also

Other fcat: `as.data.frame.fcat()`, `as.fcat.data.frame()`, `as.fcat.fmt_lst()`, `as.fcat.list()`, `as.fcat()`, `fcat()`, `is.fcat()`, `print.fcat()`, `read.fcat()`

Examples

```r
# Create format catalog
cl <- fcat(num_fmt = "%.1f",
           label_fmt = value(condition(x == "A", "Label A"),
                               condition(x == "B", "Label B"),
                               condition(TRUE, "Other")),
           date_fmt = "%d%b%Y")

# Get temp directory
tmp <- tempdir()

# Save catalog to file system
pth <- write.fcat(cl, dir_path = tmp)

# Read from file system
c2 <- read.fcat(pth)

# Use formats in the catalog
fapply(2, cl$num_fmt)
fapply(c("A", "B", "C", "B"), cl$label_fmt)
fapply(Sys.Date(), cl$date_fmt)
```
Index

* fcat
  as.data.frame.fcat, 3
  as.fcat, 6
  as.fcat.data.frame, 6
  as.fcat.fmt_lst, 8
  as.fcat.list, 9
  fcat, 32
  is.fcat, 47
  print.fcat, 52
  read.fcat, 54
  write.fcat, 58

* flist
  as.data.frame.fmt_lst, 5
  as.flist, 11
  as.flist.data.frame, 13
  as.flist.fcat, 15
  as.flist.list, 17
  as.flist.tbl_df, 18
  flist, 35
  is.flist, 48
  print.fmt_lst, 53

* fmt
  as.data.frame.fmt, 4
  as.fmt, 20
  as.fmt.data.frame, 21
  condition, 22
  is.format, 48
  labels.fmt, 51
  print.fmt, 53
  value, 55

* helpers
  fmt_cnt_pct, 38
  fmt_mean_sd, 39
  fmt_median, 40
  fmt_n, 41
  fmt_quantile_range, 42
  fmt_range, 43

  as.data.frame.fcat, 3, 6, 7, 9, 10, 32, 47, 52, 54, 59

  as.data.frame.fmt, 4, 21–23, 49, 51, 53, 56
  as.data.frame.fmt_lst, 5, 12, 14, 16, 18, 19, 35, 48, 54
  as.fcat, 3, 6, 7, 9, 10, 32, 47, 52, 54, 59
  as.fcat.data.frame, 3, 6, 6, 9, 32, 47, 52, 54, 59
  as.fcat.fmt_lst, 3, 6, 7, 8, 9, 10, 32, 47, 52, 54, 59
  as.fcat.list, 3, 6, 7, 9, 9, 10, 32, 47, 52, 54, 59
  as.fcat.tbl_df, 9
  as.flist, 5, 11, 14, 16, 18, 19, 35, 48, 54
  as.flist.data.frame, 5, 12, 13, 16, 18, 19, 35, 48, 54
  as.flist.fcat, 5, 12, 14, 15, 18, 19, 35, 48, 54
  as.flist.list, 5, 12, 14, 16, 17, 19, 35, 48, 54
  as.flist.tbl_df, 5, 12, 14, 16, 18, 18, 35, 48, 54
  as.fmt, 4, 20, 22, 23, 49, 51, 53, 56
  as.fmt.data.frame, 4, 21, 21, 23, 49, 51, 53, 56
  condition, 4, 21, 22, 22, 22, 49, 51, 53, 55, 56
  descriptions, 24
  descriptions<- (descriptions), 24
  fapply, 23, 25, 25, 29–35, 37, 38, 45, 46, 49–51, 55–57
  fapply2, 28
  fattr, 26, 29, 29, 31, 34, 37
  fattr<-, 31
  fcat, 3, 6, 7, 9, 10, 26, 29, 32, 34, 37, 38, 44, 45, 47, 52, 54, 59
  fdata, 23, 25, 26, 29–32, 33, 37, 45, 46, 50, 57
  flist, 5, 8, 12, 14, 16, 18, 19, 26, 29, 32, 35, 38, 48, 54
  fmt_cnt_pct, 38, 40, 41, 43, 44
fmt_mean_sd, 39, 39, 41, 43, 44
fmt_median, 39, 40, 40, 41, 43, 44
fmt_n, 39–41, 41, 43, 44
fmt_quantile_range, 39–41, 42, 44
fmt_range, 39–41, 43, 44
fmtr, 37
format, 33
formats, 32, 34, 37, 44
formats<- (formats), 44
FormattingStrings, 7, 10, 22, 26, 29, 30, 34, 35, 37, 45, 46
is.fcat, 3, 6, 7, 9, 10, 32, 47, 52, 54, 59
is.flist, 5, 12, 14, 16, 18, 19, 35, 48, 54
is.format, 4, 21–23, 48, 51, 53, 56
justification, 34, 49
justification<- (justification), 49
labels.fmt, 4, 21–23, 49, 51, 53, 56
levels, 23, 56
median, 40
print.fcat, 3, 6, 7, 9, 10, 32, 47, 52, 54, 59
print.fmt, 4, 21–23, 49, 51, 53, 56
print.fmt_lst, 5, 12, 14, 16, 18, 19, 35, 48, 53
quantile, 42
range, 43
read.fcat, 3, 6, 7, 9, 10, 32, 47, 52, 54, 59
sprintf, 38–40, 42, 43, 45, 46
strptime, 45, 46
sum, 41
value, 4, 7, 10, 21–23, 25, 26, 29, 34, 35, 37, 38, 45, 49–51, 53, 55, 57
Vectorize, 26
widths, 34, 57
widths<- (widths), 57
write.fcat, 3, 6, 7, 9, 10, 32, 47, 52, 54, 58