Package ‘dfidx’

October 13, 2022

Version 0.0-5
Date 2022-06-26
Title Indexed Data Frames
Depends R (>= 2.10)
Imports dplyr, Formula
Suggests knitr, rmarkdown, AER, mlogit, plm
Description Provides extended data frames, with a special data frame column which contains two indexes, with potentially a nesting structure.
License GPL (>= 2)
URL https://cran.r-project.org/package=dfidx
VignetteBuilder knitr
RoxygenNote 7.1.2
Encoding UTF-8
NeedsCompilation no
Author Yves Croissant [aut, cre]
Maintainer Yves Croissant <yves.croissant@univ-reunion.fr>
Repository CRAN
Date/Publication 2022-06-28 12:00:02 UTC

R topics documented:

dfdx ......................................................... 2
dplyr ....................................................... 4
idx .......................................................... 5
idx_name .................................................... 6
methods.dfdx ............................................. 8
model.frame.dfdx ....................................... 9
unfold_idx ............................................. 11

Index 12
### dfidx

**Data frames with indexes**

#### Description

data frames for which observations are defined by two (potentially nested) indexes and for which series have therefore a natural tabular representation

#### Usage

```r
defidx(
  data,
  idx = NULL,
  drop.index = TRUE,
  as.factor = NULL,
  pkg = NULL,
  fancy.row.names = FALSE,
  subset = NULL,
  idnames = NULL,
  shape = c("long", "wide"),
  choice = NULL,
  varying = NULL,
  sep = ".",
  opposite = NULL,
  levels = NULL,
  ranked = FALSE,
  ...
)
```

#### Arguments

- `data` : a data frame
- `idx` : an index
- `drop.index` : if TRUE (the default), remove the index series from the data.frame as stand alone series
- `as.factor` : should the indexes be coerced to factors?
- `pkg` : if set, the resulting dfidx object is of class c("dfidx_pkg", "dfidx") which enables to write specific classes
- `fancy.row.names` : if TRUE, fancy row names are computed
- `subset` : a logical which defines a subset of rows to return
- `idnames` : the names of the indexes
- `shape` : either wide or long
- `choice` : the choice
dfidx

varying, sep relevant for data sets in wide format, these arguments are passed to reshape
opposite return the opposite of the series
levels the levels for the second index
ranked a boolean for ranked data
...

Details
Indexes are stored as a data.frame column in the resulting dfidx object

Value
an object of class "dfidx"

Author(s)
Yves Croissant

Examples
if (requireNamespace("AER")){
  data("TravelMode", package = "AER")
  # the first two columns contain the index
  TM1 <- dfidx(TravelMode)
  # explicitly indicate the two indexes using either a vector or a
  # list of two characters
  TM2 <- dfidx(TravelMode, idx = c("individual", "mode"))
  TM3 <- dfidx(TravelMode, idx = list("individual", "mode"))
  # rename one or both indexes
  TM3b <- dfidx(TravelMode, idnames = c(NA, "trmode"))
  # for balanced data (with observations ordered by the first, then
  # by the second index
  # use the name of the first index
  TM4 <- dfidx(TravelMode, idx = "individual", idnames = c("individual", "mode"))
  # or an integer equal to the cardinal of the first index
  TM5 <- dfidx(TravelMode, idx = 210, idnames = c("individual", "mode"))
  # Indicate the values of the second index using the levels argument
TMSb <- dfidx(TravelMode, idx = 210, idnames = c("individual", "mode"),
levels = c("air", "train", "bus", "car"))
}

# Nesting structure for one of the index
if (requireNamespace("mlogit")){
data("JapaneseFDI", package = "mlogit")
JapaneseFDI <- dplyr::select(JapaneseFDI, 1:8)
JP1b <- dfidx(JapaneseFDI, idx = list("firm", c("region", "country")),
idnames = c("japf", "iso80"))
}

# Data in wide format
if (requireNamespace("mlogit")){
data("Fishing", package = "mlogit")
Fi <- dfidx(Fishing, shape = "wide", varying = 2:9, idnames = c("chid", "alt"))
}

---

**dplyr**

**Methods for dplyr verbs**

**Description**

methods of dplyr verbs for dfidx objects. Default functions don’t work because most of these functions returns either a tibble or a data.frame but not a dfidx.

**Usage**

```r
## S3 method for class 'dfidx'
arrange(.data, ...)

## S3 method for class 'dfidx'
filter(.data, ...)

## S3 method for class 'dfidx'
slice(.data, ...)

## S3 method for class 'dfidx'
mutate(.data, ...)

## S3 method for class 'dfidx'
transmute(.data, ...)

## S3 method for class 'dfidx'
select(.data, ...)
```

**Arguments**

- `.data` a dfidx object,
- `...` further arguments
Details

These methods always return the data frame column that contains the indexes and return a dfidx object.

Value

an object of class "dfidx"

Author(s)

Yves Croissant

Examples

```r
if (requireNamespace("AER")){
data("TravelMode", package = "AER")
TM <- dfidx(TravelMode)
select(TM, - wait, - vcost)
mutate(TM, inc2 = income ^ 2, linc = log(income))
transmute(TM, inc2 = income ^ 2, linc = log(income))
arrange(TM, desc(size), income)
filter(TM, income > 35, size <= 2)
pull(TM, income)
slice(TM, c(1:2, 5:7))
}
```

Description

The index of a dfidx is a data frame containing the different series which define the two indexes (with possibly a nesting structure). It is stored as a "sticky" data.frame column of the data.frame and is also inherited by series (of class 'xseries') which are extracted from a dfidx.

Usage

```r
idx(x, n = NULL, m = NULL)
## S3 method for class 'dfidx'
dIdx(x, n = NULL, m = NULL)

## S3 method for class 'idx'
dIdx(x, n = NULL, m = NULL)

## S3 method for class 'xseries'
dIdx(x, n = NULL, m = NULL)
```
## S3 method for class 'idx'
format(x, size = 4, ...)

**Arguments**

- `x`: a dfidx or a xseries
- `n, m`: `n` is the index to be extracted (1 or 2), `m` equal to one to get the index, greater than one to get a nesting variable.
- `size`: the number of characters of the indexes for the format method
- `...`: further arguments (for now unused)

**Details**

idx is defined as a generic with a dfidx and a xseries method.

**Value**

A data.frame containing the indexes or a series if a specific index is selected.

**Author(s)**

Yves Croissant

**Examples**

```r
if (requireNamespace("AER")){
  data("TravelMode", package = "AER")
  TM1 <- dfidx(TravelMode)
  idx(TM1)
  inc <- TM1$income
  idx(inc)
  # get the first index
  idx(TM1, 1)
  # get the second index
  idx(TM1, 2)
  idx(inc, 2)
}
```

---

**idx_name**

*Get the names of the indexes*

**Description**

This function extract the names of the indexes or the name of a specific index.
**idx_name**

**Usage**

```r
idx_name(x, n = 1, m = NULL)
```

```r
## S3 method for class 'dfidx'
idx_name(x, n = NULL, m = NULL)

## S3 method for class 'idx'
idx_name(x, n = NULL, m = NULL)

## S3 method for class 'xseries'
idx_name(x, n = NULL, m = NULL)
```

**Arguments**

- `x`: a dfidx, a idx or a xseries object
- `n`: the index to be extracted (1 or 2, ignoring the nesting variables)
- `m`: if > 1, a nesting variable

**Value**

If `n` is `NULL`, a named integer which gives the position of the `idx` column in the `dfidx` object, otherwise, a character of length 1

**Author(s)**

Yves Croissant

**Examples**

```r
if (requireNamespace("mlogit")){
data("JapaneseFDI", package = "mlogit")
JapaneseFDI <- dplyr::select(JapaneseFDI, 1:8)
JP1b <- dfidx(JapaneseFDI, idx = list("firm", c("region", "country")),
idnames = c("japf", "iso80"))
# get the position of the idx column
idx_name(JP1b)
# get the name of the first index
idx_name(JP1b, 1)
# get the name of the second index
idx_name(JP1b, 2)
# get the name of the nesting variable for the second index
idx_name(JP1b, 2, 2)
}
```
**Description**

A dfidx is a data.frame with a "sticky" data.frame column which contains the indexes. Specific methods of functions that extract lines and/or columns of a data.frame are provided.

**Usage**

```r
# S3 method for class 'dfidx'
x[i, j, drop = TRUE]

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

# S3 method for class 'dfidx'
print(x, ..., n = 10L)

# S3 method for class 'dfidx'
head(x, n = 10L, ...)

# S3 method for class 'dfidx'
x[[y]]

# S3 method for class 'dfidx'
x$y

# S3 replacement method for class 'dfidx'
object$y <- value

# S3 replacement method for class 'dfidx'
object[[y]] <- value

# S3 method for class 'xseries'
print(x, ..., n = 10L)

# S3 method for class 'idx'
print(x, ..., n = 10L)

# S3 method for class 'dfidx'
mean(x, ...)
```

**Arguments**

- `x`, `object`: a dfidx object
- `i`: the row index
j
the column index

drop
if TRUE a vector is returned if the result is a one column data.frame

row.names, optional
arguments of the generic as.data.frame method, not used

... further arguments

n the number of rows for the print method

y the name or the position of the series one wishes to extract

drop
if TRUE a vector is returned if the result is a one column data.frame

row.names, optional
arguments of the generic as.data.frame method, not used

... further arguments

n the number of rows for the print method

y the name or the position of the series one wishes to extract

value the value for the replacement method

Value

as.data.frame and mean return a data.frame, [[ and $ a vector, [ either a dfidx or a vector, $<-
and [[<- modify the values of an existing column or create a new column of a dfidx object, print
is called for its side effect

Author(s)

Yves Croissant

Examples

if (requireNamespace("AER")){
data("TravelMode", package = "AER")
TM <- dfidx(TravelMode)
# extract a series (returns as a xseries object)
TM$wait
# or
TM["wait"]
# extract a subset of series (returns as a dfidx object)
TM["wait", "income"]
# extract a subset of rows and columns
TM[TM$income > 30, c("wait", "income")]
# dfidx, idx and xseries have print methods as (like tibbles), a n
# argument
print(TM, n = 3)
print(idx(TM), n = 3)
print(TM$income, n = 3)
# a dfidx object can be coerced to a data.frame
head(as.data.frame(TM))
}

model.frame.dfidx  model.frame/matrix for dfidx objects
Description

Specific model.frame/matrix are provided for dfidx objects. This leads to an unusual order of arguments compared to the usage. Actually, the first two arguments of the model.frame method are a dfidx and a formula and the only main argument of the model.matrix is a dfidx which should be the result of a call to the model.frame method, i.e. it should have a term attribute.

Usage

```r
## S3 method for class 'dfidx'
model.frame(
  formula,
  data = NULL,
  ..., 
  lhs = NULL,
  rhs = NULL,
  dot = "previous",
  alt.subset = NULL,
  reflevel = NULL,
  balanced = FALSE
)

## S3 method for class 'dfidx'
model.matrix(object, ..., lhs = NULL, rhs = 1, dot = "separate")
```

Arguments

- `formula`: a dfidx
- `data`: a formula
- `...`, `lhs`, `rhs`, `dot` see the `Formula` method
- `alt.subset`: a subset of levels for the second index
- `reflevel`: a user-defined first level for the second index
- `balanced`: a boolean indicating if the resulting data.frame has to be balanced or not
- `object`: a dfidx object

Value

A dfidx object for the `model.frame` method and a matrix for the `model.matrix` method.

Author(s)

Yves Croissant

Examples

```r
if (requireNamespace("AER")){
data("TravelMode", package = "AER")
TM <- dfidx(TravelMode)
```
mf <- model.frame(TM, choice ~ vcost | income - 1 | travel)
head(model.matrix(mf, rhs = 1))
head(model.matrix(mf, rhs = 2))
head(model.matrix(mf, rhs = 1:3))

unfold_idx  
Fold and Unfold a dfidx object

Description

fold_idx takes a dfidx, includes the indexes as stand alone columns, remove the idx column and return a data.frame, with an ids attribute that contains the informations about the indexes. fold_idx performs the opposite operation

Usage

unfold_idx(x)
fold_idx(x, pkg = NULL)

Arguments

x       a dfidx object
pkg     if not NULL, this argument is passed to dfidx

Value

a data.frame for the unfold_dfidx function, a dfidx object for the fold_dfidx function

Author(s)

Yves Croissant

Examples

if (requireNamespace("AER")){
data("TravelMode", package = "AER")
TM <- dfidx(TravelMode)
TM2 <- unfold_idx(TM)
attr(TM2, "ids")
TM3 <- fold_idx(TM2)
identical(TM, TM3)
}
Index

[.. .dfidx (methods.dfidx), 8
[[ .dfidx (methods.dfidx), 8
[[<-.dfidx (methods.dfidx), 8
$.dfidx (methods.dfidx), 8
$<-.dfidx (methods.dfidx), 8

arrange.dfidx (dplyr), 4
as.data.frame.dfidx (methods.dfidx), 8

dfidx, 2
dplyr, 4

filter.dfidx (dplyr), 4
fold_idx (unfold_idx), 11
format.idx (idx), 5

head.dfidx (methods.dfidx), 8
id, 5
idx_name, 6

mean.dfidx (methods.dfidx), 8
methods.dfidx, 8
model.frame.dfidx, 9
model.matrix.dfidx (model.frame.dfidx), 9
mutate.dfidx (dplyr), 4

print.dfidx (methods.dfidx), 8
print.idx (methods.dfidx), 8
print.xseries (methods.dfidx), 8

select.dfidx (dplyr), 4
slice.dfidx (dplyr), 4

transmute.dfidx (dplyr), 4
unfold_idx, 11

12