Package ‘reshape2’

October 22, 2016

Title Flexibly Reshape Data: A Reboot of the Reshape Package
Version 1.4.2
Author Hadley Wickham <h.wickham@gmail.com>
Maintainer Hadley Wickham <h.wickham@gmail.com>
Description Flexibly restructure and aggregate data using just two functions: melt and 'dcast' (or 'acast').
URL https://github.com/hadley/reshape
BugReports https://github.com/hadley/reshape/issues
LinkingTo Rcpp
Imports plyr (>= 1.8.1), stringr, Rcpp
Suggests testthat (>= 0.8.0), lattice
License MIT + file LICENSE
LazyData true
RoxygenNote 5.0.1.9000
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-10-22 12:17:37

R topics documented:

addMargins ................................................................. 2
cast ................................................................. 2
colsplit .............................................................. 4
french_fries ........................................................... 5
melt ................................................................. 6
melt.array ........................................................... 6
melt.data.frame ......................................................... 8
melt.default ......................................................... 9
melt.list ........................................................... 9
melt_check ............................................................ 10
add_margins

Add margins to a data frame.

Description
Rownames are silently stripped. All margining variables will be converted to factors.

Usage
add_margins(df, vars, margins = TRUE)

Arguments
- **df**: input data frame
- **vars**: a list of character vectors giving the variables in each dimension
- **margins**: a character vector of variable names to compute margins for. TRUE will compute all possible margins.

cast

Cast functions Cast a molten data frame into an array or data frame.

Description
Use acast or dcast depending on whether you want vector/matrix/array output or data frame output. Data frames can have at most two dimensions.

Usage
dcast(data, formula, fun.aggregate = NULL, ..., margins = NULL, subset = NULL, fill = NULL, drop = TRUE, value.var = guess_value(data))

acast(data, formula, fun.aggregate = NULL, ..., margins = NULL, subset = NULL, fill = NULL, drop = TRUE, value.var = guess_value(data))
Arguments

data: molten data frame, see melt.
formula: casting formula, see details for specifics.
fun.aggregate: aggregation function needed if variables do not identify a single observation for each output cell. Defaults to length (with a message) if needed but not specified.
...: further arguments are passed to aggregating function
margins: vector of variable names (can include "grand\_col" and "grand\_row") to compute margins for, or TRUE to compute all margins. Any variables that can not be margined over will be silently dropped.
subset: quoted expression used to subset data prior to reshaping, e.g. subset = .(variable="length").
fill: value with which to fill in structural missings, defaults to value from applying fun.aggregate to 0 length vector
drop: should missing combinations dropped or kept?
value.var: name of column which stores values, see guess.value for default strategies to figure this out.

Details

The cast formula has the following format: x\_variable + x\_2 ~ y\_variable + y\_2 ~ z\_variable ~ ... The order of the variables makes a difference. The first varies slowest, and the last fastest. There are a couple of special variables: "..." represents all other variables not used in the formula and "." represents no variable, so you can do formula = var1 ~ .. Alternatively, you can supply a list of quoted expressions, in the form list(.(x\_variable, x\_2), .(y\_variable, y\_2)). The advantage of this form is that you can cast based on transformations of the variables: list(.(a + b), (c = round(c))). See the documentation for . for more details and alternative formats.

If the combination of variables you supply does not uniquely identify one row in the original data set, you will need to supply an aggregating function, fun.aggregate. This function should take a vector of numbers and return a single summary statistic.

See Also

melt, http://had.co.nz/reshape/

Examples

```r
# Air quality example
names(airquality) <- tolower(names(airquality))
aqm <- melt(airquality, id=c("month", "day"), na.rm=TRUE)

acast(aqm, day ~ month ~ variable)
acast(aqm, month ~ variable, mean)
acast(aqm, month ~ variable, mean, margins = TRUE)
dcast(aqm, month ~ variable, mean, margins = c("month", "variable"))

library(plyr) # needed to access . function
acast(aqm, variable ~ month, mean, subset = .(variable == "ozone"))
```
acast(aqm, variable ~ month, mean, subset = .(month == 5))

# Chick weight example
names(ChickWeight) <- tolower(names(ChickWeight))
chick_m <- melt(ChickWeight, id=2:4, na.rm=TRUE)
dcast(chick_m, time ~ variable, mean) # average effect of time
dcast(chick_m, diet ~ variable, mean) # average effect of diet
acast(chick_m, diet ~ time, mean) # average effect of diet & time

# How many chicks at each time? - checking for balance
acast(chick_m, time ~ diet, length)
acast(chick_m, chick ~ time, mean)
acast(chick_m, chick ~ time, mean, subset = .(time < 10 & chick < 20))

acast(chick_m, time ~ diet, length)
dcast(chick_m, diet + chick ~ time)
acast(chick_m, diet + chick ~ time)
acast(chick_m, chick ~ time ~ diet)
acast(chick_m, diet + chick ~ time, length, margins="diet")
acast(chick_m, diet + chick ~ time, length, drop = FALSE)

# Tips example
dcast(melt(tips), sex ~ smoker, mean, subset = .(variable == "total_bill"))

ff_d <- melt(french_fries, id=1:4, na.rm=TRUE)
acast(ff_d, subject ~ time, length)
acast(ff_d, subject ~ time, length, fill=0)
dcast(ff_d, treatment ~ variable, mean, margins = TRUE)
dcast(ff_d, treatment + subject ~ variable, mean, margins="treatment")
if (require("lattice")) {
lattice::xyplot("1" ~ "2" | variable, dcast(ff_d, ... ~ rep), aspect="iso")
}

---

**colsplit**

**Split a vector into multiple columns**

**Description**

Useful for splitting variable names that a combination of multiple variables. Uses `type.convert` to convert each column to correct type, but will not convert character to factor.

**Usage**

```
colsplit(string, pattern, names)
```
Arguments

- **string**: character vector or factor to split up
- **pattern**: regular expression to split on
- **names**: names for output columns

Examples

```r
x <- c("a_1", "a_2", "b_2", "c_3")
vars <- colsplit(x, ",", c("trt", "time"))
vars
str(vars)
```

---

**french_fries**

*Sensory data from a french fries experiment.*

Description

This data was collected from a sensory experiment conducted at Iowa State University in 2004. The investigators were interested in the effect of using three different fryer oils had on the taste of the fries.

Usage

```r
french_fries
```

Format

A data frame with 696 rows and 9 variables

Details

Variables:

- time in weeks from start of study.
- treatment (type of oil),
- subject,
- replicate,
- potato-y flavour,
- buttery flavour,
- grassy flavour,
- rancid flavour,
- painty flavour
melt Convert an object into a molten data frame.

Description

This the generic melt function. See the following functions for the details about different data structures:

Usage

melt(data, ..., na.rm = FALSE, value.name = "value")

Arguments

data Data set to melt

... further arguments passed to or from other methods.

na.rm Should NA values be removed from the data set? This will convert explicit missings to implicit missings.

value.name name of variable used to store values

Details

• melt.data.frame for data.frames

• melt.array for arrays, matrices and tables

• melt.list for lists

See Also

cast

melt.array Melt an array.

Description

This code is conceptually similar to as.data.frame.table
**Usage**

```r
## S3 method for class 'array'
melt(data, varnames = names(dimnames(data)), ..., 
    na.rm = FALSE, as.is = FALSE, value.name = "value")

## S3 method for class 'table'
melt(data, varnames = names(dimnames(data)), ..., 
    na.rm = FALSE, as.is = FALSE, value.name = "value")

## S3 method for class 'matrix'
melt(data, varnames = names(dimnames(data)), ..., 
    na.rm = FALSE, as.is = FALSE, value.name = "value")
```

**Arguments**

- `data` array to melt
- `varnames` variable names to use in molten data.frame
- `...` further arguments passed to or from other methods.
- `na.rm` Should NA values be removed from the data set? This will convert explicit missings to implicit missings.
- `as.is` if FALSE, the default, dimnames will be converted using `type.convert`. If TRUE, they will be left as strings.
- `value.name` name of variable used to store values

**See Also**

- `cast`

Other melt methods: `melt.data.frame`, `melt.default`, `melt.list`

**Examples**

```r
a <- array(c(1:3, NA), c(2,3,4))
melt(a)
melt(a, na.rm = TRUE)
melt(a, varnames=c("X","Y","Z"))
dimnames(a) <- lapply(dim(a), function(x) LETTERS[1:x])
melt(a)
melt(a, varnames=c("X","Y","Z"))
dimnames(a)[1] <- list(NULL)
melt(a)
```
melt.data.frame  

Melt a data frame into form suitable for easy casting.

Description

You need to tell melt which of your variables are id variables, and which are measured variables. If you only supply one of id.vars and measure.vars, melt will assume the remainder of the variables in the data set belong to the other. If you supply neither, melt will assume factor and character variables are id variables, and all others are measured.

Usage

```r
## S3 method for class 'data.frame'
melt(data, id.vars, measure.vars,
     variable.name = "variable", ..., na.rm = FALSE, value.name = "value",
     factorsAsStrings = TRUE)
```

Arguments

- `data` data frame to melt
- `id.vars` vector of id variables. Can be integer (variable position) or string (variable name). If blank, will use all non-measured variables.
- `measure.vars` vector of measured variables. Can be integer (variable position) or string (variable name). If blank, will use all non id.vars
- `variable.name` name of variable used to store measured variable names
- `...` further arguments passed to or from other methods.
- `na.rm` Should NA values be removed from the data set? This will convert explicit missings to implicit missings.
- `value.name` name of variable used to store values
- `factorsAsStrings` Control whether factors are converted to character when melted as measure variables. When FALSE, coercion is forced if levels are not identical across the measure.vars.

See Also

- `cast`

Other melt methods: `melt.array, melt.default, melt.list`

Examples

```r
names(airquality) <- tolower(names(airquality))
melt(airquality, id=c("month", "day"))
names(ChickWeight) <- tolower(names(ChickWeight))
melt(ChickWeight, id=2:4)
```
melt.default

Melt a vector. For vectors, makes a column of a data frame

Description
Melt a vector. For vectors, makes a column of a data frame

Usage

## Default S3 method:
melt(data, ..., na.rm = FALSE, value.name = "value")

Arguments

- data: vector to melt
- ...: further arguments passed to or from other methods.
- na.rm: Should NA values be removed from the data set? This will convert explicit missings to implicit missings.
- value.name: name of variable used to store values

See Also

melt, cast

Other melt methods: melt.array, melt.data.frame, melt.list

melt.list

Melt a list by recursively melting each component.

Description
Melt a list by recursively melting each component.

Usage

## S3 method for class 'list'
melt(data, ..., level = 1)

Arguments

- data: list to recursively melt
- ...: further arguments passed to or from other methods.
- level: list level - used for creating labels
melt_check

Check that input variables to melt are appropriate.

Description

If id.vars or measure.vars are missing, melt_check will do its best to impute them. If you only supply one of id.vars and measure.vars, melt will assume the remainder of the variables in the data set belong to the other. If you supply neither, melt will assume discrete variables are id variables and all other are measured.

Usage

melt_check(data, id.vars, measure.vars, variable.name, value.name)

Arguments

data: data frame
id.vars: vector of identifying variable names or indexes
measure.vars: vector of Measured variable names or indexes
variable.name: name of variable used to store measured variable names
value.name: name of variable used to store values

Value

a list giving id and measure variables names.

Examples

a <- as.list(c(1:4, NA))
melt(a)
names(a) <- letters[1:4]
melt(a)
a <- list(matrix(1:4, ncol=2), matrix(1:6, ncol=2))
melt(a)
a <- list(matrix(1:4, ncol=2), array(1:27, c(3,3,3)))
melt(a)
melt(list(1:5, matrix(1:4, ncol=2)))
melt(list(list(1:3), 1, list(as.list(3:4), as.list(1:2))))
parse_formula

Parse casting formulae.

Description

There are two ways to specify a casting formula: either as a string, or a list of quoted variables. This function converts the former to the latter.

Usage

parse_formula(formula = "... ~ variable", varnames, value.var = "value")

Arguments

formula formula to parse
varnames names of all variables in data
value.var name of variable containing values

Details

Casting formulas separate dimensions with ~ and variables within a dimension with + or *. can be used as a placeholder, and ... represents all other variables not otherwise used.

Examples

reshape::parse_formula("a + ...", letters[1:6])
reshape::parse_formula("a ~ b + d")
reshape::parse_formula("a + b ~ c ~ .")

recast

Recast: melt and cast in a single step

Description

This conveniently wraps melting and (d)casting a data frame into a single step.

Usage

recast(data, formula, ..., id.var, measure.var)

Arguments

data data set to melt
formula casting formula, see dcast for specifics
... other arguments passed to dcast
id.var identifying variables. If blank, will use all non measure.var variables
measure.var measured variables. If blank, will use all non id.var variables
See Also

http://had.co.nz/reshape/

Examples

recast(french_fries, time ~ variable, id.var = 1:4)

---

<table>
<thead>
<tr>
<th>smiths</th>
<th>Demo data describing the Smiths.</th>
</tr>
</thead>
</table>

Description

A small demo dataset describing John and Mary Smith. Used in the introductory vignette.

Usage

smiths

Format

A data frame with 2 rows and 5 variables

---

<table>
<thead>
<tr>
<th>tips</th>
<th>Tipping data</th>
</tr>
</thead>
</table>

Description

One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:

Usage

tips

Format

A data frame with 244 rows and 7 variables
Details

• tip in dollars,
• bill in dollars,
• sex of the bill payer,
• whether there were smokers in the party,
• day of the week,
• time of day,
• size of the party.

In all he recorded 244 tips. The data was reported in a collection of case studies for business statistics (Bryant & Smith 1995).

References

Index

*Topic datasets
  french_fries, 5
  smiths, 12
  tips, 12
*Topic manip
  cast, 2
  colsplit, 4
  melt, 6
  melt.array, 6
  melt.data.frame, 8
  melt.default, 9
  melt.list, 9
  recast, 11
.., 3
acast (cast), 2
add_margins, 2
as.data.frame.table, 6

cast, 2, 6–10
colsplit, 4
dcast, 11
dcast (cast), 2
french_fries, 5
guess_value, 3
melt, 3, 6, 9
melt.array, 6, 6, 8–10
melt.data.frame, 6, 7, 8, 9, 10
melt.default, 7, 8, 9, 10
melt.list, 6–9, 9
melt.matrix (melt.array), 6
melt.table (melt.array), 6
melt_check, 10
parse_formula, 11
recast, 11

smths, 12
tips, 12
type.convert, 4, 7