Package 'tidyr'

July 21, 2014

Title Easily tidy data with spread and gather functions.

Version 0.1

Description tidyr is an evolution of reshape2. It's design specifically for
data tidying (not general reshaping or aggregating) and works well with dplyr data pipelines.

Depends R (>= 3.1.0)

License MIT + file LICENSE

LazyData true

Imports reshape2, dplyr (>= 0.2)

URL https://github.com/hadley/tidyr

Suggests knitr, testthat

VignetteBuilder knitr

Author 'Hadley Wickham' [aut, cre]

Maintainer 'Hadley Wickham' <h.wickham@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-21 23:16:05

R topics documented:

extract — *Extract one column into multiple columns.*

Description

Given a regular expression with capturing groups, `extract()` turns each group into a new column.

Usage

```
extract(data, col, into, regex = "([[:alnum:]]+)", remove = TRUE,
  convert = FALSE, ...)
```

Arguments

`col`	Bare column name.
`data`	A data frame.
`into`	Names of new variables to create as character vector.
`regex`	a regular expression used to extract the desired values.
`remove`	If `TRUE`, remove input column from output data frame.
`convert`	If `TRUE`, will run `type.convert` with `as.is = TRUE` on new columns. This is useful if the component columns are integer, numeric or logical.
`...`	Other arguments passed on to `regexec` to control how the regular expression is processed.

Examples

```
library(dplyr)
df <- data.frame(x = c("a.b", "a.d", "b.c"))
df %>% extract(x, "A")
df %>% extract(x, c("A", "B"), "([[:alnum:]]+)\\.([[:alnum:]]+)")
```

extract_numeric — *Extract numeric component of variable.*

Description

Extract numeric component of variable.

Usage

```
extract_numeric(x)
```

Arguments

`x`	A character vector (or a factor).

Examples

```
extract_numeric("$1,200.34")
```

---

gather *Gather columns into key-value pairs.*

Description

Gather takes multiple columns and collapses into key-value pairs, duplicating all other columns as needed. You use `gather()` when you notice that you have columns that are not variables.

Usage

```
gather(data, key, value, ..., na.rm = FALSE, convert = FALSE)
```

Arguments

`data`	A data frame.
`key,value`	Names of key and value columns to create in output.
`...`	Specification of columns to gather. Use bare variable names. Select all variables between x and z with `x:z`, exclude y with `-y`. For more options, see the select documentation.
`na.rm`	If `TRUE`, will remove rows from output where the value column in `NA`.
`convert`	If `TRUE` will automatically run `type.convert` on the key column. This is useful if the column names are actually numeric, integer, or logical.

See Also

`gather_` for a version that uses regular evaluation and is suitable for programming with.

Examples

```
library(dplyr)
# From http://stackoverflow.com/questions/1181060
stocks <- data.frame(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4)
)

gather(stocks, stock, price, -time)
stocks %>% gather(stock, price, -time)
```

`separate` *Separate one column into multiple columns.*

Description

Given either regular expression or a vector of character positions, `separate()` turns a single character column into multiple columns.

Usage

```
separate(data, col, into, sep = "[^[:alnum:]]+", remove = TRUE,
  convert = FALSE, ...)
```

Arguments

`col`	Bare column name.
`data`	A data frame.
`into`	Names of new variables to create as character vector.
`sep`	Separator between columns. If character, is interpreted as a regular expression. The default value is a regular expression that matches any sequence of non-alphanumeric values. If numeric, interpreted as positions to split at. Positive values start at 1 at the far-left of the string; negative value start at -1 at the far-right of the string. The length of `sep` should be one less than `into`.
`remove`	If `TRUE`, remove input column from output data frame.
`convert`	If `TRUE`, will run `type.convert` with `as.is = TRUE` on new columns. This is useful if the component columns are integer, numeric or logical.
`...`	Other arguments passed on to `strsplit` to control how the regular expression is processed.

Examples

```
library(dplyr)
df <- data.frame(x = c("a.b", "a.d", "b.c"))
df %>% separate(x, c("A", "B"))
```

spread *Spread a key-value pair across multiple columns.*

Description

Spread a key-value pair across multiple columns.

Usage

```
spread(data, key, value, fill = NA, convert = FALSE)
```

Arguments

key,value	Bare (unquoted) names of key and value columns.
data	A data frame.
fill	If there isn't a value for every combination of the other variables and the key column, this value will be substituted.
convert	If TRUE, type.convert with asis = TRUE will be run on each of the new columns. This is useful if the value column was a mix of variables that was coerced to a string.

Examples

```
library(dplyr)
stocks <- data.frame(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4)
)
stocksm <- stocks %>% gather(stock, price, -time)
stocksm %>% spread(stock, price)
stocksm %>% spread(time, price)

# Spread and gather are complements
df <- data.frame(x = c("a", "b"), y = c(3, 4), z = c(5, 6))
df %>% spread(x, y) %>% gather(x, y, a:b, na.rm = TRUE)
```

spread_ *Standard-evaluation version of* spread.

Description

This is a S3 generic.

Usage

```
spread_(data, key_col, value_col, fill = NA, convert = FALSE)
```

Arguments

data	A data frame.
key_col,value_col	Strings giving names of key and value cols.
fill	If there isn't a value for every combination of the other variables and the key column, this value will be substituted.
convert	If TRUE, type.convert with asis = TRUE will be run on each of the new columns. This is useful if the value column was a mix of variables that was coerced to a string.

---

unite *Unite multiple columns into one.*

Description

Convenience function to paste together multiple functions into one.

Usage

```
unite(data, col, ..., sep = "_", remove = TRUE)
```

Arguments

col	(Bare) name of column to add
...	Specification of columns to unite. Use bare variable names. Select all variables between x and z with x:z, exclude y with -y. For more options, see the select documentation.
data	A data frame.
sep	Separator to use between values.
remove	If TRUE, remove input columns from output data frame.

See Also

separate(), the complement.

Examples

```
library(dplyr)
unite_(mtcars, "vs_am", c("vs","am"))

# Separate is the complement of unite
mtcars %>%
  unite(vs_am, vs, am) %>%
  separate(vs_am, c("vs", "am"))
```

Index