Package 'pivot'

May 1, 2018

Title 'SQL' PIVOT and UNPIVOT

Version 18.4.17

Maintainer Andrew Redd <Andrew.Redd@hsc.utah.edu>

Description Extends the 'tidyverse' packages 'dbplyr' and 'tidyr' functionality with pivot(), i.e. spread(), and unpivot(), i.e. gather(), for reshaping remote tables.
Currently only 'Microsoft SQL Server' is supported.

Depends R (>= 3.4.0)

Imports assertthat, dplyr, dbplyr (>= 1.2.1), DBI, magrittr, purrr, rlang, tidyselect, tidyr, lubridate, colorspace

Enhances odbc

Suggests testthat, covr

License Unlimited

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Collate 'pivot.R' 'unpivot.R' 'tidyr.R' 'utils.R' 'testing-utils.R'

NeedsCompilation no

Language en-US

Author Andrew Redd [aut, cre] (<https://orcid.org/000-0002-6149-2438>)

Repository CRAN

Date/Publication 2018-05-01 09:49:18 UTC

R topics documented:

`pivot` *Pivot a table*

Description

Pivot a table

Usage

```
pivot(data, key, value, ..., fill = NULL)
```

Arguments

`data`	A data frame.
`key`	Column names or positions. This is passed to `tidyselect::vars_pull()`. These arguments are passed by expression and support quasiquotation (you can unquote column names or column positions).
`value`	Column names or positions. This is passed to `tidyselect::vars_pull()`. These arguments are passed by expression and support quasiquotation (you can unquote column names or column positions).
`...`	Selection criteria for levels of key to select.
`fill`	If set, missing values will be replaced with this value. Note that there are two types of missingness in the input: explicit missing values (i.e. `NA`), and implicit missings, rows that simply aren't present. Both types of missing value will be replaced by `fill`.

Examples

```
library(dplyr)
library(dbplyr)
# establish db as a database connection

## Not run:
db_iris <- copy_to(db, iris)

## End(Not run)
result <- pivot( db_iris, Species, mean(Petal.Length, na.rm=TRUE)
               , setosa, versicolor, virginica)
sql_render(result)
```

pivot_query *Create a pivot query representation*

Description

Create a pivot table

Usage

```
pivot_query(from, key, value, levels, select = ident(), order_by = NULL,
  fill = NULL)
```

Arguments

from	the from clause
key	Variable columns originate from
value	The expression to evaluate to create the values
levels	the levels of key to turn into columns.
select	variables to select in addition to levels.
order_by	optional order by clause
fill	optional value to fill in structural missing values. It is the responsibility of the user to ensure type compatibility.

Examples

```
library(dplyr)
library(dbplyr)
con <- simulate_mssql()

query <- pivot_query( ident('##iris'), key = ident('Species')
                    , levels = ident(c('setosa', 'virginica', 'versicolor'))
                    , value  = rlang::quo(mean(Petal.Length, na.rm=TRUE))
                    )
sql_render(query, con=con)
```

sql_pivot *Create a Pivot Query*

Description

Creates a SQL pivot query. Similar to the `tidyr::spread` function.

Usage

```
sql_pivot(con, from, select, key, value, levels, ...)
```

Arguments

con	a Database connection
from	the from clause
select	variables to select in addition to levels.
key	Variable columns originate from
value	The expression to evaluate to create the values
levels	the levels of key to turn into columns.
...	arguments to pass on or ignore.

Examples

```
library(dbplyr)
query <- sql_pivot( dbplyr::simulate_mssql()
                  , from   = ident('##iris')
                  , select = ident()
                  , key    = ident('Species')
                  , value  = rlang::quo(mean(Petal.Length, na.rm=TRUE))
                  , levels = ident(c('versicolor', 'virginica'))
                  )
sql_render(query)
```

---

sql_unpivot *Create an unpivot query*

Description

Creates a SQL pivot query. Similar to the `tidyr::gather` function.

Usage

```
sql_unpivot(con, from, select, key, value, levels, order_by = NULL)
```

Arguments

con	a Database connection
from	the from clause
select	variables to select in addition to levels.
key	Variable columns originate from
value	The expression to evaluate to create the values
levels	the columns to turn into values of a variable.
order_by	optional order by clause

unpivot *Un-pivot a table*

Description

Un-pivot a table

Usage

```
unpivot(data, key, value, ...)
```

Arguments

data	A data frame.
key	Names of new key and value columns, as strings or symbols. This argument is passed by expression and supports quasiquotation (you can unquote strings and symbols). The name is captured from the expression with rlang::quo_name() (note that this kind of interface where symbols do not represent actual objects is now discouraged in the tidyverse; we support it here for backward compatibility).
value	Names of new key and value columns, as strings or symbols. This argument is passed by expression and supports quasiquotation (you can unquote strings and symbols). The name is captured from the expression with rlang::quo_name() (note that this kind of interface where symbols do not represent actual objects is now discouraged in the tidyverse; we support it here for backward compatibility).
...	Selection criteria for columns to unpivot.

Examples

```
# establish `db` as a database connection

library(dplyr)
library(dbplyr)

## Not run:
    db_iris <- copy_to(db, iris)

## End(Not run)
long.iris <- unpivot(db_iris, Variable, Value, Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)
sql_render(long.iris)
```

unpivot_query *Create a pivot query representation*

Description

Create a pivot table

Usage

```
unpivot_query(from, key, value, levels, select = character(0),
  order_by = NULL)
```

Arguments

from	the from clause
key	Variable columns originate from
value	The expression to evaluate to create the values
levels	the columns to turn into values of a variable.
select	variables to select in addition to levels.
order_by	optional order by clause

Index