Package ‘widyr’

September 9, 2019

Type Package
Title Widen, Process, then Re-Tidy Data
Version 0.1.2
Description Encapsulates the pattern of untidying data into a wide matrix, performing some processing, then turning it back into a tidy form. This is useful for several operations such as co-occurrence counts, correlations, or clustering that are mathematically convenient on wide matrices.
License MIT + file LICENSE
LazyData TRUE
Maintainer David Robinson <admiral.david@gmail.com>
URL http://github.com/dgtrtwo/widyr
BugReports http://github.com/dgtrtwo/widyr/issues
VignetteBuilder knitr
Imports dplyr, tidyr, reshape2, tidytext, purrr, Matrix, broom
Suggests ggraph, igraph, gapminder, testthat, covr, knitr, topicmodels, jameaustenr, rmarkdown, unvotes (>= 0.2.0), countrycode, fuzzyjoin, ggplot2, maps, irlba
RoxygenNote 6.1.1
NeedsCompilation no
Author David Robinson [aut, cre], Kanishka Misra [ctb]
Repository CRAN
Date/Publication 2019-09-09 21:20:02 UTC

R topics documented:
cor_sparse ............................................ 2
pairwise_cor ........................................... 2
pairwise_count ....................................... 3
pairwise_delta ........................................ 4
cor_sparse

Find the Pearson correlation of a sparse matrix efficiently

Description

Find the Pearson correlation of a sparse matrix. For large sparse matrix this is more efficient in time and memory than cor(as.matrix(x)). Note that it does not currently work on simple_triplet_matrix objects.

Usage

cor_sparse(x)

Arguments

x A matrix, potentially a sparse matrix such as a "dgTMatrix" object

Source

This code comes from mike on this Stack Overflow answer: http://stackoverflow.com/a/9626089/712603.

pairwise_cor

Correlations of pairs of items

Description

Find correlations of pairs of items in a column, based on a "feature" column that links them together. This is an example of the spread-operate-retidy pattern.

Usage

pairwise_cor(tbl, item, feature, value, method = c("pearson", "kendall", "spearman"), use = "everything", ...)

pairwise_cor_(tbl, item, feature, value, method = c("pearson", "kendall", "spearman"), use = "everything", ...)

pairwise_dist pairwise_pmi pairwise_similarity

squarely widely widely_svd
pairwise_count

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tbl</td>
<td>Table</td>
</tr>
<tr>
<td>item</td>
<td>Item to compare; will end up in item1 and item2 columns</td>
</tr>
<tr>
<td>feature</td>
<td>Column describing the feature that links one item to others</td>
</tr>
<tr>
<td>value</td>
<td>Value column. If not given, defaults to all values being 1 (thus a binary correlation)</td>
</tr>
<tr>
<td>method</td>
<td>Correlation method</td>
</tr>
<tr>
<td>use</td>
<td>Character string specifying the behavior of correlations with missing values; passed on to cor</td>
</tr>
<tr>
<td>...</td>
<td>Extra arguments passed on to squarely, such as diag and upper</td>
</tr>
</tbody>
</table>

Examples

```r
library(dplyr)
library(gapminder)

gapminder %>%
  pairwise_cor(country, year, lifeExp)

gapminder %>%
  pairwise_cor(country, year, lifeExp, sort = TRUE)

# United Nations voting data
library(unvotes)

country_cors <- un_votes %>%
  mutate(vote = as.numeric(vote)) %>%
  pairwise_cor(country, rcid, vote, sort = TRUE)

country_cors
```

Description

Count the number of times each pair of items appear together within a group defined by "feature." For example, this could count the number of times two words appear within documents.

Usage

```r
pairwise_count(tbl, item, feature, wt = NULL, ...)

pairwise_count_(tbl, item, feature, wt = NULL, ...)
```
pairwise_delta

Description

Compute the delta distances (from its two variants) of all pairs of documents in a tidy table.

Usage

pairwise_delta(tbl, item, feature, value, method = "burrows", ...)

Examples

library(dplyr)
dat <- data.frame(group = rep(1:5, each = 2),
letter = c("a", "b",
"a", "c",
"a", "c",
"b", "e",
"b", "f"))

# count the number of times two letters appear together
pairwise_count(dat, letter, group)
pairwise_count(dat, letter, group, sort = TRUE)
pairwise_count(dat, letter, group, sort = TRUE, diag = TRUE)
pairwise_delta

Arguments

tbl Table
item Item to compare; will end up in item1 and item2 columns
feature Column describing the feature that links one item to others
value Value
method Distance measure to be used; see dist
... Extra arguments passed on to squarely, such as diag and upper

See Also

squarely

Examples

library(janeaustenr)
library(dplyr)
library(tidytext)

# closest documents in terms of 1000 most frequent words
closest <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word) %>%
  top_n(1000, n) %>%
  pairwise_delta(book, word, n, method = "burrows") %>%
  arrange(delta)

closest

closest %>%
  filter(item1 == "Pride & Prejudice")

# to remove duplicates, use upper = FALSE
closest <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word) %>%
  top_n(1000, n) %>%
  pairwise_delta(book, word, n, method = "burrows", upper = FALSE) %>%
  arrange(delta)

# Can also use Argamon's Linear Delta
closest <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word) %>%
  top_n(1000, n) %>%
  pairwise_delta(book, word, n, method = "argamon", upper = FALSE) %>%
  arrange(delta)
**pairwise_dist**

*Distances of pairs of items*

**Description**

Compute distances of all pairs of items in a tidy table.

**Usage**

```r
pairwise_dist(tbl, item, feature, value, method = "euclidean", ...)
pairwise_dist_(tbl, item, feature, value, method = "euclidean", ...)
```

**Arguments**

- **tbl**: Table
- **item**: Item to compare; will end up in `item1` and `item2` columns
- **feature**: Column describing the feature that links one item to others
- **value**: Value
- **method**: Distance measure to be used; see `dist`
- **...**: Extra arguments passed on to `squarely`, such as `diag` and `upper`

**See Also**

`squarely`

**Examples**

```r
library(gapminder)
library(dplyr)

# closest countries in terms of life expectancy over time
closest <- gapminder %>%
  pairwise_dist(country, year, lifeExp) %>%
  arrange(distance)

closest

closest %>%
  filter(item1 == "United States")

# to remove duplicates, use upper = FALSE
gapminder %>%
  pairwise_dist(country, year, lifeExp, upper = FALSE) %>%
  arrange(distance)
```
# Can also use Manhattan distance

gapminder %>%
pairwise_dist(country, year, lifeExp, method = "manhattan", upper = FALSE) %>%
arrange(distance)

---

**pairwise_pmi**  
Pointwise mutual information of pairs of items

### Description

Find pointwise mutual information of pairs of items in a column, based on a "feature" column that links them together. This is an example of the spread-operate-retidy pattern.

### Usage

```r
pairwise_pmi(tbl, item, feature, sort = FALSE, ...)
```

### Arguments

- **tbl**: Table
- **item**: Item to compare; will end up in `item1` and `item2` columns
- **feature**: Column describing the feature that links one item to others
- **sort**: Whether to sort in descending order of the pointwise mutual information
- **...**: Extra arguments passed on to `squarely`, such as `diag` and `upper`

### Value

A tbl_df with three columns, `item1`, `item2`, and `pmi`.

### Examples

```r
library(dplyr)

dat <- data_frame(group = rep(1:5, each = 2),
  letter = c("a", "b",
    "a", "c",
    "a", "c",
    "b", "e",
    "b", "f"))

# how informative is each letter about each other letter
pairwise_pmi(dat, letter, group)
pairwise_pmi(dat, letter, group, sort = TRUE)
```
pairwise_similarity     Cosine similarity of pairs of items

Description
Compute cosine similarity of all pairs of items in a tidy table.

Usage
pairwise_similarity(tbl, item, feature, value, ...)
pairwise_similarity_(tbl, item, feature, value, ...)

Arguments
  tbl            Table
  item           Item to compare; will end up in item1 and item2 columns
  feature        Column describing the feature that links one item to others
  value          Value
  ...            Extra arguments passed on to squarely, such as diag and upper

See Also
  squarely

Examples
library(janeaustenr)
library(dplyr)
library(tidytext)

# Comparing Jane Austen novels
austen_words <- austen_books() %>%
  unnest_tokens(word, text) %>%
  anti_join(stop_words, by = "word") %>%
  count(book, word) %>%
  ungroup()

# closest books to each other
closest <- austen_words %>%
  pairwise_similarity(book, word, n) %>%
  arrange(desc(similarity))

closest

closest %>%
  filter(item1 == "Emma")
Description

A special case of `widely`. Used to pre-prepare and post-tidy functions that take an \( m \times n \) (\( m \) items, \( n \) features) matrix and return an \( m \times m \) (item x item) matrix, such as a distance or correlation matrix.

Usage

```
squarely(.f, diag = FALSE, upper = TRUE, ...)
squarely_(.f, diag = FALSE, upper = TRUE, ...)
```

Arguments

- `.f` Function to wrap
- `diag` Whether to include diagonal \((i = j)\) in output
- `upper` Whether to include upper triangle, which may be duplicated
- `...` Extra arguments passed on to `widely`

Value

Returns a function that takes at least four arguments:

```
tbl A table
item Name of column to use as rows in wide matrix
feature Name of column to use as columns in wide matrix
feature Name of column to use as values in wide matrix
... Arguments passed on to inner function
```

See Also

- `widely`, `pairwise_count`, `pairwise_cor`, `pairwise_dist`

Examples

```
library(dplyr)
library(gapminder)

closest_continent <- gapminder %>%
  group_by(continent) %>%
  squarely(dist)(country, year, lifeExp)
```
widely

Adverb for functions that operate on matrices in "wide" format

Description

Modify a function in order to pre-cast the input into a wide matrix format, perform the function, and then re-tidy (e.g. melt) the output into a tidy table.

Usage

widely(.f, sort = FALSE, sparse = FALSE, maximum_size = 1e+07)

widely_(.f, sort = FALSE, sparse = FALSE, maximum_size = 1e+07)

Arguments

.f Function being wrapped

sort Whether to sort in descending order of value

sparse Whether to cast to a sparse matrix

maximum_size To prevent crashing, a maximum size of a non-sparse matrix to be created. Set to NULL to allow any size matrix.

Value

Returns a function that takes at least four arguments:

tbl A table

row Name of column to use as rows in wide matrix

column Name of column to use as columns in wide matrix

value Name of column to use as values in wide matrix

... Arguments passed on to inner function

widely creates a function that takes those columns as bare names, widely_ a function that takes them as strings.

Examples

library(dplyr)
library(gapminder)

gapminder

gapminder %>%
  widely(dist)(country, year, lifeExp)

# can perform within groups
widely_svd

```r
closest.continent <- gapminder %>%
  group_by(continent) %>%
  widely(dist)(country, year, lifeExp)
closest.continent

# for example, find the closest pair in each
closest.continent %>%
  top_n(1, -value)
```

---

**widely_svd**

*Turn into a wide matrix, perform SVD, return to tidy form*

---

**Description**

This is useful for dimensionality reduction of items, especially when setting a lower `nv`.

**Usage**

```r
widely_svd(tbl, item, feature, value, nv = NULL, weight_d = FALSE, ...)
widely_svd_(tbl, item, feature, value, nv = NULL, weight_d = FALSE, ...)
```

**Arguments**

- `tbl`  
  Table
- `item`  
  Item to perform dimensionality reduction on; will end up in `item` column
- `feature`  
  Column describing the feature that links one item to others.
- `value`  
  Value
- `nv`  
  Optional; the number of principal components to estimate. Recommended for matrices with many features.
- `weight_d`  
  Whether to multiply each value by the `d` principal component.
- `...`  
  Extra arguments passed to `svd` (if `nv` is `NULL`) or `irlba` (if `nv` is given)

**Value**

A `tbl_df` with three columns. The first is retained from the `item` input, then `dimension` and `value`. Each row represents one principal component value.
Examples

```r
library(dplyr)
library(gapminder)

# principal components driving change
gapminder_svd <- gapminder %>%
  widely_svd(country, year, lifeExp)

gapminder_svd

# compare SVDs, join with other data
library(ggplot2)
library(tidyr)

gapminder_svd %>%
  spread(dimension, value) %>%
  inner_join(distinct(gapminder, country, continent), by = "country") %>%
  ggplot(aes(\texttt{\texttt{\texttt{1}}}, \texttt{\texttt{\texttt{2}}}, label = country)) +
  geom_point(aes(color = continent)) +
  geom_text(vjust = 1, hjust = 1)
```
Index

cor_sparse, 2

dist, 5, 6

pairwise_cor, 2, 9
pairwise_cor_ (pairwise_cor), 2
pairwise_count, 3, 9
pairwise_count_ (pairwise_count), 3
pairwise_delta, 4
pairwise_delta_ (pairwise_delta), 4
pairwise_dist, 6, 9
pairwise_dist_ (pairwise_dist), 6
pairwise_pmi, 7
pairwise_pmi_ (pairwise_pmi), 7
pairwise_similarity, 8
pairwise_similarity_ (pairwise_similarity), 8

squarely, 4–6, 8, 9
squarely_ (squarely), 9

widely, 9, 10
widely_ (widely), 10
widely_svd, 11
widely_svd_ (widely_svd), 11