Package: rsmatrix (via r-universe)

October 18, 2026

Title Matrices for Repeat-Sales Price Indexes

Version 0.3.0.9001

Description Calculate the matrices in Shiller (1991, <doi:10.1016/S1051-1377(05)80028-2>) that serve as the foundation for many repeat-sales price indexes.

Depends R (>= 4.0)

Imports Matrix (>= 1.5-0)

Suggests litedown, piar, spelling, tinytest

License GPL (>= 3)

Encoding UTF-8

BugReports https://codeberg.org/marberts/rsmatrix/issues

VignetteBuilder litedown

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Language en-US

Repository https://marberts.r-universe.dev

Date/Publication 2026-09-18 02:59:44 UTC

RemoteUrl https://codeberg.org/marberts/rsmatrix

RemoteRef HEAD

RemoteSha 15ddba16d980e25ad6c5066a8bc9aeb61c17f34b

Contents

rs_matrix 2
rs_pairs 4
rs_var 5

Index **7**

`rs_matrix` *Shiller's repeat-sales matrices*

Description

Create a function to compute the Z, X, y, and Y matrices in Shiller (1991, section I-II) from sales-pair data in order to calculate a repeat-sales price index.

Usage

```
rs_matrix(t2, t1, p2, p1, f = NULL, sparse = FALSE)
```

Arguments

`t2`, `t1`	`[Date | character]` A pair of vectors giving the time period of the second and first sale, respectively. Usually a vector of dates, but other values are possible if they can be coerced to character vectors and sorted in chronological order (i.e., with `base::order()`).
`p2`, `p1`	`[numeric > 0]` A pair of numeric vectors giving the price of the second and first sale, respectively.
`f`	`[factor]` An optional factor the same length as `t1` and `t2`, or a vector to be turned into a factor, that is used to group sales.
`sparse`	`[logical(1)]` Should sparse matrices from the **Matrix** package be used (faster for large datasets), or regular dense matrices (the default)?

Details

The function returned by `rs_matrix()` computes a generalization of the matrices in Shiller (1991, section I-II) that are applicable to grouped data. These are useful for calculating separate indexes for many, say, cities without needing an explicit loop.

The Z, X, and Y matrices are not well defined if either `t1` or `t2` have missing values, and an error is thrown in this case. Similarly, it should always be the case that `t2 > t1`, otherwise a warning is given.

Value

A function that takes a single argument naming the desired matrix. It returns one of two matrices (Z and X) or two vectors (y and Y), either regular matrices if `sparse = FALSE`, or sparse matrices of class `dgCMatrix` if `sparse = TRUE`.

References

Bailey MJ, Muth RF, Nourse HO (1963). "A regression method for real estate price index construction." *Journal of the American Statistical Association*, **53**(304), 933-942.

Shiller RJ (1991). "Arithmetic repeat sales price estimators." *Journal of Housing Economics*, **1**(1), 110-126. doi:10.1016/S10511377(05)800282.

See Also

`rs_pairs()` for turning sales data into sales pairs.

Examples

```
# Make some data
x <- data.frame(
  date = c(3, 2, 3, 2, 3, 3),
  date_prev = c(1, 1, 2, 1, 2, 1),
  price = 6:1,
  price_prev = 1
)

# Calculate matrices
mat <- with(x, rs_matrix(date, date_prev, price, price_prev))
Z <- mat("Z") # Z matrix
X <- mat("X") # X matrix
y <- mat("y") # y vector
Y <- mat("Y") # Y vector

# Calculate the GRS index in Bailey, Muth, and Nourse (1963)
b <- solve(crossprod(Z), crossprod(Z, y))[, 1]
# or b <- qr.coef(qr(Z), y)
(grs <- exp(b) * 100)

# Standard errors
vcov <- rs_var(y - Z %*% b, Z)
sqrt(diag(vcov)) * grs # delta method

# Calculate the ARS index in Shiller (1991)
b <- solve(crossprod(Z, X), crossprod(Z, Y))[, 1]
# or b <- qr.coef(qr(crossprod(Z, X)), crossprod(Z, Y))
(ars <- 100 / b)

# Standard errors
vcov <- rs_var(Y - X %*% b, Z, X)
sqrt(diag(vcov)) * ars^2 / 100 # delta method

# Works with grouped data
x <- data.frame(
  date = c(3, 2, 3, 2),
  date_prev = c(2, 1, 2, 1),
  price = 4:1,
  price_prev = 1,
  group = c("a", "a", "b", "b")
)

mat <- with(x, rs_matrix(date, date_prev, price, price_prev, group))
b <- solve(crossprod(mat("Z"), mat("X")), crossprod(mat("Z"), mat("Y")))[, 1]
100 / b
```

`rs_pairs` *Sales pairs*

Description

Turn repeat-sales data into sales pairs that are suitable for making repeat-sales matrices.

Usage

```
rs_pairs(period, product, match_first = TRUE)
```

Arguments

`period`	`[Date | factor]` A vector that gives the time period for each sale. Usually a date vector, or a factor with the levels in chronological order, but other values are possible if they can be sorted in chronological order (i.e., with `base::order()`).
`product`	`[factor]` A vector that gives the product identifier for each sale. Usually a factor or vector of integer codes for each product.
`match_first`	`[logical(1)]` Should products in the first period match with themselves (the default)?

Value

A numeric vector of indices giving the position of the previous sale for each `product`, with the convention that the previous sale for the first sale is itself if `match_first = TRUE`, `NA` otherwise. Ties are resolved according to the order they appear in `period`.

Note

`base::order()` is the workhorse of `rs_pairs()`, so performance can be sensitive to the types of `period` and `product`, and can be slow for large character vectors.

See Also

`rs_matrix()` for using sales pairs to make a repeat-sales index.

`rtCreateTrans()` in the **hpiR** package for a feature-rich but slower and less flexible function to make sales pairs.

Examples

```
# Make sales pairs
x <- data.frame(
  id = c(1, 1, 1, 3, 2, 2, 3, 3),
  date = c(1, 2, 3, 2, 1, 3, 4, 1),
  price = c(1, 3, 2, 3, 1, 1, 1, 2)
)

pairs <- rs_pairs(x$date, x$id)
```

```
x[c("date_prev", "price_prev")] <- x[c("date", "price")][pairs, ]

x
```

`rs_var` *Robust variance matrix for repeat-sales indexes*

Description

Convenience function to compute a cluster-robust variance matrix for a linear regression, with or without instruments, where clustering occurs along one dimension. Useful for calculating a variance matrix when a regression is calculated manually.

Usage

```
rs_var(u, Z, X = Z, ids = seq_len(nrow(X)), df = NULL)
```

Arguments

`u`	`[numeric]` An $n \times 1$ vector of residuals from a linear regression.
`Z`	`[matrix]` An $n \times k$ matrix of instruments.
`X`	`[matrix]` An $n \times k$ matrix of covariates.
`ids`	`[factor]` A factor of length n, or something that can be coerced into one, that groups observations in `u`. By default each observation belongs to its own group.
`df`	`[numeric(1) > 0]` An optional degrees of freedom correction. Default is Stata's small sample degrees of freedom correction.

Details

This function calculates the standard robust variance matrix for a linear regression, as in Manski (1988, section 8.1.2); that is, $(Z'X)^{-1}V(X'Z)^{-1}$. It is useful when a regression is calculated by hand. This generalizes the variance matrix proposed by Shiller (1991, section II) when a property sells more than twice.

This function gives the same result as `vcovHC(x, type = "sss", cluster = 'group')` from the **plm** package.

Value

A $k \times k$ covariance matrix.

References

Manski C (1988). *Analog Estimation Methods in Econometrics*. Chapman and Hall.

Shiller RJ (1991). "Arithmetic repeat sales price estimators." *Journal of Housing Economics*, **1**(1), 110-126. doi:10.1016/S10511377(05)800282.

Examples

```
# Makes some groups in mtcars
mtcars$clust <- letters[1:4]

# Matrices for regression
x <- model.matrix(~ cyl + disp, mtcars)
y <- matrix(mtcars$mpg)

# Regression coefficients
b <- solve(crossprod(x), crossprod(x, y))

# Residuals
r <- y - x %*% b

# Robust variance matrix
vcov <- rs_var(r, x, ids = mtcars$clust)

## Not run:
# Same as plm
library(plm)
mdl <- plm(mpg ~ cyl + disp, mtcars, model = "pooling", index = "clust")
vcov2 <- vcovHC(mdl, type = "sss", cluster = "group")
vcov - vcov2

## End(Not run)
```

Index

base::order(), *2*, *4*

rs_matrix, 2
rs_matrix(), *4*
rs_pairs, 4
rs_pairs(), *3*
rs_var, 5