Package 'hpiR'

February 12, 2020

Type Package

Title House Price Indexes

Version 0.3.1

Maintainer Andy Krause <andyxkrause@gmail.com>

Description Compute house price indexes and series using a variety of different methods and models common through the real estate literature. Evaluate index 'goodness' based on accuracy, volatility and revision statistics. Background on basic model construction for repeat sales models can be found at: Case and Quigley (1991) <https://ideas.repec.org/a/tpr/restat/v73y1991i1p50-58.html> and for hedonic pricing models at: Bourassa et al (2006) <doi:10.1016/j.jhe.2006.03.001>. The package author's working paper on the random forest approach to house price indexes can be found at: <http://www.github.com/andykrause/hpi_research>.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

Imports dplyr, magrittr, lubridate, robustbase, ggplot2, imputeTS (>= 3.0), purrr, forecast, gridExtra, MASS, rlang, plyr, zoo, ranger, pdp

URL https://www.github.com/andykrause/hpiR

RoxygenNote 6.1.1

Suggests markdown, testthat, covr, knitr

VignetteBuilder knitr

NeedsCompilation no

Author Andy Krause [aut, cre]

Repository CRAN

Date/Publication 2020-02-12 16:00:07 UTC

R topics documented:

`buildForecastIDs` *Create the row IDs for forecast accuracy*

Description

Generate a vector of row IDs for use in forecast accuracy tests

Usage

```
buildForecastIDs(time_cut, hpi_df, forecast_length = 1, train = TRUE)
```

Arguments

`time_cut`	Period after which to cut off data
`hpi_df`	Data to be converted to training or scoring
`forecast_length`	default = 1; Length of forecasting to do
`train`	Default=TRUE; Create training data? FALSE = Scoring data

Value

vector of row_ids indicating inclusion in the forecasting data as either the training set (train = TRUE) or the scoring set (train = FALSE)

Further Details

This function is rarely (if ever) used directly. Most often called by ‘calcForecastError()‘

It is a generic method that dispatches on the ‘hpi_df‘ object.

Examples

```
 # Load example sales
data(ex_sales)

# Create RT data
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')

# Create ids
  fc_ids <- buildForecastIDs(time_cut = 27,
                             hpi_df = rt_data,
                             forecast_length = 2,
                             train = TRUE)
```

`buildForecastIDs.heddata` *Create the row IDs for forecast accuracy (hed approach)*

Description

Generate a vector of row IDs for use in forecast accuracy tests (hed approach)

Usage

```
## S3 method for class 'heddata'
buildForecastIDs(time_cut, hpi_df, forecast_length = 1,
  train = TRUE)
```

Arguments

`time_cut`	Period after which to cut off data
`hpi_df`	Data to be converted to training or scoring
`forecast_length`	default = 1; Length of forecasting to do
`train`	Default=TRUE; Create training data? FALSE = Scoring data

buildForecastIDs.rtdata *Create the row IDs for forecast accuracy (rt approach)*

Description

Generate a vector of row IDs for use in forecast accuracy tests (rt approach)

Usage

```
## S3 method for class 'rtdata'
buildForecastIDs(time_cut, hpi_df, forecast_length = 1,
  train = TRUE)
```

Arguments

time_cut	Period after which to cut off data
hpi_df	Data to be converted to training or scoring
forecast_length	default = 1; Length of forecasting to do
train	Default=TRUE; Create training data? FALSE = Scoring data

calcAccuracy *Calculate the accuracy of an index*

Description

Estimate index accuracy using one of a variety of approaches

Usage

```
calcAccuracy(hpi_obj, test_method = "insample", test_type = "rt",
  pred_df = NULL, smooth = FALSE, in_place = FALSE,
  in_place_name = "accuracy", ...)
```

Arguments

hpi_obj	Object of class 'hpi'
test_method	default = 'insample'; Also 'kfold'
test_type	default = 'rt'; Type of data to use for test. See details.
pred_df	default = NULL; Extra data if the test_type doesn't match data in hpi_obj
smooth	default = FALSE; calculated on the smoothed index(es)
in_place	default = FALSE; Should the result be returned into an existing 'hpi' object
in_place_name	default = 'accuracy'; Name for returning in place
...	Additional Arguments

Value

object of class ‘hpiaccuracy‘ inheriting from class ‘data.frame‘ containing the following fields:

prop_id Property Identification number

price Transaction Price

pred_price Predicted price

error (Prediction - Actual) / Actual

log_error log(prediction) - log(actual)

pred_period Period of the prediction

Further Details

’rt’ test type tests the ability of the index to correctly predict the second value in a repeat transaction pair FUTURE: ’hed’ test type tests the ability of the index to improve an OLS model that doesn’t account for time. (This approach is not ready yet).

Examples

```
# Load Data
data(ex_sales)

# Create Index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Calculate insample accuracy
hpi_accr <- calcAccuracy(hpi_obj = rt_index,
                         test_type = 'rt',
                         test_method = 'insample')
```

`calcForecastError` *Calculate the forecast accuracy of series of indexes*

Description

Estimate the index accuracy with forecasting for a (progressive) series of indexes

Usage

```
calcForecastError(is_obj, pred_df, return_forecasts = FALSE,
  forecast_length = 1, ...)
```

Arguments

`is_obj`	Object of class 'hpiseries'
`pred_df`	Set of sales to be used for predictive quality of index
`return_forecasts`	default = FALSE; return the forecasted indexes
`forecast_length`	default = 1; Length of period(s) in time to forecast
`...`	Additional Arguments

Value

object of class 'hpiaccuracy' inheriting from class 'data.frame' containing the following fields:

prop_id Property Identification number

price Transaction Price

pred_price Predicted price

error (Prediction - Actual) / Actual

log_error log(prediction) - log(actual)

pred_period Period of the prediction

series Series position from which the prediction was generated

Further Details

If you set 'return_forecasts' = TRUE, the forecasted indexes for each period will be returned in the 'forecasts' attribute of the 'hpiaccuracy' object. (attr(accr_obj, 'forecasts')

For now, the 'pred_df' object must be a set of repeat transactions with the class 'rt', inheriting from 'hpidata'

Examples

```
# Load example sales
data(ex_sales)

# Create Index
hed_index <- hedIndex(trans_df = ex_sales,
                      periodicity = 'monthly',
                      max_date = '2011-12-31',
                      adj_type = 'clip',
                      date = 'sale_date',
                      price = 'sale_price',
                      trans_id = 'sale_id',
                      prop_id = 'pinx',
                      estimator = 'robust',
                      log_dep = TRUE,
                      trim_model = TRUE,
                      max_period = 24,
                      dep_var = 'price',
                      ind_var = c('tot_sf', 'beds', 'baths'),
                      smooth = FALSE)

# Create Series (Suppressing messages do to small sample size of this example)
 suppressMessages(
   hpi_series <- createSeries(hpi_obj = hed_index,
                              train_period = 12))

# Create Prediction data
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         max_date = '2011-12-31',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date',
                         min_period_dist = 12)


# Calculate forecast accuracty
fc_accr <- calcForecastError(is_obj = hpi_series,
                             pred_df = rt_data)
```

`calcInSampleError` *Calculate index errors in sample*

Description

Estimate the predictive error of an index via an in-sample approach.

Usage

```
calcInSampleError(pred_df, index, ...)
```

Arguments

pred_df	Set of sales against which to test predictions
index	Index (of class 'ts') to be tested for accuracy
...	Additional Arguments

Value

object of class 'hpiaccuracy' inheriting from class 'data.frame' containing the following fields:

pair_id Uniq Pair ID number

price Transaction Price

pred_price Predicted price

error (Prediction - Actual) / Actual

log_error log(prediction) - log(actual)

pred_period Period of the prediction

Further Details

In addition to being a stand-alone function, it is also used by 'calcForecastError' and 'calcKFold-Error"

Examples

```
# Load example data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Calculate accuracy
in_accr <- calcInSampleError(pred_df = rt_index$data,
```

```
                    index = rt_index$index$value)
```

`calcInSampleError.heddata` *Calculate index errors in sample (hed approach)*

Description

Estimate the predictive error of an index via an in-sample approach (hed approach)

Usage

```
## S3 method for class 'heddata'
calcInSampleError(pred_df, index, ...)
```

Arguments

`pred_df`	Set of sales against which to test predictions
`index`	Index (of class 'ts') to be tested for accuracy
`...`	Additional Arguments

`calcInSampleError.rtdata` *Calculate index errors in sample (rt approach)*

Description

Estimate the predictive error of an index via an in-sample approach (rt approach)

Usage

```
## S3 method for class 'rtdata'
calcInSampleError(pred_df, index, ...)
```

Arguments

`pred_df`	Set of sales against which to test predictions
`index`	Index (of class 'ts') to be tested for accuracy
`...`	Additional Arguments

`calcKFoldError` *Calculate index error with FKold (out of sample)*

Description

Use a KFold (out of sample) approach to estimate index accuracy

Usage

```
calcKFoldError(hpi_obj, pred_df, k = 10, seed = 1, smooth = FALSE,
  ...)
```

Arguments

`hpi_obj`	HPI object of class 'hpi'
`pred_df`	Data.frame of sales to be used for assessing predictive quality of index
`k`	default=10; Number of folds to apply to holdout process
`seed`	default=1; Random seed generator to control the folding process
`smooth`	default = FALSE; Calculate on the smoothed index
`...`	Additional Arguments

Value

object of class 'hpiaccuracy' inheriting from class 'data.frame' containing the following fields:

pair_id Unique Pair ID

price Transaction Price

pred_price Predicted price

error (Prediction - Actual) / Actual

log_error log(prediction) - log(actual)

pred_period Period of the prediction

Examples

```
# Load data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
```

```
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Create prediction data
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')

# Calc Accuracy
kf_accr <- calcKFoldError(hpi_obj = rt_index,
                          pred_df = rt_data,
                          k = 10,
                          seed = 123,
                          smooth = FALSE)
```

`calcRevision` *Calculate revision values of an index*

Description

Create estimates of the revision statistics for a house price index

Usage

```
calcRevision(series_obj, in_place = FALSE, in_place_name = "rev",
  smooth = FALSE, ...)
```

Arguments

`series_obj`	A list of progressively longer indexes (a 'serieshpi" object from 'createSeries()")
`in_place`	default = FALSE; Calculating in place (adding to hpi)
`in_place_name`	default = 'rev'; Name of revision object in_place
`smooth`	default = FALSE; Use smoothed indexes
`...`	Additional Arguments

Value

list of length 3 containing:

period Data.frame containing the period number, mean and median for that period

mean Mean revision for all periods

median Median revision for all periods

Further Details

The revision object can be generate "in place" inside of the ‘serieshpi‘ object by setting ‘in_place‘ equal to TRUE.

Examples

```
# Load example sales
 data(ex_sales)

 # Create Index
 rt_index <- rtIndex(trans_df = ex_sales,
                     periodicity = 'monthly',
                     min_date = '2010-06-01',
                     max_date = '2015-11-30',
                     adj_type = 'clip',
                     date = 'sale_date',
                     price = 'sale_price',
                     trans_id = 'sale_id',
                     prop_id = 'pinx',
                     estimator = 'robust',
                     log_dep = TRUE,
                     trim_model = TRUE,
                     max_period = 48,
                     smooth = FALSE)

 # Create Series (Suppressing messages do to small sample size of this example)
  suppressMessages(
    hpi_series <- createSeries(hpi_obj = rt_index,
                               train_period = 12))

 # Calculate revision
 series_rev <-  calcRevision(series_obj = hpi_series)
```

---

calcSeriesAccuracy *Calculate the accuracy of a series of indexes*

Description

Estimate the index accuracy for a (progressive) series of indexes

Usage

```
calcSeriesAccuracy(series_obj, test_method = "insample",
  test_type = "rt", pred_df = NULL, smooth = FALSE,
  summarize = FALSE, in_place = FALSE, in_place_name = "accuracy",
  ...)
```

Arguments

series_obj	Serieshpi object to be analyzed
test_method	default = 'insample'; Also 'kfold' or 'forecast'
test_type	default = 'rt'; Type of data to use for test. See details.
pred_df	default = NULL; Extra data if the test_type doesn't match data in hpi_obj
smooth	default = FALSE; Analyze the smoothed indexes
summarize	default = FALSE; When multiple accuracy measurements for single observation take the mean of them all.
in_place	default = FALSE; Should the result be returned into an existing 'hpi' object
in_place_name	default = 'accuracy'; Name for returning in place
...	Additional Arguments

Value

'seriesaccuracy' object (unless calculated 'in_place')

Further Details

Unless using 'test_method = "forecast"'' with a "forecast_length" of 1, the results will have more than one accuracy estimate per observations. Setting 'summarize = TRUE' will take the mean accuracy for each observation across all indexes.

Examples

```
# Load data
data(ex_sales)

# Create index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
```

```
                    max_period = 48,
                    smooth = FALSE)

#  Create Series (Suppressing messages do to small sample size of this example)
suppressMessages(
  hpi_series <- createSeries(hpi_obj = rt_index,
                             train_period = 12))

# Calculate insample accuracy
hpi_series_accr <- calcSeriesAccuracy(series_obj = hpi_series,
                                      test_type = 'rt',
                                      test_method = 'insample')
```

`calcSeriesVolatility` *Calculate volatility of a series of indexes*

Description

Calculates volatility over a (progressive) series of indexes

Usage

```
calcSeriesVolatility(series_obj, window = 3, smooth = FALSE,
  in_place_name = "volatility", ...)
```

Arguments

`series_obj`	Series object to be calculated
`window`	default = 3; Rolling periods over which to calculate the volatility
`smooth`	default = FALSE; Also calculate volatilities for smoothed indexes
`in_place_name`	name if saving in place
`...`	Additional Arguments

Value

'serieshpi' object

Further Details

Leaving order blank default to a moving average with order 3.

Examples

```
# Load example sales
 data(ex_sales)

 # Create Index
 rt_index <- rtIndex(trans_df = ex_sales,
                     periodicity = 'monthly',
                     min_date = '2010-06-01',
                     max_date = '2015-11-30',
                     adj_type = 'clip',
                     date = 'sale_date',
                     price = 'sale_price',
                     trans_id = 'sale_id',
                     prop_id = 'pinx',
                     estimator = 'robust',
                     log_dep = TRUE,
                     trim_model = TRUE,
                     max_period = 48,
                     smooth = FALSE)

 # Create Series (Suppressing messages do to small sample size of this example)
 suppressMessages(
    hpi_series <- createSeries(hpi_obj = rt_index,
                               train_period = 12))

 # Calculate series volatility
 series_vol <- calcSeriesVolatility(series_obj = hpi_series,
                                    window= 3)
```

`calcVolatility` *Calculate index volatility*

Description

Create estimate of index volatility given a window

Usage

```
calcVolatility(index, window = 3, in_place = FALSE,
  in_place_name = "volatility", smooth = FALSE, ...)
```

Arguments

`index`	An object of class 'hpiindex'
`window`	default = 3; Rolling periods over which to calculate the volatility
`in_place`	default = FALSE; Adds volatility metric to the 'hpiindex' object (may be within an 'hpi' object)

in_place_name	default = 'vol'; Name of volatility object in 'hpiindex' object
smooth	default = FALSE; Calculate on the smoothed index?
...	Additional arguments

Value

an 'indexvolatility' (S3) object, the 'index' slot of which is a 'ts' object

roll volatility at each rolling point

mean overall mean volatility

median overall median volatility

Further Details

You may also provide an 'hpi' object to this function. If you do, it will extract the 'hpiindex' object from the 'index' slot in the 'hpi' class object.

Examples

```
# Load Data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Calculate Volatility
index_vol <- calcVolatility(index = rt_index,
                            window = 3)
```

checkDate *Validate the date argument*

Description

Internal function to validate (or convert) the provided date field

Usage

```
checkDate(x_date, name)
```

Arguments

x_date	Date string or vector
name	Name of argument to return in error/warning message

Value

Adjusted date field

Examples

```
# Load Data
 data(ex_sales)

# Check date
 date_checked <- checkDate(x_date = ex_sales$sale_date,
                           name = 'sale date')
```

createKFoldData *Create data for KFold error test*

Description

Generic method for creating KFold testing data

Usage

```
createKFoldData(score_ids, full_data, pred_df)
```

Arguments

score_ids	Vector of row ids to be included in scoring data
full_data	Complete dataset (class 'hpidata") of this model type (rt or hed)
pred_df	Data to be used for prediction

Value

list of length 2 containing:

train Training data.frame

score Scoring data.frame

Further Details

Called from ‘calcKFoldError()“

Examples

```
# Load Data
data(ex_sales)

# Create RT Data
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')
# Create folds
k_folds <- split(x = 1:nrow(rt_data),
                 f = sample(1:10, nrow(rt_data), replace = TRUE))

# Create data from folds
kfold_data <- createKFoldData(score_ids = k_folds[[1]],
                              full_data = rt_data,
                              pred_df = rt_data)
```

---

createKFoldData.rtdata

Create data for KFold error test (rt approach)

Description

‘rtdata‘ method for creating KFold testing data

Usage

```
## S3 method for class 'rtdata'
createKFoldData(score_ids, full_data, pred_df)
```

Arguments

score_ids	Vector of row ids to be included in scoring data
full_data	Complete dataset (class 'hpidata") of this model type (rt or hed)
pred_df	Data to be used for prediction

---

createSeries *Create a series of indexes*

Description

Generate a series of progressive indexes

Usage

```
createSeries(hpi_obj, train_period = 12, max_period = NULL, ...)
```

Arguments

hpi_obj	Object of class 'hpi'
train_period	default = 12; Number of periods to use as purely training before creating indexes
max_period	default = NULL; Maximum number of periods to create the index up to
...	Additional Arguments

Value

An 'serieshpi' object – a list of 'hpi' objects.

Further Details

'train_period' Represents the shortest index that you will create. For certain approaches, such as a repeat transaction model, indexes shorter than 10 will likely be highly unstable.

If 'max_period" is left NULL, then it will forecast up to the end of the data.

Examples

```
 # Load example sales
data(ex_sales)

# Create Index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
```

```
                      price = 'sale_price',
                      trans_id = 'sale_id',
                      prop_id = 'pinx',
                      estimator = 'robust',
                      log_dep = TRUE,
                      trim_model = TRUE,
                      max_period = 48,
                      smooth = FALSE)

# Create Series (Suppressing messages do to small sample size of this example)
 suppressMessages(
   hpi_series <- createSeries(hpi_obj = rt_index,
                              train_period = 12))
```

`dateToPeriod` *Convert dates to a relative period*

Description

Create a relative period variable from a date variable

Usage

```
dateToPeriod(trans_df, date, periodicity = NULL, min_date = NULL,
  max_date = NULL, adj_type = "move", ...)
```

Arguments

`trans_df`	data.frame of raw transactions
`date`	name of field containing the date of the sale in Date or POSIXt format
`periodicity`	type of periodicity to use ('yearly', 'quarterly', 'monthly' or 'weekly)
`min_date`	default = NULL; optional minimum date to use
`max_date`	default = NULL; optional maximum date to use
`adj_type`	default = 'move'; how to handle min and max dates within the range of transactions. 'move' min and/or max date or 'clip' the data
`...`	Additional arguments

Value

original data frame (‘trans_df‘ object) with two new fields: trans_period: integer value counting from the minimum transaction date in the periodicity selected. Base value is 1. Primarily for modeling trans_date: properly formatted transaction date

Further Details

"trans_period" counts from the minimum transaction date provided. As such the period counts are relative, not absolute

Additionally, this function modifies the data.frame that it is given and return that same data.frame that it is given and returns that data.frame with the new fields attached.

Examples

```
# Load data
  data(ex_sales)

# Convert to period df
  hpi_data <- dateToPeriod(trans_df = ex_sales,
                           date = 'sale_date',
                           periodicity = 'monthly')
```

---

ex_sales *Subset of Seattle Home Sales*

Description

Seattle home sales from areas 13, 14,an 15 (central Seattle) 2010 to 2016. Includes only detached single family residences and townhomes. Data gathered from the King County Assessor's FTP site. A number of initial data munging tasks were necessary to bring the data into this format.

Usage

```
data(ex_sales)
```

Format

A "data.frame" with 5,348 rows and 16 variables

pinx The unique property identifying code. Original value is preceded by two '..'s to prevent the dropping of leading zeros

sale_id The unique transaction identifying code.

sale_price Price of the home

sale_date Date of sale

use_type Property use type

area Assessment area or zone

lot_sf Size of lot in square feet

wfnt Is property waterfront?

bldg_grade Quality of the building construction (higher is better)

tot_sf Size of home in square feet

beds Number of bedrooms

baths Number of bathrooms

age Age of home

eff_age Age of home, considering major remodels

longitude Longitude

latitude Latitude

Source

King County Assessor: http://info.kingcounty.gov/assessor/DataDownload/

`hedCreateTrans` *Create data for 'hed' approach*

Description

Generate standardized data for the 'hed' modeling approach

Usage

```
hedCreateTrans(trans_df, prop_id, trans_id, price, date = NULL,
  periodicity = NULL, ...)
```

Arguments

`trans_df`	sales transaction in either a data.frame or a trans_df class from dateToPeriod() function
`prop_id`	field contain the unique property identification
`trans_id`	field containing the unique transaction identification
`price`	field containing the transaction price
`date`	default=NULL, field containing the date of the transaction. Only necessary if not passing an 'hpidata' object
`periodicity`	default=NULL, field containing the desired periodicity of analysis. Only necessary if not passing a 'hpidata' object
`...`	Additional arguments

Value

data.frame of transactions with standardized period field. Note that a full data.frame of the possible periods, their values and names can be found in the attributes to the returned 'hed' object

Examples

```
# Load example data
data(ex_sales)

# Create Hed Data
ex_heddata <- hedCreateTrans(trans_df = ex_sales,
                             prop_id = 'pinx',
                             trans_id = 'sale_id',
                             price = 'sale_price',
                             date = 'sale_date',
                             periodicity = 'monthly')
```

---

hedIndex *Create a full index object by hedonic approach*

Description

Wrapper to create index object via entire hedonic approach

Usage

```
hedIndex(trans_df, dep_var = NULL, ind_var = NULL, hed_spec = NULL,
  ...)
```

Arguments

trans_df	data.frame of transactions
dep_var	default = NULL; Dependent variable in hedonic model
ind_var	default = NULL; Independent variables in the hedonic model
hed_spec	default = NULL; Full hedonic model specification
...	Additional Arguments

Value

'hpi" object. S3 list with:

data 'hpidata' object

model 'hpimodel' object

index 'hpiindex' object

Further Details

Additional argument need to provide necessary argument for create 'hpidata' objects if the 'trans_df' object is not of that class.

Examples

```
# Load data
data(ex_sales)

# Create index with raw transaction data
hed_index <- hedIndex(trans_df = ex_sales,
                      periodicity = 'monthly',
                      min_date = '2010-06-01',
                      max_date = '2015-11-30',
                      adj_type = 'clip',
                      date = 'sale_date',
                      price = 'sale_price',
                      trans_id = 'sale_id',
                      prop_id = 'pinx',
                      estimator = 'robust',
                      log_dep = TRUE,
                      trim_model = TRUE,
                      max_period = 48,
                      dep_var = 'price',
                      ind_var = c('tot_sf', 'beds', 'baths'),
                      smooth = FALSE)
```

---

hedModel *Estimate hedonic model for index creation*

Description

Estimate coefficients for an index via the hedonic approach (generic method)

Usage

```
hedModel(estimator, hed_df, hed_spec, ...)
```

Arguments

estimator	Type of model to estimates (base, robust, weighted)
hed_df	Repeat sales dataset from hedCreateSales()
hed_spec	Model specification (‘formula‘ object)
...	Additional arguments

Value

‘hedmodel‘ object: model object of the estimator (ex.: ‘lm‘)

Further Details

'estimator' argument must be in a class of 'base', 'weighted' or 'robust' This function is not generally called directly, but rather from 'hpiModel()'

Examples

```
# Load example data
data(ex_sales)

# Create hedonic data
hed_data <- hedCreateTrans(trans_df = ex_sales,
                           prop_id = 'pinx',
                           trans_id = 'sale_id',
                           price = 'sale_price',
                           date = 'sale_date',
                           periodicity = 'monthly')

# Estimate Model
hed_model <- hedModel(estimator = structure('base', class = 'base'),
                      hed_df = hed_data,
                      hed_spec = as.formula(log(price) ~ baths + tot_sf))
```

---

hedModel.base *Hedonic model approach with base estimator*

Description

Use of base estimator in hedonic model approach

Usage

```
## S3 method for class 'base'
hedModel(estimator, hed_df, hed_spec, ...)
```

Arguments

estimator	Type of model to estimates (base, robust, weighted)
hed_df	Repeat sales dataset from hedCreateSales()
hed_spec	Model specification ('formula' object)
...	Additional arguments

Further Details

See '?hedModel' for more information

hedModel.robust *Hedonic model approach with robust estimator*

Description

Use of robust estimator in hedonic model approach

Usage

```
## S3 method for class 'robust'
hedModel(estimator, hed_df, hed_spec, ...)
```

Arguments

estimator	Type of model to estimates (base, robust, weighted)
hed_df	Repeat sales dataset from hedCreateSales()
hed_spec	Model specification (‘formula‘ object)
...	Additional arguments

Further Details

See ‘?hedModel‘ for more information

See ‘?hedModel‘ for more information

hedModel.weighted *Hedonic model approach with weighted estimator*

Description

Use of weighted estimator in hedonic model approach

Usage

```
## S3 method for class 'weighted'
hedModel(estimator, hed_df, hed_spec, ...)
```

Arguments

estimator	Type of model to estimates (base, robust, weighted)
hed_df	Repeat sales dataset from hedCreateSales()
hed_spec	Model specification (‘formula‘ object)
...	Additional arguments

Further Details

See ‘?hedModel‘ for more information

`hpiModel` *Wrapper to estimate model approaches (generic method)*

Description

Generic method to estimate modeling approaches for indexes

Usage

```
hpiModel(model_type, hpi_df, estimator = "base", log_dep = TRUE,
  trim_model = TRUE, mod_spec = NULL, ...)
```

Arguments

`model_type`	Type of model to estimate ('rt', 'hed', 'rf')
`hpi_df`	Dataset created by one of the *CreateTrans() function in this package.
`estimator`	Type of estimator to be used ('base', 'weighted', 'robust')
`log_dep`	default TRUE, should the dependent variable (change in price) be logged?
`trim_model`	default TRUE, should excess be trimmed from model results ('lm' or 'rlm' object)?
`mod_spec`	Model specification
`...`	Additional Arguments

Value

hpimodel object consisting of:

estimator Type of estimator

coefficients Data.frame of coefficient

model_obj class 'rtmodel' or 'hedmodel'

mod_spec Full model specification

log_dep Binary: is the dependent variable in logged format

base_price Mean price in the base period

periods 'data.frame' of periods

approach Type of model used

Examples

```
# Load data
data(ex_sales)

# With a raw transaction data.frame
rt_data <- rtCreateTrans(trans_df = ex_sales,
```

```
                          prop_id = 'pinx',
                          trans_id = 'sale_id',
                          price = 'sale_price',
                          periodicity = 'monthly',
                          date = 'sale_date')

# Create model object
hpi_model <- hpiModel(model_type = 'rt',
                      hpi_df = rt_data,
                      estimator = 'base',
                      log_dep = TRUE)

# For custom weighted repeat transaction model

hpi_model_wgt <- hpiModel(model_type = 'rt',
                          hpi_df = rt_data,
                          estimator = 'weighted',
                          weights = runif(nrow(rt_data), 0, 1))
```

`hpiModel.hed` *Specific method for hpi modeling (hed) approach)*

Description

Estimate hpi models with hed approach

Usage

```
## S3 method for class 'hed'
hpiModel(model_type, hpi_df, estimator = "base",
  log_dep = TRUE, trim_model = TRUE, mod_spec = NULL,
  dep_var = NULL, ind_var = NULL, ...)
```

Arguments

`model_type`	Type of model to estimate ('rt', 'hed', 'rf')
`hpi_df`	Dataset created by one of the *CreateSales() function in this package.
`estimator`	Type of estimator to be used ('base', 'weighted', 'robust')
`log_dep`	default=TRUE; should the dependent variable (change in price) be logged?
`trim_model`	default TRUE, should excess be trimmed from model results ('lm' or 'rlm' object)?
`mod_spec`	default=NULL; hedonic model specification
`dep_var`	default=NULL; dependent variable of the model
`ind_var`	default=NULL; independent variable(s) of the model
`...`	Additional Arguments

Value

hpimodel object consisting of:

estimator Type of estimator

coefficients Data.frame of coefficient

model_obj class ‘rtmodel‘ or ‘hedmodel‘

mod_spec Full model specification

log_dep Binary: is the dependent variable in logged format

base_price Mean price in the base period

periods ‘data.frame‘ of periods

approach Type of model used

`hpiModel.rf` *Specific method for hpi modeling (hed) approach)*

Description

Estimate hpi models with hed approach

Usage

```
## S3 method for class 'rf'
hpiModel(model_type, hpi_df, estimator = "pdp",
  log_dep = TRUE, trim_model = TRUE, mod_spec = NULL,
  dep_var = NULL, ind_var = NULL, ...)
```

Arguments

`model_type`	Type of model (’rt’, ’hed’, ’rf’)
`hpi_df`	Dataset created by one of the *CreateSales() function in this package.
`estimator`	Type of estimator to be used (’base’, ’weighted’, ’robust’)
`log_dep`	default=TRUE; should the dependent variable (change in price) be logged?
`trim_model`	default TRUE, should excess be trimmed from model results (’lm’ or ’rlm’ object)?
`mod_spec`	default=NULL; hedonic model specification
`dep_var`	default=NULL; dependent variable of the model
`ind_var`	default=NULL; independent variable(s) of the model
`...`	Additional Arguments

Value

hpimodel object consisting of:

estimator Type of estimator

coefficients Data.frame of coefficient

model_obj class ‘rtmodel‘ or ‘hedmodel‘

mod_spec Full model specification

log_dep Binary: is the dependent variable in logged format

base_price Mean price in the base period

periods ‘data.frame‘ of periods

approach Type of model used

`hpiModel.rt` *Specific method for hpi modeling (rt approach)*

Description

Estimate hpi models with rt approach

Usage

```
## S3 method for class 'rt'
hpiModel(model_type, hpi_df, estimator = "base",
  log_dep = TRUE, trim_model = TRUE, mod_spec = NULL, ...)
```

Arguments

`model_type`	Type of model to estimate (’rt’, ’hed’, ’rf’)
`hpi_df`	Dataset created by one of the *CreateTrans() function in this package.
`estimator`	Type of estimator to be used (’base’, ’weighted’, ’robust’)
`log_dep`	default TRUE, should the dependent variable (change in price) be logged?
`trim_model`	default TRUE, should excess be trimmed from model results (’lm’ or ’rlm’ object)?
`mod_spec`	Model specification
`...`	Additional Arguments

Value

hpimodel object consisting of:

estimator Type of estimator
coefficients Data.frame of coefficient
model_obj class 'rtmodel' or 'hedmodel'
mod_spec Full model specification
log_dep Binary: is the dependent variable in logged format
base_price Mean price in the base period
periods 'data.frame' of periods
approach Type of model used

`hpiR` *hpiR: A package for house price indexes*

Description

House Price Indexes in R: A set of tools to create house price indexes and analyze their various performance metrics.

`matchKFold` *Helper function to make KFold data*

Description

Function to help create KFold data based on approach (Generic Method)

Usage

```
matchKFold(train_df, pred_df)
```

Arguments

`train_df`	Data.frame of training data
`pred_df`	Data.frame (class 'hpidata") to be used for prediction

Value

list

train Training data
score Scoring data

Further Details

Helper function called from createKFoldData

matchKFold.heddata *Helper function to make KFold data*

Description

Function to help create KFold data based on hed approach

Usage

```
## S3 method for class 'heddata'
matchKFold(train_df, pred_df)
```

Arguments

train_df	Data.frame of training data
pred_df	Data.frame (class 'hpidata") to be used for prediction

matchKFold.rtdata *Helper function to make KFold data*

Description

Function to help create KFold data based on rt approach

Usage

```
## S3 method for class 'rtdata'
matchKFold(train_df, pred_df)
```

Arguments

train_df	Data.frame of training data
pred_df	Data.frame (class 'hpidata") to be used for prediction

`modelToIndex` *Convert model results into a house price index*

Description

Converts model results to standardized index objects

Usage

```
modelToIndex(model_obj, max_period = max(model_obj$coefficients$time),
  ...)
```

Arguments

`model_obj`	Model results object
`max_period`	Maximum number of periods that should have been estimated.
`...`	Additional arguments

Value

'hpiindex' object containing:

`name`	vector of period names
`numeric`	vector of period in numeric form
`period`	vector of period numbers
`value`	'ts' object of the index values
`imputed`	vector of binary values indicating imputation

Examples

```
# Load data
data(ex_sales)

# With a raw transaction data.frame
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')

# Create model object
hpi_model <- hpiModel(model_type = 'rt',
                      hpi_df = rt_data,
                      estimator = 'base',
                      log_dep = TRUE)
```

```
# Create Index
hpi_index <- modelToIndex(hpi_model,
                          max_period = 84)
```

---

plot.hpi *Plot method for 'hpi' object*

Description

Specific plotting method for hpi objects

Usage

```
## S3 method for class 'hpi'
plot(x, ...)
```

Arguments

x	Object to plot of class 'hpi'
...	Additional Arguments

Value

'plotindex' object inheriting from a ggplot object

Further Details

Additional argument can include those argument for 'plot.hpindex"

Examples

```
# Load data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
```

```
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Plot data
plot(rt_index)
plot(rt_index, smooth = TRUE)
```

---

plot.hpiaccuracy *Plot method for 'hpiaccuracy' object*

Description

Specific plotting method for hpiaccuracy objects

Usage

```
## S3 method for class 'hpiaccuracy'
plot(x, return_plot = FALSE, do_plot = TRUE,
  use_log_error = FALSE, ...)
```

Arguments

x	Object to plot of class 'hpiaccuracy"
return_plot	default = FALSE; Return the plot to the function call
do_plot	default = FALSE; Execute plotting to terminal/console
use_log_error	[FALSE] Use the log error?
...	Additional Arguments

Value

'plotaccuracy' object inheriting from a ggplot object

Examples

```
# Load Data
data(ex_sales)

# Create Index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
```

```
                      trans_id = 'sale_id',
                      prop_id = 'pinx',
                      estimator = 'robust',
                      log_dep = TRUE,
                      trim_model = TRUE,
                      max_period = 48,
                      smooth = FALSE)

# Calculate insample accuracy
hpi_accr <- calcAccuracy(hpi_obj = rt_index,
                         test_type = 'rt',
                         test_method = 'insample')

# Make Plot
plot(hpi_accr)
```

`plot.hpiindex` *Plot method for 'hpiindex' object*

Description

Specific plotting method for hpiindex objects

Usage

```
## S3 method for class 'hpiindex'
plot(x, show_imputed = FALSE, smooth = FALSE, ...)
```

Arguments

`x`	Object to plot of class 'hpiindex"
`show_imputed`	default = FALSE; highlight the imputed points
`smooth`	default = FALSE; plot the smoothed index
`...`	Additional Arguments

Value

'plotindex' object inheriting from a ggplot object

Examples

```
# Load data
data(ex_sales)

# With a raw transaction data.frame
rt_data <- rtCreateTrans(trans_df = ex_sales,
```

```
                          prop_id = 'pinx',
                          trans_id = 'sale_id',
                          price = 'sale_price',
                          periodicity = 'monthly',
                          date = 'sale_date')

# Create model object
hpi_model <- hpiModel(model_type = 'rt',
                      hpi_df = rt_data,
                      estimator = 'base',
                      log_dep = TRUE)

# Create Index
hpi_index <- modelToIndex(hpi_model,
                          max_period = 84)

# Make Plot
plot(hpi_index)
```

`plot.indexvolatility` *Plot method for 'indexvolatility' object*

Description

Specific plotting method for indexvolatility objects

Usage

```
## S3 method for class 'indexvolatility'
plot(x, ...)
```

Arguments

`x`	Object to plot of class 'indexvolatility"
`...`	Additional Arguments

Value

'plotvolatility' object inheriting from a ggplot object

Examples

```
# Load Data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
```

```
                      periodicity = 'monthly',
                      min_date = '2010-06-01',
                      max_date = '2015-11-30',
                      adj_type = 'clip',
                      date = 'sale_date',
                      price = 'sale_price',
                      trans_id = 'sale_id',
                      prop_id = 'pinx',
                      estimator = 'robust',
                      log_dep = TRUE,
                      trim_model = TRUE,
                      max_period = 48,
                      smooth = FALSE)

# Calculate Volatility
index_vol <- calcVolatility(index = rt_index,
                            window = 3)

# Make Plot
plot(index_vol)
```

`plot.seriesaccuracy` *Plot method for 'seriesaccuracy' object*

Description

Specific plotting method for seriesaccuracy objects

Usage

```
## S3 method for class 'seriesaccuracy'
plot(x, return_plot = FALSE, ...)
```

Arguments

`x`	Object of class 'hpiaccuracy"
`return_plot`	default = FALSE; Return the plot to the function call
`...`	Additional argument (passed to 'plot.hpiaccuracy()")

Value

'plotaccuracy' object inheriting from a ggplot object

Examples

```
# Load data
data(ex_sales)

# Create index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

#  Create Series (Suppressing messages do to small sample size of this example)
suppressMessages(
   hpi_series <- createSeries(hpi_obj = rt_index,
                              train_period = 12))

# Calculate insample accuracy
hpi_series_accr <- calcSeriesAccuracy(series_obj = hpi_series,
                                      test_type = 'rt',
                                      test_method = 'insample')
# Make Plot
plot(hpi_series_accr)
```

`plot.serieshpi` *Plot method for 'serieshpi' object*

Description

Specific plotting method for serieshpi objects

Usage

```
## S3 method for class 'serieshpi'
plot(x, smooth = FALSE, ...)
```

Arguments

`x`	Object of class 'serieshpi'

smooth	default = FALSE; plot the smoothed object
...	Additional Arguments‘

Value

‘plotseries‘ object inheriting from a ggplot object

Examples

```
# Load data
data(ex_sales)

# Create index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Create Series (Suppressing messages do to small sample size of this example)
suppressMessages(
  hpi_series <- createSeries(hpi_obj = rt_index,
                             train_period = 12))

# Make Plot
plot(hpi_series)
```

---

plot.seriesrevision *Plot method for ‘seriesrevision‘ object*

Description

Specific plotting method for seriesrevision objects

Usage

```
## S3 method for class 'seriesrevision'
plot(x, measure = "median", ...)
```

Arguments

x	Object to plot of class ‘seriesrevision‘
measure	default = ’median’; Metric to plot (’median’ or ’mean’)
...	Additional Arguments

Value

‘plotrevision‘ object inheriting from a ggplot object

Examples

```
# Load example sales
data(ex_sales)

# Create Index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

# Create Series (Suppressing messages do to small sample size of this example)
suppressMessages(
  hpi_series <- createSeries(hpi_obj = rt_index,
                             train_period = 12))

# Calculate revision
series_rev <-  calcRevision(series_obj = hpi_series)

# Make Plot
plot(series_rev)
```

---

rfIndex *Create a full index object by random forest approach*

Description

Wrapper to create index object via entire random forest approach

Usage

```
rfIndex(trans_df, dep_var = NULL, ind_var = NULL, rf_spec = NULL,
  ...)
```

Arguments

trans_df	data.frame of transactions
dep_var	default = NULL; Dependent variable in hedonic model
ind_var	default = NULL; Independent variables in the hedonic model
rf_spec	default = NULL; Full random forest model specification
...	Additional Arguments

Value

'hpi" object. S3 list with:

data 'hpidata' object

model 'hpimodel' object

index 'hpiindex' object

Further Details

Additional argument need to provide necessary argument for create 'hpidata' objects if the 'trans_df' object is not of that class.

Examples

```
# Load data
data(ex_sales)

# Create index with raw transaction data
rf_index <- rfIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'pdp',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    dep_var = 'price',
                    ind_var = c('tot_sf', 'beds', 'baths'),
                    smooth = FALSE,
```

```
                    ntrees = 10,
                    sim_count = 2)
```

`rfModel` *Estimate random forest model for index creation*

Description

Estimate coefficients for an index via the random forest approach (generic method)

Usage

```
rfModel(estimator, rf_df, rf_spec, ntrees = 200, seed = 1, ...)
```

Arguments

`estimator`	Type of model to estimates (pdp)
`rf_df`	Transactions dataset from hedCreateSales()
`rf_spec`	Model specification (‘formula‘ object)
`ntrees`	[200] Set number of trees to use
`seed`	[1] Random seed for reproducibility
`...`	Additional arguments

Value

‘rfmodel‘ object: model object of the estimator (ex.: ‘lm‘)

Further Details

‘estimator‘ argument must be in a class of ’pdp’ This function is not generally called directly, but rather from ‘hpiModel()‘

Examples

```
# Load example data
data(ex_sales)

# Create hedonic data
hed_data <- hedCreateTrans(trans_df = ex_sales,
                           prop_id = 'pinx',
                           trans_id = 'sale_id',
                           price = 'sale_price',
                           date = 'sale_date',
                           periodicity = 'monthly')

# Estimate Model
```

```
rf_model <- rfModel(estimator = structure('pdp', class = 'pdp'),
                    rf_df = hed_data,
                    rf_spec = as.formula(log(price) ~ baths + tot_sf),
                    ntrees = 10,
                    sim_count = 1)
```

`rfModel.pdp` *Random forest model approach with pdp estimator*

Description

Use of pdp estimator in random forest approach

Usage

```
## S3 method for class 'pdp'
rfModel(estimator, rf_df, rf_spec, ntrees = 200,
  seed = 1, ...)
```

Arguments

`estimator`	Type of model to estimates (pdp)
`rf_df`	Transactions dataset from hedCreateSales()
`rf_spec`	Model specification (‘formula‘ object)
`ntrees`	[200] Set number of trees to use
`seed`	[1] Random seed for reproducibility
`...`	Additional arguments

Further Details

See ‘?rfModel‘ for more information

`rfSimDf` *Create simulation data for Random forest approach*

Description

Create data to use in PDP simulation

Usage

```
rfSimDf(rf_df, seed, sim_ids = NULL, sim_count = NULL,
  sim_per = NULL, ...)
```

Arguments

rf_df	Full training dataset
seed	Random seed for reproducibility
sim_ids	row ids to simulate
sim_count	number of random rows to simulate
sim_per	percent of rows to randomly simulate
...	Additional arguments

Further Details

See '?rfModel' for more information

---

rtCreateTrans *Create transaction data for rt approach*

Description

Generate standardized object for rt estimate approach

Usage

```
rtCreateTrans(trans_df, prop_id, trans_id, price, date = NULL,
  periodicity = NULL, seq_only = FALSE, min_period_dist = NULL, ...)
```

Arguments

trans_df	transactions in either a data.frame or a 'hpidata" class from dateToPeriod() function
prop_id	field contain the unique property identification
trans_id	field containing the unique transaction identification
price	field containing the transaction price
date	default=NULL, field containing the date of the sale. Only necessary if not passing an 'hpidata' object
periodicity	default=NULL, field containing the desired periodicity of analysis. Only necessary if not passing a 'hpidata' object
seq_only	default=FALSE, indicating whether to only include sequential repeat observations 1 to 2 and 2 to 3. False returns 1 to 2, 1 to 3 and 2 to 3.
min_period_dist	[12] Minimum number of period required between repeat sales
...	Additional arguments

Value

data.frame of repeat transactions. Note that a full data.frame of the possible periods, their values and names can be found in the attributes to the returned ‘rtdata‘ object

Further Details

Properties with greater than two transactions during the period will make pairwise matches among all sales. Any property transacting twice in the same period will remove the lower priced of the two transactions. If passing a raw data.frame (not a ‘hpidata‘‘ object) the "date" field should refer to a field containing a vector of class POSIXt or Date.

Examples

```
# Load data
data(ex_sales)

# With a raw transaction data.frame
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')
```

---

rtIndex *Create a full index object by repeat transaction approach*

Description

Wrapper to create index object via entire repeat transaction approach

Usage

```
rtIndex(trans_df, ...)
```

Arguments

trans_df	data.frame of transactions. Can be a ’hpidata’ or an ’rtdata’ object.
...	Additional Arguments

Value

‘hpi‘‘ object. S3 list with:

data ‘hpidata‘ object

model ‘hpimodel‘ object

index ‘hpiindex‘ object

Further Details

Additional argument need to provide necessary argument for create ‘hpidata‘ objects if the ‘trans_df‘ object is not of that class.

Examples

```
# Load data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)
```

---

rtModel *Estimate repeat transaction model for index creation*

Description

Estimate coefficients for an index via the repeat transaction approach (generic method)

Usage

```
rtModel(rt_df, time_matrix, price_diff, estimator, lm_recover = TRUE,
  ...)
```

Arguments

rt_df	Repeat transactions dataset from rtCreateTrans()
time_matrix	Time matrix object from rtTimeMatrix()
price_diff	Difference in price between the two transactions
estimator	Type of model to estimates (base, robust, weighted). Must be in that class.
lm_recover	(TRUE) Allows robust model to use linear model if it fails
...	Additional arguments

Value

'rtmodel' object

Further Details

Three available specific methods: 'base', 'robust' and 'weighted'

Examples

```
# Load data
data(ex_sales)

# With a raw transaction data.frame
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')

# Calc price differences
price_diff <- rt_data$price_2 - rt_data$price_1

# Create time matrix
rt_matrix <- rtTimeMatrix(rt_data)

# Calculate model
rt_model <- rtModel(rt_df = rt_data,
                    price_diff = price_diff,
                    time_matrix = rt_matrix,
                    estimator = structure('base', class='base'))
```

---

rtModel.base *Repeat transaction model approach with base estimator*

Description

Use of base estimator in repeat transactions model approach

Usage

```
## S3 method for class 'base'
rtModel(rt_df, time_matrix, price_diff, estimator, ...)
```

Arguments

rt_df	Repeat transactions dataset from rtCreateTrans()
time_matrix	Time matrix object from rtTimeMatrix()
price_diff	Difference in price between the two transactions
estimator	Type of model to estimates (base, robust, weighted). Must be in that class.
...	Additional arguments

Further Details

See '?rtModel' for more information

---

rtModel.robust *Repeat transaction model approach with robust estimator*

Description

Use of robust estimator in repeat transactions model approach

Usage

```
## S3 method for class 'robust'
rtModel(rt_df, time_matrix, price_diff, estimator,
  lm_recover = TRUE, ...)
```

Arguments

rt_df	Repeat transactions dataset from rtCreateTrans()
time_matrix	Time matrix object from rtTimeMatrix()
price_diff	Difference in price between the two transactions
estimator	Type of model to estimates (base, robust, weighted). Must be in that class.
lm_recover	(TRUE) Allows robust model to use linear model if it fails
...	Additional arguments

Further Details

See '?rtModel' for more information

rtModel.weighted *Repeat transaction model approach with weighted estimator*

Description

Use of weighted estimator in repeat transactions model approach

Usage

```
## S3 method for class 'weighted'
rtModel(rt_df, time_matrix, price_diff, estimator, ...)
```

Arguments

rt_df	Repeat transactions dataset from rtCreateTrans()
time_matrix	Time matrix object from rtTimeMatrix()
price_diff	Difference in price between the two transactions
estimator	Type of model to estimates (base, robust, weighted). Must be in that class.
...	Additional arguments

Further Details

See '?rtModel' for more information

rtTimeMatrix *Create model matrix for repeat transaction approach*

Description

Generates the array necessary to estimate a repeat transactions model

Usage

```
rtTimeMatrix(rt_df)
```

Arguments

rt_df	object of class 'rtdata': repeat transaction data.frame created by rtCreateTrans()

Value

matrix to be used on the right hand side of a repeat sales regression model

Further Details

Time periods are calculated from the data provided.

Examples

```
# Load data
data(ex_sales)

# With a raw transaction data.frame
rt_data <- rtCreateTrans(trans_df = ex_sales,
                         prop_id = 'pinx',
                         trans_id = 'sale_id',
                         price = 'sale_price',
                         periodicity = 'monthly',
                         date = 'sale_date')
# Create Matrix
rt_matrix <- rtTimeMatrix(rt_data)
```

`seattle_sales` *Seattle Home Sales*

Description

Seattle home sales from 2010 to 2016. Includes only detached single family residences and townhomes. Data gathered from the King County Assessor's FTP site. A number of initial data munging tasks were necessary to bring the data into this format.

Usage

```
data(seattle_sales)
```

Format

A `"data.frame"` with 43,313 rows and 16 variables

pinx The unique property identifying code. Original value is preceded by two '..'s to prevent the dropping of leading zeros

sale_id The unique transaction identifying code.

sale_price Price of the home

sale_date Date of sale

use_type Property use type

area Assessment area or zone

lot_sf Size of lot in square feet

wfnt Is property waterfront?

bldg_grade Quality of the building construction (higher is better)

tot_sf Size of home in square feet

beds Number of bedrooms

baths Number of bathrooms

age Age of home

eff_age Age of home, considering major remodels

longitude Longitude

latitude Latitude

Source

King County Assessor: http://info.kingcounty.gov/assessor/DataDownload/

`smoothIndex` *Smooth an index*

Description

Smooths an existing hpiindex object

Usage

```
smoothIndex(index_obj, order = 3, in_place = FALSE, ...)
```

Arguments

`index_obj`	Index to be smoothed
`order`	default = 3; Number of nearby period to smooth with, multiple means multiple iterations
`in_place`	default = FALSE; adds smoothed index to the ‘hpiindex‘ object
`...`	Additional Arguments

Value

a ‘ts“ and ’smooth_index‘ object with smoothed index

Further Details

Leaving order blank default to a moving average with order 3.

Examples

```
# Load data
data(ex_sales)

# Create index with raw transaction data
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
```

```
                      max_date = '2015-11-30',
                      adj_type = 'clip',
                      date = 'sale_date',
                      price = 'sale_price',
                      trans_id = 'sale_id',
                      prop_id = 'pinx',
                      estimator = 'robust',
                      log_dep = TRUE,
                      trim_model = TRUE,
                      max_period = 48,
                      smooth = FALSE)

# Create Smooth index
sm_index <- smoothIndex(index_obj = rt_index,
                        order = 3,
                        in_place = FALSE)

# Create Smooth index (in place)
sm_index <- smoothIndex(index_obj = rt_index,
                        order = 3,
                        in_place = TRUE)
```

`smoothSeries` *Smooth all indexes in a series*

Description

Smooths all indexes within a progressive series of indexes

Usage

```
smoothSeries(series_obj, order = 3, ...)
```

Arguments

`series_obj`	Series to be smoothed
`order`	Number of nearby period to smooth with
`...`	Additional Arguments

Value

a ‘serieshpi‘ object with a smoothed index in each ‘hpiindex‘ object

Further Details

Leaving order blank default to a moving average with order 3.

Examples

```
# Load data
data(ex_sales)

# Create index
rt_index <- rtIndex(trans_df = ex_sales,
                    periodicity = 'monthly',
                    min_date = '2010-06-01',
                    max_date = '2015-11-30',
                    adj_type = 'clip',
                    date = 'sale_date',
                    price = 'sale_price',
                    trans_id = 'sale_id',
                    prop_id = 'pinx',
                    estimator = 'robust',
                    log_dep = TRUE,
                    trim_model = TRUE,
                    max_period = 48,
                    smooth = FALSE)

#  Create Series (Suppressing messages do to small sample size of this example)
suppressMessages(
   hpi_series <- createSeries(hpi_obj = rt_index,
                              train_period = 12))
# Smooth indexes
sm_series <- smoothSeries(series_obj = hpi_series,
                          order = 5)
```

Index