Package ‘PINstimation’

May 25, 2022

Type Package
Version 0.1.0
Date 2022-05-24
Title Estimation of the Probability of Informed Trading
Maintainer Montasser Ghachem <montasser.ghachem@pinstimation.com>
Description A comprehensive bundle of utilities for the estimation of probability of informed trading models: original PIN in Easley and O'Hara (1992) and Easley et al. (1996); Multi-layer PIN (MPIN) in Ersan (2016); Adjusted PIN (AdjPIN) in Duarte and Young (2009); and volume-synchronized PIN (VPIN) in Easley et al. (2011, 2012). Implementations of various estimation methods suggested in the literature are included. Additional compelling features comprise posterior probabilities, an implementation of an expectation-maximization (EM) algorithm, and PIN decomposition into layers, and into bad/good components. Versatile data simulation tools, and trade classification algorithms are among the supplementary utilities. The package provides fast, compact, and precise utilities to tackle the sophisticated, error-prone, and time-consuming estimation procedure of informed trading, and this solely using the raw trade-level data.


BugReports https://github.com/monty-se/PINstimation/issues

License GPL (>= 3)
Encoding UTF-8
LazyData true
LazyDataCompression xz
RoxygenNote 7.1.2
Roxygen list(markdown = TRUE)
VignetteBuilder knitr
Imports Rdpack, knitr, methods, skellam, nloptr, furrr, future, dplyr, rmarkdown
RdMacros Rdpack
Depends R (>= 2.10.0)
Language en-US
R topics documented:

PINstimation-package .................................................. 2
adjpin ................................................................. 7
aggregate_trades ....................................................... 10
dailytrades ............................................................. 12
data.series-class ...................................................... 12
dataset-class .......................................................... 13
detectlayers ............................................................. 15
estimate.adjpin-class ............................................... 16
estimate.mpin-class ................................................... 18
estimate.mpin.ecm-class .............................................. 19
estimate.pin-class ..................................................... 21
estimate.vpin-class ................................................... 22
factorizations ......................................................... 23
generatedata_adjpin ................................................ 26
generatedata_mpin ................................................... 29
get_posteriors ........................................................ 33
hfdata ................................................................. 35
initials_adjpin ........................................................ 36
initials_adjpin_cl .................................................... 37
initials_adjpin_rnd ................................................... 39
initials_mpin .......................................................... 41
initials_pin_ea ........................................................ 43
initials_pin_gwj ....................................................... 45
initials_pin_yz ........................................................ 46
mpin_ecm .............................................................. 48
mpin_ml ................................................................. 52
pin ...................................................................... 54
pin_ea ................................................................. 56
pin_gwj ................................................................. 58
pin_yz ................................................................. 60
set_display_digits .................................................... 62
vpin ................................................................. 63

Index 66

PINstimation-package  An R package for estimating the probability of informed trading

Description

The package provides utilities for the estimation of probability of informed trading measures: original PIN (PIN) as introduced by Easley and Ohara (1992) and Easley et al. (1996), multilayer PIN (MPIN) as introduced by Ersan (2016), adjusted PIN (AdjPIN) model as introduced in Duarte and Young (2009), and volume-synchronized PIN (VPIN) as introduced by Easley et al. (2011) and Easley et al. (2012). Estimations of PIN, MPIN, and AdjPIN are subject to floating-point exception error, and are sensitive to the choice of initial values. Therefore, researchers developed factorizations of the model likelihood functions as well as algorithms for determining initial parameter sets for the maximum likelihood estimation - (MLE henceforth).
As for the factorizations, the package includes three different factorizations of the PIN likelihood function: `fact_pin_eho()` as in Easley et al. (2010), `fact_pin_lk()` as in Lin and Ke (2011), and `fact_pin_e()` as in Ersan (2016); one factorization for MPIN likelihood function: `fact_mpin()` as in Ersan (2016); and one factorization for AdjPIN likelihood function: `fact_adjpin()` as in Ersan and Ghachem (2022b).

The package implements three algorithms to generate initial parameter sets for the MLE of the PIN model in: `initials_pin_yz()` for the algorithm of Yan and Zhang (2012), `initials_pin_gwj()` for the algorithm of Gan et al. (2015), and `initials_pin_ea()` for the algorithm of Ersan and Alici (2016). As for the initial parameter sets for the MLE of the MPIN model, the function `initials_mpin()` implements a multilayer extension of the algorithm of Ersan and Alici (2016). Finally, three functions implement three algorithms of initial parameter sets for the MLE of the AdjPIN model, namely `initials_adjpin()` for the algorithm in Ersan and Ghachem (2022b), `initials_adjpin_cl()` for the algorithm of Cheng and Lai (2021); and `initials_adjpin_rnd()` for randomly generated initial parameter sets. The choice of the initial parameter sets can be done directly, either using specific functions implementing MLE for the PIN model, such as, `pin_yz()`, `pin_gwj()`, `pin_ea()`; or through the argument `initialsets` in generic functions implementing MLE for the MPIN and AdjPIN models, namely `mpin_ml()`, and `adjpin()`. Besides, PIN, MPIN and AdjPIN models can be estimated using custom initial parameter set(s) provided by the user and fed through the argument `initialsets` for the functions `pin()`, `mpin_ml()` and `adjpin()`. Through the function `get_posteriors()`, the package also allows users to assign, for each day in the sample, the posterior probability that the day is a no-information day, good-information day and bad-information day.

As an alternative to the standard maximum likelihood estimation, estimation via expectation conditional maximization algorithm (ECM) is suggested in Ghachem and Ersan (2022a), and is implemented through the function `mpin_ecm()` for the MPIN model, and the function `adjpin()` for the AdjPIN model.

Dataset(s) of daily aggregated numbers of buys and sells with user determined number of information layers can be simulated with the function `generatedata_mpin()` for the MPIN (PIN) model; and `generatedata_adjpin()` for the AdjPIN model. The output of these functions contains the theoretical parameters used in the data generation, empirical parameters computed from the generated data, alongside the generated data itself. Data simulation functions allow for broad customization to produce data that fit the user’s preferences. Therefore, simulated data series can be utilized in comparative analyses for the applied methods in different scenarios. Alternatively, the user can use two example datasets preloaded in the package: `dailytrades` as a representative of a quarterly trade data with daily buys and sells; and `hfdata` as a simulated high-frequency dataset comprising 100,000 trades.

Finally, the package provides two functions to deal with high-frequency data. First, the function `vpin()` estimates and provides detailed output on the order flow toxicity metric, volume-synchronized probability of informed trading, as developed in Easley et al. (2011) and Easley et al. (2012). Second, the function `aggregate_trades()` aggregates the high-frequency trade-data into daily data using several trade classification algorithms, namely the tick algorithm, the quote algorithm, LR algorithm (Lee and Ready 1991) and the EMO algorithm (Ellis et al. 2000).

The package provides fast, compact, and precise utilities to tackle the sophisticated, error-prone, and time-consuming estimation procedure of informed trading, and this solely using the raw trade-level data. Ghachem and Ersan (2022b) provides comprehensive overview of the package: it first details the underlying theoretical background, provides a thorough description of the functions, before using them to tackle relevant research questions.
Functions

- `adjpin` estimates the adjusted probability of informed trading (AdjPIN) of the model of Duarte and Young (2009).
- `aggregate_trades` aggregates the trading data per day using different trade classification algorithms.
- `detectlayers_e` detects the number of information layers present in the trade-data using the algorithm in Ersan (2016).
- `detectlayers_eg` detects the number of information layers present in the trade-data using the algorithm in Ersan and Ghachem (2022a).
- `detectlayers_ecm` detects the number of information layers present in the trade-data using the expectation-conditional maximization algorithm in Ghachem and Ersan (2022a).
- `fact_adjpin` returns the AdjPIN factorization of the likelihood function by Ersan and Ghachem (2022b) evaluated at the provided data and parameter sets.
- `fact_pin_e` returns the PIN factorization of the likelihood function by Ersan (2016) evaluated at the provided data and parameter sets.
- `fact_pin_eho` returns the PIN factorization of the likelihood function by Easley et al. (2010) evaluated at the provided data and parameter sets.
- `fact_pin_lk` returns the PIN factorization of the likelihood function by Lin and Ke (2011) evaluated at the provided data and parameter sets.
- `fact_mpin` returns the MPIN factorization of the likelihood function by Ersan (2016) evaluated at the provided data and parameter sets.
- `generatedata_adjpin` generates a dataset object or a list of dataset objects generated according to the assumptions of the AdjPIN model.
- `generatedata_mpin` generates a dataset object or a list of dataset objects generated according to the assumptions of the MPIN model.
- `get_posteriors` computes, for each day in the sample, the posterior probabilities that it is a no-information day, good-information day and bad-information day respectively.
- `initials_adjpin` generates the initial parameter sets for the ML/ECM estimation of the adjusted probability of informed trading using the algorithm of Ersan and Ghachem (2022b).
- `initials_adjpin_cl` generates the initial parameter sets for the ML/ECM estimation of the adjusted probability of informed trading using an extension of the algorithm of Cheng and Lai (2021).
- `initials_adjpin_rnd` generates random parameter sets for the estimation of the AdjPIN model.
- `initials_pin_ea` generates the initial parameter sets for the maximum likelihood estimation of the probability of informed trading (PIN) using the algorithm of Ersan and Alici (2016).
- `initials_pin_gwj` generates the initial parameter set for the maximum likelihood estimation of the probability of informed trading (PIN) using the algorithm of Gan et al. (2015).
- `initials_pin_yz` generates the initial parameter sets for the maximum likelihood estimation of the probability of informed trading (PIN) using the algorithm of Yan and Zhang (2012).
- `mpin_ecm` estimates the multilayer probability of informed trading (MPIN) using the expectation-conditional maximization algorithm (ECM) as in Ghachem and Ersan (2022a).
- `mpin_ml` estimates the multilayer probability of informed trading (MPIN) using layer detection algorithms in Ersan (2016), and Ersan and Ghachem (2022a); and standard maximum likelihood estimation.
PINstimation-package

- **pin** estimates the probability of informed trading (PIN) using custom initial parameter set(s) provided by the user.
- **pin_ea** estimates the probability of informed trading (PIN) using the initial parameter sets from the algorithm of Ersan and Alici (2016).
- **pin_gwj** estimates the probability of informed trading (PIN) using the initial parameter set from the algorithm of Gan et al. (2015).
- **pin_yz** estimates the probability of informed trading (PIN) using the initial parameter sets from the grid-search algorithm of Yan and Zhang (2012).
- **vpin** estimates the volume-synchronized probability of informed trading (VPIN).

Datasets

- **dailytrades** A dataframe representative of quarterly (60 trading days) data of simulated daily buys and sells.
- **hfdata** A dataframe containing simulated high-frequency trade-data on 100 000 timestamps with the variables {timestamp, price, volume, bid, ask}.

Estimation results

- **estimate.adjpin-class** The class `estimate.adjpin` stores the estimation results of the function `adjpin()`.
- **estimate.mpin-class** The class `estimate.mpin` stores the estimation results of the MPIN model as estimated by the function `mpin_ml()`.
- **estimate.mpin.ecm-class** The class `estimate.mpin.ecm` stores the estimation results of the MPIN model as estimated by the function `mpin_ecm()`.
- **estimate.pin-class** The class `estimate.pin` stores the estimation results of the following PIN functions: `pin()`, `pin_yz()`, `pin_gwj()`, and `pin_ea()`.
- **estimate.vpin-class** The class `estimate.vpin` stores the estimation results of the VPIN model using the function `vpin()`.

Data simulation

- **dataset-class** The class `dataset` stores the result of simulation of the aggregate daily trading data.
- **data.series-class** The class `data.series` stores a list of `dataset`.

Author(s)

Montasser Ghachem montasser.ghachem@pinstimation.com
Department of Economics at Stockholm University, Stockholm, Sweden.

Oguz Ersan oguz.ersan@pinstimation.com
Department of International Trade and Finance at Kadir Has University, Istanbul, Turkey.
References


Ersan O, Ghachem M (2022b). “A methodological approach to the computational problems in the estimation of adjusted PIN model.” *Available at SSRN 4117954*.


---

**adjpin**

*Estimation of adjusted PIN model*

**Description**

Estimates the Adjusted Probability of Informed Trading (adjPIN) as well as the Probability of Symmetric Order-flow Shock (PSOS) from the AdjPIN model of Duarte and Young (2009).

**Usage**

```
adjpin(data, method = "ECM", initialsets = "GE", num_init = 20,
       restricted = list(), ..., verbose = TRUE)
```

**Arguments**

- **data** A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
- **method** A character string referring to the method used to estimate the model of Duarte and Young (2009). It takes one of two values: "ML" refers to the standard maximum likelihood estimation, and "ECM" refers to the expectation-conditional maximization algorithm. The default value is "ECM". Details of the ECM method, and comparative results can be found in Ghachem and Ersan (2022a), and in Ghachem and Ersan (2022b).
- **initialsets** It can either be a character string referring to prebuilt algorithms generating initial parameter sets or a dataframe containing custom initial parameter sets. If initialsets is a character string, it refers to the method of generation of the initial parameter sets, and takes one of three values: "GE", "CL", or "RANDOM". "GE" refers to initial parameter sets generated by the algorithm of Ersan and Ghachem (2022b), and implemented in initials_adjpin(), "CL" refers to initial parameter sets generated by the algorithm of Cheng and Lai (2021), and implemented in initials_adjpin_cl(), while "RANDOM" generates random initial parameter sets as implemented in initials_adjpin_rnd(). The default value is "GE". If initialsets is a dataframe, the function adjpin() will estimate the AdjPIN model using the provided initial parameter sets.
- **num_init** An integer specifying the maximum number of initial parameter sets to be used in the estimation. If initialsets="GE", the generation of initial parameter sets will stop when the number of initial parameter sets reaches num_init. It can stop earlier if the number of all possible generated initial parameter sets is lower than num_init. If initialsets="RANDOM", exactly num_init initial parameter sets are returned. If initialsets="CL": then num_init is ignored, and all 256 initial parameter sets are used. The default value is 20. [i] The argument num_init is ignored when the argument initialsets is a dataframe.
- **restricted** A binary list that allows estimating restricted AdjPIN models by specifying which model parameters are assumed to be equal. It contains one or multiple of the following four elements {theta, mu, eps, d}. For instance, If theta is set to TRUE, then the probability of liquidity shock in no-information days, and in
adjpin

information days is assumed to be the same ($\theta=\theta'$). If any of the remaining rate elements \{mu, eps, d\} is set to TRUE, (say mu=TRUE), then the rate is assumed to be the same on the buy side, and on the sell side ($\mu_b=\mu_s$). If more than one element is set to TRUE, then the restrictions are combined. For instance, if the argument restricted is set to list(theta=TRUE,eps=TRUE,d=TRUE), then the restricted AdjPIN model is estimated, where $\theta=\theta'$, $\varepsilon_b=\varepsilon_s$, and $\Delta_b=\Delta_s$. If the value of the argument restricted is the empty list (list()), then all parameters of the model are assumed to be independent, and the unrestricted model is estimated. The default value is the empty list list().

... Additional arguments passed on to the function adjpin(). The recognized arguments are hyperparams, and fact. The argument hyperparams consists of a list containing the hyperparameters of the ECM algorithm. When not empty, it contains one or more of the following elements: maxeval, and tolerance. It is used only when the method argument is set to "ECM". The argument fact is a binary value that determines which likelihood functional form is used: A factorization of the likelihood function by Ersan and Ghachem (2022b) when it is set to TRUE, otherwise, the original likelihood function of Duarte and Young (2009). The default value is TRUE. More about these arguments are in the Details section.

verbose A binary variable that determines whether detailed information about the steps of the estimation of the AdjPIN model is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.

Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

If initialsets is neither a dataframe, nor a character string from the set {"GE", "CL", "RANDOM"}, the estimation of the AdjPIN model is aborted. The default initial parameters ("GE") for the estimation method are generated using a modified hierarchical agglomerative clustering. For more information, see initials_adjpin().

The argument hyperparams contains the hyperparameters of the ECM algorithm. It is either empty or contains one or two of the following elements:

- maxeval: (integer) It stands for maximum number of iterations of the ECM algorithm for each initial parameter set. When missing, maxeval takes the default value of 100.
- tolerance (numeric) The ECM algorithm is stopped when the (relative) change of log-likelihood is smaller than tolerance. When missing, tolerance takes the default value of 0.001.

Value

Returns an object of class estimate.adjpin.

References


Examples

# We use 'generatedata_adjpin()' to generate a S4 object of type 'dataset' # with 60 observations.

sim_data <- generatedata_adjpin(days = 60)

# The actual dataset of 60 observations is stored in the slot 'data' of the # S4 object 'sim_data'. Each observation corresponds to a day and contains # the total number of buyer-initiated transactions ('B') and seller- # initiated transactions ('S') on that day.

xdata <- sim_data@data

# ------------------------------------------------------------------------ # # Compare the unrestricted AdjPIN model with various restricted models # # ------------------------------------------------------------------------ #

# Estimate the unrestricted AdjPIN model using the ECM algorithm (default), # and show the estimation output

estimate.adjpin.0 <- adjpin(xdata, verbose = FALSE)
show(estimate.adjpin.0)

# Estimate the restricted AdjPIN model where mub=mus

estimate.adjpin.1 <- adjpin(xdata, restricted = list(mu = TRUE),
                           verbose = FALSE)

# Estimate the restricted AdjPIN model where eps.b=eps.s

estimate.adjpin.2 <- adjpin(xdata, restricted = list(eps = TRUE),
                           verbose = FALSE)

# Estimate the restricted AdjPIN model where d.b=d.s

estimate.adjpin.3 <- adjpin(xdata, restricted = list(d = TRUE),
                           verbose = FALSE)

# Compare the different values of adjusted PIN

estimates <- list(estimate.adjpin.0, estimate.adjpin.1,
                   estimate.adjpin.2, estimate.adjpin.3)
adjpins <- sapply(estimates, function(x) x@adjpin)
psos <- sapply(estimates, function(x) x@psos)
summary <- cbind(adjpins, psos)
rownames(summary) <- c("unrestricted", "same.mu", "same.eps", "same.d")
show(round(summary, 5))
Details

The argument algorithm takes one of four values:

- "Tick" refers to the tick algorithm: Trade is classified as a buy (sell) if the price of the trade to be classified is above (below) the closest different price of a previous trade.
- "Quote" refers to the quote algorithm: it classifies a trade as a buy (sell) if the trade price of the trade to be classified is above (below) the mid-point of the bid and ask spread. Trades executed at the mid-spread are not classified.
- "LR" refers to LR algorithm as in Lee and Ready (1991). It classifies a trade as a buy (sell) if its price is above (below) the mid-spread (quote algorithm), and uses the tick algorithm if the trade price is at the mid-spread.
- "EMO" refers to EMO algorithm as in Ellis et al. (2000). It classifies trades at the bid (ask) as sells (buys) and uses the tick algorithm to classify trades within the then prevailing bid-ask spread.

LR recommend the use of mid-spread five-seconds earlier (’5-second’ rule) mitigating trade misclassifications for many of the 150 NYSE stocks they analyze. On the other hand, in more recent studies such as Piwowar and Wei (2006) and Aktas and Kryzanowski (2014), the use of 1-second lagged midquotes are shown to yield lower rates of misclassifications. The default value is set to 0 seconds (no time-lag). Considering the ultra-fast nature of today’s financial markets, time-lag is in the unit of milliseconds. Shorter than 1-second lags can also be implemented by entering values such as 100 or 500.

Value

Returns a dataframe of two (or three) variables. If reportdays is set to TRUE, then the returned dataframe has three variables \{day, b, s\}. If reportdays is set to FALSE, then the returned dataframe has two variables \{b, s\}, and, therefore, can be directly used for the estimation of the PIN and MPIN models.

References


Examples

# There is a preloaded dataset called 'hfdata' contained in the package.
# It is an artificially created high-frequency trading data. The dataset
# contains 100 000 trades and five variables 'timestamp', 'price',
# 'volume', 'bid', and 'ask'. For more information, type ?hfdata.

xdata <- hfdata
xdata$volume <- NULL
# Use the LR algorithm with a timelag of 0 milliseconds

daytrades <- aggregate_trades(xdata, algorithm = "LR", verbose = FALSE)

# Since the argument 'reportdays' is set to FALSE by default, then the
# output 'daytrades' can be used directly for the estimation of the PIN
# model, namely using pin_ea()

estimate <- pin_ea(daytrades, verbose = FALSE)

# Show the estimate

show(estimate)

dailytrades

Example of quarterly data

Description

An example dataset representative of quarterly data containing the aggregate numbers of buyer-initiated and seller-initiated trades for each trading day.

Usage

dailytrades

Format

A data frame with 60 observations and 2 variables:

- B: total number of buyer-initiated trades.
- S: total number of seller-initiated trades.

Source

Artificially created data set.

data.series-class

List of dataset objects

Description

The class data.series is the blueprint of S4 objects that store a list of dataset objects.

Usage

## S4 method for signature 'data.series'

show(object)
Arguments

object an object of class data.series

Slots

series (numeric) returns the number of dataset objects stored.

days (numeric) returns the length of the simulated data in days common to all dataset objects stored. The default value is 60.

model (character) returns a character string, either 'MPIN' or 'adjPIN'.

layers (numeric) returns the number of information layers in all dataset objects stored. It takes the value 1 for the adjusted PIN model, i.e. when model takes the value 'adjPIN'.

datasets (list) returns the list of the dataset objects stored.

restrictions (list) returns a binary list that contains the set of parameter restrictions on the original AdjPIN model in the estimated AdjPIN model. The restrictions are imposed equality constraints on model parameters. If the value of the parameter restricted is the empty list (list()), then the model has no restrictions, and the estimated model is the unrestricted, i.e., the original AdjPIN model. If not empty, the list contains one or multiple of the following four elements {theta, mu, eps, d}. For instance, If theta is set to TRUE, then the estimated model has assumed the equality of the probability of liquidity shocks in no-information, and information days, i.e., \( \theta = \theta' \). If any of the remaining rate elements {mu, eps, d} is equal to TRUE, (say mu=TRUE), then the estimated model imposed equality of the concerned parameter on the buy side, and on the sell side (\( \mu_b = \mu_s \)). If more than one element is equal to TRUE, then the restrictions are combined. For instance, if the slot restrictions contains list(theta=TRUE, eps=TRUE, d=TRUE), then the estimated AdjPIN model has three restrictions \( \theta = \theta' \), \( \varepsilon_b = \varepsilon_s \), and \( \Delta_b = \Delta_s \), i.e., it has been estimated with just 7 parameters, in comparison to 10 in the original unrestricted model. [i] This slot only concerns datasets generated by the function generatedata_adjpin().

warnings (numeric) returns numbers referring to the warning errors caused by a conflict between the different arguments used to call the function generatedata_mpin().

runningtime (numeric) returns the running time of the data simulation in seconds.

---

dataset-class Simulated data object

Description

The class dataset is a blueprint of S4 objects that store the result of simulation of the aggregate daily trading data.

Usage

### S4 method for signature 'dataset'

show(object)

Arguments

object an object of class dataset
Details

Theoreticals are the parameters used to generate the daily buys and sells. Empiricals are computed from the generated daily buys and sells. If we generate data for a 60 days using \( \alpha=0.1 \), the most likely outcome is to obtain 6 days (0.1 \times 60) as information event days. In this case, the theoretical value of \( \alpha=0.1 \) is equal to the empirically estimated value of \( \alpha=6/60=0.1 \). The number of generated information days can, however, be different from 6; say 5. In this case, empirical (actual) \( \alpha \) parameter derived from the generated numbers would be 5/60=0.0833, which differs from the theoretical \( \alpha=0.1 \). The weak law of large numbers ensures the empirical parameters (empiricals) converge towards the theoretical parameters (theoreticals) when the number of days becomes very large. To detect the estimation biases from the models/methods, comparing the estimates with empiricals rather than theoreticals would yield more realistic results.

Slots

- **model** (character) returns the model being simulated, either "MPIN", or "adjPIN".
- **days** (numeric) returns the length of the generated data in days.
- **layers** (numeric) returns the number of information layers in the simulated data. It takes the value 1 for the adjusted PIN model, i.e. when model takes the value ‘adjPIN’.
- **theoreticals** (list) returns the list of the theoretical parameters used to generate the data.
- **empiricals** (list) returns the list of the empirical parameters computed from the generated data.
- **aggregates** (numeric) returns an aggregation of information layers’ empirical parameters alongside with \( \varepsilon_b \) and \( \varepsilon_s \). The aggregated parameters are calculated as follows: \( \alpha_{agg} = \sum \alpha_j \), \( \delta_{agg} = \delta_1 \times \delta_j \), and \( \mu_{agg} = \mu_1 \times \mu_j \).
- **emp.pin** (numeric) returns the PIN/MPIN/AdjPIN value derived from the empirically estimated parameters of the generated data.
- **data** (dataframe) returns a dataframe containing the generated data.
- **likelihood** (numeric) returns the value of the (log-)likelihood function evaluated at the empirical parameters.
- **warnings** (character) stores warning messages for events that occurred during the data generation, such as conflict between two arguments.
- **restrictions** (list) returns a binary list that contains the set of parameter restrictions on the original AdjPIN model in the estimated AdjPIN model. The restrictions are imposed equality constraints on model parameters. If the value of the parameter restricted is the empty list \( \text{list()} \), then the model has no restrictions, and the estimated model is the unrestricted, i.e., the original AdjPIN model. If not empty, the list contains one or multiple of the following four elements \{theta, mu, eps, d\}. For instance, If theta is set to TRUE, then the estimated model has assumed the equality of the probability of liquidity shocks in no-information, and information days, i.e., \( \theta=\theta' \). If any of the remaining rate elements \{mu, eps, d\} is equal to TRUE, (say mu=TRUE), then the estimated model imposed equality of the concerned parameter on the buy side, and on the sell side (\( \mu_b'=\mu_s \)). If more than one element is equal to TRUE, then the restrictions are combined. For instance, if the slot restrictions contains \text{list(theta=TRUE, eps=TRUE, d=TRUE)}, then the estimated AdjPIN model has three restrictions \( \theta=\theta', \varepsilon_b=\varepsilon_s, \text{and} \Delta_b=\Delta_s \), i.e., it has been estimated with just 7 parameters, in comparison to 10 in the original unrestricted model. [i] This slot only concerns datasets generated by the function generatedata_adjpin().
Layer detection in trade-data

Description

Detects the number of information layers present in trade-data using the algorithms in Ersan (2016), Ersan and Ghachem (2022a), and Ghachem and Ersan (2022a).

Usage

detectlayers_e(data, confidence = 0.995, correction = TRUE)
detectlayers_eg(data, confidence = 0.995)
detectlayers_ecm(data, hyperparams = list())

Arguments

data A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
confidence A number from (0.5,1), corresponding to the range of the confidence interval used to determine whether a given cluster is compact, and therefore can be considered an information layer. If all values of absolute order imbalances (AOI) within a given cluster are within the confidence interval of a Skellam distribution with level equal to 'confidence', and centered on the mean of AOI, then the cluster is considered compact, and, therefore, an information layer. If some observations are outside the confidence interval, then the data is clustered further. The default value is 0.995. [i] This is an argument of the functions detectlayers_e(), and detectlayers_eg().
correction A binary variable that determines whether the data will be adjusted prior to implementing the algorithm of Ersan (2016). The default value is TRUE.
hyperparams A list containing the hyperparameters of the ECM algorithm. When not empty, it contains one or more of the following elements: maxeval, tolerance, maxinit, and maxLayers. More about these elements are found in the Details section. [i] This is an argument of the function detectlayers_ecm().

Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The argument hyperparams contains the hyperparameters of the ECM algorithm. It is either empty or contains one or more of the following elements:

• maxeval (integer) It stands for maximum number of iterations of the ECM for each initial parameter set. When missing, maxeval takes the default value of 100.
• tolerance (numeric) The ECM algorithm is stopped when the (relative) change of log-likelihood is smaller than tolerance. When missing, tolerance takes the default value of 0.001.
- `maxinit` (integer): It is the maximum number of initial parameter sets used for the ECM estimation per layer. When missing, `maxinit` takes the default value of 20.
- `maxlayers` (integer): It is the upper limit of number of layers used in the ECM algorithm. To find the optimal number of layers, the ECM algorithm will estimate a model for each value of the number of layers between 1 and `maxlayers`, and then picks the model that has the lowest Bayes information criterion (BIC). When missing, `maxlayers` takes the default value of 8.

**Value**

Returns an integer corresponding to the number of layers detected in the data.

**References**


**Examples**

```r
# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades

xdata <- dailytrades

# Detect the number of layers present in the dataset 'dailytrades' using the
# different algorithms and display the results

e.layers <- detectlayers_e(xdata)
eg.layers <- detectlayers_eg(xdata)
em.layers <- detectlayers_ecm(xdata)

show(c(e = e.layers, eg = eg.layers, em = em.layers))
```

---

**estimate.adjpin-class**

*AdjPIN estimation results*

**Description**

The class `estimate.adjpin` is a blueprint of the S4 objects that store the results of the estimation of the AdjPIN model using `adjpin()`.

**Usage**

```r
## S4 method for signature 'estimate.adjpin'
show(object)
```
estimate.adjpin-class

Arguments

object (estimate.adjpin-class)

Slots

success (logical) takes the value TRUE when the estimation has succeeded, FALSE otherwise.

errorMessage (character) contains an error message if the estimation of the AdjPIN model has failed, and is empty otherwise.

convergent.sets (numeric) returns the number of initial parameter sets, for which the likelihood maximization converged.

method (character) contains a reference to the estimation method: "ECM" for expectation-conditional maximization algorithm and "ML" for standard maximum likelihood estimation.

factorization (character) contains a reference to the factorization of the likelihood function used: "GE" for the factorization in Ersan and Ghachem (2022b), and "NONE" for the original likelihood function in Duarte and Young (2009).

restrictions (list) returns a binary list that contains the set of parameter restrictions on the original AdjPIN model in the estimated AdjPIN model. The restrictions are imposed equality constraints on model parameters. If the value of the parameter restricted is the empty list (list()), then the model has no restrictions, and the estimated model is the unrestricted, i.e., the original AdjPIN model. If not empty, the list contains one or multiple of the following four elements {theta, mu, eps, d}. For instance, If theta is set to TRUE, then the estimated model has assumed the equality of the probability of liquidity shocks in no-information, and information days, i.e., \( \theta = \theta' \). If any of the remaining rate elements {mu, eps, d} is equal to TRUE, (say \( \mu = \text{TRUE} \)), then the estimated model imposed equality of the concerned parameter on the buy side, and on the sell side (\( \mu_b = \mu_s \)). If more than one element is equal to TRUE, then the restrictions are combined. For instance, if the slot restrictions contains list(theta=TRUE, eps=TRUE, d=TRUE), then the estimated AdjPIN model has three restrictions \( \theta = \theta' , \epsilon_b = \epsilon_s \), and \( \Delta_b = \Delta_s \), i.e., it has been estimated with just 7 parameters, in comparison to 10 in the original unrestricted model.

algorithm (character) returns the implemented initial parameter set determination algorithm. "GE" is for Ersan and Ghachem (2022b), "CL" is for Cheng and Lai (2021), "RANDOM" for random initial parameter sets, and "CUSTOM" for custom initial parameter sets.

parameters (numeric) returns the vector of the optimal maximum-likelihood estimates (\( \alpha, \delta, \theta, \theta', \epsilon_b, \epsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s \)).

likelihood (numeric) returns the value (of the factorization) of the likelihood function, as in Ersan and Ghachem (2022b), evaluated at the set of optimal parameters.

adjpin (numeric) returns the value of the adjusted probability of informed trading (Duarte and Young 2009).

psos (numeric) returns the probability of symmetric order flow shock (Duarte and Young 2009).

dataset (dataframe) returns the dataset of buys and sells used in the estimation of the AdjPIN model.

initialsets (dataframe) returns the initial parameter sets used in the estimation of AdjPIN model.

details (dataframe) returns a dataframe containing the estimated parameters for each initial parameter set.

hyperparams (list) returns the hyperparameters of the ECM algorithm, which are maxeval, and tolerance.

runningtime (numeric) returns the running time of the AdjPIN estimation in seconds.
The class `estimate.mpin` is the blueprint of S4 objects that store the results of the estimation of the MPIN model, using the function `mpin_ml()`.

## S4 method for signature 'estimate.mpin'

### show(object)

#### Arguments

- **object**: an object of class `estimate.mpin`

#### Slots

- **success** (logical): returns the value `TRUE` when the estimation has succeeded, `FALSE` otherwise.
- **errorMessage** (character): returns an error message if the estimation of the MPIN model has failed, and is empty otherwise.
- **convergent.sets** (numeric): returns the number of initial parameter sets at which the likelihood maximization converged.
- **method** (character): returns the method of estimation used, and is equal to 'Maximum Likelihood Estimation'.
- **layers** (numeric): returns the number of layers detected in the trading data, or provided by the user.
- **detection** (logical): returns a reference to the layer-detection algorithm used ("E", "EG", "ECM"), if any algorithm is used. If the number of layers is provided by the user, detection takes the value "USER".
- **parameters** (list): returns the list of the maximum likelihood estimates \((\alpha, \delta, \mu, \varepsilon, \varepsilon_s)\), where \(\alpha\), \(\delta\), and \(\mu\) are numeric vectors of length \(\text{layers}\).
- **aggregates** (numeric): returns an aggregation of information layers’ estimated parameters alongside with \(\varepsilon_b\) and \(\varepsilon_s\). The aggregated parameters are calculated as follows: \(\alpha_{agg} = \sum \alpha_j\), \(\delta_{agg} = \sum \alpha_j \times \delta_j\), and \(\mu_{agg} = \sum \alpha_j \times \mu_j\).
- **likelihood** (numeric): returns the value of the (log-)likelihood function evaluated at the optimal set of parameters.
- **mpinJ** (numeric): returns the values of the multilayer probability of informed trading per layer, calculated using the layer-specific estimated parameters.
- **mpin** (numeric): returns the global value of the multilayer probability of informed trading. It is the sum of the multilayer probabilities of informed trading per layer stored in the slot `mpinJ`.
- **mpin.goodbad** (list): returns a list containing a decomposition of \(\text{MPIN}\) into good-news, and bad-news \(\text{MPIN}\) components. The decomposition has been suggested for PIN measure in Brennan et al. (2016). The list has four elements: `mpinG`, and `mpinB` are the global good-news, and bad-news components of \(\text{MPIN}\), while `mpinGj` and `mpinBj` are two vectors containing the good-news (bad-news) components of \(\text{MPIN}\) computed per layer.
dataset (dataframe) returns the dataset of buys and sells used in the maximum likelihood estimation of the MPIN model.

initialsets (dataframe) returns the initial parameter sets used in the maximum likelihood estimation of the MPIN model.

details (dataframe) returns a dataframe containing the estimated parameters of the MLE method for each initial parameter set.

runningtime (numeric) returns the running time of the estimation of the MPIN model in seconds.

---

**estimate.mpin.ecm-class**

*MPIN estimation results (ECM)*

**Description**

The class `estimate.mpin.ecm` is the blueprint of S4 objects that store the results of the estimation of the MPIN model using the Expectation-Conditional Maximization method, as implemented in the function `mpin_ecm()`.

**Usage**

```r
## S4 method for signature 'estimate.mpin.ecm'
show(object)

selectModel(object, criterion)

## S4 method for signature 'estimate.mpin.ecm'
selectModel(object, criterion)

getSummary(object)

## S4 method for signature 'estimate.mpin.ecm'
getSummary(object)
```

**Arguments**

- `object` 
  - an object of class `estimate.mpin.ecm`.
- `criterion` 
  - a character string specifying the model selection criterion. `criterion` should take one of these values {"BIC", "AIC", "AWE"}. They stand for Bayesian Information Criterion, Akaike Information Criterion, and Approximate Weight of Evidence, respectively.

**Functions**

- `selectModel,estimate.mpin.ecm-method`: returns the optimal model among the estimated models, i.e., the model having the lowest information criterion, provided by the user.

- `getSummary,estimate.mpin.ecm-method`: returns a summary of the estimation of the MPIN model using the ECM algorithm for different values of the argument layers. For each estimation, the number of layers, the MPIN value, the log-likelihood value, as well as the values of the different information criteria, namely AIC, BIC and AWE are displayed.
Slots

- **success** (logical) returns the value TRUE when the estimation has succeeded, FALSE otherwise.
- **errorMessage** (character) returns an error message if the MPIN estimation has failed, and is empty otherwise.
- **convergent.sets** (numeric) returns the number of initial parameter sets at which the likelihood maximization converged.
- **method** (character) returns the method of estimation, and is equal to 'Expectation-Conditional Maximization Algorithm'.
- **layers** (numeric) returns the number of layers estimated by the Expectation-Conditional Maximization algorithm, or provided by the user.
- **optimal** (logical) returns whether the number of layers used for the estimation is provided by the user (optimal=FALSE), or determined by the ECM algorithm (optimal=TRUE).
- **parameters** (list) returns the list of the maximum likelihood estimates \((\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)\), where \(\alpha, \delta,\) and \(\mu\) are numeric vectors of length layers.
- **aggregates** (numeric) returns an aggregation of information layers' parameters alongside with \(\varepsilon_b\) and \(\varepsilon_s\). The aggregated parameters are calculated as follows: \(\alpha_{agg} = \sum \alpha_j \delta_{agg} = \sum \alpha_j \times \delta_j\), and \(\mu_{agg} = \sum \alpha_j \times \mu_j\).
- **likelihood** (numeric) returns the value of the (log-)likelihood function evaluated at the optimal set of parameters.
- **mpinJ** (numeric) returns the values of the multilayer probability of informed trading per layer, calculated using the layer-specific estimated parameters.
- **mpin** (numeric) returns the global value of the multilayer probability of informed trading. It is the sum of the multilayer probabilities of informed trading per layer stored in the slot mpinJ.
- **mpin.goodbad** (list) returns a list containing a decomposition of MPIN into good-news, and bad-news MPIN components. The decomposition has been suggested for PIN measure in Brennan et al. (2016). The list has four elements: \(\text{mpinG}\) and \(\text{mpinB}\) are the global good-news, and bad-news components of MPIN, while \(\text{mpinGj}\) and \(\text{mpinBj}\) are two vectors containing the good-news (bad-news) components of MPIN computed per layer.
- **dataset** (dataframe) returns the dataset of buys and sells used in the ECM estimation of the MPIN model.
- **initialsets** (dataframe) returns the initial parameter sets used in the ECM estimation of the MPIN model.
- **details** (dataframe) returns a dataframe containing the estimated parameters of the ECM method for each initial parameter set.
- **models** (list) returns the list of estimate.mpin.ecm objects storing the results of estimation using the function mpin_ecm() for different values of the argument layers. It returns NULL when the argument layers of the function mpin_ecm() take a specific value.
- **AIC** (numeric) returns the value of the Akaike Information Criterion (AIC).
- **BIC** (numeric) returns the value of the Bayesian Information Criterion (BIC).
- **AWE** (numeric) returns the value of the Approximate Weight of Evidence.
- **criterion** (character) returns the model selection criterion used to find the optimal estimate for the MPIN model. It takes one of these values 'BIC', 'AIC', 'AWE'; which stand for Bayesian Information Criterion, Akaike Information Criterion, and Approximate Weight of Evidence, respectively.
- **hyperparams** (list) returns the hyperparameters of the ECM algorithm, which are minalpha, maxeval, tolerance, and maxlayers. Check the details section of mpin_ecm() to know more about these parameters.
- **runtime** (numeric) returns the running time of the estimation in seconds.
**estimate.pin-class**

**PIN estimation results**

**Description**

The class `estimate.pin` is a blueprint of S4 objects that store the results of the different PIN functions: `pin()`, `pin_yz()`, `pin_gwj()`, and `pin_ea()`.

**Usage**

```r
## S4 method for signature 'estimate.pin'
show(object)
```

**Arguments**

- `object` - an object of class `estimate.pin`

**Slots**

- `success` (logical) takes the value TRUE when the estimation has succeeded, FALSE otherwise.
- `errorMessage` (character) contains an error message if the PIN estimation has failed, and is empty otherwise.
- `convergent.sets` (numeric) returns the number of initial parameter sets at which the likelihood maximization converged.
- `algorithm` (character) returns the algorithm used to determine the set of initial parameter sets for the maximum likelihood estimation. It takes one of the following values:
  - "YZ": Yan and Zhang (2012)
  - "GWJ": Gan, Wei and Johnstone (2015)
  - "YZ*": Yan and Zhang (2012) as modified by Ersan and Alici (2016)
  - "EA": Ersan and Alici (2016)
  - "CUSTOM": Custom initial parameter sets
- `factorization` (character) returns the factorization of the PIN likelihood function as used in the maximum likelihood estimation. It takes one of the following values:
  - "NONE": No factorization
  - "EHO": Easley, Hvidkjaer and O’Hara (2010)
  - "LK": Lin and Ke (2011)
  - "E": Ersan (2016)
- `parameters` (list) returns the list of the maximum likelihood estimates \((\alpha, \delta, \mu, \epsilon_b, \epsilon_s)\)
- `likelihood` (numeric) returns the value of (the factorization of) the likelihood function evaluated at the optimal set of parameters.
- `pin` (numeric) returns the value of the probability of informed trading.
- `pin.goodbad` (list) returns a list containing a decomposition of PIN into good-news, and bad-news PIN components. The decomposition has been suggested in Brennan et al. (2016). The list has two elements: `pinG`, and `pinB` are the good-news, and bad-news components of PIN, respectively.
- `dataset` (dataframe) returns the dataset of buys and sells used in the maximum likelihood estimation of the PIN model.
initialsets (dataframe) returns the initial parameter sets used in the maximum likelihood estimation of the PIN model.
details (dataframe) returns a dataframe containing the estimated parameters by the MLE method for each initial parameter set.
runningtime (numeric) returns the running time of the estimation of the PIN model in seconds.

estimate.vpin-class  VPIN estimation results

Description

The class estimate.vpin is a blueprint for S4 objects that store the results of the VPIN estimation method using the function vpin().

The function show() displays a description of the estimate.vpin object: descriptive statistics of the VPIN variable, the set of relevant parameters, and the running time.

Usage

## S4 method for signature 'estimate.vpin'
show(object)

Arguments

object an object of class estimate.vpin

Slots

success (logical) returns the value TRUE when the estimation has succeeded, FALSE otherwise.
errorMessage (character) returns an error message if the VPIN estimation has failed, and is empty otherwise.
parameters (numeric) returns a numeric vector of estimation parameters (tbSize, buckets, samplength, VBS, #days), where tbSize is the size of timebars (in seconds); buckets is the number of buckets per average volume day; VBS is Volume Bucket Size (daily average volume/number of buckets buckets); samplength is the length of the window used to estimate VPIN; and #days is the number of days in the dataset.
bucketdata (dataframe) returns the dataframe containing detailed information about buckets. Following the output of Abad and Yague (2012), we report for each bucket its identifier (bucket), the aggregate buy volume (agg.bVol), the aggregate sell volume (agg.sVol), the absolute order imbalance (AOI=|agg.bVol-agg.sVol|), the start time (starttime), the end time (endtime), the duration in seconds (duration) as well as the VPIN vector.
vpin (numeric) returns the vector of the volume-synchronized probabilities of informed trading.
dailyvpin (dataframe) returns the daily VPIN values. Two variants are provided for any given day: dvpin corresponds to the unweighted average of vpin values, and dvpin.weighted corresponds to the average of vpin values weighted by bucket duration.
runningtime (numeric) returns the running time of the VPIN estimation in seconds.
Factorizations of the different PIN likelihood functions

Description

The PIN likelihood function is derived from the original PIN model as developed by Easley and Ohara (1992) and Easley et al. (1996). The maximization of the likelihood function as is leads to computational problems, in particular, to floating point errors. To remedy to this issue, several log-transformations or factorizations of the different PIN likelihood functions have been suggested. The main factorizations in the literature are:

- `fact_pin_eho()`: factorization of Easley et al. (2010)
- `fact_pin_lk()`: factorization of Lin and Ke (2011)
- `fact_pin_e()`: factorization of Ersan (2016)

The factorization of the likelihood function of the multilayer PIN model, as developed in Ersan (2016).

- `fact_mpin()`: factorization of Ersan (2016)

The factorization of the likelihood function of the adjusted PIN model (Duarte and Young 2009), is derived, and presented in Ersan and Ghachem (2022b).

- `fact_adjpin()`: factorization in Ersan and Ghachem (2022b)

Usage

```r
fact_pin_eho(data, parameters = NULL)
fact_pin_lk(data, parameters = NULL)
fact_pin_e(data, parameters = NULL)
fact_mpin(data, parameters = NULL)
fact_adjpin(data, parameters = NULL)
```

Arguments

- `data`: A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
- `parameters`: In the case of the PIN likelihood factorization, it is an ordered numeric vector \((\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)\). In the case of the MPIN likelihood factorization, it is an ordered numeric vector \((\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)\), where \(\alpha, \delta, \) and \(\mu\) are numeric vectors of size \(J\), where \(J\) is the number of information layers in the data. In the case of the AdjPIN likelihood factorization, it is an ordered numeric vector \((\alpha, \delta, \theta, \theta', \varepsilon_b, \varepsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s)\). The default value is NULL.
Details

The argument `data` should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

Our tests, in line with Lin and Ke (2011), and Ersan and Alici (2016), demonstrate very similar results for fact_pin lk(), and fact_pin_e(), both having substantially better estimates than fact_pin_eho().

Value

If the argument `parameters` is omitted, returns a function object that can be used with the optimization functions `optim()`, and `neldermead()`.

If the argument `parameters` is provided, returns a numeric value of the log-likelihood function evaluated at the dataset `data` and the parameters `parameters`, where `parameters` is a numeric vector following this order \((\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)\) for the factorizations of the PIN likelihood function, \((\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s)\) for the factorization of the MPIN likelihood function, and \((\alpha, \delta, \theta, \theta', \varepsilon_b, \varepsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s)\) for the factorization of the AdjPIN likelihood function.

References


Ersan O, Ghachem M (2022b). “A methodological approach to the computational problems in the estimation of adjusted PIN model.” *Available at SSRN 4117954*.


Examples

```r
# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades
```
xdata <- dailytrades

# Using fact_pin_eho(), fact_pin lk(), fact_pin_e() to find the likelihood value as factorized by Easley(2010), Lin & Ke (2011), and Ersan(2016).

# Choose a given parameter set to evaluate the likelihood function at a
givenpoint <- c(0.4, 0.1, 800, 300, 200)

# Use the output of fact_pin_e() with the optimization function optim() to find optimal estimates of the PIN model.
model <- suppressWarnings(optim(givenpoint, fact_pin_e(xdata)))

# Collect the model estimates from the variable model and display them.
varnames <- c("alpha", "delta", "mu", "eps.b", "eps.s")
estimates <- setNames(model$par, varnames)
show(estimates)

# Find the value of the log-likelihood function at givenpoint
lklValue <- fact_pin_lk(xdata, givenpoint)
show(lklValue)

# Using fact_mpin() to find the value of the MPIN likelihood function as factorized by Ersan (2016).

givenpoint <- c(0.4, 0.5, 0.1, 0.6, 600, 1000, 300, 200)

# Use the output of fact_mpin() with the optimization function optim() to find optimal estimates of the PIN model.
model <- suppressWarnings(optim(givenpoint, fact_mpin(xdata)))

# Collect the model estimates from the variable model and display them.
varnames <- c(paste("alpha", 1:2, sep = ""), paste("delta", 1:2, sep = ""), paste("mu", 1:2, sep = ""), "eb", "es")
estimates <- setNames(model$par, varnames)
show(estimates)

# Find the value of the MPIN likelihood function at givenpoint
lklValue <- fact_mpin(xdata, givenpoint)
# Using fact_adjpin() to find the value of the DY likelihood function as factorized by Ersan and Ghachem (2022b).

# Choose a given parameter set to evaluate the likelihood function at a the initial parameter set givenpoint = (alpha, delta, theta, theta', eps.b, eps.s, muB, muS, db, ds)

givenpoint <- c(0.4, 0.1, 0.3, 0.7, 500, 600, 800, 1000, 300, 200)

# Use the output of fact_adjpin() with the optimization function neldermead() to find optimal estimates of the AdjPIN model.

low <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
up <- c(1, 1, 1, Inf, Inf, Inf, Inf, Inf, Inf, Inf)
model <- nloptr::neldermead(
givenpoint, fact_adjpin(xdata), lower = low, upper = up)

# Collect the model estimates from the variable model and display them.

varnames <- c("alpha", "delta", "theta", "thetap", "eps.b", "eps.s", "muB", "muS", "db", "ds")
estimates <- setNames(model$par, varnames)
show(estimates)

# Find the value of the log-likelihood function at givenpoint

adjlklValue <- fact_adjpin(xdata, givenpoint)
show(adjlklValue)

generatedata_adjpin

Simulation of AdjPIN model data.

Description

Generates a dataset object or a data.series object (a list of dataset objects) storing simulation parameters as well as aggregate daily buys and sells simulated following the assumption of the AdjPIN model of Duarte and Young (2009).

Usage

generatedata_adjpin(series=1, days = 60, parameters = NULL, ranges = list(), restricted = list(), verbose = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>series</td>
<td>The number of datasets to generate.</td>
</tr>
<tr>
<td>days</td>
<td>The number of trading days, for which aggregated buys and sells are generated. The default value is 60.</td>
</tr>
<tr>
<td>parameters</td>
<td>A vector of model parameters of size 10 and it has the following form {\alpha, \delta, \theta, \theta', \epsilon_b, \epsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s }.</td>
</tr>
</tbody>
</table>
generatedata_adjpin

ranges A list of ranges for the different simulation parameters having named elements alpha (\( \alpha \)), delta (\( \delta \)), theta (\( \theta \)), thetap (\( \theta' \)), eps.b (\( \varepsilon_b \)), eps.s (\( \varepsilon_s \)), mu.b (\( \mu_b \)), mu.s (\( \mu_s \)), d.b (\( \Delta_b \)), d.s (\( \Delta_s \)). The value of each element is a vector of two numbers: the first one is the minimal value min_v and the second one is the maximal value max_v. If the element corresponding to a given parameter is missing, the default range for that parameter is used, otherwise, the simulation parameters are uniformly drawn from the interval (min_v, max_v). The default value is list().

restricted A binary list that allows estimating restricted AdjPIN models by specifying which model parameters are assumed to be equal. It contains one or multiple of the following four elements {theta, mu, eps, d}. For instance, If theta is set to TRUE, then the probability of liquidity shock in no-information days, and in information days is assumed to be the same (\( \theta=\theta' \)). If any of the remaining rate elements {mu, eps, d} is set to TRUE, then the rate is assumed to be the same on the buy side, and on the sell side (\( \mu_b=\mu_s \)). If more than one element is set to TRUE, then the restrictions are combined. For instance, if the argument restricted is set to list(theta=TRUE, eps=TRUE, d=TRUE), then the restricted AdjPIN model is estimated, where \( \theta=\theta' \), \( \varepsilon_b=\varepsilon_s \), and \( \Delta_b=\Delta_s \). If the value of the argument restricted is the empty list (list()), then all parameters of the model are assumed to be independent, and the unrestricted model is estimated. The default value is the empty list list().

verbose A binary variable that determines whether detailed information about the progress of the data generation is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.

Details

If the argument parameters is missing, then the parameters are generated using the ranges specified in the argument ranges. If the argument ranges is set to list(), default ranges are used. Using the default ranges, the simulation parameters are obtained using the following procedure:

- \( \alpha, \delta \): (alpha, delta) uniformly distributed on \((0, 1)\).
- \( \theta, \theta' \): (theta, thetap) uniformly distributed on \((0, 1)\).
- \( \varepsilon_b \): (eps.b) an integer uniformly drawn from the interval \((100, 10000)\) with step 50.
- \( \varepsilon_s \): (eps.s) an integer uniformly drawn from \(((4/5) \varepsilon_b, (6/5) \varepsilon_b)\) with step 50.
- \( \Delta_b \): (d.b) an integer uniformly drawn from \(((1/2) \varepsilon_b, 2 \varepsilon_b)\).
- \( \Delta_s \): (d.s) an integer uniformly drawn from \(((4/5) \Delta_b, (6/5) \Delta_b)\).
- \( \mu_b \): (mu.b) uniformly distributed on the interval \(((1/2) \max(\varepsilon_b, \varepsilon_s), 5 \max(\varepsilon_b, \varepsilon_s))\).
- \( \mu_s \): (mu.s) uniformly distributed on the interval \(((4/5) \mu_b, (6/5) \mu_b)\).

Based on the simulation parameters parameters, daily buys and sells are generated by the assumption that buys and sells follow Poisson distributions with mean parameters:

- \( \varepsilon_b, \varepsilon_s \) in a day with no information and no liquidity shock;
- \( \varepsilon_b + \Delta_b, \varepsilon_s + \Delta_s \) in a day with no information and with liquidity shock;
- \( \varepsilon_b + \mu_b, \varepsilon_s \) in a day with good information and no liquidity shock;
- \( \varepsilon_b + \mu_b + \Delta_b, \varepsilon_s + \Delta_s \) in a day with good information and liquidity shock;
- \( \varepsilon_b, \varepsilon_s + \mu_s \) in a day with bad information and no liquidity shock;
- \( \varepsilon_b + \Delta_s, \varepsilon_s + \mu_s + \Delta_s \) in a day with bad information and liquidity shock;
Value

Returns an object of class dataset if series=1, and an object of class data.series if series>1.

References


Examples

```r
# Generate data following the AdjPIN model using generatedata_adjpin() #
# With no arguments, the function generates one dataset object spanning # 60 days, and where the parameters are chosen as described in the section # 'Details'.

sdata <- generatedata_adjpin()

# Alternatively, simulation parameters can be provided. Recall the order of # parameters (alpha, delta, theta, theta', eps.b, eps.s, mub, mus, db, ds).

givenpoint <- c(0.4, 0.1, 0.5, 0.6, 800, 1000, 2300, 4000, 500, 500)
sdata <- generatedata_adjpin(parameters = givenpoint)

# Data can be generated following restricted AdjPIN models, for example, with # restrictions 'eps.b = eps.s', and 'mu.b = mu.s'.

sdata <- generatedata_adjpin(restricted = list(eps = TRUE, mu = TRUE))

# Data can be generated using provided ranges of simulation parameters as fed # to the function using the argument 'ranges', where thetap corresponds to # theta'.

sdata <- generatedata_adjpin(ranges = list(
  alpha = c(0.1, 0.15), delta = c(0.2, 0.2),
  theta = c(0.2, 0.6), thetap = c(0.2, 0.4)
))

# The value of a given simulation parameter can be set to a specific value by # setting the range of the desired parameter takes a unique value, instead of # a pair of values.

sdata <- generatedata_adjpin(ranges = list(
  alpha = 0.4, delta = c(0.2, 0.7),
  eps.b = c(100, 7000), mu.b = 8000
))

# Display the details of the generated simulation data

show(sdata)
```

```r
# Use generatedata_adjpin() to check the accuracy of adjpin()     #
```
model <- adjpin(sdata@data, verbose = FALSE)

summary <- cbind(
c(sdata@emp.pin['adjpin'], model$adjpin, abs(model$adjpin - sdata@emp.pin['adjpin'])),
c(sdata@emp.pin['psos'], model$psos, abs(model$psos - sdata@emp.pin['psos'])))
colnames(summary) <- c('adjpin', 'psos')
rownames(summary) <- c('Data', 'Model', 'Difference')

show(knitr::kable(summary, 'simple'))

generatedata_mpin

Simulation of MPIN model data

Description

Generates a dataset object or a data.series object (a list of dataset objects) storing simulation parameters as well as aggregate daily buys and sells simulated following the assumption of the MPIN model of (Ersan 2016).

Usage

generatedata_mpin(series = 1, days = 60, layers = NULL, parameters = NULL, ranges = list(), ...,
verbose = TRUE)

Arguments

series The number of datasets to generate.
days The number of trading days for which aggregated buys and sells are generated. Default value is 60.
layers The number of information layers to be included in the simulated data. Default value is NULL. If layers is omitted or set to NULL, the number of layers is uniformly selected from the set \{1, ..., maxlayers\}.
parameters A vector of model parameters of size 3J+2 where J is the number of information layers and it has the following form \{\alpha_1, ..., \alpha_J, \delta_1, ..., \delta_J, \mu_1, ..., \mu_J, \epsilon_b, \epsilon_s\}.
ranges A list of ranges for the different simulation parameters having named elements \alpha, \delta, \epsilon_b, \epsilon_s, and \mu. The value of each element is a vector of two numbers: the first one is the minimal value min_v and the second one is the maximal value max_v. If the element corresponding to a given parameter is missing, the default range for that parameter is used. If the argument ranges is an empty list and parameters is NULL, the default ranges for the parameters are used. The simulation parameters are uniformly drawn from the interval (min_v, max_v) for the specified parameters. The default value is list().
...

Additional arguments passed on to the function generatedata_mpin(). The recognized arguments are confidence, maxlayers, eps_ratio, mu_ratio.
• confidence (numeric) denotes the range of the confidence interval associated with each layer such that all observations within the layer lie in the theoretical confidence interval of the Skellam distribution centered on the mean order imbalance, at the level 'confidence'. The default value is 0.99.

• maxlayers (integer) denotes the upper limit of number of layers for the generated datasets. If the argument layers is missing, the layers of the simulated datasets will be uniformly drawn from \{1, \ldots, maxlayers\}. When missing, maxlayers takes the default value of 5.

• eps_ratio (numeric) specifies the admissible range for the value of the ratio $\varepsilon_s/\varepsilon_b$. It can be a two-value vector or just a single value. If eps_ratio is a vector of two values: the first one is the minimal value and the second one is the maximal value; and the function tries to generate $\varepsilon_s$ and $\varepsilon_b$ satisfying that their ratios $\varepsilon_s/\varepsilon_b$ lies within the interval eps_ratio. If eps_ratio is a single number, then the function tries to generate $\varepsilon_s$ and $\varepsilon_b$ satisfying $\varepsilon_s = \varepsilon_b \times \text{eps}_\text{ratio}$. If this range conflicts with other arguments such as ranges or confidence, a warning is displayed. The default value is c(0.75, 1.25).

• mu_ratio (numeric) it is the minimal value of the ratio between two consecutive values of the vector $\mu$. If $\mu_{\text{ratio}} = 1.25$ e.g., then $\mu_{j+1}$ should be larger than $1.25 \times \mu_j$ for all $j = 1, \ldots, J$. If mu_ratio conflicts with other arguments such as ranges or confidence, a warning is displayed. The default value is NULL.

verbose (logical) a binary variable that determines whether detailed information about the progress of the data generation is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.

Details

An information layer refers to a given type of information event existing in the data. The PIN model assumes a single type of information events characterized by three parameters for $\alpha$, $\delta$, and $\mu$. The MPIN model relaxes the assumption, by relinquishing the restriction on the number of information event types. When layers = 1, generated data fit the assumptions of the PIN model.

If the argument parameters is missing, then the simulation parameters are generated using the ranges specified in the argument ranges. If the argument ranges is list(), default ranges are used. Using the default ranges, the simulation parameters are obtained using the following procedure:

• $\alpha()$: a vector of length layers, where each $\alpha_j$ is uniformly distributed on $(0, 1)$ subject to the condition:
  \[ \sum_j \alpha_j \leq 1. \]

• $\delta()$: a vector of length layers, where each $\delta_j$ uniformly distributed on $(0, 1)$.

• $\mu()$: a vector of length layers, where each $\mu_j$ is uniformly distributed on the interval $(0.5 \max(\varepsilon_b, \varepsilon_s), 5 \max(\varepsilon_b, \varepsilon_s))$. The $\mu$'s are then sorted so the excess trading increases in the information layers, subject to the condition that the ratio of two consecutive $\mu$'s should be at least 1.25.

• $\varepsilon_b$: an integer drawn uniformly from the interval $(100, 10000)$ with step 50.

• $\varepsilon_s$: an integer uniformly drawn from $((3/4)\varepsilon_b, (5/4)\varepsilon_b)$ with step 50.

Based on the simulation parameters parameters, daily buys and sells are generated by the assumption that buys and sells follow Poisson distributions with mean parameters $(\varepsilon_b, \varepsilon_s)$ on days with no information; with mean parameters $(\varepsilon_b + \mu_j, \varepsilon_s)$ on days with good information of layer $j$ and $(\varepsilon_b, \varepsilon_s + \mu_j)$ on days with bad information of layer $j$. 

Considerations for the ranges of simulation parameters: While `generatedata_mpin()` function enables the user to simulate data series with any set of theoretical parameters, we strongly recommend the use of parameter sets satisfying below conditions which are in line with the nature of empirical data and the theoretical models used within this package. When parameter values are not assigned by the user, the function, by default, simulates data series that are in line with these criteria.

- **Consideration 1**: any $\mu$’s value separable from $\varepsilon_b$ and $\varepsilon_s$ values, as well as other $\mu$ values. Otherwise, the PIN and MPIN estimation would not yield expected results.
  
  [x] Sharp example.1: $\varepsilon_b = 1000; \mu = 1$. In this case, no information layer can be captured in a healthy way by the use of the models which relies on Poisson distributions.
  
  [x] Sharp example.2: $\varepsilon_s = 1000, \mu_1 = 1000$, and $\mu_2 = 1001$. Similarly, no distinction can be made on the two simulated layers of informed trading. In real life, this entails that there is only one type of information which would also be the estimate of the MPIN model. However, in the simulated data properties, there would be 2 layers which will lead the user to make a wrong evaluation of model performance.

- **Consideration 2**: $\varepsilon_b$ and $\varepsilon_s$ being relatively close to each other. When they are far from each other, that would indicate that there is substantial asymmetry between buyer and seller initiated trades, being a strong signal for informed trading. There is no theoretical evidence to indicate that the uninformed trading in buy and sell sides deviate much from each other in real life. Besides, numerous papers that work with PIN model provide close to each other uninformed intensities. when no parameter values are assigned by the user, the function generates data with the condition of sell side uninformed trading to be in the range of $(4/5):=80\%$ and $(6/5):=120\%$ of buy side uninformed rate.
  
  [x] Sharp example.3: $\varepsilon_b = 1000, \varepsilon_s = 10000$. In this case, the PIN and MPIN models would tend to consider some of the trading in sell side to be informed (which should be the actual case). Again, the estimation results would deviate much from the simulation parameters being a good news by itself but a misleading factor in model evaluation. See for example Cheng and Lai (2021) as a misinterpretation of comparative performances. The paper’s findings highly rely on the simulations with extremely different $\varepsilon_b$ and $\varepsilon_s$ values (813-8124 pair and 8126-812).

Value

Returns an object of class `dataset` if `series=1`, and an object of class `data.series` if `series>1`.

References


Examples

```r
# There are different scenarios of using the function generatedata_mpin()
# With no arguments, the function generates one dataset object spanning
# 60 days, containing a number of information layers uniformly selected
# from "[1, 2, 3, 4, 5]", and where the parameters are chosen as
# described in the details.

sdata <- generatedata_mpin()
```
# The number of layers can be deduced from the simulation parameters, if
# fed directly to the function `generatedata_mpin()` through the argument
# 'parameters'. In this case, the output is a dataset object with one
# information layer.

givenpoint <- c(0.4, 0.1, 800, 300, 200)
sdata <- generatedata_mpin(parameters = givenpoint)

# The number of layers can alternatively be set directly through the
# argument 'layers'.
sdata <- generatedata_mpin(layers = 2)

# The simulation parameters can be randomly drawn from their corresponding
# ranges fed through the argument 'ranges'.
sdata <- generatedata_mpin(ranges = list(alpha = c(0.1, 0.7),
                           delta = c(0.2, 0.7),
                           mu = c(3000, 5000)))

# The value of a given simulation parameter can be set to a specific value by
# setting the range of the desired parameter takes a unique value, instead of
# a pair of values.
sdata <- generatedata_mpin(ranges = list(alpha = 0.4, delta = c(0.2, 0.7),
                           eps.b = c(100, 7000),
                           mu = c(8000, 12000)))

# If both arguments 'parameters', and 'layers' are simultaneously provided,
# and the number of layers detected from the length of the argument
# 'parameters' is different from the argument 'layers', the former is used
# and a warning is displayed.
sim.params <- c(0.4, 0.2, 0.9, 0.1, 400, 700, 300, 200)
sdata <- generatedata_mpin(days = 120, layers = 3, parameters = sim.params)

# Display the details of the generated data
show(sdata)

# Use `generatedata_mpin()` to compare the accuracy of estimation methods
# The example below illustrates the use of the function 'generatedata_mpin()'
# to compare the accuracy of the functions 'mpin_ml()', and 'mpin_ecm()'.

# The example will depend on three variables:
# n: the number of datasets used
# l: the number of layers in each simulated datasets
# xc : the number of extra clusters used in initials_mpin

# For consideration of speed, we will set n = 2, l = 2, and xc = 2
# These numbers can change to fit the user's preferences
n <- 1 <- xc <- 2
# We start by generating n datasets simulated according to the
# assumptions of the MPIN model.

dataseries <- generatedata_mpin(series = n, layers = l, verbose = FALSE)

# Store the estimates in two different lists: 'mllist', and 'ecmlist'

mllist <- lapply(dataseries@datasets, function(x)
    mpin_ml(x@data, xtraclusters = xc, layers = l, verbose = FALSE))

ecmlist <- lapply(dataseries@datasets, function(x)
    mpin_ecm(x@data, xtraclusters = xc, layers = l, verbose = FALSE))

# For each estimate, we calculate the absolute difference between the
# estimated mpin, and empirical mpin computed using dataset parameters.
# The absolute differences are stored in 'mldpin' ('ecmdpin') for the
# ML (ECM) method,

mldpin <- sapply(1:n,
    function(x) abs(mllist[[x]]@mpin - dataseries@datasets[[x]]@emp.pin))

ecmdpin <- sapply(1:n,
    function(x) abs(ecmlist[[x]]@mpin - dataseries@datasets[[x]]@emp.pin))

# Similarly, we obtain vectors of running times for both estimation methods.
# They are stored in 'mltime' ('ecmtime') for the ML (ECM) method.

mltime <- sapply(mllist, function(x) x@runningtime)

ecmtime <- sapply(ecmlist, function(x) x@runningtime)

# Finally, we calculate the average absolute deviation from empirical PIN
# as well as the average running time for both methods. This allows us to
# compare them in terms of accuracy, and speed.

accuracy <- c(mean(mldpin), mean(ecmdpin))
timing <- c(mean(mltime), mean(ecmtime))

comparison <- as.data.frame(rbind(accuracy, timing))
colnames(comparison) <- c("ML", "ECM")
rownames(comparison) <- c("Accuracy", "Timing")

show(round(comparison, 6))

---

**get_posteriors**

Posterior probabilities for PIN and MPIN estimates

**Description**

Computes, for each day in the sample, the posterior probability that the day is a no-information day, good-information day and bad-information day, respectively (Easley and Ohara (1992), Easley et al. (1996), Ersan (2016)).

**Usage**

`get_posteriors(object)`
get_posteriors

**Arguments**

`object` (S4 object) an object of type `estimate.pin`, `estimate.mpin`, or `estimate.mpin.ecm`.

**Value**

If the argument `object` is of type `estimate.pin`, returns a dataframe of three variables `post.N`, `post.G` and `post.B` containing in each row the posterior probability that a given day is a no-information day (N), good-information day (G), or bad-information day (B) respectively.

If the argument `object` is of type `estimate.mpin` or `estimate.mpin.ecm` with J layers, returns a dataframe of $2J+1$ variables `Post.N` and `Post.G[j]` and `Post.B[j]` for each layer j containing in each row the posterior probability that a given day is a no-information day, good-information day in layer j or bad-information day in layer j, for each layer j respectively.

If the argument `object` is of any other type, an error is returned.

**References**


**Examples**

# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades

xdata <- dailytrades

# ------------------------------------------------------------------------ #
# Posterior probabilities for PIN estimates
# ------------------------------------------------------------------------ #

# Estimate PIN using the Ersan and Alici (2016) algorithm and the

estimate <- pin_ea(xdata, "LK", verbose = FALSE)

# Display the estimated PIN value

estimate@pin

# Store the posterior probabilities in a dataframe variable and display its
# first 6 rows.

modelposteriors <- get_posteriors(estimate)
show(round(head(modelposteriors), 3))

# ------------------------------------------------------------------------ #
# Posterior probabilities for MPIN estimates
#
# Estimate MPIN via the ECM algorithm, assuming that the dataset has 2
# information layers

estimate <- mpin_ecm(xdata, layers = 2, verbose = FALSE)

# Display the estimated Multilayer PIN value

show(estimate@mpin)

# Store the posterior probabilities in a dataframe variable and display its
# first six rows. The posterior probabilities are contained in a dataframe
# with 7 variables: one for no-information days, and two variables for each
# layer, one for good-information days and one for bad-information days.

modelposteriors <- get_posteriors(estimate)
show(round(head(modelposteriors), 3))

---

**hfdata**  
*High-frequency trade-data*

**Description**

A simulated dataset containing sample timestamp, price, volume, bid and ask for 100 000 high frequency transactions.

**Usage**

hfdata

**Format**

A data frame with 100 000 observations with 5 variables:

- timestamp: time of the trade.
- price: transaction price.
- volume: volume of the transactions, in asset units.
- ask: best ask price.

**Source**

Artificially created data set.
AdjPIN initial parameter sets of Ersan & Ghachem (2022b)

Description

Based on the algorithm in Ersan and Ghachem (2022b), generates sets of initial parameters to be used in the maximum likelihood estimation of AdjPIN model.

Usage

initials_adjpin(data, xtraclusters = 4, restricted = list(), verbose = TRUE)

Arguments

data A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
xtraclusters An integer used to divide trading days into #(4 + xtraclusters) clusters, thereby resulting in #comb(4 + xtraclusters -1,4 -1) initial parameter sets in line with Ersan and Alici (2016), and Ersan and Ghachem (2022b). The default value is 4 as chosen in Ersan (2016).

restricted A binary list that allows estimating restricted AdjPIN models by specifying which model parameters are assumed to be equal. It contains one or multiple of the following four elements {theta, mu, eps, d}. For instance, If theta is set to TRUE, then the probability of liquidity shock in no-information days, and in information days is assumed to be the same (\(\theta=\theta'\)). If any of the remaining rate elements {mu, eps, d} is set to TRUE (say mu=TRUE), then the rate is assumed to be the same on the buy side, and on the sell side (\(\mu_b=\mu_s\)). If more than one element is set to TRUE, then the restrictions are combined. For instance, if the argument restricted is set to list(theta=TRUE, eps=TRUE, d=TRUE), then the restricted AdjPIN model is estimated, where \(\theta=\theta'\), \(\varepsilon_b=\varepsilon_s\), and \(\Delta_b=\Delta_s\). If the value of the argument restricted is the empty list, then all parameters of the model are assumed to be independent, and the unrestricted model is estimated. The default value is the empty list list().

verbose A binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when verbose is set to FALSE. The default value is TRUE.

Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The function initials_adjpin() implements the algorithm suggested in Ersan and Ghachem (2022b), and uses a hierarchical agglomerative clustering (HAC) to find initial parameter sets for the maximum likelihood estimation.
Value

Returns a dataframe of numerical vectors of ten elements \( \{ \alpha, \delta, \theta, \theta', \varepsilon_b, \varepsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s \} \).

References


Examples

```r
# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades

data <- dailytrades

# Obtain a dataframe of initial parameter sets for the maximum likelihood
# estimation using the algorithm of Ersan and Ghachem (2022b).

init.sets <- initials_adjpin(xdata)

# Use the list to estimate adjpin using the adjpin() method
# Show the value of adjusted PIN

estimate <- adjpin(xdata, initialsets = init.sets, verbose = FALSE)
show(estimate@adjpin)
```

initials_adjpin_cl

AdjPIN initial parameter sets of Cheng and Lai (2021)

Description

Based on an extension of the algorithm in Cheng and Lai (2021), generates sets of initial parameters to be used in the maximum likelihood estimation of AdjPIN model.

Usage

```r
initials_adjpin_cl(data, restricted = list(), verbose = TRUE)
```

Arguments

data A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
**restricted**  A binary list that allows estimating restricted AdjPIN models by specifying which model parameters are assumed to be equal. It contains one or multiple of the following four elements \{\theta, \mu, \epsilon, d\}. For instance, if \theta is set to TRUE, then the probability of liquidity shock in no-information days, and in information days is assumed to be the same \(\theta = \theta'\). If any of the remaining rate elements \{\mu, \epsilon, d\} is set to TRUE, (say \mu=TRUE), then the rate is assumed to be the same on the buy side, and on the sell side \(\mu_b = \mu_s\). If more than one element is set to TRUE, then the restrictions are combined. For instance, if the argument restricted is set to list(\theta=TRUE, \epsilon=TRUE, d=TRUE), then the restricted AdjPIN model is estimated, where \(\theta = \theta'\), \(\epsilon_b = \epsilon_s\), and \(\Delta_b = \Delta_s\). If the value of the argument restricted is the empty list, then all parameters of the model are assumed to be independent, and the unrestricted model is estimated. The default value is the empty list list().

**verbose**  a binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when verbose is set to FALSE. The default value is TRUE.

### Details

The argument `data` should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation corresponds to a trading day. NA values will be ignored.

The function implements an extension of the algorithm of Cheng and Lai (2021). In their paper, the authors assume that the probability of liquidity shock is the same in no-information, and information days, i.e., \(\theta = \theta'\), and use a procedure similar to that of Yan and Zhang (2012) to generate 64 initial parameter sets. The function implements an extension of their algorithm, by relaxing the assumption of equality of liquidity shock probabilities, and generates thereby 256 initial parameter sets for the unrestricted AdjPIN model.

### Value

Returns a dataframe of numerical vectors of ten elements \{\alpha, \delta, \theta, \theta', \epsilon_b, \epsilon_s, \mu_b, \mu_s, \Delta_b, \Delta_s\}.

### References


### Examples

```r
# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades
xdata <- dailytrades

# The function adjpin(xdata, initialsets="CL") allows the user to directly
```
# estimate the AdjPIN model using the full set of initial parameter sets
generated using the algorithm Cheng and Lai (2021)
estimate.1 <- adjpin(xdata, initialsets="CL", verbose = FALSE)

# Obtaining the set of initial parameter sets using initials_adjpin_cl
# allows us to estimate the PIN model using a subset of these initial sets.
# Use initials_adjpin_cl() to generate 256 initial parameter sets using the
initials_cl <- initials_adjpin_cl(xdata, verbose = FALSE)

# Use 50 randomly chosen initial sets from the dataframe 'initials_cl' in
# order to estimate the AdjPIN model using the function adjpin() with custom
# initial parameter sets

numberofsets <- nrow(initials_cl)
selectedsets <- initials_cl[sample(numberofsets, 50),]
estimate.2 <- adjpin(xdata, initialsets = selectedsets, verbose = FALSE)

# Compare the parameters and the pin values of both specifications

c(estimate.1@parameters, adjpin = estimate.1@adjpin, psos = estimate.1@psos),
c(estimate.2@parameters, estimate.2@adjpin, estimate.2@psos))

rownames(comparison) <- c("all", "50")
show(comparison)

---

**initials_adjpin_rnd**  
AdjPIN random initial sets

**Description**

Generates random initial parameter sets to be used in the estimation of the AdjPIN model of Duarte and Young (2009).

**Usage**

```r
initials_adjpin_rnd(data, restricted = list(), num_init = 20, verbose = TRUE)
```

**Arguments**

- `data`: A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
restricted  A binary list that allows estimating restricted AdjPIN models by specifying which model parameters are assumed to be equal. It contains one or multiple of the following four elements {theta, mu, eps, d}. For instance, if theta is set to TRUE, then the probability of liquidity shock in no-information days, and in information days is assumed to be the same (θ=θ′). If any of the remaining rate elements {mu, eps, d} is set to TRUE, (say mu=TRUE), then the rate is assumed to be the same on the buy side, and on the sell side (μ_b=μ_s). If more than one element is set to TRUE, then the restrictions are combined. For instance, if the argument restricted is set to list(theta=TRUE,eps=TRUE,d=TRUE), then the restricted AdjPIN model is estimated, where θ=θ′, ε_b=ε_s, and Δ_b=Δ_s. If the value of the argument restricted is the empty list (list()), then all parameters of the model are assumed to be independent, and the unrestricted model is estimated. The default value is the empty list list().

num_init  An integer corresponds to the number of initial parameter sets to be generated. The default value is 20.

verbose  a binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when verbose is set to FALSE. The default value is TRUE.

Details

The argument ‘data’ should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The buy rate parameters {ε_b, μ_b, Δ_b} are randomly generated from the interval (minB, maxB), where minB (maxB) is the smallest (largest) value of buys in the dataset, under the condition that ε_b+μ_b+Δ_b<maxB. Analogously, the sell rate parameters {ε_s, μ_s, Δ_s} are randomly generated from the interval (minS, maxS), where minS (maxS) is the smallest(largest) value of sells in the dataset, under the condition that ε_s+μ_s+Δ_s<maxS.

Value

Returns a dataframe of numerical vectors of ten elements {α, δ, θ, θ′, ε_b, ε_s, μ_b, μ_s, Δ_b, Δ_s}.

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 # observations. Each observation corresponds to a day and contains the # total number of buyer-initiated trades ('B') and seller-initiated # trades ('S') on that day. To know more, type ?dailytrades

xdata <- dailytrades

# Obtain a dataframe of 20 random initial parameters for the MLE of # the AdjPIN model using the initials_adjpin_rnd().
initial.sets <- initials_adjpin_rnd(xdata, num_init = 20)

# Use the dataframe to estimate the AdjPIN model using the adjpin() function.
estimate <- adjpin(xdata, initialsets = initial.sets, verbose = FALSE)

# Show the value of adjusted PIN
show(estimate@adjpin)

---

initials_mpin

**MPIN initial parameter sets of Ersan (2016)**

**Description**

Based on the algorithm in Ersan (2016), generates initial parameter sets for the maximum likelihood estimation of the MPIN model.

**Usage**

`initials_mpin(data, layers = NULL, detectlayers = "EG", xtraclusters = 4, verbose = TRUE)`

**Arguments**

- **data**
  A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).

- **layers**
  An integer referring to the assumed number of information layers in the data. If the value of `layers` is `NULL`, then the number of layers is automatically determined by one of the following functions: `detectlayers_e()`, `detectlayers_eg()`, and `detectlayers_ecm()`. The default value is `NULL`.

- **detectlayers**
  A character string referring to the layer detection algorithm used to determine the number of layers in the data. It takes one of three values: "E", "EG", and "ECM". "E" refers to the algorithm in Ersan (2016), "EG" refers to the algorithm in Ersan and Ghachem (2022a); while "ECM" refers to the algorithm in Ghachem and Ersan (2022a). The default value is "EG". Comparative results between the layer detection algorithms can be found in Ersan and Ghachem (2022a).

- **xtraclusters**
  An integer used to divide trading days into `#(1 + layers + xtraclusters)` clusters, thereby resulting in `#comb(layers + xtraclusters, layers)` initial parameter sets in line with Ersan and Alici (2016), and Ersan (2016). The default value is 4 as chosen in Ersan (2016).

- **verbose**
  A binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when `verbose` is set to `FALSE`. The default value is `TRUE`. 
Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

Value

Returns a dataframe of initial parameter sets each consisting of \(3J + 2\) variables \(\{\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s\}\). \(\alpha\), \(\delta\), and \(\mu\) are vectors of length \(J\) where \(J\) is the number of layers in the MPIN model.

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 # observations. Each observation corresponds to a day and contains the # total number of buyer-initiated trades ('B') and seller-initiated # trades ('S') on that day. To know more, type ?dailytrades
donload('dailytrades')
xdata <- dailytrades

# Obtain a dataframe of initial parameter sets for estimation of the MPIN # model using the algorithm of Ersan (2016) with 3 extra clusters. # By default, the number of layers in the data is detected using the # algorithm of Ersan and Ghachem (2022a).
initparams <- initials_mpin(xdata, xtraclusters = 3)

# Show the initial parameter sets
show(round(initparams, 2))

# Use 10 randomly selected initial parameter sets from initparams to # estimate the probability of informed trading via mpin_ecm. The number # of information layers will be detected from the initial parameter sets.
nrow(initparams)
selectedsets <- initparams[sample(nrow(initparams), 10),]
estimate <- mpin_ecm(xdata, initialsets = selectedsets, verbose = FALSE)
# Display the estimated MPIN value
show(estimate@mpin)

# Display the estimated parameters as a numeric vector.
show(unlist(estimate@parameters))

# Store the posterior probabilities in a variable, and show the first 6 rows.
modelposteriors <- get_posteriors(estimate)
show(round(head(modelposteriors), 3))

initials_pin_ea

## Description

Based on the algorithm in Ersan and Alici (2016), generates initial parameter sets for the maximum likelihood estimation of the PIN model.

## Usage

```r
initials_pin_ea(data, xtraclusters = 4, verbose = TRUE)
```

## Arguments

- **data**: A dataframe with 2 variables: the first corresponds to buyer-initiated trades (brys), and the second corresponds to seller-initiated trades (sells).
- **xtraclusters**: An integer used to divide trading days into $(2 + xtraclusters)$ clusters, thereby resulting in $\text{comb}(1 + xtraclusters, 1)$ initial parameter sets in line with Ersan and Alici (2016). The default value is 4.
- **verbose**: A binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when `verbose` is set to `FALSE`. The default value is `TRUE`.

## Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The function `initials_pin_ea()` uses a hierarchical agglomerative clustering (HAC) to find initial parameter sets for the maximum likelihood estimation. The steps in Ersan and Alici (2016) algorithm differ from those used by Gan et al. (2015), and are summarized below.

Via the use of HAC, daily absolute order imbalances (AOIs) are grouped in $2 + J$ (default $J=4$) clusters. After sorting the clusters based on AOIs, they are combined into two larger groups of days (event and no-event) by merging neighboring clusters with each other. Consequently, those groups are formed in $\text{comb}(5, 1) = 5$ different ways. For each of the 5 configurations with which, days are
grouped into two (event group and no-event group), the procedure below is applied to obtain initial parameter sets.

Days in the event group (the one with larger mean AOI) are distributed into two groups, i.e. good-event days (days with positive OI) and bad-event days (days with negative OI). Initial parameters are obtained from the frequencies, and average trade rates of three types of days. See Ersan and Alici (2016) for further details.

The higher the number of the additional clusters ($xtraclusters$), the better is the estimation. Ersan and Alici (2016), however, have shown the benefit of increasing this number beyond 4 is marginal, and statistically insignificant.

**Value**

Returns a dataframe of initial sets each consisting of five variables {$\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s$}.

**References**


**Examples**

```r
# There is a preloaded quarterly dataset called 'dailytrades' with 60 observations. Each observation corresponds to a day and contains the total number of buyer-initiated trades ('B') and seller-initiated trades ('S') on that day. To know more, type ?dailytrades

dailytrades <- dailytrades

# Obtain a dataframe of initial parameters for the maximum likelihood estimation using the algorithm of Ersan and Alici (2016).

init.sets <- initials_pin_ea(dailytrades)

# Use the obtained dataframe to estimate the PIN model using the function pin() with custom initial parameter sets

estimate.1 <- pin(dailytrades, initialsets = init.sets, verbose = FALSE)

# pin_ea() directly estimates the PIN model using initial parameter sets generated using the algorithm of Ersan & Alici (2016).

estimate.2 <- pin_ea(dailytrades, verbose = FALSE)

# Check that the obtained results are identical

show(estimate.1@parameters)
show(estimate.2@parameters)
```
**initials_pin_gwj**

*Initial parameter set of Gan et al.(2015)*

**Description**

Based on the algorithm in Gan et al. (2015), generates an initial parameter set for the maximum likelihood estimation of the PIN model.

**Usage**

`initials_pin_gwj(data, verbose = TRUE)`

**Arguments**

- `data`: A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
- `verbose`: a binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when `verbose` is set to FALSE. The default value is TRUE.

**Details**

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

**Value**

Returns a dataframe containing numerical vector of five elements \{α, δ, μ, ε_b, ε_s\}.

**References**


**Examples**

```r
# There is a preloaded quarterly dataset called 'dailytrades' with 60 observations. Each observation corresponds to a day and contains the total number of buyer-initiated trades ('B') and seller-initiated trades ('S') on that day. To know more, type ?dailytrades
xdata <- dailytrades

# Obtain the initial parameter set for the maximum likelihood estimation using the algorithm of Gan et al.(2015).
initparams <- initials_pin_gwj(xdata)

# Use the obtained dataframe to estimate the PIN model using the function pin() with custom initial parameter sets
```
estimate.1 <- pin(xdata, initialsets = initparams, verbose = FALSE)

# pin_gwj() directly estimates the PIN model using an initial parameter set
# generated using the algorithm of Gan et al. (2015).

estimate.2 <- pin_gwj(xdata, "E", verbose = FALSE)

# Check that the obtained results are identical
show(estimate.1@parameters)
show(estimate.2@parameters)

---

initials_pin_yz

**Initial parameter sets of Yan and Zhang (2012)**

**Description**

Based on the grid search algorithm of Yan and Zhang (2012), generates initial parameter sets for the maximum likelihood estimation of the PIN model.

**Usage**

`initials_pin_yz(data, grid_size = 5, ea_correction = FALSE, verbose = TRUE)`

**Arguments**

- `data`: A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
- `grid_size`: An integer between 1, and 20; representing the size of the grid. The default value is 5. See more in details.
- `ea_correction`: A binary variable determining whether the modifications of the algorithm of Yan and Zhang (2012) suggested by Ersan and Alici (2016) are implemented. The default value is FALSE.
- `verbose`: a binary variable that determines whether information messages about the initial parameter sets, including the number of the initial parameter sets generated. No message is shown when verbose is set to FALSE. The default value is TRUE.

**Details**

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered. The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The argument `grid_size` determines the size of the grid of the variables: alpha, delta, and eps.b. If grid_size is set to a given value m, the algorithm creates a sequence starting from 1/2m, and ending in 1 -1/2m, with a step of 1/m. The default value of 5 corresponds to the size of the grid.
initials_pin_yz

in Yan and Zhang (2012). In that case, the sequence starts at \(0.1 = 1/(2 \times 5)\), and ends in \(0.9 = 1 - 1/(2 \times 5)\) with a step of \(0.2 = 1/m\).

The function initials_pin_yz() implements, by default, the original Yan and Zhang (2012) algorithm as the default value of ea_correction takes the value FALSE. When the value of ea_correction is set to TRUE; then, sets with irrelevant \(\mu\) values are excluded, and sets with boundary values are reintegrated in the initial parameter sets.

Value

Returns a dataframe of initial sets each consisting of five variables \{\(\alpha, \delta, \mu, \varepsilon_b, \varepsilon_s\}\).

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades

dxdata <- dailytrades

# The function pin_yz() allows the user to directly estimate the PIN model
# using the full set of initial parameter sets generated using the algorithm
# of Yan and # Zhang (2012).

estimate.1 <- pin_yz(xdata, verbose = FALSE)

# Obtaining the set of initial parameter sets using initials_pin_yz allows
# us to estimate the PIN model using a subset of these initial sets.

initparams <- initials_pin_yz(xdata, verbose = FALSE)

# Use 10 randomly chosen initial sets from the dataframe 'initparams' in
# order to estimate the PIN model using the function pin() with custom
# initial parameter sets

numberofsets <- nrow(initparams)
selectedsets <- initparams[sample(numberofsets, 10),]

estimate.2 <- pin(xdata, initialsets = selectedsets, verbose = FALSE)

# Compare the parameters and the pin values of both specifications

comparison <- rbind(c(estimate.1@parameters, pin = estimate.1@pin),
                   c(estimate.2@parameters, estimate.2@pin))

rownames(comparison) <- c("all", "10")
mpin_ecm

MPIN model estimation via an ECM algorithm

Description

Estimates the multilayer probability of informed trading (MPIN) using an Expectation Conditional Maximization algorithm, as in Ghachem and Ersan (2022a).

Usage

\[ \text{mpin\_ecm(data, layers = NULL, xtraclusters = 4, initialsets = NULL, ..., verbose = TRUE)} \]

Arguments

data A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).

layers An integer referring to the assumed number of information layers in the data. If the argument layers is given, then the ECM algorithm will use the number of layers provided. If layers is omitted, the function mpin\_ecm() will simultaneously optimize the number of layers as well as the parameters of the MPIN model.

xtraclusters An integer used to divide trading days into \#(1 + layers + xtraclusters) clusters, thereby resulting in \#comb((layers + xtraclusters, layers) initial parameter sets in line with Ersan and Alici (2016), and Ersan (2016). The default value is 4 as chosen in Ersan (2016).

initialsets A dataframe containing initial parameter sets for estimation of the MPIN model. The default value is NULL. If initialsets is NULL, the initial parameter sets are provided by the function initials\_mpin().

... Additional arguments passed on to the function mpin\_ecm. The recognized arguments are hyperparams, and is\_parallel.

- hyperparams is a list containing the hyperparameters of the ECM algorithm. When not empty, it contains one or more of the following elements: minalpha, maxeval, tolerance, criterion, and maxlayers. More about these elements are in the details section.
- is\_parallel is a logical variable that specifies whether the computation is performed using parallel or sequential processing. The default value is FALSE.

verbose (logical) a binary variable that determines whether detailed information about the steps of the estimation of the MPIN model is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.
Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The initial parameters for the expectation-conditional maximization algorithm are computed using the function initials_mpin() with default settings. The factorization of the MPIN likelihood function used is developed by Ersan (2016), and is implemented in fact_mpin().

The argument hyperparams contains the hyperparameters of the ECM algorithm. It is either empty or contains one or more of the following elements:

- \texttt{minalpha} (numeric) It stands for the minimum share of days belonging to a given layer, i.e., layers falling below this threshold are removed during the iteration, and the model is estimated with a lower number of layers. When missing, minalpha takes the default value of 0.001.
- \texttt{maxeval} (integer) It stands for maximum number of iterations of the ECM algorithm for each initial parameter set. When missing, maxeval takes the default value of 100.
- \texttt{tolerance} (numeric) The ECM algorithm is stopped when the (relative) change of log-likelihood is smaller than tolerance. When missing, tolerance takes the default value of 0.001.
- \texttt{criterion} (character) It is the model selection criterion used to find the optimal estimate for the MPIN model. It take one of these values "BIC", "AIC" and "AWE"; which stand for Bayesian Information Criterion, Akaike Information Criterion and Approximate Weight of Evidence, respectively (Akogul and Erisoglu 2016). When missing, criterion takes the default value of "BIC".
- \texttt{maxlayers} (integer) It is the upper limit of number of layers used for estimation in the ECM algorithm. If the argument \texttt{layers} is missing, the ECM algorithm will estimate MPIN models for all layers in the integer set from 1 to maxlayers. When missing, maxlayers takes the default value of 8.
- \texttt{maxinit} (integer) It is the maximum number of initial sets used for each individual estimation in the ECM algorithm. When missing, maxinit takes the default value of 100.

If the argument \texttt{layers} is given, then the Expectation Conditional Maximization algorithm will use the number of layers provided. If \texttt{layers} is omitted, the function \texttt{mpin_ecm()} will simultaneously optimize the number of layers as well as the parameters of the MPIN model. Practically, the function \texttt{mpin_ecm()} uses the ECM algorithm to optimize the MPIN model parameters for each number of layers within the integer set from 1 to 8 (or to maxlayers if specified in the argument hyperparams); and returns the optimal model with the lowest Bayesian information criterion (BIC) (or the lowest information criterion criterion if specified in the argument hyperparams).

Value

Returns an object of class \texttt{estimate.mpin.ecm}.

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 # observations. Each observation corresponds to a day and contains the # total number of buyer-initiated trades ('B') and seller-initiated # trades ('S') on that day. To know more, type ?dailytrades

data <- dailytrades

# Estimate the MPIN model using the expectation-conditional maximization # (ECM) algorithm.

# Estimate the MPIN model, assuming that there exists 2 information layers # in the dataset

estimate <- mpin_ecm(xdata, layers = 2, verbose = FALSE)

# Show the estimation output

show(estimate)

# Display the optimal parameters from the Expectation Conditional # Maximization algorithm

show(estimate@parameters)

# Display the global multilayer probability of informed trading

show(estimate@mpin)

# Display the multilayer probability of informed trading per layer

show(estimate@mpinJ)

# Display the first five rows of the initial parameter sets used in the # expectation-conditional maximization estimation

show(round(head(estimate@initialsets, 5), 4))

# Omit the argument 'layers', so the ECM algorithm optimizes both the # number of layers and the MPIN model parameters.

estimate <- mpin_ecm(xdata, verbose = FALSE)

# Show the estimation output
show(estimate)
# Display the optimal parameters from the estimation of the MPIN model using
# the expectation-conditional maximization (ECM) algorithm
show(estimate@parameters)
# Display the multilayer probability of informed trading
show(estimate@mpin)
# Display the multilayer probability of informed trading per layer
show(estimate@mpinJ)
# Display the first five rows of the initial parameter sets used in the
# expectation-conditional maximization estimation.
show(round(head(estimate@initialsets, 5), 4))
# ------------------------------------------------------------------------ #
# Tweak in the hyperparameters of the ECM algorithm
# ------------------------------------------------------------------------ #
# Create a variable ecm.params containing the hyperparameters of the ECM
# algorithm. This will surely make the ECM algorithm take more time to give
# results
ecm.params <- list(tolerance = 0.0000001)
# If we suspect that the data contains more than eight information layers, we
# can raise the number of models to be estimated to 10 as an example, i.e.,
# maxlayers = 10.
ecm.params$maxlayers <- 10
# We can also choose Approximate Weight of Evidence (AWE) for model
# selection instead of the default Bayesian Information Criterion (BIC)
ecm.params$criterion <- 'AWE'
# We can also increase the maximum number of initial sets to 200, in
# order to obtain higher level of accuracy for models with high number of
# layers. We set the sub-argument 'maxinit' to "200". Remember that its
# default value is "100".
ecm.params$maxinit <- 200
estimate <- mpin_ecm(xdata, xtraclusters = 2, hyperparams = ecm.params,
                     verbose = FALSE)
# We can change the model selection criterion by calling selectModel()
estimate <- selectModel(estimate, "AIC")
# We get the mpin_ecm estimation results for the MPIN model with 2 layers
# using the slot models. We then show the first five rows of the
# corresponding slot details.

models <- estimate@models
show(round(head(models[[2]]@details, 5), 4))

# We can also use the function getSummary to get an idea about the change in
# the estimation parameters as a function of the number of layers in the
# MPIN model. The function getSummary returns a dataframe that contains,
# among others, the number of layers of the model, the number of layers in
# the optimal model, the MPIN value, and the values of the different
# information criteria, namely AIC, BIC and AWE.

summary <- getSummary(estimate)

# We can plot the MPIN value and the layers at the optimal model as a
# function of the number of layers to see whether additional layers in the
# model actually contribute to a better precision in the probability of
# informed trading. Remember that the hyperparameter 'minalpha' is
# responsible for dropping layers with "frequency" lower than 'minalpha'.

plot(summary$layers, summary$MPIN,
     type = "o", col = "red",
     xlab = "MPIN model layers", ylab = "MPIN value"
)

plot(summary$layers, summary$em.layers,
     type = "o", col = "blue",
     xlab = "MPIN model layers", ylab = "layers at the optimal model"
)

---

### mpin_ml

**MPIN model estimation via standard ML methods**

**Description**

Estimates the multilayer probability of informed trading (MPIN) using the standard Maximum Likelihood method.

**Usage**

```r
mpin_ml(data, layers = NULL, xtraclusters = 4, initialsets = NULL,
        detectlayers = "EG", ..., verbose = TRUE)
```

**Arguments**

- `data`  
  A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).

- `layers`  
  An integer referring to the assumed number of information layers in the data. If the argument `layers` is given, then the maximum likelihood estimation will use the number of layers provided. If `layers` is omitted, the function `mpin_ml()` will find the optimal number of layers using the algorithm developed in Ersan and Ghachem (2022a) (as default).
xtraclusters An integer used to divide trading days into \((1 + \text{layers} + \text{xtraclusters})\) clusters, thereby resulting in \(#\text{comb}(\text{layers} + \text{xtraclusters}, \text{layers})\) initial parameter sets in line with Ersan and Alici (2016), and Ersan (2016). The default value is 4 as chosen in Ersan (2016).

initialsets A dataframe containing initial parameter sets for the estimation of the MPIN model. The default value is NULL. If initialsets is NULL, the initial parameter sets are determined by the function initials_mpin()

detectlayers A character string referring to the layer detection algorithm used to determine the number of layer in the data. It takes one of three values: "E", "EG", and "ECM". "E" refers to the algorithm in Ersan (2016), "EG" refers to the algorithm in Ersan and Ghachem (2022a); while "ECM" refers to the algorithm in Ghachem and Ersan (2022a). The default value is "EG". Comparative results between the layer detection algorithms can be found in Ersan and Ghachem (2022a).

... Additional arguments passed on to the function mpin_ml. The recognized argument is is_parallel. is_parallel is a logical variable that specifies whether the computation is performed using parallel processing. The default value is FALSE.

verbose A binary variable that determines whether detailed information about the steps of the estimation of the MPIN model is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.

Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

Value

Returns an object of class estimate.mpin

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 # observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated trades ('S') on that day. To know more, type ?dailytrades

```r
data <- dailytrades
```

# Estimate MPIN model using the standard ML method

```r
estimate <- mpin_ml(xdata, layers = 1, xtraclusters = 2, verbose = FALSE)
```

# Show the estimation output

```r
show(estimate)
```

# Estimate the MPIN model using the function mpin_ml(), without specifying the number of layers. The number of layers is then detected using Ersan and Ghachem (2022a).

```r
estimate <- mpin_ml(xdata, xtraclusters = 2, verbose = FALSE)
```

# Show the estimation output

```r
show(estimate)
```

# Display the likelihood-maximizing parameters

```r
show(estimate@parameters)
```

# Display the global multilayer probability of informed trading

```r
show(estimate@mpin)
```

# Display the multilayer probabilities of informed trading per layer

```r
show(estimate@mpinJ)
```

# Display the first five initial parameters sets used in the maximum likelihood estimation

```r
show(round(head(estimate@initialsets, 5), 4))
```

---

**PIN** estimation - custom initial parameter sets

**Description**

Estimates the Probability of Informed Trading (PIN) using custom initial parameter sets.
Usage

pin(data, initialsets, factorization = "E", verbose = TRUE)

Arguments

data A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).

initialsets A dataframe with the following variables in this order ($\alpha$, $\delta$, $\mu$, $\varepsilon_b$, $\varepsilon_s$).

factorization A character string from ("EH0", "LK", "E", "NONE") referring to a given factorization. The default value is set to "E".

verbose A binary variable that determines whether detailed information about the steps of the estimation of the PIN model is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.

Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The factorization variable takes one of four values:

- "EH0" refers to the factorization in Easley et al. (2010)
- "LK" refers to the factorization in Lin and Ke (2011)
- "E" refers to the factorization in Ersan (2016)
- "NONE" refers to the original likelihood function - with no factorization

Value

Returns an object of class estimate.pin

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 # observations. Each observation corresponds to a day and contains the # total number of buyer-initiated trades ('B') and seller-initiated # trades ('S') on that day. To know more, type ?dailytrades

xdata <- dailytrades

#--------------------------------------------------------------
```
# Using generic function pin()
#--------------------------------------------------------------

# Define initial parameters:
# initialset = (alpha, delta, mu, eps.b, eps.s)
initialset <- c(0.3, 0.1, 800, 300, 200)

# Estimate the PIN model using the factorization of the PIN likelihood
# function by Ersan (2006)
estimate <- pin(xdata, initialsets = initialset, verbose = FALSE)

# Display the estimated PIN value
show(estimate@pin)

# Display the estimated parameters
show(estimate@parameters)

# Store the initial parameter sets used for MLE in a dataframe variable,
# and display its first five rows
initialsets <- estimate@initialsets
show(head(initialsets, 5))
```

---

### Description

Estimates the Probability of Informed Trading (PIN) using the initial sets from the algorithm in Ersan and Alici (2016).

### Usage

`pin_ea(data, factorization, xtraclusters = 4, verbose = TRUE)`

### Arguments

- **data**  A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
- **factorization**  A character string from \{"E","EHO","LK","NONE"\} referring to a given factorization. The default value is "E".
- **xtraclusters**  An integer used to divide trading days into \(#(2 + \text{xtraclusters})\) clusters, thereby resulting in \(#\text{comb}(1 + \text{xtraclusters}, 1)\) initial parameter sets in line with Ersan and Alici (2016). The default value is 4.
- **verbose**  A binary variable that determines whether detailed information about the steps of the estimation of the PIN model is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.
Details

The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The factorization variable takes one of four values:

- "EH0" refers to the factorization in Easley et al. (2010)
- "LK" refers to the factorization in Lin and Ke (2011)
- "E" refers to the factorization in Ersan (2016)
- "NONE" refers to the original likelihood function - with no factorization

The function pin_ea() implements the algorithm detailed in Ersan and Alici (2016). The higher the number of the additional layers (xtraclusters), the better is the estimation. Ersan and Alici (2016), however, have shown the benefit of increasing this number beyond 5 is marginal, and statistically insignificant.

The function initials_pin_ea() provides the initial parameter sets obtained through the implementation of the Ersan and Alici (2016) algorithm. For further information on the initial parameter set determination, see initials_pin_ea().

Value

Returns an object of class estimate.pin

References


Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 # observations. Each observation corresponds to a day and contains the # total number of buyer-initiated trades ('B') and seller-initiated # trades ('S') on that day. To know more, type ?dailytrades

xdata <- dailytrades

# Estimate the PIN model using the factorization of Ersan (2016), and initial # parameter sets generated using the algorithm of Ersan and Alici (2016). # The argument xtraclusters is omitted so will take its default value 4.
estimate <- pin_ea(xdata, verbose = FALSE)

# Display the estimated PIN value
show(estimate@pin)

# Display the estimated parameters
show(estimate@parameters)

# Store the initial parameter sets used for MLE in a dataframe variable,
# and display its first five rows
initialsets <- estimate@initialsets
show(head(initialsets, 5))

pin_gwj

PIN estimation - initial parameter set of Gan et al. (2015)

Description
Estimates the Probability of Informed Trading (PIN) using the initial set from the algorithm in Gan et al. (2015).

Usage
pin_gwj(data, factorization = "E", verbose = TRUE)

Arguments
data
 ~ A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).

factorization
 ~ A character string from ("EHO", "LK", "E", "NONE") referring to a given factor-
ization. The default value is set to "E".

verbose
 ~ A binary variable that determines whether detailed information about the steps
of the estimation of the PIN model is displayed. No output is produced when
verbose is set to FALSE. The default value is TRUE.

Details
The argument 'data' should be a numeric dataframe, and contain at least two variables. Only the first
two variables will be considered: The first variable is assumed to correspond to the total number
of buyer-initiated trades, while the second variable is assumed to correspond to the total number
of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be
ignored.

The factorization variable takes one of four values:

- "EHO" refers to the factorization in Easley et al. (2010)
- "LK" refers to the factorization in Lin and Ke (2011)
- "E" refers to the factorization in Ersan (2016)


• "NONE" refers to the original likelihood function - with no factorization

The function `pin_gwj()` implements the algorithm detailed in Gan et al. (2015). You can use the function `initials_pin_gwj()` in order to get the initial parameter set.

**Value**

Returns an object of class `estimate.pin`

**References**


**Examples**

# There is a preloaded quarterly dataset called 'dailytrades' with 60
# observations. Each observation corresponds to a day and contains the
# total number of buyer-initiated trades ('B') and seller-initiated
# trades ('S') on that day. To know more, type ?dailytrades

`xdata <- dailytrades`

# Estimate the PIN model using the factorization of Ersan (2016), and initial
# parameter sets generated using the algorithm of Gan et al. (2015).
# The argument xtraclusters is omitted so will take its default value 4.

`estimate <- pin_gwj(xdata, verbose = FALSE)`

# Display the estimated PIN value

`show(estimate@pin)`

# Display the estimated parameters

`show(estimate@parameters)`

# Store the initial parameter sets used for MLE in a dataframe variable,
# and display its first five rows

`initialsets <- estimate@initialsets`

`show(head(initialsets, 5))`
**Description**

Estimates the Probability of Informed Trading (PIN) using the initial parameter sets generated using the grid search algorithm of Yan and Zhang (2012).

**Usage**

```r
pin_yz(data, factorization, ea_correction = FALSE, grid_size = 5, verbose = TRUE)
```

**Arguments**

- `data`: A dataframe with 2 variables: the first corresponds to buyer-initiated trades (buys), and the second corresponds to seller-initiated trades (sells).
- `factorization`: A character string from (`"EHO","LK","E","NONE"`) referring to a given factorization. The default value is "E".
- `ea_correction`: A binary variable determining whether the modifications of the algorithm of Yan and Zhang (2012) suggested by Ersan and Alici (2016) are implemented. The default value is FALSE.
- `grid_size`: An integer between 1, and 20; representing the size of the grid. The default value is 5. See more in details.
- `verbose`: A binary variable that determines whether detailed information about the steps of the estimation of the PIN model is displayed. No output is produced when verbose is set to FALSE. The default value is TRUE.

**Details**

The argument `data` should be a numeric dataframe, and contain at least two variables. Only the first two variables will be considered: The first variable is assumed to correspond to the total number of buyer-initiated trades, while the second variable is assumed to correspond to the total number of seller-initiated trades. Each row or observation correspond to a trading day. NA values will be ignored.

The factorization variable takes one of four values:

- "EHO" refers to the factorization in Easley et al. (2010)
- "LK" refers to the factorization in Lin and Ke (2011)
- "E" refers to the factorization in Ersan (2016)
- "NONE" refers to the original likelihood function - with no factorization

The argument `grid_size` determines the size of the grid of the variables: alpha, delta, and eps.b. If `grid_size` is set to a given value m, the algorithm creates a sequence starting from 1/2m, and ending in 1-1/2m, with a step of 1/m. The default value of 5 corresponds to the size of the grid in Yan and Zhang (2012). In that case, the sequence starts at 0.1 = 1/(2 x 5), and ends in 0.9 = 1 -1/(2 x 5) with a step of 0.2 = 1/m.

The function `pin_yz()` implements, by default, the original Yan and Zhang (2012) algorithm as the default value of `ea_correction` takes the value FALSE. When the value of `ea_correction` is set to TRUE; then, sets with irrelevant mu values are excluded, and sets with boundary values are reintegrated in the initial parameter sets.
**Value**

Returns an object of class `estimate.pin`

**References**


**Examples**

# There is a preloaded quarterly dataset called 'dailytrades' with 60 observations. Each observation corresponds to a day and contains the total number of buyer-initiated trades ('B') and seller-initiated trades ('S') on that day. To know more, type ?dailytrades

```r
xdata <- dailytrades
```

# Estimate the PIN model using the factorization of Lin and Ke(2011), and initial parameter sets generated using the algorithm of Yan & Zhang (2012).
# In contrast to the original algorithm, we set the grid size for the grid search algorithm at 3. The original algorithm assumes a grid size of 5.

```r
estimate <- pin_yz(xdata, "LK", grid_size = 3, verbose = FALSE)
```

# Display the estimated PIN value

```r
show(estimate@pin)
```

# Display the estimated parameters

```r
show(estimate@parameters)
```

# Store the initial parameter sets used for MLE in a dataframe variable, and display its first five rows

```r
initialsets <- estimate@initialsets
show(head(initialsets, 5))
```
set_display_digits  Package-wide number of digits

Description

Sets the number of digits to display in the output of the different package functions.

Usage

set_display_digits(digits = list())

Arguments

digits  A list of numbers corresponding to the different display digits. The default value is list().

Details

The parameter digits is a named list. It will be containing:

- $d_1$: contains the number of display digits for the values of probability estimates such as $\alpha$, $\delta$, pin, mpin, mpin(j), adjpin, psos, $\theta$, and $\theta'$.
- $d_2$: contains the number of display digits for the values of $\mu$, $\varepsilon_b$ and $\varepsilon_s$, as well as information criteria: AIC, BIC, and AWE.
- $d_3$: contains the number of display digits for the remaining values such as vpin statistics and likelihood value.

If the function is called with no arguments, the display digits will be reset to the default values, i.e., list($d_1 = 6$, $d_2 = 2$, $d_3 = 3$). If the argument digits is not omitted, the function will only accept a list containing exactly three numerical values, each ranging between 0 and 10. The list can be named or unnamed. If the numbers in the argument digits are not integers, they will be rounded.

Examples

# There is a preloaded quarterly dataset called 'dailytrades' with 60 observations. Each observation corresponds to a day and contains the total number of buyer-initiated trades ('B') and seller-initiated trades ('S') on that day. To know more, type ?dailytrades
xdata <- dailytrades

# We show the output of the function pin_ea() using the default values of display digits. We then change these values using the function set_display_digits(), before displaying the same estimate.pin object again to see the difference.
mmodel <- pin_ea(xdata, verbose = FALSE)
show(model)

# Change the number of digits for $d_1$ to 3, of $d_2$ to 0 and of $d_3$ to 2
set_display_digits(list(3, 0, 2))
# No need to run the function mpin_ml() again to update the display of an estimate.mpin object. This holds for all estimate S4 objects.

show(model)

---

**Estimation of Volume-Synchronized PIN model**

**Description**

Estimates the Volume-Synchronized Probability of Informed Trading as developed in Easley et al. (2011) and Easley et al. (2012).

**Usage**

```r
vpin(data, timebarsize = 60, buckets = 50, samplength = 50, tradinghours = 24, verbose = TRUE)
```

**Arguments**

- `data`: A dataframe with 3 variables: `{timestamp, price, volume}`.
- `timebarsize`: An integer referring to the size of timebars in seconds. The default value is 60.
- `buckets`: An integer referring to the number of buckets in a daily average volume. The default value is 50.
- `samplength`: An integer referring to the sample length or the window size used to calculate the VPIN vector. The default value is 50.
- `tradinghours`: An integer referring to the length of daily trading sessions in hours. The default value is 24.
- `verbose`: A binary variable that determines whether detailed information about the steps of the estimation of the VPIN model is displayed. No output is produced when `verbose` is set to `FALSE`. The default value is `TRUE`.

**Details**

The dataframe data should contain at least three variables. Only the first three variables will be considered and in the following order `{timestamp, price, volume}`.

The property `@bucketdata` is created as in Abad and Yague (2012).

The argument `timebarsize` is in seconds enabling the user to implement shorter than 1 minute intervals. The default value is set to 1 minute (60 seconds) following Easley et al. (2011, 2012).

The parameter `tradinghours` is used to eventually correct the duration per bucket. The duration of a given bucket is the difference between the timestamp of the last trade endtime and the timestamp of the first trade stime in the bucket. If the first trade and the last trade in a bucket occur in two different days, and the market trading session does not cover a full day (24 hours); then the duration of the bucket will be inflated. Assume that the daily trading session is 8 hours (`tradinghours=8`), the start time of a bucket is `2018-10-12 17:06:40` and its end time is `2018-10-13 09:36:00`. A straightforward calculation gives that the duration of this bucket is 59,360 secs. However, this duration includes the time during which the market is closed (16 hours). The corrected duration takes into consideration only the time of market activity: duration=59,360-16*3600 = 1760 secs, i.e., about 30 minutes.
Value

Returns an object of class estimate.vpin.

References


Examples

# There is a preloaded dataset called 'hfdata' contained in the package.
# It is an artificially created high-frequency trading data. The dataset
# contains 100 000 trades and five variables 'timestamp', 'price',
# 'volume', 'bid' and 'ask'. For more information, type ?hfdata.

xdata <- hfdata

# Estimate VPIN model, using the following parameter set where the time
# bar size is 5 minutes, i.e., 300 seconds (timebarsize = 300), 50
# buckets per average daily volume (buckets = 50), and a window size of
# 250 for the VPIN calculation (samplength = 250).

estimate <- vpin(xdata, timebarsize = 300, buckets = 50, samplength = 250)

# Display a description of the estimate

show(estimate)

# Plot the estimated VPIN vector

plot(estimate@vpin, type = "l", xlab = "time", ylab = "VPIN", col = "blue")

# Display the parameters of VPIN estimates

show(estimate@parameters)

# Store the computed data of the different buckets in a dataframe 'buckets'.
# Display the first 10 rows of the dataframe 'buckets'.

buckets <- estimate@bucketdata

show(head(buckets, 10))

# Store the daily VPIN values (weighted and unweighted) in a dataframe
# 'dayvpin'.

# Display the first 10 rows of the dataframe 'dayvpin'.

dayvpin <- estimate@dailyvpin

show(head(dayvpin, 10))
Index

* datasets
  dailytrades, 12
  hfdata, 35

adjpin, 4, 7
aggregate_trades, 4, 10
dailytrades, 5, 12
data.series-class, 5, 12
dataset-class, 5, 13
detectlayers, 15
detectlayers.e, 4
detectlayers.e (detectlayers), 15
detectlayers.ecm, 4
detectlayers.ecm (detectlayers), 15
detectlayers_eg, 4
detectlayers_eg (detectlayers), 15
estimate.adjpin-class, 5, 16
estimate.mpin-class, 5, 18
estimate.mpin.ecm-class, 5, 19
estimate.pin-class, 5, 21
estimate.vpin-class, 5, 22
fact_adjpin, 4
fact_adjpin (factorizations), 23
fact_mpin, 4
fact_mpin (factorizations), 23
fact_pin_e, 4
fact_pin_e (factorizations), 23
fact_pin_ego, 4
fact_pin_ego (factorizations), 23
fact_pin lk, 4
fact_pin lk (factorizations), 23
factorizations, 23
generatedata_adjpin, 4, 26
generatedata_mpin, 4, 29
get_posteriors, 4, 33
getSummary (estimate.mpin.ecm-class), 19
getSummary, estimate.mpin.ecm-method
  (estimate.mpin.ecm-class), 19

hfdata, 5, 35

initials_adjpin, 4, 36
initials_adjpin_cl, 4, 37
initials_adjpin_rnd, 4, 39
initials_mpin, 4, 41
initials_pin_ea, 4, 43
initials_pin_gwj, 4, 45
initials_pin_zy, 4, 46

mpin_ecm, 4, 48
mpin_ml, 4, 52
pin, 5, 54
pin_ea, 5, 56
pin_gwj, 5, 58
pin_zy, 5, 60
PINestimation-package, 2
selectModel (estimate.mpin.ecm-class),
  19
selectModel, estimate.mpin.ecm-method
  (estimate.mpin.ecm-class), 19
set_display_digits, 62
show, data.series-method
  (data.series-class), 12
show, dataset-method (dataset-class), 13
show, estimate.adjpin-method
  (estimate.adjpin-class), 16
show, estimate.mpin-method
  (estimate.mpin-class), 18
show, estimate.mpin.ecm-method
  (estimate.mpin.ecm-class), 19
show, estimate.pin-method
  (estimate.pin-class), 21
show, estimate.vpin-method
  (estimate.vpin-class), 22

vpin, 5, 63