Package 'asharechips'

November 10, 2026

Title Estimate Holding Cost Distributions for Chinese Equities

Version 0.2.1

Description Estimates holding-cost distributions of mainland Chinese A-share equities from daily price, volume, amount and turnover data. Implements the linear turnover-based cost-distribution recurrence of Liu and Wang (2018) <doi:10.1016/j.physa.2017.11.154>, and adds a package-specific exponential-retention alternative and price-allocation choices. Provides optional adjustment factors and initial distributions, offline preparation of 'Tushare' tables, weighted cost summaries and diagnostics for reproducible sensitivity analysis.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports graphics

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Ruihong Zhao [aut, cre]

Maintainer Ruihong Zhao <3142115360@qq.com>

Repository CRAN

Date/Publication 2026-11-10 17:20:08 UTC

Contents

asharechips-package 2
chip_distribution 2
chip_mean_cost 3
estimate_chip_distribution 4
prepare_tushare_data 8

Index **10**

`asharechips-package` *Estimate A-Share Holding-Cost Distributions*

Description

Offline preparation and estimation of daily holding-cost distributions for mainland Chinese A-share equities. Models expose their initialization, turnover replacement, price allocation and price-basis assumptions.

Details

The preparation function joins locally supplied tables. The estimator returns weights, statistics and diagnostics. A weighted-mean cost utility is also provided. Examples and the vignette use synthetic data and require no token. See the introduction vignette for a worked example and limitations.

Author(s)

Ruihong Zhao <3142115360@qq.com>

See Also

`estimate_chip_distribution`

`chip_distribution` *Inspect and Plot a Chip Distribution*

Description

Print a brief model summary, extract statistics or weights, or plot estimated cost on the vertical axis against estimated chip share on the horizontal axis.

Usage

```
## S3 method for class 'chip_distribution'
print(x, ...)
## S3 method for class 'chip_distribution'
summary(object, ...)
## S3 method for class 'chip_distribution'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
## S3 method for class 'chip_distribution'
plot(x, ..., col = "#4C78A8", close_col = "#D62728")
```

Arguments

x, object	A chip_distribution object returned by estimate_chip_distribution().
...	For plot, named arguments forwarded to graphics::plot; defaults such as xlab, ylim and main can be overridden. x and y cannot be overridden. For as.data.frame, arguments forwarded to the data-frame method. Otherwise ignored.
row.names, optional	Passed to as.data.frame.
col	Colour for horizontal chip segments.
close_col	Colour for the final closing-price reference line.

Value

summary returns the one-row summary data frame; as.data.frame returns the distribution data frame. print and plot return x invisibly. plot draws on the current graphics device without opening an external viewer or saving files.

See Also

estimate_chip_distribution

Examples

```
daily <- data.frame(
  trade_date = c("20260102", "20260105"),
  high = c(10, 11), low = c(10, 11), close = c(10, 11),
  vol = c(100, 100), amount = c(100, 110), turnover_rate = c(0, 20)
)
chips <- estimate_chip_distribution(daily)
print(chips)
summary(chips)
as.data.frame(chips)
plot(chips, main = "Synthetic example", xlab = "Share (%)")
```

---

chip_mean_cost *Calculate a Share-Weighted Mean Cost*

Description

Calculate the weighted mean of prices without R's vector recycling. Prices and share counts must correspond in the same order.

Usage

```
chip_mean_cost(price, shares)
```

Arguments

price	Non-empty vector of positive, finite prices. Numeric text and factors containing numeric text are also accepted.
shares	Equally sized vector of non-negative, finite share counts or weights with a finite positive sum.

Value

A numeric scalar equal to the sum of prices multiplied by normalized share counts. Prices retain the input monetary units.

Examples

```
chip_mean_cost(c(10, 11, 12), c(50, 20, 30)) # 10.8
chip_mean_cost(c(10, 11, 12), c(0.5, 0.2, 0.3))
```

---

estimate_chip_distribution

Estimate a Holding-Cost Distribution from Daily Market Data

Description

Estimate the cost distribution of circulating shares using an explicitly specified daily turnover-replacement model. Returns auditable weights, descriptive cost statistics and a per-day diagnostic record.

Usage

```
estimate_chip_distribution(
  data, date_col = "trade_date", high_col = "high", low_col = "low",
  close_col = "close", turnover_col = "turnover_rate", volume_col = "vol",
  amount_col = "amount", adjustment_col = NULL, price_step = 0.01,
  price_method = c("vwap", "typical", "close", "uniform"),
  turnover_transform = c("exponential", "linear"),
  initial_distribution = NULL
)
```

Arguments

data	A non-empty data frame containing one row per observed day and one stock. A `ts_code` column, if present, must identify a single stock. Missing trading days are not imputed.
date_col	Date column name. Accepts `Date` objects, eight-digit `YYYYMMDD` numbers or text, or `YYYY-MM-DD` text. Dates must be valid and unique; they are normalized to `YYYYMMDD` and sorted.

`high_col`, `low_col`, `close_col`
: Column names for positive finite daily high, low and closing prices in CNY per share. High must be at least low and close must lie in that range. Use unadjusted prices when supplying factors.

`turnover_col`
: Column name for non-negative daily turnover in percentage points: `5` means five percent. Tushare's `turnover_rate` uses unrestricted circulating shares; `turnover_rate_f` uses a different denominator and should not be mixed with it.

`volume_col`, `amount_col`
: Volume and amount column names, required for VWAP. Units are lots of 100 shares and thousand CNY, respectively, as in Tushare daily data. Zero values are permitted together only on rows on which no transaction price is needed. Negative values are rejected.

`adjustment_col`
: Optional column name for strictly positive adjustment factors. Historical prices are multiplied by that day's factor divided by the last observed day's factor. Without factors, prices must already share a consistent basis.

`price_step`
: Positive price-grid spacing in CNY. Smaller spacing increases resolution and computational work. Grids above one million bins are rejected. The spacing must exceed the numerical resolution of the input prices.

`price_method`
: Placement of newly traded weight: `"vwap"` uses `amount * 10 / vol`; `"typical"` uses the mean of high, low and close; `"close"` uses close; `"uniform"` distributes weight across the high-low interval using overlap with each grid cell.

`turnover_transform`
: `"exponential"` retains $\exp(-r/100)$ of previous weight, allowing turnover above 100 percent. `"linear"` retains $1 - r/100$ and requires turnover at most 100 percent.

`initial_distribution`
: Optional non-empty data frame with positive `price` and non-negative `weight` columns. Weights are normalized and must have a finite positive sum. Prices must already use the last observed day's price basis. The distribution describes holdings immediately before the first row; that row's turnover is therefore applied.

Details

Let w_{t-1} be the previous grid distribution, g_t the normalized distribution assigned to newly traded shares and s_t the retention fraction. The recurrence is

$$w_t = s_t w_{t-1} + (1 - s_t) g_t.$$

The exponential transformation assumes homogeneous random replacement; the linear alternative assumes a fraction of existing shares changes hands once. Neither transformation identifies actual sellers or individual accounts.

If `initial_distribution` is `NULL`, all initial weight is placed using the first day's price rule. Its turnover is not used as a replacement step. If an initial distribution is supplied, every row is a replacement step. `history_coverage` describes the amount of initial weight replaced by those steps, not coverage of actual shareholder records.

Point prices are rounded to the nearest grid centre using R's ties-to-even rounding. Centres are spaced by `price_step`; the first centre is the largest non-negative grid multiple not exceeding the minimum input price, except that a zero centre is replaced by the minimum positive input price. Uniform allocation uses overlap with cells extending half a step either side of each centre. A zero-width high-low interval is allocated to one centre. No arbitrary small-weight threshold is used; only zero numerical weights are omitted. Very old weights may underflow in long histories.

VWAP values beyond the high-low range by more than `1e-6 * max(1, high)` in original price units are rejected. Smaller rounding discrepancies are clamped to the range and flagged. The optional `float_share` column must be positive; a consecutive relative change above `1e-6` produces a warning and a diagnostic flag. Issuances, unlocks, repurchases, fees and taxes are not modelled. Dividend factors provide a comparable price basis rather than literal historical cash acquisition costs.

Runtime is approximately proportional to the product of the number of days and grid bins. Memory usage is proportional to bins plus input rows.

Value

A `chip_distribution` object, a list with four components:

distribution Data frame with positive grid `price`, normalized `weight` (sum one, subject to numerical precision), and `percent = 100 * weight`.

summary One-row data frame, with these groups of fields:

- Stock identifier: `ts_code`.
- Date interval: `start_date` and `end_date`.
- Input count: `observations`. Final price: `last_close`.
- Weighted mean: `average_cost`. Weighted median: `median_cost`.
- Peak cost: `peak_price`. Peak share: `peak_percent`. Tied peaks use the lowest price.
- Profitable share: `profit_ratio` and `profit_percent`; only costs strictly below the last close are included.
- Equal-price share: `breakeven_ratio`. Higher-cost share: `loss_ratio`. Equality allows only numerical rounding error. The profit, break-even and loss ratios sum to one.
- Central interval: `cost_low_90` and `cost_high_90`, the discrete 5th and 95th weighted quantiles.
- Range width: `concentration_90`, the interval's width divided by the sum of its endpoints. Smaller values mean narrower cost ranges, not higher ownership concentration.
- Turnover sum: `cumulative_turnover_percent`, over applied rows.
- Surviving prior: `initialization_weight`, the product of retention fractions over applied update rows.
- Replaced prior: `history_coverage`, one minus surviving prior.

diagnostics One row per sorted date with the following fields:

- Date: `trade_date`. Turnover: `turnover_rate`.
- Proposed replacement: `replacement_fraction`.
- Used replacement: `applied_replacement_fraction`; zero for an initialized first row.
- Old-weight retention: `retention`.
- Final-basis price: `transaction_price`.

- Original-basis price: `raw_transaction_price`.
- Factor normalization: `adjustment_ratio`.
- Minor VWAP corrections: `vwap_rounding_adjusted`.
- Initialization flag: `initialized_here`.
- Share-count change flag: `capital_changed`.

`vwap_outside_range` is retained for backwards compatibility and is always false because material violations are errors. Uniform allocation's transaction price is a representative typical price, not the allocation itself.

settings Column mappings, grid spacing and origin, price and turnover methods, initialization rule and `price_basis = "last_trading_day"`.

The estimates depend on modelling assumptions, observed history and grid spacing. They do not establish investor identity or future returns.

References

Liu, H. and Wang, L. (2018). "The price momentum of stock in distribution." *Physica A: Statistical Mechanics and its Applications*, 492, 2336–2344. doi:10.1016/j.physa.2017.11.154.

Tushare daily prices and units: https://tushare.pro/document/2?doc_id=27. Turnover definitions: https://tushare.pro/document/2?doc_id=32. Adjustment factors: https://tushare.pro/document/2?doc_id=28.

See Also

`prepare_tushare_data`, `chip_mean_cost`, `plot.chip_distribution`

Examples

```
# Entirely synthetic data; no network access or API token is required.
daily <- data.frame(
  trade_date = c("20260106", "20260102", "20260105"),
  high = c(12, 10, 11), low = c(12, 10, 11), close = c(12, 10, 11),
  vol = c(100, 100, 100), amount = c(120, 100, 110),
  turnover_rate = c(30, 0, 20)
)
chips <- estimate_chip_distribution(daily)
summary(chips)
as.data.frame(chips)

initial <- data.frame(price = c(8, 12), weight = c(0.5, 0.5))
estimate_chip_distribution(daily, initial_distribution = initial)
```

`prepare_tushare_data` *Prepare Matching Tushare Tables without Network Access*

Description

Validate and join locally supplied daily price, turnover and optional factor tables with complete one-to-one stock/date matches. Dates are normalized and sorted. No API requests, token storage or file writes are performed.

Usage

```
prepare_tushare_data(daily, daily_basic, adj_factor = NULL,
                     ts_code = NULL, as_of = NULL)
```

Arguments

`daily`	Unadjusted daily prices, supplied as a non-empty data frame with `ts_code`, `trade_date`, `high`, `low`, `close`, `vol` (lots of 100 shares), and `amount` (thousand CNY).
`daily_basic`	Non-empty data frame with `ts_code`, `trade_date`, `turnover_rate` (percentage points) and optionally `float_share` (ten thousand shares).
`adj_factor`	Optional factor table with three columns: `ts_code` for stock codes; `trade_date` for dates; and `adj_factor` for positive factors.
`ts_code`	Optional single stock code to select from all tables. Without it, each table must refer to one and the same stock.
`as_of`	Optional cutoff date. Accepts a `Date`, `YYYYMMDD` number or text, or `YYYY-MM-DD` text. Rows after it are excluded. The cutoff need not be a trading date; the last remaining daily row determines the price basis used by the estimator.

Details

Duplicate selected stock/date keys, different stocks or a missing turnover or factor row for any retained daily date are errors. Extra dates in the indicator or factor table are ignored. Daily prices remain authoritative; for example, `daily_basic$close` does not replace `daily$close`. Columns from `daily` are preserved except reserved indicator/factor columns, which are supplied by their authoritative tables. Arithmetic and price-range validation is performed by the estimator.

Value

A data frame ready for `estimate_chip_distribution()`, with daily price columns, matched `turnover_rate`, optional `float_share` and optional `adj_factor`. If factors are included, explicitly specify `adjustment_col = "adj_factor"` in the estimator.

See Also

`estimate_chip_distribution`

Examples

```
daily <- data.frame(
  ts_code = rep("000001.SZ", 2), trade_date = c("20260105", "20260102"),
  high = c(11, 10), low = c(11, 10), close = c(11, 10),
  vol = c(100, 100), amount = c(110, 100)
)
basic <- data.frame(
  ts_code = rep("000001.SZ", 2), trade_date = c("20260102", "20260105"),
  turnover_rate = c(0, 20)
)
input <- prepare_tushare_data(daily, basic)
estimate_chip_distribution(input)
```

Index

∗ **models**
estimate_chip_distribution, 4

as.data.frame.chip_distribution
(chip_distribution), 2
asharechips (asharechips-package), 2
asharechips-package, 2

chip_distribution, 2
chip_mean_cost, 3, *7*

estimate_chip_distribution, *2*, *3*, 4, *8*

plot.chip_distribution, *7*
plot.chip_distribution
(chip_distribution), 2
prepare_tushare_data, *7*, 8
print.chip_distribution
(chip_distribution), 2

summary.chip_distribution
(chip_distribution), 2