Package 'agriME'

September 9, 2026

Type Package

Title Agricultural Marketing Efficiency and Price Spread Analysis

Version 0.1.0

Description Provides reproducible tools for analysing agricultural marketing channels, price spread, the producer's share in the consumer price, intermediary costs and margins, and alternative indices of marketing efficiency. Implements conventional, Shepherd, and Acharya measures; validates stage-level channel accounts; compares and ranks channels; and supplies bootstrap confidence intervals, scenario sensitivity analysis, loss-adjusted margins, break-even calculations, and base-graphics methods. Methodological context is provided by Acharya and Agarwal (2021, ISBN:9789389688061) and Shepherd (2007) <https://www.fao.org/4/u8770e/u8770e00.htm>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

NeedsCompilation no

Imports stats

Author Chiranjit Mazumder [aut, cre],
Bikramjeet Ghose [aut],
Pramit Pandit [aut]

Maintainer Chiranjit Mazumder <chiranjit@iari.res.in>

Repository CRAN

Date/Publication 2026-09-09 14:20:08 UTC

Contents

`agriME-package` *Agricultural Marketing Efficiency and Price Spread Analysis*

Description

Tools for reproducible price-spread, producer-share, marketing-cost, intermediary-margin, and marketing-efficiency analysis using channel totals or stage-level market-chain accounts.

Details

The principal workflows are `marketing_metrics()` for channel totals, `analyse_channels()` for stage-level accounts, and `bootstrap_marketing_metrics()` for repeated observations. All monetary inputs used in one comparison should refer to an equivalent commodity quality, form, place, time, and quantity unit.

Author(s)

Chiranjit Mazumder, Bikramjeet Ghose, and Pramit Pandit.

References

Acharya, S. S. and Agarwal, N. L. (2021). *Agricultural Marketing in India*, 7th edition. Oxford and IBH. ISBN 9789389688061.

Shepherd, A. W. (2007). *A Guide to Marketing Costs and How to Calculate Them*. Food and Agriculture Organization of the United Nations. https://www.fao.org/4/u8770e/u8770e00.htm

See Also

`marketing_metrics`, `analyse_channels`, `bootstrap_marketing_metrics`

as_market_channel *Standardise, Validate, and Analyse Stage-Level Marketing Channels*

Description

Creates an explicit market-channel data contract, detects inconsistent accounts, and calculates actor-level margins plus channel-level price-spread and efficiency statistics.

Usage

```
as_market_channel(
  data,
  channel = "channel",
  stage = "stage",
  actor = "actor",
  actor_type = "actor_type",
  purchase_price = "purchase_price",
  sale_price = "sale_price",
  marketing_cost = "marketing_cost",
  quantity = NULL,
  loss_percent = NULL,
  strict = FALSE
)

validate_market_channel(data, tolerance = 1e-08)

analyse_channel(data, tolerance = 1e-08, strict = FALSE)

analyse_channels(data, tolerance = 1e-08, strict = FALSE)
```

Arguments

data	A data frame containing one row per market actor and channel stage, or a standard market_channel object.
channel, stage, actor, actor_type, purchase_price, sale_price, marketing_cost	Column names in data.
quantity	Optional quantity column name.
loss_percent	Optional physical-loss percentage column name.
strict	If TRUE, broken inter-stage price links stop the operation.
tolerance	Non-negative numerical tolerance for matching prices and accounting identities.

Details

Each channel must contain exactly one producer at the first stage. All later rows must have `actor_type = "intermediary"`. The sale price at one stage should equal the purchase price at the next stage when prices refer to the same product form and quantity. A direct channel may contain only the producer row; the producer sale price is then also the consumer price.

The producer net price is the producer sale price minus the producer marketing cost. Intermediary net margin is sale price minus purchase price minus actor marketing cost.

Value

`as_market_channel()` returns a data frame of class `market_channel`. `validate_market_channel()` returns a diagnostics data frame. The analysis functions return an `agriME_analysis` object with `summary`, `actors`, and `diagnostics` components.

Examples

```
data(tomato_channels)
channel_data <- as_market_channel(tomato_channels)
validate_market_channel(channel_data)

fit <- analyse_channels(channel_data)
fit
summary(fit)
fit$actors
```

---

bootstrap_marketing_metrics

Bootstrap Confidence Intervals for Marketing Metrics

Description

Resamples repeated observations independently within each channel and calculates nonparametric percentile intervals for price-spread, producer-share, and marketing-efficiency indicators.

Usage

```
bootstrap_marketing_metrics(
  data,
  producer_price = "producer_price",
  consumer_price = "consumer_price",
  marketing_cost = "marketing_cost",
  marketing_margin = NULL,
  channel = NULL,
  weight = NULL,
  R = 999L,
  conf = 0.95,
  seed = NULL,
```

```
  na.rm = FALSE,
  shepherd_variant = c("ratio", "net_ratio")
)
```

Arguments

`data`	Observation-level data frame.
`producer_price`, `consumer_price`, `marketing_cost`	Column names or numeric vectors.
`marketing_margin`	Optional column name or numeric vector. If `NULL`, margin is derived from the accounting identity for every row.
`channel`	Optional grouping column name or vector. The default pools all observations.
`weight`	Optional non-negative weight column or vector.
`R`	Integer number of bootstrap replicates, at least 20.
`conf`	Confidence level strictly between zero and one.
`seed`	Optional finite random seed. The previous random-number state is restored when the function exits.
`na.rm`	Remove incomplete observations.
`shepherd_variant`	Shepherd definition passed to `marketing_metrics()`.

Details

Point estimates and bootstrap replicates are calculated from channel means. If weights are supplied, weighted means are used within each resample. Each channel must contain at least two complete observations. Percentile intervals quantify sampling variability under the empirical resampling scheme; they do not correct survey-design bias or establish causal channel effects.

Value

An `agriME_bootstrap` object with a long `summary` table, a wide `replicates` table, the matched call, and the Shepherd definition.

Examples

```
data(market_observations)
boot <- bootstrap_marketing_metrics(
  market_observations,
  marketing_margin = "marketing_margin",
  channel = "channel",
  weight = "volume_qtl",
  R = 49,
  seed = 2026
)
boot
```

consumer_rupee *Decompose the Consumer Price*

Description

Decomposes each unit of consumer expenditure into net producer price, marketing cost, net intermediary margin, and any accounting gap.

Usage

```
consumer_rupee(
  x = NULL,
  producer_price = NULL,
  consumer_price = NULL,
  marketing_cost = NULL,
  marketing_margin = NULL,
  channel = NULL
)
```

Arguments

x	An agriME_analysis object, or a numeric producer price.
producer_price	Alternative named net producer-price input when x is not supplied.
consumer_price	Consumer price.
marketing_cost	Total marketing cost.
marketing_margin	Total net intermediary margin; derived if omitted.
channel	Optional channel labels for direct numeric input.

Value

A long data frame with channel, component, monetary value, and percentage of consumer price.

Examples

```
consumer_rupee(
  producer_price = 1900,
  consumer_price = 3150,
  marketing_cost = 510,
  marketing_margin = 740,
  channel = "Wholesale-retail"
)

data(tomato_channels)
consumer_rupee(analyse_channels(tomato_channels))
```

`loss_adjusted_margin` *Marketing Margin after Physical Product Loss*

Description

Adjusts acquisition cost and trading margin for handling, transport, storage, processing, or spoilage losses.

Usage

```
loss_adjusted_margin(
  purchase_price,
  sale_price,
  quantity_purchased,
  quantity_sold,
  marketing_cost = 0,
  cost_basis = c("lot", "purchased_unit", "sold_unit")
)
```

Arguments

`purchase_price`	Price per purchased unit.
`sale_price`	Price per sold unit.
`quantity_purchased`	Quantity acquired before loss.
`quantity_sold`	Quantity sold after loss.
`marketing_cost`	Marketing cost stated according to `cost_basis`.
`cost_basis`	Whether marketing cost is for the complete lot, each purchased unit, or each sold unit.

Details

The effective purchase cost per sold unit is total purchase value divided by quantity sold. Gross margin is sales value minus purchase value; net margin also deducts total marketing cost. Quantity sold cannot exceed quantity purchased.

Value

A data frame containing quantity loss, loss percentage, purchase and sales values, total marketing cost, effective unit cost, and gross and net margins.

Examples

```
loss_adjusted_margin(
  purchase_price = 20,
  sale_price = 28,
  quantity_purchased = 100,
  quantity_sold = 92,
  marketing_cost = 180,
  cost_basis = "lot"
)
```

---

marketing_efficiency *Agricultural Marketing Efficiency and Complete Channel Metrics*

Description

Calculates alternative marketing-efficiency indices or a comprehensive table of price-spread and efficiency results.

Usage

```
marketing_efficiency(
  producer_price,
  consumer_price,
  marketing_cost,
  marketing_margin = NULL,
  method = c("acharya", "shepherd", "conventional"),
  shepherd_variant = c("ratio", "net_ratio")
)

marketing_metrics(
  producer_price,
  consumer_price,
  marketing_cost,
  marketing_margin = NULL,
  channel = NULL,
  shepherd_variant = c("ratio", "net_ratio")
)
```

Arguments

producer_price	Net price received by the producer per common unit.
consumer_price	Price paid by the final consumer per the same unit.
marketing_cost	Total marketing cost per the same unit.
marketing_margin	Total net intermediary margin. If NULL, it is derived as consumer price minus producer price minus marketing cost.

`method`	One of `"acharya"`, `"shepherd"`, or `"conventional"`.
`shepherd_variant`	`"ratio"` uses consumer price divided by marketing cost. `"net_ratio"` subtracts one from that ratio.
`channel`	Optional channel labels.

Details

The package implements the following accounting definitions:

$$ME_A = P_f/(MC + MM)$$

for Acharya efficiency,

$$ME_S = P_c/MC$$

for the Shepherd ratio, and

$$ME_C = (P_c - P_f)/MC$$

for conventional efficiency. Here P_f is net producer price, P_c is consumer price, MC is total marketing cost, and MM is total net intermediary margin. The optional net-ratio Shepherd variant is $P_c/MC - 1$.

When the supplied accounts satisfy $P_c = P_f + MC + MM$, Acharya efficiency is also equal to $P_c/(MC + MM) - 1$. Any departure is retained in the `accounting_gap` column.

Value

`marketing_efficiency()` returns a numeric vector. `marketing_metrics()` returns a data frame containing all price, spread, share, accounting, and efficiency measures.

References

Acharya, S. S. and Agarwal, N. L. (2021). *Agricultural Marketing in India*, 7th edition. Oxford and IBH. ISBN 9789389688061.

Examples

```
marketing_efficiency(1900, 3150, 510, 740, method = "acharya")
marketing_efficiency(1900, 3150, 510, method = "shepherd")

marketing_metrics(
  producer_price = c(2420, 1900),
  consumer_price = c(2600, 3150),
  marketing_cost = c(180, 510),
  marketing_margin = c(0, 740),
  channel = c("Direct", "Wholesale-retail")
)
```

`marketing_margin` *Gross and Net Margin of a Market Intermediary*

Description

Separates the intermediary's gross price margin from net margin after marketing cost.

Usage

```
marketing_margin(
  purchase_price,
  sale_price,
  marketing_cost = 0,
  consumer_price = NULL
)
```

Arguments

`purchase_price`	Purchase price per unit.
`sale_price`	Sale price per unit.
`marketing_cost`	Marketing cost incurred per unit.
`consumer_price`	Optional final consumer price used to calculate gross and net margin shares.

Details

Gross margin is sale price minus purchase price. Net margin is gross margin minus marketing cost. Markup is measured relative to purchase price; an undefined markup caused by a zero purchase price is returned as `NA`.

Value

A data frame containing purchase price, sale price, marketing cost, gross margin, net margin, markup percentage, and optional consumer-price shares.

Examples

```
marketing_margin(
  purchase_price = c(2050, 2450),
  sale_price = c(2450, 3150),
  marketing_cost = c(140, 220),
  consumer_price = 3150
)
```

`marketing_sensitivity` *Marketing Sensitivity and Efficiency-Target Analysis*

Description

Evaluates a full factorial set of price, cost, and margin shocks, or solves an efficiency equation for the value required to attain a target.

Usage

```
marketing_sensitivity(
  producer_price,
  consumer_price,
  marketing_cost,
  marketing_margin = NULL,
  producer_change = c(-0.1, 0, 0.1),
  consumer_change = 0,
  cost_change = c(-0.1, 0, 0.1),
  margin_change = 0,
  shepherd_variant = c("ratio", "net_ratio")
)

efficiency_target(
  target,
  method = c("acharya", "shepherd", "conventional"),
  solve_for = c(
    "producer_price",
    "consumer_price",
    "marketing_cost",
    "marketing_margin"
  ),
  producer_price = NULL,
  consumer_price = NULL,
  marketing_cost = NULL,
  marketing_margin = NULL,
  shepherd_variant = c("ratio", "net_ratio")
)
```

Arguments

`producer_price`, `consumer_price`, `marketing_cost`, `marketing_margin`
: Baseline or known accounting values. An omitted margin in sensitivity analysis is derived by identity.

`producer_change`, `consumer_change`, `cost_change`, `margin_change`
: Vectors of proportional changes; `0.10` means a ten percent increase.

`shepherd_variant`
: One of `"ratio"` or `"net_ratio"`.

target	Desired positive efficiency ratio.
method	Efficiency equation to solve.
solve_for	Unknown component to calculate.

Details

Sensitivity scenarios change each accounting component independently. A non-zero accounting gap can therefore be informative: it shows that the chosen scenario is not a closed price identity unless the related components are also adjusted.

Acharya efficiency can solve for producer price, marketing cost, or marketing margin. Shepherd efficiency can solve for consumer price or marketing cost. Conventional efficiency can solve for producer price, consumer price, or marketing cost.

Value

`marketing_sensitivity()` returns an `agriME_sensitivity` data frame. `efficiency_target()` returns a one-row data frame containing the required value.

Examples

```
sens <- marketing_sensitivity(
  1900, 3150, 510, 740,
  producer_change = c(0, 0.05),
  cost_change = c(-0.1, 0, 0.1)
)
sens

efficiency_target(
  target = 2,
  method = "acharya",
  solve_for = "marketing_cost",
  producer_price = 1900,
  marketing_margin = 740
)
```

---

market_observations *Repeated Illustrative Agricultural Market Observations*

Description

Synthetic weekly channel-level observations designed for bootstrap examples. The accounting fields obey the price identity. These data are not empirical survey results.

Usage

```
market_observations
```

Format

A data frame with 72 rows and 10 variables:

date Weekly observation date.

market Illustrative market name.

commodity Commodity name.

unit Price unit.

channel Marketing channel.

producer_price Net producer price in Indian rupees per quintal.

consumer_price Final consumer price in Indian rupees per quintal.

marketing_cost Total channel marketing cost.

marketing_margin Total net intermediary margin.

volume_qtl Illustrative transaction volume in quintals.

Source

Synthetic data created by the package authors.

Examples

```
data(market_observations)
head(market_observations)
```

---

plot.agriME_analysis *Methods for agriME Result Objects*

Description

Print, summary, conversion, and base-graphics methods for package result objects.

Usage

```
## S3 method for class 'agriME_analysis'
plot(
  x,
  type = c("decomposition", "efficiency", "producer_share"),
  col = NULL,
  main = NULL,
  ...
)

## S3 method for class 'agriME_bootstrap'
plot(
  x,
  metric = "acharya_efficiency",
```

```
  col = "#2C7FB8",
  main = NULL,
  ...
)

## S3 method for class 'agriME_ranking'
plot(x, col = "#756BB1", main = NULL, ...)

## S3 method for class 'agriME_sensitivity'
plot(
  x,
  metric = "acharya_efficiency",
  col = NULL,
  main = NULL,
  ...
)

## S3 method for class 'agriME_analysis'
print(x, digits = 3L, ...)

## S3 method for class 'agriME_bootstrap'
print(x, digits = 3L, ...)

## S3 method for class 'agriME_ranking'
print(x, digits = 3L, ...)

## S3 method for class 'agriME_sensitivity'
print(x, digits = 3L, ...)

## S3 method for class 'agriME_analysis'
summary(object, ...)

## S3 method for class 'agriME_analysis'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  ...
)
```

Arguments

x, object	An object created by an agriME analysis function.
type	Analysis plot type.
metric	Metric name to plot.
col	Optional colour or colour vector.
main	Optional plot title.

digits	Number of decimal places printed.
row.names, optional	Arguments passed to as.data.frame().
...	Additional arguments passed to the underlying method.

Value

Plot methods return plotted values invisibly. Print methods return x invisibly. The summary and data-frame methods return the channel summary table.

Examples

```
data(tomato_channels)
fit <- analyse_channels(tomato_channels)
summary(fit)
as.data.frame(fit)

ranking <- rank_channels(fit)
ranking


plot(fit, type = "decomposition")
plot(ranking)
```

---

price_spread *Price Spread, Producer Share, and Total Gross Marketing Margin*

Description

Calculates three closely related farm-retail price indicators from equivalent producer and consumer prices.

Usage

```
price_spread(producer_price, consumer_price, percent = FALSE)

producers_share(producer_price, consumer_price)

total_gross_marketing_margin(
  producer_price,
  consumer_price,
  percent = FALSE
)
```

Arguments

producer_price	Net price received by the producer per common unit.
consumer_price	Price paid by the final consumer per the same unit.
percent	Logical; express the result as a percentage of the consumer price.

Details

The absolute price spread is

$$PS = P_c - P_f,$$

where P_c is consumer price and P_f is the net producer price. The percentage price spread is $100PS/P_c$, while producer share is $100P_f/P_c$. With a net producer price, the total gross marketing margin is numerically equal to the price spread.

Value

A numeric vector. Producer share is always returned as a percentage.

Examples

```
price_spread(1900, 3150)
price_spread(1900, 3150, percent = TRUE)
producers_share(1900, 3150)
total_gross_marketing_margin(1900, 3150)
```

---

rank_channels *Rank Alternative Agricultural Marketing Channels*

Description

Uses transparent min-max normalisation and a weighted additive score to compare channels on multiple marketing outcomes.

Usage

```
rank_channels(
  x,
  indicators = c(
    "acharya_efficiency",
    "producer_share_percent",
    "price_spread_percent"
  ),
  directions = c("max", "max", "min"),
  weights = NULL,
  channel_col = "channel"
)
```

Arguments

x	An agriME_analysis object or channel-level data frame.
indicators	Names of numeric indicator columns.
directions	One "max" or "min" value per indicator.
weights	Optional non-negative weights. Equal weights are used by default and supplied weights are rescaled to sum to one.
channel_col	Channel identifier column when x is a data frame.

Details

A constant indicator receives a normalised score of one for every channel, so it does not create artificial differences. Rankings are descriptive and depend on the selected indicators, directions, and value judgements embodied in the weights.

Value

An agriME_ranking data frame containing original indicators, normalised indicators, composite score, and rank. Weights and directions are stored as attributes.

Examples

```
data(tomato_channels)
fit <- analyse_channels(tomato_channels)
rank_channels(fit)
rank_channels(fit, weights = c(0.5, 0.3, 0.2))
```

---

tomato_channels *Illustrative Tomato Marketing Channels*

Description

A synthetic and internally consistent stage-level dataset representing four tomato marketing channels. It is intended for teaching, examples, and software validation, not as empirical survey evidence.

Usage

```
tomato_channels
```

Format

A data frame with 10 rows and 9 variables:

channel Channel label.

stage Ordered stage number.

actor Market participant.

actor_type Producer or intermediary.

purchase_price Purchase price in Indian rupees per quintal.

sale_price Sale price in Indian rupees per quintal.

marketing_cost Marketing cost in Indian rupees per quintal.

quantity_kg Reference quantity in kilograms.

loss_percent Illustrative physical loss percentage.

Source

Synthetic data created by the package authors.

Examples

```
data(tomato_channels)
tomato_channels
```

Index