Package: fpoValue (via r-universe)

October 24, 2026

Type Package

Title Farmer Producer Organisation Economics and Member Benefit Accounting

Version 0.1.0

Description Tools to determine when collective marketing, aggregation, grading and bulk input procurement through farmer producer organisations (FPOs) and agricultural cooperatives improve member-farmer returns. The package provides channel-level transaction cost accounting (transport, commissions, market fees, time, rejection and payment-delay costs), estimation of scale economies from consolidated transport, tiered bulk input discounts and grading premiums, break-even throughput and procurement-price analysis, and comparison of individual and collective marketing scenarios. Its core is a member-level distributional accounting framework that separates the total surplus generated by an FPO from the benefits actually received by farmers through prices, transaction cost savings, input savings and dividends, and summarises their distribution across members with Gini and concentration indices and farm-size group shares.

License GPL (>= 3)

Encoding UTF-8

Language en-GB

Depends R (>= 3.5.0)

Imports graphics

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.1

NeedsCompilation no

Author Anbukkani Perumal [aut, cre, cph]

Maintainer Anbukkani Perumal <anbueconomic@gmail.com>

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

Date/Publication 2026-10-24 11:00:15 UTC

RemoteUrl https://github.com/cran/fpoValue

RemoteRef HEAD

RemoteSha 52bbcc7d2804efe55d56cf6d3ed7760e5c557f5d

Contents

aggregation_savings 2
break_even_volume 4
collective_marketing_scenarios 5
member_benefit 6
plot.collective_marketing_scenarios 9
transaction_costs 9

Index **12**

`aggregation_savings` *Scale economies from aggregation*

Description

Estimates the gross savings an FPO can generate by aggregating members' produce and purchases, from three sources:

Transport consolidation Without the FPO, each member carries produce to the market in small vehicles. With the FPO, members carry produce to a nearby collection centre and the FPO moves the pooled volume in large vehicles. Trip-based costs make the per-unit cost of both legs depend on the quantity carried.

Bulk input procurement A tiered discount schedule is applied to each member's own input purchase and to the pooled purchase.

Grading Selling sorted grades instead of an ungraded lot earns a price premium per unit equal to the grade-weighted price minus the ungraded price, less any grading cost.

Savings are reported in total, per unit and per member. Transport savings are member-specific; the bulk leg is allocated in proportion to quantity, as are grading premiums. These are *potential* savings: whether members receive them depends on the FPO's pricing and dividend policy, which [member_benefit()] evaluates.

Usage

```
aggregation_savings(
  quantity,
  distance_market,
  distance_centre = 0,
  centre_to_market = NULL,
```

```
  small_capacity = 1,
  small_cost_per_km = 0,
  small_fixed_trip = 0,
  bulk_capacity = 10,
  bulk_cost_per_km = 0,
  bulk_fixed_trip = 0,
  input_quantity = 0,
  input_price = 0,
  discount_tiers = NULL,
  grade_shares = NULL,
  grade_prices = NULL,
  ungraded_price = NULL,
  grading_cost = 0
)
```

Arguments

`quantity` Quantity marketed by each member.

`distance_market` One-way distance (km) from each farm to the market.

`distance_centre` One-way distance (km) from each farm to the collection centre.

`centre_to_market` One-way distance (km) from the collection centre to the market.

`small_capacity`, `small_cost_per_km`, `small_fixed_trip` Capacity, cost per km and fixed cost per trip of the small vehicles used by members.

`bulk_capacity`, `bulk_cost_per_km`, `bulk_fixed_trip` Capacity, cost per km and fixed cost per trip of the FPO's bulk vehicles.

`input_quantity` Quantity of inputs bought by each member.

`input_price` Undiscounted input price per unit.

`discount_tiers` Optional data frame with columns 'min_quantity' and 'discount' (share off 'input_price') defining a bulk discount schedule.

`grade_shares`, `grade_prices` Optional vectors giving the share of produce in each grade and the price of each grade.

`ungraded_price` Price of an ungraded lot.

`grading_cost` Cost of grading per unit.

Value

A list of class 'aggregation_savings' with 'by_source' (total and per-unit savings for transport, inputs and grading), 'by_member' (savings per member) and 'total'.

Examples

```
q <- c(4, 8, 15, 25, 60)
aggregation_savings(q, distance_market = 30, distance_centre = 3,
                    centre_to_market = 30, small_capacity = 5,
                    small_cost_per_km = 20, bulk_capacity = 100,
                    bulk_cost_per_km = 45,
                    input_quantity = c(2, 3, 5, 8, 20), input_price = 1300,
                    discount_tiers = data.frame(min_quantity = c(10, 30),
                                                discount = c(0.03, 0.06)),
                    grade_shares = c(0.5, 0.3, 0.2),
                    grade_prices = c(2000, 1600, 1100),
                    ungraded_price = 1550, grading_cost = 30)
```

---

break_even_volume *Break-even throughput, membership and procurement price*

Description

Finds the volume an FPO must handle to cover its fixed costs (and an optional surplus target) given the per-unit margin between its sale price and the price it pays members:

$$Q^* = \frac{F + T - M_{inputs}}{p^{sale} - p^m - h},$$

where F is fixed cost, T the surplus target and M_{inputs} the margin from the input business. It also reports the number of members needed at the average member volume, the margin of safety at the current volume, and the highest member price the FPO can pay at the current volume while breaking even, which bounds how much of the aggregation gain can be passed to farmers through prices.

Usage

```
break_even_volume(
  fixed_cost,
  sale_price,
  member_price,
  handling_cost = 0,
  input_margin = 0,
  target_surplus = 0,
  avg_member_volume = NULL,
  current_volume = NULL
)
```

Arguments

`fixed_cost`	FPO fixed operating cost for the period.
`sale_price`	Price the FPO realises per unit.
`member_price`	Price the FPO pays members per unit.

`handling_cost` FPO handling cost per unit.

`input_margin` Surplus from the input business for the period.

`target_surplus` Surplus the FPO aims to retain (for reserves or dividends).

`avg_member_volume` Optional average volume per member.

`current_volume` Optional current throughput.

Details

All arguments are vectorised and recycled, so a grid of scenarios can be evaluated in one call.

Value

A data frame of class ‘break_even_volume‘ with the inputs and ‘unit_margin‘, ‘break_even_volume‘, ‘members_needed‘, ‘margin_of_safety‘ (share by which current volume exceeds break-even) and ‘max_member_price‘ (at the current volume). Break-even volume is ‘Inf‘ when the unit margin is not positive and the costs are not already covered.

Examples

```
break_even_volume(fixed_cost = 6e5, sale_price = 1650,
                  member_price = c(1450, 1500, 1550),
                  handling_cost = 60, input_margin = 1e5,
                  avg_member_volume = 15, current_volume = 6000)
```

`collective_marketing_scenarios` *Compare individual and collective marketing scenarios*

Description

Evaluates the same set of members under alternative marketing channels, for example individual sale to a village trader, individual sale in a wholesale market, FPO aggregation, FPO aggregation with grading, and an FPO contract with a processor or retailer. Each scenario is described by one row of ‘scenarios‘:

scenario Name.

member_price Price received by members.

member_cost Members’ transaction cost per unit.

sale_price Price realised by the FPO; ‘NA‘ for individual channels.

handling_cost, fixed_cost, dividend_share Optional FPO parameters (default 0).

Member net return is $q_i(p^m - c^m) + D_i$. For FPO channels the FPO surplus and patronage dividends are computed as in [member_benefit()]. Each scenario is compared with the ‘baseline‘ scenario: the gain in members’ net returns, the FPO’s retained surplus, the total value created relative to the baseline, the member share of that value, the share of members better off, and the gain received by the smallest half of suppliers.

Usage

```
collective_marketing_scenarios(
  quantity,
  scenarios,
  baseline = NULL,
  group = NULL
)
```

Arguments

quantity	Quantity marketed by each member.
scenarios	Data frame of scenarios (see Details).
baseline	Name of the baseline scenario; defaults to the first row.
group	Optional group of each member for group-wise gains.

Value

A list of class ‘collective_marketing_scenarios‘ with ‘summary‘ (one row per scenario), ‘member_returns‘ (matrix of member net returns, members by scenarios) and, if ‘group‘ is given, ‘group_gain‘ (gain per member by group and scenario). A ‘plot()‘ method compares member gains and retained surplus across scenarios.

Examples

```
q <- c(5, 8, 12, 20, 35, 60)
sc <- data.frame(
  scenario = c("village trader", "mandi", "FPO pooled", "FPO graded"),
  member_price = c(1300, 1450, 1480, 1540),
  member_cost = c(10, 140, 40, 55),
  sale_price = c(NA, NA, 1650, 1760),
  handling_cost = c(0, 0, 60, 90),
  fixed_cost = c(0, 0, 8000, 12000),
  dividend_share = c(0, 0, 0.5, 0.5))
res <- collective_marketing_scenarios(q, sc,
         group = c("marginal", "marginal", "small", "small",
                   "medium", "large"))
res
plot(res)
```

---

member_benefit *Member-level distributional accounting of FPO benefits*

Description

Separates the total surplus generated by a farmer producer organisation (FPO) from the benefits actually received by member farmers.

Usage

```
member_benefit(
  quantity,
  outside_price,
  member_price,
  sale_price,
  outside_cost = 0,
  member_cost = 0,
  handling_cost = 0,
  fixed_cost = 0,
  input_quantity = 0,
  input_market_price = 0,
  input_member_price = 0,
  input_bulk_price = 0,
  dividend_share = 0,
  dividend_rule = c("patronage", "equity", "equal"),
  equity = NULL,
  group = NULL,
  member_id = NULL
)
```

Arguments

`quantity`	Quantity each member sells through the FPO.
`outside_price`	Price each member would receive outside the FPO.
`member_price`	Price the FPO pays each member.
`sale_price`	Price the FPO realises when selling the pooled produce.
`outside_cost`	Member transaction cost per unit outside the FPO.
`member_cost`	Member transaction cost per unit when delivering to the FPO.
`handling_cost`	FPO handling, grading and transport cost per unit.
`fixed_cost`	FPO fixed operating cost for the period (staff, rent, compliance, depreciation).
`input_quantity`	Quantity of inputs each member buys through the FPO.
`input_market_price`	Price of inputs in the open market.
`input_member_price`	Price the FPO charges members for inputs.
`input_bulk_price`	Price the FPO pays for inputs in bulk.
`dividend_share`	Share of positive FPO surplus paid out as dividends (0 to 1).
`dividend_rule`	How dividends are allocated: ‘"patronage"‘ (in proportion to quantity), ‘"equity"‘ (in proportion to ‘equity‘) or ‘"equal"‘ (equal per member).
`equity`	Share capital held by each member; required for ‘dividend_rule = "equity"‘.
`group`	Optional farm-size or social group of each member.
`member_id`	Optional member identifiers.

Details

FPO account. The FPO buys members' produce at 'member_price', sells it at 'sale_price', incurs handling costs per unit and fixed operating costs, and may run an input business buying at 'input_bulk_price' and selling to members at 'input_member_price'. Its net surplus is

$$S = \sum_i q_i(p^{sale} - p_i^m - h) + \sum_i x_i(w^m - w^{bulk}) - F.$$

A share 'dividend_share' of positive surplus is distributed as dividends; the rest is retained.

Member account. Relative to the member's outside option (selling individually at 'outside_price' with transaction cost 'outside_cost' and buying inputs at 'input_market_price'), member i gains

$$B_i = q_i(p_i^m - p_i^{out}) + q_i(c_i^{out} - c_i^m) + x_i(w^{mkt} - w^m) + D_i.$$

Value created is $\sum_i (B_i - D_i) + S$, which equals $\sum_i B_i$ plus retained surplus. The member share of value created shows how much of the FPO's economic contribution actually reaches farmers. Distribution across members is summarised by the Gini coefficient of benefits (when all benefits are non-negative), the concentration index of benefits ranked by quantity marketed (positive values mean benefits are concentrated among larger suppliers), the benefit share of the smallest half of suppliers compared with their quantity share, the share of members made worse off, and, if 'group' is given, benefit shares and benefit per member by group (for example marginal, small, medium and large farmers).

Per-member arguments accept a scalar or one value per member. Per-unit transaction costs can be taken from [transaction_costs()].

Value

A list of class 'member_benefit' with

members Data frame of benefit components per member: 'price_gain', 'cost_saving', 'input_saving', 'dividend' and 'total'.

fpo Named vector: output margin, input margin, fixed cost, net surplus, dividends paid and retained surplus.

value Named vector: value created, total member benefit, retained surplus, member share of value created, member benefit per unit marketed.

distribution List with 'metrics' and 'by_group'.

Examples

```
q <- c(5, 8, 12, 20, 35, 60)
size <- c("marginal", "marginal", "small", "small", "medium", "large")
mb <- member_benefit(q, outside_price = 1400, member_price = 1480,
                     sale_price = 1650, outside_cost = 140,
                     member_cost = 40, handling_cost = 60,
                     fixed_cost = 8000, input_quantity = q / 5,
                     input_market_price = 1350, input_member_price = 1300,
                     input_bulk_price = 1250, dividend_share = 0.5,
                     group = size)
mb
mb$distribution$by_group
```

plot.collective_marketing_scenarios *Plot collective marketing scenarios*

Description

Stacked bars of the gain in member net returns and the FPO's retained surplus, relative to the baseline, for each scenario.

Usage

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

Arguments

x	A [collective_marketing_scenarios()] object.
...	Further arguments passed to [graphics::barplot()].

Value

'x', invisibly.

Examples

```
sc <- data.frame(scenario = c("individual", "FPO"),
                 member_price = c(1400, 1480), member_cost = c(120, 40),
                 sale_price = c(NA, 1650), handling_cost = c(0, 60),
                 fixed_cost = c(0, 5000), dividend_share = c(0, 0.5))
plot(collective_marketing_scenarios(c(5, 10, 20, 40), sc))
```

transaction_costs *Transaction costs of a marketing channel*

Description

Computes, for each farmer, the full cost of selling through one marketing channel. Transport is trip-based, so costs per unit fall with the quantity carried per trip:

$$transport_i = trips_i \times (2\, d_i\, c_{km} + c_{trip}), \quad trips_i = \lceil q_i / capacity \rceil.$$

Other components are commission (a share of sale value), market fees and handling charges per unit, the opportunity cost of the farmer's time per trip, the value of produce rejected or unsold, and the interest cost of delayed payment.

Usage

```
transaction_costs(
  quantity,
  price,
  distance,
  vehicle_capacity = 1,
  cost_per_km = 0,
  fixed_per_trip = 0,
  commission = 0,
  market_fee = 0,
  handling = 0,
  time_hours = 0,
  wage = 0,
  rejection = 0,
  payment_delay = 0,
  interest_rate = 0.12,
  label = NULL
)
```

Arguments

`quantity`	Quantity sold by each farmer.
`price`	Sale price per unit (scalar or one per farmer).
`distance`	One-way distance (km) from farm to point of sale.
`vehicle_capacity`	Quantity carried per trip.
`cost_per_km`	Vehicle cost per km.
`fixed_per_trip`	Fixed cost per trip (loading, tolls, hire minimum).
`commission`	Commission as a share of sale value.
`market_fee`	Market fee or levy per unit.
`handling`	Handling, packing and loading cost per unit.
`time_hours`	Farmer time per trip (travel, waiting, bargaining).
`wage`	Opportunity cost of farmer time per hour.
`rejection`	Share of quantity rejected or sold at zero value.
`payment_delay`	Days between sale and payment.
`interest_rate`	Annual interest rate used to value payment delay.
`label`	Optional channel name stored with the result.

Details

Call the function once for each channel (for example individual sale in a wholesale market and delivery to an FPO collection centre) and pass the resulting ‘per_unit‘ costs to [member_benefit()] or [collective_marketing_scenarios()].

Value

A data frame of class ‘transaction_costs‘ with one row per farmer: ‘quantity‘, ‘trips‘, ‘transport‘, ‘commission‘, ‘fees‘, ‘handling‘, ‘time_cost‘, ‘rejection_cost‘, ‘delay_cost‘, ‘total‘, ‘per_unit‘ and ‘share_of_value‘.

Examples

```
q <- c(5, 12, 30, 80)
mandi <- transaction_costs(q, price = 1500, distance = 35,
                           vehicle_capacity = 10, cost_per_km = 25,
                           commission = 0.06, market_fee = 15,
                           time_hours = 8, wage = 50,
                           payment_delay = 7, label = "mandi")
centre <- transaction_costs(q, price = 1500, distance = 4,
                            vehicle_capacity = 10, cost_per_km = 25,
                            time_hours = 1.5, wage = 50,
                            payment_delay = 10, label = "FPO centre")
data.frame(quantity = q, mandi = mandi$per_unit,
           centre = centre$per_unit)
```

Index

aggregation_savings, 2

break_even_volume, 4

collective_marketing_scenarios, 5

member_benefit, 6

plot.collective_marketing_scenarios, 9

transaction_costs, 9