

Package: grainPlan (via r-universe)

October 11, 2026

Title Grain-Specific Decision Orchestration on Top of Decision Theory

Version 0.2.1

Description Turns an orchestra inference result into an actionable grain-production plan. It is a thin grain-semantics layer on top of the 'decideR' decision engine: it reuses that engine, its grade-band loss library, its manifest-native pipeline tail, and its grounding firewall, and adds the grain economics those decisions kept re-deriving. Named grain decisions price a nitrogen rate by expected profit, rank a variety from a multi-environment trial under downside-aware merit, and target a quality grade against a delivery standard. A reusable, documented grain value library supplies wheat protein bands, barley malting premiums, and nitrogen-rate economics built from published price structures. A season plan composes several grain decisions into one provenance-carrying object whose grounding is the worst case across its decisions, so a plan resting on any unverified input abstains rather than asserting a confident wrong recommendation.

License MIT + file LICENSE

URL <https://github.com/max578/grainPlan>

BugReports <https://github.com/max578/grainPlan/issues>

Encoding UTF-8

Language en-AU

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Depends R (>= 3.5.0)

Imports decideR, S7, stats

Suggests agridat, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

Remotes max578/decideR

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

Repository <https://max578.r-universe.dev>
Date/Publication 2026-10-11 03:36:45 UTC
RemoteUrl <https://github.com/max578/grainPlan>
RemoteRef main
RemoteSha 409791a5e57ddfbed37f2d958653e63f54aa05ba

Contents

barley_malting_premium	2
grain_decision	4
grain_is_grounded	5
grain_plan	6
grain_plan_table	7
n_rate_economics	8
plan_grade_target	9
plan_nitrogen_rate	11
plan_nitrogen_rate_from_manifest	12
plan_season	14
plan_variety	15
plan_variety_from_manifest	17
print.grain_decision	19
print.grain_plan	20
print.n_rate_economics	20
wheat_protein_bands	21

Index 23

barley_malting_premium
<i>Barley malting-premium value schedule</i>

Description

The barley counterpart to [wheat_protein_bands\(\)](#): a price per tonne that carries a malting premium when grain protein falls inside the malting window and drops to the feed price outside it. Malting barley is the textbook two-sided quality payoff – protein that is too low or too high both miss the malting grade – which a single rising step schedule cannot express, so this schedule pays the premium only across the central band.

Usage

```
barley_malting_premium(  
  window = c(9.5, 12.5),  
  feed_price = 280,  
  malting_premium = 60  
)
```

Arguments

window	A length-two numeric vector <code>c(low, high)</code> giving the malting protein window in percent (default <code>c(9.5, 12.5)</code>).
feed_price	The feed price in dollars per tonne paid outside the window (default an illustrative 280).
malting_premium	The premium in dollars per tonne added to the feed price inside the window (default an illustrative 60).

Details

The default window is the broad malting protein window in percent and the default prices are an illustrative feed price and malting premium in dollars per tonne. The window shape (a premium inside an acceptance band, feed outside) follows the published Barley Australia / Grain Trade Australia malting receival conventions; the dollar values are illustrative and not receival prices. Supply current prices for a real decision.

Value

A `decideR::grade_band_value` schedule: a vectorised function from a protein outcome to its dollar-per-tonne payoff, paying `feed_price + malting_premium` inside the window and `feed_price` outside.

References

Barley Australia and Grain Trade Australia malting barley receival standards (the malting protein window). Dollar values are illustrative.

See Also

[wheat_protein_bands\(\)](#) for the wheat counterpart; [plan_grade_target\(\)](#) for the decision that consumes it.

Other grain-value: [n_rate_economics\(\)](#), [wheat_protein_bands\(\)](#)

Examples

```
v <- barley_malting_premium()
v(c(8.0, 10.5, 13.0))
v
```

grain_decision	<i>A grain decision – a recommended grain action with context and grounding</i>
----------------	---

Description

The unit grainPlan recommends: exactly one `decideR::decision` wrapped with the grain context that gives it agronomic meaning. Where a bare decision records "action 90, expected utility 1840, grounded", a `grain_decision` records "apply 90 kg/ha of nitrogen to this wheat crop, grounded, profit \$1840/ha" – the same decision, read in grain terms.

Usage

```
grain_decision(
  decision = NULL,
  crop = NA_character_,
  kind = NA_character_,
  action = NULL,
  unit = NA_character_,
  abstained = FALSE,
  grounding = "ungrounded",
  rationale = NA_character_,
  context = list()
)
```

Arguments

decision	The wrapped <code>decideR::decision</code> object.
crop	The crop the decision concerns (e.g. "wheat", "barley", "corn").
kind	The decision kind, one of "nitrogen_rate", "variety", or "grade_target".
action	The recommended action, lifted from the wrapped decision (the <code>safe_action</code> when the decision abstained).
unit	A short string naming the units of action (e.g. "kg N/ha", "variety", "protein %").
abstained	TRUE when the wrapped decision declined the utility-maximal action.
grounding	A canonical grounding token, copied from the wrapped decision.
rationale	A short human-readable sentence explaining the recommendation, suitable for a grower-facing report.
context	A named list of free-form grain context (prices, the candidate set, the dataset attribution).

Details

The object carries the wrapped decision whole (in `decision`) so nothing `decideR` computed is lost, and lifts the fields a practitioner reads first to the top: the crop, the decision kind, the recommended action and its unit, whether the decision abstained, and the grounding token. The grounding is copied from the wrapped decision, never recomputed, so a `grain_decision` can never claim a stronger provenance than the decision it wraps.

Value

A `grain_decision` S7 object.

See Also

[grain_plan](#) for a season of decisions; [plan_nitrogen_rate\(\)](#), [plan_variety\(\)](#), and [plan_grade_target\(\)](#) for the verbs that build it.

Other grain-objects: [grain_is_grounded\(\)](#), [grain_plan\(\)](#), [grain_plan_table\(\)](#), [plan_season\(\)](#)

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax <- rnorm(800L, 5.5, 0.3)
yld <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
              numeric(800L))
gd <- plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                        crop = "wheat",
                        grounding = decideR::grounding_grounded())
gd@action
```

<code>grain_is_grounded</code>	<i>Is a grain decision or plan grounded?</i>
--------------------------------	--

Description

A convenience predicate reading the grounding label of a [grain_decision](#) or [grain_plan](#). A grower-facing report, or a downstream automation that must refuse to act on unverified evidence, gates on this. It mirrors `decideR::is_grounded()` one level up, reading the grain object's own worst-case grounding token.

Usage

```
grain_is_grounded(x)
```

Arguments

`x` A [grain_decision](#) or [grain_plan](#).

Value

TRUE when the object's grounding is `decideR::grounding_grounded()`.

See Also

Other grain-objects: `grain_decision()`, `grain_plan()`, `grain_plan_table()`, `plan_season()`

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax <- rnorm(800L, 5.5, 0.3)
yld <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
              numeric(800L))
gd <- plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                        crop = "wheat",
                        grounding = decideR::grounding_grounded())
grain_is_grounded(gd)
```

grain_plan	<i>A grain plan – an ordered season of grain decisions with combined grounding</i>
------------	--

Description

The orchestrator's output: a sequence of `grain_decision` objects gathered into one provenance-carrying season plan. The plan's grounding is the worst case across its decisions (via `decideR::combine_grounding()`), so a plan that rests on any unverified decision is itself "ungrounded" – the Independent Oracle Principle applied at the plan level. A plan is `partial_abstention` when at least one of its decisions abstained, which a report surfaces so a grower never reads a confident headline over a plan with a declined component.

Usage

```
grain_plan(
  decisions = list(),
  crop = NA_character_,
  season = NA_character_,
  grounding = "ungrounded",
  partial_abstention = FALSE,
  metadata = list()
)
```

Arguments

decisions	A list of grain_decision objects, in the order they apply through the season.
crop	The crop the plan concerns.
season	A short label for the season or field the plan covers.
grounding	A canonical grounding token, combined worst-case across the decisions.
partial_abstention	TRUE when at least one decision in the plan abstained.
metadata	A named list of free-form plan metadata.

Value

A grain_plan S7 object.

See Also

[plan_season\(\)](#) for the constructor; [grain_decision](#) for the unit.

Other grain-objects: [grain_decision\(\)](#), [grain_is_grounded\(\)](#), [grain_plan_table\(\)](#), [plan_season\(\)](#)

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax <- rnorm(800L, 5.5, 0.3)
yld <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
              numeric(800L))
n_dec <- plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                           crop = "wheat",
                           grounding = decideR::grounding_grounded())
plan_season(list(n_dec), crop = "wheat", season = "2026 paddock 7")
```

grain_plan_table	<i>Summarise a grain plan as a data frame</i>
------------------	---

Description

Flatten a [grain_plan](#) into a one-row-per-decision data.frame for a report or a console table: the decision kind, the recommended action and its unit, whether it abstained, and its grounding token. The plan's worst-case grounding and partial-abstention flag are carried as attributes so a report header can read them without re-deriving them.

Usage

```
grain_plan_table(plan)
```

Arguments

plan	A grain_plan .
------	--------------------------------

Value

A data.frame with one row per decision and columns kind, action, unit, abstained, grounding, plus the attributes plan_grounding and partial_abstention.

See Also

[plan_season\(\)](#) for the plan; [grain_plan](#) for the object.

Other grain-objects: [grain_decision\(\)](#), [grain_is_grounded\(\)](#), [grain_plan\(\)](#), [plan_season\(\)](#)

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax <- rnorm(800L, 5.5, 0.3)
yld <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
              numeric(800L))
n_dec <- plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                           crop = "wheat",
                           grounding = decideR::grounding_grounded())
grain_plan_table(plan_season(list(n_dec), crop = "wheat"))
```

n_rate_economics	<i>Nitrogen-rate economics</i>
------------------	--------------------------------

Description

The reusable economic bundle a nitrogen-rate decision prices against: the grain price, the nitrogen price, and the resulting per-draw profit utility $p_y y - p_n r$. This is the second hand-built economic that the orchestra runs kept re-deriving inline; `n_rate_economics()` consolidates it into a named, documented object holding both the prices and the utility function the decision engine consumes, so a profit-priced rate decision reads as its economics rather than as an anonymous lambda.

Usage

```
n_rate_economics(price_grain, price_n)
```

Arguments

price_grain	The grain price per unit yield (e.g. dollars per tonne).
price_n	The nitrogen price per unit rate (e.g. dollars per kilogram of nitrogen).

Details

The utility is the textbook nitrogen-response economics: revenue is the grain price times yield, the cost is the nitrogen price times the rate, and profit is their difference. Yield and rate are in whatever units the caller's yield draws and rate grid use (tonnes per hectare and kilograms of nitrogen per hectare in the common case); the prices must be in matching units (dollars per tonne and dollars per kilogram of nitrogen).

Value

A list with class `n_rate_economics` holding `price_grain`, `price_n`, and `utility`, where `utility(rate, yield)` returns the profit per draw – the function form `decideR::decide()` and `decideR::decide_rate_from_manifest()` consume.

References

Standard nitrogen-response profit economics (revenue less nitrogen cost); the per-tonne / per-kilogram convention follows the broad Australian grains gross-margin framework.

See Also

[plan_nitrogen_rate\(\)](#) for the decision that consumes it.

Other grain-value: [barley_malting_premium\(\)](#), [wheat_protein_bands\(\)](#)

Examples

```
e <- n_rate_economics(price_grain = 350, price_n = 1.3)
e$utility(rate = 80, yield = c(4.0, 4.5, 5.0))
e
```

plan_grade_target	<i>Plan a grade target under a quality-band payoff</i>
-------------------	--

Description

Choose how hard to push a quality-driving input when the grain payoff steps at a delivery-standard threshold (a wheat protein band, a barley malting window) rather than rising smoothly. It is a grain-semantics wrapper over `decideR::decide()` driven by a grade-band loss: the value schedule prices each quality band, the utility is the band payoff of the shifted quality outcome less the input cost, and the expected utility `decideR` averages over the draws is governed by each band's posterior probability – exactly the threshold economics a quality-graded decision faces. It abstains to the status-quo action when the quality evidence is unverified or no action is clearly better.

Usage

```
plan_grade_target(
  quality_draws,
  actions,
  value,
  quality_shift = NULL,
  cost = 0,
  crop = "wheat",
  grounding = decideR::grounding_unverified(),
  safe_action = NULL,
  decisive_prob = 0.6,
  ...
)
```

Arguments

quality_draws	A numeric vector of posterior draws of the baseline quality outcome under the status-quo action.
actions	A numeric vector of candidate input levels (the grid the decision chooses over).
value	A <code>decideR::grade_band_value</code> schedule pricing the quality bands – typically <code>wheat_protein_bands()</code> or <code>barley_malting_premium()</code> .
quality_shift	A function <code>function(action, quality)</code> returning the realised quality under the action, or NULL for the additive default quality + action.
cost	A per-unit input cost (a single number applied as <code>cost * action</code>), a function <code>function(action)</code> , or 0 for no cost.
crop	The crop the decision concerns (default "wheat").
grounding	One or more grounding tokens for the quality evidence; combined worst-case.
safe_action	The action returned on abstention (default the first element of actions, the status quo).
decisive_prob	Minimum posterior probability that the leading action beats the status quo (passed to <code>decideR::decide()</code>).
...	Further arguments passed to the decision core (e.g. <code>min_ess</code>).

Details

`quality_draws` is a numeric vector of posterior draws of the baseline quality outcome (grain protein, say) under no extra input. `actions` is the candidate input grid (a nitrogen-rate grid, say). `quality_shift(action, quality)` returns the realised quality under an action – the additive default quality + action suits a unit that lifts quality one-for-one, and a supplied function suits a saturating or partial response. `value` is a grade-band schedule from `grainPlan`'s value library (or any `decideR::grade_band_value()`), and `cost` prices the input.

Value

A `grain_decision` of kind "grade_target" whose action is the recommended input level.

See Also

`wheat_protein_bands()` and `barley_malting_premium()` for the value schedules; `decideR::grade_band_utility()` for the loss; `decideR::decide()` for the engine.

Other grain-decisions: `plan_nitrogen_rate()`, `plan_nitrogen_rate_from_manifest()`, `plan_variety()`, `plan_variety_from_manifest()`

Examples

```
set.seed(1L)
# Baseline protein draws; a nitrogen top-up lifts protein additively.
protein <- rnorm(2000L, mean = 11.0, sd = 0.6)
v <- wheat_protein_bands()
plan_grade_target(protein, actions = seq(0, 60, by = 10), value = v,
  quality_shift = function(rate, p) p + 0.02 * rate,
  cost = 1.3, crop = "wheat",
  grounding = decideR::grounding_grounded())
```

plan_nitrogen_rate	<i>Plan a nitrogen rate for a grain crop</i>
--------------------	--

Description

The canonical grain decision: choose how much nitrogen to apply to a crop by expected profit, abstaining to the status-quo rate when the yield evidence is unverified or no rate is clearly better. It is a grain-semantics wrapper over `decideR::decide_input_rate()` – the profit maximisation, the grounding firewall, and the abstention logic are `decideR`’s; `grainPlan` supplies the grain economics (`n_rate_economics()`), the crop context, a grower-facing rationale, and the [grain_decision](#) result.

Usage

```
plan_nitrogen_rate(
  yield_draws,
  rates,
  price_grain,
  price_n,
  crop = "wheat",
  grounding = decideR::grounding_unverified(),
  safe_rate = 0,
  decisive_prob = 0.6,
  ...
)
```

Arguments

yield_draws	A numeric matrix of posterior yield draws, one column per candidate rate (rows are draws).
rates	A numeric vector of candidate nitrogen rates, one per column of <code>yield_draws</code> .
price_grain	The grain price per unit yield (passed to n_rate_economics()).
price_n	The nitrogen price per unit rate (passed to n_rate_economics()).
crop	The crop the decision concerns (default "wheat").
grounding	One or more grounding tokens for the yield evidence; combined worst-case and passed to the decision engine.
safe_rate	The fallback rate on abstention (default 0, the no-input status quo).
decisive_prob	Minimum posterior probability that the leading rate beats the status quo (passed to <code>decideR::decide()</code>).
...	Further arguments passed to the decision core (e.g. <code>min_ess</code>). <code>constraint</code> is rejected with an error: <code>decideR::decide_input_rate()</code> does not honour a rate constraint and would otherwise drop it silently (use plan_nitrogen_rate_from_manifest() , which does honour one, or cap rates before calling).

Details

Pass either a yield-draws matrix (yield_draws, one column per candidate rate, rows are posterior draws – the object a PESTO inversion or a mechanistic yield-response posterior produces) together with rates, or an upstream orchestra_manifest to `plan_nitrogen_rate_from_manifest()`, which sources the draws and the grounding token straight from the manifest. Either way the grounding rides from the evidence through the decision: an unverified yield posterior forces the status-quo rate, never a confident wrong recommendation.

Value

A `grain_decision` of kind "nitrogen_rate" whose action is the recommended rate.

See Also

`plan_nitrogen_rate_from_manifest()` for the manifest entry point; `n_rate_economics()` for the economics; `decideR::decide_input_rate()` for the underlying engine.

Other grain-decisions: `plan_grade_target()`, `plan_nitrogen_rate_from_manifest()`, `plan_variety()`, `plan_variety_from_manifest()`

Examples

```
set.seed(1L)
rates <- seq(0, 200, by = 25)
# diminishing-returns yield response with posterior noise
mu <- 3 + 2.5 * (1 - exp(-rates / 80))
yield_draws <- vapply(mu, function(m) rnorm(1500L, m, 0.3), numeric(1500L))
plan_nitrogen_rate(yield_draws, rates, price_grain = 350, price_n = 1.3,
                  crop = "wheat",
                  grounding = decideR::grounding_grounded())
```

plan_nitrogen_rate_from_manifest

Plan a nitrogen rate from an upstream orchestra manifest

Description

The manifest entry point to the nitrogen-rate decision: source the yield draws and the producer's grounding token straight from an orchestra_manifest emitted upstream (a PESTO inversion, a mechanistic yield-response posterior), price them, and return a `grain_decision`. It is the formalised crop-pipeline tail at the grain layer – a yield posterior in, a profit-priced, grounding-aware nitrogen-rate plan out – built directly on `decideR::decide_rate_from_manifest()`, which reads the manifest by its public contract (the draws matrix and the grounding token) without grainPlan depending on the contract package.

Usage

```
plan_nitrogen_rate_from_manifest(
  manifest,
  rates,
  price_grain,
  price_n,
  crop = "wheat",
  grounding = NULL,
  safe_rate = 0,
  decisive_prob = 0.6,
  ...
)
```

Arguments

manifest	An orchestra_manifest carrying a yield-per-rate draws matrix (in outputs or metadata\$yield_draws), one column per rate.
rates	A numeric vector of candidate nitrogen rates, one per column of the manifest's draws matrix.
price_grain	The grain price per unit yield.
price_n	The nitrogen price per unit rate.
crop	The crop the decision concerns (default "wheat").
grounding	Optional override for the manifest's own grounding token; NULL (default) uses the token the manifest carries, read worst-case.
safe_rate	The fallback rate on abstention (default 0).
decisive_prob	Minimum posterior probability that the leading rate beats the status quo.
...	Further arguments passed to the decision core (e.g. min_ess, draws_key).

Details

The grounding rides from the producer through the decision: an unverified yield manifest forces the status-quo rate. grainPlan takes no dependency on the manifest constructor; the manifest is read by decideR's duck-typed tail. A caller who has independently grounded (or wants to force "ungrounded" on) a manifest can override the producer's token through grounding, matching [plan_variety_from_manifest\(\)](#).

Value

A [grain_decision](#) of kind "nitrogen_rate" whose action is the recommended rate. Held at safe_rate, unpriced, when the manifest's own producer declared an abstention (summary\$abstained), or when it declares an inferential_target other than "predictions" or "parameters".

See Also

[plan_nitrogen_rate\(\)](#) for the caller-assembled-matrix form; [decideR::decide_rate_from_manifest\(\)](#) for the underlying tail.

Other grain-decisions: [plan_grade_target\(\)](#), [plan_nitrogen_rate\(\)](#), [plan_variety\(\)](#), [plan_variety_from_manifest\(\)](#)

Examples

```
## Not run:
# `manifest` is an orchestra_manifest S7 object emitted upstream (a PESTO
# inversion or a mechanistic yield-response posterior), carrying a yield-per-rate
# draws matrix in its `outputs` slot (or `metadata$yield_draws`) and a
# grounding token in `metadata$grounding`. decideR's tail reads it by its
# public S7 contract, so grainPlan needs no dependency on the contract package.
rates <- seq(0, 200, by = 25)
plan_nitrogen_rate_from_manifest(manifest, rates,
                                price_grain = 350, price_n = 1.3)

## End(Not run)
```

plan_season

Compose grain decisions into a season plan

Description

Gather an ordered list of [grain_decision](#) objects into one [grain_plan](#) – a single provenance-carrying recommendation for a season or field. The plan's grounding is the worst case across its decisions (via `decideR::combine_grounding()`), so a plan that rests on any unverified decision is itself unverified: the Independent Oracle Principle applied one level up. The plan is flagged `partial_abstention` when at least one decision abstained, which a grower-facing report surfaces so a confident headline is never read over a plan with a declined component.

Usage

```
plan_season(decisions, crop = NULL, season = "season", metadata = list())
```

Arguments

<code>decisions</code>	A list of grain_decision objects, in the order they apply.
<code>crop</code>	The crop the plan concerns; defaults to the crop of the first decision.
<code>season</code>	A short label for the season or field (default "season").
<code>metadata</code>	A named list of free-form plan metadata.

Details

The decisions are stored in the order given – the order they apply through the season (variety choice at sowing, then a nitrogen rate, then an in-season grade-target top-up, say). v0.1 composes by sequencing and worst-case grounding only; it does not re-optimize across decisions (a deliberately simple, auditable first orchestrator).

Value

A [grain_plan](#) object.

See Also

[grain_plan](#) for the returned object; the `plan_*()` verbs that build the decisions.

Other grain-objects: [grain_decision\(\)](#), [grain_is_grounding\(\)](#), [grain_plan\(\)](#), [grain_plan_table\(\)](#)

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax <- rnorm(800L, 5.5, 0.3)
yld <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
              numeric(800L))
n_dec <- plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                           crop = "wheat",
                           grounding = decideR::grounding_grounding())
varieties <- c("Scepter", "Vixen", "Calibre")
merit <- vapply(c(5.2, 5.4, 4.9), function(m) rnorm(800L, m, 0.25),
              numeric(800L))
v_dec <- plan_variety(merit, varieties, crop = "wheat",
                     grounding = decideR::grounding_grounding())
plan_season(list(v_dec, n_dec), crop = "wheat", season = "2026 paddock 7")
```

plan_variety	<i>Plan a variety choice from a multi-environment trial or genomic posterior</i>
--------------	--

Description

Rank candidate grain varieties under a downside-aware merit loss and recommend the leading one, abstaining when the merit evidence is unverified or no variety is clearly better than the incumbent. It is a grain-semantics wrapper over `decideR::decide()`: the variety indices are the candidate actions, the utility is each variety's posterior merit less a one-sided downside penalty (`risk_aversion` times the shortfall below the variety's own mean, so a high-yielding but volatile line is discounted against a steady one), and the grounding firewall plus abstention are `decideR`'s.

Usage

```
plan_variety(
  merit_draws,
  varieties = NULL,
  risk_aversion = 0.5,
  incumbent = NULL,
  crop = "wheat",
  grounding = decideR::grounding_unverified(),
  decisive_prob = 0.6,
  ...
)
```

Arguments

merit_draws	A numeric matrix of posterior merit draws, one column per variety (rows are draws).
varieties	A character vector of variety names, one per column of merit_draws; defaults to the column names of merit_draws, then to "v1", "v2",
risk_aversion	A non-negative number weighting the one-sided downside penalty (default 0.5); 0 is plain expected merit.
incumbent	The variety recommended on abstention – the status-quo line a grower keeps when the evidence does not justify a switch; a variety name in varieties, or NULL to default to the first variety.
crop	The crop the decision concerns (default "wheat").
grounding	One or more grounding tokens for the merit evidence; combined worst-case.
decisive_prob	Minimum posterior probability that the leading variety beats the incumbent (passed to decideR::decide()).
...	Further arguments passed to the decision core (e.g. min_ess).

Details

merit_draws is a numeric matrix of posterior merit draws with one column per variety (rows are draws) – the object a factor-analytic MET model emits (a variety's mean across environments) or a genomic-prediction model emits (its genomic estimated breeding value posterior). The recommendation is the single leading variety; the full ranked ledger of every variety's expected merit is carried in the wrapped decision's candidates for a short-list.

The grounding rides from the evidence: an unverified merit posterior forces the recommendation to the incumbent variety (the safe action), so a variety is never switched on evidence the producer did not ground.

Value

A [grain_decision](#) of kind "variety" whose action is the recommended variety name; the ranked short-list is in decision@candidates of the wrapped decision.

See Also

[plan_variety_from_manifest\(\)](#) for a breeding_values manifest; [decideR::decide\(\)](#) for the underlying engine.

Other grain-decisions: [plan_grade_target\(\)](#), [plan_nitrogen_rate\(\)](#), [plan_nitrogen_rate_from_manifest\(\)](#), [plan_variety_from_manifest\(\)](#)

Examples

```
set.seed(1L)
varieties <- c("Scepter", "Vixen", "Calibre", "Denison")
means <- c(5.2, 5.4, 4.9, 5.3)
merit_draws <- vapply(means, function(m) rnorm(1000L, m, 0.25),
                      numeric(1000L))
plan_variety(merit_draws, varieties, crop = "wheat",
             grounding = decideR::grounding_grounded())
```

plan_variety_from_manifest

Plan a variety choice from a breeding-values manifest

Description

The genomic entry point to the variety decision: read genomic estimated breeding values (GEBVs) straight from a `breeding_values_orchestra_manifest` – the manifest flexyBayes’s genomics targets emit (`genomic_summary()` lifted through the composition layer) – and rank the genotypes under the same downside-aware merit loss `plan_variety()` uses. This closes the genomics-to-decision loop: a GBLUP posterior in, a grounding-aware genotype recommendation out.

Usage

```
plan_variety_from_manifest(
  manifest,
  genotypes = NULL,
  risk_aversion = 0.5,
  incumbent = NULL,
  gebv_sd = NULL,
  n_draws = 1000L,
  crop = "wheat",
  grounding = NULL,
  decisive_prob = 0.6,
  ...
)
```

Arguments

manifest	A <code>breeding_values_orchestra_manifest</code> (or any S7 object exposing outputs and metadata with a <code>gebv</code> payload).
genotypes	A character vector of genotype names; defaults to the names on the GEBV payload, then to "g1", "g2",
risk_aversion	A non-negative downside-penalty weight (default 0.5).
incumbent	The genotype recommended on abstention, or NULL for the first.
gebv_sd	Optional per-genotype GEBV standard deviation used to reconstruct a posterior from point GEBVs (a single number or one per genotype). When NULL (default) the uncertainty is read from the manifest (PEV, standard error, or reliability plus genetic variance); when the manifest carries none, supply this rather than let the function guess. Ignored when the payload is a draws matrix.
n_draws	The number of draws to reconstruct per genotype when the payload is point GEBVs (default 1000L).
crop	The crop the decision concerns (default "wheat").
grounding	Optional override for the manifest’s own grounding token; defaults to the token the manifest carries.

decisive_prob Minimum posterior probability that the leading genotype beats the incumbent.
 ... Further arguments passed to [plan_variety\(\)](#).

Details

The manifest is read by its public contract. GEBVs are sought in `outputs$gebv` (the `breeding_values` payload shape) as either a numeric vector of point GEBVs (one per genotype) or a matrix of GEBV draws (genotypes in columns). The grounding token rides from the manifest exactly as the nitrogen tail does – an unverified breeding-values manifest forces the incumbent. `grainPlan` takes no dependency on the manifest constructor; the manifest is read by duck-typed S7 property access, the composition-layer convention.

When the manifest carries point GEBVs rather than a draws matrix, the downside-aware loss still needs a distribution to weigh, so a per-genotype normal posterior is reconstructed with the GEBV as its mean. The standard deviation is the genotype's own genomic uncertainty, resolved in this order and never fabricated silently:

1. an explicit caller `gebv_sd` (a single value or one per genotype) if supplied, which always wins;
2. a per-genotype prediction-error variance on the manifest (`outputs$pev` or `metadata$pev`), used as $sd_j = \sqrt{PEV_j}$;
3. a per-genotype standard error on the manifest (`outputs$gebv_se`, `outputs$se`, or `metadata$gebv_sd`), used directly;
4. a per-genotype reliability (`outputs$reliability` or `metadata$reliability`) together with an additive-genetic variance (`metadata$genetic_var`), used as $sd_j = \sqrt{\sigma_a^2 (1 - r_j^2)}$ from the standard reliability-PEV relation (Mrode, 2014).

When none of these is available the function stops rather than invent an uncertainty scale a point GEBV cannot carry: a genotype switch must not turn on a magic constant. The resolved source is recorded in the decision's context as `gebv_sd_source`, and the reconstruction never strengthens the grounding.

Value

A [grain_decision](#) of kind "variety" whose action is the recommended genotype name. Held at the incumbent, unread, when the manifest's own producer declared an abstention (`summary$abstained`), or when it declares an `inferential_target` other than "breeding_values".

References

Mrode, R. A. (2014). *Linear Models for the Prediction of Animal Breeding Values* (3rd ed.). CABI. The reliability-PEV relation $r^2 = 1 - PEV/\sigma_a^2$.

See Also

[plan_variety\(\)](#) for the caller-assembled-matrix form.

Other grain-decisions: [plan_grade_target\(\)](#), [plan_nitrogen_rate\(\)](#), [plan_nitrogen_rate_from_manifest\(\)](#), [plan_variety\(\)](#)

Examples

```
## Not run:
# `manifest` is a breeding_values orchestra_manifest S7 object -- the manifest
# flexyBayes's genomics targets emit (genomic_summary lifted through the
# composition layer), carrying GEBVs in `outputs$gebv` and a grounding token
# in `metadata$grounding`. The manifest is read by duck-typed S7 property
# access, so grainPlan needs no dependency on the contract package.
plan_variety_from_manifest(manifest, risk_aversion = 0.5)

## End(Not run)
```

```
print.grain_decision  Print a grain decision
```

Description

Print a grain decision

Arguments

x	A grain_decision .
...	Ignored, for compatibility with print() .

Value

x, invisibly.

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax  <- rnorm(800L, 5.5, 0.3)
yld   <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
               numeric(800L))
print(plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                        crop = "wheat",
                        grounding = decideR::grounding_grounded()))
```

print.grain_plan	<i>Print a grain plan</i>
------------------	---------------------------

Description

Print a grain plan

Arguments

x	A grain_plan .
...	Ignored, for compatibility with print() .

Value

x, invisibly.

Examples

```
set.seed(1L)
rates <- seq(0, 150, by = 25)
ymax <- rnorm(800L, 5.5, 0.3)
yld <- vapply(rates, function(r) 3 + (ymax - 3) * (1 - exp(-r / 70)),
              numeric(800L))
gd <- plan_nitrogen_rate(yld, rates, price_grain = 350, price_n = 1.3,
                        crop = "wheat",
                        grounding = decideR::grounding_grounded())
print(plan_season(list(gd), crop = "wheat", season = "2026 paddock 7"))
```

print.n_rate_economics	<i>Print nitrogen-rate economics</i>
------------------------	--------------------------------------

Description

Print nitrogen-rate economics

Usage

```
## S3 method for class 'n_rate_economics'
print(x, ...)
```

Arguments

x	An n_rate_economics object.
...	Ignored, for compatibility with print() .

Value

x, invisibly.

Examples

```
print(n_rate_economics(price_grain = 350, price_n = 1.3))
```

wheat_protein_bands	<i>Wheat protein-band value schedule</i>
---------------------	--

Description

The canonical Australian milling-wheat payoff: a price per tonne that steps up as grain protein clears each receival band. This is the schedule the orchestra runs kept hand-building when they priced a quality-graded wheat decision; `wheat_protein_bands()` consolidates it into a first-class, documented `decideR::grade_band_value()` schedule so a decision can price protein without re-deriving the band table.

Usage

```
wheat_protein_bands(breaks = c(10.5, 11.5, 13), values = c(300, 330, 360, 400))
```

Arguments

breaks	A strictly increasing numeric vector of protein thresholds in percent (default <code>c(10.5, 11.5, 13.0)</code> , the milling / hard / prime-hard edges).
values	A numeric vector of dollar-per-tonne band payoffs, of length <code>length(breaks) + 1</code> , ordered from the lowest (feed) band up (default an illustrative <code>c(300, 330, 360, 400)</code>).

Details

The default breaks are the protein thresholds that separate the broad Australian wheat grades by protein – feed below the milling floor, then the standard milling grades (Australian Standard White, Australian Premium White) and the high-protein hard grades (Australian Hard, Australian Prime Hard) – and the default values are an illustrative step-up of dollars per tonne across them. The band shape (a step at each protein threshold) follows the published Grain Trade Australia wheat standards; the dollar values are illustrative and not receival prices. Supply current prices for a real decision.

Value

A `decideR::grade_band_value` schedule: a vectorised function from a protein outcome to its band payoff in dollars per tonne, carrying the schedule for inspection.

References

Grain Trade Australia, *Wheat Standards* (publicly published receival standards; band thresholds), and the broad Australian wheat grade classes (AGW Feed, ASW, APW, AH, APH). Dollar values are illustrative.

See Also

[barley_malting_premium\(\)](#) for the barley counterpart; [plan_grade_target\(\)](#) for the decision that consumes it.

Other grain-value: [barley_malting_premium\(\)](#), [n_rate_economics\(\)](#)

Examples

```
v <- wheat_protein_bands()
v(c(9.5, 11.0, 11.5, 13.4))
v
```

Index

- * **grain-decisions**
 - plan_grade_target, 9
 - plan_nitrogen_rate, 11
 - plan_nitrogen_rate_from_manifest, 12
 - plan_variety, 15
 - plan_variety_from_manifest, 17
- * **grain-objects**
 - grain_decision, 4
 - grain_is_grounded, 5
 - grain_plan, 6
 - grain_plan_table, 7
 - plan_season, 14
- * **grain-value**
 - barley_malting_premium, 2
 - n_rate_economics, 8
 - wheat_protein_bands, 21

barley_malting_premium, 2

barley_malting_premium(), 9, 10, 22

decideR::decide(), 10, 16

decideR::decide_input_rate(), 12

decideR::decide_rate_from_manifest(), 13

decideR::grade_band_utility(), 10

grain_decision, 4, 5–7, 10–14, 16, 18, 19

grain_decision(), 6–8, 15

grain_is_grounded, 5

grain_is_grounded(), 5, 7, 8, 15

grain_plan, 5, 6, 7, 8, 14, 15, 20

grain_plan(), 5, 6, 8, 15

grain_plan_table, 7

grain_plan_table(), 5–7, 15

n_rate_economics, 8, 20

n_rate_economics(), 3, 11, 12, 22

plan_grade_target, 9

plan_grade_target(), 3, 5, 12, 13, 16, 18, 22

plan_nitrogen_rate, 11

plan_nitrogen_rate(), 5, 9, 10, 13, 16, 18

plan_nitrogen_rate_from_manifest, 12

plan_nitrogen_rate_from_manifest(), 10–12, 16, 18

plan_season, 14

plan_season(), 5–8

plan_variety, 15

plan_variety(), 5, 10, 12, 13, 17, 18

plan_variety_from_manifest, 17

plan_variety_from_manifest(), 10, 12, 13, 16

print(), 19, 20

print.grain_decision, 19

print.grain_plan, 20

print.n_rate_economics, 20

wheat_protein_bands, 21

wheat_protein_bands(), 2, 3, 9, 10