Package ‘rfcipCalcPass’

October 10, 2025

Type Package

Title FCIP PASS-Based Calculators and Tools

Version 0.0.0.9000

Author Francis Tsiboe [aut, cre] (<https://orcid.org/0000-0001-5984-1072>)

Maintainer Francis Tsiboe <ftsiboe@hotmail.com>

Contributor -

Reviewer -

Creator Francis Tsiboe

Description Provides R implementations of calculators and tables used in the Federal Crop Insurance Program's (FCIP) Policy Acceptance and Storage System (PASS).
Includes tools for working with APH yields, area and individual plan pricing, premium subsidies, and ADM-based structures. Designed for use alongside the 'rfcip' package.
Functions are based on formulas published in USDA Risk Management Agency (RMA), Appendix III.M.13 Handbooks.

License GPL-3 + file LICENSE

URL https://github.com/you/rfcipCalcPass

BugReports https://github.com/you/rfcipCalcPass/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Depends R (>= 3.5)

Imports rfcip, data.table, dplyr, stats, utils, arrow, piggyback, gh, purrr, readr

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Remotes github::dylan-turner25/rfcip

LazyData true

Cite-us If you find it useful, please consider staring the repository and citing the following studies
- Tsiboe, F. and Turner, D. (2025). ``Incorporating buy-
up price loss coverage into the United States
farm safety net.'' Applied Economic Perspectives and Policy.
- Tsiboe, F., et al. (2025). ``Risk reduction impacts of crop insurance in the United States.''
Applied Economic Perspectives and Policy.
- Gaku, S. and Tsiboe, F. (2024). Evaluation of alternative farm safety net program combination strategies. Agricultural Finance Review.

Contents

`adm_insurance_plans` *Simulator Helper Datasets*

Description

A combined dataset for adm_insurance_plans

Usage

```
data(adm_insurance_plans)
```

Format

A data frame with 773 rows and 12 columns covering 1989-2025.

Source

USDA-RMA, Actuarial Data Master (ADM)

`adm_mco_2026` *adm_mco_2026*

Description

A combined dataset for adm_mco_2026

Usage

```
data(adm_mco_2026)
```

Format

A data frame with 39342 rows and 25 columns covering 2026-2026.

Source

Watts and Associates

`bfrvfr_subsidy_schedule_ob3` *BFR and VFR Premium Subsidy Schedule*

Description

Calculates the premium subsidy percentage for a Beginning Farmer and Rancher (BFR)/Veteran Farmer and Rancher (VFR) based on the number of years since the producer first qualified as a BFR/VFR, following the tiered structure established in the One Big Beautiful Bill (OB3).

Usage

```
bfrvfr_subsidy_schedule_ob3(status_years)
```

Arguments

`status_years`	Integer. Number of years in BFR/VFR status.

Details

Under this policy:

- Years 1-2: 15% subsidy
- Year 3: 13% subsidy
- Year 4: 11% subsidy
- Years 5-10: 10% subsidy
- Year 11 and beyond: 0% subsidy (outside BFR/VFR window)

Value

A named list with a single element:

bfr_subsidy_percent Numeric. The applicable premium subsidy percentage (e.g., 0.15 for 15%).

See Also

Other Producer Premium Subsidy: `fcip_premium_subsidy_schedule()`, `premium_subsidy_control_functions()`

`calc_base_premium_rate` *Calculate Base Premium Rate Components with Control Options*

Description

Given a target `rate_yield` and both current- and prior-year parameters, this helper computes all intermediate components of the RMA premium-rate pipeline-yield ratios, rate multipliers, base rates, and branch-specific base premium rates-and then applies the RMA cap-and-pmin logic to produce the final `base_premium_rate`.

Usage

```
calc_base_premium_rate(
  rate_yield,
  reference_amount,
  reference_rate,
  fixed_rate,
  rate_differential_factor,
  unit_residual_factor,
  exponent_value,
  prior_year_reference_amount,
  prior_year_reference_rate,
  prior_year_fixed_rate,
  prior_year_rate_differential_factor,
  prior_year_unit_residual_factor,
  prior_year_exponent_value,
  control = rfcipCalcPass_control()
)
```

Arguments

`rate_yield` Numeric. The target yield per acre for which to infer the base premium rate.

`reference_amount` Numeric. Current-year reference yield used to compute yield ratios.

`reference_rate` Numeric. Rate used in the base rate calculation for the current year.

`fixed_rate` Numeric. Fixed increment added after exponentiation for the current year.

`rate_differential_factor` Numeric. Differential factor applied to the current-year base rate.

`unit_residual_factor` Numeric. Unit residual factor applied to the current-year base rate.

`exponent_value` Numeric. Exponent used to raise the current-year yield ratio.

`prior_year_reference_amount`
Numeric. Reference yield for the prior year.

`prior_year_reference_rate`
Numeric. Rate used in the base rate calculation for the prior year.

`prior_year_fixed_rate`
Numeric. Fixed increment for the prior year.

`prior_year_rate_differential_factor`
Numeric. Differential factor for the prior-year base rate.

`prior_year_unit_residual_factor`
Numeric. Unit residual factor for the prior-year base rate.

`prior_year_exponent_value`
Numeric. Exponent used to raise the prior-year yield ratio.

`control` a list of control parameters; see `rfcipCalcPass_control`.

- `rma_rounding`: logical, whether to use RMA default rounding (FALSE) or disable (TRUE).
- `yield_ratio_cup_and_cap`: logical, whether to apply a 0.5-1.5 cup-and-cap to yield ratios.

Details

Internally:

1. `rma_rounding` is set to 1 (no extra scaling) or 1e6 (to emulate RMA's rounding) based on `control$rma_rounding`.
2. Yield ratios are rounded to `2 * rma_rounding` digits, then optionally cup-and-capped to `[0.5,1.5]`.
3. Ratios are raised to their exponents and rounded to `8 * rma_rounding` digits.
4. Base rates are computed by multiplying by reference rates and adding fixed rates, then rounded.
5. Branch base premium rates are computed by applying differential and unit residual factors, then rounded.
6. The final `base_premium_rate` is `pmin(current, prior * 1.2, 0.999)` rounded to `8 * rma_rounding` digits.

Value

A named list of components:

current_yield_ratio Rounded and optionally capped yield ratio (current year).

prior_year_yield_ratio Rounded and optionally capped yield ratio (prior year).

current_rate_multiplier Rounded exponentiated multiplier (current year).

prior_year_rate_multiplier Rounded exponentiated multiplier (prior year).

current_base_rate Rounded base rate before differential factors (current year).

prior_year_base_rate Rounded base rate before differential factors (prior year).

current_base_premium_rate Rounded branch premium rate (current year).

prior_year_base_premium_rate Rounded branch premium rate (prior year).

base_premium_rate Final capped-and-minimum premium rate.

See Also

Other FCIP Premium / Indemnity Calculators: `fcip_calculator()`, `fcip_calculator_area()`, `fcip_calculator_individual()`, `fcip_calculator_supplemental()`

calc_payment_rate *Calculate Insurance Payment Rate (Indemnification)*

Description

Computes the payment rate for a insurance policy based on yields, prices, and plan characteristics.

Usage

```
calc_payment_rate(
  expected_yield,
  final_yield,
  harvest_price,
  price_election_amount,
  projected_price,
  trigger_index,
  coverage_range = 1,
  insurance_plan_code,
  control = rfcipCalcPass_control()
)
```

Arguments

`expected_yield` Numeric. The expected yield level.

`final_yield` Numeric. The actual harvested yield.

`harvest_price` Numeric. The market price at harvest time.

`price_election_amount` Numeric. The price selected at signup for the insurance policy.

`projected_price` Numeric. The market price at harvest time.

`trigger_index` Numeric. trigger_index.

`coverage_range` Numeric. coverage_range.

`insurance_plan_code` Character or numeric. The insurance plan identifier used to determine how prices and revenues are handled.

`control` List. A control list typically generated by `rfcipCalcPass_control` containing elements:

- **harvest_price_inclusion_plans** Vector of plan codes that include harvest price in guarantee calculation.
- **non_price_risk_plans** Vector of plan codes that do not adjust revenue based on harvest price.

Value

Numeric. The payment rate as a proportion of the insurance guarantee (values between 0 and 1).

clear_rfcipCalcPass_cache *Clear the package cache of downloaded data files*

Description

Deletes the entire cache directory used by the **rfcipCalcPass** package to store downloaded data files. Useful if you need to force re-download of data, or free up disk space.

Usage

```
clear_rfcipCalcPass_cache()
```

Value

Invisibly returns NULL. A message is printed indicating which directory was cleared.

See Also

Other helpers: rfcipCalcPass_control()

Examples

```
## Not run:
# Remove all cached data files so they will be re-downloaded on next use
clear_rfcipCalcPass_cache()

## End(Not run)
```

fcip_adm_area *Retrieve ADM parameters for Basic Index/Area-based insurance plans*

Description

This function retrieves and aggregates Actuarial Data Master (ADM) parameters for index and area insurance plans, handling differences in data structure between the 2011-2016 and 2017+ policy years.

Usage

```
fcip_adm_area(
  farmdata,
  year = 2020,
  keep_price_volatility = FALSE,
  control = rfcipCalcPass_control()
)
```

Arguments

`farmdata` A `data.table` of farm-level input. Must contain all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`

`year` Numeric scalar giving the target insurance crop year (e.g. `2024`). Default is `2020`.

`keep_price_volatility`
Logical scalar. If `TRUE`, grouping by `price_volatility_factor` is retained in the output; otherwise actuarial parameters are averaged over all `price_volatility_factor` (default: `FALSE`).

`control` a list of control parameters used to set `liability_adjustment_factor` and `multiple_commodity_adjustment_factor`. See `rfcipCalcPass_control`

Value

A `data.table` with one row per unique combination of commodity_year and all fields `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`,containing: `price_volatility_factor`, `expected_county_yield`, `final_county_yield`, `expected_county_revenue`, `final_county_revenue`, `base_rate`, and `payment_factor` `Fixed parameters:`liability_adjustment_factorandmultiple_commodity_adjustment_factor‘. see `rfcipCalcPass_control`

See Also

Other Actuarial Data Master (ADM): `fcip_adm_individual()`

Examples

```
## Not run:
index_params <- fcip_adm_index(
  farmdata = plan_comparison[insurance_plan_code %in% c(4:6,16,17),
c("commodity_year","producer_id",FCIP_INSURANCE_POOL,FCIP_INSURANCE_ELECTION), with = FALSE],
  year   = 2020
)

## End(Not run)
```

`fcip_adm_individual` *Retrieve ADM parameters for Basic On-farm experience-based insurance plans*

Description

This function pulls together all of the Actuarial Data Master (ADM) parameters needed to price Yield Protection (YP), Actual Production History (APH), Revenue Protection (RP), and RP-Harvest Price Exclusion (RP-HPE) policies for a given set of farmer records and crop year

Usage

```
fcip_adm_individual(
  farmdata,
  year = 2020,
  insurance_plan_individual = c(1:3, 90),
  control = rfcipCalcPass_control()
)
```

Arguments

farmdata
: A data.table of farm-level input. Must contain all fields in FCIP_INSURANCE_POOL and FCIP_INSURANCE_ELECTION

year
: Numeric scalar giving the target insurance crop year (e.g. 2024). Default is 2020.

insurance_plan_individual
: can be either a character string indicating an on-farm experience based insurance plan.

control
: a list of control parameters used to set revenue_lookup_adjustment_factor, unit_structure_discount_factor, additive_optional_rate_adjustment_factor, and multiplicative_optional_rate_adjustment_factor See rfcipCalcPass_control

Value

A data.table with one row per unique combination of commodity_year and all fields FCIP_INSURANCE_POOL and FCIP_INSURANCE_ELECTION, containing:

- Base rate fields (ADM-A01010): reference_amount, reference_rate, exponent_value, fixed_rate, and their prior-year analogues.
- Coverage level differential factors (ADM-A01040): rate_differential_factor, residual_factor, and prior-year analogues.
- Insurance offer fields (ADM-A00030): beta_id
- Historical revenue capping fields (ADM-A01110): capping_reference_yield, capping_reference_rate, etc., plus beta_0_factor-beta_14_factor.
- Price fields (ADM-A00810): projected_price, harvest_price, and price_volatility_factor
- Fixed parameters: revenue_lookup_adjustment_factor, unit_structure_discount_factor, additive_optional_rate_adjustment_factor, and multiplicative_optional_rate_adjustment_factor based on the control list. see rfcipCalcPass_control

See Also

Other Actuarial Data Master (ADM): fcip_adm_area()

Examples

```
## Not run:
adm_params <- fcip_adm_individual(
  farmdata = plan_comparison[insurance_plan_code %in% c(1:3,90),
c("commodity_year","producer_id",FCIP_INSURANCE_POOL,FCIP_INSURANCE_ELECTION), with = FALSE],
  year  = 2020
)

## End(Not run)
```

`fcip_calculator` *FCIP Premium & Indemnity Calculator Dispatcher*

Description

Routes farm-level data to the appropriate FCIP premium (and optional indemnity) calculator based on insurance plan codes. Currently supports

- Yield Protection (YP) (code = 01)
- Actual Production History (APH) (code = 90)
- Revenue Protection (RP) (code = 02)
- Revenue Protection with Harvest Price Exclusion (RP-HPE) (code = 03)
- Area Yield Protection (AY) (code = 04)
- Area Revenue Protection (AR) (code = 05)
- Area Revenue Protection with Harvest Price Exclusion (AR-HPE) (code = 06)
- Margin Protection (MP) (code = 16)
- Margin Protection with Harvest Price Exclusion(code = 17)
- Supplemental Coverage Option (SCO) (code = 31, 32 ,33)
- Enhanced Coverage Option (ECO) (code = 87, 88 ,89)

Usage

```
fcip_calculator(
  farmdata,
  farm_identifiers = NULL,
  supplemental_election = NULL,
  projected_price_provided = NULL,
  keep_all_fields = FALSE,
  calculate_indemnity = FALSE,
  premium_subsidy_control = NULL,
  keep_price_volatility = FALSE,
  control = rfcipCalcPass_control()
)
```

Arguments

`farmdata` A `data.table` of farm-level input. Must contain ‘commodity_year’ and all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`

`farm_identifiers`
column name(s) that uniquely identify each record/farm/producer. If `NULL`, a "producer_id" column is created.

`supplemental_election`
Character. Indicates which supplemental coverage type(s) to calculate. Must be one or more of `"SCO"` for Supplemental Coverage Option (SCO) `"ECO90"` for Enhanced Coverage Option at 90% coverage level (ECO90) `"ECO95"` for Enhanced Coverage Option at 95% coverage level (ECO95) `"MCO90"` for Margin Coverage Option at 90% coverage level (MCO90) `"MCO95"` for Margin Coverage Option at 90% coverage level (MCO95)

`projected_price_provided`
: Numeric or `NULL`. If non-`NULL`, overrides the `projected_price` column obtained from ADM.

`keep_all_fields`
: logical(1) If FALSE, returns only identifiers and core premium fields. If TRUE, retains all intermediate columns.

`calculate_indemnity`
: logical(1) If TRUE, also computes per-acre indemnity based on planted acres, damage rates, harvest price, and farm yields. Defaults to FALSE.

`premium_subsidy_control`
: A named list of adjustment functions, typically created using `premium_subsidy_control_function` If `NULL`, the default adjustment functions are used.

`keep_price_volatility`
: Logical scalar. If `TRUE`, grouping by `price_volatility_factor` is retained in the output; otherwise actuarial parameters are averaged over all `price_volatility_factor` (default: `FALSE`).

`control`
: a list of control parameters; see `rfcipCalcPass_control()`. `reported_acres`, `insured_share_percent`, `damage_area_rate`, `price_election_percent`, `rma_rounding`, and `yield_ratio_cup_and_cap` are fixed via control

Details

This dispatcher performs:

1. **Setup & validation**
   - Coerces farmdata to data.table, sets up identifiers
   - Verifies required columns for both ADM lookup and policy calcs
2. **Per-year split**
   - For each unique commodity_year, routes data to:
     - Individual policies (plans 1-3, 90) calculator (`fcip_calculator_individual`)
     - Area (4-6) and Margin (16-17) plans calculator (`fcip_calculator_area`)
3. **Binding**
   - Row-binds results for all years and plan types

Value

#' A `data.table` with one row per farm (and, if retained, per `price_volatility_factor`). At minimum, the following columns will appear:

- farm identifiers: as specified by `farm_identifiers`
- all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`
- `price_election_percent`, `insured_share_percent`
- `reported_acres`, `insured_acres`, `liability_amount`
- `premium_rate`, `base_rate`, `total_premium_amount`, `subsidy_amount`, `producer_premium_amount`
- `indemnity_amount` (if `calculate_indemnity` = `TRUE`)

See Also

Other FCIP Premium / Indemnity Calculators: `calc_base_premium_rate()`, `fcip_calculator_area()`, `fcip_calculator_individual()`, `fcip_calculator_supplemental()`

fcip_calculator_area *Calculator for area-base Crop Insurance Policies*

Description

Calculates outcomes for;

- Area Yield Protection (AY) (code = 04)
- Area Revenue Protection (AR) (code = 05)
- Area Revenue Protection with Harvest Price Exclusion (AR-HPE) (code = 06)
- Margin Protection (MP) (code = 16)
- Margin Protection with Harvest Price Exclusion(code = 17)

Usage

```
fcip_calculator_area(
  farmdata,
  year = 2020,
  farm_identifiers = NULL,
  adm = NULL,
  projected_price_provided = NULL,
  keep_all_fields = FALSE,
  calculate_indemnity = FALSE,
  keep_price_volatility = FALSE,
  premium_subsidy_control = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`farmdata` A `data.table` of farm-level input. Must contain all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`

`year` Numeric scalar giving the target insurance crop year (e.g. `2024`). Default is `2020`.

`farm_identifiers`
column name(s) that uniquely identify each record/farm/producer. If `NULL`, a "producer_id" column is created.

`adm` Optional `data.table` of ADM parameters. If `NULL`, parameters are fetched via `fcip_adm_area`

`projected_price_provided`
Numeric or `NULL`. If non-`NULL`, overrides the `projected_price` column obtained from ADM.

`keep_all_fields`
logical(1) If FALSE, returns only identifiers and core premium fields. If TRUE, retains all intermediate columns.

`calculate_indemnity`
logical(1) If TRUE, also computes per-acre indemnity based on planted acres, damage rates, harvest price, and farm yields. Defaults to FALSE.

`keep_price_volatility`
: Logical scalar. If `TRUE`, grouping by `price_volatility_factor` is retained in the output; otherwise actuarial parameters are averaged over all `price_volatility_factor` (default: `FALSE`).

`premium_subsidy_control`
: A named list of adjustment functions, typically created using `premium_subsidy_control_function` If `NULL`, the default adjustment functions are used.

`control`
: a list of control parameters; see `rfcipCalcPass_control`. `reported_acres`, `insured_share_percent`, `damage_area_rate`, and `price_election_percent` are fixed via control

Value

A `data.table` with one row per farm (and, if retained, per `price_volatility_factor`). At minimum, the following columns will appear:

- farm identifiers: as specified by `farm_identifiers`
- all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`
- `price_election_percent`, `insured_share_percent`
- `reported_acres`, `insured_acres`, `liability_amount`
- `base_rate`, `total_premium_amount`, `subsidy_amount`, `producer_premium_amount`
- `indemnity_amount` (if `calculate_indemnity` = `TRUE`)

See Also

Other FCIP Premium / Indemnity Calculators: `calc_base_premium_rate()`, `fcip_calculator()`, `fcip_calculator_individual()`, `fcip_calculator_supplemental()`

`fcip_calculator_individual` *Calculator for Individual Crop Insurance Policies*

Description

Calculates outcomes for;

- Yield Protection (YP) (code = 01)
- Actual Production History (APH) (code = 90)
- Revenue Protection (RP) (code = 02)
- Revenue Protection with Harvest Price Exclusion (RP-HPE) (code = 03)

Usage

```
fcip_calculator_individual(
  farmdata,
  year = 2020,
  farm_identifiers = NULL,
  adm = NULL,
  projected_price_provided = NULL,
  keep_all_fields = FALSE,
  calculate_indemnity = FALSE,
  premium_subsidy_control = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`farmdata` A `data.table` of farm-level input. Must contain `rate_yield`,`approved_yield`, and all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`

`year` Numeric scalar giving the target insurance crop year (e.g. `2024`). Default is `2020`.

`farm_identifiers` column name(s) that uniquely identify each record/farm/producer. If `NULL`, a "producer_id" column is created.

`adm` Optional `data.table` of ADM parameters. If `NULL`, parameters are fetched via `fcip_adm_individual`.

`projected_price_provided` Numeric or `NULL`. If non-`NULL`, overrides the `projected_price` column obtained from ADM.

`keep_all_fields` logical(1) If FALSE, returns only identifiers and core premium fields. If TRUE, retains all intermediate columns.

`calculate_indemnity` logical(1) If TRUE, also computes per-acre indemnity based on planted acres, damage rates, harvest price, and farm yields. Defaults to FALSE.

`premium_subsidy_control` A named list of adjustment functions, typically created using `premium_subsidy_control_function`
If `NULL`, the default adjustment functions are used.

`control` a list of control parameters; see `rfcipCalcPass_control`. `reported_acres`, `insured_share_percent`, `damage_area_rate`, `price_election_percent`, `rma_rounding`, and `yield_ratio_cup_and_cap` are fixed via control

Value

A `data.table` with one row per farm. At minimum, the following columns will appear:

- farm identifiers: as specified by `farm_identifiers`
- all fields in `FCIP_INSURANCE_POOL` and `FCIP_INSURANCE_ELECTION`
- `price_election_percent`, `insured_share_percent`
- `reported_acres`, `insured_acres`, `liability_amount`
- `premium_rate`, `total_premium_amount`, `subsidy_amount`, `producer_premium_amount`
- `indemnity_amount` (if `calculate_indemnity` = `TRUE`)

See Also

Other FCIP Premium / Indemnity Calculators: `calc_base_premium_rate()`, `fcip_calculator()`, `fcip_calculator_area()`, `fcip_calculator_supplemental()`

`fcip_calculator_supplemental` *FCIP Supplemental Insurance Calculator*

Description

FCIP Supplemental Insurance Calculator

Usage

```
fcip_calculator_supplemental(
  basic_policy_data,
  year = 2023,
  farm_identifiers = NULL,
  adm = NULL,
  projected_price_provided = NULL,
  keep_price_volatility = FALSE,
  premium_subsidy_control = NULL,
  calculate_indemnity = FALSE,
  control = rfcipCalcPass_control(),
  supplemental_election = NULL
)
```

Arguments

`basic_policy_data`
: `data.frame` or `data.table`. Core insurance records including columns such as `insurance_plan_code`, `coverage_level_percent`, `unit_structure_code`, `liability_amount`, `insured_acres`, and `endorsement_area_percent`.

`year`
: Integer. The crop year for which to apply the supplemental insurance calculations (default: `2023`).

`farm_identifiers`
: Optional character vector of column names uniquely identifying farms (e.g., `c("producer_id")`). If `NULL`, unique row IDs will be created.

`adm`
: Optional `data.table`. Area Data Management (ADM) table. If `NULL`, it is generated internally using `fcip_adm_area()`.

`projected_price_provided`
: Optional numeric. If provided, this value overrides the projected price from ADM.

`keep_price_volatility`
: Logical. If `TRUE`, preserves the `price_volatility_factor` dimension in results. Defaults to `FALSE`.

`premium_subsidy_control`
: Optional control table for premium subsidy rates.

`calculate_indemnity`
: logical(1) If TRUE, also computes per-acre indemnity based on planted acres, damage rates, harvest price, and farm yields. Defaults to FALSE.

`control`
: A list of internal parameters used by calculation functions, typically from `rfcipCalcPass_control()`

`supplemental_election`
: Character. Indicates which supplemental coverage type(s) to calculate. Must be one or more of `"SCO"` for Supplemental Coverage Option (SCO) `"ECO90"` for Enhanced Coverage Option at 90% coverage level (ECO90) `"ECO95"` for Enhanced Coverage Option at 95% coverage level (ECO95) `"MCO90"` for Margin Coverage Option at 90% coverage level (MCO90) `"MCO95"` for Margin Coverage Option at 90% coverage level (MCO95)

Details

Calculates outcomes for area-based supplemental coverage options under the Federal Crop Insurance Program (FCIP), including:

- **Supplemental Coverage Option (SCO)** -
  - Yield Protection (YP) (plan code = 31)
  - Revenue Protection (RP) (plan code = 32)
  - Revenue Protection with Harvest Price Exclusion (RP-HPE) (plan code = 33)
- **Enhanced Coverage Option (ECO90 & ECO95)** -
  - YP (plan code = 87)
  - RP (plan code = 88)
  - RP-HPE (plan code = 89)
- **Margin Coverage Option (MCO90 & MCO95)** -
  - YP (plan code = 67)
  - RP (plan code = 68)
  - RP-HPE (plan code = 69)

Not implemented in this function yet (listed for completeness):

- **Post-Application Coverage Endorsement (PACE)**
- **Hurricane Insurance Protection - Wind Index (HIP-WI)**
- **Frost/Freeze Insurance Protection - Supplemental Index (FIP-SI)**
- **Margin Protection (MP)**
- **Stacked Income Protection Plan (STAX)**

Value

A `data.table` in wide format, where each calculated outcome is prefixed with its corresponding election type (e.g., `SCO_indemnity_amount`). Outcome columns include:

- `insured_acres`
- `liability_amount`
- `total_premium_amount`
- `subsidy_amount`
- `producer_premium_amount`
- `indemnity_amount`

See Also

Other FCIP Premium / Indemnity Calculators: `calc_base_premium_rate()`, `fcip_calculator()`, `fcip_calculator_area()`, `fcip_calculator_individual()`

`FCIP_FORCE_AMOUNT_VARIABLES`
Column names to coerce to numeric

Description

A character vector of column names that should be converted to character during data ingestion and cleaning.

Usage

```
FCIP_FORCE_AMOUNT_VARIABLES
```

Format

A `character` vector of column names.

Value

A `character` vector of field names to coerce to numeric

`FCIP_FORCE_CHARACTER_KEYS`
Column names to coerce to character

Description

A character vector of column names that should be converted to character during data ingestion and cleaning.

Usage

```
FCIP_FORCE_CHARACTER_KEYS
```

Format

A `character` vector of column names.

Value

A `character` vector of field names to coerce to character

FCIP_FORCE_NUMERIC_KEYS *Column names to coerce to numeric*

Description

A character vector of column names that should be converted from character to numeric during data ingestion and cleaning.

Usage

```
FCIP_FORCE_NUMERIC_KEYS
```

Format

A `character` vector of column names.

Details

The following fields, although often stored as text, represent numeric values and must be coerced for accurate calculation and analysis:

- `commodity_year`: Crop year of the record.
- Fields in `FCIP_INSURANCE_POOL`: `state_code`, `county_code`, `commodity_code`, `type_code`, `practice_code`.
- `record_category_code`: Category of the record.
- `insurance_plan_code`: Code for the insurance plan.
- `coverage_level_percent`: Coverage level percentage.

Value

A `character` vector of field names to coerce to numeric.

Examples

```
## Not run:
# View default keys
FCIP_FORCE_NUMERIC_KEYS

# Extend with a custom numeric field
rFarmPolicySim:::FCIP_FORCE_NUMERIC_KEYS <- c(
  rFarmPolicySim:::FCIP_FORCE_NUMERIC_KEYS,
  "custom_numeric_field"
)

## End(Not run)
```

FCIP_INSURANCE_ELECTION *Insurance election identifier fields*

Description

A character vector of column names that define an insurance election within the Federal Crop Insurance Program (FCIP). Each field corresponds to an attribute of a policy election.

Usage

```
FCIP_INSURANCE_ELECTION
```

Format

A `character` vector of field names:

unit_structure_code Structure of the insured unit (e.g., basic, optional).

insurance_plan_code Code for the insurance product (e.g., MPCI, CRC).

coverage_type_code Type of coverage (e.g., Actual/Assumed Yield, Yield Protection).

coverage_level_percent Elected coverage level as a percentage of approved yield or price.

Value

A `character` vector of field names used to specify insurance elections.

Examples

```
## Not run:
# Default election fields
FCIP_INSURANCE_ELECTION

# Override to include only plan and coverage level
rFarmPolicySim:::FCIP_INSURANCE_ELECTION <- c(
  "insurance_plan_code",
  "coverage_level_percent"
)

## End(Not run)
```

FCIP_INSURANCE_ELECTION_RCODED *Insurance election identifier fields (recoded)*

Description

A character vector of recoded column names that specify an insurance election within the Federal Crop Insurance Program (FCIP). These fields correspond to recoded versions of the original election attributes.

Usage

```
FCIP_INSURANCE_ELECTION_RCODED
```

Format

A `character` vector of field names:

unit_structure_recode Recoded unit structure (e.g., basic, optional).

insurance_plan_recode Recoded insurance plan code (e.g., APH, YP).

coverage_type_code Coverage type code (unchanged).

coverage_level_percent unchanged

Value

A `character` vector of recoded insurance election field names.

Examples

```
## Not run:
# View the default recoded election fields
FCIP_INSURANCE_ELECTION_RCODED

# Override to drop the recoded plan field
rFarmPolicySim::FCIP_INSURANCE_ELECTION_RCODED <- c(
  "unit_structure_recode",
  "coverage_type_code",
  "coverage_level_percent"
)

## End(Not run)
```

FCIP_INSURANCE_POOL *Insurance pool identifier fields*

Description

A character vector of column names that together define a unique insurance pool in the Federal Crop Insurance Program (FCIP).

Usage

```
FCIP_INSURANCE_POOL
```

Format

A `character` vector of field names.

Details

Insurance pools represent the most granular level of rate making within FCIP. Each pool is uniquely identified by the combination of:

- **state_code**: State FIPS code
- **county_code**: County FIPS code
- **commodity_code**: Crop commodity code
- **type_code**: Crop type (e.g., grain vs. silage)
- **practice_code**: Production practice (e.g., irrigated, organic)

Value

A `character` vector specifying the columns used to define each FCIP insurance pool.

Examples

```
## Not run:
# Default insurance pool fields
FCIP_INSURANCE_POOL

# Override to a subset of the original fields
rFarmPolicySim:::FCIP_INSURANCE_POOL <- c(
  "state_code", "county_code", "commodity_code"
)

## End(Not run)
```

`fcip_premium_subsidy_schedule` *FCIP producer premium subsidy schedule*

Description

Calculates Federal Crop Insurance Program (FCIP) producer subsidy amounts.

Usage

```
fcip_premium_subsidy_schedule(
  data,
  years,
  premium_subsidy_control = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`data` A `data.table` (or coercible to one) containing at minimum the columns `commodity_year`, `unit_structure_code`, `insurance_plan_code`, `coverage_type_code`, `coverage_level_percent` and `total_premium_amount`. It may also include election flags and percent columns for BVFR, native sod, CC, etc.

`years` Integer vector of policy years to fetch subsidy-percent schedule.

`premium_subsidy_control`
: A named list of adjustment functions, typically created using `premium_subsidy_control_function` If `NULL`, the default adjustment functions are used.

`control`
: a list of control parameters;`continuous_integration_session` set via control to determine the ADM source. see `rfcipCalcPass_control`.

Details

Notes about subsidies goes here

Value

The input `data.table` augmented with these new columns:

`premium_subsidy_percent` Base subsidy percent (0-1).

`base_subsidy_amount` Rounded producer base subsidy.

`bfrvfr_subsidy_amount` Placeholder for Beginning/Veteran FR subsidy.

`native_sod_subsidy_amount` Placeholder for Native sod subsidy.

`cc_subsidy_reduction_amount` Placeholder for CC subsidy reduction.

`subsidy_amount` Final producer subsidy after adjustments.

`ao_expense_subsidy_amount` Final A&O expense subsidy amount.

See Also

Other Producer Premium Subsidy: `bfrvfr_subsidy_schedule_ob3()`, `premium_subsidy_control_functions()`

Examples

```
## Not run:
library(data.table)
dt <- data.table(
  commodity_year = 2022,
  unit_structure_code = "BU",
  insurance_plan_code   = 1,
  coverage_type_code    = "A",
  coverage_level_percent= 0.75,
  total_premium_amount  = 1000
)
out <- fcip_premium_subsidy_schedule(dt, years = 2020:2022)
head(out)

## End(Not run)
```

`get_adm_data_extended` *Fetch and extend ADM data for analysis*

Description

Retrieves ADM data via `rfcip::get_adm_data()`, coerces key columns to numeric, and harmonizes recoding of unit structure and insurance plan codes for downstream analysis.

Usage

```
get_adm_data_extended(
  year = NULL,
  dataset = "baserate",
  control = rfcipCalcPass_control()
)
```

Arguments

`year` Integer or character. Year(s) to fetch. Passed through to `get_adm_data()`. Defaults to `NULL`.

`dataset` Character. Name of the ADM dataset to retrieve (e.g., `'baserate'`). Defaults to `'baserate'`.

`control` a list of control parameters;`continuous_integration_session` set via control to determine the ADM source. see `rfcipCalcPass_control`. if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE).

Value

A `data.table` containing the requested ADM data with numeric conversions and recoded `unit_structure_code` and `insurance_plan_code`.

See Also

Other ADM compression: `get_compressed_adm()`, `get_compressed_adm_area()`, `get_compressed_adm_individua`

Examples

```
## Not run:
dt <- get_adm_data_extended(year = 2020, dataset = 'A01010_BaseRate',control=control)

## End(Not run)
```

`get_compressed_adm` *Unified Entry Point for Compressed ADM Data*

Description

Loads and returns compressed actuarial data, automatically dispatching to either APH-based or index-based data handlers depending on the insurance plan type. Results are cached locally as `.rds` files to reduce re-processing on future runs.

Usage

```
get_compressed_adm(
  year = 2020,
  insurance_plan = NULL,
  dataset = NULL,
  variable = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`year` Integer. The data year to retrieve. Default is `2020`.

`insurance_plan` can be either a character string indicating an insurance plan (ex: `yp` and `yield protection` are both valid) or a numeric value indicating the insurance plan code (i.e. `1`).

`dataset` Character. The name of the actuarial dataset to retrieve (used only for APH-based plans).

`variable` Optional. Character. An additional variable name used for conditional filtering in APH datasets.

`control` a list of control parameters;`continuous_integration_session` set via control to determine the ADM source. see `rfcipCalcPass_control`. if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE). Determines whether APH or index-based processing is used. data is used; if `FALSE`, the full official YTD data from RMA is used. Default is `FALSE`.

Details

The function caches processed output as `.rds` files in a user-specific directory under: `rfcipCalcPass/actuarial_data_` If the file already exists, it is loaded directly from disk; otherwise, it is generated, saved, and returned.

Value

Returns either:

- A `data.table` (for APH-based plans)
- A named list of `data.tables` (`yield`, `rate`) for index-based plans
- `NULL` with a message if no matching plan type is found or no data is processed

See Also

Other ADM compression: `get_adm_data_extended()`, `get_compressed_adm_area()`, `get_compressed_adm_indivi`

`get_compressed_adm_area` *Get Aggregated Area-Based Actuarial Data (Index Plans)*

Description

Loads and processes index-based (area) actuarial data for years 2011 and later. Combines multiple ADM datasets to return aggregated yields and rates.

Usage

```
get_compressed_adm_area(
  year = 2020,
  insurance_plan = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`year` Integer. The data year to retrieve (default: 2020). if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE).

`insurance_plan` can be either a character string indicating an insurance plan (ex: `yp` and `yield protection` are both valid) or a numeric value indicating the insurance plan code (i.e. `1`). Inputting a vector with multiple values will return data for multiple insurance plans. To get a data frame containing all the available insurance plans and insurance plan codes use `get_insurance_plan_codes()`.

`control` a list of control parameters;`continuous_integration_session` set via control to determine the ADM source. see `rfcipCalcPass_control`. if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE).

Value

A list with two `data.table` objects: `yield` (county yield data) and `rate` (base rate/payment factor data).

See Also

Other ADM compression: `get_adm_data_extended()`, `get_compressed_adm()`, `get_compressed_adm_individual(`

`get_compressed_adm_individual`

Get Aggregated APH-Based Actuarial Data

Description

Loads and processes APH-based actuarial data for a given dataset and year. Applies filters, selects parameters based on record type, and aggregates key variables.

Usage

```
get_compressed_adm_individual(
  dataset,
  year = 2020,
  variable = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`dataset` Character. The actuarial dataset name (e.g., "A01010_BaseRate").

`year` Integer. The data year to retrieve (default: 2020).

`variable` Optional. A specific variable name used for some conditional logic. if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE).

`control` a list of control parameters;`continuous_integration_session` set via control to determine the ADM source. see `rfcipCalcPass_control`. if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE).

Value

A `data.table` containing aggregated APH actuarial data.

See Also

Other ADM compression: `get_adm_data_extended()`, `get_compressed_adm()`, `get_compressed_adm_area()`

get_insurance_offerings

Unified Insurance Offerings by Year

Description

Retrieves a comprehensive list of insurance offerings for a given year based on the specified insurance protection level(s). This function integrates data from multiple RFCIP datasets to return plan offerings, coverage levels, and unit structures.

Usage

```
get_insurance_offerings(
  year = 2020,
  insurance_plan = NULL,
  control = rfcipCalcPass_control()
)
```

Arguments

`year`	Integer. The year of data to retrieve. Default is `2020`.
`insurance_plan`	can be either a character string indicating an insurance plan (ex: `yp` and `yield protection` are both valid) or a numeric value indicating the insurance plan code (i.e. `1`).
`control`	List. A list of control parameters, typically created using `rfcipCalcPass_control`, to manage internal behavior and data sourcing.

Value

A `data.table` containing the combined and filtered insurance offering records.

get_insurance_plan_categories

Get Insurance Plan Codes by Plan Type and Year

Description

This internal helper function extracts insurance plan codes from a data table for specified plan types (trigger levels) and a given year or range of years.

Usage

```
get_insurance_plan_categories(
  data,
  year,
  plan_types = c("Basic Individual", "Basic Index", "Basic Dollar", "Basic Area",
    "Basic Livestock", "Supplemental SCO", "Supplemental STAX", "Basic Margin",
   "Supplemental HIPWI", "Supplemental ECO", "Supplemental MCO", "Supplemental PACE",
    "Supplemental FIPSI")
)
```

Arguments

`data`	A `data.table` or `data.frame` containing insurance plan information. Must include the columns `commodity_year`, `plan_type`, and `insurance_plan_code`.
`year`	A numeric or character scalar/vector specifying the target commodity year(s).
`plan_types`	A character vector of plan types to extract. Defaults to: `"Basic Individual"`, `"Basic Index"`, `"Basic Dollar"`, `"Basic Area"`, `"Basic Livestock"`, `"Supplemental SCO"`, `"Supplemental STAX"`, `"Basic Margin"`, `"Supplemental HIPWI"`, `"Supplemental ECO"`, `"Supplemental PACE"`, `"Supplemental FIPSI"`, `"Supplemental MCO"`

Value

A named list where each element corresponds to a plan type, each containing a numeric vector of unique insurance plan codes.

Examples

```
## Not run:
plan_codes <- get_insurance_plan_categories(adm_insurance_plans, 2025)

## End(Not run)
```

`get_insurance_plan_codes_extended`

Lookup insurance plan codes for FCIP insurance plans (Extended Version)

Description

Lookup insurance plan codes for FCIP insurance plans (Extended Version)

Usage

```
get_insurance_plan_codes_extended(
  year = as.numeric(format(Sys.Date(), "%Y")),
  plan = NULL
)
```

Arguments

`year` A single numeric value, or vector of numeric values, indicating what years of the summary of business should be used to get crop names and crop codes. Defaults to the current year.

`plan` can be either a character string indicating an insurance plan (ex: `yp` and `yield protection` are both valid) or a numeric value indicating the insurance plan code (i.e. `1`). Inputting nothing for the `plan` argument will return codes for all insurance plans in the specified year(s).

Value

Returns a tibble containing the relevant commodity year, insurance plan codes, insurance plan name, and insurance plan abbreviation

`initialize_rfcipCalcPass_cache` *Initialize the cache for rfcipCalcPass*

Description

`initialize_rfcipCalcPass_cache()` prepares the local cache of ADM datasets used by **rfcipCalcPass**. It supports two workflows:

Usage

```
initialize_rfcipCalcPass_cache(
  years = as.numeric(format(Sys.Date(), "%Y")),
  source,
  clear_cache = FALSE,
  control = rfcipCalcPass_control()
)
```

Arguments

`years` Integer or character vector of years. Defaults to current year. Used to filter downloaded assets (`source = "release"`) or to choose which years to compute (`source = "on_demand"`).

`source` Character; one of `"release"` or `"on_demand"`. Determines whether ADM data are *downloaded* from the GitHub release tag `"adm_compressed"` or *computed* locally.

`clear_cache` Logical; if `TRUE`, deletes and recreates the cache directory before initialization. Default `FALSE`.

`control` a list of control parameters;`continuous_integration_session` set via control to determine the ADM source. see `rfcipCalcPass_control`. if `FALSE`, the full official Year-To-Date (YTD) data from RMA is used. (default: FALSE). Determines whether APH or index-based processing is used. data is used; if `FALSE`, the full official YTD data from RMA is used. Default is `FALSE`.

Details

- **Release mode** (source = "release"): downloads published ADM assets from the GitHub release tag "adm_compressed" (via piggyback) into the user cache. You can target specific years by filename match (e.g., "2023").
- **On-demand mode** (source = "on_demand"): computes the ADM data locally for the requested years using your internal helpers (e.g., fcip_calculator(), get_compressed_adm()), and stores any outputs in the same cache.

After running this function, other parts of **rfcipCalcPass** can read the initialized ADM files directly from the cache directory: tools::R_user_dir("rfcipCalcPass", "cache").

When should I use it? Run this once per session (or whenever you need to refresh). If you want to ensure a clean state (e.g., to avoid stale files), call with clear_cache = TRUE.

Value

A character vector of file paths under tools::R_user_dir("rfcipCalcPass", "cache"), representing the ADM files now available to rfcipCalcPass.

Examples

```
## Not run:
# 1) Initialize from published releases (current year)
files <- initialize_rfcipCalcPass_cache(source = "release")
files

# 2) Initialize for multiple years from releases
initialize_rfcipCalcPass_cache(years = 2022:2024, source = "release")

# 3) Start fresh, then download
initialize_rfcipCalcPass_cache(source = "release", clear_cache = TRUE)

# 4) Compute on demand for specific years
initialize_rfcipCalcPass_cache(years = c(2023, 2024), source = "on_demand")

# After initialization, other functions can read from:
tools::R_user_dir("rfcipCalcPass", "cache")

## End(Not run)
```

plan_comparison *RMA Cost Estimator Comparison Table*

Description

This dataset was generated using the USDA Risk Management Agency (RMA) Cost Estimator tool. It is used to test and validate premium and liability calculations implemented in this package.

Usage

```
data("plan_comparison")
```

Format

A `data.table` with 73 rows and 21 variables:

insurance_plan_code Insurance plan code: e.g., 3 = RP-HPE, 2 = RP, 1 = YP

state_code FIPS state code (e.g., 19 = Iowa)

county_code FIPS county code (e.g., 1)

commodity_code Commodity code (e.g., 41 = Corn)

type_code Crop type code (e.g., 16)

practice_code Production practice code (e.g., 2)

rate_yield Rate yield used for premium calculation

approved_yield Approved yield (e.g., 100 bushels/acre)

coverage_type_code Coverage type code (e.g., "A")

unit_structure_code Unit structure (e.g., "OU" = Optional Units)

coverage_level_percent Coverage level as a decimal (e.g., 0.85 for 85%)

liability_amount Gross liability amount in U.S. dollars

total_premium_amount Total premium cost in U.S. dollars

subsidy_amount Federal subsidy portion in U.S. dollars

producer_premium_amount Producer-paid premium in U.S. dollars

commodity_year Crop year (e.g., 2020)

farm_yield Farm-level yield (used for validation)

projected_price_provided Projected price per unit used in calculations

plan_comparison_id Identifier for grouping or scenario comparison

price_volatility_factor Volatility factor used in premium calculation

producer_id Synthetic identifier for test farm units

Source

USDA Risk Management Agency (RMA) Cost Estimator. Retrieved July 11, 2025. Source PDFs and data: https://github.com/you/rfcipCalcPass/tree/main/data-raw/plan_comparison_data/ftsiboe

---

`premium_subsidy_control_functions` *Create a Producer premium subsidy Adjustment Control Functions*

Description

Constructs a list of adjustment functions used to override or control how various Federal Crop Insurance Program (FCIP) subsidy components are computed.

Usage

```
premium_subsidy_control_functions(
  base_subsidy_adj = function(x) {
     x[, `:=`(premium_subsidy_percent, 1 *
    premium_subsidy_percent)]
     return(x)
 },
  bfrvfr_subsidy_adj = function(x) {
     x[, `:=`(bfrvfr_subsidy_amount, 0 *
    total_premium_amount)]
     return(x)
 },
  native_sod_subsidy_adj = function(x) {
     x[, `:=`(native_sod_subsidy_amount, 0 *
    total_premium_amount)]
     return(x)
 },
  cc_subsidy_reduction_adj = function(x) {
     x[, `:=`(cc_subsidy_reduction_amount, 0 *
    total_premium_amount)]
     return(x)
 },
  pccp_subsidy_adj = function(x) {
     x[, `:=`(pccp_subsidy_amount, 0 *
    total_premium_amount)]
     return(x)
 },
  organic_transition_premium_adj = function(x) {
     x[,
    `:=`(organic_transitional_subsidy_amount, 0 * total_premium_amount)]
     return(x)

    },
  organic_certified_grain_premium_adj = function(x) {
     x[,
    `:=`(organic_certified_subsidy_amount, 0 * total_premium_amount)]
     return(x)
 },
  wfrp_organic_transition_adj = function(x) {
     x[, `:=`(organic_wfrp_subsidy_amount,
    0 * total_premium_amount)]
     return(x)
 },
  state_private_subsidy_adj = function(x) {
     x[, `:=`(state_private_subsidy_amount, 0
    * total_premium_amount)]
     return(x)
 },
  additional_subsidy_adj = function(x) {
     x[, `:=`(additional_subsidy_amount, 0 *
    total_premium_amount)]
     return(x)
  }
)
```

Arguments

`base_subsidy_adj`
: A function that modifies the base premium subsidy percentage. Default multiplies it by 1 (i.e., passes it through unchanged).

`bfrvfr_subsidy_adj`
: A function that sets the Beginning/Veteran Farmer-Rancher (BVFR) subsidy. Default sets it to 0.

`native_sod_subsidy_adj`
: A function that sets the native sod subsidy. Default sets it to 0.

`cc_subsidy_reduction_adj`
: A function that sets the cover crop (CC) subsidy reduction. Default sets it to 0.

`pccp_subsidy_adj`
: A function for the Pandemic Cover Crop Program subsidy. Default sets it to 0.

`organic_transition_premium_adj`
: A function to adjust premium subsidies for producers in transition to organic certification. Default sets it to 0.

`organic_certified_grain_premium_adj`
: A function to adjust subsidies for certified organic grain and feed producers. Default sets it to 0.

`wfrp_organic_transition_adj`
: A function for WFRP (Whole-Farm Revenue Protection) policies related to organic transition. Default sets it to 0.

`state_private_subsidy_adj`
: A function to compute state private subsidy. Default sets it to 0.

`additional_subsidy_adj`
: A function to compute additional subsidy. Default sets it to 0.

Details

Each parameter is a function that takes a `data.table` and returns a modified version of it. By default, most adjustments are zeroed out or passed through unchanged.

Value

A named list of subsidy adjustment functions.

See Also

Other Producer Premium Subsidy: `bfrvfr_subsidy_schedule_ob3()`, `fcip_premium_subsidy_schedule()`

Examples

```
## Not run:
controls <- subsidy_control(
  base_subsidy_adj = function(data) {
    data[, premium_subsidy_percent := 0.75 * premium_subsidy_percent]
    return(data)
  }
)
```

```
controls$base_subsidy_adj(data.table::data.table(premium_subsidy_percent = 0.8))

## End(Not run)
```

rfcipCalcPass_control *Create a control list of adjustment factors for PASS Calculators*

Description

This function initializes a named list of control parameters (adjustment factors) used throughout the farm policy simulation pipeline. Each element has a sensible default but can be overridden to customize behavior.

Usage

```
rfcipCalcPass_control(
  revenue_lookup_adjustment_factor = 1,
  unit_structure_discount_factor = 1,
  additive_optional_rate_adjustment_factor = 0,
  multiplicative_optional_rate_adjustment_factor = 1,
  capped_revenue_add_on_factor = 0,
  liability_adjustment_factor = 1,
  multiple_commodity_adjustment_factor = 1,
  reported_acres = 1,
  insured_share_percent = 1,
  price_election_percent = 1,
  damage_area_rate = 1,
  harvest_price_inclusion_plans = c(2, 5, 16, 32, 88),
  non_price_risk_plans = c(1, 90, 4, 31, 87),
  rma_rounding = TRUE,
  yield_ratio_cup_and_cap = TRUE,
  continuous_integration_session = FALSE,
  adm_decoy_state_abb = "ND"
)
```

Arguments

`revenue_lookup_adjustment_factor`
Numeric scalar. Multiplier applied to revenue look ups. (Default = 1)

`unit_structure_discount_factor`
Numeric scalar. Discount factor for unit structure. (Default = 1)

`additive_optional_rate_adjustment_factor`
Numeric scalar. Additive adjustment to optional rates. (Default = 0)

`multiplicative_optional_rate_adjustment_factor`
Numeric scalar. Multiplicative adjustment to optional rates. (Default = 1)

`capped_revenue_add_on_factor`
Numeric scalar. Add-on factor applied to capped revenue. (Default = 0)

`liability_adjustment_factor`
Numeric scalar. Multiplier applied to liability coverage. (Default = 1)

`multiple_commodity_adjustment_factor`
Numeric scalar. Adjustment factor when multiple commodities are insured. (Default = 1)

`reported_acres` Numeric scalar. Number of acres reported for insurance purposes. (Default = 1)

`insured_share_percent`
Numeric scalar. Share of the crop insured (0-1). (Default = 1)

`price_election_percent`
Numeric scalar. Proportion of the elected price used (0-1). (Default = 1)

`damage_area_rate`
Numeric scalar. Rate applied to damage-area calculation.(Default= 1)

`harvest_price_inclusion_plans`
Vector of plan codes that include harvest price in guarantee calculation.

`non_price_risk_plans`
Vector of plan codes that do not adjust revenue based on harvest price.

`rma_rounding` logical(1) or numeric(1) If FALSE, rounds only to integer. Otherwise multiplies the number of digits by this factor (mimics round(x, n * rma_rounding)). Defaults to TRUE.

`yield_ratio_cup_and_cap`
logical(1) If TRUE, enforces a 0.50-1.50 `cup & cap` on yield ratios. Defaults to TRUE.

`continuous_integration_session`
logical(1). If TRUE, a small deterministic subset of the Actuarial Data Master (ADM) YTD ZIP archive is used. This is designed to be safe and fast for use in continuous integration sessions. see `build_min_adm()`

`adm_decoy_state_abb`
tate abbreviation indicating which state's decoy ADM to use (default is ND).

Value

A named list of all control parameters, ready to be passed to other simulation functions.

See Also

Other helpers: `clear_rfcipCalcPass_cache()`

Examples

```
## Not run:
# Use all defaults:
ctrl <- rfcipCalcPass_control()

# Override a couple of factors:
ctrl2 <- rfcipCalcPass_control(
  free_acres_factor = 0.15,
  liability_adjustment_factor = 0.9
)


## End(Not run)
```

Index