

# Package ‘HiddenSafetynet2025’

September 9, 2025

**Type** Package

**Title** HiddenSafetynet2025

**Version** 0.0.0.9000

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

**Maintainer** Francis Tsiboe <[ftsiboe@hotmail.com](mailto:ftsiboe@hotmail.com)>

**Creator** Francis Tsiboe

**Description** Replication Package for Hidden Safety Net of Underutilized Supplemental Insurance in US Agriculture.

**License** GPL-3 + file LICENSE

**URL** <https://github.com/you/HiddenSafetynet2025>

**BugReports** <https://github.com/you/HiddenSafetynet2025/issues>

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Depends** R (>= 4.1.0)

**Imports** data.table, rfcip, stringr, urbnmapr

**Remotes** github::dylan-turner25/rfcip, github::UrbanInstitute/urbnmapr, github::dylan-turner25/rfsa

**Suggests** dplyr, tidyr, knitr, rmarkdown, mockery, withr, testthat (>= 3.0.0)

**LazyData** true

**Cite-us** If you find it useful, please consider starring 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.

R topics documented:

aggregate_expected_outcomes . . . . .	2
build_agent_simulation_data . . . . .	3
build_supplemental_offering_and_adoption . . . . .	4
clean_agents_data . . . . .	5
clean_eco_share_by_insurance_plan . . . . .	6
clean_rma_sco_and_eco_adm . . . . .	6
clean_rma_sobtpu . . . . .	7
clean_sco_share_by_coverage_level . . . . .	8
clean_supplemental_plan_shares . . . . .	8
compute_base_policy_outcomes . . . . .	9
compute_expected_outcomes . . . . .	10
compute_supplemental_current . . . . .	12
compute_supplemental_factors . . . . .	12
compute_supplemental_full . . . . .	13
compute_supplemental_incremental . . . . .	14
dispatcher_supplemental_simulation . . . . .	14
farm_performance_metrics . . . . .	15
setup_environment . . . . .	17
study_scenarios . . . . .	19
<b>Index</b>	<b>20</b>

---

aggregate_expected_outcomes
<i>Aggregate and winsorize expected outcomes (year-level)</i>

---

Description

Loads all per-task expected outcome files for a given year, aggregates them, winsorizes key relative metrics within groups (5th-95th percentiles), and saves a single cleaned file.

Usage

```
aggregate_expected_outcomes(  
  year,  
  expected_directory = NULL,  
  output_directory = NULL,  
  study_environment,  
  agent_identifiers = c("commodity_year", "state_code", "county_code", "type_cod  
  disaggregate = NULL  
)
```

Arguments

- year Integer. Year to aggregate.
- expected\_directory Character or NULL. Directory containing per-task expected\_\*.rds files for the year. If NULL, uses file.path(study\_environment\$wd\$dir\_expected, year).

`output_directory` Character or NULL. Directory to write the aggregated file. If NULL, uses `study_environment$wd$dir_expected`.

`study_environment` List. Must provide `$wd$dir_expected` (and is used to resolve defaults when directories are NULL).

`agent_identifiers` Character vector. Grouping keys used for by-group winsorization (default: `c("commodity_year")`).

`disaggregate` Character or NULL. Optional extra grouping column (e.g., "combination"). If provided but missing, it is created as "ALL".

### Details

Reads all `.rds` files under `expected_directory`, binds them, computes 5th and 95th percentiles for `Relmean`, `Relsd`, `Relcv`, `Rellapv`, `Relrrpv`, `Relnlapv`, `Relnlrpv`, `Relvar` within each group, caps values to that range, and writes `expected_<year>.rds` to `output_directory`.

### Value

Invisibly returns the path to the saved file.

---

`build_agent_simulation_data`  
*Build agent simulation panel*

---

### Description

Read cleaned agent-level simulation data for a crop year, unnest per-draw outcomes, filter to the requested draw(s), compute county-level expected yields, and add per-row revenue.

### Usage

```
build_agent_simulation_data(
  year,
  sim,
  agents_directory = "data/cleaned_agents_data"
)
```

### Arguments

`year` Integer. Crop year.

`sim` Integer vector. Draw number(s) to keep.

`agents_directory` Character. Directory containing cleaned agent data. Default: "data/cleaned\_agents\_data".

## Details

The function:

1. Loads `cleaned_agents_data_<year>.rds` from `agents_directory`.
2. Unnests draw pools: number, farm yield/price, and county yield/price.
3. Filters to `sim` (matching `rma_draw_number`).
4. Renames simulated fields to canonical names and floors negative county yields at zero.
5. Computes a planted-acre-weighted `expected_county_yield`.
6. Computes row-level `revenue = actual_farm_yield * actual_price * planted_acres`.

## Value

A [data.table](#) containing all original columns plus:

- `expected_county_yield`
- `final_county_yield`
- `harvest_price`
- `revenue`

---

```
build_supplemental_offering_and_adoption
```

*Build panel of supplemental insurance availability (offering) and adoption (acres)*

---

## Description

Creates a county-year-commodity panel with availability flags for APH/SCO/ECO90/ECO95 and adoption/acreage measures from RMA SOB/TPU. Availability is sourced from the RMA ADM (A00030\_InsuranceOffer). ECO availability applies starting in 2021.

## Usage

```
build_supplemental_offering_and_adoption(
  cleaned_rma_sobtpu_file_path = "data/cleaned_rma_sobtpu.rds",
  output_directory = "data"
)
```

## Arguments

`cleaned_rma_sobtpu_file_path`

Character. Path to cleaned RMA SOB/TPU RDS. Default: "data/cleaned\_rma\_sobtpu.rds".

`output_directory`

Character. Directory to save output RDS; created if missing. Default: "data".

**Details**

Output columns:

- commodity\_year, state\_code, county\_code, commodity\_code, county\_fips
- avail\_aph, avail\_sco, avail\_eco90, avail\_eco95 (0/1 flags)
- insured\_acres, sco, eco90, eco95 (adopted acres)

Availability aggregation uses max() (binary). Acreage aggregation uses sum(). Missing numeric values are replaced with 0.

**Value**

Invisibly returns the output file path. Also prints a brief summary.

**Examples**

```
## Not run:
path <- build_supplemental_offering_and_adoption()
readRDS(path)[1:5]

## End(Not run)
```

---

clean\_agents\_data    *Clean agent-level data for a given year*

---

**Description**

Downloads, merges, and processes agent-level insurance data for the specified year. Combines revenue draws, calibrated yields, and RMA reference data, computes premium/subsidy measures, and saves the cleaned dataset as an RDS file.

**Usage**

```
clean_agents_data(
  year,
  cleaned_rma_sobtpu_file_path = "data/cleaned_rma_sobtpu.rds",
  cleaned_rma_sco_and_eco_adm_file_path = "data/cleaned_rma_sco_and_eco_adm.rds",
  output_directory = "data/cleaned_agents_data"
)
```

**Arguments**

**year**                    Integer. Commodity year to process (e.g., 2015).

**cleaned\_rma\_sobtpu\_file\_path**                    Path to cleaned RMA SOB/TPU RDS file. Default: "data-raw/data/cleaned\_rma\_sobtpu.rds"

**cleaned\_rma\_sco\_and\_eco\_adm\_file\_path**                    Path to cleaned RMA SCO & ECO admin RDS file. Default: "data-raw/data/cleaned\_rma\_sco\_and\_eco\_adm.rds"

**output\_directory**                    Directory to save output RDS file. Created if missing. Default: "data/cleaned\_agents\_data"

**Value**

Returns the input `year` on success, with attributes for `save_path` and number of rows. Returns `NULL` on error.

**Note**

Requires **data.table**, access to GitHub-hosted RDS files, and the helper function `get_compressed_adm()`.

---

```
clean_eco_share_by_insurance_plan
      ECO share by insurance plan
```

---

**Description**

Computes Enhanced Coverage Option (ECO) **shares of insured acres** for base plans (1=YP, 2=RP, 3=RP-HPE), using ECO plan codes 87-89 mapped back to 1-3, and returns separate shares for ECO-90 and ECO-95.

**Usage**

```
clean_eco_share_by_insurance_plan(sob)
```

**Arguments**

<code>sob</code>	A <code>data.table</code> (or <code>data.frame</code> ) of SOB records that already contain aggregated acre and dollar fields. Must include at least: <code>insured_acres</code> , <code>endorsed_acres</code> , <code>insurance_plan_code</code> , <code>coverage_level_percent</code> , <code>commodity_year</code> , <code>state_code</code> , <code>county_code</code> , <code>commodity_code</code> , <code>type_code</code> , <code>practice_code</code> .
------------------	--

**Details**

- Base data are summed for plans 1 : 3.
- ECO records are selected via plan codes 87 : 89, remapped to 1 : 3 (subtract 86), and coverage levels are labeled as "eco90" / "eco95" (using `coverage_level_percent * 100`).
- ECO shares are computed as `eco_endorsed_acres / insured_acres` by key.

**Value**

A `data.table` with unique rows by `commodity_year`, `state_code`, `county_code`, `commodity_code` and columns: `eco90`, `eco95` (shares in [0,1]); missing combinations may be NA.

---

```
clean_rma_sco_and_eco_adm
```

*Build SCO/ECO/Area ADM table for a given year (adds SCO88/SCO90)*

---

### Description

Downloads yearly ADM fragments from GitHub Releases for *Supplemental SCO*, *Supplemental ECO*, and *Area* plans, aggregates key parameters by common grouping keys, linearly interpolates SCO rates to 88% and 90% (using AYP and, for years  $\geq 2021$ , ECO anchors), and returns the cleaned, stacked table.

### Usage

```
clean_rma_sco_and_eco_adm(year)
```

### Arguments

`year` Integer. commodity year (e.g., 2022).

### Value

A [data.table](#) containing original SCO/ECO/Area ADM rows plus synthesized **SCO88** (`insurance_plan_code + 10`) and **SCO90** (`insurance_plan_code + 20`) rows with non-invalid `base_rate`.

### Note

Requires internet access. Missing plan files for a year are skipped silently.

---

```
clean_rma_sobtpu
```

*Clean and aggregate RMA Summary of Business (SOB-TPU) data*

---

### Description

Retrieves RMA SOB-TPU records for requested years, combining **live** years (last 5 years, fetched via `rforcip::get_sob_data()`) with **stable** years (downloaded from a prebuilt `.rds` release), then filters, harmonizes insurance plan codes, coverage levels, and unit structure codes, and returns an analysis-ready `data.table` aggregated to common keys.

### Usage

```
clean_rma_sobtpu(
  years = as.numeric(format(Sys.Date(), "%Y")) - 1,
  insurance_plan = NULL,
  acres_only = TRUE,
  addon_only = TRUE,
  harmonize_insurance_plan_code = TRUE,
  harmonize_coverage_level_percent = TRUE,
  harmonize_unit_structure_code = TRUE
)
```

**Arguments**

years	Integer vector of commodity years.
insurance_plan	Optional integer vector of harmonized plan codes to keep after harmonization (1=YP, 2=RP, 3=RP-HPE). If NULL, keep all.
acres_only	Logical; keep only acres-level records. Default TRUE.
addon_only	Logical; exclude CAT (coverage_type_code == "C"). Default TRUE.
harmonize_insurance_plan_code	Logical; recode plans to (1,2,3). Default TRUE.
harmonize_coverage_level_percent	Logical; normalize coverage levels to decimal in 0.50 to 0.95 at 0.05 steps. Default TRUE.
harmonize_unit_structure_code	Logical; recode unit structure to (OU, BU, and EU). Default TRUE.

**Value**

A `data.table` aggregated to the keys with columns: `insured_acres`, `endorsed_acres`, `liability_amount`, `total_premium_amount`, `subsidy_amount`, `indemnity_amount`.

---

```
clean_sco_share_by_coverage_level
```

*SCO share by coverage level*

---

**Description**

Computes the Supplemental Coverage Option (SCO) **share of insured acres** by coverage level for base plans (1=YP, 2=RP, 3=RP-HPE), using SCO plan codes 31-33 mapped back to 1-3.

**Usage**

```
clean_sco_share_by_coverage_level(sob)
```

**Arguments**

sob	A <code>data.table</code> (or <code>data.frame</code> ) of SOB records that already contain aggregated acre and dollar fields. Must include at least: <code>insured_acres</code> , <code>endorsed_acres</code> , <code>insurance_plan_code</code> , <code>coverage_level_percent</code> , <code>commodity_year</code> , <code>state_code</code> , <code>county_code</code> , <code>commodity_code</code> , <code>type_code</code> , <code>practice_code</code> .
-----	--

**Details**

- Base data are summed for plans 1 : 3.
- SCO records are selected via plan codes 31 : 33 and then remapped to 1 : 3 (subtract 30) to align with corresponding base plans.
- SCO share is computed as `endorsed_acres / insured_acres` by key and coverage level.



**Value**

A `data.table` with unique rows by `commodity_year`, `state_code`, `county_code`, `commodity_code` and column: `sco` (share in [0,1]).

---

```
clean_supplemental_plan_shares
      Supplemental plan shares (SCO/ECO)
```

---

**Description**

Aggregates base plan acres/dollars and (optionally) merges in Supplemental Coverage Option (SCO) and Enhanced Coverage Option (ECO) shares for each key.

**Usage**

```
clean_supplemental_plan_shares(
  sob,
  get_sco_shares = TRUE,
  get_eco_shares = TRUE
)
```

**Arguments**

<code>sob</code>	A <code>data.table</code> or <code>data.frame</code> of SOB records. Expected columns: <code>insured_acres</code> , <code>endorsed_acres</code> , <code>insurance_plan_code</code> , <code>coverage_level_percent</code> , <code>commodity_year</code> , <code>state_code</code> , <code>county_code</code> , <code>commodity_code</code> , <code>type_code</code> , <code>practice_code</code> , <code>unit_structure_code</code> , <code>coverage_type_code</code> , <code>liability_amount</code> , <code>total_premium_amount</code> , <code>subsidy_amount</code> , <code>indemnity_amount</code> .
<code>get_sco_shares</code>	Logical; if TRUE, merge SCO shares by coverage level.
<code>get_eco_shares</code>	Logical; if TRUE, merge ECO-90/95 shares by plan.

**Details**

Assumes base plans are harmonized to 1=YP, 2=RP, 3=RP-HPE, and `coverage_level_percent` is in decimals (e.g., 0.90, 0.95). Relies on helpers: `clean_sco_share_by_coverage_level()` and `clean_eco_share_by_insurance_plan()`.

**Value**

A `data.frame` with aggregated acres/dollars and columns `sco`, `eco90`, `eco95` (shares in [0,1] where available).

---

```
compute_base_policy_outcomes
```

*Compute base-policy outcomes*

---

### Description

Vectorized **data.table** implementation of base-policy guarantees, acres/liability, premium pieces (total/subsidy/producer), and indemnity, plus a tidy column subset for downstream joins.

### Usage

```
compute_base_policy_outcomes(cleaned_agents_data)
```

### Arguments

```
cleaned_agents_data
```

A `data.frame` or `data.table` with the required columns (see error message if any are missing).

### Details

Requires a set of core inputs (e.g., yields, prices, coverages, acres) and returns the standard monetary outputs for each policy row. Price risk is handled via a `new_insurance_guarantee` that depends on plan code.

### Value

A `data.table` with key fields and outputs: `insured_acres`, `liability`, `total_premium`, `subsidy_amount`, `producer_premium`, `indemnity`, `revenue`, and supporting fields such as `harvest_price`, `expected_county_yield`, `final_county_yield`, `new_insurance_guarantee`, `projected_price`.

---

```
compute_expected_outcomes
```

*Compute expected outcomes and risk metrics from simulation outputs*

---

### Description

Joins cleaned agent records to simulation files, then computes expected (mean/sd) revenues, downside-risk measures (loss-side residual moments), relative improvements with insurance, and insurance performance statistics. Writes a single `.rds` result file and returns its path (invisibly).

### Usage

```
compute_expected_outcomes(
  year,
  task_id,
  agents_directory = "data/cleaned_agents_data",
  simulation_directory = NULL,
  output_directory = NULL,
```

```

study_environment,
agent_identifiers = c("commodity_year", "state_code", "county_code", "commodity_
  "type_code", "practice_code", "unit_structure_code", "insurance_plan_code",
  "coverage_level_percent", "insured_acres"),
disaggregate = NULL
)

```

## Arguments

<code>year</code>	Integer (scalar). Analysis year (used to resolve input/output paths).
<code>task_id</code>	Integer or integer vector. Pseudo-task partition(s) to keep; the function cycles a 1..500 index over agent rows and filters to these values.
<code>agents_directory</code>	Character. Directory containing <code>cleaned_agents_data_&lt;year&gt;.rds</code> .
<code>simulation_directory</code>	Character or NULL. Directory with simulation <code>.rds</code> files; default is <code>file.path(study_environment\$wd, year)</code> .
<code>output_directory</code>	Character or NULL. Directory to write results; default is <code>file.path(study_environment\$wd, year)</code> .
<code>study_environment</code>	List. Must include <code>wd\$dir_sim</code> and <code>wd\$dir_expected</code> if the corresponding directory arguments are NULL.
<code>agent_identifiers</code>	Character vector. Columns that identify agent units and define aggregation groups (used for joins and <code>by</code> ); default includes year, location, crop, unit structure, plan, coverage, and acres.
<code>disaggregate</code>	Character or NULL. Optional extra column to disaggregate by (for example, "combination"). If provided but missing after the join, the column is created and set to "ALL".

## Details

### Pipeline

1. Load agent data and keep only `agent_identifiers`; coerce to `data.table`.
2. Assign a pseudo task (cycles 1..500), then filter to `task_id`.
3. Guardrails:
  - Stop if no simulation files are found.
  - Stop if the combined join yields zero rows.
  - Validate required numeric columns: `revenue`, `indemnity`, `producer_premium`, `liability`, `total_premium`, `subsidy_amount`.
  - Use `safe_div()` to avoid Inf/NaN on zero or non-finite denominators.
4. Compute revenues (floored at 0): `Revenue` and `Revenue_Inc` (= `revenue` + `indemnity` + `producer premium`).
5. By `uid` (= `agent_identifiers` plus `disaggregate` if provided), compute means, sds, residual-based downside measures (loss-only squared residuals and their frequency), and derived statistics (variance, CV, LAPV, LRPV, normalized forms).

6. Compute **relative** metrics (insured vs. uninsured ratios): Relmean, Relsd, Relcv, Rellapv, Rellrpv, Relnlpv, Relnlrpv, Relvar. **Base Revenue\*** statistics are dropped before the final merge to keep results compact.
7. Aggregate insurance performance by group: mean liability, total\_premium, subsidy\_amount, producer\_premium, indemnity, premium and LCR rates (Simrate, SimrateP, Simsuby, Simlcr), and **group sums** for lr\_indemnity and lr\_premium. Merge with the relative metrics.

**Join note** The join uses `data[simdt, on = <keys>, nomatch = 0]`, i.e., it returns rows aligned to the simulation table entries that match the agent keys.

### Value

Invisibly returns the saved file path (`expected_<year>_<task-range>.rds`).

---

```
compute_supplemental_current
```

*Aggregate supplemental results for the current environment*

---

### Description

Scale selected SCO/ECO factors by base-policy weights (`sco`, `eco90`, `eco95`), aggregate by policy keys, append base outcomes, and label the rollup as "Basic+CURRENT".

### Usage

```
compute_supplemental_current(base_policy_data, supplemental_factors)
```

### Arguments

`base_policy_data`

[data.table](#). Base-policy outcomes (contains keys, weights, and monetary fields).

`supplemental_factors`

[data.table](#). Supplemental outcomes from `compute_supplemental_factors` including `sup`.

### Value

A [data.table](#) aggregated by policy keys with: `revenue`, `liability`, `total_premium`, `subsidy_amount`, `producer_premium`, `indemnity`, and `combination`.

---

```
compute_supplemental_factors
```

*Compute supplemental policy factors (SCO/ECO)*

---

### Description

Compute shallow-loss protection, premiums, and indemnities for one SCO/ECO endorsement offering, aligning plan families and joining ADM rating inputs.

### Usage

```
compute_supplemental_factors(base_policy, adm, plan, subsidy, trigger)
```

### Arguments

<code>base_policy</code>	<a href="#">data.table</a> . Base-policy rows (keys, yields, prices, liability, etc.).
<code>adm</code>	<a href="#">data.table</a> . Rating inputs with <code>base_rate</code> and join keys.
<code>plan</code>	Integer. Plan code in the offering (e.g., 31-33, 51-53, 87-89).
<code>subsidy</code>	Numeric. Subsidy factor (e.g., 0.65, 0.80, 0.44).
<code>trigger</code>	Numeric. Coverage trigger level (e.g., 0.86, 0.90, 0.95).

### Details

Handles plan families via offsets (31-33, 41-43, 51-53, 87-89). For plans 87-89 (ECO), the `coverage_level_percent` for ADM is matched to the `trigger` (with a small tolerance), and the subsidy factor special-case is applied for underlying plan code 1. Emits a standard `sup` label like "SCO8665" or "ECO9544".

### Value

A [data.table](#) with columns: `commodity_year`, `state_code`, `county_code`, `commodity_code`, `type_code`, `practice_code`, `unit_structure_code`, `insurance_plan_code`, `coverage_level_percent`, `liability`, `total_premium`, `subsidy_amount`, `producer_premium`, `indemnity`, `sup`.

---

```
compute_supplemental_full
```

*Aggregate supplemental full-participation results*

---

### Description

Given selected `sup` labels, sum their monetary fields, append base outcomes, and produce a final rollup by policy keys with a descriptive `combination` label.

### Usage

```
compute_supplemental_full(
  base_policy_data,
  supplemental_factors,
  supplemental_pick
)
```

**Arguments**

`base_policy_data`  
[data.table](#). Base-policy outcomes.

`supplemental_factors`  
[data.table](#). Results from `compute_supplemental_factors`.

`supplemental_pick`  
 Character vector of sup labels to include.

**Details**

The function self-filters `supplemental_factors` to the provided `supplemental_pick` (after dropping empties), aggregates within keys, appends base outcomes, and re-aggregates.

**Value**

A [data.table](#) aggregated by the policy keys with: revenue, liability, total\_premium, subsidy\_amount, producer\_premium, indemnity, and combination.

---

```
compute_supplemental_incremental
```

*Compute incremental supplemental results at an adoption rate*

---

**Description**

Build an incremental scenario by scaling SC08665 supplemental dollars by a user-specified adoption rate, aggregating by keys, and appending base outcomes.

**Usage**

```
compute_supplemental_incremental(  
  base_policy_data,  
  supplemental_factors,  
  adoption_rate  
)
```

**Arguments**

`base_policy_data`  
[data.table](#). Base-policy outcomes.

`supplemental_factors`  
[data.table](#). Output from `compute_supplemental_factors` filtered to `sup == "SC08665"`.

`adoption_rate`  
 Numeric. Percentage (e.g., 10 for 10\ scale incremental supplemental amounts.

**Value**

A [data.table](#) aggregated by the policy keys with: revenue, liability, total\_premium, subsidy\_amount, producer\_premium, indemnity, and combination.

---

dispatcher\_supplemental\_simulation

*Dispatcher: simulate supplemental outcomes for one draw*


---

## Description

Orchestrate the full supplemental simulation workflow for a given crop year and draw: build the agent panel, compute base-policy results, generate supplemental factors, assemble *Current*, *Full*, and *Incremental* scenarios, and write the combined results to disk.

## Usage

```
dispatcher_supplemental_simulation(
  sim,
  year,
  agents_directory = "data/cleaned_agents_data",
  cleaned_rma_sco_and_eco_adm_file_path = "data/cleaned_rma_sco_and_eco_adm.rds",
  output_directory = NULL
)
```

## Arguments

<code>sim</code>	Integer. Draw number used in data building and the filename.
<code>year</code>	Integer. Crop year.
<code>agents_directory</code>	Character. Directory for cleaned agents data.
<code>cleaned_rma_sco_and_eco_adm_file_path</code>	Character. Path to RDS of SCO/ECO ADM with join keys and <code>base_rate</code> . Default: "data/cleaned_rma_sco_and_eco_adm.rds".
<code>output_directory</code>	Character or NULL. Where to write results; see Details for default behavior.

## Details

The pipeline:

1. `build_agent_simulation_data` to construct the panel.
2. `compute_base_policy_outcomes` for base outcomes.
3. `study_scenarios` to enumerate offerings/mixes.
4. Load SCO/ECO ADM; filter to `commodity_year == year`; average `base_rate` by key; drop invalid/zero rates.
5. Loop offerings through `compute_supplemental_factors`.
6. Build scenarios:
  - *Current*: `compute_supplemental_current`.
  - *Full*: `compute_supplemental_full`.
  - *Incremental*: `compute_supplemental_incremental`.
7. Aggregate base-only results, `rbind` all scenarios, and save as `simXXX.rds` in `output_directory`.

If `output_directory` is NULL, it defaults to `file.path(study_environment$wd$dir_sim, year)` (ensure `study_environment$wd$dir_sim` exists in the calling environment).

**Value**

Invisibly writes `simXXX.rds` to `output_directory`.

---

```
farm_performance_metrics
```

*Farm performance metrics by scenario and disaggregate*

---

**Description**

Load `expected_<year>.rds`, derive outcome variables, compute deltas vs. baselines, trim extremes using quantile limits, aggregate (weighted mean/median) by requested disaggregates, and save a summarized `.rds`. Returns the saved path invisibly.

**Usage**

```
farm_performance_metrics(
  year,
  agent_identifiers = c("commodity_year", "state_code", "county_code", "commodity_year",
    "type_code", "practice_code", "unit_structure_code", "insurance_plan_code",
    "coverage_level_percent"),
  outcome_list = c("its", "Iits", "rrs1", "rrs2", "rrs3", "Irrs1", "Irrs2", "Irrs3",
    "sner1", "sner2", "sner3", "Simrate", "SimrateP", "Simsuby", "Simlcr", "rrp1",
    "rrp2", "rrp3", "itp"),
  combo,
  weight_variable = NULL,
  expected_directory = NULL,
  draw = NULL,
  draw_list_file_path = NULL,
  disaggregates = NULL,
  output_file_path = NULL,
  distributional_limits = c(0.05, 0.95)
)
```

**Arguments**

<code>year</code>	Policy year used to locate <code>expected_&lt;year&gt;.rds</code> .
<code>agent_identifiers</code>	Character vector of ID columns for grouping prior to long-pivot and averaging.
<code>outcome_list</code>	Character vector of outcome columns to reshape and aggregate.
<code>combo</code>	Target scenario (e.g., "Basic+CURRENT", "Basic+SC08665", or another).
<code>weight_variable</code>	NULL for equal weights (=1) or a character name of a numeric weight column.
<code>expected_directory</code>	Directory containing <code>expected_&lt;year&gt;.rds</code> .
<code>draw</code>	Optional draw identifier used for filtering and filename tag.
<code>draw_list_file_path</code>	Optional path to an RDS (named list) with the draw table; required if <code>draw</code> is not NULL.



disaggregates

Optional character vector of additional disaggregate columns (alongside "FCIP").

output\_file\_path

Output file path

distributional\_limits

Numeric length-2 vector of lower/upper probabilities (e.g., `c(0.05, 0.95)`); must satisfy  $0 < p1 < p2 < 1$ .

## Details

### Steps:

1. Filter rows to combination `%in% {"Basic+CURRENT", combo, "Basic+SCO8665"}`.
2. Create derived metrics: `rrs1/2/3`, `its`, `flags` `Irrs*/Iits`, `sner*`, `percent/level` transforms (`rrp*`, `itp`), and scale `Sim*` by 100.
3. Reshape to long on `outcome_list`, drop non-finite values, average within identifiers (`agent_identifiers`, and `weight_variable` if provided), `scenario`, `variable`.
4. Join baselines: if `combo != "Basic+CURRENT"`, add "Basic+CURRENT" as `base00`; if `combo = {"Basic+SCO8665", "Basic+CURRENT"}`, add "Basic+SCO8665" as `base01`. Compute `chglv100/01` and `chgpct00/01` (guard divide-by-zero).
5. Build labels `PLAN`, `RPYP`, `COV`, `STRUCT`.
6. Compute trimming limits per (`variable`, `combination`, `state_code`, `IRR`, `commodity_code`) using `distributional_limits` (default `c(0.05, 0.95)`), require `n` greater or equal to 20, and cap to `*T` columns.
7. For each of `c("FCIP", disaggregates)`, compute weighted mean and weighted median of raw and trimmed metrics; stack results and write output.

## Value

Invisibly returns the character path of the saved `.rds`.

## Required columns

All `agent_identifiers`, plus: `combination`, `state_code`, `county_code`, `commodity_code`, `type_code`, `practice_code`, `IRR`, `Relcv`, `Relnlrvp`, `Relnlapv`, `Relmean`, `Simrate`, `SimrateP`, `Simsuby`, `Simlcr`, `coverage_level_percent`, `unit_structure_code`, `insurance_plan_code`. If `weight_variable` is not `NULL`, that column must exist and be numeric.

## Note

Baseline joins use `nomatch = 0` by design, so rows missing in the baseline are dropped before delta computation. Change to `nomatch = NA` if you prefer to retain such rows with NA deltas.

## See Also

`data.table::data.table`, `data.table::melt`, `matrixStats::weightedMedian`

---

 setup\_environment    *Setup Project Environment*


---

## Description

Initializes the working environment for a project by creating required directories, setting useful global options, and fixing the random seed.

## Usage

```
setup_environment(
  year_beg = 2001,
  year_end = as.numeric(format(Sys.Date(), "%Y")),
  seed = 1980632,
  project_name,
  local_directories = list(file.path("data-raw", "output"), file.path("data-raw",
    "scripts"), file.path("data")),
  fastscratch_root = NULL,
  fastscratch_directories = NULL
)
```

## Arguments

<code>year_beg</code>	Integer. Beginning year of the analysis (default: 2001).
<code>year_end</code>	Integer. Ending year of the analysis (default: current system year).
<code>seed</code>	Integer. Random seed for reproducibility (default: 1980632).
<code>project_name</code>	Character. Project name (required). Used to build fast-scratch directory paths.
<code>local_directories</code>	List of project-local directories to create (default: <code>list("data-raw/output", "data-raw/scripts", "data")</code> ).
<code>fastscratch_root</code>	Optional character. Root directory for fast-scratch files. If <code>NULL</code> , it is set automatically: <ul style="list-style-type: none"> <li>• Windows: <code>"C:/fastscratch"</code></li> <li>• Linux/macOS: <code>"/fastscratch/&lt;username&gt;"</code></li> </ul>
<code>fastscratch_directories</code>	List of fast-scratch subdirectories (relative to <code>&lt;fastscratch_root&gt;/&lt;project_name&gt;</code> ) to create. If <code>NULL</code> , no fast-scratch subdirectories are created and <code>wd</code> is returned as an empty list.

## Details

The function ensures the requested directories exist, creating them if necessary. Directory keys in the returned `wd` list are the basenames of the provided `fastscratch_directories`.

It also sets the following options:

- `options(scipen = 999)` (turns off scientific notation)
- `options(future.globals.maxSize = 8 * 1024^3)` (~8 GiB)
- `options(dplyr.summarise.inform = FALSE)` (quiet **dplyr**)

Finally, the random number generator is seeded with the provided `seed`.

**Value**

A list with:

**wd** Named list of created fast-scratch directories. Empty if `fastscratch_directories = NULL`.

**year\_beg** Starting year (integer).

**year\_end** Ending year (integer).

**seed** Seed value used for RNG.

**Examples**

```
## Not run:
env <- setup_environment(
  year_beg = 2015,
  year_end = 2024,
  project_name = "HiddenSafetyNet2025",
  fastscratch_directories = c("output/sims", "output/expected")
)
str(env$wd)

## End(Not run)
```

---

study_scenarios	<i>Build study scenarios (SCO/ECO offerings and mixes)</i>
-----------------	--

---

**Description**

Define the endorsement offerings (plan family - trigger - subsidy - label) and the full-participation SCO/ECO mixes to evaluate for a given year.

**Usage**

```
study_scenarios(year)
```

**Arguments**

**year** Integer. Crop year used to determine available ECO variants.

**Details**

For years  $\geq 2021$ , ECO 90/44 and 95/44 variants are added and the participation set is expanded accordingly. Offerings create sup labels such as "SCO8665", "SCO9080", "ECO9044", "ECO9544".

**Value**

A named list with:

- offerings: [data.table](#) of insurance\_plan\_code, Trigger, plan, Subsidy\_factor.
- full\_participation: [data.table](#) of SCO/ECO label combinations to test (columns sco, eco).

# Index

`aggregate_expected_outcomes`, [2](#)

`build_agent_simulation_data`, [3](#), [15](#)

`build_supplemental_offering_and_adoption`,  
[4](#)

`clean_agents_data`, [5](#)

`clean_eco_share_by_insurance_plan`,  
[6](#)

`clean_rma_sco_and_eco_adm`, [6](#)

`clean_rma_sobtpu`, [7](#)

`clean_sco_share_by_coverage_level`,  
[8](#)

`clean_supplemental_plan_shares`,  
[8](#)

`compute_base_policy_outcomes`, [9](#),  
[15](#)

`compute_expected_outcomes`, [10](#)

`compute_supplemental_current`, [12](#),  
[15](#)

`compute_supplemental_factors`, [12](#),  
[12](#), [13–15](#)

`compute_supplemental_full`, [13](#), [15](#)

`compute_supplemental_incremental`,  
[14](#), [15](#)

`data.table`, [4](#), [7](#), [10](#), [12–14](#), [19](#)

`dispatcher_supplemental_simulation`,  
[14](#)

`farm_performance_metrics`, [15](#)

`rfcip::get_sob_data()`, [7](#)

`setup_environment`, [17](#)

`study_scenarios`, [15](#), [19](#)