# Package 'HiddenSafetynet2025'

September 7, 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>

**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 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.

# R topics documented:



---

`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` |
| `cleaned_rma_sco_and_eco_adm_file_path` | Path to cleaned RMA SCO & ECO admin RDS file. Default: `"data-raw/data/cleaned_rma_` |
| `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_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 >= 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 enrich RMA Summary of Business (SOB) data*

## Description

Processes RMA Summary of Business (SOB) data to produce an analysis-ready dataset with aggregated core insurance metrics and **shares** of Supplemental Coverage Option (SCO) and Enhanced Coverage Option (ECO) by coverage level.

## Usage

```
clean_rma_sobtpu(study_env = setup_environment(), output_directory = "data")
```

## Arguments

| | |
|---|---|
| `study_env` | A list-like environment produced by `setup_environment()` that must include `year_beg` and `year_end` (inclusive integers). Defaults to `setup_environment()`. |
| `output_directory` | Character string specifying the directory where the processed `.rds` file should be saved. Defaults to `"data"`. The file will be named `"cleaned_rma_sobtpu.rds"`. |

### Details

The output file will be written to `file.path(output_directory, "cleaned_rma_sobtpu.rds")`. The directory is created if it does not exist.

### Value

A character message describing the processed year range and number of output rows; the main side effect is writing an `.rds` file to disk.

---

`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", "commodit
    "type_code", "practice_code", "unit_structure_code", "insurance_plan_code",
    "coverage_level_percent", "insured_acres"),
  disaggregate = NULL
)
```

## Arguments

| | |
|---|---|
| `year` | Integer (scalar). Analysis year (used to resolve input/output paths). |
| `task_id` | 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. |
| `agents_directory` | Character. Directory containing `cleaned_agents_data_<year>.rds`. |
| `simulation_directory` | Character or `NULL`. Directory with simulation `.rds` files; default is `file.path(study_enviro year)`. |
| `output_directory` | Character or `NULL`. Directory to write results; default is `file.path(study_environment$wd year)`. |
| `study_environment` | List. Must include `wd$dir_sim` and `wd$dir_expected` if the corresponding directory arguments are `NULL`. |
| `agent_identifiers` | Character vector. Columns that identify agent units and define aggregation groups (used for joins and `by`); default includes year, location, crop, unit structure, plan, coverage, and acres. |
| `disaggregate` | 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`, `Relnlapv`, `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

| | |
|---|---|
| `base_policy` | data.table. Base-policy rows (keys, yields, prices, liability, etc.). |
| `adm` | data.table. Rating inputs with `base_rate` and join keys. |
| `plan` | Integer. Plan code in the offering (e.g., 31-33, 51-53, 87-89). |
| `subsidy` | Numeric. Subsidy factor (e.g., `0.65`, `0.80`, `0.44`). |
| `trigger` | 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_perce` 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_pe` `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 `SCO8665` 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 == "SCO8665". |
| 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

| | |
|---|---|
| sim | Integer. Draw number used in data building and the filename. |
| year | Integer. Crop year. |
| agents_directory | Character. Directory for cleaned agents data. |
| cleaned_rma_sco_and_eco_adm_file_path | Character. Path to RDS of SCO/ECO ADM with join keys and base_rate. Default: "data/cleaned_rma_sco_and_eco_adm.rds". |

output_directory
: 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.

---

# setup_environment *Setup Project Environment*

---

## Description

Loads project settings, creates working directories (both under a fast scratch area and in the project), sets useful options(), fixes the RNG seed, and stores the analysis year range.

## Usage

```
setup_environment(
  year_beg = 2015,
  year_end = 2024,
  seed = 1980632,
  fastscratch_root = NULL
)
```

### Arguments

| | |
|---|---|
| year_beg | Integer. Beginning year of the analysis (default: 2015). |
| year_end | Integer. Ending year of the analysis (default: 2024). |
| seed | Integer. Random seed for reproducibility (default: 1980632). |
| fastscratch_root | Optional character. Root directory where intermediate files from simulations and estimations will be written for later aggregation. If NULL, it is set automatically based on the operating system:<br>• Windows: "C:/fastscratch"<br>• Linux/macOS: "/fastscratch/<username>" |

### Details

Creates these directories (if absent):

- Fast scratch tree (for large, intermediate outputs): <fastscratch_root>/HiddenSafetynet2025/outpu with subfolders sims, expected, draw_farm, draw_cost.
- Project-local (for smaller, version-controlled artifacts): data/, data/output/, data/cleaned_agents_d

Sets:

- options(scipen = 999)
- options(future.globals.maxSize = 8 * 1024^3) (= 8 GiB)
- options(dplyr.summarise.inform = FALSE)
- set.seed(seed)

Requires the packages **future.apply**, **rfcip**, **data.table**, and **rfcipCalcPass**.

### Value

A list with:

**wd** Named list of working directories (fastscratch root and subfolders).

**year_beg** Starting year (integer).

**year_end** Ending year (integer).

---

## study_scenarios *Build study scenarios (SCO/ECO offerings and mixes)*

---

### 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 >= 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