Package ‘USFarmSafetyNetLab’

November 20, 2025

Type Package

Title US Farm Safety Net Lab

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 This repository centralizes research outputs, analytical tools, and resources for exploring and evaluating the United States agricultural safety net programs. It supports analysis of key programs including the Federal Crop Insurance Program (FCIP), the Noninsured Crop Disaster Assistance Program (NAP), Price Loss Coverage (PLC), Agricultural Risk Coverage (ARC), and various ad-hoc disaster assistance programs.

License GPL-3 + file LICENSE

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

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports readr, purrr, data.table, doBy, janitor, plm, sp, spdep, stringr, terra, tidyr, tigris, xml2

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

Suggests dplyr, rvest, knitr, rmarkdown, rfcip, withr, piggyback, testthat (>= 3.0.0)

LazyData true

LazyDataCompression gzip

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

`agCensusAcres` *agCensusAcres*

Description

A combined dataset for agCensusAcres

Usage

```
data(agCensusAcres)
```

Format

A data frame with 15369 rows and 15 columns covering Inf–Inf.

Source

USDA NASS Quick Stats

`agCensusBFR` *agCensusBFR*

Description

A combined dataset for agCensusBFR

Usage

```
data(agCensusBFR)
```

Format

A data frame with 255 rows and 16 columns covering Inf–Inf.

Source

USDA NASS Quick Stats

`agCensusInsurance` *agCensusInsurance*

Description

A combined dataset for agCensusInsurance

Usage

```
data(agCensusInsurance)
```

Format

A data frame with 329783 rows and 10 columns covering Inf–Inf.

Source

USDA NASS Quick Stats

`calculate_mode` *Calculate the Statistical Mode*

Description

Returns the element that occurs most frequently in a vector.

Usage

```
calculate_mode(x, na.rm = TRUE)
```

Arguments

`x`	A vector of any atomic type (numeric, character, factor,).
`na.rm`	Logical; should missing values be ignored? Defaults to `TRUE`. If `FALSE` and `x` contains any `NA`s, the function returns `NA`.

Details

Internally the function:

1. Optionally removes `NA`s (`na.rm` = `TRUE`).
2. Builds a lookup table of unique values via `unique(x)`.
3. Counts the frequency of each unique value with `tabulate(match(x, ux))`.
4. Returns the value with the maximum count.

Because it relies on base R functions, the implementation is vectorised and generally fast for typical data-frame column sizes.

Value

A single value giving the modal element of `x`. If two or more values are tied for the highest frequency, the first one encountered in `x` is returned.

clean_data *Apply standardized data cleaning opperations*

Description

Apply standardized data cleaning opperations

Usage

```
clean_data(df)
```

Arguments

df	a data frame to clean

Value

a cleaned data frame

Source

copied from https://github.com/dylan-turner25/rmaADM/blob/main/R/helpers.R

Examples

```
## Not run: clean_data(df)
```

downloaded_nass_large_datasets *Download and cache USDA NASS Quick Stats large dataset files*

Description

`downloaded_nass_large_datasets()` retrieves a Quick Stats file from the USDA National Agricultural Statistics Service (NASS) https://www.nass.usda.gov/datasets/ page and saves it locally. If the file is already present in the target directory, it is not re-downloaded.

Usage

```
downloaded_nass_large_datasets(
  large_datasets,
  dir_dest = "./data-raw/fastscratch/nass/"
)
```

Arguments

`large_datasets`	`character list` The base name of the Quick Stats file to download. For example, use `"crops"` to fetch `qs.crops_YYYYMMDD.txt.gz` or include `"census2022"` (e.g. `"census2022"`) to fetch the gzipped 2022 census version (`qs.census2022.txt.gz`). any of: "census2002","census2007","census2012","census2017","census2022", "census2007zipcode","census2017zipcode", "animals_products","crops","demographics","economic
`dir_dest`	`character(1)` Path to a directory where downloaded files will be stored. Defaults to `"./data-raw/fastscratch/nass/"`.

Details

1. Prepends `"qs."` to the provided `large_dataset`. If `large_dataset` contains `"census"`, appends `".txt.gz"`, otherwise `NULL`.
2. Ensures `dir_dest` exists (creates it if needed).
3. Scrapes the NASS datasets page (`https://www.nass.usda.gov/datasets/`) for links ending in `.txt.gz`.
4. Downloads the matching file into `dir_dest` if not already present.

Value

Invisibly returns the normalized file large_dataset (e.g. `"qs.crops_YYYYMMDD.txt.gz"` or `"qs.censusYYYY.txt.gz"`) that was downloaded or already present.

See Also

Other USDA NASS Quick Stats: `get_nass_census_data()`, `process_nass_dataset()`

Examples

```
## Not run:
# Download the 'crops' dataset if not already cached:
downloaded_nass_large_datasets(large_dataset = "crops")

# Download the 2022 census version:
downloaded_nass_large_datasets(large_dataset = "census2022",
dir_dest = "./data-raw/fastscratch/nass/")

## End(Not run)
```

`download_rma_web_data_files` *Download and Process RMA Web Data Files*

Description

Fetches one or more years of USDA RMA Summary of Business data (state-county-crop & livestock participation series), unzips, reads the pipe-delimited text, applies the correct column names, and saves each year as an `.rds` under `dest`.

Usage

```
download_rma_web_data_files(
  years,
  file_name,
  dest = "./data-raw/data_release",
  url_rma_ftp_file_access = "https://pubfs-rma.fpac.usda.gov/pub"
)
```

Arguments

`years`	Integer vector of crop or livestock years to fetch.
`file_name`	Character; one of `"sobcov"`, `"sobtpu"`, `"lgm"`, `"lrp"`, or `"colsom"`. Determines both the subdirectory and the column schema.
`dest`	Character path where the final `.rds` files will go. Defaults to `"./data-raw/data_release"`.
`url_rma_ftp_file_access`	Base URL for USDA RMA web data. Defaults to the official RMA FTP file access root.

Value

Invisibly returns `NULL`. Side effect: one `.rds` per year is written under `dest`, named `<file_name>_<year>.rds`.

Examples

```
## Not run:
# Get sobtpu data for 2018-2022
download_rma_web_data_files(2018:2022, "sobtpu")

## End(Not run)
```

`fcip_aph_base_rate` *fcip_aph_base_rate*

Description

A combined dataset for fcip_aph_base_rate

Usage

```
data(fcip_aph_base_rate)
```

Format

A data frame with 428502 rows and 6 columns covering 2001-2026.

Source

USDA-RMA, Actuarial Data Master supplemented data from legacy ADM files

---

`fcip_contiguous_county` *fcip_contiguous_county*

Description

A combined dataset for fcip_contiguous_county

Usage

```
data(fcip_contiguous_county)
```

Format

A data frame with 24307 rows and 12 columns covering Inf–Inf.

Source

USDA-RMA, Actuarial Data Master - A0123

`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_recodes_commodity_groupings`
fcip_recodes_commodity_groupings

Description

A combined dataset for fcip_recodes_commodity_groupings

Usage

```
data(fcip_recodes_commodity_groupings)
```

Format

A data frame with 3572 rows and 10 columns covering 1997-2025.

Source

USDA-RMA, Actuarial Data Master - A00400 and A00420 supplemented data from legacy ADM files

---

`fcip_recodes_insurance_plan`
fcip_recodes_insurance_plan

Description

A combined dataset for fcip_recodes_insurance_plan

Usage

```
data(fcip_recodes_insurance_plan)
```

Format

A data frame with 773 rows and 10 columns covering 1989-2025.

Source

USDA-RMA, Actuarial Data Master - A00460 supplemented data from legacy ADM files

`fcip_recodes_practice` *fcip_recodes_practice*

Description

A combined dataset for fcip_recodes_practice

Usage

```
data(fcip_recodes_practice)
```

Format

A data frame with 28640 rows and 8 columns covering 1997-2025.

Source

USDA-RMA, Actuarial Data Master - A00510 supplemented data from legacy ADM files

`fcip_recodes_type` *fcip_recodes_type*

Description

A combined dataset for fcip_recodes_type

Usage

```
data(fcip_recodes_type)
```

Format

A data frame with 232730 rows and 7 columns covering 1999-2025.

Source

Generated internally, using harmonize_crop_type_codes()

`fsa_crop_linker` *Simulator Helper Datasets*

Description

A combined dataset for fsa_crop_linker

Usage

```
data(fsa_crop_linker)
```

Format

A data frame with 8594 rows and 8 columns covering Inf–Inf.

Source

Internal innovation

`get_census_harvested_area` *Retrieve and Aggregate NASS Harvested Area Data*

Description

Extracts, filters, and aggregates harvested area information for major field crops from pre-processed USDA NASS datasets (e.g., Census of Agriculture). Multiple crop-specific descriptors (e.g., "CORN, GRAIN - ACRES HARVESTED", "CORN, SILAGE - ACRES HARVESTED") are standardized into unified commodity groups, then aggregated to county level and (optionally) rolled up to state and national levels across selected years.

Usage

```
get_census_harvested_area(
  dir_source,
  census_years = c(2002, 2007, 2012, 2017, 2022),
  aggregation_level = c("STATE", "NATIONAL", "COUNTY"),
  map_crop_area = list(barley = "BARLEY - ACRES HARVESTED", corn =
   c("CORN, GRAIN - ACRES HARVESTED", "CORN, SILAGE - ACRES HARVESTED"), cotton =
    "COTTON - ACRES HARVESTED", oats = "OATS - ACRES HARVESTED", peanuts =
    "PEANUTS - ACRES HARVESTED", rice = "RICE - ACRES HARVESTED", sorghum =
    c("SORGHUM, GRAIN - ACRES HARVESTED", "SORGHUM, SILAGE - ACRES HARVESTED",
   "SORGHUM, SYRUP - ACRES HARVESTED"), soybeans = "SOYBEANS - ACRES HARVESTED", wheat =
    "WHEAT - ACRES HARVESTED")
)
```

Arguments

`dir_source`	Character. Path to the root directory containing the downloaded or pre-processed NASS datasets.
`census_years`	Numeric vector of Census of Agriculture years to include. Default: `c(2002, 2007, 2012, 2017, 2022)`.
`aggregation_level`	Character vector specifying one or more levels of geographic aggregation to include. One or more of `"COUNTY"`, `"STATE"`, `"NATIONAL"`. COUNTY data are always queried and used as the base for rollups.
`map_crop_area`	Named list mapping standardized crop names to their corresponding NASS `short_desc` values used for filtering harvested area items. Defaults cover barley, corn, cotton, oats, peanuts, rice, sorghum, soybeans, and wheat.

Details

For each requested census dataset, the function:

1. Calls `process_nass_dataset()` with filters for harvested area.
2. Normalizes and coerces numeric values.
3. Maps NASS descriptors to standardized `commodity_name`.
4. Aggregates to COUNTY, then (optionally) rolls up to STATE and NATIONAL.

Datasets that cannot be read/processed are skipped silently.

Value

A single `data.table` with aggregated harvested area and columns:

commodity_year Numeric; year of the commodity observation.

commodity_name Character; standardized crop identifier.

state_code State FIPS code (NA at NATIONAL level).

county_code County FIPS code (NA at STATE/NATIONAL levels).

value Total harvested area (acres).

agg_level One of `"COUNTY"`, `"STATE"`, `"NATIONAL"`.

`get_file_info` *Get File Information from a Directory*

Description

Scans a specified directory for files with a given suffix and returns a data frame containing their file paths, sizes in bytes, and sizes in megabytes.

Usage

```
get_file_info(directory = "./data-raw", file_suffix = ".rds")
```

Arguments

`directory`	A character string specifying the path to the directory to scan. Defaults to `"./data-raw"`.
`file_suffix`	A character string specifying the file suffix to match. Defaults to `".rds"`.

Value

A data frame with columns:

`file_path`	Full file path
`size_bytes`	File size in bytes
`size_mb`	File size in megabytes

Source

copied from https://github.com/dylan-turner25/rmaADM/blob/main/R/helpers.R

Examples

```
## Not run:
get_file_info()
get_file_info(directory = "./my-data", file_suffix = ".csv")

## End(Not run)
```

`get_ice_data` *Download and clean Insurance Control Elements tables*

Description

`get_ice_data()` retrieves all "YTD" ICE (Insurance Control Elements) text files from the specified directory on the RMA public FTP site for one or more years, downloads them to a temporary location, reads them as pipe-delimited data, applies internal cleaning routines, and returns the combined dataset. Original text files are discarded after reading.

Usage

```
get_ice_data(
  years = 2012,
  ice_url =
   "https://pubfs-rma.fpac.usda.gov/pub/References/insurance_control_elements/PASS/",
  selected_ice = NULL
)
```

Arguments

`years`	Integer vector of calendar years to download (e.g. `2012:2020`). Defaults to `2012`.
`ice_url`	Character string giving the base URL of the ICE directory on the RMA FTP site. Must end with a slash. Defaults to `"https://pubfs-rma.fpac.usda.gov/pub/References/insur`
`selected_ice`	Character vector of keyword(s) or regular expressions to filter the filenames. Only ICE files whose names match at least one element of `selected_ice` will be downloaded. If `NULL`, all "YTD" files are processed.

Value

A single `data.table` (invisibly coercible to `data.frame`) containing the cleaned ICE data for all requested years. If no matching files are found or all downloads fail, returns an empty `data.table`.

Examples

```
## Not run:
# Download & process ICE tables for 2018 and 2019,
# filtering for any file with "IceAOExpenseSubsidy" in its name
ice_df <- get_ice_data(
  years        = 2018:2019,
  selected_ice = "IceAOExpenseSubsidy"
)

## End(Not run)
```

`get_marketing_year_avg_price` *Get Marketing Year Average Price for a Single Crop from USDA NASS Quick Stats*

Description

`get_marketing_year_avg_price()` fetches USDA NASS Quick Stats data for a specified crop (`short_desc`) at one or more aggregation levels (`agg_level_desc`), computes the mean price for the marketing year, joins to official RMA commodity codes, applies necessary unit conversions, and returns a tidy table of marketing-year average prices.

Usage

```
get_marketing_year_avg_price(
  dir_source = "./data-raw/fastscratch/nass/",
  agg_level_desc = c("NATIONAL", "STATE", "COUNTY"),
  short_desc = "CORN, GRAIN - PRICE RECEIVED, MEASURED IN $ / BU"
)
```

Arguments

`dir_source`	`character(1)` Path to the directory where Quick Stats files are stored. Defaults to `"./data-raw/fastscratch/nass/"`.
`agg_level_desc`	`character` One or more values for the `agg_level_desc` field in the Quick Stats data. Can be `"STATE"` or/and `"NATIONAL"`. Defaults to `"NATIONAL"`.
`short_desc`	`character` One or more Quick Stats `short_desc` strings identifying the crop-price series to retrieve. Defaults to `"CORN, GRAIN - PRICE RECEIVED, MEASURED IN $ / BU"`. **Currently, only the following set is supported:** c("OATS - PRICE RECEIVED, MEASURED IN $ / BU", "RYE - PRICE RECEIVED, MEASURED IN $ / BU", "TOBACCO - PRICE RECEIVED, MEASURED IN $ / LB", "CORN, GRAIN - PRICE RECEIVED, MEASURED IN $ / BU", "FLAXSEED - PRICE RECEIVED, MEASURED IN $ / BU", "BARLEY - PRICE RECEIVED, MEASURED IN $ / BU", "BEANS, DRY EDIBLE, INCL CHICKPEAS - PRICE RECEIVED, MEASURED IN $ / CWT", "HAY - PRICE RECEIVED, MEASURED IN $ / TON", "WHEAT - PRICE RECEIVED, MEASURED IN $ / BU", "COTTON - PRICE RECEIVED, MEASURED IN $ / LB", "COTTON, COTTONSEED - PRICE RECEIVED, MEASURED IN $ / TON", "COTTON, UPLAND - PRICE RECEIVED, MEASURED IN $ / LB", "SORGHUM, GRAIN - PRICE RECEIVED, MEASURED IN $ / CWT", "SOYBEANS - PRICE RECEIVED, MEASURED IN $ / BU", "SUGARBEETS - PRICE RECEIVED, MEASURED IN $ / TON", "PEAS, DRY EDIBLE - PRICE RECEIVED, MEASURED IN $ / CWT", "SUNFLOWER - PRICE RECEIVED, MEASURED IN $ / CWT", "RICE - PRICE RECEIVED, MEASURED IN $ / CWT", "RICE, MEDIUM-SHORT GRAIN - PRICE RECEIVED, MEASURED IN $ / CWT", "RICE, LONG GRAIN - PRICE RECEIVED, MEASURED IN $ / CWT", "PEANUTS - PRICE RECEIVED, MEASURED IN $ / LB", "CANOLA - PRICE RECEIVED, MEASURED IN $ / CWT", "MAPLE SYRUP - PRICE RECEIVED, MEASURED IN $ / GALLON", "MILLET, PROSO - PRICE RECEIVED, MEASURED IN $ / BU", "SUGARCANE - PRICE RECEIVED, MEASURED IN $ / TON", "SAFFLOWER - PRICE RECEIVED, MEASURED IN $ / CWT")

Value

A `data.table` with columns:

- `commodity_year` (`integer`): the marketing year
- `commodity_code` (`integer`): NASS commodity code
- `state_code` (`integer`): state FIPS code (if `agg_level_desc` includes `"STATE"`)
- `marketing_year_avg_price` (`numeric`): average price for the marketing year, in dollars per unit
- `data_source` (`character`): always `"USDA NASS Quick Stats"`

See Also

- `process_nass_dataset()` for the underlying data fetch and filtering
- `get_nass_large_datasets()` for downloading the raw Quick Stats files

Examples

```
## Not run:
# Default: national average for corn
get_marketing_year_avg_price()

# Both state and national for wheat
get_marketing_year_avg_price(
  agg_level_desc = c("STATE", "NATIONAL"),
  short_desc     = "WHEAT - PRICE RECEIVED, MEASURED IN $ / BU"
)

## End(Not run)
```

---

get_nass_census_data *Prepare USDA NASS Census Data for Release*

Description

Iterates over one or more USDA NASS census years, fetching each via `process_nass_dataset()`, and produces three sets of state-level and county-level summaries for each year:

1. Agricultural land metrics by state,
2. Crop insurance totals by state and county,
3. Broad Farm Registry (BRF) census summaries by state and national level.

Usage

```
get_nass_census_data(
  censuses = c(2022, 2017, 2012, 2007, 2002),
  dir_source = "./data-raw/fastscratch/nass/",
  dir_dest = "data-raw/data_release/nass/"
)
```

Arguments

`censuses`	Integer vector. One or more census years to process (e.g. `c(2022, 2017, 2012, 2007, 2002)`).
`dir_source`	Character. Path to the directory containing raw NASS Quick Stats census datasets (default: `"./data-raw/fastscratch/nass/"`).
`dir_dest`	Character. Directory where the processed RDS files will be saved (default: `"data-raw/data_release/nass/"`).

Details

For each year in `censuses`, the function: First, it calls `process_nass_dataset()` with the `large_dataset = paste0("census", census)` argument to load the raw census data into a `data.table`, then renames `commodity_year` to `census_year`. It then performs three blocks of aggregation:

1. **Agricultural Land by State** Filters for ECONOMICS-sector state-level records on cropland, pasture, woodland, and cropland share, cleans and converts the `value` field to numeric, computes the mean by `(census_year, state_code, short_desc)`, recodes `short_desc` to simple labels (`cropland`, `pasture`, `woodland`, `cropland_pct`), pivots wide with `data.table::dcast()`, coerces `census_year` and `state_code` to numeric, and saves `nass_census_agLand_state_<census>.rds`.

2. **Crop Insurance Summaries** Subsets for both cropland and crop-insurance acreage and operation counts plus several farm-related income receipt categories, strips commas and coerces `value` to numeric, drops invalid entries, converts location codes to numeric, filters to insurance-related domains, sums `value` by all relevant grouping fields, and saves `nass_census_insurance_data_<`
3. **BRF Census Aggregates** Filters for producer counts and acres (owned vs. rented), converts `value` to numeric, removes zeros and missing, first computes group-wise means and then sums across `(census_year, state_code, state_alpha, agg_level_desc, unit_desc, domaincat_desc)`, recodes `domaincat_desc` to `ALL_` or `BRF_` plus unit, pivots wide, reorders and renames columns, and saves `nass_census_brf_<census>.rds`.

Value

Character vector of all `nass_census_*.rds` filenames written to `dir_dest`.

See Also

- `process_nass_dataset` for loading raw Quick Stats data

Other USDA NASS Quick Stats: `downloaded_nass_large_datasets()`, `process_nass_dataset()`

`get_nass_historical_track_record_crop`
Download and Process USDA NASS Historical Crop Track Records

Description

Scrapes the USDA NASS historical track records web page to identify and download the latest `croptrXX.zip` archive, parses its index to locate CSV tables for area planted and harvested by crop, reads and reshapes each table, cleans and standardizes units and variable names, applies special-case unit conversions, and returns a unified time series data frame of crop-level measures (area planted, area harvested, production, yield, price).

Usage

```
get_nass_historical_track_record_crop(
  url = "https://usda.library.cornell.edu/concern/publications/c534fn92g?locale=en",
  dir_source = "./data-raw/fastscratch/nass/"
)
```

Arguments

`url`	Character. URL of the USDA NASS historical track record page.
`dir_source`	Character. Directory into which to download and extract the ZIP file. Default is `"./data-raw/fastscratch/nass/"`. Must exist or be creatable.

Details

1. **Link discovery & download**
   - Reads all `<a>` hrefs from `url`, filters for links containing `"c534fn92g"`, `"zip"`, and `"croptr"`.
   - Extracts the year from each filename and selects the link with the maximum year.

- Downloads that single ZIP as `croptr.zip` into `dir_source`.

2. **Index parsing**
   - Opens `crop_index.htm` inside the ZIP, extracts the second HTML table as text.
   - Splits on double line breaks, filters to lines mentioning "Area Planted" or "Area Harvested", and excludes summary or non-crop entries.
   - Splits each line into `Page`, `File`, `Description`, then further into `Description` and a year range (`start`/`end`), and derives the uppercase crop name.
3. **CSV reading & reshaping**
   - For each row of the index:
     - Reads the CSV inside the ZIP once to locate header (`h`), unit (`u`), and data (`d`) rows.
     - Reads it again skipping to the data start, then loops over each data column to build a long table with columns `crop`, `crop_yr`, `variable`, `unit`, and `value`.
   - Combines all tables into one data.frame.
4. **Cleaning & standardization**
   - Converts `crop_yr` and `value` to numeric, drops zeros and invalids.
   - Scales values when unit indicates thousands, strips formatting characters, and normalizes variable names.
   - Consolidates crop subtypes (e.g. grain/silage/grazed) under a single `crop`.
5. **Variable mapping & special conversions**
   - Identifies key measures via lookup lists (`Area Planted`, `Area Harvested`, `Production`, `Yield`, `Price`) and relabels them to `"Tracks_*"`.
   - Applies special-case unit conversions (cents to dollars, bales to lbs, cwt to lbs, tons to lbs, etc.), including crop-specific rules (cotton, canola, dry beans, rice, sugarbeet, sugarcane, sunflower).
6. **Final pivot**
   - Pivots the cleaned long table to wide format so each `"Tracks_*"` measure is its own column, and returns a data.frame with columns: `CROP`, `crop_year`, `Tracks_area_planted`, `Tracks_area_harvested`, `Tracks_production`, `Tracks_yield`, `Tracks_price`.

Value

A data.table in which each row corresponds to a crop-year, with standardized track record measures for area planted, area harvested, production, yield, and price.

`get_nass_production_data` *Retrieve and Aggregate NASS Production Data*

Description

This helper function cleans, and aggregates production and area data from the USDA NASS Quick Stats API (via your `process_nass_dataset()` function), for specified geographic aggregation levels (national, state, county). It returns a `data.table` summarizing mean production and area by the chosen levels.

Usage

```
get_nass_production_data(
  dir_source = "./data-raw/fastscratch/nass/",
  source_desc = "SURVEY",
  agg_level_desc = c("NATIONAL", "STATE", "COUNTY")
)
```

Arguments

`dir_source`	Character. Path to the directory where NASS Quick Stats raw QS files are stored (default: `"./data-raw/fastscratch/nass/"`).
`source_desc`	Character. The `source_desc` filter passed to NASS (e.g. `"SURVEY"`).
`agg_level_desc`	Character vector. Which aggregation levels to include: any combination of `"NATIONAL"`, `"STATE"`, and `"COUNTY"`. Controls which code columns (`state_code`, `county_code`) are added.

Details

This function begins by constructing the set of grouping keys (`agg_level_list`), always including the year, commodity name, aggregation descriptor, statistic category, and unit, and then conditionally adding state and/or county codes if those levels are requested. It then invokes `process_nass_dataset()` to fetch the raw crop data for the specified source, sector, domain, country, frequency, reference period, statistic categories, and aggregation levels. Once the data are loaded, any invalid or missing values are removed and the mean of the remaining values is computed for each unique combination of metadata columns. Four separate summaries are then generated: (1) overall production/utilization/class totals, (2) breakdown by commodity class, (3) breakdown by utilization practice, and (4) breakdown by production practice. These four summaries are merged back together, and any rows with unwanted units (e.g., containing a dollar sign) or total rows in the commodity name are filtered out. Next, area metrics are processed by selecting the first non-missing sum across the four summaries and averaging it, and production metrics are handled similarly after filtering out invalid unit-commodity combinations and converting cotton bale values to pounds. Finally, the area and production results are bound together and summed across the chosen grouping keys to produce the final `data.table`.

Value

A `data.table` with one row per combination of:

- `commodity_year`, `commodity_name`,
- chosen geographic codes (`state_code`, `county_code`),
- plus any other aggregation keys. Columns contain summed mean values for production and area.

See Also

- `downloaded_nass_large_datasets` - for downloading and caching USDA NASS Quick Stats large dataset files
- `process_nass_dataset` - for retrieving raw NASS Quick Stats data

`get_price_indices` *Build a price-received deflator (PPIPR) series relative to* `current_year`

Description

Constructs a table used to deflate nominal FCIP monetary amounts to a common base year. Returns two columns, `commodity_year` and `PPIPR`, where `PPIPR` equals the year's price-received index divided by the index in `current_year` (so `PPIPR(current_year) == 1`).

Usage

```
get_price_indices(current_year = NULL)
```

Arguments

`current_year`	Integer scalar. The base year used for normalization. The returned `PPIPR` equals 1 for this year.

Details

Data sources (from `rfcipDemand`):

- `nassSurveyPriceRecivedIndex` (annual; expects `commodity_year`, `index_for_price_recived`).
- `nassAgPriceMonthlyIndex` (monthly U.S. agricultural price index; expects `year`, `comm`, `index`).

Synthesizing the current year (if missing in the annual table):

- Compute the arithmetic mean of the monthly index where `comm == "Agricultural"` for both `current_year` and `current_year - 1`.
- Multiply last year's annual `index_for_price_recived` by the ratio `mean_monthly(current_year) / mean_monthly(current_year - 1)` to derive the current-year annual index.
- Append this row with `data_source = "calculated"`.

Normalization:

- Let the denominator be the (mean) `index_for_price_recived` among rows with `commodity_year == current_year` (provides stability if duplicates exist).
- Define `PPIPR = index_for_price_recived / denominator`.

Output shape:

- Returns only `commodity_year` and `PPIPR`, sorted ascending by `commodity_year`.
- If the input annual table contains multiple rows per year, duplicates are preserved in the output (each with its own `PPIPR`). Aggregate if you require strictly one row per year (see Notes).

Value

A `data.table` with two columns:

- `commodity_year` - integer year.
- `PPIPR` - numeric deflator equal to the year's price-received index divided by the `current_year` index.

Assumptions & Notes

- Assumes both reference datasets from **rfcipDemand** are available with the specified columns (including the source's spelling `index_for_price_recived`).
- Monthly means are computed with `na.rm = TRUE`.
- If you need one row per year, post-aggregate: `dt[, .(PPIPR = mean(PPIPR, na.rm = TRUE)), by = commodity_year]`.

`get_state_rental_rates` *Get State Rental Rates for Cropland*

Description

Get State Rental Rates for Cropland

Usage

```
get_state_rental_rates(dir_source = "./data-raw/fastscratch/nass/")
```

Arguments

`dir_source`	`character(1)` Path to the directory where Quick Stats files are stored. Defaults to `"./data-raw/fastscratch/nass/"`. Approximate per-acre cost of crop production using state-level rental rates retrieved from USDA NASS Quick Stats. This function: 1. Loads and aggregates NASS asset values and cash rents by state and year. 2. Joins the two series, excludes non-contiguous states/territories. 3. Estimates missing rents via a panel regression on log asset values, then corrects for systematic bias. 4. Interpolates any remaining missing values using a 5-nearest-neighbor spatial average, iterated twice.

Details

Internally this function relies on:

- `process_nass_dataset()` to pull NASS Quick Stats.
- `plm::pdata.frame()` for panel data setup.
- `lm()` to fit a state-fixed-effects trend model.
- spdep to compute spatial lags (5-nearest neighbors).
- `doBy::summaryBy()` for error-ratio corrections.
- `terra` and `tigris` to obtain state geometries.

Value

A `data.frame` with columns:

`NAME` State name

`state_code` Numeric state FIPS code

`commodity_year` Year of the observation

`rent` Adjusted per-acre rent ($/acre)

Examples

```
## Not run:
# Make sure `process_nass_dataset()` and required packages are loaded
df <- get_state_rental_rates()
head(df)

## End(Not run)
```

---

harmonize_codes_and_names

Harmonize names/codes in a data table of insurance elections

Description

This function looks for two columns and creates recoded versions grouping similar codes into broader categories.

Usage

```
harmonize_codes_and_names(df)
```

Arguments

df	A data.frame

Value

A data.frame with the same columns as df with new columns

---

harmonize_crop_type_codes

Harmonize and summarize crop type codes from raw SOBTPU data

Description

Download the raw Summary of Business by Type, Practice, and Unit Structure (SOBTPU) data, clean and harmonize the crop type names for a fixed set of commodity codes, determine a single dominant type per county, and save the resulting lookup table of type renames.

Usage

```
harmonize_crop_type_codes()
```

Details

This helper function performs the following steps:

1. Downloads the raw SOBTPU data and reads it in.
2. Filters out rows with missing, zero, infinite, or NaN `liability_amount`, and retains only the specified set of commodity codes (wheat, barley, rice, etc.).
3. Recodes `type_name` by commodity:
    - **Wheat (11):** `"DURUM"`, `"SPRING"`, or `"NO TYPE SPECIFIED"`, dropping `"FORAGE WHEAT FOR SEED"`.
    - **Barley (91):** `"SPRING"` or `"WINTER."`
    - **Rice (18):** `"SHORT"`, `"LONG"`, `"MEDIUM"`, or `"NO TYPE SPECIFIED"`, dropping certain japonica types.
    - **Other crops (41, 75, 94, 17, 81):** `"GRAIN"`, `"SILAGE"` (when `type_code == 26`), or "ALL".
4. Summarizes total `liability_amount` by `(commodity_code, state_code, county_code, type_name)`.
5. Pivots to wide form, with one column per `type_name`, filling missing with zero.
6. Determines the single "dominant" type for each county-i.e. the one with positive liability when all other types are zero.
7. Joins this `type_recode` back onto the full dataset and applies final cleanup:
    - If `type_name` is not `"NO TYPE SPECIFIED"`, it overrides the county-level rename.
    - Fills any remaining blanks or `NA` in `type_recode` with `"NO TYPE SPECIFIED"`.
8. Drops any residual invalid liability rows, selects and deduplicates the key columns, adds a `data_source` column, saves to `./data-raw/type_recodes.rds`, and returns the result.

Value

A `data.frame` with columns:

- `commodity_year`
- `state_code`
- `county_code`
- `commodity_code`
- `type_code`
- `type_recode`
- `data_source` (always "Summary of business by type, practice, and unit structure")

Examples

```
## Not run:
# Build and retrieve the lookup of harmonized crop types
lookup_dt <- harmonize_crop_type_codes()

## End(Not run)
```

ice_administrative_fee_waiver_code

`ice_administrative_fee_waiver_code` *ice_administrative_fee_waiver_code*

Description

A combined dataset for ice_administrative_fee_waiver_code

Usage

```
data(ice_administrative_fee_waiver_code)
```

Format

A data frame with 66 rows and 4 columns covering Inf–Inf.

Source

USDA-RMA, Insurance Control Elements - PASS - D00154

ice_ao_expense_subsidy_percent

`ice_ao_expense_subsidy_percent` *ice_ao_expense_subsidy_percent*

Description

A combined dataset for ice_ao_expense_subsidy_percent

Usage

```
data(ice_ao_expense_subsidy_percent)
```

Format

A data frame with 4232 rows and 7 columns covering 1998-2025.

Source

USDA-RMA, Insurance Control Elements - PASS - D00154

`ice_policy_history_request_code`
ice_policy_history_request_code

Description

A combined dataset for ice_policy_history_request_code

Usage

```
data(ice_policy_history_request_code)
```

Format

A data frame with 127 rows and 4 columns covering Inf–Inf.

Source

USDA-RMA, Insurance Control Elements - PASS - D00154

`ice_program_indicator_code`
ice_program_indicator_code

Description

A combined dataset for ice_program_indicator_code

Usage

```
data(ice_program_indicator_code)
```

Format

A data frame with 46 rows and 4 columns covering Inf–Inf.

Source

USDA-RMA, Insurance Control Elements - PASS - D00154

`ice_yield_type_code` *ice_yield_type_code*

Description

A combined dataset for ice_yield_type_code

Usage

```
data(ice_yield_type_code)
```

Format

A data frame with 913 rows and 15 columns covering Inf–Inf.

Source

USDA-RMA, Insurance Control Elements - PASS - D00154

`layouts_fcip` *Column Layouts for Federal Crop Insurance Program (FCIP) Data*

Description

Use these layouts when reading raw FCIP CSV exports or when constructing data frames for analysis. Ensures consistent column ordering and naming across multiple report types.

Usage

```
layouts_fcip
```

Format

An object of class `list` of length 8.

Source

USDA-RMA

`locate_download_link` *Locate the download link for the actuarial data master*

Description

Locate the download link for the actuarial data master

Usage

```
locate_download_link(
  year = 2012,
 adm_url = "https://pubfs-rma.fpac.usda.gov/pub/References/actuarial_data_master/",
  ice_url =
   "https://pubfs-rma.fpac.usda.gov/pub/References/insurance_control_elements/PASS/",
  data_source = "adm"
)
```

Arguments

`year`	the year of the actuarial data master to download
`adm_url`	the url where the ADM FTP site is
`ice_url`	the url where the ICE (insurance control elements) FTP site is
`data_source`	either "adm" or "ice". Defaults to "adm".

Value

a list of the data and layout file urls with the time the file was last updated on RMA's server

Source

copied from https://github.com/dylan-turner25/rmaADM/blob/main/R/helpers.R

Examples

```
## Not run: locate_download_link(year = 2012)
```

`nassAgPriceMonthlyIndex` *nassAgPriceMonthlyIndex*

Description

A combined dataset for nassAgPriceMonthlyIndex

Usage

```
data(nassAgPriceMonthlyIndex)
```

Format

A data frame with 2799 rows and 8 columns covering Inf–Inf.

Source

USDA NASS: https://www.nass.usda.gov/Charts_and_Maps/graphics/data

---

`nassSurveyAnimalInventory` *nassSurveyAnimalInventory*

Description

A combined dataset for nassSurveyAnimalInventory

Usage

```
data(nassSurveyAnimalInventory)
```

Format

A data frame with 175547 rows and 10 columns covering 1920-2025.

Source

USDA NASS Quick Stats - Animal inventory as of first of Jan

`nassSurveyMYAprice` *nassSurveyMYAprice*

Description

A combined dataset for nassSurveyMYAprice

Usage

```
data(nassSurveyMYAprice)
```

Format

A data frame with 31139 rows and 7 columns covering 1866-2024.

Source

USDA NASS Quick Stats

`nassSurveyPriceRecivedIndex` *nassSurveyPriceRecivedIndex*

Description

A combined dataset for nassSurveyPriceRecivedIndex

Usage

```
data(nassSurveyPriceRecivedIndex)
```

Format

A data frame with 35 rows and 3 columns covering 1990-2024.

Source

USDA NASS Quick Stats

`nassSurveyRentalRates` *nassSurveyRentalRates*

Description

A combined dataset for nassSurveyRentalRates

Usage

```
data(nassSurveyRentalRates)
```

Format

A data frame with 1792 rows and 5 columns covering 1994-2025.

Source

Output from get_state_rental_rates() function

`premium_subsidy_schedule` *premium_subsidy_schedule*

Description

A combined dataset for premium_subsidy_schedule

Usage

```
data(premium_subsidy_schedule)
```

Format

A data frame with 7522 rows and 7 columns covering 2001-2025.

Source

USDA-RMA, Actuarial Data Master supplemented data from legacy ADM files

`process_nass_dataset` *Process a USDA NASS Quick Stats dataset by sector and statistic category*

Description

`process_nass_dataset()` downloads (if needed) and reads one or more NASS Quick Stats large datasets files for a given sector, filters the rows by the chosen statistic category plus any additional Quick Stats API parameters, converts and cleans the `value` column, aggregates it by taking its mean over all remaining grouping columns, and then renames that aggregated column to match the requested statistic.

Usage

```
process_nass_dataset(
  dir_source = "./data-raw/fastscratch/nass/",
  large_dataset,
  statisticcat_desc = NULL,
  nassqs_params = NULL
)
```

Arguments

`dir_source`	`character(1)` **Length 1.** Path to the directory where Quick Stats large datasets files are stored (and will be downloaded to via `get_nass_large_datasets()`). Defaults to `"./data-raw/fastscratch/nass/"`.
`large_dataset`	`character(1)` The Quick Stats `large_dataset` to load (e.g. `"crops"`). one of: "census2002","census2007","census2012","census2017","census2022", "census2007zipcode","censu "animals_products","crops","demographics","economics","environmental"

`statisticcat_desc`
: `character(1)` **Length 1.** The Quick Stats `statisticcat_desc` to filter on (e.g. `"PRICE RECEIVED"`). After aggregation, the resulting column of mean values will be renamed to `gsub(" ", "_", statisticcat_desc)`.

`nassqs_params`
: `list` or `NULL` A named list of additional Quick Stats API parameters to filter by (e.g. `"domain_desc"`, `"agg_level_desc"`, `"year"`, etc.). Names must correspond to valid Quick Stats fields. If `NULL` (the default), only `sector_desc` + `statisticcat_desc` filtering is applied. Use `rnassqs::nassqs_params()` to list all valid parameter names.

Details

The full set of valid Quick Stats API parameter names can be retrieved with:

`rnassqs::nassqs_params()`

Value

A `data.table` where:

- All original columns have been lowercased.
- Rows have been filtered by `nassqs_params`.
- A `value` column has been converted to numeric (commas stripped), cleaned of non-finite entries, and then aggregated by mean over the remaining columns.
- That aggregated column is renamed to `gsub(" ", "_", statisticcat_desc)`.
- Numeric code columns `state_code`, `country_code`, `asd_code`, plus `commodity_year` and `commodity_name` have been created.

See Also

- `get_nass_large_datasets()` for downloading the raw Quick Stats files

Other USDA NASS Quick Stats: `downloaded_nass_large_datasets()`, `get_nass_census_data()`

Examples

```
## Not run:
# National annual average price received for all CROPS in 2020:
dt1 <- process_nass_dataset(
  large_dataset       = "crops",
  statisticcat_desc = "PRICE RECEIVED",
  nassqs_params = list( agg_level_desc = "NATIONAL", year = 2020 ))

# State-level marketing-year average price for soybeans:
dt2 <- process_nass_dataset(
  large_dataset       = "crops",
  statisticcat_desc = "PRICE RECEIVED",
  nassqs_params     = list(
    agg_level_desc      = "STATE",
    short_desc          = "SOYBEANS - PRICE RECEIVED, MEASURED IN $ / BU",
    reference_period_desc = "MARKETING YEAR",
    freq_desc           = "ANNUAL"
  )
)

## End(Not run)
```

`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

`year_beg`	Integer. Beginning year of the analysis (default: 2001).
`year_end`	Integer. Ending year of the analysis (default: current system year).
`seed`	Integer. Random seed for reproducibility (default: 1980632).
`project_name`	Character. Project name (required). Used to build fast-scratch directory paths.
`local_directories`	List of project-local directories to create (default: `list("data-raw/output", "data-raw/scripts", "data")`).
`fastscratch_root`	Optional character. Root directory for fast-scratch files. If `NULL`, it is set automatically: • Windows: `"C:/fastscratch"` • Linux/macOS: `"/fastscratch/<username>"`
`fastscratch_directories`	List of fast-scratch subdirectories (relative to `<fastscratch_root>/<project_name>`) to create. If `NULL`, no fast-scratch subdirectories are created and `wd` 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)
```

`standardize_fcip_column_names` *Standardize FCIP Data Frame Column Names*

Description

Rename a set of common Federal Crop Insurance Program (FCIP) data frame columns to consistent, descriptive names. This ensures uniform naming across different data sources and simplifies downstream processing.

Usage

```
standardize_fcip_column_names(df)
```

Arguments

`df`	A `data.frame` containing FCIP data with short column names such as `crop_yr`, `state_cd`, etc.

Value

The input `df` with Standardized column names:

Examples

```
## Not run:
# Suppose fcip_raw is loaded with original column names
fcip_clean <- standardize_fcip_column_names(fcip_raw)
# Now fcip_clean has standardized, descriptive column names

## End(Not run)
```

Index