Package 'arpcCost'

October 1, 2025

Type Package

Title Calibarte Production Costs by Commodity and County

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 Provides tools to calibrate and harmonize agricultural production costs by commodity and county. The package integrates data sources, applies econometric calibration methods, and produces outputs suitable for policy analysis, risk management research, and farm-level decision support.

License GPL-3 + file LICENSE

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

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports data.table, stringr, xml2, readr, readxl, urbnmapr, sf, rfcip, GWmodel, sp, methods, rlang

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

Suggests knitr, rmarkdown, tibble, tidyr, dplyr, rvest, mockery, testthat (>= 3.0.0)

LazyData true

Contents

`apply_crop_specific_region_mapping`
Apply crop-specific adjustments to ERS region mapping

Description

Adjusts the ERS resource-region crosswalk when certain crops require sub-regional differentiation. Currently supports:

- **Peanuts** - splits the *Southern Seaboard* region into two labeled subregions.
- **Rice** - relabels selected states/regions into rice-specific subregions.

If neither peanuts nor rice are present in `county_data$crop`, `regs` is returned unchanged.

Usage

```
apply_crop_specific_region_mapping(regs, county_data, verbose = FALSE)
```

Arguments

`regs`	A data frame/data.table crosswalk with columns: `fips` (character, 5-digit, may be unpadded) and `Region` (ERS resource-region name).
`county_data`	A data frame/data.table of historical COP rows containing a `crop` column (lowercase crop names). Used to detect presence of `"peanuts"` and/or `"rice"`.
`verbose`	Logical; if `TRUE`, prints a small sample of relabeled rows.

Details

Peanuts mapping

- Counties with `Region == "Southern Seaboard"` and state FIPS in AL (`"01"`) or GA (`"13"`) are relabeled `"Southern Seaboard (AL, GA)"`.
- Counties with `Region == "Southern Seaboard"` and state FIPS in VA (`"51"`), NC (`"37"`), or SC (`"45"`) are relabeled `"Southern Seaboard (VA, NC, SC)"`.

Rice mapping Uses state FIPS to create rice-specific subregions (with *Mississippi Portal* handled first):

- Any county whose *original* ERS region is "Mississippi Portal" is relabeled "Mississippi River Delta".
- California (state FIPS "06") -> "California".
- Arkansas (state FIPS "05") counties *not* in "Mississippi Portal" -> "Arkansas Non-Delta".
- Texas ("48") and Louisiana ("22") counties *not* in "Mississippi Portal" -> "Gulf Coast".

Notes

- The function defensively zero-pads fips to 5 characters.
- Rice relabeling preserves the special handling of "Mississippi Portal" by reassignment to "Mississippi River Delta" before other state-based rules.

Value

A copy of regs with crop-specific Region relabeling applied when peanuts and/or rice are present; otherwise regs unchanged.

---

arpcCost_control *Create control parameters for* arpcCost

Description

Initializes a named list of adjustment factors used throughout the ARPC simulation pipeline. Defaults are sensible for research use but can be overridden for testing, replication, or alternative policy assumptions.

Usage

```
arpcCost_control(
  continuous_integration_session = FALSE,
  adm_decoy_state_abb = "ND",
  ratio_cup_cap_source = "ERS-COP",
  ratio_cup_cap_alpha = 0.1,
  ers_floor_rate = 0.8,
  census_years = c(2002, 2007, 2012, 2017, 2022)
)
```

Arguments

`continuous_integration_session`
: logical(1). If TRUE, use a small deterministic subset of the Actuarial Data Master (ADM) ZIP archive to ensure reproducible, fast CI runs. Default: FALSE.

`adm_decoy_state_abb`
: character(1). Two-letter state abbreviation used for the ADM decoy when continuous_integration = TRUE. Default: "ND".

`ratio_cup_cap_source`
: character(1). Source used to set the cup-and-cap bounds for county:national ratios. Default: "ERS-COP" (bounds derived from ERS regional:national ratios).

`ratio_cup_cap_alpha`
: numeric(1). Symmetric clamp width for ratios; e.g., 0.10 to (+ or -)10% around the ERS bounds. Default: 0.10.

ers_floor_rate	numeric(1). Floor for scaled county estimates expressed as a share of the ERS regional benchmark (e.g., 0.80 floors county values to 80% of ERS region). Default: 0.80.
census_years	integer vector. Census years used across the pipeline. Default: c(2002, 2007, 2012, 2017, 2022).

Value

A named list of control parameters.

See Also

scale_cop_items, back_fill_cop_proportionally

Other helpers: clear_arpcCost_cache()

---

back_fill_cop_proportionally

Proportionally back-fill missing county ARPC items within a cost category

---

Description

For a given category ("Operating costs" or "Allocated overhead"), fills missing/non-finite county ARPC items in proportion to **ERS national** item shares of that category's preliminary total.

Usage

```
back_fill_cop_proportionally(
  cost_category,
  data,
  ers_cost_item_catalog,
  control = arpcCost_control()
)
```

Arguments

cost_category	character(1). "Operating costs" or "Allocated overhead".
data	data.table/data.frame with partially computed county ARPC items and ERS national series.
ers_cost_item_catalog	data.table with at least category and item.
control	list of adjustment factors; see arpcCost_control.

Details

Steps performed:

1. Collect item columns for the category (excluding any total_*).
2. Compute county and national preliminary totals (row sums over items).
3. For each missing county item i, set: arpc_cty_i = (ers_nat_i / ers_nat_total) * arpc_cty_total when ers_nat_total > 0.
4. Add diagnostic flag: diag_{oc|ao}_zero_cty_pos_nat = TRUE when county total is zero but national total is positive.

Value

The same `data.table`, modified in place, with filled ARPC values, category-level preliminary totals, and the diagnostic flag.

Notes

- Backfill occurs only if the national preliminary total is positive.
- County totals of zero produce zero backfill (but are flagged).

`clear_arpcCost_cache` *Clear the package cache of downloaded data files*

Description

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

Usage

```
clear_arpcCost_cache()
```

Value

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

See Also

Other helpers: `arpcCost_control()`

Examples

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

## End(Not run)
```

`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.

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)
```

---

estimate_gwss_by_county

Estimate geographically weighted summary statistics (GWSS) for counties

---

Description

Computes Geographically Weighted Summary Statistics (GWSS) for a scalar, county-level variable observed in a subset of counties, then evaluates the statistics at **all** county locations (points-on-surface). This is useful for spatial smoothing and gap-filling (imputation) when some counties are missing observations.

Usage

```
estimate_gwss_by_county(
  data,
  fip_col,
  variable,
  distance_metric = "Euclidean",
  kernel = "gaussian",
  target_crs = 5070,
  draw_rate = 0.5,
  approach = "CV",
  adaptive = TRUE
)
```

Arguments

`data`	A `data.frame`/`data.table` with at least `fip_col` and `variable`.
`fip_col`	Character. Name of the county ID column in `data`; copied to `"county_fips"`.
`variable`	Character. Name of the numeric column in `data` to summarize.
`distance_metric`	Character. One of `gw_distance_metric_names()`. Default: `"Euclidean"`.
`kernel`	Character. One of `"gaussian"`, `"exponential"`, `"bisquare"`, `"boxcar"`, `"tricube"`. Default: `"gaussian"`.
`target_crs`	Integer EPSG used to project county geometries when `longlat = FALSE`. Default: **5070** (NAD83 / CONUS Albers, meters).
`draw_rate`	Numeric in (0, 1]. Fraction of observed counties used during bandwidth cross-validation. Default: **0.5** (50%).
`approach`	Character. Bandwidth selection approach passed to `GWmodel::bw.gwr()`. One of `"CV"`, `"AIC"`, `"AICc"`. Default: `"CV"`.
`adaptive`	Logical. Use adaptive (nearest-neighbour count) bandwidth instead of fixed distance. Default: `TRUE`.

Details

The function:

1. pulls U.S. counties from **urbnmapr** and projects to a suitable CRS;
2. copies `data[[fip_col]]` into `"county_fips"` and joins to the map;
3. fits GWSS using **only observed counties** (points) and selects an adaptive bandwidth by cross-validation on a random subsample sized by `draw_rate`;
4. evaluates GWSS at **all counties** and returns local summaries keyed by `county_fips`.

Inputs: `data` must contain a county identifier column referenced by `fip_col` and a numeric column referenced by `variable`. Internally, a column `"county_fips"` is created for joining with `urbnmapr::get_urbn_map("counties")`.

Distance metric (`distance_metric`) defines `(p, theta, longlat)` for `GWmodel::gw.dist()`. Use `gw_distance_metric_names()` to list options. If `longlat = TRUE` (e.g., `"Great Circle"`), counties are transformed to EPSG:4326; otherwise to `target_crs` (default 5070, meters).

Bandwidth selection: CV is run on a uniform random subsample of size `ceiling(draw_rate * n_obs)`, bounded to `[5, n_obs - 1]`. Call `set.seed()` beforehand for reproducibility. Returns `NULL` (with a message) if fewer than 5 counties have finite values.

Value

A `data.table` of GW summary statistics for **all counties**, with a `county_fips` column for merging back to polygons. Column names follow **GWmodel** conventions for the internal `value` variable (created from `variable`), e.g. `value_LM`, `value_LSD`, `value_LCV`, `value_LSKe`, `value_LSSke`, etc. Attributes attached: `"bandwidth"`, `"distance_params"`, `"kernel"`, `"approach"`, `"adaptive"`. Returns `NULL` when there are < 5 observed counties.

Imputation workflow

```
set.seed(123)
gw <- estimate_gwss_by_county(
  data = my_data, fip_col = "county_fips", variable = "my_var"
)
sf_out <- counties_sf |>
  dplyr::left_join(gw[, c("county_fips","value_LM")], by = "county_fips") |>
 dplyr::mutate(my_var_imputed = dplyr::if_else(is.finite(my_var), my_var, value_LM))
```

`extrapolate_arpc_cop` *Construct county-level ARPC expenses from ERS national profiles*

Description

Builds county-level ARPC expense series (USD/ac) by scaling NASS-derived county ratios against ERS national items and (optionally) proportionally back-filling missing county items using ERS national shares.

Usage

```
extrapolate_arpc_cop(
  data,
  ers_cost_item_catalog,
  proportional_back_fill = TRUE,
  source_of_variability = "nass_expense",
  control = arpcCost_control()
)
```

Arguments

data	A data.table or data.frame prepared by prepare_extrapolation_data(). Must include ERS national (ers_national_*) and regional (ers_region_*) items. When source_of_variability = "nass_expense", expects mean_expense_<slug>_county and mean_expense_<slug>_national for the mapped slugs (e.g., seed, fertilizer, ...).
ers_cost_item_catalog	data.table with at least category and item describing ERS items and their category membership.
proportional_back_fill	logical(1). If TRUE, perform proportional backfill within each category. Default: TRUE.
source_of_variability	character(1). "nass_expense" (default) uses mean_expense_*_{county,national} ratios; otherwise, treated as a slug prefix and expects <slug>_county and <slug>_national to exist.
control	list of adjustment factors; see arpcCost_control.

Details

The function operates in two categories:

- Operating costs: seed, fertilizer, chemicals, custom services, fuel/lube/electricity, repairs, interest on operating inputs.
- Allocated overhead: hired labor, taxes & insurance, opportunity cost of land, and inferred opportunity cost of unpaid labor.

Direct scaling. For an item i with national ERS value ERS_i^{nat} and a "source" county:national ratio r_i, the county estimate is

$$\mathrm{ARPC}_i^{county} = \mathrm{ERS}_i^{nat} \times r_i.$$

When source_of_variability = "nass_expense", the "source" ratio is $\mathrm{mean\_expense}_{i,county}/\mathrm{mean\_expense}_{i,natio}$ where mean_expense_* were constructed upstream from prepare_extrapolation_data() Ratios are clamped using cup-and-cap bounds derived from ERS regional:national ratios ($\pm$ ratio_cup_cap_alpha) or fixed bounds if ratio_cup_cap_source is not "ERS-COP". A floor of ers_floor_rate times the ERS regional value is applied.

Unpaid labor (allocated overhead). Inferred via:

$$\mathrm{ARPC}_{unpaid}^{county} = \left(\frac{\mathrm{ERS}_{unpaid}^{nat}}{\mathrm{ERS}_{hired\_labor}^{nat}} \right) \times \mathrm{ARPC}_{hired\_labor}^{county}.$$

Proportional backfill (optional). If proportional_back_fill = TRUE, missing/non-finite items within a category are filled in proportion to ERS national shares of that category's preliminary total.

Output shaping. Results are returned in long form with one row per county-item and columns item, value (USD/ac). Preliminary totals are excluded.

Value

Long-form data.table with keys (when present) commodity_name, commodity_year, census_year, resource_region_code, resource_region_name, state_code, county_code, fips, and columns:

- item - ARPC item name (without prefixes).
- value - county ARPC estimate (USD per acre).

Notes

- In-place updates are used; pass a copy if inputs must be preserved.
- `arpc_oc_county_purchased_irrigation_water` is currently `NA_real_`.
- Numeric invalids (`NA`, `NaN`, `Inf`) are dropped in the final long output.

See Also

`scale_cop_items`, `back_fill_cop_proportionally`, `arpcCost_control`

`get_fcip_program_yields` *Get FCIP area-program expected/final yields by year*

Description

`get_fcip_program_yields()` returns **county-level** FCIP area-program expected and final "yields" (index values) for a given year, then derives **state** and **national** aggregates. The data source depends on `year`:

- **2003-2010**: downloads prebuilt `area_index_{year}.rds` from *USFarmSafetyNetLab* GitHub Releases; uses `area_yield_expected`, `area_yield_final`.
- **2011-2016**: pulls ADM table `"A01005_AreaRiskRate"` via `rfcip::get_adm_data()`; uses `expected_index_value`, `final_index_value`.
- **2017-current**: pulls ADM table `"A00810_Price"` via `rfcip::get_adm_data()`; uses `expected_index_value`, `final_index_value`.

Within each era:

1. rows with non-finite or zero **expected** values are dropped;
2. values are **averaged** over duplicates within (`commodity_year`, `commodity_code`, `state_code`, `county_code`);
3. columns are renamed to county-level outputs: `fcip_expected_yield_county`, `fcip_final_yield_county`.

After retrieval, the function downloads a commodity grouping map (`fcip_recodes_commodity_groupings.rds`) and merges by `commodity_code` to get `CROP`. It then collapses any 1:n mappings by averaging to (`commodity_year`, `CROP`, `state_code`, `county_code`), lowercases `CROP` to `commodity_name`, and computes **state** and **national** aggregates: `fcip_expected_yield_state`, `fcip_expected_yield_national`, `fcip_final_yield_state`, `fcip_final_yield_national`.

Usage

```
get_fcip_program_yields(year)
```

Arguments

`year`	`integer(1)` Calendar year to fetch. Supported ranges are **2003-2010**, **2011-2016**, and **2017-**`as.integer(format(Sys.Date(), "%Y"))`. The function **errors** if `year` is outside these ranges.

Details

- Filtering uses `is.finite(expected) & expected != 0`.
- Aggregation uses `lapply(.SD, mean, na.rm = TRUE)`.
- The **legacy (2003-2010)** and **groupings** steps download files from GitHub; internet access is required for those paths.

Value

A `data.table` with one row per (`commodity_year`, `commodity_name`, `state_code`, `county_code`) and columns:

- `fcip_expected_yield_county`, `fcip_final_yield_county`
- `fcip_expected_yield_state`, `fcip_final_yield_state`
- `fcip_expected_yield_national`, `fcip_final_yield_national`

---

gw_distance_metric_names

List valid GW distance metric names

Description

List valid GW distance metric names

Usage

```
gw_distance_metric_names()
```

Value

Character vector of valid preset names.

---

gw_distance_metric_presets

GW distance metric presets for GWmodel

Description

Provides a curated set of distance metric presets (Minkowski family and great-circle) for **GWmodel**. Each preset specifies `(p, theta, longlat)` for `GWmodel::gw.dist()`.

Usage

```
gw_distance_metric_presets()
```

Value

A named list of presets, each entry a `list(p, theta, longlat)`.

prepare_census_data *Prepare NASS Census expense panels (county/state/national) and convert to per-acre metrics*

Description

Builds wide **USDA NASS Census of Agriculture** panels that include cropland acres and major expense/value series at three aggregation levels (COUNTY, STATE, NATIONAL), then converts expense/value fields to **per-cropland-acre** by dividing by cropland acres at the corresponding level.

Usage

```
prepare_census_data(
  dir_source = "./data-raw/fastscratch/nass_data/",
  census_years = arpcCost_control()$census_years,
  aggregation_level = c("STATE", "NATIONAL", "COUNTY"),
  map_expense_items = c(`AG LAND, CROPLAND - ACRES` = "nass_crop_acres",
    `SEEDS & PLANTS TOTALS - EXPENSE, MEASURED IN $` = "nass_seed",
   `FERTILIZER TOTALS, INCL LIME & SOIL CONDITIONERS - EXPENSE, MEASURED IN $` =
  "nass_fertilizer", `CHEMICAL TOTALS - EXPENSE, MEASURED IN $` = "nass_chemicals",
   `AG SERVICES, CUSTOMWORK - EXPENSE, MEASURED IN $` = "nass_custom_services",
   `INTEREST, NON-REAL ESTATE - EXPENSE, MEASURED IN $` =
   "nass_interest_non_real_estate", `LABOR, HIRED - EXPENSE, MEASURED IN $` =
   "nass_hired_labor",
    `FUELS, INCL LUBRICANTS - EXPENSE, MEASURED IN $` =
   "nass_fuel_lube_and_electricity",
  `SUPPLIES & REPAIRS, (EXCL LUBRICANTS) - EXPENSE, MEASURED IN $` = "nass_repairs",
  `TAXES, PROPERTY, REAL ESTATE & NON-REAL ESTATE` = "nass_taxes_and_insurance",
   `RENT, CASH, LAND & BUILDINGS - EXPENSE, MEASURED IN $` =
   "nass_cash_rent_land_buildings",
  `AG LAND, INCL BUILDINGS - ASSET VALUE, MEASURED IN $` = "nass_ag_land_value"),
  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. Directory containing source NASS Census files used by process_nass_dataset(). Default: "./data-raw/fastscratch/nass_data/".
census_years	Integer vector of Census years (e.g., c(2002, 2007, 2012, 2017, 2022)). Defaults to arpcCost_control()$census_years.
aggregation_level	Character vector subset of c("STATE","NATIONAL","COUNTY"). Default: all three.
map_expense_items	Named character vector mapping NASS short_desc to variable roots.

`map_crop_area` Named list mapping crop names to one or more NASS `short_desc` strings for harvested area.

Details

For each requested Census year, this function calls `process_nass_dataset()` to pull:

- Cropland acres
- Seeds & plants
- Fertilizer (incl. lime & soil conditioners)
- Chemicals
- Agricultural custom services
- Interest
- Hired labor
- Fuels (incl. lubricants)
- Supplies & repairs (excl. lubricants)
- Property & other taxes / insurance (as labeled by NASS)
- Cash rent for land & buildings
- Agricultural land (incl. buildings) asset value

NASS `short_desc` values are mapped to standardized variable roots: `nass_crop_acres`, `nass_seed`, `nass_fertilizer`, `nass_chemicals`, `nass_custom_services`, `nass_interest_on_operating_inputs`, `nass_hired_labor`, `nass_fuel_lube_and_electricity`, `nass_repairs`, `nass_taxes_and_insurance`, `nass_opportunity_cost_of_land`, `nass_ag_land_value`.

After reshaping to wide, each variable is suffixed by level: `_county`, `_state`, `_national`. Expense/value variables are then converted in place to per-acre units using `nass_crop_acres_<level>` as divisors.

Within-level duplicates are averaged via `mean(, na.rm = TRUE)` over:

- COUNTY: `(commodity_year, state_code, county_code, short_desc)`
- STATE: `(commodity_year, state_code, short_desc)`
- NATIONAL: `(commodity_year, short_desc)`

Levels that fail to load are skipped quietly.

The function also retrieves **harvested area** for major crops (barley, corn, cotton, oats, peanuts, rice, sorghum, soybeans, wheat), returning columns named `<crop>_harvested_area_<level>`.

Value

A `data.table` in wide format with:

- `census_year`
- `state_code`, `county_code` (`NA` where not applicable)
- Per-acre expense/value columns at each level (USD per cropland acre)
- Cropland acre divisors retained: `nass_crop_acres_<level>`
- Crop harvested-area columns: `<crop>_harvested_area_<level>` (acres)

Assumptions & caveats

- Any duplicate observations within a group are averaged with `mean(, na.rm=TRUE)`.
- If `nass_crop_acres_<level>` is missing or zero, the corresponding per-acre variables are set to `NA_real_`.
- Non-availability at a given level (e.g., county suppressed values) yields missing per-acre variables for that level.
- `process_nass_dataset()` must return at least the columns: `commodity_year`, `state_code`, `county_code`, `agg_level_desc`, `short_desc`, `value`.

Note

Units after conversion: **USD per cropland acre** for expense/value series; **acres** for harvested-area series. `nass_ag_land_value_<level>` becomes asset value per cropland acre.

References

USDA National Agricultural Statistics Service (NASS), Quick Stats / Census of Agriculture.

---

prepare_ers_commodity_costs

Download, harmonize, and extend ERS Cost-of-Production (COP) data

---

Description

End-to-end pipeline that: (1) downloads USDA ERS **Cost and Returns** historical CSVs by crop, (2) downloads the ERS COP **forecast** CSV, (3) downloads the ERS **resource-region** crosswalk (Excel), (4) reads and standardizes historical series, (5) computes national forecast **percent changes** by cost item/type, (6) extrapolates those national forecast changes to the most-recent county-level observations, and (7) returns tidy, analysis-ready tables at national and county (with ERS resource-region attribution) geographies, plus a lightweight cost-item catalogue and the region crosswalk.

Usage

```
prepare_ers_commodity_costs(
  ers_cop_base_url =
    "https://ers.usda.gov/sites/default/files/_laserfiche/DataFiles/47913/",
  historical_urls = list(corn = "CornCostReturn.csv?v=55131", cotton =
   "CottonCostReturn.csv?v=43745", barley = "BarleyCostReturn.csv?v=13878", peanuts =
   "PeanutCostReturn.csv?v=95354", rice = "RiceCostReturn.csv?v=38466", sorghum =
   "SorghumCostReturn.csv?v=21267", oats = "OatsCostReturn.csv?v=80094", soybeans =
    "SoybeanCostReturn.csv?v=20403", wheat = "WheatCostReturn.csv?v=34104"),
  forecast_urls = c("cop_forecast.csv?v=73342"),
 ers_regions = c("https://www.ers.usda.gov/sites/default/files/images/reglink.xls"),
  output_dir = "./data-raw/fastscratch/ers_cop"
)
```

Arguments

ers_cop_base_url	Base URL for ERS COP assets (historical and forecast CSVs).
historical_urls	Named list of *relative* CSV paths for crop-specific COP files. Names become commodity_name labels (e.g., "corn", "soybeans").
forecast_urls	Character vector of *relative* forecast CSV paths; the first is used.
ers_regions	Character vector of one or more full URLs for the ERS resource-region Excel; the first is used. Expected sheet format places FIPS and region code in the first two columns with two header rows to skip.
output_dir	Directory where downloaded/intermediate files are saved.

Details

Workflow

1. Create output_dir if missing (recursively).
2. For each crop in historical_urls, download the CSV via **readr**, append a crop label, and save as <crop>_cost.rds.
3. Download the COP forecast CSV and save as cop_forecast.rds.
4. Download the ERS resource-region Excel (reglink.xls) to output_dir.
5. Reload all *_cost.rds, row-bind, coerce Value to numeric, and normalize selected Item strings (e.g., align "Interest on operating capital" to "Interest on operating inputs").
6. Build *national* and *county* slices; keep original ERS keys (Year, Region, Category, Item, Value, crop) for transformation.
7. For *national* forecast deltas: filter U.S.-total forecast rows, keep operating costs (variable) and allocated overhead (fixed), then compute percent changes relative to the most recent historical national values by (crop, cost_item, cost_type).
8. For *county* forecasts: take the most recent county value for each (crop, fips, cost_item, cost_type) and multiply by the national percent change for the corresponding forecast year.
9. Combine historical national+county with national and county forecasts; standardize cost_item to snake_case and harmonize cost_type into fixed_cost, variable_cost, or total_cost.
10. Parse reglink.xls into an ERS region crosswalk with zero-padded fips and a resource_region_name. (If peanuts are present, subdivision rules may be applied; see apply_crop_specific_region_mapping().)

Inputs expected in ERS files The historical and forecast CSVs are assumed to include columns: Year, Region (e.g., "U.S. total" or ERS-region/county labels), Category (contains "operating costs" or "allocated overhead"), Item, and Value. Some sources also include Commodity; when absent, the pipeline relies on the injected crop label during ingest.

Column conventions in the return Returned cost rows are normalized to the following columns: commodity_name, commodity_year, geography (national or ERS region/county), state_code, county_code, fips (if applicable), cost_item (snake_case), cost_type {fixed_cost|variable_cost|total_cost}, cost_per_acre, and is_forecast (logical).

Notes

- Requires internet access. Each download is wrapped in tryCatch; failures are logged via cat() and the pipeline proceeds where feasible.
- If ERS modifies column names or sheet structure, adjust string handling and/or selectors accordingly.
- Side effects: writes multiple .rds files and reglink.xls to output_dir.

Value

A named `list` with:

- `ersResourceRegion` (`data.table`): ERS region crosswalk with columns `fips` (zero-padded), `resource_region_name`, and numeric `state_code`/`county_code`.
- `ersCop` (`data.table`): Tidy national/county COP history and forecasts with standardized `cost_item`, `cost_type`, `cost_per_acre`, and flags.
- `ers_cost_item_catalog` (`data.table`): Unique triplets `commodity_name`, `category`, `item`, `units`; empty if inputs are missing.

`prepare_extrapolation_data` *Build an ERS-NASS-FCIP extrapolation panel (county/state/region)*

Description

Constructs an analysis panel that aligns **NASS Census per-acre expenses**, **ERS Cost of Production (COP)** (transposed to wide), the **ERS resource-region** crosswalk, and **FCIP program yields**. For each Census year, the function (i) expands that slice across a moving window of `commodity_year` values; (ii) duplicates rows across all COP commodities; (iii) joins COP (wide) and FCIP; (iv) re-attaches any missing region attributes; (v) stabilizes ERS region and national series via within-group means; and (vi) computes area-weighted national means for NASS per-acre expenses to aid downstream checks.

Usage

```
prepare_extrapolation_data(
  census_expense,
  ersResourceRegion,
  ersCop,
  fcip_yield
)
```

Arguments

`census_expense`	A wide `data.table`/`data.frame` of NASS Census per-acre expenses keyed by (at least) `census_year`, `state_code`, `county_code` (optional `fips` allowed), with columns such as `nass_<item>_county`, `nass_<item>_state`, `nass_<item>_national`. For each commodity appearing in COP, a column `<commodity>_harvested_area_county` is recommended; if absent, a value of 1 is assumed for area-weighted computations.
`ersResourceRegion`	ERS resource-region crosswalk including attributes (e.g., `resource_region_name`, `resource_region_code` if available) and typically `state_code`/`county_code`. An upstream `fips` may be present and is dropped internally to avoid key duplication.
`ersCop`	A tall ERS COP table with at least `commodity_year`, `commodity_name`, `geography` (e.g., `"U.S. total"` or an ERS region name), `cost_item`, `cost_per_acre`, `is_forecast`, and (for regional rows) `state_code`/`county_code`. This is transposed to wide via `transpose_ers_cop` before joining.

`fcip_yield` A wide `data.table`/`data.frame` of FCIP program yields keyed by a subset of `census_year`, `state_code`, `county_code` (optional `fips` allowed), with columns like `fcip_<metric>_county`, `fcip_<metric>_state`, `fcip_<metric>_national`.

Details

- **Temporal expansion (moving window):** For each `census_year = cy`, the Census slice is replicated for `commodity_year` in `(cy - 2):(cy + 2)`. For the most recent Census year, the upper bound is extended to `max(ersCop$commodity_year)` so the panel reaches the latest COP year present.
- **Commodity duplication:** The expanded Census slice is duplicated across all `commodity_name` values observed in the COP input, enabling joins on `(commodity_year, commodity_name)`.
- **COP join (wide) and region attributes:** COP is first transposed to wide via `transpose_ers_cop` producing `ers_region_*` and `ers_national_*` columns and merged on intersecting keys (cartesian merges allowed). Region attributes from `ersResourceRegion` are merged back on intersecting keys if any are missing.
- **ERS mean stabilization:** For each `ers_region_*` column, values are replaced by the group mean over `(commodity_year, commodity_name, resource_region_name)` (rows with missing `resource_region_name` remain `NA`). For each `ers_national_*` column, values are replaced by the mean within `(commodity_year, commodity_name)`.
- **NASS area-weighted national means:** For each county-level NASS per-acre column matching `^nass_.*_county$`, an area-weighted national mean `_national` is computed within `(census_year, commodity_year)` using the weight `<commodity_name>_harvested_area_county` when present (otherwise weight `1`). Fall back to an unweighted mean if all weights are zero. After all merges, any column ending in `_national` (including FCIP/NASS) is additionally stabilized by a within-group mean over `(commodity_year, commodity_name)`.
- **FCIP join:** FCIP wide metrics are merged on intersecting keys. No regional aggregation is performed here.

Value

A `data.table` with one or more rows per `(census_year, commodity_year, commodity_name, state_code, county_code)` containing:

- NASS per-acre expenses at county/state/national: `nass_*_county`, `nass_*_state`, `nass_*_national` (with `_national` stabilized as described).
- COP wide series: `ers_region_*` (region-mean stabilized) and `ers_national_*` (national-mean stabilized).
- Region attributes from `ersResourceRegion`, including `resource_region_name` where available.
- FCIP wide metrics (if keys intersect).

Assumptions & caveats

- COP merge may use `allow.cartesian = TRUE`. Ensure upstream uniqueness if cartesian expansion is not desired.
- ERS region/national "means" overwrite within-group duplicates using `mean(, na.rm = TRUE)`. Entirely missing groups remain `NA`.
- NASS area weights use `<commodity>_harvested_area_county` when available, else weight `1`.

- Any NaN or infinite values produced by merges or arithmetic are normalized to NA in numeric columns.

---

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

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)
```

---

resolve_distance_metric

Resolve a GW distance metric preset

Description

Resolve a GW distance metric preset

Usage

```
resolve_distance_metric(name, stop_on_error = TRUE)
```

Arguments

name	Character scalar. One of gw_distance_metric_names().
stop_on_error	Logical. If TRUE, throw for unknown names; else NULL.

Value

list(p, theta, longlat) or NULL.

---

scale_cop_items *Scale ERS national costs to county level using ratios*

Description

Produces county-level ARPC items by multiplying **ERS national** values by a bounded county:national ratio. Bounds are derived from **ERS regional:national** ratios (or set symmetrically) and a floor is applied as a share of the ERS regional value.

Usage

```
scale_cop_items(
  cost_items,
  cost_denom_map = NULL,
  cost_category,
  data,
  source_of_variability = "nass_expense",
  control = arpcCost_control()
)
```

Arguments

cost_items	character. Short item names to scale (e.g., "seed", "fertilizer").
cost_denom_map	optional named character vector mapping items to the NASS denominator slugs used when source_of_variability = "nass_expense".
cost_category	character(1). Either "Operating costs" or "Allocated overhead".
data	data.table/data.frame with ERS/NASS columns and keys (commodity_year, commodity_name).
source_of_variability	character(1). "nass_expense" (default) or a custom slug prefix (must have <slug>_county and <slug>_national columns).
control	list of adjustment factors; see arpcCost_control.

Details

For each item x:

1. Compute ers_ratio = ers_region_x / ers_national_x (where defined).
2. Within (commodity_year, commodity_name), set bounds: lower = min(ers_ratio) * (1 - alpha), upper = max(ers_ratio) * (1 + alpha) when ratio_cup_cap_source = "ERS-COP", else lower = 1 - alpha, upper = 1 + alpha.

3. Compute source_ratio:
   - If source_of_variability = "nass_expense": source_ratio = mean_expense_slug_county / mean_expense_slug_national where slug comes from cost_denom_map.
   - Otherwise: expect <slug>_county and <slug>_national and use their ratio.
4. Clamp source_ratio to [lower, upper].
5. Set arpc_{oc|ao}_county_x = ers_national_x * source_ratio.
6. Apply region floor: values < ers_region_x * ers_floor_rate are floored up.
7. Invalid results (non-finite) are set to NA_real_.

Value

The same data.table, modified in place, with arpc_{oc|ao}_county_{x} columns (USD/ac) for each requested item.

Changes

- Prevention of accidental NA propagation when ratios are already in-range.
- Clamp width set by ratio_cup_cap_alpha (default (+ or -)10%).
- Additional guards for divide-by-zero and invalid numerics.

---

transpose_ers_cop *Transpose ERS COP series to wide (ers_region_* / ers_national_*) and attach region crosswalk*

Description

Converts a **tall** ERS Cost of Production (COP) table into a **wide** layout with ers_region_* and ers_national_* columns, averaging duplicates within grouping keys, and then merges the ERS resource-region crosswalk so region attributes (e.g., resource_region_name, codes, and/or FIPS-derived state/county) are available on the COP rows.

Usage

```
transpose_ers_cop(ersResourceRegion, ersCop)
```

Arguments

ersResourceRegion	A crosswalk with resource_region_name, resource_region_code (if available), and usually state_code/county_code (often derived upstream from a 5-digit fips). May include crop-specific relabels produced by apply_crop_specific_region_mappir
ersCop	A tidy ERS COP table containing commodity_year, commodity_name, geography (e.g., "U.S. total" or ERS region), state_code/county_code where applicable, cost_item, cost_per_acre, and is_forecast.

Details

The function first prefixes each `cost_item` with either `"ers_national_"` when `geography == "U.S. total"`, or `"ers_region_"` otherwise. It then:

1. Aggregates the **regional** slice by `(commodity_name, commodity_year, cost_item, state_code, county_code, is_forecast)` using `mean(cost_per_acre, na.rm = TRUE)`, and pivots to wide.
2. Aggregates the **national** slice by `(commodity_name, commodity_year, cost_item, is_forecast)` using the same mean, and pivots to wide.
3. Left-joins the regional and national wide tables on `(commodity_name, commodity_year, is_forecast)`.
4. Merges the ERS region crosswalk (`ersResourceRegion`) into the result, using the intersection of column names as join keys (with `allow.cartesian = TRUE` to permit one-to-many expansion).

Input expectations

- `ersCop` must be **tall** with at least: `commodity_year`, `commodity_name`, `geography` (e.g., `"U.S. total"` or an ERS region name), `cost_item`, `cost_per_acre`, `is_forecast`. For regional rows, `state_code` and `county_code` are expected if you want county-level aggregation (they are part of the regional grouping keys).
- `ersResourceRegion` can include `resource_region_name`, `resource_region_code` (if available), and usually `state_code`/`county_code` (often derived from a 5-digit FIPS). It may also include crop-specific relabels produced by `apply_crop_specific_region_mapping`.

Notes

- Duplicate observations within a grouping key are averaged via `mean(na.rm = TRUE)`.
- Only columns common to both inputs are used for the final merge; any others are ignored.
- Because the final merge uses `allow.cartesian = TRUE`, one COP row can expand to multiple geographies if the crosswalk maps it so.
- The function does not alter `is_forecast`; both historical and forecast rows pass through.

Value

A `data.table` where each row corresponds to `(commodity_name, commodity_year, is_forecast, [state_code, county_code, ...])` with wide COP columns of the form `ers_region_<item>` and `ers_national_<item>` when available, plus any region attributes merged from `ersResourceRegion`.

`widen_by_levels` *Widen a long table into county/state/national wide panels by averaging and casting*

Description

Widen a long table into county/state/national wide panels by averaging and casting

Usage

```
widen_by_levels(
  data,
  aggregation_level = c("COUNTY", "STATE", "NATIONAL"),
  name_col = "short_desc",
  value_col = "value",
  year_col = "commodity_year",
  level_col = "agg_level_desc",
  rename_year_to = "census_year"
)
```

Arguments

data	data.frame/data.table with columns: • year_col (default "commodity_year") • level_col (default "agg_level_desc") with values in ("COUNTY","STATE","NATIONAL") • state_code, county_code (for COUNTY/STATE where applicable) • name_col (default "commodity_name") -> becomes wide column names • value_col (default "value") -> becomes cell values
aggregation_level	character vector subset of c("COUNTY","STATE","NATIONAL")
name_col	column names (character)
value_col	column names (character)
year_col	column names (character)
level_col	column names (character)
rename_year_to	optional new name for year column (default "census_year")

Value

data.table wide panel with keys merged; year column renamed

Index