Package 'emburden'

May 18, 2026

Title Energy Burden Analysis Using Net Energy Return Methodology

Version 0.6.2

Description Calculate and analyze household energy burden using the Net Energy Return aggregation methodology. Functions support weighted statistical calculations across geographic and demographic cohorts, with utilities for formatting results into publication-ready tables. Methods are based on Scheier & Kittner (2022) <doi:10.1038/s41467-021-27673-y>.

License AGPL (>= 3)

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, httr, rappdirs, readr, rlang, scales, spatstat.univar, stats, stringr, tibble, tidyr

Suggests covr, DBI, httptest2, jsonlite, kableExtra, knitr, mockery, rmarkdown, RSQLite, rticles, testthat (>= 3.0.0), tinytex, withr

VignetteBuilder knitr

LazyData true

URL https://github.com/ericscheier/emburden, https://ericscheier.info/emburden/

BugReports https://github.com/ericscheier/emburden/issues

NeedsCompilation no

Author Eric Scheier [aut, cre, cph]

Maintainer Eric Scheier <eric@scheier.org>

Repository CRAN

Date/Publication 2026-05-18 20:40:02 UTC

Contents

`calculate_weighted_metrics` *Calculate Weighted Metrics for Energy Burden Analysis*

Description

Calculates weighted statistical metrics (mean, median, quantiles) for a specified energy metric, with optional grouping by geographic or demographic categories. This is the primary function for aggregating household-level energy burden data using proper weighting by household counts.

Usage

```
calculate_weighted_metrics(
  graph_data,
  group_columns,
  metric_name,
  metric_cutoff_level,
```

```
  upper_quantile_view = 1,
  lower_quantile_view = 0
)
```

Arguments

graph_data	A data frame containing household energy burden data with columns for the metric of interest, household counts, and optional grouping variables
group_columns	Character vector of column names to group by, or NULL for no grouping (calculates overall statistics)
metric_name	Character string specifying the column name of the metric to analyze (e.g., "ner" for Net Energy Return)
metric_cutoff_level	Numeric value defining the poverty threshold for the metric (e.g., 15.67 for Nh corresponding to 6% energy burden)
upper_quantile_view	Numeric between 0 and 1 specifying the upper quantile to calculate (default: 1.0 for maximum)
lower_quantile_view	Numeric between 0 and 1 specifying the lower quantile to calculate (default: 0.0 for minimum)

Details

This function requires the `spatstat` package for weighted quantile calculations. It automatically handles missing values and ensures that statistics are only calculated when sufficient data points exist (n >= 3).

The function adds an "All" category row that aggregates across all groups, in addition to the individual group statistics.

Value

A data frame with one row per group (or one row if ungrouped) containing:

household_count	Total number of households in the group
households_below_cutoff	Number of households below poverty threshold
pct_in_group_below_cutoff	Proportion of group below threshold
metric_mean	Weighted mean of the metric
metric_median	Weighted median of the metric
metric_upper	Upper quantile value
metric_lower	Lower quantile value
metric_max	Maximum value in group
metric_min	Minimum value in group

Examples

```
# Calculate metrics for NC cooperatives using Nh
library(dplyr)

# Sample data
data <- data.frame(
  cooperative = rep(c("Coop A", "Coop B"), each = 3),
  ner = c(20, 15, 25, 18, 22, 12),
  households = c(1000, 500, 750, 900, 600, 400)
)

# Calculate weighted metrics by cooperative
results <- calculate_weighted_metrics(
  graph_data = data,
  group_columns = "cooperative",
  metric_name = "ner",
  metric_cutoff_level = 15.67,
  upper_quantile_view = 0.95,
  lower_quantile_view = 0.05
)
```

---

check_data_sources *Check Available Data Sources*

Description

Check which data sources are available locally (database, CSV files, or will require download from OpenEI).

Usage

```
check_data_sources(verbose = TRUE)
```

Arguments

verbose	Logical, print detailed status (default TRUE)

Value

A list with status of each data source

Examples

```
# Check what data is available
check_data_sources()
```

clear_all_cache *Clear all emburden cache and database*

Description

Nuclear option: clears ALL cached data and database. Use with caution - will require re-downloading all data.

Usage

```
clear_all_cache(confirm = FALSE, verbose = TRUE)
```

Arguments

confirm	Logical, must be TRUE to proceed (safety check)
verbose	Logical, print progress messages

Value

Invisibly returns list with: cache_cleared (logical), db_cleared (logical)

Examples

```
# Clear everything (requires confirm = TRUE)
clear_all_cache(confirm = TRUE)
```

clear_dataset_cache *Clear cache for a specific dataset*

Description

Removes cached CSV files and database entries for a specific dataset/vintage. Useful when you know a specific dataset is corrupted.

Usage

```
clear_dataset_cache(
  dataset = c("ami", "fpl"),
  vintage = c("2018", "2022"),
  verbose = TRUE
)
```

Arguments

dataset	Character, "ami" or "fpl"
vintage	Character, "2018" or "2022"
verbose	Logical, print progress messages

Value

Invisibly returns number of items cleared

Examples

```
# Clear corrupted AMI 2018 cache
clear_dataset_cache("ami", "2018")

# Clear FPL 2022 cache
clear_dataset_cache("fpl", "2022", verbose = TRUE)
```

---

colorize *Colorize Text for Knitted Documents*

Description

Wraps text in color formatting appropriate for the output format (LaTeX or HTML). This function is intended for use within R Markdown/knitr documents.

Usage

```
colorize(x, color)
```

Arguments

x	Character string to colorize
color	Character string specifying the color name (e.g., "red", "blue")

Details

This function detects the knitr output format and applies appropriate color formatting. For LaTeX output, it uses `\\textcolor{}`. For HTML output, it uses `<span style='color: ...'>`.

Value

Character string wrapped in LaTeX or HTML color commands, or unchanged if output format is neither

Examples

```
# In an R Markdown document:
colorize("Important text", "red")
```

`compare_energy_burden` *Compare Energy Burden Between Years*

Description

Compare household energy burden metrics across different data vintages, using proper Net Energy Return (Nh) aggregation methodology.

Usage

```
compare_energy_burden(
  dataset = c("ami", "fpl"),
  states = NULL,
  group_by = "income_bracket",
  counties = NULL,
  vintage_1 = "2022",
  vintage_2 = "2018",
  format = TRUE,
  strict_matching = TRUE
)
```

Arguments

`dataset`	Character, either "ami" or "fpl" for cohort data type
`states`	Character vector of state abbreviations to filter by (optional)
`group_by`	Character or character vector. Use keywords "income_bracket" (default), "state", or "none" for standard groupings. Or provide custom column name(s) for dynamic grouping (e.g., "geoid" for tract-level, c("state_abbr", "income_bracket") for multi-level grouping). Custom columns must exist in the loaded data.
`counties`	Character vector of county names or FIPS codes to filter by (optional). Requires `states` to be specified.
`vintage_1`	Character, first vintage year: "2018" or "2022" (default "2022")
`vintage_2`	Character, second vintage year: "2018" or "2022" (default "2018")
`format`	Logical, if TRUE returns formatted percentages (default TRUE)
`strict_matching`	Logical, if TRUE (default) only compares income brackets that exist in both vintages and warns about mismatched brackets. If FALSE, compares all brackets (may result in NA values for brackets unique to one vintage).

Value

A data.frame with energy burden comparison showing:

- neb_YYYY: Net Energy Burden for each vintage (where YYYY is the year)
- change_pp: Absolute change in percentage points
- change_pct: Relative percent change

Examples

```
# Single state comparison (fast, good for learning)
nc_comparison <- compare_energy_burden("ami", "NC", "income_bracket")

# Overall comparison (no grouping)
compare_energy_burden("ami", "NC", "none")



if (interactive()) {
  # Multi-state regional comparison (requires census data download)
  southeast <- compare_energy_burden(
    dataset = "fpl",
    states = c("NC", "SC", "GA", "FL"),
    group_by = "state"
  )

  # Nationwide comparison by income bracket (all 51 states)
  us_comparison <- compare_energy_burden(
    dataset = "ami",
    group_by = "income_bracket"
  )

  # Compare specific counties within a state (requires census data)
  compare_energy_burden("fpl", "NC", counties = c("Orange", "Durham", "Wake"))

  # Custom grouping by tract-level geoid (requires census data)
  compare_energy_burden("ami", "NC", group_by = "geoid")
}
```

---

dear_func *Calculate Disposable Energy-Adjusted Resources (DEAR)*

Description

Calculates DEAR as the ratio of net income after energy spending to gross income. DEAR = (G - S) / G.

Usage

```
dear_func(g, s, se = NULL)
```

Arguments

g	Numeric vector of gross income values
s	Numeric vector of energy spending values
se	Optional numeric vector of effective energy spending (defaults to s)

Value

Numeric vector of DEAR values (ratio of disposable income to gross income)

Examples

```
# Calculate DEAR
dear_func(50000, 3000)
```

---

energy_burden_func *Calculate Energy Burden*

Description

Calculates the energy burden as the ratio of energy spending to gross income. Energy burden is defined as E_b = S/G, where S is energy spending and G is gross income.

Usage

```
energy_burden_func(g, s, se = NULL)
```

Arguments

g	Numeric vector of gross income values
s	Numeric vector of energy spending values
se	Optional numeric vector of effective energy spending (defaults to s)

Value

Numeric vector of energy burden values (ratio of spending to income)

Examples

```
# Calculate energy burden for households
gross_income <- c(50000, 75000, 100000)
energy_spending <- c(3000, 3500, 4000)
energy_burden_func(gross_income, energy_spending)
```

eroi_func *Calculate Energy Return on Investment (EROI)*

Description

Calculates the Energy Return on Investment as the ratio of gross income to effective energy spending. EROI = G/Se.

Usage

```
eroi_func(g, s, se = NULL)
```

Arguments

g	Numeric vector of gross income values
s	Numeric vector of energy spending values
se	Optional numeric vector of effective energy spending (defaults to s)

Value

Numeric vector of EROI values

Examples

```
# Calculate EROI for households
eroi_func(50000, 3000)
```

get_dataset_info *Get Dataset Information*

Description

Returns metadata about available LEAD datasets.

Usage

```
get_dataset_info()
```

Value

Data frame with dataset information

Examples

```
get_dataset_info()
```

get_income_brackets *Get Available Income Brackets for a Dataset and Vintage*

Description

Returns the expected income brackets for a given dataset and vintage year. Useful for understanding what brackets are available before running analyses.

Usage

```
get_income_brackets(dataset, vintage)
```

Arguments

dataset	Character, either "ami" or "fpl"
vintage	Integer, the year of the data vintage (e.g., 2018, 2022)

Value

Character vector of income bracket names

Examples

```
# Get AMI brackets for 2022
get_income_brackets("ami", 2022)

# Get FPL brackets for 2018
get_income_brackets("fpl", 2018)
```

list_cohort_columns *List Available Columns in Cohort Data*

Description

Returns column names and descriptions for LEAD cohort datasets.

Usage

```
list_cohort_columns(dataset = NULL, vintage = NULL)
```

Arguments

dataset	Character, either "ami" or "fpl" (optional, affects available columns)
vintage	Character, "2018" or "2022" (optional, affects available columns)

Value

Data frame with columns: column_name, description, data_type

Examples

```
list_cohort_columns()
list_cohort_columns("ami", "2022")
```

---

list_income_brackets *List Available Income Brackets*

Description

Returns the income brackets available for a given dataset and vintage.

Usage

```
list_income_brackets(dataset = c("ami", "fpl"), vintage = "2022")
```

Arguments

dataset	Character, either "ami" or "fpl"
vintage	Character, "2018" or "2022"

Value

Character vector of income bracket labels

Examples

```
list_income_brackets("ami", "2022")
list_income_brackets("fpl", "2018")
```

---

list_states *List Available States*

Description

Returns all state abbreviations available in the LEAD dataset.

Usage

```
list_states()
```

Value

Character vector of 51 state abbreviations (50 states + DC)

Examples

```
list_states()
```

---

load_census_tract_data *Load Census Tract Data*

Description

Load census tract demographics and utility service territory information with automatic fallback to CSV or OpenEI download.

Usage

```
load_census_tract_data(states = NULL, verbose = TRUE)
```

Arguments

states	Character vector of state abbreviations to filter by (optional)
verbose	Logical, print status messages (default TRUE)

Value

A tibble with columns:

- geoid: Census tract identifier
- state_abbr: State abbreviation
- county_name: County name
- tract_name: Tract name
- utility_name: Electric utility serving this tract
- Additional demographic columns

Examples

```
if (interactive()) {
  # Single state (requires census data download)
  nc_tracts <- load_census_tract_data(states = "NC")

  # Multiple states (regional)
  southeast <- load_census_tract_data(states = c("NC", "SC", "GA", "FL"))

  # Nationwide (all ~73,000 census tracts)
  us_tracts <- load_census_tract_data()  # No filter = all states
}
```

`load_cohort_data` *Load DOE LEAD Tool Cohort Data*

Description

Load household energy burden cohort data with automatic fallback:

1. Try local database
2. Fall back to local CSV files
3. Auto-download from OpenEI if neither exists
4. Auto-import downloaded data to database for future use

Usage

```
load_cohort_data(
  dataset = c("ami", "fpl"),
  states = NULL,
  counties = NULL,
  vintage = "2022",
  income_brackets = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

`dataset`	Character, either "ami" (Area Median Income) or "fpl" (Federal Poverty Line)
`states`	Character vector of state abbreviations to filter by (optional)
`counties`	Character vector of county names or FIPS codes to filter by (optional). County names are matched case-insensitively. Requires `states` to be specified.
`vintage`	Character, data vintage: "2018" or "2022" (default "2022")
`income_brackets`	Character vector of income brackets to filter by (optional)
`verbose`	Logical, print status messages (default TRUE)
`...`	Additional filter expressions passed to dplyr::filter() for dynamic filtering. Allows filtering by any column in the dataset using tidyverse syntax. Example: `households > 100, total_income > 50000`

Value

A tibble with columns:

- geoid: Census tract identifier
- income_bracket: Income bracket label
- households: Number of households

- total_income: Total household income ($)
- total_electricity_spend: Total electricity spending ($)
- total_gas_spend: Total gas spending ($)
- total_other_spend: Total other fuel spending ($)
- TEN: Housing tenure category (1=Owned free/clear, 2=Owned with mortgage, 3=Rented, 4=Occupied without rent). Enables analysis of energy burden differences between renters and owners.
- TEN-YBL6: Housing tenure crossed with year structure built (6 categories). Allows analysis of how building age and ownership status interact to affect energy burden (e.g., older rental units vs newer owner-occupied homes).
- TEN-BLD: Housing tenure crossed with building type (e.g., single-family, multi-unit). Enables analysis of energy burden across different housing structures and ownership patterns.
- TEN-HFL: Housing tenure crossed with primary heating fuel type (e.g., gas, electric, oil). Critical for analyzing how heating fuel choice and tenure status jointly influence energy costs and burden.

Examples

```
# Single state (fast, good for learning)
nc_ami <- load_cohort_data(dataset = "ami", states = "NC")

# Load specific vintage
nc_2018 <- load_cohort_data(dataset = "ami", states = "NC", vintage = "2018")



if (interactive()) {
  # Multiple states (regional analysis - requires data download)
  southeast <- load_cohort_data(dataset = "fpl", states = c("NC", "SC", "GA", "FL"))

  # Nationwide (all 51 states - no filter)
  us_data <- load_cohort_data(dataset = "ami", vintage = "2022")

# Filter to specific income brackets
low_income <- load_cohort_data(
  dataset = "ami",
  states = "NC",
  income_brackets = c("0-30% AMI", "30-50% AMI")
)

# Filter to specific counties within a state
triangle <- load_cohort_data(
  dataset = "fpl",
  states = "NC",
  counties = c("Orange", "Durham", "Wake")
)

# Or use county FIPS codes
orange <- load_cohort_data(
```

```
  dataset = "fpl",
  states = "NC",
  counties = "37135"
)

# Use dynamic filtering for custom criteria
high_burden <- load_cohort_data(
  dataset = "ami",
  states = "NC",
  households > 100,
  total_electricity_spend / total_income > 0.06
)

# Analyze energy burden by housing characteristics
# Compare renters vs owners by heating fuel type
nc_housing <- load_cohort_data(dataset = "ami", states = "NC")
library(dplyr)

# Group by tenure and heating fuel to analyze energy burden patterns
housing_analysis <- nc_housing %>%
  filter(!is.na(TEN), !is.na(`TEN-HFL`)) %>%
  group_by(TEN, `TEN-HFL`) %>%
  summarise(
    total_households = sum(households),
    avg_energy_burden = weighted.mean(
      (total_electricity_spend + total_gas_spend + total_other_spend) / total_income,
      w = households,
      na.rm = TRUE
    ),
    .groups = "drop"
  )
}
```

`nc_sample` *North Carolina Complete Energy Burden Sample Data*

Description

A comprehensive dataset containing energy burden data for all counties in North Carolina. This dataset includes both Federal Poverty Line (FPL) and Area Median Income (AMI) cohort data for 2018 and 2022 vintages, aggregated to the census tract × income bracket level.

Usage

```
nc_sample
```

Format

A named list with 4 data frames:

fpl_2018 Federal Poverty Line cohort data for 2018 (~10,805 rows)

fpl_2022 Federal Poverty Line cohort data for 2022 (~13,185 rows)

ami_2018 Area Median Income cohort data for 2018 (~6,484 rows)

ami_2022 Area Median Income cohort data for 2022 (~5,091 rows)

Each data frame contains:

geoid 11-digit census tract identifier (character)

income_bracket Income bracket category (character)

households Number of households in this cohort (numeric)

total_income Total household income in dollars (numeric)

total_electricity_spend Total electricity spending in dollars (numeric)

total_gas_spend Total gas spending in dollars (numeric)

total_other_spend Total other fuel spending in dollars (numeric)

Details

This sample data provides full state coverage for more comprehensive analysis, testing, and demonstrations. For lightweight quick demos, see `orange_county_sample`.

North Carolina (all 100 counties):

- 2018: 2,163 census tracts
- 2022: 2,642 census tracts (tract boundaries changed)

Income Brackets:

- FPL: 0-100%, 100-150%, 150-200%, 200-400%, 400%+
- AMI: Varies by vintage (4-6 categories)

Size: 1.3 MB compressed (.rda)

Source

U.S. Department of Energy Low-Income Energy Affordability Data (LEAD) Tool

- 2018 vintage: https://data.openei.org/submissions/573
- 2022 vintage: https://data.openei.org/submissions/6219

See Also

- `orange_county_sample` - Lightweight sample (94 KB) for quick demos
- `load_cohort_data` - Load data for any state with county filtering
- `compare_energy_burden` - Compare energy burden across vintages
- `calculate_weighted_metrics` - Calculate weighted metrics with grouping

Examples

```
# Load sample data
data(nc_sample)

# View structure
names(nc_sample)

# Analyze energy burden by county
library(dplyr)

# Extract county FIPS (first 5 digits of geoid)
nc_sample$fpl_2022 %>%
  mutate(county_fips = substr(geoid, 1, 5)) %>%
  group_by(county_fips, income_bracket) %>%
  summarise(
    households = sum(households),
   avg_energy_burden = sum(total_electricity_spend + total_gas_spend + total_other_spend) /
                       sum(total_income),
    .groups = "drop"
  ) %>%
  filter(county_fips == "37183")  # Wake County

# Compare urban vs rural counties
urban_counties <- c("37119", "37063", "37183")  # Mecklenburg, Durham, Wake
rural_counties <- c("37069", "37095", "37131")  # Franklin, Hyde, Northampton

nc_sample$fpl_2022 %>%
  mutate(
    county_fips = substr(geoid, 1, 5),
    region = case_when(
      county_fips %in% urban_counties ~ "Urban",
      county_fips %in% rural_counties ~ "Rural",
      TRUE ~ "Other"
    )
  ) %>%
  filter(region != "Other") %>%
  group_by(region, income_bracket) %>%
  summarise(
    households = sum(households),
    energy_burden = sum(total_electricity_spend + total_gas_spend + total_other_spend) /
                    sum(total_income),
    .groups = "drop"
  )
```

---

neb_func *Calculate Net Energy Burden (NEB)*

Description

Calculates Net Energy Burden with proper aggregation methodology via the Net Energy Return (Nh) framework. For individual households, NEB = EB = S/G. When aggregating across households (with weights), automatically uses the Nh method to avoid 1-5% aggregation errors.

Usage

```
neb_func(g, s, se = NULL, weights = NULL, aggregate = FALSE)
```

Arguments

g	Numeric vector of gross income values
s	Numeric vector of energy spending values
se	Optional numeric vector of effective energy spending (defaults to s)
weights	Optional numeric vector of weights for aggregation (e.g., household counts). When provided, uses Nh method: `1 / (1 + weighted.mean(nh, weights))`
aggregate	Logical, if TRUE forces aggregation even without weights (uses unweighted mean). Default FALSE for backwards compatibility.

Details

Individual Level: NEB = EB = S/G (mathematically identical)

Aggregation Modes:

1. **No aggregation** (default): Returns vector of individual NEB values

   ```
   neb_func(income, spending)  # Returns vector
   ```

2. **Weighted aggregation**: Automatically uses Nh method when weights provided

   ```
   neb_func(income, spending, weights = households)  # Returns single value
   ```

3. **Unweighted aggregation**: Use `aggregate = TRUE` for simple mean

   ```
   neb_func(income, spending, aggregate = TRUE)  # Returns single value
   ```

Why Nh Method? Avoids 1-5% error from naive averaging:

- **CORRECT**: `neb_func(g, s, weights = w)` → Uses Nh internally
- **WRONG**: `weighted.mean(s/g, w)` → Introduces bias

The Nh method: `1 / (1 + weighted.mean(nh, weights))` where `nh = (g-s)/se` uses arithmetic mean instead of harmonic mean, providing computational simplicity and numerical stability.

Value

- If `weights = NULL` and `aggregate = FALSE`: Numeric vector of individual NEB values (S/G)
- If `weights` provided or `aggregate = TRUE`: Single aggregated NEB value via Nh method

See Also

ner_func() for the Net Energy Return (Nh) calculation

energy_burden_func() for simple EB without aggregation support

Examples

```
# Individual household - returns vector
neb_func(50000, 3000)  # 0.06
neb_func(c(30000, 50000), c(3000, 3500))  # c(0.10, 0.07)

# Aggregation with weights - returns single value (CORRECT method)
incomes <- c(30000, 50000, 75000)
spending <- c(3000, 3500, 4000)
households <- c(100, 150, 200)
neb_func(incomes, spending, weights = households)

# Unweighted aggregation
neb_func(incomes, spending, aggregate = TRUE)

# Comparison: naive mean (WRONG) vs Nh method (CORRECT)
neb_naive <- weighted.mean(spending/incomes, households)  # Biased
neb_correct <- neb_func(incomes, spending, weights = households)  # Correct
abs(neb_naive - neb_correct) / neb_correct  # ~1-5% error
```

---

ner_func *Calculate Net Energy Return (Nh)*

Description

Calculates the Net Energy Return using the formula Nh = (G - S) / Se, where G is gross income, S is energy spending, and Se is effective energy spending. This metric is the preferred aggregation variable as it properly accounts for harmonic mean behavior when aggregating across households.

Usage

```
ner_func(g, s, se = NULL)
```

Arguments

g	Numeric vector of gross income values
s	Numeric vector of energy spending values
se	Optional numeric vector of effective energy spending (defaults to s)

Details

The Net Energy Return is mathematically related to energy burden by: E_b = 1 / (Nh + 1), or equivalently: Nh = (1/E_b) - 1

Why use Nh for aggregation?

For individual household data, the Nh method enables simple arithmetic weighted mean aggregation:

- **Via Nh**: `neb = 1 / (1 + weighted.mean(nh, weights))` (arithmetic mean)
- **Direct EB**: `neb = 1 / weighted.mean(1/eb, weights)` (harmonic mean)

Computational advantages of the arithmetic mean approach:

1. **Simpler to compute** - Uses standard `weighted.mean()` function
2. **More numerically stable** - Avoids division by very small EB values (e.g., 0.01)
3. **More interpretable** - "Average net return per dollar spent on energy"
4. **Prevents errors** - Makes it obvious you can't use arithmetic mean on EB directly

For cohort data (pre-aggregated totals), direct calculation `sum(S)/sum(G)` is mathematically equivalent to the Nh method but simpler.

The 6% energy burden poverty threshold corresponds to Nh $\geq$ 15.67.

Value

Numeric vector of Net Energy Return (Nh) values

Examples

```
# Calculate Net Energy Return
gross_income <- 50000
energy_spending <- 3000
nh <- ner_func(gross_income, energy_spending)

# Convert back to energy burden
energy_burden <- 1 / (nh + 1)
```

---

orange_county_sample *Orange County NC Energy Burden Sample Data*

Description

A sample dataset containing energy burden data for Orange County, North Carolina (FIPS code 37135). This dataset includes both Federal Poverty Line (FPL) and Area Median Income (AMI) cohort data for 2018 and 2022 vintages.

Usage

```
orange_county_sample
```

Format

A named list with 4 data frames:

fpl_2018 Federal Poverty Line cohort data for 2018 (135 rows)

fpl_2022 Federal Poverty Line cohort data for 2022 (206 rows)

ami_2018 Area Median Income cohort data for 2018 (259 rows)

ami_2022 Area Median Income cohort data for 2022 (149 rows)

Each data frame contains:

geoid 11-digit census tract identifier (character)

income_bracket Income bracket category (character)

households Number of households in this cohort (numeric)

total_income Total household income in dollars (numeric)

total_electricity_spend Total electricity spending in dollars (numeric)

total_gas_spend Total gas spending in dollars (numeric)

total_other_spend Total other fuel spending in dollars (numeric)

Details

This sample data is provided for quick demos, testing, and vignettes without requiring a large download. For full state or national analysis, use `load_cohort_data()` to download complete datasets from OpenEI.

Orange County NC (Chapel Hill, Carrboro, Hillsborough):

- 2018: 27 census tracts
- 2022: 42 census tracts (tract boundaries changed)

Income Brackets:

- FPL: 0-100%, 100-150%, 150-200%, 200-400%, 400%+
- AMI: very_low, low_mod, mid_high (aggregated from 6 AMI categories)

Source

U.S. Department of Energy Low-Income Energy Affordability Data (LEAD) Tool

- 2018 vintage: https://data.openei.org/submissions/573
- 2022 vintage: https://data.openei.org/submissions/6219

See Also

- `load_cohort_data` - Load full datasets for any state
- `compare_energy_burden` - Compare energy burden across vintages
- `calculate_weighted_metrics` - Calculate weighted metrics with grouping

Examples

```
# Load sample data
data(orange_county_sample)

# View structure
names(orange_county_sample)

# Quick analysis of 2022 FPL data
library(dplyr)
orange_county_sample$fpl_2022 %>%
  group_by(income_bracket) %>%
  summarise(
    households = sum(households),
   avg_energy_burden = sum(total_electricity_spend + total_gas_spend + total_other_spend) /
                         sum(total_income)
  )
```

`print.energy_burden_comparison` *Print Comparison Summary*

Description

Pretty-print a comparison table from compare_energy_burden()

Usage

```
## S3 method for class 'energy_burden_comparison'
print(x, ...)
```

Arguments

`x`	Comparison result from compare_energy_burden()
`...`	Additional arguments (not used)

Value

Returns `x` invisibly for use in pipe chains.

to_big *Format Large Numbers with Thousand Separators*

Description

Converts numeric values to formatted strings with thousand separators (commas).

Usage

```
to_big(x)
```

Arguments

x	Numeric vector to format

Value

Character vector of formatted numbers

Examples

```
# Format large numbers
to_big(c(1000, 25000, 1000000))
```

to_billion_dollar *Format Dollar Amounts in Billions*

Description

Converts large dollar values to billions format with dollar sign prefix. Values less than 1 billion are shown in millions.

Usage

```
to_billion_dollar(x, suffix = " billion", override_to_k = TRUE)
```

Arguments

x	Numeric vector to format
suffix	Character string to append after "billion" (default: " billion")
override_to_k	Logical (currently unused, kept for compatibility)

Value

Character vector of formatted dollar amounts with "billion" or "m" suffix

Examples

```
# Format in billions
to_billion_dollar(c(5000000, 1000000000, 2500000000))
```

---

to_dollar *Format Number as Dollar Amount*

Description

Converts numeric values to formatted dollar strings with appropriate decimal places and thousand separators.

Usage

```
to_dollar(x, latex = FALSE)
```

Arguments

x	Numeric vector to format
latex	Logical indicating whether to escape dollar sign for LaTeX (default: FALSE)

Value

Character vector of formatted dollar amounts

Examples

```
# Format dollar amounts
to_dollar(c(1000, 2500.50, 10000))

# LaTeX-escaped format
to_dollar(c(1000, 2500.50), latex = TRUE)
```

---

to_million *Format Numbers in Millions*

Description

Converts large numeric values to millions format with appropriate suffix. Values less than 1 million are shown in thousands.

Usage

```
to_million(x, suffix = " million", override_to_k = TRUE)
```

Arguments

x	Numeric vector to format
suffix	Character string to append after "million" (default: " million")
override_to_k	Logical indicating whether to show values < 1M as thousands (default: TRUE)

Value

Character vector of formatted numbers with "million" or "k" suffix

Examples

```
# Format in millions
to_million(c(5000, 1000000, 2500000))
```

---

to_percent *Format Number as Percentage*

Description

Converts numeric values to formatted percentage strings with no decimal places by default.

Usage

```
to_percent(x, latex = FALSE)
```

Arguments

x	Numeric vector to format (as proportions, not percentages)
latex	Logical indicating whether to escape percent sign for LaTeX (default: FALSE)

Value

Character vector of formatted percentages

Examples

```
# Format percentages
to_percent(c(0.25, 0.50, 0.123))

# LaTeX-escaped format
to_percent(c(0.25, 0.50), latex = TRUE)
```

Index