

# Package ‘indexDesignWindows’

February 17, 2026

**Type** Package

**Title** Replication Package: Redesigning Historical Windows in Index-Based Insurance

**Version** 0.0.0.9000

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

**Maintainer** Francis Tsiboe <ftsiboe@hotmail.com>

**Creator** Francis Tsiboe

**Description** Replication package for a study evaluating alternative historical window designs used to construct the Pasture, Rangeland, and Forage (PRF) rainfall index under the U.S. Federal Crop Insurance Program (FCIP). The package systematically compares index designs based on varying lengths of historical climate data to assess implications for index stability, spatial equity, indemnity accuracy, and policy performance. It provides reproducible workflows, pre-processed outputs, and visualization tools to support robustness analysis of index-based insurance products in both U.S. and international contexts.

**License** GPL-3 + file LICENSE

**URL** <https://github.com/ftsiboe/indexDesignWindows>

**BugReports** <https://github.com/ftsiboe/indexDesignWindows/issues>

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**VignetteBuilder** knitr

**Depends** R (>= 4.1.0)

**Imports** data.table, ggplot2, ggrepel, gridExtra, gtable, sf, urbnmapr

**Remotes** github::ftsiboe/rfcipDemand, github::UrbanInstitute/urbanmapr

**Suggests**

dplyr, tidyr, knitr, rmarkdown, withr, stats, rfcipDemand, piggyback, classInt, testthat (>= 3.0.0)

**LazyData** true

**Cite-us** If you find it useful, please consider starring the repository and citing the following studies

- Tsiboe, F. and Turner, D. (2025). ``Incorporating buy-up price loss coverage into the United States farm safety net.'' *Applied Economic Perspectives and Policy*.
- Tsiboe, F., et al. (2025). ``Risk reduction impacts of crop insurance in the United States.'' *Applied Economic Perspectives and Policy*.
- Gaku, S. and Tsiboe, F. (2024). Evaluation of alternative farm safety net program combination strategies. *Agricultural Finance Review*.

## Contents

add_break_categories . . . . .	2
compute_balance . . . . .	3
ers_theme . . . . .	5
get_price_indices . . . . .	5
get_study_releases . . . . .	7
plot_fcip_main_outcomes . . . . .	8
plot_liability_and_acres . . . . .	10
plot_prf_statistics . . . . .	12
plot_us_states_choropleth . . . . .	13
policytime . . . . .	15
reshape_statistics . . . . .	16
setup_environment . . . . .	17

## Index

19

---

**add\_break\_categories**    *Add Categorical Breaks to a Numeric Variable*

---

### Description

Classifies a numeric variable into categorical intervals using either Fisher-Jenks natural breaks (via **classInt**) or user-specified break points, and appends the resulting factor variable to the input data.

### Usage

```
add_break_categories(
  data,
  variable,
  break_n = NULL,
  break_levels = NULL,
  break_labels = NULL
)
```

### Arguments

<b>data</b>	A data frame containing the variable to be classified.
<b>variable</b>	Character scalar giving the name of the numeric column in <b>data</b> to classify.
<b>break_n</b>	Integer number of classes to compute when using Fisher-Jenks breaks. Required when <b>break_levels</b> is <b>NULL</b> .
<b>break_levels</b>	Optional numeric vector of break points. If supplied, these are used directly and <b>break_n</b> is ignored.
<b>break_labels</b>	Optional character vector of labels to use for the resulting factor. If <b>NULL</b> , labels are generated from <b>break_levels</b> .

## Details

The function supports two main workflows:

- **Automatic breaks:** If `break_levels` is `NULL`, Fisher-Jenks natural breaks are computed using `classInt::classIntervals()` with `style = "fisher"`. In this case, `break_n` must be supplied.
- **Manual breaks:** If `break_levels` is provided, these are used directly and `break_n` is ignored. If `break_labels` is `NULL`, labels are generated from `break_levels` in the form "`a - b`" (two decimal places).

Intervals are constructed with `include.lowest = TRUE` and `right = FALSE`, so they are left-closed and right-open.

## Value

The original data with an additional factor column named `<variable>_cat` containing the categorical breaks.

<code>compute_balance</code>	<i>Distributional balance: one basis vs multiple comparison variables (long format)</i>
------------------------------	---

## Description

For a given **basis** numeric column (`col_x`) and one or more **comparison** numeric columns (`col_y`), this function computes a rich set of diagnostics describing bias, scale, agreement, and distributional differences between the basis and each comparison variable.

Results are returned in **long format** with one row per group (defined by `by`) and per comparison variable (`y_level`).

Metrics include:

- **Sample size & moments**
  - `n`: number of non-missing pairs
  - `mean_x`, `mean_y`
  - `var_x`, `var_y`
  - `sd_x`, `sd_y`
  - `cv_x`, `cv_y` (coefficients of variation)
- **Bias & scale**
  - `mean_bias` = `mean_x` - `mean_y`
  - `mae` = `mean(|x - y|)`
  - `rmse` = `sqrt(mean((x - y)^2))`
  - `smape` =  $2 * \text{mean}(|x - y| / (|x| + |y|))$ , with safe handling when denominator is zero
  - `ratio_means` = `mean_x` / `mean_y`
  - `cv_ratio` = `cv_x` / `cv_y`
  - `mpd` = mean percentage difference:

$$(((\bar{x} - \bar{y})/\bar{y}) - 1) * 100$$

- **Balance / distribution comparison**

- msd = mean standardized difference:  $(\text{mean}_x - \text{mean}_y) / \sqrt{(\text{sd}_x^2 + \text{sd}_y^2) / 2}$
- var\_ratio =  $\text{var}_x / \text{var}_y$
- ks\_stat = Kolmogorov-Smirnov D-statistic (two-sample)
- emd = 1D Wasserstein / Earth Mover's Distance, approximated as mean absolute difference between quantiles of x and y
- hellinger = Hellinger distance computed from common histograms
- pvalue\_mean = p-value for equality of means (paired t-test)
- pvalue\_ks = p-value from Kolmogorov-Smirnov test
- pvalue\_var = p-value from F-test of equal variances
- pvalue\_levene = p-value from Levene test (mean-based)
- pvalue\_brown\_forsythe = p-value from Brown-Forsythe test
- pvalue\_kruskal\_wallis = p-value from Kruskal-Wallis test

- **Zero-mass / structural zeros**

- zero\_share\_x = proportion of observations with  $x == 0$
- zero\_share\_y = proportion of observations with  $y == 0$
- zero\_share\_diff =  $\text{zero\_share}_x - \text{zero\_share}_y$
- pvalue\_zero\_mcnenar = p-value from McNemar test on the paired zero/non-zero indicators

- **Bounded-scale / logit-space (for (0, 1) variables)**

- mean\_logit\_x, mean\_logit\_y = means of logit-transformed x and y
- mean\_logit\_diff =  $\text{mean}(\text{logit}_x - \text{logit}_y)$
- rmse\_logit =  $\sqrt{\text{mean}((\text{logit}_x - \text{logit}_y)^2)}$
- pvalue\_mean\_logit = p-value from paired t-test on logit\_x vs logit\_y

These logit-based metrics are computed only if all non-missing x and y are in the interval (0, 1). Otherwise, they are returned as NA.

- **Agreement / association**

- pearson\_cor = Pearson correlation
- spearman\_cor = Spearman rank correlation
- kendall\_cor = Kendall's tau
- ccc = Lin's Concordance Correlation Coefficient
- pvalue\_pearson\_cor
- pvalue\_spearman\_cor
- pvalue\_kendall\_cor

- **Quantile / tail behavior**

- Quantiles at  $p = 0.10, 0.25, 0.50, 0.75, 0.90$  for x and y: q10\_x, q25\_x, q50\_x, q75\_x, q90\_x q10\_y, q25\_y, q50\_y, q75\_y, q90\_y
- Differences: q10\_diff, q25\_diff, q50\_diff, q75\_diff, q90\_diff computed as  $(q_x - q_y)$

## Usage

```
compute_balance(dt, col_x, col_y, by = NULL)
```

## Arguments

dt	A <code>data.frame</code> or <code>data.table</code> .
col_x	Character scalar. Name of the <b>basis</b> numeric column.
col_y	Character vector. Names of one or more <b>comparison</b> numeric columns. The basis variable <code>col_x</code> is always compared against each variable in <code>col_y</code> .
by	Optional character vector of grouping variables. If <code>NULL</code> (default), all rows are treated as one group.

## Value

A `data.table` in **long format** with columns: `c(by, "y_level", "n", "mean_x", "mean_y", "var_x", "var_y", "sd_x", "sd_y", "cv_x", "cv_y", "cv_ratio", "mean_bias", "mae", "rmse", "smape", "ratio_means", "mpd", "msd", "var_ratio", "ks_stat", "emd", "hellinger", "pvalue_mean", "pvalue_ks", "pvalue_var", "pvalue_levene", "pvalue_brown_forsythe", "pvalue_kruskal_wallis", "zero_share_x", "zero_share_y", "zero_share_diff", "pvalue_zero_mc nemar", "mean_logit_x", "mean_logit_y", "mean_logit_diff", "rmse_logit", "pvalue_mean_logit", "pearson_cor", "spearman_cor", "kendall_cor", "ccc", "pvalue_pearson_cor", "pvalue_spearman_cor", "pvalue_kendall_cor", "q10_x", "q25_x", "q50_x", "q75_x", "q90_x", "q10_y", "q25_y", "q50_y", "q75_y", "q90_y", "q10_diff", "q25_diff", "q50_diff", "q75_diff", "q90_diff")`.

---

ers\_theme

*ERS Theme*

---

## Description

ERS Theme

## Usage

```
ers_theme()
```

## Source

copied from <https://github.com/USDA-REE-ERS/MTED-Theme> on 08/01/2025

## Examples

```
ggplot2::ggplot() + ers_theme()
```

`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):

- `nassSurveyPriceReceivedIndex` (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_study_releases`      *Download all assets from a GitHub release with rate limiting*

## Description

Downloads all files attached to a specified GitHub release tag while **throttling requests** to avoid GitHub rate limits and abuse protection. This helper is designed for releases containing many or large assets (e.g., `.rds` outputs generated on HPC systems).

## Usage

```
get_study_releases(
  owner,
  repository,
  release_tag,
  output_directory = NULL,
  github_token = NULL,
  sleep_seconds = 3,
  max_rounds = 3
)
```

## Arguments

<code>owner</code>	Character string giving the GitHub repository owner (e.g., "ftsiboe").
<code>repository</code>	Character string giving the GitHub repository name (e.g., "indexDesignWindows").
<code>release_tag</code>	Character string specifying the GitHub release tag whose assets should be downloaded.
<code>output_directory</code>	Optional character string specifying the local directory where release assets should be saved. Defaults to <code>data-raw/releases/{release_tag}</code> .
<code>github_token</code>	Optional GitHub personal access token (PAT). Passed to <b>piggyback</b> via <code>.token</code> . Strongly recommended.
<code>sleep_seconds</code>	Numeric scalar giving the number of seconds to pause between individual file downloads. Increasing this value reduces the likelihood of triggering GitHub rate limits.
<code>max_rounds</code>	Integer giving the maximum number of retry rounds. Each round attempts to download any files still missing locally.

## Details

The function downloads assets incrementally, pauses between requests, and retries failed downloads across multiple rounds. Already-downloaded files are skipped, allowing the function to safely resume after interruptions or rate-limit errors.

The function:

1. Constructs a default output directory (`data-raw/releases/{release_tag}`) if none is supplied.
2. Queries GitHub once to obtain the list of release assets.
3. Downloads assets **one at a time** using **piggyback**.
4. Pauses for `sleep_seconds` between downloads to reduce request bursts.
5. Retries failed or missing downloads for up to `max_rounds`.
6. Skips files that already exist locally.

This approach is especially useful when GitHub returns repeated HTTP 403 (Forbidden) errors during bulk downloads.

Authentication via a GitHub personal access token (PAT) is strongly recommended, even for public repositories.

---

## Value

Invisibly returns NULL. Files are downloaded for their side effects.

---

## *plot\_fcip\_main\_outcomes*

*Plot FCIP Main Outcomes (Faceted Bar Charts)*

---

## Description

The function expects `data` to contain:

- a numeric/factor year column (specified via `column_year`);
- a numeric value column (bar heights);
- a categorical "outcome" column (specified via `column_outcome`) used for facetting; and
- an optional grouping column (specified via `column_grouping`) used to color/ fill bars and determine legend entries.

For legend ordering, the function ranks groups using the subset of rows from the most recent year *within the facet whose label equals "(B) Liability amount in U.S. dollars"*. If your outcome labels differ, you may want to harmonize them or adjust the code that builds `labs`.

## Usage

```
plot_fcip_main_outcomes(
  data,
  column_outcome,
  column_year,
  column_grouping = NULL,
  time_scale_theme = NULL,
  general_theme = NULL,
  palette = c("#003524", "#00583D", "#A0BD78", "#BED73B", "#D7E5C8", "#FEF389",
             "#FFC425", "#BE5E27", "#9DD9F7", "#51ABA0", "#0F374B")
)
```

## Arguments

<code>data</code>	A <code>data.frame</code> (or <code>data.table</code> ) with at least the columns: <code>value</code> , the column named by <code>column_year</code> , the column named by <code>column_outcome</code> , and (optionally) the column named by <code>column_grouping</code> .
<code>column_outcome</code>	Character scalar. Name of the column in <code>data</code> that defines facet panels (e.g., outcome labels such as "(A) Net reported acres" and "(B) Liability amount in U.S. dollars").
<code>column_year</code>	Character scalar. Name of the year column in <code>data</code> (x-axis, typically the FCIP commodity year).
<code>column_grouping</code>	Character scalar or <code>NULL</code> . Name of the grouping column in <code>data</code> used for fill/color and legend entries. If <code>NULL</code> , all bars are treated as a single group.
<code>time_scale_theme</code>	Optional <code>ggplot2</code> scale/theme element controlling the x-axis (e.g., <code>scale_x_continuous(...)</code> ). If <code>NULL</code> , a default scale is applied using the unique values of the supplied year column.
<code>general_theme</code>	Optional <code>ggplot2</code> theme applied to the figure. If <code>NULL</code> , <code>ers_theme()</code> is used and further tweaked (text sizes, legend layout, facet strips). Requires that <code>ers_theme()</code> is available in scope.
<code>palette</code>	Optional character vector of HEX colors used by <code>scale_fill_manual()</code> . If provided, it is applied to the grouping legend.

## Details

Creates a faceted bar chart of FCIP outcomes over time by a chosen grouping, with a shared x-axis (commodity year) and a legend. Facets are determined by an "outcome" column in `data` (e.g., liability vs. acres), allowing one or more panels depending on the unique values present.

Internally, the function:

1. Copies the columns named by `column_outcome`, `column_grouping`, and `column_year` into standard placeholders used in aesthetics.
2. Builds a ranking of groups from the latest year within the facet whose label equals "(B) Liability amount in U.S. dollars", then maps that ranking to legend order.
3. Draws a bar chart of `value ~ year`, colored/filled by group, and `facet_wrap()`s by outcome (free y-scales).
4. Applies the supplied or default x-axis scale and theme, plus optional manual fill palette.

**Value**

A ggplot object.

**Note**

The parameter names use `commodity_*` (typo preserved for backward compatibility). Consider aliasing/renaming in a future version.

**plot\_liability\_and\_acres**

*Plot Liability and Net Reported Acres Faceted by a Grouping Variable*

**Description**

Produces a two-panel bar chart:

1. **Panel A** - Liability (in U.S.\ dollars) by year and group.
2. **Panel B** - Net reported acres (in acres) by year and group.

The panels share the same x-axis (commodity years) and a single legend that is displayed beneath the figure.

**Usage**

```
plot_liability_and_acres(
  data,
  grouping_variable,
  grouping_name = NULL,
  time_scale_theme = NULL,
  general_theme = NULL,
  label_liability = "Billion U.S. dollars\n",
  label_net_reported_acres = "Million acres\n",
  palette = c("#003524", "#00583D", "#A0BD78", "#BED73B", "#D7E5C8", "#FEF389",
             "#FFC425", "#BE5E27", "#9DD9F7", "#51ABA0", "#0F374B")
)
```

**Arguments**

<code>data</code>	A <code>data.frame</code> containing at least these columns: <ul style="list-style-type: none"> <li>• <code>commodity_year</code> - numeric or factor; the x-axis.</li> <li>• <code>value</code> - numeric; the bar heights.</li> <li>• <code>variable</code> - factor with the levels "(A) Liability in U.S. dollars" and "(B) Net reported acres".</li> <li>• The column referenced by <code>grouping_variable</code>.</li> </ul>
<code>grouping_variable</code>	A character scalar giving the name of the column in <code>data</code> that defines the groups (e.g.\ "commodity", "state").
<code>grouping_name</code>	Optional character scalar used as the legend title. Defaults to the value of <code>grouping_variable</code> if NULL.

time_scale_theme	Optional ggplot2 scale or theme element that controls the x-axis breaks/labels. If NULL, the function applies scale_x_continuous(breaks = unique(data\$commodity_year), labels = unique(data\$commodity_year)).
general_theme	Optional ggplot2 theme applied to both panels. If NULL, ers_theme() is used with additional tweaks for font sizes, legend layout, and facet strips.
label_liability	Y-axis label for Panel A. Default "Billion U.S. dollars\n".
label_net_reported_acres	Y-axis label for Panel B. Default "Million acres\n".
palette	A character vector of hex colors used to fill the bars. The default is an 11-color palette aligned with ERS/NDSU branding.

## Details

Internally the function:

- 1.Duplicates the column named by grouping\_variable into data\$grouping\_variable for convenient aesthetic mapping.
- 2.Builds two separate ggplot bar charts (one per variable level), applies user themes and palettes, and hides their legends.
- 3.Extracts a shared legend and x-axis grob with gtable\_filter().
- 4.Assembles everything with gridExtra::grid.arrange().

## Value

A named list with five objects:

```
fig A gtable containing the assembled two-panel figure, shared x-axis label, and bottom legend.
figA The ggplot object for Panel A.
figB The ggplot object for Panel B.
ldgnd The extracted legend as a gtable.
xlabT The shared x-axis grob as a gtable.
```

## Examples

```
## Not run:
# Group by commodity with a custom x-axis theme
plot_liability_and_acres(
  data = data_comm,
  grouping_variable = "commodity",
  grouping_name = "Commodity",
  time_scale_theme = ggplot2::scale_x_continuous(
    breaks = seq(2008, 2024, 2),
    labels = seq(2008, 2024, 2)
  ),
  general_theme = ggplot2::theme_minimal()
)

## End(Not run)
```

`plot_prf_statistics`    *Plot PRF Summary Statistics on the Official Grid*

## Description

Creates a choropleth map of Pasture, Rangeland, Forage (PRF) statistics on the official PRF grid. Optionally facets the map by a disaggregate variable (e.g., data source: Official vs. Study Approximation).

## Usage

```
plot_prf_statistics(
  data,
  outcome_variable,
  plot_title,
  prf_polygon = NULL,
  disaggregate_variable = NULL,
  spatial_unit = "prf_grid",
  palette = c("#FEF9C5", "#FFC425", "#FEF389", "#E7F2B4", "#D7E5C8", "#C5DE91",
             "#BED73B", "#A0BD78", "#00583D", "#0F374B"),
  na.value = "white"
)
```

## Arguments

<code>data</code>	A data frame containing at least <code>grid_id</code> and the variable specified in <code>outcome_variable</code> . If <code>disaggregate_variable</code> is used, it must also be present in <code>data</code> .
<code>outcome_variable</code>	Character scalar giving the name of the variable in <code>data</code> to map (e.g., a binned category such as <code>"mean_cat"</code> ).
<code>plot_title</code>	Character scalar used as the main plot title.
<code>prf_polygon</code>	Optional <code>sf</code> object containing PRF grid polygons with a <code>grid_id</code> column. If <code>NULL</code> , the function calls <code>get_official_prf_polygon()</code> .
<code>disaggregate_variable</code>	Optional character scalar giving the name of a column in <code>data</code> used for facetting (e.g., <code>"disaggregate"</code> or <code>"source"</code> ). If <code>NULL</code> , no facets are drawn.
<code>spatial_unit</code>	Character scalar. One of <code>"prf_grid"</code> or <code>"county"</code> . Determines which polygons are used and which join key is required.
<code>palette</code>	Character vector of hex color codes used for the fill scale. The default is a 10-color sequential/diverging palette tailored to PRF maps.
<code>na.value</code>	Fill color for the background (non-data areas) and missing values.

## Details

The function:

1. Loads the official PRF grid polygons via `get_official_prf_polygon()` if `prf_polygon` is not supplied.
2. Filters PRF polygons to those with `grid_id` present in `data`.

3. Transforms the PRF polygons to match the CRS of the state boundaries from `urbnmapr::get_urbn_map(map = "states")`.
4. Joins data to the PRF polygons by `grid_id`.
5. Filters out rows with missing values in `outcome_variable`.
6. Plots a choropleth map using the supplied palette, overlaying U.S. state boundaries (excluding Alaska and Hawaii; FIPS 2 and 15).
7. Optionally facets the plot by `disaggregate_variable`, with two columns of panels.

The function assumes that `data` contains a column named `grid_id` matching the PRF grid IDs, and a column whose name is given by `outcome_variable`. When `disaggregate_variable` is provided, it should also be a column in `data`.

## Value

A `ggplot` object representing the PRF statistics map, optionally with facets by `disaggregate_variable`.

## Examples

```
## Not run:
p <- plot_prf_statistics(
  data              = df,
  outcome_variable = "mean_cat",
  plot_title       = "PRF Index Mean (2016-2024)",
  disaggregate_variable = "disaggregate"
)
print(p)

## End(Not run)
```

## plot\_us\_states\_choropleth

*Plot a US States Choropleth With State-Level Labels*

## Description

Builds a U.S. states choropleth by joining user-supplied state data (by FIPS) to the `urbnmapr` states basemap. States are filled using a categorical column, and each state is labeled with a two-line label (state abbreviation + a value column). Labels for small states are placed with repelled text to reduce overlap, with special nudges for Vermont and New Hampshire. Optionally, a table grob can be embedded in the lower-left of the map.

## Usage

```
plot_us_states_choropleth(
  data,
  value_label,
  category_label,
  legend_title = NULL,
  state_text_colors = c("black", "black"),
  palette = c("#BE5E27", "#FFC425", "#FEF389", "#BED73B", "#A0BD78", "#00583D",
```

```

  "#003524", "#D7E5C8", "#9DD9F7", "#51ABA0", "#0F374B"),
  table_grob = NULL,
  label_size = 2.5,
  na.value = "white",
  keep_all_states = FALSE
)

```

## Arguments

<code>data</code>	A <code>data.frame</code> containing (at minimum) a <code>state_code</code> column of numeric state FIPS codes used to join to the <code>urbnmapr</code> states map. It must also contain the columns referenced by <code>value_label</code> and <code>category_label</code> .
<code>value_label</code>	Character scalar. Name of the column in <code>data</code> used to populate the numeric/value portion of the state label (second line). Values are inserted as-is (not rounded/formatted) and are coerced through <code>as.data.frame(sf_object)[, value_label]</code> .
<code>category_label</code>	Character scalar. Name of the column in <code>data</code> used for choropleth fills. Rows with missing values in this column are dropped unless you set <code>keep_all_states</code> = TRUE (which still draws all states, but only colors those present after filtering).
<code>legend_title</code>	Character. Legend title for the fill scale. If <code>NULL</code> , the title is set to an empty string (i.e., no visible title).
<code>state_text_colors</code>	Character vector of length 2. The first color is used for labels on non-small states; the second color is used for repelled labels on small states.
<code>palette</code>	Character vector of hex colours used by <code>scale_fill_manual()</code> for the categories in <code>category_label</code> .
<code>table_grob</code>	Optional grob to embed on the map using <code>annotation_custom()</code> (e.g., <code>gridExtra::tableGrob(...)</code> ). If <code>NULL</code> , no table is added.
<code>label_size</code>	Numeric. Text size passed to <code>geom_sf_text()</code> and <code>geom_text_repel()</code> .
<code>na.value</code>	Character. Fill color used for missing values when drawing the basemap (and as <code>na.value</code> in <code>scale_fill_manual()</code> ).
<code>keep_all_states</code>	Logical. If <code>TRUE</code> , first draws the full U.S. states basemap in <code>na.value</code> (so states not present in <code>data</code> remain visible). If <code>FALSE</code> , only the joined/filtered states are drawn.

## Details

The function:

- Loads the states basemap via `urbnmapr::get_urbn_map(map = "states", sf = TRUE)` and creates a numeric `state_code` from `state_fips`.
- Left-joins `data` on `state_code`, then filters out rows where `category_label` is NA.
- Builds a two-line label of STATE\_ABB plus the `value_label` column.
- Flags "small" states as those with area < 50,000 km<sup>2</sup> (computed after transforming to EPSG:5070 for equal-area calculations).
- Uses `geom_sf_text()` for non-small states and `ggrepel::geom_text_repel()` for small states, nudging labels left/right based on whether the centroid lies east or west of the map midpoint; Vermont and New Hampshire get a custom nudge.
- Optionally adds `table_grob` at fixed map coordinates using `annotation_custom()`.

**Value**

A ggplot object (choropleth of U.S. states) with state labels and an optional embedded table grob.

---

policytime

*Add U.S. Farm Policy Vertical Lines and Labels to a ggplot*

---

**Description**

This function overlays vertical lines and text labels on a ggplot object to mark major U.S. agricultural policy events (e.g., Farm Bills or Acts). The vertical lines are drawn at specific years, and labels are positioned based on values provided in the pty vector.

**Usage**

```
policytime(pty, plot, size)
```

**Arguments**

pty	A numeric vector of y-axis positions for label placement. The length of the vector determines which policy annotations are added:
	<ul style="list-style-type: none"><li>• 1st: 1980 Act</li><li>• 2nd: 1994 Act</li><li>• 3rd: 1996 Farm Bill</li><li>• 4th: 2000 Agricultural Risk Protection Act</li><li>• 5th: 2008 Farm Bill</li><li>• 6th: 2014 Farm Bill</li><li>• 7th: 2018 Farm Bill</li></ul>
plot	A ggplot object to which policy lines and labels will be added.
size	A numeric value indicating the text size of the policy labels.

**Details**

The lines and labels are added in brown color with dashed lines (lty=5), and labels are rotated vertically. Labels are only drawn if the corresponding index exists in pty.

**Value**

A ggplot object with added vertical dashed lines and corresponding text labels for each policy year provided.

**Examples**

```
## Not run:  
base_plot <- ggplot(data, aes(x = commodity_year, y = value)) +  
  geom_line()  
policy_positions <- c(10, 15, 20, 25, 30, 35, 40)  
annotated_plot <- policytime(policy_positions, base_plot, size = 3)  
print(annotated_plot)  
  
## End(Not run)
```

**reshape\_statistics**      *Reshape and Label Balance Statistics*


---

## Description

Reshapes selected balance statistics from a data frame into long format and assigns descriptive labels to each statistic disaggregate. This function is commonly used to prepare datasets for balance comparison plots or summary tables.

## Usage

```
reshape_statistics(
  data,
  statistic_variables,
  disaggregate_labels,
  rename = NULL,
  panel_variables
)
```

## Arguments

<code>data</code>	A data frame containing <code>grid_id</code> and the statistic variables to be reshaped.
<code>statistic_variables</code>	Character vector of column names in <code>data</code> that contain the statistics to reshape.
<code>disaggregate_labels</code>	Character vector of descriptive labels corresponding to <code>statistic_variables</code> . Must be the same length and in the same order.
<code>rename</code>	Optional character scalar. If supplied, the output column value will be renamed to this string (e.g., "std_diff" or "index").
<code>panel_variables</code>	panel variables

## Details

The function:

1. Validates that `statistic_variables` and `disaggregate_labels` have equal length.
2. Checks that all specified statistic variables exist in `data`.
3. Selects `grid_id` and the variables listed in `statistic_variables`.
4. Converts the data from wide to long format using `tidyr::gather()`, creating columns: `grid_id`, `disaggregate`, and `value`.
5. Converts `disaggregate` to a factor and replaces raw column names with human-readable labels using `disaggregate_labels`.
6. Optionally renames the `value` column using the `rename` argument.

## Value

A long-format data frame with the columns:

**grid\_id** Identifier for the spatial/grid unit.

**disaggregate** Factor indicating the origin of each statistic, labeled using `disaggregate_labels`.

**value** Numeric value of the statistic, or renamed to `rename` if provided.

`setup_environment`      *Setup Project Environment*

## Description

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

## Usage

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

## Arguments

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

## Details

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

It also sets the following options:

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

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

## Value

A list with:

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

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

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

**seed** Seed value used for RNG.

# Index

add\_break\_categories, 2  
compute\_balance, 3  
ers\_theme, 5  
get\_price\_indices, 5  
get\_study\_releases, 7  
plot\_fcip\_main\_outcomes, 8  
plot\_liability\_and\_acres, 10  
plot\_prf\_statistics, 12  
plot\_us\_states\_choropleth, 13  
policytime, 15  
reshape\_statistics, 16  
setup\_environment, 17