Package 'paneldesc'

July 21, 2026

Type Package

Title Descriptive Analysis and Visualization for Panel Data

Version 0.2.0

Date 2026-07-21

Description Provides a comprehensive set of tools for describing and visualizing panel data structures, as well as for summarizing and visualizing variables within a panel data context.

License GPL-3

URL https://github.com/dtereshch/paneldesc, https://dtereshch.github.io/paneldesc-guides/

BugReports https://github.com/dtereshch/paneldesc/issues

LazyData true

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Dmitrii Tereshchenko [aut, cre] (ORCID: <https://orcid.org/0000-0002-8973-542X>)

Maintainer Dmitrii Tereshchenko <dtereshch@gmail.com>

Repository CRAN

Date/Publication 2026-07-21 16:50:02 UTC

Contents

`add_means` *Add Group Means to a Data Frame*

Description

This function adds group means of numeric variables to a data frame, which can be used for Mundlak-style (correlated random effects) modeling or for other purposes where within-group averages are needed.

Usage

```
add_means(data, group = NULL)
```

Arguments

`data`	A data.frame containing the variables.
`group`	A character string or vector of character strings specifying the grouping variable(s). If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes to define grouping variables. Otherwise, overall means are computed and added as constant columns.

Details

For each numeric variable (excluding the grouping variables) and for each grouping variable, the group mean is computed (ignoring NAs) and added as a new column.

For a numeric variable `x` and a grouping variable `g`, the new column is named `x_mean_g`. For example, with grouping variables `"firm"` and `"year"`, variable `"labor"` yields columns `labor_mean_firm` and `labor_mean_year`.

If no grouping variable is supplied and `data` is not a `panel_data` object, the overall mean of each numeric variable is added as a constant column named `x_mean`.

The returned object has class `"panel_data"` if the input was a `panel_data` object; otherwise it is a `data.frame` with additional attributes. In both cases, the object has the following attributes:

`metadata` List containing the function name and the arguments used. If the input was a `panel_data` object, the original metadata elements (entity, time, and delta) are preserved.

`details` A list containing data.frames with the computed group means. If overall means were computed (no grouping), it contains a named vector of the overall means.

Value

The input data frame with additional columns for group means.

See Also

See also `make_demeaned()`, `summarize_numeric()`, `plot_heterogeneity()`.

Examples

```
data(production)

# Basic usage with a single group
prod_means_1 <- add_means(production, group = "firm")

# Using multiple grouping variables
prod_means_2 <- add_means(production, group = c("firm", "year"))

# With panel_data object (automatically uses entity and time)
panel <- make_panel(production, index = c("firm", "year"))
panel_means <- add_means(panel)

# Overall means (no grouping)
prod_overall <- add_means(production)

# Accessing attributes
attr(panel_means, "metadata")
attr(panel_means, "details")
```

---

decompose_factor *Panel Data Factor Variable Decomposition*

Description

This function performs one-way tabulations and decomposes counts into between and within components for categorical (factor) variables in panel data.

Usage

```
decompose_factor(
  data,
  select = NULL,
  index = NULL,
  format = "wide",
  digits = 3
)
```

Arguments

data	A data.frame containing panel data in a long format.
select	A character vector specifying which categorical (factor) variables to analyze. If not specified, all factor variables in the data.frame will be used.
index	A character vector of length 1 or 2 specifying the names of the entity and (optionally) time variables. The first element is the entity variable; if a second element is provided, it is used as the time variable. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.
format	A character string specifying the output format: "wide" or "long". Default = "wide".
digits	An integer indicating the number of decimal places to round shares. Default = 3.

Details

The output format is controlled by the format parameter.

When format = "wide" (default), returns a data.frame with columns:

variable The name of the analyzed variable

category The category level of the variable

count_overall Overall frequency (entity-time observations)

share_overall Overall share (count_overall / total_obs)

count_between Between-entity frequency (number of entities ever having this category)

share_between Between-entity share (count_between / total_entities)

share_within Within-entity share (average share of time entities have this category)

When format = "long", returns a data.frame with columns:

variable The name of the analyzed variable

category The category level of the variable

dimension Type of decomposition: "overall", "between", or "within"

count Frequency count (NA for within dimension)

share Share proportion (0 to 1)

The returned object has class "panel_summary" and two additional attributes:

metadata List containing the function name and the arguments used.

details List containing additional information: count_entities.

Value

A data.frame with categorical panel data decomposition statistics.

References

For Stata users: This corresponds to the xttab command.

See Also

See also decompose_numeric(), summarize_transition().

Examples

```
data(production)

# Basic usage
decompose_factor(production, index = "firm")

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
decompose_factor(panel)

# Selecting specific variables
decompose_factor(production, select = "industry", index = "firm")

# Returning results in a long format
decompose_factor(production, index = "firm", format = "long")

# Custom rounding
decompose_factor(production, index = "firm", digits = 2)

# Accessing attributes
out_dec_fac <- decompose_factor(production, index = "firm")
attr(out_dec_fac, "metadata")
attr(out_dec_fac, "details")
```

---

decompose_numeric *Panel Data Numeric Variable Decomposition*

Description

This function decomposes variance of numeric variables into between and within components in panel data.

Usage

```
decompose_numeric(
  data,
  select = NULL,
  index = NULL,
  detail = TRUE,
  format = "long",
  digits = 3
)
```

Arguments

data	A data.frame containing panel data in a long format.
select	A character vector specifying which numeric variables to analyze. If not specified, all numeric variables in the data.frame will be used.
index	A character vector of length 1 or 2 specifying the names of the entity and (optionally) time variables. The first element is the entity variable; if a second element is provided, it is used as the time variable. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.
detail	A logical flag indicating whether to return detailed Stata-like output. Default = TRUE.
format	A character string specifying the output format: "long" or "wide". Default = "long".
digits	An integer indicating the number of decimal places to round statistics. Default = 3.

Details

The output format is controlled by two parameters: format and detail.

When format = "long" and detail = TRUE (default), returns a data.frame with:

variable The name of the analyzed variable

dimension Type of decomposition: "overall", "between", or "within"

mean Mean value (only for "overall" row)

std Standard deviation

min Minimum value

max Maximum value

count Number of observations or entities

When format = "long" and detail = FALSE, returns a data.frame with:

variable The name of the variable

dimension Type of decomposition: "overall", "between", or "within"

mean Mean value

`std` Standard deviation

When `format = "wide"` and `detail = TRUE`, returns a data.frame with:

`variable` The name of the variable

`mean` Overall mean

`std_overall` Overall standard deviation

`min_overall` Overall minimum

`max_overall` Overall maximum

`count_overall` Number of observations

`std_between` Between-entity standard deviation

`min_between` Minimum of entity means

`max_between` Maximum of entity means

`count_between` Number of entities

`std_within` Within-entity standard deviation

`min_within` Within-entity minimum (transformed)

`max_within` Within-entity maximum (transformed)

`count_within` Average observations per entity

When `format = "wide"` and `detail = FALSE`, returns a data.frame with:

`variable` The name of the variable

`mean` Overall mean

`std_overall` Overall standard deviation

`std_between` Between-entity standard deviation

`std_within` Within-entity standard deviation

The returned object has class `"panel_summary"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List containing additional information: `count_entities`.

Value

A data.frame with panel data decomposition statistics.

References

For Stata users: This corresponds to the `xtsum` command.

See Also

See also `decompose_factor()`, `summarize_numeric()`, `plot_heterogeneity()`.

Examples

```
data(production)

# Basic usage
decompose_numeric(production, index = "firm")

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
decompose_numeric(panel)

# Selecting specific variables
decompose_numeric(production, select = c("sales", "labor"), index = "firm")

# Returning results in a wide format without excessive details
decompose_numeric(production, index = "firm", detail = FALSE, format = "wide")

# Custom rounding
decompose_numeric(production, index = "firm", digits = 2)

# Accessing attributes
out_dec_num <- decompose_numeric(production, index = "firm")
attr(out_dec_num, "metadata")
attr(out_dec_num, "details")
```

---

describe_balance *Panel Data Balance Description*

Description

This function provides summary statistics for panel data structure with focus on balance and data completeness.

Usage

```
describe_balance(data, index = NULL, detail = FALSE, digits = 3)
```

Arguments

data	A data.frame containing panel data in a long format.
index	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.
detail	A logical flag indicating whether to return additional statistics (5th, 25th, 50th, 75th, and 95th percentiles). Default = FALSE.
digits	An integer specifying the number of decimal places for rounding mean values. Default = 3.

Details

The statistics for entities describe the distribution of the number of entities observed per time period (cross-sectional size per period). The statistics for periods describe the distribution of the number of time periods observed per entity (temporal length per entity).

The returned data.frame always contains the following columns:

`dimension` Either "entities" or "periods".

`mean` Mean number of entities per period (or periods per entity).

`std` Standard deviation.

`min` Minimum value.

`max` Maximum value.

When `detail = TRUE`, five additional percentile columns are included:

`p5` 5th percentile.

`p25` 25th percentile (first quartile).

`p50` 50th percentile (median).

`p75` 75th percentile (third quartile).

`p95` 95th percentile.

All statistics are rounded to the number of decimal places specified by `digits`.

The returned object has class `"panel_description"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List containing the full presence matrix.

Value

A data.frame with panel data summary statistics for entities and periods.

Note

An entity-time combination is considered **present** if the corresponding row contains at least one non-NA value in any substantive variable (all columns except the entity and time identifiers).

See Also

See also `describe_dimensions()`, `describe_periods()`, `describe_patterns()`, `plot_periods()`.

Examples

```
data(production)

# Basic usage
describe_balance(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
```

```
describe_balance(panel)

# Returning detailed statistics
describe_balance(production, index = c("firm", "year"), detail = TRUE)

# Custom rounding
describe_balance(production, index = c("firm", "year"), digits = 2)

# Accessing attributes
out_des_bal <- describe_balance(production, index = c("firm", "year"))
attr(out_des_bal, "metadata")
attr(out_des_bal, "details")
```

`describe_dimensions` *Panel Data Dimensions Description*

Description

This function provides basic dimension counts for panel data: number of rows, unique entities, unique time periods, and substantive variables.

Usage

```
describe_dimensions(data, index = NULL)
```

Arguments

`data`	A data.frame containing panel data in a long format.
`index`	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.

Details

The returned data.frame has the following structure:

`rows` Total number of rows in the data frame.

`entities` Number of distinct values in the entity variable.

`periods` Number of distinct values in the time variable.

`variables` Number of substantive variables (all columns except entity and time).

The returned object has class `"panel_description"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List with the actual vectors of entities, periods, and substantive variables.

Value

A data.frame containing panel dimension counts.

See Also

See also `describe_balance()`, `describe_periods()`.

Examples

```
data(production)

# Basic usage
describe_dimensions(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
describe_dimensions(panel)

# Accessing attributes
out_des_dim <- describe_dimensions(production, index = c("firm", "year"))
attr(out_des_dim, "metadata")
attr(out_des_dim, "details")
```

---

describe_incomplete *Incomplete Entities Description*

Description

This function provides a descriptive table of entities with incomplete observations (missing values).

Usage

```
describe_incomplete(data, index = NULL, detail = FALSE)
```

Arguments

`data`	A data.frame containing panel data in a long format.
`index`	A character vector of length 1 or 2 specifying the names of the entity and (optionally) time variables. The first element is the entity variable; if a second element is provided, it is used as the time variable. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.
`detail`	A logical flag indicating whether to include detailed missing counts for each variable. Default = FALSE.

Details

The returned data.frame has the following structure:

`[entity]` The entity identifier (name matches input entity variable)

`na_count` Total number of missing observations for the entity

`variables` Number of variables with at least one missing value for that entity

When `detail = TRUE`, additional columns are included for each substantive variable, showing the number of NAs in that variable for the entity.

The data.frame is sorted by:

1. Number of variables with NAs (descending)
2. Total number of NAs (descending)

The returned object has class `"panel_description"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List containing total entity counts and the IDs of incomplete entities.

Value

A data.frame with incomplete entities description.

Note

The interpretation of incomplete entities may differ depending on whether the panel is balanced or unbalanced. In a balanced panel, each entity has the same number of time periods, so the total possible observations per entity are equal. In an unbalanced panel, entities may have different numbers of time periods, so the number of missing values should be interpreted relative to the entity's total observations. The function does not adjust for the number of time periods per entity; the missing counts reflect absolute counts of NAs in the data. Users should consider the panel structure when interpreting the results.

See Also

See also `summarize_missing()`, `describe_patterns()`, `describe_periods()`.

Examples

```
data(production)

# Basic usage with entity only
describe_incomplete(production, index = "firm")

# With time variable (check duplicates)
describe_incomplete(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
describe_incomplete(panel)
```

```
# Returning detailed results
describe_incomplete(production, index = "firm", detail = TRUE)

# Accessing attributes
out_des_inc <- describe_incomplete(production, index = c("firm", "year"))
attr(out_des_inc, "metadata")
attr(out_des_inc, "details")
```

`describe_patterns` *Entities Presence Patterns Description*

Description

This function describes entities presence patterns in panel data over time.

Usage

```
describe_patterns(
  data,
  index = NULL,
  delta = NULL,
  limits = NULL,
  detail = TRUE,
  format = "wide",
  digits = 3
)
```

Arguments

`data`	A data.frame containing panel data in a long format.
`index`	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.
`delta`	An optional integer giving the expected interval between time periods. If not specified and data is a `panel_data` object with defined `delta`, the value will be extracted from the data.frame attributes.
`limits`	Either a single integer (show that many most frequent patterns) or a vector of two integers (show patterns with ranks between the two values, inclusive). If not specified, all patterns are shown.
`detail`	A logical flag indicating whether to return detailed patterns. Default = TRUE.
`format`	A character string specifying the output format: "wide" or "long". Default = "wide".
`digits`	An integer specifying the number of decimal places for rounding share column. Default = 3.

Details

The output format is controlled by `format` and `detail`.

When `format = "wide"` and `detail = TRUE` (default):

`pattern` Pattern number (ranked by frequency).

`[time1], [time2], ...` Presence (1) / absence (0) for each time period.

`count` Number of entities sharing this pattern.

`share` Proportion of entities with this pattern (rounded to `digits`).

When `format = "wide"` and `detail = FALSE`, only the `pattern` and presence columns are returned.

When `format = "long"` and `detail = TRUE`:

`pattern` Pattern number.

`[time]` Time period identifier (name equals the original time variable).

`presence` Presence (1) / absence (0).

`count` Number of entities with this pattern.

`share` Proportion of entities with this pattern.

When `format = "long"` and `detail = FALSE`, only `pattern`, time, and `presence` columns are returned.

If `delta` is supplied, the function checks that all observed time points are separated by multiples of `delta`. If gaps are detected, a message lists the missing periods (unless the interval was inherited from panel attributes), and columns for those missing periods are added to the presence matrix and to the output data.frame with all zeros. This ensures that the patterns reflect the full regular sequence of time periods.

The returned object has class `"panel_description"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List with the full presence matrix, pattern-entity mapping, and the pattern matrix.

Value

A data.frame with presence patterns.

Note

An entity-time combination is considered **present** if the corresponding row contains at least one non-NA value in any substantive variable (i.e., all columns except the entity and time identifiers).

See Also

See also `plot_patterns()`, `describe_periods()`, `describe_balance()`.

Examples

```
data(production)

# Basic usage
describe_patterns(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
describe_patterns(panel)

# Specifying time interval
describe_patterns(production, index = c("firm", "year"), delta = 1)

# Showing only the top 3 patterns
describe_patterns(production, index = c("firm", "year"), limits = 3)

# Showing patterns ranked 4 to 6
describe_patterns(production, index = c("firm", "year"), limits = c(4, 6))

# Returning results in a long format without excessive details
describe_patterns(production, index = c("firm", "year"), detail = FALSE, format = "long")

# Custom rounding
describe_patterns(production, index = c("firm", "year"), digits = 2)

# Accessing attributes
out_des_pat <- describe_patterns(production, index = c("firm", "year"))
attr(out_des_pat, "metadata")
attr(out_des_pat, "details")
```

---

describe_periods *Time Periods Completeness Description*

Description

This function calculates, for each time period, the number of entities that have at least one non-missing value in any substantive variable, and the corresponding share of all entities.

Usage

```
describe_periods(data, index = NULL, delta = NULL, digits = 3)
```

Arguments

data	A data.frame containing panel data in a long format.
index	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.

delta	An optional integer giving the expected interval between time periods. If not specified and data is a panel_data object with defined delta, the value will be extracted from the data.frame attributes.
digits	An integer specifying the number of decimal places for rounding the share column. Default = 3.

Details

The returned data.frame contains the following columns:

[time] Time period identifier (name matches the input time variable).

count Number of distinct entities observed in that period, i.e., entities with at least one row containing a non-NA value in substantive variables.

share Proportion of entities observed in that period (0 to 1), rounded to digits.

If delta is supplied, the function checks that all observed time points are separated by multiples of delta. If gaps are detected, a message lists the missing periods (unless the interval was inherited from panel attributes). For each missing period, a row is added to the output with count = 0 and share = 0, ensuring that the output covers the full regular time sequence.

The returned object has class "panel_description" and two additional attributes:

metadata List containing the function name and the arguments used.

details List with a named list entities giving, for each period, the vector of entities observed.

Value

A data.frame with entities presence summary by time period.

See Also

See also plot_periods(), describe_balance(), describe_patterns().

Examples

```
data(production)

# Basic usage
describe_periods(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
describe_periods(panel)

# Specifying time interval
describe_periods(production, index = c("firm", "year"), delta = 1)

# Custom rounding
describe_periods(production, index = c("firm", "year"), digits = 2)

# Accessing attributes
```

```
out_des_per <- describe_periods(production, index = c("firm", "year"))
attr(out_des_per, "metadata")
attr(out_des_per, "details")
```

`make_balanced` *Create a Balanced Panel Dataset*

Description

This function creates a balanced panel dataset by either keeping only entities present in all time periods, keeping only periods where all entities are present, or expanding the data to include all entity-time combinations.

Usage

```
make_balanced(data, index = NULL, delta = NULL, balance = "rows")
```

Arguments

`data`	A data.frame containing panel data in a long format.
`index`	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.
`delta`	An optional integer giving the expected interval between time periods. If not specified and data is a `panel_data` object with defined `delta`, the value will be extracted from the data.frame attributes.
`balance`	One of "rows", "entities", or "periods". Specifies the balancing method (see Details). Default = "rows".

Details

This function balances a panel dataset according to the chosen method. The returned object has class `"panel_data"` and includes metadata attributes similar to `make_panel()`.

Depending on the value of `balance`, the balancing is performed using different methods:

`balance = "rows"` Create a row for every entity-time combination. If `delta` is supplied, the full time sequence (including missing periods) is used. Missing combinations get NA in all other columns.

`balance = "entities"` Keep only entities present in all time periods.

`balance = "periods"` Keep only time periods where all entities are present.

The returned object has class `"panel_data"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List containing the unique entities and periods in the balanced data, and if `delta` is supplied, the restored full period sequence and any missing periods.

Value

A balanced panel data.frame with additional attributes.

Note

An entity-time combination is considered **present** if the corresponding row contains at least one non-NA value in any substantive variable (all columns except the entity and time identifiers).

See Also

See also `make_panel()`, `make_wide()`, `make_long()`, `describe_dimensions()`, `describe_balance()`.

Examples

```
data(production)

# Expand to full grid (default method)
balanced_rows <- make_balanced(production, index = c("firm", "year"), balance = "rows")

# Keep only entities present in all periods
balanced_entities <- make_balanced(production, index = c("firm", "year"), balance = "entities")

# Keep only periods where all entities are present
balanced_periods <- make_balanced(production, index = c("firm", "year"), balance = "periods")

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
balanced_panel <- make_balanced(panel)

# Specifying time interval
balanced_rows_delta <- make_balanced(production, index = c("firm", "year"), delta = 1)

# Accessing attributes
attr(balanced_panel, "metadata")
attr(balanced_panel, "details")
```

---

make_demeaned *Within-Group Demeaning (Centering) for Panel Data*

Description

This function performs within-group demeaning (centering) for all numeric variables in a data frame. Non-numeric variables are not demeaned and are returned unchanged.

Usage

```
make_demeaned(data, group = NULL)
```

Arguments

data	A data.frame containing variables to be demeaned.
group	A character string or vector of character strings specifying the grouping variable(s). If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes to define grouping variables. Otherwise, overall demeaning is performed.

Details

Depending on the value of group argument, the function uses different demeaning procedures:

No grouping variable (and data is not a panel_data object) Simple overall demeaning is performed: for each numeric variable, the overall mean (ignoring NAs) is subtracted.

One group variable specified The exact calculation is used: the group mean is subtracted from each observation.

Two or more groups specified Iterative Gauss–Seidel algorithm is used. The algorithm runs up to 2000 iterations with tolerance 1e-6 (matching the defaults of fixest::demean()); a warning is issued if convergence is not reached.

The returned object has class "panel_data" if the input was a panel_data object; otherwise it is a data.frame with additional attributes. In both cases, the object has the following attributes:

metadata List containing the function name and the arguments used. If the input was a panel_data object, the original metadata elements (entity, time, and delta) are preserved.

details A list containing data.frames with the computed group means. If overall demeaning was performed (no grouping), it contains a named vector of the overall means.

Value

The input data frame containing all numeric variables replaced by their demeaned versions and unchanged non-numeric variables.

See Also

See also plot_demeaned(), add_means(), make_panel().

Examples

```
data(production)

# Basic usage (overall demeaning)
prod_demeaned <- make_demeaned(production)

# Demeaning by a single group
prod_demeaned_1 <- make_demeaned(production, group = "firm")

# Demeaning by two groups
prod_demeaned_2 <- make_demeaned(production, group = c("firm", "year"))

# Demeaning by multiple groups
```

```
prod_demeaned_3 <- make_demeaned(production, group = c("industry", "region", "year"))

# With panel_data object (automatically demeans by entity and time)
panel <- make_panel(production, index = c("firm", "year"))
panel_demeaned <- make_demeaned(panel)

# Accessing attributes
attr(panel_demeaned, "metadata")
attr(panel_demeaned, "details")
```

---

make_long *Convert Panel Data from Wide to Long Format*

Description

This function reshapes panel data from wide format to long format.

Usage

```
make_long(
  data,
  select,
  index = NULL,
  static = NULL,
  spacer = "_",
  invert = FALSE
)
```

Arguments

data	A data.frame containing panel data in a wide format.
select	A character vector specifying the stubs of the names of the time-varying variables to reshape.
index	A character vector of length 2 specifying the name of the entity column (first element) and the name to give to the new time column in the long format (second element). If not specified and data is a panel_wide object, the entity and time values are extracted from the metadata.
static	An optional character vector of names of time-invariant variables.
spacer	A character string used to separate variable names and time values in the wide column names. Default = "_".
invert	A logical flag indicating the order of components in column names. If FALSE, column names are variable + spacer + time; if TRUE, they are time + spacer + variable. Default = FALSE.

Details

The function converts data from wide to long format. Below is an illustration of the transformation for two time periods (t = 1, 2) and two time-varying variables (x, y) plus a static variable z.

Wide format (input):

id	x_1	x_2	y_1	y_2	z
1	5	7	8	9	A
2	6	8	7	6	B

Long format (output) with `spacer = "_"` **and** `invert = FALSE`**:**

id	t	x	y	z
1	1	5	8	A
1	2	7	9	A
2	1	6	7	B
2	2	8	6	B

All columns not in `select` (or the entity column) are treated as time-invariant and are replicated for each time period.

The reshaped columns are ordered as follows: the entity column appears first, then the time column, then any static (time-invariant) columns, and finally all time-varying variables in the order they appear in `select`. The time periods are sorted according to their natural order (numerically if all values are numeric, otherwise lexicographically).

After reshaping, if a row has NA for all reshaped variables, the static columns for that row are also set to NA. This prevents a false impression that the entity had a valid observation at that time point (only time-varying variables were missing).

The resulting time column is converted to the most appropriate type: if all time values are integers, the column becomes `integer`; if they are numeric but not integers, it becomes `double`; otherwise, it remains `character`.

The returned object has class `"panel_data"` and two additional attributes:

`metadata` List containing the function name and the arguments used. If the input was a `panel_wide` object, the original metadata elements for the entity and time variables are preserved.

`details` List containing character vectors of reshaped variables names and static variables names.

Value

A data.frame containing panel data in a long format.

Note

The input wide data.frame should have column names that follow a consistent naming convention. **It highly recommended to use a non-empty uniform** `spacer` **and explicit variable names.** Purely numeric column names (e.g., `"2000"`, `"2001"`) are not recognised as time-varying.

If no explicit separator exists and there is no better option, use `spacer = ""` (empty string). In this case, treat the results with caution, as the function may not work perfectly.

In an attempt to avoid cases of ambiguity, when `spacer = ""` the function splits column names using the stubs given in `select` by matching the longest possible stub. The remainder of the column name after removing the matched stub is taken as the time value.

See Also

See also `make_panel()`, `make_wide()`, `make_balanced()`.

Examples

```
data(production)

# Define the time-varying variables to reshape (optional, for reuse)
vars <- c("sales", "capital", "labor", "industry", "ownership")

# First create a wide data frame (using direct specification)
wide <- make_wide(production,
                  select = c("sales", "capital", "labor", "industry", "ownership"),
                  index = c("firm", "year"))

# Direct selection of variables inside the function
long <- make_long(wide,
                  select = c("sales", "capital", "labor", "industry", "ownership"),
                  index = c("firm", "year"))

# Using a pre-defined vector
long2 <- make_long(wide, select = vars, index = c("firm", "year"))

# Custom spacer and inverted order
wide_custom <- make_wide(production,
                         select = vars,
                         index = c("firm", "year"),
                         spacer = ".", invert = TRUE)
long3 <- make_long(wide_custom,
                   select = vars,
                   spacer = ".", invert = TRUE)

# With panel_wide object (uses metadata)
panel <- make_panel(production, index = c("firm", "year"))
wide_panel <- make_wide(panel, select = vars)
long_panel <- make_long(wide_panel, select = vars)

# Using the `static` argument to explicitly mark a time-invariant variable
long_region <- make_long(wide, select = vars, index = c("firm", "year"),
                         static = "region")

# Accessing attributes
attr(long, "metadata")
attr(long, "details")
```

make_panel *Panel Data Structure Setting*

Description

This function adds panel structure attributes to a data.frame, storing entity and time variable names, and optionally checks the expected interval between time periods. The returned data is sorted by the entity and time variables.

Usage

```
make_panel(data, index, delta = NULL, ...)
```

Arguments

data	A data.frame containing panel data in a long format.
index	A character vector of length 2 specifying the names of the entity and time variables.
delta	An optional integer giving the expected interval between time periods.
...	Additional arguments (not used, except to catch deprecated balance).

Details

This function adds attributes to a data.frame to mark it as panel data.

Before returning, the data is sorted by the entity variable (first element of index) and then by the time variable (second element).

If delta is supplied, the function checks that all observed time points are separated by multiples of delta. If gaps are detected, a message lists the missing periods and the full sequence is stored in details$periods_restored.

The returned object has class "panel_data" and two additional attributes:

metadata List containing the function name and the arguments used.

details List containing the unique entities and periods, and if delta is supplied, the restored full period sequence and missing periods.

Value

The input data.frame sorted by the entity and time variables, with additional attributes.

See Also

See also make_balanced(), make_balanced(), make_wide(), make_long(), make_demeaned(), describe_dimensions().

Examples

```
data(production)

# Basic usage
panel <- make_panel(production, index = c("firm", "year"))

# Specifying time interval
panel <- make_panel(production, index = c("firm", "year"), delta = 1)

# Accessing attributes
attr(panel, "metadata")
attr(panel, "details")
```

`make_wide` *Convert Panel Data from Long to Wide Format*

Description

This function reshapes panel data from long format to wide format.

Usage

```
make_wide(
  data,
  select,
  index = NULL,
  static = NULL,
  spacer = "_",
  invert = FALSE
)
```

Arguments

`data`	A data.frame containing panel data in a long format.
`select`	A character vector specifying the names of the time-varying variables to reshape.
`index`	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.
`static`	An optional character vector of names of time-invariant variables.
`spacer`	A character string to insert between variable names and time values in the wide format column names. Default = "_".
`invert`	A logical flag indicating whether to put time values before variable names in column names. If FALSE, column names are `variable` + `spacer` + `time`; if TRUE, they are `time` + `spacer` + `variable`. Default = FALSE.

Details

The function converts data from long to wide format. Below is an illustration of the transformation for two time periods (t = 1, 2) and two time-varying variables (x, y) plus a static variable z.

Long format (input):

id	t	x	y	z
1	1	5	8	A
1	2	7	9	A
2	1	6	7	B
2	2	8	6	B

Wide format (output) with `spacer = "_"` **and** `invert = FALSE`**:**

id	x_1	x_2	y_1	y_2	z
1	5	7	8	9	A
2	6	8	7	6	B

All columns not in `select` or the index are automatically treated as time-invariant. The function verifies that they are indeed constant over time for each entity; if not, it stops with an error listing the offenders.

The reshaped columns are ordered as follows: the entity column appears first, then any time-invariant variables (in the order they appear in the input data), and finally the selected time-varying variables, grouped by variable (i.e., all time periods for the first variable, then all time periods for the second variable, and so on). Time periods are ordered by their natural order. For character or numeric time variables this means increasing order. For factor time variables the order follows the factor levels.

The returned object has class `"panel_wide"` and two additional attributes:

`metadata` List containing the function name and the arguments used. If the input was a `panel_data` object, the original metadata elements (entity, time, and delta) are preserved.

`details` List containing character vectors of reshaped variables names and static variables names.

Value

A data.frame containing panel data in a wide format.

Note

The reshaping preserves standard atomic types and factors. When extracting static variables, the function attempts to preserve the original column class (e.g., `Date`, `POSIXct`). If the class cannot be maintained (for example because the column's assignment method does not handle NA values as expected), a warning is issued and you should consider converting the column to a simpler type before calling `make_wide()`.

Internally, a temporary separator `"._TEMP_."` is used during the reshape step and later replaced by the user-provided `spacer`. The function checks that neither column names nor time values contain this exact string; if they do, an error is raised. Avoid using this substring in your variable names and time values.

See Also

See also make_panel(), make_long(), make_balanced().

Examples

```
data(production)

# Define the time-varying variables to reshape (optional, for reuse)
vars <- c("sales", "capital", "labor", "industry", "ownership")

# Direct selection of variables inside the function
wide <- make_wide(production,
                  select = c("sales", "capital", "labor", "industry", "ownership"),
                  index = c("firm", "year"))

# Using a pre-defined vector
wide2 <- make_wide(production, select = vars, index = c("firm", "year"))

# Custom spacer and inverted order
wide3 <- make_wide(production,
                   select = vars,
                   index = c("firm", "year"),
                   spacer = ".", invert = TRUE)

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
wide4 <- make_wide(panel, select = vars)

# Using `static` to explicitly mark a time-invariant variable
wide_region <- make_wide(production, select = vars,
                         index = c("firm", "year"), static = "region")

# Accessing attributes
attr(wide, "metadata")
attr(wide, "details")
```

---

plot_demeaned *Plot Demeaned Variable(s)*

Description

This function performs within-group demeaning (centering) for selected numeric variable(s) and creates a visualization.

Usage

```
plot_demeaned(data, select, group = NULL, colors = c("darkblue", "white"))
```

Arguments

`data`	A data.frame containing variables for analysis.
`select`	A character vector of length 1 or 2 specifying the name(s) of the numeric variable(s) to demean and plot.
`group`	A character string or vector of character strings specifying the grouping variable(s). If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes to define grouping variables. Otherwise, overall demeaning is performed.
`colors`	A character vector of length 2 specifying the colors for the plot. First color is for fill, second color is for the border line. Default = `c("darkblue", "white")`.

Details

The function creates a plot whose shape depends on the `select` argument:

One variable specified A histogram of the demeaned values is plotted.

Two variables specified A scatterplot of the demeaned values is plotted.

The demeaning is performed by calling `make_demeaned()` internally, so it shares all specifics of that function.

Depending on the value of `group`, the function uses different demeaning procedures:

No grouping variable (and `data` is not a `panel_data` object) Simple overall demeaning is performed: for each numeric variable, the overall mean (ignoring NAs) is subtracted.

One group variable specified The exact calculation is used: the group mean is subtracted from each observation.

Two or more groups specified Iterative Gauss–Seidel algorithm is used. The algorithm runs up to `2000` iterations with tolerance `1e-6` (matching the defaults of `fixest::demean()`); a warning is issued if convergence is not reached.

The returned list contains:

`metadata` List containing the function name and the arguments used.

`details` List containing the demeaned data. For a single variable, this is a numeric vector of demeaned values. For two variables, it is a data frame with columns `x` and `y` containing the demeaned values.

Value

Invisibly returns a list with the demeaned values and metadata.

Note

Only the selected variable(s) and the grouping variables are passed to `make_demeaned()`, so no other numeric variables in the original data are demeaned or cause message output.

See Also

See also `make_demeaned()`, `plot_heterogeneity()`, `decompose_numeric()`

Examples

```
data(production)

# Histogram of a single variable (overall demeaning)
plot_demeaned(production, select = "labor")

# Scatterplot of two variables demeaned by a single group
plot_demeaned(production, select = c("labor", "capital"), group = "firm")

# Demeaning with two grouping variables
plot_demeaned(production, select = c("labor", "capital"), group = c("firm", "year"))

# Demeaning with multiple grouping variables
plot_demeaned(production, select = c("labor", "capital"), group = c("industry", "region", "year"))

# With panel_data object (automatically demeans by entity and time)
panel <- make_panel(production, index = c("firm", "year"))
plot_demeaned(panel, select = c("labor", "capital"))

# Custom colors
plot_demeaned(production, select = c("labor", "capital"), group = c("firm", "year"),
              colors = c("gray", "black"))

# Accessing the returned list components
out_plo_dem <- plot_demeaned(production, select = "labor", group = "firm")
out_plo_dem$metadata
out_plo_dem$details
```

---

plot_heterogeneity *Heterogeneity Visualization*

Description

This function creates visualizations of heterogeneity among groups.

Usage

```
plot_heterogeneity(data, select, group = NULL, colors = c("darkblue", "gray"))
```

Arguments

data	A data.frame containing variables for analysis.
select	A character string specifying the numeric variable of interest.
group	A character string or vector of character strings specifying the grouping variable(s). If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes to define grouping variables.

`colors` A character vector of length 2 specifying the colors for the plot. First color is for mean line and points, second color is for individual points. Default = c("darkblue", "gray").

Details

This function creates one or more plots (depending on the number of grouping variables) showing the heterogeneity among groups. Each plot displays individual observations (points) and group means (connected line).

The returned list contains the following components:

`metadata` List containing the function name and the arguments used.

`details` List containing group-level statistics for each grouping variable, each containing means, standard deviations, and counts per group.

Value

Invisibly returns a list with summary statistics and metadata.

See Also

See also `decompose_numeric()`, `summarize_numeric()`.

Examples

```
data(production)

# Basic usage with regular data.frame
plot_heterogeneity(production, select = "labor", group = "year")

# Using multiple grouping variables
plot_heterogeneity(production, select = "sales", group = c("firm", "industry", "year"))

# With panel_data object (uses both entity and time)
panel <- make_panel(production, index = c("firm", "year"))
plot_heterogeneity(panel, select = "labor")

# Custom colors
plot_heterogeneity(production, select = "sales", group = "year",
                   colors = c("black", "gray"))

# Accessing the returned list components
out_plo_het <- plot_heterogeneity(panel, select = "capital", group = "year")
out_plo_het$metadata
out_plo_het$details
```

plot_missing *Missing Values Heatmap by Period*

Description

This function creates a heatmap showing the number of missing values for each variable across all time periods in panel data.

Usage

```
plot_missing(data, select = NULL, index = NULL, colors = c("darkblue", "gray"))
```

Arguments

data	A data.frame containing panel data in a long format.
select	A character vector specifying which variables to include. If not specified, all substantive variables (except entity and time) are used.
index	A character vector of length 2 giving the names of the entity and time variables. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.
colors	A character vector of length 2 defining the gradient for the heatmap. The first color represents the largest number of missing values, the second color the smallest number. Default = c("darkblue", "gray").

Details

The function creates a heatmap where rows are variables and columns are time periods. Cell color reflects the number of missing values in that variable for that period, using a continuous gradient from colors[1] (most missing) to colors[2] (least missing). Rows are ordered as the variables appear (first at the top). Columns are ordered chronologically.

The returned list contains:

metadata List containing the function name and the arguments used.

details List with the missing count matrix (variables × periods).

Value

Invisibly returns a list with summary statistics and metadata.

Note

The interpretation of missing counts may differ depending on whether the panel is balanced or unbalanced. In a balanced panel, each time period contains the same number of entities, so the raw NA counts per period are directly comparable across periods. In an unbalanced panel, the number of entities varies by period, so the raw NA counts should be interpreted relative to the number of observations available in each period. The function does not standardize the counts by period size; users should account for the panel structure when interpreting the results.

See Also

See also summarize_missing, `plot_patterns()`, `plot_periods()`.

Examples

```
data(production)

# Basic usage
plot_missing(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
plot_missing(panel)

# Selecting specific variables
plot_missing(production, select = c("labor", "capital"), index = c("firm", "year"))

# Custom colors
plot_missing(production, index = c("firm", "year"), colors = c("black", "white"))

# Accessing the returned list components
out_plo_mis <- plot_missing(production, index = c("firm", "year"))
out_plo_mis$metadata
out_plo_mis$details
```

---

plot_patterns *Entities Presence Patterns Visualization*

Description

This function creates a heatmap showing the presence/absence pattern of each entity over time.

Usage

```
plot_patterns(
  data,
  index = NULL,
  delta = NULL,
  limits = NULL,
  colors = c("darkblue", "white")
)
```

Arguments

data	A data.frame containing panel data in a long format.
index	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.

`delta`	An optional integer giving the expected interval between time periods. If not specified and data is a `panel_data` object with defined `delta`, the value will be extracted from the data.frame attributes.
`limits`	Either a single integer (show that many most frequent patterns) or a vector of two integers (show patterns with ranks between the two values, inclusive). If not specified, all patterns are shown.
`colors`	A character vector of length 2 specifying the colors for the plot. First color is for present observations, second color is for missing observations. Default = c("darkblue", "white").

Details

The function creates a heatmap where rows are entities and columns are time periods. Present cells are colored with the first color, missing cells with the second. Rows are ordered by pattern frequency: the most frequent pattern is at the top. Within each pattern block, entities appear in their original order.

If `delta` is supplied, the function checks for regular spacing and adds missing periods (with all zeros) to the plot. A message lists missing periods unless the interval was inherited from panel attributes. The heatmap will therefore show columns for the full regular time sequence, with missing periods appearing entirely white (or the color for missing).

The returned list contains:

`metadata` List containing the function name and the arguments used.

`details` List with the sorted presence matrix, pattern-entity mapping, pattern count, and the pattern matrix (unique patterns as rows).

Value

Invisibly returns a list with summary statistics and metadata.

Note

An entity-time combination is considered **present** if the corresponding row contains at least one non-NA value in any substantive variable (all columns except the entity and time identifiers).

See Also

See also `describe_patterns()`, `plot_periods()`, `plot_missing()`.

Examples

```
data(production)

# Basic usage
plot_patterns(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
plot_patterns(panel)
```

```
# Specifying time interval
plot_patterns(production, index = c("firm", "year"), delta = 1)

# Show only the top 3 patterns
plot_patterns(production, index = c("firm", "year"), limits = 3)

# Show patterns ranked 4 to 6
plot_patterns(production, index = c("firm", "year"), limits = c(4, 6))

# Custom colors
plot_patterns(production, index = c("firm", "year"), colors = c("black", "white"))

# Accessing the returned list components
out_plo_pat <- plot_patterns(production, index = c("firm", "year"))
out_plo_pat$metadata
out_plo_pat$details
```

`plot_periods` *Time Coverage Distribution Visualization*

Description

This function calculates summary statistics and creates a histogram showing the distribution of time periods covered by each entity in panel data.

Usage

```
plot_periods(data, index = NULL, colors = c("darkblue", "white"))
```

Arguments

`data`	A data.frame containing panel data in a long format.
`index`	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a `panel_data` object, the entity and time values will be extracted from the data.frame attributes.
`colors`	A character vector of length 2 specifying the colors for the plot. First color is for fill, second color is for the border line. Default = `c("darkblue", "white")`.

Details

The function creates a histogram of the number of time periods covered by each entity. The x-axis shows coverage (periods per entity), the y-axis shows the count of entities.

The returned list contains:

`metadata` List containing the function name and the arguments used.

`details` List with the coverage vector per entity and the histogram data used for plotting.

Value

Invisibly returns a list with summary statistics and metadata.

Note

An entity-time combination is considered **present** if the corresponding row contains at least one non-NA value in any substantive variable (all columns except the entity and time identifiers).

See Also

See also `describe_periods()`, `plot_patterns()`, `plot_missing()`.

Examples

```
data(production)

# Basic usage
plot_periods(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
plot_periods(panel)

# Custom colors
plot_periods(production, index = c("firm", "year"), colors = c("gray", "black"))

# Accessing the returned list components
out_plo_per <- plot_periods(production, index = c("firm", "year"))
out_plo_per$metadata
out_plo_per$details
```

---

production *Simulated Unbalanced Panel Data for Cobb-Douglas Production Function Analysis*

Description

A simulated dataset containing firm-level panel data with industry affiliation, entry, exit, random missing values, and ownership information. The data follows industry-specific production structures with occasional industry and ownership changes.

Usage

```
production
```

Format

A data frame with 180 rows (30 firms × 6 years) and 8 variables:

firm integer; firm identifier (1 to 30)

year integer; year identifier (1 to 6)

sales numeric; firm sales/output generated from a Cobb-Douglas production function with industry-specific parameters and technology shocks. Contains random missing values (~2%).

capital numeric; capital input, log-normally distributed with firm-specific effects and industry-specific time trends. Contains random missing values (~2%).

labor numeric; labor input, log-normally distributed with firm-specific effects and industry-specific time trends. Contains random missing values (~2%).

industry factor; industry affiliation with three levels: "Industry 1", "Industry 2", "Industry 3". Some firms change industry over time.

ownership factor; ownership type with three levels: "private", "public", "mixed". The variable is stable over time but changes with a probability of 5% per year.

region factor; region affiliation with four levels: "west", "east", "north", "south". Time-invariant for each firm.

Details

The dataset exhibits several realistic features of firm-level panel data:

- 50% of firms (15 firms) have complete data for all 6 years.
- 50% of firms (15 firms) have entry and exit patterns with different start and end years.
- Three industry categories with different production function parameters.
- About 20% of firms change industry affiliation at least once.
- Ownership changes occur with 5% probability per year.
- Industry-specific Cobb-Douglas parameters:
  - Industry 1: $\alpha = 0.25$, $\beta = 0.65$, $A = 2.0$ (labor-intensive)
  - Industry 2: $\alpha = 0.35$, $\beta = 0.55$, $A = 2.2$ (balanced, high productivity)
  - Industry 3: $\alpha = 0.30$, $\beta = 0.60$, $A = 1.8$ (standard)
- Additional random missing values (approx. 2%) in sales, capital, and labor.
- Firm-specific effects and industry-specific time trends in inputs.
- Technology shocks affecting output.

Source

Simulated data for econometric analysis and demonstration purposes.

Examples

```
data(production)
head(production)
table(production$ownership)
```

summarize_missing *Missing Values Summary for Panel Data*

Description

This function calculates summary statistics for missing values (NAs) in panel data, providing both overall and detailed period-specific missing value counts.

Usage

```
summarize_missing(
  data,
  select = NULL,
  index = NULL,
  detail = FALSE,
  digits = 3
)
```

Arguments

data	A data.frame containing panel data in a long format.
select	A character vector specifying which variables to analyze for missing values. If not specified, all variables (except entity and time) will be used.
index	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.
detail	A logical flag indicating whether to return detailed period-specific NA counts. Default = FALSE.
digits	An integer indicating the number of decimal places to round the share column. Default = 3.

Details

When detail = FALSE, returns columns:

variable Variable name.

na_count Total number of missing values in that variable.

na_share Proportion of missing values (rounded to digits).

entities Number of distinct entities that have at least one missing value in that variable.

periods Number of distinct time periods that have at least one missing value in that variable.

When detail = TRUE, additional columns for each time period contain the number of missing values in that variable for that period.

The returned object has class "panel_summary" and two additional attributes:

metadata List containing the function name and the arguments used.

details List with counts of variables with/without NAs, and their names.

Value

A data.frame with missing value summary statistics.

Note

The interpretation of missing counts may differ depending on whether the panel is balanced or unbalanced. In a balanced panel, each time period contains the same number of entities, so the raw NA counts per period are directly comparable across periods. In an unbalanced panel, the number of entities varies by period, so the raw NA counts should be interpreted relative to the number of observations available in each period. The function does not standardize the counts by period size; users should account for the panel structure when interpreting the results.

See Also

See also `plot_missing()`, `describe_incomplete()`, `describe_patterns()`, `describe_periods()`.

Examples

```
data(production)

# Basic usage
summarize_missing(production, index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
summarize_missing(panel)

# Selecting specific variables
summarize_missing(production, select = c("labor", "capital"), index = c("firm", "year"))

# Returning detailed results
summarize_missing(production, index = c("firm", "year"), detail = TRUE)

# Custom rounding
summarize_missing(production, index = c("firm", "year"), digits = 2)

# Accessing attributes
out_sum_mis <- summarize_missing(production, index = c("firm", "year"))
attr(out_sum_mis, "metadata")
attr(out_sum_mis, "details")
```

---

summarize_numeric *Summary Statistics for Numeric Variables*

Description

This function calculates summary statistics for numeric variables, either overall or grouped by a single grouping variable.

Usage

```
summarize_numeric(
  data,
  select = NULL,
  group = NULL,
  detail = FALSE,
  digits = 3
)
```

Arguments

data	A data.frame containing variables for analysis.
select	A character vector specifying which numeric variables to analyze. If not specified, all numeric variables in the data.frame will be used.
group	A character string specifying the grouping variable name. If not specified, overall statistics will be returned.
detail	A logical flag indicating whether to return additional statistics (25th, 50th, and 75th percentiles). Default = FALSE.
digits	An integer specifying the number of decimal places for rounding statistics. Default = 3.

Details

The returned data.frame contains columns depending on the arguments:

When no grouping variable is specified (overall):

`variable` The name of the numeric variable.

`count` Number of non-NA observations.

`mean` Arithmetic mean.

`std` Standard deviation.

`min` Minimum value.

`max` Maximum value.

When `detail = TRUE`, additional columns are included:

`p25` 25th percentile (first quartile).

`p50` 50th percentile (median).

`p75` 75th percentile (third quartile).

When a grouping variable is specified, statistics are calculated for each group, and the data.frame includes a column named after the grouping variable, followed by the same statistics columns as above.

The returned object has class `"panel_summary"` and two additional attributes:

`metadata` List containing the function name and the arguments used.

`details` List with counts of variables, groups, and total observations.

Value

A data.frame with descriptive statistics summary.

Note

Unlike other functions in this package, this function does not access the values of panel attributes from a `panel_data` object.

See Also

See also `decompose_numeric()`, `plot_heterogeneity()`, `add_means()`.

Examples

```
data(production)

# Basic usage
summarize_numeric(production)

# Selecting specific variables
summarize_numeric(production, select = "sales")
summarize_numeric(production, select = c("capital", "labor"))

# Grouped statistics
summarize_numeric(production, group = "year")

# Detailed statistics
summarize_numeric(production, detail = TRUE)

# Custom rounding
summarize_numeric(production, digits = 2)

# Accessing attributes
out_sum_num <- summarize_numeric(production)
attr(out_sum_num, "metadata")
attr(out_sum_num, "details")
```

---

summarize_transition *Transition Summary*

Description

Calculates transition counts and shares between states of a categorical (factor) variable across consecutive time periods within entities for panel data.

Usage

```
summarize_transition(data, select, index = NULL, format = "wide", digits = 3)
```

Arguments

data	A data.frame containing panel data in a long format.
select	A character string specifying the factor variable to analyze transitions for.
index	A character vector of length 2 specifying the names of the entity and time variables. If not specified and data is a panel_data object, the entity and time values will be extracted from the data.frame attributes.
format	A character string specifying the output format: "wide" (default) or "long".
digits	An integer indicating the number of decimal places to round transition shares. Default = 3.

Details

The structure depends on format:

When format = "wide", a transition matrix as a data.frame:

from_to The originating state (row label).

[state1], [state2], ... Columns for each destination state, containing the share of transitions from the row state to the column state (rounded).

When format = "long", a data.frame with columns:

from Originating state.

to Destination state.

count Number of observed transitions.

share Proportion of transitions from from that go to to (rounded).

The returned object has class "panel_summary" and two additional attributes:

metadata List containing the function name and the arguments used.

details List with the vector of all category levels.

Value

A data.frame containing transition summaries.

See Also

See also decompose_factor().

Examples

```
data(production)

# Basic usage
summarize_transition(production, select = "industry", index = c("firm", "year"))

# With panel_data object
panel <- make_panel(production, index = c("firm", "year"))
```

```
summarize_transition(panel, select = "industry")

# Returning results in a long format
summarize_transition(production, select = "industry",
                     index = c("firm", "year"), format = "long")

# Custom rounding
summarize_transition(production, select = "industry", index = c("firm", "year"), digits = 2)

# Accessing attributes
out_sum_tra <- summarize_transition(production, select = "industry", index = c("firm", "year"))
attr(out_sum_tra, "metadata")
attr(out_sum_tra, "details")
```

Index