Package ‘causalverse’

January 10, 2024

Type Package

Date 2023-08-16

Title Causality in Clarity

Version 0.0.0.9000

Maintainer The package maintainer <mikenguyen.contact@gmail.com>

Description CausalVerse: An R toolkit expediting causal research & analysis. Streamlines complex methodologies, empowering users to unveil causal relationships with precision. Your go-to for insightful causality exploration..

License GPL-3 | file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

Config/testthat/edition 3

Imports ggplot2 (>= 3.4.2),
ggthemes (>= 4.2.4),
tidyverse (>= 2.0.0),
lubridate (>= 1.9.2),
rio (>= 0.5.29),
xtable (>= 1.8.4),
dplyr (>= 1.1.1),
tidyr (>= 1.3.0),
scales (>= 1.2.1),
gridExtra (>= 2.3),
systemfit (>= 1.1.30),
Hotelling (>= 1.0.8),
MatchIt (>= 4.5.4),
rlang (>= 1.1.1),
fixest (>= 0.11.1),
stats (>= 4.2.3),
PanelMatch (>= 2.0.1),
doParallel (>= 1.0.17),
fastDummies (>= 1.7.3),
magrittr (>= 2.0.3),

did (>= 2.1.2),
synthdid (>= 0.0.9),
plm (>= 2.6.3),
foreach (>= 1.5.2)

VignetteBuilder knitr

R topics documented:

`ama_export_fig` *Function to export a figure with custom settings*

Description

This function exports a ggplot2 figure to a given path. It exports both an archived version with the current date and a current version without a date. The function supports exporting to PDF and JPG formats.

Usage

```
ama_export_fig(figure, filename, filepath, width = 7, height = 7)
```

Arguments

`figure`	A ggplot2 object.
`filename`	A character string specifying the filename without the extension.
`filepath`	A character string specifying the directory to save the file.
`width`	The width of the image in inches (default is 7 inches).
`height`	The height of the image in inches (default is 7 inches).

Examples

```
## Not run:
test_plot <- ggplot(mpg, aes(x=displ, y=hwy)) + geom_point()  # Create a ggplot2 plot
filename <- "sample_plot"  # Define a filename
filepath <- tempdir()  # Define a path using a temporary directory
ama_export_fig(test_plot, filename, filepath)  # Call the ama_export_fig function

## End(Not run)
```

`ama_export_tab` *Function to export a table with AMA style*

Description

This function exports the provided table in both Excel (`.xlsx`) and LaTeX (`.tex`) formats. The table is archived with the current date in the filename for the Excel version, while the LaTeX version is saved with just the specified filename.

Usage

```
ama_export_tab(table, filename, filepath, caption = NULL)
```

Arguments

`table`	A data frame or matrix.
`filename`	A character string specifying the filename without the extension.
`filepath`	A character string specifying the directory to save the file.
`caption`	A character string specifying the caption for the table.

Examples

```
## Not run:
data(mtcars)  # Load the mtcars dataset
ama_export_tab(mtcars[1:5, 1:5], "sample_table", tempdir(), "Sample Caption for mtcars")

## End(Not run)
```

`ama_labs` *Custom Label Formatting for ggplot2: American Marketing Association Style*

Description

This function provides custom label formatting for ggplot2 based on the guidelines set by the American Marketing Association.

Usage

```
ama_labs(
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  x = NULL,
  y = NULL,
  fill = NULL,
  color = NULL,
  ...
)
```

Arguments

`title`	Plot title.
`subtitle`	Plot subtitle.
`caption`	Plot caption.
`x`	X-axis label.
`y`	Y-axis label.
`fill`	Fill legend title.
`color`	Color legend title.
`...`	Additional arguments to be passed to `ggplot2::labs()`.

Value

Modified labels for a ggplot2 plot.

Examples

```
## Not run:
library(ggplot2)
ggplot(mtcars, aes(mpg, wt)) + geom_point() +
ama_labs(title = "Sample Plot") +
ama_theme()

## End(Not run)
```

ama_scale_color — *Custom Color Scale for ggplot2: American Marketing Association Style*

Description

This function provides a custom color scale for ggplot2 plots based on the guidelines set by the American Marketing Association.

Usage

```
ama_scale_color(
  use_color = FALSE,
  palette_name = "OkabeIto",
  grayscale_limits = c(0.2, 0.8)
)
```

Arguments

`use_color`	Logical. If TRUE, uses color, otherwise uses grayscale.
`palette_name`	Character. Name of the color palette to use.
`grayscale_limits`	Numeric vector. Limits for the grayscale gradient.

Value

A color scale for a ggplot2 plot.

Examples

```
## Not run:
library(ggplot2)
ggplot(mtcars, aes(mpg, wt, color = gear)) + geom_point(size = 4) + ama_scale_color()

## End(Not run)
```

ama_scale_fill — *Custom Fill Scale for ggplot2: American Marketing Association Style*

Description

This function provides a custom fill scale for ggplot2 plots based on the guidelines set by the American Marketing Association.

Usage

```
ama_scale_fill(
  use_color = FALSE,
  palette_name = "OkabeIto",
  grayscale_limits = c(0.2, 0.8)
)
```

Arguments

`use_color`	Logical. If TRUE, uses color, otherwise uses grayscale.
`palette_name`	Character. Name of the color palette to use.
`grayscale_limits`	Numeric vector. Limits for the grayscale gradient.

Value

A fill scale for a ggplot2 plot.

Examples

```
## Not run:
library(ggplot2)
ggplot(mtcars, aes(mpg, wt, fill = gear)) +
geom_point(shape = 21, size = 4) +
ama_scale_fill()

## End(Not run)
```

`ama_theme` *Custom Theme for ggplot2: American Marketing Association Style*

Description

This function provides a custom theme for ggplot2 following the guidelines set by the American Marketing Association.

Usage

```
ama_theme(
  base_size = 16,
  base_family = "sans",
  title_size = ggplot2::rel(1.2),
  axis_title_size = ggplot2::rel(1.2),
  legend_title_size = ggplot2::rel(0.6),
  legend_text_size = ggplot2::rel(0.6),
  axis_text_size = ggplot2::rel(1),
  ...
)
```

Arguments

`base_size`	Base font size.
`base_family`	Font family. Use "sans" for Arial and "serif" for Times New Roman.
`title_size`	Title font size as a relative value.
`axis_title_size`	Axis title font size as a relative value.
`legend_title_size`	Legend title font size as a relative value.

`legend_text_size`	Legend text font size as a relative value.
`axis_text_size`	Axis text font size as a relative value.
`...`	Additional theme elements to be passed to `ggplot2::theme()`.

Value

A ggplot2 theme.

Examples

```
## Not run:
library(ggplot2)
# Using Arial font
ggplot(mtcars, aes(mpg, wt)) + geom_point() + ama_theme()
# Using Times New Roman font
ggplot(mtcars, aes(mpg, wt)) + geom_point() + ama_theme(base_family = "serif")

## End(Not run)
```

`balance_assessment` *Assess balance between treated and control groups*

Description

This function performs a balance assessment between treated and control groups using Seemingly Unrelated Regression (SUR) and Hotelling's T-squared test.

Usage

```
balance_assessment(data, treatment_col, ...)
```

Arguments

`data`	A dataframe containing the data to be assessed.
`treatment_col`	The name of the column that contains the treatment indicator (0 for control, 1 for treated).
`...`	Names of the dependent variables.

Value

A list with two elements: 'SUR' (results of the SUR) and 'Hotelling' (results of the Hotelling's T-squared test).

Examples

```
## Not run:
set.seed(123)
data = mtcars %>%
  dplyr::select(mpg, cyl, disp, hp, wt) %>%
  dplyr::rowwise() %>%
  dplyr::mutate(treatment = sample(c(0,1), 1, replace = TRUE)) %>%
  dplyr::ungroup()

results <- balance_assessment(data, "treatment", "mpg", "cyl")
print(results$SUR)
print(results$Hotelling)

## End(Not run)
```

`balance_scatter_custom` *Custom function to visualize the balance between treatment and control groups*

Description

Custom function to visualize the balance between treatment and control groups

Usage

```
balance_scatter_custom(
  matched_set_list,
  set.names = NULL,
  show.legend = TRUE,
  legend.title = "Type",
  legend.position = "right",
  xlim = c(0, 0.8),
  ylim = c(0, 0.8),
  main = "Standardized Mean Difference of Covariates",
  pchs = NULL,
  dot.size = NULL,
  covariates,
  data,
  x.axis.label = "Before Refinement",
  y.axis.label = "After Refinement",
  theme_use = causalverse::ama_theme(),
  ...
)
```

Arguments

`matched_set_list`	List of matched sets
`set.names`	Vector of names for matched sets. Defaults to NULL.
`show.legend`	Boolean to determine if legend should be shown. Defaults to TRUE.

`legend.title`	Legend title. Defaults to "Type".
`legend.position`	Position of legend. Defaults to "right".
`xlim`	Vector defining x-axis limits. Defaults to c(0, 0.8).
`ylim`	Vector defining y-axis limits. Defaults to c(0, 0.8).
`main`	Main title for the plot. Defaults to "Standardized Mean Difference of Covariates".
`pchs`	Plot characters. Defaults to NULL.
`dot.size`	Size of dots in the scatter plot. Defaults to NULL.
`covariates`	Covariates for calculating balance.
`data`	Dataset for balance calculation.
`x.axis.label`	x-axis label. Defaults to "Before Refinement".
`y.axis.label`	y-axis label. Defaults to "After Refinement".
`theme_use`	Custom theme that follows ggplots2. Defaults to causalverse::ama_theme().
`...`	Additional arguments passed to the labs() function

Value

ggplot object

Examples

```
## Not run:
library(PanelMatch)
# Maha 4-year lag, up to 5 matches
PM.results.maha.4lag.5m <- PanelMatch::PanelMatch(
   lag = 4,
   time.id = "year",
   unit.id = "wbcode2",
   treatment = "dem",
   refinement.method = "mahalanobis",
   data = PanelMatch::dem,
   match.missing = TRUE,
   covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
   size.match = 5,
   qoi = "att",
   outcome.var = "y",
   lead = 0:4,
   forbid.treatment.reversal = FALSE,
   use.diagonal.variance.matrix = TRUE
)

# Maha 4-year lag, up to 10 matches
PM.results.maha.4lag.10m <- PanelMatch::PanelMatch(
   lag = 4,
   time.id = "year",
   unit.id = "wbcode2",
   treatment = "dem",
   refinement.method = "mahalanobis",
   data = PanelMatch::dem,
   match.missing = TRUE,
   covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
   size.match = 10,
   qoi = "att",
   outcome.var = "y",
   lead = 0:4,
   forbid.treatment.reversal = FALSE,
   use.diagonal.variance.matrix = TRUE
)

# Using the function
balance_scatter_custom(
   matched_set_list = list(PM.results.maha.4lag.5m$att, PM.results.maha.4lag.10m$att),
   set.names = c("Maha 4 Lag 5 Matches", "Maha 4 Lag 10 Matches"),
   data = dem,
   covariates = c("y", "tradewb")
)

## End(Not run)
```

`get_balanced_panel` *Extract a Balanced Panel*

Description

This function extracts a balanced panel from the data for a specific adoption cohort. It drops units with missing observations at any point within its entire specified window (leads + adoption time, adoption time + lags). Units treated in the same period as the adoption cohort are marked as "treated," and units with their first treatment after the leads time of the specified adoption cohort are marked as "control."

Usage

```
get_balanced_panel(
  data = data,
  adoption_cohort,
  lags,
  leads,
  time_var,
  unit_id_var,
  treated_period_var,
  filter_units = TRUE
)
```

Arguments

`data`	The dataset to be used.
`adoption_cohort`	Numeric, the specific adoption cohort.
`lags`	Numeric, the number of lags.
`leads`	Numeric, the number of leads.
`time_var`	String, the name of the time variable.
`unit_id_var`	String, the name of the unit ID variable.

`treated_period_var`	String, the name of the treated period variable.
`filter_units`	Logical, whether to filter only units with data on all time periods within the specified time window. Defaults to TRUE.

Value

A data frame with the balanced panel.

Examples

```
## Not run:
get_balanced_panel(data = fixest::base_stagg,
                   adoption_cohort = 5,
                   lags = 2,
                   leads = 3,
                   time_var = "year",
                   unit_id_var = "id",
                   treated_period_var = "year_treated")

## End(Not run)
```

`nice_tab` *Nice Tabulation Function*

Description

Create a custom function that takes a data frame and a number of decimal places as input, rounds all numeric columns in the data frame to the specified number of decimal places, and returns the modified data frame.

Usage

```
nice_tab(data, digit_decimal = 2)
```

Arguments

`data`	A data frame.
`digit_decimal`	A number of decimal places.

Value

A data frame with all numeric columns rounded to the specified number of decimal places.

plot_coef_par_trends *Plot Coefficients of Parallel Trends*

Description

This function generates coefplots or iplots based on fixest outputs, allowing the user to visualize interaction coefficients with ease.

Usage

```
plot_coef_par_trends(
  data,
  dependent_vars,
  time_var,
  unit_treatment_status,
  unit_id_var,
  plot_type = "coefplot",
  combined_plot = TRUE,
  legend_position = "bottomleft",
  legend_title = "Legend Title",
  legend_args = list(),
  plot_args = list()
)
```

Arguments

`data`	Data frame containing the data to be used in the model.
`dependent_vars`	Named list of dependent variables to model and their respective labels.
`time_var`	Name of the time variable in the data.
`unit_treatment_status`	Name of the treatment status variable.
`unit_id_var`	Name of the unit identification variable.
`plot_type`	Type of plot to generate. Either "coefplot" or "iplot".
`combined_plot`	Logical indicating whether to combine plots for all dependent variables.
`legend_position`	Position of the legend on the plot.
`legend_title`	Title for the legend.
`legend_args`	List of additional arguments to customize the legend.
`plot_args`	List of additional arguments to customize the plot.

Value

A plot visualizing interaction coefficients.

Examples

```
## Not run:
library(fixest)
data("base_did")

# Sample call to the function:
plot_coef_par_trends(
  data = base_did,
  dependent_vars = c(y = "Outcome 1", x1 = "Outcome 2"),
  time_var = "period",
  unit_treatment_status = "treat",
  unit_id_var = "id",
  plot_type = "coefplot",
  combined_plot = TRUE,
  plot_args = list(main = "Interaction coefficients Plot"),
  legend_title = "Metrics",
  legend_position = "bottomleft"
)

plot_coef_par_trends(
  data = base_did,
  dependent_vars = c(y = "Outcome 1", x1 = "Outcome 2"),
  time_var = "period",
  unit_treatment_status = "treat",
  unit_id_var = "id",
  plot_type = "coefplot",
  combined_plot = FALSE
)

## End(Not run)
```

`plot_covariate_balance_pretrend` *Plot Covariate Balance Over Pre-Treatment Period*

Description

This function visualizes the covariate balance over the pre-treatment period. It's particularly designed for outputs from methods like PanelMatch.

Usage

```
plot_covariate_balance_pretrend(
  balance_data,
  y_limits = c(-1, 1),
  theme_use = causalverse::ama_theme(),
  xlab = "Time to Treatment",
  ylab = "Balance (in SD unit)",
  main_title = "Covariate Balance Over Pre-Treatment Period",
  legend_title = "Covariate",
  show_legend = TRUE,
  ...
)
```

Arguments

`balance_data`	A matrix containing the covariate balance data over the pre-treatment period.
`y_limits`	A numeric vector of length 2 defining the y-axis limits.
`theme_use`	A ggplot2 theme. By default, it uses `causalverse::ama_theme()`.
`xlab`	A string indicating the label for the x-axis.
`ylab`	A string indicating the label for the y-axis.
`main_title`	A string for the main title of the plot.
`legend_title`	A string for the legend title.
`show_legend`	A logical; if TRUE, the legend is displayed, otherwise, it's hidden.
`...`	Additional arguments passed to the ggplot labs.

Value

A ggplot2 object.

Examples

```
## Not run:
  balance_data_sample <- matrix(rnorm(20), nrow = 5)
  plot_covariate_balance_pretrend(balance_data_sample)

## End(Not run)
```

`plot_density_by_treatment`

Plot Density by Treatment

Description

This function creates a list of ggplot density plots for specified variables by treatment groups.

Usage

```
plot_density_by_treatment(
  data,
  var_map,
  treatment_var,
  show_legend = TRUE,
  theme_use = ggplot2::theme_minimal(),
  ...
)
```

Arguments

`data`	A data frame containing the variables to plot and a treatment variable.
`var_map`	A named list mapping the column names in the data to display names for plotting.
`treatment_var`	A named vector where the name is the treatment column in the data and the value is the legend title.
`show_legend`	A logical value indicating whether to show the legend. Defaults to TRUE.
`theme_use`	ggplot2 theme. Defaults to `ggplot2::theme_minimal()`.
`...`	Additional arguments to be passed to `geom_density`.

Value

A list of ggplot objects for each variable in `var_map`.

Examples

```
## Not run:
data(mtcars)
data <- mtcars %>%
  dplyr::select(mpg, cyl) %>%
  dplyr::rowwise() %>%
  dplyr::mutate(treatment = sample(c(0,1), 1, replace = TRUE)) %>%
  dplyr::ungroup()

plots <- plot_density_by_treatment(
  data = data,
  var_map = list("mpg" = "Var 1",
                 "cyl" = "Var 2"),
  treatment_var = c("treatment" = "Treatment Name\nin Legend")
)

## End(Not run)
```

`plot_PanelEstimate` *Plot Estimated Effects of Treatment Over Time*

Description

This function takes an object (result of PanelEstimate or similar) and plots its estimates over time.

Usage

```
plot_PanelEstimate(
  pe.object,
  ylab = "Estimated Effect of Treatment",
  xlab = "Time Since Treatment",
  main = "Estimated Effects of Treatment Over Time",
  ylim = NULL,
  theme_use = causalverse::ama_theme(),
  ...
)
```

Arguments

`pe.object`	The object with the estimation results.
`ylab`	The y-axis label.
`xlab`	The x-axis label.
`main`	The main title for the plot.
`ylim`	The limits for the y-axis.
`theme_use`	The theme to use for the plot. Defaults to causalverse::ama_theme().
`...`	Additional parameters to pass to labs() function.

Value

A ggplot object with the desired plot.

Examples

```
## Not run:
PM.results.ps.weight <- PanelMatch(lag = 4, time.id = "year", unit.id = "wbcode2",
                                   treatment = "dem", refinement.method = "ps.weight",
                               data = dem, match.missing = FALSE, listwise.delete = TRUE,
                                 covs.formula = ~ I(lag(tradewb, 1:4)) + I(lag(y, 1:4)),
                                   size.match = 5, qoi = "att", outcome.var = "y",
                                   lead = 0:4, forbid.treatment.reversal = FALSE)

PE.results <- PanelEstimate(sets = PM.results.ps.weight,
                            data = dem,
                            se.method = "bootstrap",
                            number.iterations = 1000,
                            confidence.level = .95)

plot_PanelEstimate(PE.results)

## End(Not run)
```

`plot_par_trends` *Plot Parallel Trends*

Description

Plots parallel trends for given metrics.

Usage

```
plot_par_trends(
  data,
  metrics_and_names,
  treatment_status_var,
  time_var,
  conf_level = 0.95,
  non_negative = FALSE,
  display_CI = TRUE,
  output_format = "plot",
  smoothing_method = NULL,
  title_prefix = "Parallel Trends for",
  theme_use = causalverse::ama_theme()
)
```

Arguments

`data`	A data frame containing the data to plot.
`metrics_and_names`	A named list of metrics to plot.
`treatment_status_var`	The variable indicating treatment status.
`time_var`	The variable indicating time.
`conf_level`	Confidence level for confidence intervals (default is 0.95).
`non_negative`	Logical; if TRUE, sets negative lower confidence bounds to 0.
`display_CI`	Logical; if TRUE, displays confidence intervals.
`output_format`	Format of the output; "plot" returns a list of ggplots, "data.frame" returns a data frame.
`smoothing_method`	Method to use for smoothing; NULL means no smoothing.
`title_prefix`	A character string specifying the prefix for the plot title (default is "Parallel Trends for").
`theme_use`	Custom theme that follows ggplots2

Value

A list of ggplot objects or a data frame.

Examples

```
## Not run:
library(tidyverse)
data <- expand.grid(entity = 1:100, time = 1:10) %>%
  dplyr::arrange(entity, time) %>%
  dplyr::mutate(
    treatment = ifelse(entity <= 50, "Treated", "Control"),
    outcome1 = 0.5 * time + rnorm(n(), 0, 2) + ifelse(treatment == "Treated", 0, 0),
    outcome2 = 3 + 0.3 * time + rnorm(n(), 0, 1) + ifelse(treatment == "Treated", 0, 2)
  )
results <- plot_par_trends(
  data = data,
  metrics_and_names = list(outcome1 = "Outcome 1", outcome2 = "Outcome 2"),
  treatment_status_var = "treatment",
  time_var = list(time = "Time"),
  smoothing_method = "loess"
)
library(gridExtra)
gridExtra::grid.arrange(grobs = results, ncol = 1)

## End(Not run)
```

plot_treat_time *Plot number of treated units over time or return a dataframe.*

Description

Plot number of treated units over time or return a dataframe.

Usage

```
plot_treat_time(
  data,
  time_var,
  unit_treat,
  outlier_method = "iqr",
  show_legend = FALSE,
  theme_use = causalverse::ama_theme(),
  legend_title = "Point Type",
  legend_labels = c("Regular", "Outlier"),
  regular_size = 3,
  outlier_size = 5,
  regular_color = "black",
  outlier_color = "red",
  regular_shape = 16,
  outlier_shape = 17,
  title = "Random Time Assignment",
  xlab = "Time",
  ylab = "Number of Treated Units",
  output = "plot",
  ...
)
```

Arguments

`data`	Dataframe containing data.
`time_var`	Time variable for aggregating the number of treated units.
`unit_treat`	Variable indicating if the unit was treated in a specific time period.
`outlier_method`	Method for outlier detection ("iqr" or "z-score").
`show_legend`	Logical indicating whether to show legend.
`theme_use`	ggplot2 theme to use.
`legend_title`	Title for legend.
`legend_labels`	Labels for regular and outlier points.
`regular_size`	Size of regular points.
`outlier_size`	Size of outlier points.
`regular_color`	Color of regular points.
`outlier_color`	Color of outlier points.
`regular_shape`	Shape of regular points.
`outlier_shape`	Shape of outlier points.

`title`	Plot title.
`xlab`	X-axis label.
`ylab`	Y-axis label.
`output`	Type of output ("plot" or "dataframe").
`...`	Additional arguments to pass to ggplot2::labs.

Value

ggplot2 object or dataframe.

Examples

```
# Example usage:
## Not run:
data <- data.frame(time = c(1,1,2,2,3,3), treat = c(0,1,1,1,0,0))
plot_treat_time(data, time_var = time, unit_treat = treat)
plot_treat_time(data, time_var = time, unit_treat = treat, output = "dataframe")

## End(Not run)
```

`plot_trends_across_group`
Custom Faceted Line Plot with Optional Standard Error

Description

This function generates a faceted line plot for a given dataset, allowing the user to specify the x-axis, y-axis, grouping variable, and facet variable. Additionally, users can include standard errors and customize labels.

Usage

```
plot_trends_across_group(
  data,
  x_var,
  y_var,
  grouping_var,
  facet_var,
  se = NULL,
  include_legend = TRUE,
  title = "Dependent Variable across Years by Group and Industry",
  x_label = "Year",
  y_label = "Dependent Variable",
  theme = causalverse::ama_theme(),
  ...
)
```

Arguments

data	A data frame containing the data to be plotted.
x_var	A character string specifying the x-axis variable.
y_var	A character string specifying the y-axis variable.
grouping_var	A character string specifying the grouping variable.
facet_var	A character string specifying the facet variable.
se	A character string specifying the standard error variable, or NULL (default) if not provided.
include_legend	Logical. If TRUE, includes the legend, otherwise it does not.
title	Character string specifying the main plot title.
x_label	Character string specifying the x-axis label.
y_label	Character string specifying the y-axis label.
theme	A ggplot2 theme. Defaults to ama_theme.
...	Additional arguments passed to labs.

Value

A ggplot object.

Examples

```
## Not run:
# Create a small sample dataset
sample_data <- data.frame(
  year = rep(2001:2005, each = 2),
  dependent_variable = rnorm(10, mean = 50, sd = 10),
  group = rep(c("treated", "control"), times = 5),
  industry = rep(c("Tech", "Healthcare"), each = 5)
)

# Use the function
plot_trends_across_group(data = sample_data,
                         x_var = "year",
                         y_var = "dependent_variable",
                         grouping_var = "group",
                         facet_var = "industry",
                         title = "Sample Title")

## End(Not run)
```

---

stack_data *Stacked Data for Staggered DiD Analysis*

Description

stack_data processes datasets used in staggered Difference-in-Differences (DiD) designs. Staggered DiD designs arise when different units (e.g., firms, regions, countries) get treated at different time periods. This function creates cohorts based on the provided treatment period variable and stacks them together to create a comprehensive longitudinal format suitable for staggered DiD analyses.

Usage

```
stack_data(
  treated_period_var,
  time_var,
  pre_window,
  post_window,
  data,
  control_type = c("both", "never-treated", "not-yet-treated")
)
```

Arguments

`treated_period_var`	A character string indicating the column name of the treatment period variable.
`time_var`	A character string indicating the column name for time.
`pre_window`	An integer indicating the number of periods before the treatment to consider (i.e., leads).
`post_window`	An integer indicating the number of periods after the treatment to consider (i.e., lags).
`data`	A data frame containing the dataset to be processed.
`control_type`	A character string indicating which control type to use. One of "both", "never-treated", or "not-yet-treated".

Details

The function emphasizes the importance of having a control group, which should be represented by the value 10000 in the `treated_period_var` column of the provided dataset. The output data will be augmented with relative period dummy variables for ease of subsequent analysis.

Value

A data frame with the stacked data, augmented with relative period dummy variables, suitable for staggered DiD analysis.

Examples

```
## Not run:
  library(did)
  library(tidyverse)
  library(fixest)
  data(base_stagg)
 stacked_data <- stack_data("year_treated", "year", 3, 3, base_stagg, control_type = "both")
  feols_result <- feols(as.formula(paste0(
    "y ~ ",
    paste(paste0("`rel_period_", c(-3:-2, 0:3), "`"), collapse = " + "),
    " | id ^ df + year ^ df"
  )), data = stacked_data)
  print(feols_result)

## End(Not run)
```

synthdid_est *Synthetic DID Estimation Using synthdid Package*

Description

This function estimates synthetic difference-in-differences using the `synthdid` package. It differs from `synthdid::synthdid_estimate` in that it calculates treatment effects (TEs) for each period instead of a single TE for all treated periods.

Usage

```
synthdid_est(
  data,
  adoption_cohort,
  subgroup = NULL,
  lags,
  leads,
  time_var,
  unit_id_var,
  treated_period_var,
  treat_stat_var,
  outcome_var,
  seed = 1
)
```

Arguments

`data`	Data frame to analyze.
`adoption_cohort`	Cohort in data to use as treated.
`subgroup`	(Optional) List of IDs to use as treated subgroup.
`lags`	Number of lags to use pre-treatment.
`leads`	Number of post-treatment periods (0 for only the treatment period).
`time_var`	Name of the calendar time column.
`unit_id_var`	Name of the unit ID column.
`treated_period_var`	Name of the treatment time period column.
`treat_stat_var`	Name of the treatment indicator column.
`outcome_var`	Name of the outcome variable column.
`seed`	A numeric value for setting the random seed (only for placebo SE). Default is 1.

Value

A list containing the estimated treatment effects, standard errors, observed and predicted outcomes, synthetic control lambda weights, and counts of treated and control units.

Examples

```
## Not run:
  library(tidyverse)
  library(causalverse)
  library(synthdid)

  data <- get_balanced_panel(
    data = fixest::base_stagg,
    adoption_cohort = 5,
    lags = 2,
    leads = 3,
    time_var = "year",
    unit_id_var = "id",
    treated_period_var = "year_treated"
  ) |>
    dplyr::mutate(treatvar = if_else(time_to_treatment >= 0, 1, 0)) |>
    dplyr::mutate(treatvar = as.integer(if_else(year_treated > (5 + 2), 0, treatvar)))

  synthdid_est(
    data,
    adoption_cohort = 5,
    lags = 2,
    leads = 3,
    time_var = "year",
    unit_id_var = "id",
    treated_period_var = "year_treated",
    treat_stat_var = "treatvar",
    outcome_var = "y"
  )

## End(Not run)
```

`synthdid_est_ate` *Estimate the SynthDiD ATEs and Standard Errors*

Description

This function uses an adapted SynthDiD method (Arkhangelsky et al., 2021) to estimate the average treatment effect for staggered adoption scenarios. It combines cohort-level ATT estimates, similar to the approach in Ben-Michael et al. (2022), for synthetic controls with staggered adoption. The function is designed to handle various cohorts, lags, leads, placebo tests, and pooled analyses.

Usage

```
synthdid_est_ate(
  data,
  adoption_cohorts,
  lags,
  leads,
  time_var,
  unit_id_var,
  treated_period_var,
  treat_stat_var,
  outcome_var,
  placebo = F,
  pooled = F,
  subgroup = NULL,
  conf_level = 0.95,
  seed = 1
)
```

Arguments

`data`	A data frame in long format to be analyzed.
`adoption_cohorts`	Vector of cohorts to use for adoption times.
`lags`	Integer, number of lags of adoption time to analyze.
`leads`	Integer, number of leads of adoption time to analyze.
`time_var`	String, column name of time variables.
`unit_id_var`	String, ID column of units.
`treated_period_var`	String, column with adoption time of each unit.
`treat_stat_var`	String, column name indicating treatment status.
`outcome_var`	String, column of outcome to analyze.
`placebo`	Logical, whether to run placebo analysis.
`pooled`	Logical, whether to run pooled analysis of all treated units.
`subgroup`	Vector, IDs for subgroup analysis.
`conf_level`	Numeric, confidence level for the interval estimation (Default: 95%).
`seed`	A numeric value for setting the random seed (for placebo SE and placebo analysis). Default is 1.

Value

A list containing the following elements:

- time: Vector of time periods used in estimation from -lags to leads (relative to the adoption period)
- TE_mean: Vector of ATT in each time period
- SE_mean: Vector of Standard error of ATT each time period
- TE_mean_lower: Vector of Lower C.I. for ATT per period
- TE_mean_upper: Vector of Upper C.I. for ATT per period
- TE_mean_w, SE_mean_w, TE_mean_w_lower, TE_mean_w_upper: Weighted versions of the above metrics by the number of treated units in each time period
- Ntr: Number of treated units
- Nco: Number of control units
- TE: Treatment effect for each cohort in each time period
- SE: Standard error of TE of each cohort in each time period
- y_obs: Observed outcomes of treated units
- y_pred: Predicted outcomes of treated units
- col_names: Column names for TE and SE matrices (times and ATTs)

References

Arkhangelsky, D., Athey, S., Hirshberg, D. A., Imbens, G. W., & Wager, S. (2021). Synthetic difference-in-differences. American Economic Review, 111(12), 4088-4118. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203.

Ben-Michael, E., Feller, A., & Rothstein, J. (2022). Synthetic controls with staggered adoption. Journal of the Royal Statistical Society Series B: Statistical Methodology, 84(2), 351-381. Oxford University Press.

Examples

```
## Not run:
  library(tidyverse)
  data <- fixest::base_stagg |>
    mutate(treatvar = if_else(time_to_treatment >= 0, 1, 0)) |>
    mutate(treatvar = as.integer(if_else(year_treated > (5 + 2), 0, treatvar)))

  synthdid_est_ate(
    data = data,
    adoption_cohorts = 5:7,
    lags = 2,
    leads = 2,
    time_var = "year",
    unit_id_var = "id",
    treated_period_var = "year_treated",
    treat_stat_var = "treatvar",
    pooled = F,
    outcome_var = "y"
  )

## End(Not run)
```

`synthdid_est_per` *Estimate Treatment Effects for Each Period*

Description

Given the output from the `synthdid::synthdid_estimate` method, this function computes the treatment effects (TEs) for each post-treatment period, along with the cumulative average treatment effect (ATE). It also provides observed and predicted outcomes for treated units, synthetic control weights, and counts of treated and control units.

Usage

```
synthdid_est_per(Y, N0, T0, weights)
```

Arguments

`Y`	Data matrix with units as rows and time periods as columns.
`N0`	Number of control units.
`T0`	Number of pre-treatment periods.
`weights`	Output from `synthdid`, containing lambda and omega weights.

Value

A list containing:

- est: TEs for each post-treatment period and cumulative ATEs.
- y_obs: Observed outcomes for treated units.
- y_pred: Predicted outcomes for treated units.
- lambda.synth: Synthetic control lambda weights.
- Ntr: Number of treated units.
- Nco: Number of control units.

Examples

```
## Not run:
library(tidyverse)
library(synthdid)
library(fixest)

setup <- base_did |>
  mutate(
    id = as.factor(id),
    period = as.integer(period),
    y = as.double(y),
    post = as.integer(post)
  ) |>
  # Correct treatment
  dplyr::mutate(treatment = as.integer(if_else(treat == 0, 0, post))) |>
 synthdid::panel.matrices(unit = "id", time = "period", outcome = "y", treatment = "treatment")

sdid <- synthdid::synthdid_estimate(setup$Y, setup$N0, setup$T0)
synthdid_est_per(setup$Y, setup$N0, setup$T0, weights = attr(sdid, 'weights'))

## End(Not run)
```

---

synthdid_plot_ate *Create ATE Plot Using ggplot2*

Description

This function creates a ggplot for visualizing Average Treatment Effect (ATE) from a given estimation object.

Usage

```
synthdid_plot_ate(
  est,
  show_CI = TRUE,
  custom_title = "",
  xlab = "Relative Time Period",
  ylab = "ATE",
  y_intercept = 0,
  theme = causalverse::ama_theme(),
  fill_color = "lightgrey"
)
```

Arguments

`est`	Estimation object from `synthdid_est_ate`.
`show_CI`	Logical; if TRUE, shows confidence intervals on the plot.
`custom_title`	String; title of the plot.
`xlab`	String; label for the x-axis.
`ylab`	String; label for the y-axis.
`y_intercept`	Numeric; value at which a horizontal line is drawn.
`theme`	ggplot theme; default is set to causalverse::ama_theme().
`fill_color`	String; color used for the confidence interval shading.

Value

A ggplot object representing the ATE plot.

Examples

```
## Not run:
  # Load required libraries
  library(ggplot2)
  library(tidyverse)
  library(causalverse)

  library(tidyverse)
  data <- fixest::base_stagg |>
    dplyr::mutate(treatvar = if_else(time_to_treatment >= 0, 1, 0)) |>
    dplyr::mutate(treatvar = as.integer(if_else(year_treated > (5 + 2), 0, treatvar)))
  est <-
    synthdid_est_ate(
      data = data,
      adoption_cohorts = 5:7,
      lags = 2,
      leads = 2,
      time_var = "year",
      unit_id_var = "id",
      treated_period_var = "year_treated",
      treat_stat_var = "treatvar",
      pooled = FALSE,
      placebo = FALSE,
      outcome_var = "y"
    )
  # Generate the plot
  synthdid_plot_ate(est, show_CI = TRUE, custom_title = "Sample ATE Plot")

## End(Not run)
```

`synthdid_se_jacknife` *Calculate Jackknife Standard Errors for Synthetic DID*

Description

Computes the standard error of estimates using the jackknife method. It is specifically tailored for use with synthetic difference-in-differences estimates from the `synthdid` package. This function supports both the usual jackknife estimate of variance and the fixed-weights jackknife estimate as described by Arkhangelsky et al.

Usage

```
synthdid_se_jacknife(estimate, weights = attr(estimate, "weights"), seed = 1)
```

Arguments

`estimate`	A synthdid estimate object.
`weights`	Optional; custom weights for the fixed-weights jackknife. If NULL, the usual jackknife estimate is calculated.
`seed`	A numeric value for setting the random seed (only for placebo SE). Default is 1.

Value

Returns the standard error of the provided estimate.

References

Arkhangelsky, D., Athey, S., Hirshberg, D. A., Imbens, G. W., & Wager, S. (2021). Synthetic difference-in-differences. American Economic Review, 111(12), 4088-4118.

Examples

```
## Not run:
setup <- get_balanced_panel(
  data = fixest::base_stagg,
  adoption_cohort = 5,
  lags = 2,
  leads = 3,
  time_var = "year",
  unit_id_var = "id",
  treated_period_var = "year_treated"
) |>
  dplyr::mutate(treatvar = if_else(time_to_treatment >= 0, 1, 0)) |>
  dplyr::mutate(treatvar = as.integer(if_else(year_treated > (5 + 2), 0, treatvar))) |>
  synthdid::panel.matrices(
    unit = "id",
    time = "year",
    outcome = "y",
    treatment = "treatvar"
  )
estimate <- synthdid::synthdid_estimate(setup$Y, setup$N0, setup$T0)
se_results <- synthdid_se_jacknife(estimate, seed = 123)

## End(Not run)
```

`synthdid_se_placebo` *Calculate Placebo Standard Errors for Synthetic DID*

Description

Computes placebo standard errors for synthetic difference-in-differences (DID) estimates. This function is based on the methodology described in Arkhangelsky et al. (2021). It is particularly useful when there is only one treated unit and performs a bootstrap procedure to estimate the standard errors.

Usage

```
synthdid_se_placebo(estimate, replications = 10000, seed = 1)
```

Arguments

`estimate`	An estimate object obtained from synthetic DID estimation.
`replications`	The number of bootstrap replications to perform. Defaults to 500.
`seed`	A numeric value for setting the random seed. Default is 1.

Value

A vector of standard errors corresponding to the input estimates.

References

Arkhangelsky, D., Athey, S., Hirshberg, D. A., Imbens, G. W., & Wager, S. (2021). Synthetic Difference-in-Differences. American Economic Review, 111(12), 4088-4118. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203.

Examples

```
## Not run:
setup <- get_balanced_panel(
  data = fixest::base_stagg,
  adoption_cohort = 5,
  lags = 2,
  leads = 3,
  time_var = "year",
  unit_id_var = "id",
  treated_period_var = "year_treated"
) |>
  # get treatment status
  dplyr::mutate(treatvar = if_else(time_to_treatment >= 0, 1, 0)) |>
  # correct those control units to have treatment status to be 0
  dplyr::mutate(treatvar = as.integer(if_else(year_treated > (5 + 2), 0, treatvar))) |>
  synthdid::panel.matrices(
    unit = "id",
    time = "year",
    outcome = "y",
    treatment = "treatvar"
  )
estimate <- synthdid::synthdid_estimate(setup$Y, setup$N0, setup$T0)
```

```
se_results <- synthdid_se_placebo(estimate, replications = 1000)

## End(Not run)
```

Index