Package: PITS (via r-universe)

October 31, 2026

Title Power of Interrupted Time Series (ITS) Studies

Version 0.1.1

Description Provides tools for estimating the statistical power of Interrupted Time Series (ITS) designs, with a focus on healthcare applications. The package supports prospective power calculations before a study begins, and retrospective assessments of whether a completed study was adequately powered. It includes functions to estimate nuisance parameters (baseline, residual standard deviation, autocorrelation) from data observed before the intervention, and to estimate power via Monte Carlo simulation for single-site and multi-site designs. Utility functions for design optimisation sweeps and publication- ready plots are also provided.

License MIT + file LICENSE

Language en-GB

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.0.0)

Imports nlme, stats, graphics, grDevices, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

URL https://github.com/drdaviddelorenzo/PITS

BugReports https://github.com/drdaviddelorenzo/PITS/issues

Config/roxygen2/version 8.0.0

Repository https://drdaviddelorenzo.r-universe.dev

Date/Publication 2026-07-03 11:39:57 UTC

RemoteUrl https://github.com/drdaviddelorenzo/pits

RemoteRef HEAD

RemoteSha a6169a9f63cc8107f4f0438a31ad8b868568863b

Contents

PITS-package . . . 2
build_param_grid . . . 3
calculate_power . . . 5
calculate_power_multi . . . 6
diagnose_params . . . 8
estimate_and_calculate . . . 9
estimate_baseline . . . 10
estimate_its_params . . . 11
estimate_rho . . . 12
estimate_sigma . . . 14
estimate_trend . . . 15
example_cfr_data . . . 16
export_results . . . 17
fit_its_model . . . 17
interpret_power . . . 18
plot_its_example . . . 19
plot_power_curve . . . 20
plot_power_heatmap . . . 21
power_sweep . . . 22
print.pits_power_result . . . 23
print.pits_sweep_result . . . 24
run_its_power . . . 25
run_power_grid . . . 26
simulate_its_data . . . 27
simulate_predata . . . 29
summary.pits_power_result . . . 30
validate_params . . . 30

Index **32**

`PITS-package` *PITS: Power of Interrupted Time Series studies*

Description

The package provides a complete workflow for ITS power analysis:

Step 1 - Estimate nuisance parameters from pre-intervention data

Use `estimate_its_params()` to estimate `baseline`, `sigma` (residual SD), and `rho` (AR(1) autocorrelation) from a pre-intervention time series. Individual functions `estimate_baseline()`, `estimate_sigma()`, `estimate_rho()`, and `estimate_trend()` are also available.

Step 2 - Calculate power via Monte Carlo simulation

Use `calculate_power()` (single site) or `calculate_power_multi()` (multiple sites) to estimate the probability of detecting a specified intervention effect for a given study design.

Step 3 - Optimise the design

Use `power_sweep()` to evaluate power across a range of post-intervention durations, and `run_power_grid()` for full factorial sensitivity analyses. Visualise results with `plot_power_curve()` and `plot_power_heatmap()`.

Convenience wrappers

`run_its_power()` replicates the interactive experience of the original `its_power_tool.R` script. `estimate_and_calculate()` chains parameter estimation and power calculation in a single call.

Details

Tools for estimating the statistical power of Interrupted Time Series (ITS) designs, with a focus on healthcare applications.

Key references

- Lopez Bernal J, et al. (2017). Interrupted time series regression for the evaluation of public health interventions: a tutorial. *Int J Epidemiol* 46:348-355. doi:10.1093/ije/dyw098
- Zhang F, et al. (2011). Simulation-based power calculation for designing interrupted time series analyses of health policy interventions. *J Clin Epidemiol* 64:1252-1261.
- Wagner AK, et al. (2002). Segmented regression analysis of interrupted time series studies in medication use research. *J Clin Pharm Ther* 27:299-309.

Author(s)

Maintainer: David de Lorenzo <drdaviddelorenzo@gmail.com> (ORCID) (affiliation: UCL Great Ormond Street Institute of Child Health, London, UK; Neotree, London, UK (https://neotree.org/))

Authors:

- David de Lorenzo <drdaviddelorenzo@gmail.com> (ORCID) (affiliation: UCL Great Ormond Street Institute of Child Health, London, UK; Neotree, London, UK (https://neotree.org/))

See Also

Useful links:

- https://github.com/drdaviddelorenzo/PITS
- Report bugs at https://github.com/drdaviddelorenzo/PITS/issues

---

`build_param_grid` *Build a factorial parameter grid for power calculations*

Description

Creates a data frame of all combinations of the supplied parameter vectors, suitable for passing to `run_power_grid()`. Useful for sensitivity analyses and generating figures showing how power varies across parameter space.

Usage

```
build_param_grid(
  n_post,
  level_change,
  sigma,
  rho,
  n_pre = 24L,
  baseline = 15,
  slope_change = 0
)
```

Arguments

n_post	Integer vector. Post-intervention durations to evaluate.
level_change	Numeric vector. Effect sizes to evaluate.
sigma	Numeric vector. Residual SDs to evaluate.
rho	Numeric vector. Autocorrelation values to evaluate.
n_pre	Integer. Pre-intervention duration (fixed). Default 24.
baseline	Numeric. Baseline outcome (fixed). Default 15.
slope_change	Numeric. Slope change (fixed). Default 0.

Value

A data frame with one row per parameter combination.

See Also

run_power_grid()

Examples

```
grid <- build_param_grid(
  n_post       = c(12, 18, 24, 30),
  level_change = c(-1, -2, -3),
  sigma        = c(1.5, 2.5, 3.5),
  rho          = c(0.2, 0.4, 0.6)
)
nrow(grid)   # 4 * 3 * 3 * 3 = 108 combinations
```

`calculate_power` *Estimate statistical power for a single-site ITS design*

Description

Runs a Monte Carlo simulation to estimate the probability of detecting a specified intervention effect in a single-site ITS study.

Usage

```
calculate_power(
  n_pre,
  n_post,
  baseline,
  level_change,
  slope_change = 0,
  sigma,
  rho,
  test = c("level", "slope", "both"),
  alpha = 0.05,
  n_sim = 1000L,
  seed = 123L,
  pre_trend = 0
)
```

Arguments

`n_pre`	Integer. Number of pre-intervention time points.
`n_post`	Integer. Number of post-intervention time points. This is the primary design lever: use `power_sweep()` to find the minimum `n_post` that achieves $\geq 80\%$ power.
`baseline`	Numeric. Mean outcome in the pre-intervention period.
`level_change`	Numeric. Expected immediate step change at the intervention point (your minimum clinically meaningful effect). Set to 0 when `test = "slope"`.
`slope_change`	Numeric. Expected change in trend per time unit after the intervention. Default 0. Set to 0 when `test = "level"`.
`sigma`	Numeric. Residual standard deviation. Estimate from pre-intervention data using `estimate_sigma()` or `estimate_its_params()`.
`rho`	Numeric. AR(1) autocorrelation coefficient in $[0, 1)$. Use 0.4 as a conservative default for monthly data if unknown.
`test`	Character. Effect to test: `"level"` (default), `"slope"`, or `"both"`. See `fit_its_model()` for details.
`alpha`	Numeric. Significance threshold. Default 0.05.
`n_sim`	Integer. Number of Monte Carlo replications. Use 500 for a quick check, 1000 for a reportable estimate, 2000+ for publication.

seed	Integer or NULL. Random seed for reproducibility.
pre_trend	Numeric. Pre-intervention trend per time unit. Default 0.

Value

A named list:

power Numeric. Estimated power (proportion of simulations with $p < \alpha$).

power_pct Numeric. Power as a percentage.

interpretation Character. Qualitative label (see interpret_power()).

p_values Numeric vector. Raw p-values from all n_sim replications (NA = non-convergence).

n_converged Integer. Number of replications that converged.

n_sim Integer. Total replications requested.

params Named list. Input parameters, for traceability.

See Also

power_sweep(), calculate_power_multi(), estimate_its_params()

Examples

```
result <- calculate_power(
  n_pre = 24, n_post = 24,
  baseline = 15, level_change = -3,
  sigma = 2.5, rho = 0.4,
  n_sim = 500, seed = 42
)
print(result$power_pct)
```

---

calculate_power_multi *Estimate statistical power for a multi-site ITS design*

Description

Runs a Monte Carlo simulation to estimate the probability of detecting a common intervention effect across multiple sites (e.g. hospitals), using a mixed-effects segmented regression model.

Usage

```
calculate_power_multi(
  sites,
  test = c("level", "slope", "both"),
  alpha = 0.05,
  n_sim = 1000L,
  seed = 123L
)
```

Arguments

`sites`	A list of named parameter lists, one per site. Each element must contain: `name` (character), `n_pre`, `n_post`, `baseline`, `level_change`, `slope_change`, `sigma`, `rho`. All sites must have the same `n_pre` and `n_post`.
`test`	Character. Effect to test: `"level"` (default), `"slope"`, or `"both"`. See `fit_its_model()` for details.
`alpha`	Numeric. Significance threshold. Default 0.05.
`n_sim`	Integer. Number of Monte Carlo replications. Use 500 for a quick check, 1000 for a reportable estimate, 2000+ for publication.
`seed`	Integer or `NULL`. Random seed for reproducibility.

Details

The model is a linear mixed-effects model with site-specific random intercepts and AR(1) autocorrelation within each site:

$$y_{it} = \beta_0 + \beta_1 t + \delta D_t + \gamma T_t^* + u_i + \varepsilon_{it}$$

where $u_i \sim N(0, \tau^2)$ are site random intercepts and ε_{it} follows AR(1) within each site.

Value

Same structure as `calculate_power()`, plus:

n_sites Integer. Number of sites.

site_names Character vector. Site names.

See Also

`calculate_power()`, `power_sweep()`

Examples

```
sites <- list(
  list(name = "Hospital A", n_pre = 24, n_post = 24,
       baseline = 15, level_change = -3, slope_change = 0,
       sigma = 2.5, rho = 0.4),
  list(name = "Hospital B", n_pre = 24, n_post = 24,
       baseline = 18, level_change = -3, slope_change = 0,
       sigma = 3.0, rho = 0.4)
)
result <- calculate_power_multi(sites, n_sim = 200, seed = 42)
result$power_pct
```

`diagnose_params` *Diagnostic plots for pre-intervention data*

Description

Produces a 2x2 panel of diagnostic plots for pre-intervention data: the observed series with fitted trend, residuals over time, a Q-Q normality plot, and the residual ACF. These help assess whether the ITS model assumptions are plausible.

Usage

```
diagnose_params(
  data,
  outcome_col = "outcome",
  time_col = "time",
  main = "Pre-intervention diagnostics"
)
```

Arguments

`data`	Pre-intervention data: a data frame or numeric vector. See `estimate_its_params()` for format details.
`outcome_col`	Character. Column name for the outcome. Default `"outcome"`.
`time_col`	Character. Column name for time. Default `"time"`.
`main`	Character. Panel title prefix.

Value

Invisibly, a list with elements `params` (estimated parameters) and `residuals` (residual vector).

See Also

`estimate_its_params()`

Examples

```
data("example_cfr_data")
diagnose_params(example_cfr_data)
```

`estimate_and_calculate` *Estimate parameters and calculate power in one step*

Description

Convenience function that chains `estimate_its_params()` and `calculate_power()`. Supply a pre-intervention dataset, a `level_change`, and a target `n_post`, and receive a power estimate.

Usage

```
estimate_and_calculate(
  data,
  level_change,
  n_post,
  outcome_col = "outcome",
  time_col = "time",
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Pre-intervention data: a data frame with columns `time` and `outcome`, or a numeric vector of outcome values.
`level_change`	Numeric. Minimum clinically meaningful effect size (your clinical hypothesis). This is **not** estimated from data.
`n_post`	Integer. Planned post-intervention follow-up duration.
`outcome_col`	Character. Name of the outcome column. Default `"outcome"`.
`time_col`	Character. Name of the time column. Default `"time"`.
`verbose`	Logical. Print parameter and power summaries. Default `TRUE`.
`...`	Additional arguments passed to `calculate_power()`.

Value

Output from `calculate_power()`.

See Also

`estimate_its_params()`, `calculate_power()`, `run_its_power()`

Examples

```
data("example_cfr_data")
# Small n_sim for a fast example; use 1000+ for a reportable estimate.
result <- estimate_and_calculate(
  data         = example_cfr_data,
  level_change = -1.0,
  n_post       = 24,
  n_sim        = 100,
  verbose      = FALSE
)
result$power_pct
```

`estimate_baseline` *Estimate baseline outcome from pre-intervention data*

Description

Fits a linear trend to the pre-intervention series and returns the intercept at $t = 1$, which represents the baseline (mean) outcome at the start of the observation period.

Usage

```
estimate_baseline(outcome, time = seq_along(outcome))
```

Arguments

`outcome`	Numeric vector of outcome values in the pre-intervention period, ordered chronologically. Minimum 12 observations; 24+ recommended.
`time`	Integer vector of time indices. Defaults to `seq_along(outcome)`.

Details

The baseline is the OLS intercept from the linear model $outcome_t = \beta_0 + \beta_1 \cdot t + \varepsilon_t$. When the pre-intervention trend is close to zero (as expected for a stable outcome), this is approximately equal to the mean of `outcome`.

Value

A single numeric value: the estimated baseline outcome.

See Also

`estimate_its_params()` for extracting all parameters at once.

Examples

```
data("example_cfr_data")
estimate_baseline(example_cfr_data$outcome)
```

`estimate_its_params` *Estimate all ITS nuisance parameters from pre-intervention data*

Description

Convenience wrapper that estimates all four nuisance parameters needed for ITS power calculation from a pre-intervention data frame or numeric vector. The output can be passed directly to `calculate_power()` or `run_its_power()`.

Usage

```
estimate_its_params(
  data,
  outcome_col = "outcome",
  time_col = "time",
  verbose = TRUE
)
```

Arguments

`data`	Either: • A data frame with columns named by `time_col` and `outcome_col`; or • A numeric vector of outcome values (in which case `time_col` is ignored and a sequential time index is used).
`outcome_col`	Character. Name of the outcome column when `data` is a data frame. Default `"outcome"`.
`time_col`	Character. Name of the time column when `data` is a data frame. Default `"time"`.
`verbose`	Logical. If `TRUE` (default), prints a formatted parameter summary and guidance to the console.

Details

All four nuisance parameters are estimated jointly by maximum likelihood from a single generalised least squares model with AR(1) errors, fitted with `nlme::gls()` and `nlme::corAR1()`:

$$y_t = \beta_0 + \beta_1 t + \varepsilon_t, \quad \varepsilon_t = \rho\,\varepsilon_{t-1} + \nu_t$$

This is the recommended approach because σ and ρ are mutually dependent: ordinary least squares estimates σ without accounting for autocorrelation (biased upward when $\rho > 0$) and then estimates ρ from serially correlated residuals. Fitting both from the same likelihood avoids this inconsistency and yields parameters that are directly compatible with the simulation engine used by `calculate_power()`. If GLS fails to converge - rare, and usually only for very short series (fewer than about 12 observations) - the function falls back to the OLS two-step estimator and records this in the returned `method` element.

The individual helpers `estimate_baseline()`, `estimate_sigma()`, `estimate_rho()` and `estimate_trend()` use the simpler OLS two-step and are provided for quick, single-parameter checks; for power calculation, prefer the jointly estimated values returned here.

This function does **not** estimate `level_change` or `slope_change`. Those are clinical hypotheses - the minimum effects you would consider meaningful to detect - and must be set based on clinical judgement or published evidence, not derived from your own data.

Value

A named list with elements:

n_pre Integer. Number of pre-intervention observations.

baseline Numeric. Estimated baseline (model intercept).

sigma Numeric. Estimated residual standard deviation.

rho Numeric. Estimated AR(1) autocorrelation.

trend_pre Numeric. Estimated pre-intervention trend (slope).

method Character. Estimation method actually used: `"GLS-ML"` (the default) or `"OLS"` (the fallback, used only when GLS fails to converge).

See Also

`estimate_baseline()`, `estimate_sigma()`, `estimate_rho()`, `estimate_trend()`, `calculate_power()`, `diagnose_params()`

Examples

```
data("example_cfr_data")
params <- estimate_its_params(example_cfr_data, verbose = TRUE)
str(params)

# Use directly in calculate_power():
# calculate_power(
#   n_pre        = params$n_pre,
#   n_post       = 24,
#   baseline     = params$baseline,
#   level_change = -1.0,   # your clinical hypothesis
#   sigma        = params$sigma,
#   rho          = params$rho
# )
```

`estimate_rho` *Estimate AR(1) autocorrelation from pre-intervention data*

Description

Estimates the first-order autocorrelation coefficient (ρ) from the residuals of a linear trend fit to the pre-intervention series.

Usage

```
estimate_rho(outcome, time = seq_along(outcome))
```

Arguments

`outcome`	Numeric vector of outcome values in the pre-intervention period, ordered chronologically. Minimum 12 observations; 24+ recommended.
`time`	Integer vector of time indices. Defaults to `seq_along(outcome)`.

Details

The estimate is the Pearson correlation between consecutive residuals:

$$\hat{\rho} = \text{cor}(\hat{\varepsilon}_t,\, \hat{\varepsilon}_{t+1})$$

Positive autocorrelation is nearly universal in routine health data and reduces effective sample size, lowering power relative to a naive calculation that assumes independence.

Typical ranges by aggregation frequency:

- Daily: 0.7-0.9
- Weekly: 0.5-0.8
- Monthly: 0.3-0.5
- Quarterly: 0.1-0.4

If `outcome` has fewer than 3 observations, `NA` is returned with a warning.

Value

A single numeric value in $(-1, 1)$: the estimated AR(1) autocorrelation coefficient.

See Also

`estimate_its_params()` for extracting all parameters at once.

Examples

```
data("example_cfr_data")
estimate_rho(example_cfr_data$outcome)
```

`estimate_sigma` *Estimate residual standard deviation from pre-intervention data*

Description

Fits a linear trend to the pre-intervention series and returns the standard deviation of the residuals, used as the noise parameter σ in ITS power calculations.

Usage

```
estimate_sigma(outcome, time = seq_along(outcome))
```

Arguments

`outcome`	Numeric vector of outcome values in the pre-intervention period, ordered chronologically. Minimum 12 observations; 24+ recommended.
`time`	Integer vector of time indices. Defaults to `seq_along(outcome)`.

Details

σ is the standard deviation of the detrended pre-intervention series. It captures how much the outcome varies from one time point to the next after accounting for any underlying trend. Larger σ means noisier data and lower power.

As a rough guide, σ is typically 10-20\ monthly hospital rates. If your estimate is much larger, consider whether the outcome series is stable or whether aggregation to a lower frequency would reduce noise.

Value

A single positive numeric value: the estimated residual SD (σ).

See Also

`estimate_its_params()` for extracting all parameters at once.

Examples

```
data("example_cfr_data")
estimate_sigma(example_cfr_data$outcome)
```

`estimate_trend` *Estimate pre-intervention trend from pre-intervention data*

Description

Fits a linear trend to the pre-intervention series and returns the slope (change per time unit). Ideally this should be close to zero, indicating a stable pre-intervention period.

Usage

```
estimate_trend(outcome, time = seq_along(outcome))
```

Arguments

`outcome`	Numeric vector of outcome values in the pre-intervention period, ordered chronologically. Minimum 12 observations; 24+ recommended.
`time`	Integer vector of time indices. Defaults to `seq_along(outcome)`.

Details

A non-trivial pre-intervention trend (e.g. $|trend| > 0.05 \times baseline$) may indicate that the outcome was not stable before the intervention. This can bias ITS estimates and should be discussed in study planning.

Value

A single numeric value: the estimated trend (slope) per time unit.

See Also

`estimate_its_params()` for extracting all parameters at once.

Examples

```
data("example_cfr_data")
estimate_trend(example_cfr_data$outcome)
```

`example_cfr_data` *Example pre-intervention case fatality rate data*

Description

Monthly case fatality rate (CFR) observations from a hypothetical hospital over a 24-month pre-intervention period. This dataset is used in the package vignettes to illustrate the full PITS workflow: parameter estimation followed by power calculation. It represents the motivating example from the accompanying paper, in which a hospital evaluates whether a clinical decision support system (CDSS) reduces case fatality rate and wishes to determine how long post-intervention follow-up is needed to detect a meaningful effect.

Usage

```
example_cfr_data
```

Format

A data frame with 24 rows and 2 variables:

time Integer. Sequential monthly time index, 1 to 24.

outcome Numeric. Case fatality rate, expressed as a percentage (for example, 3.5 represents 3.5 per cent).

Details

The pre-intervention CFR is approximately 3.6 per cent, with low residual variability (sigma approximately 0.2) and moderate positive autocorrelation (rho approximately 0.3 to 0.5), consistent with monthly hospital data.

Source

Simulated dataset generated for illustrative purposes.

Examples

```
data("example_cfr_data")
head(example_cfr_data)
plot(example_cfr_data$time, example_cfr_data$outcome, type = "o",
     xlab = "Month", ylab = "CFR (per cent)",
     main = "Pre-intervention CFR")

# Estimate parameters:
params <- estimate_its_params(example_cfr_data)
```

export_results *Export PITS results to CSV and plain-text summary*

Description

Writes the output of `calculate_power()` or `power_sweep()` to timestamped files in the specified directory.

Usage

```
export_results(result, dir, prefix = "pits")
```

Arguments

`result`	Output from `calculate_power()` or `power_sweep()`.
`dir`	Character. Output directory, supplied by the user; created if it does not exist. There is no default: you must choose where files are written (e.g. a project subdirectory, or `tempdir()` for throwaway output). The function never writes to the working directory or home filespace unless you point `dir` there.
`prefix`	Character. File name prefix. Default `"pits"`.

Value

Invisibly, a named character vector of file paths written.

Examples

```
result <- calculate_power(n_pre = 24, n_post = 24, baseline = 15,
                          level_change = -3, sigma = 2.5, rho = 0.4,
                          n_sim = 100)
# Write to a temporary directory (use your own path in practice):
export_results(result, dir = tempdir())
```

fit_its_model *Fit a segmented regression model to an ITS dataset*

Description

Fits a Gaussian segmented-regression model with AR(1) autocorrelation correction using `nlme::gls()`, and returns the p-value for the coefficient(s) specified by `test`.

Usage

```
fit_its_model(data, test = c("level", "slope", "both"))
```

Arguments

`data`	A data frame as returned by `simulate_its_data()`, containing columns `time`, `D`, `time_after`, and `y`.
`test`	Character. Which effect to test: `"level"` P-value for the immediate step change (δ). Use for acute interventions. `"slope"` P-value for the slope change (γ). Use for gradual interventions. `"both"` Likelihood-ratio test p-value for the joint null $\delta = \gamma = 0$. Uses 2 degrees of freedom.

Details

The model fitted is:

$$y_t = \beta_0 + \beta_1 t + \delta D_t + \gamma T_t^* + \varepsilon_t, \quad \varepsilon_t \sim AR(1)$$

Estimation is by maximum likelihood (`method = "ML"`) to support likelihood-ratio tests. For the `"both"` option, the full model is compared against a model with only a linear trend ($\delta = \gamma = 0$).

Value

A single numeric p-value, or `NA` if the model failed to converge.

See Also

`simulate_its_data()`, `calculate_power()`

Examples

```
df <- simulate_its_data(
  n_pre = 24, n_post = 24,
  baseline = 15, level_change = -3,
  slope_change = 0, sigma = 2.5, rho = 0.4
)
fit_its_model(df, test = "level")
```

`interpret_power` *Interpret a power estimate qualitatively*

Description

Converts a numeric power estimate to a short descriptive label using conventional thresholds.

Usage

```
interpret_power(power)
```

Arguments

power	Numeric. Power estimate in $[0, 1]$.

Value

A character string: "Adequate (>= 80%)", "Borderline (60-79%)", or "Underpowered (< 60%)".

Examples

```
interpret_power(0.85)
interpret_power(0.70)
interpret_power(0.45)
```

---

plot_its_example *Plot a simulated ITS example*

Description

Generates and plots one realisation of an ITS dataset, showing the pre-intervention trend, the post-intervention trajectory, the intervention breakpoint, and the fitted segmented regression lines. Useful for illustrating the ITS model in papers and presentations.

Usage

```
plot_its_example(
  n_pre = 24L,
  n_post = 24L,
  baseline = 15,
  level_change = -3,
  slope_change = 0,
  sigma = 2.5,
  rho = 0.4,
  seed = 42L,
  xlab = "Time",
  ylab = "Outcome",
  main = "Simulated ITS example",
  pre_col = "steelblue",
  post_col = "firebrick"
)
```

Arguments

n_pre	Integer. Pre-intervention time points.
n_post	Integer. Post-intervention time points.
baseline	Numeric. Baseline outcome.

level_change	Numeric. Level change at the intervention.
slope_change	Numeric. Slope change after the intervention. Default 0.
sigma	Numeric. Residual SD.
rho	Numeric. AR(1) autocorrelation.
seed	Integer. Random seed for reproducibility. Default 42.
xlab	Character. x-axis label.
ylab	Character. y-axis label.
main	Character. Plot title.
pre_col	Character. Colour for pre-intervention points.
post_col	Character. Colour for post-intervention points.

Value

Invisibly, the simulated data frame.

Examples

```
plot_its_example(
  n_pre = 24, n_post = 24,
  baseline = 15, level_change = -3,
  sigma = 2.5, rho = 0.4
)
```

---

plot_power_curve *Plot power as a function of post-intervention duration*

Description

Produces a line plot of estimated power against n_post, as returned by power_sweep(). Highlights the 80\ adequate duration.

Usage

```
plot_power_curve(
  sweep_result,
  target = 80,
  xlab = "Post-intervention time points (n_post)",
  ylab = "Estimated power (%)",
  main = "ITS power curve",
  col = "steelblue",
  ...
)
```

Arguments

sweep_result	A data frame as returned by power_sweep(), with columns n_post and power_pct.
target	Numeric. Power target line (0-100). Default 80.
xlab	Character. x-axis label. Default "Post-intervention time points (n_post)".
ylab	Character. y-axis label. Default "Estimated power (%)".
main	Character. Plot title. Default "ITS power curve".
col	Character. Line colour. Default "steelblue".
...	Additional arguments passed to graphics::plot().

Value

Invisibly NULL. Called for its side-effect (plot).

See Also

power_sweep(), plot_power_heatmap()

Examples

```
# Small n_sim for a fast example; use 1000+ for a reportable estimate.
sweep <- power_sweep(
  sweep_post = c(12, 24, 36),
  n_pre = 24, baseline = 15, level_change = -3,
  sigma = 2.5, rho = 0.4, n_sim = 50, seed = 42, verbose = FALSE
)
plot_power_curve(sweep)
```

---

plot_power_heatmap *Plot a power heatmap across two parameters*

Description

Displays estimated power as a colour-coded grid, with one parameter on each axis. Useful for visualising how power responds to combinations of effect size, follow-up duration, noise, or auto-correlation.

Usage

```
plot_power_heatmap(
  grid_result,
  x = "n_post",
  y = "level_change",
  xlab = x,
  ylab = y,
  main = "ITS power heatmap",
  palette = NULL
)
```

Arguments

grid_result	A data frame as returned by run_power_grid(), with a power column and at least the columns specified by x and y.
x	Character. Name of the column to use as the x-axis variable.
y	Character. Name of the column to use as the y-axis variable.
xlab	Character. x-axis label. Defaults to x.
ylab	Character. y-axis label. Defaults to y.
main	Character. Plot title. Default "ITS power heatmap".
palette	Character vector of colours for the gradient from 0 to 100\ power. Defaults to a white-steelblue ramp.

Value

Invisibly NULL. Called for its side-effect (plot).

See Also

run_power_grid(), build_param_grid()

Examples

```
grid <- build_param_grid(
  n_post       = c(12, 24, 36),
  level_change = c(-1, -2, -3),
  sigma        = 2.5,
  rho          = 0.4
)
results <- run_power_grid(grid, n_sim = 100, verbose = FALSE)
plot_power_heatmap(results, x = "n_post", y = "level_change")
```

---

power_sweep *Design optimisation sweep: power across a range of n_post values*

Description

Runs calculate_power() for a vector of post-intervention durations, returning a data frame of power estimates. Use this to identify the minimum n_post needed to achieve $\geq 80\%$ power.

Usage

```
power_sweep(sweep_post = c(6L, 12L, 18L, 24L, 30L, 36L), ..., verbose = TRUE)
```

Arguments

sweep_post	Integer vector. n_post values to evaluate. Default c(6, 12, 18, 24, 30, 36).
...	All other arguments passed to calculate_power(). Note that n_post is overridden by sweep_post internally - do not pass it separately.
verbose	Logical. If TRUE (default), prints a formatted sweep table to the console.

Value

A data frame with columns:

n_post Integer. Follow-up duration evaluated.

power Numeric. Estimated power (0-1).

power_pct Numeric. Power as a percentage.

interpretation Character. Qualitative label.

n_converged Integer. Number of converged replications.

See Also

calculate_power(), plot_power_curve()

Examples

```
# Small n_sim for a fast example; use 1000+ for a reportable estimate.
sweep <- power_sweep(
  sweep_post   = c(12, 24, 36),
  n_pre        = 24,
  baseline     = 15,
  level_change = -3,
  sigma        = 2.5,
  rho          = 0.4,
  n_sim        = 100,
  seed         = 42,
  verbose      = FALSE
)
plot_power_curve(sweep)
```

---

print.pits_power_result

Print method for PITS power results

Description

Displays a formatted summary of the output from calculate_power() or calculate_power_multi().

Usage

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

Arguments

x	A pits_power_result object.
...	Ignored.

Value

Invisibly x.

`print.pits_sweep_result` *Print method for PITS sweep results*

Description

Displays a formatted table of power estimates across post-intervention durations, as returned by power_sweep().

Usage

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

Arguments

x	A pits_sweep_result data frame.
...	Ignored.

Value

Invisibly x.

run_its_power *Full single-site ITS power workflow*

Description

Convenience wrapper that replicates the interactive experience of `its_power_tool.R`. Runs the power simulation and optionally a design sweep, prints formatted output to the console, and optionally saves results.

Usage

```
run_its_power(
  n_pre,
  n_post,
  baseline,
  level_change,
  slope_change = 0,
  sigma,
  rho,
  test = c("level", "slope", "both"),
  alpha = 0.05,
  n_sim = 1000L,
  seed = 123L,
  sweep = FALSE,
  sweep_post = c(6L, 12L, 18L, 24L, 30L, 36L),
  save_output = FALSE,
  output_dir = NULL
)
```

Arguments

`n_pre`	Integer. Pre-intervention time points.
`n_post`	Integer. Post-intervention time points (primary design).
`baseline`	Numeric. Baseline outcome.
`level_change`	Numeric. Expected level change (your clinical hypothesis).
`slope_change`	Numeric. Expected slope change. Default 0.
`sigma`	Numeric. Residual SD.
`rho`	Numeric. AR(1) autocorrelation.
`test`	Character. Effect to test. Default `"level"`.
`alpha`	Numeric. Significance threshold. Default 0.05.
`n_sim`	Integer. Monte Carlo replications. Default 1000.
`seed`	Integer. Random seed. Default 123.
`sweep`	Logical. If `TRUE`, also run `power_sweep()`.

sweep_post	Integer vector. n_post values for the sweep.
save_output	Logical. If TRUE, save results via export_results(). Default FALSE (nothing is written to disk).
output_dir	Character. Directory for saved files, required only when save_output = TRUE. There is no default path: supply your own directory (e.g. a project subdirectory, or tempdir()). Nothing is ever written to the working directory or home filespace by default.

Value

Invisibly, a list with elements result (from calculate_power()) and, if sweep = TRUE, sweep (from power_sweep()).

See Also

calculate_power(), power_sweep(), estimate_its_params()

Examples

```
run_its_power(
  n_pre = 24, n_post = 24,
  baseline = 15, level_change = -3,
  sigma = 2.5, rho = 0.4,
  n_sim = 100, sweep = TRUE
)
```

---

run_power_grid *Run power calculations across a parameter grid*

Description

Applies calculate_power() to each row of a parameter grid as produced by build_param_grid(), returning a data frame with the estimated power for each combination.

Usage

```
run_power_grid(
  grid,
  test = c("level", "slope", "both"),
  alpha = 0.05,
  n_sim = 500L,
  seed = 123L,
  verbose = TRUE
)
```

Arguments

grid	A data frame as returned by build_param_grid(), or any data frame with columns n_pre, n_post, baseline, level_change, slope_change, sigma, rho.
test	Character. Effect to test: "level", "slope", or "both".
alpha	Numeric. Significance threshold. Default 0.05.
n_sim	Integer. Monte Carlo replications per cell. Default 500.
seed	Integer. Base random seed; each row adds its row index.
verbose	Logical. If TRUE, prints a progress counter.

Value

The input grid data frame with columns appended: power, power_pct, interpretation, n_converged.

See Also

build_param_grid(), plot_power_heatmap()

Examples

```
grid <- build_param_grid(
  n_post       = c(12, 24, 36),
  level_change = c(-2, -3),
  sigma        = 2.5,
  rho          = 0.4
)
results <- run_power_grid(grid, n_sim = 100, verbose = FALSE)
plot_power_heatmap(results, x = "n_post", y = "level_change")
```

---

simulate_its_data *Simulate a single ITS dataset*

Description

Generates one realisation of an ITS time series under the alternative hypothesis (i.e. with a true intervention effect), with AR(1) autocorrelated errors. Used internally by calculate_power().

Usage

```
simulate_its_data(
  n_pre,
  n_post,
  baseline,
  level_change,
  slope_change,
```

```
  sigma,
  rho,
  pre_trend = 0
)
```

Arguments

n_pre	Integer. Number of pre-intervention time points.
n_post	Integer. Number of post-intervention time points.
baseline	Numeric. Mean outcome at $t = 1$ (before any trend).
level_change	Numeric. Immediate step change in outcome at the intervention point. Set to 0 if testing slope only.
slope_change	Numeric. Change in trend per time unit after the intervention. Set to 0 if testing level only.
sigma	Numeric. Residual standard deviation (noise).
rho	Numeric. AR(1) autocorrelation coefficient. Must be in $[0, 1)$.
pre_trend	Numeric. Pre-intervention trend (slope) per time unit. Default 0 (stable pre-period).

Details

The data-generating process is the standard segmented-regression ITS model:

$$y_t = \beta_0 + \beta_1 t + \delta D_t + \gamma T_t^* + \varepsilon_t$$

where $D_t = \mathbf{1}(t > n_{pre})$, $T_t^* = (t - n_{pre}) \cdot D_t$, δ = level_change, γ = slope_change, and ε_t follows an AR(1) process with innovation SD $\sigma\sqrt{1-\rho^2}$.

Value

A data frame with columns:

time Integer time index, 1 to $n_{pre} + n_{post}$.

D Binary intervention indicator (0 = pre, 1 = post).

time_after Time elapsed since intervention (0 in pre-period).

y Simulated outcome values.

See Also

calculate_power(), fit_its_model()

Examples

```
df <- simulate_its_data(
  n_pre = 24, n_post = 24,
  baseline = 15, level_change = -3,
  slope_change = 0, sigma = 2.5, rho = 0.4
)
```

```
plot(df$time, df$y, type = "o", pch = 16,
     xlab = "Time", ylab = "Outcome")
abline(v = 24.5, lty = 2, col = "red")
```

`simulate_predata` *Generate synthetic pre-intervention data*

Description

Simulates a pre-intervention time series with known parameters. Useful for testing and for vignette examples when real data are unavailable.

Usage

```
simulate_predata(
  n = 24L,
  baseline = 15,
  sigma = 2.5,
  rho = 0.4,
  trend = 0,
  seed = 42L
)
```

Arguments

`n`	Integer. Number of time points to generate. Default 24.
`baseline`	Numeric. Mean outcome at $t = 1$. Default 15.
`sigma`	Numeric. Residual standard deviation. Default 2.5.
`rho`	Numeric. AR(1) autocorrelation. Default 0.4.
`trend`	Numeric. Linear trend per time unit. Default 0.
`seed`	Integer or `NULL`. Random seed. Default 42.

Value

A data frame with columns `time` and `outcome`.

Examples

```
pre <- simulate_predata(n = 24, baseline = 15, sigma = 2.5, rho = 0.4)
plot(pre$time, pre$outcome, type = "o")
```

summary.pits_power_result *Summary method for PITS power results*

Description

Returns an extended summary including the p-value distribution across Monte Carlo replications.

Usage

```
## S3 method for class 'pits_power_result'
summary(object, ...)
```

Arguments

object	A pits_power_result object.
...	Ignored.

Value

Invisibly, a list with power, params, and pvalue_quantiles.

validate_params *Validate ITS parameter values before simulation*

Description

Checks that a set of ITS parameters is internally consistent and within plausible ranges. Issues warnings for unusual but permitted values.

Usage

```
validate_params(
  n_pre,
  n_post,
  baseline,
  level_change,
  sigma,
  rho,
  alpha = 0.05,
  n_sim = 1000L
)
```

Arguments

n_pre	Integer. Pre-intervention time points.
n_post	Integer. Post-intervention time points.
baseline	Numeric. Baseline outcome.
level_change	Numeric. Expected level change.
sigma	Numeric. Residual SD.
rho	Numeric. AR(1) autocorrelation.
alpha	Numeric. Significance threshold.
n_sim	Integer. Number of simulations.

Value

Invisibly TRUE if all checks pass; stops with an error on critical failures.

Examples

```
validate_params(n_pre = 24, n_post = 24, baseline = 15,
                level_change = -3, sigma = 2.5, rho = 0.4)
```

Index

∗ **datasets**
example_cfr_data, 16

build_param_grid, 3
build_param_grid(), *22*, *26, 27*

calculate_power, 5
calculate_power(), *2*, *7*, *9*, *11, 12*, *17, 18*, *22, 23*, *26–28*
calculate_power_multi, 6
calculate_power_multi(), *2*, *6*, *23*

diagnose_params, 8
diagnose_params(), *12*

estimate_and_calculate, 9
estimate_and_calculate(), *3*
estimate_baseline, 10
estimate_baseline(), *2*, *11, 12*
estimate_its_params, 11
estimate_its_params(), *2*, *5, 6*, *8–10*, *13–15*, *26*
estimate_rho, 12
estimate_rho(), *2*, *11, 12*
estimate_sigma, 14
estimate_sigma(), *2*, *5*, *11, 12*
estimate_trend, 15
estimate_trend(), *2*, *11, 12*
example_cfr_data, 16
export_results, 17
export_results(), *26*

fit_its_model, 17
fit_its_model(), *5*, *7*, *28*

graphics::plot(), *21*

interpret_power, 18
interpret_power(), *6*

nlme::corAR1(), *11*

nlme::gls(), *11*, *17*

PITS *(*PITS-package*)*, 2
PITS-package, 2
plot_its_example, 19
plot_power_curve, 20
plot_power_curve(), *3*, *23*
plot_power_heatmap, 21
plot_power_heatmap(), *3*, *21*, *27*
power_sweep, 22
power_sweep(), *3*, *5–7*, *17*, *20, 21*, *24–26*
print.pits_power_result, 23
print.pits_sweep_result, 24

run_its_power, 25
run_its_power(), *3*, *9*, *11*
run_power_grid, 26
run_power_grid(), *3, 4*, *22*

simulate_its_data, 27
simulate_its_data(), *18*
simulate_predata, 29
summary.pits_power_result, 30

validate_params, 30