Package 'valytics'

January 22, 2026

Title Statistical Methods for Analytical Method Comparison and Validation

Version 0.3.0

Description Provides statistical methods for analytical method comparison and validation studies. Implements Bland-Altman analysis for assessing agreement between measurement methods (Bland & Altman (1986) <doi:10.1016/S0140-6736(86)90837-8>),
Passing-Bablok regression for non-parametric method comparison (Passing & Bablok (1983) <doi:10.1515/cclm.1983.21.11.709>), and Deming regression accounting for measurement error in both variables (Linnet (1993) <doi:10.1093/clinchem/39.3.424>). Also includes tools for setting quality goals based on biological variation (Fraser & Petersen (1993) <doi:10.1093/clinchem/39.7.1447>) and calculating Six Sigma metrics. Commonly used in clinical laboratory method validation. Provides publication-ready plots and comprehensive statistical summaries.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.1

LazyData true

Depends R (>= 3.5.0)

Imports ggplot2, robslopes

Suggests testthat (>= 3.0.0), knitr (>= 1.43), rmarkdown (>= 2.22)

Config/testthat/edition 3

URL `https://github.com/marcellogr/valytics`

BugReports `https://github.com/marcellogr/valytics/issues`

VignetteBuilder knitr

NeedsCompilation no

Author Marcello Grassi [aut, cre] (ORCID: <`https://orcid.org/0009-0008-8904-4988`>)

Maintainer Marcello Grassi <`marcello.grassi@tum.de`>

Repository CRAN

Date/Publication 2026-01-22 21:00:02 UTC

Contents

ate_assessment 2
ate_from_bv 5
ba_analysis 7
creatinine_serum 10
deming_regression 11
glucose_methods 15
pb_regression 16
plot.ba_analysis 19
plot.deming_regression 21
plot.pb_regression 24
print.ate_assessment 26
print.ate_specs 27
print.ba_analysis 27
print.deming_regression 28
print.pb_regression 29
print.sigma_metric 30
print.summary.ba_analysis 30
sigma_metric 31
summary.ate_assessment 33
summary.ate_specs 34
summary.ba_analysis 34
summary.deming_regression 35
summary.pb_regression 36
summary.sigma_metric 37
troponin_cardiac 38

Index **40**

`ate_assessment` *Assess Analytical Performance Against Allowable Total Error*

Description

Evaluates observed analytical performance (bias and imprecision) against allowable total error specifications. Provides pass/fail assessment for individual components and overall method acceptability, along with the sigma metric.

Usage

```
ate_assessment(
  bias,
  cv,
  tea,
  allowable_bias = NULL,
  allowable_cv = NULL,
  k = 1.65
)
```

Arguments

`bias`	Numeric. Observed bias (systematic error), expressed as a percentage.
`cv`	Numeric. Observed coefficient of variation (imprecision), expressed as a percentage.
`tea`	Numeric. Total allowable error specification. Can be provided directly or will be calculated if `allowable_bias` and `allowable_cv` are provided with `k`.
`allowable_bias`	Numeric. Allowable bias specification (optional). If provided, enables individual bias assessment.
`allowable_cv`	Numeric. Allowable imprecision specification (optional). If provided, enables individual CV assessment.
`k`	Numeric. Coverage factor for TEa calculation when using component specifications (default: 1.65).

Details

The assessment evaluates method performance at multiple levels:

Component Assessment (if specifications provided):

- Bias: Pass if |observed bias| <= allowable bias
- CV: Pass if observed CV <= allowable CV

Total Error Assessment:

- Observed TE = k * CV + |Bias| (linear model)
- Pass if observed TE <= TEa

Sigma Metric:

- Sigma = (TEa - |Bias|) / CV
- Provides quality rating from "World Class" to "Unacceptable"

Value

An object of class `c("ate_assessment", "valytics_ate", "valytics_result")`, which is a list containing:

assessment List with pass/fail results:

- `bias_acceptable`: Logical; TRUE if |bias| <= allowable_bias
- `cv_acceptable`: Logical; TRUE if cv <= allowable_cv
- `tea_acceptable`: Logical; TRUE if observed TE <= TEa
- `overall`: Logical; TRUE if method meets specifications

observed List with observed performance:

- `bias`: Observed bias
- `cv`: Observed CV
- `te`: Observed total error (k * CV + |Bias|)

specifications List with allowable specifications:

- `allowable_bias`: Allowable bias (or NULL)
- `allowable_cv`: Allowable CV (or NULL)
- `tea`: Total allowable error

sigma List with sigma metric results:

- `value`: Sigma metric value
- `category`: Performance category

settings List with settings:

- `k`: Coverage factor used

Overall Assessment

The overall assessment is determined as follows:

- If only TEa is provided: based on total error assessment
- If component specs provided: all components must pass
- Sigma >= 3 is generally considered minimum acceptable

References

Westgard JO (2008). *Basic Method Validation* (3rd ed.). Westgard QC, Inc.

Fraser CG (2001). *Biological Variation: From Principles to Practice*. AACC Press.

See Also

`ate_from_bv()` for calculating specifications from biological variation, `sigma_metric()` for sigma calculation details

Examples

```
# Basic assessment with TEa only
assess <- ate_assessment(bias = 1.5, cv = 2.5, tea = 10)
assess

# Assessment with all component specifications
assess_full <- ate_assessment(
  bias = 1.5,
  cv = 2.5,
  tea = 10,
  allowable_bias = 3.0,
  allowable_cv = 4.0
)
assess_full

# Using specifications from ate_from_bv()
specs <- ate_from_bv(cvi = 5.6, cvg = 7.5)
assess <- ate_assessment(
  bias = 1.5,
  cv = 2.5,
  tea = specs$specifications$tea,
  allowable_bias = specs$specifications$allowable_bias,
  allowable_cv = specs$specifications$allowable_cv
)
summary(assess)

# Check if method passes
assess$assessment$overall
```

---

ate_from_bv *Calculate Allowable Total Error from Biological Variation*

Description

Calculates analytical performance specifications (allowable imprecision, allowable bias, and total allowable error) based on biological variation data using the hierarchical model from Fraser & Petersen (1993).

Usage

```
ate_from_bv(
  cvi,
  cvg = NULL,
  level = c("desirable", "optimal", "minimum"),
  k = 1.65
)
```

Arguments

`cvi`	Numeric. Within-subject (intra-individual) biological variation coefficient of variation, expressed as a percentage.
`cvg`	Numeric. Between-subject (inter-individual) biological variation coefficient of variation, expressed as a percentage. If `NULL` (default), only imprecision specifications are calculated.
`level`	Character. Performance level: `"desirable"` (default), `"optimal"`, or `"minimum"`. See Details.
`k`	Numeric. Coverage factor for total allowable error calculation (default: 1.65 for ~95% coverage assuming normal distribution).

Details

The biological variation model for analytical performance specifications was developed by Fraser, Petersen, and colleagues. The model derives allowable analytical error from the inherent biological variability of the measurand.

Formulas (Desirable level):

$$CV_A \leq 0.50 \times CV_I$$

$$Bias \leq 0.25 \times \sqrt{CV_I^2 + CV_G^2}$$

$$TEa \leq k \times CV_A + Bias$$

Where:

- CV_I = within-subject biological variation
- CV_G = between-subject biological variation
- CV_A = allowable analytical imprecision
- k = coverage factor (default 1.65)

Performance Levels:

Three hierarchical performance levels are defined:

- **Optimal**: Most stringent; multipliers are 0.25x desirable (i.e., 0.125 for CV, 0.0625 for bias)
- **Desirable**: Standard target; multipliers are 0.50 for CV, 0.25 for bias
- **Minimum**: Least stringent; multipliers are 1.5x desirable (i.e., 0.75 for CV, 0.375 for bias)

Value

An object of class `c("ate_specs", "valytics_ate", "valytics_result")`, which is a list containing:

specifications List with calculated specifications:

- `allowable_cv`: Allowable analytical imprecision (CV_A)
- `allowable_bias`: Allowable analytical bias (NULL if cvg not provided)
- `tea`: Total allowable error (NULL if cvg not provided)

input List with input parameters:

- `cvi`: Within-subject CV
- `cvg`: Between-subject CV (or NULL)
- `level`: Performance level used
- `k`: Coverage factor used

multipliers List with level-specific multipliers used:

- `imprecision`: Multiplier for CV_I
- `bias`: Multiplier for sqrt(CV_I^2 + CV_G^2)

Data Sources

Biological variation data (CV_I and CV_G) should be obtained from authoritative sources. The recommended current source is the **EFLM Biological Variation Database**: `https://biologicalvariation.eu/`

This database provides rigorously reviewed BV estimates derived from published studies meeting defined quality specifications.

References

Fraser CG, Petersen PH (1993). Desirable standards for laboratory tests if they are to fulfill medical needs. *Clinical Chemistry*, 39(7):1447-1453.

Ricos C, Alvarez V, Cava F, et al. (1999). Current databases on biological variation: pros, cons and progress. *Scandinavian Journal of Clinical and Laboratory Investigation*, 59(7):491-500.

Aarsand AK, Fernandez-Calle P, Webster C, et al. (2020). The EFLM Biological Variation Database. `https://biologicalvariation.eu/`

Westgard JO (2008). *Basic Method Validation* (3rd ed.). Westgard QC, Inc.

See Also

`sigma_metric()` for calculating Six Sigma metrics, `ate_assessment()` for comparing observed performance to specifications

Examples

```
# Glucose: CV_I = 5.6%, CV_G = 7.5% (example values)
ate <- ate_from_bv(cvi = 5.6, cvg = 7.5)
ate

# Optimal performance level (more stringent)
ate_optimal <- ate_from_bv(cvi = 5.6, cvg = 7.5, level = "optimal")
ate_optimal

# Minimum acceptable performance
ate_min <- ate_from_bv(cvi = 5.6, cvg = 7.5, level = "minimum")
ate_min

# When only within-subject CV is known (bias goal not calculable)
ate_cv_only <- ate_from_bv(cvi = 5.6)
ate_cv_only

# Custom coverage factor (e.g., 2.0 for ~97.5% coverage)
ate_custom <- ate_from_bv(cvi = 5.6, cvg = 7.5, k = 2.0)

# Access individual specifications
ate$specifications$allowable_cv
ate$specifications$allowable_bias
ate$specifications$tea
```

`ba_analysis` *Bland-Altman Analysis for Method Comparison*

Description

Performs Bland-Altman analysis to assess agreement between two measurement methods. Calculates bias (mean difference), limits of agreement, and confidence intervals following the approach of Bland & Altman (1986, 1999).

Usage

```
ba_analysis(
  x,
  y = NULL,
  data = NULL,
  conf_level = 0.95,
  type = c("absolute", "percent"),
  na_action = c("omit", "fail")
)
```

Arguments

`x`	Numeric vector of measurements from method 1 (reference method), or a formula of the form `method1 ~ method2`.
`y`	Numeric vector of measurements from method 2 (test method). Ignored if `x` is a formula.
`data`	Optional data frame containing the variables specified in `x` and `y` (or in the formula).
`conf_level`	Confidence level for intervals (default: 0.95).
`type`	Type of difference calculation: `"absolute"` (default) for `y - x`, or `"percent"` for `100 * (y - x) / mean`.
`na_action`	How to handle missing values: `"omit"` (default) removes pairs with any NA, `"fail"` stops with an error.

Details

The Bland-Altman method assesses agreement between two quantitative measurements by analyzing the differences against the averages. The key outputs are:

- **Bias**: The mean difference between methods, indicating systematic difference. A bias significantly different from zero suggests one method consistently measures higher or lower than the other.
- **Limits of Agreement (LoA)**: The interval within which 95\ differences are expected to lie (bias +/- 1.96 x SD). These define the range of disagreement between methods.
- **Confidence Intervals**: CIs for bias and LoA quantify the uncertainty in these estimates due to sampling variability.

The confidence intervals for limits of agreement are calculated using the exact method from Bland & Altman (1999), which accounts for the uncertainty in both the mean and standard deviation.

Value

An object of class `c("ba_analysis", "valytics_comparison", "valytics_result")`, which is a list containing:

input List with original data and metadata:

- `x`, `y`: Numeric vectors (after NA handling)

- `n`: Number of paired observations
- `n_excluded`: Number of pairs excluded due to NAs
- `var_names`: Named character vector with variable names

results List with statistical results:

- `differences`: Numeric vector of differences (y - x or percent)
- `averages`: Numeric vector of means ((x + y) / 2)
- `bias`: Mean difference (point estimate)
- `bias_se`: Standard error of the bias
- `bias_ci`: Named numeric vector with lower and upper CI for bias
- `sd_diff`: Standard deviation of differences
- `loa_lower`: Lower limit of agreement (bias - 1.96 * SD)
- `loa_upper`: Upper limit of agreement (bias + 1.96 * SD)
- `loa_lower_ci`: Named numeric vector with CI for lower LoA
- `loa_upper_ci`: Named numeric vector with CI for upper LoA

settings List with analysis settings:

- `conf_level`: Confidence level used
- `type`: Type of difference calculation
- `multiplier`: Multiplier for LoA (default: 1.96 for 95\

call The matched function call.

Assumptions

The standard Bland-Altman analysis assumes:

- Differences are approximately normally distributed
- No proportional bias (constant bias across the measurement range)
- No repeated measurements per subject

References

Bland JM, Altman DG (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, 1(8476):307-310. doi:10.1016/S01406736(86)908378

Bland JM, Altman DG (1999). Measuring agreement in method comparison studies. *Statistical Methods in Medical Research*, 8(2):135-160. doi:10.1177/096228029900800204

See Also

`plot.ba_analysis()` for visualization, `summary.ba_analysis()` for detailed summary

Examples

```
# Simulated method comparison data
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- method_a + rnorm(50, mean = 2, sd = 5)  # Method B has +2 bias
```

```
# Basic analysis
ba <- ba_analysis(method_a, method_b)
ba

# Using formula interface with data frame
df <- data.frame(reference = method_a, test = method_b)
ba <- ba_analysis(reference ~ test, data = df)

# Percentage differences
ba_pct <- ba_analysis(method_a, method_b, type = "percent")
```

`creatinine_serum` *Serum Creatinine Method Comparison Dataset*

Description

Synthetic dataset comparing serum creatinine measurements from two analytical methods: an enzymatic method (reference) and the Jaffe colorimetric method. This is a classic method comparison scenario where the Jaffe method is known to have positive interference from proteins and other chromogens.

Usage

```
creatinine_serum
```

Format

A data frame with 80 observations and 3 variables:

sample_id Character. Unique sample identifier.

enzymatic Numeric. Creatinine concentration (mg/dL) measured by the enzymatic method.

jaffe Numeric. Creatinine concentration (mg/dL) measured by the Jaffe method.

Details

This synthetic dataset was designed to illustrate well-known patterns in creatinine method comparisons:

- **Concentration range**: 0.4-8.0 mg/dL, from normal kidney function to severe chronic kidney disease (CKD)
- **Bias pattern**: Jaffe method shows positive bias, more pronounced at lower concentrations due to pseudocreatinine interference
- **Outliers**: A few samples show larger positive bias, simulating interference from bilirubin, hemolysis, or ketones

The enzymatic method is more specific and has largely replaced the Jaffe method in modern clinical laboratories, though the Jaffe method remains in use in some settings due to lower reagent costs.

Source

Synthetic data generated to mimic realistic clinical patterns. See `data-raw/make_datasets.R` for the generation script.

References

Peake M, Whiting M. Measurement of serum creatinine–current status and future goals. Clin Biochem Rev. 2006;27(4):173-184.

See Also

`ba_analysis()`, glucose_methods, troponin_cardiac

Examples

```
# Bland-Altman analysis
ba <- ba_analysis(enzymatic ~ jaffe, data = creatinine_serum)
summary(ba)
plot(ba)

# Note: Jaffe typically shows positive bias vs enzymatic
```

`deming_regression` *Deming Regression for Method Comparison*

Description

Performs Deming regression to assess agreement between two measurement methods. Unlike ordinary least squares, Deming regression accounts for measurement error in both variables, making it appropriate for method comparison studies where neither method is a perfect reference.

Usage

```
deming_regression(
  x,
  y = NULL,
  data = NULL,
  error_ratio = 1,
  conf_level = 0.95,
  ci_method = c("jackknife", "bootstrap"),
  boot_n = 1999,
  weighted = FALSE,
  na_action = c("omit", "fail")
)
```

Arguments

`x`	Numeric vector of measurements from method 1 (reference method), or a formula of the form `method1 ~ method2`.
`y`	Numeric vector of measurements from method 2 (test method). Ignored if `x` is a formula.
`data`	Optional data frame containing the variables specified in `x` and `y` (or in the formula).
`error_ratio`	Ratio of error variances (Var(error_y) / Var(error_x)). Default is 1 (orthogonal regression, assuming equal error variances). Can be estimated from replicate measurements or set based on prior knowledge of method precision.
`conf_level`	Confidence level for intervals (default: 0.95).
`ci_method`	Method for calculating confidence intervals: `"jackknife"` (default) uses delete-one jackknife resampling, `"bootstrap"` uses BCa bootstrap resampling.
`boot_n`	Number of bootstrap resamples when `ci_method = "bootstrap"` (default: 1999).
`weighted`	Logical; if `TRUE`, performs weighted Deming regression where weights are inversely proportional to the variance at each point. Requires replicate measurements to estimate weights. Default is `FALSE`.
`na_action`	How to handle missing values: `"omit"` (default) removes pairs with any NA, `"fail"` stops with an error.

Details

Deming regression (also known as errors-in-variables regression or Model II regression) is designed for situations where both X and Y are measured with error. This is the typical case in method comparison studies where both the reference and test methods have measurement uncertainty.

The error ratio (lambda, λ) represents the ratio of error variances:

$$\lambda = \frac{Var(\epsilon_y)}{Var(\epsilon_x)}$$

When λ = 1 (default), this is equivalent to orthogonal regression, which minimizes perpendicular distances to the regression line. When λ != 1, the regression minimizes a weighted combination of horizontal and vertical distances.

Choosing the error ratio:

- If both methods have similar precision: use λ = 1
- If precision differs: estimate from replicate measurements as λ = CV_y^2 / CV_x^2 (squared coefficient of variation ratio)
- If one method is much more precise: consider ordinary least squares

Value

An object of class `c("deming_regression", "valytics_comparison", "valytics_result")`, which is a list containing:

input List with original data and metadata:

- `x`, `y`: Numeric vectors (after NA handling)
- `n`: Number of paired observations
- `n_excluded`: Number of pairs excluded due to NAs
- `var_names`: Named character vector with variable names

results List with statistical results:

- `intercept`: Intercept point estimate
- `slope`: Slope point estimate
- `intercept_ci`: Named numeric vector with lower and upper CI
- `slope_ci`: Named numeric vector with lower and upper CI
- `intercept_se`: Standard error of intercept
- `slope_se`: Standard error of slope
- `residuals`: Perpendicular residuals
- `fitted_x`: Fitted x values
- `fitted_y`: Fitted y values

settings List with analysis settings:

- `error_ratio`: Error variance ratio used
- `conf_level`: Confidence level used
- `ci_method`: CI method used
- `boot_n`: Number of bootstrap samples (if applicable)
- `weighted`: Whether weighted regression was used

call The matched function call.

Interpretation

- **Slope = 1**: No proportional difference between methods
- **Slope != 1**: Proportional (multiplicative) difference exists
- **Intercept = 0**: No constant difference between methods
- **Intercept != 0**: Constant (additive) difference exists

Use the confidence intervals to test these hypotheses: if 1 is within the slope CI and 0 is within the intercept CI, the methods are considered equivalent.

Comparison with Other Methods

- **Ordinary Least Squares (OLS)**: Assumes X is measured without error. Biases slope toward zero when both variables have error.
- **Passing-Bablok**: Non-parametric, robust to outliers, but assumes linear relationship and no ties.
- **Deming**: Parametric, accounts for error in both variables, allows specification of error ratio.

Assumptions

- Linear relationship between X and Y
- Measurement errors are normally distributed
- Error variances are constant (homoscedastic) or known ratio
- Errors in X and Y are independent

References

Deming WE (1943). Statistical Adjustment of Data. Wiley.

Linnet K (1990). Estimation of the linear relationship between the measurements of two methods with proportional errors. *Statistics in Medicine*, 9(12):1463-1473. doi:10.1002/sim.4780091210

Linnet K (1993). Evaluation of regression procedures for methods comparison studies. *Clinical Chemistry*, 39(3):424-432. doi:10.1093/clinchem/39.3.424

Cornbleet PJ, Gochman N (1979). Incorrect least-squares regression coefficients in method-comparison analysis. *Clinical Chemistry*, 25(3):432-438.

See Also

`plot.deming_regression()` for visualization, `summary.deming_regression()` for detailed summary, `pb_regression()` for non-parametric alternative, `ba_analysis()` for Bland-Altman analysis

Examples

```
# Simulated method comparison data
set.seed(42)
true_values <- rnorm(50, mean = 100, sd = 20)
method_a <- true_values + rnorm(50, sd = 5)
method_b <- 1.05 * true_values + 3 + rnorm(50, sd = 5)

# Basic analysis (orthogonal regression, lambda = 1)
dm <- deming_regression(method_a, method_b)
dm

# Using formula interface with data frame
df <- data.frame(reference = method_a, test = method_b)
dm <- deming_regression(reference ~ test, data = df)

# With known error ratio (e.g., test method has 2x variance)
dm <- deming_regression(method_a, method_b, error_ratio = 2)

# With bootstrap confidence intervals
dm_boot <- deming_regression(method_a, method_b, ci_method = "bootstrap")

# Using package example data
data(glucose_methods)
dm <- deming_regression(reference ~ poc_meter, data = glucose_methods)
summary(dm)
plot(dm)
```

`glucose_methods` *Glucose Method Comparison Dataset*

Description

Synthetic dataset comparing glucose measurements from two analytical methods: a reference hexokinase-based laboratory analyzer and a point-of-care (POC) glucose meter. The data mimics realistic patterns observed in clinical laboratory method validation studies.

Usage

```
glucose_methods
```

Format

A data frame with 60 observations and 3 variables:

sample_id Character. Unique sample identifier.

reference Numeric. Glucose concentration (mg/dL) measured by the reference hexokinase method.

poc_meter Numeric. Glucose concentration (mg/dL) measured by the point-of-care glucose meter.

Details

This synthetic dataset was designed to illustrate common patterns in glucose method comparisons:

- **Concentration range**: 50-350 mg/dL, covering hypoglycemia through severe hyperglycemia
- **Bias pattern**: The POC meter shows a small positive bias (~3-5 mg/dL) with slight proportional error at higher concentrations
- **Precision**: Reference method CV ~2.5%, POC meter CV ~4.5%

The data is suitable for demonstrating Bland-Altman analysis, Passing-Bablok regression, and other method comparison techniques.

Source

Synthetic data generated to mimic realistic clinical patterns. See `data-raw/make_datasets.R` for the generation script.

See Also

`ba_analysis()`, creatinine_serum, troponin_cardiac

Examples

```
# Bland-Altman analysis
ba <- ba_analysis(reference ~ poc_meter, data = glucose_methods)
summary(ba)
plot(ba)

# Check for proportional bias
plot(ba, title = "POC Glucose Meter vs Reference")
```

---

pb_regression *Passing-Bablok Regression for Method Comparison*

Description

Performs Passing-Bablok regression to assess agreement between two measurement methods. This non-parametric regression method is robust to outliers and does not assume normally distributed errors. The implementation uses a fast O(n log n) algorithm from the `robslopes` package for point estimation.

Usage

```
pb_regression(
  x,
  y = NULL,
  data = NULL,
  conf_level = 0.95,
  ci_method = c("analytical", "bootstrap"),
  boot_n = 1999,
  na_action = c("omit", "fail")
)
```

Arguments

`x`	Numeric vector of measurements from method 1 (reference method), or a formula of the form `method1 ~ method2`.
`y`	Numeric vector of measurements from method 2 (test method). Ignored if `x` is a formula.
`data`	Optional data frame containing the variables specified in `x` and `y` (or in the formula).
`conf_level`	Confidence level for intervals (default: 0.95).
`ci_method`	Method for calculating confidence intervals: `"analytical"` (default) uses the original Passing-Bablok (1983) method, `"bootstrap"` uses BCa bootstrap resampling.
`boot_n`	Number of bootstrap resamples when `ci_method = "bootstrap"` (default: 1999).
`na_action`	How to handle missing values: `"omit"` (default) removes pairs with any NA, `"fail"` stops with an error.

Details

Passing-Bablok regression is a non-parametric method for fitting a linear relationship between two measurement methods. Unlike ordinary least squares, it:

- Makes no assumptions about error distribution
- Accounts for measurement error in both variables
- Is robust to outliers
- Produces results independent of which variable is assigned to X or Y (when using the equivariant form)

The slope is estimated as the median of all pairwise slopes (in absolute value for the equivariant version), and the intercept is the median of `y - slope * x`.

Value

An object of class `c("pb_regression", "valytics_comparison", "valytics_result")`, which is a list containing:

input List with original data and metadata:

- `x`, `y`: Numeric vectors (after NA handling)
- `n`: Number of paired observations
- `n_excluded`: Number of pairs excluded due to NAs
- `var_names`: Named character vector with variable names

results List with statistical results:

- `intercept`: Intercept point estimate
- `slope`: Slope point estimate
- `intercept_ci`: Named numeric vector with lower and upper CI
- `slope_ci`: Named numeric vector with lower and upper CI
- `residuals`: Perpendicular residuals
- `fitted_x`: Fitted x values
- `fitted_y`: Fitted y values

cusum List with CUSUM linearity test results (if calculable):

- `statistic`: CUSUM test statistic
- `critical_value`: Critical value at alpha = 0.05
- `p_value`: Approximate p-value
- `linear`: Logical; TRUE if linearity assumption holds

settings List with analysis settings:

- `conf_level`: Confidence level used
- `ci_method`: CI method used
- `boot_n`: Number of bootstrap samples (if applicable

call The matched function call.

Interpretation

- **Slope = 1**: No proportional difference between methods
- **Slope != 1**: Proportional (multiplicative) difference exists
- **Intercept = 0**: No constant difference between methods
- **Intercept != 0**: Constant (additive) difference exists

Use the confidence intervals to test these hypotheses: if 1 is within the slope CI and 0 is within the intercept CI, the methods are considered equivalent.

Assumptions

- Linear relationship between X and Y (test with CUSUM)
- Measurement range covers the intended clinical range
- Data are continuously distributed

CUSUM Test for Linearity

The CUSUM (cumulative sum) test checks the linearity assumption. A significant result ($p < 0.05$) suggests non-linearity, and Passing-Bablok regression may not be appropriate.

References

Passing H, Bablok W (1983). A new biometrical procedure for testing the equality of measurements from two different analytical methods. Application of linear regression procedures for method comparison studies in clinical chemistry, Part I. *Journal of Clinical Chemistry and Clinical Biochemistry*, 21(11):709-720. doi:10.1515/cclm.1983.21.11.709

Passing H, Bablok W (1984). Comparison of several regression procedures for method comparison studies and determination of sample sizes. Application of linear regression procedures for method comparison studies in Clinical Chemistry, Part II. *Journal of Clinical Chemistry and Clinical Biochemistry*, 22(6):431-445. doi:10.1515/cclm.1984.22.6.431

Bablok W, Passing H, Bender R, Schneider B (1988). A general regression procedure for method transformation. Application of linear regression procedures for method comparison studies in clinical chemistry, Part III. *Journal of Clinical Chemistry and Clinical Biochemistry*, 26(11):783-790. doi:10.1515/cclm.1988.26.11.783

Raymaekers J, Dufey F (2022). Equivariant Passing-Bablok regression in quasilinear time. *arXiv preprint*. doi:10.48550/arXiv.2202.08060

See Also

`plot.pb_regression()` for visualization, `summary.pb_regression()` for detailed summary, `ba_analysis()` for Bland-Altman analysis

Examples

```
# Simulated method comparison data
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
```

```
method_b <- 1.05 * method_a + 3 + rnorm(50, sd = 5)  # slope=1.05, intercept=3

# Basic analysis
pb <- pb_regression(method_a, method_b)
pb

# Using formula interface with data frame
df <- data.frame(reference = method_a, test = method_b)
pb <- pb_regression(reference ~ test, data = df)

# With bootstrap confidence intervals
pb_boot <- pb_regression(method_a, method_b, ci_method = "bootstrap")

# Using package example data
data(glucose_methods)
pb <- pb_regression(reference ~ poc_meter, data = glucose_methods)
summary(pb)
plot(pb)
```

`plot.ba_analysis` *Plot method for ba_analysis objects*

Description

Creates a Bland-Altman plot (difference vs. average) for visualizing agreement between two measurement methods. The plot displays the bias (mean difference) and limits of agreement with optional confidence intervals.

Usage

```
## S3 method for class 'ba_analysis'
plot(
  x,
  show_ci = TRUE,
  show_points = TRUE,
  point_alpha = 0.6,
  point_size = 2,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)

## S3 method for class 'ba_analysis'
autoplot(
  object,
  show_ci = TRUE,
  show_points = TRUE,
  point_alpha = 0.6,
  point_size = 2,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

`x`	An object of class `ba_analysis`.
`show_ci`	Logical; if `TRUE` (default), displays confidence interval bands for bias and limits of agreement.
`show_points`	Logical; if `TRUE` (default), displays individual data points.
`point_alpha`	Numeric; transparency of points (0-1, default: 0.6).
`point_size`	Numeric; size of points (default: 2).
`line_colors`	Named character vector with colors for `"bias"`, `"loa"`, and `"ci"`. Defaults to a clean color scheme.
`title`	Character; plot title. If `NULL` (default), generates an automatic title.
`xlab`	Character; x-axis label. If `NULL`, uses "Mean of methods".
`ylab`	Character; y-axis label. If `NULL`, auto-generates based on difference type.
`...`	Additional arguments (currently ignored).
`object`	An object of class `ba_analysis`.

Details

The Bland-Altman plot displays:

- **Points**: Each point represents a paired observation, plotted as the difference (y - x) against the average ((x + y) / 2).
- **Bias line**: Solid horizontal line at the mean difference.
- **Limits of agreement**: Dashed horizontal lines at bias +/- 1.96 x SD.
- **Confidence intervals**: Shaded bands showing the uncertainty in the bias and LoA estimates.

Patterns to look for:

- **Funnel shape**: Suggests proportional bias (variance increases with magnitude).
- **Trend**: Suggests systematic relationship between difference and magnitude.
- **Outliers**: Points outside the LoA may warrant investigation.

Value

A `ggplot` object that can be further customized.

See Also

`ba_analysis()` for performing the analysis, `summary.ba_analysis()` for detailed results

Examples

```
# Basic Bland-Altman plot
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- method_a + rnorm(50, mean = 2, sd = 5)

ba <- ba_analysis(method_a, method_b)
plot(ba)

# Without confidence intervals
plot(ba, show_ci = FALSE)

# Customized appearance
plot(ba,
     point_alpha = 0.8,
     point_size = 3,
     title = "Method Comparison: A vs B")

# Further customization with ggplot2
library(ggplot2)
plot(ba) +
  theme_minimal() +
  scale_color_brewer(palette = "Set1")

# Using autoplot (ggplot2-style)
autoplot(ba)
```

`plot.deming_regression` *Plot method for deming_regression objects*

Description

Creates publication-ready plots for Deming regression results. Multiple plot types are available: scatter plot with regression line and residual plot.

Usage

```
## S3 method for class 'deming_regression'
plot(
  x,
  type = c("scatter", "residuals"),
  show_ci = TRUE,
  show_identity = TRUE,
  residual_type = c("fitted", "rank"),
  point_alpha = 0.6,
  point_size = 2,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)

## S3 method for class 'deming_regression'
autoplot(
  object,
  type = c("scatter", "residuals"),
  show_ci = TRUE,
  show_identity = TRUE,
  residual_type = c("fitted", "rank"),
  point_alpha = 0.6,
  point_size = 2,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

`x`	An object of class `deming_regression`.
`type`	Character; type of plot to create: • `"scatter"` (default): Scatter plot with regression line, CI band, and identity line • `"residuals"`: Residuals vs. fitted values or rank
`show_ci`	Logical; if `TRUE` (default), displays confidence band for the regression line (only for `type = "scatter"`).
`show_identity`	Logical; if `TRUE` (default), displays the identity line (y = x) for reference.
`residual_type`	Character; for `type = "residuals"`, plot residuals against `"fitted"` (default) or `"rank"` (ordered by x).
`point_alpha`	Numeric; transparency of points (0-1, default: 0.6).
`point_size`	Numeric; size of points (default: 2).
`line_colors`	Named character vector with colors for `"regression"`, `"identity"`, and `"ci"`. Defaults to a clean color scheme.
`title`	Character; plot title. If `NULL` (default), generates an automatic title.
`xlab, ylab`	Character; axis labels. If `NULL`, auto-generates based on variable names.
`...`	Additional arguments (currently ignored).
`object`	An object of class `deming_regression`.

Details

Scatter plot (`type = "scatter"`): Displays the raw data with the fitted Deming regression line and optional confidence band. The identity line (y = x) is shown for reference. If the regression line overlaps substantially with the identity line, the methods are in good agreement.

Residual plot (`type = "residuals"`): Displays perpendicular residuals. Look for:

- Random scatter around zero (good)
- Patterns or trends (suggests non-linearity)
- Funnel shape (suggests heteroscedasticity)

Value

A `ggplot` object that can be further customized.

See Also

`deming_regression()` for performing the analysis, `summary.deming_regression()` for detailed results

Examples

```
set.seed(42)
true_vals <- rnorm(50, 100, 20)
method_a <- true_vals + rnorm(50, sd = 5)
method_b <- 1.05 * true_vals + 3 + rnorm(50, sd = 5)
dm <- deming_regression(method_a, method_b)

# Scatter plot with regression line
plot(dm)

# Without identity line
plot(dm, show_identity = FALSE)

# Residual plot
plot(dm, type = "residuals")

# Residuals by rank
plot(dm, type = "residuals", residual_type = "rank")

# Customized appearance
plot(dm, point_size = 3, title = "Glucose: POC vs Reference")
```

`plot.pb_regression` *Plot method for pb_regression objects*

Description

Creates publication-ready plots for Passing-Bablok regression results. Multiple plot types are available: scatter plot with regression line, residual plot, and CUSUM plot for linearity assessment.

Usage

```
## S3 method for class 'pb_regression'
plot(
  x,
  type = c("scatter", "residuals", "cusum"),
  show_ci = TRUE,
  show_identity = TRUE,
  residual_type = c("fitted", "rank"),
  point_alpha = 0.6,
  point_size = 2,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)

## S3 method for class 'pb_regression'
autoplot(
  object,
  type = c("scatter", "residuals", "cusum"),
  show_ci = TRUE,
  show_identity = TRUE,
  residual_type = c("fitted", "rank"),
  point_alpha = 0.6,
  point_size = 2,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

`x`	An object of class `pb_regression`.
`type`	Character; type of plot to create:

	• `"scatter"` (default): Scatter plot with regression line, CI band, and identity line • `"residuals"`: Residuals vs. fitted values or rank • `"cusum"`: CUSUM plot for linearity assessment
`show_ci`	Logical; if `TRUE` (default), displays confidence band for the regression line (only for `type = "scatter"`).
`show_identity`	Logical; if `TRUE` (default), displays the identity line (y = x) for reference.
`residual_type`	Character; for `type = "residuals"`, plot residuals against `"fitted"` (default) or `"rank"` (ordered by x).
`point_alpha`	Numeric; transparency of points (0-1, default: 0.6).
`point_size`	Numeric; size of points (default: 2).
`line_colors`	Named character vector with colors for `"regression"`, `"identity"`, and `"ci"`. Defaults to a clean color scheme.
`title`	Character; plot title. If `NULL` (default), generates an automatic title.
`xlab, ylab`	Character; axis labels. If `NULL`, auto-generates based on variable names.
`...`	Additional arguments (currently ignored).
`object`	An object of class `pb_regression`.

Details

Scatter plot (`type = "scatter"`): Displays the raw data with the fitted Passing-Bablok regression line and optional confidence band. The identity line (y = x) is shown for reference. If the regression line overlaps substantially with the identity line, the methods are in good agreement.

Residual plot (`type = "residuals"`): Displays perpendicular residuals. Look for:

- Random scatter around zero (good)
- Patterns or trends (suggests non-linearity)
- Funnel shape (suggests heteroscedasticity)

CUSUM plot (`type = "cusum"`): Displays the cumulative sum of residual signs, used to assess linearity. The CUSUM should stay within the critical bounds if the linearity assumption holds.

Value

A `ggplot` object that can be further customized.

See Also

`pb_regression()` for performing the analysis, `summary.pb_regression()` for detailed results

Examples

```
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- 1.05 * method_a + 3 + rnorm(50, sd = 5)
pb <- pb_regression(method_a, method_b)

# Scatter plot with regression line
plot(pb)

# Without identity line
plot(pb, show_identity = FALSE)

# Residual plot
plot(pb, type = "residuals")

# Residuals by rank
plot(pb, type = "residuals", residual_type = "rank")

# CUSUM plot
plot(pb, type = "cusum")

# Customized appearance
plot(pb, point_size = 3, title = "Glucose: POC vs Reference")
```

`print.ate_assessment` *Print method for ate_assessment objects*

Description

Displays a concise summary of the performance assessment.

Usage

```
## S3 method for class 'ate_assessment'
print(x, digits = 2, ...)
```

Arguments

`x`	An object of class `ate_assessment`.
`digits`	Number of decimal places to display (default: 2).
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the input object `x`.

Examples

```
assess <- ate_assessment(bias = 1.5, cv = 2.5, tea = 10)
print(assess)
```

`print.ate_specs` *Print method for ate_specs objects*

Description

Displays a concise summary of allowable total error specifications calculated from biological variation.

Usage

```
## S3 method for class 'ate_specs'
print(x, digits = 2, ...)
```

Arguments

`x`	An object of class `ate_specs`.
`digits`	Number of decimal places to display (default: 2).
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the input object `x`.

Examples

```
ate <- ate_from_bv(cvi = 5.6, cvg = 7.5)
print(ate)
```

`print.ba_analysis` *Print method for ba_analysis objects*

Description

Displays a concise summary of Bland-Altman analysis results.

Usage

```
## S3 method for class 'ba_analysis'
print(x, digits = 3, ...)
```

Arguments

`x`	An object of class `ba_analysis`.
`digits`	Number of significant digits to display (default: 3).
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the input object `x`.

Examples

```
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- method_a + rnorm(50, mean = 2, sd = 5)
ba <- ba_analysis(method_a, method_b)
print(ba)
```

`print.deming_regression` *Print method for deming_regression objects*

Description

Displays a concise summary of Deming regression results, including slope and intercept estimates with confidence intervals.

Usage

```
## S3 method for class 'deming_regression'
print(x, digits = 3, ...)
```

Arguments

`x`	An object of class `deming_regression`.
`digits`	Number of significant digits to display (default: 3).
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the input object.

See Also

`summary.deming_regression()` for detailed output

Examples

```
set.seed(42)
true_vals <- rnorm(50, 100, 20)
method_a <- true_vals + rnorm(50, sd = 5)
method_b <- 1.05 * true_vals + 3 + rnorm(50, sd = 5)
dm <- deming_regression(method_a, method_b)
print(dm)
```

`print.pb_regression` *Print method for pb_regression objects*

Description

Displays a concise summary of Passing-Bablok regression results, including slope and intercept estimates with confidence intervals.

Usage

```
## S3 method for class 'pb_regression'
print(x, digits = 3, ...)
```

Arguments

`x`	An object of class `pb_regression`.
`digits`	Number of significant digits to display (default: 3).
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the input object.

See Also

`summary.pb_regression()` for detailed output

Examples

```
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- 1.05 * method_a + 3 + rnorm(50, sd = 5)
pb <- pb_regression(method_a, method_b)
print(pb)
```

print.sigma_metric *Print method for sigma_metric objects*

Description

Displays a concise summary of the sigma metric calculation.

Usage

```
## S3 method for class 'sigma_metric'
print(x, digits = 2, ...)
```

Arguments

x	An object of class sigma_metric.
digits	Number of decimal places to display (default: 2).
...	Additional arguments (currently ignored).

Value

Invisibly returns the input object x.

Examples

```
sm <- sigma_metric(bias = 1.5, cv = 2.0, tea = 10)
print(sm)
```

print.summary.ba_analysis *Print method for summary.ba_analysis objects*

Description

Print method for summary.ba_analysis objects

Usage

```
## S3 method for class 'summary.ba_analysis'
print(x, digits = 4, ...)
```

Arguments

x	An object of class summary.ba_analysis.
digits	Number of significant digits to display (default: 4).
...	Additional arguments (currently ignored).

Value

Invisibly returns the input object `x`.

`sigma_metric` *Calculate Six Sigma Metric for Analytical Performance*

Description

Calculates the sigma metric, which quantifies analytical performance in terms of the number of standard deviations between observed performance and the allowable total error limit. Higher sigma values indicate better performance and lower defect rates.

Usage

```
sigma_metric(bias, cv, tea)
```

Arguments

`bias`	Numeric. Observed bias (systematic error), expressed as a percentage or in the same units as `tea`.
`cv`	Numeric. Observed coefficient of variation (imprecision), expressed as a percentage.
`tea`	Numeric. Total allowable error specification, expressed as a percentage or in the same units as `bias`.

Details

The sigma metric is calculated as:

$$\sigma = \frac{TEa - |Bias|}{CV}$$

Where:

- TEa = Total allowable error (quality specification)
- Bias = Observed systematic error (absolute value used)
- CV = Observed coefficient of variation

Interpretation Guidelines:

The sigma metric provides a standardized way to assess method performance:

- **>= 6 sigma**: World class performance (<3.4 defects per million)
- **>= 5 sigma**: Excellent performance (~230 defects per million)
- **>= 4 sigma**: Good performance (~6,200 defects per million)
- **>= 3 sigma**: Marginal performance (~66,800 defects per million)
- **< 3 sigma**: Poor performance (unacceptable defect rates)

Note: These defect rates assume a 1.5 sigma shift (industry standard for long-term process variation).

Value

An object of class `c("sigma_metric", "valytics_ate", "valytics_result")`, which is a list containing:

sigma Numeric. The calculated sigma metric value.

input List with input parameters:

- `bias`: Observed bias
- `cv`: Observed CV
- `tea`: Total allowable error

interpretation List with performance interpretation:

- `category`: Performance category (e.g., "World Class", "Good")
- `defect_rate`: Approximate defect rate per million

Clinical Laboratory Context

In clinical laboratories, a sigma metric of 4 or higher is generally considered acceptable for routine testing, while 6 sigma is the gold standard. Methods with sigma < 3 require stringent QC procedures and may not be suitable for clinical use without improvement.

References

Westgard JO, Westgard SA (2006). The quality of laboratory testing today: an assessment of sigma metrics for analytic quality using performance data from proficiency testing surveys and the CLIA criteria for acceptable performance. *American Journal of Clinical Pathology*, 125(3):343-354.

Westgard JO (2008). *Basic Method Validation* (3rd ed.). Westgard QC, Inc.

See Also

`ate_from_bv()` for calculating TEa from biological variation, `ate_assessment()` for comprehensive performance assessment

Examples

```
# Basic sigma calculation
sm <- sigma_metric(bias = 1.5, cv = 2.0, tea = 10)
sm

# World-class performance example
sm_excellent <- sigma_metric(bias = 0.5, cv = 1.0, tea = 8)
sm_excellent

# Marginal performance example
sm_marginal <- sigma_metric(bias = 3.0, cv = 3.0, tea = 12)
sm_marginal

# Using with ate_from_bv() for glucose
ate <- ate_from_bv(cvi = 5.6, cvg = 7.5)
# Assume observed bias = 1.5%, CV = 2.5%
```

```
sm <- sigma_metric(bias = 1.5, cv = 2.5, tea = ate$specifications$tea)
sm

# Access the sigma value directly
sm$sigma
```

summary.ate_assessment

Summary method for ate_assessment objects

Description

Provides a detailed summary of the performance assessment, including calculations and interpretation guidance.

Usage

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

Arguments

`object`	An object of class `ate_assessment`.
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the object.

Examples

```
assess <- ate_assessment(
  bias = 1.5, cv = 2.5, tea = 10,
  allowable_bias = 3.0, allowable_cv = 4.0
)
summary(assess)
```

`summary.ate_specs` *Summary method for ate_specs objects*

Description

Provides a detailed summary of allowable total error specifications, including the formulas used and all three performance tiers for comparison.

Usage

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

Arguments

`object`	An object of class `ate_specs`.
`...`	Additional arguments (currently ignored).

Value

An object of class `summary.ate_specs` containing detailed specification information, printed as a side effect.

Examples

```
ate <- ate_from_bv(cvi = 5.6, cvg = 7.5)
summary(ate)
```

`summary.ba_analysis` *Summary method for ba_analysis objects*

Description

Provides a detailed summary of Bland-Altman analysis results, including additional diagnostics and descriptive statistics.

Usage

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

Arguments

`object`	An object of class `ba_analysis`.
`...`	Additional arguments (currently ignored).

Value

An object of class `summary.ba_analysis` containing:

call The original function call.

n Number of paired observations.

n_excluded Number of pairs excluded due to NAs.

var_names Variable names for x and y.

type Type of difference calculation.

conf_level Confidence level used.

descriptives Data frame with descriptive statistics.

agreement Data frame with agreement statistics.

normality_test Shapiro-Wilk test result for differences.

Examples

```
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- method_a + rnorm(50, mean = 2, sd = 5)
ba <- ba_analysis(method_a, method_b)
summary(ba)
```

---

summary.deming_regression *Summary method for deming_regression objects*

Description

Provides a detailed summary of Deming regression results, including regression coefficients, confidence intervals, standard errors, and interpretation guidance.

Usage

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

Arguments

`object`	An object of class `deming_regression`.
`...`	Additional arguments (currently ignored).

Details

The summary includes:

- Regression coefficients with standard errors and confidence intervals
- Interpretation of slope and intercept CIs
- Method agreement conclusion
- Residual summary statistics

Value

Invisibly returns a list with summary statistics.

See Also

`print.deming_regression()` for concise output

Examples

```
set.seed(42)
true_vals <- rnorm(50, 100, 20)
method_a <- true_vals + rnorm(50, sd = 5)
method_b <- 1.05 * true_vals + 3 + rnorm(50, sd = 5)
dm <- deming_regression(method_a, method_b)
summary(dm)
```

`summary.pb_regression` *Summary method for pb_regression objects*

Description

Provides a detailed summary of Passing-Bablok regression results, including regression coefficients, confidence intervals, linearity test (CUSUM), and interpretation guidance.

Usage

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

Arguments

`object`	An object of class `pb_regression`.
`...`	Additional arguments (currently ignored).

Details

The summary includes:

- Regression coefficients with confidence intervals
- CUSUM test for linearity assumption
- Interpretation of slope and intercept CIs
- Method agreement conclusion

Value

Invisibly returns a list with summary statistics.

See Also

`print.pb_regression()` for concise output

Examples

```
set.seed(42)
method_a <- rnorm(50, mean = 100, sd = 15)
method_b <- 1.05 * method_a + 3 + rnorm(50, sd = 5)
pb <- pb_regression(method_a, method_b)
summary(pb)
```

`summary.sigma_metric` *Summary method for sigma_metric objects*

Description

Provides a detailed summary of the sigma metric calculation, including the formula and interpretation scale.

Usage

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

Arguments

`object`	An object of class `sigma_metric`.
`...`	Additional arguments (currently ignored).

Value

Invisibly returns the object.

Examples

```
sm <- sigma_metric(bias = 1.5, cv = 2.0, tea = 10)
summary(sm)
```

`troponin_cardiac` *Cardiac Troponin Method Comparison Dataset*

Description

Synthetic dataset comparing high-sensitivity cardiac troponin I (hs-cTnI) measurements from two different immunoassay platforms. This dataset illustrates challenges in comparing troponin assays, which lack standardization across manufacturers.

Usage

```
troponin_cardiac
```

Format

A data frame with 50 observations and 3 variables:

sample_id Character. Unique sample identifier.

platform_a Numeric. Troponin I concentration (ng/L) measured by platform A.

platform_b Numeric. Troponin I concentration (ng/L) measured by platform B.

Details

This synthetic dataset was designed to illustrate common patterns in cardiac troponin method comparisons:

- **Concentration range**: 2-5000 ng/L, from near the limit of detection to acute myocardial infarction levels
- **Distribution**: Log-normal (most values low, few very high), reflecting typical clinical populations
- **Bias pattern**: Systematic proportional difference between platforms (~15%), reflecting lack of troponin assay standardization
- **Precision**: Higher CV at low concentrations near the detection limit

Unlike many analytes, cardiac troponin assays are not standardized, meaning results from different manufacturers are not directly comparable. This has clinical implications for interpreting troponin values when patients are tested at different institutions.

Source

Synthetic data generated to mimic realistic clinical patterns. See `data-raw/make_datasets.R` for the generation script.

References

Apple FS, et al. Cardiac Troponin Assays: Guide to Understanding Analytical Characteristics and Their Impact on Clinical Care. Clin Chem. 2017;63(1):73-81.

See Also

`ba_analysis()`, glucose_methods, creatinine_serum

Examples

```
# Bland-Altman analysis with percent differences
# (appropriate for proportional bias)
ba <- ba_analysis(platform_a ~ platform_b,
                  data = troponin_cardiac,
                  type = "percent")
summary(ba)
plot(ba)
```

Index

∗ **datasets**
    creatinine_serum, 10
    glucose_methods, 15
    troponin_cardiac, 38

ate_assessment, 2
ate_assessment(), *7*, *32*
ate_from_bv, 5
ate_from_bv(), *4*, *32*
autoplot.ba_analysis
        *(*plot.ba_analysis*)*, 19
autoplot.deming_regression
        *(*plot.deming_regression*)*, 21
autoplot.pb_regression
        *(*plot.pb_regression*)*, 24

ba_analysis, 7
ba_analysis(), *11*, *14*, *15*, *18*, *21*, *39*

creatinine_serum, 10, *15*, *39*

deming_regression, 11
deming_regression(), *23*

glucose_methods, *11*, 15, *39*

pb_regression, 16
pb_regression(), *14*, *25*
plot.ba_analysis, 19
plot.ba_analysis(), *9*
plot.deming_regression, 21
plot.deming_regression(), *14*
plot.pb_regression, 24
plot.pb_regression(), *18*
print.ate_assessment, 26
print.ate_specs, 27
print.ba_analysis, 27
print.deming_regression, 28
print.deming_regression(), *36*
print.pb_regression, 29
print.pb_regression(), *37*
print.sigma_metric, 30
print.summary.ba_analysis, 30

sigma_metric, 31
sigma_metric(), *4*, *7*
summary.ate_assessment, 33
summary.ate_specs, 34
summary.ba_analysis, 34
summary.ba_analysis(), *9*, *21*
summary.deming_regression, 35
summary.deming_regression(), *14*, *23*, *28*
summary.pb_regression, 36
summary.pb_regression(), *18*, *25*, *29*
summary.sigma_metric, 37

troponin_cardiac, *11*, *15*, 38