Package: valytics (via r-universe)

June 7, 2026

Title Statistical Methods for Analytical Method Comparison and Validation

Version 0.4.1

Date 2026-02-19

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, precision experiments with variance component analysis, precision profiles for functional sensitivity estimation (Kroll & Emancipator (1993) <https://pubmed.ncbi.nlm.nih.gov/8448849/>). Commonly used in clinical laboratory method validation. Provides publication-ready plots and comprehensive statistical summaries.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

LazyData true

Depends R (>= 3.5.0)

Imports ggplot2, robslopes, stats, utils

Suggests testthat (>= 3.0.0), knitr (>= 1.43), rmarkdown (>= 2.22), lme4 (>= 1.1-33)

Config/testthat/edition 3

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

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

VignetteBuilder knitr

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

Date/Publication 2026-02-24 10:25:13 UTC

RemoteUrl https://github.com/marcellogr/valytics

RemoteRef HEAD

RemoteSha a46a33d76a33cd3d821fe556241a164c5b311a70

Contents

`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_y² / CV_x² (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")
```

---

plot.precision_profile

Plot method for precision_profile objects

Description

Creates publication-ready visualization of precision profile results, showing CV vs concentration with the fitted model curve.

Usage

```
## S3 method for class 'precision_profile'
plot(
  x,
  show_ci = TRUE,
  show_targets = TRUE,
  show_points = TRUE,
  point_alpha = 0.8,
  point_size = 3,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
```

```
  log_x = FALSE,
  ...
)

## S3 method for class 'precision_profile'
autoplot(
  object,
  show_ci = TRUE,
  show_targets = TRUE,
  show_points = TRUE,
  point_alpha = 0.8,
  point_size = 3,
  line_colors = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  log_x = FALSE,
  ...
)
```

Arguments

x	An object of class `precision_profile`.
show_ci	Logical; if `TRUE` (default), displays prediction interval bands for the fitted curve.
show_targets	Logical; if `TRUE` (default), displays horizontal lines at functional sensitivity target CV values.
show_points	Logical; if `TRUE` (default), displays the observed data points.
point_alpha	Numeric; transparency of points (0-1, default: 0.8).
point_size	Numeric; size of points (default: 3).
line_colors	Named character vector with colors for `"fitted"`, `"ci"`, and `"target"`. 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 "Concentration".
ylab	Character; y-axis label. If `NULL`, uses "CV (%)".
log_x	Logical; if `TRUE`, uses logarithmic scale for x-axis (default: `FALSE`).
...	Additional arguments (currently ignored).
object	An object of class `precision_profile`.

Details

The precision profile plot displays:

- **Observed points**: CV values at each tested concentration
- **Fitted curve**: Model-predicted CV across the concentration range
- **Prediction intervals**: Confidence bands showing uncertainty

- **Target lines**: Horizontal lines at functional sensitivity thresholds

The plot helps visualize:

- How measurement precision changes with concentration
- Model fit quality (points should follow the curve)
- Functional sensitivity estimates (intersection of curve with target lines)

Value

A ggplot object that can be further customized.

Examples

```
# See ?precision_profile for complete examples
```

---

plot.precision_study *Plot method for precision_study objects*

Description

Creates visualizations for precision study results. Multiple plot types are available: variance component chart, CV profile across samples, and precision estimates summary.

Usage

```
## S3 method for class 'precision_study'
plot(
  x,
  type = c("variance", "cv", "precision"),
  show_ci = TRUE,
  colors = NULL,
  title = NULL,
  ...
)

## S3 method for class 'precision_study'
autoplot(
  object,
  type = c("variance", "cv", "precision"),
  show_ci = TRUE,
  colors = NULL,
  title = NULL,
  ...
)
```

Arguments

x	An object of class precision_study.
type	Character; type of plot to create: • "variance" (default): Bar chart showing variance components as percentage of total variance • "cv": CV profile across samples (requires multi-sample data) • "precision": Forest plot of precision estimates with CIs
show_ci	Logical; if TRUE (default), displays confidence intervals where applicable.
colors	Character vector of colors for the plot elements. If NULL, uses a default color palette.
title	Character; plot title. If NULL (default), generates an automatic title.
...	Additional arguments (currently ignored).
object	An object of class precision_study.

Details

Variance component chart (type = "variance"): Displays the proportion of total variance attributable to each source (between-day, between-run, error/repeatability). Helps identify which factors contribute most to measurement variability.

CV profile (type = "cv"): For multi-sample studies, displays how CV varies across concentration levels. Typically CV is higher at low concentrations. Requires data from multiple samples/levels.

Precision summary (type = "precision"): Forest plot showing precision estimates (repeatability, intermediate precision, reproducibility) with confidence intervals.

Value

A ggplot object that can be further customized.

See Also

precision_study() for performing the analysis, summary.precision_study() for detailed results

Examples

```
# Create example data
set.seed(42)
prec_data <- data.frame(
  day = rep(1:5, each = 6),
  run = rep(rep(1:2, each = 3), 5),
  value = rnorm(30, mean = 100, sd = 5)
)
prec_data$value <- prec_data$value + rep(rnorm(5, 0, 3), each = 6)

prec <- precision_study(prec_data, value = "value", day = "day", run = "run")

# Variance component chart (default)
```

```
plot(prec)
plot(prec, type = "variance")

# Precision estimates with CIs
plot(prec, type = "precision")
```

`precision_profile` *Precision Profile Analysis*

Description

Constructs a precision profile (CV vs concentration relationship) from precision study results and estimates functional sensitivity. The precision profile characterizes how measurement imprecision changes across the analytical measurement interval.

Usage

```
precision_profile(
  x,
  concentration = "concentration",
  cv = "cv_pct",
  model = c("hyperbolic", "linear"),
  cv_targets = c(10, 20),
  conf_level = 0.95,
  bootstrap = FALSE,
  boot_n = 1999
)
```

Arguments

`x`	An object of class `precision_study` with multiple concentration levels, OR a data frame with columns for concentration and CV values.
`concentration`	Character string specifying the column name for concentration values (only used if `x` is a data frame). Default is `"concentration"`.
`cv`	Character string specifying the column name for CV values (only used if `x` is a data frame). Default is `"cv_pct"`.
`model`	Regression model for CV-concentration relationship: `"hyperbolic"` (default) fits CV = sqrt(a^2 + (b/x)^2), `"linear"` fits CV = a + b/x.
`cv_targets`	Numeric vector of target CV percentages for functional sensitivity estimation. Default is `c(10, 20)`.
`conf_level`	Confidence level for prediction intervals (default: 0.95).
`bootstrap`	Logical; if `TRUE`, uses bootstrap resampling for confidence intervals on functional sensitivity estimates. Default is `FALSE`.
`boot_n`	Number of bootstrap resamples when `bootstrap = TRUE` (default: 1999).

Details

Precision Profile:

The precision profile describes how analytical imprecision (CV) varies across the analytical measurement interval. Typically, CV decreases as concentration increases, following a hyperbolic relationship.

Hyperbolic Model:

The hyperbolic model is:

$$CV = \sqrt{a^2 + (b/x)^2}$$

where:

- a represents the asymptotic CV at high concentrations
- b represents the concentration-dependent component
- x is the analyte concentration

This model captures the characteristic behavior where CV approaches a constant value at high concentrations and increases hyperbolically at low concentrations.

Linear Model:

The linear model is:

$$CV = a + b/x$$

This is a simpler alternative that may be appropriate when the relationship is approximately linear when plotted as CV vs 1/concentration.

Functional Sensitivity:

Functional sensitivity is defined as the lowest concentration at which a measurement procedure achieves a specified level of precision (CV). Common thresholds are:

- **10% CV**: Modern standard for high-sensitivity assays (e.g., cardiac troponin)
- **20% CV**: Traditional standard (originally defined for TSH assays)

The functional sensitivity is calculated by solving the fitted model equation for the concentration that yields the target CV.

Value

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

input List with original data:

- `concentration`: Numeric vector of concentrations
- `cv`: Numeric vector of CV values (percent)
- `n_levels`: Number of concentration levels
- `conc_range`: Concentration range (min, max)
- `conc_span`: Fold-difference (max/min)

model List with fitted model information:

- `type`: Model type ("hyperbolic" or "linear")
- `parameters`: Named vector of fitted parameters
- `equation`: Character string describing the fitted equation

fitted Data frame with fitted values:

- `concentration`: Concentration values
- `cv_observed`: Observed CV values
- `cv_fitted`: Model-fitted CV values
- `residual`: Residuals (observed - fitted)
- `ci_lower`: Lower prediction interval
- `ci_upper`: Upper prediction interval

fit_quality List with goodness-of-fit statistics:

- `r_squared`: Coefficient of determination
- `adj_r_squared`: Adjusted R-squared
- `rmse`: Root mean squared error
- `mae`: Mean absolute error

functional_sensitivity Data frame with functional sensitivity estimates:

- `cv_target`: Target CV percentage
- `concentration`: Estimated concentration at target CV
- `ci_lower`: Lower confidence limit (if bootstrap)
- `ci_upper`: Upper confidence limit (if bootstrap)
- `achievable`: Logical; TRUE if target CV is achievable

settings List with analysis settings

call The matched function call

Minimum Requirements

- At least 4 concentration levels
- Concentration span of at least 2-fold (warning if less)
- Valid CV estimates at each level (from precision study)

References

Armbruster DA, Pry T (2008). Limit of blank, limit of detection and limit of quantitation. *Clinical Biochemist Reviews*, 29(Suppl 1):S49-S52.

CLSI EP17-A2 (2012). Evaluation of Detection Capability for Clinical Laboratory Measurement Procedures; Approved Guideline - Second Edition. Clinical and Laboratory Standards Institute, Wayne, PA.

Kroll MH, Emancipator K (1993). A theoretical evaluation of linearity. *Clinical Chemistry*, 39(3):405-413.

See Also

`precision_study()` for variance component analysis, `plot.precision_profile()` for visualization

Examples

```
# Example with simulated multi-level precision data
set.seed(42)

# Generate data for 6 concentration levels
conc_levels <- c(5, 10, 25, 50, 100, 200)
n_levels <- length(conc_levels)

prec_data <- data.frame()
for (i in seq_along(conc_levels)) {
  level_data <- expand.grid(
    level = conc_levels[i],
    day = 1:5,
    replicate = 1:5
  )

  # Simulate CV that decreases with concentration
  true_cv <- sqrt(3^2 + (20/conc_levels[i])^2)
  level_data$value <- conc_levels[i] * rnorm(
    nrow(level_data),
    mean = 1,
    sd = true_cv/100
  )

  prec_data <- rbind(prec_data, level_data)
}

# Run precision study
prec <- precision_study(
  data = prec_data,
  value = "value",
  sample = "level",
  day = "day"
)

# Generate precision profile
profile <- precision_profile(prec)
print(profile)
summary(profile)

# Hyperbolic model with bootstrap CIs
profile_boot <- precision_profile(
  prec,
  model = "hyperbolic",
  cv_targets = c(10, 20),
  bootstrap = TRUE,
  boot_n = 499
)

# Linear model
profile_linear <- precision_profile(prec, model = "linear")
```

precision_study *Precision Study Analysis*

Description

Performs variance component analysis for precision experiments following established methodology for clinical laboratory method validation. Estimates repeatability, Within-laboratory precision, and reproducibility from nested experimental designs.

Usage

```
precision_study(
  data,
  value = "value",
  sample = NULL,
  site = NULL,
  day = "day",
  run = NULL,
  replicate = NULL,
  conf_level = 0.95,
  ci_method = c("satterthwaite", "mls", "bootstrap"),
  boot_n = 1999,
  method = c("anova", "reml")
)
```

Arguments

data	A data frame containing the precision experiment data.
value	Character string specifying the column name containing measurement values. Default is "value".
sample	Character string specifying the column name for sample/level identifier. Use when multiple concentration levels are tested. Default is NULL (single sample).
site	Character string specifying the column name for site/device identifier. Use for multi-site reproducibility studies. Default is NULL (single site).
day	Character string specifying the column name for day identifier. Default is "day".
run	Character string specifying the column name for run identifier (within day). Default is NULL (assumes single run per day).
replicate	Character string specifying the column name for replicate identifier. If NULL (default), replicates are inferred from the data structure.
conf_level	Confidence level for intervals (default: 0.95).
ci_method	Method for calculating confidence intervals: "satterthwaite" (default) uses the Satterthwaite approximation, "mls" uses the Modified Large Sample method, "bootstrap" uses BCa bootstrap resampling.
boot_n	Number of bootstrap resamples when ci_method = "bootstrap" (default: 1999).

`method` Estimation method for variance components: `"anova"` (default) uses ANOVA-based method of moments, `"reml"` uses Restricted Maximum Likelihood (requires lme4 package).

Details

This function implements variance component analysis for nested experimental designs commonly used in clinical laboratory precision studies. The analysis follows methodology consistent with international standards.

Supported Experimental Designs:

- **Single-site, day/run/replicate**: Classic 20 x 2 x 2 design (20 days, 2 runs per day, 2 replicates per run)
- **Single-site, day/replicate**: Simplified design without run factor (e.g., 5 days x 5 replicates for verification)
- **Multi-site**: 3 sites x 5 days x 5 replicates for reproducibility
- **Custom designs**: Any fully-nested combination of factors

Variance Components:

For a design with site/day/run/replicate, the model is:

$$y_{ijkl} = \mu + S_i + D_{j(i)} + R_{k(ij)} + \epsilon_{l(ijk)}$$

where S = site, D = day (nested in site), R = run (nested in day), and epsilon = residual error.

Precision Measures:

- **Repeatability**: Within-run precision (sqrt of error variance)
- **Between-run precision**: Additional variability between runs
- **Between-day precision**: Additional variability between days
- **Within-laboratory precision**: Within-laboratory precision (combines day, run, and error variance)
- **Reproducibility**: Total precision including between-site variability (for multi-site designs)

Value

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

input List with original data and metadata:

- `data`: The input data frame (after validation)
- `n`: Total number of observations
- `n_excluded`: Number of observations excluded due to NAs
- `factors`: Named list of factor column names used
- `value_col`: Name of the value column

design List describing the experimental design:

- `type`: Design type (e.g., "single_site", "multi_site")
- `structure`: Character string describing nesting (e.g., "day/run")
- `levels`: Named list with number of levels for each factor
- `balanced`: Logical; TRUE if design is balanced
- `n_samples`: Number of distinct samples/concentration levels

variance_components Data frame with variance component estimates:

- `component`: Name of variance component
- `variance`: Estimated variance
- `sd`: Standard deviation (sqrt of variance
- `pct_total`: Percentage of total variance
- `df`: Degrees of freedom

precision Data frame with precision estimates:

- `measure`: Precision measure name (repeatability, intermediate, etc.)
- `sd`: Standard deviation
- `cv_pct`: Coefficient of variation (percent)
- `ci_lower`: Lower confidence limit
- `ci_upper`: Upper confidence limit

anova_table ANOVA table with SS, MS, DF for each source of variation

by_sample If multiple samples: list of results per sample

settings List with analysis settings

call The matched function call

Confidence Intervals

Three methods are available for confidence interval estimation:

- **Satterthwaite** (default): Uses Satterthwaite's approximation for degrees of freedom of linear combinations of variance components.
- **MLS**: Modified Large Sample method, which can provide better coverage when variance components may be estimated as negative.
- **Bootstrap**: BCa bootstrap resampling. Most robust but computationally intensive.

ANOVA vs REML

- **ANOVA** (default): Method of moments estimation. Works well for balanced designs. May produce negative variance estimates for small variance components (set to zero by default).
- **REML**: Restricted Maximum Likelihood. Preferred for unbalanced designs. Requires the lme4 package. Always produces non-negative estimates.

References

Chesher D (2008). Evaluating assay precision. *Clinical Biochemist Reviews*, 29(Suppl 1):S23-S26.

ISO 5725-2:2019. Accuracy (trueness and precision) of measurement methods and results - Part 2: Basic method for the determination of repeatability and reproducibility of a standard measurement method.

Searle SR, Casella G, McCulloch CE (1992). *Variance Components*. Wiley, New York.

Satterthwaite FE (1946). An approximate distribution of estimates of variance components. *Biometrics Bulletin*, 2:110-114.

See Also

`verify_precision()` for comparing results to manufacturer claims, `plot.precision_study()` for visualization, `summary.precision_study()` for detailed summary

Examples

```
# Example with simulated precision data
set.seed(42)

# Generate study design: 20 days x 2 runs x 2 replicates
n_days <- 20
n_runs <- 2
n_reps <- 2

prec_data <- expand.grid(
  day = 1:n_days,
  run = 1:n_runs,
  replicate = 1:n_reps
)

# Add realistic variance components
day_effect <- rep(rnorm(n_days, 0, 1.5), each = n_runs * n_reps)
run_effect <- rep(rnorm(n_days * n_runs, 0, 1.0), each = n_reps)
error <- rnorm(nrow(prec_data), 0, 2.0)

prec_data$value <- 100 + day_effect + run_effect + error

# Run precision study
prec <- precision_study(
  data = prec_data,
  value = "value",
  day = "day",
  run = "run"
)

print(prec)
summary(prec)
```

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.precision_profile *Print method for precision_profile objects*

Description

Displays a concise summary of precision profile results, including model fit and functional sensitivity estimates.

Usage

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

Arguments

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

Value

Invisibly returns the input object x.

Examples

```
# See ?precision_profile for examples
```

---

print.precision_study *Print method for precision_study objects*

Description

Displays a concise summary of precision study results, including variance components and key precision estimates.

Usage

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

Arguments

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

Value

Invisibly returns the input object x.

See Also

summary.precision_study() for detailed output

Examples

```
# Create example data
set.seed(42)
data <- data.frame(
  day = rep(1:5, each = 4),
  value = rnorm(20, mean = 100, sd = 5)
)
data$value <- data$value + rep(rnorm(5, 0, 3), each = 4)

prec <- precision_study(data, value = "value", day = "day")
print(prec)
```

---

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.

print.summary.precision_profile

Print method for summary.precision_profile objects

Description

Print method for summary.precision_profile objects

Usage

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

Arguments

x	An object of class summary.precision_profile.
...	Additional arguments (currently ignored).

Value

Invisibly returns the input object x.

`print.summary.precision_study` *Print method for summary.precision_study objects*

Description

Print method for summary.precision_study objects

Usage

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

Arguments

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

Value

Invisibly returns the input object `x`.

`print.summary.verify_precision` *Print method for summary.verify_precision objects*

Description

Print method for summary.verify_precision objects

Usage

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

Arguments

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

Value

Invisibly returns the input object `x`.

print.verify_precision *Print method for verify_precision objects*

Description

Displays verification results including observed vs. claimed precision, test statistics, and verification outcome.

Usage

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

Arguments

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

Value

Invisibly returns the input object x.

See Also

summary.verify_precision() for detailed output

Examples

```
set.seed(42)
measurements <- rnorm(25, mean = 100, sd = 3.5)
result <- verify_precision(measurements, claimed_cv = 4, mean_value = 100)
print(result)
```

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.precision_profile

Summary method for precision_profile objects

Description

Provides a detailed summary of precision profile results, including fitted values, residuals, fit statistics, and functional sensitivity estimates.

Usage

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

Arguments

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

Value

An object of class `summary.precision_profile` containing summary statistics.

Examples

```
# See ?precision_profile for examples
```

summary.precision_study *Summary method for precision_study objects*

Description

Provides a detailed summary of precision study results, including variance components, ANOVA table (for ANOVA method), precision estimates with confidence intervals, and design information.

Usage

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

Arguments

object	An object of class precision_study.
...	Additional arguments (currently ignored).

Value

An object of class summary.precision_study containing:

call The original function call.

n Number of observations.

n_excluded Number of observations excluded due to NAs.

design Design information list.

settings Analysis settings.

variance_components Data frame of variance components.

precision Data frame of precision estimates.

anova_table ANOVA table (if method = "anova").

by_sample Results by sample (if multiple samples).

See Also

print.precision_study() for concise output

Examples

```
# Create example data
set.seed(42)
data <- data.frame(
  day = rep(1:5, each = 4),
  value = rnorm(20, mean = 100, sd = 5)
)
data$value <- data$value + rep(rnorm(5, 0, 3), each = 4)
```

```
prec <- precision_study(data, value = "value", day = "day")
summary(prec)
```

---

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)
```

---

summary.verify_precision *Summary method for verify_precision objects*

Description

Provides a detailed summary of precision verification results, including confidence intervals, test details, and interpretation guidance.

Usage

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

Arguments

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

Value

An object of class `summary.verify_precision` containing:

call The original function call.

input Input information.

observed Observed precision statistics.

claimed Claimed precision statistics.

test Hypothesis test results.

verification Verification outcome.

ci Confidence intervals.

settings Analysis settings.

See Also

`print.verify_precision()` for concise output

Examples

```
set.seed(42)
measurements <- rnorm(25, mean = 100, sd = 3.5)
result <- verify_precision(measurements, claimed_cv = 4, mean_value = 100)
summary(result)
```

`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)
```

`troponin_precision` *High-Sensitivity Cardiac Troponin I Precision Dataset*

Description

Synthetic dataset for evaluating precision of a high-sensitivity cardiac troponin I (hs-cTnI) assay across multiple concentration levels. The data is designed for demonstrating precision studies with variance component analysis and precision profile estimation.

Usage

```
troponin_precision
```

Format

A data frame with 120 rows and 6 variables:

level Concentration level factor (L1-L6)

day Day of measurement (D1-D5)

run Run within day (R1-R2)

replicate Replicate within run (1-2)

value Measured concentration (ng/L)

target Nominal target concentration (ng/L)

Details

This synthetic dataset simulates a multi-level precision study for a high-sensitivity cardiac troponin I assay. The data characteristics:

- **Concentration levels**: 5, 10, 25, 50, 100, 500 ng/L
- **Design**: 5 days x 2 runs x 2 replicates per level (EP05-style)
- **Total observations**: 120 (6 levels x 20 measurements each)
- **Precision pattern**: CV decreases with concentration following a hyperbolic relationship

The precision profile follows the model:

$$CV = \sqrt{a^2 + (b/x)^2}$$

where approximately:

- $a \approx 3\%$ (asymptotic CV at high concentrations)
- $b \approx 25$ (concentration-dependent component)

This gives expected CVs of approximately:

- 5 ng/L: ~5.5%

- 10 ng/L: ~3.5%
- 25 ng/L: ~3.0%
- 50 ng/L: ~2.0%
- 100 ng/L: ~2.0%
- 500 ng/L: ~3.5%

Clinical Context

High-sensitivity cardiac troponin assays are used to diagnose acute myocardial infarction (AMI). Key clinical decision points include:

- 99th percentile upper reference limit (URL): typically 14-26 ng/L
- Functional sensitivity (CV <= 10%): should be <= 50% of 99th percentile
- Precision at low concentrations is critical for early AMI detection

Source

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

See Also

`precision_study()` for variance component analysis, `precision_profile()` for CV-concentration modeling, glucose_methods, creatinine_serum, troponin_cardiac

Examples

```
# Load the dataset
data(troponin_precision)
head(troponin_precision)

# Precision study with multiple concentration levels
prec <- precision_study(
  data = troponin_precision,
  value = "value",
  sample = "level",
  day = "day",
  run = "run"
)
print(prec)

# Results for each concentration level
names(prec$by_sample)

# Generate precision profile
profile <- precision_profile(prec, cv_targets = c(10, 20))
print(profile)
plot(profile)

# Functional sensitivity at 10% CV
```

```
profile$functional_sensitivity
```

`verify_precision` *Precision Verification Against Manufacturer Claims*

Description

Compares observed precision (CV or SD) to manufacturer's claimed performance using statistical hypothesis testing. This function implements verification protocols for validating that an analytical method meets specified precision goals.

Usage

```
verify_precision(
  x,
  claimed_cv = NULL,
  claimed_sd = NULL,
  mean_value = NULL,
  alpha = 0.05,
  alternative = c("less", "two.sided", "greater"),
  conf_level = 0.95,
  value = "value",
  day = "day",
  run = NULL,
  ...
)
```

Arguments

`x`	Either a numeric vector of measurements, a `precision_study` object, or a data frame containing precision study data.
`claimed_cv`	Manufacturer's claimed coefficient of variation (as percent). Either `claimed_cv` or `claimed_sd` must be provided.
`claimed_sd`	Manufacturer's claimed standard deviation. Either `claimed_cv` or `claimed_sd` must be provided.
`mean_value`	Mean concentration of the sample. Required when `x` is a numeric vector and `claimed_cv` is used, or when `claimed_sd` is used and CV-based comparison is desired.
`alpha`	Significance level for the hypothesis test (default: 0.05).
`alternative`	Type of alternative hypothesis: `"less"` (default) tests if observed is not worse than claimed, `"two.sided"` tests for any difference, `"greater"` tests if observed is worse than claimed.
`conf_level`	Confidence level for intervals (default: 0.95).

value	Character string specifying the column name containing measurement values when x is a data frame. Default is "value".
day	Character string specifying the column name for day identifier when x is a data frame. Default is "day".
run	Character string specifying the column name for run identifier when x is a data frame. Default is NULL.
...	Additional arguments passed to precision_study() when x is a data frame.

Details

Statistical Test:

The verification uses a chi-square test comparing observed variance to claimed variance:

$$\chi^2 = \frac{(n-1) \cdot s^2}{\sigma^2_{claimed}}$$

where s^2 is the observed sample variance and $\sigma^2_{claimed}$ is the manufacturer's claimed variance.

Hypothesis Testing:

For alternative = "less" (default, recommended for verification):

- H0: True precision is worse than or equal to claimed
- H1: True precision is better than or equal to claimed
- Verification passes if observed precision is not significantly worse

For typical verification studies, the observed CV should not exceed the manufacturer's claimed CV by more than expected from sampling variability.

Verification Limit:

The upper verification limit (UVL) represents the maximum observed CV that would still be consistent with the claimed CV at the given significance level:

$$UVL = CV_{claimed} \cdot \sqrt{\frac{\chi^2_{1-\alpha,df}}{df}}$$

If observed CV <= UVL, precision is verified.

Value

An object of class c("verify_precision", "valytics_precision", "valytics_result"), which is a list containing:

input List with input data and metadata:

- n: Number of observations
- df: Degrees of freedom for the test
- mean_value: Mean of measurements
- source: Description of input source

observed List with observed precision:

- `sd`: Observed standard deviation
- `cv_pct`: Observed CV (percent)
- `variance`: Observed variance

claimed List with manufacturer's claimed precision:

- `sd`: Claimed SD
- `cv_pct`: Claimed CV (percent)
- `variance`: Claimed variance

test List with hypothesis test results:

- `statistic`: Chi-square test statistic
- `df`: Degrees of freedom
- `p_value`: P-value
- `alternative`: Alternative hypothesis used
- `method`: Test method description

verification List with verification outcome:

- `verified`: Logical; TRUE if precision is verified
- `ratio`: Ratio of observed variance to claimed variance
- `cv_ratio`: Ratio of observed CV to claimed CV
- `upper_verification_limit`: Upper limit for verification

ci List with confidence intervals:

- `sd_ci`: CI for standard deviation
- `cv_ci`: CI for CV (percent)
- `variance_ci`: CI for variance

settings List with analysis settings

call The matched function call

Input Options

The function accepts three types of input:

- **Numeric vector**: Raw measurements (simplest case)
- **precision_study object**: Uses within-laboratory precision from a previous analysis
- **Data frame**: Runs `precision_study()` internally with specified factors

References

Chesher D (2008). Evaluating assay precision. *Clinical Biochemist Reviews*, 29(Suppl 1):S23-S26.

ISO 5725-6:1994. Accuracy (trueness and precision) of measurement methods and results - Part 6: Use in practice of accuracy values.

See Also

`precision_study()` for full precision analysis, `ate_assessment()` for total error assessment

Examples

```
# Example 1: Verify precision from raw measurements
set.seed(42)
measurements <- rnorm(25, mean = 100, sd = 3.5)

# Manufacturer claims CV = 4%
result <- verify_precision(measurements, claimed_cv = 4, mean_value = 100)
print(result)

# Example 2: Verify precision from a precision_study object
prec_data <- data.frame(
  day = rep(1:5, each = 5),
  value = rnorm(25, mean = 100, sd = 3)
)
prec_data$value <- prec_data$value + rep(rnorm(5, 0, 1.5), each = 5)

prec <- precision_study(prec_data, value = "value", day = "day")
result <- verify_precision(prec, claimed_cv = 5)
print(result)

# Example 3: Verify precision directly from data frame
result <- verify_precision(
  prec_data,
  claimed_cv = 5,
  value = "value",
  day = "day"
)
print(result)
```

Index