

Package ‘ivdtools’

August 20, 2026

Title Statistical Tools for Evaluation of in Vitro Diagnostic Reagents

Version 0.1.2

Description Provides statistical workflows used in the evaluation of in vitro diagnostic reagents. Facilities include method comparison and Bland-Altman analysis, receiver operating characteristic analysis, qualitative agreement, precision and variance-component analysis, reference intervals, stability studies, quality-control charts, curve fitting, analytical sensitivity, outlier and normality assessment, and sample-size calculations. For methodological details, see Bland and Altman (1986) <[doi:10.1016/S0140-6736\(86\)90837-8](https://doi.org/10.1016/S0140-6736(86)90837-8)>, Passing and Bablok (1983) <[doi:10.1515/cclm.1983.21.11.709](https://doi.org/10.1515/cclm.1983.21.11.709)>, Linnet (1993) <[doi:10.1093/clinchem/39.3.424](https://doi.org/10.1093/clinchem/39.3.424)>, Hanley and McNeil (1982) <[doi:10.1148/radiology.143.1.7063747](https://doi.org/10.1148/radiology.143.1.7063747)>, Horn et al. (1998) <[doi:10.1093/clinchem/44.3.622](https://doi.org/10.1093/clinchem/44.3.622)>, Westgard et al. (1981) <[doi:10.1093/clinchem/27.3.493](https://doi.org/10.1093/clinchem/27.3.493)>, and Lu et al. (2016) <[doi:10.1515/ijb-2015-0039](https://doi.org/10.1515/ijb-2015-0039)>.

License MIT + file LICENSE

URL <https://github.com/hiox-tech/ivdtools>

BugReports <https://github.com/hiox-tech/ivdtools/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ggplot2, ggrepel, minpack.lm, nloptr, nls2, nortest, rlang, stats, utils, VCA, VFP

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Author hiox-tech [cph, aut, cre]

Maintainer hiox-tech <GeorgeBinDragon@outlook.com>

Repository CRAN

Date/Publication 2026-08-20 17:00:02 UTC

Contents

arrhenius	4
auc.roc	5
bias.mcr	6
bland_altman.mcr	7
bottle_anova	8
ci.precision	9
coef.fit_equation	10
compare_equation	10
continuous_to_binary	11
correlation.mcr	12
counts_to_table	12
cutoff.roc	13
describe	14
describe.fourfold_table	15
describe.mcr	16
describe.roc	17
diagnostics.fourfold_table	17
fit_equation	18
ivd_bottle_example	21
ivd_mcr_example	21
ivd_qualitative_example	22
ivd_roc_example	22
kappa.fourfold_table	23
list_equation	24
list_sadler	24
list_westgard	25
lob_lod_loq	26
mcnemar.fourfold_table	28
mcr	29
mkt	30
mlr.roc	30
normal.precision	31
normal_test	32
outlier	33
outlier.mcr	33
outlier.precision	34
outliers_test	35
plot.fit_equation	36
plot.mcr	37
plot.precision	38
plot.reference_interval	39
plot.roc	39
precision	40
predict.fit_equation	41
predict.mcr	42
predict.roc	43

print.bottle_anova	44
print.describe_table	45
print.diagnostics	45
print.fit_equation	46
print.fourfold_table	47
print.kappa_table	47
print.mcnemar_table	48
print.mcr	49
print.mcr_bias	49
print.mcr_bland_altman	50
print.mcr_correlation	50
print.mcr_describe	51
print.mcr_outlier	51
print.mcr_regression	52
print.precision	52
print.precision_ci	53
print.precision_normal	53
print.precision_outlier	54
print.precision_profile	54
print.precision_vc	55
print.reference_interval	55
print.roc	56
print.roc_auc_table	56
print.roc_cutoff_table	57
print.roc_describe	57
print.roc_mlr	58
print.tukey	58
profile.precision	59
qc_chart	59
raw_to_table	60
reference_interval	61
regression.mcr	63
replicate_to_mean	64
residuals.fit_equation	65
roc	65
sample_size_bland_altman	66
sample_size_proportion	68
sample_size_proportion_ci	69
stability_bias	70
stability_plan	71
stability_regression	72
stability_time	73
summary.fourfold_table	74
summary.mcr	75
summary.precision	76
summary.roc	76
tukey	77
tukey.bottle_anova	78

variance.precision	79
youden_plot	79

Index	81
--------------	-----------

arrhenius	<i>arrhenius — Arrhenius accelerated stability analysis</i>
-----------	---

Description

Estimates degradation rates at multiple elevated temperatures, fits the Arrhenius equation, and extrapolates to the target storage temperature to predict shelf life.

Usage

```
arrhenius(  
  data,  
  temperature,  
  time,  
  value,  
  target_temp,  
  order = c("auto", "zero", "first"),  
  direction = c("auto", "decrease", "increase"),  
  limit,  
  bias_type = c("relative", "absolute"),  
  temp_unit = "C",  
  time_unit = "d",  
  conf.level = 0.95  
)
```

Arguments

data	A data frame.
temperature	Column name for storage temperature (numeric).
time	Column name for time point (numeric).
value	Column name for measurement value (numeric).
target_temp	Target storage temperature.
order	"auto", "zero", or "first".
direction	"auto", "decrease", or "increase".
limit	A positive number for symmetric allowable bias.
bias_type	"relative" (percent) or "absolute".
temp_unit	Temperature unit: "C" or "K".
time_unit	Time unit: m, min, h, d, month, or y.
conf.level	Confidence level, default 0.95.

Value

An S3 object of class "arrhenius".

Examples

```
temp_df <- data.frame(temp = rep(c(25,30,37), each = 3),
                      time = rep(c(0,1,3), 3),
                      val = c(100,98,95, 100,96,90, 100,94,85))
arrhenius(temp_df, "temp", "time", "val", target_temp = 5, limit = 10)
```

auc.roc	<i>Compute ROC curves and AUC for each evaluation column.</i>
---------	---

Description

Compute ROC curves and AUC for each evaluation column.

Usage

```
## S3 method for class 'roc'
auc(x, cols = NULL, ...)
```

Arguments

- x roc object
- cols column names to analyze; default is all
- ... reserved arguments

Value

Updated roc object with results stored in \$auc_list and \$auc_table

Examples

```
obj <- roc(ivd_roc_example, cols = c("x1", "x2"), reference = "ref")
obj <- auc(obj)
obj <- auc(obj, cols = "x1")
```

bias.mcr	<i>Bias analysis (S3 method) – estimate bias at given medical decision levels</i>
----------	---

Description

Compute bias at specified medical decision levels (MDL) based on stored regression results: $\text{bias} = \text{candidate} - \text{predicted_reference} = x_0 - (\text{intercept} + \text{slope} * x_0) = -\text{intercept} + (1 - \text{slope}) * x_0$

Usage

```
## S3 method for class 'mcr'
bias(
  x,
  mdl,
  level = 0.95,
  interval = c("", "confidence", "prediction", "both"),
  ...
)
```

Arguments

x	mcr object (must call regression() first)
mdl	Medical decision levels (numeric vector), i.e., values of the test method
level	Confidence level, default 0.95
interval	Interval type: "" (point estimate, default), "confidence" (confidence interval), "prediction" (prediction interval), "both" (confidence + prediction)
...	Additional arguments (currently unused).

Details

Bias is the difference between the candidate method value and the estimated reference method value; a positive value indicates the candidate method is higher.

Value

Updated mcr object (invisible), results stored in \$bias

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
obj <- regression(obj, method = "ols")
bias(obj, mdl = c(200, 400, 600))
bias(obj, mdl = c(200, 400, 600), interval = "both")
```

bland_altman.mcr

*Bland-Altman analysis (S3 method)***Description**

Calculate Limits of Agreement between two measurement methods, supporting three y-axis metrics: difference, ratio, and percent difference.

Usage

```
## S3 method for class 'mcr'
bland_altman(
  x,
  type = c("difference", "ratio", "percent"),
  x_axis = c("mean", "candidate", "reference"),
  agree.level = 0.95,
  conf.level = 0.95,
  ...
)
```

Arguments

<code>x</code>	mcr object
<code>type</code>	Y-axis metric: "difference" (default), "ratio" or "percent"
<code>x_axis</code>	X-axis: "mean" (average of the two methods, default), "candidate" (test method values), "reference" (reference method values)
<code>agree.level</code>	Agreement level for limits of agreement, default 0.95 (i.e., 95% LoA)
<code>conf.level</code>	Confidence level for LoA confidence intervals, default 0.95
<code>...</code>	Additional arguments

Value

Updated mcr object (invisible), results stored in `$bland_altman`

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
bland_altman(obj)
bland_altman(obj, type = "ratio")
bland_altman(obj, type = "percent", x_axis = "reference")
```

bottle_anova	<i>bottle_anova</i> constructor
--------------	---------------------------------

Description

Create a bottle/batch ANOVA analysis object. Supports one-way and nested designs, with automatic assumption checks, missing value reporting, and ANOVA table output.

Usage

```
bottle_anova(data, formula, conf.level = 0.95)
```

Arguments

data	A data frame
formula	A formula, e.g. value ~ batch or value ~ batch/vial
conf.level	Confidence level (default 0.95)

Value

An S3 object of class "bottle_anova" with components:

call	Function call
data	Original data
formula	Input formula
conf.level	Confidence level
response_name	Response variable name
design_type	Design type ("one_way" or "nested")
rhs_vars	Right-hand side grouping variable names
term_labels	Model term labels
n_total	Total sample size
n_complete	Number of complete cases
missing_rows	Indices of excluded missing rows
cc_data	Complete-case data frame
aov_fit	aov fitted object
aov_table	ANOVA table
desc_stats	Descriptive statistics table
desc_group_col	Grouping column label
assumptions	Assumption test results (Bartlett + Shapiro-Wilk)
group_means	Group means
tukey_results	Tukey HSD results (NULL initially, filled by tukey())

Examples

```
# One-way design
df1 <- data.frame(
  bottle = rep(c("A", "B", "C"), each = 5),
  value = c(rnorm(5, 10, 1), rnorm(5, 12, 1), rnorm(5, 11, 1))
)
obj <- bottle_anova(df1, value ~ bottle)

# Nested design
df2 <- data.frame(
  lot = rep(c("L1", "L2"), each = 10),
  vial = rep(c("V1", "V2"), each = 5, times = 2),
  value = c(rnorm(5, 10, 1), rnorm(5, 10.5, 1),
            rnorm(5, 12, 1), rnorm(5, 12.5, 1))
)
obj2 <- bottle_anova(df2, value ~ lot/vial)
```

ci.precision

*S3 confidence interval method***Description**

Compute confidence intervals for each variance component (SD and %CV) based on Satterthwaite approximation. Requires `variance()` to be run first to populate the `$results` slot.

Usage

```
## S3 method for class 'precision'
ci(x, digits = 4, ...)
```

Arguments

<code>x</code>	precision object
<code>digits</code>	Number of decimal places, default 4
<code>...</code>	Reserved arguments

Value

Updated precision object with results stored in `$ci`

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
obj <- variance(obj)
obj <- ci(obj)
```

coef.fit_equation	<i>Extract coefficients from an equation fit</i>
-------------------	--

Description

Extract coefficients from an equation fit

Usage

```
## S3 method for class 'fit_equation'  
coef(object, ...)
```

Arguments

object	A fit_equation object.
...	Additional arguments (currently unused).

Value

A named numeric vector of model coefficients.

Examples

```
df <- data.frame(x = 1:5, y = c(2.1, 4.0, 6.2, 7.9, 10.1))  
f <- fit_equation("E01", df, "x", "y")  
coef(f)
```

compare_equation	<i>Fit and compare multiple equations</i>
------------------	---

Description

Fit and compare multiple equations

Usage

```
compare_equation(data, x = "x", y = "y", eqs = NULL, ...)
```

Arguments

data	a data frame
x	x column name
y	y column name
eqs	equation name/ID vector; NULL = all Response equations
...	additional arguments passed to fit_equation

Value

An object of class "compare_equation" with components:

table	data.frame sorted by AIC: Eq, Name, Formula, nPar, AIC, deltaAIC, R2, RSE
fits	list of fit_equation objects, named by eq_id
failed	character vector of eq IDs that failed to fit
data, x, y	input data reference

Examples

```
df <- data.frame(
  x = c(0.1, 0.3, 1, 3, 10, 30, 100),
  y = c(12, 25, 48, 72, 88, 96, 99)
)
compare_equation(df, "x", "y", eqs = c("E01", "E06", "E07"))
```

continuous_to_binary *Convert quantitative columns to binary based on cutoff values*

Description

Creates new binary columns (0/1) from existing quantitative columns using specified cutoff values, appending them to the original data frame. New columns are named <col>_binary.

Usage

```
continuous_to_binary(data, cols, cutoff)
```

Arguments

data	data frame
cols	character vector of column names to binarize
cutoff	numeric vector of cutoff values; recycled to length(cols) if a single value is given

Value

data frame with original columns plus new <col>_binary columns

Examples

```
df <- data.frame(x = 1:10, y = rnorm(10))
continuous_to_binary(df, "x", 5)
continuous_to_binary(df, c("x", "y"), c(5, 0))
```

correlation.mcr	<i>Correlation analysis (S3 method)</i>
-----------------	---

Description

Perform correlation analysis between the test method (X) and reference method (Y), supporting Pearson, Spearman, and Kendall methods.

Usage

```
## S3 method for class 'mcr'
correlation(x, method = c("pearson", "spearman", "kendall"), ...)
```

Arguments

x	mcr object
method	Correlation method: "pearson" (default), "spearman", or "kendall"
...	Additional arguments passed to cor.test

Value

Updated mcr object (invisible), results stored in \$correlation

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
correlation(obj)
correlation(obj, method = "spearman")
```

counts_to_table	<i>Construct a 2x2 contingency table from TP / FP / TN / FN frequencies (silent construction)</i>
-----------------	---

Description

When raw data are not available and only the four diagnostic test frequencies are known, use this function to directly construct a result object with the same structure as raw_to_table(), compatible with print() for displaying the fourfold table.

Usage

```
counts_to_table(  
  tp,  
  fp,  
  tn,  
  fn,  
  candidate = "candidate",  
  reference = "reference",  
  candidate_levels = c("positive", "negative"),  
  reference_levels = c("positive", "negative")  
)
```

Arguments

- tp True positives (candidate=level1, reference=level1)
- fp False positives (candidate=level1, reference=level2)
- tn True negatives (candidate=level2, reference=level2)
- fn False negatives (candidate=level2, reference=level1)
- candidate Candidate method name (character)
- reference Reference method name (character)
- candidate_levels Two levels of the candidate method, default c("positive", "negative")
- reference_levels Two levels of the reference method, default c("positive", "negative")

Value

A fourfold_table object (S3 class "fourfold_table") with the same structure as raw_to_table(): - \$freq_raw Long-format frequency data.frame - \$table 3x3 matrix with margins - \$n Total sample size - \$print Print slot - \$candidate_levels Levels of the candidate method - \$reference_levels Levels of the reference method Use print() directly to display the fourfold table.

Examples

```
result <- counts_to_table(tp = 45, fp = 12, tn = 80, fn = 5,  
  candidate = "new method", reference = "gold standard")
```

cutoff.roc	<i>Optimal Cutoff Analysis (S3 method)</i>
------------	--

Description

For each evaluation column, compute diagnostic metrics at all cutoff values, and identify the optimal cutoff (maximum Youden index, closest to (0,1) corner).

Usage

```
## S3 method for class 'roc'
cutoff(x, cols = NULL, ...)
```

Arguments

x	roc object
cols	column names to analyze; default is all
...	reserved arguments

Value

Updated roc object with results stored in \$cutoff_table

Examples

```
obj <- roc(ivd_roc_example, cols = c("x1", "x2"), reference = "ref")
obj <- auc(obj)
cutoff(obj)
cutoff(obj, cols = "x1")
```

describe	<i>Describe an analysis object</i>
----------	------------------------------------

Description

Describe an analysis object

Usage

```
describe(x, ...)

correlation(x, ...)

regression(x, ...)

bias(x, ...)

bland_altman(x, ...)

profile(object, ...)

normal(x, ...)

ci(x, ...)

variance(x, ...)
```

```
vc(x, ...)
diagnostics(object, ...)
kappa(x, ...)
mcnemar(x, ...)
cutoff(x, ...)
mlr(x, ...)
auc(x, ...)
```

Arguments

x	An S3 analysis object.
...	Additional arguments passed to a method.
object	An S3 analysis object.

Value

The value returned by the dispatched method.

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
describe(obj)
```

```
describe.fourfold_table
```

Descriptive Statistics: Raw Data Overview (S3 Method)

Description

Print variable description information for a fourfold_table object (from raw_to_table). If the object comes from counts_to_table (no raw data), indicates unavailability.

Usage

```
## S3 method for class 'fourfold_table'
describe(x, ...)
```

Arguments

x	a fourfold_table object
...	reserved arguments

Value

Updated fourfold_table object, with \$describe slot filled with description results

Examples

```
df <- data.frame(
  new = c("positive", "positive", "negative", "negative", "positive", "negative"),
  gold = c("positive", "negative", "positive", "negative", "positive", "positive"),
  age = c(45, 52, 38, 61, 47, 55),
  gender = c("M", "F", "F", "M", "M", "F"),
  id = c("S001", "S002", "S003", "S004", "S005", "S006")
)
tab <- raw_to_table(df, "new", "gold", id = "id")
tab <- describe(tab)
```

describe.mcr	Compute descriptive statistics and store results
--------------	--

Description

Computes data overview (rows/columns/ID duplicates), per-column summaries, and a side-by-side comparison table of candidate, reference, and Diff. Results are stored in x\$describe (class mcr_describe) and displayed via print.mcr_describe() (e.g., when calling summary()).

Usage

```
## S3 method for class 'mcr'
describe(x, cols = NULL, digits = 4L, ...)
```

Arguments

x	mcr object
cols	Column name vector to analyze; by default, analyzes all columns except id, candidate, and reference. Supports exclusion with - prefix, e.g., cols = -c("age", "sex").
digits	Number of decimal places, default 4
...	Additional arguments

Value

Updated mcr object (invisible), results stored in x\$describe

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
describe(obj)
```

describe.roc	<i>Descriptive Statistics (S3 method)</i>
--------------	---

Description

Print distribution summaries for evaluation columns and the reference column (if numeric). When the reference is binary, evaluation column distributions are shown grouped by positive/negative.

Usage

```
## S3 method for class 'roc'
describe(x, cols = NULL, digits = 4L, ...)
```

Arguments

x	roc object
cols	columns to describe; defaults to all evaluation columns + reference column (if numeric)
digits	number of decimal places, default 4
...	reserved arguments

Value

Updated roc object with results stored in \$describe

Examples

```
obj <- roc(ivd_roc_example, cols = "x1", reference = "ref")
describe(obj)
```

diagnostics.fourfold_table

Diagnostic performance metrics: calculate sensitivity, specificity, likelihood ratios, etc. with confidence intervals for a fourfold_table

Description

Diagnostic performance metrics: calculate sensitivity, specificity, likelihood ratios, etc. with confidence intervals for a fourfold_table

Usage

```
## S3 method for class 'fourfold_table'
diagnostics(
  object,
  conf.level = 0.95,
  ci.method = "wilson",
  prevalence = NULL,
  ...
)
```

Arguments

object	a fourfold_table object
conf.level	Confidence level, default 0.95
ci.method	Proportion confidence interval method, default "wilson". Supports "wald", "wald-cc", "wilson", "wilson-cc", "agresti-coull", "jeffreys", "clopper-pearson"
prevalence	User-specified prevalence. If provided, PPV/NPV will be calculated based on this prevalence via Bayes' theorem (rather than the observed prevalence in the data).
...	Additional arguments passed to print

Value

Updated fourfold_table object, with diagnostic performance metrics stored in \$diagnostics: - \$tp / \$fp / \$fn / \$tn / \$n Four-fold frequencies and total sample size - \$sensitivity Sensitivity with CI - \$specificity Specificity with CI - \$ppv Positive predictive value with CI - \$npv Negative predictive value with CI - \$accuracy Accuracy with CI - \$prevalence Prevalence with CI - \$lr_positive Positive likelihood ratio with CI - \$lr_negative Negative likelihood ratio with CI - \$odds_ratio Odds ratio with CI - \$conf_level Confidence level used - \$ci_method Method name used - \$prev_specified Whether user specified prevalence

Examples

```
tab <- raw_to_table(ivd_qualitative_example, "new", "gold",
  positive = "positive", id = "id")
tab <- diagnostics(tab)
tab <- diagnostics(tab, prevalence = 0.3)
tab <- diagnostics(tab, conf.level = 0.90)
tab <- diagnostics(tab, ci.method = "clopper-pearson")
```

fit_equation	General-purpose equation fitting
--------------	----------------------------------

Description

General-purpose equation fitting

Usage

```
fit_equation(
  eq,
  data,
  x = "x",
  y = "y",
  start = NULL,
  df_col = NULL,
  lower = NULL,
  upper = NULL,
  constraints = NULL,
  weights = NULL,
  ...
)
```

Arguments

eq	equation identifier: ID("E07"), name("4PLC"), category("sadler"), or a custom formula string ("y ~ aexp(-bx)")
data	a data frame
x	x variable column name (default "x")
y	y variable column name (default "y")
start	named list of starting values (NULL=auto)
df_col	df column name for Sadler models (NULL=default 50)
lower	named list/vector of lower bounds (e.g. list(Vmax=0, Km=0))
upper	named list/vector of upper bounds
constraints	list of constraint functions: list(ineq = f, eq = g) f(par) returns a vector; all values must be non-positive g(par) returns a vector; must satisfy all(g(par) = 0)
weights	weights: NULL(equal) / numeric vector / method name string methods: "equal", "1/y", "1/y^2", "1/x", "1/x^2"
...	additional arguments passed to nlsLM() / fit_vfp() / nloptr()

Value

an S3 object of class "fit_equation"

Examples

```
# ==== Response curve fitting ====

# 1) Linear
df <- data.frame(x = 1:5, y = c(2.1, 4.0, 6.2, 7.9, 10.1))
f1 <- fit_equation("E01", df, "x", "y")
print(f1)

# 2) Logarithmic
```

```

f2 <- fit_equation("E06", df, "x", "y")
print(f2)

# 3) 4PLC (dose-response, descending)
df4plc <- data.frame(
  conc = c(0.1, 0.3, 1, 3, 10, 30, 100),
  resp = c(99, 96, 88, 72, 48, 25, 12)
)
f3 <- fit_equation("E07", df4plc, "conc", "resp")
print(f3)

# 4) Custom formula
f4 <- fit_equation("a + b * log(x)", df4plc, "conc", "resp")
print(f4)

# 5) Box constraints (parameter bounds)
f5 <- fit_equation("E07", df4plc, "conc", "resp",
  lower = list(D = 0), upper = list(D = 5))
print(f5)

# 6) Inequality constraints (parameter relationship)

# require D > 1 (Hill slope cannot be too shallow)
f6 <- fit_equation("E07", df4plc, "conc", "resp",
  constraints = list(ineq = function(p) 1 - p["D"]))
print(f6)

# 7) Equality constraints (fixed parameter relationship)

# fix b = 0.5 in exponential decay y = a*exp(-b*x)
df_exp <- data.frame(x = 0:5, y = c(10, 6.1, 3.7, 2.2, 1.4, 0.8))
f7 <- fit_equation("E04", df_exp, "x", "y",
  constraints = list(eq = function(p) p["b"] - 0.5))
print(f7)

# 8) Weighted fitting via replicate_to_mean

# data with replicate measurements (8 dose levels, 3 replicates each)
df_rep <- data.frame(
  dose = rep(c(0.3, 0.6, 1.5, 4, 10, 25, 60, 150), each = 3),
  resp = c(4.1, 3.8, 3.9, 8.5, 8.9, 8.6,
    18.2, 17.9, 18.5,
    34.6, 35.1, 34.8,
    54.3, 53.9, 54.7,
    73.0, 73.6, 73.2,
    87.5, 87.1, 87.8,
    97.9, 98.3, 98.0)
)
# first summarize, generate inverse-variance weights
rep <- replicate_to_mean(df_rep, "dose", "resp", weights = "1/sd^2")

```

```

# fit 4PLC with summarized means and explicit start values
f8 <- fit_equation("E07", rep, x = "x", y = "y_mean",
  weights = rep$weight,
  start = list(A = 2, B = 100, C = 8, D = -1.5))
print(f8)

# ==== Sadler variance function fitting ====

df_var <- data.frame(
  conc = c(1, 2, 5, 10, 20, 50),
  var = c(0.1, 0.3, 0.8, 3.2, 12.5, 78.4)
)

# 9) Single Sadler model
f9 <- fit_equation("V03", df_var, "conc", "var")
print(f9)

# 10) All 10 Sadler models
f10 <- fit_equation("sadler", df_var, "conc", "var")
print(f10)

```

ivd_bottle_example	<i>Small deterministic one-way ANOVA example</i>
--------------------	--

Description

Small deterministic one-way ANOVA example

Usage

```
ivd_bottle_example
```

Format

A data frame with 12 observations from three bottles.

ivd_mcr_example	<i>Small deterministic method-comparison example</i>
-----------------	--

Description

Small deterministic method-comparison example

Usage

```
ivd_mcr_example
```

Format

A data frame with 12 rows and columns sid, test, and ref.

ivd_qualitative_example	<i>Small deterministic qualitative-method example</i>
-------------------------	---

Description

Small deterministic qualitative-method example

Usage

ivd_qualitative_example

Format

A data frame with paired binary results and sample identifiers.

ivd_roc_example	<i>Small deterministic ROC example</i>
-----------------	--

Description

Small deterministic ROC example

Usage

ivd_roc_example

Format

A data frame with 12 rows, two markers, and a binary reference.

kappa.fourfold_table *Cohen's Kappa / PABAK Agreement Analysis (2x2 table only)*

Description

Calculate Cohen's Kappa coefficient or PABAK (Prevalence-Adjusted Bias-Adjusted Kappa) and its confidence interval, used to evaluate agreement between two binary classification methods.

Usage

```
## S3 method for class 'fourfold_table'
kappa(x, prevalence = NULL, conf.level = 0.95, ...)
```

Arguments

x	a fourfold_table object
prevalence	Patient prevalence. If provided, calculates PABAK instead of Cohen's Kappa.
conf.level	Confidence level, default 0.95
...	Additional arguments (currently unused).

Details

When data prevalence is extreme, Cohen's Kappa may be low even when observed agreement is high. In such cases, specify the prevalence parameter to compute PABAK as a correction. PABAK = $2 \times p_{\text{obs}} - 1$, which assumes expected agreement = 0.5.

Value

Updated fourfold_table object, with Kappa results stored in \$kappa: - \$kappa Kappa / PABAK coefficient - \$se Standard error - \$lower / \$upper Lower/upper confidence interval bounds - \$p_obs Observed agreement - \$p_exp Expected agreement - \$n Total sample size - \$conf_level Confidence level used - \$method Method name used

Examples

```
tab <- counts_to_table(tp = 85, fp = 3, tn = 90, fn = 2,
                       candidate = "new method", reference = "gold standard")
tab <- kappa(tab)                # Cohen's Kappa
tab <- kappa(tab, prevalence = 0.5) # PABAK
```

list_equation	<i>Print the equation registry</i>
---------------	------------------------------------

Description

Print the equation registry

Usage

```
list_equation(category = NULL, engine = NULL)
```

Arguments

category	filter category: "Response", "Sadler", or NULL (all)
engine	filter engine: "lm", "nls", "vfp", or NULL (all)

Value

invisible data.frame (matching rows), prints a formatted table to the console

Examples

```
# All equations
list_equation()

# Response only
list_equation(category = "Response")

# lm engine only
list_equation(engine = "lm")
```

list_sadler	<i>List Sadler precision profile model equations</i>
-------------	--

Description

List Sadler precision profile model equations

Usage

```
list_sadler()
```

Arguments

This function has no arguments.

Details

List the 10 candidate Sadler precision profile model formulas and types used in the VFP package. Includes: Constant SD, Constant CV, Linear (variance), Power model, Exponential model, etc.

Value

Invisibly returns NULL. The model table is printed to the console as a side effect.

Examples

```
list_sadler()
```

list_westgard	<i>List Westgard QC rules</i>
---------------	-------------------------------

Description

Prints commonly used Westgard multi-rule QC rules and their meanings. Rule names printed here can be passed to `qc_chart(rules = "...")`.

Usage

```
list_westgard(brief = FALSE)
```

Arguments

brief Set to TRUE to suppress printing and return only the named character vector.

Value

A named character vector of rule descriptions (invisibly).

Examples

```
list_westgard()
```

lob_lod_loq

*Limit of blank, detection, and quantitation (LoB / LoD / LoQ)***Description**

Estimates analytical sensitivity limits from a data.frame of *summarized* measurements (one row per sample: a sample-name column, a mean column, an SD column, and a replicate-count column). The workflow follows CLSI EP17-A2:

Usage

```
lob_lod_loq(
  data,
  sample_col = "x",
  mean_col = "y_mean",
  sd_col = "y_sd",
  n_col = "y_n",
  blank = NULL,
  target_cv = 0.2,
  na.rm = TRUE,
  ...
)
```

Arguments

data	a data.frame of summarized measurements with a sample name column, a mean column, an SD column, and an n (replicate count) column. Column names are free-form; specify them via sample_col, mean_col, sd_col, and n_col. The default column names match the output of replicate_to_mean().
sample_col	name of the column identifying each sample (default "x").
mean_col	name of the column holding each sample's mean of replicates (default "y_mean"). This is used as the concentration axis of the precision profile.
sd_col	name of the column holding each sample's SD (default "y_sd").
n_col	name of the column holding each sample's number of replicates (default "y_n").
blank	sample name(s) in sample_col that are blank samples (default NULL). When NULL, only the LoQ is computed. When provided, the LoB (and then LoD) are computed from the pooled blank rows.
target_cv	target coefficient of variation for the LoQ (default 0.20, i.e. 20%).
na.rm	remove rows with NA in any of the used columns? (default TRUE).
...	reserved for future use.

Details

- **LoB** (Section 5.3.3.1, parametric option): $\text{LoB} = M_B + cp * SD_B$, pooling all blank samples, with $cp = 1.645 / \sqrt{1 - 1/(4*(B - K))}$.
- **LoD** (Section 5.4.3): a fixed-point iteration on the fitted precision profile, $X = \text{LoB} + cp * SD_{WL}(X)$, solved for the smallest $X \geq \text{LoB}$.
- **LoQ** (Appendix D1, Example 1 — functional sensitivity): the concentration X where the precision profile meets the target CV, i.e. $\sqrt{\text{Var}(X)} / X = \text{target_cv}$.

The precision profile is fit internally with `fit_equation("sadler", ...)` (all 10 Sadler variance models; the best model by AIC is used for prediction).

Note that the LoQ reported here reflects **precision only**; bias is not included. If the precision-based LoQ is below the LoD, the LoQ is reported as $\max(\text{LoQ}, \text{LoD})$ with a warning.

Value

An object of class "sensitivity" (printed and plottable) with components:

lob	LoB (parametric, pooled blanks) or NA when no blanks are given or the blank degrees of freedom are insufficient.
lod	LoD (fixed point on the precision profile) or NA when no fixed point exists in the search range.
loq	LoQ (functional sensitivity at target_cv), already constrained to be $\geq \text{lod}$ when both are finite.
n_blank, B, K	blank result count (B) and blank sample count (K).
cp_lob, cp_lod	the EP17 small-sample corrections for the LoB and LoD multipliers.
target_cv	the requested CV target.
fit	the internal <code>fit_equation("sadler", ...)</code> result, or NULL.
model	best Sadler model name (by AIC), or NA.
notes	character vector of any conditions encountered.

A warning is issued whenever a limit cannot be estimated (components set to NA).

Examples

```
# summarized data: two blank samples + six low levels
summ <- data.frame(
  sample = c("blank_A", "blank_B", "L1", "L2", "L3", "L4", "L5", "L6"),
  mean   = c(0, 0, 0.5, 1, 2, 5, 10, 20),
  sd     = 0.05 + 0.10 * c(0, 0, 0.5, 1, 2, 5, 10, 20),
  n      = c(15L, 15L, 5L, 5L, 5L, 5L, 5L, 5L)
)

# 1. blank + low levels: LoB / LoD / LoQ
lob_lod_loq(summ, "sample", "mean", "sd", "n",
  blank = c("blank_A", "blank_B"))

# 2. blanks only: LoB
```

```
lob_lod_loq(summ[1:2, ], "sample", "mean", "sd", "n",
            blank = c("blank_A", "blank_B"))

# 3. no blanks: LoQ only
lob_lod_loq(summ[3:8, ], "sample", "mean", "sd", "n")
```

```
mcnemar.fourfold_table
```

McNemar Test (for fourfold_table only)

Description

Perform McNemar's test on a 2x2 paired contingency table to evaluate whether there is a systematic difference between two binary classification methods (i.e., whether marginal probabilities are equal).

Usage

```
## S3 method for class 'fourfold_table'
mcnemar(x, conf.level = 0.95, ...)
```

Arguments

<code>x</code>	a fourfold_table object
<code>conf.level</code>	Confidence level, default 0.95
<code>...</code>	Additional arguments (currently unused).

Details

When discordant pairs ($b + c$) are few (< 25), automatically uses the exact binomial test (`binom.test`); otherwise uses McNemar's chi-squared test with continuity correction.

Value

Updated fourfold_table object, with McNemar results stored in `$mcnemar`: - `$chi_sq` McNemar chi-squared statistic (NA for exact test) - `$df` Degrees of freedom (NA for exact test) - `$p_value` Test p-value - `$method` Test method used - `$b` Discordant pairs (positive, negative) count - `$c` Discordant pairs (negative, positive) count - `$n_bc` Total discordant pairs - `$conf.level` Confidence level used

Examples

```
tab <- counts_to_table(tp = 45, fp = 12, tn = 80, fn = 5,
                      candidate = "new method", reference = "gold standard")
tab <- mcnemar(tab)
```

mcr	<i>mcr constructor</i>
-----	------------------------

Description

Create a method comparison regression (mcr) object containing sample IDs, test method, and reference method data.

Usage

```
mcr(data, id, candidate, reference, weights = NULL)
```

Arguments

data	Data frame with id, candidate, reference columns
id	ID variable name (character), used to identify samples
candidate	Test method variable name (character, method under evaluation)
reference	Reference method variable name (character, standard method)
weights	Optional column name (character) in data containing non-negative finite numeric weights for weighted regression. Default NULL (no weighting).

Value

Returns an S3 "mcr" object containing:

call	Original function call
data	Complete data frame
id / id_name	ID vector and column name
candidate / reference	Test/reference method numeric vectors (complete pairs only)
candidate_name / reference_name	Method column names
n	Number of complete pairs
complete_idx	Row indices of complete pairs in the original data

and analysis result storage slots: \$correlation, \$regression, \$outlier, \$bland_altman

Examples

```
df <- data.frame(sid = 1:30,
                 test = rnorm(30, 50, 10),
                 ref = rnorm(30, 50, 10))
obj <- mcr(df, "sid", "test", "ref")
```

mkt	<i>mkt — Mean Kinetic Temperature</i>
-----	---------------------------------------

Description

Calculates the mean kinetic temperature from one or more temperature probes. Supports equal-interval and time-weighted trapezoidal methods.

Usage

```
mkt(data, temp_cols, time = NULL, temp_unit = "C", ea = 83.144)
```

Arguments

data	A data frame.
temp_cols	One or more column names for temperature probes.
time	Optional time column; accepts numeric, Date, or POSIXct.
temp_unit	Temperature unit: "C" or "K".
ea	Activation energy in kJ/mol; default 83.144.

Value

An S3 object of class "mkt".

Examples

```
temp_df <- data.frame(t1 = c(25,26,27), t2 = c(24,25,26), time = 1:3)
mkt(temp_df, c("t1","t2"), "time")
```

mlr.roc	<i>Multivariate Logistic Regression (S3 method)</i>
---------	---

Description

Fit logistic regression using selected evaluation columns, compute AUC of predicted probabilities. Names must be unique; auto-generated as MLR01, MLR02 ... when not specified.

Usage

```
## S3 method for class 'roc'
mlr(x, cols = NULL, name = NULL, ...)
```

Arguments

x	roc object
cols	column names to fit; default is all
name	fit name (must be unique); auto-generated by default
...	additional arguments passed to glm

Value

Updated roc object with results stored in \$mlr_results[[name](#)]

Examples

```
obj <- roc(ivd_roc_example, cols = c("x1", "x2"), reference = "ref")
obj <- auc(obj)
obj <- mlr(obj)           # -> MLR01
obj <- mlr(obj)           # -> MLR02
obj <- mlr(obj, name = "my_model") # -> my_model
```

normal.precision	<i>S3 normality test method</i>
------------------	---------------------------------

Description

Perform normality tests on the response values for each sample, underlying call to `normal_test()`. Supports automatic test selection (Shapiro-Wilk, Anderson-Darling, Lilliefors, CVM).

Usage

```
## S3 method for class 'precision'
normal(
  x,
  method = c("auto", "shapiro", "sw", "ad", "lillie", "cvm"),
  level = 0.95,
  ...
)
```

Arguments

x	precision object
method	Test method: "auto" (default, automatic selection), "shapiro", "sw", "ad", "lillie", "cvm"
level	Confidence level, default 0.95
...	Reserved arguments

Value

Updated precision object with results stored in \$normal

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
obj <- normal(obj)
```

normal_test

Normality test

Description

Performs normality tests on a single numeric column of a data frame. Plots are generated separately via `plot()`.

Usage

```
normal_test(
  data,
  col,
  method = c("auto", "shapiro", "sw", "ad", "lillie", "cvm"),
  level = 0.95
)
```

Arguments

data	A data frame.
col	Column name to test (single string).
method	Test method: "auto" (default, chooses by sample size), "shapiro"/"sw", "ad", "lillie", "cvm".
level	Confidence level (default 0.95, used for QQ confidence bands).

Value

An S3 object of class "normal_test" containing:

call Function call.
 data_name Data object name.
 col Column name.
 method User-specified method.
 level Confidence level.
 n_total Total number of rows.
 n Number of finite values (non-missing).
 missing Number of missing values.
 test_method Actual test method used.
 statistic Test statistic.
 p_value p-value.
 y Clean numeric vector, used by `plot()`.

Examples

```
df <- data.frame(x = rnorm(50))
normal_test(df, "x")
normal_test(df, "x", method = "ad")
plot(normal_test(df, "x"))
plot(normal_test(df, "x"), type = "his")
plot(normal_test(df, "x"), type = c("qq", "his"))
```

outlier

Detect outliers in an analysis object

Description

Detect outliers in an analysis object

Usage

```
outlier(x, ...)
```

Arguments

x An S3 analysis object.

... Additional arguments passed to a method.

Value

The value returned by the dispatched method.

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
outlier(obj, method = "grubbs")
```

outlier.mcr

Outlier detection (S3 method)

Description

Perform outlier detection based on the difference or percent difference between two measurements. Reuses four methods from ../precision/outliers.R.

Usage

```
## S3 method for class 'mcr'
outlier(
  x,
  method = c("grubbs", "esd", "dixon", "iqr"),
  type = c("difference", "percent"),
  alpha = 0.05,
  coef = 1.5,
  ...
)
```

Arguments

x	mcr object
method	Outlier detection method: "grubbs" (default), "esd", "dixon", "iqr"
type	Data type to compute: "difference" (default) or "percent" (percent difference)
alpha	Significance level, default 0.05 (Grubbs/ESD/Dixon only)
coef	IQR multiplier, default 1.5 (IQR only)
...	Additional arguments passed to esd_test / dixon_test

Value

Updated mcr object (invisible), results stored in \$outlier

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
outlier(obj, method = "grubbs")
outlier(obj, method = "iqr", type = "percent")
```

outlier.precision	<i>S3 outlier detection method</i>
-------------------	------------------------------------

Description

Detect outliers in the response values for each sample, supporting Grubbs test and IQR method. Underlying call to outliers_test() (from outliers-and-normal.R).

Usage

```
## S3 method for class 'precision'
outlier(x, alpha = 0.05, method = c("grubbs", "iqr"), ...)
```

Arguments

x	precision object
alpha	Significance level, default 0.05 (Grubbs only)
method	Detection method: "grubbs" (default) or "iqr"
...	Additional arguments passed to outliers_test()

Value

Updated precision object with results stored in \$outlier

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
obj <- outlier(obj, method = "iqr")
```

outliers_test

Outlier detection

Description

Integrates Grubbs test, generalized ESD test, Dixon Q test, and IQR method under a consistent API with an S3 print method.

Usage

```
outliers_test(data, col, method = c("grubbs", "esd", "dixon", "iqr"), ...)
```

Arguments

data	A data frame.
col	Column name to test (single string).
method	Detection method: "grubbs" (default), "esd", "dixon", "iqr".
...	Additional arguments passed to internal methods:
alpha	Significance level for grubbs / esd / dixon (default 0.05). dixon only accepts 0.01 and 0.05.
r	Maximum number of outliers for ESD (default min(5, n-2)).
type	Tail to test for Dixon: "both" (default), "min", "max".
coef	IQR multiplier (default 1.5).

Value

An S3 object of class "outliers_test" containing:

method Method name used.

data_name Name of the input data.

col Column name tested.

n_total Total number of rows.

n Number of finite observations used.

missing Number of missing (non-finite) values.

parameters List of method parameters.

n_outliers Number of outliers detected.

indices Indices of outliers in the original vector (1-based).

values Outlier values.

details Method-specific details (statistics, critical values, bounds, etc.).

Examples

```
set.seed(123)
df <- data.frame(x = c(rnorm(20), 10, -8))
outliers_test(df, "x", "grubbs")
outliers_test(df, "x", "esd", r = 3)
outliers_test(df, "x", "dixon")
outliers_test(df, "x", "iqr", coef = 2)
```

plot.fit_equation	<i>Plot an equation fit</i>
-------------------	-----------------------------

Description

Plot an equation fit

Usage

```
## S3 method for class 'fit_equation'
plot(x, interval = "confidence", level = 0.95, frame = FALSE, ...)
```

Arguments

x	A fit_equation object.
interval	Interval type, either "confidence" or "prediction".
level	Confidence level.
frame	Whether to draw a frame.
...	Additional arguments (currently unused).

Value

x, invisibly.

Examples

```
df4plc <- data.frame(
  conc = c(0.1, 0.3, 1, 3, 10, 30, 100),
  resp = c(12, 25, 48, 72, 88, 96, 99)
)
f <- fit_equation("E07", df4plc, "conc", "resp")
plot(f)
plot(f, interval = "prediction", frame = TRUE)
```

plot.mcr

*Plot (S3 method) – unified plotting interface***Description**

Draw one of four plot types for an mcr object.

Usage

```
## S3 method for class 'mcr'
plot(
  x,
  type = c("scatter", "regression", "bland_altman", "bias"),
  interval = c("none", "confidence", "prediction", "both"),
  conf.level = NULL,
  mdl = NULL,
  ...
)
```

Arguments

x	mcr object
type	Plot type: "scatter" (scatter + identity line, default), "regression" (regression line + confidence/prediction bands), "bland_altman" (Bland-Altman plot), "bias" (bias trend plot)
interval	Interval type (only for type="regression"): "none" (default), "confidence", "prediction", "both"
conf.level	Confidence level; if NULL, reads from analysis slot, otherwise uses this value
mdl	Medical decision levels (only for type="bias"); if NULL, tries to read from x\$bias\$mdl
...	Additional arguments

Value

Invisible ggplot object

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
plot(obj)
plot(obj, type = "scatter")
obj <- regression(obj)
plot(obj, type = "regression", interval = "confidence")
obj <- bland_altman(obj)
plot(obj, type = "bland_altman")
obj <- bias(obj, mdl = c(200, 400))
plot(obj, type = "bias")
```

plot.precision	<i>S3 plot method</i>
----------------	-----------------------

Description

Draw one of five plot types for a precision object. dot (run-order scatter plot) and his (histogram + $N(0,1)$ density) require no prior analysis; var (variance component bar chart) requires variance() to be run first; qq (normal Q-Q plot) requires normal() first; profile (precision profile) requires profile() first.

Usage

```
## S3 method for class 'precision'
plot(x, type = c("dot", "his", "var", "qq", "profile"), ...)
```

Arguments

x	precision object
type	Plot type: "dot" (run-order scatter plot, default preferred), "his" (histogram), "var" (variance component plot), "qq" (normal Q-Q plot), "profile" (precision profile)
...	Reserved arguments

Value

Invisible ggplot object

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
plot(obj, type = "dot")
plot(obj, type = "his")
```

plot.reference_interval

Plot reference interval results

Description

One panel per method: jitter strip chart overlaid with reference interval limits, endpoints, and CI error bars.

Usage

```
## S3 method for class 'reference_interval'
plot(x, ...)
```

Arguments

x	A "reference_interval" object.
...	Additional arguments (reserved for generic).

Value

A ggplot object, invisibly. The plot is also drawn as a side effect.

plot.roc

Plot (S3 method) – ROC Curve

Description

Draw ROC curves for evaluation columns and/or MLR model predictions. Optionally mark optimal cutoff points (requires `cutoff.roc()` first).

Usage

```
## S3 method for class 'roc'
plot(x, cols = NULL, mlr = NULL, cutpoint = NULL, ...)
```

Arguments

x	roc object
cols	evaluation column names to plot; NULL (default) plots all, c() (empty) plots none.
mlr	MLR model name(s) to overlay; NULL (default) plots none, "all" plots all stored MLR models.
cutpoint	mark optimal cutoff points: NULL (default, none), "Youden", "Corner", or "all".
...	additional arguments (reserved)

Value

invisible ggplot object

Examples

```
obj <- roc(ivd_roc_example, cols = c("x1", "x2"), reference = "ref")
obj <- auc(obj)
plot(obj)
plot(obj, cols = "x1")
obj <- auc(obj); obj <- cutoff(obj); obj <- mlr(obj)
plot(obj, mlr = "all", cutpoint = "Youden")
```

precision

Precision variance component analysis

Description

For each sample (grouped by by), estimate variance components via VCA: :anovaVCA, compute SD / CV and their Satterthwaite confidence intervals.

Usage

```
precision(data, form, by = NULL, NegVC = FALSE, ...)
```

Arguments

data	Data frame containing the response variable and design factor columns.
form	Formula parsed by VCA (e.g. <code>y ~ day/run/rep</code>).
by	Column name(s) for sample grouping, supports multiple columns.
NegVC	Allow negative variance components? Default FALSE.
...	Additional arguments passed to VCA: :anovaVCA.

Value

Returns an object of class precision. Use [summary](#), [profile](#), etc. to view results.

Examples

```
data(VCAdata1, package = "VCA")
precision(VCAdata1, y ~ day/run, by = "sample")

# More complex: deeply nested factors with more sample groups
data(realData, package = "VCA")
precision(realData, y ~ lot/calibration/day/run, by = "PID")
```

predict.fit_equation *Predict fitted values or inverse-predict (from y to x)*

Description

Predict fitted values or inverse-predict (from y to x)

Usage

```
## S3 method for class 'fit_equation'
predict(
  object,
  newdata = NULL,
  x = NULL,
  y = NULL,
  inverse = FALSE,
  interval = NULL,
  level = 0.95,
  ...
)
```

Arguments

object	a fit_equation object
newdata	new data as a data.frame
x	x column name; NULL uses the original x column from the fit
y	y column name (inverse only); NULL uses the original y column
inverse	logical; if TRUE, solve for x given y values in newdata
interval	"confidence" or "prediction"
level	confidence level (default 0.95)
...	additional arguments passed to the underlying predict()

Value

data.frame with x, y, and (if requested) confidence/prediction interval columns

Examples

```
df4plc <- data.frame(
  conc = c(0.1, 0.3, 1, 3, 10, 30, 100),
  resp = c(12, 25, 48, 72, 88, 96, 99)
)
f <- fit_equation("E07", df4plc, "conc", "resp")

# Fitted values
predict(f)
```

```
# Confidence interval
predict(f, interval = "confidence")

# Prediction interval
predict(f, interval = "prediction")

# Inverse prediction: estimate x for y=50
predict(f, newdata = data.frame(resp = 50), inverse = TRUE)
```

predict.mcr

Predict (S3 method) – predict based on regression results

Description

Predict values based on stored regression results.

Usage

```
## S3 method for class 'mcr'
predict(
  object,
  newdata = NULL,
  candidate = NULL,
  reference = NULL,
  inverse = FALSE,
  interval = c("none", "confidence", "prediction", "both"),
  conf.level = 0.95,
  ...
)
```

Arguments

object	mcr object (must call regression() first)
newdata	data.frame containing predictor variables. If provided, candidate and reference are treated as column names in newdata.
candidate	Predictor variable. When newdata is not NULL, it is a column name (character) in newdata; when newdata is NULL, it is a numeric vector; if also NULL, uses original data.
reference	Predictor for inverse prediction (only used when inverse=TRUE). Usage same as candidate.
inverse	Whether to perform inverse prediction (reference to candidate), default FALSE.
interval	Interval type: "none" (default), "confidence", "prediction", "both". Only forward prediction (inverse=FALSE) supports intervals.
conf.level	Confidence level, default 0.95
...	Additional arguments

Value

data.frame containing predicted values and (if requested) interval bounds

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
obj <- regression(obj, method = "ols")
predict(obj, candidate = c(40, 50, 60))
predict(obj, candidate = c(40, 50, 60), interval = "both")
predict(obj, newdata = data.frame(xx = c(40, 50)), candidate = "xx")
predict(obj, inverse = TRUE) # Inverse prediction on original data
```

predict.roc

Predict (S3 method) – predict from multivariate logistic regression

Description

Predict positive-class probability for new data using a stored MLR model. Returns the input predictor columns together with predicted probability, standard error, confidence interval, and 0/1 classification.

Usage

```
## S3 method for class 'roc'
predict(object, mlr = NULL, newdata = NULL, column_map = NULL, ...)
```

Arguments

object	roc object (must run mlr() first)
mlr	MLR model name (character). If only one MLR exists, defaults to it; if multiple exist, must specify.
newdata	data.frame of new observations. If NULL (default), uses the training data (complete cases) from the roc object.
column_map	Named character vector mapping newdata column names to training column names, e.g. c(x1_new = "x1", x2_new = "x2"). If NULL, tries to match by name automatically.
...	Additional arguments passed to stats::predict.glm

Value

data.frame containing the predictor columns used in the model, plus:

pred_prob	predicted probability of the positive class
pred_se	standard error of the predicted probability
ci_lower	lower bound of the confidence interval (probability scale)
ci_upper	upper bound of the confidence interval (probability scale)
pred_class	binary prediction (1 if pred_prob >= 0.5, 0 otherwise)

Examples

```
df <- data.frame(sid = 1:100,
                 x1 = c(rnorm(50, 10, 2), rnorm(50, 12, 2)),
                 x2 = rnorm(100, 5, 1),
                 ref = rep(c(0, 1), each = 50))
obj <- roc(df, cols = c("x1", "x2"), reference = "ref")
obj <- auc(obj)
obj <- mlr(obj)
predict(obj)
predict(obj, newdata = df[1:5, ])
## column_map example: rename x1 to new_x1 in newdata
new_df <- df[1:5, c("x2", "ref")]
new_df$new_x1 <- df$x1[1:5]
predict(obj, newdata = new_df,
        column_map = c(new_x1 = "x1", x2 = "x2"))
```

```
print.bottle_anova      S3 print method: bottle_anova
```

Description

Print bottle ANOVA analysis results including descriptive statistics, ANOVA table, and assumption checks.

Usage

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

Arguments

x	A bottle_anova object
...	Other arguments (unused, for S3 generic signature)

Value

Invisibly returns x.

```
print.describe_table    S3 print method for describe_table
```

Description

S3 print method for describe_table

Usage

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

Arguments

```
x            describe_table object
...          reserved arguments
```

Value

Invisible describe_table object

Examples

```
df <- data.frame(
  new   = c("positive", "positive", "negative", "negative", "positive", "negative"),
  gold  = c("positive", "negative", "positive", "negative", "positive", "positive"),
  age   = c(45, 52, 38, 61, 47, 55),
  gender = c("M", "F", "F", "M", "M", "F"),
  id    = c("S001", "S002", "S003", "S004", "S005", "S006")
)
tab <- raw_to_table(df, "new", "gold", id = "id")
tab <- describe(tab)
```

```
print.diagnostics    S3 print method: format diagnostics output (diagnostic performance metrics)
```

Description

Prints four-fold table frequencies, sensitivity, specificity, PPV, NPV, accuracy, likelihood ratios, etc.

Usage

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

Arguments

x	diagnostics object
digits	Number of decimal places, default 3
...	reserved arguments

Value

Invisible diagnostics object

Examples

```
df <- data.frame(
  new    = c("positive", "positive", "negative", "negative", "positive", "negative"),
  gold   = c("positive", "negative", "positive", "negative", "positive", "positive"),
  age    = c(45, 52, 38, 61, 47, 55),
  gender = c("M", "F", "F", "M", "M", "F"),
  id     = c("S001", "S002", "S003", "S004", "S005", "S006")
)
tab <- raw_to_table(df, "new", "gold", positive = "positive", id = "id")
tab <- diagnostics(tab)
```

print.fit_equation	<i>Print an equation fit</i>
--------------------	------------------------------

Description

Print an equation fit

Usage

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

Arguments

x	A fit_equation object.
...	Additional arguments (currently unused).

Value

x, invisibly.

```
print.fourfold_table
```

S3 print method: format a fourfold_table object as a fourfold table

Description

Prints data source, total sample size, variable names, fourfold table (with margins), and analysis status.

Usage

```
## S3 method for class 'fourfold_table'
print(x, margin = TRUE, ...)
```

Arguments

x	fourfold_table object (result from raw_to_table / counts_to_table)
margin	Whether to display margins, default TRUE
...	reserved arguments

Value

Invisible fourfold_table object

Examples

```
df <- data.frame(
  new = c("positive", "positive", "negative", "negative", "positive", "negative"),
  gold = c("positive", "negative", "positive", "negative", "positive", "positive"),
  age = c(45, 52, 38, 61, 47, 55),
  gender = c("M", "F", "F", "M", "M", "F"),
  id = c("S001", "S002", "S003", "S004", "S005", "S006")
)
tab <- raw_to_table(df, "new", "gold", id = "id")
print(tab)
```

```
print.kappa_table
```

S3 print method: format kappa_table output

Description

Prints Cohen's Kappa / PABAK coefficient, standard error, confidence interval, and agreement rates.

Usage

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

Arguments

<code>x</code>	kappa_table object
<code>digits</code>	Number of decimal places, default 4
<code>...</code>	reserved arguments

Value

Invisible kappa_table object

Examples

```
tab <- counts_to_table(tp = 85, fp = 3, tn = 90, fn = 2,
                      candidate = "new method", reference = "gold standard")
tab <- kappa(tab)
```

`print.mcnemar_table` *S3 print method: format mcnemar_table output*

Description

Prints McNemar test results, including discordant pair counts, test statistic, and p-value.

Usage

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

Arguments

<code>x</code>	mcnemar_table object
<code>...</code>	reserved arguments

Value

Invisible mcnemar_table object

Examples

```
tab <- counts_to_table(tp = 45, fp = 12, tn = 80, fn = 5,
                      candidate = "new method", reference = "gold standard")
tab <- mcnemar(tab)
```

print.mcr	<i>S3 print method</i>
-----------	------------------------

Description

Print data overview and analysis slot status based on \$print slot.

Usage

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

Arguments

x	mcr object
...	Additional arguments

Value

Invisible mcr object

Examples

```
df <- data.frame(sid = 1:30,  
                 test = rnorm(30, 50, 10),  
                 ref = rnorm(30, 50, 10))  
obj <- mcr(df, "sid", "test", "ref")  
print(obj)
```

print.mcr_bias	<i>Print method for bias results</i>
----------------	--------------------------------------

Description

Print method for bias results

Usage

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

Arguments

x	mcr_bias object
...	additional arguments

Value

The unchanged `mcr_bias` data frame, invisibly.

```
print.mcr_bland_altman
```

Print method for Bland-Altman results

Description

Print method for Bland-Altman results

Usage

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

Arguments

<code>x</code>	<code>mcr_bland_altman</code> object
<code>...</code>	additional arguments

Value

The unchanged `mcr_bland_altman` object, invisibly.

```
print.mcr_correlation
```

Print method for correlation results

Description

Print method for correlation results

Usage

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

Arguments

<code>x</code>	<code>mcr_correlation</code> object
<code>...</code>	additional arguments

Value

The unchanged `mcr_correlation` object, invisibly.

print.mcr_describe	<i>Print method for describe results</i>
--------------------	--

Description

Print method for describe results

Usage

```
## S3 method for class 'mcr_describe'  
print(x, digits = NULL, ...)
```

Arguments

x	mcr_describe object
digits	Number of decimal places, default 4
...	additional arguments

Value

The unchanged mcr_describe object, invisibly.

print.mcr_outlier	<i>Print method for outlier results</i>
-------------------	---

Description

Print method for outlier results

Usage

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

Arguments

x	mcr_outlier object
...	additional arguments

Value

The unchanged mcr_outlier object, invisibly.

print.mcr_regression *Print method for regression results*

Description

Print method for regression results

Usage

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

Arguments

x	mcr_regression object
...	additional arguments

Value

The unchanged mcr_regression object, invisibly.

print.precision *Print a precision object overview*

Description

Print the data overview of a precision object (data class, total rows, formula, grouping variables, sample count, factor levels, balance warnings) and the analysis slot status (`[x]` / `[ ]`).

Usage

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

Arguments

x	precision object
...	Reserved arguments

Value

Invisible precision object

```
print.precision_ci
```

Print confidence intervals

Description

Prints confidence interval tables for SD and \ sample, component, estimate, lower, upper.

Usage

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

Arguments

x	precision_ci object (a list of data frames, keyed by "SD" and "CV")
...	Reserved arguments

Value

Invisibly returns x.

```
print.precision_normal
```

Print normality test results

Description

Prints a table of Shapiro-Wilk test results per sample, including sample size, W statistic, and p-value.

Usage

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

Arguments

x	precision_normal object
...	Reserved arguments

Value

Invisibly returns x.

```
print.precision_outlier
```

Print outlier detection results

Description

Prints the outlier detection method, count of outliers found, and the detail table (row, sample, value, statistic, critical value).

Usage

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

Arguments

x	precision_outlier object
...	Reserved arguments

Value

Invisibly returns x.

```
print.precision_profile
```

Print precision profile fits

Description

Prints the best Sadler model formula, AIC, and R-squared for each variance component.

Usage

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

Arguments

x	precision_profile object (a list of per-component fit results)
...	Reserved arguments

Value

Invisibly returns x.

print.precision_vc	<i>Print variance component table</i>
--------------------	---------------------------------------

Description

Prints the variance component summary table with columns: sample, component, VC (variance component), \

Usage

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

Arguments

x	precision_vc object (a data frame)
...	Reserved arguments

Value

Invisibly returns x.

print.reference_interval	<i>Print reference interval analysis results</i>
--------------------------	--

Description

Print reference interval analysis results

Usage

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

Arguments

x	A "reference_interval" object.
...	Additional arguments (reserved for generic).

Value

The unchanged reference_interval object, invisibly.

print.roc	<i>Print a ROC analysis object</i>
-----------	------------------------------------

Description

Print a ROC analysis object

Usage

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

Arguments

- x A roc object.
- ... Additional arguments (currently unused).

Value

x, invisibly.

print.roc_auc_table	<i>Print method for roc AUC table</i>
---------------------	---------------------------------------

Description

Print method for roc AUC table

Usage

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

Arguments

- x roc_auc_table object (data.frame)
- ... additional arguments

Value

The unchanged roc_auc_table data frame, invisibly.

```
print.roc_cutoff_table
```

Print method for roc cutoff table

Description

Print method for roc cutoff table

Usage

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

Arguments

x	roc_cutoff_table object (data.frame)
...	additional arguments

Value

The unchanged roc_cutoff_table data frame, invisibly.

```
print.roc_describe
```

Print method for roc describe results

Description

Print method for roc describe results

Usage

```
## S3 method for class 'roc_describe'  
print(x, digits = NULL, ...)
```

Arguments

x	roc_describe object
digits	number of decimal places, default 4
...	additional arguments

Value

The unchanged roc_describe object, invisibly.

print.roc_mlr	<i>Print a multivariate ROC model</i>
---------------	---------------------------------------

Description

Print a multivariate ROC model

Usage

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

Arguments

x	A roc_mlr object.
...	Additional arguments (currently unused).

Value

x, invisibly.

print.tukey	<i>S3 print method: tukey</i>
-------------	-------------------------------

Description

Print Tukey HSD post-hoc test results including pairwise comparison table, significance markers, and compact letter display.

Usage

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

Arguments

x	A tukey object
...	Other arguments (unused, for S3 generic signature)

Value

Invisibly returns x.

profile.precision	<i>S3 precision profile method</i>
-------------------	------------------------------------

Description

Fit precision (SD) vs. mean across samples using the Sadler model selection algorithm (10 candidate models) based on `VFP::fit_vfp()`.

Usage

```
## S3 method for class 'precision'
profile(object, model.no = 1:10, ...)
```

Arguments

<code>object</code>	precision object (requires <code>variance()</code> to be run first)
<code>model.no</code>	Vector of candidate model numbers, default 1:10
<code>...</code>	Additional arguments passed to <code>.sadler()</code> (e.g. <code>K</code>)

Value

Updated precision object with results stored in `$profile`

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
obj <- variance(obj)
obj <- profile(obj, model.no = 1)
```

qc_chart	<i>Levey-Jennings QC chart + Westgard rule evaluation</i>
----------	---

Description

Draws a Levey-Jennings QC chart and detects out-of-control points based on selected Westgard rules. If target mean and SD are not provided, they are estimated from the data.

Usage

```
qc_chart(  
  data,  
  value,  
  rules = "1-2s,1-3s,2-2s,R-4s,4-1s,10x",  
  mean = NULL,  
  sd = NULL,  
  group = NULL,  
  run = NULL  
)
```

Arguments

data	A data frame.
value	Column name (numeric) containing measured values.
rules	Comma-separated rule names, e.g. "1-2s,1-3s,10x". Rule names must match those printed by list_westgard().
mean	Target mean; NULL to estimate from the selected data column.
sd	Target SD; NULL to estimate from the selected data column.
group	Optional grouping column name (e.g. QC lot), used for faceted plots.
run	Run-order column name (optional, default 1:n).

Value

An object of class "qc_chart" with violation results and plot data. Use print() to show summary and plot() to draw the Levey-Jennings chart.

Examples

```
set.seed(42)  
df <- data.frame(  
  run = 1:30,  
  value = c(rnorm(27, 100, 5), 108, 115, 85)  
)  
res <- qc_chart(df, "value", rules = "1-2s,1-3s,10x")  
print(res)  
plot(res)
```

raw_to_table	Core function: raw data -> frequency fourfold table (silent construction)
--------------	---

Description

Core function: raw data -> frequency fourfold table (silent construction)

Usage

```
raw_to_table(
  data,
  candidate,
  reference,
  id = NULL,
  na.rm = TRUE,
  positive = NULL
)
```

Arguments

<code>data</code>	Data frame, one observation per row
<code>candidate</code>	Candidate method variable name (character), e.g. "method_new"
<code>reference</code>	Reference method variable name (character), e.g. "method_standard"
<code>id</code>	ID variable name (character), optional, used for duplicate detection in describe
<code>na.rm</code>	Whether to remove NA, default TRUE
<code>positive</code>	Specify the positive label (character); if non-NULL, levels will be uniformly mapped to positive/negative

Value

A `fourfold_table` object (list with S3 class "fourfold_table"), containing: - `$freq_raw` Raw frequencies (long format) - `$table` Fourfold matrix (with margins) - `$n` Total sample size - `$print` Print slot (descriptive metadata) - `$candidate_levels` Levels of the candidate method - `$reference_levels` Levels of the reference method Use `print()` directly to display the fourfold table.

Examples

```
df <- data.frame(
  new    = c("positive", "positive", "negative", "negative", "positive", "negative"),
  gold   = c("positive", "negative", "positive", "negative", "positive", "positive"),
  age    = c(45, 52, 38, 61, 47, 55),
  gender = c("M", "F", "F", "M", "M", "F"),
  id     = c("S001", "S002", "S003", "S004", "S005", "S006")
)
result <- raw_to_table(df, "new", "gold", id = "id")
```

Description

Features:

1. Reference interval calculation (CLSI EP28-A3c)
2. Three methods: percentile (non-parametric), parametric (normal), robust (Horn & Pesce)
3. Data quality report: missing values, duplicate IDs
4. Visualization: jitter strip plot + method reference intervals with CI

Usage

```
reference_interval(
  data,
  col,
  interval = 0.95,
  ci = 0.95,
  method = "percentile",
  id = NULL
)
```

Arguments

data	A data frame.
col	Column name (character) of the numeric variable.
interval	Reference interval coverage, default 0.95 (i.e. 2.5%–97.5%).
ci	Confidence level for the reference limits, default 0.95.
method	Calculation method(s): "percentile" (default), "parametric", "robust", or "all" for all three. Multiple methods can be passed as a vector.
id	Optional column name for ID-based duplicate detection.

Details

Dependencies: base R + ggplot2

Value

An object of S3 class "reference_interval" with fields: \$call, \$col, \$id_name, \$n_total, \$n_miss, \$n_complete, \$n_duplicates, \$dup_rows, \$interval, \$ci, \$methods, \$percentile, \$parametric, \$robust, \$values

Examples

```
df <- data.frame(val = rnorm(200, 100, 15))
reference_interval(df, "val")
reference_interval(df, "val", method = "all")
reference_interval(df, "val", method = c("percentile", "robust"))
```

regression.mcr	<i>Regression analysis (S3 method)</i>
----------------	--

Description

Perform regression analysis between the test method (X) and reference method (Y). Supports five methods: Ordinary Least Squares (OLS), Weighted Least Squares (WLS), Deming regression, Weighted Deming regression, and Passing-Bablok regression.

Usage

```
## S3 method for class 'mcr'
regression(
  x,
  method = c("ols", "wls", "deming", "wdeming", "pb"),
  conf.level = 0.95,
  lambda = 1,
  weights = NULL,
  ...
)
```

Arguments

x	mcr object
method	Regression method: "ols" (default), "wls", "deming", "wdeming", "pb"
conf.level	Confidence level, default 0.95
lambda	Variance ratio for Deming regression (reference variance divided by candidate variance), default 1
weights	Weights specification, default NULL (unweighted). Character string for built-in modes: "equal", "1/y", "1/y^2", "1/x", "1/x^2", or a column name in data containing non-negative weights. For OLS: weights are passed to lm() when provided. For WLS / Weighted Deming: weights are required (error if NULL). For Deming / Passing-Bablok: weights are ignored (with a warning).
...	Additional arguments passed to lm (OLS/WLS only)

Value

Updated mcr object (invisible), results stored in \$regression

Examples

```
obj <- mcr(ivd_mcr_example, "sid", "test", "ref")
regression(obj, method = "ols")
regression(obj, method = "deming")
regression(obj, method = "pb")
```

replicate_to_mean	<i>Summarize replicate measurements</i>
-------------------	---

Description

Groups data by x, computes mean, SD, and sample size for each group. When weights is specified, returns an additional weight column. Summary metadata (call, weighting method, column names) is stored as attributes for downstream reference.

Usage

```
replicate_to_mean(data, x, y, weights = NULL, na.rm = TRUE)
```

Arguments

data	a data frame
x	x column name (grouping variable)
y	y column name (response variable)
weights	weighting method: NULL / "n" / "1/sd" / "1/sd^2" NULL – summary only, no weight column "n" – w = number of replicates "1/sd" – w = 1/sd(y) "1/sd^2" – w = 1/var(y) (inverse-variance weighting)
na.rm	remove NA? (default TRUE)

Value

A data.frame with columns:

x	grouping variable
y_mean	group mean
y_sd	group standard deviation
y_n	number of observations per group
weight	(only when weights is specified) weight values

Attributes: call, weights (method name), x_col, y_col

Examples

```
# Basic summary
df <- data.frame(
  conc = rep(c(1, 2, 5, 10), each = 3),
  resp = c(3.1, 3.2, 2.9, 5.8, 6.1, 5.9,
           14.2, 14.5, 14.0, 28.1, 27.9, 28.3)
)
rep <- replicate_to_mean(df, "conc", "resp")
rep
```

```
# Inverse-variance weighting
rep_w <- replicate_to_mean(df, "conc", "resp", weights = "1/sd^2")
rep_w
```

```
residuals.fit_equation
```

Extract residuals from an equation fit

Description

Extract residuals from an equation fit

Usage

```
## S3 method for class 'fit_equation'
residuals(object, ...)
```

Arguments

object	A fit_equation object.
...	Additional arguments passed to the fitted model.

Value

A numeric vector of residuals.

Examples

```
df <- data.frame(x = 1:5, y = c(2.1, 4.0, 6.2, 7.9, 10.1))
f <- fit_equation("E01", df, "x", "y")
residuals(f)
```

```
roc
```

roc constructor

Description

Create an ROC analysis object.

Usage

```
roc(data, cols, reference, id = NULL, positive = NULL)
```

Arguments

data	data frame
cols	evaluation column names (character vector, quantitative, can be multiple)
reference	reference column name (character, single column). Must be binary 0/1, a factor with 2 levels, or a character vector with 2 unique values. If the original reference is quantitative, use <code>continuous_to_binary()</code> first to convert it to 0/1 and pass the new column name.
id	ID column name (optional, for duplicate detection)
positive	specify the positive level of reference (for factor/character)

Details

Supports multiple evaluation columns (quantitative) and a single reference column (binary 0/1, factor, or character with two levels). Automatically detects direction, reports missing values and ID duplicates.

Value

Returns an S3 object of class "roc"

Examples

```
df <- data.frame(
  sid = 1:100,
  x1 = c(rnorm(50, 10, 2), rnorm(50, 12, 2)),
  ref = rep(c(0, 1), each = 50),
  age = 1:100,
  sex = rep(c("F", "M"), each = 50)
)
obj <- roc(df, cols = "x1", reference = "ref", id = "sid")
print(obj)
```

sample_size_bland_altman

Sample size and power for Bland-Altman agreement assessment

Description

Given the clinical limit δ and either n (sample size) or power (target power), calculates the other using the method of Lu et al. (2016).

Usage

```
sample_size_bland_altman(
  n = NULL,
  power = NULL,
  mu,
  sd,
  delta,
  conf.level = 0.95,
  agree.level = 0.95,
  n.min = 3L,
  n.max = 1e+05
)
```

Arguments

n	Sample size (an integer of at least 3). Provide to compute power, or leave NULL to compute n from power.
power	Target power (in (0, 1)). Provide to compute n, or leave NULL to compute power from n.
mu	Mean difference between the two measurement methods.
sd	Standard deviation of the differences. Must be greater than zero.
delta	Clinical agreement limit. Maximum acceptable difference. Must be > 0.
conf.level	Confidence level for the agreement limit confidence interval (default 0.95).
agree.level	Agreement level for the limits of agreement (default 0.95).
n.min	Minimum sample size when searching for n (only used when computing n). Default 3.
n.max	Maximum sample size when searching for n (only used when computing n). Default 1e5.

Value

An object of class `sample_size_bland_altman`: a numeric power value when `n` is supplied, or an integer sample size when `power` is supplied. Analysis inputs are retained as attributes.

References

Lu MJ, Zhong WH, Liu YX, Miao HZ, Li YC, Ji MH (2016) Sample size for assessing agreement between two methods of measurement by Bland-Altman method. *International Journal of Biostatistics*, 12(2). doi:10.1515/ijb20150039

Examples

```
# Mode 1: compute power given sample size
sample_size_bland_altman(n = 203, mu = 0.2, sd = 1, delta = 2.5)

# Mode 2: compute sample size given target power
sample_size_bland_altman(power = 0.80, mu = 0.2, sd = 1, delta = 2.5)
```

sample_size_proportion

Single-arm target value test – sample size / significance level / power

Description

Given any **two** of {n, alpha, power}, compute the third using a confidence-interval inversion approach for a one-sample proportion test against a known target value p_0 .

Usage

```
sample_size_proportion(
  p0 = 0.2,
  p1 = 0.35,
  n = NULL,
  alpha = NULL,
  power = NULL,
  alternative = c("greater", "less", "two.sided"),
  method = c("wilson", "wald", "wald-cc", "agresti-coull", "jeffreys", "wilson-cc",
    "clopper-pearson"),
  n.max = 1e+06,
  tol = 1e-08
)
```

Arguments

p0	Target (null hypothesis) proportion. Must be in (0, 1).
p1	Expected (alternative hypothesis) proportion. Must be in (0, 1).
n	Sample size (integer). Leave NULL to compute it.
alpha	Significance level (Type I error). Leave NULL to compute it.
power	Desired test power (1 - Type II error). Leave NULL to compute it.
alternative	Direction of the alternative hypothesis. One of "greater", "less", or "two.sided".
method	Confidence interval method for test inversion. One of "wald", "wald-cc", "wilson", "wilson-cc", "agresti-coull", "jeffreys", or "clopper-pearson".
n.max	Maximum sample size (only used when computing n).
tol	Tolerance passed to uniroot when solving alpha. Default 1e-8.

Value

An object of class `sample_size_proportion`.

- If computing n: returns an integer sample size.
- If computing power: returns a numeric power in $[0, 1]$.
- If computing alpha: returns a numeric significance level in $(0, 1)$.

Metadata is stored in attributes (method, p0, p1, alpha, power, x_crit, alternative, type). Use [print](#) for full context.

Examples

```
# Mode 1 -- compute n (given alpha and power)
sample_size_proportion(p0 = 0.2, p1 = 0.35,
                       alpha = 0.025, power = 0.8, method = "wilson")

# Mode 2 -- compute power (given n and alpha)
sample_size_proportion(p0 = 0.2, p1 = 0.35,
                       n = 60, alpha = 0.025, method = "wald-cc")

# Mode 3 -- compute alpha (given n and power)
sample_size_proportion(p0 = 0.2, p1 = 0.35,
                       n = 60, power = 0.8, method = "clopper-pearson")
```

sample_size_proportion_ci

Sample size or half-width for a single proportion confidence interval

Description

Given a target proportion p and either d (half-width) or n (sample size), compute the unknown quantity using one of seven confidence interval methods.

Usage

```
sample_size_proportion_ci(
  p,
  n = NULL,
  d = NULL,
  conf.level = 0.95,
  method = c("wilson", "wald", "wald-cc", "agresti-coull", "jeffreys", "wilson-cc",
             "clopper-pearson"),
  n.max = 1e+06,
  tol = 1e-06
)
```

Arguments

<code>p</code>	Expected proportion. Must lie in (0, 1). Required – no default.
<code>n</code>	Sample size (integer). Provide this to compute d , or leave <code>NULL</code> to compute n from d .
<code>d</code>	Desired half-width (margin of error). Provide this to compute n , or leave <code>NULL</code> to compute d from n .
<code>conf.level</code>	Confidence level. Default 0.95.
<code>method</code>	Confidence interval method. One of: "wald", "wald-cc", "wilson", "wilson-cc", "agresti-coull", "jeffreys", "clopper-pearson".
<code>n.max</code>	Maximum sample size to search (only used when solving for n). Default 1e6.
<code>tol</code>	Convergence tolerance. Default 1e-6.

Value

An object of class `sample_size_proportion_ci`.

- If `d` is given: returns an integer sample size `n` with metadata stored in attributes (`method`, `p`, `d`, `conf.level`, `type`).
- If `n` is given: returns a numeric half-width `d` with metadata stored in attributes (`method`, `p`, `n`, `conf.level`, `type`).

Use `print` to see full context.

Examples

```
# Mode 1: compute n (given p and d)
sample_size_proportion_ci(p = 0.3, d = 0.05)

# Mode 2: compute d (given p and n)
sample_size_proportion_ci(p = 0.3, n = 320, conf.level = 0.99)
```

stability_bias

stability_bias — Node-wise bias analysis across storage conditions

Description

Each storage condition is baseline-corrected to its own first time point. Absolute and relative biases are computed and compared across conditions at matched time points.

Usage

```
stability_bias(
  data,
  condition,
  time,
  value,
  reference_condition = NULL,
  limit = NULL,
  bias_type = c("relative", "absolute"),
  time_unit = "d"
)
```

Arguments

<code>data</code>	A data frame.
<code>condition</code>	Column name for storage condition (e.g. temperature).
<code>time</code>	Column name for time point.
<code>value</code>	Column name for measurement value.

reference_condition	Reference condition for between-condition comparison; NULL uses the first occurring condition.
limit	A positive number for symmetric allowable bias; NULL to skip.
bias_type	"relative" (percent) or "absolute".
time_unit	Time unit: m, min, h, d, month, or y.

Value

An S3 object of class "stability_bias".

Examples

```
set.seed(42)
df <- data.frame(cond = rep(c("2-8C", "25C"), each = 8),
                 t = rep(c(0,1,3,7), 4),
                 val = c(rnorm(4,100,2), rnorm(4,98,2),
                        rnorm(4,100,2), rnorm(4,92,3)))
stability_bias(df, "cond", "t", "val", limit = 5)
```

stability_plan	<i>stability_plan — CLSI EP25 Appendix A time-point planning</i>
----------------	--

Description

Based on the ratio of expected drift to allowable drift and the reproducibility variability, looks up the minimum number of time points required per regression series in the EP25 Appendix A tables.

Usage

```
stability_plan(
  allowable_drift,
  variability,
  replicates = 1L,
  expected_drift = NULL,
  power = c(0.8, 0.9),
  mode = c("simple", "components"),
  bias_type = c("relative", "absolute")
)
```

Arguments

allowable_drift	Allowable drift (positive number).
variability	Repeatability CV/SD (simple mode) or a data frame of variance components.
replicates	Number of replicate measurements per time point.

expected_drift Expected drift; NULL defaults to half of allowable_drift.
 power Statistical power: 0.8 or 0.9.
 mode "simple" or "components".
 bias_type "relative" or "absolute".

Value

An S3 object of class "stability_plan".

Examples

```

stability_plan(4, 1.5, replicates = c(1,2,3))

comp <- data.frame(source = c("repeatability", "lot", "operator"),
                   variability = c(1.5, 1.0, 0.8),
                   levels = c(1, 3, 2))
stability_plan(4, comp, replicates = c(1,2,3), mode = "components")

```

stability_regression *stability_regression — CLSI EP25 regression-based stability assessment*

Description

Performs simple linear regression on a single condition's measurements (self mode) or on paired biases between test and reference conditions (compare mode), with one-sided confidence limits to evaluate whether drift remains within allowable limits.

Usage

```

stability_regression(
  data,
  time,
  value,
  condition = NULL,
  mode = c("self", "compare"),
  test_condition = NULL,
  reference_condition = NULL,
  bias_type = c("relative", "absolute"),
  direction = c("auto", "decrease", "increase"),
  conf.level = 0.95,
  limit = NULL,
  time_unit = "d"
)

```

Arguments

<code>data</code>	A data frame.
<code>time</code>	Column name for time point.
<code>value</code>	Column name for measurement value.
<code>condition</code>	Column name for storage condition; required for compare mode.
<code>mode</code>	"self" (single-condition self-reference) or "compare" (test vs reference condition).
<code>test_condition</code>	Test condition; auto-selected if NULL.
<code>reference_condition</code>	Reference condition; NULL uses the first condition.
<code>bias_type</code>	"relative" (percent) or "absolute".
<code>direction</code>	"auto", "decrease", or "increase".
<code>conf.level</code>	One-sided confidence level, default 0.95.
<code>limit</code>	A positive number for symmetric allowable bias; NULL to skip.
<code>time_unit</code>	Time unit: m, min, h, d, month, or y.

Value

An S3 object of class "stability_regression".

Examples

```
set.seed(42)
df <- data.frame(cond = rep(c("2-8C", "25C"), each = 8),
  t = rep(c(0,1,3,7), 4),
  val = c(rnorm(4,100,2), rnorm(4,98,2),
    rnorm(4,100,2), rnorm(4,92,3)))
stability_regression(df, "t", "val", "cond", test_condition = "25C", limit = 5)

set.seed(42)
df <- data.frame(cond = rep(c("Ref", "Test"), each = 8),
  t = rep(c(0,1,3,7), 4),
  val = c(rnorm(4,100,2), rnorm(4,99,2),
    rnorm(4,100,2), rnorm(4,94,3)))
stability_regression(df, "t", "val", "cond", mode = "compare",
  test_condition = "Test", reference_condition = "Ref", limit = 5)
```

stability_time

stability_time — Predict time-to-limit from a stability regression

Description

Given a fitted regression model, computes the time at which the point estimate or one-sided confidence limit first reaches the allowable bias. May extrapolate beyond the observed time range.

Usage

```
stability_time(object, limit = NULL, max_time = NULL)
```

Arguments

object	A stability_regression object.
limit	Symmetric allowable bias.
max_time	Maximum search time; NULL defaults to 100x the maximum observed time.

Value

An S3 object of class "stability_time".

Examples

```
set.seed(42)
df <- data.frame(cond = rep(c("2-8C", "25C"), each = 8),
                 t = rep(c(0,1,3,7), 4),
                 val = c(rnorm(4,100,2), rnorm(4,98,2),
                        rnorm(4,100,2), rnorm(4,92,3)))
r <- stability_regression(df, "t", "val", "cond", test_condition = "25C", limit = 5)
stability_time(r)

set.seed(42)
df <- data.frame(cond = rep(c("Ref", "Test"), each = 8),
                 t = rep(c(0,1,3,7), 4),
                 val = c(rnorm(4,100,2), rnorm(4,99,2),
                        rnorm(4,100,2), rnorm(4,94,3)))
r2 <- stability_regression(df, "t", "val", "cond", mode = "compare",
                        test_condition = "Test", reference_condition = "Ref", limit = 5)
stability_time(r2, limit = 5)
```

```
summary.fourfold_table
```

S3 summary method

Description

Summarizes all key analysis information already executed in a fourfold_table object, formatted following summary.mcr() pattern (mcr.R).

Usage

```
## S3 method for class 'fourfold_table'
summary(object, digits = 3, ...)
```

Arguments

object	a fourfold_table object
digits	Number of decimal places, default 3
...	Reserved arguments

Value

Invisible fourfold_table object

Examples

```
tab <- counts_to_table(tp = 85, fp = 3, tn = 90, fn = 2,
                      candidate = "new method", reference = "gold standard")
summary(tab)
tab <- diagnostics(tab); tab <- kappa(tab); tab <- mcnemar(tab)
summary(tab)
```

summary.mcr	<i>S3 summary method</i>
-------------	--------------------------

Description

Summarize key information from all executed analyses in the mcr object (descriptive statistics, correlation analysis, regression analysis, outlier detection, Bland-Altman analysis, and bias analysis).

Usage

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

Arguments

object	mcr object
...	Additional arguments

Value

Invisible mcr object

Examples

```
df <- data.frame(sid = 1:30,
                 test = rnorm(30, 50, 10),
                 ref = rnorm(30, 50, 10))
obj <- mcr(df, "sid", "test", "ref")
summary(obj)
obj <- correlation(obj)
obj <- regression(obj, method = "ols")
```

```
obj <- bland_altman(obj)
summary(obj)
```

summary.precision	<i>S3 summary method</i>
-------------------	--------------------------

Description

Summarize all executed analyses in a precision object, including data description, analysis status checklist, outlier detection, normality tests, variance components, confidence intervals, and precision profile.

Usage

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

Arguments

object	precision object
...	Reserved arguments

Value

Invisible precision object

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
summary(obj)
```

summary.roc	<i>Summary (S3 method)</i>
-------------	----------------------------

Description

Output key information from all analyses performed on the roc object.

Usage

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

Arguments

object	roc object
...	reserved arguments

Value

invisible roc object

Examples

```
obj <- roc(ivd_roc_example, cols = c("x1", "x2"), reference = "ref")
summary(obj)
obj <- auc(obj)
obj <- cutoff(obj)
obj <- mlr(obj)
summary(obj)
```

tukey	<i>Tukey HSD post-hoc test</i>
-------	--------------------------------

Description

Tukey HSD post-hoc test

Usage

```
tukey(x, ...)
```

Arguments

x	A bottle_anova object
...	Other arguments (passed to methods)

Value

The value returned by the dispatched method.

Examples

```
obj <- bottle_anova(ivd_bottle_example, value ~ bottle)
res <- tukey(obj)
```

tukey.bottle_anova	<i>Tukey HSD post-hoc test (S3 method)</i>
--------------------	--

Description

Perform Tukey HSD pairwise comparisons on an ANOVA result, with compact letter display.

Usage

```
## S3 method for class 'bottle_anova'  
tukey(x, term = NULL, conf.level = NULL, ...)
```

Arguments

x	A bottle_anova object
term	Effect term to analyse (character vector); NULL for all non-residual terms
conf.level	Confidence level; NULL uses the value stored in the object
...	Other arguments (unused)

Value

An S3 object of class "tukey" with components:

response_name	Response variable name
conf.level	Confidence level
tukey_list	List, one element per term, each with matrix, means, cld

Note: The original bottle_anova object's \$tukey_results is also updated.

Examples

```
obj <- bottle_anova(ivd_bottle_example, value ~ bottle)  
res <- tukey(obj)  
res
```

variance.precision	<i>S3 variance component estimation method</i>
--------------------	--

Description

For each sample, run `VCA::anovaVCA()` to estimate variance components, compute standard deviation (SD), coefficient of variation (CV%), and their percentage of total variance. Results are stored in the `$results` and `$vc` slots.

Usage

```
## S3 method for class 'precision'
variance(x, digits = 4, ...)
```

Arguments

- `x` precision object
- `digits` Number of decimal places, default 4
- `...` Additional arguments passed to `VCA::anovaVCA()` (e.g. `NegVC`, `conf.level`)

Value

Updated precision object, `$results` contains per-sample VCA results, `$vc` is the summary variance component table

Examples

```
data(VCAdata1, package = "VCA")
obj <- precision(VCAdata1, y ~ day/run, by = "sample")
obj <- variance(obj)
```

youden_plot	<i>Youden plot</i>
-------------	--------------------

Description

Draws a Youden plot comparing two measurement systems. If target means and SDs are not provided, they are estimated from the data.

Usage

```
youden_plot(  
  data,  
  sample1,  
  sample2,  
  mean1 = NULL,  
  mean2 = NULL,  
  sd1 = NULL,  
  sd2 = NULL  
)
```

Arguments

data	A data frame.
sample1	Column name for sample 1.
sample2	Column name for sample 2.
mean1	Target mean for sample 1; NULL to estimate.
mean2	Target mean for sample 2; NULL to estimate.
sd1	Target SD for sample 1; NULL to estimate.
sd2	Target SD for sample 2; NULL to estimate.

Value

An object of class "youden_plot" with summary statistics and plot data. Use `print()` to show summary and `plot()` to draw the Youden plot.

Examples

```
set.seed(42)  
df <- data.frame(  
  m1 = rnorm(30, 100, 5),  
  m2 = rnorm(30, 98, 5)  
)  
res <- youden_plot(df, "m1", "m2", mean1 = 100, mean2 = 100, sd1 = 5, sd2 = 5)  
print(res)  
plot(res)
```

Index

* datasets

- ivd_bottle_example, 21
- ivd_mcr_example, 21
- ivd_qualitative_example, 22
- ivd_roc_example, 22

arrhenius, 4

auc (describe), 14

auc.roc, 5

bias (describe), 14

bias.mcr, 6

bland_altman (describe), 14

bland_altman.mcr, 7

bottle_anova, 8

ci (describe), 14

ci.precision, 9

coef.fit_equation, 10

compare_equation, 10

continuous_to_binary, 11

correlation (describe), 14

correlation.mcr, 12

counts_to_table, 12

cutoff (describe), 14

cutoff.roc, 13

describe, 14

describe.fourfold_table, 15

describe.mcr, 16

describe.roc, 17

diagnostics (describe), 14

diagnostics.fourfold_table, 17

fit_equation, 18

ivd_bottle_example, 21

ivd_mcr_example, 21

ivd_qualitative_example, 22

ivd_roc_example, 22

kappa (describe), 14

kappa.fourfold_table, 23

list_equation, 24

list_sadler, 24

list_westgard, 25

lob_lod_loq, 26

mcnemar (describe), 14

mcnemar.fourfold_table, 28

mcr, 29

mkt, 30

mlr (describe), 14

mlr.roc, 30

name, 31

normal (describe), 14

normal.precision, 31

normal_test, 32

outlier, 33

outlier.mcr, 33

outlier.precision, 34

outliers_test, 35

plot.fit_equation, 36

plot.mcr, 37

plot.precision, 38

plot.reference_interval, 39

plot.roc, 39

precision, 40

predict.fit_equation, 41

predict.mcr, 42

predict.roc, 43

print, 68, 70

print.bottle_anova, 44

print.describe_table, 45

print.diagnostics, 45

print.fit_equation, 46

print.fourfold_table, 47

print.kappa_table, 47

print.mcnemar_table, 48
print.mcr, 49
print.mcr_bias, 49
print.mcr_bland_altman, 50
print.mcr_correlation, 50
print.mcr_describe, 51
print.mcr_outlier, 51
print.mcr_regression, 52
print.precision, 52
print.precision_ci, 53
print.precision_normal, 53
print.precision_outlier, 54
print.precision_profile, 54
print.precision_vc, 55
print.reference_interval, 55
print.roc, 56
print.roc_auc_table, 56
print.roc_cutoff_table, 57
print.roc_describe, 57
print.roc_mlr, 58
print.tukey, 58
profile, 40
profile(describe), 14
profile.precision, 59

qc_chart, 59

raw_to_table, 60
reference_interval, 61
regression(describe), 14
regression.mcr, 63
replicate_to_mean, 64
residuals.fit_equation, 65
roc, 65

sample_size_bland_altman, 66
sample_size_proportion, 68
sample_size_proportion_ci, 69
stability_bias, 70
stability_plan, 71
stability_regression, 72
stability_time, 73
summary, 40
summary.fourfold_table, 74
summary.mcr, 75
summary.precision, 76
summary.roc, 76

tukey, 77

tukey.bottle_anova, 78

uniroot, 68

variance(describe), 14
variance.precision, 79
vc(describe), 14

youden_plot, 79