Package 'trialSizing'

August 5, 2026

Type Package

Title Tools for Experimental Design Sizing

Version 0.1.0

Description Sizes field experiments from uniformity-trial data, following the relationship between the coefficient of variation and plot size. Checks a trial for the spatial structure the sizing methods assume (semivariogram, Moran's I, kriged field map), summarises the coefficient of variation over every plot shape the grid admits, and estimates the optimal plot size by the modified maximum curvature method of Meier and Lessman (1971), by the linear response plateau (LRP) and quadratic response plateau (QRP) models, and by the closed form of Paranaiba, Ferreira and Morais (2009), which can be compared side by side. From the coefficient of variation at the optimum it derives the number of replications needed to detect a given difference between treatment means, as in Cargnelutti Filho and others (2014). Every method returns standardised diagnostic statistics, optional bootstrap uncertainty for the breakpoint, and publication-style plots.

License GPL (>= 3)

URL https://willyanjnr.github.io/trialSizing/, https://github.com/willyanjnr/trialSizing

BugReports https://github.com/willyanjnr/trialSizing/issues

Encoding UTF-8

Language en

LazyData true

Depends R (>= 3.5)

Imports ggplot2, stats, utils

Suggests knitr, rmarkdown, patchwork, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

NeedsCompilation no

Author Willyan Bandeira [aut, cre] (ORCID:
<https://orcid.org/0000-0002-9430-3664>),
Leonardo Pradebon [aut] (ORCID:
<https://orcid.org/0000-0001-7827-6312>),
Ivan Carvalho [aut] (ORCID: <https://orcid.org/0000-0001-7947-4900>),
Murilo Loro [aut] (ORCID: <https://orcid.org/0000-0003-0241-4226>)

Maintainer Willyan Bandeira <bandeira.wjab@gmail.com>

Repository CRAN

Date/Publication 2026-08-05 09:30:02 UTC

Contents

`calc_cv_shapes` *CV by plot shape from a uniformity-trial grid*

Description

Builds the coefficient-of-variation table that `fit_lrp()`, `fit_qrp()` and `fit_mcm()` consume, starting from the raw grid of basic experimental units (BEU) as the trial was harvested. Adjacent BEU are grouped into every rectangular plot shape the grid admits: for a grid of L rows by C columns, each shape is $X_L \times X_C$ basic units with X_L a divisor of L and X_C a divisor of C. For each shape the plot totals are formed and their mean, standard deviation and CV reported.

Usage

```
calc_cv_shapes(
  .data,
  value = NULL,
  row_id = NULL,
  col_id = NULL,
  trial = NULL,
  min_plots = 2
)
```

Arguments

`.data`	a matrix (one trial), a named list of matrices (several trials), or a long data frame with one row per basic unit.
`value`	name of the column holding the BEU measurement (long format).
`row_id`, `col_id`	names of the row and column index columns (long format).
`trial`	optional column name identifying the trial (long format).
`min_plots`	minimum number of plots a shape must yield to be kept (default 2).

Details

This closes the pipeline: the grid goes in, the CV table comes out, and it feeds the fitters directly, since the returned columns are already named `x`, `cv` and `trial`.

Value

A data frame with one row per shape and trial: `trial`, `X_L`, `X_C` (shape in basic units), `x` (plot size), `n` (number of plots), `mean`, `sd` and `cv` (percent), ordered by plot size.

Shapes and plot counts

A shape of area $X = X_L X_C$ yields $n = LC/X$ plots, so the CV of a large plot size rests on few plots and is estimated with little information: the last rows of the table are the least reliable. The `n` column is returned for exactly this reason and is the natural weight for a weighted fit. Shapes

leaving fewer than `min_plots` plots are dropped, which by default removes only the whole grid (a single plot, no variance).

Shapes are not interchangeable at equal area: 1×2 and 2×1 cover the same two basic units but run along different directions of the field, and their CVs differ whenever fertility is not isotropic. Both are reported, which is why the CV table has repeated `x` values.

References

Cargnelutti Filho, A. et al. (2025). Determinacao do tamanho de parcela para avaliar a massa de parte aerea de grao-de-bico. *Revista Vivencias*, 21(43), 499-513.
Paranaiba, P. F., Ferreira, D. F. & Morais, A. R. (2009). *Revista Brasileira de Biometria*, 27(2), 255-268.

See Also

`fit_lrp()`, `fit_qrp()`, `fit_mcm()`, `calc_paranaiba()`

Examples

```
## Three trials of the bundled simulated uniformity trial, each an 8 x 12
## grid of basic units (see ?uniformity_trial).
grids <- lapply(split(uniformity_trial, uniformity_trial$trial),
                function(d) as.matrix(d[, grep("^col", names(d))]))

tab <- calc_cv_shapes(grids)
head(tab, 8)

## Same area, different orientation, different CV
tab[tab$trial == "T1" & tab$x == 2, ]

## The table feeds the fitters unchanged
fit_lrp(tab[tab$trial == "T1", ], x = "x", cv = "cv", step = 0.05)


## One model per trial, straight from the grids
fit_lrp(tab, x = "x", cv = "cv", trial = "trial", step = 0.01)$summary

## n falls as the plot grows: the last shapes rest on very few plots
unique(tab[, c("x", "n")])
```

`calc_paranaiba` *Optimal plot size by the Paranaiba method*

Description

Estimates the optimal plot size from the raw uniformity-trial grid using the maximum curvature of the coefficient of variation model (Paranaiba, Ferreira & Morais, 2009). Unlike `fit_lrp()`, `fit_qrp()` and `fit_mcm()`, which take CV values already computed for several plot sizes, this method works directly on the basic experimental units (BEU) and returns a closed-form estimate:

$$X_o = \frac{10\sqrt[3]{2(1-\rho^2)s^2 m}}{m}, \qquad CV_{Xo} = \frac{100\sqrt{(1-\rho^2)s^2/m^2}}{\sqrt{X_o}}$$

where m and s^2 are the mean and variance of the BEU values and ρ is the first-order spatial autocorrelation.

Usage

```
calc_paranaiba(
  .data,
  value = NULL,
  row_id = NULL,
  col_id = NULL,
  trial = NULL,
  n_row = NULL,
  n_col = NULL,
  rho_direction = c("row", "col", "mean")
)
```

Arguments

`.data`	a matrix, a list of matrices, or a data frame.
`value`	name of the column holding the BEU measurement (long format).
`row_id`, `col_id`	names of the row and column index columns (long format).
`trial`	optional column name identifying the trial.
`n_row`, `n_col`	grid dimensions; required only for a matrix supplied as a plain vector, or to check the grid of a long data frame.
`rho_direction`	`"row"` (default), `"col"` or `"mean"`.

Value

An object of class `"paranaiba_fit"`: a list with `summary` (one row per trial: mean, variance, CV, rho_row, rho_col, rho, Xo, CVxo, valid) and `meta`.

Direction of the autocorrelation

ρ is estimated along a serpentine walk through the grid. The original method walks in the direction of the rows (`rho_direction = "row"`, the default). `"col"` walks down the columns and `"mean"` averages both directions; these can give visibly different ρ and so a different X_o.

Input

Supply either a matrix (one trial), a list of matrices (several trials), or a data frame in long format with the value column plus row/column indices, or a data frame whose `n_col` value columns hold the grid (see `value`, `row_id`, `col_id`, `trial`).

References

Paranaiba, P. F., Ferreira, D. F. & Morais, A. R. (2009). Tamanho otimo de parcelas experimentais: proposicao de metodos de estimacao. *Revista Brasileira de Biometria*, 27(2), 255-268.
Cargnelutti Filho, A. et al. (2014). Tamanho de parcela e numero de repeticoes em aveia preta. *Ciencia Rural*, 44(10), 1732-1739.

See Also

`fit_lrp()`, `fit_mcm()`, `calc_replicates()`

Examples

```
## The bundled simulated uniformity trial holds three trials, each an 8 x 12
## grid of basic experimental units (columns col01-col12); see
## ?uniformity_trial.
col_cols <- grep("^col", names(uniformity_trial))
rep1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1", col_cols])
dim(rep1)

## One trial, from a matrix
calc_paranaiba(rep1)

## Several trials, from a named list of matrices
grids <- lapply(split(uniformity_trial, uniformity_trial$trial),
                function(d) as.matrix(d[, grep("^col", names(d))]))
par_fit <- calc_paranaiba(grids)
par_fit$summary

## rho is estimated along a serpentine walk. The original method walks the
## rows; walking the columns instead can give a visibly different Xo.
cbind(
  row  = calc_paranaiba(grids)$summary$Xo,
  col  = calc_paranaiba(grids, rho_direction = "col")$summary$Xo,
  mean = calc_paranaiba(grids, rho_direction = "mean")$summary$Xo
)

## Long format: one row per basic unit, with row and column indices
long <- data.frame(
  trial = rep(uniformity_trial$trial, times = length(col_cols)),
  row   = rep(uniformity_trial$row,   times = length(col_cols)),
  col   = rep(names(uniformity_trial)[col_cols], each = nrow(uniformity_trial)),
  value = unlist(uniformity_trial[, col_cols], use.names = FALSE)
)
calc_paranaiba(long, value = "value", row_id = "row", col_id = "col",
               trial = "trial")$summary
```

```
plot(par_fit)

## CVxo of each trial feeds the number of replications
calc_replicates(treatments = c(5, 10, 20),
                cv_percent = mean(par_fit$summary$CVxo),
                lsd_percent = c(10, 20))
```

`calc_replicates` *Estimate the optimal number of replications*

Description

Estimates the number of replications for CRD or RCBD experiments following Cargnelutti Filho et al. (2014), from the number of treatments, the experimental coefficient of variation, the least significant difference (LSD, as a percent of the mean) and the significance level, using the Tukey (studentized range) critical value.

Usage

```
calc_replicates(
  treatments,
  cv_percent,
  lsd_percent,
  alpha = 0.05,
  design = c("CRD", "RCBD"),
  tol = 1e-09,
  max_iter = 1000
)
```

Arguments

`treatments`	numeric vector; numbers of treatments (>= 2).
`cv_percent`	single numeric; experimental CV (%), e.g. the CVxo from `fit_lrp()`/`fit_qrp()`/`fit_mcm()`.
`lsd_percent`	numeric vector; least significant differences (% of mean).
`alpha`	significance level for the Tukey test (default 0.05).
`design`	"CRD" or "RCBD".
`tol`, `max_iter`	convergence tolerance and iteration cap for the fixed point (defaults 1e-9 and 1000).

Details

The required replications solve $r = (q_\alpha CV/LSD)^2$, where q_α is the Tukey critical value. Since q_α depends on the error df, which depends on r, the problem is solved iteratively. Two readings are returned: `r_continuous`, the continuous fixed point (this reproduces the published tables), and `r_optimal`, the practical `ceiling(r_continuous)` floored at 2 (a design needs at least two replications). Error df: $t(r-1)$ for CRD and $(t-1)(r-1)$ for RCBD.

Value

An object of class "replicates_fit": data (Treatments, CV_percent, LSD_percent, Alpha, Design, r_continuous, r_optimal, df_error, q_tukey, converged) and meta.

References

Cargnelutti Filho, A. et al. (2014). Tamanho de parcela e numero de repeticoes em aveia preta. *Ciencia Rural*, 44(10), 1732-1739.

Examples

```
## CV = 9.25% is the CVxo of the black oat trial (Cargnelutti Filho et al.,
## 2014). How many replications to detect a difference of 10% or 20% of the
## mean, for a few treatment numbers?
fit <- calc_replicates(treatments = c(5, 10, 20, 30), cv_percent = 9.25,
                       lsd_percent = c(10, 20), design = "CRD")
fit

## r_continuous is the fixed point the published tables report; r_optimal is
## the practical ceiling, floored at 2.
fit$data[fit$data$LSD_percent == 10, ]

## A randomized complete block design spends one df per block, so it needs
## slightly more replications than a completely randomized design.
rcbd <- calc_replicates(treatments = c(5, 10, 20, 30), cv_percent = 9.25,
                        lsd_percent = c(10, 20), design = "RCBD")
cbind(CRD = fit$data$r_optimal, RCBD = rcbd$data$r_optimal)


## The full published table: every treatment number from 3 to 50, at three
## precision levels.
full <- calc_replicates(treatments = 3:30, cv_percent = 9.25,
                        lsd_percent = c(10, 20, 30), design = "CRD")
head(full$data)
plot(full)

## A stricter test costs replications
calc_replicates(treatments = 10, cv_percent = 9.25, lsd_percent = 10,
                alpha = 0.01)$data[, c("Alpha", "r_continuous", "r_optimal")]
```

---

check_trial *Check a uniformity trial before sizing plots*

Description

Inspects the raw grid of basic experimental units and reports what should be settled before any plot-size method is run: whether the grid is complete and usable, whether the values are sane, and whether the field has spatial structure worth sizing plots against. The result carries a plot() method drawing the field map.

Usage

```
check_trial(
  .data,
  value = NULL,
  row_id = NULL,
  col_id = NULL,
  trial = NULL,
  cell_size = c(1, 1),
  n_bins = 15,
  max_dist = NULL,
  variogram = TRUE
)
```

Arguments

.data	a matrix (one trial), a named list of matrices, or a long data frame with one row per basic unit.
value	name of the measurement column (long format).
row_id, col_id	names of the row and column index columns (long format).
trial	optional column name identifying the trial.
cell_size	numeric c(row, col): the distance between the centres of adjacent basic units, in the units you want distances reported in. Default c(1, 1).
n_bins	number of distance classes for the empirical variogram.
max_dist	largest separation used; NULL (default) takes half the maximum distance in the field, the usual rule.
variogram	logical; fit the variogram (default TRUE). Turning it off skips the only appreciable computation.

Value

An object of class "trial_check": a list with checks (one element per trial, each holding the dimensions, statistics, outliers, trend, spatial measures, fitted variogram and any issues) and summary, one row per trial.

What is checked

Structure Grid dimensions, missing cells, and how many rectangular plot shapes the grid admits. A grid whose sides are prime yields almost no shapes, which stops the CV-based methods before they start; the count is flagged when it falls below six.

Values Missing, negative, zero and constant values, and outliers by the boxplot rule. In a uniformity trial an outlying basic unit is usually a harvest failure or a typing error, and it contaminates every shape that contains it.

Trend Means along rows and columns, with the p-value of a monotone trend. A gradient in one direction is field fertility, and it is why shapes of equal area but different orientation give different CVs.

Spatial structure Moran's I with its p-value, the first-order autocorrelations used by calc_paranaiba(), and a fitted variogram.

Why the variogram matters here

The fitted model gives three numbers that bear directly on plot size. The *range* is the distance beyond which basic units stop being correlated: a plot larger than the range is averaging units that are already independent, which is the spatial reading of the plateau that `fit_lrp()` and `fit_qrp()` estimate empirically. The *nugget-to-sill ratio* is the share of variance with no spatial structure; following Cambardella et al. (1994) it is read as strong (below 0.25), moderate (0.25 to 0.75) or weak (above 0.75) spatial dependence. When it is weak, the field varies almost at random from unit to unit and no choice of plot size will buy much precision – worth knowing before fitting five models to the CV table.

The model is fitted by profiling the range over a grid and solving the nugget and partial sill exactly at each candidate, under non-negativity. Spherical, exponential and Gaussian shapes are all tried and the best weighted fit is kept; exponential and Gaussian use the effective-range convention, where the range is the distance reaching 95% of the sill.

References

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, 58(8), 1246-1266.
Moran, P. A. P. (1950). Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2), 17-23.
Cambardella, C. A. et al. (1994). Field-scale variability of soil properties in central Iowa soils. *Soil Science Society of America Journal*, 58(5), 1501-1511.
Cressie, N. (1993). *Statistics for Spatial Data*. Wiley, New York.
Webster, R. & Oliver, M. A. (2007). *Geostatistics for Environmental Scientists*, 2nd ed. Wiley, Chichester.

See Also

`calc_cv_shapes()`, `calc_paranaiba()`, `plot.trial_check()`

Examples

```
## The bundled simulated uniformity trial (see ?uniformity_trial)
grid1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                    grep("^col", names(uniformity_trial))])
chk <- check_trial(grid1)
chk

## the fitted variogram, as numbers rather than as a picture
chk$checks[[1]]$variogram[c("model", "nugget", "sill", "range",
                            "nugget_ratio", "dependence")]


## the field map: kriged surface with the basic units drawn on top
plot(chk)

## several trials at once
grids <- lapply(split(uniformity_trial, uniformity_trial$trial),
                function(d) as.matrix(d[, grep("^col", names(d))]))
check_trial(grids)$summary

## a grid whose sides are prime admits almost no plot shapes
```

```
check_trial(matrix(rnorm(77, 100, 10), nrow = 7))$checks[[1]]$issues
```

`compare_methods` *Compare the plot-size methods on the same trial*

Description

Runs the CV-based methods on one set of data and tabulates what each recommends: the optimal plot size X_o, the CV at that size, and comparable fit statistics. Given the raw grid instead of a CV table, it also builds the table with `calc_cv_shapes()` and adds `calc_paranaiba()`, which works on the basic units rather than on CV values.

Usage

```
compare_methods(
  .data,
  x = NULL,
  cv = NULL,
  trial = NULL,
  methods = NULL,
  step = 0.001,
  weights = FALSE,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95
)
```

Arguments

`.data`	one of: a data frame holding a CV table (with `x` and `cv` columns); a matrix, being the raw grid of basic experimental units of one trial; or a named list of such matrices. Grids are expanded into CV tables with `calc_cv_shapes()`, and only then can the Paranaiba method take part.
`x`, `cv`	column names in `.data` when it is a data frame (default `"x"` and `"cv"`, the names `calc_cv_shapes()` produces).
`trial`	optional column name identifying the trial.
`methods`	which methods to run; `NULL` (default) runs every applicable one. Any subset of `"mcm"`, `"lrp"`, `"qrp"` and `"paranaiba"`, the last requiring a grid.
`step`	grid step for the breakpoint search of the LRP and QRP.
`weights`	weighting for the fits, as in `fit_lrp()`. `TRUE` uses the `n` column, which a grid always provides. It reaches the MCM through that method's Federer `df` argument, the same weighted least squares by another name. The Paranaiba row is unaffected: it is a closed form over the basic units and never sees the CV table.
`bootstrap`	logical; add a confidence interval for each X_o, and the breakpoint-existence p-value where the method has one.
`n_boot`, `conf_level`	bootstrap size and confidence level.

Value

An object of class "method_comparison": a list with summary (one row per method and trial: trial, method, Xo, CVxo, R2, RMSE, and Xo_lwr, Xo_upr, p_breakpoint when bootstrap = TRUE) and meta.

What the table is for

The methods disagree systematically: the MCM optimum is the smallest, the LRP intermediate and the QRP the largest, an ordering reported across many crops. Seeing the three side by side, with intervals when bootstrap = TRUE, shows whether that ordering is a real difference or three readings of the same imprecise quantity.

Which statistics are comparable

R2 and RMSE are recomputed here from the residuals on the original CV scale, so they mean the same thing for every method even when the fit itself was weighted. AIC and BIC are deliberately absent: the MCM has two parameters against the plateau models' three plus a breakpoint, and the breakpoint is not an ordinary parameter, so the information criteria are not on a common footing. The Paranaiba estimate is a closed form with no fitted residuals, so its R2 and RMSE are NA by nature, not by omission.

References

Cargnelutti Filho, A. et al. (2025). *Revista Vivencias*, 21(43), 499-513, which compares the same three methods and reports 4.81, 7.19 and 10.25 m2 for the MCM, LRP and QRP respectively.

See Also

fit_lrp(), fit_qrp(), fit_mcm(), calc_paranaiba(), calc_cv_shapes()

Examples

```
## The bundled simulated uniformity trial (see ?uniformity_trial): one
## 8 x 12 grid of 1 m2 basic units per trial.
grid1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                    grep("^col", names(uniformity_trial))])

## From a CV table (built here from the grid)
cv_tab <- calc_cv_shapes(list(T1 = grid1))
cmp <- compare_methods(data.frame(x = cv_tab$x, cv = cv_tab$cv), step = 0.05)
cmp


## From the raw grid: the CV table is built on the way, and the Paranaiba
## method joins because it needs the basic units.
compare_methods(grid1, step = 0.05)

## Weighted by the number of plots per shape
compare_methods(grid1, step = 0.05, weights = TRUE)$summary

## With intervals, which is the point: the spread between methods is often
```

```
## smaller than the uncertainty within each one.
set.seed(1)
compare_methods(grid1, step = 0.05, bootstrap = TRUE, n_boot = 200)

## Several trials at once
grids <- lapply(split(uniformity_trial, uniformity_trial$trial),
                function(d) as.matrix(d[, grep("^col", names(d))]))
compare_methods(grids, step = 0.05)$summary
```

`fit_lrp` *Fit the Linear Response Plateau (LRP) model by grid search*

Description

Fits the linear-plateau (broken-line) model

$$f(x) = a + b\,x \;\text{ if } x \le X_0, \qquad f(x) = a + b\,X_0 \;\text{ if } x > X_0$$

by profiling the breakpoint X_0 over a fine grid. For each candidate breakpoint the linear coefficients are obtained by least squares and the plateau is set to $a + b\,X_0$; the breakpoint minimizing the residual sum of squares over all observations is returned. Profiling the breakpoint avoids the starting-value sensitivity and local-minima of a direct `nls` fit, so no initial values are required.

Usage

```
fit_lrp(
  .data = NULL,
  x = NULL,
  cv = NULL,
  trial = NULL,
  step = 0.001,
  method = c("segment", "ramp"),
  search_range = NULL,
  start = NULL,
  local_min_tol = 0.1,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  weights = FALSE
)
```

Arguments

`.data`	optional data frame. When supplied, `x`, `cv` and `trial` are interpreted as column names (character strings). When `NULL` (default) the vector interface is used.
`x`, `cv`	either numeric vectors (vector interface) or, when `.data` is a data frame, the names of the predictor and response columns.

trial	optional name of the column identifying the trial. When given, one model is fit per trial.
step	grid step for the breakpoint search. Default 0.001; larger values run faster with a slightly coarser breakpoint.
method	how the linear coefficients are estimated at each candidate breakpoint. "segment" (default) uses only observations with x <= X0, reproducing the Paranaiba et al. (2009) procedure. "ramp" uses all observations on the basis pmin(x, X0), the standard least-squares LRP.
search_range	optional numeric c(lower, upper) restricting the interval (in units of x) where the breakpoint is searched. Must fall within the data range. Useful when the optimum is known to lie in a region and an outlier could otherwise pull the breakpoint outside it. NULL (default) searches the full feasible interval.
start	optional single breakpoint value. The fit is unchanged, but the result also carries a $compat element holding the local minimum of the basin containing start: the solution a gradient-based fitter (nls, nlsLM) seeded there would return. Use it to reproduce published results obtained with such implementations, and to see how much worse they fit.
local_min_tol	relative SSE tolerance (default 0.10) deciding which local minima count as competing. A second basin fitting within 10% of the optimum means the breakpoint is not sharply identified. Only labelling is affected (the competing column of $local_minima and the stars in print()); the fit itself never changes, and no warning is issued.
bootstrap	logical; if TRUE, also estimate the uncertainty of the breakpoint by resampling (default FALSE). Off by default because the published procedure reports the point estimate alone. See the section "Uncertainty of the breakpoint".
n_boot	number of bootstrap resamples (default 1000). Used only when bootstrap = TRUE.
conf_level	confidence level of the percentile interval (default 0.95). Used only when bootstrap = TRUE.
weights	weights for a weighted least-squares fit. FALSE (default) fits unweighted, as the published procedure does. TRUE uses the n column of .data, the number of plots behind each CV, as returned by calc_cv_shapes(). A single column name or a numeric vector are also accepted. See the section "Weighting by the number of plots".

Details

The response is typically the coefficient of variation (CV, percent) and the predictor the plot size. Supply the individual CV values (one per basic-unit form), not the means per plot size; repeated x values are expected.

Value

For a single series, an object of class "lrp_fit": a list with coefficients (a, b); parameters (Breakpoint, Breakpoint_Response, R2, RMSE, AIC, BIC); fitted; residuals; data; method; step; local_min_tol; local_minima (the competing basins, with their SSE excess over the optimum and a competing flag, or NULL); sse_profile (the SSE at every candidate breakpoint);

compat when start was given; and bootstrap when bootstrap = TRUE, a list with ci, se, p_value (existence of the breakpoint), statistic, replicates, n_valid and conf_level.
With trial, an object of class "lrp_multi": a list with summary (one row per trial), fits (the individual "lrp_fit" objects) and method.

Competing breakpoints

With few distinct plot sizes the residual sum of squares is a stepped function of the breakpoint, and it often has several local minima. The grid search always returns the global optimum, but a second basin may fit almost as well, in which case the breakpoint is not sharply identified and different implementations legitimately disagree. Every local minimum of the profile is reported in $local_minima; those fitting within local_min_tol of the optimum are flagged in the competing column and starred by print(). No warning is issued: on stepped profiles competing basins are common, so a warning would fire on almost every fit. Inspect $local_minima and $sse_profile instead, and lower local_min_tol to flag only near-ties. Gradient-based fitters return whichever basin their starting value lands in; start reproduces that.

Weighting by the number of plots

The CV values are not equally reliable. A shape of area X leaves $n = LC/X$ plots in the grid, so the CV of the largest plot size may rest on two plots while the smallest rests on dozens. weights = TRUE fits by weighted least squares with n as the weight, which is the natural measure of how much information stands behind each point.

This is off by default because it is not what the published procedure does, and it moves the answer: the small plot sizes, where the CV is highest, gain most of the weight, so the fitted line is pulled towards them and the breakpoint typically falls. Report both if you use it.

Two cautions. Weighting corrects for unequal information, not for dependence: every CV in the table comes from the same grid of basic units, so the points are not independent with or without weights. And a weighted fit's R^2, RMSE, AIC and BIC follow the lm convention of being computed on weighted residuals, so they cannot be compared with those of the unweighted fit.

Uncertainty of the breakpoint

bootstrap = TRUE adds two things the point estimate cannot give.

The first is a percentile confidence interval: the shapes (the rows of the CV table) are resampled with replacement, the model is refit on each resample, and the empirical quantiles of the resulting breakpoints form the interval. Resamples with fewer than three distinct plot sizes cannot place a breakpoint and are discarded; $bootstrap$n_valid reports how many were kept. The interval is usually wide, which is the honest reading of a breakpoint estimated from a handful of plot sizes.

The second is a test of whether the breakpoint exists at all. Under the null hypothesis that the CV falls linearly and never plateaus, the breakpoint is not identified, so the usual likelihood-ratio statistic does not have a chi-square distribution (Davies, 1987). The null distribution is simulated instead: residuals of the straight-line fit are resampled, the plateau model is refit to each simulated series, and the p-value is the proportion of simulated SSE reductions that match or exceed the observed one. A large p-value means a straight line explains the data as well as the plateau, and the optimal plot size should not be read off this fit.

Both are computed on the same breakpoint grid as the main fit, so step controls their resolution too. Set the random seed before calling to make the result reproducible.

Two ways to call

Vectors `fit_lrp(x, cv)` fits a single series and returns an `"lrp_fit"`.

Data frame `fit_lrp(.data, x = "x", cv = "cv", trial = "trial")` takes a data frame plus column names. With `trial`, one model is fit per trial and an `"lrp_multi"` object (summary table plus the individual fits) is returned. Column names default to `"x"`, `"cv"` and `"trial"`; missing columns raise a clear error.

References

Paranaiba, P. F., Ferreira, D. F. & Morais, A. R. (2009). Tamanho otimo de parcelas experimentais: proposicao de metodos de estimacao. *Revista Brasileira de Biometria*, 27(2), 255-268.
Cargnelutti Filho, A. et al. (2025). *Revista Vivencias*, 21(43), 499-513.
Davies, R. B. (1987). Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika*, 74(1), 33-43.
Efron, B. & Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, New York.

See Also

`plot.lrp_fit()`, `predict.lrp_fit()`

Examples

```
## A simulated uniformity trial bundled with the package (see
## ?uniformity_trial). Group its 1 m2 basic units into plots of every shape,
## then read one CV per shape -- the input the plateau models expect. Several
## shapes share an area, so the plot sizes repeat.
grid1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                    grep("^col", names(uniformity_trial))])
cv_tab <- calc_cv_shapes(list(T1 = grid1))
X   <- cv_tab$x    # plot size (m2)
CV1 <- cv_tab$cv   # CV (%) among plots of that shape

## A coarse grid runs fast and already reproduces Xo to two decimals; the
## default step = 0.001 refines the third.
fit <- fit_lrp(X, CV1, step = 0.01)
fit
coef(fit)
fit$parameters[c("Breakpoint", "Breakpoint_Response")]

## CV expected at plot sizes that were not evaluated
predict(fit, newx = c(2, 5, 7.5, 15))


## Full precision is the default; it costs about ten times more time and
## only refines the third decimal, so it is shown here rather than used
## throughout:
# fit_lrp(X, CV1)$parameters["Breakpoint"]

## Title and styling belong to plot(), not to the fit
plot(fit, title = "Uniformity trial, T1")
```

```
## Weighting by the number of plots ---------------------------------------
## The CV of a large shape rests on very few plots, that of a 1 m2 shape on
## many. Weighting by n (carried in the CV table) pulls the fit towards the
## small sizes, so Xo falls.
c(unweighted = unname(fit_lrp(X, CV1, step = 0.01)$parameters["Breakpoint"]),
  weighted   = unname(fit_lrp(X, CV1, step = 0.01,
                              weights = cv_tab$n)$parameters["Breakpoint"]))

## Straight from the grid table, `weights = TRUE` finds the n column itself
fit_lrp(cv_tab, x = "x", cv = "cv", step = 0.05,
        weights = TRUE)$parameters["Breakpoint"]

## Uncertainty of Xo ------------------------------------------------------
## Off by default, because the published procedure reports the point alone.
set.seed(1)
unc <- fit_lrp(X, CV1, step = 0.01, bootstrap = TRUE, n_boot = 200)
unc
unc$bootstrap$ci

## p_value tests whether a breakpoint exists at all: a large value means a
## straight line explains the CV just as well, and no plateau should be read.
unc$bootstrap$p_value

## Competing breakpoints --------------------------------------------------
## Every local minimum of the SSE profile is reported; those fitting within
## local_min_tol of the optimum are flagged as competing.
fit$local_minima

## Lower the tolerance to flag only near-ties
fit_lrp(X, CV1, step = 0.01, local_min_tol = 0.02)$local_minima

## The whole profile, for inspection
plot(fit$sse_profile, type = "l", xlab = "Breakpoint", ylab = "SSE")
abline(v = fit$parameters["Breakpoint"], col = "forestgreen")

## What a gradient fitter (nls, nlsLM) seeded at 12 would have returned.
## The reported fit does not change; $compat shows the cost in SSE.
fit_lrp(X, CV1, step = 0.01, start = 12)$compat

## Other arguments --------------------------------------------------------
## Restrict the search when an outlier pulls the breakpoint away
fit_lrp(X, CV1, step = 0.01, search_range = c(5, 12))$parameters["Breakpoint"]

## "ramp" estimates the descending line from every observation instead of
## only those below the breakpoint, which can shift the optimum
fit_lrp(X, CV1, step = 0.01, method = "ramp")$parameters["Breakpoint"]

## Several trials at once -------------------------------------------------
trials <- rbind(
  data.frame(x = X, cv = CV1,        trial = "T1"),
  data.frame(x = X, cv = CV1 * 0.85, trial = "T2")
)
res <- fit_lrp(trials, x = "x", cv = "cv", trial = "trial", step = 0.01)
```

```
res$summary

## CVxo feeds the number of replications
calc_replicates(treatments = c(5, 10, 20),
                cv_percent = unname(fit$parameters["Breakpoint_Response"]),
                lsd_percent = c(10, 20))
```

`fit_mcm` *Fit the Modified Maximum Curvature (MCM) plot-size model*

Description

Estimates the optimal plot size by the modified maximum curvature method of Meier & Lessman (1971). The relationship $CV = a\,X^{-b}$ is fit by linear regression of $\log CV$ on $\log X$, and the optimum is the point of maximum curvature

$$X_c = \left[a^2 b^2 (2b+1)/(b+2)\right]^{1/(2b+2)}.$$

Usage

```
fit_mcm(
  .data = NULL,
  x = NULL,
  cv = NULL,
  df = NULL,
  trial = NULL,
  method = c("nls", "loglinear"),
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95
)
```

Arguments

`.data`	optional data frame; when supplied the other arguments are column names.
`x`, `cv`	numeric vectors (plot size and CV in percent), or column names.
`df`	optional degrees of freedom per point for the Federer (1955) weighting, as a vector or a column name. `NULL` = unweighted.
`trial`	optional column name identifying the trial.
`method`	estimation method: `"nls"` (default, original scale, as in the modern Brazilian plot-size papers) or `"loglinear"` (Meier & Lessman regression).
`bootstrap`	logical; if `TRUE`, also estimate the uncertainty of X_c by resampling the shapes (default `FALSE`). See the section "Uncertainty of the optimum".

n_boot	number of bootstrap resamples (default 1000), used only when bootstrap = TRUE.
conf_level	confidence level of the percentile interval (default 0.95), used only when bootstrap = TRUE.

Value

An "mcm_fit" (single series) or "mcm_multi" (per trial). With bootstrap = TRUE the fit also carries bootstrap, a list with ci, se, ci_b, replicates, n_valid and conf_level.

Estimation method

method = "nls" (default) fits $CV = aX^{-b}$ by nonlinear least squares on the original scale (seeded from the log-log fit), reproducing Cargnelutti Filho et al. (2025) and the modern Brazilian plot-size papers. method = "loglinear" fits the line $\log CV = \log a - b \log X$ (the classic Meier & Lessman route). If "nls" fails to converge it falls back to the log-linear estimate with a warning.

Degrees-of-freedom correction

Pass df, the degrees of freedom of each point, to weight the fit as proposed by Federer (1955); this down-weights the CV of larger plot sizes, estimated from fewer plots. Some authors apply it, others do not; df = NULL (default) is unweighted. The cost-factor modification (K1, K2) of the classic method is not applied; X_c is in the units of x.

Uncertainty of the optimum

bootstrap = TRUE resamples the shapes with replacement, refits, and returns a percentile interval for X_c in $bootstrap$ci.

Unlike fit_lrp() and fit_qrp(), there is no test for the existence of the optimum, because there is no breakpoint whose presence is in doubt: the curve $CV = aX^{-b}$ has a point of maximum curvature whenever $b > 0$. What can be questioned is b. Its bootstrap interval is therefore reported as $bootstrap$ci_b; if that interval covers zero, the CV does not demonstrably fall with plot size and X_c carries no meaning.

Set the random seed before calling to make the result reproducible.

Two ways to call

Vectors fit_mcm(x, cv, df = NULL) returns an "mcm_fit".

Data frame fit_mcm(.data, x = "x", cv = "cv", df = "df", trial = "trial"); with trial, one model per trial is fit and an "mcm_multi" object is returned.

References

Meier, V. D. & Lessman, K. J. (1971). Estimation of optimum field plot shape and size for testing yield in *Crambe abyssinica* Hochst. *Crop Science*, 11, 648-650.
Federer, W. T. (1955). *Experimental Design*. Macmillan, New York.

See Also

fit_lrp(), fit_qrp(), plot.mcm_fit()

Examples

```
## CV per plot shape from the bundled simulated uniformity trial
## (see ?uniformity_trial).
grid1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                    grep("^col", names(uniformity_trial))])
cv_tab <- calc_cv_shapes(list(T1 = grid1))
X   <- cv_tab$x
CV1 <- cv_tab$cv

fit <- fit_mcm(X, CV1)
fit

## The fitted decay CV = a * X^-b, and the CV expected at unobserved sizes
coef(fit)
predict(fit, newx = c(2, 5, 7.5, 15))

## "loglinear" is the classic Meier & Lessman route (regression of log CV on
## log X); "nls" (default) fits on the original scale and is what the modern
## plot-size articles report. They rarely agree exactly.
c(nls       = unname(fit$parameters["Breakpoint"]),
  loglinear = unname(fit_mcm(X, CV1, method = "loglinear")$parameters["Breakpoint"]))

## Federer (1955) weighting: the CV of a large plot size rests on fewer
## plots, so pass the degrees of freedom of each point (n - 1) to down-weight
## it. The plot count n is carried in the CV table.
fit_mcm(X, CV1, df = cv_tab$n - 1)$parameters["Breakpoint"]

## Uncertainty of Xc, off by default. There is no existence test here: the
## curve always has a maximum-curvature point when b > 0, so what gets an
## interval is b itself.
set.seed(1)
unc <- fit_mcm(X, CV1, bootstrap = TRUE, n_boot = 200)
unc
unc$bootstrap$ci_b

## One model per trial
trials <- rbind(
  data.frame(x = X, cv = CV1,        trial = "T1"),
  data.frame(x = X, cv = CV1 * 0.85, trial = "T2")
)
fit_mcm(trials, x = "x", cv = "cv", trial = "trial")$summary


plot(fit, title = "Uniformity trial, T1")

## The three CV-based methods on the same data. MCM is the most
## conservative and QRP the most generous; the ordering is systematic.
c(MCM = unname(fit$parameters["Breakpoint"]),
```

```
  LRP = unname(fit_lrp(X, CV1, step = 0.01)$parameters["Breakpoint"]),
  QRP = unname(fit_qrp(X, CV1, step = 0.01)$parameters["Breakpoint"]))
```

`fit_qrp` *Fit the Quadratic Response Plateau (QRP) model by grid search*

Description

Fits the quadratic-plateau (smooth broken-line) model

$$f(x) = a + bx + cx^2 \ \text{ if } x \le X_0, \qquad f(x) = a - b^2/(4c) \ \text{ if } x > X_0,$$

with $X_0 = -b/(2c)$. The breakpoint is profiled over a grid: for each candidate X_0 the model is linear in the plateau level and the curvature, so it is fit by least squares with no starting values, and the X_0 minimizing the residual sum of squares is returned. This mirrors `fit_lrp()` and avoids the convergence problems of a direct nls fit.

Usage

```
fit_qrp(
  .data = NULL,
  x = NULL,
  cv = NULL,
  trial = NULL,
  step = 0.001,
  search_range = NULL,
  start = NULL,
  local_min_tol = 0.1,
  bootstrap = FALSE,
  n_boot = 1000,
  conf_level = 0.95,
  weights = FALSE
)
```

Arguments

`.data`	optional data frame; when supplied, `x`/`cv`/`trial` are column names.
`x`, `cv`	numeric vectors, or column names when `.data` is a data frame.
`trial`	optional column name identifying the trial.
`step`	grid step for the breakpoint search (default 0.001).
`search_range`	optional `c(lower, upper)` restricting the breakpoint search; must fall within the data range.
`start`	optional single breakpoint value. The fit is unchanged, but the result also carries `$compat`: the local minimum of the basin containing `start`, i.e. what a gradient-based fitter seeded there would return. Rarely needed for the QRP, whose SSE profile is usually a single smooth basin.

`local_min_tol` relative SSE tolerance (default 0.10) deciding which local minima count as competing. Only labelling is affected (the `competing` column of `$local_minima` and the stars in `print()`); the fit never changes, and no warning is issued.

`bootstrap` logical; if `TRUE`, also estimate the uncertainty of the breakpoint by resampling (default `FALSE`). See the section "Uncertainty of the breakpoint".

`n_boot` number of bootstrap resamples (default 1000), used only when `bootstrap = TRUE`.

`conf_level` confidence level of the percentile interval (default 0.95), used only when `bootstrap = TRUE`.

`weights` weights for a weighted least-squares fit. `FALSE` (default) fits unweighted, as the published procedure does. `TRUE` uses the `n` column of `.data` (the number of plots behind each CV, as returned by `calc_cv_shapes()`); a column name or a numeric vector are also accepted. The caveats are the same as for `fit_lrp()`: the breakpoint typically falls, the points remain dependent, and the weighted fit statistics are not comparable with the unweighted ones.

Value

A `"qrp_fit"` (single series) or `"qrp_multi"` (per trial). The fit also carries `local_minima` (competing basins with their SSE excess over the optimum and a `competing` flag, or `NULL`), `local_min_tol`, `sse_profile`, `compat` when `start` was given, and `bootstrap` when `bootstrap = TRUE` (a list with `ci`, `se`, `p_value`, `statistic`, `replicates`, `n_valid`, `conf_level` and `null_model`). Because the quadratic-plateau joins smoothly, this profile is typically a single basin, unlike the stepped profile of `fit_lrp()`.

Two ways to call

Vectors `fit_qrp(x, cv)` returns a `"qrp_fit"`.

Data frame `fit_qrp(.data, x = "x", cv = "cv", trial = "trial")`; with `trial`, one model per trial is fit and a `"qrp_multi"` object is returned. Column names default to "x", "cv", "trial".

Uncertainty of the breakpoint

`bootstrap = TRUE` resamples the shapes (the rows of the CV table) with replacement, refits on each resample, and returns a percentile interval for X_o in `$bootstrap$ci`, together with a bootstrap standard error.

It also tests whether the plateau is warranted. The null here is not the one used by `fit_lrp()`: a quadratic-plateau that never plateaus is simply a quadratic, so the null model is the unconstrained quadratic fitted on every observation, and the p-value is the proportion of null resamples whose SSE reduction matches or exceeds the observed one. A large p-value means the plateau segment buys nothing over a plain quadratic, and X_o should not be read as an optimal plot size.

Set the random seed before calling to make the result reproducible.

See Also

`fit_lrp()`, `plot.qrp_fit()`

Examples

```
## CV per plot shape from the bundled simulated uniformity trial
## (see ?uniformity_trial).
grid1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                    grep("^col", names(uniformity_trial))])
cv_tab <- calc_cv_shapes(list(T1 = grid1))
X   <- cv_tab$x
CV1 <- cv_tab$cv

## A coarse grid runs fast; the default step = 0.001 refines the third decimal
fit <- fit_qrp(X, CV1, step = 0.01)
fit

## The quadratic joins its plateau smoothly, so the optimum is larger than
## the one the broken-line model gives on the same data
coef(fit)
predict(fit, newx = c(2, 5, 7.5, 15))


## Full precision is the default; it costs about eight times more time and
## only refines the third decimal, so it is shown here rather than used
## throughout:
# fit_qrp(X, CV1)$parameters["Breakpoint"]

plot(fit, title = "Uniformity trial, T1")

## The smooth join makes the SSE profile a single basin, unlike the stepped
## profile of fit_lrp(): competing minima are rare and much worse.
plot(fit$sse_profile, type = "l", xlab = "Breakpoint", ylab = "SSE")
fit$local_minima

## Uncertainty of Xo, off by default. The p-value asks whether the plateau
## buys anything over a plain quadratic.
set.seed(1)
unc <- fit_qrp(X, CV1, step = 0.01, bootstrap = TRUE, n_boot = 200)
unc
unc$bootstrap$ci

## Restrict the breakpoint search
fit_qrp(X, CV1, step = 0.01, search_range = c(5, 15))$parameters["Breakpoint"]

## One model per trial, with a summary table
trials <- rbind(
  data.frame(x = X, cv = CV1,        trial = "T1"),
  data.frame(x = X, cv = CV1 * 0.85, trial = "T2")
)
fit_qrp(trials, x = "x", cv = "cv", trial = "trial", step = 0.01)$summary
```

`plot.lrp_fit` *Plot an LRP fit (publication style)*

Description

Draws the observed points, the fitted broken line, dotted guides to the breakpoint and the plateau-model annotations, in the layout used in plot-size articles: the model block (equations and R^2) centred at the top and Xo/CVxo next to the breakpoint. Axis limits come from the data.

Usage

```
## S3 method for class 'lrp_fit'
plot(
  x,
  title = "Linear Plateau",
  annotate_model = TRUE,
  xlab = NULL,
  ylab = "CV (%)",
  decimal_mark = c(".", ","),
  cond_word = "if",
  digits_coef = 3,
  digits_stat = 2,
  point_size = 2.3,
  line_size = 0.8,
  point_colour = "black",
  line_colour = "black",
  bp_colour = "red",
  base_size = 12,
  label_size = 4.6,
  title_size = NULL,
  family = "sans",
  theme = NULL,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 12,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

x	an object of class "lrp_fit".
title	plot title.
annotate_model	logical; draw the model equations and statistics.
xlab, ylab	axis titles. xlab = NULL uses "Plot size (m^2)".
decimal_mark	decimal separator for the annotations, "." or ",".
cond_word	conditional word in the equations (e.g. "if" or "se").

digits_coef, digits_stat
: decimals for the coefficients (a, b) and for the statistics (Xo, CVxo, R2).

point_size, line_size
: sizes of the points and the fitted line.

point_colour, line_colour, bp_colour
: colours of the points, the fitted line and the breakpoint marker.

base_size
: base font size for the theme; axis titles and text scale from it.

label_size
: size of the annotation text (equations, Xo, CVxo).

title_size
: size of the plot title. NULL uses base_size.

family
: font family for the theme and annotations.

theme
: optional **ggplot2** theme object used instead of the package default (the shared trialSizing theme); the title_size tweak is applied on top.

save
: logical; if TRUE, write the figure to disk (default FALSE).

file
: output file name; if NULL, derived from title.

format
: one of "tiff", "png", "jpeg", "pdf", "eps". TIFF/PDF/EPS are preferred for journals; TIFF is written with LZW compression.

dpi
: resolution for raster formats.

width, height, units
: figure size.

compression
: TIFF compression (default "lzw").

...
: ignored.

Value

A ggplot object (invisibly when saved).

---

plot.lrp_multi *Plot LRP fits for several trials (paginated grid)*

Description

Arranges the per-trial plots in a grid of at most 6 panels (3 rows by 2 columns). With more than 6 trials the panels are paginated: each page is a separate 3x2 figure, and with save = TRUE each page is written to its own file. Requires the **patchwork** package.

Usage

```
## S3 method for class 'lrp_multi'
plot(
  x,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
```

```
  dpi = 300,
  width = 18,
  height = 24,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

x	an object of class "lrp_multi".
save	logical; write the page(s) to disk (default FALSE).
file	base file name; page number and extension are appended when there is more than one page.
format, dpi, width, height, units, compression	passed to the saver.
...	styling arguments forwarded to plot.lrp_fit() (for example decimal_mark, cond_word, label_size).

Value

A list of page objects, invisibly.

---

plot.mcm_fit *Plot an MCM fit (publication style)*

Description

Draws the observed points, the fitted power curve aX^{-b}, dotted guides to the maximum-curvature point and the model annotation. Unlike the plateau models there is no flat segment: the curve keeps decreasing.

Usage

```
## S3 method for class 'mcm_fit'
plot(
  x,
  title = "Modified Maximum Curvature",
  annotate_model = TRUE,
  xlab = NULL,
  ylab = "CV (%)",
  decimal_mark = c(".", ","),
  digits_coef = 3,
  digits_stat = 2,
  point_size = 2.3,
  line_size = 0.8,
```

```
  point_colour = "black",
  line_colour = "black",
  bp_colour = "red",
  base_size = 12,
  label_size = 4.6,
  title_size = NULL,
  family = "sans",
  theme = NULL,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 12,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

x	an "mcm_fit" object.
title, annotate_model, xlab, ylab, decimal_mark, digits_coef, digits_stat	as in plot.lrp_fit().
point_size, line_size, point_colour, line_colour, bp_colour	aesthetics.
base_size, label_size, title_size, family, theme	theme and text controls.
save, file, format, dpi, width, height, units, compression	saving controls.
...	ignored.

Value

A ggplot object (invisibly when saved).

---

plot.mcm_multi *Plot MCM fits for several trials (paginated grid)*

Description

Plot MCM fits for several trials (paginated grid)

Usage

```
## S3 method for class 'mcm_multi'
plot(
  x,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 24,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

x	an "mcm_multi" object.
save, file, format, dpi, width, height, units, compression	saving controls.
...	styling arguments forwarded to plot.mcm_fit().

Value

A list of page objects, invisibly.

---

plot.paranaiba_fit *Plot Paranaiba estimates by trial*

Description

Shows the optimal plot size per trial, with the mean across trials as a dashed reference line.

Usage

```
## S3 method for class 'paranaiba_fit'
plot(
  x,
  y_var = c("Xo", "CVxo"),
  title = "Paranaiba method",
  xlab = "Trial",
  ylab = NULL,
  show_mean = TRUE,
  fill_colour = "grey35",
  base_size = 12,
  title_size = NULL,
```

```
  family = "sans",
  theme = NULL,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 12,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

x	a "paranaiba_fit" object.
y_var	"Xo" (default) or "CVxo".
title, xlab, ylab	labels; ylab = NULL is chosen from y_var.
show_mean	draw the across-trial mean as a dashed line.
fill_colour	bar fill colour.
base_size, title_size, family, theme	theme controls.
save, file, format, dpi, width, height, units, compression	saving controls.
...	ignored.

Value

A ggplot object (invisibly when saved).

---

plot.qrp_fit *Plot a QRP fit (publication style)*

Description

Same layout and options as plot.lrp_fit(), with the quadratic descending arm and the quadratic equation in the annotation.

Usage

```
## S3 method for class 'qrp_fit'
plot(
  x,
  title = "Quadratic Plateau",
  annotate_model = TRUE,
```

```
  xlab = NULL,
  ylab = "CV (%)",
  decimal_mark = c(".", ","),
  cond_word = "if",
  digits_coef = 3,
  digits_c = 4,
  digits_stat = 2,
  point_size = 2.3,
  line_size = 0.8,
  point_colour = "black",
  line_colour = "black",
  bp_colour = "red",
  base_size = 12,
  label_size = 4.6,
  title_size = NULL,
  family = "sans",
  theme = NULL,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 12,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

`x`	a `"qrp_fit"` object.
`title`	plot title.
`annotate_model`	draw the model equations and statistics.
`xlab`, `ylab`	axis titles. `xlab = NULL` uses "Plot size (m^2)".
`decimal_mark`	decimal separator, "." or ",".
`cond_word`	conditional word in the equations (e.g. "if" or "se").
`digits_coef`, `digits_c`, `digits_stat`	decimals for a/b, for c, and for the statistics.
`point_size`, `line_size`	sizes of points and fitted line.
`point_colour`, `line_colour`, `bp_colour`	point, line and breakpoint colours.
`base_size`, `label_size`, `title_size`, `family`	theme and annotation sizes and font family.
`theme`	optional ggplot2 theme used instead of the default.

save, file, format, dpi, width, height, units, compression
: saving controls; see plot.lrp_fit().

...
: ignored.

Value

A ggplot object (invisibly when saved).

---

plot.qrp_multi *Plot QRP fits for several trials (paginated grid)*

Description

Arranges the per-trial plots in a grid of at most 6 panels (3 x 2), paginating when there are more. Requires **patchwork**. See plot.lrp_multi().

Usage

```
## S3 method for class 'qrp_multi'
plot(
  x,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 24,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

x
: a "qrp_multi" object.

save, file, format, dpi, width, height, units, compression
: saving controls.

...
: styling arguments forwarded to plot.qrp_fit().

Value

A list of page objects, invisibly.

`plot.replicates_fit` *Plot the number of replications*

Description

Draws replications against the number of treatments, one line per LSD level.

Usage

```
## S3 method for class 'replicates_fit'
plot(
  x,
  y_var = c("r_optimal", "r_continuous"),
  title = "Number of replications",
  xlab = "Number of treatments",
  ylab = NULL,
  colour_lab = "LSD (%)",
  line_size = 0.8,
  point_size = 1.8,
  base_size = 12,
  title_size = NULL,
  family = "sans",
  theme = NULL,
  save = FALSE,
  file = NULL,
  format = c("tiff", "png", "jpeg", "pdf", "eps"),
  dpi = 300,
  width = 18,
  height = 12,
  units = "cm",
  compression = "lzw",
  ...
)
```

Arguments

`x`	a `"replicates_fit"` object.
`y_var`	which value to plot: `"r_optimal"` (integer, default) or `"r_continuous"` (the article's tabulated value).
`title`, `xlab`, `ylab`, `colour_lab`	labels.
`line_size`, `point_size`	line and point sizes.
`base_size`, `title_size`, `family`, `theme`	theme controls.

`save`, `file`, `format`, `dpi`, `width`, `height`, `units`, `compression`
: saving controls.

`...`
: ignored.

Value

A `ggplot` object (invisibly when saved).

`plot.trial_check` *Field map of a checked uniformity trial*

Description

Draws the trial as a map: an ordinary-kriging surface built from the variogram fitted by `check_trial()`, with the basic experimental units drawn on top and filled on the same colour scale. The surface shows where the field is systematically better or worse; the points show the data the surface came from, so an interpolation artefact cannot be mistaken for a measurement.

Usage

```
## S3 method for class 'trial_check'
plot(
  x,
  resolution = 120,
  points = TRUE,
  point_values = TRUE,
  point_size = 2.4,
  point_stroke = 0.4,
  point_colour = "grey20",
  surface = TRUE,
  palette = c("viridis", "blues", "greys", "terrain"),
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  legend_title = NULL,
  xlab = "Row",
  ylab = "Column",
  base_size = 12,
  family = "sans",
  ...
)
```

Arguments

`x`
: an object of class `"trial_check"`.

`resolution`
: number of interpolation cells along the longer side of the field (default 120). The surface costs one linear solve, so raising this is cheap.

points	logical; draw the basic units (default TRUE).
point_values	logical; fill the points with their own value (default TRUE). FALSE draws them as plain markers, showing only the sampling positions.
point_size, point_stroke, point_colour	size of the unit markers, the width of their outline, and its colour. A dark rim keeps the markers visible over the pale end of any palette.
surface	logical; draw the kriged surface (default TRUE). With FALSE only the units are drawn, which is the honest picture when the variogram shows weak spatial dependence.
palette	one of "viridis" (default), "blues", "greys" or "terrain". The default is perceptually uniform and readable in greyscale and to colour-blind readers, which several journals now require; "blues" gives the classic look.
title, subtitle, caption, legend_title	plot labels. NULL leaves a sensible default; NA removes the element.
xlab, ylab	axis titles.
base_size, family	base font size and family.
...	ignored.

Details

With several trials the maps are faceted and share one colour scale, which is what makes them comparable.

Value

A ggplot object.

See Also

check_trial()

Examples

```
grid1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                    grep("^col", names(uniformity_trial))])
chk <- check_trial(grid1)


plot(chk)

## the classic look, and points as plain position markers
plot(chk, palette = "blues", point_values = FALSE)

## data only, no interpolation
plot(chk, surface = FALSE)
```

predict.lrp_fit *Predictions from an LRP fit*

Description

Predictions from an LRP fit

Usage

```
## S3 method for class 'lrp_fit'
predict(object, newx = NULL, ...)
```

Arguments

object	an object of class "lrp_fit".
newx	numeric vector of predictor values. Defaults to the fitted data.
...	ignored.

Value

A numeric vector of predicted responses.

predict.mcm_fit *Predictions from an MCM fit*

Description

Predictions from an MCM fit

Usage

```
## S3 method for class 'mcm_fit'
predict(object, newx = NULL, ...)
```

Arguments

object	an "mcm_fit" object.
newx	numeric predictor values; defaults to the fitted data.
...	ignored.

Value

numeric vector of predicted CV values.

predict.qrp_fit *Predictions from a QRP fit*

Description

Predictions from a QRP fit

Usage

```
## S3 method for class 'qrp_fit'
predict(object, newx = NULL, ...)
```

Arguments

object	a "qrp_fit" object.
newx	numeric predictor values; defaults to the fitted data.
...	ignored.

Value

numeric vector of predicted responses.

print.lrp_fit *Print an LRP fit*

Description

Print an LRP fit

Usage

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

Arguments

x	an object of class "lrp_fit".
...	ignored.

Value

x, invisibly.

print.lrp_multi *Print LRP fits for several trials*

Description

Print LRP fits for several trials

Usage

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

Arguments

x	an object of class "lrp_multi".
...	ignored.

Value

x, invisibly.

print.method_comparison *Print a method comparison*

Description

Print a method comparison

Usage

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

Arguments

x	an object of class "method_comparison".
digits	decimals for the printed table.
...	ignored.

Value

x, invisibly.

print.trial_check *Print a trial check*

Description

Print a trial check

Usage

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

Arguments

x	an object of class "trial_check".
...	ignored.

Value

x, invisibly.

summary.lrp_fit *Summarize an LRP fit*

Description

Summarize an LRP fit

Usage

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

Arguments

object	an object of class "lrp_fit".
...	ignored.

Value

object, invisibly.

summary.lrp_multi *Summarize LRP fits for several trials*

Description

Summarize LRP fits for several trials

Usage

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

Arguments

object	an object of class "lrp_multi".
...	ignored.

Value

object, invisibly.

summary.method_comparison *Summarize a method comparison*

Description

Summarize a method comparison

Usage

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

Arguments

object	an object of class "method_comparison".
...	ignored.

Value

The summary data frame, invisibly.

summary.trial_check *Summarize a trial check*

Description

Summarize a trial check

Usage

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

Arguments

`object`	an object of class `"trial_check"`.
`...`	ignored.

Value

The summary data frame, invisibly.

uniformity_trial *Simulated uniformity trial*

Description

A simulated uniformity trial used throughout the package examples and vignettes. A uniformity trial is a field sown uniformly – one genotype, one management – and harvested in a fine grid of small *basic experimental units* (BEU). The spatial variation that remains is environmental noise, and it is what plot-size methods use to decide how large a plot must be.

Usage

```
uniformity_trial
```

Format

A data frame with 24 rows (8 grid rows $\times$ 3 trials) and 14 variables:

trial trial identifier, `"T1"`, `"T2"` or `"T3"`.

row grid row index, 1-8.

col01 response (g m^{-2}) at grid column 1.

col02 response (g m^{-2}) at grid column 2.

col03 response (g m^{-2}) at grid column 3.

col04 response (g m^{-2}) at grid column 4.

col05 response (g m^{-2}) at grid column 5.

col06 response (g m^{-2}) at grid column 6.

col07 response (g m^{-2}) at grid column 7.

col08 response (g m^{-2}) at grid column 8.

col09 response (g m^{-2}) at grid column 9.

col10 response (g m^{-2}) at grid column 10.

col11 response (g m^{-2}) at grid column 11.

col12 response (g m^{-2}) at grid column 12.

To recover a trial as a numeric matrix (the form the grid functions expect):

```
g1 <- as.matrix(uniformity_trial[uniformity_trial$trial == "T1",
                                 grep("^col", names(uniformity_trial))])
```

Details

The data are entirely synthetic (a separable first-order autoregressive field plus independent noise), so they carry no usage restriction. Three independent trials are provided, each an 8 (rows) by 12 (columns) grid of 1 m^2 BEU whose response is a biomass-like measurement in g m^{-2}. The parameters were chosen so the coefficient of variation falls with plot size and levels off at a plateau, as in a real trial. The generating script is in `data-raw/uniformity_trial.R`.

Source

Simulated data for teaching; see `data-raw/uniformity_trial.R`.

See Also

`calc_cv_shapes()`, `calc_paranaiba()`, `check_trial()`, `fit_lrp()`

Index