Package: inferMM (via r-universe)

October 30, 2026

Type Package

Title Variance-Aware Michaelis-Menten Estimation and Inference

Version 0.1.1

Description Variance-aware Michaelis-Menten estimation, model screening, grouped enzyme-kinetic analyses, and clustered repeated-measurement workflows. The package implements profile-score estimators under built-in or user-defined fixed working variance functions for single curves, optional higher-moment estimating equations for independent observations with asymmetric errors, and a lightweight cluster-aware working-covariance extension with optional conditional projected scores for an unspecified random-effect density, Wald and bootstrap confidence intervals, prediction utilities, and simulation helpers. Related methodology is discussed by Kim and Ma (2012) <doi:10.1007/s10463-011-0332-y>, Kim (2023) <doi:10.1002/sta4.606>, and Ma and Genton (2010) <doi:10.1111/j.1467-9868.2010.00741.x>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports graphics, grDevices, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Mijeong Kim [aut, cre], Minkyoung Cha [aut], Ah Young Jeong [aut]

Maintainer Mijeong Kim <m.kim@ewha.ac.kr>

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

Date/Publication 2026-10-30 07:17:35 UTC

RemoteUrl https://github.com/cran/inferMM

RemoteRef HEAD

RemoteSha bcb205aa4aebbcf31f2dae18fce3e4a0e6ffec80

Contents

inferMM-package 2
alves_demo 3
cluster_mm 3
fit_mm 8
fit_mm_moments 12
group_mm 16
report_mm 19
screen_mm 20
sdl_demo 22
simulate_mm_data 23
variance_function 24

Index **26**

`inferMM-package` *inferMM: Variance-Aware Michaelis-Menten Estimation and Inference*

Description

The **inferMM** package provides variance-aware tools for Michaelis-Menten estimation, clustered repeated-measurement analysis, model screening, grouped curve analysis, simulation, and prediction. Single-curve fits and grouped screening accept user-defined fixed positive variance functions in addition to the built-in variance shapes. An optional higher-moment estimator fits independent curves using working skewness and kurtosis, with joint sandwich or model-based covariance.

Details

The package focuses on enzyme-kinetic settings where measurement variability may increase with substrate concentration, making standard homoscedastic nonlinear least squares inadequate for interval estimation and uncertainty quantification.

See Also

`fit_mm`, `fit_mm_moments`, `cluster_mm`, `report_mm`, `screen_mm`, `group_mm`, `simulate_mm_data`, `variance_function`

alves_demo *Subset of the Alves Soil Enzyme Kinetics Data*

Description

A filtered subset of the Alves et al. soil exoenzyme kinetics data, restricted to depth `00-10` and temperature `10` for fast reproducible package examples.

Format

A data frame with 247 rows and 6 variables:

`core` Soil core identifier.

`depth` Depth interval.

`enzyme` Enzyme name.

`temperature` Incubation temperature.

`substrate_conc` Substrate concentration.

`activity` Observed enzyme activity.

Source

Prepared from the Alves study files in the project workspace.

Examples

```
head(alves_demo)
table(alves_demo$enzyme)
```

cluster_mm *Fit a Cluster-Aware Michaelis-Menten Model*

Description

Estimate shared Michaelis-Menten mean parameters from repeated or clustered assay data using a random effect on $V_{\max}$ and a residual working variance function. An opt-in Ma-Genton-style conditional projected-score construction leaves the random-effect density unspecified.

Usage

```
cluster_mm(data, s, v, cluster,
  variance = c("log", "sqrt", "cuberoot", "constant", "power", "auto"),
  power = NULL, power_selection = c("quasi_aic", "quasi_bic"),
  power_grid = default_power_grid(), interval_level = 0.95,
  invalid_response_values = c(-9999),
  invalid_substrate_values = numeric(0), allow_zero_substrate = TRUE,
  sort_x = TRUE, method = c("profile", "projected"), km_bounds = NULL)

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

## S3 method for class 'cluster_mm'
vcov(object, ...)

## S3 method for class 'cluster_mm'
confint(object, parm = c("Vmax", "Km"),
  level = object$interval_level, method = c("wald", "bootstrap"),
  B = 399L, bootstrap_type = c("pairs"),
  bootstrap_ci = c("studentized", "basic", "percentile"),
  seed = NULL, ...)

## S3 method for class 'cluster_mm'
print(x,
  digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'cluster_mm'
summary(object, level = object$interval_level,
  method = c("wald", "bootstrap"), B = 399L,
  bootstrap_type = c("pairs"),
  bootstrap_ci = c("studentized", "basic", "percentile"),
  seed = NULL, ...)

## S3 method for class 'summary.cluster_mm'
print(x,
  digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'cluster_mm'
predict(object, newdata = NULL, se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = object$interval_level, ...)

## S3 method for class 'cluster_mm'
plot(x, interval = TRUE,
  interval_type = c("prediction", "confidence"),
  level = x$interval_level, n_points = 200,
  xlab = "Substrate concentration", ylab = "Reaction velocity",
  main = NULL, pch = 19, col.points = "#1f78b4",
```

```
  col.line = "#222222",
  col.band = grDevices::adjustcolor("#9ecae1", alpha.f = 0.45), ...)
```

Arguments

data	A data frame containing clustered Michaelis-Menten observations.
s	Name of the substrate concentration column.
v	Name of the response column.
cluster	Character vector naming one or more clustering columns. Rows with missing values in any clustering column are removed with a warning before fitting, cluster counting, and variance-model screening.
variance	Character string naming a built-in residual working variance model. Use "power" with a fixed exponent or "auto" to compare log(S + 1) against a grid of power functions.
power	Optional exponent used when variance = "power".
power_selection	Criterion used when variance = "auto": "quasi_aic" or "quasi_bic".
power_grid	Candidate exponents considered when variance = "auto".
interval_level	Confidence level used for Wald intervals.
invalid_response_values	Values to remove before fitting.
invalid_substrate_values	Values to remove before fitting.
allow_zero_substrate	Logical; if FALSE, substrate concentrations must be strictly positive.
sort_x	Logical; sort observations within each cluster by substrate concentration.
object	A fitted "cluster_mm" object.
parm	Parameter names for confint().
level	Confidence level for intervals.
method	In cluster_mm(), the estimator: "profile" retains the existing profile/moment algorithm; "projected" uses conditional projected scores with an identity instrument and a stacked cluster-sandwich covariance. In summary() and confint(), the interval method: "wald" or a cluster-pairs "bootstrap" sensitivity analysis.
km_bounds	Optional positive increasing search bounds for K_m, available only with method = "projected".
B	Number of bootstrap resamples when method = "bootstrap".
bootstrap_type	Bootstrap scheme for clustered fits. The current implementation supports "pairs" only.
bootstrap_ci	Bootstrap confidence-interval calibration: "studentized", "basic", or "percentile".
seed	Deprecated and ignored. Call set.seed() before bootstrap-based summaries or intervals if reproducibility is needed.
x	A fitted "cluster_mm" object for the print and plot methods.

digits	Number of printed digits.
newdata	Optional numeric vector of new substrate concentrations.
se.fit	Logical; if TRUE, return standard errors for mean predictions.
interval	Prediction type: none, confidence, or prediction.
interval_type	For the plot method, draw either a prediction or confidence band when interval = TRUE.
n_points	Number of grid points used for plotting the fitted curve.
xlab, ylab, main, pch, col.points, col.line, col.band	Graphical parameters for the plot method.
...	Additional arguments passed to S3 methods.

Details

The current implementation treats repeated measurements through a working covariance of the form $Z_i B Z_i^\top + \gamma A_i$, where A_i is induced by the chosen residual variance function and the random-effect component is specialized to a first-order perturbation of $V_{\max}$. This provides a lightweight cluster-aware extension of the single-curve profile-score estimator without requiring a fully parametric nonlinear mixed effects specification.

Both methods remove observations with missing values (including numeric NaN) in any clustering column before constructing clusters. A warning reports the number of removed observations. Cluster counts, sizes, residuals, and fitted values refer only to the retained data. Unused factor levels do not count as clusters; a literal character ID "NA" is not a missing value. If no observations or too few clusters remain, fitting stops with an error.

With method = "profile", the routine initializes the fit from the corresponding pooled single-curve estimator and then alternates a one-dimensional update for K_m with plug-in updates for $V_{\max}$ and moment-based updates for the variance components. The pooled start is obtained from the same single-curve profile-score machinery used by fit_mm(), including the constant-variance special case. Fits that reach the maximum number of outer iterations without meeting the convergence tolerance remain available but carry an explicit convergence warning in the object summary. The profile method permits singleton clusters alongside repeated-measurement clusters; singleton covariance blocks retain their one-by-one matrix shape. The projected method still requires at least two readings per retained cluster.

With method = "projected", the assumed conditional model is $Y_i = z_i(K_m)T_i + e_i$, where $T_i = V_{\max} + b_i$, $E(b_i \mid S_i) = 0$, and residuals are conditionally independent Gaussian with covariance γH_i, $H_i = \mathrm{diag}(h(S_{ij}))$. The random-effect density is not specified. Clusters must be independent, the residual variance shape must be correct and fixed, and the random effect must have finite fourth moments for the reported joint covariance. At least five clusters, at least two readings per cluster, a positive loading in every cluster, and within-cluster variation in positive concentrations are required.

For a candidate k, define $c_i = z_i^\top H_i^{-1} z_i$, $W_i = z_i^\top H_i^{-1} Y_i / c_i$, $u_i = Y_i - z_i W_i$, and $q_i = z_i'(k)^\top H_i^{-1} u_i$. This is an MM-specific adaptation of the conditional-centering construction of Ma and Genton (2010, Section 3.2), using the identity instrument $a(W) = W$, not a general GLLVM implementation. Conditional Gaussianity makes W_i and u_i independent given the design, regardless of the random-effect density, so $E(W_i q_i \mid S_i) = 0$. The scalar equation $\sum_i W_i q_i = 0$ estimates K_m. Then $\hat{V}_{\max} = \mathrm{mean}(W_i)$ and $\hat{\gamma} = \sum_i u_i^\top H_i^{-1} u_i / \sum_i (n_i - 1)$. The random-effect variance moment is $\mathrm{mean}\{(W_i - \hat{V}_{\max})^2 - \hat{\gamma}/c_i\}$. A negative value is truncated to zero with a warning, only for reporting the variance component and marginal variance; it does not enter the mean fit.

The new path searches 400 log-spaced points for scalar roots, compares verified interior candidates by within-cluster weighted residual sum of squares, and rejects a standardized score above 10^{-7}. Invalid roots, degenerate variance and singular derivative/covariance calculations stop the fit, with class `inferMM_cluster_projected_error`. No silent Gaussian fallback or covariance ridge is applied. The `stage`, `reason` and `diagnostics` fields on failure conditions to identify the numerical stage and available score/conditioning values. An incomplete grid/refinement search can retain a verified interior root but emits `inferMM_cluster_projected_warning`, with reason `"incomplete_search"`; the message is stored in `convergence$search_warning`. A non-positive variance moment instead has warning reason `"variance_component_boundary"`. Neither warning silently changes the estimation method. Both scalar and joint scaled score residuals are checked against 10^{-7}. Unrepresentable original-unit scales or non-positive covariance diagonals stop fitting. The joint covariance uses the derivatives and empirical covariance of the four cluster-level equations for $V_{\max}$, K_m, γ, and the unconstrained variance moment. Its mean block drives standard errors and Wald intervals; it is a stacked sandwich, not inverse efficient information. `vcov_joint` labels the last coordinate `tau2_unconstrained`. General semiparametric efficiency is not claimed for the identity instrument. Variance-model screening and its intervals condition on the selected variance shape. Quasi-AIC/BIC are retained only as working-normal marginal screening scores, not likelihood information criteria for the unspecified latent-density model.

Projected fits provide mean-confidence intervals. Prediction intervals are not available because an unspecified random-effect density does not determine predictive quantiles. The default plot for a projected fit uses a confidence band instead. This differs from the legacy working-normal prediction band.

Because small numbers of clusters can make first-order interval calibration fragile, printed summaries suppress interval output by default when fewer than four clusters are available or when there are fewer than six distinct substrate concentrations. In that regime, `confint()` remains available as an explicit sensitivity analysis and issues a warning.

Value

The main fitting function returns an object of class `"cluster_mm"`. `predict()` returns either a numeric vector or a data frame containing pointwise confidence or prediction intervals together with working and marginal variance estimates.

References

Ma, Y. and Genton, M. G. (2010). Explicit estimating equations for semiparametric generalized linear latent variable models. *Journal of the Royal Statistical Society: Series B*, 72, 475–495. doi:10.1111/j.14679868.2010.00741.x.

Examples

```
cluster_fit <- cluster_mm(
  data = subset(alves_demo, enzyme == "BG"),
  s = "substrate_conc",
  v = "activity",
  cluster = "core",
  variance = "sqrt"
)
```

```
coef(cluster_fit)
summary(cluster_fit)
confint(cluster_fit)
head(
  predict(
    cluster_fit,
    newdata = seq(0, 700, length.out = 6),
    interval = "confidence"
  )
)

set.seed(351)
S <- c(2, 5, 10, 20, 50, 100, 200)
b <- 12 * (rgamma(60, shape = 2) - 2) / sqrt(2)
dat <- data.frame(id = rep(seq_len(60), each = length(S)), S = rep(S, 60))
dat$Y <- (100 + b[dat$id]) * dat$S / (20 + dat$S) +
  rnorm(nrow(dat), sd = sqrt(2 * sqrt(dat$S)))
projected <- cluster_mm(dat, "S", "Y", "id", variance = "sqrt",
                        method = "projected")
summary(projected)
confint(projected)
predict(projected, c(10, 20, 80), interval = "confidence")


plot(cluster_fit, interval_type = "confidence")
report_mm(cluster_fit, interval_type = "confidence")
```

`fit_mm` *Fit a Variance-Aware Michaelis-Menten Model*

Description

Estimate Michaelis-Menten parameters under a constant or concentration-dependent working variance specification.

Usage

```
fit_mm(x, y,
  variance = c("log", "sqrt", "cuberoot", "constant", "power", "auto"),
  power = NULL, power_selection = c("quasi_aic", "quasi_bic"),
  power_grid = default_power_grid(), interval_level = 0.95,
  km_bounds = NULL, allow_zero_substrate = TRUE)

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

## S3 method for class 'fit_mm'
vcov(object, ...)
```

```
## S3 method for class 'fit_mm'
confint(object, parm = c("Vmax", "Km"),
  level = object$interval_level, method = c("wald", "bootstrap"),
  B = 399L, bootstrap_type = c("wild", "pairs"),
  bootstrap_ci = c("studentized", "basic", "percentile"),
  wild_weights = c("mammen", "rademacher"), seed = NULL, ...)

## S3 method for class 'fit_mm'
print(x, digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'fit_mm'
summary(object, level = object$interval_level,
  method = c("wald", "bootstrap"), B = 399L,
  bootstrap_type = c("wild", "pairs"),
  bootstrap_ci = c("studentized", "basic", "percentile"),
  wild_weights = c("mammen", "rademacher"), seed = NULL, ...)

## S3 method for class 'summary.fit_mm'
print(x,
  digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'fit_mm'
predict(object, newdata = NULL, se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = object$interval_level, ...)

## S3 method for class 'fit_mm'
plot(x, interval = TRUE,
  interval_type = c("prediction", "confidence"),
  level = x$interval_level, n_points = 200,
  xlab = "Substrate concentration", ylab = "Reaction velocity",
  main = NULL, pch = 19, col.points = "#1f78b4",
  col.line = "#222222",
  col.band = grDevices::adjustcolor("#9ecae1", alpha.f = 0.45), ...)
```

Arguments

x	Numeric vector of substrate concentrations. For the plot method, a fitted object of class "fit_mm".
y	Numeric vector of observed reaction velocities.
variance	A built-in working variance model name or a fixed, vectorized function of concentration returning strictly positive values. Use variance_function to attach a label to a custom function. Use "power" with a fixed exponent or "auto" to compare log(S + 1) against a grid of power functions.
power	Optional exponent used when variance = "power".
power_selection	Criterion used when variance = "auto": "quasi_aic" or "quasi_bic".

power_grid	Candidate exponents considered when variance = "auto".
interval_level	Confidence level used for interval estimation.
km_bounds	Numeric vector of length two giving the search bounds for K_m.
allow_zero_substrate	Logical; if FALSE, substrate concentrations must be strictly positive.
object	A fitted "fit_mm" object.
parm	Parameter names for confint().
level	Confidence level for intervals.
method	Interval construction method: "wald" or "bootstrap".
B	Number of bootstrap resamples when method = "bootstrap".
bootstrap_type	Bootstrap scheme: "wild" or "pairs".
bootstrap_ci	Bootstrap interval calibration used when method = "bootstrap": "studentized", "basic", or "percentile".
wild_weights	Wild-bootstrap multipliers used when bootstrap_type = "wild": "mammen" or "rademacher".
seed	Deprecated and ignored. Call set.seed() before bootstrap-based summaries or intervals if reproducibility is needed.
digits	Number of printed digits.
newdata	Optional numeric vector of new substrate concentrations.
se.fit	Logical; if TRUE, return standard errors for mean predictions.
interval	Prediction type: none, confidence, or prediction.
interval_type	For the plot method, draw either a prediction or confidence band when interval = TRUE.
n_points	Number of grid points used for plotting the fitted curve.
xlab, ylab, main, pch, col.points, col.line, col.band	Graphical parameters for the plot method.
...	Additional arguments passed to S3 methods.

Details

Built-in working variance models are "constant", "log", "sqrt", "cuberoot", "power", and "auto". All single-curve fits use the Michaelis-Menten profile-score estimator from the companion methodology work, with the constant-variance case treated as the special case $h(S) \equiv 1$. Confidence intervals can be computed either from the default asymptotic Wald approximation or from a bootstrap refitting scheme. The bootstrap option supports percentile, basic, and studentized calibration, together with wild or pairs resampling. The default printed summaries suppress interval output when the fitted curve has fewer than 20 observations or fewer than 6 distinct substrate concentrations; in that setting, confint() remains available as an explicit sensitivity analysis tool and issues a warning. The automatic option compares log(S + 1) with a user-supplied grid of S^p working variance functions and keeps the best fit according to quasi-AIC or quasi-BIC.

The profile search examines sign-changing brackets on a 400-point log grid and verifies the mean score equations after solving. The diagnostic projects the K_m score direction off the eliminated

$V_{\max}$ direction and standardizes its residual inner product by both weighted norms; the $V_{\max}$ equation is also checked. The maximum absolute standardized score must not exceed 10^{-7}. If no bracketed root is verified, minimization of the squared standardized profile score supplies additional candidates, not automatically accepted estimates. Among detected positive interior roots passing these checks, the smallest weighted residual sum of squares determines the fitted root. A finite grid does not guarantee finding every root or a global minimum.

Candidates within 10^{-6} of either log search bound are rejected, as are non-positive or non-finite parameters and numerically zero residual variance. The mean information is scaled by its gradient-column magnitudes; a reciprocal condition number below 10^{-12} stops covariance calculation. No ridge term is added to obtain the single-curve Wald covariance. These checks do not guarantee identifiability or accurate finite-sample coverage.

Profile validation errors have class `inferMM_profile_error`, also inheriting from `inferMM_convergence_error`. They identify the failure stage, reason, search bounds, score norm, tolerance and candidate count, with information conditioning where relevant. No fit or confidence intervals are returned on these errors. Use `tryCatch()` to inspect the condition. Automatic variance selection records unsuccessful candidates in `variance_candidate_table` using `success = 0` and an `error_message`; they cannot be selected. If every candidate fails, selection stops. Screening and grouped workflows likewise record failures without reporting estimates or intervals for those curves. The profile pilot of `fit_mm_moments` uses the same root-validation checks; the later joint higher-moment solve has its own diagnostics. These new single-curve checks do not replace the separate clustered solver.

A custom function specifies $h(S)$ in $\mathrm{Var}(Y \mid S) = \gamma h(S)$; see `variance_function` for its input/output contract. Only K_m requires numerical search, followed by closed-form updates for $V_{\max}$ and γ. The stored `variance_fun` is reused for new-data prediction and wild or pairs bootstrap refitting. Custom shapes are normalized internally for numerical scaling, without clipping; the reported `scale` is γ for the original supplied shape. Multiplying the shape by a positive constant changes γ inversely, not the fitted variance. Extremely ill-scaled shapes can still produce numerical errors.

Custom shape parameters are not estimated. Wald covariance and working-Gaussian prediction bands condition on the fixed supplied shape. Quasi-AIC and quasi-BIC count three fitted parameters ($V_{\max}$, K_m, γ); they do not account for coefficients fitted externally inside a custom function or for variance-model selection uncertainty. Free parameters, shapes depending on the fitted mean parameters, and data-adaptive variance-function estimation are outside this interface. The feature does not implement conditional third/fourth moment estimation or imply an efficiency gain over identically weighted nonlinear least squares. Prediction bands retain the existing working-normal approximation rather than estimating skewed predictive quantiles.

Value

The main fitting function returns an object of class `"fit_mm"`. Its `convergence` component records the standardized score norm, tolerance, search method, candidate roots, and scaled information condition. `predict()` returns either a numeric vector or a data frame containing prediction intervals.

Examples

```
one_curve <- subset(sdl_demo, enzyme == "1111")

fit <- fit_mm(
  x = one_curve$s_uM,
```

```
  y = one_curve$v_uM_per_min,
  variance = "sqrt"
)

fit_power <- fit_mm(
  x = one_curve$s_uM,
  y = one_curve$v_uM_per_min,
  variance = "power",
  power = 0.4
)

fit_auto <- fit_mm(
  x = one_curve$s_uM,
  y = one_curve$v_uM_per_min,
  variance = "auto",
  power_selection = "quasi_aic"
)

coef(fit)
confint(fit)
set.seed(1)
confint(
  fit,
  method = "bootstrap",
  B = 99,
  bootstrap_ci = "studentized",
  wild_weights = "mammen"
)
head(predict(fit, newdata = seq(0, 80, length.out = 6), interval = "prediction"))

h <- variance_function(function(S) 1 + S / 20,
                       label = "Var = gamma (1 + S/20)")
fit_custom <- fit_mm(one_curve$s_uM, one_curve$v_uM_per_min, variance = h)
summary(fit_custom)
confint(fit_custom)
head(predict(fit_custom, newdata = c(0, 10, 40), interval = "prediction"))


plot(fit)
report_mm(fit, interval_type = "confidence")
```

`fit_mm_moments` *Fit Michaelis-Menten Conditional Higher-Moment Estimating Equations*

Description

Fit independent observations using the first two conditional residual moments, weighted by a covariance involving working third and fourth moments. This is an optional joint estimating-equation extension of the default profile-score estimator, not a parametric gamma or Student-t likelihood fit.

Usage

```
fit_mm_moments(x, y,
  variance = c("log", "sqrt", "cuberoot", "constant", "power"),
  power = NULL,
  moments = c("estimate", "normal", "gamma", "student_t", "custom"),
  shape = NULL, df = NULL, skewness = NULL, kurtosis = NULL,
  covariance = c("model", "sandwich"), interval_level = 0.95,
  km_bounds = NULL, allow_zero_substrate = TRUE, control = list())

## S3 method for class 'fit_mm_moments'
confint(object, parm = c("Vmax", "Km"),
  level = object$interval_level, method = c("wald", "bootstrap"),
  B = 399L, bootstrap_type = "pairs",
  bootstrap_ci = c("studentized", "basic", "percentile"), ...)

## S3 method for class 'fit_mm_moments'
summary(object, level = object$interval_level,
  method = c("wald", "bootstrap"), B = 399L, bootstrap_type = "pairs",
  bootstrap_ci = c("studentized", "basic", "percentile"), ...)

## S3 method for class 'fit_mm_moments'
print(x,
  digits = max(3L, getOption("digits") - 2L), ...)
## S3 method for class 'summary.fit_mm_moments'
print(x,
  digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'fit_mm_moments'
predict(object, newdata = NULL, se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  level = object$interval_level, ...)

## S3 method for class 'fit_mm_moments'
plot(x, interval = TRUE,
  interval_type = "confidence", ...)
```

Arguments

x, y	Concentrations and responses. For print and plot methods, x is the fitted object or summary.
variance	A fixed built-in variance-shape name or a vectorized, fixed positive function $h(S)$. See variance_function. Automatic variance selection is not supported by this estimator.
power	Fixed positive exponent when variance = "power".
moments	Working standardized third and fourth moments: "estimate" estimates constant skewness and kurtosis from centered, standardized pilot residuals; "normal" uses 0 and 3; "gamma" uses $2/\sqrt{a}$ and $3 + 6/a$; "student_t" uses 0 and $3 + 6/(\nu - 4)$; "custom" uses the supplied values or functions.

`shape`	Fixed gamma shape $a > 0$; defaults to 4 in gamma mode. It is not estimated. Only used in gamma mode.
`df`	Fixed Student-t degrees of freedom $\nu > 4$; defaults to 10 in Student-t mode. Only used in that mode.
`skewness`, `kurtosis`	In custom mode, finite numbers or deterministic, pointwise vectorized functions of concentration. Functions must return one value per input. Kurtosis is the standardized fourth moment, not excess kurtosis. The difference $k - 1 - s^2$ must be safely positive. Keep captured constants unchanged after fitting.
`covariance`	Default `"model"` uses joint inverse working information and requires correct higher-moment specifications for its interpretation. Optional `"sandwich"` uses derivatives and empirical outer products of the full joint estimating equations.
`interval_level`, `level`	Confidence level strictly between zero and one.
`km_bounds`	Positive search bounds for K_m, as in `fit_mm()`.
`allow_zero_substrate`	Whether zero concentrations are allowed.
`control`	Named list: `maxit` (100 iterations per continuation attempt), `tol` (10^{-7} scaled-score tolerance), and `min_step` (1/64 minimum continuation step).
`object`	Fitted object.
`parm`	Mean-parameter names to include in the confidence intervals.
`method`	Wald or pairs-bootstrap confidence intervals.
`B`	Number of bootstrap resamples; at least 20.
`bootstrap_type`	Only `"pairs"` is supported. Observation-level wild multipliers do not generally preserve the required higher moments.
`bootstrap_ci`	Studentized, basic, or percentile bootstrap interval.
`digits`	Printed digits.
`newdata`	New concentrations for mean prediction.
`se.fit`	Return standard errors for mean predictions.
`interval`	For prediction, `"none"` or `"confidence"`. `"prediction"` raises an error because four moments do not specify predictive quantiles. For plotting, logical indicating whether to show bands.
`interval_type`	Only pointwise mean-confidence bands are supported.
`...`	Additional arguments to the underlying summary, bootstrap, or graphical methods.

Details

Write $r_i = Y_i - \mu(S_i; \beta)$, $v_i = \gamma h(S_i)$, and $u_i = (r_i, r_i^2 - v_i)^\top$. For working skewness s_i and raw kurtosis k_i, the conditional covariance of u_i is

$$\Omega_i = \begin{pmatrix} v_i & s_i v_i^{3/2} \\ s_i v_i^{3/2} & (k_i - 1) v_i^2 \end{pmatrix}.$$

The derivative matrix is

$$D_i = \begin{pmatrix} g_{i1} & g_{i2} & 0 \\ 0 & 0 & h(S_i) \end{pmatrix},$$

where g_i is the two-parameter MM mean gradient. The estimator solves $\sum_i D_i^\top \Omega_i^{-1} u_i = 0$ jointly for $\theta = (V_{\max}, K_m, \gamma)^\top$. This implements the conditional first/second-moment score of Kim and Ma (2012), Section 3, with working third and fourth moments. No error density is fitted.

Computation uses log parameters and continuation from the profile-score pilot, with damped Newton steps. Working skewness is interpolated from zero and kurtosis from three. A failed solve, boundary solution, or ill-conditioned covariance causes an error rather than a silent return of the pilot estimate. Convergence failures have class `inferMM_convergence_error`; their messages identify the fitting stage, failure reason, total iterations, scaled score norm, tolerance, and suggested checks. Continuation failures also report the completed fraction. No fitted object or confidence intervals are returned. The condition fields can be inspected using `tryCatch()`. Failures during non-Gaussian higher-moment continuation also suggest manually retrying with `moments = "normal"` while keeping the same variance function. This changes the working moments and guarantees neither convergence nor efficiency under the original non-Gaussian moments. No automatic fallback is performed. Only the pilot has the original scalar/closed-form reduction; the new fit uses a joint numerical solve. Multiple roots and weak identification remain possible. Internal variance normalization is undone in the reported scale. The pilot uses the root validation described in `fit_mm`. Its validation errors have class `inferMM_profile_error`, also inheriting from `inferMM_convergence_error`, with profile-specific diagnostic fields.

The optional sandwich is $A^{-1}BA^{-\top}/n$, with A the negative sample-average derivative of all three equations and B their empirical second-moment matrix. Numerical derivatives are evaluated in log parameters and the covariance is transformed back. The default model covariance is the inverse of $\sum_i D_i^\top \Omega_i^{-1} D_i$. Both include the covariance with the estimated scale; mean-parameter intervals use the leading two-by-two block of the joint inverse, not an inverse that treats the scale as known.

With correct conditional mean and variance, suitable regularity and a consistent root, working higher moments need not be correct for consistency. Efficiency in the conditional mean/variance model additionally requires appropriate higher-moment estimation. The constant standardized-moment pilot model is not nonparametric estimation of arbitrary concentration-varying skewness/kurtosis. Its estimation is first-order orthogonal under the model, but can materially affect finite-sample performance. Neither the sandwich nor this feature establishes efficiency or nominal coverage for every sample. Unlike ordinary weighted mean equations, misspecification of $h(S)$ can bias this estimator because it also uses the squared-residual equation.

Normal working moments recover the profile mean/variance equations. Constant symmetric Student-t moments also recover those point estimates; heavy tails alone do not imply an efficiency gain. Gamma moments describe centered, standardized gamma errors, not strictly positive raw gamma errors. At least ten observations are required; estimating higher moments below 50 observations issues an exploratory-use warning, not a universal threshold for reliable inference. The usual small-sample summary policy also applies.

Pairs bootstrap refits this estimator and re-estimates pilot moments in `"estimate"` mode. Predictive bands are deliberately unavailable; mean-confidence bands are pointwise. Quasi-AIC and quasi-BIC are not reported because these roots do not maximize the working Gaussian likelihood. Variance-selection uncertainty and externally estimated variance-shape parameters are not covered. Screening, grouped selection, and the clustered routine retain their existing profile estimators; this extension is for independent single curves only.

Value

An object inheriting from `"fit_mm_moments"` and `"fit_mm"`. `coef()`, `vcov()`, and `confint()` concern $V_{\max}, K_m$. Additional fields include `scale`, `vcov_joint` (all three parameters), `vcov_joint_model`, `vcov_joint_sandwich`, `standardized_moments`, `moment_spec`, `covariance_type`, and `convergence` (scaled score norm, iterations, continuation stages, and derivative reciprocal condition number).

References

Kim, M. and Ma, Y. (2012). The efficiency of the second-order nonlinear least squares estimator and its extension. *Annals of the Institute of Statistical Mathematics*, 64, 751–764. doi:10.1007/s104630110332y.

Examples

```
set.seed(19)
dat <- simulate_mm_data(n = 200, error = "skewed", skew_shape = 4)
h <- function(S) 1 + 9 * S / (20 + S)
fit <- fit_mm_moments(dat$x, dat$y, variance = h)
summary(fit)
confint(fit, level = 0.95)
fit$vcov_joint

# Fixed working moments, not a gamma likelihood fit.
fit_gamma <- fit_mm_moments(dat$x, dat$y, variance = h,
                            moments = "gamma", shape = 4)
coef(fit_gamma)
predict(fit_gamma, c(10, 20, 80), interval = "confidence")

set.seed(20)
confint(fit, method = "bootstrap", B = 99)
plot(fit)
```

---

group_mm *Fit Multiple Michaelis-Menten Curves by Group*

Description

Clean a data frame, fit one or more working variance models within each group, and compare the competing fits using quasi-AIC as the primary criterion, with quasi-BIC and RMSE reported alongside it.

Usage

```
group_mm(data, s, v, groups = NULL,
  variance_models = default_variance_models(), power_values = NULL,
  include_auto = FALSE, power_selection = c("quasi_aic", "quasi_bic"),
  power_grid = default_power_grid(), interval_level = 0.95,
```

```
  invalid_response_values = c(-9999),
  invalid_substrate_values = numeric(0), allow_zero_substrate = TRUE,
  sort_x = TRUE, quiet = FALSE, variance_functions = NULL)

## S3 method for class 'group_mm'
print(x,
  digits = max(3L, getOption("digits") - 2L), ...)

## S3 method for class 'group_mm'
plot(x, which_groups = NULL, model = NULL,
  interval = TRUE, interval_type = c("prediction", "confidence"),
  level = NULL, ncol = NULL, main = NULL,
  xlab = "Substrate concentration", ylab = "Reaction velocity", ...)
```

Arguments

`data`	A data frame containing Michaelis-Menten observations.
`s`	Name of the substrate concentration column.
`v`	Name of the response column.
`groups`	Optional character vector of grouping columns.
`variance_models`	Character vector of built-in variance models. The special value `"auto"` adds a candidate that compares `log(S + 1)` with a grid of power functions and keeps the better fit.
`power_values`	Optional numeric vector of fixed exponents for additional `S^power` candidates.
`include_auto`	Logical; if `TRUE`, add one automatic candidate that selects between `log(S + 1)` and the supplied `power_grid`.
`power_selection`	Criterion used by the automatic candidate: `"quasi_aic"` or `"quasi_bic"`.
`power_grid`	Candidate exponents used by the automatic candidate.
`interval_level`	Confidence level forwarded to `fit_mm`.
`invalid_response_values`	Values to remove before fitting.
`invalid_substrate_values`	Values to remove before fitting.
`allow_zero_substrate`	Logical; if `FALSE`, substrate concentrations must be strictly positive.
`sort_x`	Logical; sort observations within each group by concentration.
`quiet`	Logical; if `TRUE`, suppress progress messages.
`variance_functions`	Optional named list of fixed, vectorized positive functions $h(S)$, forwarded to `screen_mm` within every group. Use `variance_models = character(0)` for custom-only comparisons.
`x`	A grouped fit object for the print and plot methods.

which_groups	Optional subset of groups to plot, matched against group_id or group_label.
model	Optional model name to use for every group. When NULL, the plotting method uses the best model recorded for each group.
interval	Logical; if TRUE, add interval bands to each panel.
interval_type	Either "prediction" or "confidence".
level	Optional interval level. Defaults to the level stored in each fit.
ncol	Optional number of panel columns in the grouped display.
main	Optional panel title or titles.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
digits	Number of printed digits.
...	Additional arguments passed to S3 methods.

Details

Custom variance functions use the same definition in every group and must obey the contract in variance_function. Group-specific scales are estimated, but shape parameters in the functions are treated as fixed. Ranking criteria and intervals omit uncertainty from external shape estimation or model selection. This is a set of separate single-curve fits, not a clustered model.

Value

An object of class "group_mm" containing cleaned data, fit summaries, augmented residual output, grouped rankings, and fitted objects.

Examples

```
grouped <- group_mm(
  data = sdl_demo,
  s = "s_uM",
  v = "v_uM_per_min",
  groups = "enzyme",
  variance_models = c("constant", "sqrt"),
  power_values = 0.4,
  include_auto = TRUE,
  quiet = TRUE
)

grouped
head(
  grouped$comparison$best_by_group[
    ,
    c("group_label", "model", "selected_model", "quasi_aic", "quasi_bic", "rmse")
  ]
)

plot(grouped, interval_type = "confidence")
```

report_mm *Print and Plot a Michaelis-Menten Fit*

Description

Convenience wrapper that prints a fitted-model summary and draws the fitted curve with a confidence or prediction band. Higher-moment fits support pointwise mean-confidence bands, not prediction bands, because their working moments do not specify predictive quantiles.

Usage

```
report_mm(object, level = object$interval_level,
  method = c("wald", "bootstrap"), B = 399L,
  bootstrap_type = c("wild", "pairs"),
  bootstrap_ci = c("studentized", "basic", "percentile"),
  wild_weights = c("mammen", "rademacher"), seed = NULL,
  interval_type = c("confidence", "prediction", "none"),
  digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

object	A fitted object returned by fit_mm, fit_mm_moments, or cluster_mm.
level	Confidence level used in the printed summary and plot.
method	Interval method used in the printed summary: "wald" or "bootstrap".
B	Number of bootstrap resamples when method = "bootstrap".
bootstrap_type	Bootstrap scheme used when method = "bootstrap": "wild" or "pairs". Higher-moment fits default to and require "pairs".
bootstrap_ci	Bootstrap interval calibration used when method = "bootstrap": "studentized", "basic", or "percentile".
wild_weights	Wild-bootstrap multipliers used when bootstrap_type = "wild": "mammen" or "rademacher".
seed	Deprecated and ignored. Call set.seed() before bootstrap-based summaries if reproducibility is needed.
interval_type	Character string indicating whether the plotted band should be "confidence", "prediction", or "none".
digits	Number of printed digits.
...	Additional graphical arguments passed to plot.

Value

The original fitted object, returned invisibly. When method = "bootstrap" and the object inherits from "fit_mm", the printed summary uses bootstrap standard errors and confidence intervals; the plotted band remains the usual pointwise confidence or prediction band from plot.fit_mm(). For very small curves (fewer than 20 observations or fewer than 6 distinct substrate concentrations),

the printed summary reports point estimates only by default and leaves interval construction to an explicit `confint()` call. For sparse clustered fits, the printed summary similarly suppresses interval output by default and treats `confint()` as an explicit sensitivity analysis; bootstrap summaries are not currently available for `"cluster_mm"` objects.

Examples

```
one_curve <- subset(sdl_demo, enzyme == "1111")
fit <- fit_mm(
  x = one_curve$s_uM,
  y = one_curve$v_uM_per_min,
  variance = "sqrt"
)


report_mm(fit, interval_type = "confidence")
set.seed(1)
report_mm(
  fit,
  method = "bootstrap",
  B = 99,
  bootstrap_ci = "studentized",
  wild_weights = "mammen",
  interval_type = "confidence"
)

cluster_fit <- cluster_mm(
  data = subset(alves_demo, enzyme == "BG"),
  s = "substrate_conc",
  v = "activity",
  cluster = "core",
  variance = "sqrt"
)
report_mm(cluster_fit, interval_type = "confidence")
```

`screen_mm` *Screen Working Variance Models*

Description

Fit several working variance models to the same Michaelis-Menten curve and order them primarily by quasi-AIC, while also reporting quasi-BIC, RMSE, and weighted residual sums of squares for reference.

Usage

```
screen_mm(x, y, variance_models = default_variance_models(),
  power_values = NULL, include_auto = FALSE,
  power_selection = c("quasi_aic", "quasi_bic"),
```

```
  power_grid = default_power_grid(), interval_level = 0.95,
  allow_zero_substrate = TRUE, km_bounds = NULL, quiet = FALSE,
  variance_functions = NULL)

## S3 method for class 'screen_mm'
print(x, digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

x	Numeric vector of substrate concentrations, or a screening object for the print method.
y	Numeric vector of observed reaction velocities.
variance_models	Character vector of built-in variance models to compare. The special value "auto" adds a candidate that compares log(S + 1) with a grid of power functions and keeps the better fit.
power_values	Optional numeric vector of fixed exponents for additional S^power candidates.
include_auto	Logical; if TRUE, add one automatic candidate that selects between log(S + 1) and the supplied power_grid.
power_selection	Criterion used by the automatic candidate: "quasi_aic" or "quasi_bic".
power_grid	Candidate exponents used by the automatic candidate.
interval_level	Confidence level forwarded to fit_mm.
allow_zero_substrate	Logical; if FALSE, substrate concentrations must be strictly positive.
km_bounds	Optional search bounds for K_m.
quiet	Logical; if TRUE, suppress progress messages.
variance_functions	Optional named list of fixed, vectorized positive functions $h(S)$. Names must be unique, non-empty, and distinct from built-in candidate names. Set variance_models = character(0) to compare only these custom candidates.
digits	Number of printed digits.
...	Additional arguments passed to S3 methods.

Details

Custom functions follow the contract in variance_function and are fitted by fit_mm. Their names identify rows in the ranking table and entries in fits. A candidate whose function fails validation is recorded with success = 0 and an error_message; other candidates are still evaluated. Malformed or duplicate candidate names cause an error. Custom functions are added to the outer comparison; they do not change the log/power search performed by include_auto.

All custom shapes are treated as fixed and add no estimated shape parameters. Criteria and intervals do not account for externally estimated function coefficients or for selection uncertainty. Adding more candidates does not guarantee better estimation or interval coverage.

Value

An object of class "screen_mm" with a fitted-model table and the successful fits.

Examples

```
one_curve <- subset(sdl_demo, enzyme == "1111")
screen <- screen_mm(
  x = one_curve$s_uM,
  y = one_curve$v_uM_per_min,
  power_values = c(0.4, 0.6),
  include_auto = TRUE,
  quiet = TRUE
)

screen$table[, c("model", "selected_model", "quasi_aic", "quasi_bic", "rmse")]

custom_screen <- screen_mm(
  one_curve$s_uM, one_curve$v_uM_per_min,
  variance_models = c("constant", "sqrt"),
  variance_functions = list(
    linear_floor = function(S) 1 + S / 20,
    saturation = function(S) 1 + 9 * S / (20 + S)
  ), quiet = TRUE
)
custom_screen$table[, c("model", "quasi_aic", "success")]
```

---

sdl_demo *Self-Driving Laboratory Michaelis-Menten Demo Data*

Description

A lightweight self-driving-laboratory enzyme-kinetic dataset used in the package examples and vignette.

Format

A data frame with 168 rows and 5 variables:

`enzyme` Enzyme identifier.

`s_uM` Substrate concentration in micromolar units.

`replicate` Replicate index.

`v_uM_per_s` Velocity in micromolar units per second.

`v_uM_per_min` Velocity in micromolar units per minute.

Source

Prepared from the SDL study files in the project workspace.

Examples

```
head(sdl_demo)
table(sdl_demo$enzyme)
```

`simulate_mm_data` *Simulate Michaelis-Menten Data*

Description

Generate a synthetic Michaelis-Menten dataset with bounded heteroscedastic variance and Gaussian, centered gamma, or standardized Student-t errors.

Usage

```
simulate_mm_data(n = 100, x = seq(1, 100, length.out = n),
  vmax = 100, km = 20, variance_shape = c("mm", "exp", "hill"),
  error = c("normal", "skewed", "student_t"), s0 = 1, t2 = 9, c = 20,
  a = 0.05, alpha = 2, skew_shape = 4, seed = NULL, df = 10)
```

Arguments

`n`	Number of observations when `x` is not supplied.
`x`	Optional substrate concentration grid.
`vmax`	True $V_{\max}$ value.
`km`	True K_m value.
`variance_shape`	Character string naming the true variance pattern.
`error`	Character string naming the error distribution.
`s0`, `t2`, `c`, `a`, `alpha`	Variance-shape parameters.
`skew_shape`	Shape parameter for the skewed gamma-based error.
`seed`	Deprecated and ignored. Call `set.seed()` before `simulate_mm_data()` if reproducibility is needed.
`df`	Degrees of freedom greater than four for Student-t errors, ensuring a finite fourth moment. Ignored for other error choices.

Details

The standardized error has mean zero and variance one. For `"skewed"`, it is $(G - a)/\sqrt{a}$ with $G \sim \mathrm{Gamma}(a, 1)$ (shape a=`skew_shape`, scale one). For `"student_t"`, it is $T\sqrt{(\nu - 2)/\nu}$ with $T \sim t_\nu$. These are generating distributions, not options that select the fitting algorithm. See `fit_mm_moments` for higher-moment estimation.

Value

A data frame containing the simulated observations.

Examples

```
set.seed(1)
sim_dat <- simulate_mm_data(variance_shape = "hill", error = "skewed")
head(sim_dat)
heavy_tail_dat <- simulate_mm_data(error = "student_t", df = 10)
```

`variance_function` *Create Working Variance Functions*

Description

Construct a working variance function for variance-aware Michaelis-Menten fitting.

Usage

```
variance_function(model = c("constant", "log", "sqrt", "cuberoot", "power"),
  power = NULL, label = NULL)
```

Arguments

`model`	A built-in model name or a vectorized function of concentration returning one finite, strictly positive numeric variance-shape value per input.
`power`	Optional exponent used when `model = "power"`.
`label`	Optional non-empty descriptive label for a user-defined function.

Details

A custom function represents a fixed shape $h(S)$ in $\mathrm{Var}(Y \mid S) = \gamma h(S)$. It must be deterministic, pointwise, and vectorized: a value at a concentration must not depend on other elements of the input vector. It must return a numeric vector of the same length as its input, not a scalar recycled across observations. Values must be strictly positive and finite at all fitting and prediction concentrations, including zero if it is used. Invalid custom outputs cause an error; they are not clipped. For example, use `function(S) rep(1, length(S))` for a constant shape.

All constants captured by a custom function must be fixed, not parameters to be estimated by `fit_mm()`. Keep its enclosing environment unchanged between fitting and prediction/refitting; use a local closure to capture external constants when needed. `label` describes the function but does not change its values. Custom functions are supported by `fit_mm()`, `screen_mm()`, and `group_mm()`, not currently by `cluster_mm()`.

Value

A function of substrate concentration `x`.

Examples

```
h_log <- variance_function("log")
h_sqrt <- variance_function("sqrt")
h_log(c(0, 5, 10))
h_sqrt(c(0, 5, 10))
h_custom <- variance_function(function(S) 1 + S / 20,
                              label = "Var = gamma (1 + S/20)")
h_custom(c(0, 5, 10))
```

Index

∗ **package**
    inferMM-package, 2

alves_demo, 3

cluster_mm, *2*, 3, *19*
coef.cluster_mm *(*cluster_mm*)*, 3
coef.fit_mm *(*fit_mm*)*, 8
confint.cluster_mm *(*cluster_mm*)*, 3
confint.fit_mm *(*fit_mm*)*, 8
confint.fit_mm_moments
        *(*fit_mm_moments*)*, 12

fit_mm, *2*, 8, *15*, *17*, *19*, *21*
fit_mm_moments, *2*, *11*, 12, *19*, *23*

group_mm, *2*, 16

inferMM *(*inferMM-package*)*, 2
inferMM-package, 2

plot, *19*
plot.cluster_mm *(*cluster_mm*)*, 3
plot.fit_mm *(*fit_mm*)*, 8
plot.fit_mm_moments *(*fit_mm_moments*)*, 12
plot.group_mm *(*group_mm*)*, 16
predict.cluster_mm *(*cluster_mm*)*, 3
predict.fit_mm *(*fit_mm*)*, 8
predict.fit_mm_moments
        *(*fit_mm_moments*)*, 12
print.cluster_mm *(*cluster_mm*)*, 3
print.fit_mm *(*fit_mm*)*, 8
print.fit_mm_moments *(*fit_mm_moments*)*,
        12
print.group_mm *(*group_mm*)*, 16
print.screen_mm *(*screen_mm*)*, 20
print.summary.cluster_mm *(*cluster_mm*)*, 3
print.summary.fit_mm *(*fit_mm*)*, 8
print.summary.fit_mm_moments
        *(*fit_mm_moments*)*, 12

report_mm, *2*, 19

screen_mm, *2*, *17*, 20
sdl_demo, 22
simulate_mm_data, *2*, 23
summary.cluster_mm *(*cluster_mm*)*, 3
summary.fit_mm *(*fit_mm*)*, 8
summary.fit_mm_moments
        *(*fit_mm_moments*)*, 12

variance_function, *2*, *9*, *11*, *13*, *18*, *21*, 24
vcov.cluster_mm *(*cluster_mm*)*, 3
vcov.fit_mm *(*fit_mm*)*, 8