Package 'admixr2'

September 16, 2026

Type Package

Title Aggregate Data Modelling

Version 0.4.1

Description
Fit pharmacokinetic/pharmacodynamic (PK/PD) models to aggregate-level data (mean vector and covariance matrix per study) rather than individual-level data, for meta-analysis across studies. Integrates with the 'nlmixr2'/'rxode2' ecosystem via four estimation methods: a First-Order ('FO') analytical estimator, a Monte Carlo (MC) estimator, a Gauss-Hermite quadrature ('GH') estimator, and an Iterative Reweighting Monte Carlo ('IRMC') estimator. Methods are based on Välitalo (2021) <doi:10.1007/s10928-021-09760-1>; software described in van de Beek et al. (2025) <doi:10.1007/s10928-025-10011-w>.

License GPL (>= 3)

URL https://leidenpharmacology.github.io/admixr2/, https://github.com/LeidenPharmacology/admixr2

BugReports https://github.com/LeidenPharmacology/admixr2/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports checkmate, digest, nlmixr2est (>= 6.0.1), nloptr, randtoolbox, Rcpp, rxode2 (>= 5.1.2), symengine

LinkingTo Rcpp, RcppEigen

Suggests expm, ggplot2, knitr, lotri, MASS, memuse, mirai, mnormt, nlmixr2 (>= 5.0.0), nlmixr2data, numDeriv, patchwork, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author H. van de Beek [aut, cre],
P.A.J. Välitalo [aut],
L.B. Zwep [aut],
J.G.C. van Hasselt [aut]

Maintainer H. van de Beek <h.van.de.beek@lacdr.leidenuniv.nl>

Repository CRAN

Date/Publication 2026-09-16 07:10:08 UTC

Contents

`adfoControl` *Control settings for the FO (First-Order) estimator*

Description

Creates a control object for `nlmixr2(est = "adfo")`. The FO estimator linearises model predictions at $\eta = 0$: it is faster than the MC estimator but less accurate for models with large IIV or strongly non-linear individual predictions.

Usage

```
adfoControl(
  studies = list(),
  grad = c("analytical", "none", "fd"),
  algorithm = NULL,
  maxeval = 500L,
  ftol_rel = .Machine$double.eps^(1/2),
```

```
  print = 10L,
  seed = 12345L,
  cores = rxode2::rxCores(),
  nDisplayProgress = .Machine$integer.max,
  grad_h = 1e-04,
  grad_bounds = 5,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/5),
  covMethod = c("r,s", "r", "none"),
  n_restarts = 1L,
  restart_sd = 0.5,
  workers = 1L,
  rxControl = NULL,
  calcTables = FALSE,
  compress = TRUE,
  ci = 0.95,
  sigdig = NULL,
  sigdigTable = NULL,
  addProp = c("combined2", "combined1"),
  optExpression = TRUE,
  sumProd = FALSE,
  literalFix = TRUE,
  returnAdmr = FALSE,
  resid_nodes = 81L,
  ...
)
```

Arguments

studies	Named list of study specifications (same format as admControl(): E, V, n, times, ev, optional method; or an observations list for multi-compartment fits – see admControl()).
grad	Gradient mode. "analytical" (default) uses the closed-form FO gradient with LBFGS; "none" uses derivative-free BOBYQA; "fd" uses central finite differences of the full NLL. Forward differencing was removed in 0.4.1 – it was 10^2 to 10^4 times less accurate than a central difference at every site measured, and the one solve per parameter it saved did not pay for a gradient the optimizer struggles to descend. The default was "none" up to 0.4.0, because the structural thetas were finite-differenced through the whole NLL and the resulting gradient was too noisy for a quasi-Newton step to pay off. They are now differentiated analytically from a second-order sensitivity model (relative error ~1e-7 against a central difference, where the finite-difference pass reached 1e-2), so LBFGS on the exact gradient is the better default. A model that cannot build that sensitivity model falls back to the finite-difference gradient automatically, and grad = "none" remains available.
algorithm	nloptr algorithm, or NULL (default) to pick the default that matches grad: "NLOPT_LD_LBFGS" with a gradient, "NLOPT_LN_BOBYQA" when grad = "none". Any algorithm re-

ported by nloptr::nloptr.print.options() is accepted. An explicit algorithm is reconciled with grad: when grad = "none" a gradient-based algorithm (NLOPT_LD_* / NLOPT_GD_*) falls back to "NLOPT_LN_BOBYQA"; when a gradient is requested a derivative-free algorithm (NLOPT_LN_* / NLOPT_GN_*) turns the gradient off. Both emit a message.

maxeval — Maximum function evaluations (default 500).

ftol_rel — Relative tolerance (default sqrt(.Machine$double.eps)).

print — Print-frequency for live progress (0 = silent).

seed — Random seed (used for restarts).

cores — OpenMP threads for rxSolve(). Defaults to rxode2::rxCores(). When workers > 1 it is a *total* budget, split across the workers.

nDisplayProgress — Passed to rxSolve(): show the solver's text progress bar only once a single solve exceeds this many subjects. The default (.Machine$integer.max) keeps it off for clean script/vignette output; lower it (e.g. 1000L) to see progress during long fits.

grad_h — Finite-difference step for unpaired struct theta gradient and FD Jacobian.

grad_bounds — Box-constraint half-width when using gradients: the fit is confined to p0 +/- grad_bounds on the optimizer scale, which for a log-scale parameter is a factor of exp(grad_bounds) (~148 at the default 5). This bound is admixr2's, not the model's – an unbounded parameter has no other – and nloptr reports normal convergence at a box corner, so a warning is emitted if an estimate finishes on it.

cov_h — Inner FD step for the gradient-based Hessian (only used when covMethod = "r" and grad != "none"). Default 1e-3.

cov_h_outer — Outer step scale for NLL-FD Hessian.

covMethod — "r,s" (the DEFAULT) computes the sandwich H^-1 J H^-1; "r" the numerical Hessian alone, 2H^-1; "none" skips the covariance. All three span the structural, residual-error and omega parameters. Omega is included because excluding it also biases the STRUCTURAL standard errors downward – a theta carrying an eta is correlated with that eta's variance. If the weakly-identified omega Cholesky makes the Hessian non-positive definite, the structural + residual sub-block is reported with a warning.

"r,s" adds a sandwich correction, H^-1 J H^-1, on the same Hessian. For FO this does more than correct kurtosis: V = J Omega J' + Sigma is the covariance of an exactly normal individual law, so the reported standard errors otherwise answer to the linearisation rather than to the model. The correction scores the FO fit against the model's true nonlinear law, built post-fit on a quadrature ensemble, and so absorbs part of the linearisation error as well. Point estimates are untouched. **Transform-both-sides endpoints.** adfo composes the residual by a second-order expansion about the linearised moments, because FO carries no node ensemble to compose over – that is what the method is. adgh and admc compose exactly at their nodes/draws, so an adfo fit of a boxCox, yeoJohnson, logitNorm or probitNorm endpoint differs from theirs by the expansion's truncation: roughly 0.3% in V at moderate between-subject variability, rising to ~3% for a tightly-bounded logit/probit at high variability. That is a property of the estimator, not a discrepancy.

It is the default because it is the conservative choice, not the aggressive one. Under correct specification `J = 2H` and the sandwich returns what `"r"` returns, so defaulting to it costs nothing when the normal-theory assumption holds and corrects the standard errors when it does not. Anything it cannot build degrades to `"r"` and reports `"r"`, so no fit loses its covariance by asking. Pass `covMethod = "r"` for the pre-0.4.1 behaviour.

Applies to every residual family whose conditional law is independent across timepoints, which is all of them except `ar()`: the conditionally-normal set (`add`, `prop`, `pow`, `combined1`, `combined2`), the closed-form distributional ones (`lnorm`, `pois`, `binom`, `nbinomMu`, `beta`, and `t()` with `nu > 4`), and the transform-both-sides ones (`boxCox`, `yeoJohnson`, `logitNorm`, `probitNorm`), whose third and fourth conditional moments come off the same quadrature that already gives their mean and variance. Refused, and degraded to `"r"`: `ar()`, because it correlates the residual ACROSS timepoints and the cross terms the expansion drops are then real; `t()` with `nu <= 4`, whose kurtosis does not exist; and `ordinal()` and same-subject `joint` studies, which stack several outputs into one covariance the per-output node ensemble does not describe. These four are refusals by construction rather than failures, so the fit reports the reason as a message and falls back to `"r"`; a sandwich that was attempted and could not be built still warns.

`"r,s"` **is more sensitive to an ill-conditioned Hessian than** `"r"` **is.** `"r"` reports `2H^-1` and inverts `H` once; the sandwich reports `H^-1 J H^-1` and inverts it twice, so in a direction the data barely identifies any gap between `J` and `2H` is amplified quadratically. A residual SD contributing 0.01 variance against 1.7 from between-subject variability is such a direction: measured on one 1-cmt fixture at `cond(H) = 3.5e5`, the reported residual SE moved by a factor of 0.11 and two omega entries by 0.59 and 1.55, while the same model and design on a study the residual IS identified in (`cond(H) = 247`) reproduced `"r"` to four decimals on every parameter. Neither number is a correction there – both methods are reporting an unidentified direction, and `"r,s"` is louder about it. admixr2 says so: when the Hessian's reciprocal condition number falls below `eps^(1/4)` – the point at which squaring the conditioning reaches the bound a single inversion is already called singular at – the fit records a note naming the parameter that loads most heavily on the offending direction. It arrives on `fit$runInfo` and is listed by `print(fit)`, which is where `nlmixr2est` routes an estimator's warnings. The sandwich is still reported, because the well-determined parameters of the same fit are unaffected; check the named parameter's relative standard error before reading its `"r,s"` value as a finding.

All three blocks are reported on the scale the ESTIMATES are printed on, as `nlmixr2est` does: structural thetas on the log/optimizer scale, residual error as an SD, and omega as the variance/covariance entries (named `om.<eta>` and `cov.<eta_i>.<eta_j>`). The omega block is rotated by the full Jacobian of Omega with respect to the log-Cholesky, which is not diagonal once omega is correlated.

An adfo standard error describes scatter, not accuracy. FO linearises the model at eta = 0, and on a non-additive residual (or a saturating endpoint, or a large omega) the resulting point estimates carry a bias of several standard errors – measured 5-20 SE, giving 0% coverage for a nominal 95% interval even where the SE itself matches the sampling SD. Use `adgh` or `admc` when the uncertainty

matters.

`n_restarts` Number of optimizer restarts (1 = no multi-start).

`restart_sd` Standard deviation for random perturbations of initial struct thetas at each restart (> 1).

`workers` Number of parallel workers (mirai daemons) for multi-restart (default 1 = sequential). Requires the `mirai` package.

`rxControl` `rxode2::rxControl()` object. Created automatically when `NULL`.

`calcTables, compress, ci, sigdigTable, optExpression, sumProd, literalFix`

Passed to `nlmixr2est::foceiControl()` for the table/output machinery.

`sigdig` Significant digits asked of the ODE solver, or `NULL` (the default) to leave rxode2's own solver tolerances alone. When set, it is passed to `rxode2::rxSolve()`'s own `sigdig` argument for every solve the estimator issues – rxode2 owns the mapping to `atol`/`rtol` and has changed it between releases, which is why the digits, not the tolerances, are what travels – and to `nlmixr2est::foceiControl()` for the post-fit tables.

It is a speed lever, and an opt-in one because it is not free. The estimators finite-difference the solve with steps of the same order: `grad_h` (1e-4), `cov_h` (1e-3) and `cov_h_outer` (~2.5e-3), while `sigdig = 4` maps to a relative tolerance of ~1e-4 on current rxode2. Differencing a solution whose own noise is 1e-4 with a 1e-4 step returns noise, and it surfaces as a moved objective and an indefinite covariance Hessian (every `SE` reported `NA`) rather than as an error. Most worthwhile where the gradient is fully analytic and nothing differences the solve – `adfoControl(grad = "analytical")` measured ~4.8x faster at `sigdig = 4` with standard errors unchanged to 4 significant figures. Elsewhere, compare the objective and the standard errors against `NULL` before relying on it. Table formatting is unaffected either way: `sigdigTable` defaults to 4 regardless.

`addProp` How combined additive+proportional error is parameterised in the nlmixr2 output tables: `"combined2"` (default, variance form) or `"combined1"` (SD form). Has no effect on admixr2's own estimation.

`returnAdmr` If `TRUE`, return a plain list instead of the full nlmixr2 fit object.

`resid_nodes` Gauss-Hermite nodes used to integrate the RESIDUAL for a transform-both-sides endpoint (`boxCox`, `yeoJohnson`, `logitNorm`, `probitNorm`), where `y = g(h(f) + sigma*eps)` has no closed-form mean and variance. Ignored by every other error model, which has closed forms. Default 81. Measured worst-case relative error against an independent quadrature, over all four transforms and residual SD of 0.5, 1, 2 and 3: n = 15 gives 5.7e-2, 31 gives 4.5e-3, 81 gives 5.0e-5. The error is dominated by large residual SD; at SD <= 1, n = 31 already gives 1e-7 or better.

This is an ACCURACY dial, not a speed one. The quadrature is linear in `resid_nodes` in isolation (~50 us at 15, 300 us at 81 for an 8-row study) but negligible beside the ODE solve: a full NLL evaluation measured 0.750 s per 60 evaluations at BOTH 31 and 81 nodes. Raise it if you have a saturating endpoint with a large residual SD; there is little to gain by lowering it.

`...` Unused arguments (trigger an error).

Value

An `adfoControl` object (a named list).

Installing memuse

`rxode2::rxSolve()` estimates free RAM on every call. When the `memuse` package is not installed its fallback ends up shelling out to `vm_stat`, a macOS-only command, so on Windows and Linux every solve spawns a process that can only fail. Because the FO estimator issues many small solves, this overhead is measurable (roughly 17% of an FO gradient). Installing `memuse` makes the fallback unreachable:

```
install.packages("memuse")
```

See Also

`admControl()`, `adirmcControl()`

Examples

```
# Inspect defaults
ctl <- adfoControl()
ctl$grad
ctl$maxeval

# Analytical gradient, more evaluations
ctl2 <- adfoControl(grad = "analytical", maxeval = 1000L)


library(rxode2)
library(nlmixr2)

data("examplomycin")
obs    <- examplomycin[examplomycin$EVID == 0, ]
obs    <- obs[order(obs$ID, obs$TIME), ]
times  <- sort(unique(obs$TIME))
ids    <- unique(obs$ID)
dv_mat <- do.call(rbind, lapply(ids, function(i) {
  sub <- obs[obs$ID == i, ]; sub$DV[order(sub$TIME)]
}))
E <- colMeans(dv_mat)
V <- cov.wt(dv_mat, method = "ML")$cov

pk_model <- function() {
  ini({
    tcl <- log(5); tv <- log(30)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v ~ 0.04
  })
  model({
    cl <- exp(tcl + eta.cl)
    v  <- exp(tv  + eta.v)
```

```
    d/dt(central) <- -(cl/v) * central
    cp <- central / v
    cp ~ prop(prop.sd)
  })
}

fit <- nlmixr2(
  pk_model, admData(), est = "adfo",
  control = adfoControl(
    studies = list(study1 = list(E = E, V = V, n = length(ids),
                                 times = times, ev = et(amt = 100))),
    maxeval = 100L
  )
)
print(fit)
```

---

adghControl *Control settings for the Gauss-Hermite (GH) quadrature estimator*

Description

Creates a control object for `nlmixr2(est = "adgh")`. The GH estimator integrates model predictions against the random-effects prior $\eta \sim N(0, \Omega)$ using a deterministic tensor-product Gauss-Hermite quadrature grid. It is unbiased at any IIV magnitude (unlike FO), noise-free (unlike MC), and much faster than MC for models with up to ~4 etas.

Usage

```
adghControl(
  studies = list(),
  n_nodes = 5L,
  grad = c("analytical", "fd", "none"),
  algorithm = NULL,
  maxeval = 500L,
  ftol_rel = .Machine$double.eps^(1/2),
  print = 10L,
  seed = 12345L,
  cores = rxode2::rxCores(),
  nDisplayProgress = .Machine$integer.max,
  grad_h = 1e-04,
  grad_bounds = 5,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/4),
  covMethod = c("r,s", "r", "none"),
  n_restarts = 1L,
  restart_sd = 0.5,
```

```
  workers = 1L,
  rxControl = NULL,
  calcTables = FALSE,
  compress = TRUE,
  ci = 0.95,
  sigdig = NULL,
  sigdigTable = NULL,
  addProp = c("combined2", "combined1"),
  optExpression = TRUE,
  sumProd = FALSE,
  literalFix = TRUE,
  returnAdmr = FALSE,
  resid_nodes = 81L,
  ...
)
```

Arguments

studies	Named list of study specifications (same format as admControl(): E, V, n, times, ev, optional method; or an observations list for multi-compartment fits – see admControl()).
n_nodes	Number of quadrature nodes per eta dimension (default 5). For a transform-both-sides endpoint (boxCox, yeoJohnson, logitNorm, probitNorm) this also controls the accuracy of the RESIDUAL composition: those endpoints have a conditional mean that is nonlinear in the structural prediction, so the residual is composed at each node and aggregated rather than expanded about the ensemble mean. Before that, n_nodes had no effect at all on a TBS fit's accuracy – the expansion's error was a floor no node count removed. Total nodes = n_nodes^n_eta. n_nodes = 5 achieves near-exact covariance moments for IIV SD up to ~0.5; n_nodes = 7 extends coverage to SD ~0.7. For models with >= 5 etas the node count grows steeply; consider reducing n_nodes or using a different estimator.
grad	Gradient mode. "analytical" (default) uses closed-form contractions through the sensitivity equations – cheapest and exact. "fd" uses central finite differences (forward differencing was removed in 0.4.1; see adfoControl()). "none" uses derivative-free BOBYQA.
algorithm	nloptr algorithm, or NULL (default) to pick the default that matches grad: "NLOPT_LD_LBFGS" with a gradient, "NLOPT_LN_BOBYQA" when grad = "none". Any algorithm reported by nloptr::nloptr.print.options() is accepted. An explicit algorithm is reconciled with grad: when grad = "none" a gradient-based algorithm (NLOPT_LD_* / NLOPT_GD_*) falls back to "NLOPT_LN_BOBYQA"; when a gradient is requested a derivative-free algorithm (NLOPT_LN_* / NLOPT_GN_*) turns the gradient off. Both emit a message.
maxeval	Maximum function evaluations (default 500).
ftol_rel	Relative tolerance (default sqrt(.Machine$double.eps)).
print	Print-frequency for live progress (0 = silent).
seed	Random seed (used for restarts).

cores — OpenMP threads for rxSolve(). Defaults to rxode2::rxCores(). When workers > 1 it is a *total* budget, split across the workers.

nDisplayProgress — Passed to rxSolve(): show the solver's text progress bar only once a single solve exceeds this many subjects. The default (.Machine$integer.max) keeps it off for clean script/vignette output; lower it (e.g. 1000L) to see progress during long fits.

grad_h — Finite-difference step for unpaired struct theta gradient and FD Jacobian fallback.

grad_bounds — Box-constraint half-width when using gradients: the fit is confined to p0 +/- grad_bounds on the optimizer scale, which for a log-scale parameter is a factor of exp(grad_bounds) (~148 at the default 5). This bound is admixr2's, not the model's – an unbounded parameter has no other – and nloptr reports normal convergence at a box corner, so a warning is emitted if an estimate finishes on it.

cov_h — Inner FD step for the gradient-based Hessian (only used when covMethod = "r" and grad != "none").

cov_h_outer — Outer step scale for numerical Hessian. Default eps^(1/4) (tighter than admc's eps^(1/5) because the GH surface is noise-free).

covMethod — "r,s" (the DEFAULT) computes the sandwich H^-1 J H^-1; "r" the numerical Hessian alone, 2H^-1; "none" skips the covariance. All three span the structural, residual-error and omega parameters. Omega is included because excluding it also biases the STRUCTURAL standard errors downward – a theta carrying an eta is correlated with that eta's variance. If the weakly-identified omega Cholesky makes the Hessian non-positive definite, the structural + residual sub-block is reported with a warning.

"r,s" adds a sandwich correction, H^-1 J H^-1, on the same Hessian. The aggregate objective scores the reported mean and covariance as though the subjects behind them were multivariate normal; they are not, because the model is nonlinear in the random effects, so the sampling law of (E, V) is not the one the objective assumes. "r,s" scores that law from the model instead. Point estimates are untouched – only the reported uncertainty changes – and under correct specification it reduces to "r" exactly. **It is the default because it is the conservative choice, not the aggressive one.** Under correct specification J = 2H and the sandwich returns what "r" returns, so defaulting to it costs nothing when the normal-theory assumption holds and corrects the standard errors when it does not. Anything it cannot build degrades to "r" and reports "r", so no fit loses its covariance by asking. Pass covMethod = "r" for the pre-0.4.1 behaviour.

Applies to every residual family whose conditional law is independent across timepoints, which is all of them except ar(): the conditionally-normal set (add, prop, pow, combined1, combined2), the closed-form distributional ones (lnorm, pois, binom, nbinomMu, beta, and t() with nu > 4), and the transform-both-sides ones (boxCox, yeoJohnson, logitNorm, probitNorm), whose third and fourth conditional moments come off the same quadrature that already gives their mean and variance. Refused, and degraded to "r": ar(), because it correlates the residual ACROSS timepoints and the cross terms the expansion drops are then real; t() with nu <= 4, whose kurtosis does not exist; and ordinal() and same-subject joint studies, which stack several outputs into one covariance

the per-output node ensemble does not describe. These four are refusals by construction rather than failures, so the fit reports the reason as a message and falls back to "r"; a sandwich that was attempted and could not be built still warns.

"r,s" **is more sensitive to an ill-conditioned Hessian than** "r" **is.** "r" reports 2H^-1 and inverts H once; the sandwich reports H^-1 J H^-1 and inverts it twice, so in a direction the data barely identifies any gap between J and 2H is amplified quadratically. A residual SD contributing 0.01 variance against 1.7 from between-subject variability is such a direction: measured on one 1-cmt fixture at cond(H) = 3.5e5, the reported residual SE moved by a factor of 0.11 and two omega entries by 0.59 and 1.55, while the same model and design on a study the residual IS identified in (cond(H) = 247) reproduced "r" to four decimals on every parameter. Neither number is a correction there – both methods are reporting an unidentified direction, and "r,s" is louder about it. admixr2 says so: when the Hessian's reciprocal condition number falls below eps^(1/4) – the point at which squaring the conditioning reaches the bound a single inversion is already called singular at – the fit records a note naming the parameter that loads most heavily on the offending direction. It arrives on fit$runInfo and is listed by print(fit), which is where nlmixr2est routes an estimator's warnings. The sandwich is still reported, because the well-determined parameters of the same fit are unaffected; check the named parameter's relative standard error before reading its "r,s" value as a finding.

All three blocks are reported on the scale the ESTIMATES are printed on, as nlmixr2est does: structural thetas on the log/optimizer scale, residual error as an SD, and omega as the variance/covariance entries (named om.<eta> and cov.<eta_i>.<eta_j>). The omega block is rotated by the full Jacobian of Omega with respect to the log-Cholesky, which is not diagonal once omega is correlated.

`n_restarts` Number of optimizer restarts (1 = no multi-start).

`restart_sd` SD of random perturbations of initial struct thetas at each restart.

`workers` Number of parallel workers (mirai daemons) for multi-restart (default 1 = sequential). Requires the mirai package.

`rxControl` rxode2::rxControl() object. Created automatically when NULL.

`calcTables, compress, ci, sigdigTable, optExpression, sumProd, literalFix`

Passed to nlmixr2est::foceiControl() for the table/output machinery.

`sigdig` Significant digits asked of the ODE solver, or NULL (the default) to leave rxode2's own solver tolerances alone. When set, it is passed to rxode2::rxSolve()'s own sigdig argument for every solve the estimator issues – rxode2 owns the mapping to atol/rtol and has changed it between releases, which is why the digits, not the tolerances, are what travels – and to nlmixr2est::foceiControl() for the post-fit tables.

It is a speed lever, and an opt-in one because it is not free. The estimators finite-difference the solve with steps of the same order: grad_h (1e-4), cov_h (1e-3) and cov_h_outer (~2.5e-3), while sigdig = 4 maps to a relative tolerance of ~1e-4 on current rxode2. Differencing a solution whose own noise is 1e-4 with a 1e-4 step returns noise, and it surfaces as a moved objective and

an indefinite covariance Hessian (every SE reported NA) rather than as an error. Most worthwhile where the gradient is fully analytic and nothing differences the solve – adfoControl(grad = "analytical") measured ~4.8x faster at sigdig = 4 with standard errors unchanged to 4 significant figures. Elsewhere, compare the objective and the standard errors against NULL before relying on it. Table formatting is unaffected either way: sigdigTable defaults to 4 regardless.

addProp — How combined additive+proportional error is parameterised in the nlmixr2 output tables: "combined2" (default) or "combined1".

returnAdmr — If TRUE, return a plain list instead of the full nlmixr2 fit object.

resid_nodes — Gauss-Hermite nodes used to integrate the RESIDUAL for a transform-both-sides endpoint (boxCox, yeoJohnson, logitNorm, probitNorm), where y = g(h(f) + sigma*eps) has no closed-form mean and variance. Ignored by every other error model, which has closed forms. Default 81. Measured worst-case relative error against an independent quadrature, over all four transforms and residual SD of 0.5, 1, 2 and 3: n = 15 gives 5.7e-2, 31 gives 4.5e-3, 81 gives 5.0e-5. The error is dominated by large residual SD; at SD <= 1, n = 31 already gives 1e-7 or better.

This is an ACCURACY dial, not a speed one. The quadrature is linear in resid_nodes in isolation (~50 us at 15, 300 us at 81 for an 8-row study) but negligible beside the ODE solve: a full NLL evaluation measured 0.750 s per 60 evaluations at BOTH 31 and 81 nodes. Raise it if you have a saturating endpoint with a large residual SD; there is little to gain by lowering it.

... — Unused arguments (trigger an error).

Value

An adghControl object (a named list).

See Also

admControl(), adfoControl(), adirmcControl()

Examples

```
ctl <- adghControl()
ctl$n_nodes
ctl$grad

# More nodes for large IIV, analytical gradient
ctl2 <- adghControl(n_nodes = 7L, grad = "analytical", maxeval = 300L)


library(rxode2)
library(nlmixr2)

data("examplomycin")
obs    <- examplomycin[examplomycin$EVID == 0, ]
obs    <- obs[order(obs$ID, obs$TIME), ]
times  <- sort(unique(obs$TIME))
```

```
ids    <- unique(obs$ID)
dv_mat <- do.call(rbind, lapply(ids, function(i) {
  sub <- obs[obs$ID == i, ]; sub$DV[order(sub$TIME)]
}))
E <- colMeans(dv_mat)
V <- cov.wt(dv_mat, method = "ML")$cov

pk_model <- function() {
  ini({
    tcl <- log(5); tv <- log(30)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v ~ 0.04
  })
  model({
    cl <- exp(tcl + eta.cl)
    v  <- exp(tv  + eta.v)
    d/dt(central) <- -(cl/v) * central
    cp <- central / v
    cp ~ prop(prop.sd)
  })
}

fit <- nlmixr2(
  pk_model, admData(), est = "adgh",
  control = adghControl(
    studies = list(study1 = list(E = E, V = V, n = length(ids),
                                 times = times, ev = et(amt = 100)))
  )
)
```

`adirmcControl` *Control settings for the IRMC estimator*

Description

Constructs a control object for `est = "adirmc"`, the Iterative Reweighting Monte Carlo estimator.

Usage

```
adirmcControl(
  studies = list(),
  n_sim = 2500L,
  outer_iter = 50L,
  sampling = c("sobol", "halton", "torus", "lhs", "rnorm"),
  algorithm = NULL,
  maxeval = 5000L,
  ftol_rel = .Machine$double.eps,
```

```
  print = 1L,
  omega_expansion = 1,
  seed = 12345L,
  cores = rxode2::rxCores(),
  nDisplayProgress = .Machine$integer.max,
  grad = c("analytical", "none", "fd"),
  kappa_method = c("exact", "linearized", "linearized_gh"),
  kappa_n_nodes = 5L,
  grad_h = 1e-06,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/5),
  phases = c(2, 1, 0.5, 0.01),
  convcrit = 1e-05,
  max_worse = 5L,
  covMethod = c("r,s", "r", "none"),
  cov_n_sim = 10000L,
  n_restarts = 1L,
  restart_sd = 0.2,
  workers = 1L,
  rxControl = NULL,
  calcTables = FALSE,
  compress = TRUE,
  ci = 0.95,
  sigdig = NULL,
  sigdigTable = NULL,
  addProp = c("combined2", "combined1"),
  optExpression = TRUE,
  sumProd = FALSE,
  literalFix = TRUE,
  returnAdmr = FALSE,
  resid_nodes = 81L,
  ...
)
```

Arguments

`studies` Named list of study specifications. Each element is a list with:

- `E` – observed mean vector
- `V` – observed covariance matrix or variance vector (auto-detected)
- `n` – sample size
- `times` – numeric vector of observation times
- `ev` – `rxode2::et()` dosing event table
- `method` – `"cov"` or `"var"` (optional; auto-detected from `V`)
- `v_denom` – `"ml"` (default) or `"unbiased"`, declaring which denominator the supplied `V` uses. The likelihood is the exact one for `n` iid draws only under the ML (`n`) covariance, which is what `cov.wt(method = "ML")` and `datagen()` produce. A **published** SD is the unbiased (`n - 1`) SD, so a digitised figure gives `V = SD^2` on the `n - 1` scale: declare `v_denom = "unbiased"`

and admixr2 converts it. Declared per study, since a meta-analysis routinely mixes a digitised source with a model-derived one and the two need not share a denominator. At `n = 60` the factor is 1.7%; it matters more the smaller `n` is, and more again for any method that scores the reported covariance against its own sampling law.

Multi-compartment (multiple observed outputs). To fit several observed compartments simultaneously (e.g. plasma and brain/CSF), give the study an `observations` list instead of top-level `E`/`V`/`times`. Each entry is one observed output with its own `output` (the model prediction variable, e.g. `"cp"` or `"cCSF"`), `times`, `E`, `V` and – for independent fits – `ev` and `n`. Pass the endpoint names to `admData()`, e.g. `admData(c("cp", "cCSF"))`, so nlmixr2 recognises every endpoint. There are two modes:

- *Independent* – each observed output has its own `n`/`ev` (separate experiments / subjects, e.g. a plasma study and a brain study combined for meta-analysis). The outputs are independent likelihood blocks and the aggregate `-2LL` is their sum.
- *Joint (same subjects)* – the outputs are measured on the SAME subjects. Give the study a shared `n` and `ev`, and a joint covariance either as a study-level full matrix `V` (blocks in `observations` order) or as per-output marginal `V` plus a `cross` list of cross-covariance blocks keyed `"outA:outB"` (each `length(times_A)` x `length(times_B)`; omitted pairs are zero). The compartments are then scored by a single MVN over the stacked vector with shared random effects. `est = "adirmc"` does not support multiple observed outputs; use `"admc"`, `"adfo"` or `"adgh"`.

Long format (one row per endpoint/time). As an alternative to the `observations` list, a study may carry a `data` frame that keys each observed summary by endpoint, the way nlmixr2 keys observations by `DVID`/`CMT`. The frame needs an endpoint column (`DVID`, `CMT` or `output`), a time column (`TIME`), a mean column (`E`) and – unless a joint `V` is given – a variance column (`V`) or an SD column (`SD`). It is normalised into exactly the same units as the `observations` form, so the two are interchangeable:

```
# independent blocks: per-row variances; optional per-endpoint `n` column
# and per-endpoint `ev` (a list of event tables keyed by endpoint)
list(n = 60L, ev = ev,
   data = data.frame(DVID = c("cp", "cp", "cCSF"), TIME = c(1, 2, 2),
                     E = c(9.1, 7.4, 2.2), V = c(1.2, 0.9, 0.1)))

# joint (same subjects): ONE stacked covariance whose rows/cols align with
# the rows of `data` -- no `cross` blocks to assemble by hand
list(n = 60L, ev = ev, data = data.frame(DVID = ..., TIME = ..., E = ...),
     V = V_joint)
```

A study-level `V` (or an explicit `joint = TRUE`) marks the endpoints as same-subject; without one, each endpoint is an independent likelihood block. Endpoints are stacked in the order they first appear in `data`.

`n_sim` Number of Monte Carlo samples per NLL evaluation.

`outer_iter` Maximum inner optimiser iterations per phase.

`sampling` Sampling method for eta draws: `"sobol"` (Sobol, default), `"halton"` (Halton), `"torus"` (Kronecker/torus), `"lhs"` (Latin hypercube), or `"rnorm"` (iid normal).

`algorithm` nloptr algorithm string, or `NULL` (default) to pick the default that matches `grad`: `"NLOPT_LD_LBFGS"` with a gradient, `"NLOPT_LN_BOBYQA"` when `grad = "none"`. Any algorithm reported by `nloptr::nloptr.print.options()` is accepted (e.g. `"NLOPT_LD_MMA"`, `"NLOPT_LN_NELDERMEAD"`). An explicit algorithm is reconciled with `grad`: when `grad = "none"` a gradient-based algorithm (`NLOPT_LD_*` / `NLOPT_GD_*`) falls back to `"NLOPT_LN_BOBYQA"`; when a gradient is requested a derivative-free algorithm (`NLOPT_LN_*` / `NLOPT_GN_*`) turns the gradient off. Both emit a message.

`maxeval` Maximum number of optimizer function evaluations.

`ftol_rel` Relative function-value tolerance for convergence.

`print` Print progress every this many evaluations (0 = silent).

`omega_expansion`
Inflate proposal Omega by this factor (>= 1).

`seed` Random seed for reproducibility.

`cores` Number of OpenMP threads for `rxSolve()`. Defaults to `rxode2::rxCores()`. `rxSolve()` parallelises over subjects, so this is the main speed lever for the MC estimators; when `workers > 1` it is a *total* budget, split across the workers.

`nDisplayProgress`
Passed to `rxSolve()`: the solver shows its text progress bar only once a single solve exceeds this many subjects. The default (`.Machine$integer.max`) keeps the bar off, which is what you want for scripts, vignettes and logs; lower it (e.g. `1000L`) to see solver progress during long interactive fits.

`grad` Gradient mode for the inner optimiser: `"analytical"` (default, closed-form weight-path gradient), `"none"` (derivative-free BOBYQA), or `"fd"` (central finite differences). Note: `"sens"` is not available for the IRMC estimator.

`kappa_method` Kappa correction method for models with non-mu-referenced struct thetas: `"exact"` (default, re-evaluates population prediction `f(theta, 0)` via rxSolve at each inner step), `"linearized"` (precomputes `J = df/d(theta)` once per outer iteration using `f(theta, 0)` as baseline — zero rxSolve per inner step), or `"linearized_gh"` (same linear approximation but baseline and Jacobian use Gauss-Hermite quadrature `E_GH[f(theta, eta)]` instead of `f(theta, 0)` — more accurate baseline at any IIV magnitude, still zero rxSolve per inner step).

`kappa_n_nodes` Number of GH nodes per eta dimension for `kappa_method = "linearized_gh"` (default 5). Total quadrature points = `kappa_n_nodes^n_eta`. Ignored for other kappa methods.

`grad_h` Step size for the inner optimiser's finite-difference gradient (`grad = "fd"`). Defaults to `1e-6`, not the `1e-4` the other three controls use: the IRMC inner NLL is deterministic given fixed proposals, so there is no Monte Carlo noise to step over and the truncation-versus-noise balance that sets `1e-4` elsewhere does not apply. The inner step was a hard-coded `1e-6` until it was made to honour `grad_h`; inheriting the coarser default would have changed the gradient the loop was tuned for.

`cov_h` Inner FD step for the gradient-based Hessian (only used when `covMethod = "r"` and `grad != "none"`). Each gradient evaluation has MC noise of order `sigma / cov_h`; the Hessian divides that noise by the outer step, giving total noise `sigma / (cov_h * cov_h_outer * |p|)`. `cov_h = 1e-3` balances truncation error and noise amplification. Increase to `1e-2` if the Hessian is non-positive definite.

`cov_h_outer` Outer step scale for the numerical Hessian. The actual step for parameter `p` is `max(|p|, 0.1) * cov_h_outer`. Applied to both the gradient-FD Hessian (`grad != "none"`) and the NLL-FD Hessian (`grad = "none"`). Default `eps^(1/5)` (~2.5e-3) is larger than the textbook `eps^(1/4)` to account for MC noise in NLL and gradient evaluations; empirically it matches the analytical (sensitivity-equation) Hessian ground truth. Increase (e.g. to `5e-3` or `1e-2`) if the Hessian is non-positive definite.

`phases` Numeric vector of box-constraint half-widths, one per phase. Phases progressively tighten the search region.

`convcrit` Convergence criterion: phase ends when `|approx - exact| < convcrit`.

`max_worse` Stop a phase after this many consecutive worsening iterations.

`covMethod` `"r,s"` (the DEFAULT) computes the sandwich `H^-1 J H^-1`; `"r"` the numerical Hessian alone, `2H^-1`; `"none"` skips the covariance. All three span the structural, residual-error and omega parameters, and are reported on the scale the estimates are printed on. `admControl()` documents what the sandwich is, why it is the conservative default, and why it is more sensitive than `"r"` to an ill-conditioned Hessian; the same implementation runs here. What does NOT carry over is the family coverage, because `adirmc` itself is narrower: the estimator accepts only `add`, `prop`, `pow`, `combined1`, `combined2` and `lnorm` residuals, and `"r,s"` applies to all six. The count, `beta`, transform-both-sides, `ordinal` and `ar()` endpoints `admControl()` lists are refused by `est = "adirmc"` itself, before any covariance is reached – they are not models whose sandwich degrades to `"r"` here, they are models this estimator does not fit.

`cov_n_sim` Number of MC samples for the covariance (Hessian) step. More samples reduce MC noise in NLL evaluations. The NLL-based Hessian (`grad = "none"`) uses a central second difference of the NLL with the same Sobol sequence (CRN) at every perturbed point, so noise largely cancels and `cov_n_sim = 10000` (default) is sufficient for most models.

`n_restarts` Number of optimization restarts. Runs in parallel when `workers > 1`.

`restart_sd` Standard deviation of structural theta perturbations for restart initialisation.

`workers` Number of parallel workers for multi-restart. `1` (default) runs restarts sequentially. Values `> 1` run the restarts on a pool of background R processes (mirai daemons), which behaves the same way on every platform. Requires the `mirai` package. Workers are stopped automatically after the restart phase so all cores are available for the Hessian step; if a fit is interrupted, `admStopWorkers()` cleans up any survivors.

`rxControl` `rxode2::rxControl()` object. Created automatically when `NULL`.

`calcTables, compress, ci, sigdigTable, optExpression, sumProd, literalFix`

Passed to `nlmixr2est::foceiControl()` for the table/output machinery.

`sigdig` Significant digits asked of the ODE solver, or `NULL` (the default) to leave rxode2's own solver tolerances alone. When set, it is passed to `rxode2::rxSolve()`'s own `sigdig` argument for every solve the estimator issues – rxode2 owns the mapping to `atol`/`rtol` and has changed it between releases, which is why the digits, not the tolerances, are what travels – and to `nlmixr2est::foceiControl()` for the post-fit tables.

It is a speed lever, and an opt-in one because it is not free. The estimators finite-difference the solve with steps of the same order: `grad_h` (1e-4), `cov_h` (1e-3) and `cov_h_outer` (~2.5e-3), while `sigdig = 4` maps to a relative tolerance of ~1e-4 on current rxode2. Differencing a solution whose own noise is 1e-4 with a 1e-4 step returns noise, and it surfaces as a moved objective and an indefinite covariance Hessian (every `SE` reported `NA`) rather than as an error. Most worthwhile where the gradient is fully analytic and nothing differences the solve – `adfoControl(grad = "analytical")` measured ~4.8x faster at `sigdig = 4` with standard errors unchanged to 4 significant figures. Elsewhere, compare the objective and the standard errors against `NULL` before relying on it. Table formatting is unaffected either way: `sigdigTable` defaults to 4 regardless.

`addProp` How combined additive+proportional error is parameterised in the nlmixr2 output tables: `"combined2"` (default, variance form) or `"combined1"` (SD form). Has no effect on admixr2's own estimation; passed to `nlmixr2est::foceiControl()` for the table/output machinery only.

`returnAdmr` If `TRUE`, return a plain list instead of a full nlmixr2 fit object (useful for debugging).

`resid_nodes` Gauss-Hermite nodes used to integrate the RESIDUAL for a transform-both-sides endpoint (`boxCox`, `yeoJohnson`, `logitNorm`, `probitNorm`), where `y = g(h(f) + sigma*eps)` has no closed-form mean and variance. Ignored by every other error model, which has closed forms. Default 81. Measured worst-case relative error against an independent quadrature, over all four transforms and residual SD of 0.5, 1, 2 and 3: n = 15 gives 5.7e-2, 31 gives 4.5e-3, 81 gives 5.0e-5. The error is dominated by large residual SD; at SD <= 1, n = 31 already gives 1e-7 or better.

This is an ACCURACY dial, not a speed one. The quadrature is linear in `resid_nodes` in isolation (~50 us at 15, 300 us at 81 for an 8-row study) but negligible beside the ODE solve: a full NLL evaluation measured 0.750 s per 60 evaluations at BOTH 31 and 81 nodes. Raise it if you have a saturating endpoint with a large residual SD; there is little to gain by lowering it.

`...` Additional arguments (none allowed; triggers an error).

Details

Multi-compartment fits (a study `observations` list with several observed outputs) are **not** supported by `adirmc`; use `est = "admc"`, `"adfo"`, or `"adgh"` for those. Single-output studies are fit as usual.

Value

An object of class `adirmcControl`.

Examples

```
# Inspect defaults
ctl <- adirmcControl()
ctl$phases
ctl$omega_expansion

# Tighter phases, more restarts
ctl2 <- adirmcControl(
  n_sim           = 1000L,
  omega_expansion = 1.5,
  phases          = c(2, 1, 0.5, 0.01),
  n_restarts      = 3L
)


library(rxode2)
library(nlmixr2)

data("examplomycin")
obs   <- examplomycin[examplomycin$EVID == 0, ]
obs   <- obs[order(obs$ID, obs$TIME), ]
times <- sort(unique(obs$TIME))
ids   <- unique(obs$ID)
dv_mat <- do.call(rbind, lapply(ids, function(i) {
  sub <- obs[obs$ID == i, ]; sub$DV[order(sub$TIME)]
}))
E <- colMeans(dv_mat)
V <- diag(diag(cov.wt(dv_mat, method = "ML")$cov))

pk_model <- function() {
  ini({
    tcl <- log(5);  tv1 <- log(12); tv2 <- log(25)
    tq  <- log(12); tka <- log(1.2)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v1 ~ 0.09; eta.v2 ~ 0.09
    eta.q  ~ 0.09; eta.ka ~ 0.09
  })
  model({
    cl <- exp(tcl + eta.cl); v1 <- exp(tv1 + eta.v1)
    v2 <- exp(tv2 + eta.v2); q  <- exp(tq  + eta.q)
    ka <- exp(tka + eta.ka)
    d/dt(depot)      <- -ka * depot
    d/dt(central)    <- ka * depot - (cl/v1 + q/v1) * central + (q/v2) * peripheral
    d/dt(peripheral) <- (q/v1) * central - (q/v2) * peripheral
    cp <- central / v1
    cp ~ prop(prop.sd)
  })
}

fit <- nlmixr2(
  pk_model, admData(), est = "adirmc",
  control = adirmcControl(
```

```
    studies = list(study1 = list(E = E, V = V, n = length(ids),
                                 times = times, ev = et(amt = 100))),
    n_sim   = 500L
  )
)
print(fit)
```

---

admControl *Control settings for the ADM estimator*

Description

Constructs a control object for `est = "admc"`, the Monte Carlo aggregate data modelling estimator.

Usage

```
admControl(
  studies = list(),
  n_sim = 5000L,
  sampling = c("sobol", "halton", "torus", "lhs", "rnorm"),
  algorithm = NULL,
  maxeval = 500L,
  ftol_rel = .Machine$double.eps^2,
  print = 10L,
  seed = 12345L,
  cores = rxode2::rxCores(),
  nDisplayProgress = .Machine$integer.max,
  grad = c("sens", "fd", "none"),
  grad_h = 1e-04,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/5),
  grad_bounds = 5,
  covMethod = c("r,s", "r", "none"),
  cov_n_sim = 10000L,
  n_restarts = 1L,
  restart_sd = 0.5,
  workers = 1L,
  rxControl = NULL,
  calcTables = FALSE,
  compress = TRUE,
  ci = 0.95,
  sigdig = NULL,
  sigdigTable = NULL,
  addProp = c("combined2", "combined1"),
  optExpression = TRUE,
  sumProd = FALSE,
```

```
  literalFix = TRUE,
  returnAdmr = FALSE,
  resid_nodes = 81L,
  ...
)
```

Arguments

`studies` Named list of study specifications. Each element is a list with:

- `E` – observed mean vector
- `V` – observed covariance matrix or variance vector (auto-detected)
- `n` – sample size
- `times` – numeric vector of observation times
- `ev` – `rxode2::et()` dosing event table
- `method` – `"cov"` or `"var"` (optional; auto-detected from `V`)
- `v_denom` – `"ml"` (default) or `"unbiased"`, declaring which denominator the supplied `V` uses. The likelihood is the exact one for `n` iid draws only under the ML (`n`) covariance, which is what `cov.wt(method = "ML")` and `datagen()` produce. A **published** SD is the unbiased (`n - 1`) SD, so a digitised figure gives `V = SD^2` on the `n - 1` scale: declare `v_denom = "unbiased"` and admixr2 converts it. Declared per study, since a meta-analysis routinely mixes a digitised source with a model-derived one and the two need not share a denominator. At `n = 60` the factor is 1.7%; it matters more the smaller `n` is, and more again for any method that scores the reported covariance against its own sampling law.

Multi-compartment (multiple observed outputs). To fit several observed compartments simultaneously (e.g. plasma and brain/CSF), give the study an `observations` list instead of top-level `E`/`V`/`times`. Each entry is one observed output with its own `output` (the model prediction variable, e.g. `"cp"` or `"cCSF"`), `times`, `E`, `V` and – for independent fits – `ev` and `n`. Pass the endpoint names to `admData()`, e.g. `admData(c("cp", "cCSF"))`, so nlmixr2 recognises every endpoint. There are two modes:

- *Independent* – each observed output has its own `n`/`ev` (separate experiments / subjects, e.g. a plasma study and a brain study combined for meta-analysis). The outputs are independent likelihood blocks and the aggregate `-2LL` is their sum.
- *Joint (same subjects)* – the outputs are measured on the SAME subjects. Give the study a shared `n` and `ev`, and a joint covariance either as a study-level full matrix `V` (blocks in `observations` order) or as per-output marginal `V` plus a `cross` list of cross-covariance blocks keyed `"outA:outB"` (each `length(times_A)` x `length(times_B)`; omitted pairs are zero). The compartments are then scored by a single MVN over the stacked vector with shared random effects. `est = "adirmc"` does not support multiple observed outputs; use `"admc"`, `"adfo"` or `"adgh"`.

Long format (one row per endpoint/time). As an alternative to the `observations` list, a study may carry a `data` frame that keys each observed summary by endpoint, the way nlmixr2 keys observations by `DVID`/`CMT`. The frame needs an end-

point column (DVID, CMT or output), a time column (TIME), a mean column (E) and – unless a joint V is given – a variance column (V) or an SD column (SD). It is normalised into exactly the same units as the observations form, so the two are interchangeable:

```
# independent blocks: per-row variances; optional per-endpoint `n` column
# and per-endpoint `ev` (a list of event tables keyed by endpoint)
list(n = 60L, ev = ev,
   data = data.frame(DVID = c("cp", "cp", "cCSF"), TIME = c(1, 2, 2),
                     E = c(9.1, 7.4, 2.2), V = c(1.2, 0.9, 0.1)))

# joint (same subjects): ONE stacked covariance whose rows/cols align with
# the rows of `data` -- no `cross` blocks to assemble by hand
list(n = 60L, ev = ev, data = data.frame(DVID = ..., TIME = ..., E = ...),
     V = V_joint)
```

A study-level V (or an explicit joint = TRUE) marks the endpoints as same-subject; without one, each endpoint is an independent likelihood block. Endpoints are stacked in the order they first appear in data.

n_sim — Number of Monte Carlo samples per NLL evaluation.

sampling — Sampling method for eta draws: "sobol" (Sobol, default), "halton" (Halton), "torus" (Kronecker/torus), "lhs" (Latin hypercube), or "rnorm" (iid normal).

algorithm — nloptr algorithm string, or NULL (default) to pick the default that matches grad: "NLOPT_LD_LBFGS" with a gradient, "NLOPT_LN_BOBYQA" when grad = "none". Any algorithm reported by nloptr::nloptr.print.options() is accepted (e.g. "NLOPT_LD_MMA", "NLOPT_LN_NELDERMEAD"). An explicit algorithm is reconciled with grad: when grad = "none" a gradient-based algorithm (NLOPT_LD_* / NLOPT_GD_*) falls back to "NLOPT_LN_BOBYQA"; when a gradient is requested a derivative-free algorithm (NLOPT_LN_* / NLOPT_GN_*) turns the gradient off. Both emit a message.

maxeval — Maximum number of optimizer function evaluations.

ftol_rel — Relative function-value tolerance for convergence.

print — Print progress every this many evaluations (0 = silent).

seed — Random seed for reproducibility.

cores — Number of OpenMP threads for rxSolve(). Defaults to rxode2::rxCores(). rxSolve() parallelises over subjects, so this is the main speed lever for the MC estimators; when workers > 1 it is a *total* budget, split across the workers.

nDisplayProgress — Passed to rxSolve(): the solver shows its text progress bar only once a single solve exceeds this many subjects. The default (.Machine$integer.max) keeps the bar off, which is what you want for scripts, vignettes and logs; lower it (e.g. 1000L) to see solver progress during long interactive fits.

grad — Gradient mode: "sens" (sensitivity equations, default), "fd" (central finite differences; forward was removed in 0.4.1), or "none" (derivative-free). A warning is issued when "sens" is requested but the sensitivity model is unavailable; the estimator then falls back to central finite differences.

`grad_h` Step size for finite-difference gradient evaluation during optimization (used by `grad = "fd"`). This is the FALLBACK step: the step is normally measured per parameter by the Shi (2021) procedure, and `grad_h` is what a parameter falls back to when that measurement cannot be made (a direction the objective is flat in, or a failed noise estimate).

`cov_h` Inner FD step for the gradient-based Hessian (only used when `covMethod = "r"` and `grad != "none"`). Each gradient evaluation has MC noise of order `sigma / cov_h`; the Hessian divides that noise by the outer step, giving total noise `sigma / (cov_h * cov_h_outer * |p|)`. `cov_h = 1e-3` balances truncation error and noise amplification. Increase to `1e-2` if the Hessian is non-positive definite.

`cov_h_outer` Outer step scale for the numerical Hessian. The actual step for parameter `p` is `max(|p|, 0.1) * cov_h_outer`. Applied to both the gradient-FD Hessian (`grad != "none"`) and the NLL-FD Hessian (`grad = "none"`). Default `eps^(1/5)` (~2.5e-3) is larger than the textbook `eps^(1/4)` to account for MC noise in NLL and gradient evaluations; empirically it matches the analytical (sensitivity-equation) Hessian ground truth. Increase (e.g. to `5e-3` or `1e-2`) if the Hessian is non-positive definite.

`grad_bounds` Box-constraint half-width when using gradients: the fit is confined to `p0 +/- grad_bounds` on the optimizer scale, which for a log-scale parameter is a factor of `exp(grad_bounds)` (~148 at the default 5). This bound is admixr2's, not the model's – an unbounded parameter has no other – and nloptr reports normal convergence at a box corner, so a warning is emitted if an estimate finishes on it.

`covMethod` `"r,s"` (the DEFAULT) computes the sandwich `H^-1 J H^-1`; `"r"` the numerical Hessian alone, `2H^-1`; `"none"` skips the covariance. All three span the structural, residual-error and omega parameters. Omega is included because excluding it also biases the STRUCTURAL standard errors downward – a theta carrying an eta is correlated with that eta's variance. If the weakly-identified omega Cholesky makes the Hessian non-positive definite, the structural + residual sub-block is reported with a warning.

`"r,s"` adds a sandwich correction, `H^-1 J H^-1`, on the same Hessian. The aggregate objective scores the reported mean and covariance as though the subjects behind them were multivariate normal; they are not, because the model is nonlinear in the random effects. `"r,s"` scores that law from the model instead, on a quadrature ensemble rather than on this fit's own MC draws, so the reported uncertainty carries no sampling noise of its own. Point estimates are untouched, and under correct specification it reduces to `"r"` exactly. **It is the default because it is the conservative choice, not the aggressive one.** Under correct specification `J = 2H` and the sandwich returns what `"r"` returns, so defaulting to it costs nothing when the normal-theory assumption holds and corrects the standard errors when it does not. Anything it cannot build degrades to `"r"` and reports `"r"`, so no fit loses its covariance by asking. Pass `covMethod = "r"` for the pre-0.4.1 behaviour.

Applies to every residual family whose conditional law is independent across timepoints, which is all of them except `ar()`: the conditionally-normal set (`add`, `prop`, `pow`, `combined1`, `combined2`), the closed-form distributional ones (`lnorm`, `pois`, `binom`, `nbinomMu`, `beta`, and `t()` with `nu > 4`), and the transform-both-sides ones (`boxCox`, `yeoJohnson`, `logitNorm`, `probitNorm`), whose third and

fourth conditional moments come off the same quadrature that already gives their mean and variance. Refused, and degraded to `"r"`: `ar()`, because it correlates the residual ACROSS timepoints and the cross terms the expansion drops are then real; `t()` with `nu <= 4`, whose kurtosis does not exist; and `ordinal()` and same-subject `joint` studies, which stack several outputs into one covariance the per-output node ensemble does not describe. These four are refusals by construction rather than failures, so the fit reports the reason as a message and falls back to `"r"`; a sandwich that was attempted and could not be built still warns.

`"r,s"` is more sensitive to an ill-conditioned Hessian than `"r"` is. `"r"` reports `2H^-1` and inverts `H` once; the sandwich reports `H^-1 J H^-1` and inverts it twice, so in a direction the data barely identifies any gap between `J` and `2H` is amplified quadratically. A residual SD contributing 0.01 variance against 1.7 from between-subject variability is such a direction: measured on one 1-cmt fixture at `cond(H) = 3.5e5`, the reported residual SE moved by a factor of 0.11 and two omega entries by 0.59 and 1.55, while the same model and design on a study the residual IS identified in (`cond(H) = 247`) reproduced `"r"` to four decimals on every parameter. Neither number is a correction there – both methods are reporting an unidentified direction, and `"r,s"` is louder about it. admixr2 says so: when the Hessian's reciprocal condition number falls below `eps^(1/4)` – the point at which squaring the conditioning reaches the bound a single inversion is already called singular at – the fit records a note naming the parameter that loads most heavily on the offending direction. It arrives on `fit$runInfo` and is listed by `print(fit)`, which is where `nlmixr2est` routes an estimator's warnings. The sandwich is still reported, because the well-determined parameters of the same fit are unaffected; check the named parameter's relative standard error before reading its `"r,s"` value as a finding.

All three blocks are reported on the scale the ESTIMATES are printed on, as `nlmixr2est` does: structural thetas on the log/optimizer scale, residual error as an SD, and omega as the variance/covariance entries (named `om.<eta>` and `cov.<eta_i>.<eta_j>`). The omega block is rotated by the full Jacobian of Omega with respect to the log-Cholesky, which is not diagonal once omega is correlated.

`cov_n_sim` Number of MC samples for the covariance (Hessian) step. More samples reduce MC noise in NLL evaluations. The NLL-based Hessian (`grad = "none"`) uses a central second difference of the NLL with the same Sobol sequence (CRN) at every perturbed point, so noise largely cancels and `cov_n_sim = 10000` (default) is sufficient for most models.

`n_restarts` Number of optimization restarts. Runs in parallel when `workers > 1`.

`restart_sd` Standard deviation of structural theta perturbations for restart initialisation.

`workers` Number of parallel workers for multi-restart. `1` (default) runs restarts sequentially. Values `> 1` run the restarts on a pool of background R processes (mirai daemons), which behaves the same way on every platform. Requires the `mirai` package. Workers are stopped automatically after the restart phase so all cores are available for the Hessian step; if a fit is interrupted, `admStopWorkers()` cleans up any survivors.

`rxControl` `rxode2::rxControl()` object. Created automatically when `NULL`.

`calcTables`, `compress`, `ci`, `sigdigTable`, `optExpression`, `sumProd`, `literalFix`
: Passed to `nlmixr2est::foceiControl()` for the table/output machinery.

`sigdig`
: Significant digits asked of the ODE solver, or `NULL` (the default) to leave rxode2's own solver tolerances alone. When set, it is passed to `rxode2::rxSolve()`'s own `sigdig` argument for every solve the estimator issues – rxode2 owns the mapping to `atol`/`rtol` and has changed it between releases, which is why the digits, not the tolerances, are what travels – and to `nlmixr2est::foceiControl()` for the post-fit tables.

  It is a speed lever, and an opt-in one because it is not free. The estimators finite-difference the solve with steps of the same order: `grad_h` (1e-4), `cov_h` (1e-3) and `cov_h_outer` (~2.5e-3), while `sigdig = 4` maps to a relative tolerance of ~1e-4 on current rxode2. Differencing a solution whose own noise is 1e-4 with a 1e-4 step returns noise, and it surfaces as a moved objective and an indefinite covariance Hessian (every `SE` reported `NA`) rather than as an error. Most worthwhile where the gradient is fully analytic and nothing differences the solve – `adfoControl(grad = "analytical")` measured ~4.8x faster at `sigdig = 4` with standard errors unchanged to 4 significant figures. Elsewhere, compare the objective and the standard errors against `NULL` before relying on it. Table formatting is unaffected either way: `sigdigTable` defaults to 4 regardless.

`addProp`
: How combined additive+proportional error is parameterised in the nlmixr2 output tables: `"combined2"` (default, variance form) or `"combined1"` (SD form). Has no effect on admixr2's own estimation; passed to `nlmixr2est::foceiControl()` for the table/output machinery only.

`returnAdmr`
: If `TRUE`, return a plain list instead of a full nlmixr2 fit object (useful for debugging).

`resid_nodes`
: Gauss-Hermite nodes used to integrate the RESIDUAL for a transform-both-sides endpoint (`boxCox`, `yeoJohnson`, `logitNorm`, `probitNorm`), where `y = g(h(f) + sigma*eps)` has no closed-form mean and variance. Ignored by every other error model, which has closed forms. Default 81. Measured worst-case relative error against an independent quadrature, over all four transforms and residual SD of 0.5, 1, 2 and 3: n = 15 gives 5.7e-2, 31 gives 4.5e-3, 81 gives 5.0e-5. The error is dominated by large residual SD; at SD <= 1, n = 31 already gives 1e-7 or better.

  This is an ACCURACY dial, not a speed one. The quadrature is linear in `resid_nodes` in isolation (~50 us at 15, 300 us at 81 for an 8-row study) but negligible beside the ODE solve: a full NLL evaluation measured 0.750 s per 60 evaluations at BOTH 31 and 81 nodes. Raise it if you have a saturating endpoint with a large residual SD; there is little to gain by lowering it.

`...`
: Additional arguments (none allowed; triggers an error).

Value

An object of class `admControl`.

Examples

```
# Minimal control object -- inspect defaults
```

```
ctl <- admControl()
ctl$n_sim
ctl$algorithm

# Override key settings without fitting
ctl2 <- admControl(
  n_sim    = 2000L,
  maxeval  = 300L,
  grad     = "fd",
  seed     = 42L
)


library(rxode2)
library(nlmixr2)

data("examplomycin")
obs   <- examplomycin[examplomycin$EVID == 0, ]
obs   <- obs[order(obs$ID, obs$TIME), ]
times <- sort(unique(obs$TIME))
ids   <- unique(obs$ID)
dv_mat <- do.call(rbind, lapply(ids, function(i) {
  sub <- obs[obs$ID == i, ]; sub$DV[order(sub$TIME)]
}))
E <- colMeans(dv_mat)
V <- cov.wt(dv_mat, method = "ML")$cov

pk_model <- function() {
  ini({
    tcl <- log(5);  tv1 <- log(12); tv2 <- log(25)
    tq  <- log(12); tka <- log(1.2)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v1 ~ 0.09; eta.v2 ~ 0.09
    eta.q  ~ 0.09; eta.ka ~ 0.09
  })
  model({
    cl <- exp(tcl + eta.cl); v1 <- exp(tv1 + eta.v1)
    v2 <- exp(tv2 + eta.v2); q  <- exp(tq  + eta.q)
    ka <- exp(tka + eta.ka)
    d/dt(depot)      <- -ka * depot
    d/dt(central)    <- ka * depot - (cl/v1 + q/v1) * central + (q/v2) * peripheral
    d/dt(peripheral) <- (q/v1) * central - (q/v2) * peripheral
    cp <- central / v1
    cp ~ prop(prop.sd)
  })
}

fit <- nlmixr2(
  pk_model, admData(), est = "admc",
  control = admControl(
    studies  = list(study1 = list(E = E, V = V, n = length(ids),
                                  times = times, ev = et(amt = 100))),
    n_sim    = 1000L,
```

```
    maxeval  = 200L
  )
)
print(fit)
```

---

admData *Dummy data frame for nlmixr2 dispatch*

Description

Returns a minimal NONMEM-style data frame that satisfies nlmixr2's data argument requirement. The single observation row carries a non-NA placeholder DV (1); the dose row keeps DV = NA. The placeholder is purely for dispatch and output construction and never enters the reported objective – each estimator overwrites fitenvobjective (and OBJF/logLik/AIC/BIC) with its own aggregate -2LL. A non-NA observation is required because nlmixr2's post-fit output construction (nlmixr2CreateOutputFromUi) solves the model over this frame, and rxode2's event-table translation rejects a dataset with no non-NA observation rows ("no rows in event table or input data"), the same reason the multi-endpoint frame below uses a placeholder DV.

Usage

```
admData(outputs = NULL)
```

Arguments

outputs	Optional character vector of observed output (endpoint) names for a multi-compartment model with several prediction lines (e.g. c("cp", "cCSF")). One observation row is emitted per endpoint, keyed by name in the DVID column (nlmixr2's endpoint identifier; CMT is NA on those rows), so nlmixr2's data translation recognises every endpoint. These rows carry a non-NA placeholder DV (1) because nlmixr2's multi-endpoint translator rejects an all-NA-DV dataset; the placeholder is purely for dispatch and never enters the reported objective (each estimator overwrites it with its own aggregate -2LL). When NULL (default) the single-endpoint dummy frame is returned unchanged.

Value

A data frame with columns ID, TIME, DV, AMT, EVID, CMT (single-endpoint), plus a DVID endpoint column when outputs is given.

Examples

```
admData()
admData(c("cp", "cCSF"))
```

admStopWorkers *Stop parallel workers*

Description

Stops any worker processes (mirai daemons) started by a parallel-restart fit (`admControl(workers = N)`). Workers are stopped automatically after the restart phase completes, so this function is only needed if a fit was interrupted before cleanup could run.

Usage

```
admStopWorkers()
```

Value

`NULL`, invisibly.

Examples

```
# Safe to call at any time; no-op if no workers are running
admStopWorkers()
```

anova.admFit *Compare nested admixr2 fits by a likelihood-ratio test*

Description

The ordinary LRT: the objective difference against a chi-squared reference with `Df` equal to the number of parameters the larger model adds.

Usage

```
## S3 method for class 'admFit'
anova(object, ...)
```

Arguments

`object`	An `admFit`.
`...`	Further `admFit`s to compare it with.

Details

Both fits must come from the same estimator and, for the quadrature estimators, the same node count. Each scores its own approximation to the likelihood, so objectives from different ones are not comparable and the comparison is refused rather than reported.

Testing a variance AT ZERO puts the null on the boundary of the parameter space, where the exact reference is a chi-bar-squared mixture rather than a chi-squared. The p-value reported there is CONSERVATIVE – too large – so a significant result stays significant, but treat a borderline one with care.

Value

A data frame of class `anova.admFit`, smallest model first.

`datagen` *Generate aggregate study data from (possibly different) pharmacometric models*

Description

Generates population mean vectors (`E`) and covariance matrices (`V`) for each study by integrating over the IIV distribution – either by Monte Carlo (the default) or by a deterministic First-Order expansion (`method = "fo"`, see `datagenControl()`). Each study may specify its own PK/PD model (as would be the case when digitising data from several published studies, each fit with a different structural model). True parameter values are taken from the `ini()` block of each study's model. Each element of the returned list is ready to supply directly to `admControl(studies = ...)`.

Usage

```
datagen(studies, model = NULL, control = datagenControl())
```

Arguments

`studies` A named list of study specifications. Each element is a list with:

- `model` An nlmixr2-style model function with `ini()` and `model()` blocks. Serves as the data-generating model for this study. May differ between studies. Can be omitted if a top-level default is supplied via the `model` argument.
- `times` Numeric vector of observation times.
- `ev` A dosing event table created with `rxode2::et()`.
- `n` (Optional) integer sample size; stored as metadata and used when supplying the result to `admControl()`.
- `observations` (Optional) a named list to generate data for several observed outputs (multi-compartment). Each entry gives one output's `output` (model prediction variable, e.g. `"cp"`), `times`, and optionally `ev`/`n` (inherited from the study otherwise). When present, the study result carries a matching `observations` list of per-output `E`/`V`, ready to pass straight to `admControl(studies = ...)`.

`model`	Optional default model function used for any study that does not supply its own `model` element. At least one of `model` or each study's `model` must be non-`NULL`.
`control`	A `datagenControl()` object.

Details

With `control = datagenControl(method = "mc")` (the default) population moments are computed via the same Monte Carlo engine as `est = "admc"`:

$$E_t = \bar{f}_s(\hat{\theta}_s, \eta_i, t)$$

$$V_{ts} = \widehat{\mathrm{Cov}}_\eta[f_{s,t}, f_{s,s'}] + \Sigma_s$$

where f_s and $\hat{\theta}_s$ are the model and initial estimates from the `ini()` block of study s, the sample covariance uses the ML denominator `n_sim`, and Σ_s is diagonal with entries determined by that study model's residual error type (additive, proportional, or log-normal).

With `method = "fo"` the moments are instead the deterministic First-Order expansion used by `est = "adfo"`:

$$E = f_s(\hat{\theta}_s, 0)$$

$$V = J\Omega_s J^\top + \Sigma_s, \quad J_{tj} = \partial f_{s,t}/\partial \eta_j|_{\eta=0}$$

with the Jacobian J obtained from the sensitivity model (or finite differences if that is unavailable). This is the natural choice for design evaluation and optimal design: the moments are fast and reproducible, and because the data-generating and data-analytic models coincide, the FO Hessian of the log-likelihood (the expected information matrix) is evaluated at the true maximum rather than at a point that is not an MLE of the generated data. Note `est = "adfo"` always adds Σ to its predicted covariance, so for a consistent FIM keep the residual error in the generating model; omit it only when residual-free (IIV-only) moments are genuinely what you want.

With `method = "gh"` the moments are computed by deterministic Gauss-Hermite quadrature over the random-effects prior $\eta \sim N(0, \Omega)$:

$$E = \sum_q w_q f(\hat{\theta}, \eta_q), \quad V = \sum_q w_q (f_q - E)(f_q - E)^\top + \Sigma$$

where (η_q, w_q) are the Cholesky-scaled tensor-product GH nodes and weights. Unlike FO this is unbiased at any IIV magnitude; unlike MC the result is noise-free and exactly reproducible. Matching the moments of `est = "adgh"` makes `method = "gh"` the natural choice for optimal design with that estimator.

Models are compiled and cached on first use (keyed by model expression digest), so repeated calls or multiple studies sharing the same model incur only a single compilation.

Value

A named list with one element per study. Each element contains:

`E` Population mean vector at `times`.

`V` Population covariance matrix (`length(times)` x `length(times)`; ML denominator `n_sim` for `method = "mc"`, the analytical FO covariance for `method = "fo"`, or the GH weighted covariance for `method = "gh"`). The diagonal carries the model's residual-error variance; to generate residual-free (IIV-only) moments, omit the error term from the model.

`n` Sample size (`NA_integer_` if not supplied).

`times` Observation times.

`ev` Dosing event table.

`samples` Raw `n_sim x length(times)` prediction matrix (only when `control$return_samples = TRUE`).

See Also

`datagenControl()`, `admControl()`

Examples

```
library(rxode2)

pk_model <- function() {
  ini({
    tcl <- log(5); tv <- log(30)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v ~ 0.04
  })
  model({
    cl <- exp(tcl + eta.cl)
    v  <- exp(tv  + eta.v)
    d/dt(central) <- -(cl/v) * central
    cp <- central / v
    cp ~ prop(prop.sd)
  })
}

study_data <- datagen(
  studies = list(
    study1 = list(times = c(1, 2, 4, 8, 12, 24),
                  ev = rxode2::et(amt = 100), n = 200L)
  ),
  model   = pk_model,
  control = datagenControl(n_sim = 2000L)
)

# E and V plug directly into admControl(studies = ...)
round(study_data$study1$E, 2)
```

`datagenControl` *Control parameters for* `datagen()`

Description

Control parameters for `datagen()`

Usage

```
datagenControl(
  method = c("mc", "fo", "gh"),
  n_sim = 5000L,
  n_nodes = 5L,
  sampling = c("sobol", "halton", "torus", "lhs", "rnorm"),
  seed = 12345L,
  cores = 1L,
  return_samples = FALSE,
  resid_nodes = 81L
)
```

Arguments

`method`	Moment approximation used to generate `E` and `V`: `"mc"` (default) draws Monte Carlo samples over the IIV distribution, as in `est = "admc"`; `"fo"` uses the deterministic First-Order expansion (`mu = f(theta, 0)`, `V = J Omega J' + Sigma`), matching `est = "adfo"`; `"gh"` uses deterministic Gauss-Hermite quadrature over the random-effects prior, matching `est = "adgh"` – unbiased at any IIV magnitude and noise-free. Use `"fo"` or `"gh"` for design evaluation where the data-generating and data-analytic models must coincide.
`n_sim`	Number of Monte Carlo samples used to approximate population moments. Ignored when `method = "fo"` or `"gh"`.
`n_nodes`	Number of Gauss-Hermite nodes per eta dimension for `method = "gh"` (default 5). Total nodes = `n_nodes^n_eta`. Ignored for `"mc"` and `"fo"`.
`sampling`	Quasi-random sampling method: `"sobol"` (default), `"halton"`, `"torus"`, `"lhs"`, or `"rnorm"`. Ignored when `method = "fo"` or `"gh"`.
`seed`	Integer seed. Applied before stochastic methods (`"rnorm"`, `"lhs"`). Ignored when `method = "fo"` or `"gh"`.
`cores`	Number of `rxSolve` threads.
`return_samples`	Include the raw `n_sim x length(times)` prediction matrix as `$samples` in each study's output. No effect when `method = "fo"` or `"gh"` (those methods draw no samples).
`resid_nodes`	Gauss-Hermite nodes used to integrate the RESIDUAL for a transform-both-sides endpoint (`boxCox`, `yeoJohnson`, `logitNorm`, `probitNorm`), where `y = g(h(f) + sigma*eps)` has no closed-form mean and variance. Ignored by every other error model. Default 81 – the same default the four estimator controls use, so `datagen()` and the fit it feeds agree unless you deliberately change one of them. See `admControl()` for the measured convergence.

Value

A list of class `"datagenControl"`.

See Also

`datagen()`

Examples

```
ctrl <- datagenControl(n_sim = 2000L)
ctrl$sampling  # "sobol"

# Deterministic FO moments for design evaluation:
datagenControl(method = "fo")$method  # "fo"

# GH quadrature moments (unbiased, noise-free):
datagenControl(method = "gh", n_nodes = 5L)$n_nodes
```

`examplomycin` *Examplomycin dataset*

Description

A simulated pharmacokinetic dataset for the fictional drug examplomycin, intended as a worked example for aggregate data modelling with `admixr2`. The dataset contains 500 subjects, each with 9 observation time points, generated from a two-compartment model with first-order absorption.

Usage

```
examplomycin
```

Format

A data frame with 5000 rows and 6 columns:

- `ID`: Subject identifier (integer, 1–500).
- `TIME`: Time after dose (hours).
- `DV`: Observed plasma concentration (mg/L).
- `AMT`: Dose amount (mg); 100 for dosing records, 0 otherwise.
- `EVID`: Event type (101 = dose, 0 = observation).
- `CMT`: Compartment (1 = depot, 2 = central).

Details

True population parameters:

Parameter	Value
CL (L/hr)	5
V1 (L)	10
V2 (L)	30
Q (L/hr)	10
ka (1/hr)	1
IIV (all, SD on log scale)	0.3
Proportional error (SD)	0.2

Single oral dose of 100 mg; sampling at 0.1, 0.25, 0.5, 1, 2, 3, 5, 8, and 12 hours post-dose.

Source

Generated from a two-compartment PK model using `rxode2::rxSolve()`. See `vignette("admixr2")` for a full modelling example.

Examples

```
data("examplomycin")
head(examplomycin)

# Compute aggregate statistics
obs <- examplomycin[examplomycin$EVID == 0, ]
obs <- obs[order(obs$ID, obs$TIME), ]
times <- sort(unique(obs$TIME))
E <- sapply(times, function(t) mean(obs$DV[obs$TIME == t]))
round(E, 3)
```

---

nlmixr2Est.adfo *Fit an aggregate data model via First-Order (FO) approximation*

Description

Called automatically by `nlmixr2(model, admData(), est = "adfo", control = adfoControl(...))`. Not typically called directly.

Usage

```
## S3 method for class 'adfo'
nlmixr2Est(env, ...)
```

Arguments

`env`	nlmixr2 environment containing `ui` and `control`.
`...`	Unused.

Value

An `admFit` nlmixr2 fit object.

nlmixr2Est.adgh *Fit an aggregate data model via Gauss-Hermite quadrature*

Description

Called automatically by `nlmixr2(model, admData(), est = "adgh", control = adghControl(...))`. Not typically called directly.

Usage

```
## S3 method for class 'adgh'
nlmixr2Est(env, ...)
```

Arguments

`env`	nlmixr2 environment containing `ui` and `control`.
`...`	Unused.

Value

An `admFit` nlmixr2 fit object.

nlmixr2Est.adirmc *Fit an aggregate data model via Iterative Reweighting MC (adirmc estimator)*

Description

Called automatically by `nlmixr2(model, admData(), est = "adirmc", control = adirmcControl(...))`. Not typically called directly.

Usage

```
## S3 method for class 'adirmc'
nlmixr2Est(env, ...)
```

Arguments

`env`	nlmixr2 environment containing `ui` and `control`.
`...`	Unused.

Value

An `admFit` nlmixr2 fit object.

nlmixr2Est.admc *Fit an aggregate data model via Monte Carlo (admc estimator)*

Description

Called automatically by `nlmixr2(model, admData(), est = "admc", control = admControl(...))`. Not typically called directly.

Usage

```
## S3 method for class 'admc'
nlmixr2Est(env, ...)
```

Arguments

`env`	nlmixr2 environment containing `ui` and `control`.
`...`	Unused.

Value

An `admFit` nlmixr2 fit object.

plot.admFit *Diagnostic plots for an admixr2 fit*

Description

Generates up to four diagnostic panels:

Usage

```
## S3 method for class 'admFit'
plot(x, which = c("mean", "cov", "nll", "par"), n_sim = NULL, seed = 1L, ...)
```

Arguments

`x`	An `admFit` object returned by `nlmixr2()` with `est = "adfo"`, `est = "admc"`, `est = "adgh"`, or `est = "adirmc"`.
`which`	Character vector selecting which panel types to produce. Any subset of `c("mean", "cov", "nll", "par")`. Defaults to all four.
`n_sim`	Number of MC samples for the final prediction. Defaults to the value used during fitting. Only used when `"mean"` or `"cov"` is in `which`.
`seed`	Random seed for reproducibility.
`...`	Unused.

Details

1. `"mean"` – Observed vs predicted mean per study (2x2 grid). Upper row: observed and predicted mean lines with +/-1 SD ribbon on a shared y scale (black throughout). Lower row: raw residual lollipop with +/-2 SE band and standardised residual z-scores with +/-1.96 reference lines.
2. `"cov"` – Observed vs predicted (co)variance heatmaps per study (2x2 grid). Upper row shares a common colour scale (blue-white-red). Lower row uses distinct diverging scales: residual (red-white-green) and standardised residual (gold-white-purple). Significance stars overlaid on the standardised residual panel.
3. `"nll"` – NLL trace per restart over optimizer evaluations. Restarts coloured with the Okabe-Ito palette.
4. `"par"` – Parameter trace per restart on the natural scale (struct thetas back-transformed, sigma as SD, omega diagonal as variance labelled `V(eta.x)`). Facets ordered as in the model `ini()` block. Restarts coloured with the Okabe-Ito palette.

Value

A named list of ggplot2 objects, invisibly. Prints each selected top-level panel. For the `"mean"` and `"cov"` panels the returned list also contains each sub-panel individually so a single panel (or a few) can be extracted in code without reprinting the whole grid. Elements can be pulled out by name – `plot(fit, which = "mean")$mean_study1_pred` or `plot(fit, which = "cov")$cov_study1_std_resid` – or by position, with the combined 2x2 grid stored first per study (`plot(fit, which = "mean")[[1]]` is the full grid, `[1]` the length-1 named sub-list). The sub-panel keys are `<type>_<study>_obs`, `_pred`, `_resid`, and `_std_resid`; the combined grid stays under `<type>_<study>`. The extra sub-panel keys are not printed on their own.

Aggregate data slot

Every admixr2 fit also carries the observed and predicted aggregate data in `fit$env$aggData`, a named list with one entry per study. Each entry holds the observation `times`, the study `n`, and two moment sets – `obs` (from the data) and `pred` (predicted at the fitted parameters) – each a list with the mean vector `E` and the (co)variance matrix `V`:

```
fit$env$aggData$study1$obs$E    # observed mean vector
fit$env$aggData$study1$obs$V    # observed covariance matrix
fit$env$aggData$study1$pred$E   # predicted mean vector
fit$env$aggData$study1$pred$V   # predicted covariance matrix
```

The predicted moments are computed by one MC simulation at the fitted parameters using the fit's own `n_sim` and a fixed seed, so they match the default `plot(fit)` mean/cov panels. The slot is absent only when the fit cannot be simulated (no simulation model available).

nlmixr2 `traceplot()`

admixr2 fits also plug into the nlmixr2 `traceplot()` generic. During fitting the parameter iteration history of the best restart is stored on the fit in the standard `parHistData` slot (natural scale), so `traceplot(fit)` produces the familiar per-parameter, free-y facetted trace used elsewhere in the nlmixr2 ecosystem. There is no burn-in marker (admixr2 records optimizer evaluations, not SAEM

iterations), and only the best restart is shown – the per-restart overlay and the NLL trace remain available via `plot(fit, which = c("par", "nll"))`. The trace stores only improving evaluations (steps that lowered the best NLL), so the `iter` axis indexes those improvement steps rather than raw optimizer iterations.

Examples

```
library(rxode2)
library(nlmixr2)

data("examplomycin")
obs    <- examplomycin[examplomycin$EVID == 0, ]
obs    <- obs[order(obs$ID, obs$TIME), ]
times  <- sort(unique(obs$TIME))
ids    <- unique(obs$ID)
dv_mat <- do.call(rbind, lapply(ids, function(i) {
  sub <- obs[obs$ID == i, ]; sub$DV[order(sub$TIME)]
}))
E <- colMeans(dv_mat)
V <- cov.wt(dv_mat, method = "ML")$cov

pk_model <- function() {
  ini({
    tcl <- log(5); tv <- log(30)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v ~ 0.04
  })
  model({
    cl <- exp(tcl + eta.cl)
    v  <- exp(tv  + eta.v)
    d/dt(central) <- -(cl/v) * central
    cp <- central / v
    cp ~ prop(prop.sd)
  })
}

fit <- nlmixr2(
  pk_model, admData(), est = "adfo",
  control = adfoControl(
    studies = list(study1 = list(E = E, V = V, n = length(ids),
                                 times = times, ev = et(amt = 100))),
    maxeval = 100L
  )
)
plot(fit)
```

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

Description

Delegates to `print.nlmixr2FitCore` for the standard nlmixr2 coloured output. `admFit` class is kept on the object during the call so that `head.admFit` intercepts any `head(fit)` calls that arise in the paged- output path (R Markdown / notebooks), preventing the `[.data.frame(.subset2(env, integer))` crash that occurs when an environment-backed fit is subscripted like a plain list.

Usage

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

Arguments

`x`	An `admFit` object.
`...`	Passed to `print.nlmixr2FitCore`.

Value

`x`, invisibly.

Examples

```
library(rxode2)
library(nlmixr2)

data("examplomycin")
obs    <- examplomycin[examplomycin$EVID == 0, ]
obs    <- obs[order(obs$ID, obs$TIME), ]
times  <- sort(unique(obs$TIME))
ids    <- unique(obs$ID)
dv_mat <- do.call(rbind, lapply(ids, function(i) {
  sub <- obs[obs$ID == i, ]; sub$DV[order(sub$TIME)]
}))
E <- colMeans(dv_mat)
V <- cov.wt(dv_mat, method = "ML")$cov

pk_model <- function() {
  ini({
    tcl <- log(5); tv <- log(30)
    prop.sd <- c(0, 0.2)
    eta.cl ~ 0.09; eta.v ~ 0.04
  })
  model({
    cl <- exp(tcl + eta.cl)
    v  <- exp(tv  + eta.v)
    d/dt(central) <- -(cl/v) * central
    cp <- central / v
    cp ~ prop(prop.sd)
  })
}
```

```
fit <- nlmixr2(
  pk_model, admData(), est = "adfo",
  control = adfoControl(
    studies = list(study1 = list(E = E, V = V, n = length(ids),
                                 times = times, ev = et(amt = 100))),
    maxeval = 100L
  )
)
print(fit)
```

Index