

Package ‘admixr2’

June 2, 2026

Type Package

Title Aggregate Data Modelling

Version 0.1.0

Description

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

RoxygenNote 7.3.3

Imports checkmate, digest, nlmixr2est, nloptr, qs2, randtoolbox, Rcpp,
rxode2

LinkingTo Rcpp, RcppEigen

Suggests expm, furrr, future, ggplot2, knitr, mnormt, nlmixr2,
patchwork, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation yes

Author H. van de Beek [aut, cre],
P.A.J. Väilitalo [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 2026-06-02 10:50:02 UTC

Contents

adfoControl	2
adirmcControl	5
admControl	9
admData	13
admStopWorkers	13
datagen	14
datagenControl	16
examplomycin	17
nlmixr2Est.adfo	18
nlmixr2Est.adirmc	18
nlmixr2Est.admc	19
plot.admFit	19
print.admFit	21
Index	23

adfoControl	<i>Control settings for the FO (First-Order) estimator</i>
-------------	--

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("none", "analytical", "fd", "cfd"),
  algorithm = "NLOPT_LN_BOBYQA",
  maxeval = 500L,
  ftol_rel = .Machine$double.eps^(1/2),
  print = 10L,
  seed = 12345L,
  cores = 1L,
  grad_h = 1e-04,
  grad_bounds = 5,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/5),
```

```

covMethod = c("r", "none"),
n_restarts = 1L,
restart_sd = 0.5,
workers = 1L,
rxControl = NULL,
calcTables = FALSE,
compress = TRUE,
ci = 0.95,
sigdig = 4,
sigdigTable = NULL,
addProp = c("combined2", "combined1"),
optExpression = TRUE,
sumProd = FALSE,
literalFix = TRUE,
returnAdmr = FALSE,
...
)

```

Arguments

studies	Named list of study specifications (same format as <code>admControl()</code> : E, V, n, times, ev, optional method).
grad	Gradient mode. "none" (default) uses derivative-free BOBYQA; "analytical" uses the closed-form FO gradient (requires sensitivity equations); "fd" uses forward finite differences of the full NLL; "cfd" uses central finite differences for struct theta gradient (more accurate than "fd", roughly twice as many NLL evaluations per step).
algorithm	nloptr algorithm. Automatically coerced to "NLOPT_LD_LBFGS" when grad != "none".
maxeval	Maximum function evaluations (default 500).
ftol_rel	Relative tolerance (default <code>sqrt(.Machine\$double.eps)</code>).
print	Print-frequency for live progress (0 = silent).
seed	Random seed (used for restarts).
cores	OpenMP threads for <code>rxSolve()</code> (default 1).
grad_h	Finite-difference step for unpaired struct theta gradient and FD Jacobian.
grad_bounds	Box-constraint half-width when using gradients.
cov_h	Inner FD step for the gradient-based Hessian (only used when <code>covMethod = "r"</code> and <code>grad != "none"</code>). Default <code>1e-3</code> .
cov_h_outer	Outer step scale for NLL-FD Hessian.
covMethod	"r" computes covariance via numerical Hessian; "none" skips it.
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 PSOCK/fork workers for multi-restart (default 1 = sequential).

rxControl	rxode2::rxControl() object. Created automatically when NULL.
calcTables, compress, ci, sigdig, sigdigTable, optExpression, sumProd, literalFix	Passed to nlmixr2est::foceiControl() for the table/output machinery.
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.
...	Unused arguments (trigger an error).

Value

An adfoControl object (a named list).

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("exemplomycin")
obs <- exemplomycin[exemplomycin$EVID == 0, ]
obs <- obs[order(obs$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)

```

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 = "NLOPT_LN_BOBYQA",
  maxeval = 5000L,
  ftol_rel = .Machine$double.eps,
  print = 1L,
  omega_expansion = 1,
  seed = 12345L,
  cores = 1L,
  grad = c("analytical", "none", "fd"),
  kappa_method = c("exact", "linearized"),
  grad_h = 1e-04,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/5),
  phases = c(2, 1, 0.5, 0.01),
  convcrit = 1e-05,
  max_worse = 5L,

```

```

covMethod = c("r", "none"),
cov_n_sim = 10000L,
n_restarts = 1L,
restart_sd = 0.2,
workers = 1L,
rxControl = NULL,
calcTables = FALSE,
compress = TRUE,
ci = 0.95,
sigdig = 4,
sigdigTable = NULL,
addProp = c("combined2", "combined1"),
optExpression = TRUE,
sumProd = FALSE,
literalFix = TRUE,
returnAdmr = FALSE,
...
)

```

Arguments

<code>studies</code>	Named list of study specifications. Each element is a list with: <ul style="list-style-type: none"> • <code>E</code> – observed mean vector • <code>V</code> – observed covariance matrix or variance vector (auto-detected) • <code>n</code> – sample size • <code>times</code> – numeric vector of observation times • <code>ev</code> – <code>rxode2::et()</code> dosing event table • <code>method</code> – "cov" or "var" (optional; auto-detected from <code>V</code>)
<code>n_sim</code>	Number of Monte Carlo samples per NLL evaluation.
<code>outer_iter</code>	Maximum inner optimiser iterations per phase.
<code>sampling</code>	Sampling method for eta draws: "sobol" (Sobol, default), "halton" (Halton), "torus" (Kronecker/torus), "lhs" (Latin hypercube), or "rnorm" (iid normal).
<code>algorithm</code>	nloptr algorithm string. Automatically switched to "NLOPT_LD_LBFGS" when <code>grad != "none"</code> .
<code>maxeval</code>	Maximum number of optimizer function evaluations.
<code>ftol_rel</code>	Relative function-value tolerance for convergence.
<code>print</code>	Print progress every this many evaluations (0 = silent).
<code>omega_expansion</code>	Inflate proposal Omega by this factor (≥ 1).
<code>seed</code>	Random seed for reproducibility.
<code>cores</code>	Number of OpenMP threads for <code>rxSolve()</code> .
<code>grad</code>	Gradient mode for the inner optimiser: "analytical" (default, closed-form weight-path gradient), "none" (derivative-free BOBYQA), or "fd" (finite differences). Note: "sens" and "cfd" are 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 $f(\theta, \theta)$ via rxSolve at each inner step) or "linearized" (precomputes $J = df/d(\theta)$ once per outer iteration, approximates kappa via linear expansion — zero rxSolve per inner step).
grad_h	Step size for finite-difference gradient evaluation during optimization (used by grad = "fd" or "cfd"). The default 1e-4 is near the optimal balance between truncation error (grows with h) and MC noise amplification (grows as 1/h) for forward FD. Central FD ("cfd") has a slightly wider optimum around 1e-3, but 1e-4 works well for both.
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 $\sigma / \text{cov_h}$; the Hessian divides that noise by the outer step, giving total noise $\sigma / (\text{cov_h} * \text{cov_h_outer} * p)$. 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 $\max(p , 0.1) * \text{cov_h_outer}$. Applied to both the gradient-FD Hessian (grad != "none") and the NLL-FD Hessian (grad = "none"). Default $\text{eps}^{(1/5)}$ (~2.5e-3) is larger than the textbook $\text{eps}^{(1/4)}$ 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 $ \text{approx} - \text{exact} < \text{convcrit}$.
max_worse	Stop a phase after this many consecutive worsening iterations.
covMethod	Covariance method: "r" (numerical Hessian) or "none".
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 use a Psock cluster on Windows and fork workers on Unix/macOS. Workers are stopped automatically after the restart phase so all cores are available for the Hessian step.
rxControl	rxode2::rxControl() object. Created automatically when NULL.
calcTables, compress, ci, sigdig, sigdigTable, optExpression, sumProd, literalFix	Passed to nlmixr2est::foceiControl() for the table/output machinery.
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).

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

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("examplomyacin")
obs <- examplomyacin[examplomyacin$EVID == 0, ]
obs <- obs[order(obs$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({
    tc1 <- log(5); tv1 <- log(12); tv2 <- log(25)
    tq <- log(12); tka <- log(1.2)
    prop.sd <- c(0, 0.2)
    eta.c1 ~ 0.09; eta.v1 ~ 0.09; eta.v2 ~ 0.09
    eta.q ~ 0.09; eta.ka ~ 0.09
  })
  model({
    c1 <- exp(tc1 + eta.c1); 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 - (c1/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 = "NLOPT_LN_BOBYQA",
  maxeval = 500L,
  ftol_rel = .Machine$double.eps^2,
  print = 10L,
  seed = 12345L,
  cores = 1L,
  grad = c("sens", "fd", "cfd", "none"),
  grad_h = 1e-04,
  cov_h = 0.001,
  cov_h_outer = .Machine$double.eps^(1/5),
  grad_bounds = 5,
  covMethod = c("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 = 4,
    sigdigTable = NULL,
    addProp = c("combined2", "combined1"),
    optExpression = TRUE,
    sumProd = FALSE,
    literalFix = TRUE,
    returnAdmr = FALSE,
    ...
)

```

Arguments

studies	Named list of study specifications. Each element is a list with: <ul style="list-style-type: none"> • 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)
n_sim	Number of Monte Carlo samples per NLL evaluation.
sampling	Sampling method for eta draws: "sobel" (Sobol, default), "halton" (Halton), "torus" (Kronecker/torus), "lhs" (Latin hypercube), or "rnorm" (iid normal).
algorithm	nloptr algorithm string. Automatically switched to "NLOPT_LD_LBFGS" when grad != "none".
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().
grad	Gradient mode: "sens" (sensitivity equations, default), "fd" (forward finite differences), "cfd" (central finite differences), or "none" (derivative-free). A warning is issued when "sens" is requested but the sensitivity model is unavailable; the estimator then falls back to forward finite differences.
grad_h	Step size for finite-difference gradient evaluation during optimization (used by grad = "fd" or "cfd"). The default 1e-4 is near the optimal balance between truncation error (grows with h) and MC noise amplification (grows as 1/h) for forward FD. Central FD ("cfd") has a slightly wider optimum around 1e-3, but 1e-4 works well for both.
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 $\max(p , 0.1) * \text{cov_h_outer}$. Applied to both the gradient-FD Hessian (grad != "none") and the NLL-FD Hessian (grad = "none"). Default $\text{eps}^{(1/5)}$ (~2.5e-3) is larger than the textbook $\text{eps}^{(1/4)}$ 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.
covMethod	Covariance method: "r" (numerical Hessian) or "none".
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 use a PSOCK cluster on Windows and fork workers on Unix/macOS. Workers are stopped automatically after the restart phase so all cores are available for the Hessian step.
rxControl	rxode2::rxControl() object. Created automatically when NULL.
calcTables, compress, ci, sigdig, sigdigTable, optExpression, sumProd, literalFix	Passed to nlmixr2est::foceiControl() for the table/output machinery.
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).
...	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("exemplomycin")
obs <- exemplomycin[exemplomycin$EVID == 0, ]
obs <- obs[order(obs$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({
    tc1 <- log(5); tv1 <- log(12); tv2 <- log(25)
    tq <- log(12); tka <- log(1.2)
    prop.sd <- c(0, 0.2)
    eta.c1 ~ 0.09; eta.v1 ~ 0.09; eta.v2 ~ 0.09
    eta.q ~ 0.09; eta.ka ~ 0.09
  })
  model({
    c1 <- exp(tc1 + eta.c1); 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 - (c1/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	<i>Dummy data frame for nlmixr2 dispatch</i>
---------	--

Description

Returns a minimal NONMEM-style data frame that satisfies nlmixr2's data argument requirement. All DV values are NA so nlmixr2 adds zero $\log(2\pi)$ constants to OBJF, keeping `fit$objective == our -2LL` exactly.

Usage

```
admData()
```

Value

A data frame with columns ID, TIME, DV, AMT, EVID, CMT.

Examples

```
admData()
```

admStopWorkers	<i>Stop PSOCK workers</i>
----------------	---------------------------

Description

Stops any PSOCK worker processes 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()
```

datagen	<i>Generate aggregate study data from (possibly different) pharmacometric models</i>
---------	--

Description

Simulates population mean vectors (E) and covariance matrices (V) for each study using Monte Carlo integration over the IIV distribution. 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 <code>nlmixr2</code> -style model function with <code>ini()</code> and <code>model()</code> 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 <code>model</code> argument. times Numeric vector of observation times. ev A dosing event table created with <code>rxode2::et()</code> . n (Optional) integer sample size; stored as metadata and used when supplying the result to <code>admControl()</code> .
model	Optional default model function used for any study that does not supply its own model element. At least one of <code>model</code> or each study's <code>model</code> must be non-NULL.
control	A <code>datagenControl()</code> object.

Details

Population moments are computed via the same Monte Carlo engine as `est = "admc"`:

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

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

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).

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 ($\text{length}(\text{times}) \times \text{length}(\text{times})$, ML denominator n_{sim}). Residual error is added to the diagonal when `control$add_residual_error = TRUE`.

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

times Observation times.

ev Dosing event table.

samples Raw $n_{\text{sim}} \times \text{length}(\text{times})$ 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(  
  n_sim = 5000L,  
  sampling = c("sobol", "halton", "torus", "lhs", "rnorm"),  
  seed = 12345L,  
  cores = 1L,  
  add_residual_error = TRUE,  
  return_samples = FALSE  
)
```

Arguments

n_sim	Number of Monte Carlo samples used to approximate population moments.
sampling	Quasi-random sampling method: "sobol" (default), "halton", "torus", "lhs", or "rnorm".
seed	Integer seed. Applied before stochastic methods ("rnorm", "lhs").
cores	Number of rxSolve threads.
add_residual_error	Add residual-error variance to the diagonal of V (TRUE by default), matching the admixr2 NLL convention.
return_samples	Include the raw n_sim x length(times) prediction matrix as \$samples in each study's output.

Value

A list of class "datagenControl".

See Also

[datagen\(\)](#)

Examples

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

 exemplomycin

Exemplomycin dataset

Description

A simulated pharmacokinetic dataset for the fictional drug exemplomycin, 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

```
exemplomycin
```

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("exemplomycin")
head(exemplomycin)

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

```

nlmixr2Est.adfo	<i>Fit an aggregate data model via First-Order (FO) approximation</i>
-----------------	---

Description

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

Usage

```

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

```

Arguments

<code>env</code>	nlmixr2 environment containing ui and control.
<code>...</code>	Unused.

Value

An `admFit` nlmixr2 fit object.

nlmixr2Est.adirmc	<i>Fit an aggregate data model via Iterative Reweighting MC (adirmc estimator)</i>
-------------------	--

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	<i>Fit an aggregate data model via Monte Carlo (admc estimator)</i>
-----------------	---

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	<i>Diagnostic plots for an admixr2 fit</i>
-------------	--

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 = "adm", 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 plot.

Examples

```
library(rxode2)
library(nlmixr2)

data("exemplomycin")
obs <- exemplomycin[exemplomycin$EVID == 0, ]
obs <- obs[order(obs$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("exemplomycin")
obs <- exemplomycin[exemplomycin$EVID == 0, ]
obs <- obs[order(obs$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

* datasets

- exemplomycin, [17](#)

- adfoControl, [2](#)
- adirmcControl, [5](#)
- adirmcControl(), [4](#)
- admControl, [9](#)
- admControl(), [3](#), [4](#), [15](#)
- admData, [13](#)
- admStopWorkers, [13](#)

- datagen, [14](#)
- datagen(), [16](#)
- datagenControl, [16](#)
- datagenControl(), [14](#), [15](#)

- exemplomycin, [17](#)

- nlmixr2Est.adfo, [18](#)
- nlmixr2Est.adirmc, [18](#)
- nlmixr2Est.admc, [19](#)

- plot.admFit, [19](#)
- print.admFit, [21](#)