Package 'nmw'

June 9, 2026

Version 0.3.1

Date 2026-06-09

Title Understanding Nonlinear Mixed Effects Modeling for Population Pharmacokinetics

Description This shows how 'NONMEM' (Beal SL, Sheiner LB, Boeckmann AJ, Bauer RJ. NONMEM 7.5 Users Guides. Icon plc, 2020) software works. 'NONMEM' classical estimation methods such as 'First Order (FO) approximation', 'First Order Conditional Estimation (FOCE)', and 'Laplacian approximation' are explained. Functions are also provided for post-run processing of NONMEM output files, generating PDF diagnostic reports including objective function value analysis, parameter estimates, prediction and residual diagnostics, empirical Bayes estimate (EBE) analysis, input data summary, and individual pharmacokinetic parameter distributions. Helper utilities for building NONMEM-ready datasets from SDTM-style source tables are also included.

Depends R (>= 3.5.0), numDeriv

Imports MASS, grDevices, graphics, stats, utils

ByteCompile yes

License GPL-3

Copyright 2017-, Kyun-Seop Bae

Author Kyun-Seop Bae [aut, cre]

Maintainer Kyun-Seop Bae <k@acr.kr>

URL https://cran.r-project.org/package=nmw

Contact Kyun-Seop Bae <k@acr.kr>

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.3

Repository CRAN

Date/Publication 2026-06-09 07:00:08 UTC

Contents

`nmw-package` *Understanding Nonlinear Mixed Effects Modeling for Population Pharmacokinetics*

Description

This shows how NONMEM(R) <http://www.iconplc.com/innovation/nonmem/> software works. Additionally, provides functions for post-run processing of NONMEM output files, generating comprehensive PDF diagnostic reports.

Details

This package explains 'First Order(FO) approximation' method, 'First Order Conditional Estimation(FOCE)' method, and 'Laplacian(LAPL)' method of NONMEM software. It also provides post-run processing functions including PDF diagnostic reports for OFV analysis, parameter estimates, prediction diagnostics, residual diagnostics, EBE analysis, input data summary, and individual PK parameter distributions.

Author(s)

Kyun-Seop Bae <k@acr.kr>

References

1. NONMEM Users guide
2. Wang Y. Derivation of various NONMEM estimation methods. J Pharmacokinet Pharmacodyn. 2007.
3. Kang D, Bae K, Houk BE, Savic RM, Karlsson MO. Standard Error of Empirical Bayes Estimate in NONMEM(R) VI. K J Physiol Pharmacol. 2012.
4. Kim M, Yim D, Bae K. R-based reproduction of the estimation process hidden behind NONMEM Part 1: First order approximation method. 2015.

5. Bae K, Yim D. R-based reproduction of the estimation process hidden behind NONMEM Part 2: First order conditional estimation. 2016.

Examples

```
DataAll = Theoph
colnames(DataAll) = c("ID", "BWT", "DOSE", "TIME", "DV")
DataAll[,"ID"] = as.numeric(as.character(DataAll[,"ID"]))

nTheta = 3
nEta = 3
nEps = 2

THETAinit = c(2, 50, 0.1)
OMinit = matrix(c(0.2, 0.1, 0.1, 0.1, 0.2, 0.1, 0.1, 0.1, 0.2), nrow=nEta, ncol=nEta)
SGinit = diag(c(0.1, 0.1))

LB = rep(0, nTheta) # Lower bound
UB = rep(1000000, nTheta) # Upper bound

FGD = deriv(~DOSE/(TH2*exp(ETA2))*TH1*exp(ETA1)/(TH1*exp(ETA1) - TH3*exp(ETA3))*
             (exp(-TH3*exp(ETA3)*TIME)-exp(-TH1*exp(ETA1)*TIME)),
            c("ETA1","ETA2","ETA3"),
            function.arg=c("TH1", "TH2", "TH3", "ETA1", "ETA2", "ETA3", "DOSE", "TIME"),
            func=TRUE, hessian=TRUE)
H = deriv(~F + F*EPS1 + EPS2, c("EPS1", "EPS2"), function.arg=c("F", "EPS1", "EPS2"), func=TRUE)

PRED = function(THETA, ETA, DATAi)
{
 FGDres = FGD(THETA[1], THETA[2], THETA[3], ETA[1], ETA[2], ETA[3], DOSE=320, DATAi[,"TIME"])
  Gres = attr(FGDres, "gradient")
  Hres = attr(H(FGDres, 0, 0), "gradient")

  if (e$METHOD == "LAPL") {
    Dres = attr(FGDres, "hessian")
    Res = cbind(FGDres, Gres, Hres, Dres[,1,1], Dres[,2,1], Dres[,2,2], Dres[,3,])
   colnames(Res) = c("F", "G1", "G2", "G3", "H1", "H2", "D11", "D21", "D22", "D31", "D32", "D33")
  } else {
    Res = cbind(FGDres, Gres, Hres)
    colnames(Res) = c("F", "G1", "G2", "G3", "H1", "H2")
  }
  return(Res)
}

####### First Order Approximation Method # Commented out for the CRAN CPU time
#InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
#         Pred=PRED, METHOD="ZERO")
#(EstRes = EstStep())           # 4 sec
#(CovRes = CovStep())           # 2 sec
#PostHocEta() # Using e$FinalPara from EstStep()
#TabStep()

######## First Order Conditional Estimation with Interaction Method
```

```
#InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
#         Pred=PRED, METHOD="COND")
#(EstRes = EstStep())           # 2 min
#(CovRes = CovStep())           # 1 min
#get("EBE", envir=e)
#TabStep()

######## Laplacian Approximation with Interaction Method
#InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
#         Pred=PRED, METHOD="LAPL")
#(EstRes = EstStep())           # 4 min
#(CovRes = CovStep())           # 1 min
#get("EBE", envir=e)
#TabStep()
```

AddCox *Add a Covariate Column to an Existing NONMEM dataset*

Description

A new covariate column can be added to an existing NONMEM dataset.

Usage

```
AddCox(nmData, coxData, coxCol, dateCol = "DATE", idCol = "ID")
```

Arguments

nmData	an existing NONMEM dataset
coxData	a data table containing a covariate column
coxCol	the covariate column name in the coxData table
dateCol	date column name in the NONMEM dataset and the covariate data table
idCol	ID column name in the NONMEM dataset and the covariate data table

Details

It first carry forward for the missing data. If NA is remained, it carry backward.

Value

A new NONMEM dataset containing the covariate column

Author(s)

Kyun-Seop Bae <k@acr.kr>

AddDoNoTaLD *Add Dose Number, Time of Latest Dose, and Time after Latest Dose*

Description

Adds DoNo (Dosing Occasion Number), ToLD (Time of Latest Dose), and TaLD (Time after Latest Dose) columns to a NONMEM dataset.

Usage

```
AddDoNoTaLD(
  NMData,
  ID = "ID",
  TIME = "TIME",
  AMT = "AMT",
  II = "II",
  ADDL = "ADDL",
  MDV = "MDV"
)
```

Arguments

NMData	data.frame of NONMEM dataset
ID	character, column name for subject ID
TIME	character, column name for time
AMT	character, column name for dose amount
II	character, column name for interdose interval
ADDL	character, column name for additional doses
MDV	character, column name for missing dependent variable flag

Value

data.frame with DoNo, ToLD, TaLD columns added

AddPage *Add a New Page to PDF*

Description

Creates a new page in the PDF output with optional headers and footers.

Usage

```
AddPage(
  Cex = 0.8,
  Header1 = "",
  Header2 = "",
  Header3 = "",
  Footer1 = "",
  Footer2 = "",
  Footer3 = "",
  PrintRowNum = FALSE,
  StartRowNum = 1
)
```

Arguments

Cex	numeric, character expansion factor (0.8 or 0.6)
Header1	character, left header
Header2	character, center header
Header3	character, right header
Footer1	character, left footer
Footer2	character, center footer
Footer3	character, right footer
PrintRowNum	logical, whether to print row numbers
StartRowNum	integer, starting row number

---

add_crcl_cg *Add CRCL Column via Cockcroft-Gault*

Description

Add a CRCL column to a NM-style data.frame using Cockcroft-Gault. Column names are configurable.

Usage

```
add_crcl_cg(
  df,
  age_col = "AGE",
  bwt_col = "BWT",
  sex_col = "SEX",
  crea_col = "CREA",
  out_col = "CRCL"
)
```

Arguments

df	data.frame containing AGE, BWT, SEX, CREA columns.
age_col	name of the age column (years). Default "AGE".
bwt_col	name of the body-weight column (kg). Default "BWT".
sex_col	name of the sex column (0=M, 1=F). Default "SEX".
crea_col	name of the creatinine column (mg/dL). Default "CREA".
out_col	name of the output column. Default "CRCL".

Value

data.frame with the CRCL column added.

BtwTagMat *Extract Matrix from NONMEM XML Output*

Description

Parses NONMEM XML output to extract omega or sigma matrices.

Usage

```
BtwTagMat(Tag, RawRead, nRow)
```

Arguments

Tag	character, the tag name (e.g., "omega", "sigma")
RawRead	character vector of XML lines
nRow	integer, dimension of the matrix

Value

numeric matrix

BtwTagVals *Extract Vector of Values Between XML Tags*

Description

Parses NONMEM XML output to extract theta, omega, or sigma vectors.

Usage

```
BtwTagVals(Tag, RawRead)
```

Arguments

Tag	character, the tag name (e.g., "nm:theta")
RawRead	character vector of XML lines

Value

numeric vector of values

build_dose_records *Build NONMEM Dose Records*

Description

Build NONMEM dose records from an EX-like data.frame. EX must contain: SUBJID, DAT2, TIME, AMT, RATE. Optional: CMT (defaults to dose_cmt). Output rows have MDV = 1, DV = 0.

Usage

```
build_dose_records(EX, dose_cmt = 1L)
```

Arguments

EX	data.frame with dose information.
dose_cmt	default compartment for dose records lacking a CMT column.

Value

data.frame of dose records with columns SUBJID, DAT2, TIME, AMT, RATE, CMT, DV, MDV.

`build_nm_dataset` *Build a NONMEM-Format Dataset from DM/EX/PC*

Description

Build a NONMEM-format dataset from cleaned DM/EX/PC tables. Time-varying covariates (VS, LB, ...) are NOT merged here. Use `merge_cov_locf` afterwards to add them. CRCL is added separately via `add_crcl_cg`. Molar conversion is done with `nm_to_molar`.

Usage

```
build_nm_dataset(
  DM,
  EX,
  PC,
  IDs = NULL,
  id_prefix = "",
  id_func = NULL,
  dose_cmt = 1L,
  verbose = FALSE
)
```

Arguments

`DM`	data.frame with one row per subject. Must contain SUBJID; any other columns are merged in as subject-level constants.
`EX`	dose records (see `build_dose_records`).
`PC`	observation records (see `build_obs_records`).
`IDs`	character vector of SUBJIDs to keep. NULL (default) is the intersect of EX and PC where DV > 0.
`id_prefix`	character. When `id_func` is NULL, ID = paste0(id_prefix, SUBJID).
`id_func`	function(SUBJID) -> character ID. Overrides `id_prefix`.
`dose_cmt`	default compartment for dose records lacking a CMT column.
`verbose`	print progress. Default FALSE.

Details

Records are sorted by SUBJID, DAT2, TIME, CMT, MDV, AMT. At tied (SUBJID, DAT2, TIME, CMT, MDV), AMT = 0 (observation) sorts before AMT > 0 (dose) so a pre-dose observation precedes the dose given at the same minute.

Value

data.frame with columns ID, SUBJID, DAT2, TIME, AMT, RATE, CMT, DV, MDV, <DM-columns ...>.

build_obs_records *Build NONMEM Observation Records*

Description

Build NONMEM observation records from a PC-like data.frame. PC must contain: SUBJID, DAT2, TIME, DV, CMT. Output rows have AMT = 0, RATE = 0; DV that is NA or 0 -> MDV = 1.

Usage

```
build_obs_records(PC)
```

Arguments

PC	data.frame with observation information.

Value

data.frame of observation records with columns SUBJID, DAT2, TIME, AMT, RATE, CMT, DV, MDV.

ClosePDF *Close PDF Output*

Description

Closes the current PDF device.

Usage

```
ClosePDF()
```

code_race *Code RACE as Integer*

Description

Code RACE as integer: ASIAN -> 1, WHITE -> 2, BLACK -> 3, otherwise -> 4. Numeric input is returned unchanged (cast to integer).

Usage

```
code_race(x)
```

Arguments

x	character or numeric vector of race values.

Value

integer vector.

---

code_sex *Code SEX as Integer*

Description

Code SEX as integer: M -> 0, F -> 1, otherwise -> 2. Numeric input is returned unchanged (cast to integer).

Usage

```
code_sex(x)
```

Arguments

x	character or numeric vector of sex values.

Value

integer vector.

CombDmExPc *Combine the demographics(DM), dosing(EX), and DV(PC) tables into a new NONMEM dataset*

Description

A new NONMEM dataset can be created from the demographics, dosing, and DV tables.

Usage

```
CombDmExPc(dm, ex, pc)
```

Arguments

dm	A demographics table. It should contain a row per subject.
ex	An exposure table. Drug administration (dosing) history table.
pc	A DV(dependent variable) or PC(drug concentration) table

Details

Combining a demographics, a dosing, and a concentration table can produce a new NONMEM dataset.

Value

A new NONMEM dataset

Author(s)

Kyun-Seop Bae <k@acr.kr>

`CovStep` *Covariance Step*

Description

It calculates standard errors and various variance matrices with the `e$FinalPara` after estimation step.

Usage

```
CovStep()
```

Details

Because `EstStep` uses nonlinear optimization, covariance step is separated from estimation step. It calculates variance-covariance matrix of estimates in the original scale.

Value

`Time`	consumed time
`Standard Error`	standard error of the estimates in the order of theta, omega, and sigma
`Covariance Matrix of Estimates`	covariance matrix of estimates in the order of theta, omega, and sigma. This is inverse(R) x S x inverse(R) by default.
`Correlation Matrix of Estimates`	correlation matrix of estimates in the order of theta, omega, and sigma
`Inverse Covariance Matrix of Estimates`	inverse covariance matrix of estimates in the order of theta, omega, and sigma
`Eigen Values`	eigen values of covariance matrix
`R Matrix`	R matrix of NONMEM, the second derivative of log likelihood function with respect to estimation parameters
`S Matrix`	S matrix of NONMEM, sum of individual cross-product of the first derivative of log likelihood function with respect to estimation parameters

Author(s)

Kyun-Seop Bae <k@acr.kr>

References

NONMEM Users Guide

See Also

EstStep, InitStep

Examples

```
# Only after InitStep and EstStep
#CovStep()
```

`crcl_cg` *Cockcroft-Gault Creatinine Clearance*

Description

Cockcroft-Gault creatinine clearance (mL/min).

Usage

```
crcl_cg(age, bwt, sex, crea)
```

Arguments

`age`	numeric, age in years.
`bwt`	numeric, body weight in kg.
`sex`	integer/numeric, 0 = male, 1 = female.
`crea`	numeric, serum creatinine in mg/dL.

Value

numeric vector of CRCL in mL/min.

dat2_time_to_posix *Combine Date and Time Strings into POSIXct*

Description

Combine "YYYY-MM-DD" and "HH:MM" character vectors into POSIXct.

Usage

```
dat2_time_to_posix(dat2, time, tz = "UTC")
```

Arguments

dat2	character vector of dates in "YYYY-MM-DD" format.
time	character vector of times in "HH:MM" format.
tz	time zone. Default "UTC".

Value

POSIXct vector.

DxPlotPost *Diagnostic Plot for Post-Processing*

Description

Creates spaghetti-style diagnostic plots with individual ID labels.

Usage

```
DxPlotPost(x, y, mat, xlbl, ylbl, smooth, xlm = "", ylm = "", Log = "")
```

Arguments

x	numeric vector, x-axis values
y	numeric vector, y-axis values
mat	data.frame with an ID column
xlbl	character, x-axis label
ylbl	character, y-axis label
smooth	character, "T" for lowess smoothing, "F" for identity line
xlm	numeric vector of length 2, x-axis limits
ylm	numeric vector of length 2, y-axis limits
Log	character, log transformation for axes (e.g., "y")

EBEpair *EBE Pair Plot*

Description

Creates pairs plot of ETAs vs covariates with histograms and correlations.

Usage

```
EBEpair(tabEta, RunNumber, each = FALSE)
```

Arguments

tabEta	data.frame with ID, covariates, and ETA columns
RunNumber	character, model/run identifier
each	logical, if TRUE creates separate plots per covariate

EstStep *Estimation Step*

Description

This estimates upon the conditions with InitStep.

Usage

```
EstStep()
```

Details

It does not have arguments. All necessary arguments are stored in the e environment. It assumes "INTERACTION" between eta and epsilon for "COND" and "LAPL" options. The output is basically same to NONMEM output.

Value

Initial OFV	initial value of the objective function
Time	time consumed for this step
Optim	the raw output from optim function
Final Estimates	final estimates in the original scale

Author(s)

Kyun-Seop Bae <k@acr.kr>

References

NONMEM Users Guide

See Also

InitStep

Examples

```
# Only After InitStep
#EstStep()
```

ExpandDoseHist *Expand Dose History with ADDL/II Records*

Description

Expands compressed dosing records (ADDL/II) into individual dose records.

Usage

```
ExpandDoseHist(DoseHistTab)
```

Arguments

DoseHistTab	data.frame with columns TIME, AMT, II, ADDL

Value

data.frame with expanded dose records

GetAICc *Get Corrected AIC (AICc)*

Description

Get Corrected AIC (AICc)

Usage

```
GetAICc()
```

Value

numeric, the corrected AIC value

GetCountEps *Get Count of Epsilons*

Description

Get Count of Epsilons

Usage

```
GetCountEps()
```

Value

integer, number of epsilon parameters, or NA if file not found

GetCountEta *Get Count of Etas*

Description

Get Count of Etas

Usage

```
GetCountEta()
```

Value

integer, number of eta parameters, or NA if file not found

GetCountObs *Get Count of Observations*

Description

Get Count of Observations

Usage

```
GetCountObs()
```

Value

integer, total number of observation records

GetCountPara *Get Count of Parameters*

Description

Get Count of Parameters

Usage

```
GetCountPara()
```

Value

integer, total number of estimated parameters, or NA if file not found

GetEstMethod *Get Estimation Method*

Description

Identifies the estimation method from the .ext file header.

Usage

```
GetEstMethod()
```

Value

character, abbreviation of the estimation method

GetOFV *Get Objective Function Value*

Description

Retrieves the final OFV from the .ext file in the current directory.

Usage

```
GetOFV()
```

Value

numeric, the objective function value

`InitStep` *Initialization Step*

Description

It receives parameters for the estimation and stores them into `e` environment.

Usage

```
InitStep(DataAll, THETAinit, OMinit, SGinit, LB, UB, Pred, METHOD)
```

Arguments

`DataAll`	Data for all subjects. It should contain columns which `Pred` function uses.
`THETAinit`	Theta initial values
`OMinit`	Omega matrix initial values
`SGinit`	Sigma matrix initial values
`LB`	Lower bounds for theta vector
`UB`	Upper bounds for theta vector
`Pred`	Prediction function name
`METHOD`	one of the estimation methods `"ZERO"`, `"COND"`, or `"LAPL"`

Details

Prediction function should return not only prediction values(F or IPRED) but also G (first derivative with respect to etas) and H (first derivative of Y with respect to epsilon). For the `"LAPL"`, prediction function should return second derivative with respect to eta also. "INTERACTION" is `TRUE` for `"COND"` and `"LAPL"` option, and `FALSE` for `"ZERO"`. Omega matrix should be full block one. Sigma matrix should be diagonal one.

Value

This does not return values, but stores necessary values into the environment `e`.

Author(s)

Kyun-Seop Bae <k@acr.kr>

References

NONMEM Users Guide

Examples

```
DataAll = Theoph
colnames(DataAll) = c("ID", "BWT", "DOSE", "TIME", "DV")
DataAll[,"ID"] = as.numeric(as.character(DataAll[,"ID"]))

nTheta = 3
nEta = 3
nEps = 2

THETAinit = c(2, 50, 0.1) # Initial estimate
OMinit = matrix(c(0.2, 0.1, 0.1, 0.1, 0.2, 0.1, 0.1, 0.1, 0.2), nrow=nEta, ncol=nEta)
OMinit
SGinit = diag(c(0.1, 0.1))
SGinit

LB = rep(0, nTheta) # Lower bound
UB = rep(1000000, nTheta) # Upper bound

FGD = deriv(~DOSE/(TH2*exp(ETA2))*TH1*exp(ETA1)/(TH1*exp(ETA1) - TH3*exp(ETA3))*
             (exp(-TH3*exp(ETA3)*TIME)-exp(-TH1*exp(ETA1)*TIME)),
            c("ETA1","ETA2","ETA3"),
            function.arg=c("TH1", "TH2", "TH3", "ETA1", "ETA2", "ETA3", "DOSE", "TIME"),
            func=TRUE, hessian=TRUE)
H = deriv(~F + F*EPS1 + EPS2, c("EPS1", "EPS2"), function.arg=c("F", "EPS1", "EPS2"), func=TRUE)

PRED = function(THETA, ETA, DATAi)
{
 FGDres = FGD(THETA[1], THETA[2], THETA[3], ETA[1], ETA[2], ETA[3], DOSE=320, DATAi[,"TIME"])
  Gres = attr(FGDres, "gradient")
  Hres = attr(H(FGDres, 0, 0), "gradient")

  if (e$METHOD == "LAPL") {
    Dres = attr(FGDres, "hessian")
    Res = cbind(FGDres, Gres, Hres, Dres[,1,1], Dres[,2,1], Dres[,2,2], Dres[,3,])
   colnames(Res) = c("F", "G1", "G2", "G3", "H1", "H2", "D11", "D21", "D22", "D31", "D32", "D33")
  } else {
    Res = cbind(FGDres, Gres, Hres)
    colnames(Res) = c("F", "G1", "G2", "G3", "H1", "H2")
  }
  return(Res)
}

######### First Order Approximation Method
InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
         Pred=PRED, METHOD="ZERO")

######### First Order Conditional Estimation with Interaction Method
InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
         Pred=PRED, METHOD="COND")

######### Laplacian Approximation with Interaction Method
InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
```

```
Pred=PRED, METHOD="LAPL")
```

`locf_value` *LOCF / LOCB Lookup*

Description

For each target time, return the most-recent source value at source_t <= target_t (LOCF). If no preceding source exists, return the earliest source_v at source_t > target_t (LOCB). Empty source returns `fallback` for all targets.

Usage

```
locf_value(target_t, source_t, source_v, fallback = NA_real_)
```

Arguments

`target_t`	vector of target times (numeric or POSIXct).
`source_t`	vector of source times (same type as target_t).
`source_v`	numeric vector of source values, parallel to source_t.
`fallback`	value used when no source is available. Default NA.

Value

numeric vector, length(target_t).

`LogDV` *Check if Model Uses Log-Transformed DV (from Control File)*

Description

Check if Model Uses Log-Transformed DV (from Control File)

Usage

```
LogDV(CtlFileName)
```

Arguments

`CtlFileName`	character, path to NONMEM control file

Value

logical, TRUE if LOG(F) is found after $PRED or $ERROR

LogDV2 *Check if Model Uses Log-Transformed DV (from FSUBS)*

Description

Check if Model Uses Log-Transformed DV (from FSUBS)

Usage

```
LogDV2()
```

Value

logical, TRUE if LOG(F) is found after SUBROUTINE PRED or ERROR in FSUBS

merge_cov_locf *Merge Time-Varying Covariates by LOCF*

Description

Merge time-varying covariate columns into a NM-style data.frame using LOCF (with LOCB for records before the first source). Subjects/records still NA after LOCF/LOCB can be filled with the column-wise median of `cov_df`.

Usage

```
merge_cov_locf(
  nm,
  cov_df,
  value_cols,
  time_col,
  median_fb = TRUE,
  verbose = FALSE,
  subj_col = "SUBJID",
  dat_col = "DAT2",
  tim_col = "TIME"
)
```

Arguments

`nm`	data.frame with key columns `subj_col`, `dat_col`, `tim_col`.
`cov_df`	data.frame with `subj_col`, `time_col`, and one or more value columns.
`value_cols`	character vector of value columns in `cov_df` to bring into `nm`.
`time_col`	name of the time column in `cov_df`, parsed by `parse_dtc`.

median_fb	logical. If TRUE (default), fill remaining NA values with the column-wise median of cov_df.
verbose	logical. Print median fill counts. Default FALSE.
subj_col	subject ID column name. Default "SUBJID".
dat_col	date column name in nm. Default "DAT2".
tim_col	time column name in nm. Default "TIME".

Value

nm with value_cols added.

MinSuccess *Check Minimization Success*

Description

Check Minimization Success

Usage

```
MinSuccess()
```

Value

logical, TRUE if minimization was successful

---

mlr2 *Multiple Linear Regression with Influence Diagnostics*

Description

Performs multiple linear regression and computes influence diagnostics including DFFITS, DFBETAS, Cook's Distance, COVRATIO, and R-Student.

Usage

```
mlr2(y, x.raw, standardize = 0)
```

Arguments

y	numeric vector of response
x.raw	data.frame of predictors
standardize	integer, standardization method (0=none, 1=center, 2=scale by mean, 3=z-score)

Value

list with model estimates and influence diagnostics

NMIDStat *NONMEM Individual (ID) Statistics*

Description

Computes per-subject statistics including record counts, DV counts, dosing information, and sorting checks.

Usage

```
NMIDStat(NMTable)
```

Arguments

NMTable	data.frame of NONMEM data

Value

list with IDStat data.frame, sortedID logical, sortedDT logical

NMVarStat *NONMEM Variable Statistics*

Description

Computes statistics for each variable in a NONMEM dataset including counts of NA, zero, positive values, unique values, and functional dependency checks.

Usage

```
NMVarStat(NMTable)
```

Arguments

NMTable	data.frame of NONMEM data

Value

matrix with variable statistics

nmw_report_ebe *Generate EBE Diagnostics Report (S5-EBE.PDF)*

Description

Generates a PDF report with empirical Bayes estimate (EBE) analysis including ETA distributions, shrinkage, covariate relationships, correlations, and multiple linear regression with influence diagnostics.

Usage

```
nmw_report_ebe(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_indipk *Generate Individual PK Parameter Report (SC-IndiPK.PDF)*

Description

Generates a PDF report with individual PK parameter distributions including summary statistics, histograms, and QQ plots.

Usage

```
nmw_report_indipk(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_input *Generate Input Data Summary Report (SA-Input.PDF)*

Description

Generates a PDF report with input data validation including individuals with no dosing/DV, duplicate records, and summary statistics.

Usage

```
nmw_report_input(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_ofv *Generate OFV Diagnostic Report (S1-OFV.PDF)*

Description

Generates a PDF report with objective function value diagnostics including total OFV, AICc, BIC, individual OFV per DV summary, and gradient analysis.

Usage

```
nmw_report_ofv(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_output *Generate NONMEM Output Report (SB-Output.PDF)*

Description

Includes the NONMEM PRINT.OUT file content in a PDF report.

Usage

```
nmw_report_output(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_param *Generate Parameter Estimates Report (S2-Parameters.PDF)*

Description

Generates a PDF report with theta, omega, and sigma parameter estimates, standard errors, confidence intervals, and significance flags.

Usage

```
nmw_report_param(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_pred *Generate Prediction Diagnostics Report (S3-Predictions.PDF)*

Description

Generates a PDF report with spaghetti plots, DV vs PRED/IPRE, and individual fitting curves.

Usage

```
nmw_report_pred(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_report_resid *Generate Residual Diagnostics Report (S4-Residuals.PDF)*

Description

Generates a PDF report with WRES, CWRES, IWRES diagnostics including scatter plots, histograms, normality tests, binomial count tests, and individual residual curves with run tests.

Usage

```
nmw_report_resid(run_dir = getwd())
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)

nmw_run *Run All NONMEM Post-Processing Reports*

Description

Generates all diagnostic PDF reports for a NONMEM run in sequence: S1-OFV, S2-Parameters, S3-Predictions, S4-Residuals, S5-EBE, SA-Input, SB-Output, SC-IndiPK.

Usage

```
nmw_run(
  run_dir = getwd(),
  reports = c("ofv", "param", "pred", "resid", "ebe", "input", "output", "indipk")
)
```

Arguments

run_dir	character, path to the NONMEM run directory (default: current directory)
reports	character vector, which reports to generate. Options: "ofv", "param", "pred", "resid", "ebe", "input", "output", "indipk". Default is all reports.

Examples

```
## Not run:
# Generate all reports
nmw_run("/path/to/nonmem/run/directory")

# Generate only OFV and parameter reports
nmw_run("/path/to/run", reports = c("ofv", "param"))

## End(Not run)
```

nm_to_molar *Convert NM Dataset from Mass to Molar Units*

Description

Convert AMT/RATE/DV columns of a NM data.frame from mass units to molar units.

Usage

```
nm_to_molar(
  df,
  mw_dose,
  mw_dv = mw_dose,
  amt_cols = c("AMT", "RATE"),
  dv_col = "DV",
  cmt_col = "CMT",
  dose_factor = 1000,
  dv_factor = 1
)
```

Arguments

df	data.frame.
mw_dose	numeric scalar. MW for AMT and RATE columns.
mw_dv	numeric scalar OR named numeric. Scalar: applied to all DV regardless of CMT. Named: names are CMT values; rows whose CMT is not in names() are left unchanged. Default = mw_dose.
amt_cols	columns to convert with mw_dose. Default c("AMT", "RATE").
dv_col	DV column. Default "DV".
cmt_col	CMT column. Default "CMT".
dose_factor	multiplier applied after AMT/RATE divided by mw_dose. 1000 = mg -> umol (default).
dv_factor	multiplier applied after DV divided by mw_dv. 1 = ng/mL -> umol/L (default).

Details

Typical use:

- AMT, RATE in mg, DV in ng/mL, MW in g/mol
- mg / MW * 1000 -> umol
- ng/mL / MW -> umol/L

For multiple analytes (e.g., parent at CMT=1 and metabolite at CMT=3), supply `mw_dv` as a named numeric vector with names equal to the CMT values (as character): `c(`1` = 394.47, `3` = 380.47)`.

Value

data.frame with converted columns; other columns unchanged.

Outline — *Draw Model Development Flow Diagram*

Description

Creates a graphical model development flow diagram from SumOut results.

Usage

```
Outline(MDL, MDLName, out.start = "0", out.end = "zzzzzzzz", Target = "PDF")
```

Arguments

MDL	data.frame from SumOut
MDLName	character, model development name for title
out.start	character, filter start
out.end	character, filter end
Target	character, "PDF" or other

ParseOut — *Parse Value After Tag from NONMEM Output*

Description

Extracts the value following a tag string (after the colon) in NONMEM output lines.

Usage

```
ParseOut(TagString, RawRead)
```

Arguments

TagString	character, the tag to search for
RawRead	character vector of lines from NONMEM output

Value

character, the value after the tag

parse_dtc *Parse SDTM-Style DTC Strings into POSIXct*

Description

Parse SDTM-style datetime strings into POSIXct. Accepts "YYYY-MM-DDTHH:MM:SS", "YYYY-MM-DDTHH:MM", "YYYY-MM-DD HH:MM:SS", "YYYY-MM-DD HH:MM", or "YYYY-MM-DD". POSIXct / Date inputs are returned coerced to POSIXct.

Usage

```
parse_dtc(x, tz = "UTC")
```

Arguments

x	character vector (or POSIXct/Date) of datetime values.
tz	time zone. Default "UTC".

Value

POSIXct vector.

PrepPDF *Prepare PDF Output*

Description

Opens a PDF device for NONMEM report generation.

Usage

```
PrepPDF(FileName, Paper = "letter", FontFamily = "Courier")
```

Arguments

FileName	character, output PDF filename
Paper	character, paper size (default "letter")
FontFamily	character, font family (default "Courier")

PrinMTxt *Print Multiple Lines of Text*

Description

Prints a character vector of text lines across multiple pages as needed.

Usage

```
PrinMTxt(
  MTxt,
  Cex = 0.8,
  Header1 = "",
  Header2 = "",
  Header3 = "",
  Footer1 = "",
  Footer2 = "",
  Footer3 = "",
  PrintRowNum = FALSE
)
```

Arguments

MTxt	character vector of text lines
Cex	numeric, character expansion factor
Header1	character, left header
Header2	character, center header
Header3	character, right header
Footer1	character, left footer
Footer2	character, center footer
Footer3	character, right footer
PrintRowNum	logical, whether to print row numbers

PrinTxt *Print Text at Position*

Description

Prints text at a specific row and column position on the PDF page.

Usage

```
PrinTxt(Row, Col, Text, Cex = 0.8)
```

Arguments

Row	numeric, row position
Col	numeric, column position
Text	character, text to print
Cex	numeric, character expansion factor

RemoveNA *Remove Rows with NA Values*

Description

Remove Rows with NA Values

Usage

```
RemoveNA(RawData)
```

Arguments

RawData	data.frame

Value

data.frame with rows containing NA removed

ResTest *Residual Count Test Using Binomial Distribution*

Description

Tests whether residual counts at various Z-value thresholds follow expected binomial distributions.

Usage

```
ResTest(
  ResTab,
  TestCols = c("WRES", "CWRE", "IWRE"),
  ZVals = c(0, 1, 2, 3),
  PctVals = c(0.005, 0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5)
)
```

Arguments

ResTab	data.frame with residual columns
TestCols	character vector of column names to test
ZVals	numeric vector of Z-value thresholds
PctVals	numeric vector of percentile values

Value

matrix of test results

---

run.test.nm *Runs Test for NONMEM Residuals*

Description

Performs a runs test on residuals. Zeros are omitted.

Usage

```
run.test.nm(RES)
```

Arguments

RES	numeric vector of residuals

Value

numeric, two-sided p-value

SESuccess *Check Standard Error Success*

Description

Check Standard Error Success

Usage

```
SESuccess()
```

Value

logical, TRUE if standard errors were computed successfully

SumOut *Summarize Multiple NONMEM Runs*

Description

Scans subdirectories matching RunExt and collects summary metrics for model development flow analysis.

Usage

```
SumOut(FileExt = ".CTL", RunExt = ".R76", OutExt = ".OUT")
```

Arguments

FileExt	character, control file extension
RunExt	character, run folder extension
OutExt	character, output file extension

Value

data.frame with summary of all runs

TabStep *Table Step*

Description

This produces standard table.

Usage

```
TabStep()
```

Details

It does not have arguments. All necessary arguments are stored in the e environment. This is similar to other standard results table.

Value

A table with ID, TIME, DV, PRED, RES, WRES, derivatives of G and H. If the estimation method is other than 'ZERO'(First-order approximation), it includes CWRES, CIPREDI(formerly IPRED), CIRESI(formerly IRES).

Author(s)

Kyun-Seop Bae <k@acr.kr>

References

NONMEM Users Guide

See Also

EstStep

Examples

```
# Only After EstStep
#TabStep()
```

`TrimOut` *Trimming and beautifying NONMEM original OUTPUT file*

Description

TrimOut removes unnecessary parts from NONMEM original OUTPUT file.

Usage

```
TrimOut(inFile, outFile="PRINT.OUT")
```

Arguments

`inFile`	NONMEM original untidy OUTPUT file name
`outFile`	Output file name to be written

Details

NONMEM original OUTPUT file contains unnecessary parts such as CONTROL file content, Start/End Time, License Info, Print control characters such as "+", "0", "1". This function trims those.

Value

outFile will be written in the current working folder or designated folder. This returns TRUE if the process was smooth.

Author(s)

Kyun-Seop Bae <k@acr.kr>

Index