Package ‘nmw’

October 23, 2017

Version 0.1.3

Title Understanding Nonlinear Mixed Effects Modeling for Population Pharmacokinetics

Description This shows how NONMEM(R) <http://www.iconplc.com/innovation/nonmem/> software works. NONMEM's classical estimation methods like 'First Order(FO) approximation', 'First Order Conditional Estimation(FOCE)', and 'Laplacian approximation' are explained.

Depends R (>= 3.0.0), numDeriv

ByteCompile yes

License GPL-3

Copyright 2017, Kyun-Seop Bae

Author Kyun-Seop Bae

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

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

NeedsCompilation no

Repository

Date 2017-10-23 00:01:03 KST

R topics documented:

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

Description

This shows how NONMEM(R) <http://www.iconplc.com/innovation/nonmem/> software works.

Details

This package explains 'First Order(FO) approximation' method, 'First Order Conditional Estimation(FOCE)' method, and 'Laplacian(LAPL)' method of NONMEM software.

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: firstorder 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=(e$METHOD == "LAPL"))
```

```
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")
(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, Pred=PRED, METHOD="COND")
#(EstRes = EstStep())           # 4.64 min
#(CovRes = CovStep())           # 1.14 min
#get("EBE", envir=e)
#TabStep()

######## Laplacian Approximation with Interacton Method
#InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, Pred=PRED, METHOD="LAPL")
#(EstRes = EstStep())           # 4.53 min
#(CovRes = CovStep())           # 1.09 min
#get("EBE", envir=e)
#TabStep()
```

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 caculcates variance-covariance matrix of estimates on 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, second derivative of log likelihood function with respect to esimation parameters

`S Matrix` S matrix of NONMEM, sum of individual cross-product of first derivative of log likelihood function with respect to esimation parameters

Author(s)

Kyun-Seop Bae <k@acr.kr>

References

NONMEM Users Guide

See Also

`EstStep`, `InitStep`

Examples

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

`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 with NONMEM output.

Value

`Initial OFV`	initial value of 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()
```

`InitStep` *Initialization Step*

Description

It recevies 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=(e$METHOD == "LAPL"))
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 Interacton Method
InitStep(DataAll, THETAinit=THETAinit, OMinit=OMinit, SGinit=SGinit, LB=LB, UB=UB,
         Pred=PRED, METHOD="LAPL")
```

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, derivaties 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()
```

Index