Package: PLmixed (via r-universe)

September 17, 2024

Title Estimate (Generalized) Linear Mixed Models with Factor Structures

Version 0.1.7

Description Utilizes the 'lme4' and 'optimx' packages (previously the optim() function from 'stats') to estimate (generalized) linear mixed models (GLMM) with factor structures using a profile likelihood approach, as outlined in Jeon and Rabe-Hesketh (2012) <doi:10.3102/1076998611417628> and Rockwood and Jeon (2019) <doi:10.1080/00273171.2018.1516541>. Factor analysis and item response models can be extended to allow for an arbitrary number of nested and crossed random effects, making it useful for multilevel and cross-classified models.

Depends R (>= 3.2.2)

Imports lme4, Matrix (>= 1.1.1), numDeriv, stats, optimx

Suggests knitr, rmarkdown, irtoys

VignetteBuilder knitr

Encoding UTF-8

License GPL (>= 2)

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Minjeong Jeon [aut], Nicholas Rockwood [aut, cre]

Maintainer Nicholas Rockwood <`njrockwood@gmail.com`>

Date/Publication 2023-08-23 22:20:02 UTC

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

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

RemoteRef HEAD

RemoteSha 7378a400f2fa59e20a1c90acff76dc5a3ee39efc

Contents

`PLmixed-package` *PLmixed: A package for estimating GLMMs with factor structures.*

Description

The `PLmixed` package's main function is `PLmixed`, which estimates the model through nested maximizations using the **lme4** and **optimx** packages (previously the `optim` function). This extends the capabilities of **lme4** to allow for estimated factor structures, making it useful for estimating multilevel factor analysis and item response theory models with an arbitrary number of hierarchical levels or crossed random effects.

References

Rockwood, N. J., & Jeon, M. (2019). Estimating complex measurement and growth models using the R package PLmixed.*Multivariate Behavioral Research, 54*(2), 288-306.

Jeon, M., & Rabe-Hesketh, S. (2012). Profile-likelihood approach for estimating generalized linear mixed models with factor structures. *Journal of Educational and Behavioral Statistics, 37*(4), 518-542.

`coef.PLmod` *coef.PLmod*

Description

Obtain coefficients for a model of class PLmod.

Usage

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

Arguments

`object`	an object of class PLmod
`...`	Additional arguments from `coef.merMod`.

Value

sum of the random and fixed effects coefficients for each explanatory variable for each level of the grouping factor.

`fitted.PLmod` *fitted.PLmod*

Description

Obtain fitted values for a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
fitted(object, ...)
```

Arguments

`object`	an object of class PLmod
`...`	Additional arguments from `fitted.merMod`.

`fixef.PLmod` *fixef.PLmod*

Description

Obtain fixed effect estimates for a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
fixef(object, ...)
```

Arguments

`object`	an object of class PLmod
`...`	Additional arguments from `fixef.merMod`.

`IRTsim` *Simulated multilevel IRT dataset.*

Description

A simulated dataset that replicates the dataset from CITO.

Usage

```
IRTsim
```

Format

A data frame with 2500 rows and 4 variables:

sid Student ID

school School ID

item Item ID

y Response

`iterPlot` *iterPlot*

Description

Plot parameter estimates at each `optim` iteration.

Usage

```
iterPlot(object)
```

Arguments

`object`	an object of class PLmod

`JUDGEsim` *Simulated Multi-rater Multi-response dataset.*

Description

A simulated dataset that replicates the dataset from a multi-rater mult-reponse study where teachers and students provided responses about two student traits.

Usage

```
JUDGEsim
```

Format

A data frame with 54462 rows and 7 variables:

item Item ID

method 1 = teacher response, 2 = student response

trait 1 = trait 1, 2 = trait 2

stu Student ID

class Classroom ID

tch Teacher ID

response Item response

`KYPSitemsim` *Simulated KYPS item-level dataset.*

Description

A simulated dataset that replicates the dataset item-level data from KYPS.

Usage

```
KYPSitemsim
```

Format

A data frame with 66947 rows and 6 variables:

id Student ID

time Time Identifier

item Item ID

mid Middle School ID

hid High School ID

response Item Response

`KYPSsim` *Simulated KYPS dataset.*

Description

A simulated dataset that replicates the dataset from KYPS.

Usage

```
KYPSsim
```

Format

A data frame with 11494 rows and 5 variables:

mid Middle School ID

hid High School ID

sid Student ID

time Time Identifier

esteem Self Esteem

`PLmixed` *Fit GLMM with Factor Structure*

Description

Fit a (generalized) linear mixed effects model (GLMM) with factor structures. Utilizes both the **lme4** package and `optim` function for estimation using a profile-likelihood based approach.

Usage

```
PLmixed(
  formula,
  data,
  family = gaussian,
  load.var = NULL,
  lambda = NULL,
  factor = NULL,
  init = 1,
  nlp = NULL,
  init.nlp = 1,
  nAGQ = 1,
  method = "L-BFGS-B",
  lower = -Inf,
  upper = Inf,
  lme4.optimizer = "bobyqa",
  lme4.start = NULL,
  lme4.optCtrl = list(),
  opt.control = NULL,
  REML = FALSE,
  SE = 1,
  ND.method = "simple",
  check = "stop",
  est = TRUE,
  iter.count = TRUE
)
```

Arguments

`formula`	A formula following that of **lme4**, with the addition that factors can be specified as random effects. Factor names should not be names of variables in the data set, and are instead defined with the `factor` argument.
`data`	A data frame containing the variables used in the model (but not factor names).
`family`	A GLM family, see `glm` and `family`.
`load.var`	A variable in the dataframe identifying what the factors load onto. Each unique element in `load.var` will have a unique factor loading. All rows in the dataset with the same value for `load.var` will have the same factor loading.

`lambda`	A matrix or list of matrices corresponding to the loading matrices. A value of NA indicates the loading is freely estimated, while a numeric entry indicates a constraint.
`factor`	A list of factors corresponding to the loading matrices and factors specified in model.
`init`	A scalar (default = `1`) or vector of initial lambda values. If a scalar, the value is applied to all lambda parameters. If a vector, the values apply in row by column by matrix order.
`nlp`	A character vector containing the names of additional nonlinear parameters that are in the model formula.
`init.nlp`	A scalar (default = `1`) or vector of initial nlp values. If a scalar, the value is applied to all nlp parameters. If a vector, the values apply in the order listed.
`nAGQ`	If family is non-gaussian, the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. Defaults to `1`, corresponding to the Laplace approximation. See `glmer`.
`method`	The `optimx` optimization method. Defaults to `L-BFGS-B`.
`lower`	Lower bound on lambda parameters if applicable.
`upper`	Upper bound on lambda parameters if applicable.
`lme4.optimizer`	The **lme4** optimization method.
`lme4.start`	Start values used for **lme4**.
`lme4.optCtrl`	A list controlling the lme4 optimization. See `lmerControl` or `glmerControl`
`opt.control`	Controls for the `optimx` optimization.
`REML`	Use REML if model is linear? Defaults to `FALSE`.
`SE`	Method of calculating standard errors for fixed effects.
`ND.method`	Method of calculating numerical derivatives.
`check`	Check number of observations vs. levels and number of observations vd. random effects.
`est`	Return parameter estimates.
`iter.count`	Print the iteration counter during optimization.

Details

Factors are listed within the `formula` in the same way that random effects are specified in **lme4**. The grouping variable listed after `|` defines what the factor values randomly vary over, just as `|` does for other random effects. The names of factors and other random effect terms can be listed within the same set of parentheses, allowing the covariance between the factor(s) and random effect(s) to be estimated. The same factor may be specified for multiple grouping variables, allowing for multilevel or crossed effects.

The `factor` argument must list any factor that appears in the `formula`. The ordering will depend on the ordering of the matrices listed within `lambda`. The matrices in `lambda` specify the factor loading matrices. The number of matrices in `lambda` should equal the number of character vectors in `factor` and the number of elements in `load.var`. The number of rows in the kth matrix listed in `lambda` should correspond to the number of unique elements in the dataset for the kth variable listed in `load.var`, and the number of columns in the *k*th matrix should correspond to the number of factors listed in the *k*th character vector of `factor`.

Within the *k*th matrix, the *(i, j)* cell corresponds to the factor loading for the *i*th unique element of the *k*th variable listed in `load.var` on the *j*th factor listed in the *k*th character vector of `factor`. Each element of the matrix should be either a number or `NA`. If the element is a number, the loading will be constrained to that value. If the element is an `NA`, the loading will be freely estimated. For identification, it is necessary (but not sufficient) for at least one element in each column to be constrained.

The `nlp` argument can be viewed as a special case of the `factor` argument, where the character vector listed in `nlp` is automatically linked to 1 x p lambda matrix, where p is the number of elements in `nlp`. The `load.var` for these parameters is viewed as a constant, so that the `nlp` parameters are equivalent for all rows in the dataset. Thus, `nlp` simplifies the process of adding additional nonlinear parameters to the model without having to specify corresponding `lambda` and `load.var` values.

Value

An object of class `PLmod`, which contains an object of class `merMod` as one of its elements. Some functions for class `merMod` have been adapted to work with class `PLmod`. Others can be utilized using `object$'lme4 Model'`, where `object` is an object of class `PLmod`.

References

Rockwood, N. J., & Jeon, M. (2019). Estimating complex measurement and growth models using the R package PLmixed.*Multivariate Behavioral Research, 54*(2), 288-306.

Jeon, M., & Rabe-Hesketh, S. (2012). Profile-likelihood approach for estimating generalized linear mixed models with factor structures. *Journal of Educational and Behavioral Statistics, 37*(4), 518-542.

See Also

lme4

`glmer`

`lmer`

Examples

```
data("IRTsim") # Load the IRTsim data

IRTsub <- IRTsim[IRTsim$item < 4, ] # Select items 1-3
set.seed(12345)
IRTsub <- IRTsub[sample(nrow(IRTsub), 300), ] # Randomly sample 300 responses

IRTsub <- IRTsub[order(IRTsub$item), ] # Order by item
irt.lam = c(1, NA, NA) # Specify the lambda matrix

# Below, the # in front of family = binomial can be removed to change the response distribution
# to binomial, where the default link function is logit.
```

```
irt.model <- PLmixed(y ~ 0 + as.factor(item) + (0 + abil.sid |sid) +(0 + abil.sid |school),
                     data = IRTsub, load.var = c("item"), # family = binomial,
                     factor = list(c("abil.sid")), lambda = list(irt.lam))
summary(irt.model)

## Not run:
# A more time-consuming example.
# ~ 5-10 minutes

data("KYPSsim") # Load the KYPSsim data

kyps.lam <- rbind(c( 1,  0),  # Specify the lambda matrix
                  c(NA,  0),
                  c(NA,  1),
                  c(NA, NA))

kyps.model <- PLmixed(esteem ~ as.factor(time) +  (0 + hs | hid)
                      + (0 + ms | mid) + (1 | sid), data = KYPSsim,
                      factor = list(c("ms", "hs")), load.var = c("time"),
                      lambda = list(kyps.lam))
summary(kyps.model)

data("JUDGEsim")
JUDGEsim <- JUDGEsim[order(JUDGEsim$item), ] # Order by item
unique(JUDGEsim$item)

# Specify Lambda matrix
judge.lam <- rbind(c( 1,  0,  1,  0,  0,  0),
                   c(NA,  0, NA,  0,  0,  0),
                   c(NA,  0, NA,  0,  0,  0),
                   c( 0,  1,  0,  1,  0,  0),
                   c( 0, NA,  0, NA,  0,  0),
                   c( 0, NA,  0, NA,  0,  0),
                   c( 0,  0,  0,  0,  1,  0),
                   c( 0,  0,  0,  0, NA,  0),
                   c( 0,  0,  0,  0, NA,  0),
                   c( 0,  0,  0,  0,  0,  1),
                   c( 0,  0,  0,  0,  0, NA),
                   c( 0,  0,  0,  0,  0, NA))

# Conduct analysis
judge.example <- PLmixed(response ~ 0 + as.factor(item) + (1 | class)
                         + (0 + trait1.t + trait2.t + trait1.s + trait2.s | stu)
                         + (0 + teacher1 + teacher2 | tch), data = JUDGEsim,
                         lambda = list(judge.lam), load.var = "item",
                         factor = list(c("teacher1", "teacher2", "trait1.t",
                                         "trait2.t", "trait1.s", "trait2.s")))

summary(judge.example)

data("KYPSitemsim")

time.lam <- rbind(c( 1,  0),  # Specify time lambda matrix
                      c(NA,  0),
                      c(NA,  1),
                      c(NA, NA))

item.lam <- c(1, NA, NA, NA, NA, NA) # Specify item lambda matrix

KYPSitemsim$time2 <- (KYPSitemsim$time == 2) * 1
KYPSitemsim$time3 <- (KYPSitemsim$time == 3) * 1
KYPSitemsim$time4 <- (KYPSitemsim$time == 4) * 1

kyps.item.model <- PLmixed(response ~ 0 + as.factor(item) + lat.var:time2
                           + lat.var:time3 + lat.var:time4 + (0 + hs:lat.var | hid)
                          + (0 + ms:lat.var | mid) + (0 + lat.var:as.factor(time) | id),
                           data = KYPSitemsim, lambda = list(time.lam, item.lam),
                           factor = list(c("ms", "hs"), "lat.var"),
                           load.var = c("time", "item"))

summary(kyps.item.model)


## End(Not run)
```

`plot.PLmod` *plot.PLmod*

Description

Diagnostic plots for a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
plot(x, ...)
```

Arguments

`x`	an object of class PLmod
`...`	Additional arguments from `plot.merMod`.

predict.PLmod *predict.PLmod*

Description

Predict response values from a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	an object of class PLmod
`newdata`	data frame to obtain predictions for
`...`	Additional arguments from `predict.merMod`.

print.PLmod *print.PLmod*

Description

Print the fitted PLmixed model object of class PLmod.

Usage

```
## S3 method for class 'PLmod'
print(x, digits = 4, ...)
```

Arguments

`x`	an object of class PLmod
`digits`	minimal number of significant digits, see `print.default`.
`...`	Additional arguments.

`print.summary.PLmod` *print.summary.PLmod*

Description

Print the output for a PLmixed model object of class PLmod.

Usage

```
## S3 method for class 'summary.PLmod'
print(x, digits = 4, ...)
```

Arguments

`x`	an object of class PLmod
`digits`	minimal number of significant digits, see `print.default`.
`...`	Additional arguments.

`ranef.PLmod` *ranef.PLmod*

Description

Obtain conditional modes of the random effects for a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
ranef(object, ...)
```

Arguments

`object`	an object of class PLmod
`...`	Additional arguments from `ranef.merMod`.

residuals.PLmod *residuals.PLmod*

Description

Obtain residuals for a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
residuals(object, ...)
```

Arguments

`object`	an object of class PLmod
`...`	Additional arguments from `residuals.merMod`.

simulate.PLmod *simulate.PLmod*

Description

Simulate responses from a model of class PLmod.

Usage

```
## S3 method for class 'PLmod'
simulate(object, ...)
```

Arguments

`object`	an object of class PLmod
`...`	Additional arguments from `simulate.merMod`.

`summary.PLmod` *summary.PLmod*

Description

Obtain key output for a fitted PLmixed model object of class PLmod.

Usage

```
## S3 method for class 'PLmod'
summary(object, ...)
```

Arguments

`object`	an object of class PLmod
`...`	Additional arguments.

Value

An object containing all parameter estimates and model characteristics.

Index