

# Package ‘Dire’

January 20, 2025

**Version** 2.2.0

**Date** 2023-10-24

**Title** Linear Regressions with a Latent Outcome Variable

**Maintainer** Paul Bailey <pbailey@air.org>

**Depends** R (>= 4.0.0)

**Imports** foreach, iterators, methods, Matrix, haven, Rcpp (>= 1.0.8.3),  
lbfgs, MASS

**Description** Fit latent variable linear models, estimating score distributions for groups of people, following Cohen and Jiang (1999) <doi:10.2307/2669917>. In this model, a latent distribution is conditional on students item response, item characteristics, and conditioning variables the user includes. This latent trait is then integrated out. This software is intended to fit the same models as the existing software 'AM' <<https://am.air.org/>>. As of version 2, also allows the user to draw plausible values.

**License** GPL-2

**VignetteBuilder** knitr

**Suggests** knitr, rmarkdown, testthat, withr, doParallel, parallel,  
EdSurvey

**URL** <https://american-institutes-for-research.github.io/Dire/>

**BugReports** <https://github.com/American-Institutes-for-Research/Dire/issues>

**ByteCompile** true

**Note** This publication was prepared for NCES under Contract No. ED-IES-12-D-0002/0004 with the American Institutes for Research. Mention of trade names, commercial products, or organizations does not imply endorsement by the U.S. Government.

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**LinkingTo** Rcpp, RcppArmadillo

**NeedsCompilation** yes

**Author** Emmanuel Sikali [pdr],  
 Paul Bailey [aut, cre],  
 Eric Buehler [aut],  
 Sun-joo Lee [aut],  
 Harold Doran [aut],  
 Blue Webb [ctb],  
 Claire Kelley [ctb]

**Repository** CRAN

**Date/Publication** 2023-10-26 22:30:02 UTC

## Contents

drawPVs . . . . .	2
mml . . . . .	4

**Index** **12**

---

drawPVs	<i>Draw plausible values (PVs) from an mml fit</i>
---------	--

---

## Description

Draw plausible values (PVs) from an mml fit

## Usage

```
drawPVs(x, npv, pvVariableNameSuffix = "_dire", ...)

## S3 method for class 'summary.mmlMeans'
drawPVs(x, npv = 5L, pvVariableNameSuffix = "_dire", ...)

## S3 method for class 'mmlMeans'
drawPVs(
  x,
  npv = 5L,
  pvVariableNameSuffix = "_dire",
  stochasticBeta = FALSE,
  normalApprox = TRUE,
  newStuDat = NULL,
  newStuItems = NULL,
  returnPosterior = FALSE,
  construct = NULL,
  ...
)

## S3 method for class 'mmlCompositeMeans'
drawPVs(
```

```

x,
npv = 5L,
pvVariableNameSuffix = "_dire",
stochasticBeta = FALSE,
normalApprox = TRUE,
newStuDat = NULL,
newStuItems = NULL,
verbose = TRUE,
...
)

```

### Arguments

x	a fit from a call to <a href="#">mml</a>
npv	integer indicating the number of plausible values to draw
pvVariableNameSuffix	suffix added to new PV variables after construct name and before the plausible value ID. For example, if there is a construct <code>math</code> and the suffix is the default <code>_dire</code> , then the fourth plausible value would have a column name, <code>math_dire4</code> .
...	additional parameters
stochasticBeta	logical when TRUE the regression coefficients will be drawn from their posterior distribution. Can also be a data frame of values (see Details).
normalApprox	logical must be TRUE to use the normal approximation to the posterior distribution rather than drawing from the actual posterior distribution.
newStuDat	new <code>stuDat</code> object, (see <a href="#">mml</a> ) for which plausible values will be drawn
newStuItems	new <code>stuItems</code> object, (see <a href="#">mml</a> ); unlike in <code>mml</code> students with no items can be passed to this function
returnPosterior	logical set to TRUE to change output to include two additional data frames (see Value).
construct	character, changes the name of the columns in the final data frame
verbose	logical set to TRUE to see the status of the processing

### Details

When the argument passed to `stochasticBeta` is a data frame then each column is an element that will be used as a regression coefficient for that index of the coefficients vector. The row index used for the *n*th PV will be the *n*th row.

### Value

when `returnPosterior` is FALSE returns a object of class `DirePV` which is a list of two elements. first, a data frame with a row for every row of `newStuDat` (or the original `stuDat` object)

- `id` the value of `idVar` in the model run

- `[construct][pvVariableNameSuffix][L]` every other column is a plausible value of this format. The `[construct]` is the name of the construct, the `[pvVariableNameSuffix]` is the value of the `pvVariableNameSuffix` argument, and the `[L]` part is the plausible value index, from 1 to `npv`.

The second argument is named `newpvvars` and is a list with an element for each set of construct that lists all of the variables in that construct.

When `returnPosterior` is `TRUE` returns `list` with three elements. One is named `posterior` and has one row per `idvar` level in the `newStuDat` argument and three columns:

- `id` the value of `idVar` in the model run
- `mu` the posterior mean
- `sd` the posterior standard deviation

the second list element is named `X` that is the design matrix for `newStuDat` (see Value for `mml`). The third list element is the `rr1` element returned from `mml` with one column for each individual in `newStuDat` (see Value in `mml`).

### Author(s)

Paul Bailey, Sun-joo Lee, and Eric Buehler

### Examples

```
# See Examples in mml
```

---

mml

*Marginal Maximum Likelihood Estimation of Linear Models*

---

### Description

Implements a survey-weighted marginal maximum estimation, a type of regression where the outcome is a latent trait (such as student ability). Instead of using an estimate, the likelihood function marginalizes student ability. Includes a variety of variance estimation strategies.

### Usage

```
mml(
  formula,
  stuItems,
  stuDat,
  idVar,
  dichotParamTab = NULL,
  polyParamTab = NULL,
  testScale = NULL,
  Q = 30,
  minNode = -4,
  maxNode = 4,
```

```

polyModel = c("GPCM", "GRM"),
weightVar = NULL,
multiCore = FALSE,
bobyqaControl = NULL,
composite = TRUE,
strataVar = NULL,
PSUVar = NULL,
fast = TRUE,
calcCor = TRUE,
verbose = 0
)

```

### Arguments

formula	a formula object in the style of <code>lm</code>
stuItems	a <code>data.frame</code> where each row represents a single student's response to one item. The columns must include the <code>idVar</code> column, a key column, and a score column. Values in the score column are checked against expectations (based on <code>dichotParamTab</code> and <code>polyParamTab</code> ) and when <code>verbose</code> is $\geq 1$ a table of expected and actual levels is printed.
stuDat	a <code>data.frame</code> with a single row per student. Predictors in the formula must be in <code>stuDat</code> .
idVar	a variable name on <code>stuDat</code> that is the identifier. Every ID from <code>stuDat</code> must appear on <code>stuItems</code> and vice versa.
dichotParamTab	a <code>data.frame</code> of dichotomous item information, see Details
polyParamTab	a <code>data.frame</code> of polytomous item information, see Details
testScale	a <code>data.frame</code> of scaling information, see Details
Q	an integer; the number of integration points
minNode	a numeric; the smallest integration point for the latent variable
maxNode	a numeric; the largest integration point for the latent variable
polyModel	polytomous response model; one of <code>GPCM</code> for the Graded Partial Credit Model or <code>GRM</code> for the Graded Response Model
weightVar	a variable name on <code>stuDat</code> that is the full sample weight
multiCore	allows the <code>foreach</code> package to be used. You should have already setup the <code>registerDoParallel</code> function in the <code>doParallel</code> package.
bobyqaControl	deprecated. A list that gets passed to the <code>bobyqa</code> optimizer in <code>minqa</code>
composite	a logical indicating if an overall test should be treated as a composite score; a composite is a weighted average of the subscales in it.
strataVar	character naming a variable on <code>stuDat</code> , the variable indicating the stratum for each row. Used in post-hoc robust variance estimation.
PSUVar	character naming a variable on <code>stuDat</code> ; the primary sampling unit (PSU) variable. Used in post-hoc robust variance estimation. The values do not need to be unique across strata.

<code>fast</code>	a logical indicating if cpp code should be used in mml processes. This should yield speed-ups to runs.
<code>calcCor</code>	set to TRUE to calculate covariances. Needed to estimate variances and form plausible values
<code>verbose</code>	integer, negative or zero for no details, increasingly verbose messages at one and two

## Details

The `mml` function models a latent outcome conditioning on item response data, covariate data, item parameter information, and scaling information. These four parts are broken up into at least one argument each. Student item response data go into `stuItems`; whereas student covariates, weights, and sampling information go into `stuDat`. The `dichotParamTab` and `polyParamTab` contain item parameter information for dichotomous and polytomous items, respectively—the item parameter data is the result of an existing item parameter scaling. In the case of the National Assessment of Educational Progress (NAEP), they can be found online, for example, at [NAEP technical documentation](#). Finally, information about scaling and subscale weights for composites are put in `testScale`.

The model for dichotomous responses data is, by default, three Parameter Logit (3PL), unless the item parameter information provided by users suggests otherwise. For example, if the scaling used a two Parameter Logit (2PL) model, then the guessing parameter can simply be set to zero. For polytomous responses data, the model is dictated by the `polyModel` argument.

The `dichotParamTab` argument is a `data.frame` with a column named `ItemID` that identifies the items and agrees with the key column in the `stuItems` argument, and, for a 3PL item, columns `slope`, `difficulty`, and `guessing` for the “a”, “d”, and “g” parameters, respectively; see the vignette for details of the 3PL model. Users can also use the column names directly from the vignette notation (“a”, “d”, and “g”) if they prefer. Items that are missing (NA) are not used in the likelihood function. Users wishing to apply a special behavior for a subset of items can use `set` those items to an invalid score and put that in the `dichotParamTab` column `missingCode`. They are then scored as if they are `missingValue` proportion correct. To use the guessing parameter for the proportion correct set `missingValue` to “c”.

The `polyParamTab` has columns `ItemID` that must match with the key from `stuItems`, as well as `slope` (which can also be called `a`) that corresponds to the `a` parameter in the vignette. Users must also specify the location of the cut points ( $d_{c,j}$  in the vignette) which are named `d1`, `d2`, ..., up to `dn` where `n` is one less than the number of score points. Some people prefer to also apply a shift to all of these and this shift is applied when there is a column named `itemLocation` by simply adding that to every `d*` column. Items are not included in the likelihood for an individual when their value on `stuItems` is NA, but no provision is made for guessing, nor special provision for missing codes in polytomous items.

For both `dichotParamTab` and `polyParamTab` users wishing to use a D parameter of 1.7 (or any other value) may specify that, per item, in a column named `D`.

When there are multiple constructs, subscales, or the user wants a composite score, additional, optional, columns `test` and `subtest` can be used. these columns can be numeric or text, they must agree with the same columns in `testScale` to scale the results.

Student data are broken up into two parts. The item response data goes into `stuItems`, and the student covariates for the formula go into `stuDat`. Information about items, such as item difficulties,

is in `paramTab`. All dichotomous items are assumed to be 3PL, though by setting the guessing parameter to zero, the user can use a 2PL or the one Parameter Logit (1PL) or Rasch models. The model for polytomous responses data is dictated by the `polyModel` argument.

The marginal maximum likelihood then integrates the product of the student ability from the assessment data, and the estimate from the linear model estimates each student's ability based on the formula provided and a residual standard error term. This integration happens from the minimum node to the maximum node in the `control` argument (described later in this section) with `Q` quadrature points.

The `stuItems` argument has the scored student data. It is a list where each element is named with student ID and contains a `data.frame` with at least two columns. The first required column is named `key` and shows the item name as it appears in `paramTab`; the second column is named `score` and shows the score for that item. For dichotomous items, the score is 0 or 1. For GPCM items, the scores start at zero as well. For GRM, the scores start at 1.

For a GPCM model,  $P_0$  is the "a" parameter, and the other columns are the "d" parameters; see the vignette for details of the GPCM model.

The quadrature points then are a range from `minNode` to `maxNode` with a total of `Q` nodes.

## Value

When called for a single subscale or overall score, returns object of class `mmlMeans`. This is a list with elements:

- `call` the call used to generate this `mml.means` object
- `coefficients` the unscaled marginal maximum likelihood regression coefficients
- `LogLik` the log-likelihood of the fit model
- `X` the design matrix of the marginal maximum likelihood regression
- `Convergence` a convergence note from the optimizer
- `location` used for scaling the estimates
- `scale` used for scaling the estimates
- `lnlf` the log-likelihood function of the unscaled parameters
- `rr1` the density function of each individual, conditional only on item responses in `stuItems`
- `stuDat` the `stuDat` argument
- `weightVar` the name of the weight variable on `stuDat`
- `nodes` the nodes the likelihood was evaluated on
- `iterations` the number of iterations required to reach convergence
- `obs` the number of observations used
- `weightedObs` the weighted N for the observations
- `strataVar` the column name of the stratum variable on `stuDat`; potentially used for variance estimation
- `PSUVar` the column name of the PSU variable on `stuDat`; potentially used for variance estimation
- `itemScorePoints` a data frame that shows item IDs, the number of score points, expected scores (both from the `paramTab` arguments), as well as the occupied score points

- `stuItems` the data frame passed to `mml` reformatted for use in `mml`
- `formula` the formula passed to `mml`
- `contrasts` the contrasts used in forming the design matrix
- `xlevels` the levels of the covariates used in forming the design matrix
- `polyModel` the value of the argument of the same name passed to `mml`
- `paramTab` a data frame that condenses `dichotParamTab` and `polyParamTab`
- `fast` the value of the argument of the same name passed to `mml`
- `idVar` the value of the argument of the same name passed to `mml`
- `posteriorEsts` the posterior estimates for the people in `stuDat` included in the model

When a composite score is computed there are several subscales run and the return is a `mmlCompositeMeans`. Many elements are themselves list with one element per construct. this is a list with elements:

- `call` the call used to generate this `mml.means` object
- `coefficients` matrix of the unscaled marginal maximum likelihood regression coefficients, each row represents a subscale, each column represents a coefficient
- `X` the design matrix of the marginal maximum likelihood regression
- `rr1` a list of elements, each the `rr1` object for a subscale (see `mmlMeans` output)
- `ids` The ID variable used for each row of `stuDat`
- `Convergence` a vector of convergence notes from the optimizer
- `lnlfl` a list of log-likelihood functions of the unscaled parameters, by construct
- `stuDat` a list of `stuDat` data frames, as used when fitting each construct, filtered to just relevant student records
- `weightVar` the name of the weight variable on `stuDat`
- `nodes` the nodes the likelihood was evaluated on
- `iterations` a vector of the number of iterations required to reach convergence on each construct
- `obs` a vector of the the number of observations used on each construct
- `testScale` the `testScale` used to scale the data
- `weightedObs` a vector of the weighted N for the observations
- `SubscaleVC` the covariance matrix of subscales. The residuals are assumed to be multivariate normal with this covairiance matrix
- `idVar` the name of the identifier used on `stuDat` and `stuItems` data
- `res1` list of `mmlMeans` objects, one per construct
- `strataVar` the column name of the stratum variable on `stuDat`; potentially used for variance estimation
- `PSUVar` the column name of the PSU variable on `stuDat`; potentially used for variance estimation
- `stuItems` the data frame passed to `mml` reformatted for use in `mml`
- `formula` the formula passed to `mml`



- contrasts the contrasts used in forming the design matrix
- xlevels the levels of the covariates used in forming the design matrix
- polyModel the value of the argument of the same name passed to mml
- posteriorEsts the list of posterior estimates for the people in stuDat included in the model
- SubscaleVC the matrix of latent correlations across constructs

LogLik is not returned because there is no likelihood for a composite model.

### Author(s)

Harold Doran, Paul Bailey, Claire Kelley, Sun-joo Lee, and Eric Buehler

### Examples

```
## Not run:
require(EdSurvey)

# 1) make param tab for dichotomous items
dichotParamTab <- data.frame(ItemID = c("m109801", "m020001", "m111001",
                                         "m046301", "m046501", "m051501",
                                         "m111601", "m111301", "m111201",
                                         "m110801", "m110101"),
                             test = rep("composite",11),
                             subtest = c(rep("num",6),rep("alg",5)),
                             slope = c(0.96, 0.69, 0.83,
                                         0.99, 1.03, 0.97,
                                         1.45, 0.59, 0.34,
                                         0.18, 1.20),
                             difficulty = c(-0.37, -0.55, 0.85,
                                             -0.97, -0.14, 1.21,
                                             0.53, -1.84, -0.46,
                                             2.43, 0.70),
                             guessing = c(0.16, 0.00, 0.17,
                                           0.31, 0.37, 0.18,
                                           0.28, 0.15, 0.09,
                                           0.05, 0.18),
                             D = rep(1.7, 11),
                             MODEL = rep("3p1", 11))

# param tab for GPCM items
polyParamTab <- data.frame(ItemID = factor(c("m0757c1", "m066501")),
                            test = rep("composite",2),
                            subtest = rep("alg",2),
                            slope = c(0.43, 0.52), # could also be called "a"
                            itemLocation = c(-1.21, -0.96), # added to d1 ... dn
                            d1 = c(2.38, -0.56),
                            d2 = c(-0.57, 0.56),
                            d3 = c(-1.18, NA),
                            D = c(1.7, 1.7),
                            scorePoints = c(4L, 3L)) # number of score points, read d1 to d(n-1)

# read-in NAEP Primer data
```

```

sdf <- readNAEP(system.file("extdata/data", "M36NT2PM.dat", package = "NAEPprimer"))
# read in these items
items <- c(as.character(dichotParamTab$ItemID), as.character(polyParamTab$ItemID))
# dsex, student sex
# origwt, full sample weights
# repgrp1, stratum indicator
# jkunit, PSU indicator
edf <- getData(data=sdf, varnames=c(items, "dsex", "origwt", "repgrp1", "jkunit", "sdracem"),
               omittedLevels = FALSE, returnJKreplicates=FALSE)
# make up a student ID
edf$sid <- paste0("S", 1:nrow(edf))
# apply simplified scoring
for(i in 1:length(items)) {
  coli <- items[i]
  # save the original
  rawcol <- paste0(coli, "raw")
  edf[,rawcol] <- edf[,coli]
  if( coli %in% dichotParamTab$ItemID) {
    edf[,coli] <- ifelse(grepl("ABCDE", edf[,rawcol]), 0, NA)
    edf[,coli] <- ifelse(grepl("*", edf[,rawcol]), 1, edf[,coli])
  } else {
    # scale for m066501
    edf[,coli] <- ifelse(grepl("Incorrect", edf[,rawcol]), 0, NA)
    edf[,coli] <- ifelse(grepl("Partial", edf[,rawcol]), 1, edf[,coli])
    edf[,coli] <- ifelse(grepl("Correct", edf[,rawcol]), 2, edf[,coli])
    # scale for m0757c1
    edf[,coli] <- ifelse(grepl("None correct", edf[,rawcol]), 0, edf[,coli])
    edf[,coli] <- ifelse(grepl("One correct", edf[,rawcol]), 1, edf[,coli])
    edf[,coli] <- ifelse(grepl("Two correct", edf[,rawcol]), 2, edf[,coli])
    edf[,coli] <- ifelse(grepl("Three correct", edf[,rawcol]), 3, edf[,coli])
  }
  edf[,rawcol] <- NULL # delete original
}

# stuItems has one row per student/item combination
stuItems <- edf[,c("sid", items)]
stuItems <- reshape(data=stuItems, varying=c(items), idvar=c("sid"),
                   direction="long", v.names="score", times=items, timevar="key")
# stuDat is one row per student and contains student-level information
stuDat <- edf[,c("sid", "origwt", "repgrp1", "jkunit", "dsex", "sdracem")]

# testDat shows scaling and weights for subtests, an overall score can be treated as a subtest
testDat <- data.frame(test=c("composite", "composite"),
                     subtest=c("num", "alg"),
                     location=c(277.1563, 280.2948),
                     scale=c(37.7297, 36.3887),
                     subtestWeight=c(0.3, 0.7))

# estimate a regression for Algebra subscale
mmlA <- mml(alg ~ dsex,
            stuItems=stuItems, stuDat=stuDat,
            dichotParamTab=dichotParamTab, polyParamTab=polyParamTab,
            testScale=testDat,

```

```

        idVar="sid", weightVar="origwt", # these are column names on stuDat
        strataVar="repgrp1", PSUVar="jkunit")
# summary, with Taylor standard errors
mmlAs <- summary.mmlMeans(mmlA, varType="Taylor")

# estimate a regression for Numeracy subscale
mmlN <- mml(num ~ dsex,
            stuItems=stuItems, stuDat=stuDat,
            dichotParamTab=dichotParamTab, polyParamTab=polyParamTab,
            testScale=testDat,
            idVar="sid", weightVar="origwt", # these are column names on stuDat
            strataVar="repgrp1", PSUVar="jkunit")
# summary, with Taylor standard errors
mmlNs <- summary.mmlMeans(mmlN, varType="Taylor")
mmlNs

# draw plausible values for mmlA
head(pvd <- drawPVs.mmlMeans(mmlA))
# alternative specification
head(pvs <- drawPVs.mmlMeans(summary.mmlMeans(mmlA, varType="Taylor"), stochasticBeta=TRUE))

# composite regression
mmlC <- mml(composite ~ dsex ,
            stuItems=stuItems, stuDat=stuDat,
            dichotParamTab=dichotParamTab, polyParamTab=polyParamTab,
            testScale=testDat,
            idVar="sid", weightVar="origwt", # these are column names on stuDat
            strataVar="repgrp1", PSUVar="jkunit")
# summary, with Taylor standard errors
summary(mmlC, varType="Taylor")

# draw plausible values for mmlC
head(pvd <- drawPVs.mmlCompositeMeans(mmlC))
# alternative specification
mmlCsum <- summary.mmlCompositeMeans(mmlC, varType="Taylor")
head(pvs <- drawPVs.mmlCompositeMeans(mmlCsum, stochasticBeta=TRUE))

## End(Not run)

```

# Index

drawPVs, 2

mm1, 3, 4, 4