Package ‘lpme’

June 24, 2025

Title Measurement Error Analysis and Correction Under Identification Restrictions

Version 0.1

Description An R package for analyzing latent variable models with measurement error correction, including Item Response Theory (IRT) models. It provides tools for implementing various correction methods such as Bayesian MCMC, overimputation, bootstrapping for robust standard errors, OLS, and IV-based approaches. The package supports flexible specification of observable indicators and groupings, making it suitable for latent variable analyses in social sciences and other fields.

Depends R (>= 3.3.3)

License GPL-3

Encoding UTF-8

LazyData false

Maintainer Connor Jerzak <connor.jerzak@gmail.com>

Imports reticulate,
stats,
sensemakr,
pscl,
AER,
sandwich,
mvtnorm,
Amelia,
emIRT,
gtools

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

URL https://github.com/cjerzak/lpme

BugReports https://github.com/cjerzak/lpme/issues

Contents

`build_backend` *A function to build the environment for lpme. Builds a conda environment in which 'JAX', 'numpyro', and 'numpy' are installed. Users can also create a conda environment where 'JAX' and 'numpy' are installed themselves.*

Description

A function to build the environment for lpme. Builds a conda environment in which 'JAX', 'numpyro', and 'numpy' are installed. Users can also create a conda environment where 'JAX' and 'numpy' are installed themselves.

Usage

```
build_backend(conda_env = "lpme", conda = "auto")
```

Arguments

`conda_env`	(default = `"lpme"`) Name of the conda environment in which to place the backends.
`conda`	(default = `auto`) The path to a conda executable. Using `"auto"` allows reticulate to attempt to automatically find an appropriate conda binary.

Value

Invisibly returns NULL; this function is used for its side effects of creating and configuring a conda environment for `lpme`. This function requires an Internet connection. You can find out a list of conda Python paths via: `Sys.which("python")`

Examples

```
## Not run:
# Create a conda environment named "lpme"
# and install the required Python packages (jax, numpy, etc.)
build_backend(conda_env = "lpme", conda = "auto")

# If you want to specify a particular conda path:
# build_backend(conda_env = "lpme", conda = "/usr/local/bin/conda")

## End(Not run)
```

infer_orientation_signs — *Infer orientation signs for each observable indicator*

Description

This helper analyzes observable indicators and returns a numeric vector of 1 or -1 for use with the orientation_signs argument in lpme. Each sign is chosen so that the correlation between the oriented indicator and either the outcome Y or the first principal component of the indicators is positive.

Usage

```
infer_orientation_signs(Y, observables, method = c("Y", "PC1"))
```

Arguments

Y	Numeric outcome vector. Only used when method = "Y".
observables	A matrix or data frame of binary observable indicators.
method	Character string specifying how to orient the indicators. "Y" orient each indicator so that its correlation with Y is positive. "PC1" orient each indicator so that its correlation with the first principal component of observables is positive. Default is "Y".

Value

A numeric vector of length ncol(observables) containing 1 or -1.

Examples

```
set.seed(1)
Y <- rnorm(10)
obs <- data.frame(matrix(sample(c(0,1), 20, replace = TRUE), ncol = 2))
infer_orientation_signs(Y, obs)
```

lpme — *lpme*

Description

Implements bootstrapped analysis for latent variable models with measurement error correction

Usage

```
lpme(
  Y,
  observables,
  observables_groupings = colnames(observables),
  orientation_signs = NULL,
  make_observables_groupings = FALSE,
  n_boot = 32L,
  n_partition = 10L,
  boot_basis = 1:length(Y),
  return_intermediaries = TRUE,
  ordinal = FALSE,
  estimation_method = "em",
  latent_estimation_fn = NULL,
 mcmc_control = list(backend = "pscl", n_samples_warmup = 500L, n_samples_mcmc = 1000L,
    batch_size = 512L, chain_method = "parallel", subsample_method = "full",
    anchor_parameter_id = NULL, n_thin_by = 1L, n_chains = 2L),
  conda_env = "lpme",
  conda_env_required = TRUE
)
```

Arguments

`Y` A vector of observed outcome variables

`observables` A matrix of observable indicators used to estimate the latent variable

`observables_groupings`
A vector specifying groupings for the observable indicators. Default is column names of observables.

`orientation_signs`
(optional) A numeric vector of length equal to the number of columns in ‘observables‘, containing 1 or -1 to indicate the desired orientation of each column. If provided, each column of ‘observables‘ will be oriented by this sign before analysis. Default is NULL (no orientation applied).

`make_observables_groupings`
Logical. If TRUE, creates dummy variables for each level of the observable indicators. Default is FALSE.

`n_boot` Integer. Number of bootstrap iterations. Default is 32.

`n_partition` Integer. Number of partitions for each bootstrap iteration. Default is 10.

`boot_basis` Vector of indices or grouping variable for stratified bootstrap. Default is 1:length(Y).

`return_intermediaries`
Logical. If TRUE, returns intermediate results. Default is TRUE.

`ordinal` Logical indicating whether the observable indicators are ordinal (TRUE) or binary (FALSE).

`estimation_method`
Character specifying the estimation approach. Options include:

- "em" (default): Uses expectation-maximization via `emIRT` package. Supports both binary (via `emIRT::binIRT`) and ordinal (via `emIRT::ordIRT`) indicators.
- "averaging": Uses feature averaging.

- "mcmc": Markov Chain Monte Carlo estimation using either `pscl::ideal` (R backend) or `numpyro` (Python backend)
- "mcmc_joint": Full Bayesian model that simultaneously estimates latent variables and outcome relationship using `numpyro`
- "mcmc_overimputation": Two-stage MCMC approach with measurement error correction via over-imputation
- "custom": In this case, latent estimation performed using `latent_estimation_fn`.

`latent_estimation_fn`

Custom function for estimating latent trait from `observables` if `estimation_method="custom"` (optional). The function should accept a matrix of observables (rows are observations) and return a numeric vector of length equal to the number of observations.

`mcmc_control` A list indicating parameter specifications if MCMC used.

- `backend` Character string indicating the MCMC engine to use. Valid options are:
  - `"numpyro"` (default): Uses the Python `numpyro` package via `reticulate`.
  - `"pscl"`: Uses the R-based `pscl::ideal` function.
- `n_samples_warmup` Integer specifying the number of warm-up (a.k.a. burn-in) iterations before samples are collected. Default is `500`.
- `n_samples_mcmc` Integer specifying the number of post-warmup MCMC iterations to retain. Default is `1000`.
- `chain_method` Character string passed to `numpyro` specifying how to run multiple chains. Typical options include:
  - `"parallel"` (default): Runs chains in parallel.
  - `"sequential"`: Runs chains sequentially.
  - `"vectorized"`: Vectorized evaluation of multiple chains.
- `n_thin_by` Integer indicating the thinning factor for MCMC samples (i.e., retaining every `n_thin_by`-th sample). Default is `1`.
- `n_chains` Integer specifying the number of parallel MCMC chains to run. Default is `2`.

`conda_env` A character string specifying the name of the conda environment to use via `reticulate`. Default is `"lpme"`.

`conda_env_required`

A logical indicating whether the specified conda environment must be strictly used. If `TRUE`, an error is thrown if the environment is not found. Default is `TRUE`.

Details

This function implements a bootstrapped latent variable analysis with measurement error correction. It performs multiple bootstrap iterations, each with multiple partitions. For each partition, it calls the `lpme_onerun` function to estimate latent variables and apply various correction methods. The results are then aggregated across partitions and bootstrap iterations to produce final estimates and bootstrap standard errors.

Value

A list containing various estimates and statistics (in snake_case):

- `ols_coef`: Coefficient from naive OLS regression.

- `ols_se`: Standard error of naive OLS coefficient.
- `ols_tstat`: T-statistic of naive OLS coefficient.
- `iv_coef`: Coefficient from instrumental variable (IV) regression.
- `iv_se`: Standard error of IV regression coefficient.
- `iv_tstat`: T-statistic of IV regression coefficient.
- `corrected_iv_coef`: IV regression coefficient corrected for measurement error.
- `corrected_iv_se`: Standard error of the corrected IV coefficient (currently `NA`).
- `corrected_iv_tstat`: T-statistic of the corrected IV coefficient.
- `var_est`: Estimated variance of the measurement error (split-half variance).
- `corrected_ols_coef`: OLS coefficient corrected for measurement error.
- `corrected_ols_se`: Standard error of the corrected OLS coefficient (currently `NA`).
- `corrected_ols_tstat`: T-statistic of the corrected OLS coefficient (currently `NA`).
- `corrected_ols_coef_alt`: Alternative corrected OLS coefficient (if applicable).
- `corrected_ols_se_alt`: Standard error for the alternative corrected OLS coefficient (if applicable).
- `corrected_ols_tstat_alt`: T-statistic for the alternative corrected OLS coefficient (if applicable).
- `bayesian_ols_coef_outer_normed`: Posterior mean of the OLS coefficient under MCMC, after normalizing by the overall sample standard deviation.
- `bayesian_ols_se_outer_normed`: Posterior standard error corresponding to `bayesian_ols_coef_outer_normed`
- `bayesian_ols_tstat_outer_normed`: T-statistic for `bayesian_ols_coef_outer_normed`.
- `bayesian_ols_coef_inner_normed`: Posterior mean of the OLS coefficient under MCMC, after normalizing each posterior draw individually.
- `bayesian_ols_se_inner_normed`: Posterior standard error corresponding to `bayesian_ols_coef_inner_normed`
- `bayesian_ols_tstat_inner_normed`: T-statistic for `bayesian_ols_coef_inner_normed`.
- `m_stage_1_erv`: Extreme robustness value (ERV) for the first-stage regression (`x_est2` on `x_est1`), if computed.
- `m_reduced_erv`: ERV for the reduced model (`Y` on `x_est1`), if computed.
- `x_est1`: First set of latent variable estimates.
- `x_est2`: Second set of latent variable estimates.

Examples

```
# Generate some example data
set.seed(123)
Y <- rnorm(1000)
observables <- as.data.frame( matrix(sample(c(0,1), 1000*10, replace = TRUE), ncol = 10) )

# Run the bootstrapped analysis
results <- lpme(Y = Y,
                observables = observables,
                n_boot = 10,    # small values for illustration only
                n_partition = 5 # small for size
                )

# View the corrected IV coefficient and its standard error
print(results)
```

lpme_onerun *lpme_onerun*

Description

Implements analysis for latent variable models with measurement error correction

Usage

```
lpme_onerun(
  Y,
  observables,
  observables_groupings = colnames(observables),
  make_observables_groupings = FALSE,
  estimation_method = "em",
  latent_estimation_fn = NULL,
 mcmc_control = list(backend = "pscl", n_samples_warmup = 500L, n_samples_mcmc = 1000L,
   batch_size = 512L, chain_method = "parallel", subsample_method = "full", n_thin_by =
    1L, n_chains = 2L),
  ordinal = FALSE,
  conda_env = "lpme",
  conda_env_required = TRUE
)
```

Arguments

`Y` A vector of observed outcome variables

`observables` A matrix of observable indicators used to estimate the latent variable

`observables_groupings`
A vector specifying groupings for the observable indicators. Default is column names of observables.

`make_observables_groupings`
Logical. If TRUE, creates dummy variables for each level of the observable indicators. Default is FALSE.

`estimation_method`
Character specifying the estimation approach. Options include:

- "em" (default): Uses expectation-maximization via `emIRT` package. Supports both binary (via `emIRT::binIRT`) and ordinal (via `emIRT::ordIRT`) indicators.
- "averaging": Uses feature averaging.
- "mcmc": Markov Chain Monte Carlo estimation using either `pscl::ideal` (R backend) or `numpyro` (Python backend)
- "mcmc_joint": Joint Bayesian model that simultaneously estimates latent variables and outcome relationship using `numpyro`
- "mcmc_overimputation": Two-stage MCMC approach with measurement error correction via over-imputation
- "custom": In this case, latent estimation performed using `latent_estimation_fn`.

latent_estimation_fn
: Custom function for estimating latent trait from observables if estimation_method="custom" (optional). The function should accept a matrix of observables (rows are observations) and return a numeric vector of length equal to the number of observations.

mcmc_control
: A list indicating parameter specifications if MCMC used.

- backend Character string indicating the MCMC engine to use. Valid options are:
  - "numpyro" (default): Uses the Python numpyro package via reticulate.
  - "pscl": Uses the R-based pscl::ideal function.
- n_samples_warmup Integer specifying the number of warm-up (a.k.a. burn-in) iterations before samples are collected. Default is 500.
- n_samples_mcmc Integer specifying the number of post-warmup MCMC iterations to retain. Default is 1000.
- chain_method Character string passed to numpyro specifying how to run multiple chains. Typical options include:
  - "parallel" (default): Runs chains in parallel.
  - "sequential": Runs chains sequentially.
  - "vectorized": Vectorized evaluation of multiple chains.
- n_thin_by Integer indicating the thinning factor for MCMC samples (i.e., retaining every n_thin_by-th sample). Default is 1.
- n_chains Integer specifying the number of parallel MCMC chains to run. Default is 2.

ordinal
: Logical indicating whether the observable indicators are ordinal (TRUE) or binary (FALSE).

conda_env
: A character string specifying the name of the conda environment to use via reticulate. Default is "lpme".

conda_env_required
: A logical indicating whether the specified conda environment must be strictly used. If TRUE, an error is thrown if the environment is not found. Default is TRUE.

Details

This function implements a latent variable analysis with measurement error correction. It splits the observable indicators into two sets, estimates latent variables using each set, and then applies various correction methods including OLS correction and instrumental variable approaches.

Value

A list containing various estimates and statistics:

- ols_coef: Coefficient from naive OLS regression
- ols_se: Standard error of naive OLS coefficient
- ols_tstat: T-statistic of naive OLS coefficient
- corrected_ols_coef: OLS coefficient corrected for measurement error
- corrected_ols_se: Standard error of corrected OLS coefficient (currently NA)
- corrected_ols_tstat: T-statistic of corrected OLS coefficient (currently NA)
- corrected_ols_coef_alt: Alternative corrected OLS coefficient (currently NA)

- `iv_coef`: Coefficient from instrumental variable regression
- `iv_se`: Standard error of IV regression coefficient
- `iv_tstat`: T-statistic of IV regression coefficient
- `corrected_iv_coef`: IV regression coefficient corrected for measurement error
- `corrected_iv_se`: Standard error of corrected IV coefficient
- `corrected_iv_tstat`: T-statistic of corrected IV coefficient
- `var_est_split`: Estimated variance of the measurement error
- `x_est1`: First set of latent variable estimates
- `x_est2`: Second set of latent variable estimates

Examples

```
# Generate some example data
set.seed(123)
library( lpme )
Y <- rnorm(1000)
observables <- as.data.frame( matrix(sample(c(0,1), 1000*10, replace = TRUE), ncol = 10) )

# Run the analysis
results <- lpme_onerun(Y = Y,
                       observables = observables)

# View the corrected estimates
print(results)
```

QJEData *QJEData: Agricultural Treatment Experiment Data*

Description

Data from a field experiment studying moral hazard in tenancy contracts in agriculture.

After subsetting, this dataset includes observations on 968 experimental units with the following variables of interest: household composition, treatment assignment, and agricultural outcomes.

Usage

```
data(QJEData)
```

Format

A data frame with 968 rows and 7 columns:

children Numeric (integer). Number of children in the household. Larger numbers may reflect increased household labor needs and different investment or effort incentives.

married Numeric/binary. Whether the household head is currently married (1) or not (0). Marital status may influence decision-making and risk preferences in farming.

hh_size Numeric (integer). Household size. Differences in family labor availability or consumption needs can influence effort levels and thus relate to moral hazard in production decisions.

hh_sexrat Numeric. The ratio of adult men to adult women in the household. Imbalances in the male–female ratio can affect labor division and investment decisions.

treat1 Numeric/binary. Primary treatment indicator (e.g., whether a farmer is offered a specific tenancy contract or cost-sharing arrangement).

R_yield_ELA_sqm Numeric. Crop yield per square meter (e.g., kilograms of output per square meter). This is a principal outcome measure for evaluating productivity and treatment impact on farm performance.

ELA_Fertil_D Numeric/binary. Indicator for whether fertilizer was used (1) or not (0). This measures input investment—a key mechanism in moral hazard models (farmers may alter input use under different contracts).

Source

Burchardi, K.B., Ghatak, M., & Johanssen, A. (2019). Moral hazard: Experimental evidence from tenancy contracts. *The Quarterly Journal of Economics*, 134(1), 281-347.

Index