Package 'lpmec'

Title Measurement Error Analysis and Correction Under Identification Restrictions

Version 1.1.4

Author Connor Jerzak [aut, cre], Stephen Jessee [aut]

Description Implements methods for analyzing latent variable models with measurement error correction, including Item Response Theory (IRT) models. Provides tools for various correction methods such as Bayesian Markov Chain Monte Carlo (MCMC), over-imputation, bootstrapping for robust standard errors, Ordinary Least Squares (OLS), and Instrumental Variables (IV) based approaches. Supports flexible specification of observable indicators and groupings for latent variable analyses in social sciences and other fields. Methods are described in a working paper (2025) <doi:10.48550/arXiv.2507.22218>.

Depends R (>= 3.5.0)

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

SystemRequirements Python (>= 3.10) with jax, numpy, numpyro (optional; for NumPyro backend via reticulate)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

URL https://github.com/cjerzak/lpmec-software

BugReports https://github.com/cjerzak/lpmec-software/issues

Contents

---

build_backend | *A function to build the environment for lpmec. 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 lpmec. 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 = "lpmec", conda = "auto")
```

Arguments

conda_env | (default = "lpmec") 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 lpmec. 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 "lpmec"
# and install the required Python packages (jax, numpy, etc.)
build_backend(conda_env = "lpmec", conda = "auto")

# If you want to specify a particular conda path:
```

```
# build_backend(conda_env = "lpmec", 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 lpmec. 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)
```

KnowledgeVoteDuty	*KnowledgeVoteDuty: Survey Respondents' Views of Voting as a Duty and Political Knowledge Questions*

Description

KnowledgeVoteDuty is a modified set of responses to a small set of questions on the American National Election Study's 2024 Time Series Study. These data only include respondents who had non-missing values on all of the variables included, dropping respondents with one or more missing values.

Usage

```
data(KnowledgeVoteDuty)
```

Format

A data frame with 3,059 observations and 5 variables:

voteduty Whether respondents feel that voting is a duty or a choice. Values range from 1 to 7, with 1 being "Very strongly a duty" and 7 being "Very strongly a choice," created based on variable V241218x.

SenateTerm Dummy variable (0 or 1) for whether respondent correctly stated the length of a U.S. Senate term. Created based on variable V241612.

SpendLeast Dummy variable (0 or 1) for whether respondent correctly identified "Foreign aid" from a list as the category the federal government spends the least on. Created based on variable V241613.

HouseParty Dummy variable (0 or 1) for whether respondent correctly identified the political party that currently has the most members in the U.S. House of Representatives. Created based on variable V241614.

SenateParty Dummy variable (0 or 1) for whether respondent correctly identified the political party that currently has the most members in the U.S. Senate. Created based on variable V241615.

References

American National Election Studies. 2024. ANES 2024 Time Series Study Full Release [dataset and documentation]. Available at electionstudies.org.

Examples

```
data(KnowledgeVoteDuty)
voteduty <- KnowledgeVoteDuty$voteduty
knowledge <- scale(rowMeans(KnowledgeVoteDuty[ , -1]))
summary(lm(voteduty ~ knowledge))
```

---

lpmec   *lpmec*

Description

Implements bootstrapped analysis for latent variable models with measurement error correction

Usage

```
lpmec(
  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 = "lpmec",
  conda_env_required = FALSE
)
```

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.
- "pca": First principal component of observables.
- "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 `"pscl"` (default, uses the R-based `pscl::ideal` function) or `"numpyro"` (uses the Python numpyro package via reticulate).

n_samples_warmup Integer specifying the number of warm-up (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. Options: `"parallel"` (default), `"sequential"`, or `"vectorized"`.

n_thin_by Integer indicating the thinning factor for MCMC samples. 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 `"lpmec"`.

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 FALSE.

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 `lpmec_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.

References

Jerzak, C. T. and Jessee, S. A. (2025). Attenuation Bias with Latent Predictors. arXiv:2507.22218 [stat.AP]. https://arxiv.org/abs/2507.22218

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 <- lpmec(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)
```

---

lpmec_onerun	*lpmec_onerun*

Description

Implements analysis for latent variable models with measurement error correction

Usage

```
lpmec_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 = "lpmec",
  conda_env_required = FALSE
)
```

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.
- "pca": First principal component of observables.
- "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 `"pscl"` (default, uses the R-based `pscl::ideal` function) or `"numpyro"` (uses the Python numpyro package via reticulate).

n_samples_warmup Integer specifying the number of warm-up (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. Options: `"parallel"` (default), `"sequential"`, or `"vectorized"`.

n_thin_by Integer indicating the thinning factor for MCMC samples. 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 `"lpmec"`.

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 FALSE.

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
- `iv_coef_a`: IV coefficient using first split as instrument
- `iv_coef_b`: IV coefficient using second split as instrument
- `iv_coef`: Averaged IV coefficient from both splits
- `iv_se`: Standard error of IV regression coefficient
- `iv_tstat`: T-statistic of IV regression coefficient
- `corrected_iv_coef_a`: Corrected IV coefficient using first split as instrument
- `corrected_iv_coef_b`: Corrected IV coefficient using second split as instrument
- `corrected_iv_coef`: Averaged corrected IV coefficient from both splits
- `corrected_iv_se`: Standard error of corrected IV coefficient
- `corrected_iv_tstat`: T-statistic of corrected IV coefficient
- `corrected_ols_coef_a`: Corrected OLS coefficient using first split
- `corrected_ols_coef_b`: Corrected OLS coefficient using second split
- `corrected_ols_coef`: Averaged corrected OLS coefficient from both splits
- `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)
- `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

Standard Errors

The following standard errors and t-statistics are currently returned as NA because their analytical derivation is not yet implemented:

- `corrected_ols_se`: Standard error for the corrected OLS coefficient
- `corrected_ols_tstat`: T-statistic for the corrected OLS coefficient
- `corrected_ols_coef_alt`: Alternative corrected OLS coefficient

For inference on these quantities, use the bootstrap approach via `lpmec`, which provides valid confidence intervals and standard errors through resampling.

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 analysis
results <- lpmec_onerun(Y = Y,
```

```
                            observables = observables)

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

---

plot.lpmec   *Plot method for lpmec objects*

Description

Creates visualizations of LPMEC model results. Can plot either the latent variable estimates or the bootstrap distribution of coefficients.

Usage

```
## S3 method for class 'lpmec'
plot(x, type = "latent", ...)
```

Arguments

x	An object of class lpmec returned by [lpmec]().
type	Character string specifying the plot type. Either "latent" (default) for a scatter plot of split-half latent estimates, or "coefficients" for a density plot of bootstrap coefficient estimates.
...	Additional arguments passed to [plot]() or [density]().

Value

No return value, called for side effects (creates a plot).

See Also

[lpmec](), [summary.lpmec](), [print.lpmec]()

---

plot.lpmec_onerun   *Plot method for lpmec_onerun objects*

Description

Creates a scatter plot comparing the two split-half latent variable estimates.

Usage

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

Arguments

x   An object of class lpmec_onerun returned by `lpmec_onerun`.

...   Additional arguments passed to `plot`.

Value

No return value, called for side effects (creates a plot).

See Also

`lpmec_onerun`, `summary.lpmec_onerun`, `print.lpmec_onerun`

---

print.lpmec   *Print method for lpmec objects*

Description

Prints a concise summary of bootstrapped LPMEC model results.

Usage

```
## S3 method for class 'lpmec'
print(x, ...)
```

Arguments

x   An object of class lpmec returned by `lpmec`.

...   Additional arguments (currently unused).

Value

The input object x, returned invisibly.

See Also

`lpmec`, `summary.lpmec`, `plot.lpmec`

print.lpmec_onerun   *Print method for lpmec_onerun objects*

Description

Prints a concise summary of single-run LPMEC model results.

Usage

```
## S3 method for class 'lpmec_onerun'
print(x, ...)
```

Arguments

x	An object of class lpmec_onerun returned by [lpmec_onerun](#).
...	Additional arguments (currently unused).

Value

The input object x, returned invisibly.

See Also

[lpmec_onerun](#), [summary.lpmec_onerun](#), [plot.lpmec_onerun](#)

summary.lpmec   *Summary method for lpmec objects*

Description

Provides a comprehensive summary of bootstrapped LPMEC model results including OLS, IV, corrected, and Bayesian coefficient estimates with confidence intervals.

Usage

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

Arguments

object	An object of class lpmec returned by [lpmec](#).
...	Additional arguments (currently unused).

Value

A data frame containing coefficient estimates, standard errors, and confidence intervals, returned invisibly. The data frame has rows for OLS, IV, Corrected IV, Corrected OLS, and Bayesian OLS estimates.

See Also

[lpmec](#), [print.lpmec](#), [plot.lpmec](#)

summary.lpmec_onerun   *Summary method for lpmec_onerun objects*

Description

Provides a summary of single-run LPMEC model results including OLS, IV, and corrected coefficient estimates.

Usage

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

Arguments

object	An object of class lpmec_onerun returned by `lpmec_onerun`.
...	Additional arguments (currently unused).

Value

A data frame containing coefficient estimates and standard errors, returned invisibly. The data frame has rows for OLS, IV, Corrected IV, and Corrected OLS estimates.

See Also

`lpmec_onerun`, `print.lpmec_onerun`, `plot.lpmec_onerun`

Index