Package: klue (via r-universe)

May 28, 2026

Type Package

Title Hybrid Machine Learning and Random-Utility Workflow for Latent Class Multinomial Logit Model Specification

Version 0.6.3

Author Oliver Frings [aut, cre]

Maintainer Oliver Frings <oliver.frings@agroparistech.fr>

Description Implements the three-step workflow from Frings (2026, working paper) for specifying latent class multinomial logit (LCMNL) models. The maximum-likelihood multi-start is initialised from six clusterings of respondents' revealed-preference signatures (k-means, Gaussian mixture, hierarchical clustering with Ward, complete, and average linkage, and partitioning around medoids); LCMNL is estimated across a user-specified range of class counts; and a mixed multinomial logit (MMNL) benchmark is reported alongside BIC, AIC, ICL, and a classification-entropy diagnostic. Accepts long- or wide-format discrete-choice data with optional availability columns. Validated against five public reference datasets (Vittel, Apollo mode and route choice, Electricity, Swissmetro). Wraps the 'apollo' package for maximum-likelihood estimation.

License MIT + file LICENSE

URL https://github.com/o-frings/klue

BugReports https://github.com/o-frings/klue/issues

Encoding UTF-8

LazyData false

Depends R (>= 3.6.0), apollo (>= 0.3.0)

Imports mclust, cluster, stats, utils

Suggests mlogit, testthat

RoxygenNote 7.3.0

Config/pak/sysreqs libicu-dev

Repository https://o-frings.r-universe.dev

Date/Publication 2026-05-28 17:33:25 UTC

RemoteUrl https://github.com/o-frings/klue

RemoteRef main

RemoteSha 6ec0a5dc345dfb6786ab0e1ee8a986b5ac652571

Contents

`klue-package` *klue: Hybrid Machine Learning and Random-Utility Workflow for Latent Class Multinomial Logit Model Specification*

Description

Implements the three-step workflow from Frings (2026, working paper) for specifying latent class multinomial logit (LCMNL) models: initialise the maximum-likelihood multi-start from six clusterings of respondents' revealed-preference signatures (k-means, Gaussian mixture, hierarchical clustering with Ward / complete / average linkage, and partitioning around medoids); estimate LCMNL across a user-specified range of class counts; and benchmark against a mixed multinomial logit (MMNL, independent and optionally correlated normals) with BIC, AIC, ICL, and a classification-entropy diagnostic.

Public API by topic

Entry point `klue` – main workflow runner. `klue_demo` – zero-setup demo on bundled data.

Data harmonising `klue_database`, `klue_database_long`, `klue_database_wide` – reshape long- or wide-format discrete-choice data to the canonical format the engine consumes; handle availability filtering and per-attribute scaling.

Data simulation (Monte Carlo) `klue_simulate`, `klue_simulate_cov`, `klue_simulate_deff` – generate synthetic panel choice data from a known LCMNL DGP. The standard random-attribute design; a variant with concomitant covariates driving class membership; and a D-efficient design variant.

Estimators (advanced) `klue_lcmnl` – LCMNL multi-start. `klue_mmnl` – MMNL with independent normals. `klue_mmnl_corr` – MMNL with correlated normals (full Cholesky covariance).

Configuration klue_dgp – design / parameter config (number of attributes, alternatives, mean-beta vector). klue_mmnl_defaults – active default settings for both MMNL estimators (draws, cores, routine, bounds).

Paper replication klue_study – runs all 11 component drivers below (~12 h end-to-end). klue_study_main – 420-condition Monte Carlo. klue_study_mmnl, klue_study_mmnl_corr – MMNL benchmark and correlated-MMNL robustness. klue_study_convergence, klue_study_initialisation – starting-value ablations. klue_study_unbalanced, klue_study_design, klue_study_concomitant, klue_study_clustering, klue_study_recovery, klue_study_sample – robustness analyses across nuisance design factors.

Quick start

```
library(klue)
klue_demo()                              # see it work
res <- klue(data = "my.csv", format = "long",
            id_col = "id", task_col = "task",
            alt_col = "alt", choice_col = "chosen",
            attribute_cols = c("a", "b"), price_col = "price")
res$summary; res$best_C; res$class_betas
```

Backward compatibility

Every klue_* function has a deprecated alias preserving the pre-v0.4.0 name (run_lcmnl_workflow, build_database*, estimate_*, make_dgp_config, generate_data*, run_*). Same function object, slated for removal in a future major release.

Citation

See citation("klue") for the BibTeX bundle: the klue paper plus **apollo** (Hess & Palma 2019; the underlying estimator), **mclust** (Scrucca et al. 2016; GMM clustering) and **cluster** (Maechler et al.; PAM).

Validation

The hybrid workflow has been validated bit-exactly against five public reference applications (Vittel water-quality discrete choice experiment, Apollo mode and route choice, Electricity, Swissmetro).

Author(s)

Oliver Frings <oliver.frings@agroparistech.fr>

References

Frings, O. (2026). *A Hybrid Machine Learning and Random Utility Framework for Latent Class Model Specification*. Working paper.

Hess, S., & Palma, D. (2019). Apollo: a flexible, powerful and customisable freeware package for choice model estimation and application. *Journal of Choice Modelling*, 32, 100170.

klue — *Run the full LCMNL specification workflow on a discrete-choice dataset*

Description

Estimates LCMNL for a range of class counts, runs an MMNL benchmark (independent and optionally correlated normals), produces a BIC / AIC / ICL / entropy summary table and a class-specific coefficient table, and writes CSVs to disk.

`run_lcmnl_workflow` is a deprecated alias for the same function.

Usage

```
klue(database = NULL,
     data = NULL,
     format = c("auto", "long", "wide"),
     C_cands = 1:6,
     run_mmnl = TRUE,
     run_mmnl_corr = FALSE,
     mmnl_opts = list(),
     attr_labels = NULL,
     output_prefix = "workflow",
     output_dir = NULL,
     write_csv = TRUE,
     verbose = TRUE,
     ...)
```

Arguments

`database`	An already-built canonical wide data frame (columns `ID`, `TASK`, `x1_1..xN_J`, `price_1..price_J`, `CHOICE`). If supplied, `data` and the column-mapping arguments are ignored.
`data`	A data frame or CSV path in long or wide format. `klue_database` is invoked internally; pass the appropriate column-mapping arguments via `....`
`format`	Forwarded to `klue_database`.
`C_cands`	Integer vector of class counts to estimate. Default `1:6`. `C = 1` is the MNL baseline.
`run_mmnl`	If `TRUE`, also estimates an MMNL benchmark with independent normal random coefficients (log-normal on price).
`run_mmnl_corr`	If `TRUE`, additionally estimates a correlated MMNL benchmark with a full Cholesky-parameterised covariance matrix over random coefficients (log-normal on price).
`mmnl_opts`	Named list of arguments forwarded to BOTH `klue_mmnl` and `klue_mmnl_corr`. See `klue_mmnl_defaults` for the active defaults.
`attr_labels`	Override display column names in the betas CSV.

output_prefix	Filename prefix for the written CSVs.
output_dir	Output directory. Default: getOption("klue.output_dir", "output").
write_csv	Whether to write CSV files (TRUE by default).
verbose	Per-C progress lines and a final printout.
...	Column-mapping arguments forwarded to klue_database.

Value

Invisibly, a list:

database	Canonical wide data frame used for estimation.
dgp	Output of klue_dgp.
lcmnl	Named list of per-C fits.
mmnl	Independent-normals MMNL fit (NULL if disabled or failed).
mmnl_corr	Correlated MMNL fit (NULL if run_mmnl_corr = FALSE or it failed).
summary	Data frame with one row per C: LL, k, BIC, AIC, ICL, ICL_BIC, dBIC, dAIC, dICL, best_method.
class_betas	Per-class coefficients for the BIC-best model.
comparison	MNL vs LCMNL-best vs MMNL (independent) vs MMNL (correlated), with whichever subset was estimated.
best_C	BIC-best number of classes.
best_lcmnl	The BIC-best fit.

Files written

When write_csv = TRUE:

- <prefix>_summary.csv
- <prefix>_class_betas.csv
- <prefix>_model_comparison.csv (if run_mmnl = TRUE or run_mmnl_corr = TRUE)

See Also

klue_database, klue_demo

Examples

```
## Not run:
# Long-format CSV
res <- klue(
  data = "data.csv", format = "long",
  id_col = "id", task_col = "task",
  alt_col = "alt", choice_col = "chosen",
  attribute_cols = c("a", "b", "c"), price_col = "price",
  C_cands = 1:6
)
```

```
res$summary
res$best_C

## End(Not run)
```

`klue_database` *Build a canonical wide database from long- or wide-format input*

Description

Reshape an arbitrary discrete-choice dataset into the canonical wide format expected by the estimation engine. Both helpers also handle availability filtering, per-attribute scaling, and balanced-panel enforcement.

`build_database`, `build_database_long` and `build_database_wide` are deprecated aliases.

Argument naming is homogenised across long and wide format: the same identifiers `attribute_cols`, `price_col`, `avail_col` are used in both, though their shape differs (scalar in long, vector of J column names in wide).

Usage

```
klue_database(data, format = c("auto", "long", "wide"), ...)

klue_database_long(data,
                   id_col, task_col, alt_col, choice_col,
                   attribute_cols, price_col,
                   choice_format = c("indicator", "alt_index"),
                   avail_col = NULL,
                   scalings = NULL,
                   price_scaling = 1,
                   verbose = TRUE)

klue_database_wide(data,
                   id_col, task_col = NULL, choice_col,
                   attribute_cols = NULL, price_col = NULL,
                   avail_col = NULL,
                   scalings = NULL,
                   attributes = NULL, price = NULL,
                   availability = NULL,
                   verbose = TRUE)
```

Arguments

`data`	A data frame or path to a CSV (or `.dat`) file.
`format`	`"long"` (one row per (id, task, alternative)) or `"wide"` (one row per (id, task) with alternative-suffixed columns). `"auto"` infers from whether `attribute_cols` is a named list ($\Rightarrow$ wide) or `alt_col` is supplied ($\Rightarrow$ long).

`id_col`, `task_col`, `alt_col`, `choice_col`
: Column names. For wide format, `task_col = NULL` synthesises 1..T per respondent; `choice_col` is the chosen alternative index (1..J).

`attribute_cols`
: In long format, a character vector of generic attribute column names. In wide format, a named list mapping each attribute to a length-J character vector of column names (one per alternative). `NA` in a wide slot encodes a structural zero on that alternative.

`price_col`
: In long format, the single price column name. In wide format, a length-J character vector of price column names.

`choice_format`
: (long) `"indicator"` = 0/1 chosen-row flag; `"alt_index"` = chosen-alt integer repeated on every row of the (id, task) group.

`avail_col`
: Optional availability column(s). Scalar in long, length-J in wide. Tasks where any alternative has `avail == 0` are dropped, with a drop-rate message.

`scalings`
: Optional named list of per-attribute scalings; each column is divided by the corresponding scalar. Keys: in long, match the entries of `attribute_cols` / `price_col`; in wide, match `names(attribute_cols)` or the literal `"price"`.

`price_scaling`
: (long, shortcut) Equivalent to `scalings = list(<price_col> = price_scaling)`.

`attributes`, `price`, `availability`
: Deprecated aliases for `attribute_cols`, `price_col`, `avail_col` in wide format. Accepted silently for backward compatibility; please switch to the canonical names.

`verbose`
: Print build progress, filter rates, and final dimensions.

`...`
: Passed to `klue_database_long` or `klue_database_wide`.

Value

A data frame with columns `ID`, `TASK`, `x1_1..xN_J`, `price_1..price_J`, `CHOICE`. Attributes set: `attr_labels` (display names for the betas table), `n_alternatives`, `n_generic`.

See Also

`klue`

Examples

```
## Not run:
# Long format
db <- klue_database_long(
  data = "data.csv",
  id_col = "respondent_id", task_col = "task",
  alt_col = "alternative", choice_col = "chosen",
  attribute_cols = c("attr1", "attr2"),
  price_col = "price",
  scalings = list(price = 10)
)

# Wide format
db <- klue_database_wide(
```

```
  data = mydata,
  id_col = "ID", task_col = "task", choice_col = "choice",
  attribute_cols = list(
    time = c("time_a", "time_b", "time_c"),
    qual = c("qual_a", "qual_b", "qual_c")
  ),
  price_col = c("cost_a", "cost_b", "cost_c"),
  avail_col = c("av_a", "av_b", "av_c"),
  scalings = list(time = 60, price = 10)
)

## End(Not run)
```

`klue_demo` *Run a zero-setup demo of the klue workflow*

Description

Estimates the klue workflow on Apollo's bundled Swiss route-choice dataset (348 commuters, 9 binary tasks). Useful as a first call to see the output shape before wiring your own data.

Usage

```
klue_demo(full = FALSE, verbose = TRUE)
```

Arguments

`full`	If `FALSE` (default), runs a fast slice: `C = 1:2`, no MMNL benchmark (~3 sec). If `TRUE`, runs the full workflow: `C = 1:6` + MMNL (~30 sec).
`verbose`	Print progress lines and the final summary. Default `TRUE`.

Value

Invisibly, the same results list returned by `klue`.

See Also

`klue`

Examples

```
## Not run:
library(klue)
klue_demo()              # fast slice
res <- klue_demo(full = TRUE)
res$summary
res$class_betas
res$best_C

## End(Not run)
```

klue_engine *Engine functions (advanced)*

Description

Low-level estimation engine used internally by klue. Most users will not need to call these directly.

- klue_dgp(n_generic, n_alternatives) returns a design configuration (attribute names, parameter counts) consumed by the estimators.
- klue_lcmnl(database, C, dgp) estimates an LCMNL with C classes from six clustering-based starting values and keeps the best by log-likelihood. Returns a fit object with LL, BIC, AIC, ICL, ICL_BIC, betas, class_probs, posteriors, and best_method.
- klue_mmnl(database, dgp, ...) estimates a mixed multinomial logit with independent normal random coefficients (log-normal on price). Returns LL, BIC, AIC, mu, sigma on success; reason + apollo_log_tail on failure.
- klue_mmnl_corr(database, dgp, ...) estimates a mixed multinomial logit with a full Cholesky-parameterised covariance matrix over random coefficients (log-normal on price). Same return shape as klue_mmnl but the number of free parameters scales as $n_{beta}(n_{beta}+1)/2$.
- klue_mmnl_defaults() returns the current default settings for both MMNL estimators (n_draws, n_draws_stage1, draws_type, estimation_routine, n_cores, quiet, mu_price_bounds, sigma_price_bounds). Override individually via the mmnl_opts argument of klue or with options(klue.mmnl.*).

The expected database is canonical wide format: columns ID, TASK, x1_1..xN_J, price_1..price_J, CHOICE, with CHOICE an integer index in 1..J.

make_dgp_config, estimate_lcmnl_multistart, estimate_mmnl, and estimate_mmnl_corr are deprecated aliases for the above.

Usage

```
klue_dgp(n_generic = 4, n_alternatives = 3)
klue_lcmnl(database, C, dgp = DGP_DEFAULT)
klue_mmnl(database, n_draws = NULL, n_draws_stage1 = NULL,
          draws_type = NULL, estimation_routine = NULL,
          n_cores = NULL, quiet = NULL, mu_price_bounds = NULL,
          sigma_price_bounds = NULL, dgp = DGP_DEFAULT)
klue_mmnl_corr(database, n_draws = NULL, n_draws_stage1 = NULL,
               draws_type = NULL, estimation_routine = NULL,
               n_cores = NULL, quiet = NULL, dgp = DGP_DEFAULT)
klue_mmnl_defaults()
```

Arguments

n_generic	Number of generic attributes (excluding price).
n_alternatives	Number of alternatives J. Last alternative is the reference (no ASC).

database	Canonical wide data frame.
C	Number of latent classes (C >= 1; C = 1 = MNL).
n_draws, n_draws_stage1, draws_type, estimation_routine, n_cores, quiet, mu_price_bounds, sigma_price_bounds	Tuning knobs; NULL means "use the current klue_mmnl_defaults()".
dgp	Output of klue_dgp.

See Also

klue, klue_database

`klue_simulate` *Simulate panel discrete-choice data*

Description

Generate synthetic panel choice data from a known latent-class DGP. The returned database element is already in canonical wide format and can be passed directly to klue for estimation. This is the data-generating process used in the $>$ 400-condition Monte Carlo study in Frings (2026, working paper).

Three flavours:

- klue_simulate – the standard random-attribute design.
- klue_simulate_cov – same design plus two concomitant covariates ($Z_1 \sim N(0,1)$, $Z_2 \sim Bernoulli(0.5)$) driving class membership; useful for testing covariate-aware LCMNL.
- klue_simulate_deff – D-efficient experimental design (lower observational noise per task than the random design).

Old names generate_data, generate_data_with_covariates and generate_data_defficient are deprecated aliases.

Usage

```
klue_simulate(N_per_class = 150, T_tasks = 20, true_K = 2,
              separation = 1.0, heterogeneity = 0.25,
              seed = 12345, class_proportions = NULL,
              dgp = DGP_DEFAULT, sep_profile = NULL)

klue_simulate_cov(N_per_class = 150, T_tasks = 20, true_K = 2,
                  separation = 1.0, heterogeneity = 0.25,
                  seed = 12345, covariate_strength = 1.0,
                  dgp = DGP_DEFAULT)

klue_simulate_deff(N_per_class = 150, T_tasks = 20, true_K = 2,
                   separation = 1.0, heterogeneity = 0.25,
                   seed = 12345, dgp = DGP_DEFAULT)
```

Arguments

N_per_class	Respondents per latent class. Total N = N_per_class * true_K (unless class_proportions is set).
T_tasks	Choice tasks per respondent.
true_K	Number of latent classes in the data-generating process.
separation	Inter-class distance multiplier (κ in the paper). Larger values make classes easier to recover.
heterogeneity	Within-class taste-variation s.d. (σ in the paper). Larger values blur the discrete-class structure.
seed	RNG seed for reproducibility.
class_proportions	Optional length-true_K vector of proportions summing to 1; default = balanced. (klue_simulate only.)
covariate_strength	Strength of the covariate-to-class mapping in klue_simulate_cov. 0 = covariates are noise; large values make class membership nearly deterministic in (Z_1, Z_2).
dgp	Design configuration from klue_dgp (number of generic attributes, alternatives, mean-beta vector).
sep_profile	Optional per-attribute weights for the separation vector; NULL = uniform across attributes.

Value

A list with elements:

database	Canonical wide data frame ready for klue: ID, TASK, x1_1..xN_J, price_1..price_J, CHOICE.
true_betas	true_K x n_beta matrix of class means.
true_class	Length-N integer vector of true class memberships.
individual_betas	N x n_beta matrix of person-level taste vectors.
N, T, K, dgp	Echoed dimensions and config.

klue_simulate_cov additionally returns the simulated covariates Z1, Z2.

See Also

klue, klue_dgp

Examples

```
## Not run:
# Generate 300 respondents from 2 classes, 20 tasks each, moderate separation
sim <- klue_simulate(N_per_class = 150, T_tasks = 20, true_K = 2,
                     separation = 1.0, heterogeneity = 0.25, seed = 42)
```

```
# Run the workflow on the simulated data
res <- klue(database = sim$database, C_cands = 1:4, run_mmnl = TRUE)
res$summary       # BIC should pick C = 2
res$best_C

## End(Not run)
```

`klue_study` *Replicate the Monte Carlo simulation study from Frings (2026)*

Description

The `klue_study*()` family reproduces the simulation study reported in Frings (2026, working paper). `klue_study()` is the umbrella driver that runs the 11 component analyses below in sequence; each component is also callable standalone for partial replication.

All results are written to `getOption("klue.output_dir", "output")` as CSV files (one per analysis) and returned as a named list.

Runtime warning: `klue_study()` takes ~12 hours on a modern laptop. Individual components range from ~5 min (`klue_study_unbalanced`, `klue_study_design`) to ~2-3 hours (`klue_study_main`, `klue_study_mmnl_corr`).

The `run_*` aliases (e.g. `run_full_study`, `run_main_simulation`) are silent aliases retained for backward compatibility with the paper's original simulation scripts.

Usage

```
klue_study(run_main = TRUE, run_mmnl = TRUE, run_supp = TRUE,
           verbose = TRUE, dgp = DGP_DEFAULT)

klue_study_main(true_K_values = c(1, 2, 3, 4, 5),
                kappa_values = c(0.5, 0.75, 1.0, 1.25, 1.5),
                sigma_values = c(0.1, 0.15, 0.2, 0.25),
                n_reps = 5, C_cands = 1:6,
                dgp = DGP_DEFAULT, sep_profile = NULL, verbose = TRUE)

klue_study_mmnl(n_cond = 80, n_draws = N_DRAWS_MMNL,
                dgp = DGP_DEFAULT, verbose = TRUE)

klue_study_convergence(n_random = 50, n_cond = 40,
                       verbose = TRUE, dgp = DGP_DEFAULT)

klue_study_initialisation(n_random = 50, n_cond = 40,
                          verbose = TRUE, dgp = DGP_DEFAULT)

klue_study_unbalanced(verbose = TRUE, dgp = DGP_DEFAULT)
klue_study_design(verbose = TRUE, dgp = DGP_DEFAULT)
```

```
klue_study_concomitant(verbose = TRUE, dgp = DGP_DEFAULT)
klue_study_recovery(n_cond = 80, verbose = TRUE, dgp = DGP_DEFAULT)
klue_study_clustering(verbose = TRUE, dgp = DGP_DEFAULT)
klue_study_sample(verbose = TRUE, dgp = DGP_DEFAULT)
klue_study_mmnl_corr(n_draws = N_DRAWS_MMNL, verbose = TRUE,
                     dgp = DGP_DEFAULT)
```

Arguments

run_main, run_mmnl, run_supp	Logical switches for which blocks of klue_study() to execute. run_supp controls the nine supplementary robustness analyses.
true_K_values, kappa_values, sigma_values, n_reps	Main simulation factor grid. Default is the 420-condition design of the paper: 5 true K's x 5 separations x 4 heterogeneities x 5 reps + the K=1 cell. n_reps replications per cell.
n_cond	Number of conditions sampled in the MMNL-comparison and recovery analyses.
n_draws	Number of simulation draws for MMNL fits.
n_random	Number of random starting-value restarts in the convergence / initialisation ablations.
C_cands	Class counts to evaluate per condition.
dgp	Design configuration (see klue_dgp). Default is 4 generic attributes + price, 3 alternatives.
sep_profile	Optional per-attribute separation weights; NULL = uniform.
verbose	Print progress lines.

Value

Each driver returns a data.frame (or list of data frames for klue_study()) with one row per condition / iteration. Columns vary by analysis but always include the factor levels that define the condition plus the BIC-selected K, log-likelihood, and any analysis-specific diagnostics.

Files written

Under getOption("klue.output_dir", "output"):

- main_results.csv
- mmnl_results.csv
- convergence_results.csv
- uninformed_convergence_results.csv
- unbalanced_results.csv
- design_results.csv
- concomitant_results.csv
- recovery_results.csv

- clustering_comparison_results.csv
- sample_sensitivity_results.csv
- mmnl_correlated_results.csv

See Also

klue, klue_simulate, klue_dgp

Examples

```
## Not run:
# Quickest individual driver: ~5 min on a 4-class unbalanced design
res <- klue_study_unbalanced()

# Run a custom slice of the main simulation: 2-class only, 3 reps
res <- klue_study_main(true_K_values = 2, n_reps = 3, C_cands = 1:4)

# Full paper replication (~12 hours)
all_results <- klue_study()

## End(Not run)
```

Index