Package: klue (via r-universe)

October 27, 2026

Type Package

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

Version 0.10.0

Author Oliver Frings [aut, cre]

Maintainer Oliver Frings <oliver.frings@ik.me>

Description Implements the three-step workflow of 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 models are estimated across a user-specified range of class counts, and a mixed multinomial logit (MMNL) benchmark, estimated with 'apollo', is reported alongside BIC, AIC, ICL and a classification-entropy diagnostic. For large samples the clustering starts are computed on a capped respondent subsample, and all starts are screened at a loose optimiser tolerance before the best is refined. Accepts long- or wide-format discrete-choice data with optional availability columns.

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)

Imports mclust, cluster, stats, utils, parallel

Suggests apollo (>= 0.3.0), idefix, mlogit, testthat

RoxygenNote 7.2.3

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

Date/Publication 2026-10-26 19:25:39 UTC

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

RemoteRef main

RemoteSha a41c20aa9de50e90963ec67a00f37711e7663670

RemoteSubdir klue

Contents

cleanup_apollo . . . 3
estimate_lcmnl . . . 3
estimate_lcmnl_cov . . . 4
estimate_lcmnl_em . . . 5
get_all_starts . . . 6
get_mnl_perturbation_starts . . . 7
get_random_partition_starts . . . 8
klue . . . 9
klue_database . . . 10
klue_database_long . . . 11
klue_database_wide . . . 12
klue_demo . . . 14
klue_derror . . . 14
klue_design . . . 15
klue_dgp . . . 16
klue_draw_betas . . . 17
klue_fieller . . . 17
klue_lcmnl . . . 18
klue_lincom . . . 19
klue_mean_fn . . . 20
klue_mmnl . . . 20
klue_mmnl_defaults . . . 23
klue_nlcom . . . 23
klue_par . . . 24
klue_predict . . . 24
klue_refit . . . 25
klue_simulate . . . 26
klue_simulate_choices . . . 27
klue_starts . . . 28
klue_tost . . . 29
klue_wald . . . 29
klue_wtp . . . 30
klue_wtp_fn . . . 31

Index **32**

cleanup_apollo *Remove Apollo's global objects*

Description

Deletes the `apollo_*` objects that Apollo requires in the global environment, so a session can mix klue and Apollo runs.

Usage

```
cleanup_apollo()
```

estimate_lcmnl *Direct-MLE LCMNL (MNL when C = 1)*

Description

BFGS on the full parameter vector with analytic gradients. Bypasses Apollo entirely: LCMNL involves only discrete mixing.

Usage

```
estimate_lcmnl(
  database,
  C,
  start_betas = NULL,
  start_shares = NULL,
  dgp = DGP_DEFAULT,
  maxit = MAX_ITER,
  reltol = 1e-10,
  start_par = NULL,
  vcov = TRUE
)
```

Arguments

`database`	Data frame in long format, sorted by respondent in blocks of `T_per_n` rows, with columns `ID`, `CHOICE`, and the attribute columns named by the DGP (`x*_j`, `price_j`). Optional availability columns `av_1..av_J` (1 = available, 0 = not) mask unavailable alternatives out of each task's choice set; if absent or all ones, every alternative is available (the default). The chosen alternative must be available in every task.
`C`	Integer number of latent classes. `C = 1` fits a plain MNL.
`start_betas`	Optional `C` x `n_beta` matrix of starting taste coefficients. If `NULL`, k-means starting values are computed via `klue_starts`.

start_shares	Optional length-C vector of starting class shares. If NULL, equal shares (1 / C) are used.
dgp	Data-generating-process specification list giving the design dimensions (n_alternatives, n_beta, n_generic, n_asc, npc). Defaults to DGP_DEFAULT.
maxit, reltol	BFGS iteration cap and relative tolerance. The defaults fit to full precision; the multistart screen passes looser values.
start_par	Optional full starting parameter vector (betas, ASCs, and share deltas as one vector), used to warm-start from a previous fit. Overrides start_betas/start_shares.
vcov	If TRUE (default), compute the parameter covariance at the optimum. Set FALSE to skip the Hessian on throw-away fits (the multistart uses this on its screening fits, attaching the covariance only to the winner).

Value

A list with the fit. converged (logical) flags a successful optimisation; C is the number of classes; LL the maximised log-likelihood; BIC, AIC, and ICL the corresponding information criteria (ICL_BIC is the entropy penalty ICL - BIC); k the number of free parameters; betas a C x n_beta matrix of estimated taste coefficients; class_probs the length-C class shares; posteriors the N x C matrix of posterior class-membership probabilities; and par the full parameter vector (reusable as start_par). When vcov = TRUE, vcov (inverse observed information) and robust_vcov (sandwich estimator clustered on respondent) give the covariance of par for any C; if the observed information is rank-deficient both are set to NA, singular is TRUE and aliased names the parameters that are not separately identified (estimates, LL and class assignment are unaffected, but k still counts every parameter, so the information criteria are penalised for parameters the design cannot identify). For C = 1 the parameters are labelled b1..b{n_beta} (taste coefficients, price last) and asc1..asc{n_asc}; for C >= 2 each class c contributes b1_class{c}..b{n_beta}_class{c} and asc1_class{c}..asc{n_asc}_class{c}, followed by the C - 1 share parameters delta1..delta{C-1} (last class is the reference). model_type is "MNL" when C = 1 and "LCMNL" otherwise. A fit that hits maxit returns its (finite) values with converged = FALSE; an optimiser error returns the failure skeleton with LL = -Inf.

---

estimate_lcmnl_cov *Concomitant LCMNL: covariate-driven class membership*

Description

Direct-MLE LCMNL in which the class-membership probabilities depend on respondent-level covariates, pi_{nc} = softmax_c(t(Z_n) gamma_c) with the last class as reference (gamma_C = 0). The class-specific taste model is identical to estimate_lcmnl; only the share model changes. With no covariates this reduces exactly to the constant-share LCMNL.

Usage

```
estimate_lcmnl_cov(
  database,
  C,
```

```
  membership = character(0),
  start_betas = NULL,
  start_shares = NULL,
  dgp = DGP_DEFAULT,
  maxit = MAX_ITER,
  reltol = 1e-10,
  start_par = NULL,
  vcov = TRUE
)
```

Arguments

`database`	Long-format choice data (as for `estimate_lcmnl`), plus the membership covariate columns, which must be constant within respondent.
`C`	Integer number of latent classes.
`membership`	Character vector of covariate column names entering the class-membership model (an intercept is always included). Default none (recovers the constant-share model).
`start_betas, start_shares`	Optional taste and share starting values; if `NULL`, k-means starts are computed. Membership slopes start at zero.
`dgp, maxit, reltol, start_par, vcov`	As in `estimate_lcmnl`.

Value

A list like `estimate_lcmnl` with, additionally, `gamma` (a `p x (C-1)` matrix of membership coefficients, rows named by `Z`'s columns) and `shares` the `N x C` matrix of respondent-level class probabilities (`class_probs` is their mean). `vcov`/`robust_vcov` cover taste params then the membership parameters `{cov}_class{c}`.

See Also

`estimate_lcmnl`

`estimate_lcmnl_em` *EM estimator for the same LCMNL likelihood*

Description

E-step: posterior class weights; M-step: shares + per-class weighted MNL (warm-started at the class's current parameters). From identical starts, EM and direct ML should reach the same optimum. C = 1 delegates to `estimate_lcmnl` so the two estimators agree exactly there.

Usage

```
estimate_lcmnl_em(
  database,
  C,
  start_betas = NULL,
  start_shares = NULL,
  dgp = DGP_DEFAULT,
  max_em_iter = 500L,
  tol = 1e-06,
  verbose = FALSE
)
```

Arguments

database	Data frame in long format, sorted by respondent in blocks of T_per_n rows, with columns ID, CHOICE, and the attribute columns named by the DGP (x*_j, price_j).
C	Integer number of latent classes. C = 1 delegates to estimate_lcmnl.
start_betas	Optional C x n_beta matrix of starting taste coefficients. If NULL, k-means starting values are computed via klue_starts.
start_shares	Optional length-C vector of starting class shares. If NULL, equal shares (1 / C) are used.
dgp	Data-generating-process specification list giving the design dimensions (n_alternatives, n_beta, n_generic, n_asc, npc). Defaults to DGP_DEFAULT.
max_em_iter	Maximum number of EM iterations. Default 500.
tol	Convergence tolerance on the change in log-likelihood between successive EM iterations. Default 1e-6.
verbose	If TRUE, print the log-likelihood at each iteration. Default FALSE.

Value

A list with the fit. converged (logical) flags a successful run; C is the number of classes; LL the maximised log-likelihood; BIC, AIC, and ICL the corresponding information criteria (ICL_BIC is the entropy penalty ICL - BIC); k the number of free parameters; betas a C x n_beta matrix of estimated taste coefficients; class_probs the length-C class shares; posteriors the N x C matrix of posterior class-membership probabilities; em_iters the number of EM iterations run; and estimator the string "em". model_type is "MNL" when C = 1 and "LCMNL" otherwise. A failed fit returns the same fields with converged set to FALSE.

---

get_all_starts *All clustering-based starting values*

Description

Starting taste coefficients and class shares for C classes from every clustering method in KLUE_CLUSTER_METHODS, as used by the multistart in klue_lcmnl.

Usage

```
get_all_starts(
  database,
  C,
  dgp = DGP_DEFAULT,
  feature_type = c("rp", "onehot"),
  cluster_cap = 2000L,
  screen = getOption("klue.screen", TRUE)
)
```

Arguments

database	Canonical wide database.
C	Number of latent classes.
dgp	DGP specification from klue_dgp.
feature_type	Feature space for clustering: "rp" (revealed-preference signatures) or "onehot".
cluster_cap	Maximum respondents clustered (a fixed-seed subsample above it).
screen	See klue_starts.

Value

A list of starting values: betas (C x n_beta) and shares, or a list of such lists.

---

get_mnl_perturbation_starts

Starting values by perturbing the pooled MNL

Description

Random starts around the pooled-MNL solution, used in the initialisation ablations.

Usage

```
get_mnl_perturbation_starts(
  database,
  C,
  n_starts = 50L,
  jitter_sd = 0.5,
  seed = 1L,
  dgp = DGP_DEFAULT
)
```

Arguments

database	Canonical wide database.
C	Number of latent classes.
n_starts	Number of random starts.
jitter_sd	See klue_starts.
seed	Integer seed.
dgp	DGP specification from klue_dgp.

Value

A list of starting values: betas (C x n_beta) and shares, or a list of such lists.

---

get_random_partition_starts

Starting values from random partitions

Description

Class-specific MNL fits on random respondent partitions, used in the initialisation ablations.

Usage

```
get_random_partition_starts(
  database,
  C,
  n_starts = 50L,
  seed = 1L,
  dgp = DGP_DEFAULT
)
```

Arguments

database	Canonical wide database.
C	Number of latent classes.
n_starts	Number of random partitions.
seed	Integer seed.
dgp	DGP specification from klue_dgp.

Value

A list of starting values: betas (C x n_beta) and shares, or a list of such lists.

klue *Run the full LCMNL specification workflow*

Description

Three calling conventions: (1) ‘database =‘ canonical wide format; (2) ‘data =‘ + long-format column mapping; (3) ‘data =‘ + wide-format column mapping (both forwarded to klue_database). Loops over ‘C_cands‘, optionally estimates the MMNL benchmarks, prints and writes summary CSVs, and returns the full results list invisibly.

Usage

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

Arguments

database	canonical wide-format data frame with columns ID, TASK, CHOICE plus attribute columns. Supply this or ‘data‘, not both.
data	raw data frame passed to klue_database together with the column-mapping arguments in ‘...‘ when ‘database‘ is not supplied.
format	one of "auto", "long", "wide": the layout of ‘data‘, forwarded to klue_database. Default "auto".
C_cands	integer vector of class counts to estimate. Default 1:6.
asc_map	optional integer vector of length ‘n_alternatives‘ mapping each alternative to an ASC group (0 = reference / no ASC; alternatives sharing a group share one ASC), forwarded to klue_dgp; see there. Default NULL (the klue_dgp default).
run_mmnl	logical; if TRUE estimate the independent-normals MMNL benchmark. Default TRUE.
run_mmnl_corr	logical; if TRUE estimate the correlated-normals MMNL benchmark. Default FALSE.

mmnl_opts	named list of extra arguments forwarded to klue_mmnl for both MMNL flavours. Default an empty list.
attr_labels	character vector of length ‘n_generic + 1‘ naming the attribute coefficients (the last entry is price). If NULL, taken from the database attribute or generated as x1, x2, ..., price.
output_prefix	character prefix for the written CSV file names. Default "workflow".
output_dir	directory for the written CSVs. If NULL, uses OUTPUT_DIR.
write_csv	logical; if TRUE write the summary, class-coefficient and model-comparison CSVs. Default TRUE.
verbose	logical; if TRUE print progress and result tables. Default TRUE.
n_cores	cores for the per-start LCMNL fits within each C (forwarded to klue_lcmnl). Default 1 (sequential). Setting it to the number of physical cores fits the six clustering starts concurrently and is the main speed lever for a single dataset (~3x on a typical multicore machine).
...	column-mapping and scaling arguments forwarded to klue_database when ‘data‘ is supplied instead of ‘database‘.

Value

Invisibly, a list with elements ‘database‘, ‘dgp‘, ‘lcmnl‘ (the per-C fits), ‘mmnl‘, ‘mmnl_corr‘, ‘summary‘, ‘class_betas‘, ‘comparison‘, ‘best_C‘ (the BIC-best class count) and ‘best_lcmnl‘ (the BIC-best fit).

---

klue_database *Build the canonical database (format dispatcher)*

Description

‘format = "auto"‘ infers: named-list ‘attribute_cols‘ -> wide; ‘alt_col‘ present -> long.

Usage

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

Arguments

data	A data.frame, or a single file path to a ‘.dat‘ (whitespace-delimited, header) or ‘.csv‘ file to read.
format	One of "auto", "long", or "wide". With "auto", a named-list ‘attribute_cols‘ selects the wide builder and an ‘alt_col‘ argument selects the long builder.
...	Further arguments passed to [klue_database_long()] or [klue_database_wide()] according to the resolved format.

Value

A data.frame in the canonical wide format with columns ‘ID‘, ‘TASK‘, the generic attribute blocks ‘x1_1..xN_J‘, the price columns ‘price_1..price_J‘, and ‘CHOICE‘ (the chosen alternative index in ‘1..J‘). The last alternative is the reference. Attributes ‘attr_labels‘, ‘n_alternatives‘, and ‘n_generic‘ are attached.

---

klue_database_long *Build the canonical database from long-format data*

Description

One row per (respondent, task, alternative). ‘choice_format‘ is "indicator" (0/1 on the chosen row) or "alt_index" (chosen alt repeated on every row of the task). ‘scalings‘ is a named list of divisors keyed by attribute/price column name; ‘price_scaling‘ is a shortcut for the price column. Tasks where any alternative is unavailable (avail_col == 0) and respondents with fewer tasks than the panel maximum are dropped.

Usage

```
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,
  keep_unavailable = FALSE,
  scalings = NULL,
  price_scaling = 1,
  covariate_cols = NULL,
  verbose = TRUE
)
```

Arguments

data	A data.frame, or a single file path to a ‘.dat‘ (whitespace-delimited, header) or ‘.csv‘ file to read.
id_col	Name of the respondent identifier column.
task_col	Name of the choice-task identifier column (unique within respondent).
alt_col	Name of the alternative identifier column within a task.
choice_col	Name of the choice column, interpreted per ‘choice_format‘.

`attribute_cols` Character vector of generic attribute column names; each becomes the ‘x*_*‘ block in the order supplied.

`price_col` Length-one name of the price column.

`choice_format` Either "indicator" (0/1, one chosen row per task) or "alt_index" (the chosen alternative index repeated on every row of the task).

`avail_col` Optional name of an availability column (0/1). By default, tasks where not all alternatives are available are dropped.

`keep_unavailable`
If ‘TRUE‘ (with ‘avail_col‘ supplied), keep those tasks and emit per-alternative ‘av_1..av_J‘ columns instead of dropping them; the estimator then masks unavailable alternatives out of each task’s choice set. Default ‘FALSE‘ (drop, the historical behaviour).

`scalings` Optional named list of positive divisors keyed by attribute or price column name; matching columns are divided before use.

`price_scaling` Positive divisor applied to the price column; shortcut for adding the price column to ‘scalings‘.

`covariate_cols` Optional character vector of respondent-level columns (constant within respondent, e.g. class-membership covariates for `estimate_lcmnl_cov`) copied onto the database, one value per (respondent, task) row.

`verbose` Logical; if ‘TRUE‘, print drop counts and a build summary.

Value

A data.frame in the canonical wide format with columns ‘ID‘, ‘TASK‘, the generic attribute blocks ‘x1_1..xN_J‘, the price columns ‘price_1..price_J‘, ‘CHOICE‘ (the chosen alternative index in ‘1..J‘) and any ‘covariate_cols‘. Respondents are numbered in the sorted order of ‘id_col‘ (attribute ‘ids‘ maps ‘ID = k‘ back to the k-th sorted id) and tasks in the sorted order of ‘task_col‘ within respondent. The last alternative is the reference. Attributes ‘attr_labels‘ (attribute and price labels, suffixed by scaling where applicable), ‘n_alternatives‘ (J), and ‘n_generic‘ (number of generic attributes) are attached.

`klue_database_wide` *Build the canonical database from wide-format data*

Description

One row per (respondent, task). ‘attribute_cols‘ is a named list; each entry is a length-J character vector of column names (NA = structural zero on that alternative). ‘price_col‘ is a length-J vector; ‘avail_col‘ optionally too. ‘scalings‘ is keyed by attribute name or "price".

Usage

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

Arguments

data	A data.frame, or a single file path to a '.dat' (whitespace-delimited, header) or '.csv' file to read.
id_col	Name of the respondent identifier column.
task_col	Optional name of the choice-task identifier column; if 'NULL', tasks are numbered sequentially within each respondent.
choice_col	Name of the column holding the chosen alternative index in '1..J'.
attribute_cols	Named list of length-J character vectors of column names, one vector per generic attribute; 'NA' marks a structural zero for that alternative. The list element names become attribute labels.
price_col	Length-J character vector of per-alternative price column names.
avail_col	Optional length-J character vector of per-alternative availability column names (0/1). By default, tasks where not all alternatives are available are dropped.
keep_unavailable	If 'TRUE' (with 'avail_col' supplied), keep those tasks and emit per-alternative 'av_1..av_J' columns instead of dropping them; the estimator then masks unavailable alternatives out of each task's choice set. Default 'FALSE' (drop, the historical behaviour).
scalings	Optional named list of positive divisors keyed by attribute name or "price"; matching columns are divided before use.
attributes	Deprecated alias for 'attribute_cols'.
price	Deprecated alias for 'price_col'.
availability	Deprecated alias for 'avail_col'.
verbose	Logical; if 'TRUE', print drop counts and a build summary.

Value

A data.frame in the canonical wide format with columns 'ID', 'TASK', the generic attribute blocks 'x1_1..xN_J', the price columns 'price_1..price_J', and 'CHOICE' (the chosen alternative index in '1..J'). The last alternative is the reference. Attributes 'attr_labels' (attribute and price labels, suffixed by scaling where applicable), 'n_alternatives' (J), and 'n_generic' (number of generic attributes) are attached.

---

klue_demo *Zero-setup demonstration on Apollo's Swiss route-choice data*

Description

'full = FALSE' (default): C = 1..2, no MMNL (~3 sec). 'full = TRUE': C = 1..6 + MMNL benchmark with 500 MLHS draws (a few minutes on one core).

Usage

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

Arguments

full	logical; if TRUE estimate C = 1..6 with the MMNL benchmark, otherwise C = 1..2 without MMNL. Default FALSE.
verbose	logical; if TRUE print progress and result tables. Default TRUE.

Value

Invisibly, the list returned by klue for the Swiss route-choice data.

---

klue_derror *D-error of a choice design*

Description

MNL D-error $\det(I(\beta))^{-1/K}$ of the task rows of a canonical database (one row per choice task; ID and CHOICE are not used), with the constants among the K = npc parameters (Ngene's ;con). The information sums over the rows, so the value matches Ngene's D-error for one respondent answering every row.

Usage

```
klue_derror(database, dgp, par = NULL, draws = NULL, tol = 1e-10)
```

Arguments

database	Canonical database of the design's tasks.
dgp	Its dgp (klue_dgp).
par	Optional prior vector (taste coefficients, then constants); default zeros.
draws	Optional matrix of prior draws (one per row); the D-error is then their mean (Bayesian D-error).
tol	Relative eigenvalue tolerance of the rank.

Value

A list: derror, rank and full_rank of the information at the first prior, and K.

---

klue_design *Blocked D-efficient design from a population-mean prior*

Description

One locally-D-efficient design is constructed for the whole study (the analyst designs around population means, NOT the unknown latent classes), then split into blocks. Each respondent answers one block, so all respondents in a block face identical cards. This is the realistic-DCE baseline. Requires the idefix package.

Usage

```
klue_design(
  n_cards = 48L,
  n_blocks = 4L,
  dgp = DGP_DEFAULT,
  priors = NULL,
  n_lvls = 4L,
  n_start = 8L,
  seed = 20240601L
)
```

Arguments

n_cards	Total number of choice cards (sets) in the design. Must be a multiple of n_blocks.
n_blocks	Number of blocks the cards are split into. Each respondent answers one block.
dgp	Data-generating-process specification supplying n_alternatives, n_beta, n_generic, and beta_bar.
priors	Prior coefficients: a numeric vector gives a locally D-efficient design at that point; a matrix (rows = prior draws, columns = coefficients) gives a Bayesian D-efficient design with the D-error averaged over the draws. Defaults to dgp$beta_bar when NULL.
n_lvls	Number of attribute levels per attribute.
n_start	Number of random starting designs for the Modfed search.
seed	Integer seed for reproducible design construction and blocking.

Value

A list with elements `cards` (an `n_cards` by `n_alternatives` by `n_beta` array of attribute values), `blocks` (a list mapping each block to its card indices), `T` (cards per block), `n_cards`, `n_blocks`, `dgp`, and `Derror` (the D-error of the chosen design: Bayesian when `priors` is a matrix of draws, local otherwise).

`klue_dgp` *Data-generating-process configuration*

Description

Builds the problem-dimension specification shared by the estimators, simulators, and study drivers, so they work with a variable number of generic attributes and alternatives.

Usage

```
klue_dgp(
  n_generic = 4,
  n_alternatives = 3,
  asc_map = NULL,
  labels = NULL,
  asc_labels = NULL,
  common = NULL
)
```

Arguments

`n_generic`	number of generic attributes (the price attribute is added on top of these).
`n_alternatives`	number of alternatives per choice task; the last alternative is the reference (no ASC).
`asc_map`	optional integer vector of length `n_alternatives` mapping each alternative to an ASC group (0 = reference / no ASC; group ids must be contiguous 1..K). Alternatives sharing a group share one ASC, e.g. `c(1, 1, 2, 0)` gives two offering tiles a common ASC, the third alternative its own, and the fourth as reference. Default `NULL` reproduces the per-alternative structure `c(1, .., n_alt-1, 0)`.
`labels`	optional names of the `n_generic + 1` taste coefficients (price last). Fits then name their parameters `<label>` (MNL) or `<label>_class<c>` (latent class). Default `NULL` keeps `b1..b{n_beta}`.
`asc_labels`	optional names of the `n_asc` constants. Default `NULL` keeps `asc1..asc{n_asc}`.
`common`	optional coefficients shared by all latent classes, as labels or as positions in the per-class vector (taste coefficients, then constants). A latent class fit then estimates one value for each, and its `k` falls by `C - 1` per common coefficient. Default none.

Value

A named list with `n_generic`, `n_alternatives`, `n_beta` (generic + price), `n_asc` (number of ASC groups), `npc` (parameters per class), `asc_map`, `beta_bar` (population-mean coefficients), `attr_names`, `price_idx`, `par_labels` (the per-class parameter names) and `common` (positions of the common coefficients).

`klue_draw_betas` *Draw respondent coefficients from a fitted model*

Description

Returns an `N` x `npc` matrix of coefficients (taste coefficients, then constants, named by label) for `klue_simulate_choices`: the MNL vector repeated, or for a latent class fit each row the coefficients of a class drawn with the class shares (common coefficients share their one value). Uses the caller's RNG; the drawn classes are in `attr(, "class_draw")`.

Usage

```
klue_draw_betas(fit, N)
```

Arguments

`fit`	An MNL or constant-share LCMNL klue fit.
`N`	Number of respondents.

`klue_fieller` *Fieller confidence interval for a ratio of linear combinations*

Description

95%-by-default CI for $(a'\beta)/(b'\beta)$. Returns `NA` bounds when the denominator is not significantly different from zero (unbounded CI).

Usage

```
klue_fieller(fit, num, den, level = 0.95, robust = TRUE)
```

Arguments

`fit`	A klue fit.
`num`, `den`	Named weight vectors for the numerator and denominator, or functions of the parameter vector (a generalised Fieller interval from their delta-method covariance), e.g. from `klue_mean_fn`.
`level`	Confidence level. Default `0.95`.
`robust`	Use the cluster-robust covariance (default `TRUE`).

Value

Named vector: `ratio`, `lo`, `hi`.

`klue_lcmnl` *Multi-start LCMNL: best of the six clustering initialisations*

Description

For C = 1 (MNL) a single run suffices; for C >= 2 the model is estimated from all six clustering starts and the best log-likelihood wins. With the `"ml"` estimator the starts are screened at a loose tolerance (`maxit = 200`, `reltol = 1e-6`; fits hitting the cap still count) and only the winner is polished to full precision, warm-started from its own solution. Set `screen = FALSE` (or session-wide `options(klue.screen = FALSE)`) to fit every start at full precision with no polish step – the pre-0.9.1 code path, kept for exact reproduction of results produced with it. Each start logs its LL and elapsed time via `message()`.

Usage

```
klue_lcmnl(
  database,
  C,
  dgp = DGP_DEFAULT,
  estimator = c("ml", "em"),
  feature_type = c("rp", "onehot"),
  n_cores = 1L,
  screen = getOption("klue.screen", TRUE),
  clean = FALSE,
  maxit = MAX_ITER,
  reltol = 1e-10,
  cluster_cap = 2000L
)
```

Arguments

`database`	Data frame in long format, sorted by respondent in blocks of `T_per_n` rows, with columns `ID`, `CHOICE`, and the attribute columns named by the DGP (`x*_j`, `price_j`). Optional availability columns `av_1..av_J` (1 = available, 0 = not) mask unavailable alternatives out of each task's choice set; if absent or all ones, every alternative is available (the default). The chosen alternative must be available in every task.
`C`	Integer number of latent classes. `C = 1` fits a plain MNL.
`dgp`	Data-generating-process specification list giving the design dimensions (`n_alternatives`, `n_beta`, `n_generic`, `n_asc`, `npc`). Defaults to `DGP_DEFAULT`.
`estimator`	"ml" (direct BFGS, default) or "em".
`feature_type`	"rp" (default) or "onehot" clustering features.

n_cores	number of cores for the six per-start fits. Default 1 (sequential). With n_cores > 1 the starts are fit concurrently via parallel::mclapply (fork-based; not available on Windows). The best-of-six selection is order-deterministic and so is independent of n_cores. Leave at 1 inside the study drivers, which already parallelise across conditions – nesting would oversubscribe the cores.
screen	logical; the 0.9.1 fast-screening behaviour (default). With the "ml" estimator the six starts are screened at the loose tolerance and only the winner is polished; the cluster-wise MNL start fits also run loose (see klue_starts). FALSE restores the pre-0.9.1 path end to end: tight cluster-MNL start fits, every start estimated at full precision, no polish. The default reads getOption("klue.screen", TRUE), so options(klue.screen = FALSE) restores the old path in every caller (klue(), the study drivers, and the start generators) without changing call sites.
clean	logical; fit every start at full precision with its covariance and return the highest-LL start whose fit has not failed (no polish step). If none qualifies, the highest-LL start is returned with failed = TRUE. The fit then also reports n_same_optimum, the number of starts within 1e-4 of its LL. Default FALSE keeps the 0.10.0 path. Needs the "ml" estimator.
maxit, reltol	BFGS settings of the full-precision fits.
cluster_cap	maximum respondents clustered for the starts (see get_all_starts).

Value

The best-fitting per-start result, a list with the same fields as the chosen estimator (estimate_lcmnl for "ml", estimate_lcmnl_em for "em"): converged, C, LL, BIC, AIC, ICL (and ICL_BIC), k, betas, class_probs, posteriors, and for the EM estimator em_iters and estimator. Two extra fields are added: best_method, the name of the winning clustering start, and method_results, a named list of the individual fits from every start. If no start converges, the failure result is returned with converged = FALSE and best_method = NA.

---

klue_lincom *Linear combination of coefficients from a klue fit*

Description

Linear combination of coefficients from a klue fit

Usage

```
klue_lincom(fit, weights, robust = TRUE)
```

Arguments

fit	A klue fit (from estimate_lcmnl / estimate_lcmnl_cov) carrying par and a covariance.
weights	Named numeric vector c(param = weight, ...) (or a character vector of parameter names for unit weights) defining $c'\beta$.
robust	Use the cluster-robust covariance (default TRUE).

Value

Named vector: `estimate`, `se`, `z`, two-sided `p`.

`klue_mean_fn` *Population mean of a coefficient as a function of the parameters*

Description

Returns `function(par)` giving the share-weighted mean of a labelled coefficient over the latent classes, $\sum_c \pi_c \beta_c$ (the coefficient itself in an MNL). Pass it to `klue_nlcom`, or as numerator and denominator to `klue_fieller` for a ratio of means.

Usage

```
klue_mean_fn(fit, label)
```

Arguments

`fit`	An MNL or constant-share LCMNL klue fit.
`label`	Coefficient label.

`klue_mmnl` *MMNL benchmark (independent or correlated normals)*

Description

Estimates a mixed logit with Apollo the way Apollo's own example scripts do: random coefficients $\mu + \sigma\xi$ with σ unconstrained, a negative log-normal price, 'apollo_estimate' at Apollo's default estimation settings, and convergence as Apollo reports it. The departures from the examples (pooled-MNL starts, zero starting correlations, 3000 MLHS draws, BIC on respondents, and utilities measured from the chosen alternative on data with availability columns) are listed in the source file's header. By default every generic attribute is a random normal, the price a negative log-normal, and the constants fixed; 'random' and 'price' change that. Correlated flavour ('correlation = TRUE'): a full lower-triangular Cholesky covariance over the random parameters, started from the independent estimates (Apollo's 'apollo_readBeta' step: the same means, the independent standard deviations on the diagonal, zero correlations).

Usage

```
klue_mmnl(
  database,
  correlation = FALSE,
  n_draws = NULL,
  draws_type = NULL,
  estimation_routine = NULL,
  n_cores = NULL,
  quiet = NULL,
  start = c("informed", "zero", "sign"),
  random = NULL,
  price = c("lognormal", "normal", "fixed"),
  warm_start = NULL,
  dgp = DGP_DEFAULT,
  memory_saver = NULL,
  n_draws_stage1 = NULL,
  mu_price_bounds = NULL,
  sigma_price_bounds = NULL
)

klue_mmnl_corr(database, ...)
```

Arguments

database	Data frame in klue's canonical wide format (one row per task) with an 'ID' column and the attribute and choice columns expected by the data-generating process.
correlation	Logical; if 'TRUE', estimate the correlated specification with a full lower-triangular Cholesky covariance instead of independent normals.
n_draws	Number of inter-individual draws. 'NULL' uses the package default.
draws_type	Apollo inter-draws type (for example Halton or MLHS). 'NULL' uses the package default.
estimation_routine	Apollo estimation routine passed to 'apollo_estimate'. 'NULL' uses the package default (Apollo's default, BGW).
n_cores	Number of cores for Apollo. 'NULL' uses the package default (Apollo's default, 1).
quiet	Logical; if 'TRUE', redirect Apollo output to a temporary log file whose tail is attached to a failing result. 'NULL' uses the package default.
start	Starting-value scheme. '"informed"' (default) starts the independent model from the pooled MNL estimates (means, fixed coefficients and constants; a log-normal price at $\mu = \log(-\beta_{MNL})$, or at Apollo's recommended -3 when the pooled price coefficient is not negative) and the correlated model from an independent fit with the same draws. '"zero"' and '"sign"' are the conventional neutral starts: attribute means at '0' and at '+0.01' respectively, with a log-normal price coefficient started at '-0.01' in both (a log-normal cannot start at exactly zero, and

'mu_price = 0' would mean 'b_price = -1', not a neutral start). Standard deviations start at '0.01' under every scheme, as in Apollo's examples. Use a neutral start to show that a fit does not depend on being warm-started. Ignored (with a warning) when 'warm_start' is given.

`random` Character vector naming the random parameters among 'x1..xN', 'price' and 'asc1..ascK', where 'ascK' indexes the ASC groups of 'dgp$asc_map' (with the default map, one per non-reference alternative). 'NULL' (default) makes every generic attribute and the price random and the constants fixed. Parameters not named are fixed coefficients. Random constants are normal, with the reference alternative's constant fixed at zero, so an independent random-constant model depends on which alternative is the reference.

`price` Distribution of a random price coefficient: '"lognormal"' (default; negative log-normal), '"normal"', or '"fixed"' (the price is not random, whatever 'random' says by default).

`warm_start` Optional earlier 'klue_mmnl' fit (klue >= 0.10.0) of the same data, 'random' set and 'price' distribution to start from. An independent fit given to a correlated call is mapped to the point where the two models coincide (same means, its standard deviations on the Cholesky diagonal, zero correlations), so with the same draws the correlated fit cannot end below it; the result's 'nested_ok' reports whether it did not. A fit of the same flavour is copied by name.

`dgp` Data-generating-process specification giving the number of alternatives, generic attributes, parameters and the ASC groups.

`memory_saver` Logical; Apollo's 'apollo_control$memorySaver'. 'TRUE' computes the analytic gradient in chunks of about two respondents, which cuts its memory (about one observations-by-draws array per parameter) without changing the estimates. 'NULL' uses the package default (Apollo's default, 'FALSE'; set 'options(klue.mmnl.memory_saver = TRUE)' or the environment variable 'KLUE_MMNL_MEMORY_SAV to change it).

`n_draws_stage1`, `mu_price_bounds`, `sigma_price_bounds`
Deprecated and ignored (with a warning). Since klue 0.10.0 there is no coarse-draw first stage, and Apollo 0.3.5 has no 'bounds' setting, so the price box of earlier versions was never applied.

`...` For `klue_mmnl_corr`, arguments passed to `klue_mmnl`.

Value

A list describing the fit. 'converged' is Apollo's 'successfulEstimation'. On success: the log-likelihood 'LL' (Apollo's final log-likelihood), information criteria 'BIC' and 'AIC' (with 'N' = respondents, as for the LCMNL; Apollo's own BIC, on observations, is 'BIC_apollo'), the number of free parameters 'k', 'reason = "ok"', the resolved 'settings', Apollo's status ('apollo_status': 'successfulEstimation', 'code', 'message', 'nIter'), the starting values 'start_values' and their source 'start_used', the full estimate vector 'par' with its classical and respondent-clustered covariances 'vcov' and 'robust_vcov' (Apollo's 'varcov' and 'robvarcov'; all 'NA' when Apollo could not invert the Hessian), the resolved 'random' set and 'price' distribution, 'sd_scale = "linear"', and, for the independent specification, the means 'mu' and standard deviations 'sigma = |sigma_p|' of the random parameters (for a log-normal price, of the underlying normal). A correlated fit that fitted its own independent start also returns it as 'independent_fit'. On failure: 'converged = FALSE' with

'LL', 'BIC', 'AIC', and 'k' set to non-informative values, a 'reason' string, Apollo's status and the estimates where Apollo returned them ('LL_at_stop', 'par'), the 'independent_fit' where there was one, and the Apollo log tail and path when 'quiet' is 'TRUE'.

Side effects: during estimation the Apollo objects live in the global environment, as Apollo requires; any 'apollo_*' objects the caller had there are restored afterwards, and so is the random-number stream, which Apollo reseeds when it makes its draws.

`klue_mmnl_defaults` *MMNL default settings*

Description

Returns the active MMNL defaults as a named list. Values can be overridden per-call (arguments to `klue_mmnl` / `klue`) or globally via `options()` entries `klue.mmnl.<name>`.

Usage

```
klue_mmnl_defaults()
```

Value

A named list of the currently active defaults.

`klue_nlcom` *Delta-method inference on a function of a klue fit's parameters*

Description

Delta-method inference on a function of a klue fit's parameters

Usage

```
klue_nlcom(fit, fn, robust = TRUE, level = 0.95)
```

Arguments

`fit`	A klue fit carrying `par` and a covariance.
`fn`	Function of the full parameter vector returning one number, e.g. from `klue_wtp_fn` or `klue_mean_fn`.
`robust`	Use the cluster-robust covariance (default `TRUE`).
`level`	Confidence level of `lo` and `hi`. Default `0.95`.

Value

Named vector: `estimate`, `se`, `z`, two-sided `p`, `lo`, `hi`. The gradient is a central difference with step `1e-6 * max(|par_i|, 1)`.

klue_par *Parameter name of a labelled coefficient in a klue fit*

Description

Resolves a coefficient label (see the `labels` of `klue_dgp`) to the name it carries in `fit$par`: the label itself in an MNL, and `<label>_class<c>` in a latent class fit. A coefficient common to all classes resolves to its class-1 entry when `class` is not given (every class holds the same value). Use the result as weight names in `klue_lincom`, `klue_wtp` and the other tests.

Usage

```
klue_par(fit, label, class = NULL)
```

Arguments

`fit`	A klue fit.
`label`	Coefficient label.
`class`	Optional class number(s) for a latent class fit.

Value

Character vector of parameter names.

klue_predict *Expected choice probabilities from a fitted model*

Description

Returns the `n_obs` x `J` matrix of choice probabilities of every task row of `database`: the MNL probabilities, or for a latent class fit their mixture over the classes with the class shares.

Usage

```
klue_predict(fit, database, dgp)
```

Arguments

`fit`	An MNL or constant-share LCMNL klue fit.
`database`	Canonical database (`CHOICE` is not used).
`dgp`	The dgp of the fit.

`klue_refit` *Re-estimate a fitted MNL or latent class model*

Description

Fits the same model class to `database`, starting from the fit's estimates matched by label: an MNL, or a constant-share LCMNL at `fit$C` (with the common coefficients of `dgp`), or with `membership` the concomitant model at `fit$C`. Labels new to `dgp` start at 0, labels it drops are ignored, and `start` overrides any start value. A common coefficient starts at the mean of its class values.

Usage

```
klue_refit(
  fit,
  database,
  dgp,
  start = NULL,
  multistart = FALSE,
  membership = NULL,
  maxit = MAX_ITER,
  reltol = 1e-10,
  n_cores = 1L
)
```

Arguments

`fit`	An MNL or constant-share LCMNL klue fit.
`database`	Canonical database.
`dgp`	The dgp of the model to fit (`klue_dgp` with labels).
`start`	Optional named start values: full parameter names, or labels (set in every class).
`multistart`	Latent class only: also run `klue_lcmnl(clean = TRUE)` and keep it if it has not failed and has the higher log-likelihood.
`membership`	Optional respondent-level covariate columns: fit the concomitant model (`estimate_lcmnl_cov`); the intercepts start at the fit's share parameters and the slopes at 0.
`maxit, reltol`	BFGS settings.
`n_cores`	Cores of the multistart.

Value

The fit, with `failed`.

`klue_simulate` *Simulate panel choice data from the Frings (2026) DGP*

Description

One entry point for the plain DGP and the concomitant-covariate DGP (`covariates = TRUE`; class membership driven by Z1 ~ N(0,1) and a binary Z2). With `design` from `klue_design()`, respondents share their block's cards; otherwise attributes are drawn fresh per row, optionally correlated via `attr_corr`.

Usage

```
klue_simulate(
  N_per_class = 150,
  T_tasks = 20,
  true_K = 2,
  separation = 1,
  heterogeneity = 0.25,
  seed = 12345,
  class_proportions = NULL,
  dgp = DGP_DEFAULT,
  sep_profile = NULL,
  attr_corr = NULL,
  design = NULL,
  seg_scale = NULL,
  covariates = FALSE,
  covariate_strength = 1
)
```

Arguments

`N_per_class`	Number of respondents per latent class.
`T_tasks`	Number of choice tasks per respondent. Overridden by the block size when `design` is supplied.
`true_K`	Number of latent classes.
`separation`	Scalar controlling the distance of class means from the grand mean `beta_bar`.
`heterogeneity`	Within-class standard deviation of the individual coefficients.
`seed`	Integer seed for reproducible simulation.
`class_proportions`	Optional numeric vector of class shares summing to one. When `NULL`, classes are equally sized. Ignored when `covariates = TRUE`.
`dgp`	Data-generating-process specification supplying `n_beta`, `beta_bar`, `n_alternatives`, and `n_generic`.
`sep_profile`	Optional numeric vector of per-attribute separation weights. When `NULL`, all attributes are weighted equally. Ignored when `covariates = TRUE`.

attr_corr	Optional attribute correlation: NULL (independent), a scalar equicorrelation, or a full correlation matrix. Applies only to the random (non-design) path and is ignored when covariates = TRUE.
design	Optional blocked design from klue_design(). When supplied, respondents share their block's cards.
seg_scale	Optional per-class scale (recycled to length true_K) placing segments at unequal distances from the grand mean. Ignored when covariates = TRUE.
covariates	Logical; when TRUE, class membership is driven by covariates Z1 ~ N(0, 1) and a binary Z2 rather than fixed proportions.
covariate_strength	Scalar scaling the covariate effect on class membership when covariates = TRUE.

Value

A list with elements database (the simulated panel with a CHOICE column), true_betas, true_class, individual_betas, N, T, K, and dgp; when covariates = TRUE it also contains Z1 and Z2 (and adds these columns to database).

---

klue_simulate_choices *Simulate choices for a canonical database from given coefficients*

Description

Draws type-1 extreme-value choices for every (respondent, task) row of a canonical wide database from the utilities $V_j = x_j'\beta$. There is no separate ASC block: alternative-specific constants and framing shifts are carried as indicator attributes of the database.

Usage

```
klue_simulate_choices(database, betas, dgp)
```

Arguments

database	Canonical wide database (e.g. from klue_database_long); an existing CHOICE column is replaced.
betas	Numeric vector of length dgp$n_beta (generic attributes in order, then price), or an N x n_beta matrix of respondent-specific coefficients whose row k belongs to ID == k.
dgp	Data-generating-process specification giving n_alternatives, n_generic and n_beta (klue_dgp).

Value

The database with a simulated CHOICE column.

`klue_starts` *Clustering-based starting values for one method*

Description

Clusters respondents by their choice features, then fits a class-wise MNL to produce starting coefficients and class shares for the latent-class estimator.

Usage

```
klue_starts(
  database,
  C,
  method = "kmeans",
  features = NULL,
  feature_type = c("rp", "onehot"),
  dgp = DGP_DEFAULT,
  cluster_cap = 2000L,
  screen = getOption("klue.screen", TRUE)
)
```

Arguments

`database`	choice data frame with respondent ID, CHOICE, and attribute columns, as consumed by the estimator.
`C`	number of latent classes (number of clusters to form). With C = 1 the data are treated as a single class and no clustering is performed.
`method`	clustering method, one of "kmeans", "gmm", "hc_ward", "hc_complete", "hc_average", "pam".
`features`	optional pre-computed N-by-p feature matrix. When NULL, the features are computed from ‘database‘ according to ‘feature_type‘.
`feature_type`	"rp" (revealed-preference contrasts, the default) or "onehot" (choice indicators; the ablation arm).
`dgp`	data-generating-process specification supplying dimensions such as the number of alternatives, coefficients, and generic coefficients.
`cluster_cap`	maximum number of respondents used for clustering and the cluster-wise MNL start fits; larger samples are subsampled with a fixed seed. Ignored when ‘features‘ is supplied.
`screen`	logical; fit the cluster-wise MNLs at a loose tolerance, keeping non-converged finite-LL fits (the default, since 0.9.1). `FALSE` restores the pre-0.9.1 path: tight fits, fallback on non-convergence. Default reads `getOption("klue.screen", TRUE)`.

Value

A list with ‘betas‘ (a C-by-n_beta matrix of class-wise starting coefficients) and ‘shares‘ (a length-C vector of class shares), or NULL if the chosen clustering method failed to return labels.

---

klue_tost *Two-one-sided-tests (TOST) equivalence test for a contrast*

Description

Tests whether $c'\beta$ lies within $\pm$`margin`. A small `p_tost` rejects non-equivalence (i.e. supports equivalence).

Usage

```
klue_tost(fit, weights, margin, robust = TRUE)
```

Arguments

`fit`	A klue fit.
`weights`	Named weight vector defining the contrast $c'\beta$, or a function of the parameter vector (delta method, `klue_nlcom`).
`margin`	Positive equivalence margin.
`robust`	Use the cluster-robust covariance (default `TRUE`).

Value

Named vector: `estimate`, `se`, `margin`, `p_tost`.

---

klue_wald *Joint Wald test that a set of coefficients equals a value*

Description

Joint Wald test that a set of coefficients equals a value

Usage

```
klue_wald(fit, terms = NULL, value = 0, robust = TRUE, fn = NULL, tol = 1e-08)
```

Arguments

fit	A klue fit.
terms	Character vector of parameter names to test jointly.
value	Null value(s) (recycled). Default 0.
robust	Use the cluster-robust covariance (default TRUE).
fn	Optional function of the parameter vector returning the vector tested against value (delta method). The statistic then uses a generalised inverse of its covariance, and df is that covariance's rank (eigenvalues above tol times the largest). terms is ignored.
tol	Relative eigenvalue tolerance of the rank (with fn).

Value

Named vector: chisq, df, p.

---

klue_wtp *Delta-method willingness-to-pay (ratio of linear combinations)*

Description

Returns $-\text{scale} \times (a'\beta)/(b'\beta)$ with a Delta-method standard error and 95% interval. For a plain WTP pass num = c(attr = 1), den = c(cost = 1).

Usage

```
klue_wtp(fit, num, den, scale = 1, robust = TRUE)
```

Arguments

fit	A klue fit.
num, den	Named weight vectors for the numerator and denominator.
scale	Multiplier (e.g. currency scaling). Default 1.
robust	Use the cluster-robust covariance (default TRUE).

Value

Named vector: wtp, se, lo, hi.

`klue_wtp_fn` *Willingness to pay as a function of the parameters*

Description

Returns `function(par)` giving $-\text{scale}\, a'\beta/\beta_{den}$: in an MNL the ratio, in an LCMNL its share-weighted mean over the classes (the population mean), or one class's ratio with `class`. Common coefficients enter every class with the same value.

Usage

```
klue_wtp_fn(fit, num, den = NULL, scale = 1, class = NULL)
```

Arguments

`fit`	An MNL or constant-share LCMNL klue fit.
`num`	Named weights over coefficient labels (constants included).
`den`	Label of the denominator; default the fit's price coefficient.
`scale`	Multiplier (e.g. currency units). Default 1.
`class`	Optional class number for that class's ratio.

Index

cleanup_apollo, 3

estimate_lcmnl, 3, *4, 5*, *19*
estimate_lcmnl_cov, 4, *19*, *25*
estimate_lcmnl_em, 5

get_all_starts, 6, *19*
get_mnl_perturbation_starts, 7
get_random_partition_starts, 8

klue, 9
klue_database, 10
klue_database_long, 11, *27*
klue_database_wide, 12
klue_demo, 14
klue_derror, 14
klue_design, 15
klue_dgp, *7, 8*, *15*, 16, *24, 25*, *27*
klue_draw_betas, 17
klue_fieller, 17, *20*
klue_lcmnl, *6*, 18, *25*
klue_lincom, 19, *24*
klue_mean_fn, *17*, 20, *23*
klue_mmnl, 20
klue_mmnl_corr *(*klue_mmnl*)*, 20
klue_mmnl_defaults, 23
klue_nlcom, *20*, 23, *29*
klue_par, 24
klue_predict, 24
klue_refit, 25
klue_simulate, 26
klue_simulate_choices, *17*, 27
klue_starts, *7, 8*, *19*, 28
klue_tost, 29
klue_wald, 29
klue_wtp, *24*, 30
klue_wtp_fn, *23*, 31