Package 'apollo'

March 13, 2026

Type Package

Title Tools for Choice Model Estimation and Application

Version 0.3.7

Description Choice models are a widely used technique across numerous scientific disciplines. The Apollo package is a very flexible tool for the estimation and application of choice models in R. Users are able to write their own model functions or use a mix of already available ones. Random heterogeneity, both continuous and discrete and at the level of individuals and choices, can be incorporated for all models. There is support for both standalone models and hybrid model structures. Both classical and Bayesian estimation is available, and multiple discrete continuous models are covered in addition to discrete choice. Multi-threading processing is supported for estimation and a large number of pre and post-estimation routines, including for computing posterior (individual-level) distributions are available. For examples, a manual, and a support forum, visit <https://www.ApolloChoiceModelling.com>. For more information on choice models see Train, K. (2009) <isbn:978-0-521-74738-7> and Hess, S. & Daly, A.J. (2014) <isbn:978-1-781-00314-5> for an overview of the field.

License GPL-2

URL https://www.ApolloChoiceModelling.com

BugReports https://www.ApolloChoiceModelling.com/forum/

Encoding UTF-8

LazyData true

Depends R (>= 4.3.0), stats, utils

Imports Rcpp (>= 1.0.0), maxLik, mnormt, mvtnorm, graphics, randtoolbox, numDeriv, parallel, Deriv, matrixStats, RSGHB, coda, tibble, stringr, bgw (>= 0.1.4), cli, Rsolnp, rstudioapi

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation yes

Author Stephane Hess [aut, cre],
David Palma [aut],
Thomas Hancock [ctb]

Maintainer Stephane Hess <S.Hess@leeds.ac.uk>

Repository CRAN

Date/Publication 2026-03-13 08:00:23 UTC

Contents

.onAttach 4
apollo_addCovariance 5
apollo_attach 5
apollo_avgInterDraws 6
apollo_avgIntraDraws 7
apollo_basTest 9
apollo_bootstrap 9
apollo_checkArguments 12
apollo_choiceAnalysis 13
apollo_classAlloc 14
apollo_cnl 15
apollo_cnl2 17
apollo_combineModels 19
apollo_combineResults 21
apollo_compareInputs 22
apollo_conditionals 23
apollo_deltaMethod 24
apollo_detach 25
apollo_dft 26
apollo_diagnostics 29
apollo_drugChoiceData 29
apollo_dVdB 31
apollo_dVdBOld 31
apollo_el 32
apollo_emdc 34
apollo_emdc1 35
apollo_emdc2 37
apollo_estimate 38
apollo_estimateHB 40
apollo_expandLoop 42
apollo_firstRow 42
apollo_fitsTest 43
apollo_fmnl 44
apollo_fnl 46
apollo_initialise 48

apollo_insertComponentName . . . 49
apollo_insertFunc . . . 49
apollo_insertOLList . . . 50
apollo_insertRows . . . 51
apollo_insertRRMQuotes . . . 51
apollo_insertScaling . . . 52
apollo_keepRows . . . 52
apollo_lc . . . 53
apollo_lcConditionals . . . 54
apollo_lcEM . . . 55
apollo_lcEM_new . . . 57
apollo_lcUnconditionals . . . 58
apollo_llCalc . . . 59
apollo_loadModel . . . 60
apollo_longToWide . . . 60
apollo_lrTest . . . 61
apollo_makeCluster . . . 62
apollo_makeDraws . . . 63
apollo_makeGrad . . . 64
apollo_makeHessian . . . 65
apollo_makeLogLike . . . 65
apollo_mdcev . . . 67
apollo_mdcev2 . . . 69
apollo_mdcnev . . . 71
apollo_mixConditionals . . . 74
apollo_mixEM . . . 75
apollo_mixUnconditionals . . . 76
apollo_mlhs . . . 77
apollo_mnl . . . 78
apollo_modeChoiceData . . . 80
apollo_modelOutput . . . 81
apollo_modifyUserDefFunc . . . 82
apollo_nl . . . 84
apollo_normalDensity . . . 86
apollo_ol . . . 88
apollo_op . . . 90
apollo_outOfSample . . . 92
apollo_ownModel . . . 94
apollo_panelProd . . . 96
apollo_prediction . . . 98
apollo_preEstimate . . . 99
apollo_prepareProb . . . 101
apollo_preprocess . . . 103
apollo_print . . . 104
apollo_readBeta . . . 105
apollo_rrm . . . 105
apollo_saveOutput . . . 108
apollo_searchStart . . . 110

apollo_setRows . . . 112
apollo_setWorkDir . . . 112
apollo_sharesTest . . . 113
apollo_sink . . . 114
apollo_speedTest . . . 115
apollo_swissRouteChoiceData . . . 116
apollo_timeUseData . . . 117
apollo_tobit . . . 118
apollo_unconditionals . . . 120
apollo_validate . . . 121
apollo_validateControl . . . 122
apollo_validateData . . . 124
apollo_validateHBControl . . . 124
apollo_validateInputs . . . 126
apollo_varcov . . . 129
apollo_varList . . . 130
apollo_weighting . . . 131
apollo_writeF12 . . . 132
apollo_writeTheta . . . 133
aux_validateRows . . . 133
print.apollo . . . 134
summary.apollo . . . 134

Index **136**

`.onAttach` *Prints package startup message*

Description

This function is only called by R when attaching the package.

Usage

```
.onAttach(libname, pkgname)
```

Arguments

`libname`	Name of library.
`pkgname`	Name of package.

Value

Nothing

apollo_addCovariance *Adds covariance matrix to Apollo model*

Description

Receives an estimated model object, calculates its Hessian, and classical and robust covariance matrix, and returns the same model object, but with these additional elements.

Usage

```
apollo_addCovariance(model, apollo_inputs)
```

Arguments

model	Model object. Estimated model object as returned by function apollo_estimate.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.

Value

model.

apollo_attach *Attaches predefined variables.*

Description

Attaches parameters and data to allow users to refer to individual variables by name without reference to the object that contains them.

Usage

```
apollo_attach(apollo_beta, apollo_inputs)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.

Details

This function should be called at the beginning of apollo_probabilities to make writing the log-likelihood more user-friendly. If used, then apollo_detach should be called at the end of apollo_probabilities, or more conveniently, using on.exit after the initial call to apollo_attach. apollo_attach attaches apollo_beta, database, draws, and the output of apollo_randCoeff and apollo_lcPars, if they are defined by the user.

Value

Nothing.

`apollo_avgInterDraws` *Averages across inter-individual draws.*

Description

Averages individual-specific likelihood across inter-individual draws.

Usage

```
apollo_avgInterDraws(P, apollo_inputs, functionality)
```

Arguments

`P`	List of vectors, matrices or 3-dim arrays. Likelihood of the model components.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`functionality`	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are: • `"components"`: For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • `"conditionals"`: For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • `"estimate"`: For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • `"gradient"`: For model estimation, produces analytical gradients of the likelihood, where possible. • `"output"`: Prepares output for post-estimation reporting. • `"prediction"`: For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws. • `"preprocess"`: Prepares likelihood functions for use in estimation. • `"raw"`: For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws. • `"report"`: Prepares output summarising model and choiceset structure. • `"shares_LL"`: Produces overall model likelihood with constants only. • `"utilities"`: Returns utilities at provided parameter values. • `"validate"`: Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • `"zero_LL"`: Produces overall model likelihood with all parameters at zero.

Value

Argument P with (for most functionalities) the original contents averaged over inter-individual draws. Shape depends on argument functionality.

- "components": Returns P without changes.
- "conditionals": Returns P without averaging across draws. Drops all components except "model".
- "estimate": Returns P containing the likelihood of the model averaged across inter-individual draws. Drops all components except "model".
- "gradient": Returns P containing the gradient of the likelihood averaged across inter-individual draws. Drops all components except "model".
- "output": Returns P containing the likelihood of all model components averaged across inter-individual draws.
- "prediction": Returns P containing the probabilities/likelihoods of all alternatives for all model components averaged across inter-individual draws.
- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "report": Returns P without changes.
- "shares_LL": Returns P without changes.
- "utilities": Returns P without changes.
- "validate": Returns P containing the likelihood of the model averaged across inter-individual draws. Drops all components except "model".
- "zero_LL": Returns P without changes.

---

apollo_avgIntraDraws *Averages across intra-individual draws.*

Description

Averages observation-specific likelihood across intra-individual draws.

Usage

```
apollo_avgIntraDraws(P, apollo_inputs, functionality)
```

Arguments

P	List of vectors, matrices or 3-dim arrays. Likelihood of the model components.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

Argument P with (for most functionalities) the original contents averaged over intra-individual draws. Shape depends on argument functionality.

- "components": Returns P without changes.
- "conditionals": Returns P containing the likelihood of the model averaged across intra-individual draws. Drops all components except for "model".
- "estimate": Returns P containing the likelihood of the model averaged across intra-individual draws. Drops all components except "model".
- "gradient": Returns P containing the gradient of the likelihood averaged across intra-individual draws. Drops all components except "model".
- "output": Returns P containing the likelihood of all model components averaged across intra-individual draws.
- "prediction": Returns P containing the probabilities of all alternatives for all model components averaged across intra-individual draws.
- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "report": Returns P without changes.
- "utilities": Returns P without changes.

- "validate": Returns P containing the likelihood of the model averaged across intra-individual draws. Drops all components but "model".
- "zero_LL": Returns P without changes.

---

apollo_basTest *Ben-Akiva & Swait test*

Description

Carries out the Ben-Akiva & Swait test for non-nested models and reports the corresponding p-value.

Usage

```
apollo_basTest(model1, model2)
```

Arguments

model1	Either a character variable with the name (and possibly path) of a previously estimated model, or an estimated model in memory, as returned by apollo_estimate.
model2	Either a character variable with the name (and possibly path) of a previously estimated model, or an estimated model in memory, as returned by apollo_estimate.

Details

The two models need to both be discrete choice, and need to have been estimated on the same data.

Value

Ben-Akiva & Swait test p-value (invisibly)

---

apollo_bootstrap *Bootstrap a model*

Description

Samples individuals with replacement from the database, and estimates the model for each sample.

Usage

```
apollo_bootstrap(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
 estimate_settings = list(estimationRoutine = "bgw", maxIterations = 200, writeIter =
  FALSE, hessianRoutine = "none", printLevel = 2L, silent = FALSE, maxLik_settings =
    list()),
 bootstrap_settings = list(nRep = 30, samples = NA, calledByEstimate = FALSE, recycle =
    TRUE)
)
```

Arguments

`apollo_beta` Named numeric vector. Names and values for parameters.

`apollo_fixed` Character vector. Names (as defined in `apollo_beta`) of parameters whose value should not change during estimation.

`apollo_probabilities`
Function. Returns probabilities of the model to be estimated. Must receive three arguments:

- `apollo_beta`: Named numeric vector. Names and values of model parameters.
- `apollo_inputs`: List containing options of the model. See apollo_validateInputs.
- `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.

`apollo_inputs` List grouping most common inputs. Created by function apollo_validateInputs.

`estimate_settings`
List. Options controlling the estimation process. See apollo_estimate. `hessianRoutine="none"` by default.

`bootstrap_settings`
List containing settings for the sampling procedure. User input is required for all settings except those with a default or marked as optional.

- **calledByEstimate**: Logical. TRUE if `apollo_bootstrap` is called by apollo_estimate. FALSE by default.
- **nRep**: Numeric scalar. Number of times the model must be estimated with different samples. Default is 30.
- **recycle**: Logical. If TRUE, the function will look for old output files and append new repetitions to them. If FALSE, output files will be overwritten.
- **samples**: Numeric matrix or data.frame. Optional argument. Must have as many rows as observations in the `database`, and as many columns as number of repetitions wanted. Each column represents a re-sample, and each element the number of times that observation must be included in the sample. If this argument is provided, then `nRep` is ignored. Note that this

allows sampling at the observation rather than the individual level, which is not recommended for panel data.

- **seed**: **DEPRECATED,** `apollo_control$seed` **is used since v0.2.5**. Numeric scalar (integer). Random number generator seed to generate the bootstrap samples. Only used if `samples` is `NA`. Default is 24.

Details

This function implements a basic block bootstrap. It estimates the model parameters on `nRep` different samples. Each new sample is constructed by sampling **with replacement** from the original full sample. Each new sample has as many individuals as the original sample, though some of them may be repeated. Sampling is done at the **individual** level, therefore if different individuals have different number of observations, each re-sample does not necessarily have the same number of observations.

If the sampling should be done at the individual level (not recommended for panel data), then the optional `bootstrap_settings$samples` argument should be provided.

For each sample, only the parameters and log-likelihood are estimated. Standard errors are not calculated (they may be added in future versions). The composition of the re-samples is stored in a file, but is stable with the same seed.

This function writes three different files to the working or output directory:

- `modelName_bootstrap_params.csv`: estimated parameters, final log-likelihood, and number of observations for each re-sample
- `modelName_bootstrap_samples.csv`: composition of each re-sample.
- `modelName_bootstrap_vcov.csv`: variance-covariance matrix of the estimated parameters across re-samples.

The first two files are updated throughout the run of this function, while the last one is only written once the function finishes.

When run, this function will look for the first two files above in the working/output directory. If they are found, the function will attempt to pick up re-sampling from where those files left off. This is useful in cases where the original bootstrapping was interrupted, or when additional re-sampling runs are to be performed.

Value

List with three elements.

- `estimates`: Matrix containing the parameter estimates for each repetition. As many rows as repetitions and as many columns as parameters.
- `LL`: Vector of final log-likelihoods of each repetition.
- `varcov`: Covariance matrix of the estimated parameters across the repetitions.

This function also writes three output files to the working/output directory, with the following names ('x' represents the model name):

- **x_bootstrap_params.csv**: Table containing the parameter estimates, log-likelihood, and number of observations for each repetition.

- **x_bootstrap_samples.csv**: Table containing the description of the sample used in each repetition. Same format than `bootstrap_settings$samples`.
- **x_bootstrap_vcov**: Table containing the covariance matrix of estimated parameters across the repetitions.

`apollo_checkArguments` *Checks definitions of Apollo functions*

Description

Checks that the user-defined functions used by Apollo are correctly defined by the user.

Usage

```
apollo_checkArguments(
  apollo_probabilities = NA,
  apollo_randCoeff = NA,
  apollo_lcPars = NA
)
```

Arguments

`apollo_probabilities`
: Function. Likelihood function as defined by the user.

`apollo_randCoeff`
: Function. Used with mixing models. Constructs the random parameters of a mixing model. Receives two arguments:
  - `apollo_beta`: Named numeric vector. Names and values of model parameters.
  - `apollo_inputs`: The output of this function (`apollo_validateInputs`).

`apollo_lcPars`
: Function. Used with latent class models. Constructs a list of parameters for each latent class. Receives two arguments:
  - `apollo_beta`: Named numeric vector. Names and values of model parameters.
  - `apollo_inputs`: The output of this function (`apollo_validateInputs`).

Details

It only checks that the functions have the correct definition of inputs. It does not run the functions.

Value

Returns (invisibly) TRUE if definitions are correct, and FALSE otherwise.

`apollo_choiceAnalysis` *Reports market share for subsamples*

Description

Compares market shares across subsamples in dataset, and conducts statistical tests.

Usage

```
apollo_choiceAnalysis(choiceAnalysis_settings, apollo_control, database)
```

Arguments

`choiceAnalysis_settings`
: List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `alternatives`: Named numeric vector. Names of alternatives and their corresponding value in `choiceVar`. Note that these need not necessarily be the alternatives as defined in the model, but could e.g. relate to cheapest/most expensive.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.
- `choiceVar`: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- `explanators`: data.frame. Variables determining subsamples of the database. Values in each column must describe a group or groups of individuals (e.g. socio-demographics). Most usually a subset of columns from the database.
- `printToScreen`: Logical. TRUE for returning output to screen as well as file. TRUE by default.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`apollo_control`
: List. Options controlling the running of the code. See apollo_validateInputs.

`database`
: data.frame. Data used by model.

Details

Saves the output to a csv file in the working/output directory.

Value

Silently returns a matrix containing the mean value for each explanator for those cases where an alternative is chosen and where it is not chosen, as well as the t-test comparing those means (H0: equivalence). The table is also written to a file called `modelName_choiceAnalysis.csv` and printed to screen.

`apollo_classAlloc` *Calculates class allocation probabilities for a Latent Class model*

Description

Calculates class allocation probabilities for a Latent Class model using a Multinomial Logit model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_classAlloc(classAlloc_settings)
```

Arguments

`classAlloc_settings`

List of inputs of the MNL model. It should contain the following.

- `avail`: Named list of numeric vectors or scalars. Availabilities of classes, one element per class Names of elements must match those in `classes`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.
- `utilities`: Named list of deterministic utilities . Utilities of the classes in class allocation model. Names of elements must match those in `avail`, if provided.

Value

The returned object depends on the value of argument `functionality`, which it fetches from the calling stack (see apollo_validateInputs).

- `"components"`: Same as `"estimate"`.
- `"conditionals"`: Same as `"estimate"`.
- `"estimate"`: List of vector/matrices/arrays with the allocation probabilities for each class.

- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate".
- "prediction": Same as "estimate".
- "preprocess": Returns a list with pre-processed inputs, based on classAlloc_settings.
- "raw": Same as "estimate".
- "report": Same as "estimate".
- "shares_LL": List with probabilities for each class in an equal shares setting.
- "utilities": List of vectors/matrices/arrays. Returns a list with the utilities for all classes in class allocation model.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": List with probabilities for each class in an equal shares setting.

---

apollo_cnl *Calculates Cross-Nested Logit probabilities*

Description

Calculates the probabilities of a Cross-nested Logit model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_cnl(cnl_settings, functionality)
```

Arguments

cnl_settings — List of inputs of the CNL model. User input is required for all settings except those with a default or marked as optional.

- alternatives: Named numeric vector. Names of alternatives and their corresponding value in choiceVar.
- avail: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in alternatives. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify avail=1 to indicate universal availability, or omit the setting completely.
- choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in alternatives.
- cnlNests: List of numeric scalars or vectors. Lambda parameters for each nest. Elements must be named according to nests. The lambda at the root is fixed to 1, and therefore does not need to be defined.
- cnlStructure: Numeric matrix. One row per nest and one column per alternative. Each element of the matrix is the alpha parameter of that (nest, alternative) pair.

- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- utilities: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in alternatives.
- rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

For the model to be consistent with utility maximisation, the estimated value of the lambda parameter of all nests should be between 0 and 1. Lambda parameters are inversely proportional to the correlation between the error terms of alternatives in a nest. If lambda=1, there is no relevant correlation between the unobserved utility of alternatives in that nest. Alpha parameters inside cnlStructure should be between 0 and 1. Using a transformation to ensure this constraint is satisfied is recommended for complex structures (e.g. logistic transformation).

Value

The returned object depends on the value of argument `functionality` as follows.

- `"components"`: Same as `"estimate"`
- `"conditionals"`: Same as `"estimate"`
- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- `"gradient"`: Not implemented.
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"prediction"`: List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the chosen alternative probability.
- `"preprocess"`: Returns a list with pre-processed inputs, based on `cnl_settings`.
- `"raw"`: Same as `"prediction"`.
- `"report"`: List with tree structure and choice overview.
- `"shares_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- `"utilities"`: List of vectors/matrices/arrays. Returns the utilities.
- `"validate"`: Same as `"estimate"`.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

`apollo_cnl2` *Calculates Cross-Nested Logit probabilities*

Description

Calculates the probabilities of a Cross-nested Logit model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_cnl2(cnl_settings, functionality)
```

Arguments

`cnl_settings` List of inputs of the CNL model. User input is required for all settings except those with a default or marked as optional.

- `alternatives`: Named numeric vector. Names of alternatives and their corresponding value in `choiceVar`.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.

- `choiceVar`: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- `cnlNests`: List of numeric scalars or vectors. Lambda parameters for each nest. Elements must be named according to nests. The lambda at the root is fixed to 1, and therefore does not need to be defined.
- `cnlStructure`: Numeric matrix. One row per nest and one column per alternative. Each element of the matrix is the alpha parameter of that (nest, alternative) pair.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `utilities`: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in `alternatives`.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- `"components"`: For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- `"conditionals"`: For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- `"estimate"`: For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- `"gradient"`: For model estimation, produces analytical gradients of the likelihood, where possible.
- `"output"`: Prepares output for post-estimation reporting.
- `"prediction"`: For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- `"preprocess"`: Prepares likelihood functions for use in estimation.
- `"raw"`: For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- `"report"`: Prepares output summarising model and choiceset structure.
- `"shares_LL"`: Produces overall model likelihood with constants only.
- `"utilities"`: Returns utilities at provided parameter values.
- `"validate"`: Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- `"zero_LL"`: Produces overall model likelihood with all parameters at zero.

Details

For the model to be consistent with utility maximisation, the estimated value of the lambda parameter of all nests should be between 0 and 1. Lambda parameters are inversely proportional to the correlation between the error terms of alternatives in a nest. If lambda=1, there is no relevant correlation between the unobserved utility of alternatives in that nest. Alpha parameters inside `cnlStructure` should be between 0 and 1. Using a transformation to ensure this constraint is satisfied is recommended for complex structures (e.g. logistic transformation).

Value

The returned object depends on the value of argument `functionality` as follows.

- `"components"`: Same as `"estimate"`
- `"conditionals"`: Same as `"estimate"`
- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- `"gradient"`: Not implemented.
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"prediction"`: List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the chosen alternative probability.
- `"preprocess"`: Returns a list with pre-processed inputs, based on `cnl_settings`.
- `"raw"`: Same as `"prediction"`.
- `"report"`: List with tree structure and choice overview.
- `"shares_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- `"utilities"`: List of vectors/matrices/arrays. Returns the utilities.
- `"validate"`: Same as `"estimate"`.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

`apollo_combineModels` *Combines separate model components.*

Description

Combines model components to create likelihood for overall model.

Usage

```
apollo_combineModels(
  P,
  apollo_inputs,
  functionality,
  components = NULL,
  asList = TRUE
)
```

Arguments

P	List of vectors, matrices or 3-dim arrays. Likelihood of the model components.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "gradient": For model estimation, produces analytical gradients of the likelihood, where possible. • "output": Prepares output for post-estimation reporting. • "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws. • "preprocess": Prepares likelihood functions for use in estimation. • "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws. • "report": Prepares output summarising model and choiceset structure. • "shares_LL": Produces overall model likelihood with constants only. • "utilities": Returns utilities at provided parameter values. • "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "zero_LL": Produces overall model likelihood with all parameters at zero.
components	Character vector. Optional argument. Names of elements in P that should be multiplied to construct the whole model likelihood. If a single element is provided, it is interpreted as a regular expression. Default is to include all components in P.
asList	Logical. Only used if functionality is "conditionals","estimate","validate","zero_LL" or "output". If TRUE, it will return a list as described in the 'Value' section. If FALSE, it will only return a vector/matrix/3-dim array of the product of likelihoods inside P. Default is TRUE.

Details

This function should be called inside apollo_probabilities after all model components have been produced.

It should be called before apollo_avgInterDraws, apollo_avgIntraDraws, apollo_panelProd and apollo_prepareProb, whichever apply, except where these functions are called inside any latent class components of the overall model.

Value

Argument P with (for most functionalities) an extra element called "model", which is the product of all the other elements. Shape depends on argument functionality.

- "components": Returns P without changes.
- "conditionals": Returns P with an extra component called "model", which is the product of all other elements of P.
- "estimate": Returns P with an extra component called "model", which is the product of all other elements of P.
- "gradient": Returns P containing the gradient of the likelihood after applying the product rule across model components.
- "output": Returns P with an extra component called "model", which is the product of all other elements of P.
- "prediction": Returns P without changes.
- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "shares_LL": Returns P with an extra component called "model", which is the product of all other elements of P.
- "utilities": Returns P without changes.
- "validate": Returns P with an extra component called "model", which is the product of all other elements of P.
- "zero_LL": Returns P with an extra component called "model", which is the product of all other elements of P.

---

apollo_combineResults *Write model results to file*

Description

Writes results from various models to a single csv file.

Usage

```
apollo_combineResults(combineResults_settings = NULL)
```

Arguments

`combineResults_settings`

List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `estimateDigits`: Numeric scalar. Number of decimal places to print for estimates. Default is 4.
- `modelNames`: Character vector. Optional names of models to combine. Omit or use an empty vector to combine results from all models in the working/output directory.
- `pDigits`: Numeric scalar. Number of decimal places to print for p-values. Default is 2.
- `printClassical`: Boolean. TRUE for printing classical standard errors. FALSE by default.
- `printPVal`: Boolean. TRUE for printing p-values. FALSE by default.
- `printT1`: Boolean. If TRUE, t-test for H0: apollo_beta=1 are printed. FALSE by default.
- `sortByDate`: Boolean. If TRUE, models are ordered by date. Default is TRUE.
- `tDigits`: Numeric scalar. Number of decimal places to print for t-ratios values. Default is 2.

Value

Nothing, but writes a file called 'model_comparison_[date].csv' in the working/output directory.

`apollo_compareInputs` *Compares the content of apollo_inputs to their counterparts in the global environment*

Description

Compares the content of apollo_inputs to their counterparts in the global environment

Usage

```
apollo_compareInputs(apollo_inputs)
```

Arguments

`apollo_inputs` List grouping most common inputs. Created by function apollo_validateInputs.

Value

Logical. TRUE if the content of `apollo_inputs` is the same than the one in the global environment, FALSE otherwise.

`apollo_conditionals` *Calculates conditionals*

Description

Calculates posterior expected values (conditionals) of random coefficient models (continuous or discrete mixtures/latent class)

Usage

```
apollo_conditionals(model, apollo_probabilities, apollo_inputs)
```

Arguments

`model`	Model object. Estimated model object as returned by function apollo_estimate.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • `apollo_beta`: Named numeric vector. Names and values of model parameters. • `apollo_inputs`: List containing options of the model. See apollo_validateInputs. • `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.

Details

This functions is only meant for use with models using either continuous distributions or latent classes, not both at the same time

Value

Depends on whether the model uses continuous mixtures or latent class.

- If the model contains a continuous mixture, the function returns a list of matrices. Each matrix has dimensions nIndiv x 3. One matrix per random component. Each row of each matrix contains the indivID of an individual, and the posterior mean and s.d. of this random component for this individual.
- If the model contains latent classes, the function returns a matrix with the posterior class allocation probabilities for each individual.
- If the model contains both continuous mixtures and latent classes, the function fails.

`apollo_deltaMethod` *Delta method for Apollo models*

Description

Applies the Delta method to calculate the standard errors of transformations of parameters.

Usage

```
apollo_deltaMethod(model, deltaMethod_settings)
```

Arguments

`model` Model object. Estimated model object as returned by function apollo_estimate.

`deltaMethod_settings`
List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `expression`: Character vector. A character vector with a single or multiple arbitrary functions of the estimated parameters, as text. For example: `c(VTT="b1/b2*60")`. Each expression can only contain model parameters (estimated or fixed), numeric values, and operands. At least one of the parameters used needs to not have been fixed in estimation. Variables in the database cannot be included. If the user does not provide a name for an expression, then the expression itself is used in the output. If this setting is provided, then `operation`, `parName1`, `parName2`, `multPar1` and `multPar2` are ignored.
- `allPairs`: Logical. If set to TRUE, Delta method calculations are carried out for the ratio and difference for all pairs of parameters and returned as two separate matrices with values and t-ratios. FALSE by default.
- `varcov`: Character. Type of variance-covariance matrix to use in calculations. It can take values `"classical"`, `"robust"` and `"bootstrap"`. Default is `"robust"`.
- `printPVal`: Logical or Scalar. TRUE or 1 for printing p-values for one-sided test, 2 for printing p-values for two-sided test, FALSE for not printing p-values. FALSE by default.
- `operation`: Character. Function to calculate the delta method for. See details. Not used if `expression` is provided.
- `parName1`: Character. Name of the first parameter if `operation` is used. See details. Not used if `expression` is provided.
- `parName2`: Character. Name of the second parameter if `operation` is used. See details. Not used if `expression` is provided.. Optional depending on `operation`.
- `multPar1`: Numeric scalar. An optional value to scale `parName1`. Not used if `expression` is provided.
- `multPar2`: Numeric scalar. An optional value to scale `parName2`. Not used if `expression` is provided.

Details

apollo_deltaMethod can be used in two ways. The first and recommended way is to provide an element called expression inside its argument deltaMethod_settings. expression should contain the expression or expressions for which the standard error is/are to be calculated, as text. For example, to calculate the ratio between parameters b1 and b2, expression=c(vtt="b1/b2") should be used.

The second method is to provide the name of a specific operation inside deltaMethod_settings. The following five operations are supported.

- diff: Calculates the s.e. of parName1 - parName2 and parName2 - parName1
- exp: Calculates the s.e. of exp(parName1)
- logistic: If only parName1 is provided, it calculates the s.e. of exp(parName1)/(1+exp(parName1)) and 1/(1+exp(parName1)). If parName1 and parName2 are provided, it calculates exp(par_i)/(1+exp(parName1)+e for i=1, 2, and 3 (par_3 = 1).
- lognormal: Calculates the mean and s.d. of a lognormal distribution based on the mean (parName1) and s.d. (parName2) of the underlying normal.
- prod: Calculates the s.e. of parName1*parName2
- ratio: Calculates the s.e. of parName1/parName2 and parName2/parName1
- sum: Calculates the s.e. of parName1 + parName2

By default, apollo_deltaMethod uses the robust covariance matrix. However, the user can change this through the varcov setting.

Value

Matrix containing value, s.e. and t-ratio resulting from the requested expression or operation. This is also printed to screen.

---

apollo_detach *Detaches parameters and the database.*

Description

Detaches variables attached by apollo_attach.

Usage

```
apollo_detach(apollo_beta = NA, apollo_inputs = NA)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.

Details

This function detaches the variables attached by apollo_attach. It should be called at the end of apollo_probabilities, only if apollo_attach was called and the beginning. This can also be achieved by adding the line on.exit(apollo_detach(apollo_beta, apollo_inputs)) right after calling apollo_attach. This function can also be called without any arguments, i.e. apollo_detach().

Value

Nothing.

---

apollo_dft *Calculate DFT probabilities*

Description

Calculate probabilities of a Decision Field Theory (DFT) model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_dft(dft_settings, functionality)
```

Arguments

dft_settings	List of settings for the DFT model. It should contain the following elements.

- alternatives: Named numeric vector. Names of alternatives and their corresponding value in choiceVar.
- avail: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in alternatives. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify avail=1 to indicate universal availability, or omit the setting completely.
- altStart: A named list with as many elements as alternatives. Each element can be a scalar or vector containing the starting preference value for the alternative.
- attrScalings: A named list with as many elements as attributes, or fewer. Each element is a factor that scale the attribute, and can be a scalar, a vector or a matrix/array. They do not need to add up to one for each observation. attrWeights and attrScalings are incompatible, and they should not be both defined for an attribute. Default is 1 for all attributes.
- attrValues: A named list with as many elements as alternatives. Each element is itself a named list of vectors of the alternative attributes for each observation (usually a column from the database). All alternatives must have the same attributes (can be set to zero if not relevant).

- attrWeights: A named list with as many elements as attributes, or fewer. Each element is the weight of the attribute, and can be a scalar, a vector with as many elements as observations, or a matrix/array if random. They should add up to one for each observation and draw (if present), and will be re-scaled if they do not. attrWeights and attrScalings are incompatible, and they should not be both defined for an attribute. Default is 1 for all attributes.
- choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in alternatives.
- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- **procPars**: A list containing the four DFT 'process parameters'
  - **error_sd**: Numeric scalar or vector. The standard deviation of the the error term in each timestep.
  - **timesteps**: Numeric scalar or vector. Number of timesteps to consider. Should be an integer bigger than 0.
  - **phi1**: Numeric scalar or vector. Sensitivity.
  - **phi2**: Numeric scalar or vector. Process parameter.
- rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.

- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": Not implemented.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the chosen alternative probability.
- "preprocess": Returns a list with pre-processed inputs, based on dft_settings.
- "raw": Same as "prediction"
- "report": Choice overview.
- "shares_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "validate": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

References

Hancock, T.; Hess, S. and Choudhury, C. (2018) Decision field theory: Improvements to current methodology and comparisons with standard choice modelling techniques. Transportation Research 107B, 18 - 40. Hancock, T.; Hess, S.; Marley A.A.J. and Choudhury, C. (2021), An accumulation of preference: two alternative dynamic models for understanding transport choices, Transportation Research Part B, Volume 149, July 2021, Pages 250-282. Roe, R.; Busemeyer, J. and Townsend, J. (2001) Multialternative decision field theory: A dynamic connectionist model of decision making. Psychological Review 108, 370

apollo_diagnostics *Pre-process input for common models return*

Description

Pre-process input for common models return

Usage

```
apollo_diagnostics(inputs, modelType, apollo_inputs, data = TRUE, param = TRUE)
```

Arguments

inputs	List of settings
modelType	Character. Type of model, e.g. "mnl", "nl", "cnl", etc.
apollo_inputs	List of main inputs to the model estimation process. See apollo_validateInputs.
data	Boolean. TRUE for printing report related to dependent and independent variables. FALSE for not printing it. Default is TRUE.
param	Boolean. TRUE for printing report related to estimated parameters (e.g. model structure). FALSE for not printing it. Default is TRUE.

Value

(invisibly) TRUE if no error happend during execution.

apollo_drugChoiceData *Simulated dataset of medication choice.*

Description

A simulated dataset containing 10,000 stated medication choices among four alternatives.

Usage

```
apollo_drugChoiceData
```

Format

A data.frame with 10,000 rows and 33 variables:

ID Numeric. Identification number of the individual.

task Numeric. Index of choice situations for each individual, going from 1 to 10.

best Numeric. Index of alternative selected as best option.

second_pref Numeric. Index of alternative selected as second-best option.

third_pref Numeric. Index of alternative selected as third-best option.

worst Numeric. Index of alternative selected as worst option.

brand_1 Character. Brand for alternative 1.

country_1 Character. Country of origin for alternative 1.

char_1 Character. Characteristics of alternative 1 (standard, fast acting, or double strength).

side_effects_1 Numeric. Chance of suffering negative side effects with alternative 1 (out of 100,000).

price_1 Numeric. Cost of alternative 1 in Pounds sterling (GBP).

brand_2 Character. Brand for alternative 2.

country_2 Character. Country of origin for alternative 2.

char_2 Character. Characteristics of alternative 2 (standard, fast acting, or double strength).

side_effects_2 Numeric. Chance of suffering negative side effects with alternative 2 (out of 100,000).

price_2 Numeric. Cost of alternative 2 in Pounds sterling (GBP).

brand_3 Character. Brand for alternative 3.

country_3 Character. Country of origin for alternative 3.

char_3 Character. Characteristics of alternative 3 (standard, fast acting, or double strength).

side_effects_3 Numeric. Chance of suffering negative side effects with alternative 3 (out of 100,000).

price_3 Numeric. Cost of alternative 3 in Pounds sterling (GBP).

brand_4 Character. Brand for alternative 4.

country_4 Character. Country of origin for alternative 4.

char_4 Character. Characteristics of alternative 4 (standard, fast acting, or double strength).

side_effects_4 Numeric. Chance of suffering negative side effects with alternative 4 (out of 100,000).

price_4 Numeric. Cost of alternative 4 in Pounds sterling (GBP).

regular_user Numeric. 1 if the respondent is a regular user of headache medicine, 0 otherwise.

university_educated Numeric. 1 if the respondent holds a university degree, 0 otherwise.

over_50 Numeric. 1 if the respondent is 50 years old or older, 0 otherwise.

attitude_quality Numeric. Level of agreement from 1 (strongly disagree) to 5 (strongly agree) with the phrase 'I am concerned about the quality of drugs developed by unknown companies'.

attitude_ingredients Numeric. Level of agreement from 1 (strongly disagree) to 5 (strongly agree) with the phrase 'I believe that ingredients are the same no matter what brand'.

attitude_patent Numeric. Level of agreement from 1 (strongly disagree) to 5 (strongly agree) with the phrase 'The original patent holders have valuable experience with their medicines'.

attitude_dominance Numeric. Level of agreement from 1 (strongly disagree) to 5 (strongly agree) with the phrase 'I believe the dominance of big pharmaceutical companies is unhelpful'.

Details

This dataset is to be used for discrete choice modelling. Data comes from 1,000 individuals, each with ten stated choice (SC) scenarios involving a choice among headache medication. There are 10,000 choices in total. Data is simulated. Each observation contains attributes of the alternatives, characteristics of the respondent, and their answers to four attitudinal questions. All four alternatives are always available for all individuals. Alternatives 1 and 2 are branded, while alternatives 3 and 4 are generic. Respondents provide a full ranking of alternatives for each choice task (i.e. observation).

Source

https://www.ApolloChoiceModelling.com/

`apollo_dVdB` *Calculates gradients of utility functions*

Description

Calculates gradients (derivatives) of utility functions.

Usage

```
apollo_dVdB(apollo_beta, apollo_inputs, V)
```

Arguments

`apollo_beta`	Named numeric vector of parameters.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`V`	List of functions

Value

Named list. Each element is itself a list of functions: the partial derivatives of the elements of V.

`apollo_dVdBOld` *Calculates gradients of utility functions*

Description

Calculates gradients (derivatives) of utility functions.

Usage

```
apollo_dVdBOld(apollo_beta, apollo_inputs, V)
```

Arguments

`apollo_beta`	Named numeric vector of parameters.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`V`	List of functions

Value

Named list. Each element is a function that returns a list, where each element is the partial derivatives of the elements of V.

apollo_el *Calculates Exploded Logit probabilities*

Description

Calculates the probabilities of an Exploded Logit model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_el(el_settings, functionality)
```

Arguments

el_settings — List of inputs of the Exploded Logit model. It shoud contain the following.

- alternatives: Named numeric vector. Names of alternatives and their corresponding value in choiceVar.
- avail: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in alternatives. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify avail=1 to indicate universal availability, or omit the setting completely.
- choiceVars: List of numeric vectors. Contain choices for each position of the ranking. The list must be ordered with the best choice first, second best second, etc. It will usually be a list of columns from the database. Use value -1 if a stage does not apply for a given observations (e.g. when some individuals have shorter rankings).
- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- utilities: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in alternatives.
- rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).
- scales: List of vectors. Scale factors of each Logit model. At least one element should be normalized to 1. If omitted, scale=1 for all positions is assumed.

functionality — Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.

- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

The function calculates the probability of a ranking as a product of Multinomial Logit models with gradually reducing availability, where scale differences can be allowed for.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": Not applicable (NA).
- "preprocess": Returns a list with pre-processed inputs, based on el_settings.
- "raw": Same as "estimate"
- "report": Choice overview across stages.
- "shares_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "utilities": List of vectors/matrices/arrays. Returns the utilities.

- "validate": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

---

apollo_emdc *MDC model with exogenous budget*

Description

Calculates the likelihood function of the MDC model with exogenous budget. Can also predict and validate inputs.

Usage

```
apollo_emdc(emdc_settings, functionality = "estimate")
```

Arguments

emdc_settings	List of settings for the model. It includes the following. • avail: Named list of numeric vectors. Availability of each product. Can also be called "A". • budget: Optional numeric vector. Budget. Must be bigger that the expenditure on all inside goods. Can also be called "B". • cost: Named list of numeric vectors. Price of each product. • delta: Lower triangular numeric matrix, or list of lists. Complementarity/substitution parameter. • continuousChoice: Named list of numeric vectors. Amount consumed of each inside good. Outside good must not be included. Can also be called "X". • gamma: Named list of numeric vectors. Satiation parameter of each product. • nRep: Scalar positive integer. Number of repetitions used when prediction • sigma: Numeric vector or scalar. Standard deviation of the error term. Default is one. • timeLimit: Positive scalar. Maximum amount of seconds the optimiser can spend calculating a prediction before setting it to NA. • tol: Positive scalar. Tolerance of the prediction algorithm. • utilities: Named list of numeric vectors (or matrices or arrays). Base utility of each product. Can also be called "V". • utilityOutside: Numeric vector (or matrix or array). Shadow price of the budget. Must be normalised to 0 for at least one individual. Default is 0 for every observation. Can also be called "V0".
functionality	Character. Either "validate", "zero_LL", "estimate", "conditionals", "raw", "output" or "prediction"

Details

This model extends the traditional multiple discrete-continuous (MDC) framework by (i) making the marginal utility of the outside good deterministic, and (ii) including complementarity and substitution in the model formulation. See the following paper for more details:

Palma, D. & Hess, S. (2022) Extending the Multiple Discrete Continuous (MDC) modelling framework to consider complementarity, substitution, and an unobserved budget. Transportation Research 161B, 13 - 35. https://doi.org/10.1016/j.trb.2022.04.005

Value

The returned object depends on the value of argument `functionality` as follows.

- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- `"prediction"`: List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

---

apollo_emdc1 *MDC model with exogenous budget*

Description

Calculates the likelihood function of the MDC model with exogenous budget. Can also predict and validate inputs.

Usage

```
apollo_emdc1(emdc_settings, functionality = "estimate")
```

Arguments

`emdc_settings` List of settings for the model. It includes the following.

- `avail`: Named list of numeric vectors. Availability of each product. Can also be called "A".
- `budget`: Numeric vector. Budget. Must be bigger that the expenditure on all inside goods. Can also be called "B".
- `continuousChoice`: Named list of numeric vectors. Amount consumed of each inside good. Outside good must not be included. Can also be called "X".

- `cost`: Named list of numeric vectors. Price of each product.
- `delta`: Lower triangular numeric matrix, or list of lists. Complementarity/substitution parameter.
- `gamma`: Named list of numeric vectors. Satiation parameter of each product.
- `nRep`: Scalar positive integer. Number of repetitions used when prediction
- `sigma`: Numeric vector or scalar. Standard deviation of the error term. Default is one.
- `timeLimit`: Positive scalar. Maximum amount of seconds the optimiser can spend calculating a prediction before setting it to NA.
- `tol`: Positive scalar. Tolerance of the prediction algorithm.
- `utilities`: Named list of numeric vectors (or matrices or arrays). Base utility of each product. Can also be called "V".
- `utilityOutside`: Numeric vector (or matrix or array). Shadow price of the budget. Must be normalised to 0 for at least one individual. Default is 0 for every observation. Can also be called "V0".

`functionality` Character. Either "validate", "zero_LL", "estimate", "conditionals", "raw", "output" or "prediction"

Details

This model extends the traditional multiple discrete-continuous (MDC) framework by (i) making the marginal utility of the outside good deterministic, and (ii) including complementarity and substitution in the model formulation. See the following paper for more details:

Palma, D. & Hess, S. (2022) Extending the Multiple Discrete Continuous (MDC) modelling framework to consider complementarity, substitution, and an unobserved budget. Transportation Reserarch 161B, 13 - 35. https://doi.org/10.1016/j.trb.2022.04.005

Value

The returned object depends on the value of argument `functionality` as follows.

- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- `"prediction"`: List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

apollo_emdc2 *Extended MDC*

Description

Calculates the likelihood function of the extended MDC model. Can also predict and validate inputs.

Usage

```
apollo_emdc2(emdc_settings, functionality = "estimate")
```

Arguments

emdc_settings	List of settings for the model. It includes the following. • avail: Named list of numeric vectors. Availability of each product. Can also be called "A". • continuousChoice: Named list of numeric vectors. Amount consumed of each inside good. Outside good must not be included. Can also be called "X". • cost: Named list of numeric vectors. Price of each product. • delta: Lower triangular numeric matrix, or list of lists. Complementarity/substitution parameter. • gamma: Named list of numeric vectors. Satiation parameter of each product. • sigma: Numeric scalar. Scale parameter. • nIter: Vector of two positive integers. Number of maximum iterations used during prediction, for the upper and lower iterative levels. • nRep: Scalar positive integer. Number of repetitions used when predictiong • rawPrediction: Scalar logical. When functionality is equal to "prediction", it returns the full set of simulations. Defaults is FALSE. • tolerance: Positive scalar Tolerance of the prediction algorithm. • utilities: Named list of numeric vectors (or matrices or arrays). Base utility of each product. Can also be called "V". • utilityOutside: Numeric vector (or matrix or array). Shadow price of the budget. Must be normalised to 0 for at least one individual. Default is 0 for every observation. Can also be called "V0".
functionality	Character. Either "validate", "zero_LL", "estimate", "conditionals", "raw", "output" or "prediction"

Details

This model extends the traditional multiple discrete-continuous (MDC) framework by (i) dropping the need to define a budget, (ii) making the marginal utility of the outside good deterministic, and (iii) including complementarity and substitution in the model formulation. See the following paper for more details:

Palma, D. & Hess, S. (2022) Extending the Multiple Discrete Continuous (MDC) modelling framework to consider complementarity, substitution, and an unobserved budget. Transportation Research 161B, 13 - 35. https://doi.org/10.1016/j.trb.2022.04.005

Value

The returned object depends on the value of argument `functionality` as follows.

- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- `"prediction"`: List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

---

apollo_estimate *Estimates model*

Description

Estimates a model using the likelihood function defined by `apollo_probabilities`.

Usage

```
apollo_estimate(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  estimate_settings = NA
)
```

Arguments

`apollo_beta`	Named numeric vector. Names and values for parameters.
`apollo_fixed`	Character vector. Names (as defined in `apollo_beta`) of parameters whose value should not change during estimation.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments:

- apollo_beta: Named numeric vector. Names and values of model parameters.
- apollo_inputs: List containing options of the model. See apollo_validateInputs.
- functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".

apollo_inputs List grouping most common inputs. Created by function apollo_validateInputs.

estimate_settings
List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- bgw_settings: List. Additional arguments to the BGW optimisation method. See bgw_mle for more details.
- bootstrapSE: Numeric. Number of bootstrap samples to calculate standard errors. Default is 0, meaning no bootstrap s.e. will be calculated. Number must zero or a positive integer. Only used if apollo_control$estMethod!="HB".
- bootstrapSeed: Numeric scalar (integer). Random number generator seed to generate the bootstrap samples. Only used if bootstrapSE>0. Default is 24.
- constraints: Character vector. Linear constraints on parameters to estimate. For example c('b1>0', 'b1 + 2*b2>1'). Only >, < and = can be used. Inequalities cannot be mixed with equality constraints, e.g. c(b1-b2=0, b2>0) will fail. All parameter names must be on the left side. Fixed parameters cannot go into constraints. Alternatively, constraints can be defined as in maxLik. Constraints can only be used with maximum likelihood estimation and the BFGS routine in particular.
- estimationRoutine: Character. Estimation method. Can take values "bfgs", "bgw", "bhhh", or "nr". Used only if apollo_control$HB is FALSE. Default is "bgw".
- hessianRoutine: Character. Name of routine used to calculate the Hessian of the log-likelihood function after estimation. Valid values are "analytic" (default), "numDeriv" (to use the numeric routine in package numDeric), "maxLik" (to use the numeric routine in packahe maxLik), and "none" to avoid calculating the Hessian and the covariance matrix. Only used if apollo_control$HB=FALSE.
- maxIterations: Numeric. Maximum number of iterations of the estimation routine before stopping. Used only if apollo_control$HB is FALSE. Default is 200.
- maxLik_settings: List. Additional settings for maxLik. See argument control in maxBFGS, maxBHHH and maxNM for more details. Only used for maximum likelihood estimation.
- numDeriv_method: Character. Method used for numerical differentiation when calculating the covariance matrix. Can be "Richardson" or "simple", Only used if analytic gradients are available. See argument method in grad for more details.
- numDeriv_settings: List. Additional arguments to the method used by numDeriv to calculate the Hessian. See argument method.args in grad for more details.

- `printLevel`: Higher values render more verbous outputs. Can take values 0, 1, 2 or 3. Ignored if apollo_control$HB is TRUE. Default is 3.
- `scaleAfterConvergence`: Logical. Used to increase numerical precision of convergence. If TRUE, parameters are scaled to 1 after convergence, and the estimation is repeated from this new starting values. Results are reported scaled back, so it is a transparent process for the user. Default is FALSE.
- `scaleHessian`: Logical. If TRUE, parameters are scaled to 1 for Hessian estimation. Default is FALSE.
- `scaling`: Named vector. Names of elements should match those in `apollo_beta`. Optional scaling for parameters. If provided, for each parameter `i`, `(apollo_beta[i]/scaling[i])` is optimised, but `scaling[i]*(apollo_beta[i]/scaling[i])` is used during estimation. For example, if parameter b3=10, while b1 and b2 are close to 1, then setting `scaling = c(b3=10)` can help estimation, specially the calculation of the Hessian. Reports will still be based on the non-scaled parameters.
- `silent`: Logical. If TRUE, no information is printed to the console during estimation. Default is FALSE.
- `validateGrad`: Logical. If TRUE, the analytical gradient (if used) is compared to the numerical one. Default is FALSE.
- `writeIter`: Logical. Writes value of the parameters in each iteration to a csv file. Works only if `estimation_routine=="bfgs"|"bgw"`. Default is TRUE.

Details

This is the main function of the Apollo package. The estimation process begins by running a number of checks on the `apollo_probabilities` function provided by the user. If all checks are passed, estimation begins. There is no limit to estimation time other than reaching the maximum number of iterations. If Bayesian estimation is used, estimation will finish once the predefined number of iterations are completed. By default, this functions writes the estimated parameter values in each iteration to a file in the working/output directory. Writing can be turned off by setting `estimate_settings$writeIter` to `FALSE`. By default, **final results are not written into a file nor printed to the console**, so users must make sure to call function apollo_modelOutput and/or apollo_saveOutput afterwards. Users are strongly encouraged to visit `https://www.ApolloChoiceModelling.com/` to download examples on how to use the Apollo package. The webpage also provides a detailed manual for the package, as well as a user-group to get further help.

Value

model object

`apollo_estimateHB` *Estimates model using Bayesian estimation*

Description

Estimates a model using Bayesian estimation on the likelihood function defined by `apollo_probabilities`.

Usage

```
apollo_estimateHB(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  estimate_settings = NA
)
```

Arguments

`apollo_beta`	Named numeric vector. Names and values for parameters.
`apollo_fixed`	Character vector. Names (as defined in `apollo_beta`) of parameters whose value should not change during estimation.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • `apollo_beta`: Named numeric vector. Names and values of model parameters. • `apollo_inputs`: List containing options of the model. See apollo_validateInputs. • `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`estimate_settings`	List. Options controlling the estimation process, as used for in apollo_estimate.

Details

This is a sub function of apollo_estimate which is called when using Bayesian estimation.

Value

model object

apollo_expandLoop *Expands loops in a function or expression*

Description

Expands loops replacing the index by its value. It also evaluates `paste` and `paste0`, and removes `get`.

Usage

```
apollo_expandLoop(f, apollo_inputs, validate = TRUE)
```

Arguments

`f`	function (usually `apollo_probabilities`) inside which the name of the components are inserted.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`validate`	Logical. If TRUE, the new function will be validated before being returned

Details

For example, the expression `for(j in 1:3) V[[paste0('alt',j)]] = b1*get(paste0('x',j)) + b2*X[,j]`

would be expanded into:

`V[[alt1]] = b1*x1 + b2*X[,1] V[[alt2]] = b1*x2 + b2*X[,2] V[[alt3]] = b1*x3 + b2*X[,3]`

Value

A function or an expression (same type as input `f`)

apollo_firstRow *Keeps only the first row for each individual*

Description

Given a multi-row input, keeps only the first row for each individual.

Usage

```
apollo_firstRow(P, apollo_inputs)
```

Arguments

`P`	List of vectors, matrices or 3-dim arrays. Likelihood of the model components (or other object).
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.

Details

This a function to keep only the first row of an object per indidividual. It can handle multiple types of components, including scalars, vectors and three-dimensional arrays (cubes). The argument database MUST contain a column called 'apollo_sequence', which is created by apollo_validateData.

Value

If P is a list, then it returns a list where each element has only the first row of each individual. If P is a single element, then it returns a single element with only the first row of each individual. The size of the element is changed only in the first dimension. If input is a scalar, then it returns a vector with the element repeated as many times as individuals in database. If the element is a vector, its length will be changed to the number of individuals. If the element is a matrix, then its first dimension will be changed to the number of individuals, while keeping the size of the second dimension. If the element is a cube, then only the first dimension's length is changed, preserving the others.

---

apollo_fitsTest *Compares log-likelihood of model across categories*

Description

Given the estimates of a model, it compares the log-likelihood at the observation level across categories of observations.

Usage

```
apollo_fitsTest(model, apollo_probabilities, apollo_inputs, fitsTest_settings)
```

Arguments

model	Model object. Estimated model object as returned by function apollo_estimate.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
fitsTest_settings	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • **subsamples**: Named list of boolean vectors. Each element of the list defines whether a given observation belongs to a given subsample (e.g. by sociodemographics).

Details

Prints a table comparing the average log-likelihood at the observation level for each category.

Value

Matrix with average log-likelihood at observation level per category (invisibly).

`apollo_fmnl` *Calculates Fractional Multinomial Logit probabilities*

Description

Calculates the probabilities of a Fractional Multinomial Logit model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_fmnl(fmnl_settings, functionality)
```

Arguments

`fmnl_settings` List of inputs of the FMNL model. It should contain the following.

- `alternatives`: Character vector. Names of alternatives, elements must match the names in list 'utilities'.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.
- `choiceShares`: Named list of numeric vectors. Share allocated to each alternative. One element per alternative, as long as the number of observations or a scalar. Names must match those in `alternatives`.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.
- `utilities`: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in `alternatives`.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on fmnl_settings.
- "raw": Same as "prediction"
- "report": Overview of dependent variable
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "utilities": List of vectors/matrices/arrays. Returns the utilities.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

apollo_fnl *Calculates Fractional Nested Logit probabilities*

Description

Calculates the probabilities of a Fractional Nested Logit (FNL) model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_fnl(fnl_settings, functionality)
```

Arguments

`fnl_settings` List of inputs of the FNL model. It should contain the following.

- `alternatives`: Character vector. Names of alternatives, elements must match the names in list 'utilities'.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.
- `choiceShares`: Named list of numeric vectors. Share allocated to each alternative. One element per alternative, as long as the number of observations or a scalar. Names must match those in `alternatives`.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `nlNests`: List of numeric scalars or vectors. Lambda parameters for each nest. Elements must be named with the nest name. The lambda at the root is automatically fixed to 1 if not provided by the user.
- `nlStructure`: Named list of character vectors. As many elements as nests, it must include the "root". Each element contains the names of the nests or alternatives that belong to it. Element names must match those in `nlNests`.
- `utilities`: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in `alternatives`.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

In this implementation of the Nested Logit model, each nest must have a lambda parameter associated to it. For the model to be consistent with utility maximisation, the estimated value of the Lambda parameter of all nests should be between 0 and 1. Lambda parameters are inversely proportional to the correlation between the error terms of alternatives in a nest. If lambda=1, then there is no relevant correlation between the unobserved utility of alternatives in that nest. The tree must contain an upper nest called "root". The lambda parameter of the root is automatically set to 1 if not specified in nlNests, but can be changed by the user if desired (though not advised).

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": Not implemented.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.

- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on fnl_settings.
- "raw": Same as "prediction"
- "report": List with tree structure and choice overview.
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "utilities": List of vectors/matrices/arrays. Returns the utilities.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

---

apollo_initialise *Prepares environment*

Description

Prepares environment (the global environment if called by the user) for model definition and estimation.

Usage

```
apollo_initialise()
```

Details

This function detaches variables and makes sure that output is directed to console. It does not delete variables from the working environment.

Value

Nothing.

apollo_insertComponentName *Adds componentName2 to model calls*

Description

Adds componentName2 to model calls

Usage

```
apollo_insertComponentName(e)
```

Arguments

e	An expression or a function. It will usually be apollo_probabilities.

Value

The original argument 'e' but modified to incorporate a new setting called 'componentName2' to every call to apollo_<model> (e.g. apollo_mnl, apollo_nl, etc.).

apollo_insertFunc *Modifies function to make it compatible with analytic gradients*

Description

Takes a likelihood function and inserts `function ()` before key elements to allow for analytic gradient calculation

Usage

```
apollo_insertFunc(f, like = TRUE, randCoeff = FALSE, lcPars = FALSE)
```

Arguments

f	Function. Expressions inside it will be turned into functions. Usually `apollo_probabilities` or `apollo_randCoeff`.
like	Logical. Must be TRUE if `f` is `apollo_probabilities`. FALSE otherwise.
randCoeff	Logical. Must be TRUE if `f` is `apollo_randCoeff`. FALSE otherwise.
lcPars	Logical. Must be TRUE if `f` is `apollo_lcPars`. FALSE otherwise.

Details

It modifies the definition of the following models.

- apollo_mnl: Turns all elements inside mnl_settings$V into functions.
- apollo_ol: Turns ol_settings$V and all elements inside ol_settings$tau into functions.
- apollo_op: Turns op_settings$V and all elements inside op_settings$tau into functions.
- apollo_normalDensity: Turns normalDensity_settings$xNormal, normalDensity_settings$mu and normalDensity_settings$sigma into functions.

It can only track a maximum of 3 levels of depth in definitions. For example: V <- list() V[["A"]] <- b1*x1A + b2*x2A V[["B"]] <- b1*x1B + b2*x2B mnl_settings1 <- list(alternatives=c("A", "B"), V = V, choiceVar= Y, avail = 1, componentName="MNL1") P[["MNL1"]] <- apollo_mnl(mnl_settings1, functionality) But it may not be able to deal with the following: VA <- b1*x1A + b2*x2A V <- list() V[["A"]] <- VA V[["B"]] <- b1*x1B + b2*x2B mnl_settings1 <- list(alternatives=c("A", "B"), V = V, choiceVar= Y, avail = 1, componentName="MNL1") P[["MNL1"]] <- apollo_mnl(mnl_settings1, functionality) But that might be enough given how apollo_dVdB works.

Value

Function f but with relevant expressions turned into function definitions.

---

apollo_insertOLList *Replaces* tau=c(...) *by* tau=list(...) *in calls to* apollo_ol

---

Description

Takes a function, looks for calls to apollo_ol, identifies the corresponding ol_settings, then goes inside the definition of ol_settings and replaces tau=c(...) for tau=list(...).

Usage

```
apollo_insertOLList(f)
```

Arguments

f	Function. Usually apollo_probabilities, apollo_randCoeff, or apollo_lcPars.

Details

This only goes one level deep in definitions. For example, it will work correctly in the following cases: ol_settings = list(outcomeOrdered = y1, V = b1*x1, tau = c(tau11, tau12)) P[["OL1"]] = apollo_ol(ol_settings, functionality) P[["OL2"]] = apollo_ol(list(outcomeOrdered=y2, V=b2*x2, tau=c(tau21, tau22)), functionality) But it will not work on the following cases: Tau = c(tau1, tau2, tau3) ol_settings = list(outcomeOrdered = y2, V = b2*x2, tau = Tau) P[["OL1"]] = apollo_ol(ol_settings, functionality) P[["OL2"]] = apollo_ol(list(outcomeOrdered=y1, V=b1*x1, tau=Tau), functionality)

This function is called by apollo_modifyUserDefFunc to allow for analytical gradients when using apollo_ol.

Value

Function f with tau=c(...) replaced by tau=list(...).

---

apollo_insertRows *Inserts rows*

Description

Given a numeric object (scalar, vector, matrix or 3-dim array) inserts rows in the specified places.

Usage

```
apollo_insertRows(v, r, val)
```

Arguments

v	Numeric scalar, vector, matrix or 3-dim array.
r	Boolean vector. TRUE for inserting a row from v, FALSE to insert a new row with value val.
val	Numeric scalar. Value that will fill new rows.

Details

In general, r should be longer than the number of rows in utilities, and sum(r)=nrow(v). If not, then a new object with as many rows as r will be returned. Old rows will be taken from utilities from the top down.

Value

The same argument v but with rows added where r==FALSE.

---

apollo_insertRRMQuotes *Introduces quotes into rrm_settings*

Description

Takes a function, looks for the definition of relevant parts of rrm_settings, and introduces quotes on them. This is to facilitate their processing by apollo_rrm under functionality="preprocessing".

Usage

```
apollo_insertRRMQuotes(f)
```

Arguments

f	Function. Usually apollo_probabilities.

Value

Function f with relevant expressions turned into character.

---

apollo_insertScaling *Scales variables inside a function*

Description

It changes the syntax of the function by replacing variable names for their scaled form, e.g. x –> x*apollo_inputs$apollo_scale[["x"]]. In assignments, it only scales the right side of the assignment.

Usage

```
apollo_insertScaling(e, sca)
```

Arguments

e	Function, expression, call or symbol to alter.
sca	Named numeric vector with the scales. The names in these vectors determine which variables should be scaled.

Value

A function, expression, call or symbol with the corresponding variables scaled.

---

apollo_keepRows *Keeps only some rows*

Description

Given a numeric object (scalar, vector, matrix or 3-dim array) keeps only the specified rows.

Usage

```
apollo_keepRows(v, r)
```

Arguments

v	Numeric scalar, vector, matrix or 3-dim array.
r	Boolean vector. As many elements as rows in utilities. TRUE for keeping the row. FALSE to drop it.

Value

The same argument `utilities` but with the rows where `r==FALSE` removed.

`apollo_lc` *Calculates the likelihood of a latent class model*

Description

Given within class probabilities, and class allocation probabilities, calculates the probabilities of an Exploded Logit model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_lc(lc_settings, apollo_inputs, functionality)
```

Arguments

`lc_settings`	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • **classProb**: List of probabilities. Allocation probability for each class. One element per class, in the same order as `inClassProb`. • **componentName**: Character. Name given to model component (optional). • **inClassProb**: List of probabilities. Conditional likelihood for each class. One element per class, in the same order as `classProb`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`functionality`	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are: • `"components"`: For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • `"conditionals"`: For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • `"estimate"`: For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • `"gradient"`: For model estimation, produces analytical gradients of the likelihood, where possible. • `"output"`: Prepares output for post-estimation reporting. • `"prediction"`: For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws. • `"preprocess"`: Prepares likelihood functions for use in estimation.

- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "utilities": Returns utilities at provided parameter values.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Returns nothing.
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all models components, for each class.
- "preprocess": Returns a list with pre-processed inputs, based on lc_settings.
- "raw": Same as "prediction"
- "report": Class allocation overview.
- "shares_LL": Same as "estimate"
- "utilities": List of vectors/matrices/arrays. Returns a list with the utilities for all models components, for each class.
- "validate": Same as "estimate", but also runs a set of tests on the given arguments.
- "zero_LL": Same as "estimate"

---

apollo_lcConditionals *Calculates conditionals for latent class models.*

Description

Calculates posterior expected values (conditionals) of class allocation probabilities for each individual.

Usage

```
apollo_lcConditionals(model, apollo_probabilities, apollo_inputs)
```

Arguments

`model`	Model object. Estimated model object as returned by function apollo_estimate.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • `apollo_beta`: Named numeric vector. Names and values of model parameters. • `apollo_inputs`: List containing options of the model. See apollo_validateInputs. • `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.

Details

This function can only be used with latent class models without continuous heterogeneity.

Value

A matrix with the posterior class allocation probabilities for each individual.

---

apollo_lcEM *Uses EM for latent class model*

Description

Uses the EM algorithm for estimating a latent class model.

Usage

```
apollo_lcEM(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  lcEM_settings = NA,
  estimate_settings = NA
)
```

Arguments

`apollo_beta`	Named numeric vector. Names and values for parameters.
`apollo_fixed`	Character vector. Names (as defined in `apollo_beta`) of parameters whose value should not change during estimation.

`apollo_probabilities`
: Function. Returns probabilities of the model to be estimated. Must receive three arguments:

  - `apollo_beta`: Named numeric vector. Names and values of model parameters.
  - `apollo_inputs`: List containing options of the model. See apollo_validateInputs.
  - `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.

`apollo_inputs`
: List grouping most common inputs. Created by function apollo_validateInputs.

`lcEM_settings`
: List. Options controlling the EM process.

  - **EMmaxIterations**: Numeric. Maximum number of iterations of the EM algorithm before stopping. Default is 100.
  - **postEM**: Numeric scalar. Determines the tasks performed by this function after the EM algorithm has converged. Can take values `0`, `1` or `2` only. If value is `0`, only the EM algorithm will be performed, and the results will be a model object without a covariance matrix (i.e. estimates only). If value is `1`, after the EM algorithm, the covariance matrix of the model will be calculated as well, and the result will be a model object with a covariance matrix. If value is `2`, after the EM algorithm, the estimated parameter values will be used as starting value for a maximum likelihood estimation process, which will render a model object with a covariance matrix. Performing maximum likelihood estimation after the EM algorithm is useful, as there may be room for further improvement. Default is `2`.
  - **silent**: Boolean. If TRUE, no information is printed to the console during estimation. Default is FALSE.
  - **stoppingCriterion**: Numeric. Convergence criterion. The EM process will stop when improvements in the log-likelihood fall below this value. Default is 10^-5.

`estimate_settings`
: List. Options controlling the estimation process within each EM iteration. See apollo_estimate for details.

Details

This function uses the EM algorithm for estimating a Latent Class model. It is only suitable for models without continuous mixing. All parameters need to vary across classes and need to be included in the `apollo_lcPars` function which is used by `apollo_lcEM`.

Value

model object

apollo_lcEM_new *Uses EM for latent class model*

Description

Uses the EM algorithm for estimating a latent class model.

Usage

```
apollo_lcEM_new(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  lcEM_settings = NA,
  estimate_settings = NA
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
lcEM_settings	List. Options controlling the EM process. • **EMmaxIterations**: Numeric. Maximum number of iterations of the EM algorithm before stopping. Default is 100. • **postEM**: Numeric scalar. Determines the tasks performed by this function after the EM algorithm has converged. Can take values 0, 1 or 2 only. If value is 0, only the EM algorithm will be performed, and the results will be a model object without a covariance matrix (i.e. estimates only). If value is 1, after the EM algorithm, the covariance matrix of the model will be calculated as well, and the result will be a model object with a covariance matrix. If value is 2, after the EM algorithm, the estimated parameter values will be used as starting value for a maximum likelihood estimation process, which will render a model object with a covariance matrix. Performing

maximum likelihood estimation after the EM algorithm is useful, as there may be room for further improvement. Default is 2.

- **silent**: Boolean. If TRUE, no information is printed to the console during estimation. Default is FALSE.
- **stoppingCriterion**: Numeric. Convergence criterion. The EM process will stop when improvements in the log-likelihood fall below this value. Default is 10^-5.

`estimate_settings`
: List. Options controlling the estimation process within each EM iteration. See apollo_estimate for details.

Details

This function uses the EM algorithm for estimating a Latent Class model. It is only suitable for models without continuous mixing. All parameters need to vary across classes and need to be included in the `apollo_lcPars` function which is used by `apollo_lcEM`.

Value

model object

`apollo_lcUnconditionals`
: *Returns unconditionals for a latent class model model*

Description

Returns values for random parameters and class allocation probabilities in a latent class model model.

Usage

```
apollo_lcUnconditionals(model, apollo_probabilities, apollo_inputs)
```

Arguments

`model`
: Model object. Estimated model object as returned by function apollo_estimate.

`apollo_probabilities`
: Function. Returns probabilities of the model to be estimated. Must receive three arguments:

- `apollo_beta`: Named numeric vector. Names and values of model parameters.
- `apollo_inputs`: List containing options of the model. See apollo_validateInputs.
- `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.

`apollo_inputs`
: List grouping most common inputs. Created by function apollo_validateInputs.

Value

List of object, one per random component and one for the class allocation probabilities.

`apollo_llCalc` *Calculates log-likelihood of all model components*

Description

Calculates the log-likelihood of each model component as well as the whole model.

Usage

```
apollo_llCalc(apollo_beta, apollo_probabilities, apollo_inputs, silent = FALSE)
```

Arguments

`apollo_beta`	Named numeric vector. Names and values for parameters.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • `apollo_beta`: Named numeric vector. Names and values of model parameters. • `apollo_inputs`: List containing options of the model. See apollo_validateInputs. • `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`silent`	Boolean. If TRUE, no information is printed to the console by the function. Default is FALSE.

Details

This function calls apollo_probabilities with functionality="output". It then reorders the list of likelihoods so that "model" goes first.

Value

A list of vectors. Each vector corresponds to the log-likelihood of the whole model (first element) or a model component.

apollo_loadModel *Loads model from file*

Description

Loads a previously estimated model object from a file.

Usage

```
apollo_loadModel(modelName)
```

Arguments

modelName	Character. Name of the model to load.

Details

This function looks for a file named `modelName_model.rds` in the working or output directory, loads the object contained in it, and returns it.

Value

A model object.

apollo_longToWide *Converts data from long to wide format.*

Description

Converts choice data from long to wide format, with one row per observation as opposed to one row per alternative/observation.

Usage

```
apollo_longToWide(longData, longToWide_settings)
```

Arguments

longData	data.frame. Data in long format.
longToWide_settings	List. Contains settings for this function. User input is required for all settings.

- `altColumn`: Character. Name of column in long data that contains the names of the alternatives (either numeric or character).
- `altSpecAtts`: Character vector. Names of columns in long data with attributes that vary across alternatives within an observation.

- `choiceColumn`: Character. Name of column in long data that contains the choice.
- `idColumn`: Character. Name of column in long data that contains the ID of individuals.
- `obsColumn`: Character. Name of column in long data that contains the observation index.

Value

Silently returns a data.frame with the wide format version of the data. An overview of the data is printed to screen.

`apollo_lrTest` *Likelihood ratio test*

Description

Calculates the likelihood ratio test value between two models and reports the corresponding p-value.

Usage

```
apollo_lrTest(model1, model2)
```

Arguments

`model1`	Either a character variable with the name of a previously estimated model, or an estimated model in memory, as returned by apollo_estimate.
`model2`	Either a character variable with the name of a previously estimated model, or an estimated model in memory, as returned by apollo_estimate.

Details

The two models need to have been estimated on the same data, and one model needs to be nested within the other model.

Value

LR-test p-value (invisibly)

apollo_makeCluster *Creates cluster for estimation.*

Description

Splits data, creates cluster and loads different pieces of the database on each worker.

Usage

```
apollo_makeCluster(
  apollo_probabilities,
  apollo_inputs,
  silent = FALSE,
  cleanMemory = FALSE
)
```

Arguments

apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
silent	Boolean. If TRUE, no messages are printed to the terminal. FALSE by default. It overrides apollo_inputs$silent.
cleanMemory	Boolean. If TRUE, it saves apollo_inputs to disc, and removes database and draws from the apollo_inputs in .GlobalEnv and the parent environment.

Details

Internal use only. Called by apollo_estimate before estimation. Using multiple cores greatly increases memory consumption.

Value

Cluster (i.e. an object of class cluster from package parallel)

`apollo_makeDraws` *Creates draws for models with mixing*

Description

Creates a list containing all draws necessary to estimate a model with mixing.

Usage

```
apollo_makeDraws(apollo_inputs, silent = FALSE)
```

Arguments

`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`silent`	Boolean. If true, then no information is printed to console or default output. FALSE by default.

Details

Internal use only. Called by `apollo_validateInputs`. This function creates a list whose elements are the sets of draws requested by the user for use in a model with mixing. If the model does not include mixing, then it is not necessary to run this function. The number of draws has a massive impact on memory usage and estimation time. Memory usage and number of computations scale geometrically as N*interNDraws*intraNDraws (where N is the number of observations). Special care should be taken when using both inter and intra-individual draws, as memory usage can easily reach the GB order of magnitude. Also, keep in mind that using several threads (i.e. multicore) at least doubles the memory usage. This function returns a list, with each element representing a random component of the mixing model. The dimensions of the array depend on the type of draws used.

1. If only inter-individual draws are used, then draws are stored as 2-dimensional arrays (i.e. matrices).
2. If intra-individual draws are used, then draws are stored as 3-dimensional arrays.
3. The first dimension of the arrays (rows) correspond with the observations in the database.
4. The second dimension of the arrays (columns) correspond to the number of inter-individual draws.
5. The third dimension of the arrays correspond to the number of intra-individual draws.

Value

List. Each element is an array of draws representing a random component of the mixing model.

apollo_makeGrad *Creates gradient function.*

Description

Creates gradient function from the likelihood function apollo_probabilities provided by the user. Returns NULL if the creation of gradient function fails.

Usage

```
apollo_makeGrad(
  apollo_beta,
  apollo_fixed,
  apollo_logLike,
  validateGrad = FALSE
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_logLike	Function to calculate the log-likelihood of the model, as created by apollo_makeLogLike If provided, the value of the analytical gradient will be compared to the value of the numerical gradient as calculated using apollo_logLike and the numDeriv package. If the difference between the two is bigger than 1 that the analytical gradient is wrong and NULL will be returned.
validateGrad	Logical. If TRUE, it compares the value of the analytical gradient evaluated at apollo_beta against the numeric gradient (using numDeriv) at the same value. If the difference is bigger than 1 return NULL.

Details

Internal use only. Called by apollo_estimate before estimation. The returned function can be single-threaded or multi-threaded based on the model options.

Value

apollo_gradient function. It receives the following arguments

- b Numeric vector of _variable_ parameters (i.e. must not include fixed parameters).
- countIter Not used. Included only to mirror inputs of apollo_logLike.
- getNIter Not used. Included only to mirror inputs of apollo_logLike.
- sumLL Not used. Included only to mirror inputs of apollo_logLike.
- writeIter Not used. Included only to mirror inputs of apollo_logLike.

If the creation of the gradient function fails, then it returns NULL.

apollo_makeHessian *Creates hessian function.*

Description

Creates hessian function from the likelihood function apollo_probabilities provided by the user. Returns NULL if the creation of gradient function fails.

Usage

```
apollo_makeHessian(apollo_beta, apollo_fixed, apollo_logLike)
```

Arguments

apollo_beta	Named numeric vector. Names and values for (all) parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_logLike	Function to calculate the log-likelihood of the model, as created by apollo_makeLogLike If provided, the value of the analytical gradient will be compared to the value of the numerical gradient as calculated using apollo_logLike and the numDeriv package. If the difference between the two is bigger than 1 that the analytical gradient is wrong and NULL will be returned.

Details

Internal use only. Called by apollo_estimate before estimation. The returned function can be single-threaded or multi-threaded based on the model options.

Value

apollo_hessian function. It receives a single argument called b, which are the _variable_ parameters (i.e. must not include fixed parameters).

apollo_makeLogLike *Creates log-likelihood function.*

Description

Creates log-likelihood function from the likelihood function apollo_probabilities provided by the user.

Usage

```
apollo_makeLogLike(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  apollo_estSet = list(estimationRoutine = "bgw"),
  cleanMemory = FALSE
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
apollo_estSet	List of estimation options. It must contain at least one element called estimationRoutine defining the estimation algorithm. See apollo_estimate.
cleanMemory	Logical. If TRUE, then apollo_inputs$draws and apollo_inputs$database are erased throughout the calling stack. Used to reduce memory usage in case of multithreading and a large database or number o draws.

Details

Internal use only. Called by apollo_estimate before estimation. The returned function can be single-threaded or multi-threaded based on the model options.

Value

apollo_logLike function.

apollo_mdcev *Calculates MDCEV likelihoods*

Description

Calculates the likelihoods of a Multiple Discrete Continuous Extreme Value (MDCEV) model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_mdcev(mdcev_settings, functionality)
```

Arguments

`mdcev_settings` List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `alpha`: Named list. Alpha parameters for each alternative, including for any outside good. As many elements as alternatives.
- `alternatives`: Character vector. Names of alternatives, elements must match the names in list 'utilities'.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability.
- `budget`: Numeric vector. Budget for each observation.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `continuousChoice`: Named list of numeric vectors. Amount of consumption of each alternative. One element per alternative, as long as the number of observations or a scalar. Names must match those in `alternatives`.
- `cost`: Named list of numeric vectors. Price of each alternative. One element per alternative, each one as long as the number of observations or a scalar. Names must match those in `alternatives`.
- `gamma`: Named list. Gamma parameters for each alternative, excluding any outside good. As many elements as inside good alternatives.
- `nRep`: Numeric scalar. Number of simulations of the whole dataset used for forecasting. The forecast is the average of these simulations. Default is 100.
- `outside`: Character. Optional name of the outside good.
- `rawPrediction`: Logical scalar. TRUE for prediction to be returned at the draw level (a 3-dim array). FALSE for prediction to be returned averaged across draws. Default is FALSE.

- rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).
- sigma: Numeric scalar. Scale parameter of the model extreme value type I error.
- utilities: Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'.

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the observed consumption for each observation.
- "gradient": Not implemented

- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": A matrix with one row per observation, and columns indicating means and s.d. of continuous and discrete predicted consumptions.
- "preprocess": Returns a list with pre-processed inputs, based on mdcev_settings.
- "raw": Same as "estimate"
- "report": Dependent variable overview.
- "shares_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.

---

apollo_mdcev2 *Calculates MDCEV likelihoods*

Description

Calculates the likelihoods of a Multiple Discrete Continuous Extreme Value (MDCEV) model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_mdcev2(mdcev_settings, functionality)
```

Arguments

mdcev_settings List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- alpha: Named list. Alpha parameters for each alternative, including for any outside good. As many elements as alternatives.
- alternatives: Character vector. Names of alternatives, elements must match the names in list 'utilities'.
- avail: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in alternatives. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify avail=1 to indicate universal availability, or omit the setting completely.
- budget: Numeric vector. Budget for each observation.
- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- continuousChoice: Named list of numeric vectors. Amount of consumption of each alternative. One element per alternative, as long as the number of observations or a scalar. Names must match those in alternatives.

- cost: Named list of numeric vectors. Price of each alternative. One element per alternative, each one as long as the number of observations or a scalar. Names must match those in alternatives.
- fastPred: Boolean scalar. TRUE to mix parameter draws with repetition draws. This is formally incorrect, but a good a approximation to the true prediction, and much faster. FALSE by default.
- gamma: Named list. Gamma parameters for each alternative, excluding any outside good. As many elements as inside good alternatives.
- nRep: Numeric scalar. Number of simulations of the whole dataset used for forecasting. The forecast is the average of these simulations. Default is 100.
- outside: Character. Optional name of the outside good.
- rawPrediction: Logical scalar. TRUE for prediction to be returned at the draw level (a 3-dim array). FALSE for prediction to be returned averaged across draws. Default is FALSE.
- rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).
- sigma: Numeric scalar. Scale parameter of the model extreme value type I error.
- utilities: Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'.

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.

- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the observed consumption for each observation.
- "gradient": Not implemented
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": A matrix with one row per observation, and columns indicating means and s.d. of continuous and discrete predicted consumptions.
- "preprocess": Returns a list with pre-processed inputs, based on mdcev_settings.
- "raw": Same as "estimate"
- "report": Dependent variable overview.
- "shares_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.

---

apollo_mdcnev *Calculates MDCNEV likelihoods*

Description

Calculates the likelihoods of a Multiple Discrete Continuous Nested Extreme Value (MDCNEV) model with an outside good and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_mdcnev(mdcnev_settings, functionality)
```

Arguments

`mdcnev_settings`

List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `alpha`: Named list. Alpha parameters for each alternative, including for the outside good. As many elements as alternatives.
- `alternatives`: Character vector. Names of alternatives, elements must match the names in list 'utilities'.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.
- `budget`: Numeric vector. Budget for each observation.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `continuousChoice`: Named list of numeric vectors. Amount of consumption of each alternative. One element per alternative, as long as the number of observations or a scalar. Names must match those in `alternatives`.
- `cost`: Named list of numeric vectors. Price of each alternative. One element per alternative, each one as long as the number of observations or a scalar. Names must match those in `alternatives`.
- `gamma`: Named list. Gamma parameters for each alternative, including for the outside good. As many elements as alternatives.
- `mdcnevNests`: Named list. Lambda parameters for each nest. Elements must be named with the nest name. The lambda at the root is fixed to 1, and therefore must be no be defined. The value of the estimated mdcnevNests parameters should be between 0 and 1 to ensure consistency with random utility maximization.
- `mdcnevStructure`: Numeric matrix. One row per nest and one column per alternative. Each element of the matrix is 1 if an alternative belongs to the corresponding nest.
- `outside`: Character. Alternative name for the outside good. Default is "outside"
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.
- `utilities`: Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the observed consumption for each observation.
- "gradient": Not implemented
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": A matrix with one row per observation, and columns indicating means and s.d. of continuous and discrete predicted consumptions.
- "preprocess": Returns a list with pre-processed inputs, based on mdcnev_settings.
- "raw": Same as "estimate"
- "report": Dependent variable overview.
- "shares_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.

apollo_mixConditionals *Calculates conditionals for continuous mixture models*

Description

Calculates posterior expected values (conditionals) of continuously distributed random coefficients, as well as their standard deviations.

Usage

```
apollo_mixConditionals(model, apollo_probabilities, apollo_inputs)
```

Arguments

model	Model object. Estimated model object as returned by function apollo_estimate.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.

Details

This functions is only meant for use with continuous distributions

Value

List of matrices. Each matrix has dimensions nIndiv x 3. One matrix per random component. Each row of each matrix contains the indivID of an individual, and the posterior mean and s.d. of this random component for this individual

apollo_mixEM *Uses EM for models with continuous random coefficients*

Description

Uses the EM algorithm for estimating a model with continuous random coefficients.

Usage

```
apollo_mixEM(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  mixEM_settings = NA,
  estimate_settings = NA
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters. These need to be provided in the following order. With K random parameters, K means for the underlying Normals, followed by the elements of the lower triangle of the Cholesky matrix, by row.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
mixEM_settings	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • **EMmaxIterations**: Numeric. Maximum number of iterations of the EM algorithm before stopping. Default is 100. • **postEM**: Numeric scalar. Determines the tasks performed by this function after the EM algorithm has converged. Can take values 0, 1 or 2 only. If value is 0, only the EM algorithm will be performed, and the results will be a model object without a covariance matrix (i.e. estimates only). If value is 1, after the EM algorithm, the covariance matrix of the model will be

calculated as well, and the result will be a model object with a covariance matrix. If value is 2, after the EM algorithm, the estimated parameter values will be used as starting value for a maximum likelihood estimation process, which will render a model object with a covariance matrix. Performing maximum likelihood estimation after the EM algorithm is useful, as there may be room for further improvement. Default is 2.

- **silent**: Boolean. If TRUE, no information is printed to the console during estimation. Default is FALSE.
- **stoppingCriterion**: Numeric. Convergence criterion. The EM process will stop when improvements in the log-likelihood fall below this value. Default is 10^-5.
- **transforms**: List. Optional argument, with one entry per parameter, showing the inverse transform to return from beta to the underlying Normal. E.g. if the first parameter is specified as negative logormal inside apollo_randCoeff, then the entry in transforms should be transforms[[1]]=function(x) log(-x)

`estimate_settings`
: List. Options controlling the estimation process within each EM iteration. See apollo_estimate for details.

Details

This function uses the EM algorithm for estimating a model with continuous random coefficients. It is only suitable for models where all parameters are random, with a full covariance matrix. All random parameters need to be based on underlying Normals with a full covariance matrix, but any transform thereof can be used.

Value

model object

`apollo_mixUnconditionals`
: *Returns draws for continuously distributed random parameters in mixture model*

Description

Returns draws (unconditionals) for random parameters in model, including interactions with deterministic covariates.

Usage

```
apollo_mixUnconditionals(
  model,
  apollo_probabilities,
  apollo_inputs,
  obsLevel = FALSE
)
```

Arguments

`model`	Model object. Estimated model object as returned by function apollo_estimate.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • `apollo_beta`: Named numeric vector. Names and values of model parameters. • `apollo_inputs`: List containing options of the model. See apollo_validateInputs. • `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`obsLevel`	Logical. If `TRUE`, unconditionals are returned at the observation level rather than person level. This setting only applies to continuous mixture models and is set to TRUE by default only in the presence of intra-individual draws. Otherwise, the default is FALSE.

Details

This functions is only meant for use with continuous distributions

Value

List of object, one per random coefficient. With inter-individual draws only, this will be a matrix, with one row per individual, and one column per draw. With intra-individual draws, this will be a three-dimensional array, with one row per observation, inter-individual draws in the second dimension, and intra-individual draws in the third dimension.

`apollo_mlhs` *Generate random draws using MLHS algorithm*

Description

Generate random draws using the Modified Latin Hypercube Sampling algorithm.

Usage

```
apollo_mlhs(N, d, i)
```

Arguments

`N`	Numeric. The number of draws to generate in each dimension
`d`	Numeric. The number of dimensions to generate draws in
`i`	Numeric. The number of individuals to generate draws for

Details

Internal use only. Algorithm described in Hess, S., Train, K., and Polak, J. (2006) Transportation Research Part B, 40, 147 - 163.

Value

A (N*i) x d matrix with random draws

---

apollo_mnl *Calculates Multinomial Logit probabilities*

Description

Calculates the probabilities of a Multinomial Logit model and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_mnl(mnl_settings, functionality)
```

Arguments

`mnl_settings` List of inputs of the MNL model. It should contain the following.

- `alternatives`: Named numeric vector. Names of alternatives and their corresponding value in `choiceVar`.
- `avail`: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in `alternatives`. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify `avail=1` to indicate universal availability, or omit the setting completely.
- `choiceVar`: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`. Set to `"all"` by default if omitted.
- `utilities`: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in `alternatives`.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on mnl_settings.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "raw": Same as "prediction"
- "report": Choice overview
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "utilities": List of vectors/matrices/arrays. Returns the utilities.

- "validate": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

---

apollo_modeChoiceData *Simulated dataset of mode choice.*

Description

A simulated dataset containing 8,000 mode choices among four alternatives.

Usage

```
apollo_modeChoiceData
```

Format

A data.frame with 8,000 rows and 25 variables:

ID Numeric. Identification number of the individual.

RP Numeric. 1 if the row corresponds to a revealed preference (RP) observation. 0 otherwise.

RP_journey Numeric. Consecutive ID of RP observations. 0 if SP observation.

SP Numeric. 1 if the row corresponds to a stated preference (SP) observation. 0 otherwise.

SP_task Numeric. Consecutive ID of SP choice tasks. 0 if RP observation.

access_air Numeric. Access time (in minutes) of mode air.

access_bus Numeric. Access time (in minutes) of mode bus.

access_rail Numeric. Access time (in minutes) of mode rail.

av_air Numeric. 1 if the mode air (plane) is available. 0 otherwise.

av_bus Numeric. 1 if the mode bus is available. 0 otherwise.

av_car Numeric. 1 if the mode car is available. 0 otherwise.

av_rail Numeric. 1 if the mode rail (train) is available. 0 otherwise.

business Numeric. Purpose of the trip. 1 for business, 0 for other.

choice Numeric. Choice indicator, 1=car, 2=bus, 3=air, 4=rail.

cost_air Numeric. Cost (in GBP) of mode air.

cost_bus Numeric. Cost (in GBP) of mode bus.

cost_car Numeric. Cost (in GBP) of mode car.

cost_rail Numeric. Cost (in GBP) of mode rail.

female Numeric. Sex of individual. 1 for female, 0 for male.

income Numeric. Income (in GBP per annum) of the individual.

service_air Numeric. Additional services for the air alternative. 1 for no-frills, 2 for wifi, 3 for food. This is not used in the RP data, where it is set to 0.

service_rail Numeric. Additional services for the rail alternative. 1 for no-frills, 2 for wifi, 3 for food. This is not used in the RP data, where it is set to 0.

time_air Numeric. Travel time (in minutes) of mode air.

time_bus Numeric. Travel time (in minutes) of mode bus.

time_car Numeric. Travel time (in minutes) of mode car.

time_rail Numeric. Travel time (in minutes) of mode rail.

Details

This dataset is to be used for discrete choice modelling. Data comes from 500 individuals, each with two revealed preferences (RP) observation, and 14 stated stated (SC) observations. There are 8,000 choices in total. Data is simulated. Each observation contains attributes for the alternatives, availability of alternatives, and characteristics of the individuals.

Source

https://www.ApolloChoiceModelling.com/

---

apollo_modelOutput *Prints estimation results to console*

Description

Prints estimation results to console. Amount of information presented can be adjusted through arguments.

Usage

```
apollo_modelOutput(model, modelOutput_settings = NA)
```

Arguments

`model` Model object. Estimated model object as returned by function apollo_estimate.

`modelOutput_settings`
List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `printBHHH`: Logical. TRUE for printing BHHH standard errors. FALSE by default.
- `printChange`: Logical. TRUE for printing difference between starting values and estimates. FALSE by default.
- `printClassical`: Logical. TRUE for printing classical standard errors. TRUE by default.
- `printCorr`: Boolean. TRUE for printing parameters correlation matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. For Bayesian estimation, this setting is used for the covariane of random parameters. FALSE by default.

- printCovar: Boolean. TRUE for printing parameters covariance matrix. If printClassical=TRUE, both classical and robust matrices are printed. For Bayesian estimation, this setting is used for the correlation of random parameters. FALSE by default.
- printDataReport: Logical. TRUE for printing summary of choices in database and other diagnostics. FALSE by default.
- printFixed: Logical. TRUE for printing fixed parameters among estimated parameter. TRUE by default.
- printFunctions: Logical. TRUE for printing apollo_control, apollo_randCoeff (when available), apollo_lcPars (when available) and apollo_probabilities. FALSE by default.
- printHBconvergence: Boolean. TRUE for printing Geweke convergence tests. FALSE by default.
- printHBiterations: Boolean. TRUE for printing an iterations report for HB estimation. TRUE by default.
- printModelStructure: Logical. TRUE for printing model structure. TRUE by default.
- printOutliers: Logical or Scalar. TRUE for printing 20 individuals with worst average fit across observations. FALSE by default. If Scalar is given, this replaces the default of 20.
- printPVal: Logical or Scalar. TRUE or 1 for printing p-values for one-sided test, 2 for printing p-values for two-sided test, FALSE for not printing p-values. FALSE by default.
- printRobust: Logical. TRUE for printing robust standard errors. TRUE by default.
- printT1: Logical. If TRUE, t-test for H0: apollo_beta=1 are printed. FALSE by default.

Details

Prints to screen the output of a model previously estimated by apollo_estimate()

Value

A matrix of coefficients, s.d. and t-tests (invisible)

---

apollo_modifyUserDefFunc

Checks and modifies Apollo user-defined functions

Description

Checks and enhances user defined functions apollo_probabilities, apollo_randCoeff and apollo_lcPars.

Usage

```
apollo_modifyUserDefFunc(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  validate = TRUE,
  noModification = FALSE
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names of parameters inside apollo_beta whose values should be kept constant throughout estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
validate	Logical. If TRUE, the original and modified apollo_probabilities functions are estimated. If their results do not match, then the original functions are returned, and success is set to FALSE inside the returned list.
noModification	Logical. If TRUE, loop expansion etc are skipped.

Details

Internal use only. Called by apollo_estimate before estimation. Checks include: no re-definition of variables, no (direct) calls to database, calling of apollo_weighting if weights are defined.

Value

List with four elements: apollo_probabilities, apollo_randCoeff, apollo_lcPars and a dummy called success (TRUE if modification was successful, FALSE if not. FALSE will be only be returnes if the modifications are validated).

apollo_nl *Calculates Nested Logit probabilities*

Description

Calculates the probabilities of a Nested Logit model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_nl(nl_settings, functionality)
```

Arguments

nl_settings	List of inputs of the NL model. It should contain the following. • alternatives: Named numeric vector. Names of alternatives and their corresponding value in choiceVar. • avail: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in alternatives. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify avail=1 to indicate universal availability, or omit the setting completely. • choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in alternatives. • componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed. • nlNests: List of numeric scalars or vectors. Lambda parameters for each nest. Elements must be named with the nest name. The lambda at the root is automatically fixed to 1 if not provided by the user. • nlStructure: Named list of character vectors. As many elements as nests, it must include the "root". Each element contains the names of the nests or alternatives that belong to it. Element names must match those in nlNests. • utilities: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in alternatives. • rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.

- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

In this implementation of the Nested Logit model, each nest must have a lambda parameter associated to it. For the model to be consistent with utility maximisation, the estimated value of the Lambda parameter of all nests should be between 0 and 1. Lambda parameters are inversely proportional to the correlation between the error terms of alternatives in a nest. If lambda=1, then there is no relevant correlation between the unobserved utility of alternatives in that nest. The tree must contain an upper nest called "root". The lambda parameter of the root is automatically set to 1 if not specified in nlNests, but can be changed by the user if desired (though not advised).

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": Not implemented.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on nl_settings.
- "raw": Same as "prediction"

- "report": List with tree structure and choice overview.
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "utilities": List of vectors/matrices/arrays. Returns the utilities.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

---

apollo_normalDensity *Calculates density for a Normal distribution*

Description

Calculates density for a Normal distribution at a specific value with a specified mean and standard deviation and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_normalDensity(normalDensity_settings, functionality)
```

Arguments

normalDensity_settings

List of arguments to the functions. It must contain the following.

- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- mu: Numeric scalar. Intercept of the linear model.
- outcomeNormal: Numeric vector. Dependent variable.
- rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs).
- sigma: Numeric scalar. Variance of error component of linear model to be estimated.
- xNormal: Numeric vector. Single explanatory variable.

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.

- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

This function calculates the probability of the linear model outcomeNormal = mu + xNormal + epsilon, where epsilon is a random error distributed Normal(0,sigma). If using this function in the context of an Integrated Choice and Latent Variable (ICLV) model with continuous indicators, then outcomeNormal would be the value of the indicator, xNormal would be the value of the latent variable (possibly multiplied by a parameter to measure its correlation with the indicator, e.g. xNormal=lambda*LV), and mu would be an additional parameter to be estimated (the mean of the indicator, which should be fixed to zero if the indicator is centered around its mean beforehand).

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the likelihood for each observation.
- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": Predicted value at the observation level.
- "preprocess": Returns a list with pre-processed inputs, based on normalDensity_settings.
- "raw": Same as "estimate"
- "report": Dependent variable overview.

- "shares_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "utilities": List of vectors/matrices/arrays. Returns the x values.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.

---

apollo_ol *Calculates Ordered Logit probabilities*

Description

Calculates the probabilities of an Ordered Logit model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_ol(ol_settings, functionality)
```

Arguments

ol_settings — List of settings for the OL model. It should include the following.

- coding: Numeric or character vector. Optional argument. Defines the order of the levels in outcomeOrdered. The first value is associated with the lowest level of outcomeOrdered, and the last one with the highest value. If not provided, is assumed to be 1:(length(tau) + 1).
- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- outcomeOrdered: Numeric vector. Dependent variable. The coding of this variable is assumed to be from 1 to the maximum number of different levels. For example, if the ordered response has three possible values: "never", "sometimes" and "always", then it is assumed that outcomeOrdered contains "1" for "never", "2" for "sometimes", and 3 for "always". If another coding is used, then it should be specified using the coding argument.
- rows: Boolean vector. TRUE if a row must be considered in the calculations, FALSE if it must be excluded. It must have length equal to the length of argument outcomeOrdered. Default value is "all", meaning all rows are considered in the calculation.
- tau: List of numeric vectors/matrices/3-dim arrays. Thresholds. As many as number of different levels in the dependent variable - 1. Extreme thresholds are fixed at -inf and +inf. Mixing is allowed in thresholds. Can also be a matrix with as many rows as observations and as many columns as thresholds.
- utility: Numeric vector/matrix/3-sim array. A single explanatory variable (usually a latent variable). Must have the same number of rows as outcomeOrdered.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- `"components"`: For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- `"conditionals"`: For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- `"estimate"`: For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- `"gradient"`: For model estimation, produces analytical gradients of the likelihood, where possible.
- `"output"`: Prepares output for post-estimation reporting.
- `"prediction"`: For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- `"preprocess"`: Prepares likelihood functions for use in estimation.
- `"raw"`: For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- `"report"`: Prepares output summarising model and choiceset structure.
- `"shares_LL"`: Produces overall model likelihood with constants only.
- `"utilities"`: Returns utilities at provided parameter values.
- `"utilities"`: Returns utilities at provided parameter values.
- `"validate"`: Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- `"zero_LL"`: Produces overall model likelihood with all parameters at zero.

Details

This function estimates an Ordered Logit model of the type: y* = V + epsilon outcomeOrdered = 1 if -Inf < y* < tau[1] 2 if tau[1] < y* < tau[2] ... maxLvl if tau[length(tau)] < y* < +Inf Where epsilon is distributed standard logistic, and the values 1, 2, ..., maxLvl can be replaces by coding[1], coding[2], ..., coding[maxLvl]. The behaviour of the function changes depending on the value of the `functionality` argument.

Value

The returned object depends on the value of argument `functionality` as follows.

- `"components"`: Same as `"estimate"`
- `"conditionals"`: Same as `"estimate"`
- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.

- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all possible levels, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on ol_settings.
- "raw": Same as "prediction"
- "report": Dependent variable overview.
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "utilities": List of vectors/matrices/arrays. Returns the utilities.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.

---

apollo_op *Calculates Ordered Probit probabilities*

Description

Calculates the probabilities of an Ordered Probit model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_op(op_settings, functionality)
```

Arguments

op_settings List of settings for the OP model. It should include the following.

- coding: Numeric or character vector. Optional argument. Defines the order of the levels in outcomeOrdered. The first value is associated with the lowest level of outcomeOrdered, and the last one with the highest value. If not provided, is assumed to be 1:(length(tau) + 1).
- componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
- outcomeOrdered: Numeric vector. Dependent variable. The coding of this variable is assumed to be from 1 to the maximum number of different levels. For example, if the ordered response has three possible values: "never", "sometimes" and "always", then it is assumed that outcomeOrdered contains "1" for "never", "2" for "sometimes", and 3 for "always". If another coding is used, then it should be specified using the coding argument.
- rows: Boolean vector. TRUE if a row must be considered in the calculations, FALSE if it must be excluded. It must have length equal to the length of argument outcomeOrdered. Default value is "all", meaning all rows are considered in the calculation.

- tau: List of numeric vectors/matrices/3-dim arrays. Thresholds. As many as number of different levels in the dependent variable - 1. Extreme thresholds are fixed at -inf and +inf. Mixing is allowed in thresholds. Can also be a matrix with as many rows as observations and as many columns as thresholds.
- utility: Numeric vector/matrix/3-sim array. A single explanatory variable (usually a latent variable). Must have the same number of rows as outcomeOrdered.

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

This function estimates an ordered probit model of the type:

$$y^* = V + \epsilon y = 1 if -\infty < y^* < \tau_1, 2 if \tau_1 < y^* < \tau_2, ..., max(y) if \tau_{max(y)-1} < y^* < \infty$$

Where ϵ is distributed standard normal, and the values 1, 2, ..., $max(y)$ can be replaced by coding[1], coding[2], ..., coding[maxLvl]. The behaviour of the function changes depending on the value of the functionality argument.

Value

The returned object depends on the value of argument `functionality` as follows.

- `"components"`: Same as `"estimate"`
- `"conditionals"`: Same as `"estimate"`
- `"estimate"`: vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- `"gradient"`: List containing the likelihood and gradient of the model component.
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"prediction"`: List of vectors/matrices/arrays. Returns a list with the probabilities for all possible levels, with an extra element for the probability of the chosen alternative.
- `"preprocess"`: Returns a list with pre-processed inputs, based on `op_settings`.
- `"raw"`: Same as `"prediction"`
- `"report"`: Dependent variable overview.
- `"shares_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- `"utilities"`: List of vectors/matrices/arrays. Returns the utilities.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: Not implemented. Returns a vector of NA with as many elements as observations.

`apollo_outOfSample` *Cross-validation of fit (LL)*

Description

Randomly generates estimation and validation samples, estimates the model on the first and calculates the likelihood for the second, then repeats.

Usage

```
apollo_outOfSample(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
 estimate_settings = list(estimationRoutine = "bgw", maxIterations = 200, writeIter =
    FALSE, hessianRoutine = "none", printLevel = 3L, silent = TRUE),
 outOfSample_settings = list(nRep = 10, validationSize = 0.1, samples = NA, rmse = NULL)
)
```

Arguments

`apollo_beta` Named numeric vector. Names and values for parameters.

`apollo_fixed` Character vector. Names (as defined in `apollo_beta`) of parameters whose value should not change during estimation.

`apollo_probabilities`
Function. Returns probabilities of the model to be estimated. Must receive three arguments:

- `apollo_beta`: Named numeric vector. Names and values of model parameters.
- `apollo_inputs`: List containing options of the model. See apollo_validateInputs.
- `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.

`apollo_inputs` List grouping most common inputs. Created by function apollo_validateInputs.

`estimate_settings`
List. Options controlling the estimation process. See apollo_estimate.

`outOfSample_settings`
List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

nRep Numeric scalar. Number of times a different pair of estimation and validation sets are to be extracted from the full database. Default is 30.

rmse Character matrix with two columns. Used to calculate Root Mean Squared Error (RMSE) of prediction. The first column must contain the names of observed outcomes in the database. The second column must contain the names of the predicted outcomes as returned by `apollo_prediction`. If omitted or NULL, no RMSE is calculated. This only works for models with a single component.

samples Numeric matrix or data.frame. Optional argument. Must have as many rows as observations in the `database`, and as many columns as number of repetitions wanted. Each column represents a re-sample, and each element must be a 0 if the observation should be assigned to the estimation sample, or 1 if the observation should be assigned to the prediction sample. If this argument is provided, then `nRep` and `validationSize` are ignored. Note that this allows sampling at the observation rather than the individual level.

validationSize Numeric scalar. Size of the validation sample. Can be a percentage of the sample (0-1) or the number of individuals in the validation sample (>1). Default is 0.1.

Details

A common way to test for overfitting of a model is to measure its fit on a sample not used during estimation that is, measuring its out-of-sample fit. A simple way to do this is splitting the complete available dataset in two parts: an estimation sample, and a validation sample. The model of interest is estimated using only the estimation sample, and then those estimated parameters are used to measure the fit of the model (e.g. the log-likelihood of the model) on the validation sample. Doing this with only one validation sample, however, may lead to biased results, as a particular validation

sample need not be representative of the population. One way to minimise this issue is to randomly draw several pairs of estimation and validation samples from the complete dataset, and apply the procedure to each pair.

The splitting of the database into estimation and validation samples is done at the individual level, not at the observation level. If the sampling wants to be done at the individual level (not recommended on panel data), then the optional `outOfSample_settings$samples` argument should be provided.

This function writes two different files to the working/output directory:

- `modelName_outOfSample_params.csv`: Records the estimated parameters, final log-likelihood, and number of observations on each repetition.
- `modelName_outOfSample_samples.csv`: Records the sample composition of each repetition.

The first two files are updated throughout the run of this function, while the last one is only written once the function finishes.

When run, this function will look for the two files above in the working/output directory. If they are found, the function will attempt to pick up re-sampling from where those files left off. This is useful in cases where the original bootstrapping was interrupted, or when additional re-sampling wants to be performed.

Value

A matrix with the average log-likelihood per observation for both the estimation and validation samples, for each repetition. Two additional files with further details are written to the working/output directory.

`apollo_ownModel` *Calculates own model probabilities*

Description

Receives functions or expressions for each functionality so that a user-defined model can interface with Apollo.

Usage

```
apollo_ownModel(ownModel_settings, functionality)
```

Arguments

`ownModel_settings`

List of arguments. Only likelihood is mandatory.

- `gradient`: Function or expression used to calculate the gradient of the likelihood. If not provided, Apollo will attempt to calculate it automatically.
- `likelihood`: Function or expression used to calculate the likelihood of the model. Should evaluate to a vector, matrix, or 3-dimensional array.

- prediction: Function or expression used to calculate the prediction of the model. Should evaluate to a vector, matrix, or 3-dimensional array.
- report: List of functions or expressions used to produce a text report summarising the input and parameter estimates of the model. Should contain two elements: "data" (with a summary of the input data), and "param" (with a summary of the estimated parameters).
- shares_LL: Function or expression used to calculate the likelihood of the constants-only model.
- zero_LL: Function or expression used to calculate the likelihood of the base model (e.g. equiprobable model).

functionality Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are:

- "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.

- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on mnl_settings.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "raw": Same as "prediction"
- "report": Choice overview
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "validate": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

---

apollo_panelProd *Calculates product across observations from same individual.*

Description

Multiplies likelihood of observations from the same individual, or adds the log of them.

Usage

```
apollo_panelProd(P, apollo_inputs, functionality)
```

Arguments

P	List of vectors, matrices or 3-dim arrays. Likelihood of the model components.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "gradient": For model estimation, produces analytical gradients of the likelihood, where possible. • "output": Prepares output for post-estimation reporting.

- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "utilities": Returns utilities at provided parameter values.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

This function should be called inside apollo_probabilities only if the data has a panel structure. It should be called after apollo_avgIntraDraws if intra-individual draws are used.

Value

Argument P with (for most functionalities) the original contents after multiplying across observations at the individual level. Shape depends on argument functionality.

- "components": Returns P without changes.
- "conditionals": Returns P without averaging across draws. Drops all components except "model".
- "estimate": Returns P containing the likelihood of the model after multiplying observations at the individual level. Drops all components except "model".
- "gradient": Returns P containing the gradient of the likelihood after applying the product rule across observations for the same individual.
- "output": Returns P containing the likelihood of the model after multiplying observations at the individual level.
- "prediction": Returns P containing the probabilities/likelihoods of all alternatives for all model components averaged across inter-individual draws.
- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "report": Returns P without changes.
- "shares_LL": Returns P containing the likelihood of the model after multiplying observations at the individual level.
- "validate": Returns P containing the likelihood of the model averaged across inter-individual draws. Drops all components except "model".
- "utilities": Returns P without changes.
- "zero_LL": Returns P containing the likelihood of the model after multiplying observations at the individual level.

apollo_prediction *Predicts using an estimated model*

Description

Calculates apollo_probabilities with functionality="prediction".

Usage

```
apollo_prediction(
  model,
  apollo_probabilities,
  apollo_inputs,
  prediction_settings = list(),
  modelComponent = NA
)
```

Arguments

model	Model object. Estimated model object as returned by function apollo_estimate. A user can also simply provide a vector of parameter values instead of a whole model object, allowing for prediction to take place without having estimated a model.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
prediction_settings	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • modelComponent: Character. Name of component of apollo_probabilities output to calculate predictions for. Default is to predict for all components. • nRep: Scalar integer. Only used for models that require simulation for prediction (e.g. MDCEV). Number of draws used to calculate prediction. Default is 100. • runs: Numeric. Number of runs to use for computing confidence intervals of predictions. • silent: Boolean. If TRUE, this function won't print any output to screen.

- simChoice: Boolean. If TRUE, simulated choices are added to the output. FALSE by default.
- summary: Boolean. If TRUE, a summary of the prediction is printed to screen. TRUE by default.

modelComponent **Deprecated**. Same as modelComponent inside prediction_settings.

Details

Structure of predictions are simplified before returning, e.g. list of vectors are turned into a matrix.

Value

A list containing predictions for component modelComponent of the model described in apollo_probabilities. The particular shape of the prediction will depend on the model component.

---

apollo_preEstimate *Conducts all the pre-estimation steps for a model*

Description

Conducts all the pre-estimation steps for a model using the likelihood function defined by apollo_probabilities.

Usage

```
apollo_preEstimate(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  estimate_settings = NA
)
```

Arguments

apollo_beta Named numeric vector. Names and values for parameters.

apollo_fixed Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.

apollo_probabilities
Function. Returns probabilities of the model to be estimated. Must receive three arguments:

- apollo_beta: Named numeric vector. Names and values of model parameters.
- apollo_inputs: List containing options of the model. See apollo_validateInputs.
- functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".

`apollo_inputs` List grouping most common inputs. Created by function apollo_validateInputs.

`estimate_settings`

List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `bgw_settings`: List. Additional arguments to the BGW optimisation method. See bgw_mle for more details.
- `bootstrapSE`: Numeric. Number of bootstrap samples to calculate standard errors. Default is 0, meaning no bootstrap s.e. will be calculated. Number must zero or a positive integer. Only used if `apollo_control$estMethod!="HB"`.
- `bootstrapSeed`: Numeric scalar (integer). Random number generator seed to generate the bootstrap samples. Only used if `bootstrapSE>0`. Default is 24.
- `constraints`: Character vector. Linear constraints on parameters to estimate. For example `c('b1>0', 'b1 + 2*b2>1')`. Only >, < and = can be used. Inequalities cannot be mixed with equality constraints, e.g. `c(b1-b2=0, b2>0)` will fail. All parameter names must be on the left side. Fixed parameters cannot go into constraints. Alternatively, constraints can be defined as in maxLik. Constraints can only be used with maximum likelihood estimation and the BFGS routine in particular.
- `estimationRoutine`: Character. Estimation method. Can take values "bfgs", "bgw", "bhhh", or "nr". Used only if `apollo_control$HB` is FALSE. Default is "bgw".
- `hessianRoutine`: Character. Name of routine used to calculate the Hessian of the log-likelihood function after estimation. Valid values are `"analytic"` (default), `"numDeriv"` (to use the numeric routine in package numDeric), `"maxLik"` (to use the numeric routine in packahe maxLik), and `"none"` to avoid calculating the Hessian and the covariance matrix. Only used if `apollo_control$HB=FALSE`.
- `maxIterations`: Numeric. Maximum number of iterations of the estimation routine before stopping. Used only if `apollo_control$HB` is FALSE. Default is 200.
- `maxLik_settings`: List. Additional settings for maxLik. See argument `control` in maxBFGS, maxBHHH and maxNM for more details. Only used for maximum likelihood estimation.
- `numDeriv_method`: Character. Method used for numerical differentiation when calculating the covariance matrix. Can be `"Richardson"` or `"simple"`, Only used if analytic gradients are available. See argument `method` in grad for more details.
- `numDeriv_settings`: List. Additional arguments to the method used by numDeriv to calculate the Hessian. See argument `method.args` in grad for more details.
- `printLevel`: Higher values render more verbous outputs. Can take values 0, 1, 2 or 3. Ignored if apollo_control$HB is TRUE. Default is 3.
- `scaleAfterConvergence`: Logical. Used to increase numerical precision of convergence. If TRUE, parameters are scaled to 1 after convergence, and the estimation is repeated from this new starting values. Results are

reported scaled back, so it is a transparent process for the user. Default is FALSE.

- `scaleHessian`: Logical. If TRUE, parameters are scaled to 1 for Hessian estimation. Default is TRUE.
- `scaling`: Named vector. Names of elements should match those in `apollo_beta`. Optional scaling for parameters. If provided, for each parameter `i`, `(apollo_beta[i]/scaling[i])` is optimised, but `scaling[i]*(apollo_beta[i]/scaling[i])` is used during estimation. For example, if parameter b3=10, while b1 and b2 are close to 1, then setting `scaling = c(b3=10)` can help estimation, specially the calculation of the Hessian. Reports will still be based on the non-scaled parameters.
- `silent`: Logical. If TRUE, no information is printed to the console during estimation. Default is FALSE.
- `validateGrad`: Logical. If TRUE, the analytical gradient (if used) is compared to the numerical one. Default is FALSE.
- `writeIter`: Logical. Writes value of the parameters in each iteration to a csv file. Works only if `estimation_routine=="bfgs"|"bgw"`. Default is TRUE.

Value

model object

`apollo_prepareProb` *Checks likelihood function*

Description

Checks that the likelihood function for the mode is in the appropriate format to be returned.

Usage

```
apollo_prepareProb(P, apollo_inputs, functionality)
```

Arguments

`P`	List of vectors, matrices or 3-dim arrays. Likelihood of the model components.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`functionality`	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- `"components"`: For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.

- "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "utilities": Returns utilities at provided parameter values.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Details

This function should be called inside apollo_probabilities, near the end of it, just before return(P). This function only performs checks on the shape of P, but does not change its values.

Value

Argument P with (for most functionalities) the original contents. Output depends on argument functionality.

- "components": Returns P without changes.
- "conditionals": Returns only the "model" component of argument P.
- "estimate": Returns only the "model" component of argument P.
- "gradient": Returns only the "model" component of argument P.
- "output": Returns argument P without any changes to its content, but gives names to unnamed elements.
- "prediction": Returns argument P without any changes.
- "preprocess": Returns argument P without any changes to its content, but gives names to elements corresponding to componentNames.
- "raw": Returns argument P without any changes.
- "report": Returns P without changes.
- "shares_LL": Returns argument P without any changes to its content, but gives names to unnamed elements.

- "validate": Returns argument P without any changes.
- "utilities": Returns P without changes.
- "zero_LL": Returns argument P without any changes to its content, but gives names to unnamed elements.

---

apollo_preprocess *Pre-process input for multiple models return*

Description

Pre-process input for multiple models return

Usage

```
apollo_preprocess(inputs, modelType, functionality, apollo_inputs)
```

Arguments

inputs	List of settings
modelType	Character. Type of model, e.g. "mnl", "nl", "cnl", etc.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "gradient": For model estimation, produces analytical gradients of the likelihood, where possible. • "output": Prepares output for post-estimation reporting. • "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws. • "preprocess": Prepares likelihood functions for use in estimation. • "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws. • "report": Prepares output summarising model and choiceset structure. • "shares_LL": Produces overall model likelihood with constants only.

- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

apollo_inputs List grouping most common inputs. Created by function apollo_validateInputs.

Value

The returned object is a pre-processed version of the model settings. This is independent of functionality, but the function is only called during preprocessing.

---

apollo_print *Prints message to terminal*

Description

Prints message to terminal if apollo_inputs$silent is FALSE

Usage

```
apollo_print(txt, nSignifD = 4, widthLim = 11, pause = 0, type = "t")
```

Arguments

txt	Character, what to print.
nSignifD	Optional numeric integer. Minimum number of significant digits when printing numeric matrices. Default is 4.
widthLim	Optional numeric integer. Minimum width (in characters) of each column when printing numeric matrices. Default is 11
pause	Scalar integer. Number of seconds the execution will pause after printing the message. Default is 0.
type	Character. "t" for regular text (default), "w" for warning, "i" for information.

Value

Nothing

apollo_readBeta *Reads parameters from file*

Description

Reads in parameters from a previously estimated model and copies the values to the given apollo_beta vector, only for those parameters whose name matches.

Usage

```
apollo_readBeta(
  apollo_beta,
  apollo_fixed,
  inputModelName,
  overwriteFixed = FALSE
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
inputModelName	Character. modelName for model from which results are used as starting values.
overwriteFixed	Boolean. TRUE if starting values for fixed parameters should also be updated from input file.

Details

This function will update the values of the parameters in its argument apollo_beta with the matching values in the file (inputModelName)_estimates.csv. If there is no match for a given parameter in apollo_beta, its value will not be updated.

Value

Named numeric vector. Names and updated starting values for parameters.

apollo_rrm *Calculates Random Regret Minimisation model probabilities*

Description

Calculates the probabilities of a Random Regret Minimisation model and can also perform other operations based on the value of the functionality argument.

Usage

```
apollo_rrm(rrm_settings, functionality)
```

Arguments

rrm_settings	List of inputs of the RRM model. It should contain the following. • alternatives: Named numeric vector. Names of alternatives and their corresponding value in choiceVar. • avail: Named list of numeric vectors or scalars. Availabilities of alternatives, one element per alternative. Names of elements must match those in alternatives. Values can be 0 or 1. These can be scalars or vectors (of length equal to rows in the database). A user can also specify avail=1 to indicate universal availability, or omit the setting completely. • choiceset_scaling: Vector. One entry per row in the database, often set to 2 divided by the number of available alternatives. • choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in alternatives. • componentName: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed. • regret_inputs: Named list of regret functions. This should contain one list per attribute, where these lists themselves contain two vectors, namely a vector of attributes (at the alternative level) and parameters (either generic or attribute specific). Zeros can be used for omitted attributes for some alternatives. The order for each attribute needs to be the same as the order in alternatives.. • regret_scale: Named list of regret scales. This should have the same length as 'rrm_settings$regret_inputs' or be a single entry in the case of a generic scale parameter across regret attributes. • rows: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs). • rum_inputs: Named list of (optional) deterministic utilities. Utilities of the alternatives to be included in combined RUM-RRM models. Names of elements must match those in alternatives.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.

- "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.
- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The returned object depends on the value of argument functionality as follows.

- "components": Same as "estimate"
- "conditionals": Same as "estimate"
- "estimate": vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.
- "gradient": List containing the likelihood and gradient of the model component.
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "prediction": List of vectors/matrices/arrays. Returns a list with the probabilities for all alternatives, with an extra element for the probability of the chosen alternative.
- "preprocess": Returns a list with pre-processed inputs, based on rrm_settings.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "raw": Same as "prediction"
- "report": Choice overview
- "shares_LL": vector/matrix/array. Returns the probability of the chosen alternative when only constants are estimated.
- "utilities": List of vectors/matrices/arrays. Returns the regret functions
- "validate": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

`apollo_saveOutput` *Saves estimation results to files.*

Description

Writes files in the working/output directory with the estimation results.

Usage

```
apollo_saveOutput(model, saveOutput_settings = NA)
```

Arguments

`model` Model object. Estimated model object as returned by function apollo_estimate.

`saveOutput_settings`

List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional.

- `printBHHH`: Logical. TRUE for printing BHHH standard errors. FALSE by default.
- `printChange`: Boolean. TRUE for printing difference between starting values and estimates. FALSE by default.
- `printClassical`: Boolean. TRUE for printing classical standard errors. TRUE by default.
- `printCorr`: Boolean. TRUE for printing parameters correlation matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. For Bayesian estimation, this setting is used for the covariane of random parameters. TRUE by default.
- `printCovar`: Boolean. TRUE for printing parameters covariance matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. For Bayesian estimation, this setting is used for the correlation of random parameters. TRUE by default.
- `printDataReport`: Boolean. TRUE for printing summary of choices in database and other diagnostics. FALSE by default.
- `printFixed`: Logical. TRUE for printing fixed parameters among estimated parameter. TRUE by default.
- `printFunctions`: Boolean. TRUE for printing apollo_control, apollo_randCoeff (when available), apollo_lcPars (when available) and apollo_probabilities. TRUE by default.
- `printHBconvergence`: Boolean. TRUE for printing Geweke convergence tests. TRUE by default.
- `printHBiterations`: Boolean. TRUE for printing an iterations report for HB estimation. TRUE by default.
- `printModelStructure`: Boolean. TRUE for printing model structure. TRUE by default.

- `printOutliers`: Boolean or Scalar. TRUE for printing 20 individuals with worst average fit across observations. FALSE by default. If Scalar is given, this replaces the default of 20.
- `printPVal`: Boolean or Scalar. TRUE or 1 for printing p-values for one-sided test, 2 for printing p-values for two-sided test, FALSE for not printing p-values. FALSE by default.
- `printRobust`: Logical. TRUE for printing robust standard errors. TRUE by default.
- `printT1`: Boolean. If TRUE, t-test for H0: apollo_beta=1 are printed. FALSE by default.
- `saveEst`: Boolean. TRUE for saving estimated parameters and standard errors to a CSV file. TRUE by default.
- `saveCorr`: Boolean. TRUE for saving estimated correlation matrix to a CSV file. FALSE by default.
- `saveCov`: Boolean. TRUE for saving estimated covariance matrix to a CSV file. FALSE by default.
- `saveHBiterations`: Boolean. TRUE for including HB iterations in the saved model object. FALSE by default.
- `saveModelObject`: Boolean. TRUE to save the R model object to a file (use apollo_loadModel to load it to memory). TRUE by default.
- `saveOld`: Boolean. If TRUE, existing files are kept with an added OLD suffix. If not, they are overwritten. TRUE by default.
- `writeF12`: Boolean. TRUE for writing results into an F12 file (ALOGIT format). FALSE by default.

Details

Estimation results are saved different files in the working/output directory:

- `(modelName)_corr.csv` CSV file with the estimated classical correlation matrix. Only when bayesian estimation was not used.
- `(modelName)_covar.csv` CSV file with the estimated classical covariance matrix. Only when bayesian estimation was not used.
- `(modelName)_estimates.csv` CSV file with the estimated parameter values, their standars errors, and t-ratios.
- `(modelName).F12` F12 file with model results. Compatible with ALOGIT.
- `(modelName)_output.txt` Text file with the output produced by function `apollo_modelOutput`.
- `(modelName)_robcorr.csv` CSV file with the estimated robust correlation matrix. Only when bayesian estimation was not used.
- `(modelName)_robcovar.csv` CSV file with the estimated robust covariance matrix. Only when bayesian estimation was not used.

Value

nothing

apollo_searchStart *Searches for better starting values.*

Description

Given a set of starting values and a range for them, searches for points with a better likelihood and steeper gradients.

Usage

```
apollo_searchStart(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  searchStart_settings = NA
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
searchStart_settings	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • apolloBetaMax: Vector. Maximum possible value of parameters when generating candidates. Ignored if smartStart is TRUE. Default is apollo_beta + 0.1. • apolloBetaMin: Vector. Minimum possible value of parameters when generating candidates. Ignored if smartStart is TRUE. Default is apollo_beta - 0.1. • bfgsIter: Numeric scalar. Number od BFGS iterations to perform at each stage to each remaining candidate. Default is 20.

- dTest: Numeric scalar. Tolerance for test 1. A candidate is discarded if its distance in parameter space to a better one is smaller than dTest. Default is 1.
- gTest: Numeric scalar. Tolerance for test 2. A candidate is discarded if the norm of its gradient is smaller than gTest AND its LL is further than llTest from a better candidate. Default is 10^(-3).
- llTest: Numeric scalar. Tolerance for test 2. A candidate is discarded if the norm of its gradient is smaller than gTest AND its LL is further than llTest from a better candidate. Default is 3.
- maxStages: Numeric scalar. Maximum number of search stages. The algorithm will stop when there is only one candidate left, or if it reaches this number of stages. Default is 5.
- nCandidates: Numeric scalar. Number of candidate sets of parameters to be used at the start. Should be an integer bigger than 1. Default is 100.
- smartStart: Boolean. If TRUE, candidates are randomly generated with more chances in the directions the Hessian indicates improvement of the LL function. Default is FALSE.

Details

This function implements a simplified version of the algorithm proposed by Bierlaire, M., Themans, M. & Zufferey, N. (2010), A Heuristic for Nonlinear Global Optimization, INFORMS Journal on Computing, 22(1), pp.59-70. The main difference lies in it implementing only two out of three tests on the candidates described by the authors. The implemented algorithm has the following steps.

1. Randomly draw nCandidates candidates from an interval given by the user.
2. Label all candidates with a valid log-likelihood (LL) as active.
3. Apply bfgsIter iterations of the BFGS algorithm to each active candidate.
4. Apply the following tests to each active candidate:
   (a) Has the BGFS search converged?
   (b) Are the candidate parameters after BFGS closer than dTest from any other candidate with higher LL?
   (c) Is the LL of the candidate after BFGS further than distLL from a candidate with better LL, and its gradient smaller than gTest?
5. Mark any candidates for which at least one test results in yes as inactive.
6. Go back to step 3, unless only one candidate is active, or the maximum number of iterations (maxStages) has been reached.

This function will write a CSV file to the working/output directory summarising progress. This file is called modelName_searchStart.csv .

Value

named vector of model parameters. These are the best values found.

apollo_setRows *Sets specified rows to a given value*

Description

Given a numeric object (scalar, vector, matrix or 3-dim array) sets a subset of rows to a given value.

Usage

```
apollo_setRows(v, r, val)
```

Arguments

v	Numeric scalar, vector, matrix or 3-dim array. Rows of this object will be replaced by val and
r	Boolean vector. As many elements as rows in utilities. TRUE for replacing that row, FALSE for not changing it.
val	Numeric scalar. Value to which the specified rows must be set to.

Value

The same argument utilities but with the rows where r==TRUE set to val.

apollo_setWorkDir *Automatically sets working directory to active file directory*

Description

This function only works in Rstudio. If called outside RStudio will just print a message to screen saying it could not set the working directory.

Usage

```
apollo_setWorkDir()
```

Value

(invisibly) TRUE if it manages to set the working directory, FALSE if not.

apollo_sharesTest *Compares predicted and observed shares*

Description

Comparing the shares predicted by the model with the shares observed in the data, and conducts statistical tests.

Usage

```
apollo_sharesTest(
  model,
  apollo_probabilities,
  apollo_inputs,
  sharesTest_settings
)
```

Arguments

model	Model object. Estimated model object as returned by function apollo_estimate.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
sharesTest_settings	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • alternatives: Named numeric vector. Names of alternatives and their corresponding value in choiceVar. • choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in alternatives. • modelComponent: Name of model component. Set to model by default. • newAlts: Optional list describing the new alternatives to be used by apollo_sharesTest. This should have as many elements as new alternatives, with each entry being a matrix of 0-1 entries, with one row per observation, and one column per alternative used in the model. • newAltsOnly: Boolean. If TRUE, results will only be printed for the 'new' alternatives defined in newAlts, not the original alternatives used in the model. Set to FALSE by default.

- `subsamples`: Named list of boolean vectors. Each element of the list defines whether a given observation belongs to a given subsample (e.g. by sociodemographics).

Details

This is an auxiliary function to help guide the definition of utility functions in a choice model. By comparing the predicted and observed shares of alternatives for different categories of the data, it is possible to identify what additional explanatory variables could improve the fit of the model.

Value

Nothing

`apollo_sink` *Starts or stops writing output to a text file.*

Description

Starts or stops writing the output shown in the console to a file named "modelName_additional_output.txt".

Usage

```
apollo_sink(apollo_inputs = NULL)
```

Arguments

`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs. If not provided, it will be looked for in the global environment.

Details

After the first time this function is called, all output shown in the console will also be written to a text file called "modelName_additional_output.txt", where "modelName" is the modelName set inside apollo_control. The second time this function is called, it stops writing the console output to the file. The user should always call this function an even number of times to close the output file and prevents data loss.

Value

Nothing.

apollo_speedTest *Measures evaluation time of a model*

Description

Measures the evaluation time of a model for different number of cores and draws.

Usage

```
apollo_speedTest(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
  speedTest_settings = NA
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
speedTest_settings	List. Contains settings for this function. User input is required for all settings except those with a default or marked as optional. • nCoresTry: Numeric vector. Number of threads to try. Default is from 1 to the detected number of cores. • nDrawsTry: Numeric vector. Number of inter and intra-person draws to try. Default value is c(50, 100, 200). • nRep: Numeric scalar. Number of times the likelihood is evaluated for each combination of threads and draws. Default is 10.

Details

This function evaluates the function `apollo_probabilities` several times using different number of threads (a.k.a. processor cores), and draws (if the model uses mixing). It then plots the estimation time for each combination. Estimation time grows at least linearly with number of draws, while time savings decrease with the number of threads. This function can help decide what number of draws and cores to use for estimation, though a high number of draws is always recommended. If the computer will be used for additional activities during estimation, no more than (machine number of cores - 1) should be used. Using more threads than cores available in the machine will lead to reduce dperformance. The use of additional cores come at the expense of additional memory usage. If R uses more memory than the physical RAM available, then significant slow-downs in processing time can be expected. This function can help avoiding such pitfalls.

Value

A list of two matrices, one with the average time per evaluation for each number of threads and draws combination, and one with the RAM used. A graph is also plotted.

`apollo_swissRouteChoiceData`

Dataset of route choice.

Description

A Stated Preference dataset containing 3,492 route choices among two alternatives.

Usage

```
apollo_swissRouteChoiceData
```

Format

A data frame with 3,492 rows and 16 variables:

ID Numeric. Identification number of the individual.

choice Numeric. Choice indicator, 1 for alternative 1, and 2 for alternative 2.

tt1 Numeric. Travel time (in minutes) for alternative 1.

tc1 Numeric. Travel cost (in CHF) for alternative 1.

hw1 Numeric. Headway time (in minutes) for alternative 1.

ch1 Numeric. Number of interchanges for alternative 1.

tt2 Numeric. Travel time (in minutes) for alternative 2.

tc2 Numeric. Travel cost (in CHF) for alternative 2.

hw2 Numeric. Headway time (in minutes) for alternative 2.

ch2 Numeric. Number of interchanges for alternative 2.

hh_inc_abs Numeric. Household income (in CHF per annum).

car_availability Numeric. 1 if respondent has a car available, 0 otherwise.

commute Numeric. 1 if the purpose of the trip is commuting. 0 otherwise.

shopping Numeric. 1 if the purpose of the trip is shopping. 0 otherwise.

business Numeric. 1 if the purpose of the trip is business. 0 otherwise.

leisure Numeric. 1 if the purpose of the trip is leisure. 0 otherwise.

Details

This dataset is to be used for discrete choice modelling. Data comes from 388 individuals who participated in a Stated Choice (SC) survey, providing a total of 3,492 observations. Each choice scenario includes two rail alternatives described in terms of travel time, cost, headway and interchanges. Additional information on respondents is available. This dataset comes from the following publication: Axhausen, K.W., Hess, S., König, A., Abay, G., Bates, J.J. & Bierlaire, M. (2008), Income and distance elasticities of values of travel time savings: New Swiss results, Transport Policy, 15(3), pp. 173-185. https://doi.org/10.1016/j.tranpol.2008.02.001

Source

https://www.ApolloChoiceModelling.com/

---

apollo_timeUseData *Dataset of time use.*

Description

A Revealed Preference dataset containing 2,826 full-day observations.

Usage

```
apollo_timeUseData
```

Format

An object of class `data.frame` with 2826 rows and 20 columns.

Details

This dataset is to be used for Multiple Discrete Continuous (MDC) modelling. Data comes from 447 individuals who provided activitry diaries for a total of 2,826 days. Each observation summarizes the amount of time spent in each of twelve different activities. The dataset also incluides characteristics of the participants. This dataset comes from the following publication. Calastri, C., Crastes dit Sourd, R. and Hess, S. (2020) We want it all: experiences from a survey seeking to capture social network structures, lifetime events and short-term travel and activity planning. Transportation, 47(1), pp. 175-201.

indivID Numeric. Identification number of the individual.

day Numeric. Index of the day for each observation (day 1 was excluded).

date Numeric. Date in format yyyymmdd.

budget Numeric. Total amount of time registered during the day (in minutes).

t_a01 Numeric. Time spent dropping-of or picking up other people (in minutes).

t_a02 Numeric. Time spent working (in minutes).

t_a03 Numeric. Time spent on educational activities (in minutes).

t_a04 Numeric. Time spent shopping (in minutes).

t_a05 Numeric. Time spent on private business (in minutes).

t_a06 Numeric. Time spent getting petrol (in minutes).

t_a07 Numeric. Time spent on social or leasure activities (in minutes).

t_a08 Numeric. Time spent on vacation or long (inter-city) travel (in minutes).

t_a09 Numeric. Time spent doing exercise (in minutes).

t_a10 Numeric. Time spent at home (in minutes).

t_a11 Numeric. Time spent travelling (everyday travelling) (in minutes).

t_a12 Numeric. Non-allocated time (in minutes).

female Numeric. 1 if respondent is female. 0 otherwise.

age Numeric. Age of respondent (in years, approximate).

occ_full_time Numeric. 1 if the respondent works full time.

weekend Numeric. 1 if the current date is a weekend.

Source

https://www.ApolloChoiceModelling.com/

`apollo_tobit` *Calculates density for a Tobit model (censored Normal)*

Description

Calculates density for a censored Normal distribution at a specific value with a specified mean and standard deviation and user provided bounds, and can also perform other operations based on the value of the `functionality` argument.

Usage

```
apollo_tobit(tobit_settings, functionality)
```

Arguments

`tobit_settings` List of arguments to the functions. It must contain the following.

- `componentName`: Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in `P` to which the function output is directed.
- `lowerLimit`: Numeric scalar. Lower bound beyond which the density is 0. If not provided by the user, this will be set to -Inf.
- `mu`: Numeric scalar. Intercept of the linear model.
- `outcomeTobit`: Numeric vector. Dependent variable.
- `rows`: Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is `"all"`, equivalent to `rep(TRUE, nObs)`.
- `sigma`: Numeric scalar. Variance of error component of linear model to be estimated.
- `upperLimit`: Numeric scalar. Upper bound beyond which the density is 0. If not provided by the user, this will be set to +Inf.
- `xTobit`: Numeric vector. Single explanatory variable.

`functionality` Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call `apollo_probabilities`, though the user can also call `apollo_probabilities` manually with a given functionality for testing/debugging. Possible values are:

- `"components"`: For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations.
- `"conditionals"`: For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws.
- `"estimate"`: For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- `"gradient"`: For model estimation, produces analytical gradients of the likelihood, where possible.
- `"output"`: Prepares output for post-estimation reporting.
- `"prediction"`: For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- `"preprocess"`: Prepares likelihood functions for use in estimation.
- `"raw"`: For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- `"report"`: Prepares output summarising model and choiceset structure.
- `"shares_LL"`: Produces overall model likelihood with constants only.
- `"utilities"`: Returns utilities at provided parameter values.
- `"validate"`: Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- `"zero_LL"`: Produces overall model likelihood with all parameters at zero.

Details

This function calculates the probability of the linear model outcomeTobit = mu + xTobit + epsilon, where epsilon is a random error distributed Normal(0,sigma), but with optional lower and upper bounds imposed by the user (outside of which the density would be 0).

Value

The returned object depends on the value of argument `functionality` as follows.

- `"components"`: Same as `"estimate"`
- `"conditionals"`: Same as `"estimate"`
- `"estimate"`: vector/matrix/array. Returns the likelihood for each observation.
- `"gradient"`: List containing the likelihood and gradient of the model component.
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"prediction"`: Predicted value at the observation level.
- `"preprocess"`: Returns a list with pre-processed inputs, based on `tobit_settings`.
- `"raw"`: Same as `"estimate"`
- `"report"`: Dependent variable overview.
- `"shares_LL"`: Not implemented. Returns a vector of NA with as many elements as observations.
- `"utilities"`: List of vectors/matrices/arrays. Returns the x values.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: Not implemented. Returns a vector of NA with as many elements as observations.

---

apollo_unconditionals *Returns unconditionals for models with random heterogeneity*

Description

Returns unconditionals for random parameters in model, both for continuous mixtures and latent class.

Usage

```
apollo_unconditionals(
  model,
  apollo_probabilities,
  apollo_inputs,
  obsLevel = FALSE
)
```

Arguments

`model`	Model object. Estimated model object as returned by function apollo_estimate.
`apollo_probabilities`	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • `apollo_beta`: Named numeric vector. Names and values of model parameters. • `apollo_inputs`: List containing options of the model. See apollo_validateInputs. • `functionality`: Character. Can be either `"components"`, `"conditionals"`, `"estimate"` (default), `"gradient"`, `"output"`, `"prediction"`, `"preprocess"`, `"raw"`, `"report"`, `"shares_LL"`, `"validate"` or `"zero_LL"`.
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`obsLevel`	Logical. If `TRUE`, unconditionals are returned at the observation level rather than person level. This setting only applies to continuous mixture models and is set to TRUE by default only in the presence of intra-individual draws. Otherwise, the default is FALSE.

Details

This functions is only meant for use with models using continuous distributions or latent classes, or both at the same time.

Value

Depends on whether the model uses continuous mixtures or latent class.

- If the model contains a continuous mixture, it returns a list with one object per random coefficient. When using inter-individual draws only, each element will be a matrix with one row per individual, and one column per draw. When using intra- individual draws, each element will be a three-dimensional array, with one row per observation, inter-individual draws in the second dimension, and intra- individual draws in the third dimension.
- If the model contains latent classes, it returns a list with as many elements as random coefficients in the model, plus one additional element containing the class allocation probabilities.
- If the model contains both continuous mixing and latent classes, a list with the two elements described above will be returned.

`apollo_validate` *Pre-process input for common models return*

Description

Pre-process input for common models return

Usage

```
apollo_validate(inputs, modelType, functionality, apollo_inputs)
```

Arguments

inputs	List of settings
modelType	Character. Type of model, e.g. "mnl", "nl", "cnl", etc.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "gradient": For model estimation, produces analytical gradients of the likelihood, where possible. • "output": Prepares output for post-estimation reporting. • "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws. • "preprocess": Prepares likelihood functions for use in estimation. • "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws. • "report": Prepares output summarising model and choiceset structure. • "shares_LL": Produces overall model likelihood with constants only. • "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "zero_LL": Produces overall model likelihood with all parameters at zero.
apollo_inputs	List of main inputs to the model estimation process. See apollo_validateInputs.

Value

The returned object depends on the value of argument operation

---

apollo_validateControl

Validates apollo_control

Description

Validates the options controlling the running of the code apollo_control and sets default values for the omitted ones.

Usage

```
apollo_validateControl(database, apollo_control, silent = FALSE)
```

Arguments

database	data.frame. Data used by model.
apollo_control	List. Options controlling the running of the code. User input is required for all settings except those with a default or marked as optional.

- calculateLLC: Boolean. TRUE if user wants to calculate LL at constants (if applicable). - TRUE by default.
- HB: Boolean. TRUE if using RSGHB for Bayesian estimation of model.
- indivID: Character. Name of column in the database with each decision maker's ID.
- memorySaver: Boolean. TRUE to reduce memory usage when calculating analytical gradients and hessian - FALSE by default.
- mixing: Boolean. TRUE for models that include random parameters.
- modelDescr: Character. Description of the model. Used in output files.
- modelName: Character. Name of the model. Used when saving the output to files.
- nCores: Numeric>0. Number of cores to use in calculations of the model likelihood.
- noDiagnostics: Boolean. TRUE if user does not wish model diagnostics to be printed - FALSE by default.
- noValidation: Boolean. TRUE if user does not wish model input to be validated before estimation - FALSE by default.
- outputDirectory: Character. Optional directory for outputs if different from working director - empty by default
- panelData: Boolean. TRUE if there are multiple observations (i.e. rows) for each decision maker - Automatically set based on indivID by default.
- seed: Numeric. Seed for random number generation.
- weights: Character. Name of column in database containing weights for estimation.
- workInLogs: Boolean. TRUE for increased numeric precision in models with panel data - FALSE by default.

silent	Boolean. If TRUE, no messages are printed to screen.

Details

This function should be run before running apollo_validateData.

Value

Validated version of apollo_control, with additional element called panelData set to TRUE for repeated choice data.

apollo_validateData *Validates data*

Description

Checks consistency of the database with apollo_control, sorts it by indivID, and adds an internal ID variable (apollo_sequence)

Usage

```
apollo_validateData(database, apollo_control, silent)
```

Arguments

database	data.frame. Data used by model.
apollo_control	List. Options controlling the running of the code. See apollo_validateInputs.
silent	Boolean. TRUE to prevent the function from printing to the console. Default is FALSE.

Details

This function should be called after calling apollo_validateControl. Observations are sorted only if apollo_control$panelData=TRUE.

Value

Data.frame. Validated version of database.

apollo_validateHBControl

Validates the apollo_HB *list of parameters*

Description

Validates the apollo_HB list of parameters and sets default values for the omitted ones.

Usage

```
apollo_validateHBControl(
  apollo_HB,
  apollo_beta,
  apollo_fixed,
  apollo_control,
  silent = FALSE
)
```

Arguments

apollo_HB	List. Contains options for Bayesian estimation. See ?RSGHB::doHB for details. Parameters modelname, gVarNamesFixed, gVarNamesNormal, gDIST, svN and FC are automatically set based on the other arguments of this function. Other settings to include are the following. • constraintsNorm: Character vector. Constraints for *random* coefficients in bayesian estimation. Constraints can be written as "b1>b2", "b1<b2", "b1>0", or "b1<0". • fixedA: Named numeric vector. Contains the names and fixed mean values of random parameters. For example, c(b1=0) fixes the mean of b1 to zero. • fixedD: Named numeric vector. Contains the names and fixed variance of random parameters. For example, c(b1=1) fixes the variance of b1 to zero. • gFULLCV: Boolean. Whether the full variance-covariance structure should be used for random parameters (TRUE by default). • gNCREP: Numeric. Number of burn-in iterations to use prior to convergence (default=10^5). • gNEREP: Numeric. Number of iterations to keep for averaging after convergence has been reached (default=10^5). • gINFOSKIP: Numeric. Number of iterations between printing/plotting information about the iteration process (default=250). • hbDist: *Mandatory* setting. A named character vector determining the distribution of each parameter to be estimated. Possible values are as follows. – "CN+": Positive censored normal. – "CN-": Negative censored normal. – "JSB": Johnson SB. – "LN+": Positive log-normal. – "LN-": Negative log-normal. – "N": Normal. – "NR": Fixed (as in non-random) parameter. • nodiagnostics: Boolean. Turn off pre-estimation diagnostics for RSGHB. Set to TRUE by default.
apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation. value is constant throughout estimation).
apollo_control	List. Options controlling the running of the code. See apollo_validateInputs.
silent	Boolean. TRUE to keep the function from printing to the console. Default is FALSE.

Details

This function is only necessary when using bayesian estimation.

Value

Validated apollo_HB

apollo_validateInputs *Prepares input for* apollo_estimate

Description

Searches the user work space (.GlobalEnv) for all necessary input to run apollo_estimate, and packs it in a single list.

Usage

```
apollo_validateInputs(
  apollo_beta = NA,
  apollo_fixed = NA,
  database = NA,
  apollo_control = NA,
  apollo_HB = NA,
  apollo_draws = NA,
  apollo_randCoeff = NA,
  apollo_lcPars = NA,
  recycle = TRUE,
  silent = FALSE
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
database	data.frame. Data used by model.
apollo_control	List. Options controlling the running of the code. User input is required for all settings except those with a default or marked as optional.

- analyticGrad: Boolean. TRUE to use analytical gradients during parameter estimation, if they are available. FALSE to use numerical gradients. - TRUE by default.
- calculateLLC: Boolean. TRUE if user wants to calculate LL at constants (if applicable). - TRUE by default.
- HB: Boolean. TRUE if using RSGHB for Bayesian estimation of model.
- indivID: Character. Name of column in the database with each decision maker's ID.
- mixing: Boolean. TRUE for models that include random parameters.
- modelDescr: Character. Description of the model. Used in output files.
- modelName: Character. Name of the model. Used when saving the output to files.
- nCores: Numeric>0. Number of cores to use in calculations of the model likelihood.

- noDiagnostics: Boolean. TRUE if user does not wish model diagnostics to be printed - FALSE by default.
- noValidation: Boolean. TRUE if user does not wish model input to be validated before estimation - FALSE by default.
- outputDirectory: Character. Optional directory for outputs if different from working director - empty by default
- panelData: Boolean. TRUE if there are multiple observations (i.e. rows) for each decision maker - Automatically set based on indivID by default.
- seed: Numeric. Seed for random number generation.
- weights: Character. Name of column in database containing weights for estimation.
- workInLogs: Boolean. TRUE for increased numeric precision in models with panel data - FALSE by default.

apollo_HB — List. Contains options for Bayesian estimation. See ?RSGHB::doHB for details. Parameters modelname, gVarNamesFixed, gVarNamesNormal, gDIST, svN and FC are automatically set based on the other arguments of this function. Other settings to include are the following.

- constraintNorm: Character vector. Constraints for *random* coefficients in bayesian estimation. Constraints can be written as "b1>b2", "b1<b2", "b1>0", or "b1<0".
- fixedA: Named numeric vector. Contains the names and fixed mean values of random parameters. For example, c(b1=0) fixes the mean of b1 to zero.
- fixedD: Named numeric vector. Contains the names and fixed variance of random parameters. For example, c(b1=1) fixes the variance of b1 to zero.
- gNCREP: Numeric. Number of burn-in iterations to use prior to convergence (default=10^5).
- gNEREP: Numeric. Number of iterations to keep for averaging after convergence has been reached (default=10^5).
- gINFOSKIP: Numeric. Number of iterations between printing/plotting information about the iteration process (default=250).
- hbDist: *Mandatory* setting. A named character vector determining the distribution of each parameter to be estimated. Possible values are as follows.
  - "CN+": Positive censored normal.
  - "CN-": Negative censored normal.
  - "DNE": Parameter kept at its starting value (not estimated).
  - "JSB": Johnson SB.
  - "LN+": Positive log-normal.
  - "LN-": Negative log-normal.
  - "N": Normal.
  - "NR": Fixed (as in non-random) parameter.

apollo_draws — List of arguments describing the inter and intra individual draws. Required only if apollo_control$mixing = TRUE. Unused elements can be ommited.

- interDrawsType: Character. Type of inter-individual draws ('halton','mlhs','pmc','sobol','sobolOw 'sobolFaureTezuka', 'sobolOwenFaureTezuka' or the name of an object loaded in memory, see manual in www.ApolloChoiceModelling.com for details).

- `interNDraws`: Numeric scalar (>=0). Number of inter-individual draws per individual. Should be set to 0 if not using them.
- `interNormDraws`: Character vector. Names of normaly distributed inter-individual draws.
- `interUnifDraws`: Character vector. Names of uniform-distributed inter-individual draws.
- `intraDrawsType`: Character. Type of intra-individual draws ('halton','mlhs','pmc','sobol','sobolOw 'sobolOwenFaureTezuka' or the name of an object loaded in memory).
- `intraNDraws`: Numeric scalar (>=0). Number of intra-individual draws per individual. Should be set to 0 if not using them.
- `intraUnifDraws`: Character vector. Names of uniform-distributed intra-individual draws.
- `intraNormDraws`: Character vector. Names of normaly distributed intra-individual draws.

`apollo_randCoeff`
: Function. Used with mixing models. Constructs the random parameters of a mixing model. Receives two arguments:
  - `apollo_beta`: Named numeric vector. Names and values of model parameters.
  - `apollo_inputs`: The output of this function (`apollo_validateInputs`).

`apollo_lcPars`
: Function. Used with latent class models. Constructs a list of parameters for each latent class. Receives two arguments:
  - `apollo_beta`: Named numeric vector. Names and values of model parameters.
  - `apollo_inputs`: The output of this function (`apollo_validateInputs`).

`recycle`
: Logical. If TRUE, an older version of `apollo_inputs` is looked for in the calling environment (parent frame), and any element in that old version created by the user is copied into the new `apollo_inputs` returned by this function. For `recycle=TRUE` to work, the old version of `apollo_inputs` **must** be named "apollo_inputs". If FALSE, nothing is copied from any older version of apollo_inputs. TRUE is the default.

`silent`
: Logical. TRUE to keep the function from printing to the console. Default is FALSE.

Details

All arguments to this function are optional. If the function is called without arguments, then it it will look in the user workspace (i.e. the global environment) for variables with the same name as its omitted arguments. We strongly recommend users to visit https://www.ApolloChoiceModelling.com/ for examples on how to use Apollo. In the website, users will also find a detailed manual and a user-group for help and further reference.

Value

List grouping several required input for model estimation.

apollo_varcov *Calculates variance-covariance matrix of an Apollo model*

Description

Calculates the Hessian, variance-covariance matrix and standard errors of an Apollo model as defined by its likelihood function and apollo_inputs list of settings. Performs automatic scaling for increased numeric stability.

Usage

```
apollo_varcov(apollo_beta, apollo_fixed, varcov_settings)
```

Arguments

apollo_beta	Named numeric vector. Names and values of parameters at which to calculate the covariance matrix. Values **must not be scaled**, and they must include any fixed parameter.
apollo_fixed	Character vector. Names of fixed parameters.
varcov_settings	List of settings defining the behaviour of this function. It must contain at least one of the following: apollo_logLike, apollo_grad or apollo_inputs together with apollo_probabilities.

- apollo_grad: Function to calculate the gradient of the model, as returned by apollo_makeGrad.
- apollo_hessian: Function to calculate the hessian of the model, as returned by apollo_makeHessian.
- apollo_inputs: List grouping most common inputs. Created by function apollo_validateInputs.
- apollo_logLike: Function to calculate the log-likelihood of the model, as returned by apollo_makeLogLike.
- apollo_probabilities: apollo_probabilities Function. Returns probabilities of the model to be estimated. Must receive three arguments:
  - apollo_beta: Named numeric vector. Names and values of model parameters.
  - apollo_inputs: List containing options of the model. See apollo_validateInputs.
  - functionality: Character. Can be either "components", "conditionals", "estimate" (default), "gradient", "output", "prediction", "preprocess", "raw", "report", "shares_LL", "validate" or "zero_LL".
- BHHHhessian: Matrix. Optional input, providing the BHHH matrix so it does not get recalculated.
- hessianRoutine: Character. Name of routine used to calculate the Hessian. Valid values are "analytic", "numDeriv", "maxLik" or "none" to avoid estimating the Hessian and covariance matrix.

- numDeriv_method: Character. Method used for numerical differentiation. Can be "Richardson" or "simple", Only used if analytic gradients are available. See argument method in grad for more details.
- numDeriv_settings: List. Additional arguments to the Richardson method used by numDeriv to calculate the Hessian. See argument method.args in grad for more details.
- scaleBeta: Logical. If TRUE (default), parameters are scaled by their own value before calculating the Hessian to increase numerical stability. However, the output is de-scaled, so they are in the same scale as the apollo_beta argument.

Details

It calculates the Hessian, variance-covariance, and standard errors at apollo_beta values of an estimated model. At least one of the following settings must be provided (ordered by speed of computation): apollo_grad, apollo_logLike, or (apollo_probabilities and apollo_inputs). If more than one is provided, then the priority is: apollo_grad, apollo_logLike, (apollo_probabilities and apollo_inputs).

Value

List with the following elements

- apollo_beta: Named numerical vector. Parameter estimates (model$estimate, not scaled).
- corrmat: Numerical matrix. Correlation between parameter estimates.
- hessian: Numerical matrix. Hessian of the model at parameter estimates (model$estimate).
- hessianScaling: Named numeric vector. Scales used on the paramaters to calculate the Hessian (non-fixed only).
- methodsAttempted: Character vector. Name of methods attempted to calculate the Hessian.
- methodUsed: Character. Name of method used to calculate the Hessian.
- robcorrmat: Numerical matrix. Robust correlation between parameter estimates.
- robse: Named numerical vector. Robust standard errors of parameter estimates.
- robvarcov: Numerical matrix. Robust variance-covariance matrix.
- se: Named numerical vector. Standard errors of parameter estimates.
- varcov: Numerical matrix. Variance-covariance matrix.

---

apollo_varList *Lists variable names and definitions used inside a function*

Description

Returns a list containing the names and definitions of variables in f, apollo_randCoeff and apollo_lcPars

Usage

```
apollo_varList(f, apollo_inputs)
```

Arguments

f	A function, usually apollo_probabilities
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.

Details

It looks for variable definitions inside f, apollo_randCoeff, and apollo_lcPars. It returns them in a list.

Value

A list of expressions containing all definitions in f, apollo_randCoeff and apollo_probabilities

---

apollo_weighting *Applies weights*

Description

Applies weights to individual observations in likelihood function.

Usage

```
apollo_weighting(P, apollo_inputs, functionality)
```

Arguments

P	List of vectors, matrices or 3-dim arrays. Likelihood of the model components.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
functionality	Character. Setting instructing Apollo what processing to apply to the likelihood function. This is in general controlled by the functions that call apollo_probabilities, though the user can also call apollo_probabilities manually with a given functionality for testing/debugging. Possible values are: • "components": For further processing/debugging, produces likelihood for each model component (if multiple components are present), at the level of individual draws and observations. • "conditionals": For conditionals, produces likelihood of the full model, at the level of individual inter-individual draws. • "estimate": For model estimation, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws. • "gradient": For model estimation, produces analytical gradients of the likelihood, where possible.

- "output": Prepares output for post-estimation reporting.
- "prediction": For model prediction, produces probabilities for individual alternatives and individual model components (if multiple components are present) at the level of an observation, after averaging across draws.
- "preprocess": Prepares likelihood functions for use in estimation.
- "raw": For debugging, produces probabilities of all alternatives and individual model components at the level of an observation, at the level of individual draws.
- "report": Prepares output summarising model and choiceset structure.
- "shares_LL": Produces overall model likelihood with constants only.
- "validate": Validates model specification, produces likelihood of the full model, at the level of individual decision-makers, after averaging across draws.
- "zero_LL": Produces overall model likelihood with all parameters at zero.

Value

The likelihood (i.e. probability in the case of choice models) of the model in the appropriate form for the given functionality, multiplied by individual-specific weights.

`apollo_writeF12` *Writes an F12 file*

Description

Writes an F12 file (ALogit format) with the results of a model estimation.

Usage

```
apollo_writeF12(model, truncateCoeffNames = TRUE)
```

Arguments

`model`	Model object. Estimated model object as returned by function apollo_estimate.
`truncateCoeffNames`	Boolean. TRUE to truncate parameter names to 10 characters. TRUE by default.

Value

Nothing.

apollo_writeTheta *Writes the vector [beta,ll] to a file called* `modelname_iterations.csv`

Description

Writes the vector [beta,ll] to a file called `modelname_iterations.csv`

Usage

```
apollo_writeTheta(
  beta,
  ll,
  modelName,
  scaling = NULL,
  outDir = NULL,
  apollo_beta = NULL
)
```

Arguments

`beta`	vector of parameters to be written (including fixed ones).
`ll`	scalar representing the log-likelihood of the whole model.
`modelName`	Character. Name of the model.
`scaling`	Numeric vector of scales applied to beta
`outDir`	Scalar character. Name of output directory
`apollo_beta`	Named numeric vector of starting values.

Value

Nothing.

aux_validateRows *Validates and expands rows if necessary.*

Description

Validates and expands rows if necessary.

Usage

```
aux_validateRows(rows, componentName = NULL, apollo_inputs = NULL)
```

Arguments

rows	Boolean vector. Consideration of which rows to include. Length equal to the number of observations (nObs), with entries equal to TRUE for rows to include, and FALSE for rows to exclude. Default is "all", equivalent to rep(TRUE, nObs). Set to "all" by default if omitted.
componentName	Character. Name given to model component. If not provided by the user, Apollo will set the name automatically according to the element in P to which the function output is directed.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.

---

print.apollo *Prints brief summary of Apollo model*

Description

Receives an estimated model object and prints a brief summary using the generic print function.

Usage

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

Arguments

x	Model object. Estimated model object as returned by function apollo_estimate.
...	further arguments passed to or from other methods.

Value

nothing.

---

summary.apollo *Prints summary of Apollo model*

Description

Receives an estimated model object and prints a summary using the generic summary function.

Usage

```
## S3 method for class 'apollo'
summary(object, ..., pTwoSided = FALSE)
```

Arguments

`object`	Model object. Estimated model object as returned by function apollo_estimate.
`...`	further arguments passed to or from other methods.
`pTwoSided`	Logical. Should two-sided p-values be printed instead of one-sided p-values. FALSE by default. #' @return nothing.

Index

∗ **datasets**
    apollo_drugChoiceData, 29
    apollo_modeChoiceData, 80
    apollo_swissRouteChoiceData, 116
    apollo_timeUseData, 117
.onAttach, 4

apollo_addCovariance, 5
apollo_attach, 5, *25, 26*
apollo_avgInterDraws, 6, *21*
apollo_avgIntraDraws, 7, *21*
apollo_basTest, 9
apollo_bootstrap, 9
apollo_checkArguments, 12
apollo_choiceAnalysis, 13
apollo_classAlloc, 14
apollo_cnl, 15
apollo_cnl2, 17
apollo_combineModels, 19
apollo_combineResults, 21
apollo_compareInputs, 22
apollo_conditionals, 23
apollo_deltaMethod, 24
apollo_detach, *5*, 25
apollo_dft, 26
apollo_diagnostics, 29
apollo_drugChoiceData, 29
apollo_dVdB, 31
apollo_dVdBOld, 31
apollo_el, 32
apollo_emdc, 34
apollo_emdc1, 35
apollo_emdc2, 37
apollo_estimate, *5*, *9, 10*, *23, 24*, 38, *41*, *43*, *55, 56*, *58*, *61*, *66*, *74*, *76, 77*, *81*, *93*, *98*, *108*, *113*, *121*, *132*, *134, 135*
apollo_estimateHB, 40
apollo_expandLoop, 42
apollo_firstRow, 42
apollo_fitsTest, 43
apollo_fmnl, 44
apollo_fnl, 46
apollo_initialise, 48
apollo_insertComponentName, 49
apollo_insertFunc, 49
apollo_insertOLList, 50
apollo_insertRows, 51
apollo_insertRRMQuotes, 51
apollo_insertScaling, 52
apollo_keepRows, 52
apollo_lc, 53
apollo_lcConditionals, 54
apollo_lcEM, 55
apollo_lcEM_new, 57
apollo_lcUnconditionals, 58
apollo_llCalc, 59
apollo_loadModel, 60, *109*
apollo_longToWide, 60
apollo_lrTest, 61
apollo_makeCluster, 62
apollo_makeDraws, 63
apollo_makeGrad, 64, *129*
apollo_makeHessian, 65, *129*
apollo_makeLogLike, *64*, *65*, 65, *129*
apollo_mdcev, 67
apollo_mdcev2, 69
apollo_mdcnev, 71
apollo_mixConditionals, 74
apollo_mixEM, 75
apollo_mixUnconditionals, 76
apollo_mlhs, 77
apollo_mnl, 78
apollo_modeChoiceData, 80
apollo_modelOutput, *40*, 81
apollo_modifyUserDefFunc, 82
apollo_nl, 84
apollo_normalDensity, 86
apollo_ol, 88
apollo_op, 90

apollo_outOfSample, 92
apollo_ownModel, 94
apollo_panelProd, *21*, 96
apollo_prediction, 98
apollo_preEstimate, 99
apollo_prepareProb, *21*, 101
apollo_preprocess, 103
apollo_print, 104
apollo_readBeta, 105
apollo_rrm, 105
apollo_saveOutput, *40*, 108
apollo_searchStart, 110
apollo_setRows, 112
apollo_setWorkDir, 112
apollo_sharesTest, 113
apollo_sink, 114
apollo_speedTest, 115
apollo_swissRouteChoiceData, 116
apollo_timeUseData, 117
apollo_tobit, 118
apollo_unconditionals, 120
apollo_validate, 121
apollo_validateControl, 122, *124*
apollo_validateData, *43*, 124
apollo_validateHBControl, 124
apollo_validateInputs, *5–7*, *10*, *13*, *14*, *20*, *22*, *23*, *25*, *29*, *31*, *39*, *41–43*, *53*, *55–59*, *62*, *63*, *66*, *74*, *75*, *77*, *83*, *93*, *96*, *98–101*, *104*, *110*, *113–115*, *121*, *122*, *124*, *125*, 126, *129*, *131*, *134*
apollo_varcov, 129
apollo_varList, 130
apollo_weighting, 131
apollo_writeF12, 132
apollo_writeTheta, 133
aux_validateRows, 133

bgw_mle, *39*, *100*

grad, *39*, *100*, *130*

maxBFGS, *39*, *100*
maxBHHH, *39*, *100*
maxLik, *39*, *100*
maxNM, *39*, *100*

on.exit, *5*

print.apollo, 134

summary.apollo, 134