Package 'apollo'

May 8, 2019

Type Package

Title Tools for Choice Model Estimation and Application

Version 0.0.7

Description The Choice Modelling Centre (CMC) at the University of Leeds has developed flexible code 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 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

Encoding UTF-8

LazyData true

Depends R (>= 3.1.0), stats, utils

Imports Rcpp (>= 1.0.0), maxLik, mnormt, mvtnorm, graphics, coda, sandwich, randtoolbox, numDeriv, RSGHB, parallel

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation yes

Author Stephane Hess [aut],
David Palma [aut, cre]

Maintainer David Palma <D.Palma@leeds.ac.uk>

Repository CRAN

Date/Publication 2019-05-08 14:20:03 UTC

R topics documented:

`.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_addLog *Writes an entry to* apolloLog

Description

Writes an entry to the apolloLog, which lives inside apollo_inputs.

Usage

```
apollo_addLog(title = "", content = "", apolloLog)
```

Arguments

title	Character. Title of the log entry.
content	Content of the log entry. Can be a single element or a list. Each element will be converted to character using print, and concatenated with a line feed in between.
apolloLog	Environment. It contains the character vectors of titles and content.

Details

The variable apolloLog is an environment created inside apollo_inputs by apollo_validateInputs, but re-set by apollo_estimate. As an environment, it can be modified in place, i.e. all changes done within this function are recorded in apolloLog, even if it belongs to another environment.

Value

TRUE if writing was succesful, FALSE if not.

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 they are contained in.

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 `apollo_probabilities`, or more conveniently, using on.exit. `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.

Examples

```
apollo_beta  <- c(b1=0.3, b2=-0.5)
apollo_fixed <- c()
apollo_control <- list(indivID="id", mixing = FALSE, panelData = FALSE)
database <- data.frame(id=1:100, x1=stats::runif(100), x2=stats::runif(100))
apollo_inputs <- apollo_validateInputs()
apollo_attach(apollo_beta, apollo_inputs)
V = b1*x1 + b2*x2
apollo_detach(apollo_beta, apollo_inputs)
```

`apollo_avgInterDraws` *Averages 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. Description of the desired output from `apollo_probabilities`. Can take the values: "estimate", "prediction", "validate", "zero_LL", "conditionals", "output", "raw".

Value

Likelihood averaged over inter-individual draws (shape depends on argument `functionality`).

- "estimate": Returns the likelihood of the model averaged across inter-individual draws.
- "prediction": Returns the likelihood of all alternatives and all model components across inter-individual draws.

- "validate": Returns P without changes.
- "zero_LL": Returns P without changes.
- "conditionals": Returns P without changes.
- "output": Returns the same as "estimate", but also prints a summary of the estimation data.
- "raw": Returns P without changes.

`apollo_avgIntraDraws` *Averages 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. Description of the desired output from `apollo_probabilities`. Can take the values: "estimate", "prediction", "validate", "zero_LL", "conditionals", "output", "raw".

Value

Likelihood averaged over intra-individual draws (shape depends on argument `functionality`).

- "estimate": Returns the likelihood of the model averaged across intra-individual draws.
- "prediction": Returns the likelihood of all alternatives and all model components across intra-individual draws.
- "validate": Returns P without changes.
- "zero_LL": Returns P without changes.
- "conditionals": Returns P without changes.
- "output": Returns the same than "estimate", but also prints a summary of estimation data.
- "raw": Returns P without changes.

`apollo_choiceAnalysis` *Reports market share for subsamples*

Description

Compares market shares across subsamples in dataset, and writes results to a file.

Usage

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

Arguments

`choiceAnalysis_settings`
List containing settings for this function. The settings must be:

- 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.
- 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 database.

`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 directory.

Value

nothing, but prints the output to screen and writes a csv file to the working directory.

apollo_cnl *Calculates probabilities of a cross nested logit*

Description

Calculates probabilities of a cross nested logit model.

Usage

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

Arguments

`cnl_settings` List of inputs of the CNL 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.
- choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- V: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those 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.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Can take different values depending on desired output.

- "estimate": Used for model estimation.
- "prediction": Used for model predictions.
- "validate": Used for validating input.
- "zero_LL": Used for calculating null likelihood.
- "conditionals": Used for calculating conditionals.
- "output": Used for preparing output after model estimation.
- "raw": Used for debugging.

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. 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`. And while setting it to another value is possible, it is not recommended. Alpha parameters inside `cnlStructure` should be between 0 and 1. Using a transformation to ensure this constraint is satisfied is recommended (e.g. logistic transformation).

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 chosen alternative probability.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

Examples

```
### Load data
data(apollo_modeChoiceData)
database <- apollo_modeChoiceData
rm(apollo_modeChoiceData)

### Parameters
b = list(asc_1=0, asc_2=0, asc_3=0, asc_4=0, tt=0, tc=0, acc=0,
         lambda_fastPT=0.5, lambda_groundPT=0.5, alpha_rail_fastPT=0.5)

### List of utilities
V = list()
V[['car' ]] = b$asc_1 + b$tt*database$time_car  + b$tc*database$cost_car
V[['bus' ]] = b$asc_2 + b$tt*database$time_bus  + b$tc*database$cost_bus  +
              b$acc*database$access_bus
V[['air' ]] = b$asc_3 + b$tt*database$time_air  + b$tc*database$cost_air  +
              b$acc*database$access_air
V[['rail']] = b$asc_4 + b$tt*database$time_rail + b$tc*database$cost_rail +
              b$acc*database$access_rail

cnlStructure     = matrix(0, nrow=3, ncol=4)
cnlStructure[1,] = c( 0,  0, 1, b$alpha_rail_fastPT  ) # fastPT
```

```
cnlStructure[2,] = c( 0,  1, 0, 1-b$alpha_rail_fastPT) # groundPT
cnlStructure[3,] = c( 1,  0, 0, 0                    ) # car

### CNL settings
cnlSettings <- list(
   alternatives = c(car=1, bus=2, air=3, rail=4),
   avail        = list(car=database$av_car, bus=database$av_bus,
                       air=database$av_air, rail=database$av_rail),
   choiceVar    = database$choice,
   V            = V,
   cnlNests     = list(fastPT=b$lambda_fastPT, groundPT=b$lambda_groundPT, car=1),
   cnlStructure = cnlStructure
)

### Compute choice probabilities using CNL model
apollo_cnl(cnlSettings, functionality="estimate")
```

`apollo_combineModels` *Combines separate model components.*

Description

Combines model components to create probability for overall model.

Usage

```
apollo_combineModels(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. Can take different values depending on desired output. • "estimate": For model estimation, returns likelihood of model. • "prediction": For model predictions, returns probabilities of all alternatives. • "validate": Validates input. • "zero_LL": Return probabilities with all parameters at zero. • "conditionals": For conditionals, returns likelihood of model. • "output": Checks that the model is well defined. • "raw": For debugging, returns probabilities of all alternatives.

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.

Value

Argument P with an extra element called "model", which is the product of all the other elements.

---

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 of options. It can include the following.

- modelNames: Character vector. List of names of models to combine. Use an empty vector to combine results from all models in the directory.
- printClassical: Boolean. TRUE for printing classical standard errors. TRUE 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.
- estimateDigits: Numeric scalar. Number of decimal places to print for estimates. Default is 4.
- tDigits: Numeric scalar. Number of decimal places to print for t-ratios values. Default is 2.
- pDigits: Numeric scalar. Number of decimal places to print for p-values. Default is 2.

Value

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

apollo_conditionals *Calculates conditionals*

Description

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

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
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_deltaMethod *Delta method*

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 of arguments. It must contain the following. • operation: Character. Function to calculate the delta method for. See details. • parName1: Character. Name of the first parameter. • parName2: Character. Name of the second parameter. Optional depending on operation. • multPar1: Numeric scalar. A value to scale parName1. • multPar2: Numeric scalar. A value to scale parName2.

Details

apollo_deltaMethod supports the following five operations.

sum Calculates the s.e. of parName1 + parName2

diff Calculates the s.e. of parName1 - parName2 and parName2 - parName1

ratio 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.

Value

Value, s.e. and t-ratio resulting from the operation.

---

apollo_detach *Detaches parameters and the database.*

Description

Detaches variables attached by apollo_attach.

Usage

```
apollo_detach(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 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 be achieved by adding the line on.exit(apollo_detach(apollo_beta, apollo_inputs)) right after calling apollo_attach.

Value

Nothing.

Examples

```
apollo_beta  <- c(b1=0.3, b2=-0.5)
apollo_fixed <- c()
apollo_control <- list(indivID="id", mixing = FALSE, panelData = FALSE)
database <- data.frame(id=1:100, x1=stats::runif(100), x2=stats::runif(100))
apollo_inputs <- apollo_validateInputs()
apollo_attach(apollo_beta, apollo_inputs)
V = b1*x1 + b2*x2
apollo_detach(apollo_beta, apollo_inputs)
```

---

apollo_dft *Calculate DFT probabilities*

Description

Calculate probabilities of a Decision Field Theory (DFT) with external thresholds.

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.
- choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- 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).

- altStart: A named list with as many elements as alternatives. Each elment can be a scalar or vector containing the starting preference value for the alternative.
- 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.
- 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.
- 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 rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Can take different values depending on desired output.

- "estimate": Used for model estimation.
- "prediction": Used for model predictions.
- "validate": Used for validating input.
- "zero_LL": Used for calculating null likelihood.
- "conditionals": Used for calculating conditionals.
- "output": Used for preparing output after model estimation.
- "raw": Used for debugging.

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 chosen alternative probability.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

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. and Choudhury, C. (Submitted) An accumulation of preference: two alternative dynamic models for understanding transport choices. Roe, R.; Busemeyer, J. and Townsend, J. (2001) Multialternative decision field theory: A dynamic connectionist model of decision making. Psychological Review 108, 370

`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 10000 rows and 33 variables:

ID Numeric. Identification number of the individual.

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

best Numeric. Consecutive ID of RP observation.

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

third_pref Numeric. Consecutive ID of SP choice task.

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

brand_1 Character. Name of alternative's brand.

country_1 Character. Name of alternative's country of origin.

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

side_effects_1 Numeric. Chance of suffering negative side effects if this alternative is consumed.

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

brand_2 Character. Name of alternative's brand.

country_2 Character. Name of alternative's country of origin.

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

side_effects_2 Numeric. Chance of suffering negative side effects if this alternative is consumed.

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

brand_3 Character. Name of alternative's brand.

country_3 Character. Name of alternative's country of origin.

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

side_effects_3 Numeric. Chance of suffering negative side effects if this alternative is consumed.

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

brand_4 Character. Name of alternative's brand.

country_4 Character. Name of alternative's country of origin.

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

side_effects_4 Numeric. Chance of suffering negative side effects if this alternative is consumed.

price_4 Numeric. Cost of this alternative 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 preferences (SP) observations 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

http://www.apollochoicemodelling.com/

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

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.
- 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.
- V: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in alternatives.
- scales: List of vectors. Scale factors of each logit model. Should have one element less than choiceVars. At least one element should be normalized to 1. If omitted, scale=1 for all positions is assumed.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is "all", equivalent to rep(TRUE, nObs).

functionality — Character. Can take different values depending on desired output.

- "estimate": Used for model estimation.
- "prediction": Used for model predictions.
- "validate": Used for validating input.
- "zero_LL": Used for calculating null likelihood.
- "conditionals": Used for calculating conditionals.
- "output": Used for preparing output after model estimation.
- "raw": Used for debugging.

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": Not applicable.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
`apollo_inputs`	List grouping most common inputs. Created by function apollo_validateInputs.
`estimate_settings`	List. Options controlling the estimation process.

- estimationRoutine: Character. Estimation method. Can take values "bfgs", "bhhh", or "nr". Used only if `apollo_control$HB` is FALSE. Default is "bfgs".
- maxIterations: Numeric. Maximum number of iterations of the estimation routine before stopping. Used only if `apollo_control$HB` is FALSE. Default is 200.
- writeIter: Boolean. Writes value of the parameters in each iteration to a csv file. Works only if `estimation_routine="bfgs"`. Default is TRUE.
- hessianRoutine: Character. Name of routine used to calculate the Hessian of the loglikelihood function after estimation. Valid values are `"numDeriv"` (default) and `"maxLik"` to use the routines in those packages, and `"none"` to avoid estimating the Hessian (and the covariance matrix). Only used if `apollo_control$HB=FALSE`.
- printLevel: Higher values render more verbous outputs. Can take values 0, 1, 2 or 3. Ignored if apollo_control$HB is TRUE. Default is 3.
- constraints: Constraints on parameters to estimate. Should ignore fixed parameters. See argument `constraints` in maxBFGS for more details.
- 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.
- 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.
- silent: Boolean. If TRUE, no information is printed to the console during estimation. Default is FALSE.

Details

This is the main function of the Apollo package. The estimation process begins by checking the definition of `apollo_probabilities` by estimating it at the starting values. Then it runs the function with argument `functionality="validate"`. If the user requested more than one core for estimation (i.e. `apollo_control$nCores>1`), and no bayesian estimation is used (i.e. `apollo_control$HB=FALSE`), then a cluster is created. Using a cluster at least doubles the requires RAM, as the database must be copied into the cluster. 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. This functions does not save results into a file nor prints them into the console, so if users want to see and store estimation the results, they must make sure to call function `apollo_modelOutput` and/or `apollo_saveOutput` afterwards.

Value

model object

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 components, 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.

Examples

```
database <- data.frame(ID=rep(1:5, each=3), apollo_sequence=rep(1:3, 5))
apollo_inputs <- list(database=database)
attach(database)

P0 <- 0.5
apollo_firstRow(P0, apollo_inputs)

P1 <- rep(c(0.1, 0.2, 0.3, 0.4, 0.5), each=3)
apollo_firstRow(P1, apollo_inputs)

P2 <- matrix(rep(P1,10), nrow=15, ncol=10)
apollo_firstRow(P2, apollo_inputs)
```

```
P3 <- array(rep(P1, 10*10), dim=c(15, 10, 10))
apollo_firstRow(P3, apollo_inputs)

P4 <- list(P0, P1, P2, P3)
apollo_firstRow(P4, apollo_inputs)
```

`apollo_fitsTest` *Compares fit of model across categories*

Description

Given the predictions of a model, it compares the fit 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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".

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

`fitsTest_settings`
List of arguments. It must contain the following elements.

- 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).
- modelComponent: Name of model component. Set to model by default.

Details

Prints a table comparing the average fit for each category.

Value

Matrix with average fit per category (invisibly).

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_lc *Calculates the likelihood of a latent class model*

Description

Using the conditional likelihoods of each latent class, as well as teir classification probabilities, calculate the weighted likelihood of the whole model.

Usage

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

Arguments

lc_settings	List of arguments used by apollo_lc. It must include the following. • inClassProb: List of probabilities. Conditional likelihood for each class. One element per class, in the same order as classProb. • classProb: List of probabilities. Allocation probability for each class. One element per class, in the same order as inClassProb.
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
functionality	Character. Can take different values depending on desired output. • "estimate" Used for model estimation. • "prediction" Used for model predictions.

- "validate" Used for validating input.
- "zero_LL" Used for calculating null likelihood.
- "conditionals" Used for calculating conditionals.
- "output" Used for preparing output after model estimation.
- "raw" Used for debugging.

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 chosen alternative probability.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

Examples

```
data(apollo_modeChoiceData)
database <- apollo_modeChoiceData
rm(apollo_modeChoiceData)
N <- nrow(database)
lc_settings <- list(inClassProb=list(rnorm(N), rnorm(N)),
                    classProb=list(stats::runif(N), stats::runif(N)))
apollo_control <- list(indivID="ID")
x <- apollo_lc(lc_settings, apollo_control, functionality="estimate")
summary(x)
```

---

apollo_lcConditionals *Calculates conditionals of a latent class model.*

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
`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_lcUnconditionals` *Returns draws for random parameters in a latent class model model*

Description

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

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
`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 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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
`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". Then, it 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 an estimated model object from a file in the current working directory.

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 directory, loads the object contained in it, and returns it.

Value

A model object.

apollo_lrTest *Likelihood ratio test*

Description

Calculates the likelihood ratio test and prints result.

Usage

```
apollo_lrTest(baseModel, generalModel)
```

Arguments

baseModel	Character. Name of a previously estimated model whose results were written to disk by apollo_saveOutput. This is the restricted model, i.e. the one with fewer parameters.
generalModel	Either a character variable with the name of a previously estimated model, or an estimated model in memory, as returned by apollo_estimate. This model nests baseModel, and it should have more parameters than it.

Value

LL ratio test statistic (invisibly)

apollo_makeCluster *Creates cluster for estimation.*

Description

Creates cluster and loads pieces of the database for each worker.

Usage

```
apollo_makeCluster(apollo_probabilities, apollo_inputs, silent = 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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
silent	Boolean. If TRUE, it reports progress to the console. Default is FALSE.

Details

Internal use only. Called by apollo_estimate before estimation. AT least doubles up memory usage. But during the splitting it uses even more (~250

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 have 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 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_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, cl = 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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
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.
cl	Cluster as provided by makeCluster. Assumed to be PSock.

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 likelihood of a Multiple Discrete Continuous Extreme Value (MDCEV) model.

Usage

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

Arguments

mdcev_settings	List of settings for the MDCEV model. It must include the following. • V: Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'. • alternatives: Character vector. Names of alternatives, elements must match the names in list 'V'.

- alpha: Named list. Alpha parameters for each alternative, including for the outside good. As many elements as alternatives.
- gamma: Named list. Gamma parameters for each alternative, including for the outside good. As many elements as alternatives.
- sigma: Numeric scalar. Scale parameter of the model extreme value type I error.
- 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`.
- avail: Named list. Availabilities of alternatives, one element per alternative. Names of elements must match those in argument 'alternatives'. Value for each element can be 1 (scalar if always available) or a vector with values 0 or 1 for each observation. If all alternatives are always available, then user can just omit this argument.
- 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`.
- budget: Numeric vector. Budget for each observation.
- minConsumption: Named list of scalars or numeric vectors. Minimum consumption of the alternatives, if consumed. As many elements as alternatives. Names must match those in `alternatives`.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Can take different values depending on desired output.

- "estimate" Used for model estimation.
- "prediction" Used for model predictions.
- "validate" Used for validating input.
- "zero_LL" Used for calculating null likelihood.
- "conditionals" Used for calculating conditionals.
- "output" Used for preparing output after model estimation.
- "raw" Used for debugging.

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": A matrix with one row per observation, and means and s.d. of predicted consumptions.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": Not applicable.
- "conditionals": Same as "prediction".

- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by apollo_modelOutput).
- "raw": Same as "prediction".

---

apollo_mdcevInside *Calculates MDCEV likelihoods without an outside good.*

Description

Calculates the likelihood of a Multiple Discrete Continuous Extreme Value (MDCEV) model without an outside good.

Usage

```
apollo_mdcevInside(V, alternatives, alpha, gamma, sigma, cost, avail,
  continuousChoice, budget, functionality, minConsumption = NA,
  rows = "all")
```

Arguments

V	Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'.
alternatives	Character vector. Names of alternatives, elements must match the names in list 'V'.
alpha	Named list. Alpha parameters for each alternative. As many elements as alternatives.
gamma	Named list. Gamma parameters for each alternative. As many elements as alternatives.
sigma	Numeric scalar. Scale parameter of the model extreme value type I error.
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.
avail	Named list. Availabilities of alternatives, one element per alternative. Names of elements must match those in argument 'alternatives'. Value for each element can be 1 (scalar if always available) or a vector with values 0 or 1 for each observation. If all alternatives are always available, then user can just omit this argument.
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.
budget	Numeric vector. Budget for each observation.
functionality	Character. Can take different values depending on desired output. • "estimate" Used for model estimation.

- "prediction" Used for model predictions.
- "validate" Used for validating input.
- "zero_LL" Used for calculating null likelihood.
- "conditionals" Used for calculating conditionals.
- "output" Used for preparing output after model estimation.
- "raw" Used for debugging.

`minConsumption` Named list of scalars or numeric vectors. Minimum consumption of the alternatives, if consumed. As many elements as alternatives. Names must match those in `alternatives`.

`rows` Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

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": A matrix with one row per observation, and means and s.d. of predicted consumptions.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": Not applicable.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

`apollo_mdcevOutside` *Calculates MDCEV likelihoods with an outside good.*

Description

Calculates the likelihood of a Multiple Discrete Continuous Extreme Value (MDCEV) model with an outside good.

Usage

```
apollo_mdcevOutside(V, alternatives, alpha, gamma, sigma, cost, avail,
  continuousChoice, budget, functionality, minConsumption = NA,
  rows = "all")
```

Arguments

V	Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'.
alternatives	Character vector. Names of alternatives, elements must match the names in list 'V'.
alpha	Named list. Alpha parameters for each alternative, including for the outside good. As many elements as alternatives.
gamma	Named list. Gamma parameters for each alternative, including for the outside good. As many elements as alternatives.
sigma	Numeric scalar. Scale parameter of the model extreme value type I error.
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.
avail	Named list. Availabilities of alternatives, one element per alternative. Names of elements must match those in argument 'alternatives'. Value for each element can be 1 (scalar if always available) or a vector with values 0 or 1 for each observation. If all alternatives are always available, then user can just omit this argument.
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.
budget	Numeric vector. Budget for each observation.
functionality	Character. Can take different values depending on desired output. • "estimate" Used for model estimation. • "prediction" Used for model predictions. • "validate" Used for validating input. • "zero_LL" Used for calculating null likelihood. • "conditionals" Used for calculating conditionals. • "output" Used for preparing output after model estimation. • "raw" Used for debugging.
minConsumption	Named list of scalars or numeric vectors. Minimum consumption of the alternatives, if consumed. As many elements as alternatives. Names must match those in alternatives.
rows	Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is "all", equivalent to rep(TRUE, nObs).

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": A matrix with one row per observation, and means and s.d. of predicted consumptions.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": Not applicable.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

`apollo_mdcnev` *Calculates MDCNEV likelihoods with an outside good.*

Description

Calculates the likelihood of a Multiple Discrete Continuous Nested Extreme Value (MDCNEV) model with an outside good.

Usage

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

Arguments

`mdcnev_settings`

List of settings for the MDCEV model. It must include the following.

- V: Named list. Utilities of the alternatives. Names of elements must match those in argument 'alternatives'.
- alternatives: Character vector. Names of alternatives, elements must match the names in list 'V'.
- alpha: Named list. Alpha parameters for each alternative, including for the outside good. As many elements as 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.
- 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`.

- avail: Named list. Availabilities of alternatives, one element per alternative. Names of elements must match those in argument 'alternatives'. Value for each element can be 1 (scalar if always available) or a vector with values 0 or 1 for each observation. If all alternatives are always available, then user can just omit this argument.
- 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`.
- budget: Numeric vector. Budget for each observation.
- minConsumption: Named list of scalars or numeric vectors. Minimum consumption of the alternatives, if consumed. As many elements as alternatives. Names must match those in `alternatives`.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Can take different values depending on desired output.

- "estimate" Used for model estimation.
- "prediction" Used for model predictions.
- "validate" Used for validating input.
- "zero_LL" Used for calculating null likelihood.
- "conditionals" Used for calculating conditionals.
- "output" Used for preparing output after model estimation.
- "raw" Used for debugging.

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": A matrix with one row per observation, and means and s.d. of predicted consumptions.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": Not applicable.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

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	The number of draws to generate in each dimension
d	The number of dimensions to generate draws in
i	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 40B, 147 - 163.

Value

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

apollo_mnl *Calculates multinomial logit probabilities*

Description

Calculates probabilities of a multinomial logit model.

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.
- choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- V: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in `alternatives`.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Can take different values depending on desired output.

- "estimate": Used for model estimation.
- "prediction": Used for model predictions.
- "validate": Used for validating input.
- "zero_LL": Used for calculating null likelihood.
- "conditionals": Used for calculating conditionals.
- "output": Used for preparing output after model estimation.
- "raw": Used for debugging.

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 chosen alternative probability.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

Examples

```
### Load data
data(apollo_modeChoiceData)
database <- apollo_modeChoiceData
rm(apollo_modeChoiceData)
```

```
### Parameters
b = list(asc_1=0, asc_2=0, asc_3=0, asc_4=0, tt=0, tc=0, acc=0)

### List of utilities
V = list()
V[['car' ]] = b$asc_1 + b$tt*database$time_car  + b$tc*database$cost_car
V[['bus' ]] = b$asc_2 + b$tt*database$time_bus  + b$tc*database$cost_bus  +
              b$acc*database$access_bus
V[['air' ]] = b$asc_3 + b$tt*database$time_air  + b$tc*database$cost_air  +
              b$acc*database$access_air
V[['rail']] = b$asc_4 + b$tt*database$time_rail + b$tc*database$cost_rail +
              b$acc*database$access_rail

### MNL settings
mnl_settings <- list(
   alternatives = c(car=1, bus=2, air=3, rail=4),
   avail        = list(car=database$av_car, bus=database$av_bus,
                       air=database$av_air, rail=database$av_rail),
   choiceVar    = database$choice,
   V            = V
)

### Compute choice probabilities using MNL model
apollo_mnl(mnl_settings, functionality="estimate")
```

`apollo_modeChoiceData` *Simulated dataset of mode choice.*

Description

A simulated dataset containing 8000 mode choices among four alternatives.

Usage

```
apollo_modeChoiceData
```

Format

A data frame with 8000 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 observation.

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

SP_task Numeric. Consecutive ID of SP choice task.

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 in the air mode. 0 for none, 1 for a meal, 2 for wifi, 3 for meal and wifi.

service_rail Numeric. Additional services in the rail mode. 0 for none, 1 for a meal, 2 for wifi, 3 for meal and wifi.

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 one revealed preferences (RP) observation, and 15 stated preferences (SP) observations. There are 8000 choices in total. Data is simulated. Each observation contains attributes of the alternatives, availability of alternatives, and characteristics of the individuals.

Source

http://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 of options. It can include the following.

- printClassical: Boolean. TRUE for printing classical standard errors. TRUE 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.
- printDiagnostics: Boolean. TRUE for printing summary of choices in database and other diagnostics. TRUE by default.
- printCovar: Boolean. TRUE for printing parameters covariance matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. FALSE by default.
- printCorr: Boolean. TRUE for printing parameters correlation matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. FALSE by default.
- printOutliers: Boolean. TRUE for printing 20 individuals with worst average fit across observations. FALSE by default.
- printChange: Boolean. TRUE for printing difference between starting values and estimates. 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_nl *Calculates probabilities of a nested logit*

Description

Calculates probabilities of a nested logit model.

Usage

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

Arguments

`nl_settings` List of inputs of the NL 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.
- choiceVar: Numeric vector. Contains choices for all observations. It will usually be a column from the database. Values are defined in `alternatives`.
- V: Named list of deterministic utilities . Utilities of the alternatives. Names of elements must match those in `alternatives`.
- 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 fixed to 1 if excluded (recommended).
- 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`.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality` Character. Can take different values depending on desired output.

- "estimate": Used for model estimation.
- "prediction": Used for model predictions.
- "validate": Used for validating input.
- "zero_LL": Used for calculating null likelihood.
- "conditionals": Used for calculating conditionals.
- "output": Used for preparing output after model estimation.
- "raw": Used for debugging.

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`. And while setting it to another value is possible, it is not recommended.

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 chosen alternative probability.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "prediction".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "prediction".

Examples

```
### Load data
data(apollo_modeChoiceData)
database <- apollo_modeChoiceData
rm(apollo_modeChoiceData)

### Parameters
b = list(asc_1=0, asc_2=0, asc_3=0, asc_4=0, tt=0, tc=0, acc=0, lambda=0.5)

V = list()
V[['car' ]] = b$asc_1 + b$tt*database$time_car  + b$tc*database$cost_car
V[['bus' ]] = b$asc_2 + b$tt*database$time_bus  + b$tc*database$cost_bus  +
              b$acc*database$access_bus
V[['air' ]] = b$asc_3 + b$tt*database$time_air  + b$tc*database$cost_air  +
              b$acc*database$access_air
V[['rail']] = b$asc_4 + b$tt*database$time_rail + b$tc*database$cost_rail +
              b$acc*database$access_rail

### NL settings
nl_settings <- list(
   alternatives = c(car=1, bus=2, air=3, rail=4),
   avail        = list(car=database$av_car, bus=database$av_bus,
                       air=database$av_air, rail=database$av_rail),
```

```
   choiceVar    = database$choice,
   V            = V,
   nlNests      = list(root=1, public=b$lambda),
   nlStructure  = list(root=c("car", "public"), public=c("bus","air","rail"))
)

### Compute choice probabilities using NL model
apollo_nl(nl_settings, functionality="estimate")
```

`apollo_normalDensity` *Calculates density from a Normal distribution*

Description

Calculates density from a Normal distribution at a specific value with a specified mean and standard deviation.

Usage

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

Arguments

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

- outcomeNormal: Numeric vector. Dependant variable.
- xNormal: Numeric vector. Single explanatory variable.
- mu: Numeric scalar. Intercept of the linear model.
- sigma: Numeric scalar. Variance of error component of linear model to be estimated.
- rows: Boolean vector. Consideration of rows in the likelihood calculation, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE, nObs)`.

`functionality`
: Character. Can take different values depending on desired output.

- "estimate": Used for model estimation.
- "prediction": Used for model predictions.
- "validate": Used for validating input.
- "zero_LL": Used for calculating null likelihood.
- "conditionals": Used for calculating conditionals.
- "output": Used for preparing output after model estimation.
- "raw": Used for debugging.

Details

This function estimates 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.

- "estimate": vector/matrix/array. Returns the probabilities for the chosen value for each observation.
- "prediction": Not applicable.
- "validate": Boolean. Returns TRUE if all tests are passed.
- "zero_LL": Not applicable.
- "conditionals": Same as "estimate".
- "output": Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).
- "raw": Same as "estimate".

Examples

```
### Load data
xNormal <- runif(100)
outcomeNormal <- 1 + 2*xNormal + rnorm(100, mean=0, sd=0.5)

### Parameters
b = list(a=1, m=2)

### normalDensity settings
normalDensity_settings <- list(
   outcomeNormal     = outcomeNormal,
   xNormal     = 2*xNormal,
   mu    = 1,
   sigma = 0.5
)

### Compute choice probabilities using MNL model
apollo_normalDensity(normalDensity_settings, functionality="estimate")
```

apollo_ol *Calculates the probability of an ordered logit model*

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.

outcomeOrdered Numeric vector. Dependant 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.

V Numeric vector. A single explanatory variable (usually a latent variable). Must have the same number of rows as outcomeOrdered.

tau Numeric vector. Thresholds. As many as number of different levels in the dependent variable - 1. Extreme thresholds are fixed at -inf and +inf. No mixing allowed in thresholds.

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 `V`, and the last one with the highest value. If not provided, is assumed to be `1:(length(tau) + 1)`.

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 `choiceVar`. Default value is `"all"`, meaning all rows are considered in the calculation.

`functionality` Character. Can take different values depending on desired output.

"estimate" Used for model estimation.

"prediction" Used for model predictions.

"validate" Used for validating input.

"zero_LL" Used for calculating null likelihood.

"conditionals" Used for calculating conditionals.

"output" Used for preparing output after model estimation.

"raw" Used for debugging.

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.

"estimate" vector/matrix/array. Returns the probabilities for the chosen alternative for each observation.

"prediction" List. Returns a list with the probabilities for each possible outcome.

"validate" Boolean. Returns TRUE if all tests are passed.

"zero_LL" vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.

"conditionals" Same as "prediction".

"output" Same as "estimate" but also writes summary of choices into temporary file (later read by `apollo_modelOutput`).

"raw" Same as "prediction".

`apollo_outOfSample` *Out-of-sample 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 = "bfgs",
  maxIterations = 200, writeIter = FALSE, hessianRoutine = "numDeriv",
  printLevel = 3L, silent = TRUE), outOfSample_settings = list(nRep = 10,
  validationSize = 0.1))
```

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".

`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. Options defining the sampling procedure. The following are valid options.

  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.

  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 validaion samples is done at the individual level, not at the observation level.

Value

A matrix with the log-likelihood in both the estimation and validation samples. If the model has multiple components, the log-likelihood is reported for each of them. A more complete matrix also containing the estimates is written to a file called <model_name>_outOfSample.csv in the current working directory.

---

apollo_panelProd *Calculates product of panel observations.*

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. Can take different values depending on desired output.
	"estimate" For model estimation, returns probabilities of chosen alternatives.
	"prediction" For model predictions, returns probabilities of all alternatives.
	"validate" Validates input.
	"zero_LL" Return probabilities with all parameters at zero.
	"conditionals" For conditionals, returns probabilities of chosen alternatives.
	"output" Checks that the model is well defined.
	"raw" For debugging, returns probabilities of all alternatives

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

Probabilities at the individual level.

---

apollo_prediction *Predicts using an estimated model*

Description

Calculates apollo_probabilities with functionality="prediction" and extracts one element from the returned list.

Usage

```
apollo_prediction(model, apollo_probabilities, apollo_inputs,
  modelComponent = "model")
```

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".

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

`modelComponent` Character. Name of component of apollo_probabilities output to calculate predictions for. Default is "model", i.e. the whole model.

Details

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

Value

A vector containing predictions for component `modelComponent` of the model described in `apollo_probabilities`.

`apollo_prepareProb` *Checks likelihood*

Description

Checks that likelihoods, i.e. probabilities in the case of choice models, are in the appropiate 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. Can take different values depending on desired output of `apollo_probabilities`.

"estimate" For model estimation, returns probabilities of chosen alternatives.

"prediction" For model predictions, returns probabilities of all alternatives.

"validate" Validates input.

"zero_LL" Return probabilities with all parameters at zero.

"conditionals" For conditionals, returns probabilities of chosen alternatives.

"output" Checks that the model is well defined.

"raw" For debugging, returns probabilities of all alternatives

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 in any way.

Value

The likelihood (i.e. probability in the case of choice models) of the model in the appropriate form for the given functionality.

`apollo_printLog` *Returns the log of Apollo*

Description

Returns the apolloLog variable either as a list or as text.

Usage

```
apollo_printLog(apolloLog)
```

Arguments

`apolloLog`	Environment. It contains the character vectors of titles and content.

Details

The variable apolloLog is a list whose elements are character vectors with two elements. The first element is the title of the entry, and the second element is content of the entry. ApolloLog lives in the namespace environment of the Apollo package.

Value

A list or a scalar character variable.

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.

Examples

```
### Define starting values and fixed parameters
apollo_beta <- c(asc1=0, asc2=0, b1=0, b2=0)
apollo_fixed <- c("asc1")
## Not run:
## Not run:
### Update starting values
apollo_beta <- apollo_readBeta(apollo_beta, apollo_fixed,
                               "oldModelName", overwriteFixed=FALSE)
## End(Not run)

## End(Not run)
```

`apollo_saveOutput` *Saves estimation results to files.*

Description

Writes files in the working 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 of options. Valid options are the following.

- printClassical: Boolean. TRUE for printing classical standard errors. TRUE 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.
- printDiagnostics: Boolean. TRUE for printing summary of choices in database and other diagnostics. TRUE by default.
- printCovar: Boolean. TRUE for printing parameters covariance matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. TRUE by default.
- printCorr: Boolean. TRUE for printing parameters correlation matrix. If `printClassical=TRUE`, both classical and robust matrices are printed. TRUE by default.
- printOutliers: Boolean. TRUE for printing 20 individuals with worst average fit across observations. TRUE by default.
- printChange: Boolean. TRUE for printing difference between starting values and estimates. TRUE by default.
- saveEst: Boolean. TRUE for saving estimated parameters and standard errors to a CSV file. TRUE by default.
- saveCov: Boolean. TRUE for saving estimated correlation matrix to a CSV file. TRUE by default.
- saveCorr: Boolean. TRUE for saving estimated correlation matrix to a CSV file. TRUE 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.
- writeF12: Boolean. TRUE for writing results into an F12 file (ALOGIT format). FALSE by default.

Details

Estimation results are printed to different files in the working directory:

(modelName)_output.txt Text file with the output produced by function `apollo_modelOutput`.

(modelName)_estimates.csv CSV file with the estimated parameter values, their standars errors, and t-ratios.

(modelName)_covar.csv CSV file with the estimated classical covariance 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.

(modelName)_corr.csv CSV file with the estimated classical correlation matrix. Only when bayesian estimation was not used.

(modelName)_robcorr.csv CSV file with the estimated robust correlation matrix. Only when bayesian estimation was not used.

(modelName).F12 F12 file with model results. Compatible with ALOGIT.

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.

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".

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

`searchStart_settings`
List containing options for the search of starting values. The following are valid options.

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.

apolloBetaMin Vector. Minimum possible value of parameters when generating candidates. Ignored if smartStart is TRUE. Default is `apollo_beta - 0.1`.

apolloBetaMax Vector. Maximum possible value of parameters when generating candidates. Ignored if smartStart is TRUE. Default is `apollo_beta + 0.1`.

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.

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.

bfgsIter Numeric scalar. Number od BFGS iterations to perform at each stage to each remaining candidate. Default is 20.

Details

This function implements a simplified version of the algorithm proposed by Bierlaire, Themans, & Zufferey (2010). 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.

Value

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

`apollo_sharesTest` *Compares predicted and observed shares*

Description

Prints tables comparing the shares predicted by the model with the shares observed in the data.

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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".

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

`sharesTest_settings`
List of arguments. It must include the following.

- 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`.
- 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).
- modelComponent: Name of model component. Set to model by default.

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_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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
speedTest_settings	List containing options for the speed test. The following are valid options. • nDrawsTry: Numeric vector. Number of inter and intra-person draws to try. Default value is c(50, 100, 200). • nCoresTry: Numeric vector. Number of threads to try. Default is from 1 to the detected number of cores. • 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). Then it plots the estimation time for each combination. Estimation time grows at least linearly with number of draws, while time savings are decreasing 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 performance. 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 matrix with the average time per evaluation for each number of threads and draws combination. A graph is also plotted.

`apollo_splitDataDraws` *Splits data and draws for loading in cluster*

Description

Copies `apollo_inputs` as many times as cores, but each copy contains only part of `database` and `draws`.

Usage

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

Arguments

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

Details

Internal use only. This function is called by apollo_makeCluster.

Value

List. Each element is a copy of apollo_inputs, but with only a piece of `database` and `draws`.

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. 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 on a Stated Choice experiment (SC), providing a total of 3,492 observations. Each choice scenario includes two alternatives described in terms of travel time, cost, headway and interchanges. Additional information on respondents is available. This dataset comes from the following publication. Vrtic, Axhausen 2003, The impact of tilting trains in Switzerland: A route choice model of regional and long distance public transport trips. 82nd annual meeting of the transportation research board, Washington, DC.

Source

http://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, Crastes dit Sourd and Hess (2018) We want it all: experiences from a survey seeking to capture social network structures, lifetime events and short-term travel and activity planning. Transportation 2018 1-27.

indivID Numeric. Identification number of the individual.

day Numeric. Index of the day for each individual (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

http://www.apollochoicemodelling.com/

`apollo_unconditionals` *Returns draws for random parameters in a latent class model model*

Description

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

Usage

```
apollo_unconditionals(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 "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
`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 object, one per random coefficient. With inter-individual draws only, this will be a matrix, with one row per observation, 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_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.
	modelName Character. Name of the model. Used when saving the output to files.
	modelDescr Character. Description of the model. Used in output files.
	indivID Character. Name of column in the database with each decision maker's ID.
	mixing Boolean. TRUE for models that include random parameters.
	nCores Numeric>0. Number of cores to use in calculations of the model likelihood.
	seed Numeric. Seed for random number generation.
	HB Boolean. TRUE if using RSGHB for Bayesian estimation of model.
	noValidation Boolean. TRUE if user does not wish model input to be validated before estimation - FALSE by default.
	noDiagnostics Boolean. TRUE if user does not wish model diagnostics to be printed - FALSE by default.
	weights Character. Name of column in database containing weights for estimation.
`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_validateControl for details.
silent	Boolean. TRUE to keep 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)
```

Arguments

apollo_HB	List. Contains options for bayesian estimation. See doHB for details. Parameters modelname, gVarNamesFixed, gVarNamesNormal, gDIST, svN and FC are automatically set based on the other arguments of this function. It should also include a named character vector called hbDist identifying the distribution of each parameter to be estimated. Possible values are as follows. • "DNE": Parameter kept at its starting value (not estimated). • "F": Fixed (as in non-random) parameter. • "N": Normal. • "LN+": Positive log-normal. • "LN-": Negative log-normal. • "CN+": Positive censored normal. • "CN-": Negative censored normal. • "JSB": Johnson SB.
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.

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 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,
  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.

- modelName: Character. Name of the model. Used when saving the output to files.
- modelDescr: Character. Description of the model. Used in output files.
- indivID: Character. Name of column in the database with each decision maker's ID.
- mixing: Boolean. TRUE for models that include random parameters.
- nCores: Numeric>0. Number of threads (processors) to use in estimation of the model.
- workInLogs: Boolean. TRUE for higher numeric stability at the expense of computational time. Useful for panel models only. Default is FALSE.
- seed: Numeric. Seed for random number generation.
- HB: Boolean. TRUE if using RSGHB for Bayesian estimation of model.
- noValidation: Boolean. TRUE if user does not wish model input to be validated before estimation - FALSE by default.
- noDiagnostics: Boolean. TRUE if user does not wish model diagnostics to be printed - FALSE by default.
- panelData: Boolean. TRUE if using panelData data (created automatically by `apollo_validateControl`).
- weights: Character. Name of column in database containing weights for estimation.

`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. It should also include a named character vector called `hbDist` identifying the distribution of each parameter to be estimated. Possible values are as follows.

- "DNE": Parameter kept at its starting value (not estimated).
- "F": Fixed (as in non-random) parameter.
- "N": Normal.
- "LN+": Positive log-normal.
- "LN-": Negative log-normal.
- "CN+": Positive censored normal.
- "CN-": Negative censored normal.
- "JSB": Johnson SB.

`apollo_draws` List of arguments describing the inter and intra individual draws.

- interDrawsType: Character. Type of inter-individual draws ('halton','mlhs','pmc','sobol','sobolOwe
'sobolOwenFaureTezuka' or the name of an object loaded in memory).

- interNDraws: Numeric scalar (>=0). Number of inter-individual draws per individual. Should be set to 0 if not using them.
- interUnifDraws: Character vector. Names of uniform-distributed inter-individual draws.
- interNormDraws: Character vector. Names of normaly distributed inter-individual draws.
- intraDrawsType: Character. Type of intra-individual draws ('halton','mlhs','pmc','sobol','sobolOwe '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`).

`silent`
: Boolean. 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 ommited arguments.

Value

List grouping several required input for model estimation.

`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. Can take different values depending on desired output of apollo_probabilities. **"estimate"** For model estimation, returns probabilities of chosen alternatives. **"prediction"** For model predictions, returns probabilities of all alternatives. **"validate"** Validates input. **"zero_LL"** Return probabilities with all parameters at zero. **"conditionals"** For conditionals, returns probabilities of chosen alternatives. **"output"** Checks that the model is well defined. **"raw"** For debugging, returns probabilities of all alternatives

Details

This function should be called inside apollo_probabilities, near the end of it, just before return(P).

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 with the results of a model estimation.*

Description

Writes an F12 file 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. FALSE 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)
```

Arguments

`beta`	vector of parameters to be written.
`ll`	scalar representing the loglikelihood of the whole model.
`modelName`	Character. Name of the model.

Value

Nothing.

Index