Package 'apollo'

October 28, 2020

Type Package

Title Tools for Choice Model Estimation and Application

Version 0.2.1

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

URL http://www.apolloChoiceModelling.com

BugReports https://groups.google.com/d/forum/apollo-choice-modelling

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0), stats, utils

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

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2020-10-28 21:10:12 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_addCovariance *Adds covariance matrix to Apollo model*

Description

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

Usage

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

Arguments

model	A model object, as returned by apollo_estimate
apollo_inputs	List of settings. as returned by apollo_validateInputs

Value

model.

apollo_attach *Attaches predefined variables.*

Description

Attaches parameters and data to allow users to refer to individual variables by name without reference to the object that contains them. Also applies scaling if in use.

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. The use of apollo_attach is mandatory in models using scaling.

Value

Nothing.

apollo_avgInterDraws *Averages across inter-individual draws.*

Description

Averages individual-specific likelihood across inter-individual draws.

Usage

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

Arguments

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

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

Value

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

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

- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "validate": Returns P containing the likelihood of the model averaged across inter-individual draws. Drops all components except "model".
- "zero_LL": Returns P without changes.

---

apollo_avgIntraDraws *Averages across intra-individual draws.*

Description

Averages observation-specific likelihood across intra-individual draws.

Usage

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

Arguments

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

Value

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

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

---

apollo_basTest *Ben-Akiva & Swait test*

Description

Calculates the p-value for the Ben-Akiva & Swait test for non-nested models. The two models need to both be discrete choice, and estimated on the same data.

Usage

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

Arguments

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

Value

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

apollo_bootstrap *Bootstrap a model*

Description

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

Usage

```
apollo_bootstrap(
  apollo_beta,
  apollo_fixed,
  apollo_probabilities,
  apollo_inputs,
 estimate_settings = list(estimationRoutine = "bfgs", maxIterations = 200, writeIter =
  FALSE, hessianRoutine = "none", printLevel = 2L, silent = FALSE, maxLik_settings =
    list()),
 bootstrap_settings = list(nRep = 30, samples = NA, seed = 24, calledByEstimate =
    FALSE, recycle = TRUE)
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "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. hessianRoutine='none' by default.
bootstrap_settings	List. Options defining the sampling procedure. The following are valid options. • **nRep**: Numeric scalar. Number of times the model must be estimated with different samples. Default is 30.

- **samples**: Numeric matrix or data.frame. Optional argument. Must have as many rows as observations in the `database`, and as many columns as number of repetitions wanted. Each column represents a re-sample, and each element the number of times that observation must be included in the sample. If this argument is provided, then `nRep` is ignored. Note that this allows sampling at the observation rather than the individual level, which is not recommended for panel data.
- **seed**: Numeric scalar (integer). Random number generator seed to generate the bootstrap samples. Only used if `samples` is `NA`. Default is 24.
- **calledByEstimate**: Logical. TRUE if `apollo_bootstrap` is called by `apollo_estimate`. FALSE by default.
- **recycle**: Logical. If TRUE, the function will look for old output files and append new repetitions to them. If FALSE, output files will be overwritten.

Details

This function implements a basic block bootstrap. It estimates the model parameters on `nRep` number of different samples. Each new sample is constructed by sampling **with replacement** from the original full sample. Each new sample has as many individuals as the original sample, though some of them may be repeated. Sampling is done at the **individual** level, therefore if different individuals have different number of observations, each re-sample could have different number of observations.

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

For each sample, only the parameters and loglikelihood are estimated. Standard errors are not calculated (they may be in future versions). The composition of each re-sample is stored on a file, though it should be consistent across runs.

This function writes three different files to the working directory:

- `modelName_bootstrap_params.csv`: Records the estimated parameters, final loglikelihood, and number of observations on each re-sample
- `modelName_bootstrap_samples.csv`: Records the composition of each re-sample.
- `modelName_bootstrap_vcov.csv`: Variance-covariance matrix of the estimated parameters across re-samples.

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

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

Value

List with three elements.

- `estimates`: Matrix containing the parameter estimates for each repetition. As many rows as repetitions and as many columns as parameters.

- varcov: Covariance matrix of the estimated parameters across the repetitions.
- LL: Vector of final loglikelihoods of each repetition.

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

- **x_bootstrap_params.csv**: Table containing the parameter estimates, loglikelihood, and number of observations for each repetition.
- **x_bootstrap_samples.csv**: Table containing the description of the sample used in each repetition. Same format than bootstrap_settings$samples.
- **x_bootstrap_vcov**: Table containing the covariance matrix of estimated parameters across the repetitions.

---

apollo_checkArguments *Checks definitions Apollo functions*

Description

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

Usage

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

Arguments

apollo_probabilities	Function. Likelihood function as defined by the user.
apollo_randCoeff	Function. Defines the random components used inside apollo_probabilities.
apollo_lcPars	Function. Defines the class allocation probabilities and (optionally) the lists of parameters of each class for latent class models.

Details

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

Value

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

`apollo_choiceAnalysis` *Reports market share for subsamples*

Description

Compares market shares across subsamples in dataset, and 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.
- **rows**: Boolean vector. Consideration of rows to include, FALSE to exclude. Length equal to the number of observations (nObs). Default is `"all"`, equivalent to `rep(TRUE,nObs)`.

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

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

Details

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

Value

Silently returns a matrix containg the mean ehen chosen and un chose for each explanator, as well as the t-test comparing those means (H0: equivalence). The table is also writen to a file called `modelName_choiceAnalysis.csv`.

`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)`.
- **componentName**: Character. Name given to model component.

`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"`: Same as `"estimate"`
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

---

apollo_combineModels *Combines separate model components.*

Description

Combines model components to create likelihood for overall model.

Usage

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

Arguments

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

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

components Character vector. Optional argument. Names of elements in P that should be multiplied to construct the whole model likelihood. If a single element is provided, it is interpreted as a regular expression. Default is to include all components in P.

Details

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

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

Value

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

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

- "output": Returns P with an extra component called "model", which is the product of all other elements of P.
- "prediction": Returns P without changes.
- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "validate": Returns P with an extra component called "model", which is the product of all other elements of P.
- "zero_LL": Returns P with an extra component called "model", which is the product of all other elements of P.

---

apollo_combineResults *Write model results to file*

Description

Writes results from various models to a single CSV file.

Usage

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

Arguments

combineResults_settings
: List 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. FALSE by default.
  - printPVal: Boolean. TRUE for printing p-values. FALSE by default.
  - printT1: Boolean. If TRUE, t-test for H0: apollo_beta=1 are printed. FALSE by default.
  - 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.
  - sortByDate: Boolean. If TRUE, models are ordered by date. Default is TRUE.

Value

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

apollo_compareInputs *Compares the content of apollo_inputs to that in the global environment*

Description

Compares the content of apollo_inputs to that in the global environment

Usage

```
apollo_compareInputs(apollo_inputs)
```

Arguments

apollo_inputs	List of main inputs for model estimation as generated by apollo_validateInputs.

Value

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

apollo_conditionals *Calculates conditionals*

Description

Calculates posterior expected values (conditionals) of random 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_cppScript` *Generate C++ function to calculate V*

Description

Returns an R function that calls a C++ function which calculates V efficiently.

Usage

```
apollo_cppScript(apollo_probabilities, apollo_beta, apollo_inputs, V)
```

Arguments

`apollo_probabilities`	Likelihood function of the whole model.
`apollo_beta`	Named numeric vector of parameters to be estimated.
`apollo_inputs`	List of arguments and settings generated by apollo_validateInputs.
`V`	Named list of functions.

Details

V must be a list of functions.

Value

A function that receives three arguments:

```
# DataFrame db, NumericVector b, List drw
```

- `db`: Data.frame containing observations.
- `b`: Named numeric vector containing paramters to be estimated.
- `drw`: Named list containing draws, as contained inside `apollo_inputs`.

And returns a named list with numeric elements.

apollo_deltaMethod *Delta method*

Description

Applies the delta method to calculate the standard errors of transformations of parameters. If the bootstrap covariance matrix is available, it is used. If not, the robust covariance matrix is used.

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

Matrix contating calue, s.e. and t-ratio resulting from the operation. This is also printed to screen.

apollo_detach *Detaches parameters and the database.*

Description

Detaches variables attached by apollo_attach.

Usage

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

Arguments

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

Details

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

Value

Nothing.

apollo_dft *Calculate DFT probabilities*

Description

Calculate probabilities of a Decision Field Theory (DFT) 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 rescaled 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)`.
- **componentName**: Character. Name given to model component.

`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": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "estimate"
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "raw": Same as "prediction"

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_diagnostics *Pre-process input for multiple models return*

Description

Pre-process input for multiple models return

Usage

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

Arguments

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

Value

(invisibly) TRUE if no error happend during execution.

---

apollo_drugChoiceData *Simulated dataset of medication choice.*

Description

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

Usage

```
apollo_drugChoiceData
```

Format

A data frame with 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_dVdB` *Calculates gradients of utility functions*

Description

Calculates gradients (derivatives) of utility functions, considering definitions in apollo_randCoeff.

Usage

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

Arguments

`apollo_beta`	Named numeric vector of parameters.
`apollo_inputs`	List of Apollo main settings.
`V`	List of functions

Value

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

`apollo_el` *Calculates Exploded Logit probabilities*

Description

Calculates the probabilities of an Exploded Logit model and can also perform other operations based on the value of the `functionality` argument. The function calculates the probability of a ranking as a product of Multinomial Logit models with gradually reducing availability, where scale differences can be allowed for.

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. Use value -1 if a stage does not apply for a given observations (e.g. when some individuals have shorter rankings).
- `"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. 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)`.
- `"componentName"`: Character. Name given to model component.

`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 (NA).
- "validate": Same as "estimate"
- "zero_LL": vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- "conditionals": Same as "estimate"
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "raw": Same as "estimate"

---

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**: Logical. 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 `"analytic"` (default), `"numDeriv"` (to use the numeric routine in package numDeric), `"maxLik"` (to use the numeric routine in packahe maxLik), 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.
- **maxLik_settings**: List. Additional settings for maxLik. See argument `control` in maxBFGS, maxBHHH and maxNM for more details. Only used for maximum likelihood estimation.
- **numDeriv_settings**: List. Additional arguments to the Richardson method used by numDeriv to calculate the Hessian. See argument `method.args` in grad for more details.
- **bootstrapSE**: Numeric. Number of bootstrap samples to calculate standard errors. Default is 0, meaning no bootstrap s.e. will be calculated. Number must zero or a positive integer. Only used if `apollo_control$HB` is `FALSE`.
- **bootstrapSeed**: Numeric scalar (integer). Random number generator seed to generate the bootstrap samples. Only used if `bootstrapSE>0`. Default is 24.
- **silent**: Logical. If TRUE, no information is printed to the console during estimation. Default is FALSE.
- **scaleHessian**: Logical. If TRUE, parameters are scaled to 1 for Hessian estimation. Default is TRUE.

Details

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

Value

model object

---

apollo_estimateHB *Estimates model*

Description

Estimates a model using the likelihood function defined by apollo_probabilities.

Usage

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

Arguments

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

- functionality: Character. Can be either "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.
- **bootstrapSE**: Numeric. Number of bootstrap samples to calculate standard errors. Default is 0, meaning no bootstrap s.e. will be calculated. Number must zero or a positive integer. Only used if `apollo_control$HB` is `FALSE`.
- **bootstrapSeed**: Numeric scalar (integer). Random number generator seed to generate the bootstrap samples. Only used if `bootstrapSE>0`. Default is 24.
- **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.

`apollo_fitsTest` *Compares log-likelihood of model across categories*

Description

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

Usage

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

Arguments

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

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

- apollo_beta: Named numeric vector. Names and values of model parameters.
- apollo_inputs: List containing options of the model. See apollo_validateInputs.
- functionality: Character. Can be either "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 log-likelihood at the observation level for each category.

Value

Matrix with average log-likelihood at observation level 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_insertComponentName` *Adds componentName2 to model calls*

Description

Adds componentName2 to model calls

Usage

```
apollo_insertComponentName(e)
```

Arguments

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

Value

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

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

Description

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

Usage

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

Arguments

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

Details

It modifies the definition of the following models.

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

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

Value

Function f but with relevant expressions turned into function definitions.

apollo_insertRows *Inserts rows*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

apollo_insertScaling *Scales variables inside a function*

Description

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

Usage

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

Arguments

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

Value

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

`apollo_keepRows` *Keeps only some rows*

Description

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

Usage

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

Arguments

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

Value

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

`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`. • **componentName**: Character. Name given to model component.
`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. • "components" Returns `P` without changes.

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 models components, for each class.
- `"validate"`: Same as `"estimate"`, but also runs a set of tests on the given arguments.
- `"zero_LL"`: Same as `"estimate"`
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

`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_lcEM` *Uses EM for latent class model*

Description

Uses the EM algorithm for estimating a latent class model.

Usage

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

Arguments

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

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

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

`lcEM_settings` List. Options controlling the EM process.

- **stoppingCriterion**: Numeric. Convergence criterion. The EM process will stop when improvements in the log-likelihood fall below this value. Default is 10^-5.
- **EMmaxIterations**: Numeric. Maximum number of iterations of the EM algorithm before stopping. Default is 100. Used only if `apollo_control$HB` is FALSE. Default is 200.
- **postEM**: Numeric scalar. Determines the number of tasks performed by this function after the EM algorithm has converged. Can take values `0`, `1` or `2` only. If value is `0`, only the EM algorithm will be performed, and the results will be a model object without a covariance matrix (i.e. estimates only.). If value is `1`, after the EM algorithm the covariance matrix of the model will be calculated as well, and the result will be a model object with a covariance matrix. If value is `2`, after the EM algorithm, the estimated parameter values will be used as starting value for a maximum likelihood estimation process, which will render a model object with a covariance matrix. Performing maximum likelihood estimation after the EM algorithm is useful, as there may be room for further improvement. Default is `2`.
- **silent**: Boolean. If TRUE, no information is printed to the console during estimation. Default is FALSE.

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

Details

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

Value

model object

`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 value between two models and reports the corresponding p-value. The two models need to have been estimated on the same data, and one model needs to be nested within the other model.

Usage

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

Arguments

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

Value

LR-test p-value (invisibly)

apollo_makeCluster *Creates cluster for estimation.*

Description

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

Usage

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

Arguments

`apollo_probabilities`
: Function. Returns probabilities of the model to be estimated. Must receive three arguments:
  - apollo_beta: Named numeric vector. Names and values of model parameters.
  - apollo_inputs: List containing options of the model. See apollo_validateInputs.
  - functionality: Character. Can be either "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 messages are printed to the terminal. FALSE by default. It overrides `apollo_inputs$silent`.

`cleanMemory`
: Boolean. If TRUE, it saves apollo_inputs to disc, and removes database and draws from the apollo_inputs in .GlobalEnv and the parent environment.

Details

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

Value

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

`apollo_makeDraws` *Creates draws for models with mixing*

Description

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

Usage

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

Arguments

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

`silent`
: Boolean. If true, then no information is printed to console or default output. FALSE by default.

Details

Internal use only. Called by apollo_validateInputs. #' This function creates a list whose elements are the sets of draws requested by the user for use in a model with mixing. If the model does not include mixing, then it is not necessary to run this function. The number of draws 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_makeGrad *Creates gradient function.*

Description

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

Usage

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

Arguments

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

`apollo_logLike` Function to calculate the loglikelihood of the model, as created by apollo_makeLogLike If provided, the value of the analytical gradient will be compared to the value of the numerical gradient as calculated using apollo_logLike and the numDeriv package. If the difference between the two is bigger than 1 that the analytical gradient is wrong and NULL will be returned.

`validateGrad` Logical. If TRUE, it compares the value of the analytical gradient evaluated at apollo_beta against the numeric gradient (using numDeriv) at the same value. If the difference is bigger than 1 return NULL.

Details

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

Value

apollo_gradient function. It receives the following arguments

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

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

`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,
  cleanMemory = FALSE
)
```

Arguments

apollo_beta	Named numeric vector. Names and values for parameters.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "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.
cleanMemory	Logical. If TRUE, then apollo_inputs$draws and apollo_inputs$database are erased throughout the calling stack. Used to reduce memory usage in case of multithreading and a large database or number o draws.

Details

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

Value

apollo_logLike function.

---

apollo_mdcev *Calculates MDCEV likelihoods.*

Description

Calculates the 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 any outside good. As many elements as alternatives.
- `gamma`: Named list. Gamma parameters for each alternative, excluding any outside good. As many elements as inside good 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.
- `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`.
- `outside`: Character. Optional name of the outside good.
- `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)`.
- `componentName`: Character. Name given to model component.
- `nRep`: Numeric scalar. Number of simulations of the whole dataset used for forecasting. The forecast is the average of these simulations. Default is 100.

`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 observed consumption for each observation.
- `"prediction"`: A matrix with one row per observation, and columns indicating means and s.d. of continuous and discrete predicted consumptions.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: Not implemented. Returns a vector of NA with as many elements as observations.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"estimate"`

---

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.
- outside: Character. Alternative name for the outside good. Default is "outside"
- 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).
- componentName: Character. Name given to model component.

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 observed consumption for each observation.
- "prediction": A matrix with one row per observation, and columns indicating means and s.d. of continuous and discrete predicted consumptions.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "conditionals": Same as "estimate"
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "raw": Same as "estimate"

apollo_mixEM *Uses EM for models with continuous random coefficients*

Description

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

Usage

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

Arguments

apollo_beta	Named numeric vector. Names and values for parameters. These need to be provided in the following order. With K random parameters, K means for the underlying Normals, followed by the elements of the lower triangle of the Cholesky matrix, by row.
apollo_fixed	Character vector. Names (as defined in apollo_beta) of parameters whose value should not change during estimation.
apollo_probabilities	Function. Returns probabilities of the model to be estimated. Must receive three arguments: • apollo_beta: Named numeric vector. Names and values of model parameters. • apollo_inputs: List containing options of the model. See apollo_validateInputs. • functionality: Character. Can be either "estimate" (default), "prediction", "validate", "conditionals", "zero_LL", or "raw".
apollo_inputs	List grouping most common inputs. Created by function apollo_validateInputs.
mixEM_settings	List. Options controlling the EM process. • **stoppingCriterion**: Numeric. Convergence criterion. The EM process will stop when improvements in the log-likelihood fall below this value. Default is 10^-5. • **EMmaxIterations**: Numeric. Maximum number of iterations of the EM algorithm before stopping. Default is 100. Used only if apollo_control$HB is FALSE. Default is 200.

- **postEM**: Numeric scalar. Determines the number of tasks performed by this function after the EM algorithm has converged. Can take values `0`, `1` or `2` only. If value is `0`, only the EM algorithm will be performed, and the results will be a model object without a covariance matrix (i.e. estimates only.). If value is `1`, after the EM algorithm the covariance matrix of the model will be calculated as well, and the result will be a model object with a covariance matrix. If value is `2`, after the EM algorithm, the estimated parameter values will be used as starting value for a maximum likelihood estimation process, which will render a model object with a covariance matrix. Performing maximum likelihood estimation after the EM algorithm is useful, as there may be room for further improvement. Default is `2`.
- **silent**: Boolean. If TRUE, no information is printed to the console during estimation. Default is FALSE.
- **transforms**: List. Optional argument, with one entry per parameter, showing the inverse transform to return from beta to the underlying Normal. E.g. if the first parameter is specified as negative logormal inside apollo_randCoeff, then the entry in transforms should be transforms[[1]]=function(x) log(-x)

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

Details

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

Value

model object

`apollo_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)`.
- `componentName`: Character. Name given to model component.

`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 probability of the chosen alternative.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

---

apollo_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 or Scalar. TRUE or 1 for printing p-values for one-sided test, 2 for printing p-values for two-sided test, FALSE for not printing p-values. FALSE by default.
- `printT1`: Boolean. If TRUE, t-test for H0: apollo_beta=1 are printed. FALSE by default.
- `printDataReport`: Boolean. TRUE for printing summary of choices in database and other diagnostics. FALSE by default.
- `printModelStructure`: Boolean. TRUE for printing model structure. 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 or Scalar. TRUE for printing 20 individuals with worst average fit across observations. FALSE by default. If Scalar is given, this replaces the default of 20.
- `printChange`: Boolean. TRUE for printing difference between starting values and estimates. FALSE by default.
- `printFunctions`: Boolean. TRUE for printing apollo_control, apollo_randCoeff (when available), apollo_lcPars (when available) and apollo_probabilities. 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)`.
- `componentName`: Character. Name given to model component.

`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 probability of the chosen alternative.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: vector/matrix/array. Returns the probability of the chosen alternative when all parameters are zero.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

---

apollo_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)`.
- `componentName`: Character. Name given to model component.

`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 likelihood for each observation.
- `"prediction"`: Not implemented. Returns NA.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: Not implemented. Returns NA.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"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` List of numeric vector/matrix/3-dim arrays. Thresholds. As many elements as number of different levels in the dependent variable - 1. Extreme thresholds are fixed at -inf and +inf. Mixing is 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 `outcomeOrdered`, 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 `outcomeOrdered`. Default value is `"all"`, meaning all rows are considered in the calculation.
- `componentName` Character. Name given to model component.

`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 of vectors/matrices/arrays. Returns a list with the probabilities for all possible levels, with an extra element for the probability of the chosen alternative.
- "validate": Same as "estimate", but it also runs a set of tests to validate the function inputs.
- "zero_LL": Not implemented. Returns a vector of NA with as many elements as observations.
- "conditionals": Same as "estimate"
- "output": Same as "estimate" but also writes summary of input data to internal Apollo log.
- "raw": Same as "prediction"

---

apollo_op *Calculates the probability of an ordered probit model*

Description

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

Usage

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

Arguments

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

- 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/matrix/3-sim array. A single explanatory variable (usually a latent variable). Must have the same number of rows as outcomeOrdered.
- tau List of numeric vectors/matrices/3-dim arrays. Thresholds. As many as number of different levels in the dependent variable - 1. Extreme thresholds are fixed at -inf and +inf. Mixing is allowed in thresholds. Can also be a matrix with as many rows as observations and as many columns as thresholds.
- coding Numeric or character vector. Optional argument. Defines the order of the levels in outcomeOrdered. The first value is associated with the lowest level of outcomeOrdered, and the last one with the highest value. If not provided, is assumed to be 1:(length(tau) + 1).
- rows Boolean vector. TRUE if a row must be considered in the calculations, FALSE if it must be excluded. It must have length equal to the length of argument outcomeOrdered. Default value is "all", meaning all rows are considered in the calculation.

- `componentName` Character. Name given to model component.

`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.n Not implemented for ordered probit.
- **"conditionals"** Used for calculating conditionals.
- **"output"** Used for preparing output after model estimation.
- **"raw"** Used for debugging.

Details

This function estimates an ordered probit model of the type:

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

Where ϵ is distributed standard normal, and the values 1, 2, ..., $max(y)$ can be replaced by `coding[1],coding[2],...,codi`
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 of vectors/matrices/arrays. Returns a list with the probabilities for all possible levels, with an extra element for the probability of the chosen alternative.
- `"validate"`: Same as `"estimate"`, but it also runs a set of tests to validate the function inputs.
- `"zero_LL"`: Not implemented. Returns a vector of NA with as many elements as observations.
- `"conditionals"`: Same as `"estimate"`
- `"output"`: Same as `"estimate"` but also writes summary of input data to internal Apollo log.
- `"raw"`: Same as `"prediction"`

---

apollo_outOfSample *Cross-validation of fit (LL)*

Description

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

Usage

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

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

Details

A common way to test for overfitting of a model is to measure its fit on a sample not used during estimation that is, measuring its out-of-sample fit. A simple way to do this is splitting the complete

available dataset in two parts: an estimation sample, and a validation sample. The model of interest is estimated using only the estimation sample, and then those estimated parameters are used to measure the fit of the model (e.g. the log-likelihood of the model) on the validation sample. Doing this with only one validation sample, however, may lead to biased results, as a particular validation sample need not be representative of the population. One way to minimise this issue is to randomly draw several pairs of estimation and validation samples from the complete dataset, and apply the procedure to each pair.

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

This function writes two different files to the working directory:

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

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

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

Value

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

`apollo_panelProd` *Calculates product across observations from same individual.*

Description

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

Usage

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

Arguments

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

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

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

Details

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

Value

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

- `"components"`: Returns `P` without changes.
- `"conditionals"`: Returns `P` without averaging across draws. Drops all components except `"model"`.
- `"estimate"`: Returns `P` containing the likelihood of the model averaged across inter-individual draws. Drops all components except `"model"`.
- `"gradient"`: Returns `P` containing the gradient of the likelihood after applying the product rule across observations for the same individual.
- `"output"`: Returns `P` containing the likelihood of all model components averaged across inter-individual draws.
- `"prediction"`: Returns `P` containing the probabilities/likelihoods of all alternatives for all model components averaged across inter-individual draws.

- "preprocess": Returns P without changes.
- "raw": Returns P without changes.
- "validate": Returns P containing the likelihood of the model averaged across inter-individual draws. Drops all components except "model".
- "zero_LL": Returns P without changes.

---

apollo_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,
  prediction_settings = list(),
  modelComponent = NA
)
```

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.
prediction_settings	List of settings. It can have the following elements. • modelComponent Character. Name of component of apollo_probabilities output to calculate predictions for. Default is "model", i.e. the whole model. • runs Numeric. Number of runs to use for computing confidence intervals of predictions. • silent Boolean. If TRUE, this function won't print any output to screen.
modelComponent	**Deprecated**. Same as modelComponent inside prediction_settings.

Details

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

Value

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

`apollo_prepareProb` *Checks likelihood function*

Description

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

Usage

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

Arguments

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

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

Details

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

Value

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

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

---

apollo_preprocess *Pre-process input for multiple models return*

Description

Pre-process input for multiple models return

Usage

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

Arguments

inputs	List of settings
modelType	Character. Type of model, e.g. "mnl", "nl", "cnl", etc.
functionality	Character. Either "estimate","prediction","validate","zero_LL","conditionals","output","raw", or "preprocess". Only used for validation, it does not influence the return values.
apollo_inputs	List of main inputs to the model estimation process. See apollo_validateInputs.

Value

The returned object depends on the value of argument operation

---

`apollo_print` *Prints message to terminal*

Description

Prints message to terminal if apollo_inputs$silent is FALSE

Usage

```
apollo_print(txt, nSignifD = 4, widthLim = 11, highlight = FALSE)
```

Arguments

`txt`	Character, what to print.
`nSignifD`	Optional numeric integer. Minimum number of significant digits when printing numeric matrices. Default is 4.
`widthLim`	Optional numeric integer. Minimum width (in characters) of each column when printing numeric matrices. Default is 11
`highlight`	Optional logical. If TRUE, the message will be highlighted and will pause execution for 5 seconds.

Value

Nothing

`apollo_readBeta` *Reads parameters from file*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

---

apollo_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 or Scalar. TRUE or 1 for printing p-values for one-sided test, 2 for printing p-values for two-sided test, FALSE for not printing p-values. FALSE by default.
- printT1: Boolean. If TRUE, t-test for H0: apollo_beta=1 are printed. FALSE by default.
- printDataReport: Boolean. TRUE for printing summary of choices in database and other diagnostics. FALSE by default.
- printModelStructure: Boolean. TRUE for printing model structure. 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 or Scalar. TRUE for printing 20 individuals with worst average fit across observations. FALSE by default. If Scalar is given, this replaces the default of 20.
- printChange: Boolean. TRUE for printing difference between starting values and estimates. TRUE by default.
- printFunctions: Boolean. TRUE for printing apollo_control, apollo_randCoeff (when available), apollo_lcPars (when available) and apollo_probabilities. 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.

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

Value

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

apollo_setRows *Sets specified rows to a given value*

Description

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

Usage

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

Arguments

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

Value

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

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_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 individual, and one column per draw. With intra-individual draws, this will be a three-dimensional array, with one row per observation, inter-individual draws in the second dimension, and intra-individual draws in the third dimension.

apollo_validate *Pre-process input for multiple models return*

Description

Pre-process input for multiple models return

Usage

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

Arguments

inputs	List of settings
modelType	Character. Type of model, e.g. "mnl", "nl", "cnl", etc.
functionality	Character. Either "estimate","prediction","validate","zero_LL","conditionals","output","raw", or "preprocess". Only used for validation, it does not influence the return values.
apollo_inputs	List of main inputs to the model estimation process. See apollo_validateInputs.

Value

The returned object depends on the value of argument operation

apollo_validateControl *Validates apollo_control*

Description

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

Usage

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

Arguments

database	data.frame. Data used by model.
apollo_control	List. Options controlling the running of the code. • 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.
- workInLogs: Boolean. TRUE for increased numeric precision in models with panel data - FALSE by default.
- panelData: Boolean. TRUE if there are multiple obsrvations (i.e. rows) for each decision maker - Automatically set based on indivID by default.

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

Details

This function should be run before running apollo_validateData.

Value

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

---

apollo_validateData *Validates data*

Description

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

Usage

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

Arguments

database	data.frame. Data used by model.
apollo_control	List. Options controlling the running of the code. See ?apollo_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,
  silent = FALSE
)
```

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. • "F": Fixed - parameter kept at its starting value (not estimated). • "NR": Non-random parameter, i.e. with a generic value across individuals. • "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.
silent	Boolean. TRUE to keep the function from printing to the console. Default is FALSE.

Details

This function is only necessary when using bayesian estimation.

Value

Validated apollo_HB

---

apollo_validateInputs *Prepares input for* apollo_estimate

Description

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

Usage

```
apollo_validateInputs(
  apollo_beta = NA,
  apollo_fixed = NA,
  database = NA,
  apollo_control = NA,
  apollo_HB = NA,
  apollo_draws = NA,
  apollo_randCoeff = NA,
  apollo_lcPars = NA,
  recycle = FALSE,
  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. Avoid characters not allowed in file names, such as \, *, :, etc. • 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. Other settings to include are the following.

- hbDist *Mandatory* setting. A named character vector determining 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.
- constraintNorm Character vector. Constraints for *random* coefficients in bayesian estimation. Constraints can be written as "b1>b2", "b1<b2", "b1>0", or "b1<0".
- fixedA Named numeric vector. Contains the names and fixed mean values of random parameters. For example, c(b1=0) fixes the mean of b1 to zero.
- fixedD Named numeric vector. Contains the names and fixed variance of random parameters. For example, c(b1=1) fixes the variance of b1 to zero.

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

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

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

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

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

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

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

Details

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

Value

List grouping several required input for model estimation.

apollo_varcov *Calculates varcov matrix of an Apollo model*

Description

Calculates the Hessian, varcov matrix and s.e. of an Apollo model as defined buy its likelihood function and apollo_inputs list of settings. Performs automatic scaling for increased numeric stability.

Usage

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

Arguments

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

- **hessianRoutine**: Character. Name of routine used to calculate the Hessian. Valid values are "analytic", "numDeriv", "maxLik" or "none" to avoid estimating the Hessian and covariance matrix.
- **scaleBeta**: Logical. If TRUE (default), parameters are scaled by their own value before calculating the Hessian to increase numerical stability. However, the output is de-scaled, so they are in the same scale as the apollo_beta argument.
- **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.
- **apollo_logLike**: Function to calculate the loglikelihood of the model, as returned by apollo_makeLogLike.
- **apollo_grad**: Function to calculate the gradient of the model, as returned by apollo_makeGrad.
- **apollo_probabilities**: Function. Likelihood function of the model. Must receive three arguments:
  - apollo_beta: Named numeric vector.
  - apollo_inputs: List of settings.
  - functionality: Character.
- **apollo_inputs**: List of inputs to estimate a model, as returned by apollo_validateInputs.

Details

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

Value

List with the following elements

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

`apollo_varList` *Lists variables names and definitions used in V*

Description

Returns a list containing the names and definitions of variables used in V

Usage

```
apollo_varList(
  apollo_probabilities,
  apollo_beta,
  apollo_inputs,
  V,
  cpp = FALSE
)
```

Arguments

apollo_probabilities	Likelihood function of the whole model.
apollo_beta	Named numeric vector of parameters to be estimated.
apollo_inputs	List of arguments and settings generated by apollo_validateInputs.
V	Named list of functions.
cpp	Scalar logical. If TRUE, expressions are modified to match C++ syntax (e.g. x^y -> pow(x,y)). FALSE by default.

Details

It looks for variables in apollo_beta, apollo_randCoeff, draws, and apollo_probabilities. It returns them in a list ordered by origin.

Value

A list containing the following elements (all of type character):

- b: Vector with the names of variables contained in apollo_beta.
- x: Vector with the names of variables contained in database.
- d: Vector with the names of variables contained in draws.
- r: Matrix with the names (first column) and definitions (second column) of variables contained in apollo_randCoeff.
- p: Matrix with the names (first column) and definitions (second column) of variables contained in apollo_probabilities.
- v: Matrix with the names (first column) and definitions (second column) of utilities contained in V.

---

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

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