An introduction to *Apollo*

Key concepts & study plan

Experimental design

Data collection & processing

Model specification & estimation

Interpretation & application

Software for choice modelling

Software for estimation

- Model estimation is the key step in any choice study
- Users rely on software for this
- Several options exist, with big differences in capabilities

Software for choice modelling

Choice modelling: history

- Used across different disciplines for over four decades
- Initially, small number of (advanced) users
- Similarly, small number of software packages
- Explosion in number of users and fields since 1990s
- Ever broader group of users access to ever more advanced models

Software for choice modelling

Choice modelling: software split

- Fragmentation of community in terms of software, also along discipline lines
- Advanced users develop their own code
- Others split between commercial software and freeware tools
  - Commercial packages generally more powerful but with limitations in terms of available model structures or possibility for customisation
  - Freeware packages may have limitations in performance but benefit from more regular developments to accommodate new model structures

Software for choice modelling

ALogit

- Commercial software package
- By far the fastest package
- Can work with many 1,000s of alternatives and very large samples
- Widely used solution for large scale modelling
- Limited set of model structures, and restrictions on flexibility for model specification
- Limited ability for user to code models
- Classical estimation only
- Windows only

www.alogit.com

Software for choice modelling

Biogeme

- Free package, runs using Python
- Very powerful and flexible
- Large user base
- Runs on Windows and MacOS
- Classical estimation only
- Requires some Python knowledge

BIOGEME

https://biogeme.epfl.ch/

Software for choice modelling

NLogit

- Commercial software package
- Powerful solution for applied modelling
- Includes graphical interface
- Limited ability for user to code models
- Classical estimation only
- Windows only

https://www.limdep.com/

Software for choice modelling

Other packages

- ALogit, Biogeme and NLogit are longstanding packages developed by leading choice modellers
- Also many smaller packages in Matlab, Python, R and Stata
- Generally focus on just a small number of features
- Big differences in flexibility, speed and accessibility
- Incomplete list of examples
  - Matlab: DCE, Kenneth Train's code
  - R: mixl, RSGHB
  - Python: Pylogit
  - Stata: bayesmlogit, clogit, gmnl, mixlogit

Software for choice modelling

Black boxes are a problem!

- Many packages are black box tools
- User has little or no knowledge of what goes on "under the hood"
- Has made advanced models accessible to a broader group of users
- Disconnect between theory and software increases risk of misinterpretations and misspecifications
- Can also hide relevant nuances of modelling process and mistakenly give the impression that choice models are "easy tools" to use

Software for choice modelling

Bayesians vs the rest

- Existing software almost exclusively allows use of only either classical estimation techniques or Bayesian techniques
- Fragmentation again runs largely in parallel with discipline boundaries
- Has only served to further contribute to lack of interaction/dialogue between classical and Bayesian communities

An introduction to *Apollo*

Key concepts & study plan

Experimental design

Data collection & processing

Model specification & estimation

Interpretation & application

An introduction to *Apollo*

Existing limitations led to development of Apollo

- Culmination of 20 years of work
- Started with Ox code developed by Hess at Imperial College, translated into `R` at University of Leeds
- Combined with code developed by Palma at Pontificia Universidad Católica de Chile
- Major continuous further developments

- Couldn't base name on single model family as *Apollo* is more general
- Failure to come up with clever acronym led us to Greek mythology
- *Apollo* was the Greek god of prophecy

Approach in *Apollo*

Free access: *Apollo* is a completely free package which does not rely on commercial statistical software as a host environment.

Big community: *Apollo* relies on `R`, a free software environment for statistical computing and graphics, which is very widely used across disciplines and works well across different operating systems.

Transparent, yet accessible: *Apollo* is neither a blackbox nor does it require expert econometric skills. The user can see as much or as little detail of the underlying methodology as desired, but the link between inputs and outputs remains.

Ease of use: *Apollo* combines easy to use `R` functions with new intuitive functions without unnecessary jargon or complexity.

Approach in *Apollo*

Modular nature: *Apollo* uses the same code structure independently of whether the simplest multinomial logit model is to be estimated, or a complex structure using random coefficients and combining multiple model components.

Fully customisable: *Apollo* provides functions for many well known models but the user is able to add new structures and still make use of the overall code framework. This for example extends to coding expectation-maximisation routines.

Discrete and continuous: *Apollo* incorporates functions not just for commonly used discrete choice models but also for a family of models that looks jointly at discrete and continuous choices.

Approach in *Apollo*

Novel structures: *Apollo* goes beyond standard choice models by incorporating the ability to estimate Decision Field Theory (DFT) models, a popular accumulator model from mathematical psychology.

Classical and Bayesian: *Apollo* does not restrict the user to either classical or Bayesian estimation but easily allows changing from one to the other.

Easy multi-threading: *Apollo* allows users to split the computational work across multiple processors without making changes to the model code.

Not limited to estimation: *Apollo* provides a number of pre and post-estimation tools, including diagnostics as well as prediction/forecasting capabilities and posterior analysis of model estimates.

Files required

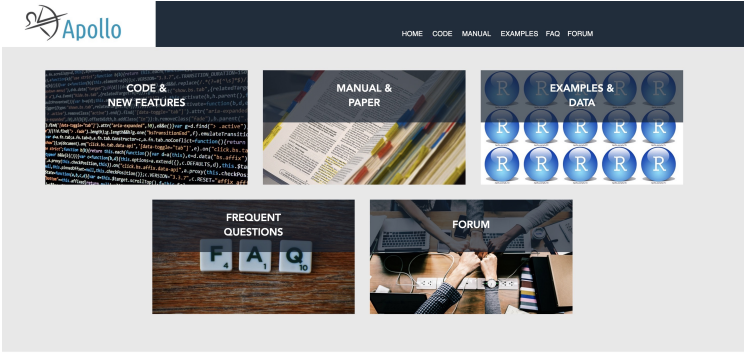
- A data file (various formats can be read into R)
  - not needed if data is read from a package
- A model file
  - contains all details on the model and tells R what routines to run
- Easier if data and model files are in the same directory

Data format

- *Apollo* uses "wide" format
- All necessary information to calculate likelihood of a single observation should be contained in a single row of the data
- In more practical terms, for an MNL model, this means that attributes for all alternatives should be contained in each row
- Most common format in choice modelling, uses less space, and more general in allowing for a mixture of different dependent variables in the same data
- Some other software uses "long" format, with one row per alternative
  - *Apollo* is not set up for such data, but reshaping the data is a straightforward task using `apollo_longToWide`

An introduction to *Apollo*

Resources: www.apollochoicemodelling.com



An introduction to *Apollo*

Documentation

- Website includes a detailed manual and shorter research paper
- In addition, syntax help for all functions is available directly in R
  - e.g. `?apollo_mnl`

R: Calculates Multinomial Logit probabilities - Find in Topic

apollo_mnl {apollo} R Documentation

Calculates Multinomial Logit probabilities

Description

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

Usage

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

Arguments

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

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

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

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