Apollo practical 1

Stephane Hess
stephane.hess@gmail.com

Apollo practical 1

Outline

1. An introduction to *Apollo*
2. First model in *Apollo*
3. Changes to a model: adding explanatory variables
4. Mode specific time coefficients
5. WTP space

An introduction to *Apollo*

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

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

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

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

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/

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/

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

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

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

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.

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.

Why R?

- Very powerful open source package
- Large number of packages available for econometric and statistical analysis
- Allows user to take the outputs from the choice model and process them directly into further output

Installing and updating *Apollo*

How do I install *Apollo*? enter `install.packages("apollo")` in R console

How do I update *Apollo*? You update *Apollo* by re-installing it

Error during installation that package not available for the R version I am running? Update R to the latest version and re-install *Apollo*

Why does it not work on my company laptop/desktop computer Often computers from big organisations will install R packages in shared libraries. As a general recommendation, always install packages in local libraries, i.e. in a folder in the local hard drive. You can see your active libraries by typing `.libPaths()`. This will list the active libraries. If the local library is, for example, the second one in the list, you should keep only that one by typing `.libPaths(.libPaths()[2])`. Then you should try installing *Apollo* again.

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

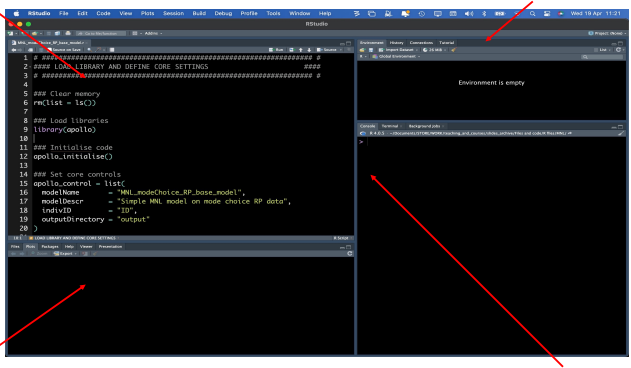
Data format

- *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`

A language and an interface

- R is a command line language
- Various shells exist to run R in
- This allows the user to have a script visible, which can then be run line by line or altogether
- We use RStudio
- Need to make sure to open your .R model files in RStudio, not directly in R

RStudio windows



First model in *Apollo*

First model in *Apollo*

Data from package: `apollo_modeChoiceData`

- Mode choice dataset, with four alternatives (car, bus, air, rail)
- 2 revealed preference choices per person
- Followed by 14 stated preference choices
- Travel time and cost for all, plus access time for non-car options
- Also a service quality attribute for air and rail in the SP data (1=no frills, 2=wifi, 3=food)
- 500 people, giving us 1,000 RP observations and 7,000 SP observations
- Not all modes available for all individuals
- Respondent characteristics: income, gender, purpose (business *vs* leisure)

First model in *Apollo*

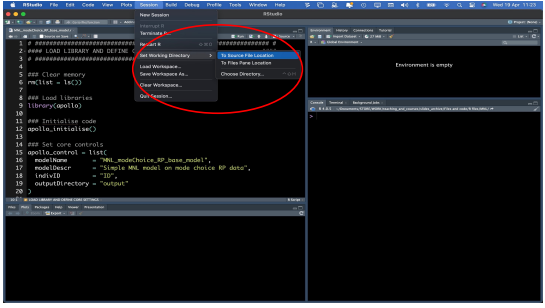
Start with a simple model

- Should always start with a base model
- No interactions, just main effects
- Shows us that the data works, and an important starting point for model comparison

First model in *Apollo*

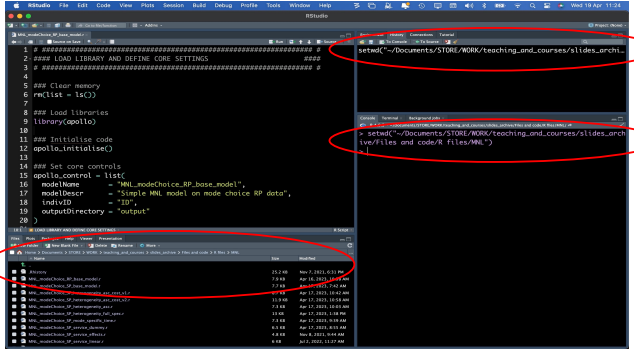
R file: MNL_modeChoice_RP_base_model.r

- First we need to set our working directory



First model in *Apollo*

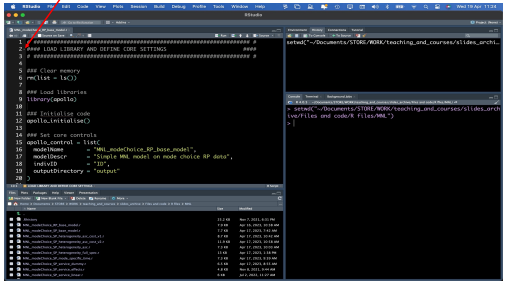
Visible in history, console and files panel



First model in *Apollo*

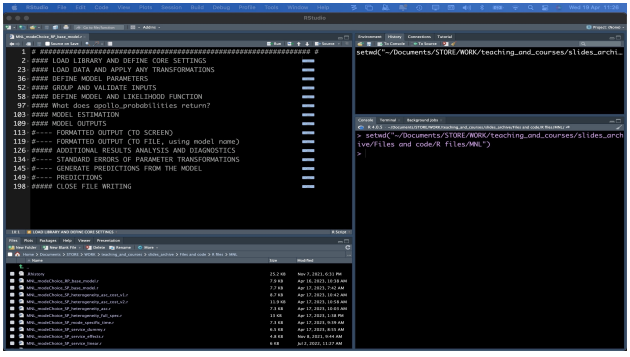
R syntax

- A line beginning with a # is a comment
- We use ### to indicate a comment, and lines starting with a single # are lines that a user can turn on/off
- Lines ending in #### or ---- are section headings
- Can collapse code into section headings by clicking on downwards arrow in margin



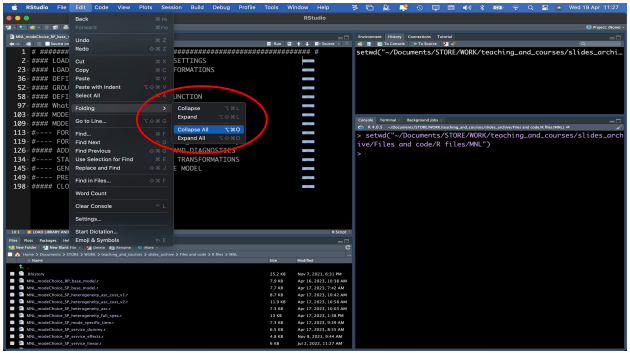
First model in *Apollo*

View with all sections collapsed



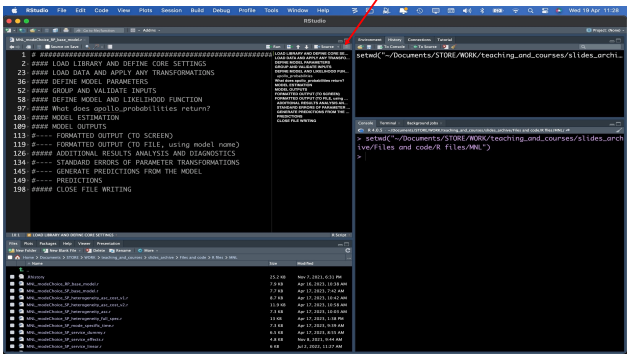
First model in *Apollo*

Also possible via menu



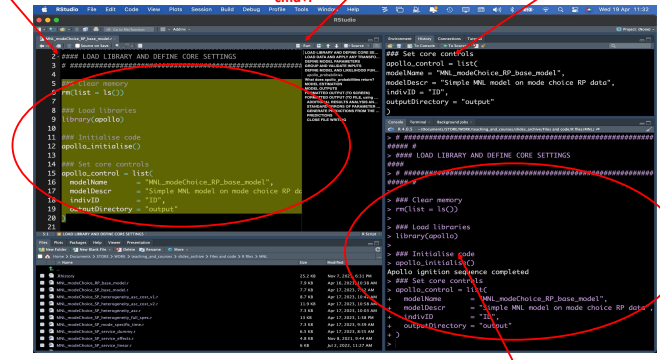
First model in *Apollo*

Document outline is useful (sections/functions)



First model in *Apollo*

Running our first few lines (not whole script!)



First model in *Apollo*

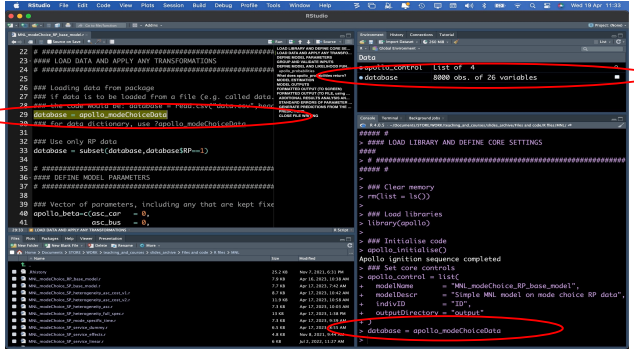
What have we just done?

- Working directory is set so that R can find the files we're working with
- Then we cleared the memory from objects we might have loaded previously
- We loaded the *Apollo* package
- apollo_initialise removes attachments and closes any file writing that is still open
- Next have a number of important settings

```
apollo_control = list(
  modelName       = "MNL_modeChoice_RP_base_model",
  modelDescr      = "Simple MNL model on mode choice RP data",
  indivID         = "ID",
  outputDirectory = "output"
)
```

First model in *Apollo*

Loading the data: from the package in this case

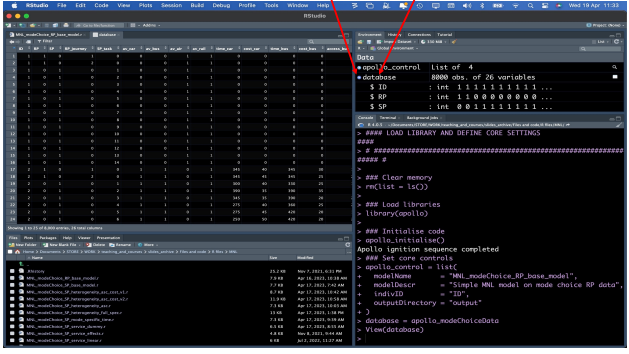


First model in *Apollo*

Can see details of data

Button shows contents

Clicking on name opens data in preview

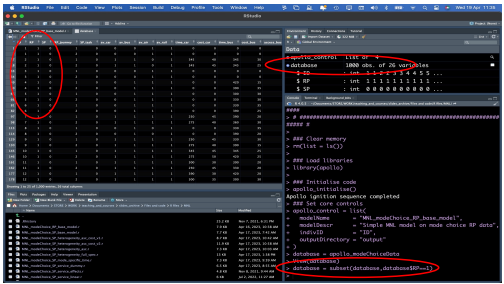


First model in *Apollo*

Uses RP data only (two choice observations per person)

- This uses the subset of database where the RP colum has a value of 1 (== is a boolean operator, equals 1 if true, 0 if false)

```
database = subset(database,database$RP==1)
```



First model in *Apollo*

Defining parameters used by model

- Need to define starting values for all parameters

```
apollo_beta=c(asc_car    = 0,
              asc_bus    = 0,
              asc_air    = 0,
              asc_rail   = 0,
              b_tt       = 0,
              b_access   = 0,
              b_cost     = 0)
```

- Possible to fix some parameters to their starting values, ASC for car in our model

```
apollo_fixed = c("asc_car")
```

- Now run those lines of code

First model in *Apollo*

Validating data and control

- Checks whether all settings have been set and whether ID column exists in data
- Also sorts the data by ID

```
apollo_inputs = apollo_validateInputs()
```

First model in *Apollo*

The apollo_probabilities function

- Core part of the code
- Calculates and returns probabilities for individual choices
- Need to run function as a whole block
- Inputs are controlled by code (so user should not change them):
  - apollo_beta: parameter values used in evaluation of probabilities
  - apollo_inputs: all other model inputs (e.g. database, apollo_control)
  - functionality: mainly estimate or prediction, but also conditionals and output

First model in *Apollo*

Initialisation

- "Attach" data and parameters, meaning they can be referred to by name

```
apollo_attach(apollo_beta, apollo_inputs)
on.exit(apollo_detach(apollo_beta, apollo_inputs))
```

- Create a list called P which we will use for our probabilities
- Lists are a flexible R object , whose individual elements can be scalars, matrices, ...

```
P = list()
```

First model in *Apollo*

MNL model component

- *Apollo* contains a predefined function for MNL models

```
### List of utilities: these must use the same names as in mnl_settings, order is
    ↪irrelevant
V = list()
V[["car"]]  = asc_car  + b_tt * time_car                                  + b_cost * cost_car
V[["bus"]]  = asc_bus  + b_tt * time_bus  + b_access * access_bus  + b_cost * cost_bus
V[["air"]]  = asc_air  + b_tt * time_air  + b_access * access_air  + b_cost * cost_air
V[["rail"]] = asc_rail + b_tt * time_rail + b_access * access_rail + b_cost * cost_rail

### Define settings for MNL model component
mnl_settings = list(
alternatives  = c(car=1, bus=2, air=3, rail=4),
avail         = list(car=av_car, bus=av_bus, air=av_air, rail=av_rail),
choiceVar     = choice,
utilities     = V
)
```

First model in *Apollo*

Moving towards probabilities

- We now use the predefined function to calculate MNL probabilities

```
P[["model"]] = apollo_mnl(mnl_settings, functionality)
```

- At this point, this contains one row per observation, and possibly multiple columns if we use random draws (in mixture models)
- For models with multiple choices per person, we want to work with the sequence of choices per individual, so we take the product across choices, i.e. $P_n = \prod_{t=1}^{T} P_{nt}$

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

- We finally use apollo_prepareProb to prepare the output of the apollo_probabilities function and return it

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

First model in *Apollo*

The apollo_probabilities function: illustration

- At the moment, our parameters are all at zero (the starting values)
- Let's see what happens if we use the model to calculate probabilities of the choices

```
> apollo_probabilities(apollo_beta,apollo_inputs,functionality="estimate")
  [1] 0.2500000 0.1111111 0.2500000 0.1111111 0.1111111
  [6] 0.1111111 0.0625000 0.2500000 0.0625000 0.0625000
 [11] 0.0625000 0.0625000 0.0625000 0.2500000 0.1111111
 [16] 0.1111111 0.1111111 0.0625000 0.1111111 0.1111111
```

- The values depend on what alternatives are available
- The first person only has air and rail available, so the probability for the actual choice (rail in both choices) is 0.5, and $0.5 \cdot 0.5 = 0.25$

First model in *Apollo*

Running the model

- Read in all lines up to the end of the apollo_probabilities function
- Then we run the following line

```
model = apollo_estimate(apollo_beta, apollo_fixed, apollo_probabilities, apollo_inputs)
```

- Here, we just use the default settings, but user can change estimation routine, etc

First model in *Apollo*

Running the model: first part

- First runs some diagnostics, checking names of alternatives match utilities, etc

```
Testing probability function (apollo_probabilities)
```

- Then presents some overview of choices

```
Overview of choices for MNL model component:
                                    car    bus    air   rail
Times available                  778.00 902.00 752.00 874.00
Times chosen                     332.00 126.00 215.00 327.00
Percentage chosen overall         33.20  12.60  21.50  32.70
Percentage chosen when available  42.67  13.97  28.59  37.41
```

First model in *Apollo*

Running the model: actual estimation

- LL starts at value obtained with our starting values, then improves until convergence
- We minimise the negative of the LL, which is the same as maximising the LL

```
Pre-processing likelihood function...
Testing influence of parameters
Starting main estimation
BGW using analytic model derivatives supplied by caller...

Iterates will be written to:
 output/MNL_modeChoice_RP_base_model_iterations.csv      it      nf       F
      ↪RELDF     PRELDF     RELDX     MODEL stppar
     0      1 1.170860076e+03
     1      4 1.109496887e+03 5.241e-02 5.578e-02 1.00e+00    G    1.62e+00
     2      5 1.051799389e+03 5.200e-02 5.486e-02 4.63e-01    G    2.68e-01
     3      6 1.031148818e+03 1.963e-02 1.851e-02 5.56e-01    S    0.00e+00
     4      7 1.030967733e+03 1.756e-04 1.730e-04 4.02e-02    G    0.00e+00
     5      8 1.030966941e+03 7.681e-07 7.688e-07 3.02e-03    S    0.00e+00
     6      9 1.030966937e+03 4.312e-09 4.142e-09 1.63e-04    S    0.00e+00
     7     10 1.030966937e+03 1.385e-11 1.305e-11 5.03e-06    G    0.00e+00

***** Relative function convergence *****
```

First model in *Apollo*

Running the model: post-convergence

```
Calculating log-likelihood at equal shares (LL(0)) for applicable models...
Calculating log-likelihood at observed shares from estimation data (LL(c)) for
  applicable models...
Calculating LL of each model component...
Calculating other model fit measures
Computing covariance matrix using analytical gradient.
 0%....25%....50%....75%.100%
Negative definite Hessian with maximum eigenvalue: -4.701829
Computing score matrix...
```

First model in *Apollo*

Functions for output

- Output to screen (apollo_modelOutput) and file (apollo_saveOutput)
- Need a model object as input
- Both can take a list of settings, or just use defaults

First model in *Apollo*

Outputs: model details

```
Model name                              : MNL_modeChoice_RP_base_model
Model description                       : Simple MNL model on mode choice RP data
Model run at                            : 2023-11-17 23:10:22.655457
Estimation method                       : bgw
Model diagnosis                         : Relative function convergence
Optimisation diagnosis                  : Maximum found
     hessian properties                 : Negative definite
     maximum eigenvalue                 : -4.701829
     reciprocal of condition number     : 4.26742e-07
Number of individuals                   : 500
Number of rows in database              : 1000
Number of modelled outcomes             : 1000
```

First model in *Apollo*

Outputs: model fit

```
LL(start)                                 : -1170.86
LL at equal shares, LL(0)                 : -1170.86
LL at observed shares, LL(C)              : -1085.14
LL(final)                                 : -1030.97
Rho-squared vs equal shares               :  0.1195
Adj.Rho-squared vs equal shares           :  0.1144
Rho-squared vs observed shares            :  0.0499
Adj.Rho-squared vs observed shares        :  0.0472
AIC                                       :  2073.93
BIC                                       :  2103.38
```

First model in *Apollo*

Outputs: parameters, time, iterations

```
Estimated parameters                : 6
Time taken (hh:mm:ss)               :  00:00:1.09
     pre-estimation                 :  00:00:0.54
     estimation                     :  00:00:0.37
          initial estimation        :  00:00:0.34
          estimation after rescaling:  00:00:0.03
     post-estimation                :  00:00:0.18
Iterations                          :  8
     initial estimation             :  7
     estimation after rescaling     :  1
```

First model in *Apollo*

Estimates

Estimates:

	Estimate	s.e.	t.rat.(0)	Rob.s.e.	Rob.t.rat.(0)
asc_car	0.000000	NA	NA	NA	NA
asc_bus	−1.290273	0.159172	−8.106	0.157807	−8.176
asc_air	−0.434473	0.409593	−1.061	0.421850	−1.030
asc_rail	−0.658200	0.234678	−2.805	0.243331	−2.705
b_tt	−0.006299	0.001287	−4.896	0.001360	−4.633
b_access	−0.007732	0.005935	−1.303	0.005807	−1.332
b_cost	−0.032041	0.003211	−9.979	0.003117	−10.278

First model in *Apollo*

Covariance and correlation matrix (in saved output)

- Look out for any really high correlations between coefficients (say above 0.95)
- Would imply two parameters explaining the same thing (opposite if negative)

Robust correlation matrix:

	asc_bus	asc_air	asc_rail	b_tt	b_access	b_cost
asc_bus	1.00000	0.03099	−0.14071	−0.437504	−0.490217	0.06161
asc_air	0.03099	1.00000	0.88117	0.658441	−0.673149	−0.12187
asc_rail	−0.14071	0.88117	1.00000	0.838270	−0.365026	0.04522
b_tt	−0.43750	0.65844	0.83827	1.000000	−0.002725	0.31855
b_access	−0.49022	−0.67315	−0.36503	−0.002725	1.000000	0.15505
b_cost	0.06161	−0.12187	0.04522	0.318547	0.155054	1.00000

First model in *Apollo*

MRS calculations for RP model

- Use `apollo_deltaMethod` to compute the VTT and its standard errors, and difference between in vehicle and access time
- Calculate VTT both in base units (£ and minutes) as well as in £ and hours
- Also instruct `R` to dump this output to a text file

```
> apollo_sink()
> apollo_deltaMethod(model,
+                    deltaMethod_settings = list(
+                      expression=c(VTT_per_minute="b_tt/b_cost",
+                                   VTT_per_hour="60*b_tt/b_cost",
+                                   b_tt_vs_access="b_tt-b_access"
+ )))

Running Delta method computation for user-defined function:

      Expression    Value Robust s.e. Rob t-ratio (0)
 VTT_per_minute   0.1966       0.0406            4.84
   VTT_per_hour 11.7949       2.4366            4.84
 b_tt_vs_access  0.0014       0.0060            0.24
```

Changes to a model: adding explanatory variables

Changes to a model: adding explanatory variables

Good practice

- Rule 1: whenever you change a model specification, save it as a new file!
- Rule 2: use the same text for the modelname as for the filename, to ensure output files have the same name as the model!

Changes to a model: adding explanatory variables

Moving to SP data

- We will now make use of the larger SP sample
- Change file name to MNL_modeChoice_SP_base_model.r
- Update `modelName` in `apollo_control`
- Think about what you need to change to use the SP part of the sample

Changes to a model: adding explanatory variables

Outputs of model estimation

- More data, and thus also more negative LL and smaller standard errors

```
LL(start)                                  : -8196.02
LL at equal shares, LL(0)                  : -8196.02
LL at observed shares, LL(C)               : -6706.94
LL(final)                                  : -5788.59
Rho-squared vs equal shares                :   0.2937
Adj.Rho-squared vs equal shares            :   0.293
Rho-squared vs observed shares             :   0.1369
Adj.Rho-squared vs observed shares         :   0.1365
AIC                                        :  11589.18
BIC                                        :  11630.3

Estimates:
          Estimate        s.e.   t.rat.(0)    Rob.s.e. Rob.t.rat.(0)
asc_car    0.00000          NA          NA          NA          NA
asc_bus   -2.05474    0.074994   -27.3987    0.092092   -22.3118
asc_air    0.09090    0.171812     0.5290    0.183719     0.4948
asc_rail  -0.20922    0.097365   -2.1488    0.105401   -1.9850
b_tt      -0.01031  5.3691e-04   -19.1985  5.7363e-04   -17.9695
b_access  -0.01860    0.002483   -7.4902    0.002472   -7.5233
b_cost    -0.05456    0.001418   -38.4621    0.001718   -31.7592
```

Changes to a model: adding explanatory variables

MRS calculations for SP model

- MRS very similar to RP results

```
> apollo_deltaMethod(model,
+                    deltaMethod_settings = list(
+                      expression=c(VTT_per_minute="b_tt/b_cost",
+                                   VTT_per_hour="60*b_tt/b_cost",
+                                   b_tt_vs_access="b_tt-b_access"
+ )))

Running Delta method computation for user-defined function:

      Expression    Value Robust s.e. Rob t-ratio (0)
 VTT_per_minute   0.1889       0.0100           18.83
   VTT_per_hour 11.3360       0.6019           18.83
 b_tt_vs_access  0.0083       0.0025            3.26
```

Changes to a model: adding explanatory variables

Instructions

- Include categorical service quality for air and rail (1=no frills, 2=wifi, 3=food)
- Three levels of difficulty:
  - level 1 treat as a continuous (linear) attribute, same as e.g. travel time
  - level 2 treat as categorical, using dummy coding (remember normalisation, and convince yourself that the choice of base is arbitrary)
    - can create a new variable for each level before `apollo_probabilities` (e.g. `database$service_air_wifi=(database$service_air==2)`)
    - or use the code directly inside the utilities (e.g. `...+beta_wifi*(service_air==2)+...`)
  - level 3 contrast with effects coding (base level β is negative sum of others)
- Can base these on `MNL_modeChoice_SP_base_model.r`
  - BUT CHANGE THE NAME, and add to the bit on WTP calculation at the end

Changes to a model: adding explanatory variables

Results (try out `apollo_combineResults()`)

Model name	MNL_modeChoice_SP_service_linear	MNL_modeChoice_SP_service_dummy	MNL_modeChoice_SP_service_effects
Estimated parameters	7	8	8
LL(final)	-5761.741	-5615.391	-5615.391
Adj.Rho-square (0)	0.2962	0.3139	0.3139
AIC	11537.48	11246.78	11246.78
BIC	11585.46	11301.61	11301.61

	estimate	Rob.t-ratio(0)	estimate	Rob.t-ratio(0)	estimate	Rob.t-ratio(0)
asc_car	0	NA	0	NA	0	NA
asc_bus	-2.0507	-22.25	-2.0429	-22.15	-2.0429	-22.15
asc_air	-0.3053	-1.57	-0.5878	-2.98	-0.1334	-0.7
asc_rail	-0.6061	-5.03	-0.862	-7.32	-0.4076	-3.73
b_tt	-0.0106	-18.38	-0.0121	-20.24	-0.0121	-20.24
b_access	-0.0188	-7.6	-0.0199	-8	-0.0199	-8
b_cost	-0.0555	-32.15	-0.0587	-34.95	-0.0587	-34.95
b_service	0.1826	7.43	NA	NA	NA	NA
b_no_frills	NA	NA	0	NA	NA	NA
b_wifi	NA	NA	0.9515	17.25	0.4971	16.1
b_food	NA	NA	0.4117	7.8	-0.0427	-1.45

Changes to a model: adding explanatory variables

WTP from linear model

- Service quality is a desirable attribute, so we can change the sign in the WTP calculations (but of course we could just do this manually)
- Small change in VTT, but linear assumption for service quality clearly not reasonable

```
> apollo_deltaMethod(model,
+                    deltaMethod_settings = list(
+                      expression=c(VTT_per_minute="b_tt/b_cost",
+                                   VTT_per_hour="60*b_tt/b_cost",
+                                   WTP_wifi="-b_service/b_cost",
+                                   WTP_food="-2*b_service/b_cost"
+ )))
Running Delta method computation for user-defined function:

      Expression    Value Robust s.e. Rob t-ratio (0)
 VTT_per_minute   0.1916       0.0099           19.26
   VTT_per_hour 11.4931       0.5968           19.26
       WTP_wifi  3.2902       0.4421            7.44
       WTP_food  6.5803       0.8843            7.44
```

Changes to a model: adding explanatory variables

WTP from dummy coded model

- Small changes for VTT, but we now see that Wifi is better than food
- With categorical attributes, can only work in differences

```
> apollo_deltaMethod(model,
+                    deltaMethod_settings = list(
+                      expression=c(VTT_per_minute="b_tt/b_cost",
+                                   VTT_per_hour="60*b_tt/b_cost",
+                                   WTP_wifi="-(b_wifi-b_no_frills)/b_cost",
+                                   WTP_food="-(b_food-b_no_frills)/b_cost"
+                    )))
The expression WTP_wifi includes parameters that were fixed in estimation: b_no_frills
These have been replaced by their fixed values, giving: -(b_wifi-0)/b_cost
The expression WTP_food includes parameters that were fixed in estimation: b_no_frills
These have been replaced by their fixed values, giving: -(b_food-0)/b_cost

Running Delta method computation for user-defined function:

     Expression    Value Robust s.e. Rob t-ratio (0)
VTT_per_minute   0.2053       0.0095           21.56
  VTT_per_hour  12.3200       0.5714           21.56
      WTP_wifi  16.2084       1.0033           16.15
      WTP_food   7.0127       0.8949            7.84
```

Changes to a model: adding explanatory variables

Dummy *vs* effects coding

- Only asc_air, asc_rail, b_wifi and b_food change
- But all differences remain the same
- The change in ASC is what has caused confusion in the literature
  - But only differences in utility matter, so there is no confounding
  - And the dummy coded one has a clear base (as opposed to an unweighted average)

	dummy	effects
asc_car	0	0
asc_bus	-2.0429	-2.0429
asc_air	-0.5878	-0.1334
asc_rail	-0.862	-0.4076
b_tt	-0.0121	-0.0121
b_access	-0.0199	-0.0199
b_cost	-0.0587	-0.0587
b_no_frills	0	-0.4544
b_wifi	0.9515	0.4971
b_food	0.4117	-0.0427
b_food-b_no_frills	0.4117	0.4117
b_wifi-b_no_frills	0.9515	0.9515
b_wifi-b_food	0.5398	0.5398

Changes to a model: adding explanatory variables

Likelihood ratio test: linear *vs* base

- The model with linear service quality is the more general model
- If service quality goes to zero, the model collapses to our base model

```
> apollo_lrTest("MNL_modeChoice_SP_base_model",model)
                                              LL par
MNL_modeChoice_SP_base_model      -5788.59    6
MNL_modeChoice_SP_service_linear  -5761.74    7
Difference                           26.85    1

Likelihood ratio test-value:     53.7
Degrees of freedom:              1
Likelihood ratio test p-value:   2.336e-13
```

- Improvement is large enough to allow us to reject H_0 that the models are equivalent
- This function can also be run for two previously saved models (rather than one in memory against a saved one)

Changes to a model: adding explanatory variables

Likelihood ratio test: dummy *vs* base

- Can also compare the dummy coded model to the model without service quality
- If both service quality parameters go to zero, the model collapses to our base model

```
> apollo_lrTest("MNL_modeChoice_SP_base_model",model)
                                              LL  par
MNL_modeChoice_SP_base_model     -5788.59    6
MNL_modeChoice_SP_service_dummy  -5615.39    8
Difference                         173.20    2

Likelihood ratio test-value:     346.4
Degrees of freedom:              2
Likelihood ratio test p-value:   6.028e-76
```

- Improvement is large enough to allow us to reject H_0 that the models are equivalent

Changes to a model: adding explanatory variables

Ben-Akiva & Swait test: dummy *vs* linear

- Compare model with dummy coding and model using linear service quality

```
> apollo_basTest("MNL_modeChoice_SP_service_linear",model)
                                              LL0        LL par adj.rho2
MNL_modeChoice_SP_service_linear  -8196.02 -5761.74   7   0.2962
MNL_modeChoice_SP_service_dummy   -8196.02 -5615.39   8   0.3139
Difference                            0.00   146.35   1   0.0177

p-value for Ben-Akiva & Swait test: 1.404e-65
```

- Can reject H_0 that the models are equivalent

Mode specific time coefficients

Mode specific time coefficients

Your next task

- We already account for differences in VTT between in vehicle time and access time
- But a minute spent on a bus is not the same as a minute spent on a train
- Two levels of difficulty:
  - level 1 use mode specific coefficients for in vehicle time
  - level 2 model fit comparisons against base model, and statistical tests for differences in time sensitivities across modes
- You can base these on the model with dummy coded service quality, i.e. `MNL_modeChoice_SP_service_dummy.r`

Mode specific time coefficients

Output with mode specific time coefficients

- Solution in `MNL_modeChoice_SP_mode_specific_time.r`
- Negative time sensitivities for all four modes

```
LL(final)                                          : -5598.9

Estimates:
                Estimate          s.e.   t.rat.(0)     Rob.s.e. Rob.t.rat.(0)
asc_car         0.000000            NA          NA           NA           NA
asc_bus         0.062411      0.538550      0.1159     0.533037       0.1171
asc_air         0.238277      0.340124      0.7006     0.329272       0.7236
asc_rail       -1.481370      0.327325     -4.5257     0.309844      -4.7810
b_tt_car       -0.011602 6.5322e-04     -17.7614 6.7831e-04     -17.1044
b_tt_bus       -0.017368      0.001452     -11.9630     0.001464     -11.8612
b_tt_air       -0.019483      0.002587      -7.5300     0.002475      -7.8717
b_tt_rail      -0.006365      0.001704      -3.7350     0.001623      -3.9207
b_access       -0.023193      0.002689      -8.6264     0.002645      -8.7676
b_cost         -0.058756      0.001487     -39.5176     0.001660     -35.3946
b_no_frills     0.000000            NA          NA           NA           NA
b_wifi          0.937555      0.052981     17.6961     0.055184     16.9898
b_food          0.409560      0.052180      7.8489     0.052628      7.7822
```

Mode specific time coefficients

Model fit comparison

- Smaller improvement over model with generic time coefficients
- But large enough to reject H_0 of no differences

```
> apollo_lrTest("MNL_modeChoice_SP_service_dummy",model)
                                                 LL par
MNL_modeChoice_SP_service_dummy          -5615.39   8
MNL_modeChoice_SP_mode_specific_time -5598.90  11
Difference                                 16.49   3

Likelihood ratio test-value:     32.98
Degrees of freedom:              3
Likelihood ratio test p-value:   3.252e-07
```

Mode specific time coefficients

WTP implications

- Big differences visible across modes
- Can also look at t-ratio for those differences

```
> apollo_deltaMethod(model,
+                    deltaMethod_settings = list(
+                      expression=c(VTT_car="60*b_tt_car/b_cost",
+                                   VTT_bus="60*b_tt_bus/b_cost",
+                                   VTT_air="60*b_tt_air/b_cost",
+                                   VTT_rail="60*b_tt_rail/b_cost",
+                                   TT_car_vs_rail="b_tt_car-b_tt_rail",
+                                   VTT_car_vs_rail="60*((b_tt_car-b_tt_rail)/b_cost)"
+                      )))
Running Delta method computation for user-defined function:

      Expression    Value Robust s.e. Rob t-ratio (0)
         VTT_car 11.8479       0.6641           17.84
         VTT_bus 17.7348       1.5188           11.68
         VTT_air 19.8961       2.4955            7.97
        VTT_rail  6.4997       1.6373            3.97
  TT_car_vs_rail -0.0052       0.0017           -3.04
 VTT_car_vs_rail  5.3482       1.7717            3.02
```

WTP space

WTP space

Your next task

- Take `MNL_modeChoice_SP_mode_specific_time.r` and write this model in WTP space (keeping ASCs in preference space)
  - remember: moving to WTP space simply involves multiplying some of the non-cost components by the cost coefficient
  - think about how to code this efficiently

WTP space

Results for WTP space model

- Solution in MNL_modeChoice_SP_WTP_space.r

```
LL(final)                                          :  -5598.9

Estimates:
                  Estimate        s.e.   t.rat.(0)  Rob.s.e.  Rob.t.rat.(0)
asc_car            0.00000          NA          NA        NA             NA
asc_bus            0.06241    0.538550      0.1159  0.533037         0.1171
asc_air            0.23828    0.340124      0.7006  0.329272         0.7236
asc_rail          -1.48137    0.327325     -4.5257  0.309844        -4.7810
v_tt_car           0.19746    0.010970     18.0006  0.011068        17.8407
v_tt_bus           0.29560    0.024965     11.8408  0.025314        11.6774
v_tt_air           0.33160    0.042851      7.7383  0.041592         7.9727
v_tt_rail          0.10833    0.028564      3.7924  0.027288         3.9697
v_access           0.39473    0.045270      8.7195  0.044973         8.7770
b_cost            -0.05876    0.001487    -39.5176  0.001660       -35.3946
v_no_frills        0.00000          NA          NA        NA             NA
v_wifi           -15.95678    0.897545    -17.7782  0.989345       -16.1286
v_food            -6.97052    0.881668     -7.9061  0.889295        -7.8383
```


Questions?
Apollo
www.ApolloChoiceModelling.com
The most flexible choice modelling software (up to a probability)