

# Apollo practical 3

Stéphane Hess

[stephane.hess@gmail.com](mailto:stephane.hess@gmail.com)

# Making predictions: Apollo implementation

# Making predictions: Apollo implementation

## Example files

- We reuse the two models we worked on yesterday :
  - `MNL_modeChoice_RP_base_model.r`
  - `MNL_modeChoice_SP_base_model.r`

# Making predictions: Apollo implementation

## Generating predictions in *Apollo*

- ❑ `apollo_prediction` returns probabilities for all alternatives in the model (plus chosen) with given values for parameters
- ❑ We do this at the estimates but could also look at changes in parameters

```
> predictions_base = apollo_prediction(model, apollo_probabilities, apollo_inputs)
Running predictions from model using parameter estimates...
Prediction at model estimates
```

	car	bus	air	rail
Aggregate	332.00	126.00	215.00	327.00
Average	0.33	0.13	0.22	0.33

```
> ### keep only columns for four alternatives (removing ID, choice task and chosen)
> predictions_base=predictions_base[,3:6]
> summary(predictions_base)
```

car		bus		air		rail	
Min.	:0.0000	Min.	:0.0000	Min.	:0.00000	Min.	:0.0000
1st Qu.:	0.2377	1st Qu.:	0.0798	1st Qu.:	0.07715	1st Qu.:	0.2202
Median	:0.3536	Median	:0.1108	Median	:0.18674	Median	:0.3096
Mean	:0.3320	Mean	:0.1260	Mean	:0.21500	Mean	:0.3270
3rd Qu.:	0.4748	3rd Qu.:	0.1523	3rd Qu.:	0.33406	3rd Qu.:	0.4578
Max.	:0.8804	Max.	:0.6056	Max.	:0.87700	Max.	:0.8709

# Making predictions: Apollo implementation

## After a 1% increase in cost for rail

```
> database$cost_rail=1.01*database$cost_rail
>
> ### Revalidate data
> apollo_inputs=apollo_validateInputs()
...
>
> ### Rerun predictions with the new data, and save into a separate matrix
> predictions_new = apollo_prediction(model,apollo_probabilities,apollo_inputs)
Running predictions from model using parameter estimates...
Prediction at model estimates
```

	car	bus	air	rail
Aggregate	333.52	126.69	215.99	323.80
Average	0.33	0.13	0.22	0.32

```
...
> summary(predictions_new)
```

car		bus		air		rail	
Min.	:0.0000	Min.	:0.00000	Min.	:0.00000	Min.	:0.0000
1st Qu.	:0.2387	1st Qu.	:0.08023	1st Qu.	:0.07765	1st Qu.	:0.2161
Median	:0.3557	Median	:0.11135	Median	:0.18806	Median	:0.3059
Mean	:0.3335	Mean	:0.12669	Mean	:0.21599	Mean	:0.3238
3rd Qu.	:0.4771	3rd Qu.	:0.15275	3rd Qu.	:0.33560	3rd Qu.	:0.4540
Max.	:0.8804	Max.	:0.60559	Max.	:0.87701	Max.	:0.8696

# Making predictions: Apollo implementation

## Demand before and after

- Can also see that MNL with full set of ASC recovers market shares

```
> predictions_overview=rbind(table(database$choice)/nrow(database),  
+                             colMeans(predictions_base),  
+                             colMeans(predictions_new))  
>  
> rownames(predictions_overview)=c("Data",  
+                                  "Base predictions",  
+                                  "1% increase in car costs")  
> print(round(predictions_overview,4))
```

	1	2	3	4
Data	0.3320	0.1260	0.215	0.3270
Base predictions	0.3320	0.1260	0.215	0.3270
1% increase in car costs	0.3335	0.1267	0.216	0.3238

# Making predictions: Apollo implementation

## Calculating elasticities

- We can approximate the elasticity by making a prediction with a very small increase in cost
- Then we use  $\frac{\log\left(\frac{\text{forecast demand}}{\text{base demand}}\right)}{\log(1.01)}$

```
> ### Computing elasticities
> ### Own elasticity for rail:
> log(sum(predictions_new[,4])/sum(predictions_base[,4]))/log(1.01)
[1] -0.9883048
> ### Cross-elasticities for other modes
> log(sum(predictions_new[,1])/sum(predictions_base[,1]))/log(1.01)
[1] 0.459145
> log(sum(predictions_new[,2])/sum(predictions_base[,2]))/log(1.01)
[1] 0.5493301
> log(sum(predictions_new[,3])/sum(predictions_base[,3]))/log(1.01)
[1] 0.4612738
```

# Making predictions: Apollo implementation

## SP elasticities higher than RP

```
> print(round(predictions_overview,4))
              1          2          3          4
Data          0.2780 0.0511 0.2174 0.4534
Base predictions 0.2780 0.0511 0.2174 0.4534
1% increase in rail costs 0.2808 0.0517 0.2193 0.4482
>
> ### Computing elasticities
> ### Own elasticity for rail:
> log(sum(predictions_new[,4])/sum(predictions_base[,4]))/log(1.01)
[1] -1.169577
> ### Cross-elasticities for other modes
> log(sum(predictions_new[,1])/sum(predictions_base[,1]))/log(1.01)
[1] 0.9929864
> log(sum(predictions_new[,2])/sum(predictions_base[,2]))/log(1.01)
[1] 1.1788
> log(sum(predictions_new[,3])/sum(predictions_base[,3]))/log(1.01)
[1] 0.8663578
```





# Questions?



**[www.ApolloChoiceModelling.com](http://www.ApolloChoiceModelling.com)**

The most flexible choice modelling software (up to a probability)