Apollo practical 4

Stephane Hess
stephane.hess@gmail.com

Outline

First MMNL model

First MMNL model

Data in package: apollo_swissRouteChoiceData

- Public transport stated choice survey from Switzerland
- Two unlabelled alternatives, 9 choices per person
- Alternatives described by travel time (tt), travel cost (tc), headway (hw) and changes (ch)
- Some basic covariates (household income, car availability, season ticket ownership and journey purpose)

```
ID,choice,tt1,tc1,hw1,ch1,tt2,tc2,hw2,ch2,hh_inc_abs,car_availability,
    ↪half_discount_ticket,full_discount_ticket,commute,shopping,business,leisure
2439,2,58,7,30,1,50,8,30,0,50000,1,0,0,1,0,0,0
2439,1,30,8,60,0,41,7,15,2,50000,1,0,0,1,0,0,0
2439,1,41,7,30,0,34,8,15,2,50000,1,0,0,1,0,0,0
2439,1,44,10,60,1,52,9,60,2,50000,1,0,0,1,0,0,0
2439,2,43,9,60,0,34,10,30,0,50000,1,0,0,1,0,0,0
2439,2,36,8,60,1,43,7,15,1,50000,1,0,0,1,0,0,0
...
```

First MMNL model

MNL model: specification

- Model file MNL_swiss.r

```
### List of utilities: these must use the same names as in mnl_settings, order is
    ↪irrelevant
V = list()
V[["alt1"]] = asc1 + b_tt * tt1 + b_tc * tc1 + b_hw * hw1 + b_ch * ch1
V[["alt2"]] =        b_tt * tt2 + b_tc * tc2 + b_hw * hw2 + b_ch * ch2

### Define settings for MNL model component
mnl_settings = list(
  alternatives  = c(alt1=1, alt2=2),
  avail         = list(alt1=1, alt2=1),
  choiceVar     = choice,
  utilities     = V
)

### Compute probabilities using MNL model
P[["model"]] = apollo_mnl(mnl_settings, functionality)
```

First MMNL model

MNL model: results

```
LL(final)                        : -1665.62

Estimates:
         Estimate        s.e.   t.rat.(0)    Rob.s.e.  Rob.t.rat.(0)
asc1     -0.01583    0.042869    -0.3692    0.045657       -0.3467
b_tt     -0.05973    0.004257   -14.0321    0.006741       -8.8602
b_tc     -0.13164    0.013503    -9.7488    0.023631       -5.5705
b_hw     -0.03744    0.001847   -20.2668    0.002317      -16.1590
b_ch     -1.15213    0.043419   -26.5350    0.061363      -18.7754

> apollo_deltaMethod(model,
+                    deltaMethod_settings = list(
+                      expression=c(VTT="60*b_tt/b_tc"
+                      )))
Running Delta method computation for user-defined function:

 Expression    Value Robust s.e. Rob t-ratio (0)
        VTT  27.2243      3.3403            8.15
```

First MMNL model

Moving to MMNL specification

- Model file MMNL_swiss_CS_uniform.r
- We reuse the simple Swiss data and use Uniform coefficients for time and cost
- For now, we use a cross-sectional (CS) specification, a point we return to later on
- We use 3 cores for improved speed

```
apollo_control = list(
  modelName       = "MMNL_swiss_CS_uniform",
  modelDescr      = "MMNL model with Uniform distributions for time and
       ↪cost on Swiss route choice data, cross-sectional spec",
  indivID         = "ID",
  nCores          = 3,
  outputDirectory = "output"
)
```

First MMNL model

Define parameters

- We define new parameters as we will now estimate offsets and ranges

```
apollo_beta = c(asc1        = 0,
                b_tt_a      = 0,
                b_tt_b      = 0,
                b_tc_a      = 0,
                b_tc_b      = 0,
                b_hw        = 0,
                b_ch        = 0)
```

First MMNL model

Defining draws

- Define names, number and type of draws, as well as dimension of integration (inter or intra)
- Determine which need to be translated to Normals (also for Lognormal), by e.g. using `interNormDraws` and `interUnifDraws`
- Empty entries can be omitted

```
### Set parameters for generating draws
apollo_draws = list(
  interDrawsType = "",
  interNDraws    = 0,
  interNormDraws = c(),
  interUnifDraws = c(),

  intraDrawsType = "halton",
  intraNDraws    = 100,
  intraNormDraws = c(),
  intraUnifDraws = c("draws_tt","draws_tc")
)
```

First MMNL model

Defining random components

- So far, we have created draws, and the parameters that will describe the distribution of the random components in our model
- Now we create the random components themselves
- This is a function that will be called at each iteration during estimation, updating the random components on the basis of the parameter values from that iteration
- With a Uniform distribution, we simply have that $\beta = a + b \cdot \xi$, where $\xi \sim U[0, 1]$

```
### Create random parameters
apollo_randCoeff = function(apollo_beta, apollo_inputs){
  randcoeff = list()

  randcoeff[["b_tt"]] = b_tt_a + b_tt_b * draws_tt
  randcoeff[["b_tc"]] = b_tc_a + b_tc_b * draws_tc

  return(randcoeff)
}
```

First MMNL model

Use inside apollo_probabilities

- We can now use the same code as before for the utilities
- The difference is that `b_tt` and `b_tc` now come from `apollo_randcoeff`

```
V[["alt1"]] = asc1 + b_tt * tt1 + b_tc * tc1 + b_hw * hw1 + b_ch * ch1
V[["alt2"]] =        b_tt * tt2 + b_tc * tc2 + b_hw * hw2 + b_ch * ch2
```

First MMNL model

Averaging across draws before multiplying

- Likelihood function means we need to average across draws before taking the product across choices

$$SL(\Omega) = \prod_{n=1}^{N} \prod_{t=1}^{T_n} \sum_{r=1}^{R} \frac{1}{R} P_{n,j_{n,t}}\left(\beta_n^{(r)}\right)$$

- Step 1: we average across intra-individual draws, ending up with a column vector, with one row per choice task

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

- We then take the product across choices for the same person, reducing the number of rows in our vector to one per person

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

First MMNL model

Outputs: Cross-sectional MMNL with two Uniforms

```
LL(final)                        : -1608.076

Estimates:
           Estimate        s.e.    t.rat.(0)    Rob.s.e.  Rob.t.rat.(0)
asc1      -0.005185    0.057043     -0.09090    0.059813       -0.08669
b_tt_a     0.011939    0.017238      0.69262    0.018540        0.64397
b_tt_b    -0.285197    0.045799     -6.22711    0.051620       -5.52491
b_tc_a     0.145333    0.061465      2.36450    0.075539        1.92394
b_tc_b    -1.038612    0.178269     -5.82611    0.274768       -3.77995
b_hw      -0.052523    0.003281    -16.00797    0.004220      -12.44720
b_ch      -1.581800    0.083028    -19.05147    0.114735      -13.78654
```

```
> apollo_lrTest("MNL_swiss", model)
                           LL  par
MNL_swiss            -1665.62    5
MMNL_swiss_CS_uniform -1608.08    7
Difference              57.54    2

Likelihood ratio test-value:    115.08
Degrees of freedom:             2
Likelihood ratio test p-value:  1.025e-25
```

Changing distributions: part 1

Changing distributions: part 1

Transformation of draws

- We typically do not want to just use Uniform distributions
- Draws generated using PMC or QMC approaches are typically uniform draws between 0 and 1, say r_u
- Can transform into a draw from a standard normal distribution, i.e. $N(0,1)$, using the inverse CDF, such that $\Phi\left(r_n\right) = r_u$, where Φ is the standard normal CDF
- To obtain a $N\left(\mu, \sigma\right)$ draw, we use $\beta_{N(\mu,\sigma)} = \mu + \sigma r_n$
  - this explains why the sign of σ is irrelevant in estimation (but matters in multivariate distributions)
- For Lognormal, use $\beta_{LN\left(\mu_{(\log\beta)},\sigma_{(\log\beta)}\right)} = exp\left(\mu_{(\log\beta)} + \sigma_{(\log\beta)} r_n\right)$, giving

  $\mu_{\beta_{LN}} = e^{\mu_{(\log\beta)} + \frac{\sigma^2_{(\log\beta)}}{2}}$ and $\sigma_{\beta_{LN}} = \mu_{\beta_{LN}}\sqrt{e^{\sigma^2_{(\log\beta)}} - 1}$
- For a triangular draw, we sum two independent uniform draws

Changing distributions: part 1

Producing draws in R: draws.r

```
> draws=runif(10000)

> draws_N_0_1=qnorm(draws)
> mean(draws_N_0_1)
[1] -0.002410429
> sd(draws_N_0_1)
[1] 1.003914

> draws_N_neg2_1=-2+1*draws_N_0_1
> mean(draws_N_neg2_1)
[1] -2.00241
> sd(draws_N_neg2_1)
[1] 1.003914

> LNdraws=exp(draws_N_neg2_1)
> mean(LNdraws)
[1] 0.2224649
> sd(LNdraws)
[1] 0.2770684

> exp(-2+(1^2)/2)
[1] 0.2231302
> exp(-2+(1^2)/2)*sqrt(exp(1^2)-1)
[1] 0.2924863
```

Changing distributions: part 1

Replacing Uniform with Normal distribution

- Model file MMNL_swiss_CS_normal.r
- We reuse the simple Swiss data and use Normal coefficients for time and cost
- New parameter names

```
apollo_beta = c(asc1        = 0,
                b_tt_mu     = 0,
                b_tt_sig    = 0,
                b_tc_mu     = 0,
                b_tc_sig    = 0,
                b_hw        = 0,
                b_ch        = 0)
```

Changing distributions: part 1

Defining draws

- Normal instead of Uniform draws

```
### Set parameters for generating draws
apollo_draws = list(
  intraDrawsType = "halton",
  intraNDraws    = 100,
  intraNormDraws = c("draws_tt","draws_tc")
)
```

- With a Normal distribution, we simply have that $\beta = \mu + \sigma \cdot \xi$, where $\xi \sim N([0,1)$

```
### Create random parameters
apollo_randCoeff = function(apollo_beta, apollo_inputs){
  randcoeff = list()

  randcoeff[["b_tt"]] = b_tt_mu + b_tt_sig * draws_tt
  randcoeff[["b_tc"]] = b_tc_mu + b_tc_sig * draws_tc

  return(randcoeff)
}
```

Changing distributions: part 1

Outputs: Cross-sectional MMNL with two Normals

- Significant heterogeneity for both random coefficients
- Sign of σ is irrelevant
- Fit essentially the same as with Uniform

```
LL(final)                        : -1608.448

Estimates:
             Estimate        s.e.   t.rat.(0)    Rob.s.e.  Rob.t.rat.(0)
asc1        -0.005955    0.057534     -0.1035    0.060285       -0.09879
b_tt_mu     -0.128574    0.013694     -9.3889    0.019448       -6.61113
b_tt_sig    -0.082387    0.015908     -5.1791    0.017234       -4.78059
b_tc_mu     -0.367095    0.051642     -7.1085    0.086749       -4.23167
b_tc_sig    -0.335441    0.068490     -4.8977    0.111342       -3.01272
b_hw        -0.053162    0.003437    -15.4678    0.004516      -11.77105
b_ch        -1.600115    0.087674    -18.2507    0.123450      -12.96160
```

- But does the use of a Normal distribution really make sense?

Panel specification

Panel specification

MMNL with panel specification

- Model file MMNL_swiss_panel_normal.r
- We reuse the simple Swiss data and use Normal coefficients for time and cost, with heterogeneity at the level of an individual rather than an observation

```
### Set parameters for generating draws
apollo_draws = list(
  interDrawsType = "halton",
  interNDraws    = 100,
  interNormDraws = c("draws_tt","draws_tc")
)
```

Panel specification

Multiplying and averaging

- Likelihood function means take product across choices before averaging across

$$SL\left(\Omega\right)=\prod_{n=1}^{N}\sum_{r=1}^{R}\frac{1}{R}\prod_{t=1}^{T_n}P_{j^*_{nt}}\left(\beta^{(r)}\right)$$

- This reduces the number of rows in our matrix to one per individual

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

- We finally average across inter-individual draws, ending up with a column vector

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

Panel specification

Outputs: Panel MMNL with two Normals

```
Estimates:
              Estimate        s.e.    t.rat.(0)    Rob.s.e.  Rob.t.rat.(0)
asc1          -0.02063    0.050366      -0.4096    0.053547        -0.3852
b_tt_mu       -0.10281    0.007823     -13.1414    0.009697       -10.6019
b_tt_sig       0.04394    0.007920       5.5477    0.007070         6.2146
b_tc_mu       -0.33391    0.033748      -9.8943    0.043980        -7.5923
b_tc_sig       0.32009    0.037329       8.5748    0.053307         6.0046
b_hw          -0.04778    0.002371     -20.1505    0.003057       -15.6321
b_ch          -1.43312    0.056582     -25.3281    0.081931       -17.4919
```

- Bigger improvement than with cross-sectional MMNL

```
> apollo_lrTest("MNL_swiss", model)
                                  LL  par
MNL_swiss                   -1665.62    5
MMNL_swiss_panel_normal     -1544.87    7
Difference                    120.75    2

Likelihood ratio test-value:     241.5
Degrees of freedom:              2
Likelihood ratio test p-value:   3.622e-53
```

Changing distributions: part 2

Changing distributions

- Uniform has a flat profile
- Normal is unbounded
- Many other choices of distributions

Changing distributions

- Our base model `MMNL_swiss_panel_normal.r` uses two random coefficients (time and cost), both with Normals
- Four levels of difficulty:

  level 1 make all four coefficients random, using Normal distributions

  level 2 make all four coefficients random, using Uniform distributions

  level 3 make all four coefficients random, using negative Lognormal distributions
  - $\beta = -e^{(\mu+\sigma r_N)}$, where $r_N \sim N(0,1)$
  - use something like -3 as the starting value for μ

  level 4 make all four coefficients random, using symmetric Triangular
  - Symmetrical Triangular: $\beta = a + b\left(r_{U,1} + r_{U,2}\right)$ where $r_{U,1}$ and $r_{U,2}$ are independent $U(0,1)$ variates
    - do not use Haltons...

Changing distributions: part 2

MMNL_swiss_panel_all_normal.r

- All four standard deviations different from zero, and big improvement over model with two random coeffs

```
LL(final)                              : -1469.931
             Estimate         s.e.   t.rat.(0)    Rob.s.e. Rob.t.rat.(0)
asc1         -0.04357     0.061023     -0.7140    0.066882      -0.6515
b_tt_mu      -0.13795     0.010767    -12.8127    0.014589      -9.4562
b_tt_sig      0.05866     0.007693      7.6257    0.008101       7.2413
b_tc_mu      -0.46673     0.044004    -10.6065    0.061826      -7.5491
b_tc_sig      0.39151     0.040603      9.6422    0.057013       6.8670
b_hw_mu      -0.06318     0.004597    -13.7449    0.005622     -11.2372
b_hw_sig      0.03743     0.005177      7.2297    0.006416       5.8334
b_ch_mu      -2.02443     0.123127    -16.4418    0.145869     -13.8784
b_ch_sig     -1.22013     0.122582     -9.9536    0.133585      -9.1338

> apollo_lrTest("MMNL_swiss_panel_normal", model)
                                   LL  par
MMNL_swiss_panel_normal      -1544.87    7
MMNL_swiss_panel_all_normal  -1469.93    9
Difference                      74.94    2
Likelihood ratio test-value:    149.88
Degrees of freedom:            2
Likelihood ratio test p-value:  2.844e-33
```

Changing distributions: part 2

MMNL_swiss_panel_all_uniform.r

- Slightly better fit than with Normals, but sign violations remain except for travel time

```
LL(final)                        :  -1461.697
           Estimate        s.e.   t.rat.(0)   Rob.s.e.  Rob.t.rat.(0)
asc1      -0.037069    0.062450     -0.5936   0.069926        -0.5301
b_tt_a    -0.040172    0.014145     -2.8400   0.015330        -2.6205
b_tt_b    -0.223788    0.031767     -7.0446   0.034818        -6.4274
b_tc_a     0.217221    0.043895      4.9487   0.050062         4.3390
b_tc_b    -1.471131    0.161619     -9.1025   0.240877        -6.1074
b_hw_a     0.005663    0.006856      0.8259   0.007467         0.7584
b_hw_b    -0.142276    0.018536     -7.6757   0.020844        -6.8258
b_ch_a     0.061118    0.140104      0.4362   0.140804         0.4341
b_ch_b    -4.562336    0.434400    -10.5026   0.477411        -9.5564

> apollo_basTest("MMNL_swiss_panel_all_normal", model)
                                LL0        LL  par  adj.rho2
MMNL_swiss_panel_all_normal  -2420.47  -1469.93    9    0.3890
MMNL_swiss_panel_all_uniform -2420.47  -1461.70    9    0.3924
Difference                       0.00      8.23    0    0.0034

p-value for Ben-Akiva & Swait test: 2.485e-05
```

Changing distributions: part 2

MMNL_swiss_panel_all_negLN.r

```
LL(final)                    :  -1444.249
Estimates:
                  Estimate        s.e.   t.rat.(0)    Rob.s.e.  Rob.t.rat.(0)
asc1              -0.03748     0.06223     -0.6023     0.06974        -0.5375
b_log_tt_mu       -2.01215     0.08609    -23.3736     0.11181       -17.9961
b_log_tt_sig       0.50657     0.09900      5.1168     0.14219         3.5626
b_log_tc_mu       -1.12714     0.12416     -9.0785     0.15019        -7.5047
b_log_tc_sig       0.97176     0.08724     11.1389     0.10996         8.8372
b_log_hw_mu       -2.94766     0.07744    -38.0641     0.08584       -34.3400
b_log_hw_sig       0.70195     0.07221      9.7206     0.06619        10.6043
b_log_ch_mu        0.65622     0.07586      8.6501     0.08555         7.6704
b_log_ch_sig       0.94054     0.09056     10.3859     0.09495         9.9056

> apollo_basTest("MMNL_swiss_panel_all_uniform", model)
                                  LL0       LL par adj.rho2
MMNL_swiss_panel_all_uniform -2420.47 -1461.70   9   0.3924
MMNL_swiss_panel_all_negLN   -2420.47 -1444.25   9   0.3996
Difference                       0.00    17.45   0   0.0072

p-value for Ben-Akiva & Swait test: 1.776e-09
```

Changing distributions: part 2

MMNL_swiss_panel_all_triangular.r

LL(final) : −1468.06

	Estimate	s.e.	t.rat.(0)	Rob.s.e.	Rob.t.rat.(0)
asc1	−0.056869	0.06158	−0.9235	0.06780	−0.8388
b_tt_a	0.009458	0.01596	0.5924	0.01513	0.6250
b_tt_b	−0.151161	0.01908	−7.9244	0.02216	−6.8208
b_tc_a	0.540210	0.07427	7.2733	0.08530	6.3327
b_tc_b	−0.996480	0.09461	−10.5320	0.12515	−7.9625
b_hw_a	0.029789	0.01063	2.8012	0.01228	2.4248
b_hw_b	−0.091225	0.01229	−7.4249	0.01415	−6.4450
b_ch_a	0.952730	0.23511	4.0522	0.24726	3.8532
b_ch_b	−3.085073	0.29670	−10.3979	0.33537	−9.1990

Changing distributions: part 2

Analysing outputs from lognormal

- Lognormal has a long tail, but has some desirable properties
- Inverse of lognormal exists (and is a lognormal)
- Ratio of two lognormals is a lognormal (so preference space and WTP space are the same)
- Estimated parameters relate to $log(\beta)$, i.e. we get $\mu_{log(\beta)}$ and $\sigma_{log(\beta)}$
- Can calculate actual moments quite easily:

$$\mu_\beta = exp(\mu_{log(\beta)} + \frac{\sigma^2_{log(\beta)}}{2})$$

and

$$\sigma_\beta = \mu_\beta * \sqrt{exp(\sigma^2_{log(\beta)}) - 1}$$

Changing distributions: part 2

In *Apollo*: MMNL_swiss_panel_all_negLN.r

```
> ### analytical
> mean_tt=-exp(model$estimate["b_log_tt_mu"]+
    ↪model$estimate["b_log_tt_sig"]^2/2)
> sd_tt=abs(mean_tt*sqrt(exp(model$estimate["
    ↪b_log_tt_sig"]^2)-1))
> mean_tc=-exp(model$estimate["b_log_tc_mu"]+
    ↪model$estimate["b_log_tc_sig"]^2/2)
> sd_tc=abs(mean_tc*sqrt(exp(model$estimate["
    ↪b_log_tc_sig"]^2)-1))
> log_vtt_mu=model$estimate["b_log_tt_mu"]-
    ↪model$estimate["b_log_tc_mu"]
> log_vtt_sig=sqrt(model$estimate["b_log_tt_sig
    ↪"]^2+model$estimate["b_log_tc_sig"]^2)
> mean_vtt=exp(log_vtt_mu+log_vtt_sig^2/2)
> sd_vtt=abs(mean_vtt*sqrt(exp(log_vtt_sig^2)
    ↪-1))
> mean_vtt=60*mean_vtt
> sd_vtt=60*sd_vtt

> ### simulated
> beta=apollo_unconditionals(model,
    ↪apollo_probabilities,apollo_inputs)
 Updating inputs...Done.
 Unconditional distributions computed
```

```
> output=matrix(0,nrow=6,ncol=2)
> colnames(output)=c("Analytical","Simulated")
> rownames(output)=c("tt_mu","tt_sig","tc_mu","tc_sig","
    ↪vtt_mu","vtt_sig")
> output[1,1]=mean_tt
> output[2,1]=sd_tt
> output[3,1]=mean_tc
> output[4,1]=sd_tc
> output[5,1]=mean_vtt
> output[6,1]=sd_vtt
> output[1,2]=mean(beta[["b_tt"]])
> output[2,2]=sd(beta[["b_tt"]])
> output[3,2]=mean(beta[["b_tc"]])
> output[4,2]=sd(beta[["b_tc"]])
> output[5,2]=mean(beta[["b_tt"]]/beta[["b_tc"]])*60
> output[6,2]=sd(beta[["b_tt"]]/beta[["b_tc"]])*60
> round(output,2)
        Analytical Simulated
tt_mu        -0.15     -0.15
tt_sig        0.08      0.08
tc_mu        -0.52     -0.52
tc_sig        0.65      0.64
vtt_mu       45.14     45.14
vtt_sig      68.80     68.01
```

Apollo

Changing distributions: part 2

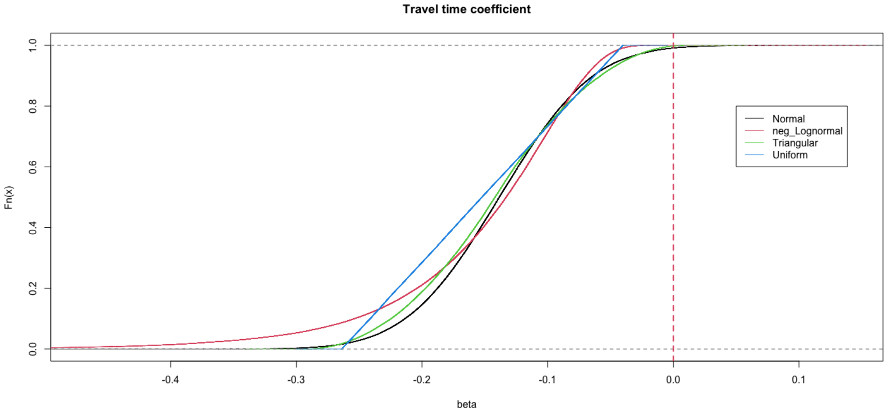
Comparing distributions

- R is useful for comparing results (`code_for_plots.r`)

```
rN1=rnorm(100000)
rN2=rnorm(100000)
rN3=rnorm(100000)
rN4=rnorm(100000)

estimates=read.csv("output/MMNL_swiss_panel_all_normal_estimates.csv",row.names=1)

beta=estimates[,1]
names(beta)=row.names(estimates)
tt_Normal=(beta["b_tt_mu"]+beta["b_tt_sig"]*rN1)
tc_Normal=(beta["b_tc_mu"]+beta["b_tc_sig"]*rN2)
hw_Normal=(beta["b_hw_mu"]+beta["b_hw_sig"]*rN3)
ch_Normal=(beta["b_ch_mu"]+beta["b_ch_sig"]*rN4)

plot(ecdf(tt_Normal),col=1,main="Travel time coefficient",xlab="beta",ylim=c(0,1))
lines(ecdf(tt_neg_Lognormal),col=2)
lines(ecdf(tt_Triangular),col=3)
lines(ecdf(tt_Uniform),col=4)
abline(v=0,lty=2,col=2,lwd=2)
legend(0.05,0.8,c("Normal","neg_Lognormal","Triangular","Uniform"),col=c(1,2,3,4),lty=1,cex=1)
```

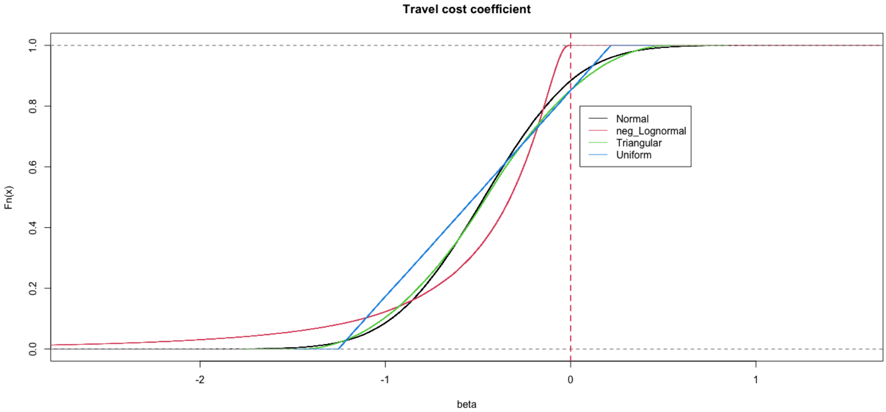
Changing distributions: part 2

Distribution of travel time coefficient



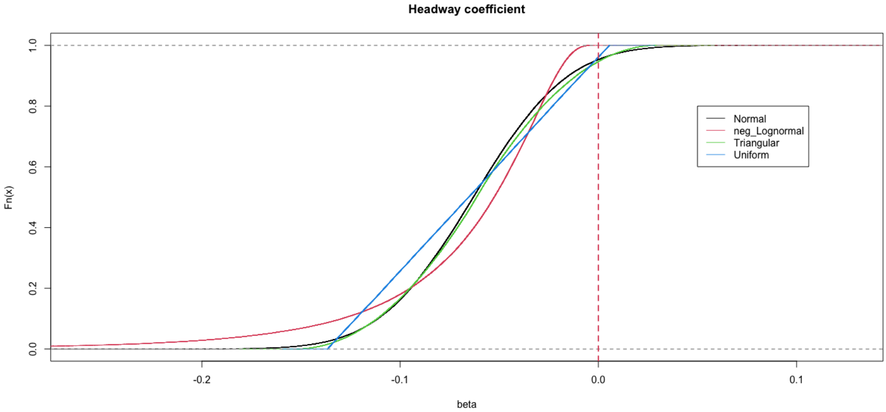
Changing distributions: part 2

Distribution of travel cost coefficient



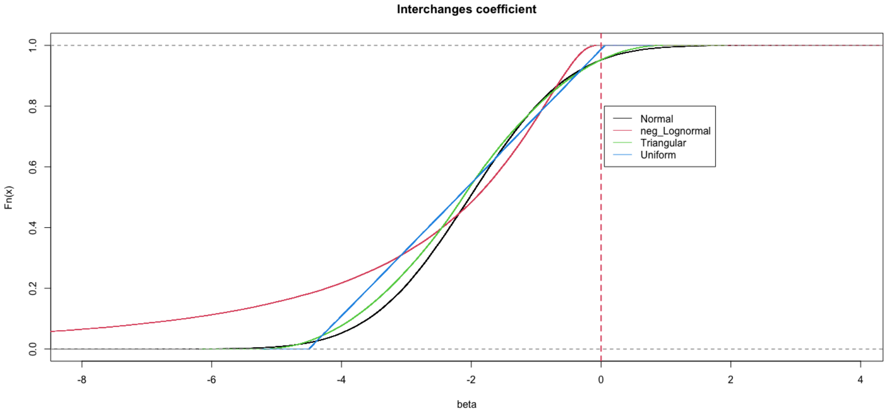
Changing distributions: part 2

Distribution of headway coefficient



Changing distributions: part 2

Distribution of interchanges coefficient



WTP space

WTP space

A task for you

- Take the base model `MMNL_swiss_panel_normal.r` and turn it into WTP space
- Of course behavioural silly to have Normals but at least mathematically no problems for WTP

WTP space

WTP space results: MMNL_swiss_panel_normal_WTP_space.r

- Fit is very similar to MMNL_swiss_panel_normal.r and mean VTT is very similar to MNL

```
LL(final)                          : -1543.75

Estimates:
             Estimate        s.e.   t.rat.(0)    Rob.s.e. Rob.t.rat.(0)
asc1         -0.01349     0.05125     -0.2631     0.05573       -0.2420
v_tt_mu       0.42499     0.03536     12.0201     0.04702        9.0390
v_tt_sig     -0.33562     0.03829     -8.7651     0.04598       -7.2990
b_tc_mu      -0.25857     0.02793     -9.2575     0.04110       -6.2914
b_tc_sig     -0.11875     0.01745     -6.8037     0.02142       -5.5447
v_hw          0.21257     0.02214      9.6031     0.03462        6.1402
v_ch          6.67859     0.65370     10.2166     0.99250        6.7291
```

Changing precision of simulation

Changing precision of simulation

Your task

- Any simulation is an approximation
- The more draws we use, the better, that's a fact
- Our model with four negative Lognormals (`MMNL_swiss_panel_all_negLN.r`) uses 100 Halton draws per individual and per random parameter
- Four tasks:
  - level 1 reverse the order of the primes used for Halton (by changing order in `interNormDraws`)
  - level 2 use 100 MLHS draws instead
  - level 3 use 500 draws instead

Changing precision of simulation

Changing the number and type of draws

- Differences between type of draws, number of draws, and even order of primes

Model name	MLHS_100	MLHS_500	Halton_100	Halton_500	rev_Halton_100	rev_Halton_500	Halton_5000
LL(final)	-1445.813	-1443.06	-1444.249	-1444.017	-1445.936	-1443.625	-1444.319
b_log_tt_mu	-1.9661	-2.0072	-2.0122	-1.9792	-2.0036	-2.0007	-1.99469
b_log_tt_sig	0.3886	0.5035	0.5066	-0.4543	-0.4268	0.4667	-0.47384
b_log_tc_mu	-0.9934	-1.0848	-1.1271	-1.0302	-1.0787	-1.062	-1.03507
b_log_tc_sig	-0.996	-1.0473	0.9718	1.0004	0.9528	1.0216	1.00306
b_log_hw_mu	-2.9433	-2.9434	-2.9477	-2.9376	-2.9593	-2.9268	-2.93514
b_log_hw_sig	0.9219	0.8306	0.7019	0.8324	0.8417	0.8239	0.8207
b_log_ch_mu	0.5919	0.6567	0.6562	0.6294	0.6096	0.6274	0.62675
b_log_ch_sig	-0.7762	-0.8486	0.9405	-0.8477	-0.8363	-0.8138	-0.83043

Changing precision of simulation

Changing the number and type of draws

- If we take Halton with 5,000 draws as the truth

Model name	MLHS_100	MLHS_500	Halton_100	Halton_500	rev_Halton_100	rev_Halton_500
LL(final)	-1.494	1.259	0.07	0.302	-1.617	0.694
b_log_tt_mu	-1.43%	0.63%	0.88%	-0.78%	0.45%	0.30%
b_log_tt_sig	-17.99%	6.26%	6.91%	-4.12%	-9.93%	-1.51%
b_log_tc_mu	-4.03%	4.80%	8.89%	-0.47%	4.22%	2.60%
b_log_tc_sig	-0.70%	4.41%	-3.12%	-0.27%	-5.01%	1.85%
b_log_hw_mu	0.28%	0.28%	0.43%	0.08%	0.82%	-0.28%
b_log_hw_sig	12.33%	1.21%	-14.48%	1.43%	2.56%	0.39%
b_log_ch_mu	-5.56%	4.78%	4.70%	0.42%	-2.74%	0.10%
b_log_ch_sig	-6.53%	2.19%	13.25%	2.08%	0.71%	-2.00%

Questions?

Apollo

www.ApolloChoiceModelling.com

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