Annotated MM-ML programs

To apply the MM-ML procedures, several estimation programs have to be defined and uploaded to STATA. The general estimation command for the full model (equation 8) with 7 free parameters is:

```
program define ML_choice_time_conf
version 1.0
args lnf p mu0T sigmaT distT  mu0C sigmaC distC
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1,`p')-
binomial(total_pred, $ML_y1-1,`p')) if index==1 & miss == 0
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1 & miss == 1
quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T'+abs(`distT')*
total_pred, `sigmaT' )) if index==2
quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C'+abs(`distC')*
total_pred, `sigmaC' )) if index==3
end
```

The variable *index* codes whether it is a choice (1), decision time (2), or confidence (3) data point; the variable *total_pred* codes the total number of observations per category (for choice data points), and the contrast weight (for decision time and confidence) (see Table 1); the variable *miss* is used for choice data points only and codes if random choices (i.e., missing values) are predicted for the respective type of items (1) or not (0) (as it is often the case for an EQW strategy, see Table 1).

After installing the program (e.g., by copying it into the command window and pressing Enter), the program *ML_choice_time_conf* can be used as part of any ML estimation as follows:

```
ml model lf ML_choice_time_conf (epsilon: dv=) (mu_Time:)
(sigma_Time:) (R_Time:) (mu_Conf:) (sigma_Conf:) (R_Conf:) if
Participant == 1 & strat ==1
ml init 0.5 500 100 100 10 5 5, copy
ml maximize
```

'ml model' defines the model which contains the parameters listed in parentheses; *dv* defines the variable containing data for choices, confidence and time. The 'if' sub-command

defines for which person and strategy the command should be executed. ‘ml init’ defines initial values for the parameters in the order listed in the model command, and helps for quicker convergence. ‘ml maximize’ starts the ML routine. Note that convergence can only be found if the observed error rate fulfills $0 < \varepsilon < 1$. For persons with a perfect strategy use one artificial error has to be added to the data before the analysis is conducted.

An example data file and the complete syntax files (i.e., do-files) for uploading the estimation programs and carrying out a comprehensive analysis can be downloaded as supplementary material to this article. The commented code is also listed here:

```
******** ML Estimation programs: Definitions
*** Just copy into execution window of STATA and press enter*******
** After their installation these programs can be used as separate
estimations commands *
* The code includes
* a) the total model (estimating strategies with predictions for
choices, time and confidence)
* b) the flat time prediction model(for strategies that predict no
differences in decision times between items)
* c) the flat confidence prediction model(for strategies that
predict no differences in confidence between items)
* d) the flat confidence and time prediction model(for strategies
that predict no differences in decision times and confidence between
items)
* e) the random choice model (predicting random choices and no
difference between time and confidence for items).
* Additionally three models are provided that estimate choices, time
and confidence separately. The first model implements the choice
based strategy classification method by Bröder & Schiffer (2003).


*** total model.

program define ML_choice_time_conf
version 1.0
args lnf p mu0T sigmaT distT  mu0C sigmaC distC
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1,`p')-
binomial(total_pred, $ML_y1-1,`p')) if index==1 & miss == 0
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1 & miss == 1

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T'+abs(`distT')*
total_pred, `sigmaT' )) if index==2

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C'+abs(`distC')*
total_pred, `sigmaC' )) if index==3
end
```

```
* Explanation: 'args' defines the parameters; the 4 'replace'
commands generate the log-likelihood function. The first two
'replace' commands use the binomial distribution to generate the
function for choices (the second of it implements random choices
with error = 0.5), the third 'replace' command uses the normal-
density function to general the likelihood function for time, the
last command does the same for confidence. The 'abs' command
produces the absolute value of the rescaling parameter and therefore
prevents negative predictions. The following programs are derived
from the total model by deleting parts of it.

** Flat time prediction

program define ML_choice_time_conf_flat_time
version 1.0
args lnf p mu0T sigmaT mu0C sigmaC distC
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1,`p')-
binomial(total_pred, $ML_y1-1,`p')) if index==1 & miss == 0
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1 & miss == 1

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T', `sigmaT' )) if
index==2

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C'+abs(`distC')*
total_pred, `sigmaC' )) if index==3
end

** Flat Confidence Prediction

program define ML_choice_time_conf_flat_conf
version 1.0
args lnf p mu0T sigmaT distT  mu0C sigmaC
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1,`p')-
binomial(total_pred, $ML_y1-1,`p')) if index==1 & miss == 0
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1 & miss == 1

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T'+abs(`distT')*
total_pred, `sigmaT' )) if index==2

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C', `sigmaC' )) if
index==3
end


** Flat time and confidence prediction

program define ML_choice_time_conf_flat_both
version 1.0
args lnf p mu0T sigmaT mu0C sigmaC
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1,`p')-
binomial(total_pred, $ML_y1-1,`p')) if index==1 & miss == 0
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1 & miss == 1
```

```
quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T', `sigmaT' )) if
index==2

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C', `sigmaC' )) if
index==3
end


** Random Choice Model

program define ML_choice_time_conf_random
version 1.0
args lnf mu0T sigmaT mu0C sigmaC

quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T', `sigmaT' )) if
index==2

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C', `sigmaC' )) if
index==3
end



****** Single estimations

**choices only

program define ML_choice
version 1.0
args lnf p
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1,`p')-
binomial(total_pred, $ML_y1-1,`p')) if index==1 & miss == 0
quietly replace `lnf'=ln(binomial(total_pred, $ML_y1, 0.5)-
binomial(total_pred, $ML_y1-1, 0.5)) if index==1 & miss == 1
end


**time only

program define ML_time
version 1.0
args lnf mu0T sigmaT distT

quietly replace `lnf'=ln(normalden( $ML_y1, `mu0T'+`distT'*
total_pred, `sigmaT' )) if index==2

end


**confidence only

program define ML_conf
version 1.0
args lnf mu0C sigmaC distC
```

```
quietly replace `lnf'=ln(normalden( $ML_y1, `mu0C'+`distC'*
total_pred, `sigmaC' )) if index==3
end

*********End Definition


*** Total estimation program for all Participants and all strategies
producing a participant x strategy matrix with BIC scores
* The example uses 10 participants and 5 strategies but can be
easily extended to any number of strategies and participants.


matrix strat_vec = [0,1,1,0,4]
*** determines which model is used per strategy: 0-full model, 1-
flat decision time prediction, 2-flat confidence prediction, 3-flat
both, 4-random choice model
* each value for a strategy is separated by ‘,’

matrix N_obs_cat = [18,18,18,18,18]
*** replace by number or observations representing independent
categories for each strategy considered


matrix result = J(10,5,0)
** defines the output matrix and sets all initial values to 0. The
first parameter represents the number of participants, the second
the number of strategies.


***** The following loops define the search for persons (u) and
strategies (v). Replace the numbers after ‘/’ by total number of
participants (u) and strategies (v). The subject ID in the data
(variable: Participant) has to be coded starting from 1 to the
number of persons.

forvalues u= 1/10 {

forvalues v = 1/5 {

disp “****************************************”

disp “**** Subject: “`u' “   **** Strategy: “`v'

     if strat_vec[1,`v'] ==0 {

          ml model lf ML_choice_time_conf (epsilon: dv=) (mu_Time:)
          (sigma_Time:) (R_Time:) (mu_Conf:) (sigma_Conf:)
          (R_Conf:) if Participant ==`u'& strat == `v'

          ** defines the model and the variable used. ‘dv=’ defines
          that all observations for choices, confidence and time
          are coded in the same variable dv.

          ml init 0.5 0  2500 1000 37  45 30, copy
```

```
      ** gives initial values for the 7 parameters (error rate,
      mean time, sd time, R time, mean confidence, sd
      confidence, R confidence; see also model above).
      Convergence is quicker with better intial values.

      ml maximize

      ** starts the ML estimation and generates the individual
      output for subject u strategy v.

      matrix coef = e(b)

      if coef[1,1] < .50 {
       scalar BIC = -2* e(ll)+ln(N_obs_cat[1, `v'])*7
      matrix result[`u', `v'] = BIC

      disp "Subj: " `u'  "  Strat: " `v' "   - BIC:" BIC
      }
      ** coef[1,1]contains the error rate epsilon which can be
      uses as a limit for classification, I use .50 as limit
      here.
      ** The BIC score is calculated correcting for 7
      parameters for the full model.

      if coef[1,1] >= .50 {
      disp "BIC: not calculated because error bigger than .50"
      disp "Subj:  " `u'  "   Strat: " `v' "   - BIC: not
      calculated because epsilon bigger or equal than .50"
      }
}

if strat_vec[1,`v'] ==1 {

      ml model lf ML_choice_time_conf_flat_time (epsilon: dv=)
      (mu_Time:) (sigma_Time:) (mu_Conf:) (sigma_Conf:)
      (R_Conf:) if Participant ==`u'& strat == `v'
      ml init 0.5 0  2500  37  45 30, copy
      ml maximize
      matrix coef = e(b)
      if coef[1,1] < .50 {
       scalar BIC = -2* e(ll)+ln(N_obs_cat[1, `v'])*6
      matrix result[`u', `v'] = BIC

      disp "Subj: " `u'  "  Strat: " `v' "   - BIC:" BIC
      }

      ** The BIC score is calculated correcting for 6
      parameters for the flat time prediction model.

      if coef[1,1] >= .50 {
      disp "BIC: not calculated because error bigger than .50"
      disp "Subj:  " `u'  "   Strat: " `v' "   - BIC: not
      calculated because epsilon bigger or equal than .50"
      }
}

if strat_vec[1,`v'] ==2 {
```

```
      ml model lf ML_choice_time_conf_flat_conf (epsilon: dv=)
      (mu_Time:) (sigma_Time:) (R_Time:) (mu_Conf:)
      (sigma_Conf:) if Participant ==`u'& strat == `v'
      ml init 0.5 0  2500 1000 37  45, copy
      ml maximize
      matrix coef = e(b)
      if coef[1,1] < .50 {
       scalar BIC = -2* e(ll)+ln(N_obs_cat[1, `v'])*6
      matrix result[`u', `v'] = BIC

      disp "Subj: " `u'  "  Strat: " `v' "   - BIC:" BIC
      }

      ** The BIC score is calculated correcting for 6
      parameters for the flat confidence prediction model.

      if coef[1,1] >= .50 {
      disp "BIC: not calculated because error bigger than .50"
      disp "Subj:  " `u'  "   Strat: " `v' "   - BIC: not
      calculated because epsilon bigger or equal than .50"
      }
}
if strat_vec[1,`v'] ==3 {

      ml model lf ML_choice_time_conf_flat_both (epsilon: dv=)
      (mu_Time:) (sigma_Time:) (mu_Conf:) (sigma_Conf:) if
      Participant ==`u'& strat == `v'
      ml init 0.5 0  2500 37  45 , copy
      ml maximize
      matrix coef = e(b)
      if coef[1,1] < .50 {
       scalar BIC = -2* e(ll)+ln(N_obs_cat[1, `v'])*5
      matrix result[`u', `v'] = BIC

      disp "Subj: " `u'  "  Strat: " `v' "   - BIC:" BIC
      }
      ** The BIC score is calculated correcting for 5
      parameters for the flat time and confidence prediction
      model.

      if coef[1,1] >= .50 {
      disp "BIC: not calculated because error bigger than .50"
      disp "Subj:  " `u'  "   Strat: " `v' "   - BIC: not
      calculated because epsilon bigger or equal than .50"
      }
}
if strat_vec[1,`v'] ==4 {

      ml model lf ML_choice_time_conf_random (mu_Time: dv=)
      (sigma_Time:) (mu_Conf:) (sigma_Conf:) if Participant
      ==`u'& strat == 4
      ml init 0  2500 37  45, copy
      ml maximize
      matrix coef = e(b)
      scalar BIC = -2* e(ll)+ln(N_obs_cat[1, `v'])*4
      matrix result[`u', `v'] = BIC
```

```
            ** The BIC score is calculated correcting for 4
            parameters for the random choice model.
            ** No limit for maximum error in choices is used for
            random choice strategies.


            disp "Subj: " `u'  "  Strat: " `v' "   - BIC:" BIC
      }
}
}


matrix list result

** the last command prints the output matrix for all subjects and
all strategies
```