Supplementary Information Part B:

'Estimating Mode Effects From a Sequential Mixed-Mode Experiment Using Structural Moment Models'

S9 List of UKHLS Wave 8 variables

The variables are were selected without any consideration as to whether mode effects would be present or absent. These variables are listed in the table below. All variables are prefixed by h in the UKHLS data set and further details can be found here https://www.understandingsociety.ac.uk/documentation/mainstage/dataset-documentation. Finally, variables in bold were dropped because the estimation routine would not converge.

aidhrs	bendis1	disdif11	othben2	S_rtcon3
basrate	bendis2	disdif12	**othben3**	S_rtcon4
debty	bendis3	disdif96	**othben4**	**S_rtpro6**
dvage	bendis4	hcondn1	othben5	S_scghqa
fimnlabnet_dv	bendis5	hcondn2	othben6	S_scghqb
fimnnet_dv	bendis7	**hcondn3**	othben7	S_scghqc
fiyrdia	bendis8	**hcondn4**	othben8	**S_scghqd**
howlng	**bendis10**	hcondn5	othben9	scghqe
j2pay_dv	bendis12	**hcondn6**	othben96	scghqf
j2paynet_dv	bendis96	hcondn7	othben97	scghqg
jbhrs	bendis97	**hcondn8**	save	scghqh
jbot	benpen1	**hcondn9**	sex	scghqi
jbotpd	benpen2	**hcondn10**	smoker	scghqj
jshrs	benpen3	hcondn11	svacts1	scghqk
nadoptch	benpen4	hcondn12	**svacts2**	scghql
nchild_dv	benpen5	hcondn13	svacts3	sclfsat1
nchresp	benpen6	hcondn14	**svacts4**	sclfsat2
ncigs	**benpen7**	**hcondn15**	**svacts5**	sclfsat7
nnatch	benpen8	hcondn16	svacts96	sclfsato
nnewborn	benpen96	**hcondn17**	**svacts97**	scsf1
paygl	caruse	hcondn18	**urban_dv**	scsf2a
paynl	ccare	**hcondn19**	wkplsam	scsf2b
saved	**cohab**	**hcondn20**	ethclose1	scsf3a
scghq1_dv	**debt1**	hcondn96	ethclose2a	scsf3b
scghq2_dv	**debt2**	hcondns1	ethclose3	scsf4a
sf12mcs_dv	**debt4**	health	ethclose4a	scsf4b
sf12pcs_dv	debt5	hospc1	ethclose4b	scsf5
workdis	debt6	j2has	ethclose11	scsf6a
aidhh	debt7	jbhas	ethid4a	scsf6b
aidxhh	debt8	**jboff**	**ethid14**	scsf6c
basnset	debt96	jbpen	finfut	scsf7
basrest	**debt97**	jbpenm	finnow	scwhoruage
benbase1	disdif1	jbsamr	food1	scwhoruedu
benbase2	disdif2	jbsect	jbsat	scwhorufam
benbase3	disdif3	jbsemp	oprlg3	scwhorupol

benbase4	disdif4	julk4wk	pride5	scwhorupro
benbase96	disdif5	julkjb	**pride7**	scwhorurac
benctc	disdif6	**lkmove**	retsuf	scwhorusex
	disdif7	mstatsam1	rtcon1	**twkcar**
	disdif8	ncrr1	rtcon2	wkaut5
	disdif9	**oprlg**		wkends
	disdif10	othben1		

S10 Stata code for g-estimation of SMoMs, SVM and SCMs

Throughout, the survey variables are denoted by X or Y. The IV is (randomized) mode M and the final mode chosen by the individual by D. The analyst is responsible for restricting estimation to the complete cases sample for a particular survey variable(s).

S10.1 Estimating the Structural Variance Model (SVM)

Note that the association model (1), linear SMM for the mean of X (2) and SVM for the variance of X (3) are fitted jointly. The `vce(robust)` option is necessary to ensure the correct standard errors are output. Analytical derivatives are specified to speed up fitting but are optional. Users should specify weight as a global name using an empty string if they have no weights or "`weightvarname`" otherwise).

```
global weight = "h_indinub_xw" // set = "" if no weight required
svyset h_psu [pweight = $weight], strata(h_strata) singleunit(scaled) // omit if i.i.d.

gmm (1: X - {b0_X} - {b1_X}*M - {b2_X}*D - {b12_X}*MD) ///
(2: X - {mu0_X} - {mu1_X}*D) ///
(3: -{lambda0_X} + exp(-D*{lambda1_X})*(X - {b0_X} - {b1_X}*M - {b2_X}*D - {b12_X}*MD)^2 + ///
(({b0_X} + {b1_X}*M + ({b2_X}-{mu1_X})*D + {b12_X}*MD)^2)) ///
[pweight=$weight], ///
onestep winitial(identity) vce(robust) ///
instruments(1: M D MD) instruments(2 3: M) ///
deriv(1/b0_X = -1) deriv(1/b1_X = -M) deriv(1/b2_X = -D) deriv(1/b12_X = -MD) ///
deriv(2/mu0_X = -1) deriv(2/mu1_X = -D) ///
deriv(3/lambda0_X = -1) ///
deriv(3/lambda1_X = -D*exp(-{lambda1_X}*D)*(X-{b0_X}-{b1_X}*M-{b2_X}*D-{b12_X}*MD)^2) ///
deriv(3/mu1_X = -2*({b0_X}+{b1_X}*M+({b2_X}-{mu1_X})*D+{b12_X}*MD)*D) ///
deriv(3/b0_X = -2*(exp(-{lambda1_X}*D)*(X-{b0_X}-{b1_X}*M-{b2_X}*D-{b12_X}*MD) -
({b0_X}+{b1_X}*M+({b2_X}-{mu1_X})*D+{b12_X}*MD))*1) ///
deriv(3/b1_X = -2*(exp(-{lambda1_X}*D)*(X-{b0_X}-{b1_X}*M-{b2_X}*D-{b12_X}*MD) -
({b0_X}+{b1_X}*M+({b2_X}-{mu1_X})*D+{b12_X}*MD))*M) ///
deriv(3/b2_X = -2*(exp(-{lambda1_X}*D)*(X-{b0_X}-{b1_X}*M-{b2_X}*D-{b12_X}*MD) -
({b0_X}+{b1_X}*M+({b2_X}-{mu1_X})*D+{b12_X}*MD))*D) ///
deriv(3/b12_X = -2*(exp(-{lambda1_X}*D)*(X-{b0_X}-{b1_X}*M-{b2_X}*D-{b12_X}*MD) -
({b0_X}+{b1_X}*M+({b2_X}-{mu1_X})*D+{b12_X}*MD))*MD)
```

S10.2 Calculating the linearized variance estimate for the SVM

It is assumed the user has run the command above.

```
matrix b=e(b) // store point estimates from fit above
local N = e(N) // store sample size from fit above
matrix srsV=e(V) // store estimated (i.i.d.) variance-covariance matrix from above
matrix ses_srs=vecdiag(cholesky(diag(vecdiag(srsV)))) //extract the standing errors
quietly summarize $weight
scalar WN=r(mean)*r(N)
matrix eG=e(G)

capture drop e_X \\ Create the association model residual
predict e_X, residual eq(#1)
capture drop u_X \\ Create the linear SMM residual
```

```
predict u_X, residual eq(#2)
capture drop v_X
predict v_X, residual eq(#3)

* Generate variables for the columns of sumSn (generally need to check ordering)
* Note that implicitly specified constant instrument is always last
capture drop sn1
gen sn1=e_X*M
capture drop sn2
gen sn2=e_X*D
capture drop sn3
gen sn3=e_X*MD
capture drop sn4
gen sn4=e_X
capture drop sn5
gen sn5=u_X*M
capture drop sn6
gen sn6=u_X
capture drop sn7
gen sn7=v_X*M
capture drop sn8
gen sn8=v_X

 *Calculate the design-consistent estimate of the sum of 'scores'
quietly svy: total sn1 sn2 sn3 sn4 sn5 sn6 sn7 sn8
mattix sumSn=e(V)
matrix eG=eG*scalar(WN)

* Calculate V
matrix tempV=inv(eG)*sumSn*inv(eG')
mat ses_lin=vecdiag(cholesky(diag(vecdiag(tempV)))) \\ extracting the SEs
matrix est=b' , ses_srs', ses_lin' \\ Store estimates and both sets of estimated SEs
```

The same approach can be used for all of the subsequent commands.

S10.3 Estimating the Structural Variance Model (SCM)

Note: the association models for X and Y (1, 2), the linear SMMs for X and Y (3, 4) and the SCM (5) are all fitted jointly.

```
gmm (1: X - {b0_X} - {b1_X}*M - {b2_X}*D - {b12_X}*MD) (2: Out - {b0_Y} - {b1_Y}*M - {b2_Y}*D
- {b12_Y}*MD) ///
(3: X - {mu0_X} - {mu1_X}*D) (4: Out - {mu0_Y} - {mu1_Y}*D) ///
(5: (X - {b0_X} - {b1_X}*M - {b2_X}*D - {b12_X}*MD)*(Out - {b0_Y} - {b1_Y}*M - {b2_Y}*D -
{b12_Y}*MD) - {gamma0_XY} - {gamma1_XY}*D ///
 + ({b0_X} + {b1_X}*M + ({b2_X}-{mu1_X})*D + {b12_X}*MD)*({b0_Y} + {b1_Y}*M + ({b2_Y}-
{mu1_Y})*D + {b12_Y}*MD)) ///
[pweight=$wweight], ///
instruments(1 2: M D MD) instruments(3 4 5: M) ///
onestep winitial(identity) vce(robust) ///
deriv(1/b0_X = -1) deriv(1/b1_X = -M) deriv(1/b2_X = -D) deriv(1/b12_X = -MD) ///
deriv(2/b0_Y = -1) deriv(2/b1_Y = -M) deriv(2/b2_Y = -D) deriv(2/b12_Y = -MD) ///
deriv(3/mu0_X = -1) deriv(3/mu1_X = -D) ///
deriv(4/mu0_Y = -1) deriv(4/mu1_Y = -D) ///
deriv(5/gamma0_XY = -1) deriv(5/gamma1_XY = -D) ///
deriv(5/b0_X = -(Out -2*{b0_Y} - 2*{b1_Y}*M - 2*{b2_Y}*D + {mu1_Y}*D - 2*{b12_Y}*MD)) ///
deriv(5/b1_X = -(Out -2*{b0_Y} - 2*{b1_Y}*M - 2*{b2_Y}*D + {mu1_Y}*D - 2*{b12_Y}*MD)*M) ///
deriv(5/b2_X = -(Out -2*{b0_Y} - 2*{b1_Y}*M - 2*{b2_Y}*D + {mu1_Y}*D - 2*{b12_Y}*MD)*D) ///
deriv(5/b12_X = -(Out -2*{b0_Y} - 2*{b1_Y}*M - 2*{b2_Y}*D + {mu1_Y}*D - 2*{b12_Y}*MD)*MD) ///
deriv(5/b0_Y = -(X -2*{b0_X} - 2*{b1_X}*M - 2*{b2_X}*D + {mu1_X}*D - 2*{b12_X}*MD)) ///
deriv(5/b1_Y = -(X -2*{b0_X} - 2*{b1_X}*M - 2*{b2_X}*D + {mu1_X}*D - 2*{b12_X}*MD)*M) ///
deriv(5/b2_Y = -(X -2*{b0_X} - 2*{b1_X}*M - 2*{b2_X}*D + {mu1_X}*D - 2*{b12_X}*MD)*D) ///
deriv(5/b12_Y = -(X -2*{b0_X} - 2*{b1_X}*M - 2*{b2_X}*D + {mu1_X}*D - 2*{b12_X}*MD)*MD) ///
deriv(5/mu1_X = -({b0_Y} + {b1_Y}*M + {b2_Y}*D - {mu1_Y}*D + {b12_Y}*MD)*D) ///
deriv(5/mu1_Y = -({b0_X} + {b1_X}*M + {b2_X}*D - {mu1_X}*D + {b12_X}*MD)*D)
```

S10.4 Estimating the multivariate SMM for categorical variables

Users must define the reference category for each of their categorical variables and set up the dummy variables to represent it in the fit.

```
* Ensure the variable is recoded so its levels run consecultively from 1 to L + 1
local Xvar="S_scsf2a" \\ Create local string containing variable name
local refcat = 1  \\ Define the reference category

capture drop dumm1_*
quietly tab1 `Xvar', gen(dumm1_)
scalar nolevs=r(r)
drop dumm1_*

local startk=1
local endk=scalar(nolevs)

local cmmd="gmm "
forvalues k=`startk'/`endk' {
   if (`k' != `refcat') {
      local tmp="(`k': dumm1_`k' - {beta0_`k'} - {beta1_`k'}*T_mode) "
      local cmmd="`cmmd'" + "`tmp'"
  }
}
local cmmd="`cmmd'" + "[pweight=$wgtvar], winitial(identity) onestep vce(robust)
instruments(IV_mode)"
display "`cmmd' "  \\ Show the command to be run
`cmmd'             \\ Run the command
```

Note that `beta1_2, beta1_3`, etc. are the effects of mode on the non-reference categories.

S10.5 Estimating the multivariate SMM for the joint distribution of two categorical variables

Both variables should be coded to run from 1 to L and 1 to M and the reference categories specified.

```
local Xvar="S_scsf2a"
local refcatX = 1
local Yvar="S_scsf3a"
local refcatY = 1

capture drop dummX_*
quietly tab1 `Xvar' if e(sample), gen(dummX_)
scalar nolevX=r(r)
capture drop dummY_*
quietly tab1 `Yvar' if e(sample), gen(dummY_)
scalar nolevY=r(r)

local endX=scalar(nolevX)
local endY=scalar(nolevY)

local cmmd="gmm "
local derivs = ""
local m=0
forvalues k=1/`endX' {
   forvalues l = 1/`endY' {
      if (`k' != `refcatX') & (`l' != `refcatY') {
        local m=`m'+1
        local tmp="( `m': dummX_`k'*dummY_`l' - {beta0_`k'_`l'} - ({betaX_`k'} + {betaY_`l'} +
{int_`k'_`l'})*T_mode) "
         local cmmd="`cmmd'" + "`tmp'"
         local tmp="deriv(`m'/beta0_`k'_`l' = -1) deriv(`m'/betaX_`k' = -T_mode)
deriv(`m'/betaY_`l' = -T_mode) deriv(`m'/int_`k'_`l' = -T_mode) "
         local derivs = "`derivs'" + "`tmp'"
      }
      else if (`k' != `refcatX') & (`l' == `refcatY') {
         local m=`m'+1
         local tmp="(`m': dummX_`k'*dummY_`l' - {beta0_`k'_`l'} - {betaX_`k'}*T_mode) "
         local cmmd="`cmmd'" + "`tmp'"
         local tmp="deriv(`m'/beta0_`k'_`l' = -1) deriv(`m'/betaX_`k' = -T_mode) "
         local derivs = "`derivs'" + "`tmp'"
        }
```

```
         else if (`k' == `refcatX') & (`l' != `refcatY') {
            local m=`m'+1
            local tmp="(`m': dummX_`k'*dummY_`l' - {beta0_`k'_`l'} - {betaY_`l'}*T_mode) "
            local cmmd="`cmmd'" + "`tmp'"
            local tmp="deriv(`m'/beta0_`k'_`l' = -1) deriv(`m'/betaY_`l' = -T_mode) "
            local derivs = "`derivs'" + "`tmp'"
          }
 }
}

local cmmd="`cmmd'" + "[pweight=`wgtvar'], winitial(identity) onestep vce(robust)
instruments(IV_mode)"
display "`cmmd' "
`cmmd'
```

S10.6 Incorporating auxiliary data into the SVM

Incorporating auxiliary data from the ringfence and low-propensity samples into the SVM estimation (S10.1).

The data from the sequential experiment, ringfence sample and low-propensity sample are concatenated into a single data set. Then three indicator variables are defined to indicate which row belongs to which sample: `expt`, `ring` and `low` together with `notring = 1 - ring`.

Additionally, three sample-specific versions of X are created equal to X if the row is in that sample and zero otherwise: `eX`, `Xlow` and `Xring` respectively for `expt`, `ring` and `low`. Similarly, for the IV M, mode D and interaction MD, and square of X: `eM`, `Mow` and `Mring`; `eD`, `Dlow` and `Dring`; `eMD`, `MDlow` and `MDring`; and `eX2`, `X2low` and `X2ring`.

Finally, store the point estimates from fitting the SVM

```
matrix b=e(b)
matrix define bstart=0,0,0,0,0.5,b[1,1],b[1,2],b[1,3],b[1,4],b[1,6],b[1,8]
```

then

```
gmm (1: Xring - {m0_ring}*ring)       (2: Xlow - {m0_low}*low) (3: X2ring - {l0_ring}*ring)
///
(4: X2low - {l0_low}*low) (5: low*notring - {plow}*notring) (6: eX - {b0_X}*expt - {b1_X}*eM -
{b2_X}*eD - {b12_X}*eMD) ///
(7: eX - (({m0_ring}-{plow}*{m0_low})/(1-{plow}))*expt - {mu1_X}*eD) ///
(8: -(({l0_ring}-{plow}*{l0_low})/(1-{plow}))*expt + exp(-eD*{lambda1_X})*(eX - {b0_X}*expt -
{b1_X}*eM - {b2_X}*eD - {b12_X}*eMD)^2 + ///
   (({b0_X}*expt + {b1_X}*eM + ({b2_X}-{mu1_X})*eD + {b12_X}*eMD)^2)) ///
$weightvar from(bstart) ///
instruments(1 3: ring, nocons) instruments(2 4: low, nocons) instruments(5: notring, nocons)
instruments(7 8: eM, nocons) ///
instruments(6: eM eD eMD expt, nocons) ///
onestep winitial(identity) vce(robust) ///
deriv(1/m0_ring = - ring) deriv(2/m0_low = - low) deriv (3/l0_ring = -ring) deriv(4/l0_low = -
low) deriv(5/plow = -notring) ///
deriv(6/b0_X = -expt) deriv(6/b1_X = -eM) deriv(6/b2_X = -eD) deriv(6/b12_X = -eMD) ///
deriv(7/m0_ring = -expt/(1-{plow})) deriv(7/m0_low = {plow}*expt/(1-{plow})) deriv(7/mu1_X = -
eD) ///
deriv(7/plow = {m0_low}*expt/(1-{plow}) - ({m0_ring}-{plow}*{m0_low})*expt/((1-{plow})^2)) ///
deriv(8/l0_ring = -expt/(1-{plow})) deriv(8/l0_low = {plow}*expt/(1-{plow})) ///
deriv(8/plow = {l0_low}*expt/(1-{plow}) - ({l0_ring}-{plow}*{l0_low})*expt/((1-{plow})^2)) ///
deriv(8/lambda1_X = -eD*exp(-{lambda1_X}*eD)*(eX-{b0_X}*expt-{b1_X}*eM-{b2_X}*eD-
{b12_X}*eMD)^2) ///
deriv(8/mu1_X = -2*({b0_X}*expt+{b1_X}*eM+({b2_X}-{mu1_X})*eD+{b12_X}*eMD)*eD) ///
deriv(8/b0_X = -2*(exp(-{lambda1_X}*eD)*(eX-{b0_X}*expt-{b1_X}*eM-{b2_X}*eD-{b12_X}*eMD) -
({b0_X}*expt+{b1_X}*eM+({b2_X}-{mu1_X})*eD+{b12_X}*eMD))*expt) ///
deriv(8/b1_X = -2*(exp(-{lambda1_X}*eD)*(eX-{b0_X}*expt-{b1_X}*eM-{b2_X}*eD-{b12_X}*eMD) -
({b0_X}*expt+{b1_X}*eM+({b2_X}-{mu1_X})*eD+{b12_X}*eMD))*eM) ///
deriv(8/b2_X = -2*(exp(-{lambda1_X}*eD)*(eX-{b0_X}*expt-{b1_X}*eM-{b2_X}*eD-{b12_X}*eMD) -
({b0_X}*expt+{b1_X}*eM+({b2_X}-{mu1_X})*eD+{b12_X}*eMD))*eD) ///
deriv(8/b12_X = -2*(exp(-{lambda1_X}*eD)*(eX-{b0_X}*expt-{b1_X}*eM-{b2_X}*eD-{b12_X}*eMD) -
({b0_X}*expt+{b1_X}*eM+({b2_X}-{mu1_X})*eD+{b12_X}*eMD))*eMD)
```

The same approach can be used for all the other methods.

To test the NEM assumption, the SMM (7) and SVM (8) models are extended as follows:

```
(7: eX - (({m0_ring}-{plow}*{m0_low})/(1-{plow}))*expt - {mu1_X}*eD - {dmu1_X}*eMD) ///
       (8: -(({l0_ring}-{plow}*{l0_low})/(1-{plow}))*expt + exp(-eD*{lambda1_X}-
eMD*{dlambda1_X})*(eX - {b0_X}*expt - {b1_X}*eM - {b2_X}*eD - {b12_X}*eMD)^2 + ///
    (({b0_X}*expt + {b1_X}*eM + ({b2_X}-{mu1_X})*eD + ({b12_X}-{dmu1_X})*eMD)^2)) ///
```

and the extended instrument set `instruments(7 8: eM expt, nocons)` used to identify the additional parameters `dmu1` and `dlambda1`.

Inference for the combined estimate without assuming NEM is carried out using the delta method as follows:

```
quietly mean M `pweight' if (D==1 & expt==1)
matrix tmpb=e(b)
scalar p1=tmpb[1,1]
scalar p0=1-scalar(p1)
quietly regress X `pweight' if (D==1 & expt==1)
scalar S21=e(rmse)^2
quietly regress X `pweight' if (M==0 & D==1 & expt==1)
matrix tmpb=e(b)
scalar S01=tmpb[1,1]
scalar S201=e(rmse)^2
quietly regress X `pweight' if (M==1 & D==1 & expt==1)
matrix tmpb=e(b)
scalar S11=tmpb[1,1]
scalar S211=e(rmse)^2
scalar Sdiff=scalar(S11)-scalar(S01)


scalar A=scalar(p0)*scalar(S201)*exp(-`lambda1_X')+scalar(p1)*scalar(S211)*exp(-`lambda1_X'-
`dlambda1_X') + scalar(p0)*scalar(p1)*(scalar(Sdiff)^2 + `dmu1_X'^2 -
2*scalar(Sdiff)*`dmu1_X') \\ p1 = Pr(M = 1 | D = 1) and p0 = 1 - p1, S201

local mu1_adj_gmm1nem = `mu1_X' + scalar(p1)*`dmu1_X'
local lam1_adj_gmm1nem =log(scalar(S21)) - log(scalar(A))

scalar deriv1=2*scalar(p0)*scalar(p1)*(scalar(Sdiff)-`dmu1_X')/scalar(A)
scalar deriv2=(scalar(p0)*scalar(S201)*exp(-`lambda1_X') + scalar(p1)*scalar(S211)*exp(-
`lambda1_X'-`dlambda1_X'))/scalar(A)
scalar deriv3=scalar(p1)*scalar(S211)*exp(-`lambda1_X'-`dlambda1_X')/scalar(A)

matrix define H = 1, 0 \ scalar(p1), scalar(deriv1) \ 0, scalar(deriv2) \ 0, scalar(deriv3)

matrix tV=tempV[10..13,10..13] \\ Required subset of the NEM-test model covariance matrix
matrix adjest=H' * tV * H
```

A similar approach is used for the SCM but this is not shown here.

S10.7 Approximate and exact estimates for mode effect on linear regression coefficient

Begin by fitting the normal linear regression to the mixed-mode data (note CCsample is the common complete-cases sample; `vce(robust)` selects the sandwich estimator of the variance).

```
   mlexp(ln(normalden(`1', {b1}*`2' + {b2}*`3', {sig2}))) if CCsample, vce(robust)

* Store variance-covariance matrix

   scalar ssize=e(N)
   matrix bcmle=e(b)
```

```
   matrix V=e(V)


* For approximate mode effect calculation, create natural-parameter derivatives



   matrix define derivs=0, -_b[/b1]/(_b[/sig2]^2), 0, 1/(_b[/sig2]^2), 0, -
_b[/b2]/(_b[/sig2]^2) ///
    \ 0, 0, -_b[/b2]/(_b[/sig2]^2), 0, 1/(_b[/sig2]^2), -_b[/b1]/(_b[/sig2]^2) ///
    \ 1/(2*_b[/sig2]^4), _b[/b1]^2/(2*_b[/sig2]^4), _b[/b2]^2/(2*_b[/sig2]^4), ///
    -_b[/b1]/(_b[/sig2]^4), -_b[/b2]/(_b[/sig2]^4), _b[/b1]*_b[/b2]/(_b[/sig2]^4)


   matrix rownames derivs = b1 b2 sig2

   matrix colnames derivs = Y2 X12 X22 X1Y X2Y X1X2
```

Set up the necessary variables for GMM estimation of the mode effects on the sufficient statistics of the multivariate normal (variable `1' is the Y variable, `2' is predictor X1 and `3' is predictor X2: do not forget that all should be mean-centred).

```
   tempvar Y_2
   tempvar X1_2
   tempvar X2_2
   tempvar X1Y
   tempvar X2Y
   tempvar X1X2
   tempvar Y
   tempvar X1
   tempvar X2
   gen `Y'=`1'
   gen `X1'=`2'
   gen `X2'=`3'
   gen `Y_2' = (`1')^2
   gen `X1_2' = (`2')^2
   gen `X2_2'=(`3')^2
   gen `X1Y' = `2'*`1'
   gen `X2Y' = `3'*`1'
   gen `X1X2' = `2'*`3'
```

Now use GMM to estimate the SMoM for the mode effects:

```
  local i=0
   foreach var of varlist `Y_2' `X1_2' `X2_2' `X1Y' `X2Y' `X1X2' {
      gmm (1: `var' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode) onestep
winitial(unadjusted) vce(robust) deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
      matrix b=e(b)
          scalar par=b[1,1]
      if (`i'==0) {
         matrix define modests = scalar(par)
      }
      else {
         matrix define modests = modests \ scalar(par)
      }
      local i=`i'+1
   }
```

Use these estimates to adjust the observed score function and fit (also using GMM):

```
   * Solve score equation for normal-linear regression with mode-effect adjustments
   gmm (1: `X1Y' - T_mode*modests[4,1] - {beta1}*(`X1_2' - T_mode*modests[2,1]) -
{beta2}*(`X1X2' - T_mode*modests[6,1])) ///
       (2: `X2Y' - T_mode*modests[5,1] - {beta1}*(`X1X2' - T_mode*modests[6,1]) -
{beta2}*(`X2_2' - T_mode*modests[3,1])) if CCsample, ///
       onestep winitial(identity) vce(robust)
   matrix exact=e(b)

   matrix define ediff=bcmle[1,1]-exact[1,1],bcmle[1,2]-exact[1,2]
```

```
   * Need to name columns of differences vector for bootstrap command to process results

   matrix colnames ediff=eb1:_cons eb2:_cons

   display "Exact diff: beta1:", ediff[1,1]
   display "Exact diff: beta2:", ediff[1,2]
```

Now calculate the approximate estimate of the mode effect:

```
   matrix diff=scalar(ssize)*scalar(pi)*V*derivs*modests
```

S10.8 Approximate and exact estimates for mode effect on linear regression coefficient

The mode effect on the coefficient of the logistic regression is performed iteratively.

First, as in S10.7 above, set up the temporary variables for the sufficient statistics. Note also that `1' is the Y variable, X1 is the first predictor and X2 is the second predictor. None of the variables is mean-centred in this example.

```
  tempvar X1
   tempvar X2
   tempvar X1Y
   tempvar X2Y
   tempvar X1X2
   tempvar mu
   tempvar Y
   tempvar X1mu
   tempvar X2mu

   gen `Y' = `1'
   gen `X1' = `2'
   gen `X2' = `3'

   gen `X1Y' = `2'*`1'
   gen `X2Y' = `3'*`1'
   gen `X1X2' = `2'*`3'
   gen `mu' = .
   gen `X1mu' = .
   gen `X2mu' = .
```

Fit the logistic regression to the mixed-mode data to set the complete-cases sample

```
*   quietly logit `1' `2' `3' if CCsample
```

Now fit the same model but using (the equivalent) GMM estimator

```
   * Fit observed regression
   * Solve logistic score function using GMM
   gmm (1: `1'-1/(1+exp(-{beta0}-{beta1}*`2' - {beta2}*`3'))    ) ///
       (2: (`1'-1/(1+exp(-{beta0}-{beta1}*`2'-{beta2}*`3')))*`2') ///
       (3: (`1'-1/(1+exp(-{beta0}-{beta1}*`2'-{beta2}*`3')))*`3') if CCsample, ///
       onestep winitial(identity) vce(robust) quickd

* Store variance-covariance matrix
   scalar ssize=e(N)
   matrix bobs=e(b)
   matrix V=e(V)

* Calculating additive mode effects

        * Take this as the starting value
       matrix b_last=bobs
       matrix list b_last
       replace `mu'=1/(1+exp(-b_last[1,1]-b_last[1,2]*`2'-b_last[1,3]*`3')) if CCsample

    gmm (1: `Y' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode) ///
              onestep winitial(unadjusted) vce(robust) deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
    matrix tmp=e(b)
    scalar modeY=tmp[1,1]
```

```
        gmm (1: `X1Y' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode) ///
              onestep winitial(unadjusted) vce(robust) deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
     matrix tmp=e(b)
     scalar modeX1Y=tmp[1,1]
     gmm (1: `X2Y' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode) ///
              onestep winitial(unadjusted) vce(robust) deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
     matrix tmp=e(b)
     scalar modeX2Y=tmp[1,1]

   * Solve score equation for normal-linear regression with mode-effect adjustments
   * Substantially simpler than the method above to implement

        local stp=0
        local nk=0

        * Iterative fitting loop here
        while `stp'==0 {
           local nk=`nk'+1
           display "Replication -- iteration `nk'"
           * Mode effect on mu(X1)
           quietly gmm (1: `mu' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode)
onestep winitial(unadjusted) vce(robust) ///
             deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
           matrix tmp=e(b)
           scalar modeMu=tmp[1,1]
           * Mode effect on mu(X1)*X1
           replace `X1mu'=`X1'*`mu'
           quietly gmm (1: `X1mu' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode)
onestep winitial(unadjusted) vce(robust) ///
              deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
           matrix tmp=e(b)
           scalar modeX1Mu=tmp[1,1]
                  * Mode effect on mu(X2)*X2
           replace `X2mu'=`X2'*`mu'
           quietly gmm (1: `X2mu' - {mu}*T_mode - {mu0}) if CCsample, instruments(1: IV_mode)
onestep winitial(unadjusted) vce(robust) ///
           deriv(1/mu = -T_mode) deriv(1/mu0 = -1)
           matrix tmp=e(b)
           scalar modeX2Mu=tmp[1,1]

           * Fit current-estimate-adjusted logit model to update beta

               noisily capture gmm (1: `1' - scalar(modeY)*T_mode -1/(1+exp(-{beta0}-
{beta1}*`2'-{beta2}*`3')) + scalar(modeMu)*T_mode) ///
                        (2: (`1'*`2'-scalar(modeX1Y)*T_mode-`2'/(1+exp(-{beta0}-{beta1}*`2'-
{beta2}*`3')) + scalar(modeX1Mu)*T_mode)) ///
                                  (3: (`1'*`3'-scalar(modeX2Y)*T_mode-`3'/(1+exp(-{beta0}-
{beta1}*`2'-{beta2}*`3')) + scalar(modeX2Mu)*T_mode)) ///
          if CCsample, onestep winitial(identity) vce(robust) quickd

       scalar cng=e(converged)

       matrix b=e(b)
           scalar chg=mreldif(b,b_last)
*          scalar chg=abs(scalar(b_curr)-scalar(b_prev))
           display "Change = ",scalar(chg)
           if (scalar(chg)< $tol) {
           local stp = 1
                 display "`stp'"
           }
           else {
              if (`nk'>$maxiter) | (scalar(cng)==0) | (_rc ~= 0) {
                  `stp'==0
                     }
              else {
                 matrix b_last=b
                    replace `mu'=1/(1+exp(-b_last[1,1]-b_last[1,2]*`2'-b_last[1,3]*`3'))
                 }
           }

       }

   matrix bfinal=b
   matrix meff=bobs-bfinal
   matrix b=bfinal, meff
   matrix colnames b=abs:b0 abs:b1 abs:b2 mode:meff0 mode:meff1 mode:meff2
```