# ONLINE APPENDIX

**Section 1: Code for fitting NPMM (here, Equation 13 with *K*=3 and *H*=3) in Mplus**

```
TITLE:    example
DATA:    FILE =  C:\filepath\example.dat;
VARIABLE:
NAMES ARE y x w cluster_id subject_id;
          USEVARIABLES = y x w;
          CLASSES = cb (3) cw (3);
          !cb denotes level-2 class, cw denotes level-1 class
          WITHIN =x cw;
          BETWEEN = w cb;
          CLUSTER = cluster_id;
ANALYSIS:
          TYPE = TWOLEVEL MIXTURE;
          ESTIMATOR=ML;
MODEL:
%WITHIN%
%OVERALL%
  y on x;
  [cw#1] (mcw1); !multinomial intercept for level-1 class k
  [cw#2] (mcw2);
%cb#1.cw#1%
   y (rv1); !class-combination residual variance
   y on x (a1); !class-combination slope of x
%cb#1.cw#2%
   y  (rv2); y on x  (a2);
%cb#1.cw#3%
   y  (rv3); y on x  (a3);
%cb#2.cw#1%
   y  (rv4); y on x  (a4);
%cb#2.cw#2%
   y  (rv5); y on x  (a5);
%cb#2.cw#3%
   y  (rv6); y on x  (a6);
%cb#3.cw#1%
   y  (rv7); y on x  (a7);
%cb#3.cw#2%
   y  (rv8); y on x  (a8);
%cb#3.cw#3%
   y  (rv9); y on x  (a9);
%BETWEEN%
%OVERALL%
   cw#1 on cb#1 (ms11); !multinomial slope of k on h
   cw#1 on cb#2 (ms12);
   cw#2 on cb#1 (ms21);
   cw#2 on cb#2 (ms22);
   y on w; [y];
   [cb#1] (mcb1); !multinomial intercept for level-2 class h
   [cb#2] (mcb2);
%cb#1.cw#1%
   [y ] (i1); !class-combination intercept
   y on w  (b1); !class-combination slope of w
```

# ONLINE APPENDIX

```
   y@0;
%cb#1.cw#2%
   [y ] (i2); y on w  (b2); y@0;
%cb#1.cw#3%
   [y ] (i3); y on w  (b3); y@0;
%cb#2.cw#1%
   [y ] (i4); y on w  (b4); y@0;
%cb#2.cw#2%
   [y ] (i5); y on w  (b5); y@0;
%cb#2.cw#3%
   [y ] (i6); y on w  (b6); y@0;
%cb#3.cw#1%
   [y ] (i7); y on w  (b7); y@0;
%cb#3.cw#2%
   [y ] (i8); y on w  (b8); y@0;
%cb#3.cw#3%
   [y ] (i9); y on w  (b9); y@0;
```

**Section 2: Code for testing NPMM implied random slope variance in Mplus (code accompanies K=3, H=3 class model above, in Section 1). For further details, see manuscript Discussion.**

```
MODEL CONSTRAINT:
NEW
  (p_cb1cw1 p_cb1cw2 p_cb1cw3
   p_cb2cw1 p_cb2cw2 p_cb2cw3
   p_cb3cw1 p_cb3cw2 p_cb3cw3
   p_cb1 p_cb2 p_cb3
   p_11 p_12 p_13
   p_21 p_22 p_23
   p_31 p_32 p_33
   H1xslope H2xslope H3xslope);
p_cb1cw1=exp(mcw1+ms11) /
               (exp(mcw1+ms11)+
               exp(mcw2+ms21)+
                 exp(0+0));
p_cb2cw1=exp(mcw1+ms12) /
               (exp(mcw1+ms12)+
               exp(mcw2+ms22)+
                 exp(0+0));
p_cb3cw1=exp(mcw1+0) /
               (exp(mcw1+0)+
               exp(mcw2+0)+
                 exp(0+0));
p_cb1cw2=exp(mcw2+ms21) /
              (exp(mcw1+ms11)+
               exp(mcw2+ms21)+
                 exp(0+0));
p_cb2cw2=exp(mcw2+ms22) /
               (exp(mcw1+ms12)+
               exp(mcw2+ms22)+
```

# ONLINE APPENDIX

```
                exp(0+0));
p_cb3cw2=exp(0+0) /
               (exp(mcw1+0)+
                exp(mcw2+0)+
               exp(0+0));
p_cb1cw3=exp(0+0) /
               (exp(mcw1+ms11)+
               exp(mcw2+ms21)+
                exp(0+0));
p_cb2cw3=exp(0+0) /
               (exp(mcw1+ms12)+
               exp(mcw2+ms22)+
                exp(0+0));
p_cb3cw3=exp(0+0) /
               (exp(mcw1+0)+
                exp(mcw2+0)+
               exp(0+0));
p_cb1=EXP(mcb1)/(EXP(mcb1)+EXP(mcb2)+EXP(0));
p_cb2=EXP(mcb2)/(EXP(mcb1)+EXP(mcb2)+EXP(0));
p_cb3=EXP(0)/(EXP(mcb1)+EXP(mcb2)+EXP(0));
p_11=p_cb1*p_cb1cw1;
p_12=p_cb1*p_cb1cw2;
p_13=p_cb1*p_cb1cw3;
p_21=p_cb2*p_cb2cw1;
p_22=p_cb2*p_cb2cw2;
p_23=p_cb2*p_cb2cw3;
p_31=p_cb3*p_cb3cw1;
p_32=p_cb3*p_cb3cw2;
p_33=p_cb3*p_cb3cw3;
H1xslope=((p_cb1cw1)*a1+(p_cb1cw2)*a2+(p_cb1cw3)*a3);
H2xslope=((p_cb2cw1)*a4+(p_cb2cw2)*a5+(p_cb2cw3)*a6);
H3xslope=((p_cb3cw1)*a7+(p_cb3cw2)*a8+(p_cb3cw3)*a9);
MODEL TEST:
  !test implied randslope of x
  H1xslope=H2xslope;
  H2xslope=H3xslope;
```

# ONLINE APPENDIX

**Section 3: R function to implement NPMM approximation calculations**

**Description of function:**

With this R function (*npmmApproximation*), users can calculate the NPMM-approximated fixed effects, random effect (co)variances, and level-1/level-2 heteroscedasticity components in MLM, for a specified *K* and *H* (if one model is fit), or a specified range of *K* and *H* (if multiple models are fit, e.g., in the context of model selection). This function accommodates any number of level-1 and level-2 predictors in the general NPMM of Equation (15). Users first run NPMM models in Mplus for each desired combination of number of level-1 classes (*K*) and level-2 classes (*H*) while outputting results in Mplus with the following command format:
SAVEDATA: RESULTS ARE example_HbyK.dat;

This function provides a table of results and can also produce plots such as in Figures 6 and 7 of the manuscript. Data can be manually entered (for a single *K, H* model) or can be read in automatically from a set of Mplus output files for multiple models (with different combinations of *K* and *H*).

Input values:
Hmin = minimum value for number of level-2 classes *H*
Hmax = maximum value for number of level-2 classes *H* (if only one *H* is desired, set Hmin=Hmax)
Kmin = minimum value for number of level-1 classes *K*
Kmax = maximum value for number of level-1 classes *K* (if only one *K* is desired, set Kmin=Kmax)
nL1vars = number of level-1 variables (i.e., number of within-level slopes in Mplus output)
nL2vars = number of level-2 variables (i.e., number of between-level slopes in Mplus output)
resvar_con = TRUE or FALSE; specifies whether residual variance is constrained across all class-combination (TRUE) or is freely estimated across all class-combinations (FALSE); set to FALSE by default
Mplusoutput = file location and *partial* name of Mplus output files used; when reading in files, function automatically adds "HbyK.dat"
plotoutput = specifies file location and name of pdf file with approximation plots showing results across the range of H and K; if FALSE (default), no plots are produced
intslopes = (NOTE: only needed if manually entering parameter estimates for a single model; set to NA by default) vector of coefficients for all class-combination intercepts and all class-combination slopes; to be entered in the following order: (1) all intercepts going in order of increasing *k* (level-1 class) then increasing *h* (level-2 class) (e.g., k=1,h=1; k=2,h=1; k=1,h=2; k=2,h=2); (2) all within-level slopes for each class-combination (classes increasing as in (1)), e.g., xslope1_k1h1, xslope2_k1h1, xslope1_k2h1, xslope2_k2h1, etc.); (3) all between-level slopes for each class-combination (classes increasing as in (1))
resvar = (NOTE: only needed if manually entering parameter estimates for a single model; set to NA by default) vector of class-combination residual variances (entered in the order of classes described in intslopes above)
mcwi = (NOTE: only needed if manually entering parameter estimates for a single model; set to NA by default) vector of level-1 class multinomial intercepts, entered in order of increasing *k*, with 0 entered for *K*
mcws = (NOTE: only needed if manually entering parameter estimates for a single model; set to NA by default) vector of multinomial slopes of *k* on *h* (entered in the order of classes described in intslopes above, with 0 entered for every *k=K* and *h=H*)
mcbi = (NOTE: only needed if manually entering parameter estimates for a single model; set to NA by default) vector of level-2 class multinomial intercepts, entered in order of increasing *h*, with 0 entered for *H*

**##R code:**

```
npmmApproximation <-
function(Hmin,Hmax,Kmin,Kmax,nL1vars,nL2vars,resvar_con=FALSE,Mplusoutput=NA,plotoutput=FALSE,
intslopes=NA, resvar=NA, mcwi=NA, mcws=NA, mcbi=NA)
{
approx.ubertable.fixed <- array(NA,c(1+nL1vars+nL2vars,1,as.numeric(paste(c(Hmax,Kmax),collapse=""))))
approx.ubertable.tau <-
array(NA,c(1+nL1vars+nL2vars,1+nL1vars+nL2vars,as.numeric(paste(c(Hmax,Kmax),collapse=""))))
```

# ONLINE APPENDIX

```
approx.ubertable.sigma <-
array(NA,c(1+nL1vars+nL2vars,1+nL1vars+nL2vars,as.numeric(paste(c(Hmax,Kmax),collapse=""))))
fixedeffects.output <- list()
tau.output <-  list()
sigma.output <-  list()
for (z in Hmin:Hmax)
{
for (v in Kmin:Kmax)
{
##specify H and K
H <- z
K <- v
if (is.na(Mplusoutput)==FALSE)
{
mplusout <- NA
intslopes <- NA
resvar <- NA
mcwi <- NA
mcws <- NA
mcbi <- NA
##read in Mplus output
#total length of file
length <- H*K*(nL1vars+1)+H*K*(nL2vars+1)+H-1+K-1+(H-1)*(K-1)
if (resvar_con==TRUE)
{
length <- H*K*nL1vars+1+H*K*(nL2vars+1)+H-1+K-1+(H-1)*(K-1)
}
mplusout <- NA
intslopes <- NA
resvar <- NA
mcwi <- NA
mcws <- NA
mcbi <- NA
##read in Mplus output data
try(
Mplusout <- matrix(data=scan(paste(Mplusoutput,H,"by",K,".dat",sep="")),length,1)
)
if (is.na(Mplusout)==TRUE) next
##intercepts
intercepts <- Mplusout[H*K*(nL1vars+1)+1,]
if (H*K > 1)
{
for (i in 1:c(K*H-1))
{
intercepts <- c(intercepts,Mplusout[c(H*K*(nL1vars+1)+1+i*(nL2vars+1)),])
}
}
if (resvar_con==TRUE)
{
intercepts <- Mplusout[H*K*nL1vars+2,]
if (H*K > 1)
{
for (i in 1:c(K*H-1))
```

# ONLINE APPENDIX

```
{
intercepts <- c(intercepts,Mplusout[c(H*K*nL1vars+2+i*(nL2vars+1)),])
}
}
}
##xslopes
xslopes <- Mplusout[1:nL1vars,]
if (H*K > 1)
{
for (i in 0:c(K*H-2))
{
xslopes <- c(xslopes,Mplusout[c(nL1vars+2+i*(nL1vars+1)):c(nL1vars+2+i*(nL1vars+1)+nL1vars-1),])
}
}
if (resvar_con==TRUE)
{
xslopes <- Mplusout[1:nL1vars,]
if (H*K > 1)
{
xslopes <- c(xslopes,Mplusout[c(nL1vars+2):c(H*K*nL1vars+1),])
}
}
##reorder
xslopes_temp <- list()
for (i in 1:nL1vars)
{
xslopes_temp[[i]] <- xslopes[seq(i, length(xslopes), nL1vars)]
if (i==1)
{
xslopes_temp2 <- xslopes_temp[[i]]
}
if (i>1)
{
xslopes_temp2 <- c(xslopes_temp2,xslopes_temp[[i]])
}
}
xslopes <- xslopes_temp2
##wslopes
wslopes <- Mplusout[c(H*K*(nL1vars+1)+2):c(H*K*(nL1vars+1)+2+nL2vars-1),]
if (H*K > 1)
{
for (i in 1:c(K*H-1))
{
wslopes <-
c(wslopes,Mplusout[c(H*K*(nL1vars+1)+2+i*(nL2vars+1)):c(H*K*(nL1vars+1)+2+i*(nL2vars+1)+nL2vars-1),])
}
}
if (resvar_con==TRUE)
{
wslopes <- Mplusout[c(H*K*nL1vars+3):c(H*K*nL1vars+3+nL2vars-1),]
if (H*K > 1)
{
for (i in 1:c(K*H-1))
```

# ONLINE APPENDIX

```
{
wslopes <- c(wslopes,Mplusout[c(H*K*nL1vars+3+i*(nL2vars+1)):c(H*K*nL1vars+3+i*(nL2vars+1)+nL2vars-1),])
}
}
}
##reorder
wslopes_temp <- list()
for (i in 1:nL1vars)
{
wslopes_temp[[i]] <- wslopes[seq(i, length(wslopes), nL1vars)]
if (i==1)
{
wslopes_temp2 <- wslopes_temp[[i]]
}
if (i>1)
{
wslopes_temp2 <- c(wslopes_temp2,wslopes_temp[[i]])
}
}
wslopes <- wslopes_temp2
##resvar
resvar=Mplusout[nL1vars+1,]
if (resvar_con==FALSE)
{
if (H*K > 1)
{
for (i in 1:c(K*H-1))
{
resvar <- c(resvar,Mplusout[c(nL1vars+1+i*(nL1vars+1)),])
}
}
}
##full ints and slopes
intslopes=c(intercepts,xslopes,wslopes)
##mcw
if (K>1)
{
mcwi=c(Mplusout[c(H*K*(nL1vars+1)+H*K*(nL2vars+1)+1+H-1):c(H*K*(nL1vars+1)+H*K*(nL2vars+1)+1+H-1+K-
2)],0)
}
if (K==1)
{
mcwi=0
}
if (resvar_con==TRUE)
{
if (K>1)
{
mcwi=c(Mplusout[c(H*K*nL1vars+1+H*K*(nL2vars+1)+1+H-2+1):c(H*K*nL1vars+1+H*K*(nL2vars+1)+1+H-
2+1+K-2)],0)
}
if (K==1)
{
```

# ONLINE APPENDIX

```
mcwi=0
}
}
##ms
if (H>1 & K>1)
{
mcws_temp=Mplusout[c(H*K*(nL1vars+1)+H*K*(nL2vars+1)+1+H-1+K-
1):c(H*K*(nL1vars+1)+H*K*(nL2vars+1)+1+H-1+K-1+(H-1)*(K-1)-1)]
mcws <- list()
for (i in 1:H)
{
mcws <- c(mcws,mcws_temp[c(1+(i-1)*(K-1)):c(i+i*(K-2))],0)
}
mcws[c((H-1)*K+1):c(H*K)] <- 0
}
if (H==1 | K==1)
{
mcws=matrix(0,H*K,1)
}
mcws <- as.numeric(mcws)
if (resvar_con==TRUE)
{
if (H>1 & K>1)
{
mcws_temp=Mplusout[c(H*K*nL1vars+1+H*K*(nL2vars+1)+1+H-2+1+K-
2+1):c(H*K*nL1vars+1+H*K*(nL2vars+1)+1+H-2+1+K-2+1+(H-1)*(K-1)-1)]
mcws <- list()
for (i in 1:H)
{
mcws <- c(mcws,mcws_temp[c(1+(i-1)*(K-1)):c(i+i*(K-2))],0)
}
mcws[c((H-1)*K+1):c(H*K)] <- 0
}
if (H==1 | K==1)
{
mcws=matrix(0,H*K,1)
}
mcws <- as.numeric(mcws)
}
##mcb
if (H>1)
{
mcbi=c(Mplusout[c(H*K*(nL1vars+1)+H*K*(nL2vars+1)+1):c(H*K*(nL1vars+1)+H*K*(nL2vars+1)+1+H-2)],0)
}
if (H==1)
{
mcbi=0
}
if (resvar_con==TRUE)
{
if (H>1)
{
mcbi=c(Mplusout[c(H*K*nL1vars+1+H*K*(nL2vars+1)+1):c(H*K*nL1vars+1+H*K*(nL2vars+1)+1+H-2)],0)
```

# ONLINE APPENDIX

```
}
if (H==1)
{
mcbi=0
}
}
}
##read in intercepts and slopes
int_slopes <- aperm(array(data=intslopes,dim=c(K,H,1+nL1vars+nL2vars)),perm=c(2,1,3))
##read in residual variances
residual_var <- matrix(data=resvar,H,K,byrow=TRUE)
##read in multinomials
#L1 class intercepts
mcw <- matrix(data=mcwi,1,K)
mcw[1,K] <- 0
#L2 class on L1 class slopes
ms <- matrix(data=mcws,H,K,byrow=TRUE)
ms[H,1:K] <- 0
ms[1:H,K] <- 0
#L2 class intercepts
mcb <- matrix(data=mcbi,H,1)
mcb[H,1] <- 0
##compute probabilities of class membership
#denominator for p_cbcw for each cbcw
den_cbcw <- matrix(NA,H,K)
for (i in 1:H)
{
for (j in 1:K)
{
den_cbcw[i,j] <- sum(exp(mcw[1,1:K]+ms[i,1:K]))
}
}
#prob of cw given cb
prob_cwgivencb <- matrix(NA,H,K)
for (i in 1:H)
{
for (j in 1:K)
{
prob_cwgivencb[i,j] <- exp(mcw[1,j]+ms[i,j]) / den_cbcw[i,j]
}
}
#marginal prob of cb
prob_cb <- matrix(NA,H,1)
for (i in 1:H)
{
prob_cb[i,1] <- exp(mcb[i,1])/ sum(exp(mcb[1:H,1]))
}
#class combination prob
prob_cbcw <- matrix(NA,H,K)
for (i in 1:H)
{
for (j in 1:K)
{
```

# ONLINE APPENDIX

```
prob_cbcw[i,j] <- prob_cb[i,1]*prob_cwgivencb[i,j]
}
}
##compute marginal L2 class parameters
margL2params <- array(3,c(H,1,1+nL1vars+nL2vars))
for (i in 1:H)
{
for (j in 1:c(1+nL1vars+nL2vars))
{
margL2params[i,1,j] <- sum(prob_cwgivencb[i,1:K]*int_slopes[i,1:K,j])
}
}
##compute fixed effects
fixedeffects <- matrix(NA,1+nL1vars+nL2vars,1)
for (i in 1:c(1+nL1vars+nL2vars))
{
fixedeffects[i,1] <- sum(prob_cb*margL2params[,,i])
}
##compute implied tau matrix
impliedtau <- matrix(NA,1+nL1vars+nL2vars,1+nL1vars+nL2vars)
for (i in 1:c(1+nL1vars+nL2vars))
{
for (j in 1:c(1+nL1vars+nL2vars))
{
impliedtau[i,j] <- sum(prob_cb*margL2params[,1,i]*margL2params[,1,j])-
sum(prob_cb*margL2params[,1,i])*sum(prob_cb*margL2params[,1,j])
}
}
##implied sigma equation
sigmamatrix <- matrix(NA,1+nL1vars+nL2vars,1+nL1vars+nL2vars)
for (i in 1:c(1+nL1vars+nL2vars))
{
for (j in 1:c(1+nL1vars+nL2vars))
{
sigmamatrix[i,j] <- sum(prob_cbcw*((int_slopes[,,i]-margL2params[,,i])*
(int_slopes[,,j]-margL2params[,,j])))
}
}
sigmamatrix[1,1] <- sigmamatrix[1,1]+sum(prob_cbcw*residual_var)
approx.ubertable.fixed[,,as.numeric(paste(c(z,v),collapse=""))] <-
fixedeffects
approx.ubertable.tau[,,as.numeric(paste(c(z,v),collapse=""))] <-
impliedtau
approx.ubertable.sigma[,,as.numeric(paste(c(z,v),collapse=""))] <-
sigmamatrix
#fixedeffects.output[[as.numeric(paste(c(z,v),collapse=""))]] <- rbind2(paste("# L1 classes = ",K,"; # L2 classes =
",H),fixedeffects)
#tau.output[[as.numeric(paste(c(z,v),collapse=""))]] <- list(paste("# L1 classes = ",K,"; # L2 classes =
",H),impliedtau)
#sigma.output[[as.numeric(paste(c(z,v),collapse=""))]] <- list(paste("# L1 classes = ",K,"; # L2 classes =
",H),sigmamatrix)
#names(fixedeffects.output[[as.numeric(paste(c(z,v),collapse=""))]]) <- paste("# L1 classes = ",K,"; # L2 classes =
",H)
```

# ONLINE APPENDIX

```
fixedeffects.output[[as.numeric(paste(c(z,v),collapse=""))]] <- matrix(c(paste("K = ",K,"; H =
",H),fixedeffects),2+nL1vars+nL2vars,1)
rownames(fixedeffects.output[[as.numeric(paste(c(z,v),collapse=""))]]) <-
c("",seq(from=1,to=1+nL1vars+nL2vars,by=1))
colnames(fixedeffects.output[[as.numeric(paste(c(z,v),collapse=""))]]) <- ""
blank <- replicate(nL1vars+nL2vars,"")
headers <- seq(from=1,to=1+nL1vars+nL2vars,by=1)
tau.output[[as.numeric(paste(c(z,v),collapse=""))]] <- matrix(c(paste("K = ",K,"; H =
",H),blank,headers,impliedtau),3+nL1vars+nL2vars,1+nL1vars+nL2vars,byrow=TRUE)
rownames(tau.output[[as.numeric(paste(c(z,v),collapse=""))]]) <-
c("","",seq(from=1,to=1+nL1vars+nL2vars,by=1))
colnames(tau.output[[as.numeric(paste(c(z,v),collapse=""))]]) <- replicate(1+nL1vars+nL2vars,"")
sigma.output[[as.numeric(paste(c(z,v),collapse=""))]] <- matrix(c(paste("K = ",K,"; H =
",H),blank,headers,sigmamatrix),3+nL1vars+nL2vars,1+nL1vars+nL2vars,byrow=TRUE)
rownames(sigma.output[[as.numeric(paste(c(z,v),collapse=""))]]) <-
c("","",seq(from=1,to=1+nL1vars+nL2vars,by=1))
colnames(sigma.output[[as.numeric(paste(c(z,v),collapse=""))]]) <- replicate(1+nL1vars+nL2vars,"")
}
}
##plots
if (plotoutput != FALSE)
{
pdf(file = paste(plotoutput,".pdf"))
classlist <- as.numeric(paste(c(Hmin,Kmin),collapse=""))
if (Hmax > Hmin)
{
for (j in c(Hmin+1):Hmax)
{
classlist <- c(classlist,as.numeric(paste(c(j,Kmin),collapse="")))
}
}
##fixed effects plots
for (i in 1:c(1+nL1vars+nL2vars))
{
plot(y=as.numeric(approx.ubertable.fixed[i,,c(classlist)]),x=c(Hmin:Hmax),type="b",xlim=c(Hmin,Hmax),ylim=c(
min(as.numeric(approx.ubertable.fixed[i,,]),na.rm=TRUE),max(as.numeric(approx.ubertable.fixed[i,,]),na.rm=TR
UE)),lwd=3,col=Kmin,xlab=expression(italic("H")),ylab=paste("Implied fixed effect ",i))
classlist_temp <- classlist
if (Kmax > Kmin)
{
for (l in c(Kmin+1):Kmax)
{
classlist_temp <- classlist_temp+1
lines(y=as.numeric(approx.ubertable.fixed[i,,classlist_temp]),x=c(Hmin:Hmax),type="b",col=l,lty=1,lwd=3)
}
legend("topright", legend = paste("K =",Kmin:Kmax), col=c(Kmin:Kmax), lty=c(1:1),lwd=3,seg.len=4)
}
}
##tau plots
for (i in 1:c(1+nL1vars+nL2vars))
{
for (j in 1:c(1+nL1vars+nL2vars))
{
```

# ONLINE APPENDIX

```
plot(y=as.numeric(approx.ubertable.tau[i,j,c(classlist)]),x=c(Hmin:Hmax),type="b",ylim=c(min(as.numeric(approx.ubertable.tau[i,j,]),na.rm=TRUE),max(as.numeric(approx.ubertable.tau[i,j,]),na.rm=TRUE)),xlim=c(Hmin,Hmax),lwd=3,col=Kmin,xlab=expression(italic("H")),ylab=paste("Implied tau ",i,j))
classlist_temp <- classlist
if (Kmax > Kmin)
{
for (l in c(Kmin+1):Kmax)
{
classlist_temp <- classlist_temp+1
lines(y=as.numeric(approx.ubertable.tau[i,j,classlist_temp]),x=c(Hmin:Hmax),type="b",col=l,lty=1,lwd=3)
}
legend("topright", legend = paste("K =",Kmin:Kmax), col=c(Kmin:Kmax), lty=c(1:1),lwd=3,seg.len=4)
}
}
}
##sigma plots
classlist_sigma <- as.numeric(paste(c(Hmin,Kmin),collapse=""))
if (Kmax > Kmin)
{
for (j in c(Kmin+1):Kmax)
{
classlist_sigma <- c(classlist_sigma,as.numeric(paste(c(Hmin,j),collapse="")))
}
}
for (i in 1:c(1+nL1vars+nL2vars))
{
for (j in 1:c(1+nL1vars+nL2vars))
{
plot(y=as.numeric(approx.ubertable.sigma[i,j,c(classlist_sigma)]),x=c(Kmin:Kmax),type="b",ylim=c(min(as.numeric(approx.ubertable.sigma[i,j,]),na.rm=TRUE),max(as.numeric(approx.ubertable.sigma[i,j,]),na.rm=TRUE)),xlim=c(Kmin,Kmax),lwd=3,col=Kmin,xlab=expression(italic("K")),ylab=paste("Implied sigma ",i,j))
classlist_sigma_temp <- classlist_sigma
if (Hmax > Hmin)
{
for (l in c(Hmin+1):Hmax)
{
classlist_sigma_temp <- classlist_sigma_temp+10
lines(y=as.numeric(approx.ubertable.sigma[i,j,classlist_sigma_temp]),x=c(Kmin:Kmax),type="b",col=l,lty=1,lwd=3)
}
legend("topright", legend = paste("H =",Hmin:Hmax), col=c(Hmin:Hmax), lty=c(1:1),lwd=3,seg.len=4)
}
}
}
dev.off()
}
##Output
fixedeffects.output <- Filter(Negate(is.null), fixedeffects.output)
tau.output <- Filter(Negate(is.null), tau.output)
sigma.output <- Filter(Negate(is.null), sigma.output)
Output <- c(noquote(fixedeffects.output),noquote(tau.output),noquote(sigma.output))
```

# ONLINE APPENDIX

```
names(Output) <- c(replicate(length(fixedeffects.output),'Implied Fixed
Effects'),replicate(length(fixedeffects.output),"Implied Tau Matrix"),replicate(length(tau.output),"Implied Sigma
Matrix"))
return(Output)
}
```