Supplementary material: Annotated JAGS code for the two empirical applications

1. A two-state OMM for daily negative affect

```
model{
for (i in 1:N)
{ # loop over the N persons in the sample

  for (t in 1:T)
  {  # loop over the T timepoints / occasions
      m[t,i] ~ dcat(ps[i,t,])
     # the categorical likelihood for the state m
     # of person i at time t.
     # ps[i,t,] is an M-length vector with the probabilities
     # for being in each of the M possible states.
  }  # end loop over T timepoints

  for (state in 1:M)
  {  # loop over all the possible states
      ps[i,1,state] <- probs1[state]
     # The probability of being in this state at occasion 1.
     # (This way, missing data at t=1 can be imputed.)
     for (t in 2:T)
     {  # loop over all timepoints except the first one
         ps[i,t,state] <- probs[m[(t-1),i], state, i]
        # choose the probability vector for the current state
        # that corresponds to the observed previous state.
     }  # end loop over timepoints

     for (prev in 1:M)
     {  # loop over the possible states at the previous occasion
         probs[prev,state,i] <- odds[prev,state,i]/totalodds[prev,i]
        # calculate the probabilities from the underlying odds
         odds[prev,state,i] <- exp(alpha[prev,state,i] +
                               beta[prev,state]*(scaled.neuro[i]))
        # calculate the odds from the random logit intercepts
        # and the regression coefficients of neuroticism.
     }  # end loop over possible previous states

      alpha[state,state,i] <- 0
     # identify the model by fixing "stability" logits at 0
      totalodds[state,i] <- odds[state,1,i] + odds[state,2,i]
     # a step in the transformation of odds to probabilities.
     # note that this line is specific to a 2-state model.
  } # end loop over possible states

   scaled.neuro[i] <- (neuro[i]-mean(neuro[]))/(2*sd(neuro[]))
  # center and rescale each person's neuroticism score
```

```
    alpha[1,2,i] <- raneffs[i,1]
    alpha[2,1,i] <- raneffs[i,2]
    raneffs[i,1:2] ~ dmnorm(alpha.means[1:2],alpha.prec[1:2, 1:2])
  # normal distribution for the random transition logit intercepts
} # end of loop over persons

for (state in 1:M)
{ # loop over all the possible states
    beta[state,state] <- 0
  # identify the model by fixing "stability" logits at 0
} # end of loop over possible states

  beta[1,2] <- rcoeff[1]
  beta[2,1] <- rcoeff[2]
# collect the non-diagonal elements in a vector for convenience
  alpha.Sigma[1:2,1:2] <- inverse(alpha.prec[1:2,1:2])
# We invert this because we prefer the covariance matrix.

# Below are the prior specifications:
  probs1[1:M] ~ ddirich(statealpha[1:M])
# a Dirichlet prior for the initial state probability vector.
# statealpha is a vector of ones, given as data input
  for (par in 1:2)
  { # a loop over the number of parameters per type
    alpha.means[par] ~ dt(T.constant.mu,T.constant.tau,T.constant.k)
    rcoeff[par] ~ dt(T.mu, T.tau, T.k)
  # Student's t priors for the two average logit intercepts
  # and the two regression coefficients of neuroticism.
  # Note that JAGS uses the inverse scaling parameter
  # notation for this distribution.
  } # end of loop over parameters

  T.constant.mu <- 0
  T.mu <- 0
# location parameters for the priors
  T.constant.tau <- 1/T.constant.scale.squared
  T.tau <- 1/T.scale.squared
# inverse scale parameters for the priors
  T.constant.scale.squared <- T.constant.scale * T.constant.scale
  T.scale.squared <- T.scale * T.scale
  T.scale <- 2.5
  T.constant.scale <- 10
# the actual scale parameters as mentioned in the paper
  T.constant.k <- 1
  T.k <- 1
# 1 degree of freedom; this results in a Cauchy density
  alpha.prec[1:2,1:2] ~ dwish(Om[1:2,1:2],2)
# a Wishart prior for the precision matrix, i.e.,
# an inverse-Wishart prior for the covariance matrix.
```

```
  Om[1,1] <- 1
  Om[1,2] <- 0
  Om[2,1] <- 0
  Om[2,2] <- 1
# hyperparameters for the Wishart prior.

} # end of model file
```

2. A three-state LMM for family interactions with random effects in both model parts

```
model{

# First the conditional model part for the observed behavior:

for (t in 1:T)
{ # loop over timepoints / occasions 1 to T
  for (i in 1:N)
  { # loop over families 1 to N
      Mo[t,i] ~ dcat(p.Mo[m[t,i],1:4])
    # the categorical likelihood for the mother's behavior
      Fa[t,i] ~ dcat(p.Fa[m[t,i],1:4])
      Ad[t,i] ~ dcat(p.Ad[m[t,i],i,1:4])
     # the categorical likelihood for the adolescent's behavior
     # Note that p.Ad is person-specific, whereas the
     # probabilities p.Mo and p.FA are not (they are fixed).
  } # end of loop over all families

} # end of loop over timepoints 1:T

for (i in 1:N)
{ # loop over all families
  for (state in 1:M)
  { # loop over all possible latent family states
      for (beh in 1:4)
      { # loop over all possible observed behavior categories
            p.Ad[state,i,beh] <- probs.Ad[state,beh,i]
          # reshaping the array for ~dcat argument requirements
            probs.Ad[state,beh,i] <- odds.Ad[state,beh,i] /
                                     totalodds.Ad[state,i]
          # calculate probabilities from underlying odds
            odds.Ad[state,beh,i] <- exp(c[state,beh,i] +
                                    dcoeff[beh]*scaled.depr[i])
          # calculate the odds from the random logit intercepts
          # and the regression coefficients of depression.
            c[state,beh,i] <- state.intercept[state,beh] +
                              ind.deviation[i,beh]
          # each random logit intercept is comprised of a
          # fixed part and a person-level deviation.
      } # end loop over observed behavior categories
```

```
      totalodds.Ad[state,i] <- odds.Ad[state,1,i]+
      odds.Ad[state,2,i]+odds.Ad[state,3,i]+odds.Ad[state,4,i]
    # a step in the transformation of odds to probabilities.
    # Note that this line is specific to a 4-category outcome.
  } # end loop over latent family states

    ind.deviation[i,1:3] ~ dmnorm(devs.means[1:3],devs.prec[1:3,1:3])
  # A normal distribution for the individual deviations in the
  # adolescents' logit intercepts for the observed behaviors.
    scaled.depr[i] <- depr[i] - mean(depr[])
  # center the depression predictor
    ind.deviation[i,4] <- 0
  # identify the model by fixing "neutral behavior" logits at 0
} # end loop over all families

for (beh in 1:3)
{ # loop over the 3 categories with parameters to-be-estimated
    devs.means[beh] <- 0
  # fix the mean of the individual deviations at 0 so that each
  # fixed effect (state.intercept) will represent the "average".
    dcoeff[beh] ~ dt(T.mu,T.tau,T.k)
  # Student's t prior for the regression coeffs. of depression.
  # Note that JAGS uses the inverse scaling parameter
  # notation for this distribution.
} # end loop over behavior categories 1-3

  dcoeff[4] <- 0
# identify the model by fixing "neutral behavior" logits at 0
  T.mu <- 0
# location parameter for the prior
  T.tau <- 1/T.scale.squared
# inverse scale parameter for the prior
  T.scale.squared <- T.scale * T.scale
  T.scale <- 2.5
# the actual scale parameter as mentioned in the paper
  T.k <- 1
# 1 degree of freedom; this results in a Cauchy density.
  devs.prec[1:3,1:3] ~ dwish(Om[1:3,1:3],3)
# a Wishart prior for the precision matrix, i.e.,
# an inverse-Wishart prior for the covariance matrix.
  Om[3,3] <- 1
# Om is a diagonal matrix with the hyperparameters
  for (om.i in 1:2)
  { # loop over rows 1-2 in the Om matrix
      Om[om.i,om.i] <- 1
      for (om.j in (om.i+1):3)
      { # loop over column 2-3 in the Om matrix
          Om[om.i,om.j] <- 0
```

```
        Om[om.j,om.i] <- 0
      } # end loop over columns in the Om matrix

    } # end loop over rows in the Om matrix

  # the loops are a shorthand for "filling" the diagonal matrix.
    devs.Sigma[1:3,1:3] <- inverse(devs.prec[1:3,1:3])
  # We invert this because we prefer the covariance matrix.

  for (state in 1:M)
  { # loop over the latent family states
      for (beh in 1:3)
      { # loop over the 3 categories with pars to be estimated
          state.intercept[state,beh] ~ dt(T.constant.mu,
                                  T.constant.tau, T.constant.k)
        # Prior for the fixed part of the logit intercepts.
      } # end of loop over behavior categories 1-3

        state.intercept[state,4] <- 0
      # identify the model by fixing "neutral behavior" logits at 0
        p.Mo[state,1:4] ~ ddirich(cond.alphas[1:4])
        p.Fa[state,1:4] ~ ddirich(cond.alphas[1:4])
      # Dirichlet priors for the parents’ behavior probabilities.
      # cond.alphas is a vector of ones given as data input.
  } # end loop over the latent family states

    T.constant.mu <- 0
  # location parameter for the prior
    T.constant.tau <- 1/T.constant.scale.squared
  # inverse scale parameter for the prior
    T.constant.scale.squared <- T.constant.scale * T.constant.scale
    T.constant.scale <- 10
  # the actual scale parameter as mentioned in the paper
    T.constant.k <- 1
  # 1 degree of freedom; this results in a Cauchy density.

  # Now the transition model part for the latent family states:

  for (i in 1:N)
  { # loop over all families
    for (t in 1:T)
    { # loop over all timepoints
        m[t,i] ~ dcat(ps[i,t,1:M])
      # the categorical likelihood for the state m
      # of family i at time t.
      # ps[i,t,] is an M-length vector with the probabilities
      # for being in each of the M possible states.
    } # end loop over timepoints
```

```
  for (state in 1:M)
  { # loop over all possible latent family states
      ps[i,1,state] <- probs1[state]
     # The probability of being in this state at occasion 1.
     # (Not dependent on the state at the previous occasion).
      for (t in 2:T)
      { # loop over all timepoints except the first one
          ps[i,t,state] <- probs[m[(t-1),i],state,i]
        # Choose the probability vector for the current state
        # that corresponds to the observed previous state.
      } # end loop over timepoints

      for (prev in 1:M)
      { # loop over the possible states at the previous occasion
          probs[prev,state,i]<- odds[prev,state,i]/totalodds[prev,i]
        # calulate the probabilities from the underlying odds
          odds[prev,state,i] <- exp(v[prev,state,i])
        # calculate the odds from the random logits.
      } # end loop over possible previous states

      v[state,state,i] <- 0
    # identify the model by fixing "stability" logits at 0
      totalodds[state,i]<- odds[state,1,i]+odds[state,2,i]+
                           odds[state,3,i]
    # a step in the transformation of odds to probabilities.
    # Note that this line is specific to a 3-state model.
  } # end loop over possible states
  v[1,2,i] <- raneffs[i,1]
  v[1,3,i] <- raneffs[i,2]
  v[2,1,i] <- raneffs[i,3]
  v[2,3,i] <- raneffs[i,4]
  v[3,1,i] <- raneffs[i,5]
  v[3,2,i] <- raneffs[i,6]
  raneffs[i,1:6] ~ dmnorm(v.means[1:6], v.prec[1:6,1:6])
# normal distribution for the random transition logits
} # end of loop over persons

  probs1[1:M] ~ ddirich(start.alphas[1:M])
# a Dirichlet prior for the initial state probability vector.
  vmeans[1,2] <- v.means[1]
  vmeans[1,3] <- v.means[2]
  vmeans[2,1] <- v.means[3]
  vmeans[2,3] <- v.means[4]
  vmeans[3,1] <- v.means[5]
  vmeans[3,2] <- v.means[6]
for (row in 1:M)
{ # loop over rows for latent states
    vmeans[row,row] <- 0
  # identify the model by fixing "stability" logits at 0
```

```
} # end loop over rows for latent states

  v.prec[1:6,1:6] ~ dwish(v.Om[1:6,1:6],6)
# a Wishart prior for the precision matrix, i.e.,
# an inverse-Wishart prior for the covariance matrix.
  v.Om[6,6] <- 1
# v.Om is a diagonal matrix with the hyperparameters
  for (vom.i in 1:5)
  { # loop over rows 1-5 in the v.Om matrix
      v.Om[vom.i,vom.i] <- 1
      for (vom.j in (vom.i+1):6)
      { # loop over columns 2-6 in the v.Om matrix
          v.Om[vom.i,vom.j] <- 0
          v.Om[vom.j,vom.i] <- 0
      } # end loop over columns of v.Om

  } # end loop over rows of V.Om

# the loops are a shorthand for "filling" the diagonal matrix.
v.Sigma[1:6,1:6] <- inverse(v.prec[1:6,1:6])
# We invert this because we prefer the covariance matrix.

for (par in 1:6)
{ # loop over the number of fixed logit parameters
     v.means[par]~dt(T.constant.mu,T.constant.tau,T.constant.k)
 # Student’s t prior for the average transition logits,
# with the same hyperparameters used and specified above.
} # end loop over number of parameters

} # end of model file
```