Supplementary

Supplementary 1 – Multivariate Multilevel Autoregressive (MMAR) model

We model future network measures (t+1) as a function of its current value (t) and the current value of all other covariates. The estimated effect of each covariate (β) is treated as a random slope using individual identity (i) as the grouping variable.

$$\mu_{inDegree,i,t+1} = \beta_{1,i} * inDegree_{i,t} + \beta_{2,i} * outDegree_{i,t} + \beta_{3,i} * MaleC_{i,t} + \beta_{4,i} * maleFC_{i,t} + \beta_{5,i} * rank_{i,t} + \beta_{6,i} * inAgg_{i,t}$$

$$\mu_{outDegree,i,t+1} = \beta_{7,i} * inDegree_{i,t} + \beta_{8,i} * outDegree_{i,t} + \beta_{9,i} * MaleC_{i,t} + \beta_{10,i} * maleFC_{i,t} + \beta_{11,i} * rank_{i,t} + \beta_{12,i} * inAgg_{i,t}$$

$$\mu_{maleC,i,t+1} = \beta_{13,i} * inDegree_{i,t} + \beta_{14,i} * outDegree_{i,t} + \beta_{15,i} * MaleC_{i,t} + \beta_{16,i} * maleFC_{i,t} + \beta_{17,i} * rank_{i,t} + \beta_{18,i} * inAgg_{i,t}$$

$$\mu_{maleFC,i,t+1} = \beta_{19,i} * inDegree_{i,t} + \beta_{20,i} * outDegree_{i,t} + \beta_{21,i} * MaleC_{i,t} + \beta_{22,i} * maleFC_{i,t} + \beta_{23,i} * rank_{i,t} + \beta_{24,i} * inAgg_{i,t}$$

$$\mu_{rank,i,t+1} = \beta_{25,i} * inDegree_{i,t} + \beta_{26,i} * outDegree_{i,t} + \beta_{27,i} * MaleC_{i,t} + \beta_{28,i} * maleFC_{i,t} + \beta_{29,i} * rank_{i,t} + \beta_{30,i} * inAgg_{i,t}$$

$$\mu_{inAgg,i,t+1} = \beta_{31,i} * inDegree_{i,t} + \beta_{32,i} * outDegree_{i,t} + \beta_{33,i} * MaleC_{i,t} + \beta_{34,i} * maleFC_{i,t} + \beta_{35,i} * rank_{i,t} + \beta_{36,i} * inAgg_{i,t}$$

Correlations between random slopes are estimated by using a multivariate normal distribution and calculating the covariance between individual slope differences ($\Sigma_{individual}$). This multivariate normal distribution is used to estimate the effects (slopes) of each effect j for each individual i, pooling data from all individuals. All data is scaled and centered within each individual, and slopes are given weakly informative priors centered on zero. This data scaling and prior choice starts the model off with the assumption that each slope is most likely zero.

$$\beta_{j,i} = Mulivariate\ normal(\beta_{\mu}, \Sigma_{individual})$$

Finally, the likelihood of observed measures (k) are calculated by using a multivariate normal distribution of the mean predicted network measures and a covariance matrix describing the dependence in the errors between network predictions (Σ_{sigma}).

$$Y_{k,t+1} = Mulivariate\ normal(\mu_{k,t+1}, \Sigma_{sigma})$$

To fit this model we use the cholesky parameterization (see supp. 2).

Supplementary 2 – model code for the MMAR

```
data {
  int<lower=0> T; // No. of observations
  int<lower=0> M; // No. of variables in y_t
  int<lower=0> L; // No. of control variables
  vector[M] Y[T]; // Obs of state variables
  vector[M] Y_prev[T]; //Obs of state variables lagged
  int<lower=0> ID[T]; //Obs of individual id
  int<lower=1> N; //No. of unique individual ids
}

parameters {
  cholesky_factor_corr[M] L_corr_noise;
  vector<lower=0>[M] sd_noise;
  cholesky_factor_corr[36] L_corr_ind;
  vector<lower=0>[36] sd_ind;

  vector[36] Amu;
  vector[36] A[N];

}

transformed parameters {
  matrix[M,M] L_sigma;
  matrix[36,36] L_ind;
  L_sigma = diag_pre_multiply(sd_noise, L_corr_noise);
  L_ind   = diag_pre_multiply(sd_ind, L_corr_ind);
}

model {
  vector[M] mus[T];
  for (t in 1:T) {
    mus[t,1] = A[ID[t],1]  * Y_prev[t,1] + A[ID[t],2] * Y_prev[t,2]  + A[ID[t],3] * Y_prev[t,3] + A[ID[t],4] * Y_prev[t,4] + A[ID[t],5] *
Y_prev[t,5] + A[ID[t],35] * Y_prev[t,6]; //in degree
    mus[t,2] = A[ID[t],6]  * Y_prev[t,2] + A[ID[t],7] * Y_prev[t,1]  + A[ID[t],8] * Y_prev[t,3] + A[ID[t],9] * Y_prev[t,4] + A[ID[t],10] *
Y_prev[t,5] + A[ID[t],34] * Y_prev[t,6]; //out degree
    mus[t,3] = A[ID[t],11] * Y_prev[t,3] + A[ID[t],12] * Y_prev[t,1]  + A[ID[t],13] * Y_prev[t,2] + A[ID[t],14] * Y_prev[t,4] + A[ID[t],15] *
Y_prev[t,5] + A[ID[t],33] * Y_prev[t,6]; //Male C
```

```
   mus[t,4] = A[ID[t],16] * Y_prev[t,4] + A[ID[t],17] * Y_prev[t,1] + A[ID[t],18] * Y_prev[t,3]+ A[ID[t],19] * Y_prev[t,2] + A[ID[t],20] *
Y_prev[t,5] + A[ID[t],32] * Y_prev[t,6]; //Male FC

   mus[t,5] = A[ID[t],21] * Y_prev[t,5] + A[ID[t],22] * Y_prev[t,1] + A[ID[t],23] * Y_prev[t,3]+ A[ID[t],24] * Y_prev[t,2] + A[ID[t],25] *
Y_prev[t,4] + A[ID[t],31] * Y_prev[t,6];  //Rank

   mus[t,6] = A[ID[t],26] * Y_prev[t,6] + A[ID[t],27] * Y_prev[t,1] + A[ID[t],28] * Y_prev[t,3]+ A[ID[t],29] * Y_prev[t,2] + A[ID[t],30] *
Y_prev[t,4] + A[ID[t],36] * Y_prev[t,5]; //aggression in (males)

  }


  L_corr_noise ~ lkj_corr_cholesky(1.0);

  sd_noise ~ normal(0,1);

  L_corr_ind ~ lkj_corr_cholesky(1.0);

  sd_ind ~ normal(0,1);


  Amu ~ normal(0,1);

  A ~ multi_normal_cholesky(Amu,L_ind);


  Y ~ multi_normal_cholesky(mus,L_sigma);

}


generated quantities {

  matrix[M,M] Corr_sigma;

  matrix[36,36] Corr_ind;

  vector[M] mus[T];

  vector[M] Y_rep[T];


  //get correlation matrix

  Corr_sigma = L_corr_noise * L_corr_noise';

  Corr_ind = L_corr_ind * L_corr_ind';


  //get predicted values

  for (t in 1:T) {

   mus[t,1] = A[ID[t],1]  * Y_prev[t,1] + A[ID[t],2] * Y_prev[t,2]  + A[ID[t],3] * Y_prev[t,3] + A[ID[t],4] * Y_prev[t,4] + A[ID[t],5] *
Y_prev[t,5] + A[ID[t],35] * Y_prev[t,6]; //in degree

   mus[t,2] = A[ID[t],6]  * Y_prev[t,2] + A[ID[t],7] * Y_prev[t,1]  + A[ID[t],8] * Y_prev[t,3] + A[ID[t],9] * Y_prev[t,4] + A[ID[t],10] *
Y_prev[t,5] + A[ID[t],34] * Y_prev[t,6]; //out degree

   mus[t,3] = A[ID[t],11] * Y_prev[t,3] + A[ID[t],12] * Y_prev[t,1]  + A[ID[t],13] * Y_prev[t,2] + A[ID[t],14] * Y_prev[t,4] + A[ID[t],15] *
Y_prev[t,5] + A[ID[t],33] * Y_prev[t,6];  //Male C

   mus[t,4] = A[ID[t],16] * Y_prev[t,4] + A[ID[t],17] * Y_prev[t,1] + A[ID[t],18] * Y_prev[t,3]+ A[ID[t],19] * Y_prev[t,2] + A[ID[t],20] *
Y_prev[t,5] + A[ID[t],32] * Y_prev[t,6];  //Male FC

   mus[t,5] = A[ID[t],21] * Y_prev[t,5] + A[ID[t],22] * Y_prev[t,1] + A[ID[t],23] * Y_prev[t,3]+ A[ID[t],24] * Y_prev[t,2] + A[ID[t],25] *
Y_prev[t,4] + A[ID[t],31] * Y_prev[t,6];  //Rank
```

```
   mus[t,6] = A[ID[t],26] * Y_prev[t,6] + A[ID[t],27] * Y_prev[t,1] + A[ID[t],28] * Y_prev[t,3]+ A[ID[t],29] * Y_prev[t,2] + A[ID[t],30] *
Y_prev[t,4] + A[ID[t],36] * Y_prev[t,5];  //aggression in (males)

  }

  Y_rep = multi_normal_cholesky_rng(mus,L_sigma);


}
```

Supplementary 2 – Figures

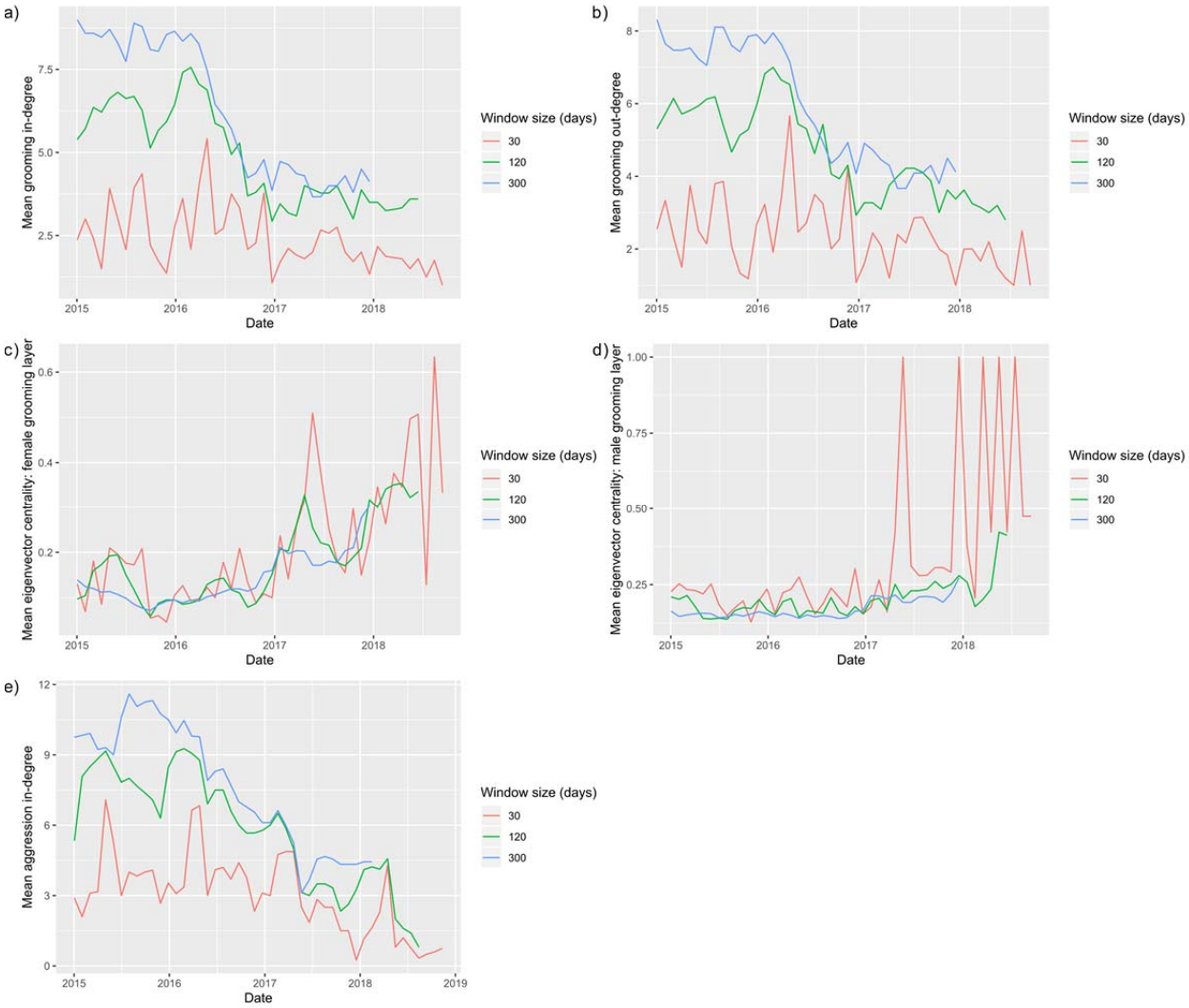

Figure S1: Visual inspection of the change in time series generated when using alternative window size choices: a) mean male grooming in-degree from females, b) mean male grooming out-degree to females, c) mean eigenvector centrality of males within the female grooming layer, d) mean eigenvector centrality of males within the male grooming layer, and e) mean male aggression in-degree from other males.

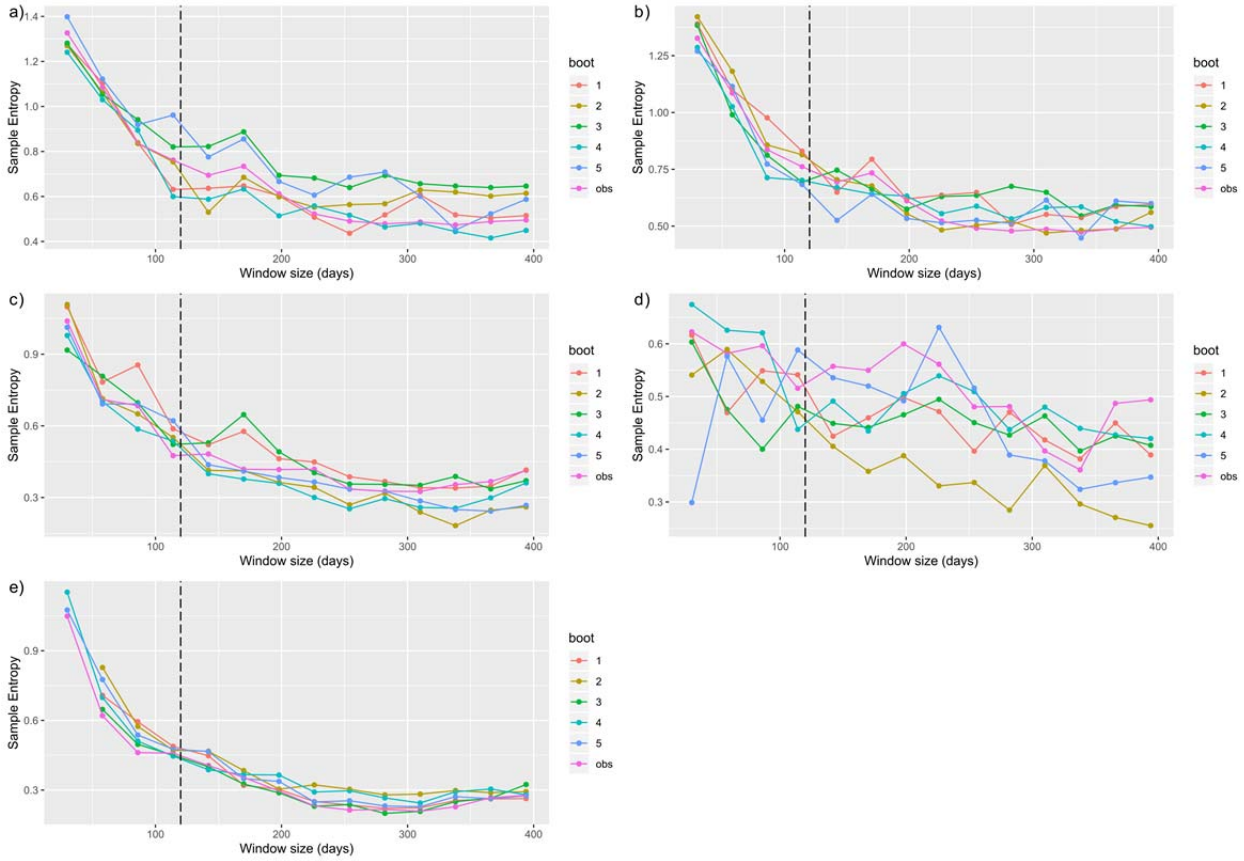

Figure S2: Sample entropy from each time series generated using window sizes ranging from 30 days to 400 days: a) mean male grooming in-degree from females, b) mean male grooming out-degree to females, c) mean eigenvector centrality of males within the female grooming layer, d) mean eigenvector centrality of males within the male grooming layer, and e) mean male aggression in-degree from other males. For each measure 5 bootstrapped replicates are shown alongside the observed data. Linear trends in each time series were removed prior to calculating sample entropy.

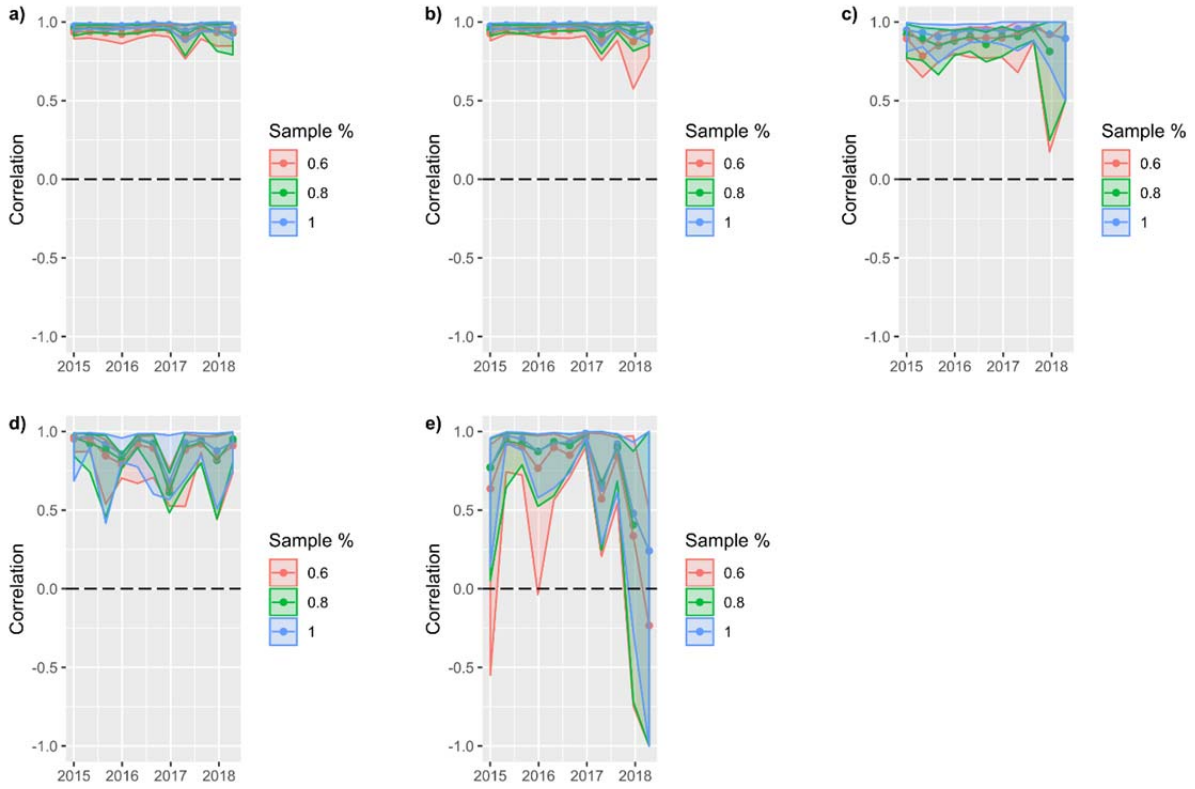

Figure S3: Results from the bootstrap test on network measures: a) grooming in-degree from females, b) grooming out-degree to females, c) aggression in-degree from males, d) eigenvector centrality of males within the female grooming layer, and e) eigenvector centrality of males within the male grooming layer. The y-axis is the correlation between the network level measures of nodes in the observed and bootstrapped networks. The lines and points represent the mean correlation, while the shaded areas represent the 95%CI calculated from 1000 bootstrapped samples. Correlations were estimated for subsamples of the data: 100%, 80%, and 60% to quantify the influence of potential missing data on network measures.

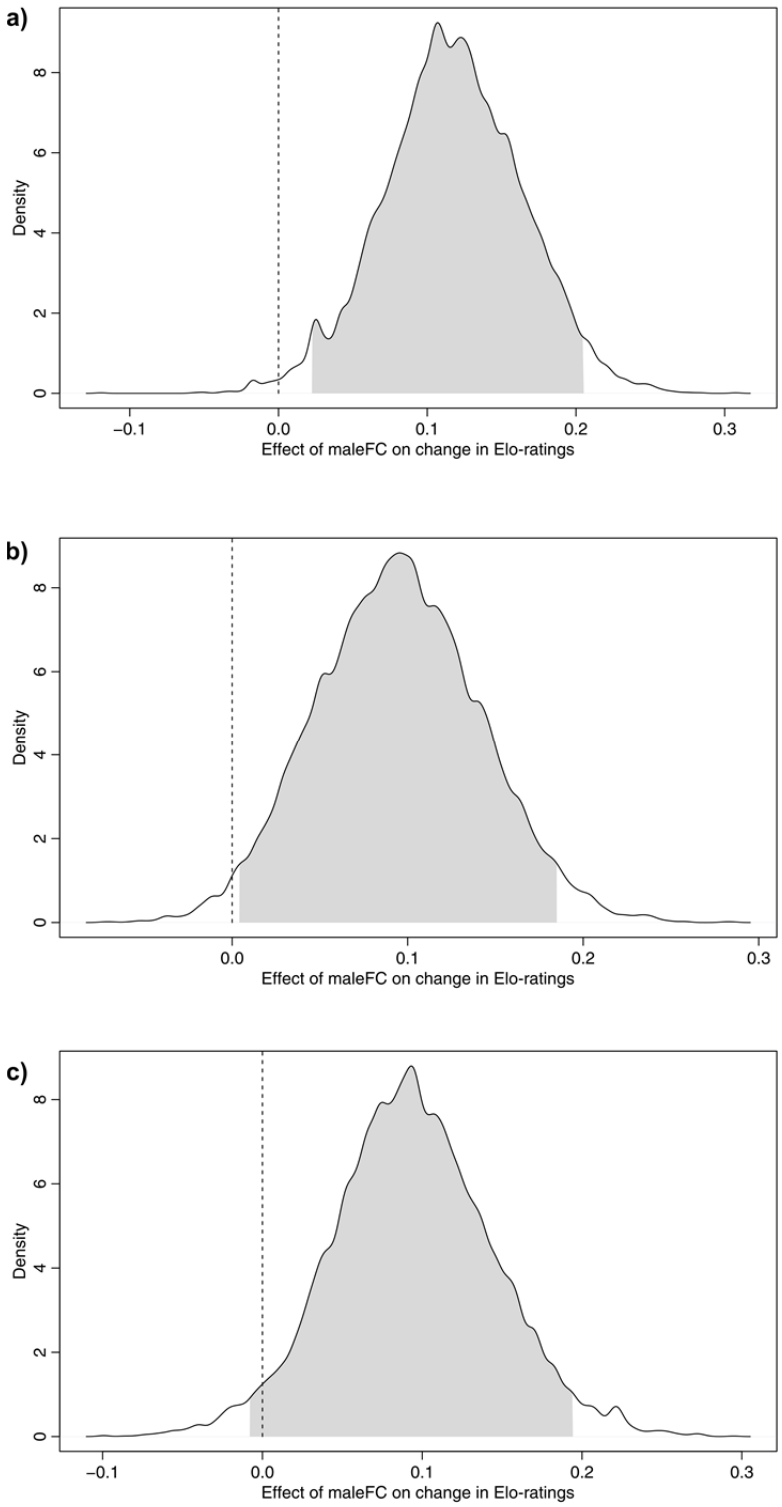

Figure S4: Estimated effect of a males centrality within the female grooming network (maleFC) on its change in Elo-ratings for a) 4 month window size, b) 5 month window size, c) 6 month window size.

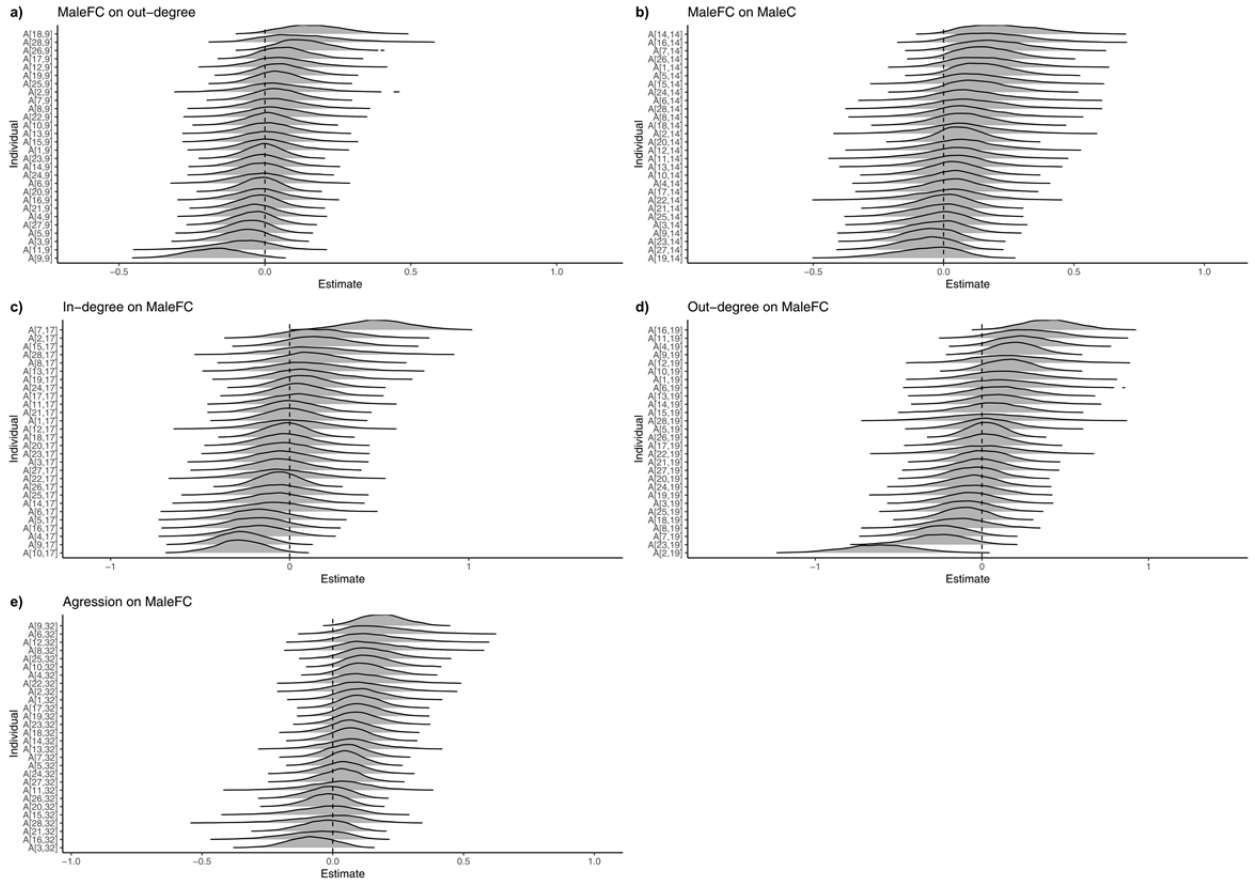

Figure S5: Individual level estimates in terms of: a) the effect of male centrality within the female grooming layer (maleFC) on out-degree, b) maleFC on male centrality within the male grooming layer (maleC), c) male grooming in-degree from females on maleFC, d) male grooming out-degree to females on maleFC, and e) male aggression in-degree on maleFC.

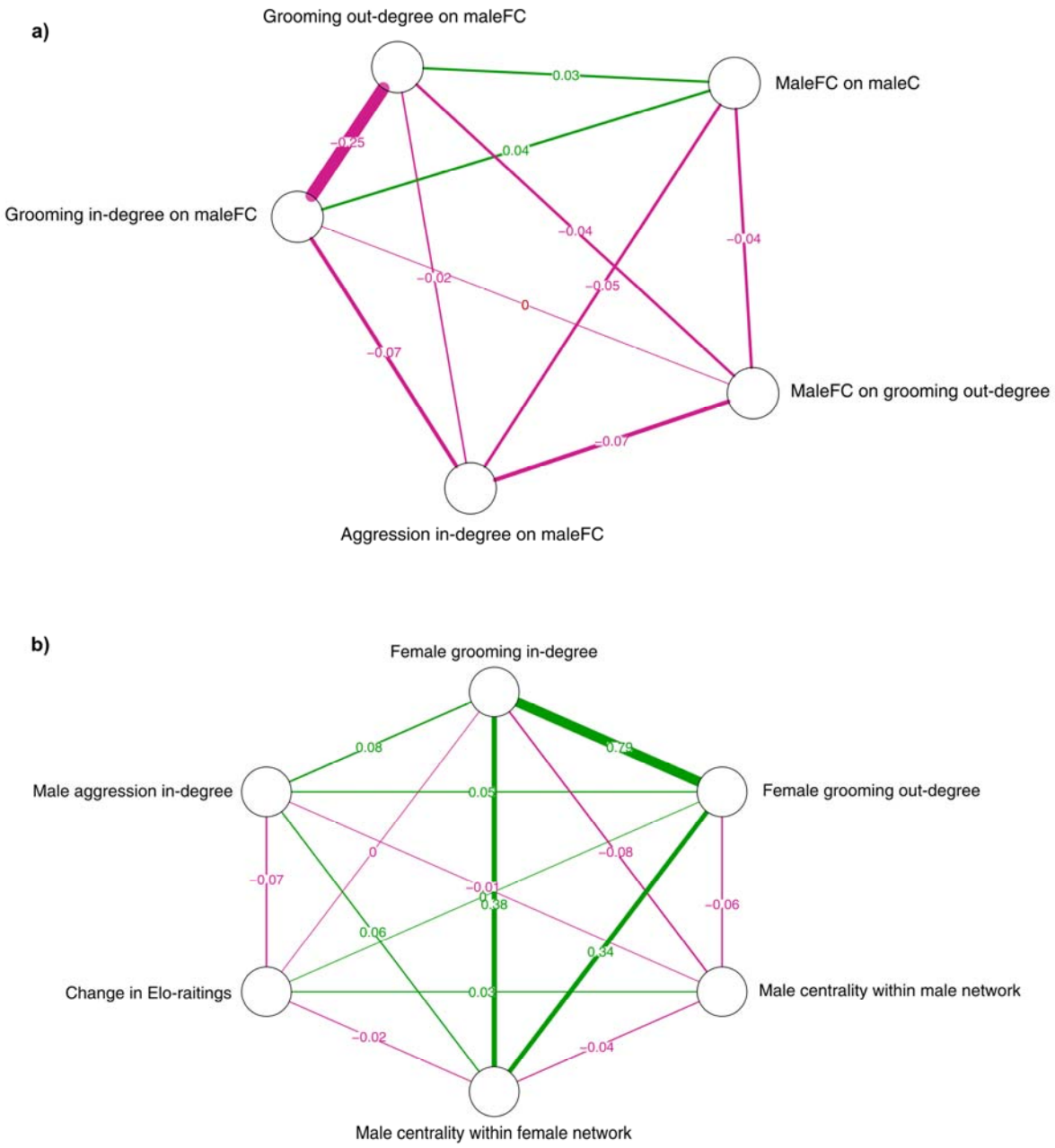

Figure S6: Estimated dependence in a) the individual level differences in lagged effects, i.e., if an individual was estimated to have a larger effect of grooming out-degree on subsequent male centrality in the female network (maleFC) were they also likely to have a lower effect of grooming in-degree on subsequent maleFC, and b) in the error of the social network measures. In b) the values associated with edges represent estimated correlation coefficients in the MMAR model between the errors of each social network measure. In a) the effects with the highest level of estimated individual level differences are presented, though all correlation estimates contain 0 when considering the 95% credible interval.

Supplementary 3 – Making predictions with the MMAR

One advantage of the multivariate multilevel autoregressive model (MMAR) approach for analyzing multilayer networks is that the dynamics of how changes in one layer influence other layers can be estimated. However, the resulting outputs from these types of models are relatively opaque. With many parameters estimates, and many interactions. One way to gain intuition about the model, and to interpret the results, is to use the fitted MMAR to make predictions about how changes in one measure cascade through the multilayered network. The steps required to do this are: 1) fit a MMAR model to data, 2) extract posterior distributions of each parameter, 3) draw from these posterior distributions to parameterize equations describing the dependence between the social network measures, 4) initialize all network measures to a starting value, and 5) use the parameterized equation to make predictions about changes in the multilayer network. For step 4, we introduced a change in one layer, and used step 5 to predict the consequences of this change through time. We used this approach to make predictions about the magnitude of perturbation caused by changing each measure. We also made these predictions at the level of the mean, i.e., μ in eq. s1, as well as predictions at the level of the outcome scale, i.e., y eq. s1, taking into account the estimated covariance matrix (Σ) from the MMAR model. We provide the code used to make these predictions below.

$$y_{i,t} \sim MVNorm(\mu_{i,t}, \Sigma) \qquad \text{Eq. s1}$$

Code:

```
sim_guitar <- function(x, post, uncertainty=FALSE, nsim=25,nreps=100, mirror =TRUE){

 #Starting point
 Y_start <- x
 Y <- matrix(NA,ncol = 1,nrow=6)

 #get parameters
 parm <-post %>% dplyr::select(contains("Amu"))
 parm_sigma <-post %>% dplyr::select(contains("L_sigma"))

 #define the number of simulated steps
 nreps = nreps
 outputs <- matrix(NA, ncol=6+2,nrow=nsim*nreps)
 outputs[1,] <- c(Y_start,1,1)

 for(j in 1:nreps){

  #choose random row from the posterior samples
  rand_row = sample(1:nrow(parm),1)

  #get covariance matrix for this run
  parm_sigma_mat <- matrix(unlist(parm_sigma[rand_row,]),ncol=6, nrow=6)
  parm_sigma_mat[upper.tri(parm_sigma_mat)] <- t(parm_sigma_mat)[upper.tri(parm_sigma_mat)]
  parm_sigma_mat<- parm_sigma_mat %*% t(parm_sigma_mat)

  #run simulation
  for(i in 1:nsim){

   if(i == 1){

    #record data
    outputs[i+((j-1)*nsim),] <- c(Y_start,1,j)

    #make the current the past
    Y_prev = Y_start
```

```
    } else {

      #update the current state
      Y[1] = parm[rand_row,1]  * Y_prev[1] + parm[rand_row,2] * Y_prev[2]  + parm[rand_row,3] * Y_prev[3] +
parm[rand_row,4] * Y_prev[4] + parm[rand_row,5] * Y_prev[5] + parm[rand_row,35] * Y_prev[6]
      Y[2] = parm[rand_row,6]  * Y_prev[2] + parm[rand_row,7] * Y_prev[1]  + parm[rand_row,8] * Y_prev[3] +
parm[rand_row,9] * Y_prev[4] + parm[rand_row,10] * Y_prev[5] + parm[rand_row,34] * Y_prev[6]
      Y[3] = parm[rand_row,11] * Y_prev[3] + parm[rand_row,12] * Y_prev[1]  + parm[rand_row,13] * Y_prev[2] +
parm[rand_row,14] * Y_prev[4] + parm[rand_row,15] * Y_prev[5] + parm[rand_row,33] * Y_prev[6]
      Y[4] = parm[rand_row,16] * Y_prev[4] + parm[rand_row,17] * Y_prev[1] + parm[rand_row,18] * Y_prev[3]+
parm[rand_row,19] * Y_prev[2] + parm[rand_row,20] * Y_prev[5] + parm[rand_row,32] * Y_prev[6]
      Y[5] = parm[rand_row,21] * Y_prev[5] + parm[rand_row,22] * Y_prev[1] + parm[rand_row,23] * Y_prev[3]+
parm[rand_row,24] * Y_prev[2] + parm[rand_row,25] * Y_prev[4] + parm[rand_row,31] * Y_prev[6]
      Y[6] = parm[rand_row,26] * Y_prev[6] + parm[rand_row,27] * Y_prev[1] + parm[rand_row,28] * Y_prev[3]+
parm[rand_row,29] * Y_prev[2] + parm[rand_row,30] * Y_prev[4] + parm[rand_row,36] * Y_prev[5]

      #make predictions for the next states
      if(uncertainty)Y <- mvrnorm(n = 1, Y, parm_sigma_mat)

      #record data
      outputs[i+((j-1)*nsim),] <- c(Y,i,j)

      #make the current the past
      Y_prev = Y

    }

   }

  }


  #simulation results
  colnames(outputs) <- c("in_deg_G","out_deg_G","cent","Fcent","elo","agg","time","rep")
  outputs.df <- as.data.frame(outputs) %>% gather(SNM,value, in_deg_G:agg)

  #get means and hpdi
  outputs.df$snm_time <- paste0(outputs.df$SNM,"_",outputs.df$time)
  if(nreps>1){
  outputs.df.summary <- outputs.df %>% group_by(snm_time) %>% summarize(mu = mean(value), hpi.l =
rethinking::HPDI(value)[1], hpi.u = rethinking::HPDI(value)[2], time=time[1],SNM=SNM[1])
  } else {
    outputs.df.summary <- outputs.df %>% group_by(snm_time) %>% summarize(mu = mean(value), hpi.l = 0, hpi.u =
0, time=time[1],SNM=SNM[1])
  }

  #whether to have the string plucked in the middle (mirrored) or have it plucked at the begginging of the plot/x-axis
(non-mirrored)
  if(mirror){
  outputs.df.before <- outputs.df.summary
  outputs.df.before$time <- -outputs.df.before$time
  outputs.df.before$mu <- 0
  outputs.df.before$hpi.l <- 0
  outputs.df.before$hpi.u <- 0
  outputs.df.summary<-rbind(outputs.df.summary,outputs.df.before)
  }

  return(outputs.df.summary)
}
```