rties: The Coupled-Oscillator Model

Emily Butler, eabutler@u.arizona.edu

Variations on the Coupled-Oscillator model are fairly common in the literature on close relationships (Boker & Laurenceau, 2006, 2007; Butner, Diamond, & Hicks, 2007; Helm, Sbarra, & Ferrer, 2012; Reed, Barnard, & Butler, 2015; Steele & Ferrer, 2011). All the models have in common that they: 1) are based on differential equations, 2) characterize oscillatory phenomena, 3) include some form of coupling (mutual influence between partners), and 4) represent damping or amplification of the oscillations over time. The last distinction is particularly important because it is central to defining co-regulation, whereby partner's mutual influence has a homeostatic effect, resulting in both partners returning to a stable set-point following some disruption to the system, versus co-dysregulation, whereby partner's mutual influence results in increasingly volatile fluctuations away from a set-point (Butler & Randall, 2013; Reed et al., 2015). This contrast is shown in Figure 3 in overview_data_prep.pdf.

One of the challenges of using COMs is that they rely on derivatives. The typical approach in social science is to estimate those derivatives from the data, which has limitations but is tractable, and this is the approach we use in *rties* (for a discussion see (Butler et al., 2017)). We make use of a Local Linear Approximation suggested and implemented in R by S. Boker (Boker, Deboeck, Edler, & Keel, 2010; Boker & Nesselroade, 2002). The *rties* version of a COM predicts the second derivative of the observed state variable from: 1) a person's own state variable, which is related to the frequency of oscillations, 2) a person's own first derivative of the state variable, which indicates damping/amplification, 3) a person's partner's state variable, which indicates coupling with respect to frequency, and 4) a person's partner's first derivative of the state variable, which indicates coupling with respect to damping/amplification. The model includes separate estimates for both partner types (e.g., men and women), resulting in a total of 8 parameters. The *lm* model used by the *rties* functions is:

```
lm(d2 ~ -1 + dist0:obs_deTrend + dist0:d1 + dist0:p_obs_deTrend +
dist0:p_d1 + dist1:obs_deTrend + dist1:d1 + dist1:p_obs_deTrend +
dist1:p_d1, na.action=na.exclude, data=datai)
```

where "d2" is the second derivative of the observed state variable (the time-series emotional experience variable in our example) with linear trends removed (e.g., it is the second derivative of the residuals from each person's state variable predicted from time) . The "-1" results in the intercept being omitted (which is part of the formulation of any coupled-oscillator model). The terms "dist0" and "dist1" are indicator variables, scored 0 or 1 to indicate which partner type is represented. In other words, terms multiplied by "dist0" indicate the estimate of that term for the partner scored 0 on the distinguishing variable provided by the user (see "overview_data_prep.pdf"), while terms multiplied by "dist1" indicate the estimate for the partner scored 1 on the original distinguishing variable. The term "obs_deTrend" is the observed state variable with individual linear trends removed. This parameter represents how quickly the observed process oscillates, but its values are not interpretable until transformed into frequency (cycles per time), or its reciprocal, period (time for one cycle). Estimates for the parameter (we will call it η here) also need to be negative in order to be interpretable. Assuming a negative estimate, $\eta < 0$, the time for one complete cycle (period) is estimated by $((2\pi) / (\sqrt{-[\eta]}))$. Larger absolute values of η are indicative of more rapid oscillations. The term "p_obs_deTrend" is the observed state variable for a person's partner with individual linear trends removed and represents coupling with respect to frequency (e.g., the impact of the partner on one's own oscillatory frequency). The term "d1" is the first derivative of the observed state variable with linear trends removed. Negative estimates of this term represent damping, or the tendency for the state variable to converge back to

homeostatic levels. Positive estimates, in contrast, represent amplification, or the tendency of the state variable to increasingly deviate away from homeostatic levels. A zero estimate suggests a continuously oscillating process of constant amplitude. Finally, the term "p_d1" is the first derivative of a person's partner's state variable with linear trend removed and represents coupling with respect to damping/amplification (e.g., the impact of the partner on one's own damping/amplification). Note that we estimate this model separately for each dyad (e.g., "datai" is the data from couple "i") and hence it is not a multilevel model

There are two sample size considerations for each of the models implemented in *rties*. The first pertains to the number of observations per dyad that are required, which is largely driven by the complexity of the dynamics to be assessed. The second is the number of dyads required, which is driven by the same issues as in regular multiple regression. The first consideration comes into play when we estimate the dynamics one dyad at a time. Greater complexity requires finer-grained measurement of time and hence more observations per dyad. The coupled-oscillator model represents fairly complex dynamics and hence requires more observations per dyad than the inertia-coordination or patterned-slopes models. The exact number of observations required and the spacing between them will depend upon the temporal processes involved, however. For a good discussion of these issues see (Boker & Nesselroade, 2002). As an over-simplified summary, the goal is to have enough observations per oscillatory cycle for at least 2 cycles to be able to "see" the oscillatory pattern. For example, if there were only 2 observations per cycle, there would be no way to estimate the curvature. Or if there were only observations for one cycle, there would be no way to estimate damping/amplification across cycles. A gross guideline that has been suggested is that between 16 to 90 observations per dyad represents the minimum, but again whether this is enough to recover the "true" oscillatory dynamics is dependent on a number of assumptions (Boker & Nesselroade, 2002). A pragmatic approach is to try using a coupled-oscillator model and if you achieve an adequate fit to the data for most dyads you have enough observations per dyad to make progress.

The second sample size consideration comes into play when we use the estimated dynamics to predict the system variable across dyads using multiple regression models. As described above, the full coupled-oscillator model includes 8 parameters, which become the predictors of the system variable. Thus, to decide the necessary number of dyads you can either apply your favorite rule of thumb along the lines of "*n* observations for each of 8 predictors", or you can conduct a power analysis.

The first step in an *rties* analysis is to follow the instructions in "overview_data_prep.pdf" to visualize and prepare the data. As described there, the end result is a dataframe (called "data3" in our example) that has the processed data ready for *rties* modeling. The next step for the coupled-oscillator model is to estimate first and second derivatives of the time-series state variable. As mentioned above, we use a Local Linear Approximation suggested and implemented in R by S. Boker (Boker, Deboeck, Edler, & Keel, 2010; Boker & Nesselroade, 2002). This method requires the user to provide settings for 3 control parameters: tau, embed, and delta. Tau is the number of time points to include when estimating the first derivative, which is the mean of two adjacent slopes across that number of time points on either side of time *t* (e.g., if tau = 2 then the estimate of the first derivative at time = *t* is based on the mean of the slopes left and right of time *t* across 2 observations each). The second derivative is the difference in the two slopes with respect to time. Tau = 1 is sensitive to noise and increasing its value acts as smoothing. Embed is relevant to the degree of derivatives that are desired and the minimum embed is 3 for 2nd order derivatives. Higher values increase smoothing. Finally, delta is the inter-observation interval and is typically set to one (e.g., if equal 2, then every second observation is used).

Choosing optimal values for tau and embed is a complex process and the resulting derivative estimates are highly sensitive to them. In *rties* we provide the "estDerivs" function to investigate sets of tau and embed for a given delta with respect to the quality of fit for an individual oscillator model for each person's data. In other words, the user provides vectors of tau and embed values and the function fits an oscillator model to each person's data using each pair of tau and embed values, and returns a list

with the maximal *R^2* for each person, the values of tau and embed that resulted in that maximal *R^2*, and the period of oscillation associated with that tau/embed combination. This information can be used to adjust the set of tau and embed until the model fit is fairly good for all people and the range of periods of oscillation is appropriate for the process being investigated. For example, if we expect the process we are studying to oscillate every 2-3 minutes, then we may choose values of tau and embed that result in lower overall *R^2* but produce a range of periods from 1-5 minutes in preference to those that produce a higher average *R^2*, but a range of periods from 10-15 minutes. The reasoning here is that it is preferable to have somewhat worse measurement of the right process, than better measurement of the wrong process. Note that you may encounter a variety of error messages during the process of selecting tau and embed vectors, all of which imply an inappropriate combination of tau and embed. The solution is to change the selection of tau and embed until no errors are encountered. The following code assigns values for taus, embeds and delta and then provides those as arguments to estDerivs. Note that the vectors for taus and embeds can be any length, but the longer they are the longer the function will take to run. We chose these taus and embeds based on prior experimentation that showed that smaller values gave poor fits, while larger values either produced inappropriate period lengths for our process of interest or resulted in error messages.

```
taus <- c(7,8,9)
embeds <- c(5,7,9,10)
delta <- 1

derivs <- estDerivs(data3, taus, embeds, delta)
```

The object "derivs" that was created from "estDerivs" contains a dataframe called "data" that holds the derivative estimates for each person using the tau/embed combination that maximized that person's *R^2*. It also contains a dataframe called "fitTable" with the fit information. Here are the first 6 entries of the fitTable from our example:

```
head(derivs$fitTable)
```

```
      id tau embed Max-Rsqr Period
[1,]   2   7    10     0.84 126.65
[2,] 502   9    10     0.85 126.71
[3,]   8   9    10     0.84 162.24
[4,] 508   8    10     0.68 143.05
[5,]  10   9    10     0.75 203.16
[6,] 510   9    10     0.80 144.98
```

The first column is the person ID, followed by the tau and embed that maximized the *R^2* for each person. From these first entries, we can see that 7, 8, and 9 were all chosen for at least one person's tau, while embed was chosen to be 10 for all of them. If that were true for the full sample, we could conclude that values for embed smaller than 10 were not helpful and we may consider changing the set to include larger values. In this example, however, an inspection of the full fitTable shows that the smaller embed values were chosen for a portion of the sample, making it reasonable to keep them as options. The next column has the maximal *R^2s*, which we can inspect in the usual ways (summary, hist, etc):

```
summary(fitTable[ ,4])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3500  0.5700  0.6700  0.6615  0.7400  0.9500
```

We can see that an individual oscillator model, with the tau and embed options we have provided, gives a fairly good fit to the data, with a mean *R^2* of .66. The next consideration is whether these tau/embed combinations are also picking up a period of oscillation that is relevant to our process of interest. In our case, we are investigating emotional experience, which we speculate should oscillate every few minutes based on theories about the time-course of emotions and our prior research. To address this, the last column in fitTable gives the estimated period of oscillation for each person, given the tau/embed combination that resulted in their maximal *R^2*. We can inspect this as usual:

```
summary(fitTable[ ,5])
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  115.3   146.0   162.6   166.7   180.8   267.7
```

The values are given in the temporal units of the raw data, which in our example was 2-second units. Thus to translate the periods back into original time units of seconds we multiply the period by the size of time units over which the observed variable was aggregated. So, for example, for a period of 115 (the minimum period estimated) and an observed variable in 2 second units, we have 115 * 2 = 230 seconds. We can then divide by 60 to get the estimate in minutes if desired, which in this case would be 3.8 minutes. Translating the 1st, 2nd and 3rd quartiles for the period gives:

```
(146*2)/60 # 4.9 minutes ~ 1st quartile
(167*2)/60 # 5.6 minutes ~ 2nd quartile (mean)
(180*2)/60 # 6 minutes ~ 3rd quartile
```

These estimates represent slightly slower oscillations than we might expect for emotional responding based on theory, but further experimentation with tau/embed showed that *R^2* was substantially worse for combinations that provided faster frequency oscillations and so we proceeded with the slower estimates. For convenience in subsequent steps, we assign the dataframe containing the derivative estimates to its own object called "cloData" with the following syntax:

```
cloData <- derivs$data
```

The next step, which is often neglected in the literature, is to assess how well the coupled-oscillator model fits the observed temporal data. Note that in the prior step we assessed the fit of an individual oscillator model to people's data one at a time. Here in the next step, we assess the fit of the coupled-oscillator model to each dyad's data. Our ultimate goal is to either predict outcomes of interest from the dynamics assessed by the coupled-oscillator model, or to test whether other variables moderate those dynamics. Either way, the results are only meaningful if the coupled-oscillator model does, in fact, realistically represent the dynamics of the system. We therefore provide two functions called "indivCloCouple" and "indivCloUncouple" that fit different versions of dyadic oscillator models to each dyad's data. The first fits the full coupled oscillator model, while the second fits a "self" only version of the model that does not include the 4 coupling terms (e.g., it only includes the "self" terms and not the "partner" terms). Both functions return a named list containing: 1) the adjusted *R^2* for each dyad (e.g., how well the model predicts the observed dyadic temporal trajectories of the data, called "r2"), 2) a dataframe ("paramData") with the parameter estimates for the model (for use later in either predicting, or being predicted by, the system variable), and 3) plots of the predicted values superimposed on the observed values for each dyad. The plots can be accessed from the returned named list ("plots") and they are also automatically saved as a .pdf file in the working directory (this process takes awhile and a blank

quartz window may appear, depending on your computer). The function takes the name of the processed dataframe containing the derivative estimates ("cloData' in our example), names for the two levels of the distinguishing variable in the correct order (0 first, then 1; these names will be used to label legends on the plots) and a name for the observed state variable (again, this name will appear on the y-axis of the plots). Here we save the results in an object called "indivCoupleModels" (note that the function "indivCloUncouple" works the same way in all respects):

```
indivCoupleModels <- indivCloCouple(cloData, "women","men","Dial")
```

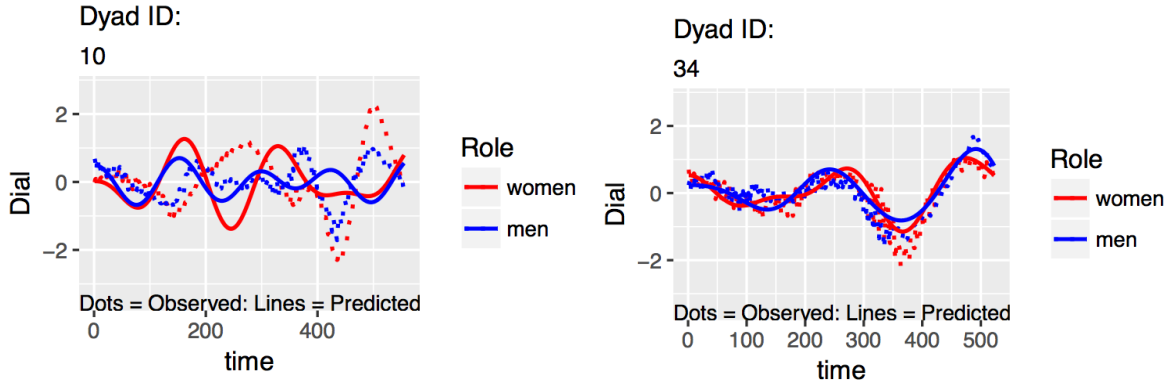
From the results, we can use the "summary" function (or "hist", or any other function) to investigate the adjusted *R^2*s across dyads as an indicator of model fit. The results are:

```
summary(indivCoupleModels$r2)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.4599  0.5780  0.6389  0.6530  0.7176  0.9083
```

Here we see that the coupled-oscillator model accounts on average for about 65% of the variance in the observed state variable (the emotional experience self-reports), and overall fits fairly similarly to the individual oscillator model we used to choose tau/embed combinations for the derivative estimates.

The plots of predicted values overlaid on observed values will be automatically written to the working directory in a file called either "cloPlotsCouple.pdf" or "cloPlotsUncouple.pdf", or they can be accessed as a named list called "plots" from the object created by the functions (e.g., in our example indivCoupleModels$plots holds the plots). The following figures shows examples, with a poor fit on the left and a good fit on the right.



We also provide a function to investigate the difference in the adjusted R-square for the two versions of the oscillator models for each dyad, which can provide evidence about whether there is substantial coupling present or not. The function is called "cloIndivCompare" and takes the data containing the derivative estimates as input. It returns a named list containing: 1) the adjusted *R^2* for each dyad for the uncoupled model ("r2uncouple"), 2) the adjusted *R^2* for each dyad for the coupled model ("r2couple"), and 3) the difference between the two *R^2s* for each dyad (coupled – uncoupled, so positive values indicate better fit for the more complex model ("r2dif")). The *R^2* output for the two models is identical to that produced by the prior functions ("indivCloCouple" and "indivCloUncouple"), but is available here in one place, along with the differences between them. It also runs faster due to not having to produce plots. As can be seen in the output below, in our example the coupled oscillator is only a small improvement over the uncoupled one, with the average improvement in *R^2* being only about .05.

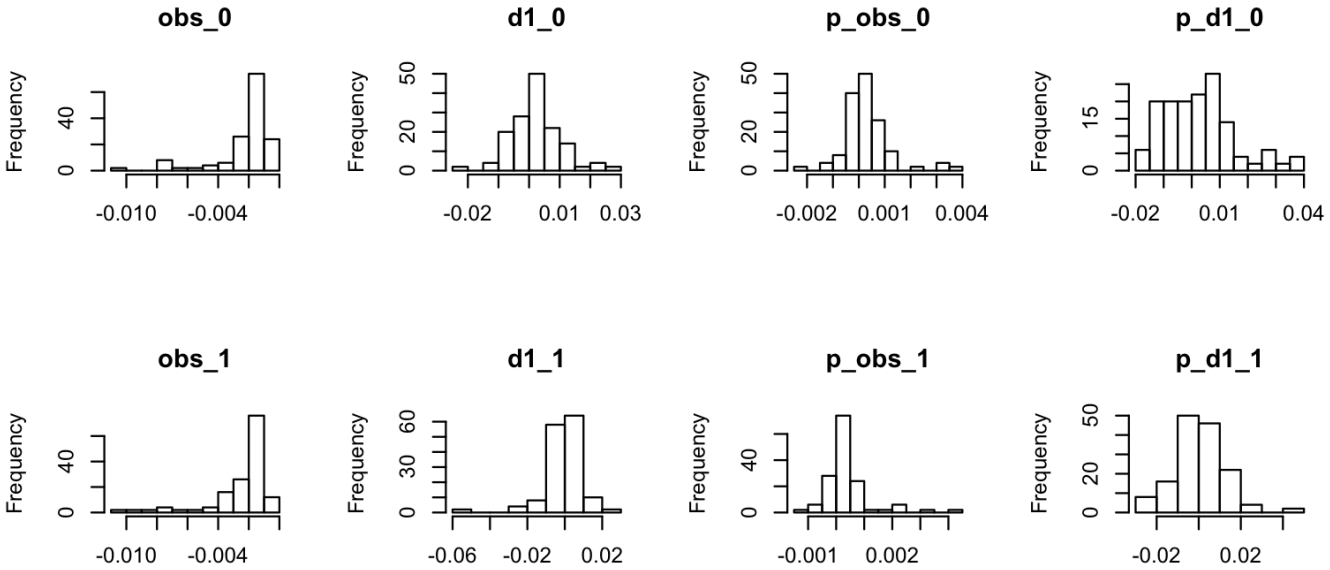
```
compare <- cloIndivCompare(cloData)

summary(compare$r2dif)

      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.0000422  0.0152010  0.0322604  0.0499348  0.0741256  0.1912137
```

The next step in the analysis is to use the parameter estimates generated by "indivClo" to either predict, or be predicted by, the system variable (which is shared unhealthy behavior in our example, called "sub"). At the time of writing, we have only implemented the functions needed to predict the system variable. Future versions of *rties* will also have functions to use it as the predictor, with the coupled-oscillator parameters as the outcomes. Either way, we start by making the parameter estimates into a stand-alone object for convenience:

```
paramData <- indivModels$paramData
```

The variables in "paramData" that are relevant for the analysis are: obs_0 = the frequency estimate for the person scored 0 (partner-0) on the distinguishing variable, obs_1 = the frequency estimate for the person scored 1 (partner-1) on the distinguishing variable, d1_0 = the damping/amplification estimate for partner-0, d1_1 = the damping/amplification estimate for partner-1, p_obs_0 = the coupling estimate with respect to frequency for partner-0, p_obs_1 = the coupling estimate with respect to frequency for partner-1, p_d1_0 = the coupling estimate with respect to damping/amplification for partner-0, and p_d1_1 = the coupling estimate with respect to damping/amplification for partner-1. It is a good idea to look at histograms of these to check that there is adequate variance across dyads to make them meaningful as predictors or outcomes of the system variable. Further, if they are to be used as outcomes, they should be fairly normally distributed (although in later versions of *rties* we intend to implement non-Gaussian options). We make use of the "histAll" function, which produces histograms for all numeric variables in a dataframe, and see that in this example all the parameters are somewhat normally distributed (although many show some skew) with adequate variance.

```
histAll(paramData)
```

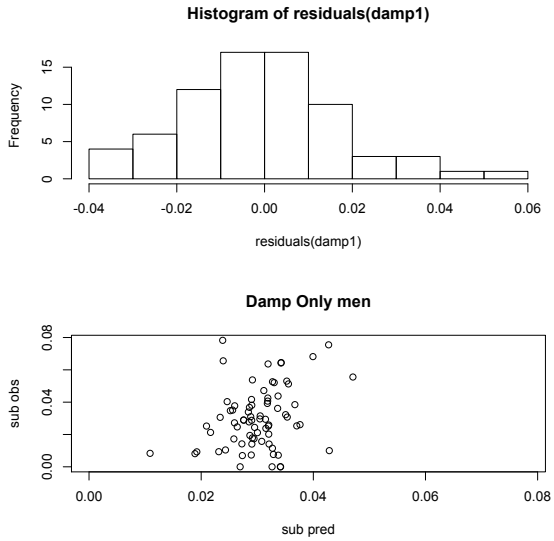


The "cloSysVarOutCompare" function uses the coupled-oscillator parameter estimates to predict the system variable across dyads using multiple regression models. It does so for three sets of predictors: 1) self frequency and damping/amplification (self-only), 2) self and partner damping/amplification (damping/amplification-only), and 3) all self and partner parameters (full coupled-oscillator). The models are currently estimated separately for the two levels of the distinguishing variable, but in future versions of *rties* we will combine them and provide significance tests and model comparisons to assess whether the results differ between the two types of partner.

The function takes the name of the dataframe containing the parameter estimates ("paramData" in our example), names for the two levels of the distinguishing variable in the correct order (0 first, then 1; these names will be used to label legends on the plots) and a name for the observed system variable (again, this name will appear on the y-axis of the plots). The function returns a named list including: 1) the lm objects containing the full results for each model for each level of the distinguishing variable (called "models0" and "models1"), 2) anova output for each model for each level of the distinguishing variable (called "anovas0" and "anovas1"), 3) summary output for each model for each level of the distinguishing variable (called "summaries0" and "summaries1") and 4) adjusted *R^2* information for each model for each level of the distinguishing variable (called "adjustR20" and "adjustR21"). The function also displays histograms of the residuals and plots of the predicted values against observed values for each model for each level of the distinguishing variable.

```
output <-cloSysVarOutCompare(paramData, "women","men","sub")
```

Here is an example of the plots produced:



We can see that the residuals of the damp-only model predicting the system variable for men are fairly normally distributed around zero, suggesting that model assumptions have been met. We also see that the predicted "sub" scores appear to have a positive association with the observed "sub" scores. We can formalize this by looking at the adjusted *R^2* results for each of the three models (self-only, damp-only, full model) predicting "sub" separately by partner type:

```
> output$adjustR20
$self0R2
[1] -0.004083835

$couple0R2
[1] 0.05551488

$damp0R2
[1] 0.04434361
```

```
> output$adjustR21
$self1R2
[1] 0.01307578

$couple1R2
[1] 0.04145539

$damp1R2
[1] 0.06366613
```

We see that the full model accounts for the most variance for partner-0, accounting for about 5.5% of the variance in the shared unhealthy behaviors, while the damp-only model fits better for partner-1, accounting for about 6.4% of the variance in unhealthy behaviors.

We next consider the results of the anovas for each model separately by partner type:

```
> output$anovas0
$self0Anova
Anova Table (Type III tests)

Response: sysVar
             Sum Sq Df F value Pr(>F)
obs_0     0.0000022  1  0.0060 0.9382
d1_0      0.0005978  1  1.6809 0.1990
Residuals 0.0252496 71

$couple0Anova
Anova Table (Type III tests)

Response: sysVar
             Sum Sq Df F value Pr(>F)
obs_0     0.0003547  1  1.0604 0.3067
d1_0      0.0007903  1  2.3625 0.1289
p_obs_0   0.0009452  1  2.8254 0.0973 .
p_d1_0    0.0004985  1  1.4903 0.2263
Residuals 0.0230818 69
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

$damp0Anova
Anova Table (Type III tests)

Response: sysVar
             Sum Sq Df F value  Pr(>F)
d1_0      0.0011944  1  3.5289 0.06441 .
p_d1_0    0.0012199  1  3.6042 0.06170 .
Residuals 0.0240318 71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> output$anovas1
$self1Anova
```

```
Anova Table (Type III tests)

Response: sysVar
             Sum Sq Df F value Pr(>F)
obs_1     0.0000676  1  0.1933 0.6615
d1_1      0.0008197  1  2.3451 0.1301
Residuals 0.0248180 71

$couple1Anova
Anova Table (Type III tests)

Response: sysVar
             Sum Sq Df F value  Pr(>F)
obs_1     0.0001205  1  0.3548 0.55336
d1_1      0.0021025  1  6.1931 0.01524 *
p_obs_1   0.0000245  1  0.0722 0.78894
p_d1_1    0.0013846  1  4.0783 0.04732 *
Residuals 0.0234254 69
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

$damp1Anova
Anova Table (Type III tests)

Response: sysVar
             Sum Sq Df F value  Pr(>F)
d1_1      0.0022504  1  6.7858 0.01118 *
p_d1_1    0.0013398  1  4.0399 0.04824 *
Residuals 0.0235459 71
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

These results suggest that most of the predictive value is coming from the damping/amplification parameters for both partners (e.g., it is those parameters that are significant, or approaching significant, for both partners) and so we focus on the damping-only model going forward. To understand the direction of the effects we look at the summary output for the damping-only model (similar summary results are produced for each of the three models):

```
output$summaries0

$damp0Summary

Call:
lm(formula = sysVar ~ d1_0 + p_d1_0, data = basedata0)

Residuals:
      Min        1Q    Median        3Q       Max
-0.030478 -0.013460 -0.001236  0.010235  0.046216

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
intercept     0.02838    0.00227  12.503   <2e-16 ***
damp_self     0.52733    0.28071   1.879   0.0644 .
damp_partner  0.34100    0.17962   1.898   0.0617 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0184 on 71 degrees of freedom
Multiple R-squared:  0.07053, Adjusted R-squared:  0.04434
F-statistic: 2.694 on 2 and 71 DF,  p-value: 0.07455

output$summaries1

$damp1Summary

Call:
lm(formula = sysVar ~ d1_1 + p_d1_1, data = basedata1)

Residuals:
      Min        1Q    Median        3Q       Max
-0.034158 -0.011681 -0.000576  0.009510  0.054424

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
intercept    0.030129   0.002118  14.223   <2e-16 ***
damp_self    0.694894   0.266758   2.605   0.0112 *
damp_partner 0.487463   0.242525   2.010   0.0482 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.01821 on 71 degrees of freedom
Multiple R-squared:  0.08932, Adjusted R-squared:  0.06367
F-statistic: 3.482 on 2 and 71 DF,  p-value: 0.0361
```

These results show that for both partner types, increases in self-damping/amplification and coupling with respect to damping/amplification predict higher levels of shared unhealthy behaviors (significantly so for men and marginally so for women). This is further borne out by the fact that the overall model accounts for a significant amount of variance in the outcome for men and a marginal amount for women (see F-statistic at the bottom of the summary outputs above). Another step we can take to decide whether the damp-only model is to be preferred is to compare it to the full model for each partner type. We focus here on the women, since for them the adjusted *R^2* was larger for full model.

```
anova(output$models0$couple0, output$models0$damp0)
```

The results below show that the more complex full model (Model 1, which has fewer residual degrees of freedom) does not provide a significant reduction in the residual sums of squares when compared to the simpler damp/amplification-only model (Model 2), a result that would lead us to prefer the simpler model.

```
Analysis of Variance Table

Model 1: sysVar ~ obs_0 + d1_0 + p_obs_0 + p_d1_0
```

```
Model 2: sysVar ~ d1_0 + p_d1_0
  Res.Df      RSS Df   Sum of Sq      F Pr(>F)
1     69 0.023082
2     71 0.024032 -2 -0.00094996 1.4199 0.2487
```

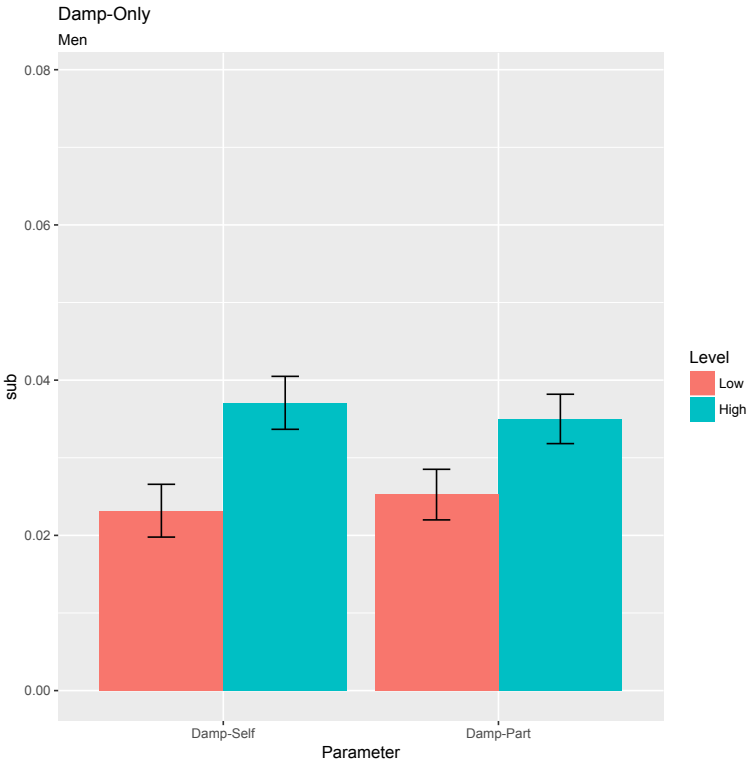
Next, 3 functions are provided to plot the results for each of the 3 models to aid in interpretation (self-only: "selfSysVarOutPlots", damping/amplification-only: "dampSysVarOutPlots", and the full model: "coupleSysVarOutPlots") for one or the other type of partner. Specifically these plots show model predicted means and standard errors for the system variable, for one of the partner types, at user-specified low and high levels of the predictors (e.g., low and high centering values for the parameter estimates). The goal in choosing centering values is that they should indicate levels of the predictor(s) that are of interest due to being representative of meaningful values in the population. If the predictor(s) are normally distributed, then minus and plus one standard deviation, or the 25th and 75th percentile, make good centering values indicative of typical cases at the low and high ends of the distribution. One might also choose theoretically or practically meaningful values, however, such as 25% below and above some clinical cut off. Or, if the predictor(s) are not normally distributed, then histograms can be used to select reasonable values. For example, when a variable is zero-inflated (e.g., there are a large number of zero observations), then zero becomes a good choice for one of the centering values since it is highly representative of part of the population.

Each of the 3 functions takes as arguments the dataframe containing the parameter estimates ("paramData" in our example), several vectors of centering values indicating prototypical low, medium and high levels of the coupled-oscillator parameter estimates (the centering values for "medium" are needed as control variables in the analysis), a name to appear on the y-axis indicating the system variable, a 0 or 1 to indicate which level of the distinguishing variable to plot the results for, and names for the two levels of the distinguishing variable in the correct order (0 first, then 1; these names will be used to label legends on the plots). As we saw when we looked at the histograms of the predictors (e.g., the coupled-oscillator parameter estimates) earlier, they were all somewhat normally distributed, but typically had noticeable skew and so we used visual inspection of the histograms to choose meaningful centering values, which we assign to vectors below:

```
cent_obs_0 <- c(-.004, -.002, -.001)
cent_d1_0 <- c(-.001, 0, .001)
cent_p_obs_0 <- c(-.001, 0, .001)
cent_p_d1_0 <- c(-.01, 0, .01)

cent_obs_1 <- c(-.004, -.002, -.001)
cent_d1_1 <- c(-.01, 0, .01)
cent_p_obs_1 <- c(-.001, 0, .001)
cent_p_d1_1 <- c(-.01, 0, .01)
```

We provide the plots for the damp-only model for men as an example, but the process would be similar for the other two models, or for women:

```
plots3 <- dampSysVarOutPlots(paramData, cent_obs_0, cent_d1_0,
 cent_p_obs_0, cent_p_d1_0, cent_obs_1, cent_d1_1, cent_p_obs_1,
 cent_p_d1_1, "sub", 1, "Women", "Men")
```



Inspection of the centering values shows that "low" damping/amplification parameters are all negative, indicating damping, while "high" parameters are positive, indicating amplification. Thus the figure shows that damping is predictive of lower shared unhealthy behaviors compared to amplification for men. With the coupled-oscillator, however, a full understanding of the dynamics associated with health behaviors requires plotting the emotional trajectories associated with low or high health behaviors, because the individual parameters operate as a set to determine non-linear trajectories. We therefore provide a function called "cloPredTraj" that plots the dynamics predicted by average levels of all the coupled-oscillator parameters, along with two other models where the user specifies the centering values for all the parameters. In our example, it makes sense to create one model with the damping/amplification values all set to negative (indicating damping, which was associated with lower shared unhealthy behaviors) and compare it to a second model where they are all set to positive (indicating amplification, which was associated with higher shared unhealthy behaviors). All other parameters can be set to NA and will be automatically centered at the sample average. The next syntax defines the two user-specified models:

```
# m1 includes all damping at low centering values
paramM1 <- list(obs_0=NA, obs_1=NA, d1_0=-.001, d1_1=-.01,
 p_obs_0=NA, p_obs_1=NA, p_d1_0= -.01, p_d1_1= -.01)

# m2 includes all damping at high centering values
paramM2 <- list(obs_0=NA, obs_1=NA, d1_0=.001, d1_1=.01, p_obs_0=NA,
 p_obs_1=NA ,p_d1_0= .01, p_d1_1=.01)
```

The "cloPredTraj" function takes our initial processed dataframe, called "data3" in this example, the dataframe containing the coupled-oscillator parameter estimates, called "paramData" here, names for the two levels of the distinguishing variable in the correct order (0 first, then 1; these names will be used to label legends on the plots), a name to appear on the y-axis indicating the state variable, and names for the two user-specified models. The plots can then be accessed from the named list created by the

function and are shown below.

```
plots <- cloPredTraj(data3, paramData, "Women", "Men", "Dial", "All
 Damp Low (sub Lower)", "All Damp High (sub Higher)")

plots$aveCloPlot
plots$m1CloPlot
plots$m2CloPlo
```