DynaMORE

Dynamic MOdelling of REsilience

H2020 - 777084

D 1.3– Open-source code of validated resilience model

Dissemination level	Deliverable
Contractual date of delivery	31.03.2024
Actual date of delivery	25.03.2024
Type	Report
Version	0.1.0
Filename	DynaMORE_Deliverable_report_D1.3.pdf
Workpackage	1
Workpackage leader	Jens Timmer

This project has received funding from the European Union’s Horizon 2020 research and innovation programm under grant agreement No 777084.

Author list

Organisation	Name	Contact information
University of Freiburg, Germany	Shakoor Pooseh	shakoor.pooseh@fdm.uni-freiburg.de
University of Freiburg, Germany	Jens Timmer	jeti@fdm.uni-freiburg.de

Executive Summary

This report outlines the development of our validated model designed for estimating dynamic networks of personal emotions over time. This model is suitable for data collected through ecological momentary assessment. A considerable portion of this document served as the supplementary material for WP1's methodological paper, *Intraindividual Time-varying Dynamic Network of Affects: Linear Autoregressive Mixed-Effects Models for Ecological Momentary Assessment*, which can be accessed online at doi.org/10.3389/fpsyt.2024.1213863.

The foundational theories and detailed explanations regarding data type construction and functional tools were previously documented in deliverable D1.2. However, we present an updated version of this material here for the sake of completeness. Additionally, we introduce new sections addressing the impact of missing values, scaling up to higher dimensions, and a practical application scenario where both the full and reduced models are fitted to data obtained from a participant in the DynaM-INT study.

Furthermore, we provide software components that illustrate the data generation process and offer a step-by-step guide to fitting the model. For fitting mixed-effects models, we utilize *MixedModels.jl* in the *Julia* programming language, and *lme4* in *R*.

Recognizing *Julia*'s enhanced computational capabilities, we have implemented the tools and conducted the analysis using the *Julia* language. However, acknowledging the widespread usage of *R* in the field, we have also developed a user-friendly graphical interface. This interface leverages the *Shiny* platform within *R*, providing an accessible means of fitting real data without requiring programming skills. This interface is hosted on a server, serving as a toolbox tailored for users with limited programming experience.

For convenience, we have made the source codes for both implementations available in separate GitHub repositories. The following links provide access to them and the web application.

- Julia: github.com/spooseh/MixedEffectsVAR
- Shiny, source code: github.com/spooseh/larmexShiny
- Shiny web app: spooseh.shinyapps.io/larmexShiny/

1 Introduction

We consider simple directed networks comprising two moods and one external factors node. The two moods interact mutually by enhancing (blue) or dampening (red) each other under the unidirectional influence of external factors.

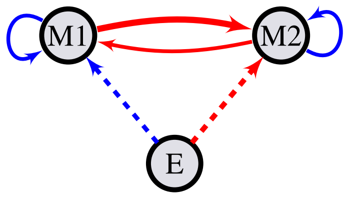


Figure 1: Nodes $M1$ and $M2$ interact with each other leading to temporal correlation, meaning that the value of the source node (variable) at time t has an impact on the value of its target at time $t+1$. Looped arrows indicate autoregressive effects. Exogenous factors influence moods at the same time in a one-way fashion. Blue and red arrows indicate activation and suppression respectively. Different thicknesses stand for distinct interaction strengths. Dashed arrows indicate contemporaneous effects while solid ones are indicative of possible temporal causalities.

Assuming intensive longitudinal data through Ecological Momentary Assessment (EMA), we formalize the mathematical representation of these networks, Figure 1, by an Exogenous Linear Autoregressive Mixed-effects model (LARMEx). The autoregressive coefficients represent the interactions and the external factors are treated as exogenous covariates. We let every parameter in the model have fixed and random components aiming at networks that are allowed to have variable structures over reasonable units of time like days or weeks depending on the study design. Harnessing the rich theoretical and computational literature that furnishes classic linear mixed-effects models, we transform this formulation to a classical one and use the existing methods and tools. We assume our model is the true data generating process and simulate data using a predefined set of parameters. Then we investigate the performance and feasibility of this approach in delivering reliable estimates for different choices of the number of observations and the intensity of noise.

2 Data generating process

Let $m_{i,t} = [m_1, m_2]^T_{i,t}$ be the 2×1 vector corresponding to the mood values from the ith day at any observation occasion $t = 1, 2, \ldots, n_i$ where n_i is the number of observations per day and T denotes the transpose of a matrix.

Assuming that a collection of external factors (E) act on mood nodes and their effect could vary between days, the evolution of $m_{i,t}$ is represented by a LARMEx model as

$$m_{i,t} = (\beta^{ar} + b_i^{ar})m_{i,t-1} + (\beta^e + b_i^e)e_{i,t} + (\beta^c + b_i^c) + \epsilon_{i,t},$$

in which β and b represent the fixed and random effects (FE and RE henceforth). The temporal (Granger-causal) connections between the moods are governed by the lag(1) autoregressive term, $(\beta^{ar} + b_i^{ar})m_{i,t-1}$. The remaining terms represent the contemporaneous effects of E and the constants (intercepts). For simplicity, we assume no connections from moods to E.

3 Generating a set of known parameters

For simulating data with this model, we need to specify the parameters. To achieve replicability, one can predefine a random number generator (RNG). In Julia, we implement this using the following `struct`.

```
mutable struct parLARMEx
    nAR::Int                # number of temporally connected nodes
    rng::AbstractRNG        # one RNG for consintency
    nL2Max::Int             # how many RE to generate
    nSamp::Int              # sample size for generating REs
    B_AR::Matrix{Float64}   # FE autoregressive coefficients
    B_E::Vector{Float64}    # FE coefficients of exogenous factors
    b_var::Matrix{Float64}# variance of group of RE
    b_cov::Matrix{Float64}# variance-covariance matrix of RE
    b_ar::Matrix{Float64}   # RE for autoregressive coefficients
    b_e::Matrix{Float64}    # RE for exogenous coefficients
    b_c::Matrix{Float64}    # RE for constant terms
end
```

To generate instances of this `struct`, a constructor is implemented as

```
parLARMEx(;b=[], nAR=2, rng=MersenneTwister(), nL2Max=100,
           nSamp=20000, B_ar=.3, B_e=.3,  b_var=[.03 .03 .03])
```

where the keyword arguments are:

- `b = []`: if provided with a matrix, RE are extracted from it otherwise generated
- `nAR = 2`: number of temporally connected nodes
- `rng = MersenneTwister()`: if left out, a `MersenneTwister` RNG is produced anew, otherwise a custom one should be provide for consistency and replication
- `nL2Max = 100`: how many RE to generate
- `nSamp = 20000`: initial sample size for generating RE, see genRE_CS() for explanation
- `B_ar = .3`: absolute value of FE autoregressive coefficients, `= []` for random values between 0.1 and 0.6
- `B_e = .3`: absolute value of FE exogenous coefficients, `= []` for random values between 0.1 and 0.6
- `b_var = [.03 .03 .03]`: for constructing a default variance-covariance of RE

For simplicity we restrict the fixed effects to be

$$\beta^{ar} = \begin{bmatrix} 0.3 & -0.3 \\ -0.3 & 0.3 \end{bmatrix}, \quad \beta^{e} = \begin{bmatrix} 0.3 \\ -0.3 \end{bmatrix}, \quad \beta^{c} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

These matrices are generated using keyword arguments to the constructor of `parLARMEx`. It is also possible to initialize these with custom matrices or random values by passing `[]`. The variance-covariance of the random effects is built using `b_var` to be

$$G = \begin{bmatrix} 0.03 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.03 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.03 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.03 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.03 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.03 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.03 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.03 \end{bmatrix}$$

This is not the final variance-covariance of the RE, because one needs to control for the stability of the autoregressive process. To this end, we sample a numger of `nSamp = 20000` sets from a multivariate normal distribution of $\mathrm{MVN}(0, G)$ and retain `nL2Max = 100` sets for which the absoloute eigenvalues of $\beta^{ar} + b^{ar}$ are less than one. These are stored in `b_ar`, `b_e`, `b_c`, and the variance-covariance of this latter sample as `b_cov`.

This process is accomplished by calling the function

`genRE_CS(rng,G,B_AR,maxSbj,nSamp)`

inside the constructor which updates the fields corresponding to the random effects and their variance-covariance structure. In what follows we import these and other function definitions from the file `helperSim.jl` on GitHub, and then construct the parameters using a replicable RNG:

[1]:
```
include("helperSim.jl");
```

[2]:
```
rng = MersenneTwister(1984);
par = parLARMEx(rng=rng);
```

4 Generating data

We assume that the data generating process has a two-level structure. The multiple observations are made on level I and these are nested in level II units. For EMA data, level I could be the daily observations, as many as `nObsL1`, which are nested in single days with a total number of `nL2`. These characteristics together with the varianc-covariance of noise, `sigma`, initial values for days, `M0`, and the vector of known exogenous factors, `E`, are stored in another `struct`.

```
mutable struct simLARMEx
    nAR::Int                # number of temporally connected nodes
    nObsL1::Int             # number of observation on level I
    nL2::Int                # number of units on level II
    sigma::Float64          # variance of noise
    M0::Matrix{Float64}     # initial values each day
    E::Vector{Float64}      # known exogenous factors
end
```

The constructor of this `struct` has the following signature,

```
simLARMEx(;rng=MersenneTwister(), nObsL1=10, nL2=36, sigma=.2,
           M0_max=.5, E=[])
```

and keyword arguments as:

- `rng = MersenneTwister()`: feed `parLARMEx.rng` for consistency and replication, initialized anew by default
- `nObsL1 = 10`: number of observation on level I
- `nL2 = 36`: number of units on level II
- `sigma = .02`: variance of noise
- `M0_max = .5`: determines the amplitude of initial values and exogenous factors, 0.5 to keep trajectories mostly in [-1,1]
- `E = []`: known exogenous factors of size [`nL2` x `nObsL1`], drawn randomly from [`0`, `M0_max`] by default

The following code snippet setts up the configuration of the desired data set. The RNG is carried over to perovide replicablity. Finally, `genData()` gives the simulated data along with the noiseless data as dataframes and the signal-to-noise-ratio (SNR).

[3]:
```
sim = simLARMEx(rng=par.rng);
simData, simData0, SNR = genData(par, sim);
```

[4]:
```
# show(first(simData,5),allcols=true)
latexify(round.(simData[1:5,:],digits=2))
```

[4]:

idL2	tL1	M1	M2	E
1	1	−0.07	0.44	0.27
1	2	−0.03	0.2	0.49
1	3	0.2	0.08	0.19
1	4	0.3	−0.04	0.23
1	5	0.19	−0.17	0.26

The returned values are

- `simData`: the simulated data according to the setup so far
- `simData0`: the simulated data without the noise component used to calculate the SNR
- `SNR`: the ratio of the variance of noiseless data to that of noise

5 Transforming to classical mixed-effects model

By stacking mood values, parameters and covariates for every day separately, and forming the design matrices for fixed and random effects, X and Z, data from a single day takes the following matrix form which is equivalent to a linear mixed effects formulation.

$$\boldsymbol{Y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b}_i + \boldsymbol{\epsilon}_i.$$

In this formulation for a network of two moods and one external nodes one has

$$\boldsymbol{Y}_i = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,n_i} & m_{2,1} & m_{2,2} & \dots & m_{2,n_i} \end{bmatrix}_i^T,$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \beta_{12} & \beta_{21} & \beta_{22} & \beta_1^e & \beta_2^e & \beta_1^c & \beta_2^c \end{bmatrix}^T.$$

and

$$\boldsymbol{X}_i = \boldsymbol{Z}_i = \begin{bmatrix} m_{1,0} & m_{2,0} & 0 & 0 & e_1 & 0 & 1 & 0 \\ m_{1,1} & m_{2,1} & 0 & 0 & e_2 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{1,n_i-1} & m_{2,n_i-1} & 0 & 0 & e_{n_i} & 0 & 1 & 0 \\ 0 & 0 & m_{1,0} & m_{2,0} & 0 & e_1 & 0 & 1 \\ 0 & 0 & m_{1,1} & m_{2,1} & 0 & e_2 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & m_{1,n_i-1} & m_{2,n_i-1} & 0 & e_{n_i} & 0 & 1 \end{bmatrix}_i.$$

In its general form, the above equation for k moods, Y_i is a $k(n_i - 1) \times 1$ vector of mood values, β is a $(k^2 + 2k) \times 1$ vector of fixed effects, b_i is a $(k^2 + 2k) \times 1$ vector of random effects, X_i and Z_i are $k(n_i - 1) \times (k^2 + 2k)$ design matrices. The random effects b_i and residuals ϵ_i are assumed to be independent with a multivariate normal (MVN) distribution of

$$\begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{\epsilon}_i \end{bmatrix} \sim MVN\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{G} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\Sigma}_i \end{bmatrix} \right).$$

This step is done by the following function which gives another data frame suitable for performing the estimation. Here, it is possible to introduce missing values to the analysis, e.g. `miss = .2` for a 20% of missing values at random. By default, it is assumed that there is no missing values, `miss = 0`. The code is available in `helperSim.jl` on GitHub.

```
prepData2Fit(rawData, idL2, endList, exgList; miss=0)
```

with the following arguments:

- `rawData`: the simulated data as a dataframe
- `idL2`: the column name for level II units, e.g., an id for each day, `"idL2"` here
- `endList`: the list of temporelly connected moods, `["M1","M2"]` here
- `exgList`: the list of exogenous factors together with the constant terms, `["E","C"]` here
- `miss`: the ratio of missing values, defaults to zero

[5]:
```
fitData = prepData2Fit(simData,"idL2",["M1","M2"],["E","C"]);
# show(first(fitData,5),allcols=true)
```

[6]:
```
df = fitData[1:5,:]; df[:,3:end] = round.(df[:,3:end],digits=2);
latexify(df)
```

[6]:

idL2	tL1	M	M11	M12	M21	M22	E1	E2	C1	C2
1	2	−0.03	−0.07	0.44	0	0	0.49	0	1	0
1	3	0.2	−0.03	0.2	0	0	0.19	0	1	0
1	4	0.3	0.2	0.08	0	0	0.23	0	1	0
1	5	0.19	0.3	−0.04	0	0	0.26	0	1	0
1	6	0.65	0.19	−0.17	0	0	0.49	0	1	0

6 Setting up the formula for fitting

The prepared data has more columns representing the response variable M, and the predictors including the connections in the network M11, M12, M21, M22, E1, E2, as well as the constant terms C1, C2. It also contains the level I time points tL1, and the level II identifiers idL2. We provide a function in helperSim.jl, on GitHub,

```
setFormula(fitData)
```

which constructs the formula suitable to feed in the Julia package MixedModels.jl. This function is restricted only to a dataframe which has the exact columns as shown above.

[7]:
```
frm = setFormula(fitData);
```

7 Fitting

We use the MixedModels.jl package in Julia to perform the estimation. It is similar to the lme4 in R but exhibits faster performance in our case.

[8]:
```
fit = MixedModels.fit(MixedModel, frm, fitData, progress=false);
show(fit)
```

```
Linear mixed model fit by maximum likelihood
 M ~ 0 + M11 + M12 + M21 + M22 + E1 + E2 +
    (0 + M11 + M12 + M21 + M22 + E1 + E2 + C1 + C2 | idL2)
   logLik   -2 logLik     AIC       AICc        BIC
   246.8663  -493.7327  -407.7327  -401.4678  -215.3554

Variance components:
         ColumnVariance Std.Dev.   Corr.
idL2     M11   0.01 0.13
         M12   0.06 0.25 +0.26
         M21  0.02 0.15 -0.50 +0.68
         M22  0.04 0.21 -0.70 +0.01 +0.37
         E1   0.02 0.16 +0.49 +0.27 -0.05 -0.58
         E2   0.02 0.14 +0.77 +0.57 -0.11 -0.45 +0.28
         C1   0.03 0.17 -0.11 +0.20 +0.41 -0.15 +0.38 -0.45
         C2   0.03 0.17 +0.33 +0.03 -0.15 -0.17 -0.31 -0.01 +0.34
Residual      0.017568 0.132545
 Number of obs: 648; levels of grouping factors: 36

  Fixed-effects parameters:
-------------------------------------------
         Coef.  Std. Error      z  Pr(>|z|)
-------------------------------------------
M11   0.16115    0.0441711   3.65    0.0003
M12  -0.482173   0.059415   -8.12    <1e-15
M21  -0.281178   0.0449767  -6.25    <1e-09
M22   0.342208   0.0553473   6.18    <1e-09
E1    0.394389   0.0587841   6.71    <1e-10
E2   -0.254405   0.0542693  -4.69    <1e-05
-------------------------------------------
```

The following parameters have been estimated. It is evident that the network's structure, with regard to the type of connections, has been accurately reconstructed. However, the precision of the coefficient estimations is not optimal.

$$\hat{\beta}^{ar} = \begin{bmatrix} 0.16 & -0.48 \\ -0.28 & 0.34 \end{bmatrix}, \quad \hat{\beta}^{e} = \begin{bmatrix} 0.39 \\ -0.25 \end{bmatrix},$$

8 Parameter recovery

In order to assess how well one can recover the parameters, we generate data with the following specifications:

- number of observations per day: 10
- $(\sigma^2, \mathrm{SNR}) \in \{(0.01, 12), (0.02, 6), (0.06, 2)\}$
- number of days, $N \in \{4, 6, \ldots, 36\}$
- β^{ar}, β^{e} and β^{c} as before
- data is generated for one simulated subject 1000 times
- at each iteration new random effects are used

This can be done using the GitHub file, `simBootstrap.jl`. It uses the following function to generate data repeatedly in a loop.

```
loopSim(csvDir, M0_max, sigma, n, P)
```

This function creates a folder called `bootstrap` in the current working directory and for each noise intensity, `sigma`, saves the true and estimated parameters of each simulation in different folders, `re` and `reh`, based on the number of days as `CSV` files. Fixed effects and the variances of random effects are collected in two separate `CSV` files.

The code could take some time, depending on the computational power. Using the estimated parameters one can construct bootstrap confidence intervals for fixed effects and the variance of random effects as well as the relative errors of the estimations. For details regarding these analyses, please refer to the main text.

For an up-to-date version please refer to the GitHub repository. Source code for generating Figures is not provided.

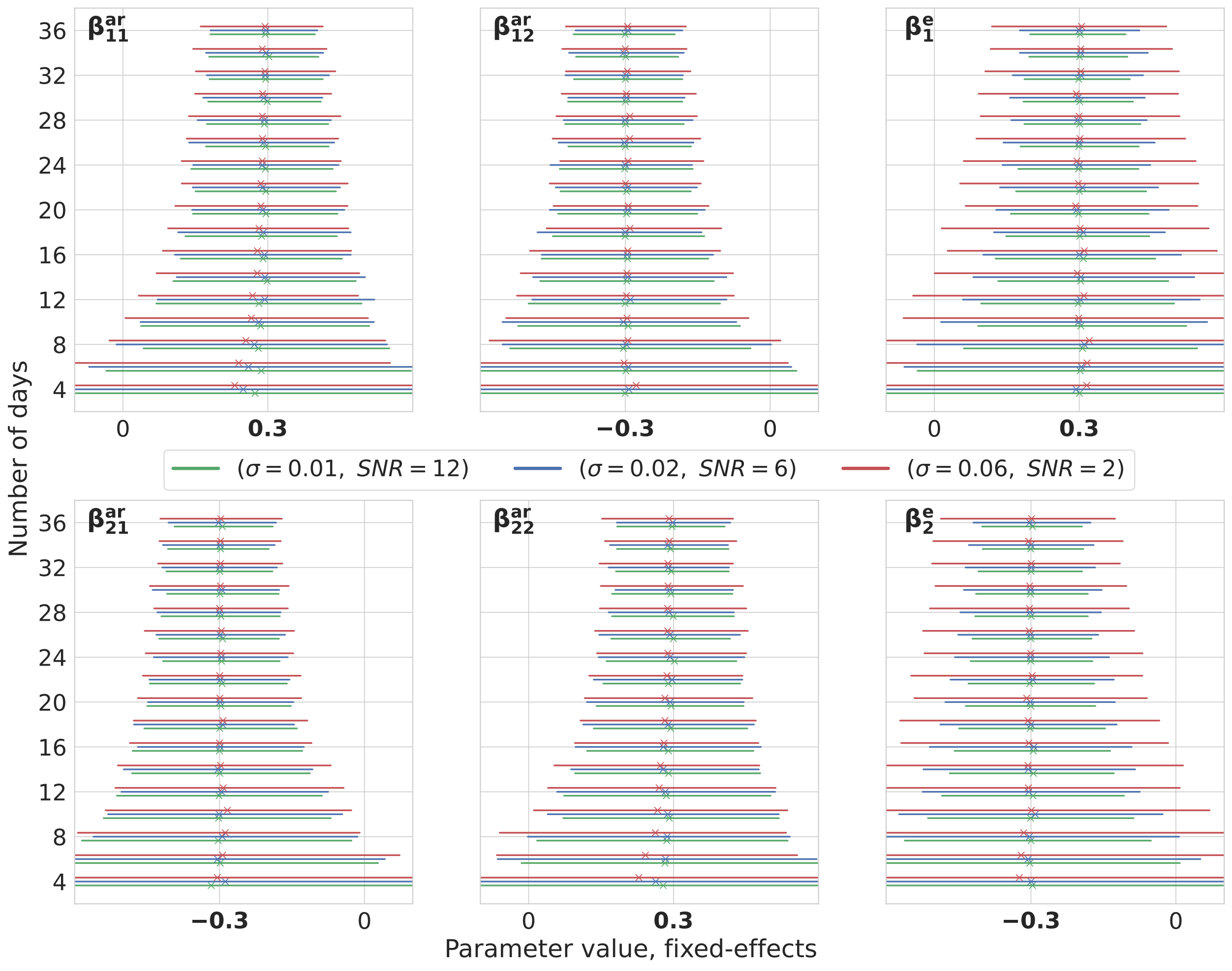


Figure 2: Fixed effects, bootstrap. Estimations of fixed effects for different levels of noise intensity and number of days. The number of observations per day is 10. True values of parameters are highlighted bold on x axes. For every number of days on the vertical axes, three lines are drawn representing the bootstrap 95% confidence intervals around the median depicted by cross signs. Every line is color- and style-coded to demonstrate one noise intensity. Data has been generated for one simulated subject, 1000 times repeatedly, with $(\sigma^2, \text{SNR}) \in \{(0.01, 12), (0.02, 6), (0.06, 2)\}$ and estimations are performed using *MixedModels.jl* package in Julia.

The number of observations per day is 10. True values of parameters are highlighted bold on x axes. For every number of days at the horizontal axes, three lines are drawn representing the bootstrap 95% confidence intervals around the median depicted by cross signs. Every line is color-coded to demonstrate one noise intensity. Data has been generated for one simulated subject, 1000 times repeatedly, with $(\sigma^2, \text{SNR}) \in \{(0.01, 12), (0.02, 6), (0.06, 2)\}$.

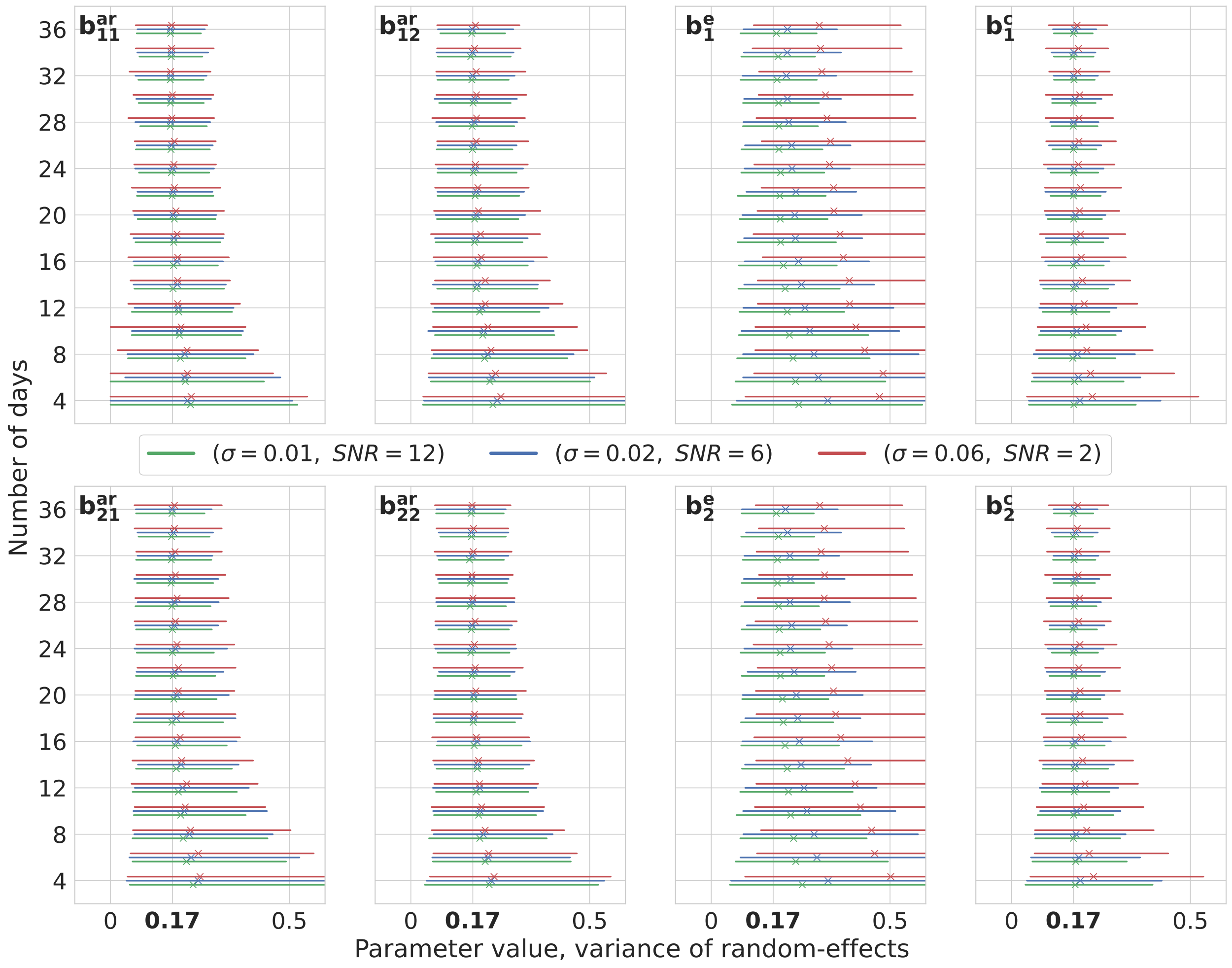


Figure 3: Variance of random effects, bootstrap. Estimations of the variance of random effects for different levels of noise intensity and number of days. The number of observations per day is 10. True values of parameters are highlighted bold on x axes, $\sqrt{0.03} \approx 0.17$. For every number of days on the vertical axes, three lines are drawn representing the bootstrap 95% confidence intervals around the median depicted by cross signs. Every line is color- and style-coded to demonstrate one noise intensity. Data has been generated for one simulated subject, 1000 times repeatedly, with $(\sigma^2, \text{SNR}) \in \{(0.01, 12), (0.02, 6), (0.06, 2)\}$ and estimations performed performed using *MixedModels.jl* package in Julia.

True values of parameters are highlighted bold on x axes, $\sqrt{0.03} \approx 0.17$.

We compute the Relative Estimation Error (REE) in the aforementioned cases as $\delta_\theta = |\hat{\theta} - \theta| / |\theta|$ in which θ and $\hat{\theta}$ represent the true and estimated parameters, respectively. To avoid computational difficulties, only random effects larger than 0.02 are considered in these analyses. We also report only the coefficients present in a network representation, i.e., $\boldsymbol{\beta}^{ar}$, $\boldsymbol{\beta}^{e}$, $\boldsymbol{b}^{ar}$ and $\boldsymbol{b}^{e}$. This analysis underscores that in a mixed-effects model the prediction of random effects is not as reliable as the estimation of fixed effects in terms of being able to recover the parameters varying over days in our case. The reason is that only the variance of random effects are present in the likelihood function and individual values are not estimated directly.

Missing values are a common occurrence in EMA data. For our study, we consider the presence of randomly missing values, leading to the omission of 10%, 20%, and 30% of observations. Employing the same methodology as earlier, we replicate the analysis for a two-variable mood network. We calculate REEs for varying numbers of days and the three distinct levels of missing values. The collective average of these REE values is succinctly presented in Figure 4. This insight reveals a potentially nonlinear influence of missing data, which may be attributed to the practice of listwise deletion, in conjunction with the restriction to consecutively recorded data points. As compliance rates decline, the probability of consecutively recorded data instances experiences a significant decrease. This underscores the necessity of considering imputation techniques depending on the severity of missing values.

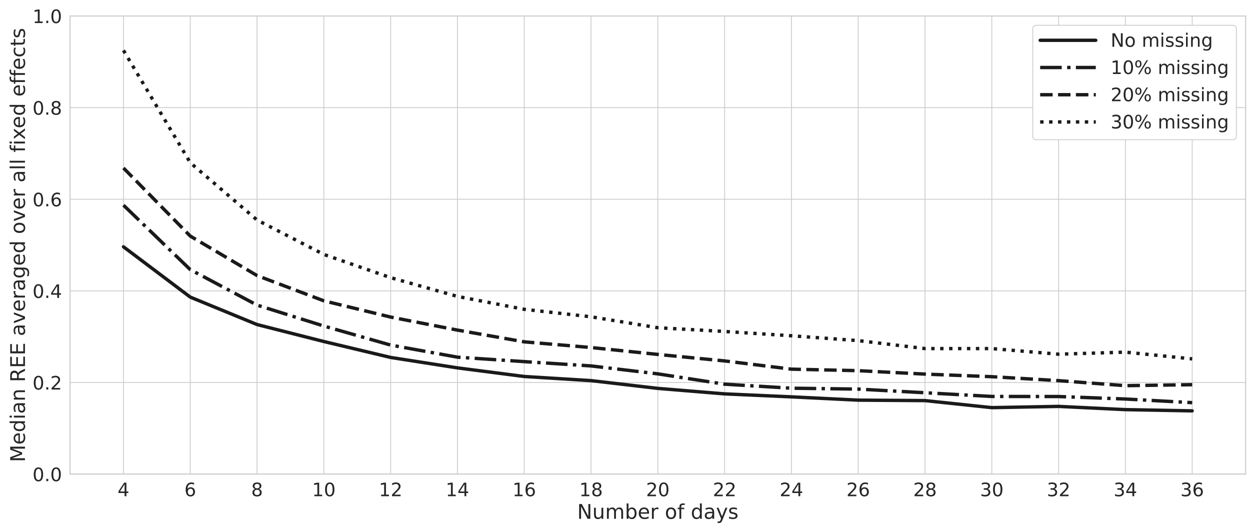


Figure 4: Relative errors, bootstrap with missing values. Lines represent the median Relative Estimation Errors (REE) averaged over fixed-effect parameters of a network with two mood nodes and one node of external factors in the presence of missing values at random. Data is generated for one simulated subject, 1000 times repeatedly, with $(\sigma^2, \text{SNR}) = (0.02, 6)$. We remove 10%, 20%, and 30% of the observations randomly for each case and estimations are performed using *MixedModels.jl* package in Julia.

To assess the scalability of this method in higher dimensions, we simulate a network comprising four interacting nodes—two positive and two negative valences—along with an external factor influencing all mood variables. Figure 5 presents the REEs for this simulation and compares them to previous results, averaging REEs across all fixed and random effects. Notably, for a two-fold increase in the number of mood variables, REEs for fixed effects almost double. This observation suggests that, for smaller networks, employing the full model might be more appropriate. However, for larger networks, it would be advisable to opt for reduced models containing a smaller number of random effects. Additionally, dimension reduction techniques could be applied to merge nodes that measure similar psychological constructs.

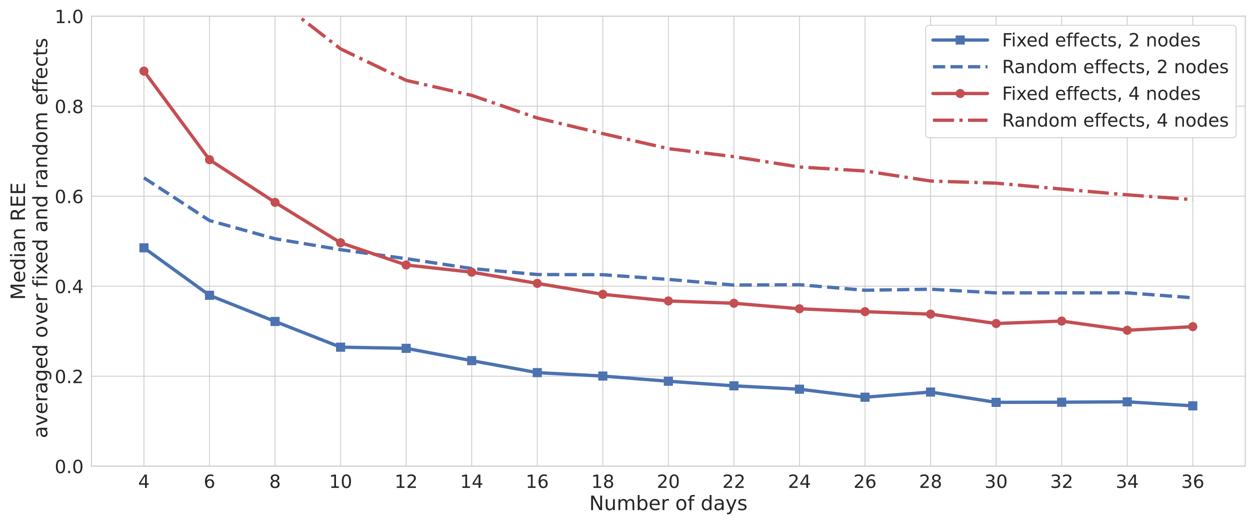
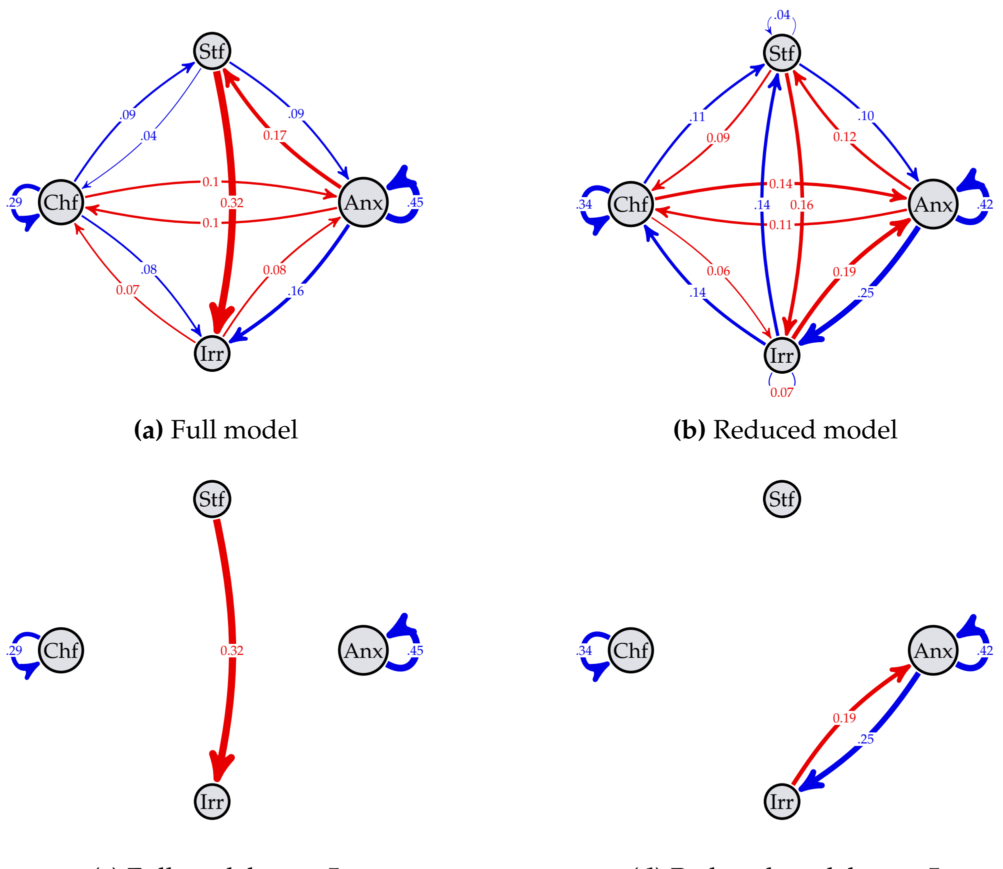


Figure 5: Relative errors, bootstrap. Solid and dashed lines represent the median Relative Estimation Errors (REE) averaged over fixed- and random-effect parameters respectively. The results are depicted with similar colors for networks with two and four mood nodes and one node of external factors. Data is generated for one simulated subject, 1000 times repeatedly, with SNR = 6 for both cases and estimations are performed using *MixedModels.jl* package in Julia.

9 Reduced models for empirical data

Based on our simulations, the EMA data collected in the DynaM-OBS and DynaM-INT studies offer potential for capturing the temporal dynamics of affects, especially within smaller networks of up to five nodes. However, our attempts to apply a multilevel model to these data revealed sparse and weak connections between different moods, even among respondents who exhibited a higher level of compliance. Consequently, we propose a refined model where only the average mood values are permitted to fluctuate across days, while maintaining a constant network structure. This refinement substantially reduces the number of random effects and improves model fits. Figure 6 depicts a network of four nodes fitted to data from a participant in the DynaM-INT study with a compliance rate of 90%. Notably, only a few edges are significantly different from zero in both models. However, reduced models entail more connections and yield superior model fits.



(a) Full model **(b)** Reduced model

(c) Full model, $p < .5$ **(d)** Reduced model, $p < .5$

Figure 6: A participant from the DynaM-INT study with a compliance rate of 90%. a) The full model with a time-varying network across days. All non-zero edges are displayed. b) The reduced model with a fixed network, where only the mean values of moods can vary over days. All non-zero edges are depicted. c) The full model with only statistically significant edges displayed. d) The reduced model with only statistically significant edges depicted.

10 Fitting data in R

Given that *R* is widely used in the field, we also provide a graphical user interface as a *Siny Web app* that is available in a separate GitHub repository. It is implemented as an R package to be installed locally by users and is deployed on a server. Without going into further details we refer the interested readers to https://github.com/spooseh/LARMExShiny for the source code. The Web application is accessible through https://spooseh.shinyapps.io/larmexShiny/

11 Conclusion

We presented an extension of linear mixed-effects models by adding an autoregressive component to model the network representation of mental state. Specifically, we assumed a simple network of two causally interacting moods under the unidirectional influence of an external-factors node. This framework is suitable for intensive longitudinal data.

We showed briefly how this representation is mathematically formulated and detailed the implementation of such a model in Julia. The practical implementation of data generating process was shown by explicit codes which is available on a public repository under GitHub. We also provided exemplary code for performing a bootstrap analysis to construct confidence intervals for the estimations. We also demonstrated how estimation precision is affected when dealing with a missing completely at random scenario as compliance rates decrease. In this analysis, missing values were addressed through listwise deletion. Our results, indicating approximately double relative estimation errors for a network with four nodes compared to one with two nodes, suggest cautious application of this approach in higher dimensions.

The presence of substantial amount of missing EMA data in DynaM-OBS and DynaM-INT exhibit higher-than-expected rates, posing challenges for model validation. Nonetheless, by restricting ourselves to a subset of respondents with a notable higher compliance rates, we were able to gain insights about the validity of our model in delivering time-varying affect networks.