Stochastic Systems: A Bayesian Networks Tutorial

Julia Fischer

Introduction

Technical objective

Learn the basics of Bayesian networks, which can represent systems of variables that are probabilistically related

Substantive research questions

How systematically do various positive emotions co-occur? Can we identify probable causal links between them?

Bayesian networks

In this tutorial, we will explore a type of network model called a **Bayesian network**. A Bayesian network model consists of a collection of variables, each probabilistically taking on various values, and probabilistic linkages among these variables. This means that the variables do not have fixed values, but rather are capable of having different values with different probabilities. Each variable thus has a probability distribution over various values, and this probability distribution is determined by the values of the variables that are causally related to that variable.

In a Bayesian network, the system of variables is represented as a directed acyclic graph (DAG) in which the nodes are variables and the edges are relationships among the variables (Ben-Gal, 2008). Crucially, as indicated by their name, DAGs do not contain any paths from one node back to itself, and the relationships between nodes have directionality. The probabilistic relationships among variables are quantified by conditional probabilities of nodes taking on values given the values of their parent nodes.

To create a Bayesian network, we typically learn the network structure and probability distributions from a set of multivariate data. Once we have generated a Bayesian network, we can fix the values of particular nodes to make inferences and test hypotheses about the system modeled by the network.

Simple Bayesian network example

Before we create a Bayesian network using actual data, it may be helpful to look at a toy example. Let's create a small, simplified network representing some factors that contribute to and result from our happiness on a given day.

Load libraries

```r
library(bnlearn)        # creating Bayesian networks
library(Rgraphviz)      # visualizing Bayesian networks
```

Specify network

Instead of learning this network from data, we simply specify by hand our variables, their distributions, and how they are related to each other. In this example, all of our variables will be limited to the values of true and false. However, in practice, the variables in a Bayesian network can take on any range of values.

Let Sleep be the event that you slept well last night, let Happy be the event that you feel happy today, let Donate be the event that you donate to a charity today, and let Volunteer be the event that you volunteer at a nonprofit today.

```r
sleep <- matrix(c(0.7, 0.3), ncol = 2, dimnames = list(NULL, c("True", "False")))
sleep
```

```
##      True False
## [1,]  0.7   0.3
```

```r
happy <- c(0.8, 0.2, 0.4, 0.6)
dim(happy) = c(2, 2)
dimnames(happy) = list("Happy" = c("True", "False"), "Sleep" = c("True", "False"))
happy
```

```
##        Sleep
## Happy   True False
##   True   0.8   0.4
##   False  0.2   0.6
```

```r
donate <- c(0.5, 0.5, 0.1, 0.9)
dim(donate) = c(2, 2)
dimnames(donate) = list("Donate" = c("True", "False"), "Happy" = c("True", "False"))
donate
```

```
##        Happy
## Donate  True False
##   True   0.5   0.1
##   False  0.5   0.9
```

```r
volunteer <- c(0.6, 0.4, 0.2, 0.8)
dim(volunteer) = c(2, 2)
dimnames(volunteer) = list("Volunteer" = c("True", "False"),
                           "Happy" = c("True", "False"))
volunteer
```

```
##          Happy
## Volunteer True False
##     True   0.6   0.2
##     False  0.4   0.8
```

Describe network

Notice that Sleep has a very simple, fixed probability distribution: The probability that it is true is 0.7, and the probability that it is false is 0.3. In mathematical notation, $P(Sleep) = 0.7$, and $P(Sleep^C) = 0.3$. This distribution is fixed because Sleep has no parent nodes in our network, meaning that none of the other variables have a downstream causal effect on Sleep.

In contrast, the other variables have their probability distributions represented as 2x2 matrices. This is because these variables each have a parent node, and thus their distribution is dependent on the value of this parent node. In more complex networks, some nodes will likely have multiple parent nodes, thus increasing the dimensionality of these matrices.

Consider the distribution matrix for Donate, which has Happy as its parent node. The first column of Donate's distribution matrix represents $P(Donate)$ and $P(Donate^C)$ when Happy is true. The second column represents $P(Donate)$ and $P(Donate^C)$ when Happy is false. Note that each column must sum to 1 to preserve the properties of a probability distribution. Additionally, although there is not a direct edge between Sleep and Donate, their connection through the Happy node allows for an indirect causal link between these two variables.

Build network

We now use the model2network function from the bnlearn package to construct the network object.

```r
simple.structure <- model2network("[Sleep][Happy|Sleep][Donate|Happy][Volunteer|Happy]")
simple.fit = custom.fit(simple.structure, dist = list(Sleep = sleep, Happy = happy,
                                                      Donate = donate,
                                                      Volunteer = volunteer))
simple.fit
```

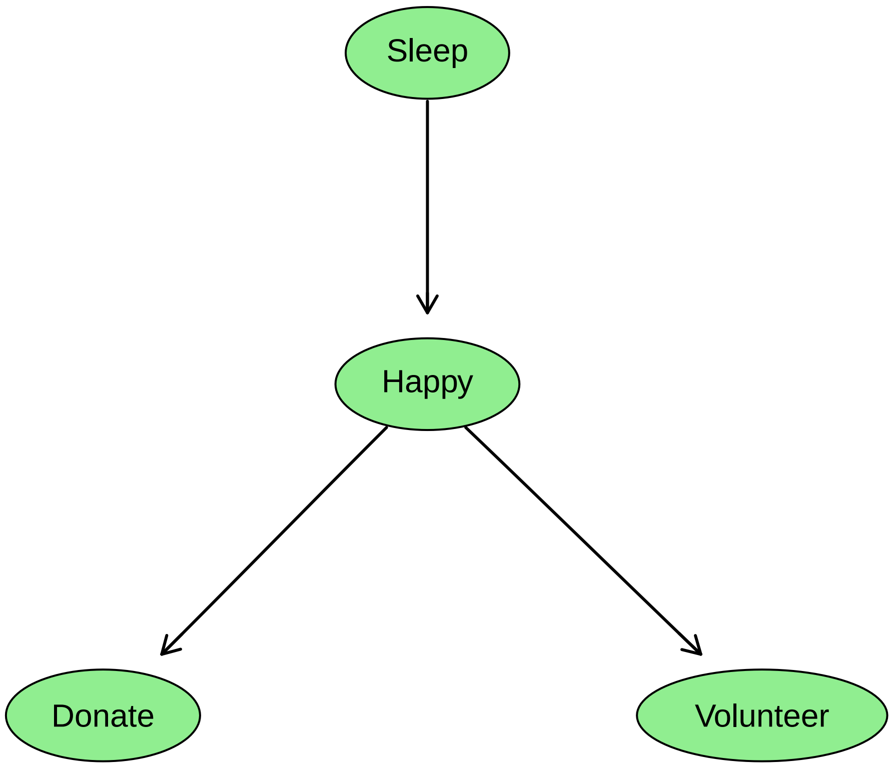
```
##
##   Bayesian network parameters
##
##   Parameters of node Donate (multinomial distribution)
##
## Conditional probability table:
##
##        Happy
## Donate  True False
##   True   0.5   0.1
##   False  0.5   0.9
##
##   Parameters of node Happy (multinomial distribution)
##
## Conditional probability table:
##
##        Sleep
## Happy   True False
##   True   0.8   0.4
##   False  0.2   0.6
##
##   Parameters of node Sleep (multinomial distribution)
##
## Conditional probability table:
##
```

```
##  True False
##   0.7   0.3
##
##   Parameters of node Volunteer (multinomial distribution)
##
## Conditional probability table:
##
##          Happy
## Volunteer True False
##     True   0.6   0.2
##     False  0.4   0.8
```

Visualize network

Next, we create a graphical representation of the model using the Rgraphviz package.

```
col_1 <- c("Sleep", "Happy", "Happy")
col_2 <- c("Happy", "Donate", "Volunteer")
arc_matrix <- cbind(col_1, col_2)
graphviz.plot(simple.fit, highlight = c(list(nodes = c("Sleep", "Happy",
                                                      "Donate", "Volunteer"),
                                         arcs = arc_matrix, col = "black",
                                         fill = "lightgreen", lwd = 1.5)),
              shape = "ellipse", fontsize = 8)
```



Learning a Bayesian network from data

Now that we have covered the basics of Bayesian networks, let's apply these concepts to actual data. The data we are using come from the AMIB study, a multiple timescale study of college students (Ram et al., 2012). In particular, we will be looking at college students' daily self-reports of their positive emotions (affect) over the course of eight days. The data can be downloaded from https://thechangelab.stanford.edu/collaborations/the-amib-data/.

Load helpful libraries

```r
library(psych)          # descriptive statistics
library(tidyverse)      # data manipulation and visualization
```

Read in day-level data

```r
# set filepath for data file
filepath <-
  "https://raw.githubusercontent.com/The-Change-Lab/collaborations/main/AMIB/AMIB_daily.csv"
# read in the .csv file using the url() function
AMIB_daily <- read.csv(file=url(filepath), header=TRUE)
```

Describe the variables of interest in our data

```r
pos_affect_data <- AMIB_daily[c("enthusiastic", "calm", "happy", "peaceful", "satisfied",
                     "proud", "relaxed", "excited", "content", "relieved")]
describe(pos_affect_data)
```

```
##              vars    n mean   sd median trimmed  mad min max range  skew
## enthusiastic    1 1441 4.21 1.56    4.0    4.24 1.48   1   7     6 -0.20
## calm            2 1441 4.40 1.46    4.0    4.46 1.48   1   7     6 -0.25
## happy           3 1441 4.67 1.35    5.0    4.72 1.48   1   7     6 -0.35
## peaceful        4 1441 4.26 1.52    4.0    4.30 1.48   1   7     6 -0.21
## satisfied       5 1438 4.34 1.47    4.0    4.39 1.48   1   7     6 -0.23
## proud           6 1440 3.41 1.65    3.5    3.34 2.22   1   7     6  0.15
## relaxed         7 1439 4.01 1.56    4.0    4.04 1.48   1   7     6 -0.13
## excited         8 1439 3.84 1.67    4.0    3.84 1.48   1   7     6  0.04
## content         9 1441 4.42 1.53    4.0    4.46 1.48   1   7     6 -0.25
## relieved       10 1439 3.04 1.62    3.0    2.92 1.48   1   7     6  0.44
##              kurtosis   se
## enthusiastic    -0.61 0.04
## calm            -0.46 0.04
## happy           -0.24 0.04
## peaceful        -0.60 0.04
## satisfied       -0.45 0.04
## proud           -0.83 0.04
## relaxed         -0.62 0.04
## excited         -0.86 0.04
## content          -0.59 0.04
## relieved         -0.60 0.04
```

Learn network

We now learn a Bayesian network from these data using bnlearn. Even though the data are nested by participant and ordered by day, we treat all observations as independent for simplicity.

We start by learning the network structure using the PC algorithm, which is a constraint-based structure learning algorithm. There are also other structure learning algorithms available through bnlearn.

```
pos_affect_data <- as.data.frame(sapply(pos_affect_data, as.numeric))
pos.affect.pdag <- pc.stable(pos_affect_data)
# set directions of undirected edges
pos.affect.dag = cextend(pos.affect.pdag)
pos.affect.dag
```

```
##
##   Bayesian network learned via Constraint-based methods
##
##   model:
##    [calm][happy][relieved][enthusiastic|calm:happy][content|happy]
##    [relaxed|calm:content][peaceful|enthusiastic:calm:happy:relaxed:content]
##    [proud|enthusiastic:peaceful:content:relieved]
##    [excited|enthusiastic:happy:proud:content:relieved]
##    [satisfied|enthusiastic:calm:happy:proud:excited:content:relieved]
##   nodes:                                 10
##   arcs:                                  26
##     undirected arcs:                     0
##     directed arcs:                       26
##   average markov blanket size:           7.80
##   average neighbourhood size:            5.20
##   average branching factor:              2.60
##
##   learning algorithm:                    PC (Stable)
##   conditional independence test:         Pearson's Correlation
##   alpha threshold:                       0.05
##   tests used in the learning procedure:  6372
```

We next learn the network's parameter values, i.e., the nodes' conditional probability distributions. We do this using maximum likelihood estimation (MLE). In particular, we use the "mle-g" method, which is appropriate for variables with continuous values. Like with learning the network structure, there are other methods available for learning the parameters.

```
pos.affect.fit = bn.fit(pos.affect.dag, pos_affect_data, method = "mle-g")
```

```
## Warning in check.data(data, allow.missing = TRUE): some observations in the
## data contain only missing values.
```

```
pos.affect.fit
```

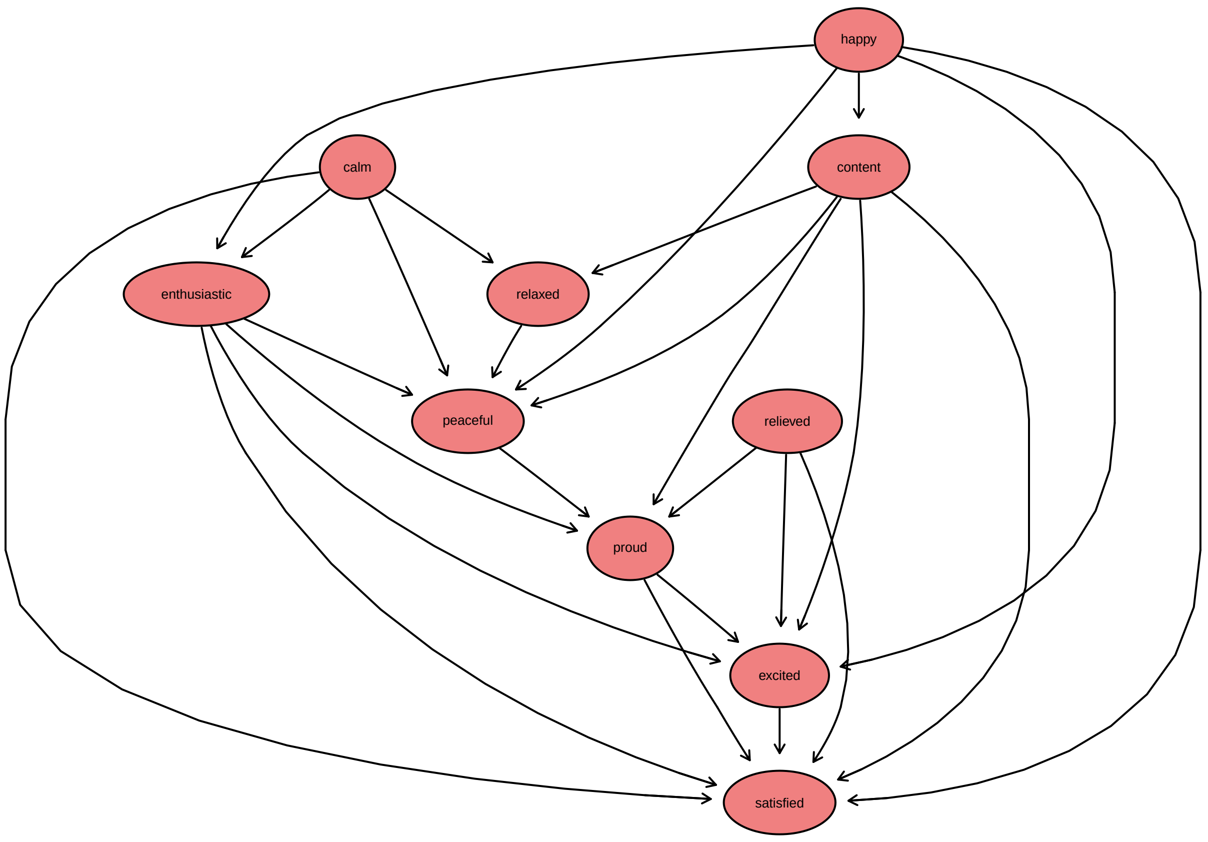
```
##
##   Bayesian network parameters
##
##   Parameters of node enthusiastic (Gaussian distribution)
##
## Conditional density: enthusiastic | calm + happy
## Coefficients:
## (Intercept)         calm        happy
##   0.4310749    0.1350353    0.6817746
## Standard deviation of the residuals: 1.173503
##
##   Parameters of node calm (Gaussian distribution)
##
## Conditional density: calm
## Coefficients:
## (Intercept)
##     4.40458
## Standard deviation of the residuals: 1.462762
##
##   Parameters of node happy (Gaussian distribution)
##
## Conditional density: happy
## Coefficients:
## (Intercept)
##    4.673838
## Standard deviation of the residuals: 1.3547
##
##   Parameters of node peaceful (Gaussian distribution)
##
## Conditional density: peaceful | enthusiastic + calm + happy + relaxed + content
## Coefficients:
##  (Intercept)  enthusiastic          calm         happy       relaxed
## -0.074266409  -0.009351932   0.368490137   0.302858355   0.239950691
##      content
##  0.084782523
## Standard deviation of the residuals: 0.9469995
##
##   Parameters of node satisfied (Gaussian distribution)
##
## Conditional density: satisfied | enthusiastic + calm + happy + proud + excited + content + relieved
## Coefficients:
##  (Intercept)  enthusiastic          calm         happy         proud
##   0.26296788    0.11133058    0.08144244    0.23180825    0.10836457
##      excited       content      relieved
##   0.04976492    0.32419586    0.05861236
## Standard deviation of the residuals: 0.947653
##
##   Parameters of node proud (Gaussian distribution)
##
## Conditional density: proud | enthusiastic + peaceful + content + relieved
## Coefficients:
##  (Intercept)  enthusiastic      peaceful       content      relieved
##  0.009524177   0.310539698   0.111214856   0.241571916   0.180928821
## Standard deviation of the residuals: 1.327437
```

```
##
##   Parameters of node relaxed (Gaussian distribution)
##
## Conditional density: relaxed | calm + content
## Coefficients:
## (Intercept)         calm      content
##   0.2551665    0.5236695    0.3276720
## Standard deviation of the residuals: 1.101658
##
##   Parameters of node excited (Gaussian distribution)
##
## Conditional density: excited | enthusiastic + happy + proud + content + relieved
## Coefficients:
##  (Intercept)  enthusiastic         happy         proud       content
##  -0.28629188    0.42410965    0.21058613    0.06189088    0.20286323
##     relieved
##   0.08043524
## Standard deviation of the residuals: 1.172746
##
##   Parameters of node content (Gaussian distribution)
##
## Conditional density: content | happy
## Coefficients:
## (Intercept)        happy
##   1.1227475    0.7055859
## Standard deviation of the residuals: 1.19516
##
##   Parameters of node relieved (Gaussian distribution)
##
## Conditional density: relieved
## Coefficients:
## (Intercept)
##    3.042391
## Standard deviation of the residuals: 1.620542
```

Visualize network

Next, we plot the structure of our Bayesian network.

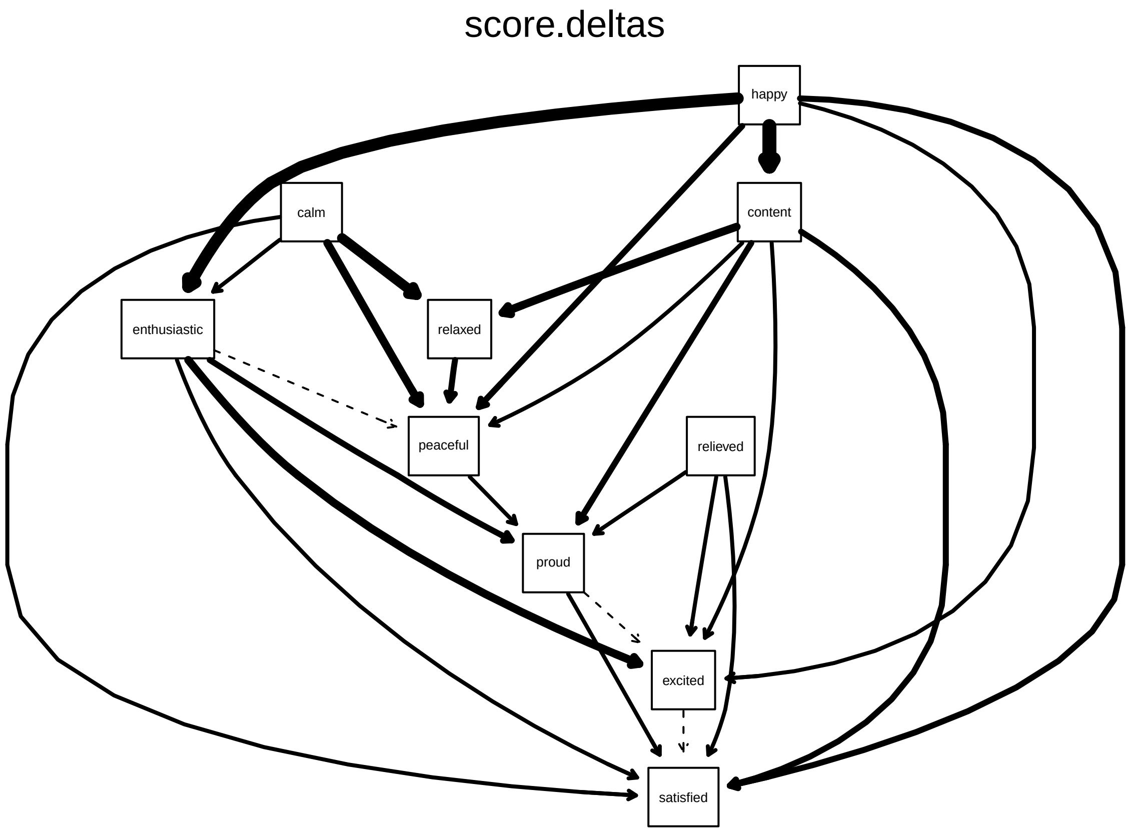
```r
graphviz.plot(pos.affect.fit, highlight = c(list(nodes = c("enthusiastic", "calm",
                                                            "happy", "peaceful",
                                                            "satisfied", "proud",
                                                            "relaxed", "excited",
                                                            "content", "relieved"),
                                                  col = "black", fill = "lightcoral")),
              shape = "ellipse", fontsize = 8)
```



We see that there is a high degree of connectivity among the nodes. This suggests that when people are feeling one positive emotion, they are likely to also feel others. Additionally, we observe that happy, calm, and relieved have no parent nodes, suggesting that they may be near the "beginning" of the positive emotion causal pathway.

To get a sense of the strengths of the relationships among these variables, we can generate a plot that weights the edges based on how "necessary" they are for the goodness of fit of the model. The value for each edge is calculated using the difference in Bayesian information criterion (BIC) when that edge is removed from the network. Note that we must omit missing data (NA values) in order to calculate BIC.

```
score.deltas = arc.strength(pos.affect.dag,
                            data = na.omit(pos_affect_data), criterion = "bic-g")
strength.plot(pos.affect.dag, strength = score.deltas, main = "score.deltas")
```



Conclusion

In this tutorial, we introduced Bayesian networks, a method for representing systems of probabilistically linked variables. We first hand-specified a simple Bayesian network, then learned a more complex network from actual data.

There are many ways to extend the basic Bayesian network beyond what was discussed in this tutorial. In particular, there are variations of networks that can be used to model processes that unfold over time, learning, and causal relationships among variables (Pearl & Russell, 2000).

Bayesian networks also have a multitude of applications in psychological research. They have been used, for example, to model symptoms of comorbid psychological disorders (McNally et al., 2017) and associations among stressful life events, psychological symptoms, and physical symptoms (García-Herrero et al., 2012). Psychological phenomena often involve multiple variables that are related in probabilistic ways, making Bayesian networks a useful analysis tool.

Additional resources

If you would like to learn more about the concepts mentioned in this tutorial, here are a few external resources:

- The mathematics behind Bayesian networks: https://doi.org/10.1007/978-3-540-85066-3_3
- The bnlearn package: https://www.bnlearn.com/
- Bayesian networks for psychopathology research: https://doi.org/10.1037/met0000479

References

Ben-Gal, I. (2008). Bayesian networks. In F. Ruggeri, R.S. Kenett, & F.W. Faltin (Eds.), *Encyclopedia of Statistics in Quality and Reliability.* https://doi.org/10.1002/9780470061572.eqr089

García-Herrero, S., Mariscal, M. A., García-Rodríguez, J., & Ritzel, D. O. (2012). Working conditions, psychological/physical symptoms and occupational accidents. Bayesian network models. *Safety Science*, 50(9), 1760–1774. https://doi.org/10.1016/j.ssci.2012.04.005

McNally, R. J., Mair, P., Mugno, B. L., & Riemann, B. C. (2017). Co-morbid obsessive-compulsive disorder and depression: A Bayesian network approach. *Psychological Medicine*, 47(7), 1204–1214. https://doi.org/10.1017/S0033291716003287

Pearl, J., & Russell, S. (2000). Bayesian networks. *Department of Statistics, UCLA.*

Ram, N., Conroy, D. E., Pincus, A. L., Hyde, A. L., & Molloy, L. E. (2012). Tethering theory to method: Using measures of intraindividual variability to operationalize individuals' dynamic characteristics. In G. Hancock & J. Harring (Eds.), *Advances in longitudinal methods in the social and behavioral sciences* (pp. 81-110). New York: Information Age.