Thesis title

by

Author name

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements
for the degree of

degree

in

department

in the field of field (if necessary)

Guelph, Ontario, Canada

ABSTRACT

THESIS TITLE

Author name
University of Guelph, year

Advisor(s):
advisor name

Despite the value that longitudinal research offers for understanding psychological processes, studies in organizational research rarely use longitudinal designs. One reason for the paucity of longitudinal designs may be the challenges they present for researchers. Three challenges of particular importance are that researchers have to determine 1) how many measurements to take, 2) how to space measurements, and 3) how to design studies when participants provide data with different response schedules (time unstructuredness). In systematically reviewing the simulation literature, I found that few studies comprehensively investigated the effects of measurement number, measurement spacing, and time structuredness (in addition to sample size) on model performance. As a consequence, researchers have little guidance when trying to conduct longitudinal research. To address these gaps in the literature, I conducted a series of simulation experiments. I found poor model performance across all measurement number/sample size pairings. That is, bias and precision were never concurrently optimized under any combination of manipulated variables. Bias was often low, however, with moderate measurement numbers and sample sizes. Although precision was frequently poor, the greatest improvements in precision resulted from using either seven measurements with $N \geq 200$ or nine measurements with $N \leq 100$. With time-unstructured data, model performance systematically

decreased across all measurement number/sample size pairings when the model incorrectly assumed an identical response pattern across all participants (i.e., time-structured data). Fortunately, when models were equipped to handle heterogeneous response patterns using definition variables, the poor model performance observed across all measurement number/sample size pairings no longer appeared. Altogether, the results of the current simulation experiments provide guidelines for researchers interested in modelling nonlinear change.

DEDICATION

[Dedication goes here if necessary]

ACKNOWLEDGEMENTS

[Acknowledgements go here]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

1 access the content in 00-abstract.Rmd file to print the abstract.

Placeholder

1.1 Quotes

1.2 Footnotes

1.3 Figures

1.4 Tables

1.5 Equations and links

To generate the data, the *multilevel logistic function* shown below in Equation (1.1) was used:

$$y_{ij} = \theta_j + \frac{\alpha_j - \theta_j}{1 + e^{\frac{\beta_j - time_i}{\gamma_j}}} + \epsilon_{ij}, \tag{1.1}$$

where θ represents the baseline parameter, α represents the maximal elevation parameter, β represents the days-to-halfway elevation parameter, and γ represents triquarter-halfway delta parameter. Note that, values for θ, α, β, and γ were generated for each j person across all i time points, with an error value being randomly generated at each i time point.

2 Experiment 2

In the `.Rmd` file, notice that a tag has been added so that a hyperlink to Experiment 2 can be done by using `[Experiment 2](#Exp2)` which produces Experiment 2.

‘

Appendix A: Ergodicity and the Need to Conduct Longitudinal Research

To understand why cross-sectional results are unlikely to agree with longitudinal results for any given analysis, a discussion of data structures is apropos. Consider an example where a researcher obtains data from 50 people measured over 100 time points such that each row contains a p person's data over the 100 time points and each column contains data from 50 people at a t time point. For didactic purposes, all data are assumed to be sampled from a normal distribution. To understand whether findings in any given cross-sectional data set yield the same findings in any given longitudinal data set, the researcher randomly samples one cross-sectional and one longitudinal data set and computes the mean and variance in each set. To conduct a cross-sectional analysis, the researcher randomly samples the data across the 50 people at a given time point and computes a mean of the scores at the sampled time point ($\bar{X}_t$) using Equation A.1 shown below:

$$\bar{X}_t = \frac{1}{P}\sum_{p=1}^{P} x_p, \tag{A.1}$$

A.0.1 Example

Example A.1. *Estimates of Taylor series approximation of* $f(x) = \cos(x)$ *as the difference between the point of evaluation* x *and the point of derivation* a *increases.*

Taylor series approximation of $\cos(x)$ (specifically, the second-order Taylor series;

$P^2[\cos(x), a])$ estimates values that are exactly equal to the values returned by $\cos(x)$ when the point of evaluation (x) is set to the point of derivation (a). The example below computes the value predicted by the Taylor series approximation of $P^2[\cos(x), a]$ and by $\cos(x)$ when $x = a = 0$.

$$P^2(\cos(x = 0), a = 0) = \cos(x = 0)$$
$$1 - \frac{1}{2}x^2 = \cos(0)$$
$$1 - \frac{1}{2}0^2 = 1$$
$$1 - 0 = 1$$
$$1 = 1$$

Taylor series approximation of $\cos(x)$ (specifically, the second-order Taylor series; $P^2[\cos(x), a])$ estimates a value that is approximately equal ($\approx$) to the value returned by $f\cos(x)$ when the difference between the point of evaluation x and the point of derivation a is small. The example below computes the value predicted by the Taylor series approximation of $P^2[\cos(x), a]$ and by $\cos(x)$ when $x = 1$ and $a = 0$.

$$P^2(\cos(x = 1), 0) \approx \cos(x = 1)$$
$$1 - \frac{1}{2}x^2 \approx \cos(1)$$
$$1 - \frac{1}{2}1^2 \approx 0.54$$
$$1 - 0.5 \approx 0.54$$
$$0.5 \approx 0.54$$

Taylor series approximation of $f\cos(x)$ (specifically, the second-order Taylor series;

$P^2[\cos(x), a])$ estimates a a value that is clearly not equal ($\neq$) to the value returned by $f\cos(x)$ when the difference between the point of evaluation x and the point of derivation a is large. The example below computes the value predicted by the Taylor series approximation of $P^2[\cos(x), a]$ and by $\cos(x)$ when $x = 4$ and $a = 0$.

$$P^2(\cos(x = 4), 0) \neq \cos(x = 4)$$

$$1 - \frac{1}{2}x^2 \neq \cos(4)$$

$$1 - \frac{1}{2}4^2 \neq -0.65$$

$$1 - 16 \neq -0.65$$

$$0.5 \neq -0.65$$

Appendix B: Code block

The code that I used to model logistic pattern of change (see data generation) is shown in Code Block B.1. Note that, the code is largely excerpted from the `run_exp_simulations()` and `create_logistic_model_ns()` functions from the `nonlinSims` package, and so readers interested in obtaining more information should consult the source code of this package. One important point to mention is that the model specified in Code Block B.1 assumes time-structured data.

Code Block B.1

OpenMx Code for Structured Latent Growth Curve Model That Assumes Time-Structured Data

```
#Days on which measurements are assumed to be taken (note that model assumes
time-structured data; that is, at each time point, participants provide data at the
exact same moment). The measurement days obtained by finding the unique values in the
`measurement_day` column of the generated data set.
measurement_days <- unique(data$measurement_day)
```

```r

#Manifest variable names (i.e., names of columns containing data at each time point,
manifest_vars <- nonlinSims:::extract_manifest_var_names(data_wide = data_wide)

#Now convert data to wide format (needed for OpenMx)
data_wide <- data[ , c(1:3, 5)] %>%
    pivot_wider(names_from = measurement_day, values_from = c(obs_score,
    actual_measurement_day))

#Remove . from column names so that OpenMx does not run into error (this occurs
because, with some spacing schedules, measurement days are not integer values.)
names(data_wide) <- str_replace(string = names(data_wide), pattern = '\\.', replacement
= '_')

#Latent variable names (theta = baseline, alpha = maximal elevation, beta =
days-to-halfway elevation, gamma = triquarter-haflway elevation)
latent_vars <- c('theta', 'alpha', 'beta', 'gamma')

latent_growth_curve_model <- mxModel(
  model = model_name,
  type = 'RAM', independent = T,
  mxData(observed = data_wide, type = 'raw'),

  manifestVars = manifest_vars,
  latentVars = latent_vars,

  #Residual variances; by using one label, they are assumed to all be equal
  (homogeneity of variance). That is, there is no complex error structure.
  mxPath(from = manifest_vars,
         arrows=2, free=TRUE,  labels='epsilon', values = 1, lbound = 0),

  #Latent variable covariances and variances (note that only the variances are
  estimated. )
  mxPath(from = latent_vars,
         connect='unique.pairs', arrows=2,
         free = c(TRUE,FALSE, FALSE, FALSE,
                  TRUE, FALSE, FALSE,
                  TRUE, FALSE,
                  TRUE),
         values=c(1, NA, NA, NA,
                  1, NA, NA,
                  1, NA,
                  1),
         labels=c('theta_rand', 'NA(cov_theta_alpha)', 'NA(cov_theta_beta)',
                  'NA(cov_theta_gamma)',
                  'alpha_rand','NA(cov_alpha_beta)', 'NA(cov_alpha_gamma)',
                  'beta_rand', 'NA(cov_beta_gamma)',
                  'gamma_rand'),
         lbound = c(1e-3, NA, NA, NA,
                    1e-3, NA, NA,
                    1, NA,
                    1),
         ubound = c(2, NA, NA, NA,
                    2, NA, NA,
                    90^2, NA,
                    45^2)),

  # Latent variable means (linear parameters). Note that the parameters of beta and
  gamma do not have estimated means because they are nonlinear parameters (i.e., the
  logistic function's first-order partial derivative with respect to each of those two
  parameters contains those two parameters. )
  mxPath(from = 'one', to = c('theta', 'alpha'), free = c(TRUE, TRUE), arrows = 1,
         labels = c('theta_fixed', 'alpha_fixed'), lbound = 0, ubound = 7,
         values = c(1, 1)),

  #Functional constraints (needed to estimate mean values of fixed-effect parameters)
```

```
  mxMatrix(type = 'Full', nrow = length(manifest_vars), ncol = 1, free = TRUE,
           labels = 'theta_fixed', name = 't', values = 1, lbound = 0,  ubound = 7),
  mxMatrix(type = 'Full', nrow = length(manifest_vars), ncol = 1, free = TRUE,
           labels = 'alpha_fixed', name = 'a', values = 1, lbound = 0,  ubound = 7),
  mxMatrix(type = 'Full', nrow = length(manifest_vars), ncol = 1, free = TRUE,
           labels = 'beta_fixed', name = 'b', values = 1, lbound = 1, ubound = 360),
  mxMatrix(type = 'Full', nrow = length(manifest_vars), ncol = 1, free = TRUE,
           labels = 'gamma_fixed', name = 'g', values = 1, lbound = 1, ubound = 360),

  mxMatrix(type = 'Full', nrow = length(manifest_vars), ncol = 1, free = FALSE,
           values = measurement_days, name = 'time'),

  #Algebra specifying first-order partial derivatives;
  mxAlgebra(expression = 1 - 1/(1 + exp((b - time)/g)), name="Tl"),
  mxAlgebra(expression = 1/(1 + exp((b - time)/g)), name = 'Al'),

  mxAlgebra(expression = -((a - t) * (exp((b - time)/g) * (1/g))/(1 + exp((b -
  time)/g))^2), name = 'Bl'),
  mxAlgebra(expression =  (a - t) * (exp((b - time)/g) * ((b - time)/g^2))/(1 + exp((b
  -time)/g))^2, name = 'Gl'),

  #Factor loadings; all fixed and, importantly, constrained to change according to
  their partial derivatives (i.e., nonlinear functions)
  mxPath(from = 'theta', to = manifest_vars, arrows=1, free=FALSE,
         labels = sprintf(fmt = 'Tl[%d,1]', 1:length(manifest_vars))),
  mxPath(from = 'alpha', to = manifest_vars, arrows=1, free=FALSE,
         labels = sprintf(fmt = 'Al[%d,1]', 1:length(manifest_vars))),
  mxPath(from='beta', to = manifest_vars, arrows=1,  free=FALSE,
         labels =  sprintf(fmt = 'Bl[%d,1]', 1:length(manifest_vars))),
  mxPath(from='gamma', to = manifest_vars, arrows=1,  free=FALSE,
         labels =  sprintf(fmt = 'Gl[%d,1]', 1:length(manifest_vars))),

  #Fit function used to estimate free parameter values.
  mxFitFunctionML(vector = FALSE)
)

#Use starting value function from OpenMx to generate good starting values (uses
weighted least squares)
latent_growth_model <- mxAutoStart(model = latent_growth_model)

#Fit model using mxTryHard(). Increases probability of convergence by attempting model
convergence by randomly shifting starting values.
model_results <- mxTryHard(latent_growth_model)
```