Is Timing Everything? Measurement Timing and the Ability to Accurately Model Longitudinal Data

by

Sebastian L.V. Sciarra

A Thesis
presented to
The University of Guelph

In partial fulfilment of requirements
for the degree of

Doctorate of Philosophy

in

Psychology

Guelph, Ontario, Canada

ABSTRACT

IS TIMING EVERYTHING? MEASUREMENT TIMING AND THE ABILITY TO ACCURATELY MODEL LONGITUDINAL DATA

Sebastian L.V. Sciarra
University of Guelph, 2022

Advisor(s):
David Stanley

The preface pretty much says it all. This is additional content. The preface pretty much says it all. This is additional content. The preface pretty much says it all. This is additional content. The preface pretty much says it all. This is additional content. The preface pretty much says it all. This is additional content.

DEDICATION

You can have a dedication here if you wish. You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.

ACKNOWLEDGEMENTS

I want to thank a few people.You can have a dedication here if you wish. You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.You can have a dedication here if you wish.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

1 thesisdown::thesis_gitbook: default

Placeholder

1.1 The Need to Conduct Longitudinal Research

1.2 Understanding Patterns of Change That Emerge Over Time

1.3 Challenges Involved in Conducting Longitudinal Research

1.3.1 Number of Measurements

1.3.2 Spacing of Measurements

1.3.3 Time Structuredness

1.3.3.1 Time-Structured Data

1.3.3.2 Time-Unstructured Data

1.3.4 Summary

1.4 Using Simulations To Assess Modelling Accuracy

1.5 Systematic Review of Simulation Literature

1.5.1 Systematic Review Methodology

1.5.2 Systematic Review Results

1.5.3 Next Steps

1.6 Methods of Modelling Nonlinear Patterns of Change Over Time

1.7 Overview of Simulation Experiments

2 Experiment 1

Placeholder

2.1 Methods

2.1.1 Variables Used in Simulation Experiment

2.1.1.1 Independent Variables

2.1.1.1.1 Spacing of Measurements

2.1.1.1.2 Number of Measurements

2.1.1.1.3 Population Values Set for The Fixed-Effect Days-to-Halfway Elevation Parameter β_{fixed} (Nature of Change)

2.1.1.2 Constants

2.1.1.3 Dependent Variables

2.1.1.3.1 Convergence Success Rate

2.1.1.3.2 Bias

2.1.1.3.3 Precision

2.1.2 Overview of Data Generation

2.1.2.1 Data Generation

2.1.2.1.1 Function Used to Generate Each Data Set

2.1.2.1.2 Population Values Used for Function Parameters

2.1.3 Modelling of Each Generated Data Set

2.1.4 Analysis of Data Modelling Output and Accompanying Visualizations

2.1.4.1 Analysis of Convergence Success Rate

2.1.4.2 Analysis and Visualization of Bias

2.1.4.3 Analysis and Visualization of Precision

2.1.4.3.1 Effect Size Computation for Precision

2.2 Results and Discussion

2.2.1 Framework for Interpreting Results

2.2.2 Pre-Processing of Data and Model Convergence

3.1 Methods

3.1.1 Variables Used in Simulation Experiment

3.1.1.1 Independent Variables

3.1.1.1.1 Spacing of Measurements

3.1.1.1.2 Number of Measurements

3.1.1.1.3 Sample Size

3.1.1.2 Constants

3.1.1.3 Dependent Variables

3.1.1.3.1 Convergence Success Rate

3.1.1.3.2 Bias

3.1.1.3.3 Precision

3.1.2 Overview of Data Generation

3.1.3 Modelling of Each Generated Data Set

3.1.4 Analysis of Data Modelling Output and Accompanying Visualizations

3.2 Results and Discussion

3.2.1 Framework for Interpreting Results

3.2.2 Pre-Processing of Data and Model Convergence

3.2.3 Equal Spacing

3.2.3.1 Bias

3.2.3.2 Precision

3.2.3.3 Qualitative Description

3.2.3.4 Summary of Results

3.2.4 Time-Interval Increasing Spacing

3.2.4.0.1 Bias

3.2.4.0.2 Precision

4.1 Methods

4.1.1 Variables Used in Simulation Experiment

4.1.1.1 Independent Variables

4.1.1.1.1 Number of Measurements

4.1.1.1.2 Sample Size

4.1.1.1.3 Time Structuredness

4.1.1.2 Constants

4.1.1.3 Dependent Variables

4.1.1.3.1 Convergence Success Rate

4.1.1.3.2 Bias

4.1.1.3.3 Precision

4.1.2 Overview of Data Generation

4.1.2.0.1 Simulation Procedure for Time Structuredness

4.1.3 Modelling of Each Generated Data Set

4.1.4 Analysis of Data Modelling Output and Accompanying Visualizations

4.2 Results and Discussion

4.2.1 Framework for Interpreting Results

4.2.2 Pre-Processing of Data and Model Convergence

4.2.3 Time-Structured Data

4.2.3.0.1 Bias

4.2.3.0.2 Precision

4.2.3.0.3 Qualitative Description

4.2.3.1 Summary of Results

4.2.4 Time-Unstructured Data Characterized by a Fast Response Rate

4.2.4.0.1 Bias

5 References

Appendix A: OpenMx Code for Structured Latent Growth Curve Model Used in Simulation Experiments

Code Block A.1

OpenMx Code for Structured Latent Growth Curve Model

```
#Manifest variable names (i.e., names of columns cotaining data points at each time point [e.g., ])
manifest_vars <- nonlinSims:::extract_manifest_var_names(data_wide = data_wide)

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

#initial checks
  tryCatch(expr = model_name, error = function(e) {message("Error: model_name is not a character vector")})

  manifest_vars <- extract_manifest_var_names(data_wide = data_wide)
  latent_vars <- c('theta', 'alpha', 'beta', 'gamma')
  measurement_days <- measurement_days

  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)
                   mxPath(from = manifest_vars,
                          arrows=2, free=TRUE,  labels='epsilon', values = 1, lbound = 0),

                   #Latent variable covariances and variances
                   mxPath(from = latent_vars,
                          connect='unique.pairs', arrows=2,
                          #aa(diff_rand), ab(cov_diff_beta), ac(cov_diff_gamma), bb(beta_rand), bc(var_beta_gamma), cc(gamma_rand)
                          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 nonlinear
                    parameters of beta and gamma do not have estimated means
                    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
                    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 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))),

                    mxFitFunctionML(vector = FALSE)

  )
```

```
names(data_wide)[2:8]
```

```
[1] "obs_score_0"   "obs_score_60"  "obs_score_120" "obs_score_180"
[5] "obs_score_240" "obs_score_300" "obs_score_360"
```

Appendix B: OpenMx Code for Structured Latent Growth Curve Model With Definition Variables

Code Block **??** OpenMx Code for Structured Latent Growth Curve Model With Definition Variables OpenMx Code for Structured Latent Growth Curve Model With Definition Variables section on Appendix B