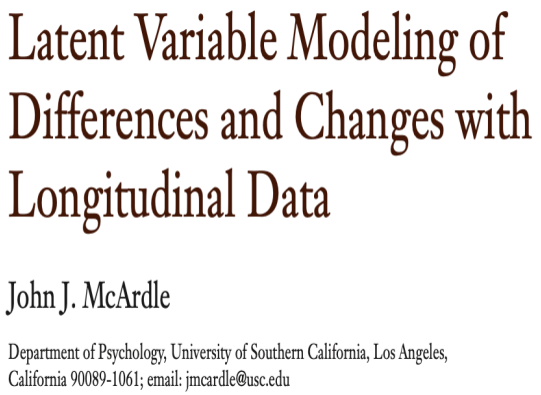
Dual Change Score Model

Rich Jones

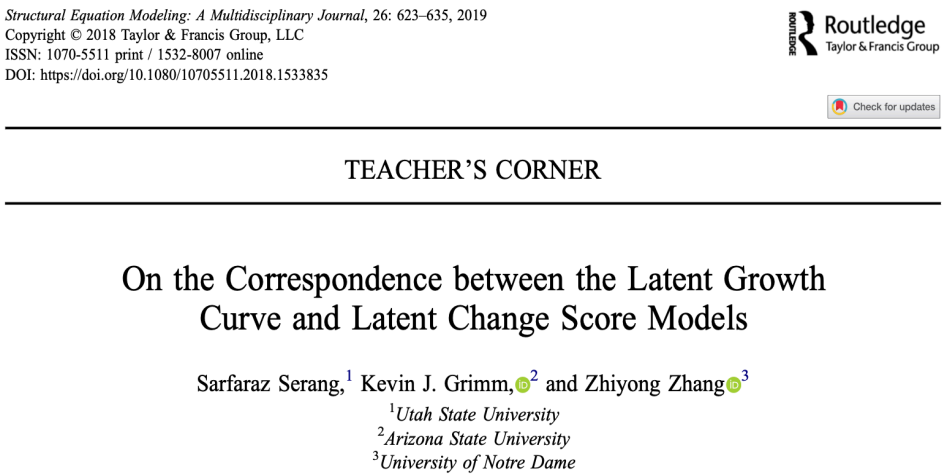
2025-09-04

Where to learn more


Latent Variable Modeling of Differences and Changes with Longitudinal Data

John J. McArdle

Department of Psychology, University of Southern California, Los Angeles, California 90089-1061; email: jmcardle@usc.edu


Structural Equation Modeling: A Multidisciplinary Journal, 26: 623–635, 2019

ISSN: 1070-5511 print / 1532-8007 online
DOI: https://doi.org/10.1080/10705511.2018.1533835

Routledge
Taylor & Francis Group

Check for updates

TEACHER'S CORNER

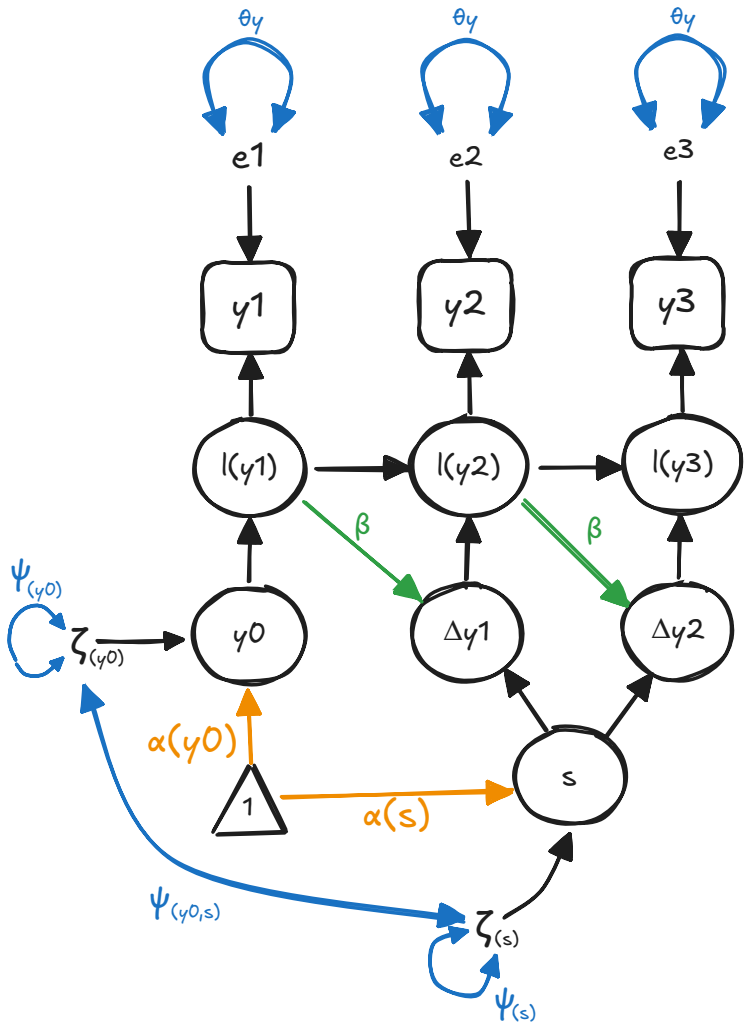
On the Correspondence between the Latent Growth Curve and Latent Change Score Models

Sarfaraz Serang,[1] Kevin J. Grimm,[2] and Zhiyong Zhang[3]
[1]*Utah State University*
[2]*Arizona State University*
[3]*University of Notre Dame*

McArdle, J. (2009). Latent variable modeling of differences and changes with longitudinal data. Annual Review of Psychology, 60, 577-605.

Serang, S., Grimm, K. J., & Zhang, Z. (2019). On the correspondence between the latent growth curve and latent change score models. Structural Equation Modeling: A Multidisciplinary Journal, 26(4), 623-635.

Dual Change Score Model



Notes: Omitted (residual) variance terms are assumed to be 0, any meanstructure part not illustrated is assumed to be 0, like-labeled parameters (i.e., θ_y, β) are assumed to be equal, any omitted path is assumed to be 0, any unlabeled path is assumed to be 1.

$$l_{y1} = y_0$$

$$l_{yt} = l_{y(t-1)} + \Delta y_{(t-1)}, \text{ where } t>1$$

$$\Delta_{yt} = l_{yt}\beta + s$$

the "change equation"

$$l_{yt} = l_{y(t-1)} + l_{y(t-1)}\beta + s, \text{ where } t>1$$

$$= l_{y(t-1)}(1 + \beta) + s, \text{ where } t>1$$

$$y_1 = l_{y1} + e_1 = y_0 + e_1$$

$$y_t = l_{y(t-1)}(1 + \beta) + s + e, \text{ where } t>1$$

this is one more parameter than a linear latent growth curve model with homoskesdasticity of item residuals.

Parameters = 7

- 1 residual variance (θ)
- 3 variance/covariance (Ψ)
- 1 structural regression (β)
- 2 latent variable means (α)

Pieces of information

- 6 variances/covariances
- 3 observed means

therefore,
(6 + 3) - 7 = 2 degrees of freedom

Growth trajectories do not (necessarily) describe linear change: growth is both proportional (approximating exponential) [(1+β)^t] and constant (linear) [s].

$$E(y_t) = (1 + \beta)^{(t-1)}\left(E(y_0) + \frac{s}{\beta}\right) - \frac{s}{\beta}$$

For slopes (derivatives) at discrete time points t (t>1) report:

$$\Delta E(y_t) = \beta[E(y_{t-1})] + s = E(\Delta y_t)$$

which are explicitly modeled given the specification we describe on the following slides.

Special case

If $\beta = 0$, $E(y_t) = E(y_0) + (t-1)\,s$

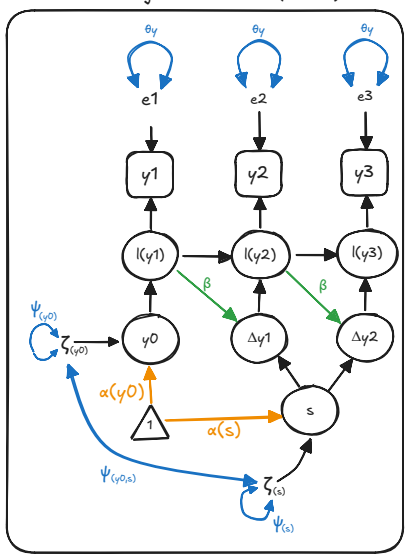
Shape of the curve driven by β

β	Equilibrium	Implication	Typical trajectory
$\beta > 0$	Exists but unstable	Proportional change, exponential growth or decay away from a equilibrium point ($-s/b$)	Moves away from equilibrium, monotone, no oscillation
$\beta = 0$	None (unless $s = 0$)	Constant change, linear growth or decay	Straight line
$-1 < \beta < 0$	Stable $y^* = -s/b$	Proportional change, exponential monotone decay toward an equilibrium point	Monotone decay toward equilibrium
$-2 < \beta < -1$	Stable	Converges to equilibrium, oscillates around ($-s/b$)	Damped oscillation toward equilibrium
$\beta < -2$	Unstable	Diverges with oscillation	Growing flips around equilibrium

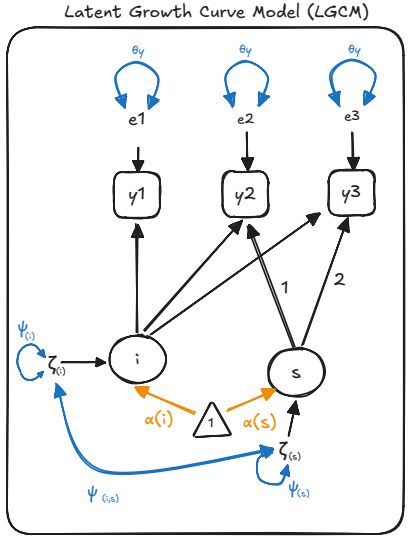
Equilibrium value for y is $y^* = -s/\beta$ and if y ever reached this value the change would be 0 and y would remain at this value. Related to an asymptote. Unstable equilibrium means that if y is perturbed away from this value it will move further away. Stable means that if y is perturbed away from this value it will move back toward it.

Baseline–Change Association

Dual Change Score Model (DCSM)



Latent Growth Curve Model (LGCM)



In a Dual Chnge Score Model (DCSM), the baseline-change association has a dual nature: a between-person component ($\mathrm{Cov}(y_0, s)$) and a within-person component (β). The random effects covariance $\mathrm{Cov}(y_0, s)$ parameter captures the correlation between the baseline level (y_0) and constant change component (s). Although they are both random effects, this effect is at the "between-person" level in a multilevel framework. The covariance $\mathrm{Cov}(y_0, s)$ tells us whether individuals who start higher also tend to have faster/slower long-run slopes. This is the between-person piece, describing population heterogeneity.

The dynamic feedback parameter (β) captures the structural regression of a person's current level on their own subsequent change. It is interpreted as a within-person dynamic: does being higher now predict more or less change next time? Unlike the LGCM, where such dynamics are absorbed into the intercept–slope covariance, the DCSM models them explicitly. The recursion multiplier means that even though β is fixed across individuals, the effect of β on change varies as a function of the individual's prior level of y.

In contrast, unconditional linear growth curve model (LGCM), the correlation between baseline and change is captured in one model parameter: the covariance between intercept and slope ($\mathrm{Cov}(i, s)$). This is a between-persons effect.

However in practice, the constant change component and the proportional change parameter in the DCSM are often highly correlated, sometimes nearly collinear. This means the dual nature of baseline–change correlation is not always empirically separable. More time points, smaller proportional change values, and constant change means near zero reduce the problem; few time points and extreme constant change means exacerbate it (Jacobucci R et al. Structural Equation Modeling: A Multidisciplinary Journal. 2019:26(6);924-930).

LDSM.EXE

https://www.psychstat.org/us/article.php/38.htm

Z. Zhang

Home | Control Panel | Login | Register | Search | Submit | Logout | About me | Contact me

My Research

- BAUW
- BMEM
- Consulting & Collaboration
- Dynamic Models
- MedCI
- R
- SAS GCM power
- SAS WinBUGS
- WebStatR
- Publications
- Programs
- In Progress
- Favorites

Google

Web
This Site
Search

Home > A c++ program to generate the Mplus bivariate latent difference score model codes

A c++ program to generate the Mplus bivariate latent difference score model codes

2005-08-15 Zhang, Z. Read: 14743 times

Cite this page: Zhang, Z. (2005). *A c++ program to generate the Mplus bivariate latent difference score model codes.* Retrieved September 5, 2025, from https://www.psychstat.org/us/article.php/38.htm.

A c++ program to generate the Mplus latent difference score model codes

To download the program, click here.

This dos program can be used to generate the Mplus codes for univariate and bivariate latent difference score models (McArdle, 2001, etc.).

To run this program, four input parameters are needed.

1. The file name to save the codes.

2. Choose univariate model or bivariate model.

3. The name of the data file.

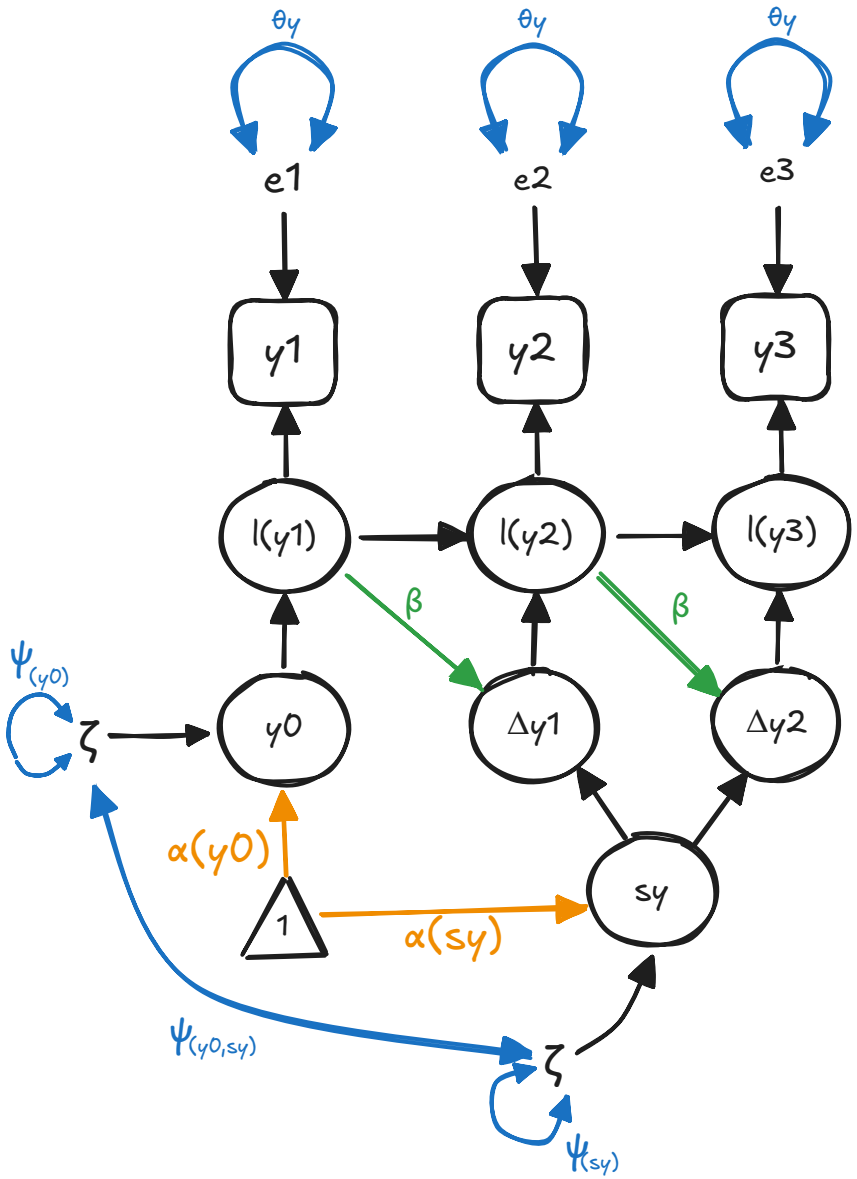
4. How many occasions in your data.

For the univariate model, the data is organized like,

y1 y2 y3 y4..

LDSM.EXE is a Windows program that generates Mplus code for latent difference score models (LDSM), including dual change score models (DCSM).

```
!*********************************************!
!This M+ latent difference score model scripts!
!are generated by LDSM generator.             !
!*********************************************!
MODEL:
!For the variable y
!Observed variables and latent level
ly1 by y1 @1;
ly2 by y2 @1;
ly3 by y3 @1;
!Autoregressive part
ly2 on ly1 @1;
ly3 on ly2 @1;
!Difference score on latent level
dy1 by ly2 @1;
dy2 by ly3 @1;
!Auto-proportion of difference score on level
dy1 on ly1 * (1);
dy2 on ly2 * (1);
!Model relationship between slope and ds
sy by dy1 @1;
sy by dy2 @1;
y0 by ly1 @1;
!Set the means and variance to be 0
[y1@0]; [ly1@0]; [dy1@0]; ly1@0; dy1@0;
[y2@0]; [ly2@0]; [dy2@0]; ly2@0; dy2@0;
[y3@0]; [ly3@0];ly3@0;
!Estimate means and variances for intercept and slope
[y0 sy];
y0 sy;
!Set all item-level residuals to be equal
y1* (2);
y2* (2);
y3* (2);
```



DCSM of MSQTOT in EPESE

- MSQTOT assessed at baseline, +3 years, and +6 years
- Scores on 0-6 scale, higher score is more correct answers

```
Mplus VERSION 8.11 (Mac)
MUTHEN & MUTHEN
09/08/2025   7:02 AM

INPUT INSTRUCTIONS

  TITLE:    DCSM, MSQ sum scores from EPESE
            Input from LDSM.EXE
            y1, y2, and y3 are MSQTOT at wave 1, 4, and 7

  DATA:     FILE = ex0201.dat;

  VARIABLE: NAMES = y1 y2 y3 ; ! msqtot1 msqtot4 msqtot7 age black ;
            MISSING  = ALL (-9999) ;

  OUTPUT:   TECH1 ;
            TECH4 ;

  SAVEDATA: SAVE = FSCORES ;
            FILE = ex0399.dat ;

  MODEL:
            !For the variable y
```

Scroll down to see the entire Mplus output

Using R to explore the output and results

- MplusAutomation package commands to extract information about the model and results
- Load the savedata file and plot the estimated latent trajectories

Load the data in the SAVEDATA file

I have a program to do this on my GitHub (rnj0nes/RNJmisc::runmplus_load_savedata.R)

This program will read the OUT file, find the name and contents of the SAVEDATA file, and read that file into R.

Also `contents.r` is a program to provide a summary of a data frame, and `eme.r` will be discussed in the Appendix.

```
library(dplyr)
library(tidyr)
library(ggplot2)
# RNJMisc programs
f1 <- "runmplus_load_savedata.R"
f2 <- "contents.r"
f3 <- "eme.r"
rnjmisc_url <- "https://raw.githubusercontent.com/rnj0nes/RNJmisc/master/"
devtools::source_url(paste0(rnjmisc_url, f1))
devtools::source_url(paste0(rnjmisc_url, f2))
devtools::source_url(paste0(rnjmisc_url, f3))
```

```r
# Read SAVEDATA from LDSM
res <- runmplus_load_savedata("ex0399.out")
df <- res$data
```

`contents` of `df` (the SAVEDATA file)

Variable	N	Mean	SD	Min	Max
y1	13698	4.768	1.273	0.000	6.000
y2	10899	4.732	1.350	0.000	6.000
y3	8169	4.569	1.426	0.000	6.000
ly1	14059	4.782	0.796	1.474	5.765
ly1_se	14059	0.513	0.071	0.461	0.656
ly2	14059	4.631	0.888	0.992	5.749
ly2_se	14059	0.529	0.109	0.445	0.723
ly3	14059	4.386	1.046	0.213	5.723
ly3_se	14059	0.705	0.156	0.580	0.970
dy1	14059	-0.151	0.118	-0.674	0.107
dy1_se	14059	0.210	0.018	0.195	0.237
dy2	14059	-0.245	0.190	-1.090	0.173
dy2_se	14059	0.340	0.029	0.316	0.384
sy	14059	-3.103	0.411	-3.574	-1.392
sy_se	14059	0.398	0.017	0.389	0.496
y0	14059	4.782	0.796	1.474	5.765
y0_se	14059	0.513	0.071	0.461	0.656

y1-y3 are the observed MSQTOT scores at waves 1, 4, and 7. Everything else is a latent variable or factor score estimate, or a derivation of a factor score (the standard errors, XXX_se). Notice the sample sizes show missing data by death and attrition for the observed variables, but not for the latent variables (which are estimated for all persons).

The `ly` variables are the estimated true scores at each time point. Their `_se` variables are estimates of the standard errors of measurement for these factor scores, and are analogous to the standard deviations of the posterior distribution of the factor scores given the model and the data, where the factor score estimates are the means of these posterior distributions.

The `dy` variables are estimated latent difference scores, `y0` is the estimated intercept, and `sy` is the estimated slope factor score.

Plots

We will make two plots. The first is a spaghetti plot of the estimated true scores (`ly1`, `ly2`, `ly3`) over time, with the mean curve overlaid. The second plot is a histogram of the person × occasion residuals (the differences between the observed scores and the estimated true scores).

Both of these will require data in “long” format, so we will reshape the data frame.

```r
# Reshape wide to long
# df$ly1, ly2, and ly3 are the expected values
df_long <- df %>%
   select(y1:y3, ly1, ly2, ly3) %>%
   mutate(id = row_number()) %>%          # step 1: row identifier
   pivot_longer(
      cols = c(y1:y3, ly1:ly3),           # step 2: reshape
      names_to = c(".value", "obs"),      # split into 'y'/'ly' and 'obs'
      names_pattern = "([a-z]+)([0-9]+)"
   ) %>%
   arrange(id, obs)
head(df_long)
```

```
# A tibble: 6 × 4
     id obs       y    ly
  <int> <chr> <dbl> <dbl>
1     1 1         4  4.36
2     1 2        NA  4.18
3     1 3        NA  3.89
4     2 1         4  4.64
5     2 2         5  4.51
6     2 3        NA  4.31
```

With the reshaped data we can easily compute correlations and R-squared values.

```r
vars <- c("y", "ly")
cor_matrix <- cor(df_long[vars], use = "pairwise.complete.obs")
print(format(round(cor_matrix, 2), nsmall = 2, trim = TRUE))
```

```
    y      ly
y  "1.00" "0.89"
ly "0.89" "1.00"
```

```r
r_sq <- cor_matrix["y", "ly"]^2
cat("R-squared:", round(r_sq, 2), "\n")
```

```
R-squared: 0.78
```

In the spaghetti plot we will overlay the mean curve from the data (in red) and the expected mean curve implied by the estimated parameters of the model (in blue). We can compute the expected means from the estimated parameters in the Mplus output. This provides a check on my understanding of the model, as these two curves should overlap exactly. In this code I use model parameters copied from the Mplus output, but it would have been better to extract them using MplusAutomation functions. I will demonstrate that in an appendix.

```r
# ensure obs is numeric (pivot_longer often makes it character)
df_long <- df_long %>% mutate(obs = as.integer(obs))

# Extract from Mplus output
mean_y0    <- 4.782  # mean of "y0" from Mplus output
beta       <- 0.617  # beta from Mplus output (Dy on Ly)
mean_sy    <- -3.103 # mean of "sy" from Mplus output
# Three different ways to compute the implied means all are equal
# implied_y1 <- mean_y0 # mean of "Y0" from Mplus output
# implied_y2 <- implied_y1 + beta*implied_y1 + mean_sy
# another way
# implied_y3 <- implied_y2 * (1 + beta) + mean_sy
# closed form showing proportional (~exponential) and linear change
implied_y1 <- (1 + beta)^(1-1) * (mean_y0 + mean_sy/beta) - (mean_sy/beta)
implied_y2 <- (1 + beta)^(2-1) * (mean_y0 + mean_sy/beta) - (mean_sy/beta)
implied_y3 <- (1 + beta)^(3-1) * (mean_y0 + mean_sy/beta) - (mean_sy/beta)

# expected means data
expected_df <- tibble::tibble(
   obs = c(1L, 2L, 3L),
   mu  = c(implied_y1, implied_y2 , implied_y3)
)
```

Scroll down to see the entire code chunk

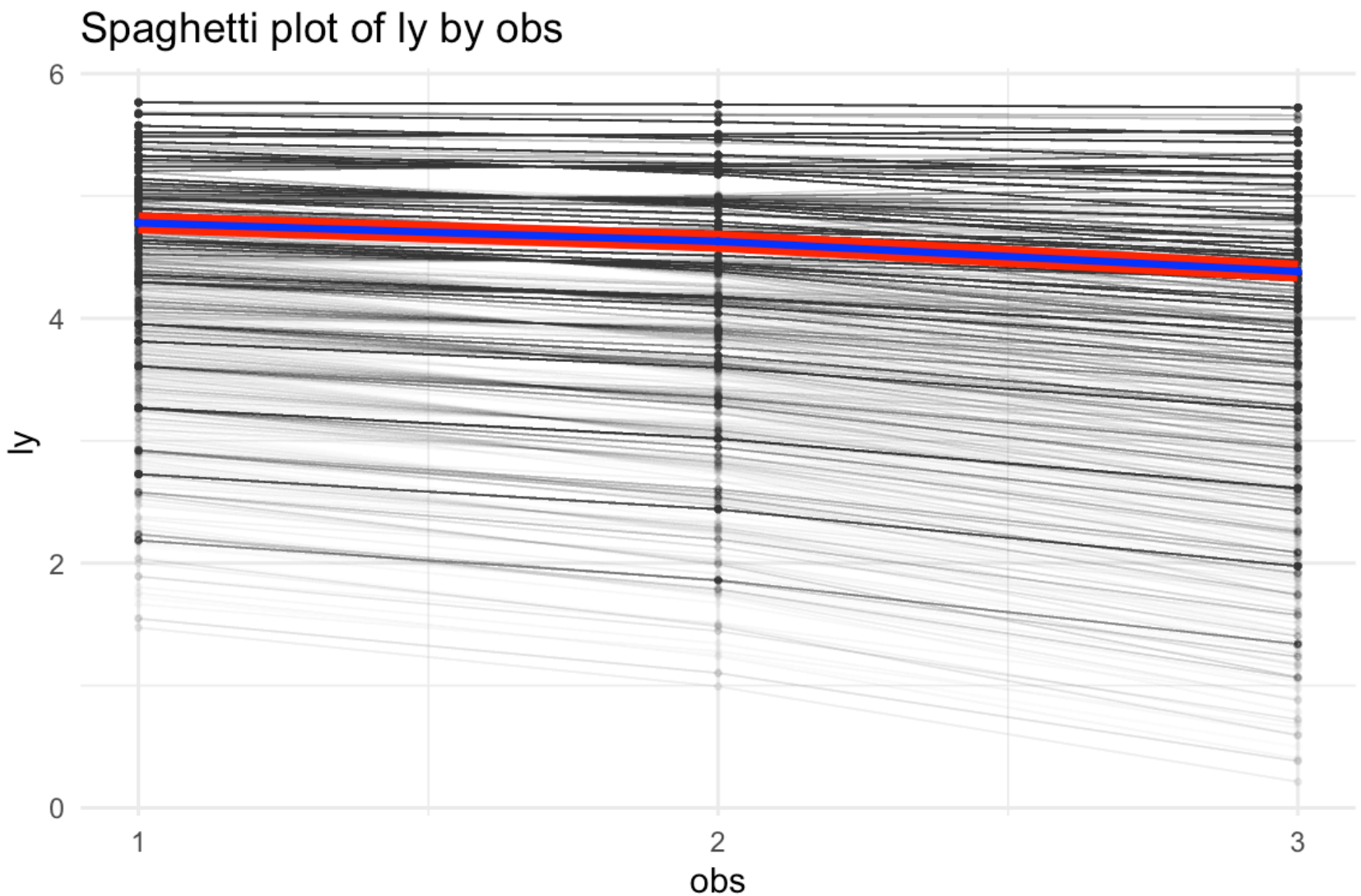
Code for the spaghetti plot using ggplot2.

```r
spagplot <- ggplot(df_long, aes(x = obs, y = ly, group = id)) +
   geom_line(alpha = 0.01, linewidth = 0.3) + # spaghetti
   geom_point(alpha = 0.01, size = 0.6) +     # optional points

   # data mean curve (red)
   geom_line(
      data = mean_curve,
      aes(x = obs, y = ly_bar, group = 1),
      linewidth = 3.1, color = "red", inherit.aes = FALSE
   ) +

   # expected mean curve (blue)
   geom_line(
      data = expected_df,
      aes(x = obs, y = mu, group = 1),
      linewidth = 1.1, color = "blue", inherit.aes = FALSE
   ) +
   geom_point(
```

Scroll down to see the entire code chunk

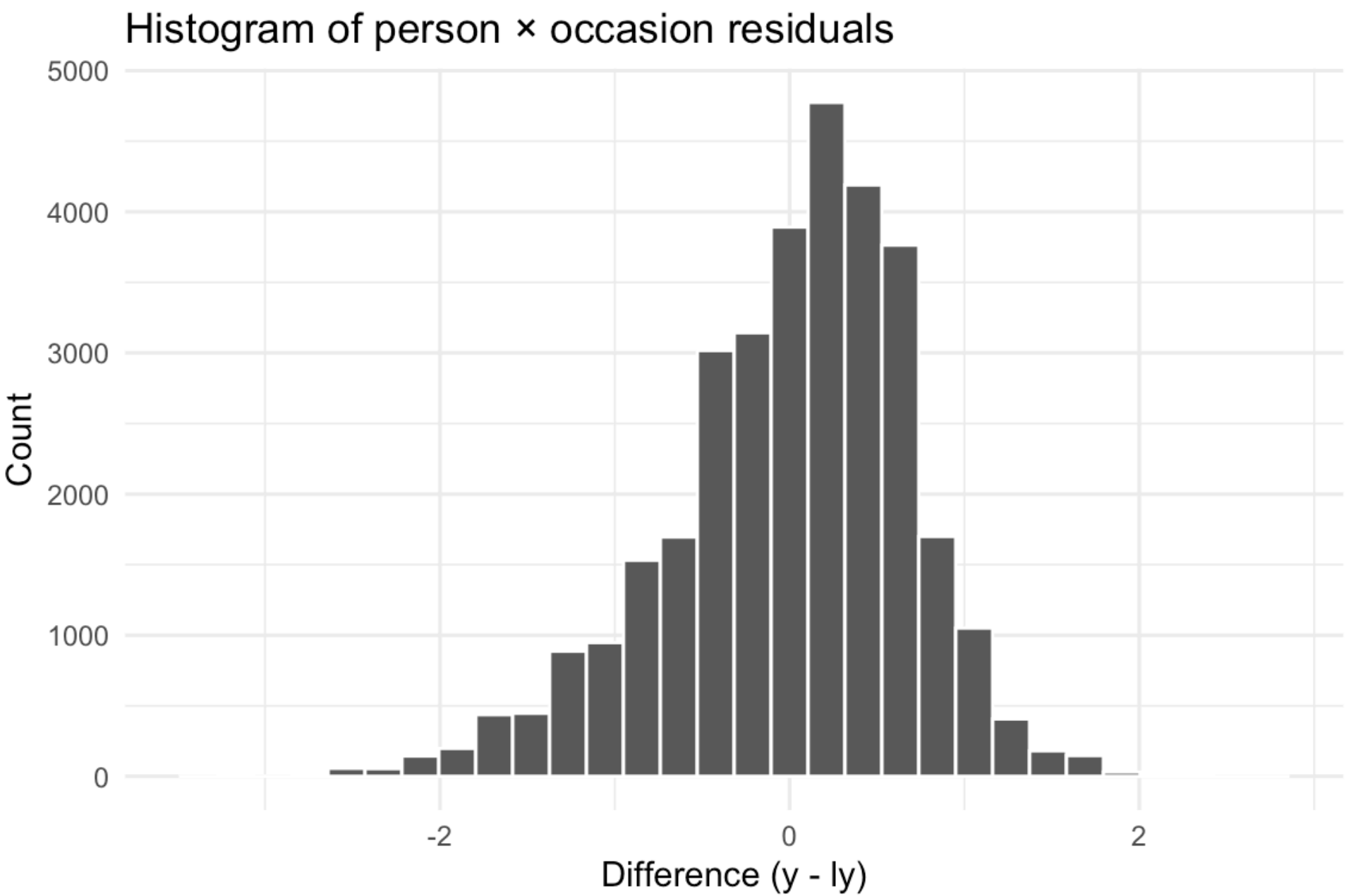


Each line is one persons estimated trajectory of the “true score” (ly) over time. The thick red line is the mean of the estimated true scores at each time point. The blue line is the expected mean trajectory implied by the estimated parameters of the model. The fact that these overlap exactly provides a check on my understanding of the model. The observed and model-implied means are identical because there are no covariates and there is no missing data.

Code for the plot of person × occasion residuals.

```r
df_long <- df_long %>%
   mutate(diff = y - ly)   # new variable: difference

residhist <- ggplot(df_long, aes(x = diff)) +
   geom_histogram(bins = 30, color = "white") +
   labs(x = "Difference (y - ly)", y = "Count",
        title = "Histogram of person × occasion residuals") +
   theme_minimal()
```


Histogram of person × occasion residuals
5000
4000
3000
2000
1000
0
Count
-2
0
2
Difference (y - ly)

Appendix: Extracting parameters from Mplus output

Here we show how to extract the estimated parameters from the Mplus output using MplusAutomation functions. This is better than copying and pasting numbers from the output, as it is less error-prone and can be automated.

Even better would be to run Mplus from R using MplusAutomation functions, and use the H5RESULTS to get parameter estimates, but that is for another time.

Assume we have already run the model and have the output file `ex0399.out`.

```r
library(MplusAutomation)
# read the Mplus output file
out <- MplusAutomation::readModels("ex0399.out")
# extract and view the parameter estimates
params <- out$parameters$unstandardized
head(params)
```

```
  paramHeader param est se est_se pval
1      LY1.BY    Y1   1  0    999  999
2      LY2.BY    Y2   1  0    999  999
3      LY3.BY    Y3   1  0    999  999
4      DY1.BY   LY2   1  0    999  999
5      DY2.BY   LY3   1  0    999  999
6       SY.BY   DY1   1  0    999  999
```

Let's just look at the parameters that are really estimates, defined by having a standard error.

```
# let's at those entries in params with SE > 0
params |> dplyr::filter(se>0) |> head(n=sum(params$se > 0))
```

```
           paramHeader param    est    se  est_se  pval
1            DY1.ON   LY1  0.617 0.144   4.274 0.000
2            DY2.ON   LY2  0.617 0.144   4.274 0.000
3           Y0.WITH    SY -0.497 0.134  -3.716 0.000
4             Means    SY -3.103 0.682  -4.552 0.000
5             Means    Y0  4.782 0.011 448.388 0.000
6         Variances    SY  0.328 0.139   2.353 0.019
7         Variances    Y0  0.902 0.021  43.107 0.000
8  Residual.Variances    Y1  0.759 0.011  66.171 0.000
9  Residual.Variances    Y2  0.759 0.011  66.171 0.000
10 Residual.Variances    Y3  0.759 0.011  66.171 0.000
```

Pull the mean of y0 into an object in R:

```
mean_y0 <- params |> dplyr::filter(paramHeader == "Means" & param == "Y0") |> pull(est)
mean_y0
```

```
[1] 4.782
```

That code is not hard but I do have a helper function to make it easier. This function is in my RNJmisc repo on GitHub, and it's called `eme.r`.

```
# this creates an object in the R environment called means_y0 (if
# there is a matching entry in `params`. Note the case insensitivity
# of the function. Syntax is eme(params_object, "paramHeader", "param")
eme(params, "means", "y0")
```

```
   paramHeader param   est    se  est_se pval
15       Means    Y0 4.782 0.011 448.388    0
```

```
means_y0$est
```

```
[1] 4.782
```

```
means_y0$se
```

```
[1] 0.011
```

```
# and if your guess at the name is not close you'll get a message
# and some help
eme(params, "y0_with","sy")
```

```
No exact match for paramHeader = 'y0_with', param = 'sy'.
Did you mean one of these (original case & punctuation preserved)?
```

```
 paramHeader param
     Y0.WITH    SY
```

(fin)