Handling Item Clustering using 2PL IRT Modeling in an SEM Framework: A Demonstration with PISA 2012 Computerized Math Problems

Daeun Im

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Education

University of Washington

2022

Committee:

Min Li, Ph.D., Chair, Professor, Measurement and Statistics

Elizabeth Sanders, Ph.D., Associate Professor, Measurement and Statistics

Program Authorized to Offer Degree:

Education

Abstract

The current study extends earlier work by Costa et al. (2021) on a method for incorporating item-varying response times into a binary latent trait model using item-level data from the 2012 Programme for International Student Achievement (PISA) U.S. dataset. Specifically, we demonstrate a 2-step factor analytic approach that incorporates three item clusters as nuisance "method" factors with 10 math problem solving items. Step 1 involves estimating separate latent speed and latent trait factor models that constrain the latent variable scales to unit normal (to estimate measurement model item parameters), and step 2 involves estimating a joint model using step 1 item parameter estimates as starting values in order to better estimate the structural regression. Despite missing data issues, results showed that scale reliability estimates improved in step 1 for both latent speed and latent trait variables when clustering was taken into account, and further, that the latent speed reliability estimate improved in the joint model in step 2. However, we did not find improvement in the latent trait ability estimate when taking latent speed into account. Our results suggest that the factor model 2-step method is a viable alternative to high-dimensional item response theory (IRT) model parameterization (which is the case when items are clustered with common prompts), but that taking latent speed into account may not necessarily improve precision in measuring the focal latent trait.

Keywords: IRT-SEM, item cluster, multidimensionality, process data, reliability

Handling Item Clustering using 2PL IRT Modeling in an SEM Framework: A Demonstration with PISA 2012 Computerized Math Problems

Moment-to-moment information about how test-takers respond to computerized assessment items, such as keystrokes, clicks, and response times, has the potential to be useful for inferring student thinking on the one hand, as well as for flagging problematic item features and validating assessments on the other hand (e.g., Xie et al., 2019a, 2019b). Broadly, these types of data are known as "process data" (Bergner & von Davier, 2019). As one example, Xie and colleagues (2019a) investigated student keystroke data to detect problematic instructions hindering students' comprehension, thereby improving assessment validity by reducing construct-irrelevant variance. Most recently, Costa et al. (2021) used item response time information to improve the measurement reliability of math skills using 10 math problem-solving items from the Programme for International Student Assessment (PISA) 2012 publicly available data. Specifically, they employed a 2-step factor analytic approach in which separate latent speed and trait models were fit in step 1, and then step 1 estimates were used as constraints in step 2 to estimate a model in which response times were parameterized to cross-load onto both factors (latent speed and latent trait). In the present study, we extend the modeling process in two ways: 1) by handling dependencies due to item clustering (three clusters of items), and 2) specifying a structural parameter to link the latent speed factor with the latent trait factor, rather than cross-loadings.

Process Data and Assessment Validity

Advances in computing technologies have changed the typical format of the educational assessments to be electronic rather than paper-and-pencil, and the recent COVID-19 pandemic further bolstered increased use of online assessments in particular (e.g., Tong, 2022). Although

the validity of electronic assessments can be complicated, one of the benefits of substituting paper-and-pencil tests with computerized testing has been that test-taker actions during the testing process can now be recorded for evaluative use. As one example, the collection of process data enabled the study of task creation, scoring procedures, and measurement models focused on task-solving strategies and approaches (Caliço, 2021).

Response time data in particular has been of interest to assessment researchers, particularly for use in increasing score validity (Dutilh et al., 2019). For example, Lu and Sirecci (2007) described how response time could be included in statistical models to increase the precision of scores in practical applications; however, this procedure could in theory be measuring construct irrelevant variance if response time is not part of the true skill being measured (e.g., if reading comprehension is the target skill, then how quickly students respond to a given item may be conflating text comprehension skill with working memory capacity). This said, Goldhammer and colleagues have found a consistent link between response time and performance across many domains; theoretically, respondents who deeply understand a subject matter should have quicker associations to draw from and therefore provide both a quicker and more correct response (Goldhammer, 2014; Goldhammer et al., 2015; Goldhammer & Klein Entink, 2011; Naumann & Goldhammer, 2017).

Novel Analytic Approaches for Incorporating Process Data in Item Response Modeling

Recently, Costa et al. (2021) demonstrated a 2-step factor analytic approach for incorporating item response time-on-task data to determine whether score reliability could be improved for 10 computerized math items from 2012 PISA, a large-scale computer-based assessment. More specifically, they treated response “speed” as a secondary latent variable (separate from math skills) in which response times were permitted to cross-load on both factors

for related items. One aspect missing from Costa et al. (2021) analysis approach was the issue of item clustering: the 10 PISA 2012 math items were actually based on three different prompts: three items related to music compact-disc production, four related to star points, and three related to body mass index. The first prompt included three items, the second included four items, and the third included three items. As is well known, responses based on the same prompt will naturally correlate with each other and therefore the residual errors among items with a common prompt will also correlate, thus violating the assumption of conditional independence for any latent variable model.

Current Study

The present paper extends the work of Costa et al. (2021) by incorporating both response time and item clustering as part of the modeling process, using two steps. Step 1 is similar to the earlier paper in that each latent construct is first estimated separately (i.e., one latent trait and one latent response time variable), but incorporates clustering as method factors, and then estimates from step 1 are used to fix parameters in step 2 in order to estimate quantities of interest. This said, our step 2 takes a different approach than Costa et al.'s approach in that it specifies a structural path between the trait and latent response time factors instead using cross-loading parameterization. In our view, this change in step 2 can more directly assess the relationship between the process data variable of response time, or "latent speed," and the focal latent trait (math skills). Last, we note that we focus on the U.S. case because a) it is one of the largest sub-samples in the data, b) because respondents from different countries may have different relations between latent speed and latent trait due to linguistic and/or cultural differences, and c) we wished to avoid use of an even more complex hierarchical structural equation model to demonstrate the new approach. Our research questions were specifically as follows.

1. Can we improve latent speed and trait reliability estimation by incorporating 'method' factors to handle item clustering?
2. What is the relationship between the latent speed and latent trait factors, controlling for item clustering?
3. Can we improve scale reliability in the latent trait after accounting for the latent speed?

Method

Data Source

The present study draws on publicly available data from the 2012 Programme for International Student Assessment (PISA) computer-based test. PISA is an international assessment that measures 15-year-old students' reading, mathematics, and science literacy every three years. The first computer-based mathematics assessment was implemented in 2012 as part of its fifth program edition, which included participants from 32 countries in total. For mathematics literacy (henceforth "math skills"), there were a total of 41 computer-based items, 10 of which included process data. The duration of the test is about 40 minutes for the computer-based section, which encompasses 20 main clusters of digital reading or problem-solving prompts (OECD, 2014a). The 10 items with process data included: CM015Q1, CM015Q2, CM015Q3, CM020Q1, CM020Q2, CM020Q3, CM020Q4, CM038Q3, CM038Q5, and CM038Q6. The original file codes answers to some of items as binary (correct/incorrect) and some as partial credit; for the purpose of the current study, partial credit items were recoded as fully correct or incorrect.

Our analytic sample included data from $N = 2572$ students from the U.S. subsample; we utilized full information maximum likelihood to estimate our models such that no students were dropped from analyses due to missing data. Tables 1-2 reports descriptive statistics for the item-

level response times (in seconds) and dichotomously coded item response accuracy levels, respectively. Translating seconds to minutes, the overall time to complete the items was approximately 14.76 minutes (*SD* = 0.47); the overall percentage correct across the items was relatively low, approximately 33% (*SD* = 19%).

Data Preparation

Data preparation was implemented in R. The raw data included two files: one containing student outcome data with country name, OECD number, student and book identifier numbers, and raw responses on each of the math items, and the other including students' clicks with start and end timestamps. We used three major steps to prepare the data for analysis. In Step 1, since we wanted to see the relationship between time duration and students' outcomes, we calculated the duration of time spent per item by individual students in the log data by subtracting start_event from end_event in the event column. In Step 2, we transformed partial credit scoring into the binary scoring format in the outcome dataset. In Step 3, the two datasets were merged by student identifier. Appendix A shows *R* programming details.

Data Analyses

We used a structural equation modeling (SEM) approach for data analyses. Factor models and item response theory (IRT) models both model the relationship between item characteristics and the latent trait being measured by those items. In most factor models for binary items, two parameters are estimated for each model: the item intercept (or threshold) and the item-factor relation (the slope). When estimating the item parameters, the latent factor mean and variance is usually constrained to a unit normal scale. Thereafter, the item parameters can be used as starting values to estimate the joint model.

Below is the IRT model specification of an SEM (e.g., Kamata & Bauer, 2008):

$$y_i^* = \nu_i + \lambda_i \xi + \varepsilon_i. \quad (1)$$

This model can be used as below comparison which can be used to estimate binary factors by setting τ_i as the *i*th item's "threshold" (Kamata & Bauer, 2008):

$$y_i = \begin{cases} 1 \text{ if } y_i^* \geq \tau_i \\ 0 \text{ if } y_i^* < \tau_i \end{cases}. \quad (2)$$

The IRT-SEM connection can be shown as:

$$p_i(y_i = 1|\xi) = f(\alpha_i \xi + \beta_i), \quad (3)$$

where α_i is item difficulty and β_i is item discrimination, as follows.

$$\alpha_i = \frac{\lambda_i}{q_i} \quad (4)$$

and

$$\beta_i = \frac{-\tau_i}{q_i} \quad (5)$$

Where q_i refers to the standard error of the i^{th} item, $V(\varepsilon_i)^{1/2}$ (Kamata & Bauer, 2008).

There were two major differences between the current study and the recent study by Costa et al. (2021): the first is that when we estimated parameters of the focal factor (math skills) and process data factor (response time, or "speed") separately, and for each, we specified three item cluster factors as 'method' factors. Then, in step 2, instead of allowing the item-level process data to be cross-loaded with the focal factor, we estimated the relationship between the focal and process data factors, and employed estimates from the previous step as starting values. In sum, we estimated three models as follows.

1. Model 1: Fit item **response time** data as the latent "speed" factor using a 4-factor CFA (1 factor for latent speed and 3 factors for clusters). Constrain latent factors as unit normal.

2. Model 2: Fit item **response accuracy** data as the latent "trait" factor using factor using a 4-factor CFA (1 factor for latent trait and 3 factors for clusters). Constrain latent factors as unit normal; use logistic model for estimating the latent trait.
3. Model 3: Fit joint 5-factor CFA. Use estimates from Models 1-2 as starting values.

For Models 2 and 3, we computed scale reliability using Raykov et al.'s (2010) generalization of omega for the trait factor, and for Models 1 and 3, we used McDonald's (1999) omega for the latent speed factor. All models were estimated using *Mplus Version 8*. Appendix B shows *Mplus* code for each of the three models.

Results

Fit Indices. Fit indices for Model 1, which modeled item response times to fit a 4-factor latent speed model (1 focal factor and 3 cluster factors) showed a significant chi-square test ($\chi^2(25) = 62.84$, $p < .001$) indicating a significant departure between the data and model-implied results, but still an acceptable fit to the data in other metrics: *RMSEA* = .06, *CFI* = .95, *TLI* = .91, and *SRMR* = .04. For Model 2, which was a non-linear model, we rely on the Likelihood ratio model fit test, which was not significant: $\chi^2(982) = 317.99$, $p = 1.000$; this indicated that Model 2 results did not deviate substantially from the observed data. Lastly, the fit for the joint model (Model 3), which also relied on a nonlinear model specification given our use of binary indicators, showed a similar well-fitting model: $\chi^2(967) = 962.645$, $p = 0.534$.

RQ1: Scale Reliability Estimates. Parameter estimates for Models 1-3 in which clustering was taken into account are shown in Table 3. Although not shown, we found that the reliability estimates for the same models in which clustering was not taken into account were worse: the latent speed reliability estimate when clustering was not taken into account was 0.78 in Models 1 and 3; but as can be seen in Table 3, reliabilities were 0.79 and 0.80 in Models 1 and

3 when clustering was taken into account (Table 3). Similarly, for the latent trait factor, reliabilities were 0.73 and 0.71 in Models 2 and 3 when clustering was not taken into account, respectively, and 0.78 and 0.72 when clustering was accounted for (Table 3). In short, taking clustering into account improved reliability, although to a small degree, in all models.

RQ2: Latent Trait-Speed Connection. Our results for the full model in which the latent trait was regressed on the latent speed factor (last columns of Table 3) showed a positive regression coefficient = 0.42 (*SE* = 0.09), $p < .001$ (standardized coefficient = 0.39), which ultimately translated into 15% of the variance in the latent trait explained by latent speed. In other words, in the U.S. sample, students who took 1 standard deviation longer to answer the items were predicted to be 0.39 standard deviations higher on their math skills.

RQ3: Latent Trait Reliability after accounting for Latent Speed. Although latent speed did explain significant variance in the latent trait, our results did not show improvement in the scale reliability estimate for the latent trait in the joint model. In fact, the estimate decreased from 0.78 when latent speed was not in the model to 0.72 when latent speed was added. This said, the reliability estimate for the latent speed factor did improve in the joint model from 0.79 to 0.80.

Discussion

Building from Costa et al.'s (2021) demonstration, the present study demonstrated a novel approach to incorporating one form of process data, item response time, into a measurement model for a latent trait, in this case, math skills as defined by 10 released items on 2012 Programme for International Student Achievement (PISA). More specifically, we sought to handle item clustering using method factors, and to directly test the relationship between response time (the "latent speed" factor) and response accuracy (the "latent trait" factor).

Although the prior paper used all countries, we focused on the U.S. dataset to avoid additional complexities due to countries' variations in language, culture, and education resources.

The results we found were mixed. First, taking item clustering into account using method factors improved scale reliability estimates in all three of our models, albeit only to a small degree. We speculate that handling clustering will improve reliability estimates as long as clustering has real effects on the observed data. In the present study, some of the items showed little clustering effects (especially Cluster 2; see again Table 3). If the clustering effects were stronger, we would anticipate better precision, and conversely, if the clustering effects were weaker, we would expect little gain in precision.

The second finding that was somewhat surprising was that greater response times predict greater response accuracy. Unfortunately, the PISA 2012 math items analyzed for this study have no details released about them other than the three topics, and further, Costa et al. (2021) did not report the direction of the relationship between response time and accuracy in their study, so we cannot compare results. All this said, Goldhammer and colleagues (e.g., Goldhammer et al., 2014) have found that the relationship between response time and task accuracy can be moderated by type of task, among other things. Taken together, we speculate that, at least in the U.S. sample, students who took more time to answer the items were more carefully thinking through multiple steps needed for arriving at a correct answer. In short, students who knew more were not necessarily answering questions faster.

Last but not least, our third finding was that, in contrast to Costa et al. (2021), taking latent speed (which they termed "time-on-task") into account using a structural equation modeling framework did not improve reliability estimates for the latent trait estimation. We suspect that our computation of scale reliability, which depends on Raykov et al.'s (2010)

method for binary factor models, is quite different from the EAP method used by Costa et al. (2021). Moreover, we directly estimated the effect of response time on response accuracy via a structural parameter that would not necessarily affect item-factor relations – and our reliability estimation method depended on item-factor relations.

Limitations and Future Directions

Of course no study is without limitations. First, we focused on the U.S. sample; thus our results are not necessarily generalizable to other countries. Second, it is unclear how our use of Raykov et al.'s (2010) reliability estimation method for binary factor analysis generalizes (or does not generalize) to Costa et al.'s (2021) EAP method of reliability estimation. Future work may want to evaluate the similarity of these two reliability estimation methods. Third, we used the raw response time data and specified linear response time-accuracy relationships; Costa et al. (2021) used log-transformed response times and Goldhammer et al. (2015) have noted nonlinear relations between response times and task performance. Thus, future research on process data involving response time should investigate varied handling of the raw response time data.

Despite the limitations, the current study demonstrated that use of item cluster factors to handle item dependencies due to a common prompt may actually improve measurement precision, even if taking response time into account does not. This said, we did find a substantial relation between response time and response accuracy to the tune of 15% shared variance; thus the growing popularity of investigating the effects of response time on response accuracy is certainly not without merit. Yet, if researchers' main interest is in improving measurement scale reliability, then, at least for now, we recommend incorporating response time at the item level, like Costa et al. (2021), rather than the factor level, as we did.

References

Bergner, Y., & von Davier, A. A. (2019). Process data in NAEP: Past, present, and future. *Journal of Educational and Behavioral Statistics, 44*(6), 706-732. https://doi.org/10.3102%2F1076998618784700

Caliço, T. (2021). Ontological and methodological barriers to the incorporation of event data in psychometric models. In: Wiberg, M., Molenaar, D., González, J., Böckenholt, U., Kim, JS. (eds) *Quantitative Psychology*. Springer Proceedings in Mathematics & Statistics, vol 353 (pp. 373-383). Springer, Cham. https://doi.org/10.1007/978-3-030-74772-5_33

Costa, D. R., Bolsinova, M., Tijmstra, J., & Andersson, B. (2021). Improving the precision of ability estimates using time-on-task variables: Insights from the PISA 2012 computer-based assessment of mathematics. *Frontiers in Psychology, 12*, 1-13. doi: 10.3389/fpsyg.2021.579128

Dutilh, G., Annis, J., Brown, S.D., et al. (2019). The quality of response time data inference: A blinded, collaborative assessment of the validity of cognitive models. *Psychonomic Bulletin & Review, 26*, 1051-1069. https://doi.org/10.3758/s13423-017-1417-2

Goldhammer, F., & Klein Entink, R. H. (2011). Speed of reasoning and its relation to reasoning ability. *Intelligence, 39*, 108-119. doi: 10.1016/j.intell.2011.02.001

Goldhammer, F., Naumann, J., & Greiff, S. (2015).More is not always better: the relation between item response and item response time in Raven's matrices. *Journal of Intelligence, 3*, 21-40. doi: 10.3390/jintelligence3010021

Goldhammer, F., Naumann, J., Stelter, A., Tóth, K., Rölke, H., & Klieme, E. (2014). The time on task effect in reading and problem solving is moderated by task difficulty and skill:

insights from a computer-based large-scale assessment. *Journal of Educational Psychology, 106,* 608-626. doi: 10.1037/a0034716

Goldhammer, F., & Zehner, F. (2017). What to make of and how to interpret process data. *Measurement, 15,* 128-132. doi: 10.1080/15366367.2017.1411651

Lu, Y., & Sirecci, S. G. (2007). Validity issues in test speededness. *Educational Measurement: Issues and Practice, 26(4),* 29-37.

Kamata, A., & Bauer, D. J. (2008). A note on the relation between factor analytic and item response theory models. *Structural Equation Modeling, 15*(1), 136-153. doi:10.1080/10705510701758406

McDonald, R. P. (1999). *Test Theory: A Unified Treatment*. Lawrence Erlbaum.

Muthén, L. K., & Muthén, B. O. (1998/2017). *Mplus User's Guide (8th Ed.).* Los Angeles, CA: Muthén & Muthén. https://www.statmodel.com/download/usersguide/MplusUserGuideVer_8.pdf

OECD (2014). *PISA 2012 Technical Report.* OECD Publishing, Paris.

Raykov, T., Dimitrov, D. M., & Asparouhov, T. (2010). Evaluation of scale reliability with binary measures using latent variable modeling. *Structural Equation Modeling: A Multidisciplinary Journal, 17*(2), 265-279. doi: 10.1080/10705511003659417

Tong, Y. (2022). NCME presidential address 2021: Assessment research and practice in the post-COVID-19 era. *Educational Measurement: Issues and Practice*. Published Online First. https://doi.org/10.1111/emip.12509

Xie, B., Davidson, M. J., Li, M., & Ko, A. J. (2019a). An item response theory evaluation of a language-independent CS1 knowledge assessment. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 699-705).

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., ... & Ko, A. J. (2019b). A theory of instruction for introductory programming skills. *Computer Science Education*, *29*(2-3), 205-253.

Table 1

Descriptive Statistics for Item Response Time (Speed) Data (in Seconds)

Item	*N*	*M*	*(SD)*
CM015 Cluster			
1. CM015Q01	402	70.61	(40.19)
2. CM015Q02	401	116.72	(83.79)
3. CM015Q03	401	114.13	(56.19)
CM020 Cluster			
4. CM020Q01	400	135.62	(56.08)
5. CM020Q02	399	53.65	(35.90)
6. CM020Q03	398	84.30	(46.45)
7. CM020Q04	396	47.85	(28.04)
CM038 Cluster			
8. CM038Q03	394	73.24	(36.47)
9. CM038Q05	390	98.67	(52.60)
10. CM038Q06	384	90.86	(50.06)

Table 2

Descriptive Statistics for Item Response Data (Items Dichotomously Scored, 1 = Correct)

Item	*N*	*M*	*(SD)*
CM015 Cluster			
1. CM015Q01	398	0.56	(0.50)
2. CM015Q02	337	0.11	(0.31)
3. CM015Q03	370	0.19	(0.39)
CM020 Cluster			
4. CM020Q01	383	0.11	(0.32)
5. CM020Q02	382	0.50	(0.50)
6. CM020Q03	374	0.25	(0.43)
7. CM020Q04	377	0.42	(0.49)
CM038 Cluster			
8. CM038Q03	390	0.65	(0.48)
9. CM038Q05	365	0.30	(0.46)
10. CM038Q06	356	0.21	(0.41)

Table 3

Model Results for Unstandardized Parameter Estimates

Parameter	Model 1		Model 2		Model 3	
	Est	*(SE)*	*Est*	*(SE)*	*Est*	*(SE)*
Speed Reliab	0.79	(0.02)***			0.80	(0.02)***
Trait Reliab			0.78	(0.03)***	0.72	(0.03)***
Speed → Trait					0.42	(0.09)***
Speed Loadings						
X1	15.57	(2.29)***			15.97	(2.21)***
X2	38.78	(4.70)***			42.27	(4.51)***
X3	24.08	(3.14)***			24.58	(3.03)***
X4	30.76	(3.36)***			31.53	(3.01)***
X5	19.24	(2.17)***			19.40	(2.10)***
X6	30.59	(2.56)***			31.19	(2.44)***
X7	11.95	(1.74)***			12.29	(1.58)***
X8	25.57	(2.23)***			24.64	(1.87)***
X9	29.74	(3.50)***			26.95	(2.80)***
X10	20.70	(2.93)***			20.07	(2.81)***
Trait Loadings						
Y1			3.67	(3.34)	1.22	(0.25)***
Y2			5.11	(5.16)	2.68	(0.63)***
Y3			1.92	(0.37)***	4.60	(8.82)
Y4			2.46	(0.54)***	2.10	(0.44)***
Y5			0.99	(0.23)***	0.81	(0.17)***
Y6			1.73	(0.33)***	1.70	(0.32)***
Y7			1.25	(0.32)***	1.04	(0.19)***
Y8			0.68	(0.19)***	0.69	(0.16)***
Y9			1.18	(0.78)	1.13	(0.21)***
Y10			1.18	(0.27)***	1.10	(0.22)***
Clust1 Loadings						
X1	2.92	(4.75)			-1.61	(3.11)
X2	34.41	(21.64)			24.37	(5.80)***
X3	17.16	(11.22)			22.47	(4.52)***
Y1			4.52	(4.79)	0.89	(0.30)**
Y2			3.69	(4.64)	1.23	(0.50)*
Y3			1.02	(0.37)**	6.49	(13.37)
Clust2 Loadings						
X4	4.92	(6.97)			-1.04	(4.92)
X5	11.05	(5.07)*			-19.09	(6.00)***
X6	5.09	(5.12)			-3.21	(3.77)
X7	10.64	(4.70)*			-6.15	(2.97)*
Y4			0.10	(0.51)	0.39	(0.42)
Y5			-0.22	(0.32)	0.37	(0.21)
Y6			0.84	(0.44)	0.33	(0.27)
Y7			1.26	(0.48)**	0.17	(0.21)

Table 3, Cont'd

	Model 1		Model 2		Model 3	
Parameter	*Est*	*(SE)*	*Est*	*(SE)*	*Est*	*(SE)*
Clust3 Loadings						
X8	5.86	(5.65)			6.81	(1.96)***
X9	21.67	(8.44)**			42.87	(3.44)***
X10	19.04	(7.74)*			13.08	(2.97)***
Y8			-0.07	(0.69)	-0.12	(0.13)
Y9			0.76	(2.39)	0.86	(0.21)***
Y10			-0.25	(0.78)	-0.22	(0.19)
Speed Intercepts						
X1	70.61	(2.01)***			70.62	(2.00)***
X2	116.64	(4.20)***			116.78	(4.22)***
X3	114.09	(2.81)***			114.20	(2.83)***
X4	135.60	(2.80)***			135.64	(2.80)***
X5	53.53	(1.81)***			53.56	(1.80)***
X6	84.11	(2.35)***			84.12	(2.33)***
X7	47.73	(1.41)***			47.75	(1.41)***
X8	73.03	(1.84)***			73.05	(1.83)***
X9	98.33	(2.71)***			98.08	(2.65)***
X10	90.36	(2.56)***			90.26	(2.54)***
Trait Thresholds						
Y1			-0.90	(0.92)	-0.35	(0.15)*
Y2			8.00	(8.05)	4.50	(0.82)***
Y3			2.51	(0.34)***	7.96	(15.24)
Y4			3.72	(0.59)***	3.57	(0.51)***
Y5			-0.01	(0.12)	0.00	(0.12)
Y6			1.76	(0.27)***	1.75	(0.25)***
Y7			0.51	(0.17)**	0.43	(0.13)***
Y8			-0.68	(0.12)***	-0.69	(0.12)***
Y9			1.16	(0.72)	1.27	(0.19)***
Y10			1.68	(0.24)***	1.67	(0.19)***

* $p < .05$, ** $p < .01$, *** $p < .001$.

Figure 1

Illustration of Final Model Implemented by Costa et al. (2021)

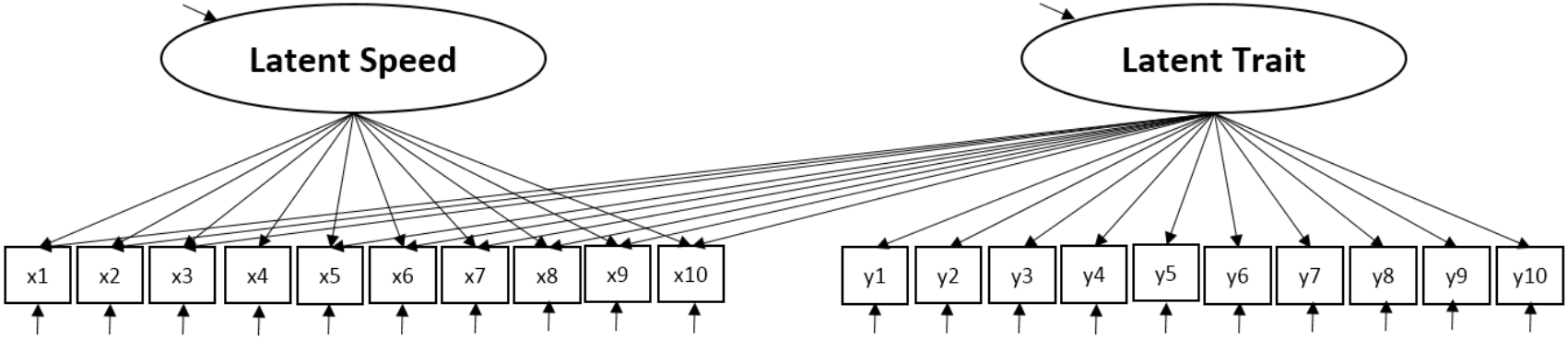
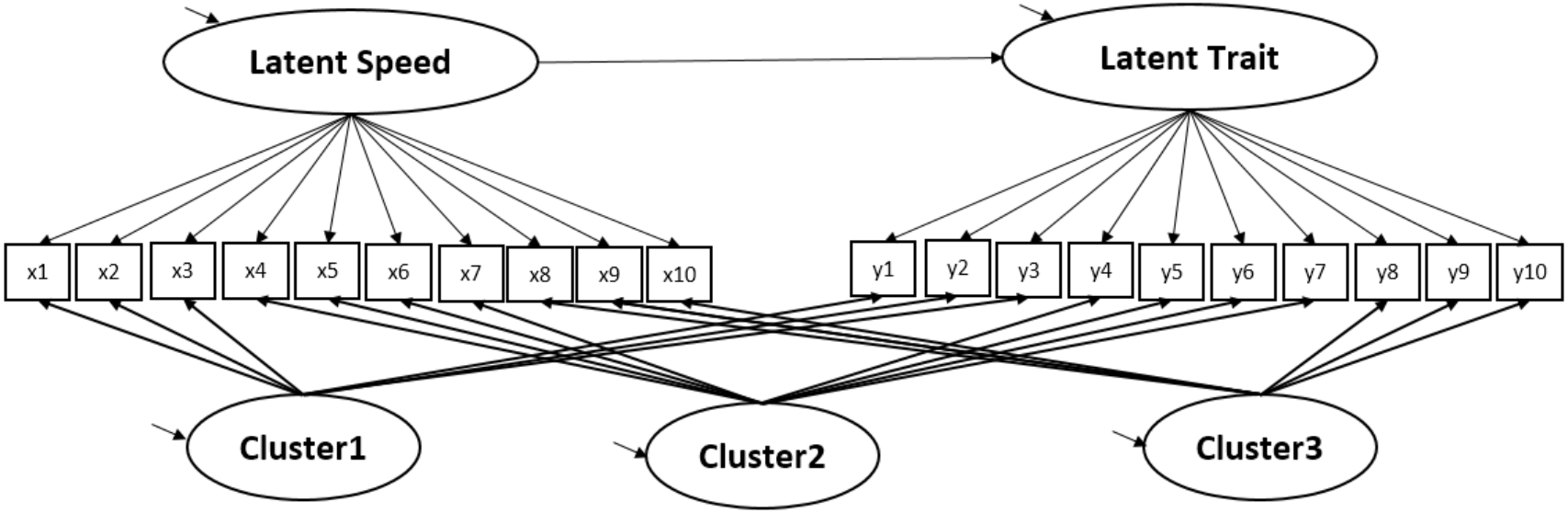


Figure 2

Illustration of Final Model (M3) Implemented in Current Study



Appendix A

R Code for Data Cleaning

```
# GENERAL PREP ----
## libraries
library(Matrix)
library(lme4)
library(lmerTest)
library(dplyr)
library(car)
library(misty)
library(readr)
library(tidyverse)
library(multiplex)

## clean up old objects
rm(list=ls())

## remove scientific notation
options(scipen=100)
options(digits=4)

# PART 1: ITEM RESPONSE "ABILITY" DATA (Latent Math Skills Trait) ----

## Import data ----
PISA2012_CM_itemresp = read.csv("filepathhere/CBA_COG12_MAR31_ANS_usa.csv")

### get column names for next step
colnames(PISA2012_CM_itemresp)

## Subset data to students and items ----
PISA2012_CM_itemresp <- subset(PISA2012_CM_itemresp, select =
                                 c(StIDStd, CM015Q01, CM015Q02D, CM015Q03D,
                                   CM020Q01, CM020Q02, CM020Q03, CM020Q04,
                                   CM038Q03T, CM038Q05, CM038Q06))

### check that subsetting was done correctly
str(PISA2012_CM_itemresp)

## Recode item response data ----

#CM015Q01
unique(PISA2012_CM_itemresp$CM015Q01)
PISA2012_CM_itemresp$CM015Q01_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM015Q01,
                                      "3=1; 1=0; 2=0; 4=0; else=NA"))
#CM015Q02D
unique(PISA2012_CM_itemresp$CM015Q02D)
PISA2012_CM_itemresp$CM015Q02D_poly <- as.numeric(recode(PISA2012_CM_itemresp$CM015Q02D,
                                        "'G'=1; 'I'=1; 'J'=1; 'O'=2; 'A'=0; else=NA"))
PISA2012_CM_itemresp$CM015Q02D_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM015Q02D_poly,
                                        "2=1; 1=0; 0=0; else=NA"))
#CM015Q03D
unique(PISA2012_CM_itemresp$CM015Q03D)
PISA2012_CM_itemresp$CM015Q03D_poly <- as.numeric(recode(PISA2012_CM_itemresp$CM015Q03D,
                                        "'G'=1; 'H'=1; 'O'=2; 'A'=0; else=NA"))
PISA2012_CM_itemresp$CM015Q03D_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM015Q03D_poly,
                                       "2=1; 1=0; 0=0; else=NA"))
#CM020Q01
unique(PISA2012_CM_itemresp$CM020Q01)
PISA2012_CM_itemresp$CM020Q01_poly <- as.numeric(recode(PISA2012_CM_itemresp$CM020Q01,
                                        "1=1; 2=2; 0=0; else=NA"))
PISA2012_CM_itemresp$CM020Q01_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM020Q01_poly,
                                      "2=1; 1=0; 0=0; else=NA"))
#CM020Q02
unique(PISA2012_CM_itemresp$CM020Q02)
PISA2012_CM_itemresp$CM020Q02_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM020Q02,
                                      "2=1; 1=0; 3=0; 4=0; else=NA"))
#CM020Q03
```

```
unique(PISA2012_CM_itemresp$CM020Q03)
PISA2012_CM_itemresp$CM020Q03_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM020Q03,
                                     "1=1; 0=0; else=NA"))
#CM020Q04
unique(PISA2012_CM_itemresp$CM020Q04)
PISA2012_CM_itemresp$CM020Q04_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM020Q04,
                                     "3=1; 1=0; 2=0; 4=0; else=NA"))
#CM038Q03T
unique(PISA2012_CM_itemresp$CM038Q03T)
PISA2012_CM_itemresp$CM038Q03T_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM038Q03T,
                                                        "2=1; 1=0; 0=0; else=NA"))
#CM038Q05
unique(PISA2012_CM_itemresp$CM038Q05)
PISA2012_CM_itemresp$CM038Q05_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM038Q05,
                                                         "1=1; 0=0; else=NA"))
#CM038Q06
unique(PISA2012_CM_itemresp$CM038Q06)
PISA2012_CM_itemresp$CM038Q06_bin <- as.numeric(recode(PISA2012_CM_itemresp$CM038Q06,
                                                        "1=1; 0=0; else=NA"))

## Re-subset data with only recoded binary data (take out old items) ----
PISA2012_CM_itemresp <- subset(PISA2012_CM_itemresp, select = -c(
  CM015Q01, CM015Q02D, CM015Q03D,
  CM020Q01, CM020Q02, CM020Q03, CM020Q04,
  CM038Q03T, CM038Q05, CM038Q06, CM015Q02D_poly, CM015Q03D_poly, CM020Q01_poly))

### check data
str(PISA2012_CM_itemresp)

## Create -999 version ----
PISA2012_CM_itemresp_999 <- PISA2012_CM_itemresp
PISA2012_CM_itemresp_999[is.na(PISA2012_CM_itemresp_999)] = -999


# PART 2: RESPONSE TIME DATA ----

## Import data ----
path = "filepathhere"

PISA2012_cm015q1 <- read.csv(paste(path, "CBA_cm015q01_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm015q1)

PISA2012_cm015q2D <- read.csv(paste(path, "CBA_cm015q02_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm015q2D)

PISA2012_cm015q3D <- read.csv(paste(path, "CBA_cm015q03_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm015q3D)

PISA2012_cm020q1 <- read.csv(paste(path, "CBA_cm020q01_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm020q1)

PISA2012_cm020q2 <- read.csv(paste(path, "CBA_cm020q02_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm020q2)

PISA2012_cm020q3 <- read.csv(paste(path, "CBA_cm020q03_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm020q3)

PISA2012_cm020q4 <- read.csv(paste(path, "CBA_cm020q04_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm020q4)

PISA2012_cm038q3T <- read.csv(paste(path, "CBA_cm038q03_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm038q3T)

PISA2012_cm038q5 <- read.csv(paste(path, "CBA_cm038q05_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm038q5)

PISA2012_cm038q6 <- read.csv(paste(path, "CBA_cm038q06_logs12_usa.csv", sep = ""))
colnames(PISA2012_cm038q6)

## unique value tracking function in event column ----
### function looks at different values in event column
```

```
### The variables are 'click', 'keyup', 'ACER_EVENT'

unique_vals <- function(df){
  return(unique(subset(df, select = event)$event))
}

unique_vals(PISA2012_cm015q1)# click, keyup
unique_vals(PISA2012_cm015q2D)# keyup
unique_vals(PISA2012_cm015q3D)# keyup
unique_vals(PISA2012_cm020q1)# click, ACER_EVENT
unique_vals(PISA2012_cm020q2)# click
unique_vals(PISA2012_cm020q3)# click, ACER_EVENT
unique_vals(PISA2012_cm020q4)# click
unique_vals(PISA2012_cm038q3T)# click

### for the items below there are no clicks, keyups, and ACER_EVENT
unique_vals(PISA2012_cm038q5)
unique_vals(PISA2012_cm038q6)

## subtract takes in data frame to calculate the total time spent by test takers ----

subtract <- function(df){

  # subdf is made since these columns are not included in the calculating process but
  # these should be included in the final process
  subdf <- subset(df, select = c(cnt, nc, schoolid, StIDStd, formid))

  # I made event subset to calculate the time of each event occurrences
  event <- subset(df, select = c(StIDStd, event, time))

  # dplyr sub-setting data frame by row values
  start <- event %>% filter(event =='START_ITEM')
  end <- event %>% filter(event =='END_ITEM')
  total <- merge(start, end, by='StIDStd', all=TRUE)

  # changing column names for easy understanding
  colnames(total) <- c("StIDStd", "event_start", "time_start",  "event_end", "time_end")

  # subtraction
  total$time_tot <- (total$time_end - total$time_start)
  total <- merge(subdf, total, by="StIDStd", all=TRUE)
  return(total)
}

cm015q1_time_tot <- subtract(PISA2012_cm015q1)
cm015q2D_time_tot <- subtract(PISA2012_cm015q2D)
cm015q3D_time_tot <- subtract(PISA2012_cm015q3D)
cm020q1_time_tot <- subtract(PISA2012_cm020q1)
cm020q2_time_tot <- subtract(PISA2012_cm020q2)
cm020q3_time_tot <- subtract(PISA2012_cm020q3)
cm020q4_time_tot <- subtract(PISA2012_cm020q4)
cm038q3T_time_tot <- subtract(PISA2012_cm038q3T)
cm038q5_time_tot <- subtract(PISA2012_cm038q5)
cm038q6_time_tot <- subtract(PISA2012_cm038q6)

## event_click_keyup takes in a data frame, ----
### calculates total number of both click and keyup by student id
### This function can be used only when both click and keyup exist in event column

event_click_keyup <- function(df){

  #subsetting df to only necessary values
  #student id column is needed to calculate variables by students
  event <- subset(df, select = c(StIDStd, event))

  # create subset of click and keyup to calculate total events without start and end log
  click <- event %>% filter(event =='click')
  keyup <- event %>% filter(event =='keyup')

  # count each events by student id
  click_counts <- click %>% count(StIDStd) # use this for the whole process
```

```
  keyup_counts <- keyup %>% count(StIDStd)

  # merge them to one dataframe
  event_count <- merge(click_counts, keyup_counts, by='StIDStd', all=TRUE)

  # rename columns
  colnames(event_count) <- c("StIDStd", "click_tot", "keyup_tot")

  # calculate total
  event_count$event_tot <- event_count$click + event_count$keyup

  # return the dataframe
  return(event_count)
}

cm015q1_events_click <- event_click_keyup(PISA2012_cm015q1)
#View(cm015q1_events_click)

## event_keyup calculates total number of keyups by student id ----

event_keyup <- function(df){

  #subsetting df to only necessary values
  #student id colum is needed for merging data later
  event <- subset(df, select = c(StIDStd, event))

  # create subset of click and keyup to calculate total events without start and end log
  keyup <- event %>% filter(event =='keyup')

  # count each events by student id
  keyup_counts <- keyup %>% count(StIDStd)
  colnames(keyup_counts) <- c("StIDStd", "keyup_tot")

  # return the dataframe
  return(keyup_counts)
}
cm015q2D_events_keyup <- event_keyup(PISA2012_cm015q2D)
cm015q3D_events_keyup <- event_keyup(PISA2012_cm015q3D)

## event_click calculates total number of clicks by student id ----

event_click <- function(df){

  #subsetting df to only necessary values
  #student id colum is needed for merging data later
  event <- subset(df, select = c(StIDStd, event))

  # create subset of click to calculate total events without start and end log
  click <- event %>% filter(event =='click')

  # count by student id
  click_counts <- click %>% count(StIDStd) # use this for the whole process
  colnames(click_counts) <- c("StIDStd", "click_tot")

  # return the dataframe
  return(click_counts)
}

cm020q2_events_click <- event_click(PISA2012_cm020q2)
cm020q4_events_click <- event_click(PISA2012_cm020q4)
cm038q3T_events_click <- event_click(PISA2012_cm038q3T)

## event_click_acer calculates total number of click and ACER_EVENT by student id ----

event_click_acer <- function(df){

  event <- subset(df, select = c(StIDStd, event))

  # create subset of click and ACER_EVENT to calculate total events without start and end log
  click <- event %>% filter(event =='click')
  acer <- event %>% filter(event =='ACER_EVENT')
```

```
  # count by student id
  click_counts <- click %>% count(StIDStd) # use this for the whole process
  acer_counts <- acer %>% count(StIDStd)

  # merge them to one dataframe
  event_count <- merge(click_counts, acer_counts, by='StIDStd', all=TRUE)

  # rename columns
  colnames(event_count) <- c("StIDStd", "click_tot", "ACER_EVENT_tot")

  # calculate total
  event_count$event_tot <- event_count$click + event_count$ACER_EVENT

  # return the dataframe
  return(event_count)
}

cm020q1_events_click_acer <- event_click_acer(PISA2012_cm020q1)
cm020q3_events_click_acer <- event_click_acer(PISA2012_cm020q3)

## merge time_tot data and events data frames by student id ----

merge_all <- function(df_1, df_2){
  new_df <- merge(df_1, df_2, by='StIDStd', all=TRUE)
  return(new_df)
}

# Notice items PISA2012_cm038q5 and PISA2012_cm038q6 are not merged since they don't have click
events
merge_cm015q1 <- merge_all(cm015q1_time_tot, cm015q1_events_click)
merge_cm015q2D <- merge_all(cm015q2D_time_tot, cm015q2D_events_keyup)
merge_cm015q3D <- merge_all(cm015q3D_time_tot, cm015q3D_events_keyup)
merge_cm020q1 <- merge_all(cm020q1_time_tot, cm020q1_events_click_acer)
merge_cm020q2 <- merge_all(cm020q2_time_tot, cm020q2_events_click)
merge_cm020q3 <- merge_all(cm020q3_time_tot, cm020q3_events_click_acer)
merge_cm020q4 <- merge_all(cm020q4_time_tot, cm020q4_events_click)
merge_cm038q3T <- merge_all(cm038q3T_time_tot, cm038q3T_events_click)

colnames(merge_cm015q1)
colnames(merge_cm015q2D)
colnames(merge_cm015q3D)
colnames(merge_cm020q1)
colnames(merge_cm020q2)
colnames(merge_cm020q3)
colnames(merge_cm020q4)
colnames(merge_cm038q3T)

## event_per_calc calculates % of entire event compared to total time by students ----

event_per_calc <- function(df){
  df$time_perc <- (df$event_tot/df$time_tot)
  return(df)
}

merge_cm015q1_per <- event_per_calc(merge_cm015q1)
merge_cm020q1_per <- event_per_calc(merge_cm020q1)
merge_cm020q3_per <- event_per_calc(merge_cm020q3)

## keyup_per calculates % of keyup event compared to total time spent by students ----

keyup_per <- function(df){
  df$keyup_perc <- (df$keyup_tot/df$time_tot)
  return(df)
}

merge_cm015q1_per <- keyup_per(merge_cm015q1_per)
merge_cm015q2D_per <- keyup_per(merge_cm015q2D)
merge_cm015q3D_per <- keyup_per(merge_cm015q3D)
```

```
## click_per calculates % of click event compared to total time spent by students ----

click_per <- function(df){
  df$click_perc <- (df$click_tot/df$time_tot)
  return(df)
}

merge_cm015q1_per <- click_per(merge_cm015q1_per)
merge_cm020q1_per <- click_per(merge_cm020q1_per)
merge_cm020q2_per <- click_per(merge_cm020q2)
merge_cm020q3_per <- click_per(merge_cm020q3_per)
merge_cm020q4_per <- click_per(merge_cm020q4)
merge_cm038q3T_per <- click_per(merge_cm038q3T)


## acer_per calculates % of ACER_EVENT compared to total time spent by students ----

acer_per <- function(df){
  df$acer_perc<- (df$ACER_EVENT_tot/df$time_tot)
  return(df)
}
merge_cm020q1_per <- acer_per(merge_cm020q1_per)
merge_cm020q3_per <- acer_per(merge_cm020q3_per)

## merge columns ----
colnames(merge_cm015q1_per)
colnames(merge_cm015q2D_per)
colnames(merge_cm015q3D_per)
colnames(merge_cm020q1_per)
colnames(merge_cm020q2_per)
colnames(merge_cm020q3_per)
colnames(merge_cm020q4_per)
colnames(merge_cm038q3T_per)
colnames(cm038q5_time_tot)

## renaming all the variables ----

names(merge_cm015q1_per)[10:16] <- c('cm015q1_time_tot', 'cm015q1_click_tot',
'cm015q1_keyup_tot', 'cm015q1_event_tot', 'cm015q1_time_perc',
                                     'cm015q1_keyup_perc', 'cm015q1_click_perc')
names(merge_cm015q2D_per)[10:12] <- c('cm015q2D_time_tot', 'cm015q2D_keyup_tot',
'cm015q2D_keyup_perc')
names(merge_cm015q3D_per)[10:12] <- c('cm015q3D_time_tot', 'cm015q3D_keyup_tot',
'cm015q3D_keyup_perc')
names(merge_cm020q1_per)[10:16] <- c('cm020q1_time_tot', 'cm020q1_click_tot',
'cm020q1_ACER_EVENT_tot', 'cm020q1_event_tot', 'cm020q1_time_perc',
                                     'cm020q1_click_perc', 'cm020q1_acer_perc')
names(merge_cm020q2_per)[10:12] <- c('cm020q2_time_tot', 'cm020q2_click_tot',
'cm020q2_click_perc')
names(merge_cm020q3_per)[10:16] <- c('cm020q3_time_tot', 'cm020q3_click_tot',
'cm020q3_ACER_EVENT_tot', 'cm020q3_event_tot', 'cm020q3_time_perc',
                                     'cm020q3_click_perc', 'cm020q3_acer_perc')
names(merge_cm020q4_per)[10:12] <- c('cm020q4_time_tot', 'cm020q4_click_tot',
'cm020q4_click_perc')
names(merge_cm038q3T_per)[10:12] <- c('cm038q3T_time_tot', 'cm038q3T_click_tot',
'cm038q3T_click_perc')
names(cm038q5_time_tot)[10] <- 'cm038q5_time_tot'
names(cm038q6_time_tot)[10] <- 'cm038q6_time_tot'

## excluding all the duplicated rows ----

cm015q1_final <- merge_cm015q1_per %>% distinct()
cm015q2D_final <- merge_cm015q2D_per %>% distinct()
cm015q3D_final <- merge_cm015q3D_per %>% distinct()
cm020q1_final <- merge_cm020q1_per %>% distinct()
cm020q2_final <- merge_cm020q2_per %>% distinct()
cm020q3_final <- merge_cm020q3_per %>% distinct()
cm020q4_final <- merge_cm020q4_per %>% distinct()
cm038q3T_final <- merge_cm038q3T_per %>% distinct()
cm038q5_final <- cm038q5_time_tot %>% distinct()
cm038q6_final <- cm038q6_time_tot %>% distinct()
```

```
## write csv including all new columns included ----

write.csv(cm015q1_final, paste(path, "cm015q1_recode.csv"))
write.csv(cm015q2D_final, paste(path, "cm015q2D_recode.csv"))
write.csv(cm015q3D_final, paste(path, "cm015q3D_recode.csv"))
write.csv(cm020q1_final, paste(path, "cm020q1_recode.csv"))
write.csv(cm020q2_final, paste(path, "cm020q2_recode.csv"))
write.csv(cm020q3_final, paste(path, "cm020q3_recode.csv"))
write.csv(cm020q4_final, paste(path, "cm020q4_recode.csv"))
write.csv(cm038q3T_final, paste(path, "cm038q3T_recode.csv"))
write.csv(cm038q5_final, paste(path, "cm038q5_recode.csv"))
write.csv(cm038q6_final, paste(path, "cm038q6_recode.csv"))

## function that takes a string value of csv files to read it ----

read_csv <- function(name){
  read.csv(paste(path, name))
}

cm015q1 <- read_csv('cm015q1_recode.csv')
colnames(cm015q1)
cm015q2D <- read_csv('cm015q2D_recode.csv')
cm015q3D <- read_csv('cm015q3D_recode.csv')
cm020q1 <- read_csv('cm020q1_recode.csv')
cm020q2 <- read_csv('cm020q2_recode.csv')
cm020q3 <- read_csv('cm020q3_recode.csv')
cm020q4 <- read_csv('cm020q4_recode.csv')
cm038q3T <- read_csv('cm038q3T_recode.csv')
cm038q5 <- read_csv('cm038q5_recode.csv')
cm038q6 <- read_csv('cm038q6_recode.csv')

## Delete unexpected index column created in the file reading process ----

delete_col <- function(df){
  df <- df[-1]
  df <- df[-c(6:9)]
}
cm015q1_del <- delete_col(cm015q1)
cm015q2D_del <- delete_col(cm015q2D)
cm015q3D_del <- delete_col(cm015q3D)
cm020q1_del <- delete_col(cm020q1)
cm020q2_del <- delete_col(cm020q2)
cm020q3_del <- delete_col(cm020q3)
cm020q4_del <- delete_col(cm020q4)
cm038q3T_del <- delete_col(cm038q3T)
cm038q5_del <- delete_col(cm038q5)
cm038q6_del <- delete_col(cm038q6)

## put all the data frames in a list to merge at once with reduce function ----
df_list <- list(cm015q1_del, cm015q2D_del, cm015q3D_del, cm020q1_del, cm020q2_del, cm020q3_del,
cm020q4_del, cm038q3T_del, cm038q5_del, cm038q6_del)
merged_all <- df_list %>% reduce(full_join, by=c('StIDStd'))

## check data ----
str(merged_all)

## subset merged_all file ----
merged_all_subset <- subset(merged_all, select =
                                c(StIDStd, cm015q1_time_tot, cm015q2D_time_tot,
cm015q3D_time_tot,
                                  cm020q1_time_tot, cm020q2_time_tot, cm020q3_time_tot,
cm020q4_time_tot,
                                  cm038q3T_time_tot, cm038q5_time_tot, cm038q6_time_tot))

### check subsetting was done correctly
str(merged_all_subset)

## change the response time data name ----
PISA2012RTdata <- merged_all_subset
str(PISA2012RTdata)
```

```
## create -999 version ----
PISA2012RTdata_999 <- PISA2012RTdata
PISA2012RTdata_999[is.na(PISA2012RTdata_999)] = -999
str(PISA2012RTdata_999)


# PART 3: Merge the two datasets ----

## full join (original sample N=4978 in the U.S. sample)
PISA2012data_NA_complete <- merge(PISA2012_CM_itemresp, PISA2012RTdata, by = "StIDStd", all=TRUE)
PISA2012data_999_join <- merge(PISA2012_CM_itemresp_999, PISA2012RTdata_999, by = "StIDStd",
all=TRUE)
PISA2012data_999_complete <- PISA2012data_999_join
PISA2012data_999_complete[is.na(PISA2012data_999_complete)] = -999
View(PISA2012data_999_complete)
## round
PISA2012data_NA_complete <- PISA2012data_NA_complete %>% mutate_if(is.numeric, round, digits=3)
PISA2012data_999_complete <- PISA2012data_999_complete %>% mutate_if(is.numeric, round, digits=3)

## only join for N=402 U.S. sample with any response time data collected on computerized math
problems
## this is the data we will use for analyses
PISA2012data_NA <- merge(PISA2012_CM_itemresp, PISA2012RTdata, by = "StIDStd")
PISA2012data_999 <- merge(PISA2012_CM_itemresp_999, PISA2012RTdata_999, by = "StIDStd")
## round
PISA2012data_NA <- PISA2012data_NA %>% mutate_if(is.numeric, round, digits=3)
PISA2012data_999 <- PISA2012data_999 %>% mutate_if(is.numeric, round, digits=3)
## check
str(PISA2012data_NA)
str(PISA2012data_999)
#View(PISA2012data_999)


# PART 4: ITEM CLUSTER ANALYSIS (DEPENDENCIES IN DATA) ----

## Create long format using NA version of data ----
data_long = reshape(PISA2012data_NA,
                    idvar = "StIDStd",
                    varying = list(c(2:11),c(12:21)), # repeated variable column locations
                    v.names = c("Ability", "RT"),     # names repeated variables in same order
                    direction = "long")               # tells R to convert from wide to long

data_long$time <- as.numeric(data_long$time)
names(data_long)[names(data_long) == 'time'] <- "item"
data_long$clust <- recode(data_long$item, "1=1;2=1;3=1;4=2;5=2;6=2;7=2;8=3;9=3;10=3")

### check data
str(data_long)
hist(data_long$Ability)
hist(data_long$RT)

## Run cross-classified models to get ICCs ----
### Ability is binary, so use logistic regression
Ability_ICC <- glmer(Ability ~ 1 + (1|StIDStd) + (1|clust/item),
data=data_long,family="binomial")
### RT is count data, so use linear regression
RT_ICC <-       lmer(RT ~ 1 + (1|StIDStd) + (1|clust/item), data=data_long, REML=FALSE)

## Get model results and compute ICCs ----

### Ability
summary(Ability_ICC)

### Random effects:
### Random effects:
###   Groups     Name        Variance      Std.Dev.
### StIDStd    (Intercept) 1.50374009477 1.2262708
### item:clust (Intercept) 1.28350279873 1.1329178
### clust      (Intercept) 0.00000000038 0.0000195
### Number of obs: 3732, groups:  StIDStd, 401; item:clust, 10; clust, 3
```

```
### Fixed effects:
###   Estimate Std. Error z value Pr(>|z|)
### (Intercept)   -1.062      0.367    -2.9   0.0038 **

### ICC calculations: add 3.29 for approx L1 variance since Ability is binary
Ability_ICC_student = 1.50374009477/(1.50374009477+1.28350279873+0.00000000038+3.29)
Ability_ICC_items = 1.28350279873/(1.50374009477+1.28350279873+0.00000000038+3.29)
Ability_ICC_clusters = 0.00000000038/(1.50374009477+1.28350279873+0.00000000038+3.29)
Ability_ICC_student
Ability_ICC_items
Ability_ICC_clusters

### Response Time
summary(RT_ICC)

### Random effects:
###   Groups     Name        Variance Std.Dev.
### StIDStd    (Intercept)  661     25.7
### item:clust (Intercept)  721     26.9
### clust      (Intercept)    0      0.0
### Residual               1926     43.9
### Number of obs: 3965, groups:  StIDStd, 402; item:clust, 10; clust, 3
###
### Fixed effects:
###   Estimate Std. Error    df t value   Pr(>|t|)
### (Intercept)    88.51       8.62 10.46    10.3 0.00000086 ***

### ICC calculations: add resid L1 error since RT is continuous
RT_ICC_student = 661/(661+721+0+1926)
RT_ICC_items = 721/(661+721+0+1926)
RT_ICC_clusters = 0/(661+721+0+1926)
RT_ICC_student
RT_ICC_items
RT_ICC_clusters


# EXPORT DATA (WITH NA and WITHOUT NA for missing data) ----
write.csv(PISA2012data_NA, "filepathhere/PISA2012data_NA.csv")
write.table(PISA2012data_999, "filepathhere/PISA2012data_999.dat",
            sep = "\t", row.names=FALSE, col.names=FALSE)
write.csv(PISA2012data_NA_complete, "filepathhere/PISA2012data_NA_complete.csv")
write.table(PISA2012data_999_complete, "filepathhere/PISA2012data_999_complete.dat",
            sep = "\t", row.names=FALSE, col.names=FALSE)
```

Appendix B

Mplus Code for Data Analysis

Model 1: Latent Speed Measurement Model

```
TITLE: Latent Speed Model for PISA2012
DATA: FILE="pathnamehere\PISA2012data_999.dat"; !update file name
VARIABLE: NAMES = ID y1-y10 x1-x10;
USEVARIABLES = !y1-y10
x1-x10;
!CATEGORICAL = y1-y10;
MISSING = ALL(-999); ! missing data must be coded, then turn on
ANALYSIS:
    ESTIMATOR = ML; !WLSMV USES PROBIT
    !LINK=LOGIT; ! automatically done with categorical indicators
    ITERATIONS = 1000000;
MODEL:
! continuous item factor
RT BY x1* (L1)
x2* (L2)
x3* (L3)
x4* (L4)
x5* (L5)
x6* (L6)
x7* (L7)
x8* (L8)
x9* (L9)
x10*2 (L10);
x1-x10* (E1-E10); ! variances
[x1-x10*] ; ! means/intercepts

RT@1; !for identifiablity
[RT@0];
! method factors
Clust1 BY x1*
x2*
x3*;
Clust2 BY x4*
x5*
X6*
X7*;
Clust3 BY x8*
x9*
x10*;
Clust1@1; !for indentifiability
Clust2@1;
Clust3@1;
[Clust1@0];
[Clust2@0];
[Clust3@0];
Clust1 WITH Clust2@0;
Clust1 WITH Clust3@0;
Clust2 WITH Clust3@0;
Clust1 WITH RT@0;
Clust2 WITH RT@0;
Clust3 WITH RT@0;

MODEL CONSTRAINT: !this is to get reliability for speed
NEW(omega);
omega =
(L1+L2+L3+L4+L5+L6+L7+L8+L9+L10)^2/
((L1+L2+L3+L4+L5+L6+L7+L8+L9+L10)^2
+(E1+E2+E3+E4+E5+E6+E7+E8+E9+E10));

OUTPUT: stdyx; cinterval;
```

Model 2: Latent Trait Measurement Model

```
TITLE: Latent Trait Model for Math Items PISA2012
DATA: FILE="pathnamehere\PISA2012data_999.dat"; !update file name
VARIABLE:
NAMES = ID y1-y10 x1-x10;
USEVARIABLES = y1-y10;! x1-x10;
CATEGORICAL = y1-y10;
MISSING = ALL(-999); ! missing data must be coded, then turn on
ANALYSIS:
    ESTIMATOR = ML; !WLSMV USES PROBIT
    LINK=LOGIT; ! automatically done with categorical indicators
    ITERATIONS = 100000;
    INTEGRATION = MONTECARLO;
    MITERATIONS = 1000;
    MCSEED = 2931;
MODEL:
! binary factor (Ability)
Ability BY y1*1.760 (p1) ! y's are for loadings; p's are for reliab (in constraints)
y2*3.429 (p2)
y3*2.196 (p3)
y4*2.164 (p4)
y5*0.845 (p5)
y6*1.446 (p6)
y7*0.944 (p7)
y8*0.672 (p8)
y9*0.960 (p9)
y10*1.117 (p10);
Ability@1; !for identifiablity
[Ability@0];
[y1$1*-0.355] (p11);
[y2$1*4.767 ] (p12);
[y3$1*2.537 ] (p13);
[y4$1*3.443 ] (p14);
[y5$1*-0.006] (p15);
[y6$1*1.532 ] (p16);
[y7$1*0.398 ] (p17);
[y8$1*-0.673] (p18);
[y9$1*1.014 ] (p19);
[y10$1*1.643] (p20);  ! ensures a 2PL model

Clust1 BY y1*4.112
y2*7.296
y3*2.151;
Clust2 BY y4*2.663
y5*0.859
y6*1.736
y7*1.296;
Clust3 BY y8*0.507
y9*1.116
y10*0.919;

Clust1@1;
Clust2@1;
Clust3@1;
[Clust1@0];
[Clust2@0];
[Clust3@0];
Clust1 WITH Clust2@0;
Clust1 WITH Clust3@0;
Clust2 WITH Clust3@0;
Clust1 WITH Ability@0;
Clust2 WITH Ability@0;
Clust3 WITH Ability@0;
```

```
MODEL CONSTRAINT: !this is to get reliability estimates (Raykov et al 2010)
NEW(
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10
q1 q2 q3 q4 q5 q6 q7 q8 q9 q10
pi1 pi2 pi3 pi4 pi5 pi6 pi7 pi8 pi9 pi10
tv1 tv2 tv3 tv4 tv5 tv6 tv7 tv8 tv9 tv10
ev1 ev2 ev3 ev4 ev5 ev6 ev7 ev8 ev9 ev10
r1 r2 r3 r4 r5 r6 r7 r8 r9 r10
errvar truvar rel
);

a1 = p1/1.702; ! loadings p1/1.702;
a2 = p2/1.702;
a3 = p3/1.702;
a4 = p4/1.702;
a5 = p5/1.702;
a6 = p6/1.702;
a7 = p7/1.702;
a8 = p8/1.702;
a9 = p9/1.702;
a10 = p10/1.702;

b1 = p11/p1; ! thresholds p11/p1;
b2 = p12/p2;
b3 = p13/p3;
b4 = p14/p4;
b5 = p15/p5;
b6 = p16/p6;
b7 = p17/p7;
b8 = p18/p8;
b9 = p19/p9;
b10 = p20/p10;

!!!! reverse sign for negative b's
! these data: 1,5,8
q1 = -a1*b1/sqrt(2+2*a1**2);
q2 = a2*b2/sqrt(2+2*a2**2);
q3 = a3*b3/sqrt(2+2*a3**2);
q4 = a4*b4/sqrt(2+2*a4**2);
q5 = -a5*b5/sqrt(2+2*a5**2);
q6 = a6*b6/sqrt(2+2*a6**2);
q7 = a7*b7/sqrt(2+2*a7**2);
q8 = -a8*b8/sqrt(2+2*a8**2);
q9 = a9*b9/sqrt(2+2*a9**2);
q10 = a10*b10/sqrt(2+2*a10**2);

!!!! add .5 instead of subtract for negative b's
! these data: 1,5,8
pi1 = .5 +.5*(1-1/(1+.278393*q1 + .230389*q1**2
+ .000972*q1**3 + .078108*q1**4)**4);
pi2 = .5 -.5*(1-1/(1+.278393*q2 + .230389*q2**2
+ .000972*q2**3 + .078108*q2**4)**4);
pi3 = .5 -.5*(1-1/(1+.278393*q3 + .230389*q3**2
+ .000972*q3**3 + .078108*q3**4)**4);
pi4 = .5 -.5*(1-1/(1+.278393*q4 + .230389*q4**2
+ .000972*q4**3 + .078108*q4**4)**4);
pi5 = .5 +.5*(1-1/(1+.278393*q5 + .230389*q5**2
+ .000972*q5**3 + .078108*q5**4)**4);
pi6 = .5 -.5*(1-1/(1+.278393*q6 + .230389*q6**2
+ .000972*q6**3 + .078108*q6**4)**4);
pi7 = .5 -.5*(1-1/(1+.278393*q7 + .230389*q7**2
+ .000972*q7**3 + .078108*q7**4)**4);
pi8 = .5 +.5*(1-1/(1+.278393*q8 + .230389*q8**2
+ .000972*q8**3 + .078108*q8**4)**4);
pi9 = .5 -.5*(1-1/(1+.278393*q9 + .230389*q9**2
+ .000972*q9**3 + .078108*q9**4)**4);
pi10 = .5 -.5*(1-1/(1+.278393*q10 + .230389*q10**2
+ .000972*q10**3 + .078108*q10**4)**4);
```

```
ev1 = (.2646 -.118*a1 + .0187*a1**2)*EXP(-.5*(b1/
(.7427 + .7081/a1 + .0074/a1**2))**2);
ev2 = (.2646 -.118*a2 + .0187*a2**2)*EXP(-.5*(b2/
(.7427 + .7081/a2 + .0074/a2**2))**2);
ev3 = (.2646 -.118*a3 + .0187*a3**2)*EXP(-.5*(b3/
(.7427 + .7081/a3 + .0074/a3**2))**2);
ev4 = (.2646 -.118*a4 + .0187*a4**2)*EXP(-.5*(b4/
(.7427 + .7081/a4 + .0074/a4**2))**2);
ev5 = (.2646 -.118*a5 + .0187*a5**2)*EXP(-.5*(b5/
(.7427 + .7081/a5 + .0074/a5**2))**2);
ev6 = (.2646 -.118*a6 + .0187*a6**2)*EXP(-.5*(b6/
(.7427 + .7081/a6 + .0074/a6**2))**2);
ev7 = (.2646 -.118*a7 + .0187*a7**2)*EXP(-.5*(b7/
(.7427 + .7081/a7 + .0074/a7**2))**2);
ev8 = (.2646 -.118*a8 + .0187*a8**2)*EXP(-.5*(b8/
(.7427 + .7081/a8 + .0074/a8**2))**2);
ev9 = (.2646 -.118*a9 + .0187*a9**2)*EXP(-.5*(b9/
(.7427 + .7081/a9 + .0074/a9**2))**2);
ev10 = (.2646 -.118*a10 + .0187*a10**2)*EXP(-.5*(b10/
(.7427 + .7081/a10 + .0074/a10**2))**2);

tv1 = pi1*(1-pi1)-ev1;
tv2 = pi2*(1-pi2)-ev2;
tv3 = pi3*(1-pi3)-ev3;
tv4 = pi4*(1-pi4)-ev4;
tv5 = pi5*(1-pi5)-ev5;
tv6 = pi6*(1-pi6)-ev6;
tv7 = pi7*(1-pi7)-ev7;
tv8 = pi8*(1-pi8)-ev8;
tv9 = pi9*(1-pi9)-ev9;
tv10 = pi10*(1-pi10)-ev10;

r1 = tv1/(tv1+ev1);
r2 = tv2/(tv2+ev2);
r3 = tv3/(tv3+ev3);
r4 = tv4/(tv4+ev4);
r5 = tv5/(tv5+ev5);
r6 = tv6/(tv6+ev6);
r7 = tv7/(tv7+ev7);
r8 = tv8/(tv8+ev8);
r9 = tv9/(tv9+ev9);
r10 = tv10/(tv10+ev10);

errvar = ev1 + ev2 + ev3 + ev4 + ev5 + ev6 + ev7 + ev8 + ev9 + ev10;

truvar = tv1 + tv2 + tv3 + tv4 + tv5 + tv6 + tv7 + tv8 + tv9 + tv10
+ 2*
(sqrt(tv1*tv2)+sqrt(tv1*tv3)+sqrt(tv1*tv4)+sqrt(tv1*tv5)
+sqrt(tv1*tv6)+sqrt(tv1*tv7)+sqrt(tv1*tv8)+sqrt(tv1*tv9)+sqrt(tv1*tv10)
+sqrt(tv2*tv3)+sqrt(tv2*tv4)+sqrt(tv2*tv5)
+sqrt(tv2*tv6)+sqrt(tv2*tv7)+sqrt(tv2*tv8)+sqrt(tv2*tv9)+sqrt(tv2*tv10)
+sqrt(tv3*tv4)+sqrt(tv3*tv5)
+sqrt(tv3*tv6)+sqrt(tv3*tv7)+sqrt(tv3*tv8)+sqrt(tv3*tv9)+sqrt(tv3*tv10)
+sqrt(tv4*tv5)
+sqrt(tv4*tv6)+sqrt(tv4*tv7)+sqrt(tv4*tv8)+sqrt(tv4*tv9)+sqrt(tv4*tv10)
+sqrt(tv5*tv6)+sqrt(tv5*tv7)+sqrt(tv5*tv8)+sqrt(tv5*tv9)+sqrt(tv5*tv10)
+sqrt(tv6*tv7)+sqrt(tv6*tv8)+sqrt(tv6*tv9)+sqrt(tv6*tv10)
+sqrt(tv7*tv8)+sqrt(tv7*tv9)+sqrt(tv7*tv10)
+sqrt(tv8*tv9)+sqrt(tv8*tv10)
+sqrt(tv9*tv10))
;
rel = truvar/(truvar+errvar); ! reliability for trait

OUTPUT: stdyx; cinterval;
```

Model 3: Combined Latent Speed and Trait Structural Model

```
TITLE: Combined Model for PISA2012 Data: 3 method factors, RT --> Ability
DATA: FILE="pathnamehere\PISA2012data_999.dat"; !update file name

VARIABLE:
NAMES = ID y1-y10 x1-x10;
USEVARIABLES = y1-y10 x1-x10;
CATEGORICAL = y1-y10;
MISSING = ALL(-999); ! missing data must be coded, then turn on

ANALYSIS:
    ESTIMATOR = ML; !WLSMV USES PROBIT
    LINK=LOGIT; ! automatically done with categorical indicators
    ITERATIONS = 100000;
    INTEGRATION = MONTECARLO;
    MITERATIONS = 1000;
    MCSEED = 2931;

MODEL:
! binary factor (Ability)
Ability BY y1*1.760 (p1) ! y's are for loadings; p's are for reliab (in constraints)
y2*3.429 (p2)
y3*2.196 (p3)
y4*2.164 (p4)
y5*0.845 (p5)
y6*1.446 (p6)
y7*0.944 (p7)
y8*0.672 (p8)
y9*0.960 (p9)
y10*1.117 (p10);
Ability@1; !for identifiablity in 1a
[Ability@0];
[y1$1*-0.355] (p11);
[y2$1*4.767 ] (p12);
[y3$1*2.537 ] (p13);
[y4$1*3.443 ] (p14);
[y5$1*-0.006] (p15);
[y6$1*1.532 ] (p16);
[y7$1*0.398 ] (p17);
[y8$1*-0.673] (p18);
[y9$1*1.014 ] (p19);
[y10$1*1.643] (p20);  ! ensures a 2PL model


! continuous item factor (Reaction Time)
RT BY x1* (L1)
x2* (L2)
x3* (L3)
x4* (L4)
x5* (L5)
x6* (L6)
x7* (L7)
x8* (L8)
x9* (L9)
x10*2 (L10);
x1-x10* (E1-E10); ! variances
[x1-x10*] ; ! means/intercepts
RT@1; !for identifiablity
[RT@0];

Ability ON RT*; ! regression

! Combined model method effects
Clust1 BY y1*
y2-y3*
x1-x3*;
Clust2 BY y4*
y5-y7*
x4-x7*;
```

```
Clust3 BY y8*
y9-y10*
x8-x10*;
Clust1@1;
Clust2@1;
Clust3@1;
[Clust1@0];
[Clust2@0];
[Clust3@0];
Clust1 WITH Clust2@0;
Clust1 WITH Clust3@0;
Clust2 WITH Clust3@0;
Clust1 WITH Ability@0;
Clust1 WITH RT@0;
Clust2 WITH Ability@0;
Clust2 WITH RT@0;
Clust3 WITH Ability@0;
Clust3 WITH RT@0;


MODEL CONSTRAINT: !this is to get reliability estimates (Raykov et al 2010)

NEW(
a1 a2 a3 a4 a5 a6 a7 a8 a9 a10
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10
q1 q2 q3 q4 q5 q6 q7 q8 q9 q10
pi1 pi2 pi3 pi4 pi5 pi6 pi7 pi8 pi9 pi10
tv1 tv2 tv3 tv4 tv5 tv6 tv7 tv8 tv9 tv10
ev1 ev2 ev3 ev4 ev5 ev6 ev7 ev8 ev9 ev10
r1 r2 r3 r4 r5 r6 r7 r8 r9 r10
errvar truvar rel omega
);

a1 = p1/1.702; ! loadings p1/1.702;
a2 = p2/1.702;
a3 = p3/1.702;
a4 = p4/1.702;
a5 = p5/1.702;
a6 = p6/1.702;
a7 = p7/1.702;
a8 = p8/1.702;
a9 = p9/1.702;
a10 = p10/1.702;

b1 = p11/p1; ! thresholds p11/p1;
b2 = p12/p2;
b3 = p13/p3;
b4 = p14/p4;
b5 = p15/p5;
b6 = p16/p6;
b7 = p17/p7;
b8 = p18/p8;
b9 = p19/p9;
b10 = p20/p10;

!!!! reverse sign for negative b's
! these data: 1,5,8
q1 = -a1*b1/sqrt(2+2*a1**2);
q2 = a2*b2/sqrt(2+2*a2**2);
q3 = a3*b3/sqrt(2+2*a3**2);
q4 = a4*b4/sqrt(2+2*a4**2);
q5 = -a5*b5/sqrt(2+2*a5**2);
q6 = a6*b6/sqrt(2+2*a6**2);
q7 = a7*b7/sqrt(2+2*a7**2);
q8 = -a8*b8/sqrt(2+2*a8**2);
q9 = a9*b9/sqrt(2+2*a9**2);
q10 = a10*b10/sqrt(2+2*a10**2);
```

```
!!!! add .5 instead of subtract for negative b's
! these data: 1,5,8
pi1 = .5 +.5*(1-1/(1+.278393*q1 + .230389*q1**2
+ .000972*q1**3 + .078108*q1**4)**4);
pi2 = .5 -.5*(1-1/(1+.278393*q2 + .230389*q2**2
+ .000972*q2**3 + .078108*q2**4)**4);
pi3 = .5 -.5*(1-1/(1+.278393*q3 + .230389*q3**2
+ .000972*q3**3 + .078108*q3**4)**4);
pi4 = .5 -.5*(1-1/(1+.278393*q4 + .230389*q4**2
+ .000972*q4**3 + .078108*q4**4)**4);
pi5 = .5 +.5*(1-1/(1+.278393*q5 + .230389*q5**2
+ .000972*q5**3 + .078108*q5**4)**4);
pi6 = .5 -.5*(1-1/(1+.278393*q6 + .230389*q6**2
+ .000972*q6**3 + .078108*q6**4)**4);
pi7 = .5 -.5*(1-1/(1+.278393*q7 + .230389*q7**2
+ .000972*q7**3 + .078108*q7**4)**4);
pi8 = .5 +.5*(1-1/(1+.278393*q8 + .230389*q8**2
+ .000972*q8**3 + .078108*q8**4)**4);
pi9 = .5 -.5*(1-1/(1+.278393*q9 + .230389*q9**2
+ .000972*q9**3 + .078108*q9**4)**4);
pi10 = .5 -.5*(1-1/(1+.278393*q10 + .230389*q10**2
+ .000972*q10**3 + .078108*q10**4)**4);

ev1 = (.2646 -.118*a1 + .0187*a1**2)*EXP(-.5*(b1/
(.7427 + .7081/a1 + .0074/a1**2))**2);
ev2 = (.2646 -.118*a2 + .0187*a2**2)*EXP(-.5*(b2/
(.7427 + .7081/a2 + .0074/a2**2))**2);
ev3 = (.2646 -.118*a3 + .0187*a3**2)*EXP(-.5*(b3/
(.7427 + .7081/a3 + .0074/a3**2))**2);
ev4 = (.2646 -.118*a4 + .0187*a4**2)*EXP(-.5*(b4/
(.7427 + .7081/a4 + .0074/a4**2))**2);
ev5 = (.2646 -.118*a5 + .0187*a5**2)*EXP(-.5*(b5/
(.7427 + .7081/a5 + .0074/a5**2))**2);
ev6 = (.2646 -.118*a6 + .0187*a6**2)*EXP(-.5*(b6/
(.7427 + .7081/a6 + .0074/a6**2))**2);
ev7 = (.2646 -.118*a7 + .0187*a7**2)*EXP(-.5*(b7/
(.7427 + .7081/a7 + .0074/a7**2))**2);
ev8 = (.2646 -.118*a8 + .0187*a8**2)*EXP(-.5*(b8/
(.7427 + .7081/a8 + .0074/a8**2))**2);
ev9 = (.2646 -.118*a9 + .0187*a9**2)*EXP(-.5*(b9/
(.7427 + .7081/a9 + .0074/a9**2))**2);
ev10 = (.2646 -.118*a10 + .0187*a10**2)*EXP(-.5*(b10/
(.7427 + .7081/a10 + .0074/a10**2))**2);

tv1 = pi1*(1-pi1)-ev1;
tv2 = pi2*(1-pi2)-ev2;
tv3 = pi3*(1-pi3)-ev3;
tv4 = pi4*(1-pi4)-ev4;
tv5 = pi5*(1-pi5)-ev5;
tv6 = pi6*(1-pi6)-ev6;
tv7 = pi7*(1-pi7)-ev7;
tv8 = pi8*(1-pi8)-ev8;
tv9 = pi9*(1-pi9)-ev9;
tv10 = pi10*(1-pi10)-ev10;

r1 = tv1/(tv1+ev1);
r2 = tv2/(tv2+ev2);
r3 = tv3/(tv3+ev3);
r4 = tv4/(tv4+ev4);
r5 = tv5/(tv5+ev5);
r6 = tv6/(tv6+ev6);
r7 = tv7/(tv7+ev7);
r8 = tv8/(tv8+ev8);
r9 = tv9/(tv9+ev9);
r10 = tv10/(tv10+ev10);

errvar = ev1 + ev2 + ev3 + ev4 + ev5 + ev6 + ev7 + ev8 + ev9 + ev10;

truvar = tv1 + tv2 + tv3 + tv4 + tv5 + tv6 + tv7 + tv8 + tv9 + tv10
+ 2*
(sqrt(tv1*tv2)+sqrt(tv1*tv3)+sqrt(tv1*tv4)+sqrt(tv1*tv5)
```

```
+sqrt(tv1*tv6)+sqrt(tv1*tv7)+sqrt(tv1*tv8)+sqrt(tv1*tv9)+sqrt(tv1*tv10)
+sqrt(tv2*tv3)+sqrt(tv2*tv4)+sqrt(tv2*tv5)
+sqrt(tv2*tv6)+sqrt(tv2*tv7)+sqrt(tv2*tv8)+sqrt(tv2*tv9)+sqrt(tv2*tv10)
+sqrt(tv3*tv4)+sqrt(tv3*tv5)
+sqrt(tv3*tv6)+sqrt(tv3*tv7)+sqrt(tv3*tv8)+sqrt(tv3*tv9)+sqrt(tv3*tv10)
+sqrt(tv4*tv5)
+sqrt(tv4*tv6)+sqrt(tv4*tv7)+sqrt(tv4*tv8)+sqrt(tv4*tv9)+sqrt(tv4*tv10)
+sqrt(tv5*tv6)+sqrt(tv5*tv7)+sqrt(tv5*tv8)+sqrt(tv5*tv9)+sqrt(tv5*tv10)
+sqrt(tv6*tv7)+sqrt(tv6*tv8)+sqrt(tv6*tv9)+sqrt(tv6*tv10)
+sqrt(tv7*tv8)+sqrt(tv7*tv9)+sqrt(tv7*tv10)
+sqrt(tv8*tv9)+sqrt(tv8*tv10)
+sqrt(tv9*tv10))
;

rel = truvar/(truvar+errvar); ! reliability for trait

omega =
(L1+L2+L3+L4+L5+L6+L7+L8+L9+L10)^2/
((L1+L2+L3+L4+L5+L6+L7+L8+L9+L10)^2
+(E1+E2+E3+E4+E5+E6+E7+E8+E9+E10)); ! reliability for speed


OUTPUT: stdyx; cinterval;
```