Policy Research Working Paper 8986

Test Score Gap Robustness to Scaling

The Scale Transformation Command

Andres Yi Chang

WORLD BANK GROUP
Human Development Global Practice
August 2019

Policy Research Working Paper 8986

Abstract

Social scientists frequently rely on the cardinal comparability of test scores to assess achievement gaps between population subgroups and their evolution over time. This approach has been criticized due to the ordinal nature of test scores and the sensibility of results to order-preserving transformations, which are theoretically plausible. Bond and Lang (2013) document the sensitivity of measured ability to scaling choices and develop a method to assess the robustness of changes in ability over time to scaling choices. This paper presents the scale transformation command, which expands the Bond and Lang method to more general cases and optimizes their algorithm to work with large data sets. The program assesses the robustness of an achievement gap between two subgroups to any arbitrary choice of scale by finding bounds for the original gap estimation. Additionally, the program finds scale transformations that are very likely and unlikely to benchmark against the results obtained. Finally, the program also allows the user to measure how much gap growth coefficients change when including controls in their specifications.

This paper is a product of the Human Development Global Practice. It is part of a larger effort by the World Bank to provide open access to its research and make a contribution to development policy discussions around the world. Policy Research Working Papers are also posted on the Web at http://www.worldbank.org/prwp. The author may be contacted at ayichang@worldbank.org.

Test Score Gap Robustness to Scaling: The *scale_transformation* Command

Andres Yi Chang
World Bank Group
Washington, D.C.
ayichang@worldbank.org

Keywords: STATA, scale_transformation, test scores, measurement, achievement gaps, robustness to scaling, psychometrics

JEL Classification: C15, C49, C87, I29

1 Introduction

With enrollments at a historical high for low- and middle-income countries, the focus in education has shifted away from enrollment and toward improving the quality of education (World Bank World Development Report, 2018). Countries around the world are focusing on both increasing learning and closing substantial gaps across sub-populations, particularly in favor of disadvantaged groups. These gaps have been extensively documented in the United States, where a large literature has identified stubborn test score gaps by race (Bertrand and Pan 2013; Fryer and Levitt 2010), socio-economic status (Reardon 2011, 2013) and sex (Hanushek and Rivkin 2006; Clotfelter et al. 2009; Fryer and Levitt 2004, 2006, 2013).

Gap measurement is essential to make inferences about equitable learning. What is surprising though, despite the seemingly large gaps across sub-groups, is that different studies often come to different conclusions. For instance, Fryer and Levitt (2004) first noted that a substantial black-white test score gap that had received extensive attention in the literature (see Coleman et al. 1981; Kaufman and Kaufman 1983; Krohn and Lamp 1989; Phillips et al. 1998; Phillips 2000) disappeared after controlling for covariates. Additionally, they found that over the first 4 years of school, blacks lose about -.10 standard deviations per year compared to other races. Bond and Lang (2013) then demonstrated that even this seemingly rock-solid racial gap in the United States is in fact, not robust to arbitrary scale transformations. That is, transformations that minimize or maximize learning by weighting test-items differently, while still preserving order, create lower and upper bounds that are too wide to be informative or definitive.

These findings contribute to an ongoing debate around test score measurement and, in particular, on whether the assumptions that justify cardinality are satisfied in most tests (Ballou 2009; Ho 2009; Bond and Lang 2013; Jacob and Rothstein 2016). Three alternatives have been proposed. The first is to argue that the test at hand has been constructed in a manner that allows for cardinal measures of ability; this is the approach of Rasch measurement and Domingue (2014) provides statistical methods to verify the assumptions required for cardinality. The second is to give up on cardinality entirely and focus only on ordinal comparisons. Reardon et al.'s (2017) *HETOP*, Williams's

(2011) *oglm*, among others, as well as other built-in Stata commands, are designed to help users follow this approach.

A third approach, which this paper focuses on, is to assess the robustness of findings to arbitrary scaling choices as proposed by Bond and Lang (2013). Specifically, their method builds on the idea that any monotone transformation of an ordinal scale is rank preserving and is therefor an equally valid measure of the underlying variable. They present an optimization method that searches for monotone transforms that minimize and maximize test score gaps, thus allowing researchers to provide bounds to their estimates of test score evolution across sub-groups over time. Here, I extend their technique to more general cases and optimize their algorithm to work with large data sets. This allows us to simulate multiple monotone transforms several times and bound differences by choosing transformations that maximize or minimize differences between sub-groups.

2 Robustness to Scaling

The ordinal nature of test scores implies that any monotonic transformation to a test score scale is potentially valid. We see this concept applied when colleges arbitrarily put more weight on grades from "harder" classes for admission purposes or when, within a particular test, a teacher decides that one question is worth more points because it is more "challenging". For this reason, when comparing test scores, we can only ascertain that someone with a high test score performed better than someone with a low one. It is much more difficult to know by how much. Standardizing test scores, which is the most common scale transformation, does not solve this issue. In fact, any analysis that relies on comparing standardized test scores implicitly assumes that test scores are interval scales. This assumption means that, in a scale from 0 to 100 points, the difference in underlying knowledge between individuals who scored 0 and 10 points is the same as that of those who score 90 and 100 points. In reality, this is seldom the case as there is no theoretical justification for this assumption: most exams have questions with varying level of difficulty and utilize scales that are arbitrary in their mean, range and distribution. Without interval scale properties, mean differences and standard deviations can be distorted depending on the weight each question receives.

Even when we equate test scores using Item Response Theory (IRT), most of the time, the underlying θ ability parameters obtained are ordinal in nature and defined only up to a linear transformation (see Lord (1975); Ballou (2009) for a detail explanation). Only in the very special case where all questions are equally informative (i.e. Rasch model), IRT scales can be interpreted as interval scales. Nonetheless, this assumption is rarely fulfilled.

An assumption that is more reasonable and appropriate in most settings is that test scores are ordinal. Unlike, temperature and distance where the magnitude of a given difference corresponds to the same underlying measurement at different parts of the distribution, test scores differences of the same "magnitude" can mean very different things at different points of the distribution. The only inference that we can make

between someone with a score of 90 points and another person with a score of 100 is that the individual with a score of 100 demonstrated a larger amount of underlying knowledge or ability than that of the individual who scored 90.

In their 2013 paper, Bond and Lang noted that the current literature does not adequately ensure that results are robust to scaling transformations and devised a methodology to find bounds for estimates measuring differences in test scores across sub-groups and over time, such as the black-white test score gap in the United States. In particular, they created an algorithm that finds the upper and lower bounds for the estimates of differences in test scores across two sub-groups at two points of time. In order to do so, they transform test scores using a 6th-degree polynomial monotonic transformation and optimize over the test score gap growth (i.e. difference in test scores between two sub-groups in t_0 minus the difference in t_1) to make it as large or small as possible.

In the next section, I describe the step-by-step methodology developed by Bond and Lang and how the `scale_transformation` command extends their methodology for general use to a wide range of applications in this literature.

3 The scale_transformation Command

3.1 Bond and Lang Methodology

The Bond and Lang (2013) methodology tests the robustness to scaling of sub-group differences using arbitrary monotone transformations of an underlying ability or test score distribution. Relying on 6th-degree polynomial transformations, the original code tries to minimize or maximize the differences in test score growth between any two sub-groups. These growth-minimizing and growth-maximizing transformations present lower and upper-bounds respectively of the differences between sub-groups.

Step-by-step methodology

1. Define the objective function as follows

$$GapGrowth = \alpha_{(1,t=1)} - \alpha_{(1,t=0)}, \tag{1}$$

   where $\alpha_{(1,t)}$ is the difference in test scores between groups at time t (see Step 2d for more details on how to obtain $\alpha_{(1,t)}$). Note that test scores need to be already vertically equated. That is, they should be completely comparable and mapped into a single scale regardless of the time in which they were captured.

2. Optimize the following function over the defined objective:

   a. Take the original test scores and scale them down to be between 0 and 1 (for computational speed).

   b. Transform original test scores (s) using a 6th-degree polynomial monotonic transformation $T(s)$, optimizing over the gap growth by choosing 5 coeffi-

cients ($\beta_2, \beta_3, \beta_4, \beta_5$ and β_6) and 1 constant (c) for the polynomial transformation:

$$T(s) = \beta_1(s-c)+\beta_2(s-c)^2+\beta_3(s-c)^3+\beta_4(s-c)^4+\beta_5(s-c)^5+\beta_6(s-c)^6 \quad (2)$$

c. Standardize transformed test scores in both periods (i.e. t_0 and t_1) together. The original Bond and Lang code standardizes test scores separately, which could artificially increase or decrease the resulting gap. By jointly standardizing both years, we ensure that they remain vertically equated and that the resulting gap is only due to the transformations.

d. Run an OLS regression separately for each period using a subgroup dummy and controlling for other desired variables. Given this set-up, the coefficient on the subgroup dummy can be interpreted as the difference between each sub-group.

$$T(s)_t = \alpha_{(0,t)} + \alpha_{(1,t)}GroupDummy + \theta_t Controls \quad (3)$$

e. Find gap growth between t_0 and t_1, as defined in Step 1.

3. Run Step 2 multiple times from different random starting values, checking that transformations preserve order (monotonic rule).

3.2 Improvements

The process described in section 3.1 is complex and computationally demanding. Most of the original public code is written in Mata, which might restrict wider use among researchers. Furthermore, the public code was written for the specific estimation in their paper. Thus, it makes some assumptions that might not necessarily extrapolate to other data and might affect the speed or magnitude of the results. This left space for two areas of improvement: (a) making the methodology more accessible to Stata users, and (b) improving the precision, flexibility and efficiency of the methodology.

In terms of accessibility, the `scale_transformation` command, together with this paper and its documentation will allow any Stata user to perform these robustness checks in one line of code.

Methodologically, the original process also sacrificed precision for efficiency. For instance, the parameter β_1 was not optimized and the monotonicity rule used did not check every single plausible value but rather checked monotonicity in small equally-spaced intervals. The `scale_transformation` command improves on these aspects as well as on computational speed.

Specifically, the original methodology has been improved by computing more accurate estimates (in quad precision) using QR decomposition tools in Mata. The monotonicity rule has also been enhanced to be more accurate and flexible with 3 variants: (a) a default rule that checks for monotonicity at every possible score value up to 4 decimals within the plausible range in a transformed scale between 0 and 1; (b) a quick

rule, which only checks for monotonicity at the unique scores values present in the sample data; and (c) a theoretical rule that checks for monotonicity at every possible value using an external file. The latter is of particular value if test scores come from a larger data set that has scores for multiple years (which are not included in the optimization) or includes scores for sub-groups not being analyzed. Furthermore, other options such as time-specific controls and weights have been included, as well as options to specify the number of times the program will run (yielding different results), the optimization methodology to be used at certain stages of the process, and the maximum number of iterations before the optimization cycle stops and returns results as if convergence was achieved.

Another important improvement of the `scale_transformation` command is that it is robust to transformations that fully change the sign of the gap regardless of its initial value. Since the optimization problem for the maximization or minimization gap growth options is complex and does not have a closed-form solution, in some specific cases the maximization could yield the solutions for the minimization problem (and vice versa). The `robust(`*integer*`)` option allows the program to use the first $K = integer$ iterations to check that the sign of the optimization is correct.

Finally, other robustness checks used by Bond and Lang have been added to the program. In particular, the `scale_transformation` command allows to search for transformations that maximize or minimize the correlation between initial and final test scores by either looking at the binary correlation between them or the resulting R-square from regressing initial test scores on final test scores. These transformations allow the user to benchmark a likely or unlikely transformation and compare it to the results obtained when maximizing or minimizing the gap. Likewise, the program allows users to optimize over the absolute value of the difference between the Gap Growth calculated with and without controls. This additional check helps users to measure how much Gap Growth coefficients could change when including controls to their specifications.

3.3 Description

The `scale_transformation` command finds a monotonic transformation for a test score scale, which optimizes a specific objective function. The optimization object includes scores at two points of time for two comparison groups (e.g. rich vs poor). This program performs a grid search from multiple random initial parameters to find the desired values. All the transformations found by this command are theoretically plausible given the ordinal nature of test scores.

A 6th-degree polynomial monotonic transformation is used because it provides flexibility to approach numerous continuous functions. Nevertheless, monotonicity checks need to be applied as this type of transformation does not necessarily preserve order. Furthermore, the monotonicity restriction introduces a discontinuity into the objective function that creates multiple local maxima and minima. These characteristics yield a complex optimization problem that does not have a closed-form solution.

The `scale_transformation` program takes advantage of the Mata [M-5] **optimize**

function and performs a grid search to find multiple solutions. The results are reported in a data set format that includes the values of the evaluated objective functions, the resulting parameters of each optimization iteration and the initial parameters that yielded that result.

To install the command on your system, open Stata and type:

`. ssc install scale_transformation`.

You can also find the latest version of the program at:

`https://github.com/andresyichang/scale_transformation`.

3.4 Syntax

`scale_transformation , type(`*integer*`) score1(`*varname*`) score2(`*varname*`) [`
`compgroup(`*varname*`) controls(`*varlist*`) controls1(`*varlist*`) controls2(`*varlist*`)`
`weights(`*varname*`) iterations(`*integer*`) maxoptiterations(`*integer*`)`
`singhmethod(`*integer*`) bounddown(`*integer*`) boundup(`*integer*`)`
`monotonicity(`*integer*`) monofile(`*filename*`) timeroff save(`*filename*`)`
`seed(`*integer*`) robust(`*integer*`) ]`

3.5 Main Options (Required)

`type(`*integer)* indicates the type of optimization and the objective function to be performed. The options are as follows:

1. Gap Growth Maximization
2. Gap Growth Minimization
3. Correlation Maximization
4. Correlation Minimization
5. R-squared Maximization
6. R-squared Minimization
7. Controls Explanation Maximization

`score1(`*varname*`)` specifies the variable that contains scores at (initial) period 1. In order to improve processing speed, this variable is automatically scaled to be between 0 and 1 prior to running the program. The transformation used for this process is the same for variables score1 and score2, and does not affect the resulting estimations in any way.

`score2(`*varname*`)` specifies the variable that contains scores at (final) period 2. In order to improve processing speed, this variable is automatically scaled to be between 0 and 1 prior to running the program. The transformation used for this process is the same for variables score1 and score2, and does not affect the resulting estimations in any way.

3.6 Secondary Options

`compgroup(`*varname*`)` specifies the variable that contains the group classification to be analyzed and from which the program will compare the top and bottom groups. This variable needs to be numeric. The program will take the lowest and highest values (i.e. groups) and compare them, while controlling for all other groups in the middle at the same time. Note that the variable compgroup should only take on integers greater or equal to 0 (all missings will be ignored). Only allowed when optimizing Gap Growth or Controls Explanation (i.e. type 1, 2 or 7).

`controls(`*varlist*`)` include varlist as controls at both period 1 and 2. Variables should be numeric as they will be used directly in regressions. Only allowed when optimizing Gap Growth or Controls Explanation (i.e. type 1, 2 or 7).

`controls1(`*varlist*`)` include varlist as controls at period 1 only. Variables should be numeric as they will be used directly in regressions. Only allowed when optimizing Gap Growth or Controls Explanation (i.e. type 1, 2 or 7).

`controls2(`*varlist*`)` include varlist as controls at period 2 only. Variables should be numeric as they will be used directly in regressions. Only allowed when optimizing Gap Growth or Controls Explanation (i.e. type 1, 2 or 7).

`weights(`*varname*`)` uses varname as inverse probability weights. If not specified, weights are set to 1.

`iterations(`*integer*`)` specifies number of times that the program will find optimal values using unique random-generated initial parameters. The default is set to `iterations(1000)`. The more iterations, the longer it will take the program to run and finish. Note that by default, if the program is stopped before ending, the results will not be stored. It is recommended that you test your program with a low number of iterations.

`maxoptiterations(`*integer*`)` specifies maximum number of iterations before the optimization program stops and return results as if convergence was achieved. The default is set to `maxoptiterations(25)`, which in practice should be enough to achieve 'convergence' with a fair degree of accuracy. This limit is necessary as the monotonicity restriction and complexity of the objective functions cause the optimization program to run indefinitely otherwise. See [M-5] **mf_optimize_maxiter** for more details.

`singhmethod(`*integer*`)` specifies what the optimizer should do when, at an iteration step, it finds that H is singular. The default is set to singhmethod(1) for "hybrid" but can be changed to 2 for "modified Marquardt algorithm". See [M-5] **mf_optimize_singularH**

for more details.

`bounddown(`*integer*`)` specifies lower bound for the random-generated initial parameters for the optimization. This program creates random initial values and then uses them to find all the different local maxima or minima, accordingly. The default is set to `bounddown(-1500)`.

`boundup(`*integer*`)` specifies upper bound for the random-generated initial parameters for the optimization. This program creates random initial values and then uses them to find all the different local maxima or minima accordingly. The default is set to `boundup(1500)`.

`monotonicity(`*integer*`)` specifies the type of monotonicity check to be applied. The default is set to `monotonicity(1)` for "Standard", which checks for monotonicity at every possible score value (up to 4 decimals within the plausible range) in the original scale that was transformed to be between 0 and 1. When less precision is needed, monotonicity can be set to 2 for "Sample", which only checks for monotonicity at the unique scores values present in the sample data. If the test scores come from a larger data set that have scores for multiple years, it is possible to do a more thorough check (but more time consuming) by putting together an external file with all theoretical or observed plausible scores where you want the program to check for monotonicity. For the latter, you must set monotonicity to 3 for "External" and include the filename in `monofile(`*filename*`)`.

`monofile(`*filename*`)` specifies the file to be used when monotonicity type is set to 3 for "External". This option should only be used with `monotonicity(3)`. You can use .dta, .csv, .xls or .xlsx files. Make sure that .csv, .xls and .xlsx files have the name of the variable on the first row.

`timeroff` turn off the timer. The default is set to include the time (in minutes) that the program took to run, which is reported only if the program finishes.

`save(`*filename*`)` save optimization results using filename once program is done.

`seed(`*integer*`)` set seed for replication, where 0 <= integer <= 2,147,483,647. If missing, the seed is randomly generated and included in the header of the program. Always save the log file to ensure you are able to recover the seed number for replication.

`robust(`*integer*`)` indicates the program to use the first $K = integer$ iterations to check that the sign of the optimization is correct. Since the optimization problem for the maximization or minimization Gap Growth options (i.e. type 1 or 2) is complex and does not have a closed-solution, in some specific cases the maximization could yield the solutions for the minimization problem (and vice versa). In order to address this issue, the user could activate the robust option to use the first $K = integer$ (where $20 <= K <= 300$; $K = even$) iterations (in addition to N specified by the iterations option) to check for a possible change of sign in Max/Min Gap Growth optimization results. Half of the K iterations are used to find the maximization solution while

the other half K iterations are used for the minimization problem. Once the correct approach is selected, the program keeps the solutions for the correct specification and runs the remaining N iterations indicated in the `iterations()` option.

3.7 Optimization Objectives

The `scale_transformation` command finds a 6th-degree polynomial monotonic transformation for a test score scale that optimizes one of the following:

1. Gap Growth: gap difference between the bottom and top groups from the compgroup variable in period 2, minus the same gap difference in period 1. In other words, it is the difference between the regression coefficient of the top compgroup indicator (controlling for groups in the middle) in period 2, minus the same regression coefficient of the top compgroup indicator (also controlling for other groups in the middle) in period 1. In both, the coefficient of the top compgroup indicator measures the difference between the top and bottom groups. Note that the Gap Growth is positive when the Gap widens and negative when the Gap shrinks (always with respect to the Gap in period 1). Thus, the Gap Growth will always be negative when the coefficient changes signs between period 1 and 2 (regardless of whether it is from + to -, or from - to +). Conversely, it will be positive when the gap widens, for instance, from -.1 to -.3;
2. Correlation: coefficient from regressing test scores in period 1 on test scores in period 2 (no controls allowed);
3. R-squared: from regressing test scores in period 1 on test scores in period 2 (no controls allowed); or
4. Controls Explanation: measures how much Gap Growth coefficients change when including controls in `controls1(`*varlist*`)` and `controls2(`*varlist*`)` - not in `controls(`*varlist*`)`. This allows to test the explanatory power of only a subset of desired control variables. That means that, if you want to measure how much Gap Growth coefficients change by introducing time-invariant controls, you must include these variables in both `controls1(`*varlist*`)` and `controls2(`*varlist*`)`. The Controls Explanation term is estimated by taking the distance (i.e. absolute value of the difference) between the Gap Growth calculated without variables included in `controls1` and `constrols2`, minus the Gap Growth calculated with these two controls varlists. To make the interpretation easier, both Gap Growths coefficients - with and without controls - are included in the results.

4 Resulting Output

4.1 Example

In this section, we use the `timss_testscores.dta` ancillary data set file to find the Gap Growth maximization solution for test scores between year 1 and 2, comparing males

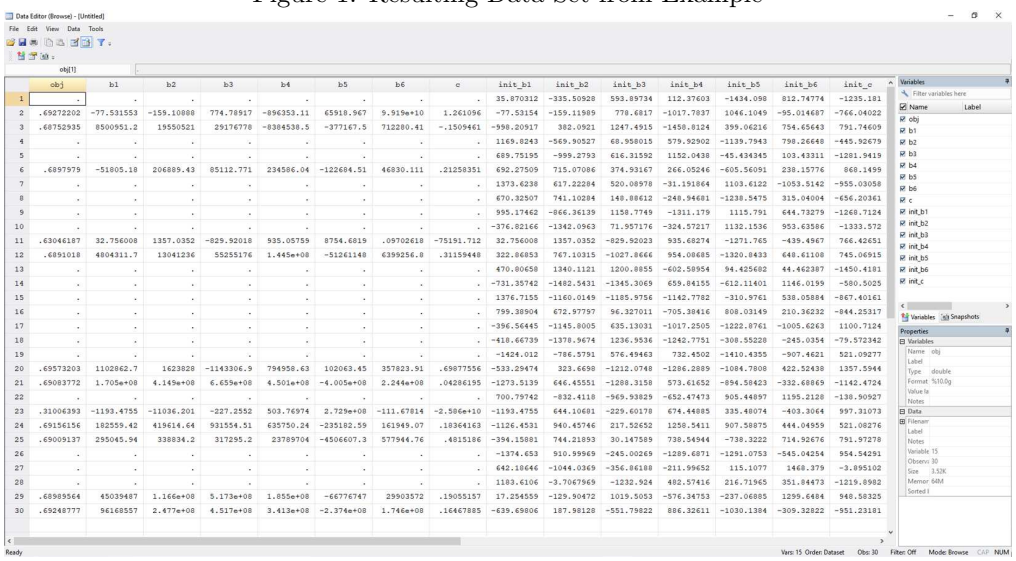
and females:

```
. ssc install scale_transformation

. sysuse timss_testscores.dta, clear

. scale_transformation, type(1) score1(score1) score2(score2) ///
    compgroup(sex) iterations(20) maxoptiterations(15) mono(2) ///
    seed(562) robust(20)
```

Note that, given the above command, the program will run 40 times from different initial parameters and each time, the program will report convergence after 15 iterations, performing a "Sample" monotonicity check. Given the option `robust(20)`, the program will use the first 20 simulations to check that the sign of the gap is correct by carrying out 10 maximizations and 10 minimizations. Once the correct direction is confirmed, the correct half is kept and the other discarded. Thus, in this example, the resulting data set will contain only 30 observations.

Figure 1 shows the resulting data set from the Gap Growth maximization example above. The missing values for the optimization object and transformation coefficients (i.e. columns 1 through 8) correspond to transformations that are not order preserving. Thus, they are omitted in the final data set.

Figure 1: Resulting Data Set from Example



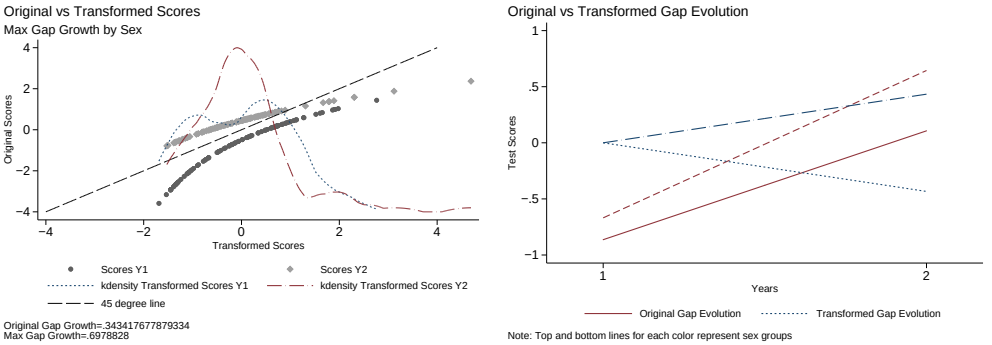
	obj	b1	b2	b3	b4	b5	b6	c	init_b1	init_b2	init_b3	init_b4	init_b5	init_b6	init_c
1	.	.	.	.	.	.	.	.	35.870312	-335.50928	593.89734	112.37603	-1434.098	812.74774	-1235.181
2	.69272202	-77.531553	-159.10888	774.78917	-896353.11	65918.967	9.919e+10	1.261096	-77.53154	-159.11989	778.6817	-1017.7837	1046.1049	-95.014687	-766.04022
3	.68752935	8500951.2	19550521	29176778	-8384538.5	-377167.5	712280.41	-.1509461	-998.20917	382.0921	1247.4915	-1458.8124	399.06216	754.65643	791.74609
4	.	.	.	.	.	.	.	.	1169.8243	-569.90527	68.958015	579.92902	-1139.7943	798.26648	-445.92679
5	.	.	.	.	.	.	.	.	689.75195	-999.2793	616.31592	1152.0438	-45.434345	103.43311	-1281.9419
6	.6897579	-51805.18	206889.43	85112.771	234586.04	-122684.51	46830.111	.21258351	692.27509	715.07086	374.93167	266.05246	-605.56091	238.15776	868.1499
7	.	.	.	.	.	.	.	.	1373.6238	617.22284	520.08978	-31.151864	1103.6122	-1053.5142	-955.03058
8	.	.	.	.	.	.	.	.	670.32507	741.10284	148.88612	-248.94681	-1238.5475	315.04004	-656.20361
9	.	.	.	.	.	.	.	.	995.17462	-866.36139	1158.7749	-1311.179	1115.791	644.73279	-1268.7124
10	.	.	.	.	.	.	.	.	-376.82166	-1342.0963	71.957176	-324.57217	1132.1536	953.63586	-1333.572
11	.63046187	32.756008	1357.0352	-829.92018	935.05759	8754.6819	.09702618	-75191.712	32.756008	1357.0352	-829.92023	935.68274	-1271.765	-439.4967	766.42651
12	.6891018	4804311.7	13041236	55255176	1.445e+08	-51261140	6399256.8	.31159448	322.86853	767.10315	-1027.8666	954.08685	-1320.8433	648.61108	745.06915
13	.	.	.	.	.	.	.	.	470.80658	1340.1121	1200.8855	-602.58954	94.425682	44.462387	-1450.4181
14	.	.	.	.	.	.	.	.	-731.35742	-1482.5431	-1345.3069	659.84155	-612.11401	1146.0199	-580.5025
15	.	.	.	.	.	.	.	.	1376.7155	-1160.0149	-1185.9756	-1142.7782	-310.9761	538.05884	-867.40161
16	.	.	.	.	.	.	.	.	799.38904	672.97797	96.327011	-705.38416	808.03149	210.36232	-844.25317
17	.	.	.	.	.	.	.	.	-396.56445	-1145.8005	635.13031	-1017.2505	-1222.8761	-1005.6263	1100.7124
18	.	.	.	.	.	.	.	.	-418.66739	-1378.9674	1236.9536	-1242.7751	-308.55228	-245.0354	-79.572342
19	.	.	.	.	.	.	.	.	-1424.012	-786.5791	576.49463	732.4502	-1410.4355	-907.4621	521.09277
20	.69573203	1102862.7	1623828	-1143306.9	794958.63	102063.45	357823.91	.69877556	-533.29474	323.6698	-1212.0748	-1286.2889	-1084.7808	422.52438	1357.5944
21	.69083772	1.705e+08	4.149e+08	6.659e+08	4.501e+08	-4.005e+08	2.244e+08	.04286195	-1273.5139	646.45551	-1288.3158	573.61652	-894.58423	-332.68869	-1142.4724
22	.	.	.	.	.	.	.	.	700.79742	-832.4118	-969.93829	-652.47473	905.44857	1195.2128	-138.90927
23	.31006393	-1193.4755	-11036.201	-227.2552	503.76974	2.729e+08	-111.67814	-2.586e+10	-1193.4755	644.10681	-229.60178	674.44885	335.48074	-403.3064	997.31073
24	.69156156	182559.42	419614.64	931554.51	635750.24	-235182.59	161949.07	.18364163	-1126.4531	940.45746	217.52652	1258.5411	907.58875	444.04959	521.08276
25	.69009137	295045.94	338834.2	317295.2	23789704	-4506607.3	577944.76	.4815186	-394.15881	744.21893	30.147589	738.54944	-738.3222	714.92676	791.97278
26	.	.	.	.	.	.	.	.	-1374.653	910.99969	-245.00269	-1289.6871	-1291.0753	-545.04254	954.54291
27	.	.	.	.	.	.	.	.	642.18646	-1044.0369	-356.86188	-211.99652	115.1077	1468.379	-3.895102
28	.	.	.	.	.	.	.	.	1183.6106	-3.7067969	-1232.924	482.57416	216.71965	351.84473	-1219.8982
29	.68985564	45039487	1.166e+08	5.173e+08	1.855e+08	-66776747	25903572	.19055157	17.254559	-129.90472	1019.5053	-576.34753	-237.06885	1299.6484	948.58325
30	.69248777	96168557	2.477e+08	4.517e+08	3.413e+08	-2.374e+08	1.746e+08	.16467885	-639.69806	187.98128	-551.79822	886.32611	-1030.1384	-309.32822	-951.23181

For this particular example, we can also plot the original and transformed scores to understand what the program did. Since we are searching for the the Gap Growth

maximization (i.e. `type(1)`), we select the transformation that yields the highest Gap Growth from the resulting data set, which is equal to .695732. Then, we use the corresponding coefficients and constant and apply the 6th-degree polynomial transformation in equation (2). Figure 2 plots the original versus transformed scores on the left and the test score gap evolution by sex between year 1 and 2. This particular Gap Growth maximization makes the gap in year 1 as small as possible while, at the same time, making the gap in year 2 as large as it can.

Note that the original gap magnitude was .343418, about half the size of the transformed gap. These results are particular to this specific scenario and, in practice, the algorithm might behave differently depending on the data and the objective function at hand.

Figure 2: Original vs Transformed Scores



4.2 Interpretation

Regardless of the optimization object chosen, the resulting data set has a similar structure. Each row contains the results for one simulation. The `obj` variable contains the optimization object, while the rest of the variables contain the parameters of the 6th-degree polynomial transformation that achieves that object and the initial random starting values for these parameters. This format allows the user to not only have the full distribution of results but also access the necessary parameters to replicate the desired scale transformation if needed.

For the Gap Growth maximizing and minimizing options, one should be aware that the program finds the most extreme transformations, some of which might be extremely unlikely (although theoretically plausible). Thus, for instances where the bounds are too wide, it might be useful to benchmark against likely transformations (such as those found by the `scale_transformation` program that maximize correlation or R-squared) and explore other properties of test scores that could help us discard some of the most extreme and very unlikely transformations. For instance, Ho and Yu (2015) present values for skewness and kurtosis from 330 United States state-level examinations. Based on this, one could look at the moments of the resulting distributions and discard those

that are extremely rare and unlikely to happen, thus narrowing the space between the resulting bounds.

5 Conclusion

One simple step that researchers can take in order to address some the challenges of scaling and the ordinal nature of test scores is to test the robustness of their results to changes in the test score scales. The `scale_transformation` command allows anyone to implement several robustness checks to scaling decisions, particularly in the context of analyzing achievement gaps between sub-groups in a panel setting. These checks build on the methodology developed by Bond and Lang (2013) and allow users to find bounds to the original gap estimation or transformations that are very likely or unlikely so as to benchmark against other results obtained. In practice, these are some of the essential checks that should be done when comparing test score averages over time and across sub-groups.

When using the `scale_transformation` command, the researcher recognizes the ordinality of test scores and tests the robustness of her/his results to arbitrary monotonic transformations. This approach is more defensible on a theoretical basis and has the potential to discern real gaps from results that are only significant due to the arbitrary choice of scales made or taken as given.

6 References

Ballou, D. 2009. Test scaling and value-added measurement. *Education finance and Policy* 4(4): 351–383.

Bertrand, M., and J. Pan. 2013. The Trouble with Boys: Social Influences and the Gender Gap in Disruptive Behavior. *American Economic Journal: Applied Economics* 5(1): 32–64. http://www.aeaweb.org/articles?id=10.1257/app.5.1.32.

Bond, T., and K. Lang. 2013. The Evolution of the Black-White Test Score Gap in Grades K3: The Fragility of Results. *The Review of Economics and Statistics* 95(5): 1468–1479. https://ideas.repec.org/a/tpr/restat/v95y2013i5p1468-1479.html.

Clotfelter, C., H. Ladd, and J. Vigdor. 2009. The Academic Achievement Gap in Grades 3 to 8. *The Review of Economics and Statistics* 91(2): 398–419. https://doi.org/10.1162/rest.91.2.398.

Coleman, J. S., E. Q. Campbell, C. J. Hobson, J. McPartland, A. M. Mood, F. D. Weinfeld, and R. L. York. 1981. Equality of Educational Opportunity (Washington, DC: US Government Printing Office, 1966). *The basic findings of the report have been affirmed again and again in the research literature. See Kahlenberg, All Together Now* 25–35.

Domingue, B. 2014. Evaluating the equal-interval hypothesis with test score scales. *Psychometrika* 79(1): 1–19.

Filmer, D. P., H. Rogers, S. Al-Samarrai, M. Bendini, T. Bteille, D. Evans, M. Kivine, S. Sabarwal, and A. Valerio. 2018. World Development Report 2018: Learning to Realize Education's Promise. Technical report. https://doi.org/10.1177/0013164414548576.

Fryer, R., and S. Levitt. 2004. Understanding The Black-White Test Score Gap in the First Two Years of School. *The Review of Economics and Statistics* .

———. 2006. The Black-White Test Score Gap Through Third Grade. *American Law and Economics Review* 8(2): 249–281. http://dx.doi.org/10.1093/aler/ahl003.

———. 2010. An Empirical Analysis of the Gender Gap in Mathematics. *American Economic Journal: Applied Economics* 2(2).

———. 2013. Testing for Racial Differences in the Mental Ability of Young Children. *American Economic Review* 103(2): 981–1005. http://www.aeaweb.org/articles?id=10.1257/aer.103.2.981.

Hanushek, E. A., and S. G. Rivkin. 2006. School Quality and the Black-White Achievement Gap. Working Paper 12651, National Bureau of Economic Research. http://www.nber.org/papers/w12651.

Ho, A. D. 2009. A Nonparametric Framework for Comparing Trends and Gaps Across Tests. *Journal of Educational and Behavioral Statistics* 34(2): 201–228. https://doi.org/10.3102/1076998609332755.

Ho, A. D., and C. C. Yu. 2015. Descriptive Statistics for Modern Test Score Distributions: Skewness, Kurtosis, Discreteness, and Ceiling Effects. *Educational and Psychological Measurement* 75(3): 365–388. https://doi.org/10.1177/0013164414548576.

Jacob, B., and J. Rothstein. 2016. The measurement of student ability in modern assessment systems. *Journal of Economic Perspectives* 30(3): 85–108. https://pubs.aeaweb.org/doi/pdfplus/10.1257/jep.30.3.85.

Kaufman, A. S., and N. L. Kaufman. 1983. *K-ABC: Kaufman assessment battery for children: Interpretive manual*. American Guidance Service.

Krohn, E. J., and R. E. Lamp. 1989. Concurrent validity of the Stanford-Binet fourth edition and K-ABC for head start children. *Journal of School Psychology* 27(1): 59–67.

Lord, F. M. 1975. Evaluation with artificial data of a procedure for estimating ability and item characteristic curve parameters. *ETS Research Report Series* 1975(2).

Phillips, M. 2000. Understanding ethnic differences in academic achievement: Empirical lessons from national data. *Analytic issues in the assessment of student achievement* 103–32.

Phillips, M., J. Brooks-Gunn, G. J. Duncan, P. Klebanov, and J. Crane. 1998. Family background, parenting practices, and the Black–White test score gap. .

Reardon, S., B. Shear, K. Castellano, and A. Ho. 2017. Using Heteroskedastic Ordered Probit Models to Recover Moments of Continuous Test Score Distributions From Coarsened Data. *Journal of Educational and Behavioral Statistics* 42(1): 3–45. https://doi.org/10.3102/1076998616666279.

Reardon, S. F. 2011. The widening academic achievement gap between the rich and the poor: New evidence and possible explanations. *Whither opportunity* 91–116.

———. 2013. The widening income achievement gap. *Educational Leadership* 70(8): 10–16.

Williams, R. 2011. Fitting heterogeneous choice models with oglm. *The Stata Journal* 10(4): 540–567.

About the authors

Andres Yi Chang is Data Coordinator for the Service Delivery Indicator program in the Human Development Chief Economist Office at the World Bank (WB). Previously, he was Research Analyst at the Development Research Group also at the WB. Andres' research focuses on the access to and quality of schooling and health services in developing contexts. Currently, he splits his time between analytical research and providing technical support to WB operation teams on data collection efforts at the facility and household levels, particularly in low- and middle-income countries.