equate: An R Package for Observed-Score Linking and Equating

Anthony D. Albano
University of Nebraska-Lincoln

Abstract

The R package **equate** (Albano 2014) contains functions for observed-score linking and equating under single-group, equivalent-groups, and nonequivalent-groups test designs. This paper introduces these designs and provides an overview of observed-score equating with details about each of the supported methods. Examples demonstrate the basic functionality of the **equate** package.

Keywords: equating, linking, R.

1. Introduction

Equating is a statistical procedure commonly used in testing programs where administrations across more than one occasion and more than one examinee group can lead to overexposure of items, threatening the security of the test. In practice, item exposure can be limited by using alternate test forms; however, multiple forms lead to multiple score scales that measure the construct of interest at differing levels of difficulty. The goal of equating is to adjust for these differences in difficulty across alternate forms of a test, so as to produce comparable score scales.

Equating defines a functional statistical relationship between multiple test score distributions and thereby between multiple score scales. When the test forms have been created according to the same specifications and are similar in statistical characteristics, this functional relationship is referred to as an *equating function*, and it serves to translate scores from one scale directly to their equivalent values on another. The term *linking* refers to test forms which have not been created according to the same specifications, for example, forms which differ in length or content; in this case, the linked scales are considered similar but not interchangeable; they are related to one another via a *linking function* (for details, see Holland and Dorans 2006).

A handful of statistical packages are available for linking and equating test forms. Kolen and Brennan (2004) demonstrate a suite of free, standalone programs for observed-score and item response theory (IRT) linking and equating. Other packages, like **equate**, have been developed within the R environment (R Core Team 2013). For example, the R package **kequate** (Andersson, Bränberg, and Wiber 2013) includes observed-score methods, but within a kernel equating framework. The R package **plink** (Weeks 2010) implements IRT linking under a variety of dichotomous, polytomous, unidimensional, and multidimensional models.

The **equate** package is designed for observed-score linking and equating. It differs from other

packages primarily in its overall structure and usability, its plotting and bootstrapping capabilities, and its inclusion of more recently developed equating and linking types such as the general-linear, synthetic, and circle-arc functions, as demonstrated below. Linking and equating are performed using a simple interface, and plotting and summary methods are provided to facilitate the comparison of results and the examination of equating error. Sample data and detailed help files are also included. These features make the package useful in teaching, research, and operational testing contexts.

This paper presents some basic linking and equating concepts and procedures. Equating designs are first discussed in Section 2. In Section 3, linear and nonlinear observed-score linking and equating functions are reviewed. In Section 4, methods are presented for linking and equating when examinee groups are not equivalent. Finally, in Section 5, the **equate** package is introduced and its basic functionality is demonstrated using three data sets.

2. Equating designs

Observed-score linking and equating procedures require data from multiple test administrations. An *equating design* specifies whether or not the test forms and the individuals sampled to take them differ across administrations. For simplicity, in this paper and in the **equate** package, equating designs are categorized as either involving a *single group*, *equivalent groups*, or *nonequivalent groups* of examinees, and test forms are then constructed based on the type of group(s) sampled.

In the single-group design, one group, sampled from the target population T, takes two different test forms X and Y, optionally with counterbalancing. Any differences in the score distributions on X and Y are attributed entirely to the test forms themselves, as group ability is assumed to be constant; thus, if the distributions are not the same, it is because the test forms differ in difficulty. Related to the single-group design is the equivalent-groups design, where one random sample from T takes X and another takes Y. Because the samples are taken randomly, group ability is again assumed to be constant, and any differences in the score distributions are again identified as form difficulty differences.

Without equivalent examinee groups, two related problems arise: 1) the target population must be defined indirectly using samples from two different examinee populations, P and Q; and 2) the ability of these groups must then be accounted for, as ability differences will be a confounding factor in the estimation of form difficulty differences. In the nonequivalent-groups design these issues are both addressed through the use of what is referred to as an *anchor test*, V, a common measure of ability available for both groups. All non-equivalence in ability is assumed to be controlled or removed via this common measure.

Equating procedures were initially developed using the single-group and equivalent-groups designs. In this simpler context, the traditional equating types include mean, linear, and equipercentile equating; these and other equating types are reviewed in Section 3. More complex procedures have been developed for use with the nonequivalent-groups design; these equating methods are presented in Section 4.

3. Equating types

Equating procedures used with the single-group and equivalent-groups designs are referred to

here and in the **equate** package as equating *types*. The type of equating refers to the equation for a line that expresses scores on one scale, or axis, in terms of the other. The available types are categorized as straight-linear (i.e., linear), including identity, mean, and linear equating, and curvilinear (i.e., nonlinear), including equipercentile and circle-arc equating. The straight-line types differ from one another in intercept and slope, and the curvilinear lines differ in the number of coordinates on the line that are estimated, whether all of them or only one. Combinations of equating lines, referred to here as composite functions, are also discussed.

The goal of equating is to summarize the difficulty difference between X to Y. As shown below, each equating type makes certain assumptions regarding this difference and how it does or does not change across the X and Y score scales. These assumptions are always expressed in the form of a line within the coordinate system for the X and Y scales.

3.1. Identity functions

Linear functions are appropriate when test form difficulties change linearly across the score scale, by a constant b and rate of change a. Scores on X are related to Y as

$$y = ax + b. \tag{1}$$

In the simplest application of Equation (1), the scales of X and Y define the line. Coordinates for scores of x and y are found based on their relative positions within each scale:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}. \tag{2}$$

Here, (x_1, y_1) and (x_2, y_2) are coordinates for any two points on the line defined by the scales of X and Y, for example, the minimum and maximum possible scale values. Solving Equation (2) for y results in the identity linking function:

$$id_Y(x) = y = \frac{\Delta_Y}{\Delta_X}x + y_1 - \frac{\Delta_Y}{\Delta_X}x_1, \tag{3}$$

where $\Delta_Y = y_2 - y_1$ and $\Delta_X = x_2 - x_1$,

$$a = \frac{\Delta_Y}{\Delta_X}, \tag{4}$$

and

$$b = y_1 - \frac{\Delta_Y}{\Delta_X}x_1. \tag{5}$$

The intercept b can also be defined using the slope a and any pair of X and Y coordinates (x_j, y_k):

$$b = y_k - ax_j, \tag{6}$$

where $j = 1, 2, \ldots, J$ indexes the points on scale X and $k = 1, 2, \ldots, K$ indexes the points on scale Y. The identity linking function is then expressed as

$$id_Y(x) = \frac{\Delta_Y}{\Delta_X}x + y_k - \frac{\Delta_Y}{\Delta_X}x_j. \tag{7}$$

When the scales of X and Y are the same, $a = 1$ and $b = 0$, and Equation (7) reduces to the identity equating function:

$$ide_Y(x) = x. \tag{8}$$

3.2. Mean functions

In mean linking and equating, form difficulty differences are estimated by the mean difference $\mu_Y - \mu_X$. Equation (7) is used to define a line that passes through the means of X and Y, rather than the point (x_j, y_k). The intercept from Equation (6) is expressed as

$$b = \mu_Y - a\mu_X. \tag{9}$$

The mean linking function is then

$$mean_Y(x) = ax + \mu_Y - a\mu_X, \tag{10}$$

where a is found using Equation (4). When the scales of X and Y are the same, the slope a is 1, which leads to the mean equating function:

$$meane_Y(x) = x + \mu_Y - \mu_X. \tag{11}$$

In mean equating, coordinates for the line are based on deviation scores:

$$x - \mu_X = y - \mu_Y. \tag{12}$$

In mean linking, coordinates are based on deviation scores relative to the scales of X and Y:

$$\frac{x - \mu_X}{\Delta_X} = \frac{y - \mu_Y}{\Delta_Y}. \tag{13}$$

3.3. Linear functions

The linear linking and equating functions also assume that the difficulty difference between X and Y changes by a constant amount a across the score scale. However, in linear equating the slope is estimated using the standard deviations of X and Y as

$$a = \frac{\sigma_Y}{\sigma_X}. \tag{14}$$

The linear equating function is defined as

$$line_Y(x) = \frac{\sigma_X}{\sigma_Y}x + \mu_Y - \frac{\sigma_X}{\sigma_Y}\mu_X, \tag{15}$$

which is also the linear linking function $lin_Y(x)$. In both linear functions, coordinates for the line are based on standardized deviation scores:

$$\frac{x - \mu_X}{\sigma_X} = \frac{y - \mu_Y}{\sigma_Y}. \tag{16}$$

3.4. General linear functions

The identity, mean, and linear linking and equating functions presented above call all be obtained as variations of a general linear function $glin_Y(x)$. The general linear function is defined based on Equation 1 as

$$glin_Y(x) = \frac{\alpha_Y}{\alpha_X}x + \beta_Y - \frac{\alpha_Y}{\alpha_X}\beta_X, \tag{17}$$

where

$$a = \frac{\alpha_Y}{\alpha_X} \tag{18}$$

and

$$b = \beta_Y - \frac{\alpha_Y}{\alpha_X}\beta_X. \tag{19}$$

Here, α is a general scaling parameter that can be estimated using σ, Δ, another fixed value, or weighted combinations of these values. β is a general centrality parameter that can be estimated using μ, x_j or y_k, other values, or weighted combinations of these values. Applications of the general linear function are discussed below.

3.5. Equipercentile functions

Equipercentile linking and equating define a nonlinear relationship between score scales by setting equal the cumulative distribution functions for X and Y: $F(x) = G(y)$. Solving for y produces the equipercentile linking function:

$$equip_Y(x) = G^{-1}[F(x)], \tag{20}$$

which is also the equipercentile equating function $equipe_Y(x)$. When the score scales are discrete, which is often the case, the cumulative distribution function can be approximated using percentile ranks. This is a simple approach to *continuizing* the discrete score distributions (for details, see Kolen and Brennan 2004, ch. 2). Kernel equating, using Gaussian kernels, offers a more flexible approach to continuization (von Davier, Holland, and Thayer 2004; Andersson, Bränberg, and Wiber 2012), but is not currently supported in the **equate** package. The equipercentile equivalent of a form-X score on the Y scale is calculated by finding the percentile rank in X of particular score, and then finding the form-Y score associated with that form-Y percentile rank.

Equipercentile equating is appropriate when X and Y differ nonlinearly in difficulty, that is, when difficulty differences fluctuate across the score scale, potentially at each score point. Each coordinate on the equipercentile curve is estimated using information from the distributions of X and Y. Thus, compared to identity, mean, and linear equating, equipercentile equating is more susceptible to sampling error because it involves the estimation of as many parameters as there are unique score points on X.

Smoothing methods are typically used to reduce irregularities due to sampling error in either the score distributions or the equipercentile equating function itself. Two commonly used smoothing methods include polynomial loglinear presmoothing (Holland and Thayer 2000) and cubic-spline postsmoothing (Kolen 1984). The **equate** package currently supports loglinear presmoothing via the `glm` function. Details are provided below.

3.6. Circle-arc functions

Circle-arc linking and equating also define a nonlinear relationship between score scales; however, they utilize only three score points in X and Y to do so: the low and high points, as defined above for the identity function, and a midpoint (x_j, y_k). On their own, the low and high points define the identity linking function $id_Y(x)$, a straight line. When (x_j, y_k) does not fall on the identity linking line, it can be connected to (x_1, y_1) and (x_2, y_2) by the circumference of a circle with center (x_c, y_c) and radius r.

There are multiple ways of solving for (x_c, y_c) and r based on the three known points (x_1, y_1), (x_j, y_k), and (x_2, y_2). For example, the center coordinates can be found by solving the following system of equations:

$$(x_1 - x_c)^2 + (y_1 - y_c)^2 = r^2 \tag{21}$$
$$(x_j - x_c)^2 + (y_k - y_c)^2 = r^2 \tag{22}$$
$$(x_2 - x_c)^2 + (y_2 - y_c)^2 = r^2. \tag{23}$$

Subtracting Equation (23) from (21) and (22) and rearranging terms leads to the following linear system:

$$2(x_1 - x_2)x_c + 2(y_1 - y_2)y_c = x_1^2 - x_2^2 + y_1^2 - y_2^2 \tag{24}$$
$$2(x_j - x_2)x_c + 2(y_k - y_2)y_c = x_j^2 - x_2^2 + y_k^2 - y_2^2. \tag{25}$$

The center coordinates can then be obtained by plugging in the known values for (x_1, y_1), (x_j, y_k), and (x_2, y_2) and again combining equations. The center and any other coordinate pair, e.g., (x_1, y_1), are then used to find the radius:

$$r = \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2}. \tag{26}$$

Finally, solving Equation (26) for y results in the circle-arc linking function:

$$circ_Y(x) = y_c \pm \sqrt{r^2 - (x - x_c)^2}, \tag{27}$$

where the second quantity, under the square root, is added to y_c when $y_k > id_Y(x_j)$ and subtracted when $y_k < id_Y(x_j)$. The circle-arc equating function $circe_Y(x)$ is obtained by using $ide_Y(x_j)$ in place of $id_Y(x_j)$ above.

Livingston and Kim (2010) refer to the circle connecting (x_1, y_1), (x_j, y_k), and (x_2, y_2) as symmetric circle-arc equating. They also present a simplified approach, where the circle-arc function is decomposed into the linear component defined by (x_1, y_1) and (x_2, y_2), which is the identity function, and the circle defined by the points $(x_1, y_1 - id_Y(x_1))$, $(x_j, y_k - id_Y(x_j))$, and $(x_2, y_2 - id_Y(x_2))$. These low and high points reduce to $(x_1, 0)$ and $(x_2, 0)$, and the center coordinates can then be found as

$$x_c^* = \frac{(x_2^2 - x_1^2)}{2(x_2 - x_1)}, \tag{28}$$

and

$$y_c^* = \frac{(x_1^2)(x_2 - x_j) - (x_j^2 + y_k^{*2})(x_2 - x_1) + (x_2^2)(x_j - x_1)}{2[y_k^*(x_1 - x_2)]}, \tag{29}$$

where $y_k^* = y_k - id_Y(x_j)$. Equation (26) is used to find the radius. Then, the simplified circle-arc function is the combination of the resulting circle-arc $circ_Y^*(x)$ and the identity function:

$$scirc_Y(x) = circ_Y^*(x) + id_Y(x). \tag{30}$$

3.7. Composite functions

The circle-arc linking and equating functions involve a curvilinear combination of the identity and mean functions, where the circle-arc overlaps with the identity function at the low and

high points, and with the mean function at the midpoint (μ_X, μ_Y). A circle then defines the coordinates that connect these three points. This is a unique example of what is referred to here as a composite function.

The composite linking function is the weighted combination of any linear and/or nonlinear linking or equating functions:

$$comp_Y(x) = \sum_i w_i link_{iY}(x), \tag{31}$$

where w_i is a weight specifying the influence of function $link_{iY}(x)$ in determining the composite.

Equation 31 is referred to as a linking function, rather than an equating function, because it will typically not meet the symmetry requirement of equating. For symmetry to hold, the inverse of the function that links X to Y must be the same as the function that links Y to X, that is, $comp_Y^{-1}(x) = comp_X(y)$, which is generally not true when using Equation 31. Holland and Strawderman (2011) show how symmetry can be maintained for any combination of two or more linear functions. The weighting system must be adjusted by the slopes for the linear functions being combined, where the adjusted weight W_i is found as

$$W_i = \frac{w_i(1 + a_i^p)^{-1/p}}{\sum_i w_i(1 + a_i^p)^{-1/p}}. \tag{32}$$

Here, a_i is the slope for a given linear function $link_i$, and p specifies the type of L_p-circle with which symmetry is defined. For details, see Holland and Strawderman (2011).

4. Equating methods

The linking and equating functions presented above are defined in terms of a single target population, and they are assumed to generalize to this population. In the nonequivalent-groups design, scores come from two distinct populations, referred to here as populations P and Q. As a result, the linking and equating functions are redefined in terms of a weighted combination of P and Q, where $T = w_P P + w_Q Q$, and w_P and w_Q are proportions that sum to 1. This mixture of P and Q is referred to as the *synthetic population* (Braun and Holland 1982), S.

The linear function from Equation (15) is rewritten in terms of the synthetic population as follows:

$$line_{Y_S}(x) = \frac{\sigma_{Y_S}}{\sigma_{X_S}} x - \frac{\sigma_{Y_S}}{\sigma_{X_S}} \mu_{X_S} + \mu_{Y_S}. \tag{33}$$

Since population S did not take forms X or Y, all of the means and standard deviations in Equation (33) are estimated indirectly using: for the means,

$$\mu_{X_S} = \mu_{X_P} - w_Q \gamma_P (\mu_{V_P} - \mu_{V_Q}), \tag{34}$$

$$\mu_{Y_S} = \mu_{Y_Q} + w_P \gamma_Q (\mu_{V_P} - \mu_{V_Q}); \tag{35}$$

and for the variances,

$$\sigma^2_{X_S} = \sigma^2_{X_P} - w_Q \gamma^2_P [\sigma^2_{V_P} - \sigma^2_{V_Q}] + w_P w_Q \gamma^2_P [\mu_{V_P} - \mu_{V_Q}]^2, \tag{36}$$

$$\sigma^2_{Y_S} = \sigma^2_{Y_Q} + w_P \gamma^2_Q [\sigma^2_{V_P} - \sigma^2_{V_Q}] + w_P w_Q \gamma^2_Q [\mu_{V_P} - \mu_{V_Q}]^2. \tag{37}$$

	nominal	tucker	levine	braun	frequency	chained
mean	√	√	√	√		√
linear		√	√	√		√
general linear	√	√	√	√		
equipercentile					√	√
circle-arc	√	√	√	√		√

Table 1: Applicable equating types and methods.

In these equations the γ terms represent the relationship between total scores on X and Y and the respective anchor scores on V. γ_P and γ_Q are used along with the weights to adjust the observed μ and σ^2 for X and Y in order to obtain corresponding estimates for the synthetic population. For example, when $w_P = 0$ and $w_Q = 1$, $\mu_{Y_S} = \mu_{Y_Q}$, and conversely μ_{X_Q} will be adjusted the maximum amount when obtaining μ_{X_S}. The same would occur with the estimation of synthetic variances. Furthermore, the adjustments would be completely removed if populations P and Q did not differ in ability, where $\mu_{V_P} = \mu_{V_Q}$ and $\sigma^2_{V_P} = \sigma^2_{V_Q}$.

A variety of techniques have been developed for estimating the linear γ terms required by Equations (34) through (37), and the terms required for equipercentile equating, as described below. These techniques all make certain assumptions about the relationships between total scores and anchor scores for populations P and Q. The techniques are referred to here as equating *methods*. The **equate** package supports the Tucker, nominal weights, Levine observed-score, Levine true score, Braun/Holland, frequency estimation, and chained equating methods (although chained equating does not rely on γ, it does make assumptions about the relationship between total and anchor scores). Table 1 shows the supported methods that apply to each equating type.

4.1. Tucker

In Tucker equating the relationship between total and anchor test scores is defined in terms of regression slopes, where γ_P is the slope resulting from the regression of X on V for population P, and γ_Q the slope from a regression of Y on V for population Q:

$$\gamma_P = \frac{\sigma_{X_P,V_P}}{\sigma^2_{V_P}} \qquad \text{and} \qquad \gamma_Q = \frac{\sigma_{Y_Q,V_Q}}{\sigma^2_{V_Q}}. \tag{38}$$

The Tucker method assumes that across populations: 1) the coefficients resulting from a regression of X on V are the same, and 2) the conditional variance of X given V is the same. These assumptions apply to the regression of Y on V and the covariance of Y given V as well.

4.2. Nominal weights

Nominal weights equating is a simplified version of the Tucker method where the total and anchor tests are assumed to have similar statistical properties and to correlate perfectly within populations P and Q. In this case the γ terms can be approximated by the ratios

$$\gamma_P = \frac{N_X}{N_V} \qquad \text{and} \qquad \gamma_Q = \frac{N_Y}{N_V}, \tag{39}$$

where N is the number of items on the test. See Babcock, Albano, and Raymond (2012) for a description and examples.

4.3. Levine

Assumptions for the Levine observed-score method are stated in terms of true scores (though only observed scores are used), where, across both populations: 1) the correlation between true scores on X and V is 1, as is the correlation between true scores on Y and V; 2) the coefficients resulting from a linear regression of true scores for X on V are the same, as with true scores for Y on V; and 3) measurement error variance is the same (across populations) for X, Y, and V. These assumptions make possible the estimation of γ as

$$\gamma_P = \frac{\sigma^2_{X_P}}{\sigma_{X_P,V_P}} \qquad \text{and} \qquad \gamma_Q = \frac{\sigma^2_{Y_Q}}{\sigma_{Y_Q,V_Q}}, \tag{40}$$

which are the inverses of the respective regression slopes for V on X and V on Y. The Levine true-score method is based on the same assumptions as the observed-score method; however, it uses a slightly different linear equating function in place of Equation (33):

$$line_Y(x) = \frac{\gamma_Q}{\gamma_P} X(x - \mu_{X_P}) + \mu_{Y_Q} + \gamma_Q(\mu_{V_P} - \mu_{V_Q}). \tag{41}$$

Hanson (1991) and Kolen and Brennan (2004) provide justifications for using this approach.

4.4. Frequency estimation

The frequency estimation method is used in equipercentile equating under the nonequivalent-groups design. It is similar to the methods described above in that it involves a synthetic population. However, in this case full score distributions for the synthetic population taking forms X and Y are required:

$$equipe_{Y_S}(x) = G_S^{-1}[F_S(x)]. \tag{42}$$

When the assumptions are made that 1) the conditional distribution of total scores on X for a given score point in V is the same across populations, and 2) the conditional distribution of total scores on Y for a given score point in V is the same across populations, the synthetic distributions can be obtained:

$$f_S(x) = w_P f_P(x) + w_Q \sum f_P(x|v) h_Q(v), \tag{43}$$

$$g_S(y) = w_Q g_Q(y) + w_P \sum g_Q(y|v) h_P(v). \tag{44}$$

Here, f, g, and h denote the distribution functions for forms X, Y, and V respectively. Percentile ranks can be taken for the cumulative versions of f and g to obtain Equation (42). As before, w_P and w_Q specify the amount of adjustment to be made to each observed distribution in the estimation of the synthetic distribution.

4.5. Braun/Holland

As a kind of extension of the frequency estimation method, the Braun/Holland method defines a linear function relating X and Y that is based on the estimates μ_{X_S}, μ_{Y_S}, σ_{X_S}, and σ_{Y_S}

for the synthetic distributions $f_S(x)$ and $g_S(y)$ obtained via frequency estimation. Thus the full synthetic distributions are estimated, as with frequency estimation, but only in order to obtain their means and standard deviations.

4.6. Chained

Finally, chained equating (Livingston, Dorans, and Wright 1990) can be applied to both linear and equipercentile equating under the nonequivalent-groups with anchor test design. The chained method differs from all other methods discussed here in that it does not explicitly reference a synthetic population. Instead, it introduces an additional equating function in the process of estimating score equivalents (see Appendix 6 for details). For both linear and equipercentile equating the steps are as follows:

1. Define the function relating X to V for population P, $link_{V_P}(x)$,
2. Define the function relating V to Y for population Q, $link_{Y_Q}(v)$,
3. Equate X to the scale of Y using both functions, where

$$chain_Y(x) = link_{Y_Q}[link_{V_P}(x)].$$

Chained methods are based on the assumptions that 1) the equating of X to V is the same for P and Q, and 2) the equating of V to Y is the same for P and Q.

4.7. Methods for circle-arc

As discussed above, the circle-arc equating function combines a linear with a curvilinear component based on three points in the X and Y score distributions. The first and third of these points are determined by the score scale, whereas the midpoint must be estimated. Thus, equating methods used with circle-arc equating in the nonequivalent-groups design apply only to estimation of this midpoint. Livingston and Kim (2009) demonstrated chained linear equating of means, under a nonequivalent-groups design. The midpoint could also be estimated using other linear methods, such as Tucker or Levine.

Note that circle-arc equating is defined here as an equating *type*, and equating *methods* are used to estimate the midpoint. When groups are considered equivalent (i.e., an anchor test is not used) equating at the midpoint is simply mean equating, as mentioned above (replace x with μ_X in Equation (15) to see why this is the case). With scores on an anchor test, both Tucker and Levine equating at the midpoint also reduce to mean equating. However, chained linear equating at the midpoint differs from chained mean (see Appendix 6).

5. Using the equate package

5.1. Sample data

The **equate** package includes three sample data sets. The first, `ACTmath`, comes from two administrations of the ACT mathematics test, and is used throughout Kolen and Brennan (2004). The test scores are based on an equivalent-groups design and are contained in a

three-column data frame where column one is the 40-point score scale and columns two and three are the number of examinees for X and Y obtaining each score point.

The second data set, `KBneat`, is also used in Kolen and Brennan (2004). It contains scores for two forms of a 36-item test administered under a nonequivalent-groups design. A 12-item anchor test is internal to the total test, that is, anchor scores contribute to an examinee's total score. Thus, the number of non-anchor items, those unique to each form, is 24, and the highest possible score is 36. Unlike the first data set, `KBneat` contains a separate total and anchor score for each examinee. It is a list of length two where the list elements `x` and `y` each contain a two-column data frame of scores on the total test and scores on the anchor test `v`.

The third data set, `PISA`, contains scored cognitive item response data from the 2009 administration of the Programme for International Assessment (PISA). Four data frames are included in `PISA`: `PISA$students` contains scores on the cognitive assessment items in math, reading, and science for all 5233 students in the USA cohort; `PISA$booklets` contains information about the structure of the test design, where multiple item sets, or clusters, were administered across 13 test booklets; `PISA$items` contains the cluster, subject, maximum possible score, item format, and number of response options for each item; and `PISA$totals` contains a list of cluster total scores for each booklet, calculated using `PISA$students` and `PISA$booklets`. For additional details, see the `PISA` help file which includes references to technical documentation.

5.2. Preparing score distributions

The **equate** package utilizes score distributions primarily as frequency tables with class `"freqtab"`. For example, to equate the `ACTmath` forms, they must first be converted to frequency tables as follows.

```
R>library("equate")
R>act.x <- as.freqtab(cbind(ACTmath[, 1], ACTmath[, 2]))
R>act.y <- as.freqtab(cbind(ACTmath[, 1], ACTmath[, 3]))
R>act.x[1:4,]
```

```
  total observed
1     0        0
2     1        1
3     2        1
4     3        3
```

Here, the function `as.freqtab` is used because the vectors for the score scale and counts are already tabulated, thus they are simply combined and the class of each object is set. Frequency tables are summarized with the `summary` function.

```
R>rbind(x = summary(act.x), y = summary(act.y))
```

```
      mean       sd      skew     kurt min max    n
x 19.85239 8.212585 0.3752283 2.301911   1  40 4329
y 18.97977 8.940397 0.3526516 2.145847   1  40 4152
```

The function `freqtab` creates a frequency table from observed scores, using a vector of scores and the corresponding score scale. With an anchor test this becomes a bivariate frequency table, and the objects sent to `freqtab` are the vectors of total scores and anchor scores, and the total and anchor score scales.

```
R>neat.x <- freqtab(KBneat$x[, 1], KBneat$x[, 2],
+         xscale = 0:36, vscale = 0:12)
R>neat.y <- freqtab(KBneat$y[, 1], KBneat$y[, 2],
+         xscale = 0:36, vscale = 0:12)
R>neat.x[50:55, ]

   total anchor observed
50     3     10        0
51     3     11        0
52     3     12        0
53     4      0        0
54     4      1        4
55     4      2        3
```

These bivariate tables contain all possible score combinations in columns 1 and 2, along with the number of examinees obtaining each combination in column 3. For example, rows 50 through 55 are displayed for form X, where counts for 6 X and V score combinations are shown. Based on the scale lengths, tables for `neat.x` and `neat.y` contain $37 \times 13 = 481$ rows, many of which have counts of zero.

The `freqtab` function can also be used to tabulate scored item responses, where the arguments `xitems` and `vitems` contain the columns over which total scores will be calculated. For example, the following syntax creates a frequency table using four reading clusters from PISA booklet 6, with clusters R3 and R6 containing the unique items and clusters R5 and R7 containing the anchor items.

```
R>attach(PISA)
R>r3items <- paste(items$itemid[items$clusterid == "r3a"])
R>r6items <- paste(items$itemid[items$clusterid == "r6"])
R>r5items <- paste(items$itemid[items$clusterid == "r5"])
R>r7items <- paste(items$itemid[items$clusterid == "r7"])
R>pisa <- freqtab(students[students$book == 6, ],
+         xitems = c(r3items, r6items),
+         vitems = c(r5items, r7items),
+         xscale = 0:31, vscale = 0:29)
R>round(data.frame(summary(pisa),
+         row.names = c("r3r6", "r5r7")), 2)

      mean   sd  skew kurt min max   n
r3r6 17.45 7.20 -0.18 2.02   1  31 396
r5r7 18.19 6.05 -0.65 2.72   1  29 396
```

A basic plot method is provided for tables of class `"freqtab"`. Univariate frequencies are plotted as vertical lines for `x`, similar to a bar chart, and as superimposed curves for y. When

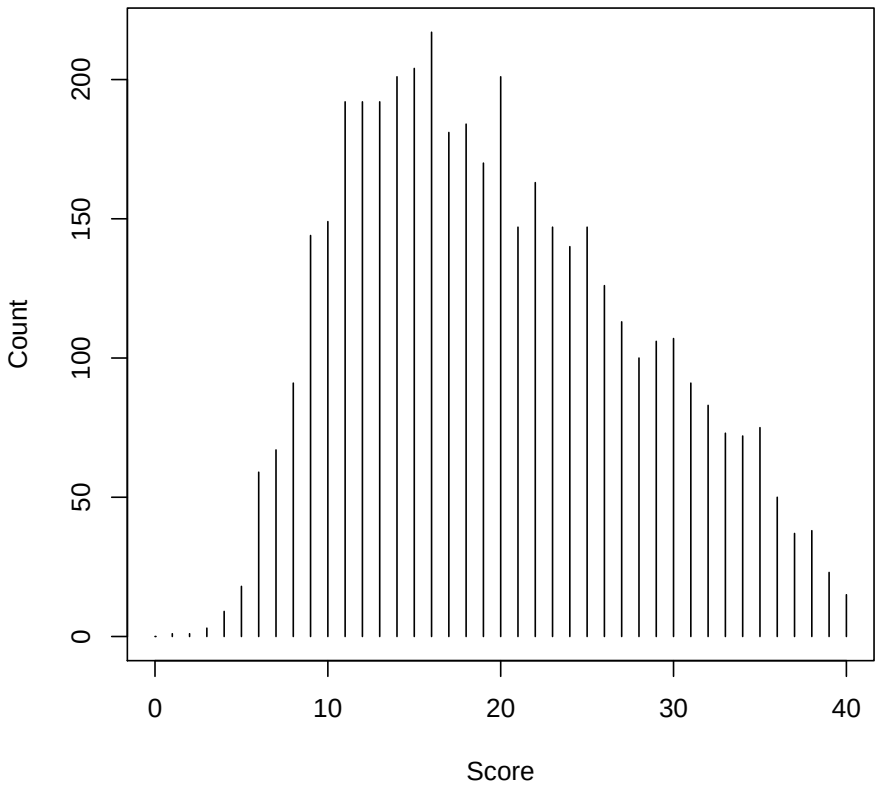

Figure 1: Univariate plot of `ACTmath` total scores for form X.

`y` is a matrix, each column of frequencies is added to the plot as a separate line. This feature is useful when examining smoothed frequencies, as discussed below. When `x` is a bivariate frequency table, a scatter plot with marginal frequency distributions is produced. See Figure 1 for an example of a univariate plot, and Figure 2 for an example of a bivariate plot.

```
R>plot(x = act.x, lwd = 2, xlab = "Score", ylab = "Count")
R>plot(neat.x)
```

Because they are based on samples, the distributions in Figures 1 and 2 are imperfect representations of the population distributions; irregularities in their shapes could merely result from sampling error. Three methods are available for presmoothing score distributions and reducing these irregularities. The first, frequency averaging (Moses and Holland 2008) replaces scores falling below `jmin` with averages based on adjacent scores. This is implemented with `smoothmethod = "average"` in the `presmoothing` function. The second, will add a small relative frequency (again, `jmin`) to each score point while adjusting the probabilities to sum to one (as described by Kolen and Brennan 2004, p. 48). This is implemented using `smoothmethod = "bump"` in the `presmoothing` function.

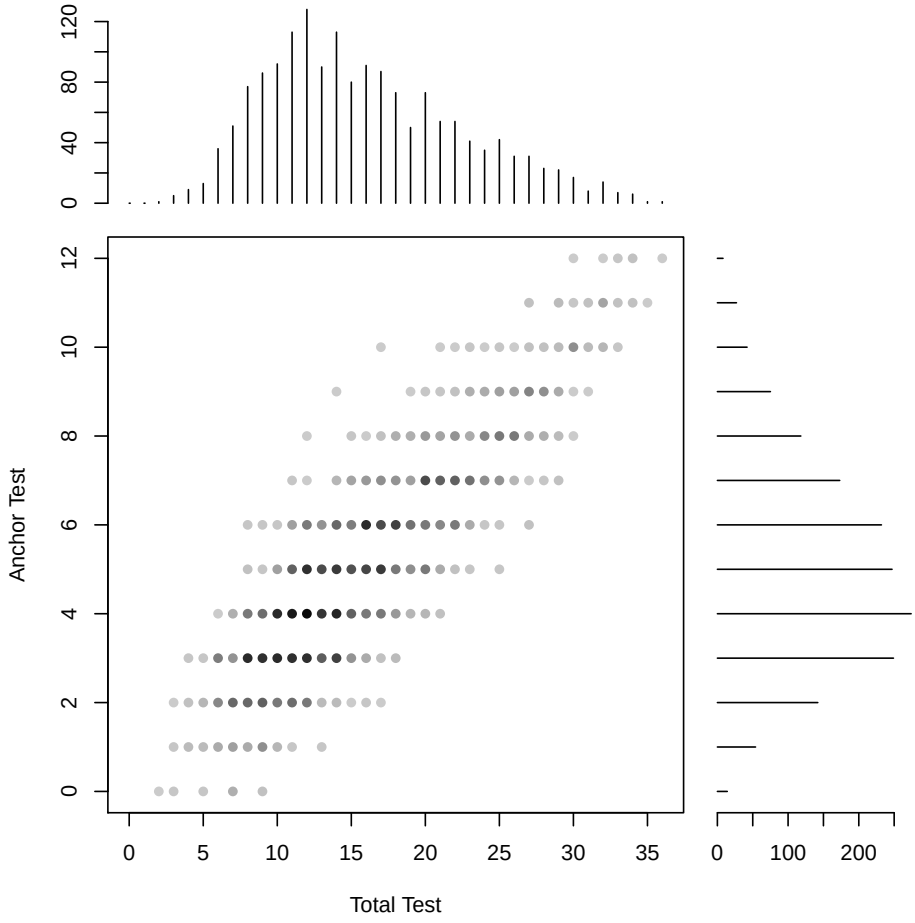

Figure 2: Bivariate plot of `KBneat` total and anchor distributions.

The third smoothing method, polynomial loglinear smoothing, described in Appendix 6, is a flexible procedure for reducing irregularities in a frequency distribution. In the **equate** package, loglinear models are fit using the `presmoothing` function with `smoothmethod = "loglinear"`, which calls on the `glm` function. Model terms are specified with either a matrix of score functions (`scorefun`) where each column is a predictor variable in the model, or simply by including the degrees of the highest desired polynomial terms (`degree` for univariate moments, `xdegree` for bivariate moments). In the example below, the bivariate distribution of X and V is smoothed with `degree = 3` and `xdegree = 1` . The smoothed distributions in Figure 3 can be compared to the unsmoothed ones in Figure 2. Figure 4 superimposes the smoothed frequencies on the unsmoothed marginal distributions for a more detailed comparison of the different smoothing models. Descriptive statistics show that the smoothed distributions match the unsmoothed in the first three moments.

```
R>neat.xs <- presmoothing(neat.x, smooth = "log", degree = 3,
+         xdegree = 1, asfreqtab = TRUE)
R>rbind(x = summary(neat.x), xs = summary(neat.xs))

               mean       sd      skew     kurt min max    n
x.total   15.820544 6.529799 0.5797331 2.720015   2  36 1655
x.anchor   5.106344 2.376742 0.4115535 2.766619   0  12 1655
xs.total  15.820544 6.529799 0.5797331 3.221789   0  36 1655
xs.anchor  5.106344 2.376742 0.4115535 2.968515   0  12 1655

R>neat.xsmat <- presmoothing(neat.x, "log",
+         degree = 3, xdegree = 1, stepup = TRUE)
R>plot(neat.xs)
R>plot(neat.x, neat.xsmat[, c(2:3, 5:7)], ycol = 1, ylty = 1:5)
```

The `loglinear` method can also be used to compare results from a sequence of nested models. The argument `stepup = TRUE` returns fitted frequencies for models based on subsets of columns in `scorefun`, starting with the first column alone, then adding the second, third, etc. When `scorefun` is omitted, it is created to have polynomial terms 1 through `degree` and `xdegree`; terms are added sequentially starting with the simplest model (`degree = 1`, `xdegree = 0`), progressing to the full univariate models (`degree = degree` for the univariate total first and then for the univariate anchor test), and finally including bivariate polynomials (from `xdegree = 1` to `xdegree = xdegree`). For example, the object `neat.xsmat` shown above is a matrix with seven columns, where each column contains the fitted frequencies for a nested model. The legend for Figure 4 shows that smoothed lines are plotted for the second and third moments of the total score scale ("x2" and "x3") and then the anchor scale ("v2" and "v3"), and finally for the first bivariate moment ("xv1"); the legend text corresponds to the new term that is included in the given model (models 1 and 4 have been omitted). Using the argument `compare = TRUE`, an ANOVA table of deviance statistics is returned for these nested models. Model fit is compare based on AIC, BIC, and likelihood ratio χ^2 tests. In the output below, AIC and BIC are smallest for the most complex model, labeled "Model 7", which also results in the largest decrease in deviance.

```
R>presmoothing(neat.x, "log", degree = 3,
+         xdegree = 1, compare = TRUE)
```

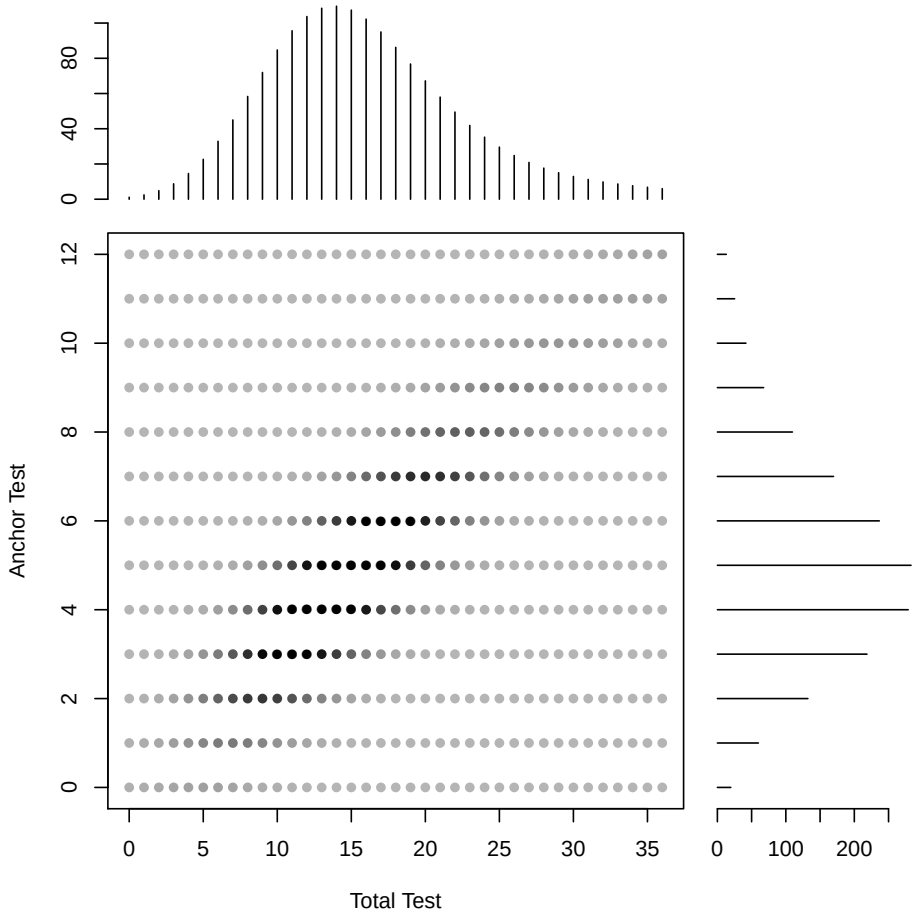

Figure 3: Bivariate plot of smoothed `KBneat` total and anchor distributions.

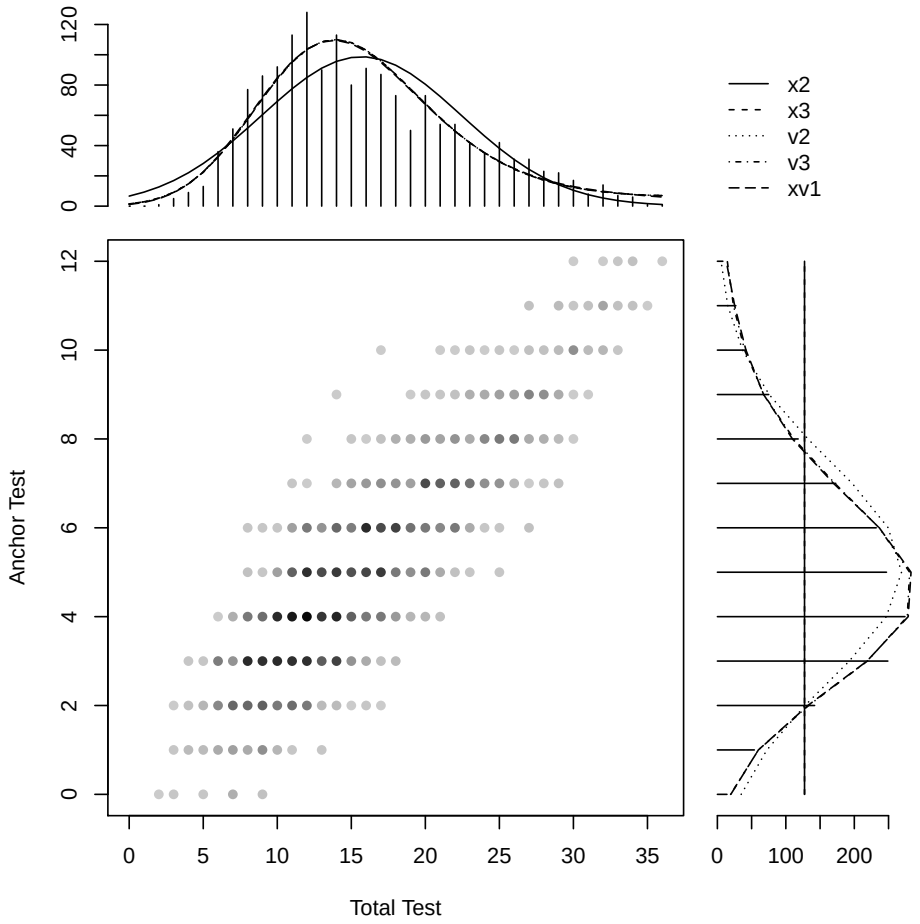

Figure 4: Bivariate plot of `KBneat` total and anchor distributions with smoothed frequencies superimposed.

```
Analysis of Deviance Table

Model 1: f ~ x1
Model 2: f ~ x1 + x2
Model 3: f ~ x1 + x2 + x3
Model 4: f ~ x1 + x2 + x3 + v1
Model 5: f ~ x1 + x2 + x3 + v1 + v2
Model 6: f ~ x1 + x2 + x3 + v1 + v2 + v3
Model 7: f ~ x1 + x2 + x3 + v1 + v2 + v3 + xv1
  Resid. Df Resid. Dev    AIC    BIC Df Deviance  Pr(>Chi)
1       479     4669.0 5301.2 5309.5
2       478     3666.3 4300.4 4313.0  1  1002.73 < 2.2e-16 ***
3       477     3559.9 4196.0 4212.7  1   106.41 < 2.2e-16 ***
4       476     3464.9 4103.1 4124.0  1    94.96 < 2.2e-16 ***
5       475     2593.3 3233.4 3258.5  1   871.64 < 2.2e-16 ***
6       474     2551.9 3194.1 3223.3  1    41.38 1.256e-10 ***
7       473      333.8  977.9 1011.4  1  2218.12 < 2.2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1
```

5.3. The `equate` function

Most of the functionality of the **equate** package can be accessed via the function `equate`, which integrates all of the equating types and methods described above. The equivalent-groups design provides a simple example: besides the X and Y frequency tables, only the equating `type` is required.

```
R>equate(act.x, act.y, type = "mean")

Mean Equating: act.x to act.y

Design: equivalent groups

Summary Statistics:
    mean   sd skew kurt  min   max    n
x  19.85 8.21 0.38 2.30 1.00 40.00 4329
y  18.98 8.94 0.35 2.15 1.00 40.00 4152
yx 18.98 8.21 0.38 2.30 0.13 39.13 4329

Coefficients:
intercept     slope        cx        cy        sx        sy
  -0.8726    1.0000   20.0000   20.0000   40.0000   40.0000
```

The nonequivalent-groups design is specified with an equating `method`, and smoothing with a `smoothmethod`.

```
R>neat.ef <- equate(neat.x, neat.y, type = "equip",
+         method = "frequency estimation", smoothmethod = "log")
```

Table 1 lists the equating methods that apply to each equating type in the nonequivalent-groups design. Levine true-score equating (`lts`) is performed by including the additional argument `lts = TRUE`.

An equating object such as `neat.ef` contains basic information about the type, method, design, smoothing, and synthetic population weighting for the equating, in addition to the frequency distributions given for `x` and `y`. The `summary` method creates separate tables for all of the frequency distributions utilized in the equating, and calculates descriptive statistics for each one.

```
R>summary(neat.ef)

Frequency Estimation Equipercentile Equating: neat.x to neat.y

Design: nonequivalent groups

Smoothing Method: loglinear presmoothing

Synthetic Weighting for x: 0.5025812

Summary Statistics:
            mean    sd  skew  kurt   min    max        n
x.obs     15.821 6.530 0.580 2.720 2.000 36.000 1655.000
x.smooth  15.821 6.530 0.580 2.720 0.000 36.000 1655.000
x.synth   16.734 6.753 0.435 2.460 0.000 36.000 1646.544
y.obs     18.673 6.881 0.205 2.301 3.000 36.000 1638.000
y.smooth  18.673 6.881 0.205 2.301 0.000 36.000 1638.000
y.synth   17.727 6.812 0.343 2.413 0.000 36.000 1646.544
yx.obs    16.808 6.605 0.484 2.624 2.142 36.267 1655.000
xv.obs     5.106 2.377 0.412 2.767 0.000 12.000 1655.000
xv.smooth  5.106 2.377 0.412 2.767 0.000 12.000 1655.000
xv.synth   5.481 2.444 0.259 2.566 0.000 12.000 1646.544
yv.obs     5.863 2.452 0.107 2.509 0.000 12.000 1638.000
yv.smooth  5.863 2.452 0.107 2.509 0.000 12.000 1638.000
yv.synth   5.481 2.444 0.259 2.566 0.000 12.000 1646.544
```

The `equate` function can also be used to convert scores from one scale to another based on the function defined in a previous equating. For example, scores on Y for a new sample of examinees taking `KBneat` form X could be obtained.

```
R>cbind(newx = c(3, 29, 8, 7, 13),
+         yx = equate(c(3, 29, 8, 7, 13), y = neat.ef))

     newx       yx
[1,]    3  3.242796
[2,]   29 29.844991
[3,]    8  8.685155
[4,]    7  7.605106
[5,]   13 14.078070
```

Here, the argument `y` passed to `equate` is the frequency estimation equipercentile equating object from above, which is an object of class `"equate"`. Since the equating function from `neat.ef` relates scores on X to the scale of Y, anchor test scores are not needed for the examinees in `newx`.

Finally, composite linkings are created using the `composite` function. For example, the identity and Tucker linear functions equating `neat.x` to `neat.y` could be combined as a weighted average function.

```
R>neat.i <- equate(neat.x, neat.y, type = "ident")
R>neat.lt <- equate(neat.x, neat.y, type = "linear",
+         method = "tucker")
R>neat.comp <- composite(list(neat.i, neat.lt), wc = .5,
+         symmetric = TRUE)
R>plot(neat.comp, addident = FALSE)
```

`neat.comp` represents what Kim, von Davier, and Haberman (2008) refer to as synthetic linear linking. The argument `symmetric = TRUE` is used to adjust the weighting system so that the resulting function is symmetric. Figure 5 shows the composite line in relation to the identity and linear components.

5.4. Linking with different scale lengths and item types

Procedures for linking scales of different lengths and item types are demonstrated here using `PISA` data. A frequency table containing four clusters, or item sets, from the PISA reading test was created above as `pisa`. This frequency table combines total scores on two item sets to create one form, R3R6, and total scores on two other item sets to create another form, R5R7. Because the same group of examinees took all of the item sets, the forms are contained within a single bivariate frequency table.

The two forms differ in length and item type. R3R6 contains 30 items, one of which has a maximum possible score of 2, and the remainder of which are scored dichotomously. This results in a score scale ranging from 0 to 31. However, 14 of the 30 items in R3R6 were multiple-choice (MC), mostly with four response options. The remaining items were either constructed-response or complex multiple-choice, where examinees were unlikely to guess the correct response. Thus, the lowest score expected by chance for R3R6 is $14/4 = 3.5$. R5R7 contains 29 items, all of which are scored dichotomously. Eight of these items are MC with four response options and the remainder are CR or complex MC, resulting in a lowest expected chance score of $8/4 = 2$. The summary statistics above show that, despite having a slightly smaller score scale, the mean for R5R7 is slightly higher than for R3R5.

Results for linking R3R6 to R5R7 are compared here for five linking types: identity, mean, linear, circle-arc, and equipercentile with loglinear presmoothing (using the default parameters). By default, the identity linking component of each linear function is based on the minimum and maximum possible points for each scale, that is, $(0, 0)$ and $(31, 29)$. The low points were modified to be $(3.5, 2)$ to reflect the lowest scores expected by chance.

```
R>pisa.i <- equate(pisa, type = "ident", lowp = c(3.5, 2))
R>pisa.m <- equate(pisa, type = "mean", lowp = c(3.5, 2))
R>pisa.l <- equate(pisa, type = "linear", lowp = c(3.5, 2))
```

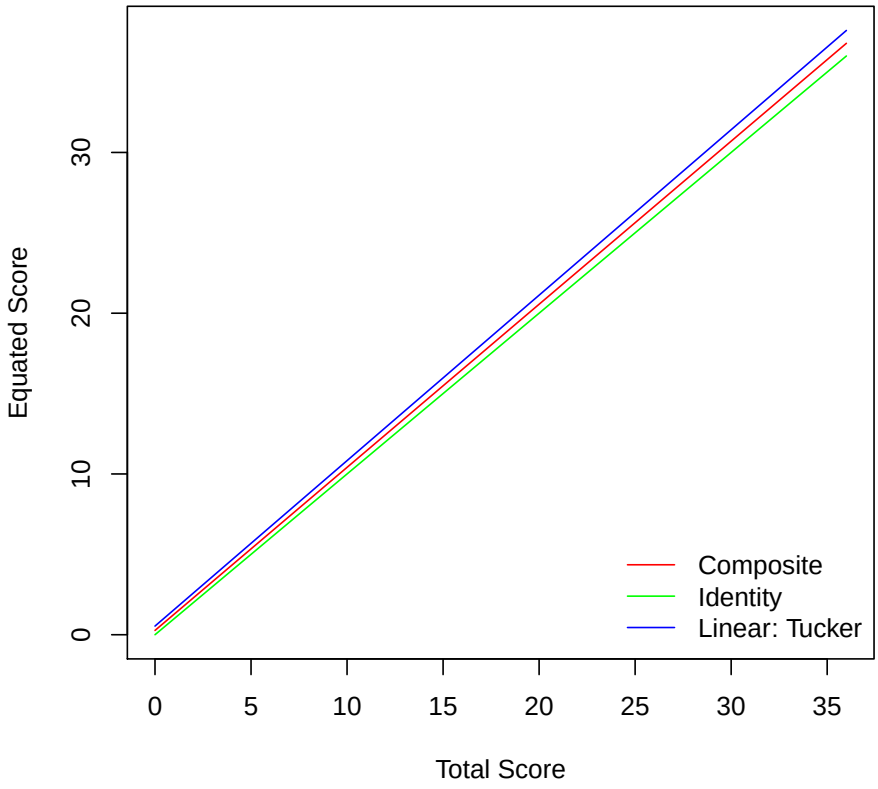

Figure 5: Identity, Tucker linear, and a composite of the two functions for equating `KBneat`.

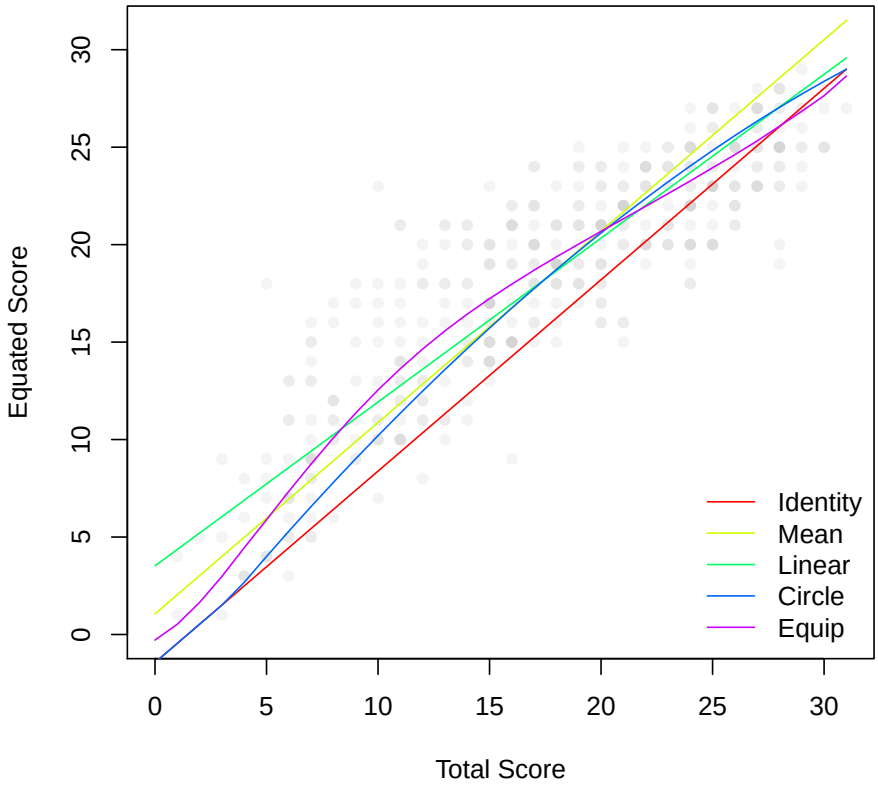

Figure 6: Five functions linking R3R6 to R5R7.

```
R>pisa.c <- equate(pisa, type = "circ", lowp = c(3.5, 2))
R>pisa.e <- equate(pisa, type = "equip", smooth = "log",
+          lowp = c(3.5, 2))
R>plot(pisa.i, pisa.m, pisa.l, pisa.c, pisa.e, addident = F,
+          xpoints = pisa, morepars = list(ylim = c(0, 31)))
```

The identity, mean, linear, circle-arc, and equipercentile linking functions are plotted in Figure 6. With a single-group design the linking lines can be plotted over the observed total scores for each form. In this way, the results can be compared in terms of how well each linking captures the difficulty difference from R3R6 to R5R7. Based on the scatterplot in Figure 6, scores on R5R7 tend to be higher, but this difference is not linear across the score scale. Instead, the difficulty difference appears curvilinear. Circle-arc linking appears to underestimate this nonlinearity, whereas equipercentile linking appears to estimate it well.

5.5. Parametric bootstrapping

All but the identity linking and equating functions estimate a statistical relationship between

score scales. Like any statistical estimate, equated scores are susceptible to bias and random sampling error, for example, as defined in Appendix 6. Standard error (SE), *bias*, and root mean square error ($RMSE$) can be estimated in the **equate** package using empirical and parametric bootstrapping.

With the argument `boot = TRUE`, the `equate` function will return bootstrap standard errors based on sample sizes of `xn` and `yn` taken across `reps = 100` replications from `x` and `y`. Individuals are sampled with replacement, and the default sample sizes `xn` and `yn` will match those observed in `x` and `y`. Equating is performed at each replication, and the estimated equating functions are saved. *Bias* and $RMSE$ can be obtained by including a vector of criterion equating scores via `crit`. Finally, the matrix of estimated equatings at each replication can be obtained with `eqs = TRUE`.

Parametric bootstrapping is performed within the `equate` function by providing the optional frequency distributions `xp` and `yp`. These simply replace the sample distributions `x` and `y` when the bootstrap resampling is performed. Additionally, the `bootstrap` function can be used directly to perform multiple equatings at each bootstrap replication. SE, *bias*, and $RMSE$ can then be obtained for each equating function using the same bootstrap data.

Parametric bootstrapping using the `bootstrap` function is demonstrated here for eight equatings of form X to Y in `KBneat`: Tucker and chained mean, Tucker and chained linear, frequency estimation and chained equipercentile, and Tucker and chained-linear circle-arc. Identity equating is also included. Smoothed population distributions are first created. Based on model fit comparisons, loglinear models were chosen to preserve 4 univariate and 2 bivariate moments in the smoothed distributions of X and Y. Plots are shown in Figures 7 and 8.

```
R>neat.xp <- presmoothing(neat.x, "log", xdegree = 2,
+         asfreqtab = TRUE)
R>neat.xpmat <- presmoothing(neat.x, "log", xdegree = 2,
+         stepup = TRUE)
R>neat.yp <- presmoothing(neat.y, "log", xdegree = 2,
+         asfreqtab = TRUE)
R>neat.ypmat <- presmoothing(neat.y, "log", xdegree = 2,
+         stepup = TRUE)
R>plot(neat.x, neat.xpmat[, c(3, 4, 7:10)])
R>plot(neat.y, neat.ypmat[, c(3, 4, 7:10)])
```

Next, the number of replications is set to 100, bootstrap sample sizes are set to 100 for X and Y, and a criterion equating function is defined as the chained equipercentile equating in the population.

```
R>set.seed(131031)
R>reps <- 100
R>xn <- 100
R>yn <- 100
R>crit <- equate(neat.xp, neat.yp, "e", "c")$conc$yx
```

Finally, to run multiple equatings in a single bootstrapping study, the arguments for each equating must be combined into a single object. Here, each element in `neat.args` is a named list of arguments for each equating. This object is then used in the `bootstrap` function, which carries out the bootstrapping.

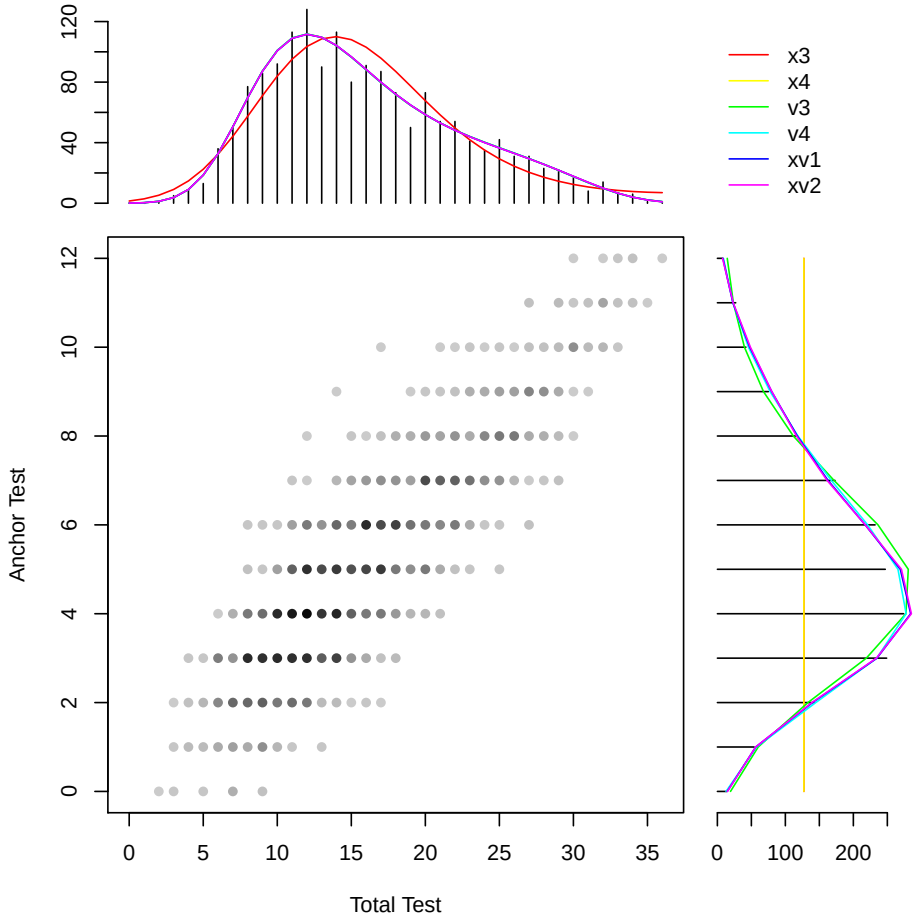

Figure 7: Smoothed population distributions for X used in parametric bootstrapping.

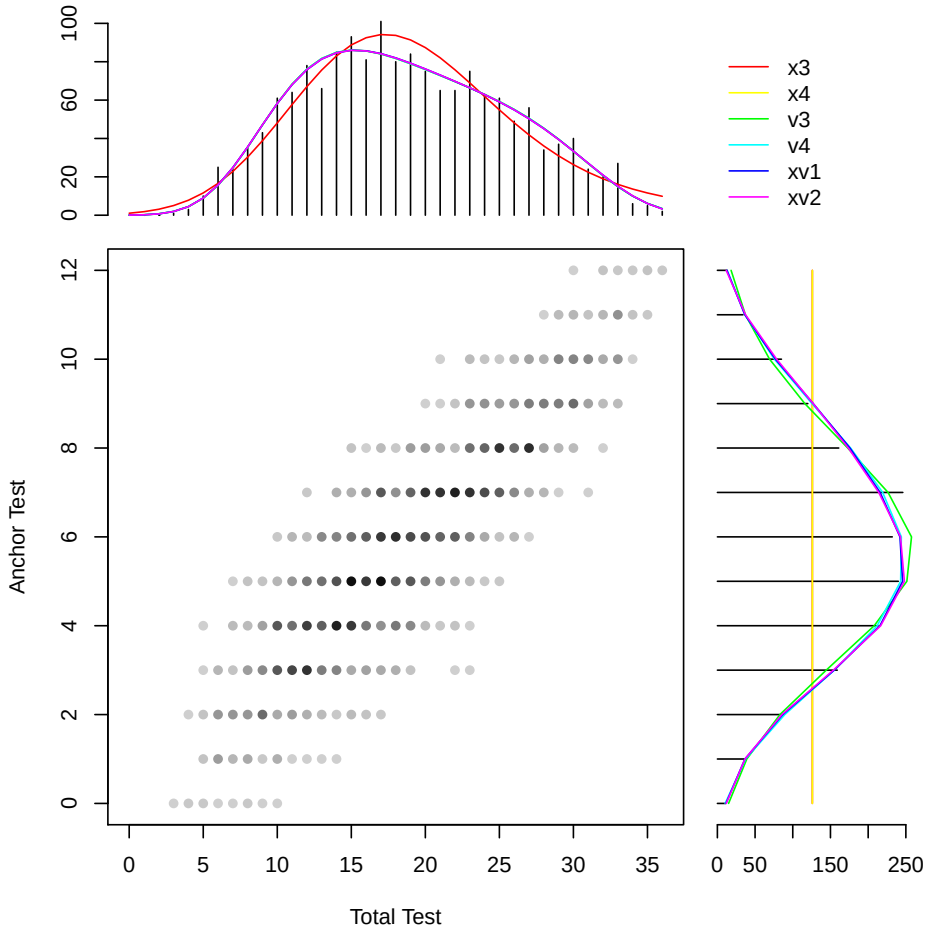

Figure 8: Smoothed population distributions for Y used in parametric bootstrapping.

```
R>neat.args <- list(i = list(type = "i"),
+         mt = list(type = "mean", method = "t"),
+         mc = list(type = "mean", method = "c"),
+         lt = list(type = "lin", method = "t"),
+         lc = list(type = "lin", method = "c"),
+         ef = list(type = "equip", method = "f", smooth = "log"),
+         ec = list(type = "equip", method = "c", smooth = "log"),
+         ct = list(type = "circ", method = "t"),
+         cc = list(type = "circ", method = "c", chainmidp = "lin"))
R>bootout <- bootstrap(x = neat.xp, y = neat.yp, xn = xn, yn = yn,
+         reps = reps, crit = crit, args = neat.args)
```

A plot method is available for visualizing output from the `bootstrap` function, as demonstrated below. Figures 9 through 12 contain the mean equated scores across replications for each method, the SE, *bias*, and $RMSE$. In Figure 9, the mean equated scores appear to be similar across much of the scale. Chained mean equating (the light orange line) consistently produces the highest mean equated scores. Mean equated scores for the remaining methods fall below those of chained mean and above those of identity equating (the black line). In Figure 10, standard errors tend to be highest for the equipercentile methods, especially chained equipercentile (the dark blue line), followed by the linear methods (green lines). SE are lowest for the circle-arc methods (purple and pink), especially in the tails of the score scale where the identity function has more of an influence. In Figure 11, *bias* is highest for chained mean equating, and is negative for the identity function; otherwise, *bias* for the remaining methods falls roughly between -0.5 and 0.5. Finally, in Figure 12, $RMSE$ tends to be highest for chained mean and the linear and equipercentile methods. $RMSE$ for Tucker mean and the circle-arc methods tended to fall at or below 0.5.

```
R>plot(bootout, addident = F, col = c(1, rainbow(8)))
R>plot(bootout, out = "se", addident = F,
+         col = c(1, rainbow(8)), legendplace = "top")
R>plot(bootout, out = "bias", addident = F,
+         col = c(1, rainbow(8)), legendplace = "top",
+         morepars = list(ylim = c(-.9, 3)))
R>plot(bootout, out = "rmse", addident = F,
+         col = c(1, rainbow(8)), legendplace = "top",
+         morepars = list(ylim = c(0, 3)))
```

A summary method is also available for output from the `bootstrap` function. Mean SE, *bias*, and $RMSE$, and weighted and absolute means, when applicable, are returned for each equating. Weighted means are calculated by multiplying the error estimate at each score point with the corresponding relative frequency in X, and absolute means are based on absolute error values. The output below summarizes what is shown in Figures 9 through 12: mean SE is lowest for identity and the circle-arc methods; mean *bias* is low for a few methods but, in terms of absolute bias, is lowest for chained equipercentile; and mean $RMSE$ is lowest on average for Tucker circle-arc. Overall, Tucker circle-arc outperforms the other methods in terms of error reduction, with mean $RMSE$ of 0.41. Mean $RMSE$ for the remaining methods are between 0.46 (chained circle-arc) and 1.51 (chained mean).

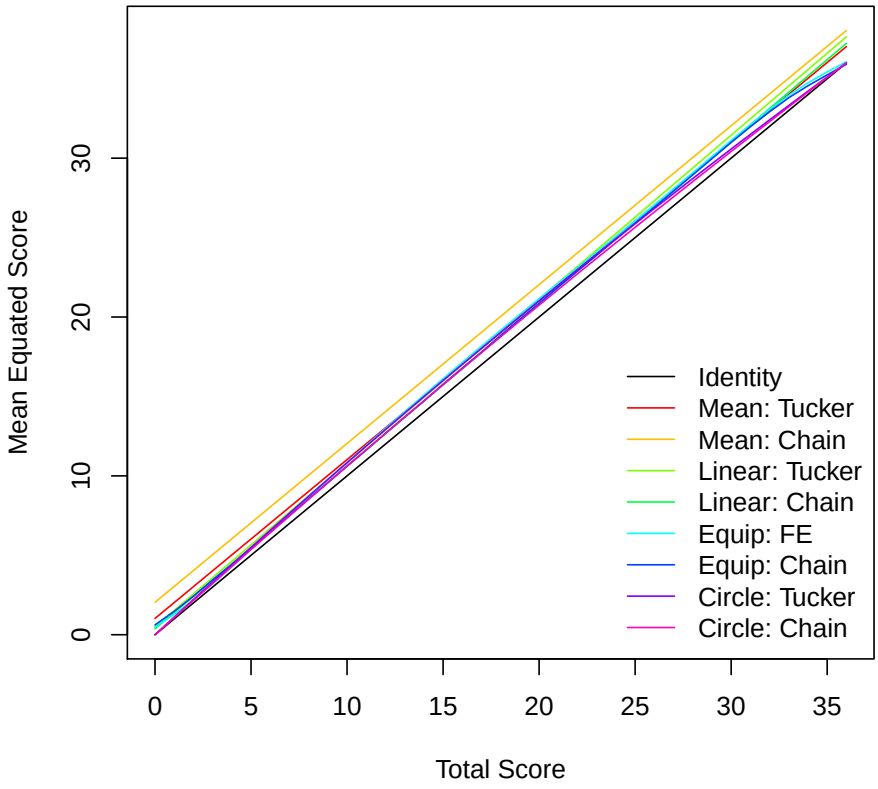

Figure 9: Parametric bootstrapped mean equated scores for eight methods.

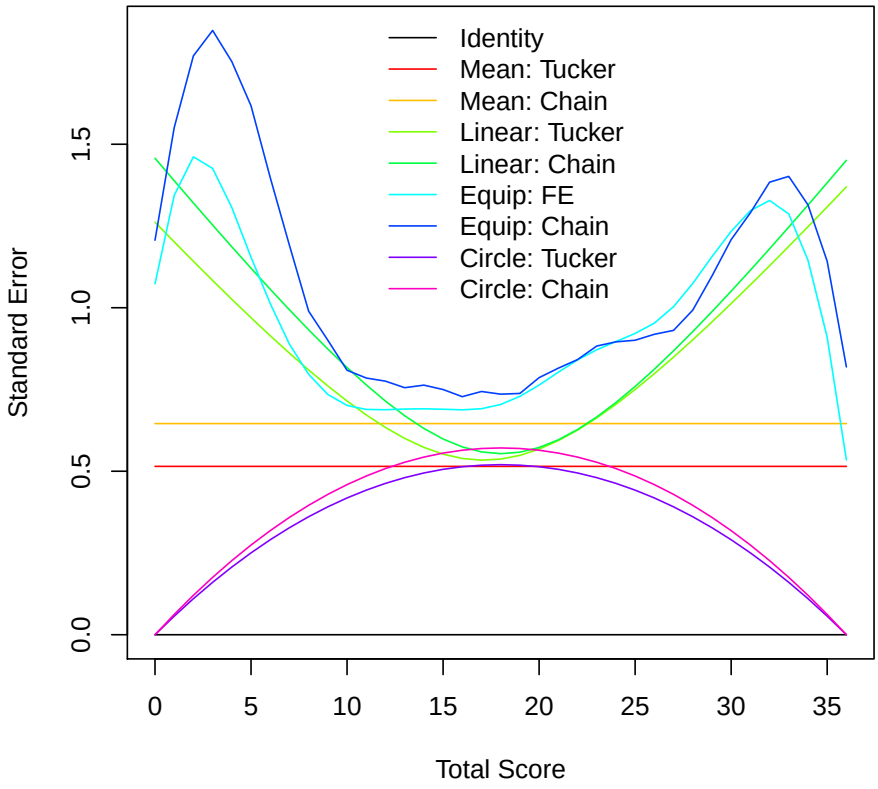

Figure 10: Parametric bootstrapped SE for eight methods.

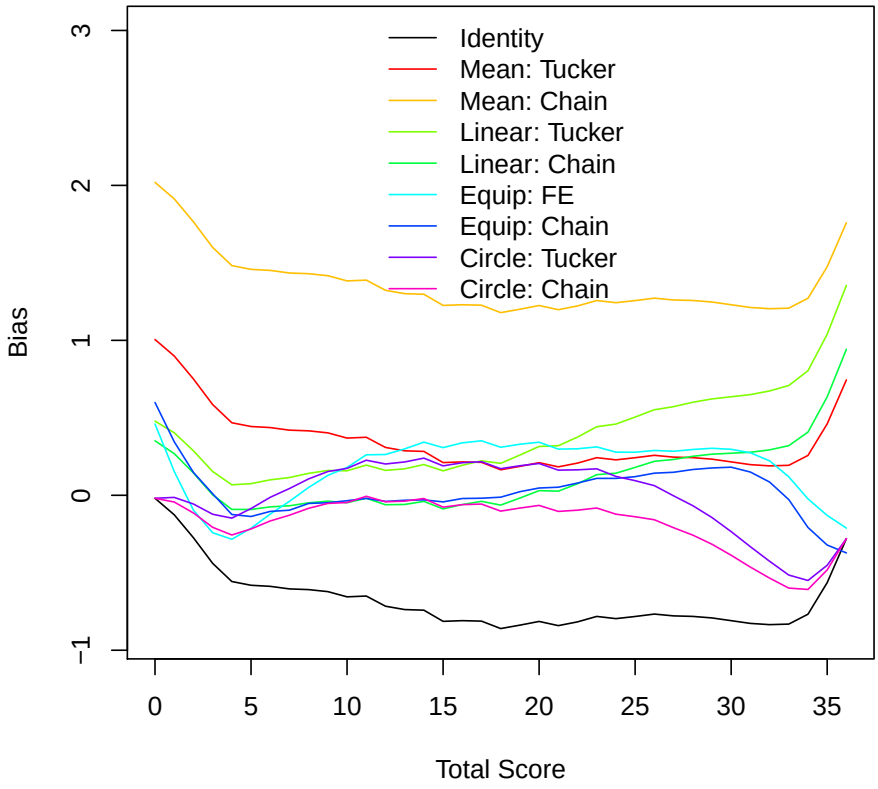

Figure 11: Parametric bootstrapped *bias* for eight methods.

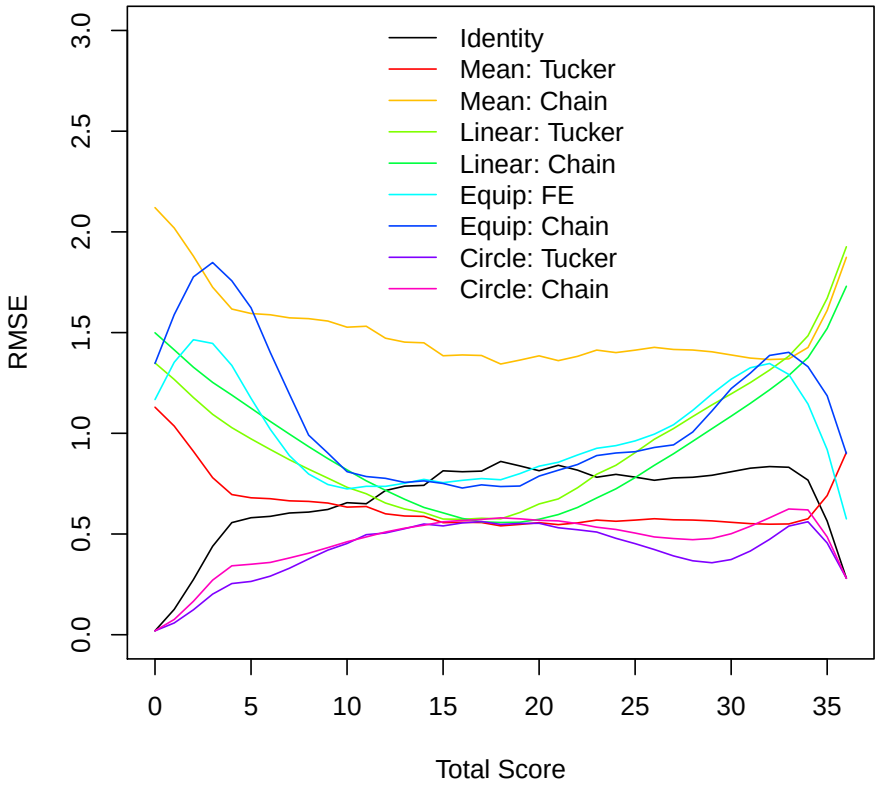

Figure 12: Parametric bootstrapped $RMSE$ for eight methods.

```
R>round(summary(bootout), 2)

     se w.se  bias a.bias w.bias wa.bias rmse
i  0.00 0.00 -0.67   0.67  -0.02    0.02 0.67
mt 0.51 0.01  0.35   0.35   0.01    0.01 0.64
mc 0.65 0.02  1.37   1.37   0.04    0.04 1.51
lt 0.86 0.02  0.39   0.39   0.01    0.01 0.96
lc 0.93 0.02  0.12   0.17   0.00    0.00 0.95
ef 0.95 0.02  0.17   0.24   0.01    0.01 0.99
ec 1.07 0.02  0.03   0.12   0.00    0.00 1.08
ct 0.34 0.01  0.00   0.18   0.00    0.00 0.41
cc 0.37 0.01 -0.18   0.18   0.00    0.00 0.46
```

6. Summary

This paper presents some basic concepts and procedures for observed-score linking and equating of measurement scales. Linear and nonlinear functions are discussed, and various methods for applying them to nonequivalent groups are reviewed. Finally, the **equate** package is introduced, and its basic functionality is demonstrated using three data sets.

The **equate** package is designed to be a resource for teaching, learning, and applying observed-score linking and equating procedures. A simple interface, via the `equate` function, can be used to control most of the necessary functionality, including data preparation, presmoothing, linking and equating, and managing output. Summary and plot methods facilitate the comparison of results. Additional features, not presented in this paper, are also available; details can be found by consulting the help files for the package. Finally, future versions of the **equate** package will be extended to support additional procedures, for example, postsmoothing, nonlinear continuization, and new composite linking functions.

Additional formulas

Chained linear equating

Chained linear equating involves two separate linear functions. In the equations below the anchor test V is distinguished by population (P taking form X and Q taking form Y), though the items on V do not change. The first linear function in slope-intercept form converts X to the scale of V_P:

$$l_{V_P}(x) = \frac{\sigma_{V_P}}{\sigma_X}x - \frac{\sigma_{V_P}}{\sigma_X}\mu_X + \mu_{V_P}. \tag{45}$$

The second function converts V_Q to the scale of Y:

$$l_Y(v_Q) = \frac{\sigma_Y}{\sigma_{V_Q}}v_Q - \frac{\sigma_Y}{\sigma_{V_Q}}\mu_{V_Q} + \mu_Y. \tag{46}$$

These functions are combined, where the first, $l_{V_P}(x)$, takes the place of v_Q in the second to obtain:

$$lchain_Y(x) = \frac{\sigma_Y}{\sigma_{V_Q}}\left[\frac{\sigma_{V_P}}{\sigma_X}x - \frac{\sigma_{V_P}}{\sigma_X}\mu_X + \mu_{V_P}\right] - \frac{\sigma_Y}{\sigma_{V_Q}}\mu_{V_Q} + \mu_Y, \tag{47}$$

or, in slope-intercept form, after some rearranging:

$$lchain_Y(x) = \frac{\sigma_Y}{\sigma_{V_Q}}\frac{\sigma_{V_P}}{\sigma_X}x + \frac{\sigma_Y}{\sigma_{V_Q}}\left[\mu_{V_P} - \frac{\sigma_{V_P}}{\sigma_X}\mu_X - \mu_{V_Q}\right] + \mu_Y. \tag{48}$$

Finally, for chained mean equating this reduces to:

$$mchain_Y(x) = x + \mu_{V_P} - \mu_X - \mu_{V_Q} + \mu_Y. \tag{49}$$

When used to obtain the midpoint coordinates in circle-arc equating, the chained method reduces even further, since x is μ_X. Here, the linear and mean functions simplify to

$$lchain_Y(\mu_X) = \frac{\sigma_Y}{\sigma_{V_Q}}\mu_{V_P} - \frac{\sigma_Y}{\sigma_{V_Q}}\mu_{V_Q} + \mu_Y, \tag{50}$$

and

$$mchain_Y(\mu_X) = \mu_{V_P} - \mu_{V_Q} + \mu_Y. \tag{51}$$

Loglinear presmoothing

Polynomial loglinear modeling is a flexible procedure for smoothing distributions of various shapes to varying degrees. The structure of a distribution can either be maintained or ignored depending on the complexity of the model, where the degree of the polynomial term included determines the moment of the raw score distribution to be preserved. For example, a model with terms to the first, second, and third powers would create a smoothed distribution which matches the raw in mean, variance, and skewness. As shown below, the log of the expected relative frequency p for score point x is modeled as a function of a normalizing constant (the intercept β_0) and the observed-score value to the first, second, and third powers:

$$log(p) = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3. \tag{52}$$

Indicator variables may also be included to preserve specific moments for subsets of score points. In the next model, the mean and variance of a sub-distribution are preserved, in addition to the first three moments of the full distribution. When $S = 1$, score point x is included in this sub-distribution, and when $S = 0$, it is ignored:

$$log(p) = \beta_0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3 + \beta_{0S} S + \beta_{1S} x^1 S + \beta_{2S} x^2 S. \tag{53}$$

An acceptable degree of smoothing is typically achieved by comparing multiple models with different numbers of polynomial terms based on their fit to the data (Kolen and Brennan 2004). The `loglinear` function in **equate** is a wrapper for the `glm` function in the **stats** package. It can be used to fit and compare nested models up to specified maximum polynomial terms. For additional details, see the `presmoothing` help file.

Error in equating

In simulation and resampling studies, equatings are typically compared based on both random and systematic error (or differences), where the first is estimated by the standard error of equating SE and the second by the *bias*. Error is defined in terms of the population equating

function $e_Y(x)$ and estimate $\hat{e}_{Yr}(x)$ for samples $r = 1, 2, \ldots, R$. Systematic error is estimated as

$$bias = \hat{\bar{e}}_Y(x) - e_Y(x), \tag{54}$$

where

$$\hat{\bar{e}}_Y(x) = \frac{1}{R}\sum_{r=1}^{R}\hat{e}_{Yr}(x) \tag{55}$$

is the average estimated equated score over R samples. The random error is estimated as

$$SE = \sqrt{\frac{1}{R}\sum_{r=1}^{R}[\hat{e}_{Yr}(x) - \hat{\bar{e}}_Y(x)]^2}. \tag{56}$$

Combining both systematic error and random error, the root mean squared error is estimated as

$$RMSE = \sqrt{bias^2 + SE^2}. \tag{57}$$

References

Albano AD (2014). ***equate**: Observed-Score Linking and Equating*. R package version 2.0-2, URL http://CRAN.R-project.org/package=equate.

Andersson B, Bränberg K, Wiber M (2012). ***kequate**: The Kernel Method of Test Equating*. R package version 0.5.1, URL http://CRAN.R-project.org/package=kequate.

Andersson B, Bränberg K, Wiber M (2013). "Performing the Kernel Method of Test Equating with the Package **kequate**." *Journal of Statistical Software*, **55**(6), 1–15. URL http://www.jstatsoft.org/v55/i06.

Babcock B, Albano AD, Raymond M (2012). "Nominal Weights Mean Equating: A Method for Very Small Samples." *Educational and Psychological Measurement*, **72**, 608–628.

Braun HI, Holland PW (1982). "Observed-Score Test Equating: A Mathematical Analysis of Some ETS Equating Procedures." In PW Holland, DB Rubin (eds.), *Test Equating*, pp. 9–49. New York, NY: Academic.

Hanson BA (1991). "A Note on Levine's Formula for Equating Unequally Reliable Tests Using Data From the Common Item Nonequivalent Groups Design." *Journal of Educational and Behavioral Statistics*, **16**, 93.

Holland PW, Dorans NJ (2006). "Linking and Equating." In RL Brennan (ed.), *Educational Measurement*, 4 edition, pp. 187–220. Westport, CT: Greenwood.

Holland PW, Strawderman WE (2011). "How to Average Equating Functions, If You Must." In AA von Davier (ed.), *Statistical Models for Test Equating, Scaling, and Linking*, pp. 89–107. New York, NY: Springer-Verlag.

Holland PW, Thayer DT (2000). "Univariate and Bivariate Loglinear Models for Discrete Test Score Distributions." *Journal of Educational and Behavioral Statistics*, **25**(2), 133–183.

Kim S, von Davier AA, Haberman S (2008). "Small-Sample Equating Using a Synthetic Linking Function." *Journal of Educational Measurement*, **45**, 325–342.

Kolen MJ (1984). "Effectiveness of Analytic Smoothing in Equipercentile Equating." *Journal of Educational and Behavioral Statistics*, **9**, 25–44.

Kolen MJ, Brennan RL (2004). *Test Equating, Scaling, and Linking*. New York, NY: Springer-Verlag.

Livingston SA, Dorans NJ, Wright NK (1990). "What Combination of Sampling and Equating Methods Works Best?" *Applied Measurement in Education*, **3**, 73–95.

Livingston SA, Kim S (2009). "The Circle-Arc Method for Equating in Small Samples." *Journal of Educational Measurement*, **46**, 330–343.

Livingston SA, Kim S (2010). "Random-groups Equating With Samples of 50 to 400 Test Takers." *Journal of Educational Measurement*, **47**, 175–185.

Moses TP, Holland PW (2008). "Notes on a General Framework for Observed Score Equating." *Ets research rep. no. rr-08-59*, Princeton, NJ: ETS.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

von Davier AA, Holland PW, Thayer DT (2004). *The Kernel Method of Test Equating*. New York, NY: Springer-Verlag.

Weeks JP (2010). "**plink**: An R Package for Linking Mixed-Format Tests Using IRT-Based Methods." *Journal of Statistical Software*, **35**(12), 1–33. URL http://www.jstatsoft.org/v35/i12/.

Affiliation:

Anthony D. Albano
Department of Educational Psychology
College of Education and Human Sciences
University of Nebraska-Lincoln
Lincoln, NE 68508, USA
E-mail: albano@unl.edu