Journal of Statistical Software
August 2014, Volume 59, Issue 7. http://www.jstatsoft.org/

SNSequate: Standard and Nonstandard Statistical Models and Methods for Test Equating

Jorge González
Pontificia Universidad Católica de Chile

Abstract

Equating is a family of statistical models and methods that are used to adjust scores on two or more versions of a test, so that the scores from different tests may be used interchangeably. In this paper we present the R package **SNSequate** which implements both standard and nonstandard statistical models and methods for test equating. The package construction was motivated by the need of having a modular, simple, yet comprehensive, and general software that carries out traditional and new equating methods. **SNSequate** currently implements the traditional mean, linear and equipercentile equating methods, as well as the mean-mean, mean-sigma, Haebara and Stocking-Lord item response theory linking methods. It also supports the newest methods such as local equating, kernel equating, and item response theory parameter linking methods based on asymmetric item characteristic functions. Practical examples are given to illustrate the capabilities of the software. A list of other programs for equating is presented, highlighting the main differences between them. Future directions for the package are also discussed.

Keywords: observed score equating (OSE), item response theory (IRT), equating and parameter linking, nonstandard equating methods, R.

1. Introduction

Many of the decisions made by administrative or policy makers in an educational system are based on examinees' scores. In making decisions, common practices are the comparison of scores in multiple forms of the same assessment. Equating is a family of statistical models and methods that are used to adjust scores on two or more versions of a test, so that scores on these versions may be used interchangeably (see, e.g., Holland and Rubin 1982; Kolen and Brennan 2004; von Davier, Holland, and Thayer 2004; Dorans, Pommerich, and Holland 2007; von Davier 2011b). The goal in equating is to obtain an appropriate transformation that maps the scores of one test form into the scale of the other. Certain requirements concerning the measured construct, the reliability of test forms, the symmetry of the transformation, and the equity and population invariance principles, are needed for this mapping to be validly called an equating (for details on these requirements, see Kolen and Brennan 2004, Section 1.3; Dorans and Holland 2000).

Methods for test equating can be classified in two main classes: observed-score equating (OSE) and item response theory (IRT) equating. Examples of OSE methods are the mean, linear, and equipercentile equating; the Tucker, Levine observed-score, and Levine true-score methods; the (Gaussian) kernel method of equating, among others. IRT methods include true score and observed score IRT equating, and the class of IRT parameter linking methods such as mean-mean, mean-sigma, Haebara and Stocking-Lord methods. A good summary of the above mentioned techniques can be found in Kolen and Brennan (2004), and von Davier *et al.* (2004). We refer to these two groups of traditional equating methods as *standard*.

The development of new theoretical and sophisticated equating methods is nowadays common in equating research (see von Davier 2011a,b,c). Some are completely novel while others are extensions of the standard methods. For example, von Davier (2011b) contains methods such that topics that are typically found in statistics books, (e.g., exponential families, Bayesian nonparametric models, time-series analysis, etc.) are explicitly put into an equating framework (González 2013). Also, hybrid methods such as local equating (van der Linden 2011), and the Levine nonlinear method (von Davier, Fournier-Zajac, and Holland 2007) have emerged as new possibilities in equating. We refer to the group of new and more theoretical and sophisticated equating methods as *nonstandard*.

While nonstandard equating methods accommodate issues that standard methods do not handle well (von Davier 2011b), they have not been widely adopted by practitioners. One reason for this is the lack of software that implements new equating methods. The aim of this paper is to introduce the R (R Core Team 2014) package **SNSequate** (González 2014) which intends to fill this gap. The package supports both standard and nonstandard statistical models and methods for test equating. Currently, **SNSequate** implements the traditional mean, linear and equipercentile equating methods; the mean-mean, mean-sigma, Haebara, and Stocking-Lord IRT parameter linking methods; and the (Gaussian) kernel method of equating (KE). Nonstandard methods such as local equating, IRT parameter linking based on asymmetric item characteristic functions, and the implementation of the logistic and uniform kernels in the KE framework are also available. Additionally, many other methods will be implemented in future versions of the package (see Section 5). Key distinguishing issues that make **SNSequate** different from current software for equating are: (i) it is the only software that currently implements local equating, (ii) it is also the only software that implements IRT parameter linking based on asymmetric ICCs, (iii) it includes many data sets that have appeared and have been well studied in the equating literature, which helps in understanding better the implemented methods.

The rest of the paper is organized as follows: Section 2 gives an introduction to equating and briefly describes methods under both the OS and IRT approaches. In Section 3 the functions that are included in the R package **SNSequate** are described. All the capabilities of **SNSequate** are illustrated by several examples in Section 4. We conclude the paper with final comments and ideas for future research.

2. The statistical inference problem in equating

Let X and Y be the random variables denoting the scores on tests $\mathcal{X}$ and $\mathcal{Y}$ which are to be equated. In what follows we assume that scores on $\mathcal{X}$ are equated to the scale of scores on $\mathcal{Y}$, but arguments and formulas for the reverse equating are analogue. Let $F(x)$ and $G(y)$ be the associated cumulative distributions functions (CDF) of the scores. We are interested in a transformation $y = \varphi(x)$ which equates the scores of $\mathcal{X}$ into $\mathcal{Y}$.

All the transformations in the equating literature are based on the so-called equipercentile function, defined as

$$\varphi(x) = G^{-1}(F(x)). \tag{1}$$

Sum scores (i.e., total number of correct responses) are commonly used test scores in measurement programs. Because the possible values that sum scores can take are consecutive integers, an evident problem with (1) is the discreteness of the score distributions rendering their inverse functions unavailable. The common solution to this problem is to continuize the discrete distributions F and G. Different continuization methods can be used each producing parametric, nonparametric, or semi-parametric statistical inference about φ (González and von Davier 2013). Regardless of the statistical approach adopted, the parameters (which are either finite or infinite-dimensional or a mixture of both) are estimated using empirical scores producing sampling variability associated with the estimated equating functions, which is quantified by the standard error of equating (SEE).

An important aspect in the estimation of the equating transformation concerns the way in which data should be collected so that differences between the tests, $\mathcal{X}$ and $\mathcal{Y}$, do not confound the assessment of the examinee differences in ability (von Davier 2013). Different approaches consider data collection designs that use either common examinees or common items. Among those that assume common examinees, are the single group design (SG), in which the same group of examinees are administrated two different forms; the equivalent groups design (EG), in which students are randomly assigned the form to be administrated; and the counterbalanced design (CB) in which two random samples of examinees from a common population take both tests in different order. There also exist designs resorting in common items such as in the nonequivalent groups with anchor tests design (NEAT), where the two groups of students each taking one of the test forms, are sampled from different populations (for details see, e.g., von Davier *et al.* 2004, Chapter 2; Kolen and Brennan 2004, Section 1.4).

2.1. Observed-score equating methods

In what follows, we first briefly describe observed-score equating methods, showing in each case the form of the equating transformation $\varphi(\cdot)$ to be estimated. Detailed descriptions of these methods can be found in Kolen and Brennan (2004), von Davier *et al.* (2004), and von Davier (2011b).

Mean and linear equating

In mean equating, the score distributions F and G only differ in their means. The scores from two tests that are the same distance from their respective means are set equal: $x - \mu_x = y - \mu_y$. It follows that

$$y = \varphi(x; \mu_x, \mu_y) = x - \mu_x + \mu_y. \tag{2}$$

Linear equating further assumes that the score distributions differ in their means and standard deviations. Scores from two tests are assumed to be an equal distance from their means in standard deviation units are set equal: $\frac{x-\mu_x}{\sigma_x} = \frac{y-\mu_y}{\sigma_y}$. It follows that

$$y = \varphi(x; \mu_x, \mu_y, \sigma_x, \sigma_y) = \frac{\sigma_x}{\sigma_y}(x - \mu_x) + \mu_y. \tag{3}$$

Theorem 1.1 in von Davier *et al.* (2004) summarizes the connection between (3) and (1). Note that in practice, the transformation of scores is made using the estimated equating function $\hat{\varphi} = \varphi(x; \hat{\boldsymbol{\pi}})$, where $\hat{\boldsymbol{\pi}} = (\hat{\mu}_x, \hat{\mu}_y, \hat{\sigma}_x, \hat{\sigma}_y)$ are directly estimated from the data. Moreover, because the equating transformation depends on parameters, both mean and linear equating methods are parametric.

Equipercentile equating

The equipercentile method generalizes mean and linear equating by not only considering differences between the first two moments, but differences between the entire score distributions. A function $e_Y(x)$ is called an equipercentile transformation if the distribution of scores which results from the conversion of $\mathcal{X}$ scores to the $\mathcal{Y}$ scale is equal to G (the distribution of scores on $\mathcal{Y}$ in the population). It follows that

$$\varphi(x) = e_Y(x) = G^{-1}(F(x)). \tag{4}$$

To employ (4) for equating, both F and G are typically continuized by linear interpolation. Because φ is built from (distribution) functions, the equipercentile equating method is non-parametric.

The kernel method of equating (KE)

In KE, the originally discrete score distributions F and G are continuized by using kernel techniques (Silverman 1986). Let h_X be a parameter controlling the degree of smoothness for the continuization. von Davier *et al.* (2004) showed that if $V \sim N(0,1)$ (a standard normal distribution) and the continuized variable is defined as $X(h_X) = a_X(X + h_X V) + (1 - a_X)\mu_X$, where $a_X^2 = \frac{\sigma_X^2}{\sigma_X^2 + \sigma_V^2 h_X^2}$, and X is the originally discrete score random variable, then the Gaussian kernel smoothing of $F(x)$, defined as

$$F_{h_X}(x) = \sum_j r_j \Phi\left(\frac{x - a_X x_j - (1 - a_x)\mu_X}{a_X h_X}\right), \tag{5}$$

is exactly the CDF of $X(h_X)$. It should be mentioned that continuization with alternative kernels other than the Gaussian is also possible (Lee and von Davier 2011). The conversion of scores is finally based on the estimated equation function

$$\hat{\varphi}(x; \boldsymbol{r}, \boldsymbol{s}) = G_{h_Y}^{-1}(F_{h_X}(x; \hat{\boldsymbol{r}}); \hat{\boldsymbol{s}}) = \hat{G}_{h_Y}^{-1}(\hat{F}_{h_X}(x)), \tag{6}$$

where $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{s}}$ are vectors of estimated score probabilities defined as $r_j = \mathsf{P}(X = x_j)$ and $s_k = \mathsf{P}(Y = y_k)$ respectively, with x_j and y_k taking values between 0 and the total number of items in $\mathcal{X}$, and $\mathcal{Y}$, respectively. Both, $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{s}}$ are obtained using the so-called design functions (DF), which take into account the chosen data collection design in the estimation. This process is typically made after smoothing the (discrete) observed score frequency distributions (univariate and/or bivariate) by using log-linear models.

In order to statistically assess the effectiveness of $\hat{\varphi}$, von Davier *et al.* (2004) give a diagnostic measure called the percent relative error (PRE) which compares the moments of the distributions of the equated scores to the moments of the original discrete reference distribution. Also, the accuracy of the estimated $\hat{\varphi}(x)$ is measured by SEE.

The main stages in the previous description of the KE method have been summarized in five steps (see, e.g., von Davier *et al.* 2004): (i) pre-smoothing, (ii) estimation of scores probabilities, (iii) continuization, (iv) computing and diagnosing the equating function, and (v) computation of accuracy measures.

Note that the estimation of φ involves both, vector ($\boldsymbol{r}$ and $\boldsymbol{s}$) and function parameters (F and G) so that the KE method is semiparametric.

Local equating

Instead of using the "marginal" distributions of scores, as is the case in the equipercentile method, local equating methods (van der Linden 2011, 2013) utilize the "conditional" (on abilities) distributions of scores to obtain the transformation φ, leading to a family of ability-dependent equating transformations of the form

$$\varphi(x;\theta) = G^{-1}_{Y|\theta}(F_{X|\theta}(x)), \quad \theta \in \mathcal{R}. \tag{7}$$

To estimate θ, maximum likelihood (ML), maximum a posteriori (MAP) or expected a posteriori (EAP) estimates are typically computed and are obtained using the response patterns of individuals and assuming that an IRT model holds. The conditional score distributions are typically obtained by using an algorithm described by Lord and Wingersky (1984). Linear interpolation is used to continuize the resulting discrete conditional distributions. Because both vector and function parameters are involved, LE is also a semiparametric method.

2.2. IRT parameter linking and equating methods

IRT models (van der Linden and Hambleton 1997; De Boeck and Wilson 2004) are widely used nowadays for analyzing and scoring tests. As many testing programs use IRT to assemble tests, the use of IRT equating is a natural option (Skaggs and Lissitz 1986; Cook and Eignor 1991; Lord 1980, Chapter 13). Using IRT in the equating process requires a previous step, referred to here as IRT item-parameter linking. Because the parameters from different test forms need to be on the same IRT scale, linking is conducted to place the IRT parameter estimates, from separate calibrations of two test forms, on a common scale[1].

Let Y_{ij} be the random variable denoting the answer of individual i to item j. IRT models specify the respondent's probability of a correct answer to a test item, based on both a person's parameter θ_i and a vector of item characteristics (e.g., difficulty, discriminant power, etc.), ω_j. When it is assumed that $\theta_i \sim N(0, \sigma^2_\theta)$ the model is specified as

$$(Y_{ij} \mid \theta_i, \omega_j) \sim \text{Bernoulli}(p(\theta_i, \omega_j)), \tag{8}$$

where p is known as the item characteristic curve (ICC). In particular, the three parameter

[1]This is needed particularly when conducting equating under the NEAT design.

logistic model (3PL, Birnbaum 1968) employs $p(\theta_i, w_j) = c_j + (1 - c_j)\Psi(Da_j(\theta_i - \beta_j))$, where $\Psi(x) = \exp(x)/1 + \exp(x)$ is the standard logistic function. Under this specification we have

$$p_{ij} = \mathsf{P}(Y_{ij} = 1 \mid \theta_i, \omega_j) = c_j + (1 - c_j)\frac{\exp[Da_j(\theta_i - \beta_j)]}{1 + \exp[Da_j(\theta_i - \beta_j)]}, \tag{9}$$

where $\omega_j = (a_j, \beta_j, c_j)$ and D is a scaling constant.

Other IRT models are special cases of the 3PL. For instance, the two parameter logistic model (2PL) is obtained by setting $c_j = 0$ for all j, whereas the 1PL model additionally sets all a_j to be equal to 1. For details on IRT parameter estimation methods and software the reader is referred to Fischer and Molenaar (1995); Baker and Kim (2004); Tuerlinckx, Rijmen, Molenberghs, Verbeke, Briggs, den Noortgate, Meulders, and De Boeck (2004).

Parameter linking

When an IRT model is used to fit two different test forms, a linear equation can be used to convert both sets of IRT parameter estimates to the same scale (see Kolen and Brennan 2004, Section 6.2). The corresponding linear relations are

$$\theta_{\mathcal{Y}i} = A\theta_{\mathcal{X}i} + B \tag{10}$$
$$a_{\mathcal{Y}j} = a_{\mathcal{X}j}/A \tag{11}$$
$$\beta_{\mathcal{Y}j} = A\beta_{\mathcal{X}j} + B \tag{12}$$
$$c_{\mathcal{Y}j} = c_{\mathcal{X}j}, \tag{13}$$

where A and B are constants to be estimated, and the indices $\mathcal{X}$ and $\mathcal{Y}$ are used to differentiate between the scales. A detailed account of methods to calculate A and B can be found in Kolen and Brennan (2004, Chapter 6). A brief description of them is given next.

Mean-mean and mean-sigma methods

These methods use the means and standard deviations of the common item parameter estimates to obtain the constants A and B in the following way

- Mean-mean: $A = \frac{\mu(a_{\mathcal{X}})}{\mu(a_{\mathcal{Y}})}$, and $B = \mu(\beta_{\mathcal{Y}}) - A\mu(\beta_{\mathcal{X}})$.
- Mean-sigma: $A = \frac{\sigma(a_{\mathcal{X}})}{\sigma(a_{\mathcal{Y}})}$, and $B = \mu(\beta_{\mathcal{Y}}) - A\mu(\beta_{\mathcal{X}})$.

In both cases, means and standard deviations are taken only on the set of common items between tests forms $\mathcal{X}$ and $\mathcal{Y}$.

Characteristic curves methods

These methods resort on the ICCs and iteratively search for optimal A and B. The functions to be minimized for the Haebara (Hcrit) and Stocking-Lord (SLcrit) criteria are

$$\text{Hcrit} = \sum_i \sum_{j:V} \left[p_{ij}\left(\theta_{\mathcal{Y}i}, \hat{a}_{\mathcal{Y}j}, \hat{\beta}_{\mathcal{Y}j}, \hat{c}_{\mathcal{Y}j}\right) - p_{ij}\left(\theta_{\mathcal{Y}i}, \frac{\hat{a}_{\mathcal{X}j}}{A}, A\hat{\beta}_{\mathcal{X}j} + B, \hat{c}_{\mathcal{X}j}\right)\right]^2, \tag{14}$$

and,

$$\text{SLcrit} = \sum_i \left[\sum_{j:V} p_{ij} \left(\theta_{\mathcal{Y}i}, \hat{a}_{\mathcal{Y}j}, \hat{\beta}_{\mathcal{Y}j}, \hat{c}_{\mathcal{Y}j} \right) - \sum_{j:V} p_{ij} \left(\theta_{\mathcal{Y}i}, \frac{\hat{a}_{\mathcal{X}j}}{A}, A\hat{\beta}_{\mathcal{X}j} + B, \hat{c}_{\mathcal{X}j} \right) \right]^2, \tag{15}$$

respectively. In both cases, V denotes the set of common items between tests forms $\mathcal{X}$ and $\mathcal{Y}$.

IRT true-score and observed-score equating

After item parameters are estimated and placed on the same scale, IRT equating is used to relate scores from two tests forms in the following ways:

- *IRT true-score equating.* A true score ξ from form $\mathcal{X}$ associated with a given θ_i, is considered to be equivalent to the true score η from $\mathcal{Y}$ that is associated with that same θ. The relation between ability and true scores is determined using the so-called test characteristics functions, $T(\theta_i)$, defined as follows,

$$\xi = \sum_{j:\mathcal{X}} p_{ij}(\theta_i, \omega_j) = T^{\xi}(\theta_i) \tag{16}$$

$$\eta = \sum_{j:\mathcal{Y}} p_{ij}(\theta_i, \omega_j) = T^{\eta}(\theta_i), \tag{17}$$

  where the sum is over the items of the corresponding form. The resulting transformation to find an equivalent true score η of ξ is given by

$$\varphi(\xi, \theta_i) = T^{\eta}(T^{\xi^{-1}}(\xi)) \tag{18}$$

  In practice, estimates of the item parameters are used to produce an *estimated true score relationship* for each test form by means of the tests characteristics functions. Also, because true scores are not observable, the estimated true score conversion is actually applied to the observed sum scores.

- *IRT observed score equating.* This method uses both, conditional (on ability) observed-score distributions and the ability distribution. The product of these two distributions is integrated (or summed) across all ability levels to produce a marginal observed score distribution. Once this process has been completed for both $\mathcal{X}$ and $\mathcal{Y}$, equipercentile equating is applied to relate scores between the two forms. Hence, if $S^{\mathcal{X}}$, $S^{\mathcal{Y}}$, and $G(\theta)$ represents the scores and ability distributions, respectively, the resulting transformation is

$$\varphi(x) = S^{\mathcal{Y}^{-1}}(S^{\mathcal{X}}(x)), \tag{19}$$

  where $S^{\bullet}(t) = \int S(t \mid \theta) G(\theta) d\theta$.

3. The SNSequate R package

SNSequate contains several functions written in R that carry out test equating for a variety of approaches. **SNSequate** is freely available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=SNSequate. In this section we briefly describe the functions currently available in **SNSequate**.

- The functions `mea.eq()`, `lin.eq()`, and `eqp.eq()` perform mean, linear and equipercentile equating, respectively. Their main arguments are the observed scores from the two test forms that are to be equated and the values in the scales that are to be equated.

- The function `le.eq()` implements local equating. To obtain the equating transformation, this function first estimates the conditional score distributions and then performs equipercentile equating using the estimated conditional distributions.

- The function `ker.eq()` implements the kernel method of equating under the EG, SG, CB, NEAT PSE, and NEAT CE designs. Currently, the Gaussian, uniform and logistic kernels are available. `ker.eq()` makes calls to other functions in order to obtain the parameters that are needed for kernel equating (e.g., the bandwidth and score probability parameters h, r, and s, respectively). The following functions to perform different tasks in the five steps of kernel equating are available: `loglin.smooth()` is a design specific function helping in the pre-smoothing (Step 1). The function also estimates the score probabilities (Step 2) needed for the computation of the equating function, and the C matrices needed in the calculation of the SEE. The `bandwidth()` function is both a design specific and kernel specific function used to obtain optimal h values used in the continuization step. For the final stage (Step 5), the `SEED()` function performs standard error of equating differences by using two objects of class '`ker.eq`' returned by `ker.eq()`. The function `PREp()` is also available to calculate the PREp diagnostic measure (see the information on KE in Section 2.1).

- The `irt.link()` function implements four IRT parameter linking methods: the mean-mean, mean-sigma, Haebara, and Stocking-Lord methods. For the characteristic curve methods (i.e., the Haebara and Stocking-Lord), besides the traditional logistic ICC used in IRT modeling, an asymmetric cloglog link is also available as an option.

- Print and summary methods are provided for most of objects returned by the above described functions.

- In order to illustrate the use of the functions in **SNSequate**, six data sets widely analyzed in the equating literature are provided: `Math20EG`, `Math20SG`, and `CBdata` to illustrate the KE method under the equivalent, single and counterbalanced group designs respectively (von Davier *et al.* 2004); and `ACTmKB`, `KB36`, and `KB36.1PL` to illustrate mean, linear, and equipercentile equating as well as mean-mean, mean-sigma, Haebara and SL irt parameter linking methods (Kolen and Brennan 2004).

4. Examples

In the following sections the main features of **SNSequate** are demonstrated using the provided data sets for each of the functions described in Section 3.

4.1. Mean, linear and equipercentile equating

Kolen and Brennan (2004) use data from two 40 items test forms of the ACT mathematics test. Form $\mathcal{X}$ is administered to 4,329 examinees, whereas form $\mathcal{Y}$ to 4,152. The following code can be used to reproduce Table 2.7 of Kolen and Brennan (2004), where equated scores are obtained by using mean, linear and equipercentile equating.

```
R> data("ACTmKB", package = "SNSequate")
R> act.m <- mea.eq(rep(0:40, ACTmKB[, 1]), rep(0:40, ACTmKB[, 2]), 0:40)
R> act.l <- lin.eq(rep(0:40, ACTmKB[, 1]), rep(0:40, ACTmKB[, 2]), 0:40)
R> act.e <- eqp.eq(rep(0:40, ACTmKB[, 1]), rep(0:40, ACTmKB[, 2]), 0:40)
R> Table2.7 <- cbind(0:40, act.m$resu, act.l$resu, act.e$resu)
R> colnames(Table2.7) <- c("Score", "Mean", "Linear", "Equipercentile")
R> Table2.7
```

```
      Score       Mean     Linear Equipercentile
 [1,]     0 -0.8726221 -2.6319708      0.0000000
 [2,]     1  0.1273779 -1.5433493      0.9795565
 [3,]     2  1.1273779 -0.4547278      1.6462231
 [4,]     3  2.1273779  0.6338937      2.2856318
 [5,]     4  3.1273779  1.7225152      2.8931979
 [6,]     5  4.1273779  2.8111367      3.6204666
 [7,]     6  5.1273779  3.8997582      4.4996535
 [8,]     7  6.1273779  4.9883797      5.5148375
 [9,]     8  7.1273779  6.0770012      6.3124157
[10,]     9  8.1273779  7.1656227      7.2242386
[11,]    10  9.1273779  8.2542443      8.1606665
[12,]    11 10.1273779  9.3428658      9.1826961
[13,]    12 11.1273779 10.4314873     10.1858956
[14,]    13 12.1273779 11.5201088     11.2513015
[15,]    14 13.1273779 12.6087303     12.3896334
[16,]    15 14.1273779 13.6973518     13.3928909
[17,]    16 15.1273779 14.7859733     14.5240050
[18,]    17 16.1273779 15.8745948     15.7169010
[19,]    18 17.1273779 16.9632163     16.8234423
[20,]    19 18.1273779 18.0518378     18.0092239
[21,]    20 19.1273779 19.1404593     19.1647208
[22,]    21 20.1273779 20.2290808     20.3676007
[23,]    22 21.1273779 21.3177023     21.4556277
[24,]    23 22.1273779 22.4063238     22.6871228
[25,]    24 23.1273779 23.4949453     23.9156570
[26,]    25 24.1273779 24.5835668     25.0291585
[27,]    26 25.1273779 25.6721883     26.1612293
[28,]    27 26.1273779 26.7608098     27.2632870
[29,]    28 27.1273779 27.8494313     28.1800647
[30,]    29 28.1273779 28.9380528     29.1424331
[31,]    30 29.1273779 30.0266743     30.1304817
[32,]    31 30.1273779 31.1152958     31.1297014
[33,]    32 31.1273779 32.2039173     32.1357069
[34,]    33 32.1273779 33.2925388     33.0780678
[35,]    34 33.1273779 34.3811603     34.0171864
```

```
[36,]    35 34.1273779 35.4697818     35.1016041
[37,]    36 35.1273779 36.5584033     36.2425502
[38,]    37 36.1273779 37.6470248     37.1247622
[39,]    38 37.1273779 38.7356463     38.1320883
[40,]    39 38.1273779 39.8242678     39.0807346
[41,]    40 39.1273779 40.9128893     39.9005544
```

The three functions, `mea.eq()`, `lin.eq()`, and `eqp.eq()` receive as the first two arguments the observed scores from forms $\mathcal{X}$ and $\mathcal{Y}$, and the third corresponds to values on the scale that are to be equated.

4.2. Local equating

Using the item parameter estimates in the `KB36` data set, two test forms each administered to a total of 2,500 individuals are simulated to illustrate local equating. For each value of $\theta = (-2.0, -1.0, 0.0, 1.0, 2.0)$, 500 response patterns are generated using the `DataGen()` function described below. From each of the two simulated test forms, sum scores are obtained and used as inputs in the `le.eq()` function to obtain the true equating transformation. The following code shows each of the described steps:

```
R> data("KB36", package = "SNSequate")
R> Itx <- KB36$KBformX_par; Ity <- KB36$KBformY_par
R> Itx.m <- t(Itx[, c(2, 1, 3)])
R> Ity.m <- t(Ity[, c(2, 1, 3)])
R> Th <- rep(seq(-2, 2, 1), each = 500)
R> Pr <- function(theta, b, a = 1, c = 0) c + (1 - c) /
+    (1 + exp(- a * (theta - b)))
R> Pattern <- function(theta, b, a = rep(1, length(b)),
+    c = rep(0, length(b))) rbinom(length(b), 1, Pr(theta, b, a, c))
R> DataGen <- function(Theta, ItemPar) {
+    res <- NULL
+    N <- length(Theta)
+    for (i in 1:N) res <- rbind(res, Pattern(Theta[i], ItemPar[1, ],
+      ItemPar[2, ], ItemPar[3, ]))
+    return(res)
+ }
R> set.seed(10)
R> X <- DataGen(Th, Itx.m); Y <- DataGen(Th, Ity.m)
R> sx <- rowSums(X); sy <- rowSums(Y)
R> true <- le.eq(sx, Itx.m, Ity.m, Th)
R> res <- cbind(true$Theta, true$Obs.Sc, true$resu)
R> unique(res[res[, 2] == 18, ])

     [,1] [,2]     [,3]
[1,]   -1   18 20.13200
[2,]    0   18 20.67140
[3,]    1   18 20.66928
```

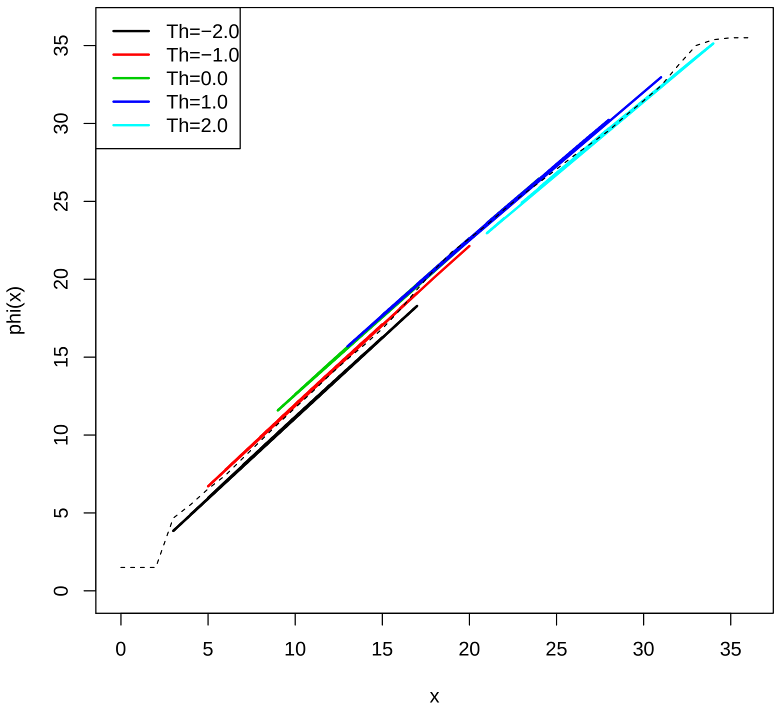

Figure 1: True equating transformations for $\theta = (-2, -1, 0, 1, 2)$ under local equating and equipercentile equating transformation (dashed line)

Because local equating generates a family of equating transformations, different individuals (i.e., with different values of θ) obtain different equated values, regardless of the fact that all of them got the same sum score. This situation is exemplified by restricting the output to show only the results for individuals with $\theta = (-1, 0, 1)$ where all of them obtain a sum score of 18.

A graphical display of the resulting family of transformation is shown in Figure 1, where also the equipercentile transformation has been plotted for comparison. The figure was generated using the following code.

```
R> resu <- vector("list", 9)
R> ind <- seq(-2, 2, 1)
R> for (i in 1:5)
+    resu[[i]] <- unique(res[res[, 1] == ind[i], 2:3])
R> est.eqp <- eqp.eq(sx, sy, 0:36)
R> est.e <- est.eqp$resu[match(rowSums(X), est.eqp$X)]
R> plot(resu[[1]], type = "l", xlim = c(0, 36), ylim = c(0, 36), col = 1,
+    lwd = 2.0, ylab = "phi(x)", xlab = "x")
R> for (i in 2:5)
+    lines(resu[[i]], type = "l", col = i, lwd = 2.0)
R> lines(0:36, est.eqp$resu, type = "l", lty = 2)
R> legend("topleft", lty = rep(1, 5), lwd = rep(2, 5),
+    paste("Th =", format(seq(-2, 2, by = 1), nsmall = 1)), col = 1:5)
```

4.3. Kernel equating

As mentioned in Section 3, different functions are available to carry out the five steps that describe the KE method. In what follows we show examples of the use of `loglin.smooth()`, `bandwidth()`, `ker.eq()`, `PREp()`, and `SEED()` functions, for different kernel and equating designs.

Log-linear models for pre-smoothing

Log-linear models are mostly used for the pre-smoothing of score distributions (see, e.g., Holland and Thayer 1987, 2000; Moses 2011). In this step, the objective is to find a model that best describe the data (i.e., that adequately fits the score distribution), as parsimoniously as possible.

The `loglin.smooth()` function fits log-linear models following the general equation

$$\log(o_{gh}) = \alpha_m + Z_m(z_g) + W_m(w_h) + ZW_m(z_g, w_h), \tag{20}$$

where

$$\begin{aligned}
Z_m(z_g) &= \textstyle\sum_{i=1}^{T_{Zm}} \beta_{zmi}(z_g)^i & (21)\\
W_m(w_h) &= \textstyle\sum_{i=1}^{T_{Wm}} \beta_{Wmi}(w_h)^i & (22)\\
ZW_m(z_g, w_h) &= \textstyle\sum_{i=1}^{I_{Zm}} \sum_{i'=1}^{I_{Wm}} \beta_{ZWmii'}(z_g)^i(w_h)^{i'} & (23)
\end{aligned}$$

Equation 20 can be used to represent various models according to the different designs used. For instance, if the SG design is considered, then $o = p$, $g = j$, $h = k$, $Z = X$, $W = Y$, $z = x$, $w = y$, leading to the log-linear models used in an example below. In general, the possible values for the symbols in (20) are $o = \{p_{(12)}, p_{(21)}, p, q\}$, $Z = \{X, Y\}$, $W = \{Y, A\}$, $z = \{x, y\}$, $w = \{y, a\}$, $m = \{(12), (21), P, Q\}$, $g = \{j, k\}$, $h = \{l, k\}$.

We illustrate the pre-smoothing step using the `loglin.smooth()` function and the `Match20EG` and `Match20SG` data for both univariate and bivariate frequency score distributions according to the EG and SG design, respectively. The example replicates some results reported by von Davier *et al.* (2004) from where the data are obtained.

In the following example a simple log-linear model with 2 power moments is used to obtain the estimated score probabilities $\hat{r}_j$ for X scores. The `degree = 2` argument means that a two-moment fit of the model is required and corresponds to the T_r term in the resulting log-linear equation

$$\log(r_j) = \alpha_r + \sum_{i=1}^{T_r} \beta_{ri}(x_j)^i. \tag{24}$$

```
R> data("Math20EG", package = "SNSequate")
R> loglin.smooth(scores = Math20EG[, 1], degree = 2, design = "EG")

Call:
loglin.smooth.default(scores = Math20EG[, 1], degree = 2, design = "EG")

Estimated score probabilities:
   Score Est.Score.Prob.
1      0     0.002270957
2      1     0.004428770
3      2     0.008098301
4      3     0.013884856
5      4     0.022321610
6      5     0.033646997
7      6     0.047555830
8      7     0.063022827
9      8     0.078312061
10     9     0.091242272
11    10     0.099678200
12    11     0.102103574
13    12     0.098065977
14    13     0.088314586
15    14     0.074573268
16    15     0.059043294
17    16     0.043832338
18    17     0.030510924
19    18     0.019913736
20    19     0.012186718
```

Similar code can be written in order to obtain the estimated score probabilities for Y scores, $\hat{s}_k$. A useful tool in the election of an appropriate log-linear model that helps to assess discrepancies is a plot showing both the observed and fitted score distributions. The following code can be used to obtain Figure 2.

```
R> data("Math20EG", package = "SNSequate")
R> rj <- loglin.smooth(scores = Math20EG[, 1], degree = 2,
+    design = "EG")$sp.est
R> sk <- loglin.smooth(scores = Math20EG[, 2], degree = 3,
+    design = "EG")$sp.est
R> score <- 0:20
R> plot(score, Math20EG[, 1], pch = 0, ylab = "Frequency", xlab = "Score")
R> points(score, rj * 1453, pch = 15)
R> legend("topright", pch = c(0, 15), c("Observed", "Fitted"))
```

When pre-smoothing bivariate frequencies, the data matrix contains the (joint) bivariate sample frequencies for X in rows, and for Y in columns looking as follows:

```
R> data("Math20SG", package = "SNSequate")
R> head(Math20SG, 10)
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
 [1,]    0    0    1    0    0    0    0    0    0     0     0     0     0
 [2,]    0    0    1    0    1    1    0    0    0     0     0     0     0
 [3,]    0    0    1    0    2    1    3    0    0     1     0     0     0
```

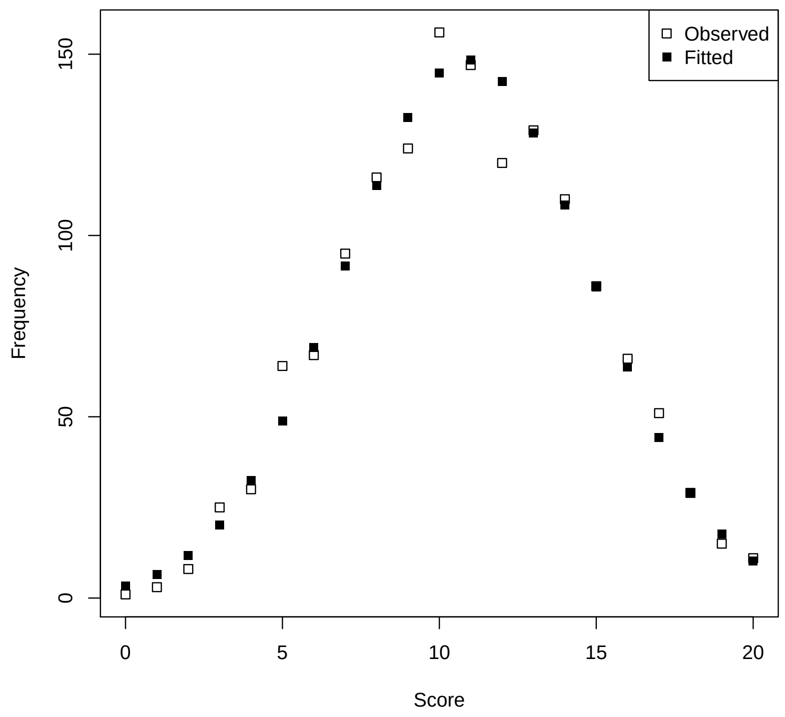

Figure 2: The observed and the fitted distribution of X.

```
 [4,]    0    0    1    5    6    3    8    1    1     0     0     0     0
 [5,]    0    0    2    7    4    6    4    3    1     3     0     0     0
 [6,]    0    0    3    3    5   12   14    8    9     6     3     1     0
 [7,]    0    0    1    4   10    9   12    9    8    10     4     0     0
 [8,]    0    0    1    3    5    7   16   16   11    17    10     5     3
 [9,]    0    0    1    1    3    8   16   14   12    24    20    11     3
[10,]    0    0    0    1    3    4    8   19   20    17    17    13    11
      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]
 [1,]     0     0     0     0     0     0     0     0
 [2,]     0     0     0     0     0     0     0     0
 [3,]     0     0     0     0     0     0     0     0
 [4,]     0     0     0     0     0     0     0     0
 [5,]     0     0     0     0     0     0     0     0
 [6,]     0     0     0     0     0     0     0     0
 [7,]     0     0     0     0     0     0     0     0
 [8,]     0     1     0     0     0     0     0     0
 [9,]     3     0     0     0     0     0     0     0
[10,]     9     2     0     0     0     0     0     0
```

For instance, if one wants to pre-smooth the bivariate frequency distribution according to the SG design, using 3 power moments for each (marginal) X and Y score and one crossed interaction term XY, the log-linear equation model reads as

$$\log(p_{jk}) = \alpha + \sum_{i=1}^{T_X} \beta_{Xi}(x_j)^i + \sum_{i=1}^{T_Y} \beta_{Yi}(y_k)^i + \beta_{XY} x_j y_k, \tag{25}$$

and the corresponding code is

```
R> data("Math20SG", package = "SNSequate")
R> loglin.smooth(scores = Math20SG, degree = c(3, 3, 1, 1), design = "SG")

Call:
loglin.smooth.default(scores = Math20SG, degree = c(3, 3, 1,
    1), design = "SG")

Estimated score probabilities:

   Score           r           s
1      0 0.001583093 0.001578752
2      1 0.003561066 0.003623786
3      2 0.007203015 0.007473589
4      3 0.013230277 0.013976282
5      4 0.022240753 0.023870355
6      5 0.034421089 0.037425573
7      6 0.049260072 0.054058035
8      7 0.065405878 0.072113498
9      8 0.080796899 0.089016839
10     9 0.093091894 0.101848078
11    10 0.100276704 0.108179978
12    11 0.101221187 0.106838498
13    12 0.095968537 0.098250225
14    13 0.085656941 0.084233492
15    14 0.072132287 0.067359828
16    15 0.057425354 0.050197807
17    16 0.043288323 0.034746895
18    17 0.030921249 0.022198047
19    18 0.020916943 0.012962520
20    19 0.013366505 0.006835540
21    20 0.008031934 0.003212383
```

The first and second components of the vector in the `degree = c(3, 3, 1, 1)` argument corresponds respectively to T_X and T_Y in (25). The third and fourth components correspond respectively to the values of I and L in the expression $\sum_{i=1}^{I} \sum_{l=1}^{L} \beta_{XYil}(x_j)^i(y_k)^l$.

Besides the visual inspection of the pre-smoothed score distributions, formal statistical procedures used to assess the fit of log-linear models can be used to select an appropriate polynomial degree. For details on these methods, the reader is referred to Hanson (1994, 1996) and Holland and Thayer (2000).

Selecting an optimal bandwidth parameter

The way to optimally select the h bandwidth parameter used in kernel equating is described in von Davier *et al.* (2004). The `bandwidth()` function automatically selects h by minimizing

$$\mathrm{PEN}_1(h) + K \times \mathrm{PEN}_2(h), \tag{26}$$

where $\mathrm{PEN}_1(h) = \sum_j (\hat{r}_j - \hat{f}_h(x_j))^2$, and $\mathrm{PEN}_2(h) = \sum_j A_j(1 - B_j)$. The terms A and B are such that PEN_2 acts as a smoothness penalty term that avoids rapid fluctuations in the approximated density (see, e.g., Chapter 10 in von Davier 2011b, for more details). The K term in (26) corresponds to the `Kp` argument of the `bandwidth()` function and its default value is set to 1. The $\hat{r}$ values are assumed to be estimated by polynomial log-linear models of a specific `degree`, which come from a call to `loglin.smooth()`. The following example shows how to obtain h_X:

```
R> hx.gauss <- bandwidth(scores = Math20EG[, 1], kert = "gauss", degree = 2,
+    design = "EG")
R> hx.gauss

Automatically selected bandwidth parameter:

[1] 0.6222771
```

Note that the `bandwidth()` function is design specific. That is, it will find the optimal values of bandwidth parameters according to the selected design. For example, in the CB design, both h_X and h_Y depend on weights w_x and w_y, respectively. The arguments `wx` and `wy` can be varied to obtain, for instance, F_1 and G_1 or $F_{1/2}$ and $G_{1/2}$ as shown in the following example:

```
R> data("CBdata", package = "SNSequate")
R> bandwidth(scores = CBdata$datx1y2, kert = "gauss",
+    degree = c(2, 2, 1, 1), design = "CB", Kp = 0, scores2 = CBdata$datx2y1,
+    J = 76, K = 77, wx = 1, wy = 1)

Automatically selected bandwidth parameter:

         hx        hy
1 0.5582462 0.6100749

R> bandwidth(scores = CBdata$datx1y2, kert = "gauss", degree = c(2, 2, 1, 1),
+    design = "CB", Kp = 0, scores2 = CBdata$datx2y1, J = 76, K = 77,
+    wx = 0.5, wy = 0.5)

Automatically selected bandwidth parameter:

         hx        hy
1 0.5580289 0.6246032
```

Note that in the previous examples, a call to `loglin.smooth()` is made in order to obtain $\hat{r}_j$ and $\hat{s}_j$ by fitting log-linear models with power degree 2 for both X and Y and no interaction term.

An additional feature of `bandwidth()` is that it is also a kernel-specific function. This means that we can also find optimal values of h considering different kernels other than the Gaussian as shown in the following example which replicates part of Table 10.1 in Lee and von Davier (2011):

```
R> hx.logis <- bandwidth(scores = Math20EG[, 1], kert = "logis", degree = 2,
+    design = "EG")$h
R> hx.unif <- bandwidth(scores = Math20EG[, 1], kert = "unif", degree = 2,
+    design = "EG")$h
R> hx.gauss <- bandwidth(scores = Math20EG[, 1], kert = "gauss", degree = 2,
+    design = "EG")$h
R> hy.logis <- bandwidth(scores = Math20EG[, 2], kert = "logis", degree = 3,
+    design = "EG")$h
R> hy.unif <- bandwidth(scores = Math20EG[, 2], kert = "unif", degree = 3,
+    design = "EG")$h
R> hy.gauss <- bandwidth(scores = Math20EG[, 2], kert = "gauss", degree = 3,
+    design = "EG")$h
R> partialTable10.1 <- rbind(c(hx.logis, hx.unif, hx.gauss),
+    c(hy.logis, hy.unif, hy.gauss))
R> dimnames(partialTable10.1) <- list(c("h.x", "h.y"),
+    c("Logistic", "Uniform", "Gaussian"))
R> partialTable10.1
```

```
     Logistic  Uniform  Gaussian
h.x 0.5117115 1.002887 0.6222771
h.y 0.4462512 1.002701 0.5706367
```

The dependence on a specific kernel is clearly seen in the expression for $\text{PEN}_1(h)$ in Equation 26.

Obtaining equated scores

As described in Section 2.1, the conversion of scores from test X to test Y is based on $\hat{e}_Y(x) = G^{-1}_{h_Y}(F_{h_X}(x;\hat{r});\hat{s})$ where $\hat{r}$ and $\hat{s}$ are estimated score probabilities obtained in a pre-smoothing stage (e.g., by using the `loglin.smooth()` function) and h_X and h_Y are bandwidth parameters (that can optimally be obtained by using the `bandwidth()` function). The `ker.eq()` function computes the equating transformation used in the KE method for various designs. The function makes calls to both `bandwidth()` (in case the bandwidth parameters are not specified by the user) and `loglin.smooth()`. Further, the `ker.eq()` function calculates many useful quantities such as summary statistics for the used data, the actual equated values, the SEE, and the SEE vector used by `SEED()` to obtain standard errors of equating difference between two equating functions, among others. A `summary()` method which summarizes the most important output is also available.

The following example shows the implementation of the KE method of equating under the EG design and replicates results obtained by von Davier *et al.* (2004, Chapter 7).

```
R> mod.gauss <- ker.eq(scores = Math20EG, kert = "gauss", hx = NULL,
+    hy = NULL, degree = c(2, 3), design = "EG")
R> summary(mod.gauss)
```

```
Call:
ker.eq.default(scores = Math20EG, kert = "gauss", hx = NULL,
    hy = NULL, degree = c(2, 3), design = "EG")

Summary statistics
                 X         Y
Total    1453.0000 1455.0000
Mean       10.8183   11.5931
SD          3.8072    3.9356
Skewness    0.0026   -0.0627
Kurtosis    2.5322    2.5012

Bandwidth parameters used
         hx        hy
1 0.6222771 0.5706367

Kernel type used
[1] "Gaussian"

Equated values and SEE under the EG design
   Score       eqYx       eqXy      SEEYx     SEEXy
1      0  0.3937442 -0.3215729 0.22003960 0.1453396
2      1  1.5813111  0.4964543 0.28953082 0.2253615
3      2  2.6403736  1.3862030 0.28750519 0.2750657
4      3  3.6443827  2.3557595 0.26639230 0.2794128
5      4  4.6316374  3.3603884 0.24103584 0.2607145
6      5  5.6177604  4.3748434 0.21694965 0.2350742
7      6  6.6099737  5.3870361 0.19666336 0.2102841
8      7  7.6120208  6.3912272 0.18124181 0.1896855
9      8  8.6259784  7.3847060 0.17074920 0.1742651
10     9  9.6530124  8.3661662 0.16457116 0.1637154
11    10 10.734827  9.3353834 0.16187098 0.1571470
12    11 11.7471374 10.2925429 0.16210073 0.1537034
13    12 12.8126165 11.2385839 0.16533579 0.1529790
14    13 13.8868780 12.1751816 0.17213294 0.1551647
15    14 14.9641259 13.1051931 0.18265190 0.1608539
16    15 16.0338580 14.0334208 0.19504820 0.1705237
17    16 17.0781117 14.9680024 0.20375790 0.1838001
18    17 18.0676535 15.9236014 0.19900325 0.1986204
19    18 18.9607437 16.9287558 0.16999801 0.2098634
20    19 19.7183061 18.0476999 0.11860262 0.2047477
21    20 20.3929890 19.4152753 0.07030393 0.1441086
```

In this example, the arguments `hx = NULL, hy = NULL` instruct the program to automatically select the optimal bandwidth parameters. The `degree = c(2, 3)` argument is used by `loglin.smooth()` in the way explained earlier.

Evaluation of KE results

As mentioned in Section 2.1, the percentage of relative error (PRE) serves as a measure to assess the adequacy of the estimated equating function $\hat{e}_Y(x)$. The measure is formally defined as

$$\text{PRE}(p) = 100\frac{\mu_p(e_Y(X)) - \mu_p(Y)}{\mu_p(Y)}, \tag{27}$$

where $\mu_p(Y) = \sum_k (y_k)^p s_k$ and $\mu_p(e_Y(X)) = \sum_j (e_Y(x_j))^p r_j$. Similar formulas can be found when equating from Y to X. The `PREp()` function can be used for this purpose. It takes as an argument an object of class '`ker.eq`' and the number of moments to be evaluated, and gives the corresponding PRE values as output. For example, using the `mod.gauss` object, the current example shows how to obtain the first 10 moments.

```
R> PREp(mod.gauss, 10)
```

```
Percent Relative Error:

   Moments       X_to_Y       Y_to_X
1        1 0.005865208 -0.00629265
2        2 0.012213344 -0.02275206
3        3 0.021859915 -0.05492007
4        4 0.039627669 -0.11380496
5        5 0.072654048 -0.21331434
6        6 0.127283594 -0.36628326
7        7 0.208690241 -0.58357207
8        8 0.320864781 -0.87360195
9        9 0.466756733 -1.24225903
10      10 0.648465369 -1.69303917
```

A final step in the equating process is the calculation of accuracy measures. For the KE method, the SEE is defined as (von Davier *et al.* 2004, Equation 5.15)

$$\text{SEE}_Y(x) = \parallel \mathbf{J}_{e_Y} \mathbf{J}_{DF} \mathbf{C} \parallel . \tag{28}$$

The `ker.eq()` function internally calculates the Jacobian matrices $\mathbf{J}_{e_Y}$ and $\mathbf{J}_{DF}$. The $\mathbf{C}$ matrix is obtained as an output of `loglin.smooth()`. As shown in previous examples, the SEE is part of the `summary()` method for '`ker.eq`' objects.

A graphical display of the SEE's range for each score point can easily be obtained. Figure 3 shows an example when equating from X to Y and it was generated using the code

```
R> plot(score, mod.gauss$SEEYx, ylab = "SEEy(x)", xlab = "X raw Score")
```

In order to decide between two equating functions, von Davier *et al.* (2004) propose the use of a measure called the standard error of equating difference (SEED). The measure is formally defined as

$$\text{SEED}_Y(x) = \sqrt{\mathsf{VAR}(\hat{e}_1(x) - \hat{e}_2(x))} \tag{29}$$

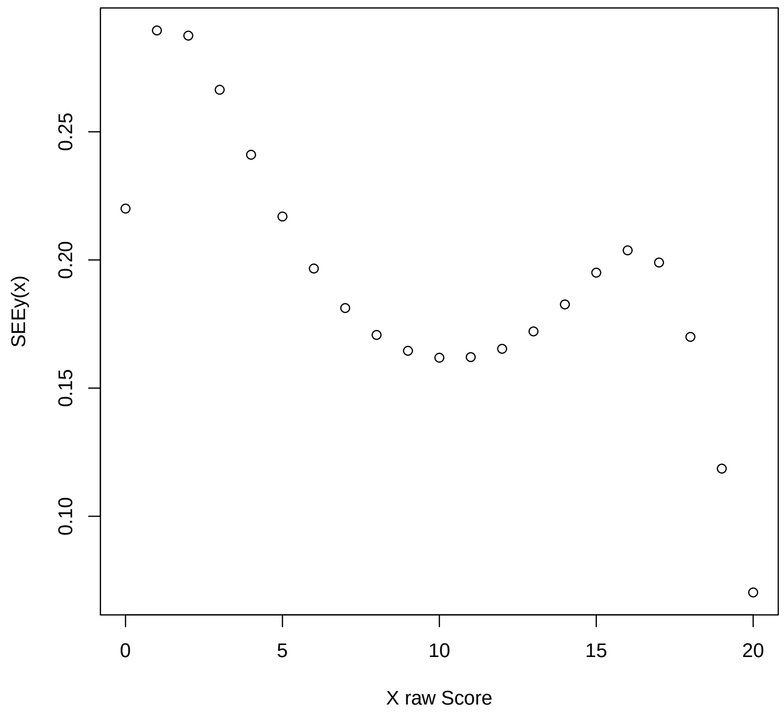

Figure 3: Standard error of equating for $\hat{e}_Y(x)$.

and is based on the SE-vector $\mathbf{J}_{e_Y}\mathbf{J}_{DF}\mathbf{C}$ of each equating function being considered in the comparison. The SE-vector is part of the output of `ker.eq()`. The `SEED()` function uses as arguments two objects of class '`ker.eq`' and returns the SEED between them. For example, following Theorem 1.1 from von Davier *et al.* (2004), the KE function approximates the standard linear equating function for large values of the bandwidth parameters. The previously obtained `mod.gauss` object is used to evaluate the difference between the Gaussian KE function and the linear equating method when equating from X to Y using `SEED()` as follows:

```
R> mod.linear <- ker.eq(scores = Math20EG, kert = "gauss", hx = 20, hy = 20,
+    degree = c(2, 3), design = "EG")
R> seed <- SEED(mod.gauss, mod.linear)$SEEDYx
R> seed
```

```
 [1] 0.21110509 0.19417797 0.16552026 0.12977915 0.09313094 0.05870588
 [7] 0.02900790 0.01319329 0.02533356 0.03842340 0.04544569 0.04544507
[13] 0.03870275 0.02801723 0.02520540 0.04104023 0.06278147 0.08237360
[19] 0.10567748 0.14907852 0.20230771
```

where it can be seen that the SEE differences range from 0.01 to 021. The difference between each of the equated values obtained using the Gaussian KE function with those obtained using the approximated linear equating function can be assessed for significance comparing them to their uncertainty (i.e., its SEED). A graphical alternative is to plot such differences of equated values along ±2SEED. Figure 4 shows an example of this and it was generated using the following code

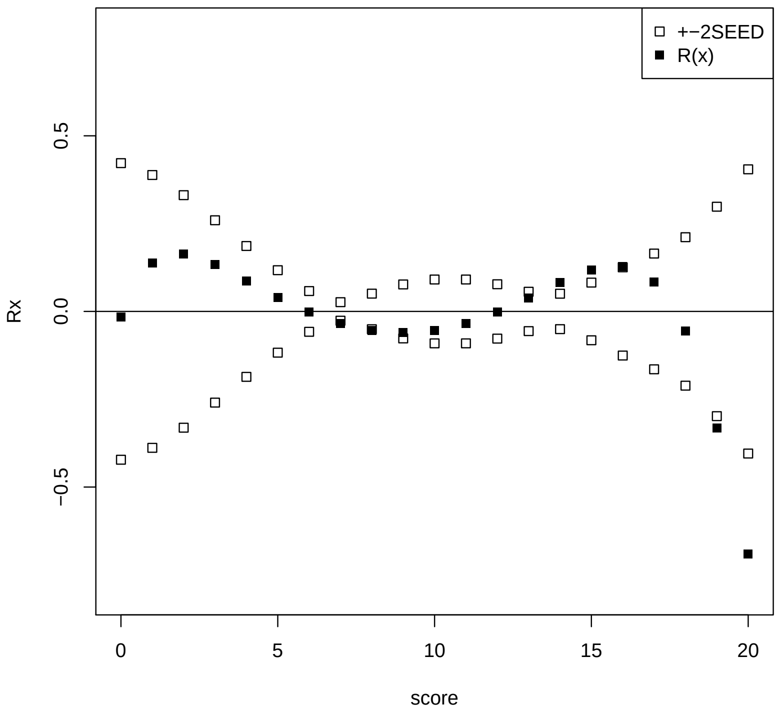

Figure 4: Difference between linear and KE equating functions ±2SEED.

```
R> Rx <- mod.gauss$eqYx - mod.linear$eqYx
R> plot(score, Rx, ylim = c(-0.8, 0.8), pch = 15)
R> abline(h = 0)
R> points(score, 2 * seed, pch = 0)
R> points(score, -2 * seed, pch = 0)
R> legend("topright", pch = c(0, 15), c("+-2SEED", "R(x)"))
```

If different types of kernels, other than the Gaussian are used for equating, the SEED is a useful tool to compare and decide between them. Figure 5 shows the difference between the Gaussian kernel equating function and both the logistic and uniform kernel equating functions. The figure was generated using the following code and replicate results found in Lee and von Davier (2011).

```
R> mod.unif <- ker.eq(Math20EG, "unif", hx = NULL, hy = NULL, c(2, 3), "EG")
R> mod.logis <- ker.eq(Math20EG, "logis", hx = NULL, hy = NULL, c(2, 3),
+    "EG")
R> Rx1 <- mod.logis$eqYx - mod.gauss$eqYx
R> Rx2 <- mod.unif$eqYx - mod.gauss$eqYx
R> seed1 <- SEED(mod.logis, mod.gauss)$SEEDYx
R> seed2 <- SEED(mod.unif, mod.gauss)$SEEDYx
R> plot(score, Rx1, ylim = c(-0.2, 0.2), pch = 15, main = "LK vs GK",
+    ylab = "", xlab = "Scores")
R> abline(h = 0)
R> points(score, 2 * seed1, pch = 0)
R> points(score, -2 * seed1, pch = 0)
```

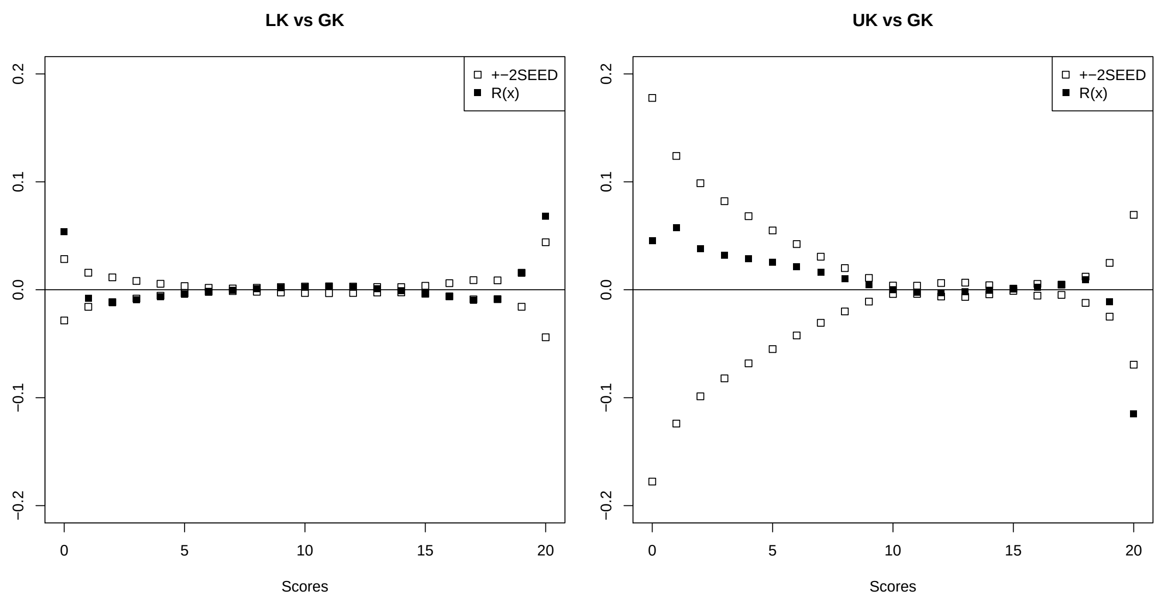

Figure 5: Difference and SEED between the estimated Gaussian kernel (GK) equating function and both the logistic (LK) and uniform (UK) kernel equating functions.

4.4. IRT item parameter linking methods

Kolen and Brennan (2004) use data from two 36 items test forms administered to 1,655 (form $\mathcal{X}$) and 1,638 (form $\mathcal{Y}$) examinees, respectively. In this test, 12 out of the 36 items are common between both test forms (items 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36). The following example code illustrates how to use the `irt.link()` function to reproduce results reported in Table 6.6 by Kolen and Brennan (2004).

```
R> data("KB36", package = "SNSequate")
R> parm.x <- KB36$KBformX_par
R> parm.y <- KB36$KBformY_par
R> comitems <- seq(3, 36, 3)
R> parm <- data.frame(parm.y, parm.x)
R> irt.link(parm, comitems, model = "3PL", icc = "logistic", D = 1.7)

Call:
irt.link.default(parm = parm, common = comitems, model = "3PL",
    icc = "logistic", D = 1.7)

IRT parameter-linking constants:

                     A          B
Mean-Mean      1.217266 -0.5571557
Mean-Sigma     1.168891 -0.5155426
Haebara        1.093600 -0.4582959
Stocking-Lord  1.101954 -0.4770156
```

The `irt.link()` function receives as arguments a data frame containing the item parameter estimates (`parm`), a numerical vector indicating the position where the common items are located (`comitems`), and both the type of IRT model and ICC that were used to obtain the item parameter estimates. It gives as output the values of the constants A and B calculated using the mean-mean, mean-sigma, Haebara, and Stocking-Lord method.

Although the logistic ICC is the most commonly used method in IRT modeling for binary data, symmetric links are not always appropriate for modeling this kind of data (Chen, Dey, and Shao 1999; Chen 2004). When asymmetric ICCs are used in IRT models, methods of item parameter linking based on ICCs (i.e., Haebara and Stocking-Lord) should accordingly be based on those asymmetric ICCs (Estay 2012). This option is available in the `irt.link()` function when a 1PL model with asymmetric cloglog ICC is used to fit the data. The following example uses item parameter estimates which were obtained under a 1PL model with asymmetric cloglog ICCs

```
R> data("KB36.1PL", package = "SNSequate")
R> b.log.y <- KB36.1PL$b.logistic[, 2]
R> b.log.x <- KB36.1PL$b.logistic[, 1]
R> b.clog.y <- KB36.1PL$b.cloglog[, 2]
R> b.clog.x <- KB36.1PL$b.cloglog[, 1]
R> parm2 <- data.frame(1, b.log.y, 0, 1, b.log.x, 0)
R> parm3 <- data.frame(1, b.clog.y, 0, 1, b.clog.x, 0)
R> comitems <- seq(3, 36, 3)
R> symlink <- irt.link(parm2, comitems, model = "1PL", icc = "logistic",
+    D = 1.7)
R> asylink <- irt.link(parm3, comitems, model = "1PL", icc = "cloglog",
+    D = 1.7)
R> res <- rbind(c(asylink$Haebara[2], asylink$StockLord[2]),
+    c(symlink$Haebara[2], symlink$StockLord[2]))
R> rownames(res) <- c("Asymmetric ICC", "Symmetric ICC")
R> colnames(res) <- c("Haebara", "SL")
R> res
```

```
                  Haebara         SL
Symmetric ICC  -0.2446079 -0.2435928
Asymmetric ICC -0.3449492 -0.3436453
```

Because a 1PL model is fitted, the constant A is equal to 1 and the output is then restricted to show only the values of B. Although both the Haebara and Stocking-Lord methods give similar results, it is noted that the obtained linking constant B is quite different for the symmetric and asymmetric model, respectively.

5. Concluding remarks

The development of new equating models has become common in equating research. Motivated by the lack of software implementing traditional and new equating methods, this paper introduced the **SNSequate** R package which implements both standard and nonstandard statistical models and methods for test equating.

The examples used to demonstrate the capabilities of **SNSequate** were based on widely known data that has been analyzed in the equating literature, which helps illustrate the implemented methods. Moreover, all the examples can be easily modified to accommodate user's data.

Many improvements can be made in future versions of the package. For instance, `plot()` methods can be added for the objects returned by the `loglin.smooth()`, `ker.eq()`, and `SEED()` functions. This would allow for easily obtaining plots of the fitted score distributions, and of the observed and fitted conditional means and standard deviations of the two distributions being considered for the equating. In the case of `ker.eq()`, plots of the continuized distributions as well as the estimated equating functions evaluated at each score point will be considered. For `SEED()` this would allow to reproduce a plot like the one shown in Figure 4. These improvements will be implemented in the next version of **SNSequate**.

The flexibility and modularity of **SNSequate** allow one to easily extend existing methods and implement new ones. The following methods are currently being investigated by the author and will be part of future versions of **SNSequate**:

- *Bayesian nonparametric methods of equating.* These methods have shown good performance compared to alternative approaches (e.g., Karabatsos and Walker 2009). A Bayesian nonparametric model for test equating which allows the use of covariates in the estimation of the score distribution functions that lead to the equating transformation is described in González, Barrientos, and Quintana (2013). A major feature of this approach is that the complete shape of the score distribution may change as a function of the covariates. As a consequence, the form of the equating transformation can change according to covariate values.

- *Epanechnikov and adaptive kernels.* Cid and von Davier (2009) examined potential boundary bias effects in the score distributions continuized by kernel smoothing. The use of the Epanechnikov and adaptive kernels in the framework of kernel equating is currently being investigated by the author.

- *IRT equating methods based on asymmetric ICCs.* Although the use of asymmetric ICCs has been studied in IRT models (e.g., Samejima 2000), their role in IRT equating methods is unclear. IRT equating methods based on asymmetric ICCs are planned to be included in future versions of **SNSequate**. Additionally, besides the cloglog link currently implemented in the `irt.link()` function, other asymmetric links such as the skew-normal, will be included.

Acknowledgments

The author is financially supported by grant FONDECYT 11110044. The author thanks Alina von Davier for valuable comments and discussion on previous versions of the paper; Garritt Page for help with proofreading; and to Gabriel Villalobos and Gionella Ramenzoni for assistance in some of the R programming.

References

Albano A (2014). ***equate**: Statistical Methods for Test Score Equating.* R package version 2.0-2, URL http://CRAN.R-project.org/package=equate.

Andersson B, Bränberg K, Wiberg M (2013). "Performing the Kernel Method of Test Equating with the Package **kequate**." *Journal of Statistical Software*, **55**(6), 1–25. URL http://www.jstatsoft.org/v55/i06/.

Andersson B, Branberg K, Wiberg M (2014). ***kequate**: The Kernel Method of Test Equating.* R package version 1.4.0, URL http://CRAN.R-project.org/package=kequate.

Baker FB (1993). ***EQUATE**: A Computer Program for the Characteristic Curve Method of IRT Equating.* University of Wisconsin, Madison. Version 1.0.

Baker FB, Kim SH (2004). *Item Response Theory: Parameter Estimation Techniques.* Marcel Dekker, New York.

Birnbaum A (1968). "Some Latent Trait Models and Their Use in Inferring any Examinee's Ability." In FM Lord, MR Novick (eds.), *Statistical Theories of Mental Test Scores*, pp. 395–479. Adison-Wesley.

Branberg K, Wiberg M (2011). "Observed Score Linear Equating with Covariates." *Journal of Educational Measurement*, **48**(4), 419–440.

Chen MH (2004). "Skewed Link Models for Categorical Response Data." In M Genton (ed.), *Skew-Elliptical Distributions and Their Applications: A Journey beyond Normality*, volume 1, pp. 131–152. Chapman & Hall/CRC.

Chen MH, Dey DK, Shao QM (1999). "A New Skewed Link Model for Dichotomous Quantal Response Data." *Journal of the American Statistical Association*, **94**(448), 1172–1186.

Choi SW, Gibbons LE, Crane PK (2011). "**lordif**: An R Package for Detecting Differential Item Functioning Using Iterative Hybrid Ordinal Logistic Regression/Item Response Theory and Monte Carlo Simulations." *Journal of Statistical Software*, **39**(8), 1–30. URL http://www.jstatsoft.org/v39/i08/.

Cid J, von Davier A (2009). "Examining Potential Boundary Bias Effects in Kernel Smoothing on Equating." Paper presented at the Annual Meeting of the National Council of Measurement in Education.

Cook LL, Eignor DR (1991). "IRT Equating Methods." *Educational Measurement: Issues and Practice*, **10**(3), 37–45.

De Boeck P, Wilson M (eds.) (2004). *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach.* Springer-Verlag.

Doran HC (2010). ***MiscPsycho**: Miscellaneous Psychometrics.* R package version 1.6, URL http://CRAN.R-project.org/package=MiscPsycho.

Dorans N, Holland P (2000). "Population Invariance and Equitability of Tests: Basic Theory and the Linear Case." *Journal of Educational Measurement*, **37**(4), 281–306.

Dorans NJ, Pommerich M, Holland PW (2007). *Linking and Aligning Scores and Scales.* Springer-Verlag.

Estay G (2012). *Characteristic Curves Scale Transformation Methods Using Asymmetric ICCs for IRT Equating.* Unpublished master's thesis, Department of Statistics, Pontificia Universidad Catolica de Chile.

ETS (2011). ***KE** Software (Version 3).* Educational Testing Service, Princeton.

Fischer G, Molenaar I (eds.) (1995). *Rasch Models: Foundations and Recent Developments.* Springer-Verlag.

González J (2013). "Book Review: Statistical Models for Test Equating, Scaling, and Linking." *Applied Psychological Measurement*, **37**(4), 336–339.

González J (2014). ***SNSequate**: Standard and Nonstandard Statistical Models and Methods for Test Equating.* R package version 1.1-1, URL `http://CRAN.R-project.org/package=SNSequate`.

González J, Barrientos AF, Quintana FA (2013). "A Dependent Bayesian Nonparametric Model for Test Equating." *Technical report*, Pontificia Universidad Católica de Chile.

González J, von Davier M (2013). "Statistical Models and Inference for the True Equating Transformation in the Context of Local Equating." *Journal of Educational Measurement*, **50**(3), 315–320.

Han KT (2007). ***IRTEQ**: Windows Application that Implements IRT Scaling and Equating.* University of Massachusetts, Amherst. Version 1.0, URL `http://www.umass.edu/remp/software/irteq/`.

Hanson BA (1994). "An Investigation of Methods for Improving Estimation of Test Score Distributions." *Research Report ACT 90-4*, American College Testing Program.

Hanson BA (1996). "Testing for Differences in Test Score Distributions Using Log-Linear Models." *Applied Measurement in Education*, **9**, 305–321.

Hanson BA (2000). ***mcmequate**: Equating Software for the Multiple-Choice Model.* Version 1.0, URL `http://www.b-a-h.com/software/mcmequate/`.

Hanson BA, Zeng L (1995). ***ST**: A Computer Program for IRT Scale Transformation.* University of Iowa. Version 1.0, URL `http://www.education.uiowa.edu/casma/computer_programs.htm#irt`.

Holland PW, Rubin DB (1982). *Test Equating.* Academic Press.

Holland PW, Thayer DT (1987). "Notes on the Use of Log-linear Models for Fitting Discrete Probability Distributions." *Research Report RR-87-31*, Educational Testing Service, Princeton.

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.

Karabatsos G, Walker SG (2009). "A Bayesian Nonparametric Approach to Test Equating." *Psychometrika*, **74**(2), 211–232.

Kim S, Kolen MJ (2003). ***POLYST**: A Computer Program for Polytomous IRT Scale Transformation.* University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Kim S, Kolen MJ (2004). ***STUIRT**: A Computer Program for Scale Transformation under Unidimensional Item Response Theory Models.* University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Kolen MJ (2004). ***POLYEQUATE***. University of Iowa. Version 1.0, URL http://www.education.uiowa.edu/casma/computer_programs.htm#irt.

Kolen MJ, Brennan RL (2004). *Test Equating, Scaling, and Linking: Methods and Practices.* Springer-Verlag.

Lee K, Oshima TC (1997). ***IpLink**: Item Parameter Linking.* Version 2.0, URL http://education.gsu.edu/eps/4493.html.

Lee YH, von Davier AA (2011). "Equating through Alternative Kernels." In AA von Davier (ed.), *Statistical Models for Test Equating, Scaling, and Linking*, pp. 159–173. Springer-Verlag.

Lord FM (1980). *Applications of Item Response Theory to Practical Testing Problems.* Lawrence Erlbaum Associates, Hillsdale.

Lord FM, Wingersky MS (1984). "Comparison of IRT True-Score and Equipercentile Observed-Score Equatings." *Applied Psychological Measurement*, **8**(4), 453–461.

Moses T, von Davier A (2011). "A SAS IML Macro for Loglinear Smoothing." *Applied Psychological Measurement*, **35**(3), 250–251.

Moses TP (2011). "Log-Linear Models as Smooth Operators: Holland's Statistical Applications and Their Practical Uses." In N Dorans, S Sinharay (eds.), *Looking Back: Proceedings of a Conference in Honor of Paul W. Holland*, pp. 185–202. Springer-Verlag.

Partchev I (2009). ***irtoys**: Simple Interface to the Estimation and Plotting of IRT Models.* R package version 0.1.2, URL http://CRAN.R-project.org/package=irtoys.

R Core Team (2014). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Samejima F (2000). "Logistic Positive Exponent Family of Models: Virtue of Asymmetric Item Characteristic Curves." *Psychometrika*, **65**(3), 319–335.

SAS Institute Inc (2011). *SAS/IML 9.3 User's Guide.* SAS Institute Inc., Cary, NC. URL http://www.sas.com/.

Silverman BW (1986). *Density Estimation for Statistics and Data Analysis.* Chapman and Hall.

Skaggs G, Lissitz RW (1986). "IRT Test Equating: Relevant Issues and a Review of Recent Research." *Review of Educational Research*, **56**(4), 495–529.

Stocking ML, Lord FM (1983). "Developing a Common Metric in Item Response Theory." *Applied Psychological Measurement*, **7**(2), 201–210.

Tuerlinckx F, Rijmen F, Molenberghs G, Verbeke G, Briggs D, den Noortgate WV, Meulders M, De Boeck P (2004). "Estimation and Software." In P De Boeck, M Wilson (eds.), *Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach*, pp. 343–373. Springer-Verlag.

van der Linden W (2011). "Local Observed-Score Equating." In A von Davier (ed.), *Statistical Models for Test Equating, Scaling, and Linking*, pp. 201–223. Springer-Verlag.

van der Linden W (2013). "Some Conceptual Issues in Observed-Score Equating." *Journal of Educational Measurement*, **50**(3), 249–285.

van der Linden WJ, Hambleton RK (1997). *Handbook of Modern Item Response Theory*. Springer-Verlag.

von Davier A (2013). "Observed-Score Equating: An Overview." *Psychometrika*, **78**(4), 605–623.

von Davier AA (2011a). "An Observed-Score Equating Framework." In N Dorans, S Sinharay (eds.), *Looking Back: Proceedings of a Conference in Honor of Paul W. Holland*, pp. 221–238. Springer-Verlag.

von Davier AA (2011b). *Statistical Models for Test Equating, Scaling, and Linking*. Springer-Verlag.

von Davier AA (2011c). "A Statistical Perspective on Equating Test Scores." In A von Davier (ed.), *Statistical Models for Test Equating, Scaling, and Linking*, pp. 1–17. Springer-Verlag.

von Davier AA, Fournier-Zajac S, Holland PW (2007). "An Equipercentile Version of the Levine Linear Observed Score Equating Function Using the Methods of Kernel Equating." *Research Report RR-87-31*, Educational Testing Service, Princeton NJ.

von Davier AA, Holland PW, Thayer DT (2004). *The Kernel Method of Test Equating*. 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/`.

A. Other software for equating

There currently exists other software for either linking or equating or both. A good summary of the main capabilities of some of these software can be found in Weeks (2010). The list given by Weeks (2010) includes the following software packages: **EQUATE** (Baker 1993), **ST** (Hanson and Zeng 1995), **mcmequate** (Hanson 2000), **IpLink** (Lee and Oshima 1997), **POLYST** (Kim and Kolen 2003), **STUIRT** (Kim and Kolen 2004), **POLYEQUATE** (Kolen 2004), **IRTEQ** (Han 2007), **irtoys** (Partchev 2009), and **MiscPsycho** (Doran 2010) software. These software are mainly designed for conducting IRT linking rather than equating, with the exception of **POLYEQUATE** and **IRTEQ** which implement IRT true score equating. We enhanced the list by mentioning further: **CIPE**, **Equating Recipes**, **RAGE-RGEQUATE**, **Equating Error**, **PIE**, **LEGS** (Kolen and Brennan 2004); the **KE** software (ETS 2011); the SAS/IML (SAS Institute Inc. 2011) macro for log-linear smoothing (Moses and von Davier 2011); and the R packages **plink** (Weeks 2010); **lordif** (Choi, Gibbons, and Crane 2011); **equate** (Albano 2014) and **kequate** (Andersson, Bränberg, and Wiberg 2013; Andersson, Branberg, and Wiberg 2014). A description for the non-R packages software can be found on the web site `http://www.education.uiowa.edu/centers/casma/computer-programs.aspx`. In what follows we give some details on the R packages cited.

Although originally developed for purposes other than equating, the **lordif** package (Choi *et al.* 2011) implements the Stocking-Lord method (Stocking and Lord 1983). The **plink** package (Weeks 2010) is designed for linking mixed-format tests using IRT-based methods. It implements IRT true-score and observed-score equating. The **equate** package (Albano 2014) contains functions for non-IRT equating under both random groups and nonequivalent groups with anchor test designs. Mean, linear, equipercentile and circle-arc equating are supported, as are methods for univariate and bivariate pre-smoothing of score distributions. Specific equating methods currently supported include Tucker, Levine observed score, Levine true score, Braun/Holland, frequency estimation, and chained equating. The **kequate** package (Andersson *et al.* 2014) implements exclusively the kernel method of test equating considering the equivalent, single and counterbalanced designs, as well as designs for nonequivalent groups using chain equating, post-stratification, and the nonequivalent groups using covariates (NEC) design (Branberg and Wiberg 2011). An additional feature of the **kequate** package is that it implements item response theory observed score equating (IRT-OSE) in the kernel equating framework.

When comparing some of the methods available in **SNSequate** which are also implemented in other software, all these programs gave very similar results (i.e., the IRT parameter linking methods, and the mean, linear and equipercentile equating methods). Most (but not all) results concerning KE agreed between **kequate** and **SNSequate**. Mainly, there were differences in the calculation of the SEE for all the equating designs. It should be remarked that the comparisons were made using well-known data appearing in the equating literature and that all the results reported by **SNSequate** were in agreement with those appearing in the literature.

Affiliation:

Jorge González
Pontificia Universidad Católica de Chile

Department of Statistics
Av. Vicuña Mackenna 4860
Macul, Santiago, Chile
E-mail: jgonzale@mat.puc.cl
URL: http://mat.puc.cl/~jgonzale/

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 59, Issue 7 *Submitted:* 2012-06-10
August 2014 *Accepted:* 2014-05-06