Journal of Statistical Software

October 2013, Volume 55, Issue 6. *http://www.jstatsoft.org/*

Performing the Kernel Method of Test Equating with the Package kequate

Björn Andersson
Uppsala University

Kenny Bränberg
Umeå University

Marie Wiberg
Umeå University

Abstract

In standardized testing it is important to equate tests in order to ensure that the test takers, regardless of the test version given, obtain a fair test. Recently, the kernel method of test equating, which is a conjoint framework of test equating, has gained popularity. The kernel method of test equating includes five steps: (1) pre-smoothing, (2) estimation of the score probabilities, (3) continuization, (4) equating, and (5) computing the standard error of equating and the standard error of equating difference. Here, an implementation has been made for six different equating designs: equivalent groups, single group, counter balanced, non-equivalent groups with anchor test using either chain equating or post-stratification equating, and non-equivalent groups using covariates. An R package for the kernel method of test equating called **kequate** is presented. Included in the package are also diagnostic tools aiding in the search for a proper log-linear model in the pre-smoothing step for use in conjunction with the R function `glm`.

Keywords: kernel equating, observed-score test equating, item-response theory, R.

1. Introduction

One of the main concerns when using standardized achievement tests is that they are fair. In order to ensure fairness when a test is given at different points in time, or in different versions of the same standardized test, a statistical procedure known as equating is used. The ultimate goal of equating is to adjust scores on different test forms so that the test forms can be used interchangeably (Kolen and Brennan 2004).

The kernel method of test equating is a single unified approach to observed-score test equating, usually presented as a process involving five different steps: pre-smoothing, score probability estimation, continuization, computation of the equating function, and computation of the standard errors of the equating function (von Davier, Holland, and Thayer 2004). The method has a number of advantages over other observed-score test equating methods. In particular, it provides explicit formulas for the standard errors of equating in different designs and directly uses information from the pre-smoothing step in the estimation of these. Kernel equating can also handle equating using covariates in a non-equivalent groups setting and provides a method to compare two different equatings using the standard error of the difference between two equating functions. Since the kernel method of test equating is a unified equating framework with large applicability both for the testing industry and the research community, it is of great interest to create a software package which anyone interested in equating can use.

The aim of this paper is to introduce package **kequate** (Andersson, Bränberg, and Wiberg 2013), an implementation of the kernel method of test equating using five different data collection designs in the statistical programming environment R (R Core Team 2013). For an introduction to R see Venables, Smith, and R Core Team (2013). The content of the package consists of the entirety of the equating aspects of the book *The Kernel Method of Test Equating* by von Davier *et al.* (2004). In addition, our implementation of the kernel method of test equating in R includes the option to use information from an item response theory (IRT) model to conduct an IRT observed-score equating and the option to use unsmoothed input frequencies directly, to enable the comparison of different approaches to observed-score test equating.

The paper is structured as follows. In Section 2 a brief introduction to the kernel equating framework is given. Section 3 introduces the functionality of **kequate**, and Section 4 provides examples of the main functions. Section 5 contains some concluding remarks and presents possible future additions to the package.

2. The kernel method of test equating

This section will comprise a brief description of the kernel method of test equating. For a complete description please read the excellent book by von Davier *et al.* (2004).However, before we can go through the steps of kernel equating, we need to describe the different data collection designs that are available in the package. The first four are standard data collection designs (see, e.g., Kolen and Brennan 2004; von Davier *et al.* 2004). The last data collection design is a more uncommon case and is used if we have additional information which is correlated with the test scores. For a detailed description please refer to Bränberg (2010) and Bränberg and Wiberg (2011).

2.1. Data collection designs

We have incorporated the possibility of five different data collection designs:

- The equivalent groups design (EG): Two independent random samples are drawn from a common population of test takers, P, and the test form X is administered to one sample while test form Y is administered to the other sample. No test takers are taking both X and Y.

- The single group design (SG): Two test forms X and Y are administered to the same group of test takers drawn from a single population P. All test takers are taking both X and Y.

- The counter balanced design (CB): Two test forms X and Y are administered to the same group of test takers drawn from a single population P. One part of the group first takes test form X and then test form Y. The other part of the group takes the test forms in a counterbalanced order, i.e., first test form Y and then test form X. This could also be viewed as two EG designs or as two SG designs.
- The non-equivalent groups with anchor test design (NEAT): A sample of test takers from population P are administered test form X, and another sample of test takers from population Q are administered test form Y. Both samples are also administered a set of common (i.e., anchor) items (test form A). With the NEAT design there are two commonly used equating methods:
  - Chain equating (CE): The idea is to first link test form X to the anchor test form A and then link test form A to test form Y.
  - Post-stratification equating (PSE): The idea is to link both test form X and test form Y to test form A using a synthetic population, which is a blend of populations P and Q. The equating is performed on the synthetic population.
- The non-equivalent groups with covariates design (NEC): A sample of test takers from population P are administered test form X, and another sample of test takers from population Q are administered test form Y. For both samples we also have observations on background variables correlated with the test scores (i.e., covariates). Using a method similar to the NEAT PSE case, a synthetic population is defined and an equating is performed on this population.

2.2. The kernel method of test equating

Following the notation in von Davier *et al.* (2004), let X and Y be the names of the two test forms to be equated and $\mathbf{X}$ and $\mathbf{Y}$ the scores on X and Y. Let T be the target population for which the equating is to be performed. We will assume that the test takers taking the tests are random samples from a population of test takers, so $\mathbf{X}$ and $\mathbf{Y}$ are regarded as random variables. Observations on $\mathbf{X}$ will be denoted by x_j for $j = 1, \ldots, J$. Observations on $\mathbf{Y}$ will be denoted by y_k for $k = 1, \ldots, K$. If $\mathbf{X}$ and $\mathbf{Y}$ are number-right scores, J and K will be the number of items plus one.

We will use

$$r_j = \mathsf{P}\left(\mathbf{X} = x_j \mid T\right) \tag{1}$$

for the probability of a randomly selected individual in population T scoring x_j on test X, and

$$s_k = \mathsf{P}\left(\mathbf{Y} = y_k \mid T\right) \tag{2}$$

for the probability of a randomly selected individual in population T scoring y_k on test Y.

The goal is to find the link between $\mathbf{X}$ and $\mathbf{Y}$ in the form of an equipercentile equating function in the target population T, the population on which the equating is to be done. The equipercentile equating function is defined in terms of the cumulative distribution functions (CDFs) of $\mathbf{X}$ and $\mathbf{Y}$ in the target population. Let

$$F\left(x\right) = \mathsf{P}\left(\mathbf{X} \leq x \mid T\right) \tag{3}$$

and

$$G(y) = \mathsf{P}(\mathbf{Y} \leq y \mid T) \tag{4}$$

be the CDFs of $\mathbf{X}$ and $\mathbf{Y}$ over the target population T. If the two CDFs are continuous and strictly increasing, then the equipercentile equating function of $\mathbf{X}$ to $\mathbf{Y}$ is defined by

$$y = \mathrm{Equi}_Y(x) = G^{-1}(F(x)). \tag{5}$$

With test score data, the CDFs are discrete step functions so the CDFs have to be made continuous somehow. In traditional equipercentile equating this is done using linear interpolation, but kernel equating handles this issue by employing a kernel method instead (Holland and Thayer 1989). The kernel method of test equating includes five steps: pre-smoothing, estimation of the score probabilities, continuization, equating, and computing the standard error of equating (SEE) and the standard error of equating difference (SEED). Most of the steps which comprise what is called the kernel method of test equating were available before the kernel framework was developed, see, e.g., Angoff (1984) for a description of equipercentile equating using linear interpolation and Fairbank (1987) for a discussion on pre-smoothing in equating. SEEs were also derived by Lord (1982) and Jarjoura and Kolen (1985), but prior to the introduction of kernel equating SEEs were not available when using pre-smoothing (Holland, King, and Thayer 1989).

Step 1: Pre-smoothing

In pre-smoothing, a statistical model is fitted to the empirical distribution obtained from the sampled data. We assume that much of the irregularities seen in the empirical distributions are due to sampling error, and the goal of smoothing is to reduce this error. In equating, the raw data are two sets of univariate, bivariate or multivariate discrete distributions (depending on the data collection design). One way to perform pre-smoothing is by fitting a polynomial log-linear model to the relative frequencies obtained from the raw data. We will show this for the NEAT design. For details the interested reader is referred to, e.g., Holland and Thayer (2000) or von Davier *et al.* (2004).

In the NEAT design each test taker has a score on one of the test forms and a score on an anchor test. Let $\mathbf{A}$ be the score on the anchor test form A. Observations on $\mathbf{A}$ will be denoted by a_l for $l = 1, \ldots, L$. Let n_{Xjl} be the number of test takers with $\mathbf{X} = x_j$ and $\mathbf{A} = a_l$, and n_{Ykl} be the number of test takers with $\mathbf{Y} = y_k$ and $\mathbf{A} = a_l$. We assume that $\mathbf{n}_{XA} = (n_{X11}, \ldots, n_{XJL})^\top$ and $\mathbf{n}_{YA} = (n_{Y11}, \ldots, n_{YKL})^\top$ are independent and that they each have a multinomial distribution. The log likelihood function for $\mathbf{X}$ is given by

$$L_X = c_X + \sum_{j,l} n_{Xjl} \log(p_{jl}) \tag{6}$$

where $p_{jl} = \mathsf{P}(\mathbf{X} = x_j, \mathbf{A} = a_l \mid T)$. The target population T is a mixture of the two populations P and Q, $T = wP + (1-w)Q$, where w is selected from the interval $[0, 1]$.

The log-linear model for p_{jl} is given by

$$\log(p_{jl}) = \alpha_X + \sum_{i=1}^{T_X} \beta_{Xi} x_j^i + \sum_{i=1}^{T_A} \beta_{Ai} a_l^i + \sum_{i=1}^{I_X} \sum_{i'=1}^{I_A} \beta_{XAii'} x_j^i a_l^{i'}, \tag{7}$$

where T_X and T_A are the number of univariate moments for tests X and A, respectively, and I_X and I_A are the number of cross-moments for the tests X and A, respectively. The log likelihood function for $\mathbf{Y}$ and the log-linear model for $q_{kl} = \mathsf{P}\left(\mathbf{Y} = y_k, \mathbf{A} = a_l \mid T\right)$ can be written in a similar way. The log-linear models can also contain additional parameters, to take care of lumps and spikes in the marginal distributions. The specification of such models is, however, not discussed further herein (the interested reader is referred to von Davier *et al.* 2004).

Step 2: Estimation of the score probabilities

The score probabilities are obtained from the estimated score distributions from step 1. The most important part of step 2 is the definition and use of the design function. The design function is a function mapping the (estimated) population score distributions into (estimates of) $\mathbf{r}$ and $\mathbf{s}$, where $\mathbf{r} = (r_1, r_2, \ldots, r_J)^\top$ and $\mathbf{s} = (s_1, s_2, \ldots, s_K)^\top$. The function will vary between different data collection designs. For example, in an EG design it is simply the identity function as compared with PSE in a NEAT design, where the design function is given by

$$\begin{pmatrix} \mathbf{r} \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} \sum_l \left(w + \frac{(1-w)\sum_k q_{kl}}{\sum_j p_{jl}} \right) \mathbf{p}_l \\ \sum_l \left((1-w) + \frac{w \sum_j p_{jl}}{\sum_k q_{kl}} \right) \mathbf{q}_l \end{pmatrix}, \tag{8}$$

where $\mathbf{p}_l = (p_{1l}, p_{2l}, \ldots, p_{Jl})^\top$ and $\mathbf{q}_l = (q_{1l}, q_{2l}, \ldots, q_{Kl})^\top$.

Step 3: Continuization

Test score distributions are discrete, and the definition of the equipercentile equating function given in Equation 5 cannot be used unless we deal with this discreteness in some way. Previous to the development of kernel equating, linear interpolation was employed to obtain continuous CDFs from the discrete CDFs (Kolen and Brennan 2004). In kernel equating continuous CDFs are used as approximations to the estimated discrete step-function CDFs generated in the pre-smoothing step. Following von Davier *et al.* (2004), we will use a Gaussian kernel. Logistic and uniform kernels have also been described in the literature (Lee and von Davier 2011) and are available as options in package **kequate**. In what follows, only the formulas for $\mathbf{X}$ are shown, but the computations for $\mathbf{Y}$ are analogous. The discrete CDF $F(x)$ is approximated by

$$F_{h_X}(x) = \sum_j r_j \Phi\left(\frac{x - a_X x_j - (1 - a_X)\mu_X}{h_X a_X} \right), \tag{9}$$

where $\mu_X = \sum_j x_j r_j$ is the mean of $\mathbf{X}$ in the target population T, h_X is the bandwidth, and $\Phi(\cdot)$ is the standard Normal distribution function. The constant a_X is defined as

$$a_X = \sqrt{\frac{\sigma_X^2}{\sigma_X^2 + h_X^2}}, \tag{10}$$

where $\sigma_X^2 = \sum_j (x_j - \mu_X)^2 r_j$ is the variance of $\mathbf{X}$ in the target population T. There are several ways of choosing the bandwidth h_X. We want the density functions to be as smooth as possible without losing the characteristics of the distributions. We recommend the use of a penalty function to deal with this problem, see von Davier *et al.* (2004). For h_X the penalty function is given by

$$\mathbf{PEN}\left(h_X\right) = \sum_j \left(\hat{r}_j - \hat{f}_{h_X}\left(x_j\right)\right)^2 + \kappa \sum_j B_j, \tag{11}$$

where $\hat{f}_{h_X}\left(x\right)$ is the estimated density function, i.e., the derivative of $\hat{F}_{h_X}\left(x\right)$, and κ is a constant. B_j is an indicator that is equal to one if the derivative of the density function is negative a little to the left of x_j and positive a little to the right of x_j, or if the derivative is positive a little to the left of x_j and negative a little to the right of x_j. Otherwise, B_j is equal to zero. With a bandwidth that minimizes $\mathbf{PEN}\left(h_X\right)$ in Equation 11, the estimated continuous density function $\hat{f}_{h_X}\left(x\right)$ will be a good approximation of the discrete distribution of $\mathbf{X}$, without too many modes.

Step 4: Equating

Assume that we are interested in equating $\mathbf{X}$ to $\mathbf{Y}$. If we use the continuized CDFs described previously, we can define the kernel equating function as

$$\hat{e}_Y\left(x\right) = \hat{G}_{h_Y}^{-1}\left(\hat{F}_{h_X}\left(x\right)\right), \tag{12}$$

which is analogous to the equipercentile equating function defined in Equation 5.

Step 5: Calculating the SEE and the SEED

One of the advantages with the kernel method of test equating is that it provides a neat way to compute the SEE. The SEE for equating $\mathbf{X}$ to $\mathbf{Y}$ is given by

$$\mathrm{SEE}_Y\left(x\right) = \sqrt{\mathsf{VAR}\left(\hat{e}_Y\left(x\right)\right)}. \tag{13}$$

In kernel equating the δ-method is used to compute an estimate of the SEE. Let $\mathbf{R}$ and $\mathbf{S}$ be the vectors of pre-smoothed score distributions. If $\mathbf{R}$ and $\mathbf{S}$ are estimated independently, the covariance can be written as

$$\mathrm{Cov}\begin{pmatrix} \hat{\mathbf{R}} \\ \hat{\mathbf{S}} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_R\mathbf{C}_R^\top & 0 \\ 0 & \mathbf{C}_S\mathbf{C}_S^\top \end{pmatrix} = \mathbf{C}\mathbf{C}^\top, \tag{14}$$

where

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_R & 0 \\ 0 & \mathbf{C}_S \end{pmatrix}. \tag{15}$$

The pre-smoothed score distributions are transformed into $\mathbf{r}$ and $\mathbf{s}$ using the design function. The Jacobian of this function is

$$\mathbf{J}_{DF} = \begin{pmatrix} \frac{\partial \mathbf{r}}{\partial \mathbf{R}} & \frac{\partial \mathbf{r}}{\partial \mathbf{S}} \\ \frac{\partial \mathbf{s}}{\partial \mathbf{R}} & \frac{\partial \mathbf{s}}{\partial \mathbf{S}} \end{pmatrix}. \tag{16}$$

In the final step of kernel equating, estimates of $\mathbf{r}$ and $\mathbf{s}$ are used in the equating function to calculate equated scores. The Jacobian of the equating function is given by

$$\mathbf{J}_{e_Y} = \begin{pmatrix} \frac{\partial e_Y}{\partial \mathbf{r}}, & \frac{\partial e_Y}{\partial \mathbf{s}} \end{pmatrix}. \tag{17}$$

If $\begin{pmatrix} \hat{\mathbf{R}} \\ \hat{\mathbf{S}} \end{pmatrix}$ is approximately normally distributed with mean $\begin{pmatrix} \mathbf{R} \\ \mathbf{S} \end{pmatrix}$ and variance given in Equation 14, then

$$\mathsf{VAR}\left(\hat{e}_Y\left(x\right)\right) = \left\|\mathbf{J}_{e_Y}\mathbf{J}_{DF}\mathbf{C}\right\|^2 \tag{18}$$

and

$$\mathrm{SEE}_Y\left(x\right) = \left\|\mathbf{J}_{e_Y}\mathbf{J}_{DF}\mathbf{C}\right\|, \tag{19}$$

where $\|v\|$ denotes the Euclidean norm of vector v.
The SEED, which can be used to compare different kernel equating functions, is defined as

$$\mathrm{SEED}_Y\left(x\right) = \sqrt{\mathsf{VAR}\left(\hat{e}_1\left(x\right) - \hat{e}_2\left(x\right)\right)} = \left\|\mathbf{J}_{e_1}\mathbf{J}_{DF_1}\mathbf{C} - \mathbf{J}_{e_2}\mathbf{J}_{DF_2}\mathbf{C}\right\|, \tag{20}$$

i.e., the Euclidean norm of the difference between the two vectors $\mathbf{J}_{e_1}\mathbf{J}_{DF_1}\mathbf{C}$ and $\mathbf{J}_{e_2}\mathbf{J}_{DF_2}\mathbf{C}$.
The equating function is designed to transform the continuous approximation of the distribution of $\mathbf{X}$ into the continuous approximation of the distribution of $\mathbf{Y}$. In order to diagnose the effectiveness of the equating function, we need to consider what this transformation does to the discrete distribution of $\mathbf{X}$. One way of doing this is to compare the moments of the distribution of $\mathbf{X}$ with the moments of the distribution of $\mathbf{Y}$. Following von Davier *et al.* (2004), we use the percent relative error (PRE) in the p-th moments, the PRE$\left(p\right)$, which is defined as

$$\mathrm{PRE}\left(p\right) = 100\frac{\mu_p\left(e_Y\left(\mathbf{X}\right)\right) - \mu_p\left(\mathbf{Y}\right)}{\mu_p\left(\mathbf{Y}\right)}, \tag{21}$$

where $\mu_p\left(e_Y\left(\mathbf{X}\right)\right) = \sum_j \left(e_Y\left(x_j\right)\right)^p r_j$ and $\mu_p\left(\mathbf{Y}\right) = \sum_k \left(y_k\right)^p s_k$.

3. kequate for R

The package **kequate** for R enables the equating of two parallel tests with the kernel method of equating for the EG, SG, CB, NEAT PSE, NEAT CE and NEC designs. The package **kequate** can use '`glm`' objects created using the R function `glm()` (from package **stats**; R Core Team 2013) as input arguments and estimate the equating function and associated standard errors directly from the information contained therein. The S4 system of classes and methods (Chambers 2008), a more formal and rigorous way of handling objects in R, is used in package **kequate**, providing methods for the generic functions `plot()` and `summary()` for a number of newly defined classes. The main function of the package is `kequate()`, which enables the equating of two parallel tests using the previously defined equating designs. The function `kequate()` has the following formal function call: `kequate(design, ...)` where `design` is a character vector indicating the design used and `...` should contain the additional arguments which depend partly on the design chosen. The possible data collection designs and the associated function calls are described below. Explanations of each argument that may be supplied to `kequate()` are collected in Tables 1 and 2.

EG :

```
kequate("EG", x, y, r, s, DMP, DMQ, N, M, hx = 0, hy = 0, hxlin = 0,
  hylin = 0, KPEN = 0, wpen = 1/4, linear = FALSE, irtx = 0, irty = 0,
  smoothed = TRUE, kernel= "gaussian", slog = 1, bunif = 0.5,
  altopt = FALSE)
```

SG :

```
kequate("SG", x, y, P, DM, N, hx = 0, hy = 0, hxlin = 0, hylin = 0,
  KPEN = 0, wpen = 1/4, linear = FALSE, irtx = 0, irty = 0,
  smoothed = TRUE, kernel = "gaussian", slog = 1, bunif = 0.5,
  altopt = FALSE)
```

CB :

```
kequate("CB", x, y, P12, P21, DM12, DM21, N, M, hx = 0, hy = 0,
  hxlin = 0, hylin = 0, wcb = 1/2, KPEN = 0, wpen = 1/4,
  linear = FALSE, irtx = 0, irty = 0, smoothed = TRUE,
  kernel = "gaussian", slog = 1, bunif = 0.5, altopt = FALSE)
```

NEAT CE :

```
kequate("NEAT_CE", x, y, a, P, Q, DMP, DMQ, N, M, hxP = 0, hyQ = 0,
  haP = 0, haQ = 0, hxPlin = 0, hyQlin = 0, haPlin = 0, haQlin = 0,
  KPEN = 0, wpen = 1/4, linear = FALSE, irtx = 0, irty = 0,
  smoothed = TRUE, kernel = "gaussian", slog = 1, bunif = 0.5,
  altopt = FALSE)
```

NEAT PSE :

```
kequate("NEAT_PSE", x, y, P, Q, DMP, DMQ, N, M, w = 0.5, hx = 0,
  hy = 0, hxlin = 0, hylin = 0, KPEN = 0, wpen = 1/4, linear = FALSE,
  irtx = 0, irty = 0, smoothed = TRUE, kernel = "gaussian", slog = 1,
  bunif = 0.5, altopt = FALSE)
```

NEC :

```
kequate("NEC", x, y, P, Q, DMP, DMQ, N, M, hx = 0, hy = 0, hxlin = 0,
  hylin = 0, KPEN = 0, wpen = 1/4, linear = FALSE, irtx = 0, irty = 0,
  smoothed = TRUE, kernel = "gaussian", slog = 1, bunif = 0.5,
  altopt = FALSE)
```

The arguments containing the score probabilities and design matrices that are supplied to **kequate** can either be objects of class 'glm' or design matrices and estimated probability vectors/matrices. For ease of use, it is recommended to estimate the log-linear models using the R function `glm()` and use the 'glm' objects as input to `kequate()`. The estimation of log-linear models using `glm()` is not covered extensively in this article. The interested reader is referred to Holland and Thayer (2000) and R help files. Optional arguments to specify the continuization parameters directly are also available for all equating designs. In addition, the option exists to only conduct a linear equating and an option to use unsmoothed input frequencies. There is also the option of selecting the kernel to be used. In the NEAT PSE case, the weighting of the synthetic populations can be specified. For all designs, if using pre-smoothed input data, the equated values and the SEE are calculated. Using unsmoothed data, the SEE is calculated only in the EG case. The SEED between the linear equating function and the kernel equipercentile equating function is also calculated. For each design there is also the option to use data from an IRT model to conduct an IRT observed-score

Argument(s)	Designs	Description
`x`, `y`	ALL	Score value vectors for test X and test Y.
`a`	CE	Score value vector for the anchor test A.
`r`, `s`	EG	Score probability vectors for tests X and Y. Alternatively objects of class '`glm`'.
`P`	SG, CE, PSE, NEC	Matrix of bivariate score probabilities for tests X and Y (SG), tests X and A (CE, PSE), or test X and covariates (NEC) on population P. Alternatively an object of class '`glm`'.
`Q`	CE, PSE, NEC	Matrix of bivariate score probabilities for tests Y and A (CE, PSE) or test Y and covariates (NEC) on population Q. Alternatively an object of class '`glm`'.
`P12`, `P21`	CB	Matrices of bivariate score probabilities for tests X and Y. Alternatively objects of class '`glm`'.
`DMP, DMQ`	CE, PSE, NEC	Design matrices for the specified bivariate log-linear models on populations P and Q, respectively (or groups taking test X and Y, respectively, in an EG design). Not needed if `P` and `Q` are of class '`glm`'.
`DM`	SG	Design matrix for the specified bivariate log-linear model. Not needed if `P` is of class '`glm`'.
`DM12`, `DM21`	CB	Design matrices for the specified bivariate log-linear models. Not needed if `P12` and `P21` are of class '`glm`'.
`N`	ALL	The sample size for population P (or the group taking test X in the EG design). Not needed if `r`, `P`, or `P12` is of class '`glm`'.
`M`	EG, CB, CE, PSE, NEC	The sample size for population Q (or the group taking test Y in the EG design). Not needed if `s`, `Q`, or `P21` is of class '`glm`'.
`w`	PSE	Optional argument to specify the weight given to population P. Default is 0.5.
`hx`, `hy`, `hxlin`, `hylin`	EG, SG, CB, PSE, NEC	Optional arguments to specify the continuization parameters manually.
`hxP`, `hyQ`, `haP`, `haQ`, `hxPlin`, `hyQlin`, `haPlin`, `haQlin`	CE	Optional arguments to specify the continuization parameters manually.
`wcb`	CB	The weighting of the two test groups in a counterbalanced design. Default is 1/2.

Table 1: Arguments supplied to `kequate()`.

equating using the kernel equating framework. This is accomplished by supplying matrices of probabilities to answer each question correctly for each ability level on two parallel tests X and Y, as estimated beforehand using an IRT model.

The package **kequate** creates an object of class '`keout`' which includes information about the equating. To access information from such an object, a number of `get*`-functions are available. They are described in Table 3. Methods for the class '`keout`' are implemented

Argument(s)	Designs	Description
`KPEN`	ALL	Optional argument to specify the constant used in deciding the optimal continuization parameter. Default is 0.
`wpen`	ALL	An argument denoting at which point the derivatives in the second part of the penalty function should be evaluated. Default is 1/4.
`linear`	ALL	Logical denoting if a linear equating only is to be performed. Default is `FALSE`.
`irtx, irty`	ALL	Optional arguments to provide matrices of probabilities to answer correctly to the questions on the parallel tests X and Y, as estimated in an IRT model.
`smoothed`	ALL	A logical argument denoting if the data provided are pre-smoothed or not. Default is `TRUE`.
`kernel`	ALL	A character vector denoting which kernel to use, with options `"gaussian"`, `"logistic"`, `"stdgaussian"` and `"uniform"`. Default is `"gaussian"`.
`slog`	ALL	The parameter used in the logistic kernel. Default is 1.
`bunif`	ALL	The parameter used in the uniform kernel. Default is 0.5.
`altopt`	ALL	Logical which sets the bandwidth parameter equal to a variant of Silverman's rule of thumb. Default is `FALSE`.

Table 2: Arguments supplied to `kequate()` (Continued.)

Function	Output
`getEquating()`	A data frame with the equated values, SEEs and other information about the equating.
`getPre()`	A data frame with the PRE for the equated distribution.
`getType()`	A character vector describing the type of equating conducted.
`getScores()`	A list containing data frames with the score values, score probabilities and the continuized distribution functions for the tests used in the equating.
`getH()`	A data frame containing the values of h used in the equating.
`getEq()`	A vector containing the equated values.
`getEqlin()`	A vector containing the equated values of the linear equating.
`getSeelin()`	A vector containing the SEEs for the equated values of the linear equating.
`getSeed()`	An object of class '`genseed`' containing the SEED between the KE-equipercentile equating and the linear equating (if applicable).

Table 3: Functions to retrieve information from the resulting '`keout`' objects.

for the functions `plot()` and `summary()`. Additionally, the function `genseed()` can be used to compare any two equatings that utilize the same log-linear models. It takes as input two objects created by **kequate** and calculates the SEED between them. A useful comparison is, for example, between a chain equating and a post-stratification equating in the NEAT design. A method for the function `plot()` is implemented for the objects created by `genseed()`. The package also includes a function `kefreq()` to tabulate frequency data from individual test score data and functions `FTres()` and `cdist()` to be used when specifying the log-linear pre-smoothing models. `FTres()` calculates the Freeman-Tukey residuals given a specified log-linear model, and `cdist()` calculates the conditional means, variances, skewnesses and kurtoses of the tests to be equated given an anchor test, for both the fitted distributions and the observed distributions.

4. Examples

We exemplify the main function `kequate()` by equating using the EG, NEAT, and NEC designs. The function calls for the other designs are very similar, only having other required arguments. We also demonstrate how to conduct an IRT observed-score equating.

4.1. EG design

Let the parallel tests X and Y have common score vectors `0:20`. The tests are each administered to a randomized group drawn from the same population, thus we have an EG design. The data used in this example is from Chapter 7 of von Davier *et al.* (2004) and we have specified identical log-linear models to the book using the `glm()` function in R. Thus, two objects `FXEGglm` and `FYEGglm` have been created. We give the summary of the log-linear model for test X below.

```
R> summary(FXEGglm)


Call:
glm(formula = freq ~ I(X) + I(X^2), family = poisson, data = FXEG,
    x = TRUE)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.93720  -0.66091   0.02267   0.28793   2.06210

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.193832   0.159873   7.467 8.18e-14 ***
I(X)         0.700114   0.030547  22.919  < 2e-16 ***
I(X^2)      -0.032194   0.001391 -23.140  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 881.023  on 20  degrees of freedom
Residual deviance:  18.353  on 18  degrees of freedom
AIC: 140.98

Number of Fisher Scoring iterations: 4
```

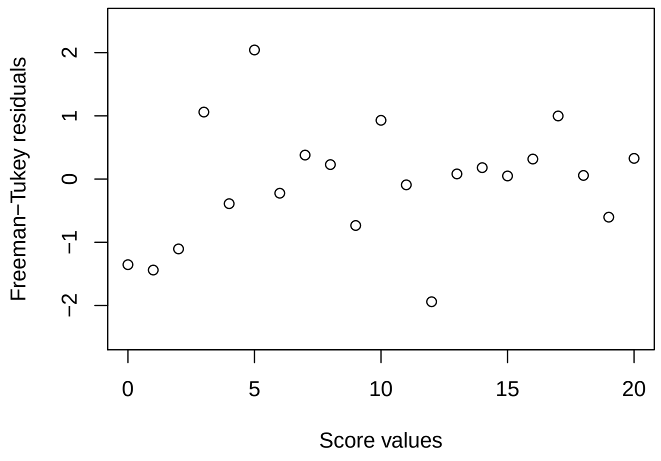

Figure 1: The Freeman-Tukey residuals for the log-linear model of test X in the EG design.

In **kequate** the function `FTres()` can be used to calculate the Freeman-Tukey residuals often considered in pre-smoothing. From our log-linear model for test X, we create a numeric vector containing the Freeman-Tukey residuals by writing:

```
R> Xres <- FTres(FXEGglm$y, FXEGglm$fitted.values)
```

We plot the resulting vector which can be seen in Figure 1. If the model fits the data well, the Freeman-Tukey residuals are approximately normal distributed, which is tenable in this case. To then equate the two tests using an equipercentile equating with pre-smoothing, we call the function `kequate()` as follows:

```
R> keEG <- kequate("EG", 0:20, 0:20, FXEGglm, FYEGglm)
```

This will create an R object `keEG` containing information about the equating, retrieved by using the functions described in Table 3. To print useful information about the equating, we can utilize the `summary()` function. With the EG example above, we write:

```
R> summary(keEG)
```

```
 Design: EG equipercentile

 Kernel: gaussian

 Sample Sizes:
   Test X: 1453
   Test Y: 1455

 Score Ranges:
   Test X:
       Min = 0 Max = 20
   Test Y:
       Min = 0 Max = 20

 Bandwidths Used:
         hx        hy    hxlin    hylin
1 0.6222656 0.5706472 3807.166 3935.618

 Equating Function and Standard Errors:
   Score       eqYx      SEEYx
1      0  0.3937428 0.22003933
2      1  1.5813115 0.28953052
3      2  2.6403737 0.28750504
4      3  3.6443599 0.26639407
5      4  4.6316372 0.24103576
6      5  5.6177604 0.21694955
7      6  6.6099736 0.19666327
8      7  7.6120208 0.18124172
9      8  8.6259786 0.17074911
10     9  9.6529836 0.16457143
11    10 10.6934738 0.16187097
12    11 11.7471382 0.16210070
13    12 12.8126160 0.16533581
14    13 13.8868792 0.17213304
15    14 14.9641247 0.18265200
16    15 16.0338852 0.19504913
17    16 17.0781109 0.20375800
18    17 18.0676527 0.19900332
19    18 18.9607427 0.16999805
20    19 19.7183057 0.11860300
21    20 20.3929906 0.07030467

 Comparing the Moments:
         PREYx
1  0.005846071
2  0.012199064
3  0.021866777
4  0.039664185
5  0.072721834
6  0.127379417
7  0.208808230
8  0.320998175
9  0.466899016
10 0.648610889
```

The `summary()` function can be used in **kequate** to print information from any object of class '`keout`'. The output is similar for all designs. The first part contains information about the score range and bandwidths. The second part contains the equating function with its standard error. Finally, the PRE is given.

With the EG design, it is also possible to equate two tests using the full kernel equating framework with observed data instead of pre-smoothed data. The additional argument `smoothed = FALSE` needs to be given to `kequate()` in such a case. As an example, by using information from the object created in the equating with pre-smoothing, we can write:

```
R> rEGobs <- getScores(keEG)$X$r
R> sEGobs <- getScores(keEG)$Y$s
R> NEG <- getScores(keEG)$N
R> MEG <- getScores(keEG)$M
R> keEGobs <- kequate("EG", 0:20, 0:20, rEGobs, sEGobs, N = NEG, M = MEG,
+    smoothed = FALSE)
```

The object created contains similar information to an object from an equating with pre-smoothed data.

IRT observed-score equating (IRT-OSE) is also enabled in **kequate**, using the arguments `irtx` and `irty`. We let `irt1` and `irt2` be matrices where each column represents an ability level in an IRT model and each row represents a question on the test to be equated. Each cell in the matrix should then contain the estimated probability to answer correctly to a question on the parallel tests for a certain ability level. To equate using IRT-OSE, we write:

```
R> keEGirt <- kequate("EG", 0:20, 0:20, FXEGglm, FYEGglm, irtx = irt2,
+    irty = irt1)
```

This function call will conduct an IRT-OSE in the kernel equating framework in addition to a regular equipercentile equating. It is possible to use unsmoothed frequencies while conducting an IRT-OSE. Specifying `linear = TRUE` will instruct `kequate()` to do a linear equating for both the regular method and for the IRT-OSE. Using IRT-OSE is not limited to an EG design. It can be used as a supplement in any of the designs available in **kequate**.

4.2. NEAT design

To illustrate how to do an equating in a NEAT design, we utilize a simulated data set provided with **kequate**. With the same data we also show how the function `kefreq()` can be used to tabulate data from individual test takers and how the function `cdist()` can be used to evaluate a log-linear model. In this example test X and test Y have the same score value vectors `0:20`, and test A (the anchor test) has the score value vector `0:10`. We now wish to equate tests X and Y. The R object `bivar1` is a data frame with columns `X` and `A` of length 1000 containing the score on test X and test A for each individual. Similarly, `bivar2` is a data frame with columns `Y` and `A` of length 1000. For all designs that utilize bivariate frequencies, the data must be sorted first by the score vector for A and then by the score vector for X. In the SG design, the data must be sorted first by the score vector for test Y and then by the score vector for test X. To create data sorted in the manner appropriate for usage with **kequate**, we write:

```
R> freq1 <- kefreq(bivar1$X, 0:20, bivar1$A, 0:10)
R> freq2 <- kefreq(bivar2$Y, 0:20, bivar2$A, 0:10)
```

The created objects `freq1` and `freq2` are data frames with three columns: `frequency`, `X`, and `A`, sorted first by the `A` column and then by the `X` column. The data frame created by `kefreq()` can then be used in the `glm()` model specification. Alternatively, the `frequency` column can be converted into relative frequencies and utilized to equate tests using observed relative frequencies directly. In this example we use pre-smoothing and assume that the '`glm`' objects `glmsim1` and `glmsim2` have been created. The summary of `glmsim1` is given below.

```
R> summary(glmsim1)

Call:
glm(formula = frequency ~ I(X) + I(X^2) + I(X^3) + I(X^4) + I(X^5) +
    I(A) + I(A^2) + I(A^3) + I(A^4) + I(A):I(X) + I(A):I(X^2) +
    I(A^2):I(X) + I(A^2):I(X^2), family = "poisson", data = freq1,
    x = TRUE)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0435  -0.6671  -0.1474   0.2255   2.7034

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.558e+00  3.088e-01   5.044 4.56e-07 ***
I(X)          -3.753e-01  2.384e-01  -1.575  0.11537
I(X^2)         1.905e-01  7.193e-02   2.648  0.00810 **
I(X^3)        -2.775e-02  8.943e-03  -3.103  0.00192 **
I(X^4)         1.298e-03  4.869e-04   2.666  0.00768 **
I(X^5)        -2.208e-05  9.685e-06  -2.280  0.02262 *
I(A)          -4.172e-01  2.282e-01  -1.828  0.06756 .
I(A^2)        -1.988e-02  7.995e-02  -0.249  0.80363
I(A^3)        -1.452e-02  1.319e-02  -1.101  0.27106
I(A^4)         8.691e-04  6.481e-04   1.341  0.17992
I(X):I(A)      1.526e-01  5.315e-02   2.870  0.00410 **
I(X^2):I(A)    7.769e-04  3.511e-03   0.221  0.82489
I(X):I(A^2)   -4.951e-03  6.362e-03  -0.778  0.43642
I(X^2):I(A^2)  1.103e-04  2.656e-04   0.415  0.67807
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 1550.4  on 230  degrees of freedom
Residual deviance:  169.0  on 217  degrees of freedom
AIC: 689.32

Number of Fisher Scoring iterations: 6
```

We fitted the model using five univariate moments for the score values of the test to be equated and four moments for the score values of the anchor test. The first four cross-moments between the test scores were also added. Adding additional parameters did not improve the model fit much. To evaluate a log-linear model for bivariate test score frequencies, it is a good idea to compare the conditional mean, variance, skewness and kurtosis of the observed and fitted frequencies. It is desirable to maintain the properties of the observed frequencies in the specified model. In **kequate** the function `cdist()` can be used to calculate the conditional parameters of observed and fitted frequencies. The input to `cdist()` are two matrices of bivariate frequencies (one for the fitted and one for the observed frequencies) and the two score value vectors. To specify the necessary input for tests X and A in our example, we write:

```
R> EGPest <- matrix(glmsim1$fitted.values, nrow = 21)
R> EGPobs <- matrix(glmsim1$y, nrow = 21)
```

We then use these objects and the score value vectors to calculate the conditional parameters and plot the result:

```
R> NEATdistP <- cdist(EGPest, EGPobs, 0:20, 0:10)
R> plot(NEATdistP)
```

The resulting object `NEATdistP` contains all the conditional parameters, but the plot shows only the conditional mean and variance of the respective tests and distributions. The plot is given in Figure 2, where it can be seen that the conditional mean is very close between the observed and fitted distributions but that the conditional variance is not as well maintained in the fitted distribution. To use the log-linear models specified above to equate the two tests in a NEAT PSE design and also display the summary, we write:

```
R> eqNEATPSE <- kequate("NEAT_PSE", 0:20, 0:20, glmsim1, glmsim2)
R> summary(eqNEATPSE)
```

```
 Design: NEAT/NEC PSE equipercentile

 Kernel: gaussian

 Sample Sizes:
   Test X: 1000
   Test Y: 1000

 Score Ranges:
   Test X:
       Min = 0 Max = 20
   Test Y:
       Min = 0 Max = 20

 Bandwidths Used:
          hx        hy    hxlin    hylin
1 0.5565059 0.5965906 4174.636 3893.065
```

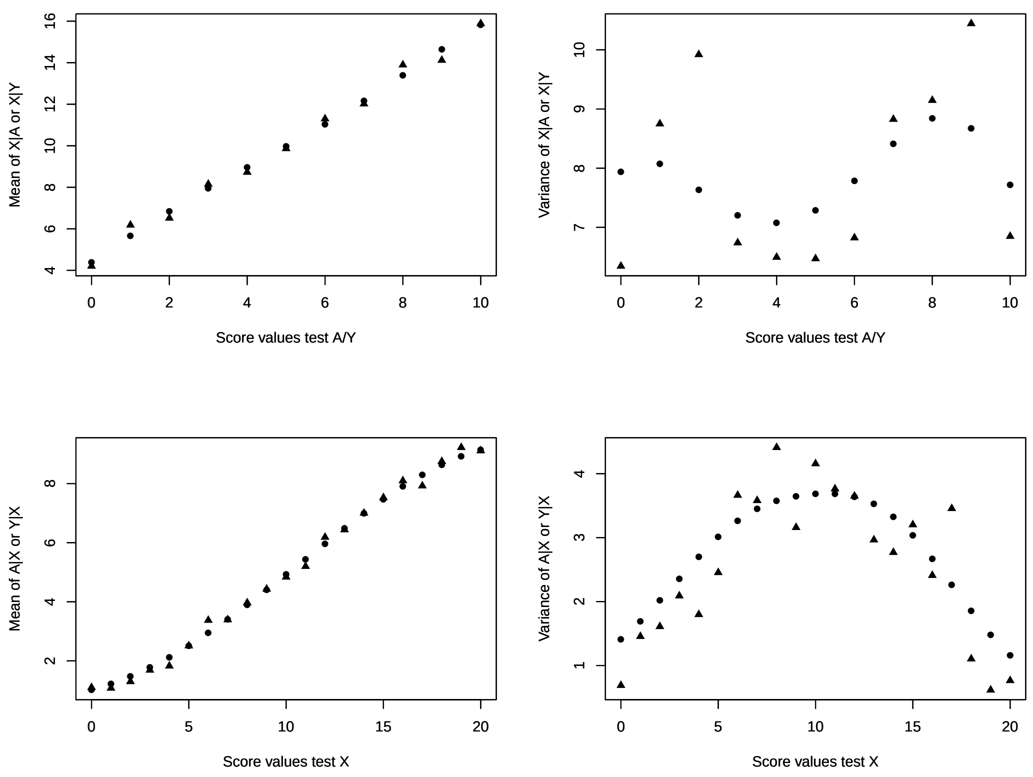

Figure 2: Conditional mean and variance of tests X and A for the observed and fitted frequencies in the NEAT case.

```
 Equating Function and Standard Errors:
   Score        eqYx     SEEYx
1      0  0.09205346 0.1942502
2      1  1.00529599 0.2936029
3      2  1.71066883 0.3000529
4      3  2.40382569 0.2784435
5      4  3.16821131 0.2484674
6      5  4.02457837 0.2170684
7      6  4.95676115 0.1879430
8      7  5.93318013 0.1647129
9      8  6.92349260 0.1493227
10     9  7.90577157 0.1410529
11    10  8.86675778 0.1379823
12    11  9.80020969 0.1391913
13    12 10.70586827 0.1455433
14    13 11.58932914 0.1582699
15    14 12.46220140 0.1769702
16    15 13.34185601 0.1999658
17    16 14.25090471 0.2274527
```

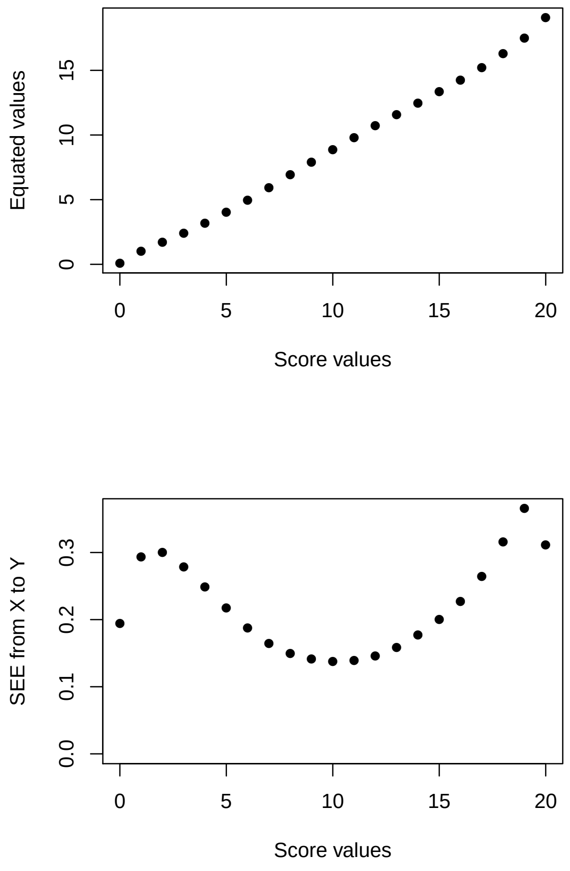

Figure 3: The equated values and corresponding SEE for each score value in a NEAT PSE design.

```
18     17 15.21769285 0.2642756
19     18 16.28082502 0.3155014
20     19 17.50960932 0.3654120
21     20 19.06346403 0.3116301

 Comparing the Moments:
          PREYx
1  -0.001950272
2  -0.030409301
3  -0.104245343
4  -0.242660855
5  -0.471507678
6  -0.814142404
7  -1.287683242
8  -1.902544819
9  -2.663200823
10 -3.569333597
```

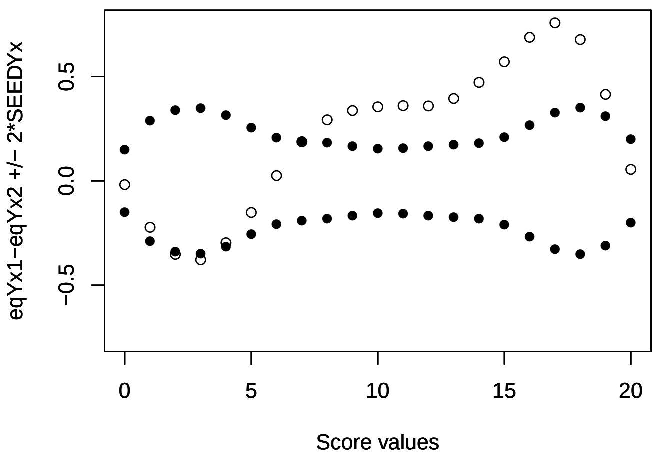

Figure 4: The difference between PSE and CE in a NEAT design for each score value with the associated SEED.

The results can also be plotted by writing:

```
R> plot(eqNEATPSE)
```

The resulting graph can be seen in Figure 3, where the first plot compares the score values on X with the equated values, and where the second plot gives the standard error of the equated values for each score value of X. The same type of graph is plotted for all equating designs. Chain equating can also be used in **kequate**. To equate the same tests as in the NEAT case above but this time using CE, we write:

```
R> eqNEATCE <- kequate("NEAT_CE", 0:20, 0:20, 0:10, glmsim1, glmsim2)
```

Given two different equating functions derived from the same log-linear models, the SEED between two equatings can be calculated. In **kequate**, the function `genseed()` takes as input two objects of class ‘`keout`’ and calculates the SEED between two kernel equipercentile or linear equatings. By default the kernel equipercentile equatings are used. To instead compare two linear equatings to each other, the logical argument `linear = TRUE` should be used when calling `genseed()`. The output from `genseed()` is an object of class ‘`genseed`’ which can be plotted using `plot()`, creating a suitable plot of the difference between the equating functions and the associated SEED. To compare the NEAT PSE and NEAT CE equatings given above and plot the results, we write:

```
R> seedPSECE <- genseed(eqNEATPSE, eqNEATCE)
R> plot(seedPSECE)
```

The resulting figure can be seen in Figure 4. The difference between the equating functions is outside of the error bands for many score values, indicating that the equatings significantly differ from each other. In the above function calls, the default settings have been used. Under the default settings, both a KE-equipercentile equating and a linear equating are done. The continuization parameters will by default be set to the optimal value in the KE-equipercentile case and to $1000 \cdot$ std error for the test scores in the linear case. It is possible to choose these parameters manually by specifying additional arguments in the function call. With a NEAT PSE design there are four continuization parameters to consider: `hx`, `hy`, `hxlin`, and `hylin`. As an example, we can write:

```
R> eqNEATPSEh1 <- kequate("NEAT_PSE", 0:20, 0:20, glmsim1, glmsim2, hx = 1,
+    hy = 1, hxlin = 1000, hylin = 1000)
```

4.3. NEC design

In the NEC design, instead of using an anchor test to enable the equating of two tests when the groups taking the test are not equivalent, we utilize background information on the individuals taking the tests. In the example used here, an equating is made of two instances of a part of the Swedish Scholastic Assessment Test (variable names `testX` and `testY`) with the aid of covariates indicating high school math grade (variable name `mattea1`) and type of high school education (`utb1`). Using the function `glm()` in R, the objects `NECYglm` and `NECXglm` for each test have been specified.

```
R> summary(NECYglm)

Call:
glm(formula = frequency ~ I(testY) + I(testY^2) + I(testY^3) +
    I(mattea1) + I(mattea1^2) + factor(utb1) + I(testY):I(mattea1) +
    I(testY):factor(utb1) + I(mattea1):factor(utb1), family = "poisson",
    data = testdata2, x = TRUE)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.4379  -0.9497  -0.0727   1.0324   4.3802

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              3.0157230  0.1107351  27.234  < 2e-16 ***
I(testY)                 0.1918915  0.0227800   8.424  < 2e-16 ***
I(testY^2)              -0.0219865  0.0019919 -11.038  < 2e-16 ***
I(testY^3)              -0.0005993  0.0000540 -11.098  < 2e-16 ***
I(mattea1)               0.7847179  0.0761327  10.307  < 2e-16 ***
I(mattea1^2)            -1.2017467  0.0189962 -63.262  < 2e-16 ***
factor(utb1)2            0.3269224  0.0866072   3.775  0.00016 ***
factor(utb1)3           -4.5725554  0.1125338 -40.633  < 2e-16 ***
I(testY):I(mattea1)      0.2636119  0.0043871  60.088  < 2e-16 ***
I(testY):factor(utb1)2   0.1836241  0.0082557  22.242  < 2e-16 ***
I(testY):factor(utb1)3   0.4452775  0.0096066  46.351  < 2e-16 ***
I(mattea1):factor(utb1)2 -0.2730027  0.0524782  -5.202 1.97e-07 ***
I(mattea1):factor(utb1)3  0.2923510  0.0557851   5.241 1.60e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 36242.64  on 206  degrees of freedom
Residual deviance:   412.48  on 194  degrees of freedom
AIC: 1379.5

Number of Fisher Scoring iterations: 5
```

We included three univariate moments for the test scores and two moments for the grade in mathematics. Additionally, interaction terms were added between the test scores and the covariates and also between the covariates. The resulting model fits the data well. A similar model was fitted for the scores of the other test administration. We now equate the two versions of the test by writing:

```
R> NECeq <- kequate("NEC", 0:22, 0:22, NECYglm, NECXglm)
```

The results from the `summary()` function are given below, showing that the two tests are fairly equal in difficulty when we have conditioned on relevant background variables. For lower scores it appears that test X is slightly more difficult, while at the higher end test Y is slightly more difficult. Due to the large sample sizes for the two tests, the estimated standard errors are small and our equating is quite reliable.

```
R> summary(NECeq)

 Design: NEAT/NEC PSE equipercentile

 Kernel: gaussian

 Sample Sizes:
   Test X: 19587
   Test Y: 16607

 Score Ranges:
   Test X:
       Min = 0 Max = 22
   Test Y:
       Min = 0 Max = 22

 Bandwidths Used:
         hx      hy    hxlin    hylin
1 0.5749784 0.58995 4306.685 4116.669

 Equating Function and Standard Errors:
   Score         eqYx      SEEYx
1      0  0.09485327 0.07349573
2      1  1.17900980 0.12298033
```

```
3       2  2.24120888 0.13666122
4       3  3.28252507 0.12961135
5       4  4.30832447 0.11459671
6       5  5.32114109 0.09779945
7       6  6.32204026 0.08225010
8       7  7.31137161 0.06936471
9       8  8.28922787 0.05960339
10      9  9.25578121 0.05274700
11     10 10.21156497 0.04811199
12     11 11.15769936 0.04485671
13     12 12.09602853 0.04229809
14     13 13.02913160 0.04009804
15     14 13.96019747 0.03830472
16     15 14.89279942 0.03729549
17     16 15.83064933 0.03760529
18     17 16.77744941 0.03959559
19     18 17.73701331 0.04307497
20     19 18.71394777 0.04705997
21     20 19.71553704 0.04948802
22     21 20.75659872 0.04608867
23     22 21.85710257 0.02953481

 Comparing the Moments:
          PREYx
1  -0.002719536
2  -0.007614830
3  -0.013701588
4  -0.024243657
5  -0.042056328
6  -0.069436523
7  -0.108144263
8  -0.159468379
9  -0.224306751
10 -0.303241828
```

In addition to the default gaussian kernel **kequate** enables the usage of logistic and uniform kernels. To utilize a different kernel the argument `kernel` is specified in the `kequate()` function call. Below, the previously defined log-linear models are used to equate the two tests in the NEC design using a logistic and a uniform kernel.

```
R> NECeqL <- kequate("NEC", 0:22, 0:22, NECYglm, NECXglm,
+    kernel = "logistic")
R> NECeqU <- kequate("NEC", 0:22, 0:22, NECYglm, NECXglm,
+    kernel = "uniform")
```

In this case the equating function is almost identical between the three kernels, but there are some slight differences in the standard error of equating, which can be seen in Figure 5.

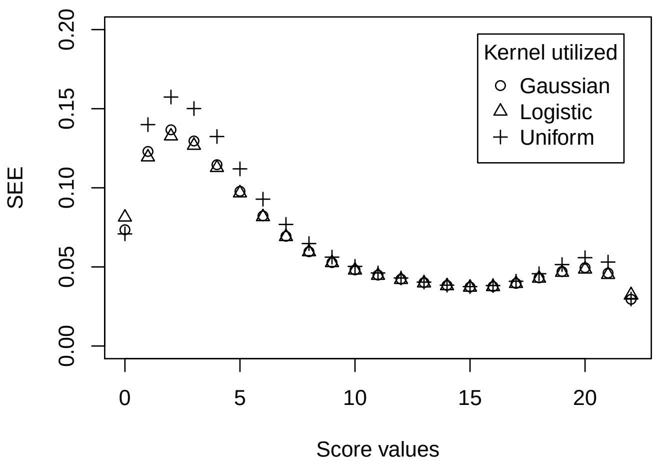

Figure 5: SEE for gaussian, logistic and uniform kernels in the NEC design.

For all designs it is also possible to specify the constants `KPEN` and `wpen` used in finding the optimal continuization parameters. Defaults are `KPEN = 0` and `wpen = 1/4`. Additionally, the logical argument `linear` can be used to specify that only a linear equating is to be performed, where the default is `linear = FALSE`.

5. Concluding remarks

In standardized achievement tests the most essential part is for all tests to be fair to test takers and between test takers. Since standardized tests are typically given at different time periods and with different test forms, it is essential to make sure that which test a test taker is given does not affect his or her results. In this paper the R package **kequate** was proposed in order to implement and to make available the kernel method of test equating. This method can be used by researchers, the testing industry, and practitioners, i.e., anyone with an interest in equating. In addition the package includes a new extension of the kernel method when we have collateral information. Finally, IRT observed-score equating was added to allow for comparisons with a well-known frequently used equating method. In the future the kernel method of test equating might be extended to incorporate more methods, and in those cases it will be easy to implement new methods in this package.

References

Andersson B, Bränberg K, Wiberg M (2013). ***kequate**: The Kernel Method of Test Equating.* R package version 1.3.2, URL http://CRAN.R-project.org/package=kequate.

Angoff WH (1984). *Scales, Norms and Equivalent Scores.* Educational Testing Service, Princeton. (Reprinted from Thorndike RL (Ed.) *Educational Measurement*, 1971).

Bränberg K (2010). *Observed Score Equating with Covariates.* Statistical Studies No. 41. Umeå University, Department of Statistics, Umeå.

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

Chambers J (2008). *Software for Data Analysis: Programming with R*. Springer-Verlag, New York.

Fairbank BA (1987). "The Use of Presmoothing and Postsmoothing to Increase the Precision of Equipercentile Equating." *Applied Psychological Measurement*, **11**(3), 245–262.

Holland PW, King BF, Thayer DT (1989). "The Standard Error of Equating for the Kernel Method of Equating Score Distributions." *Technical Report 89-83*, Educational Testing Service, Princeton.

Holland PW, Thayer DT (1989). "The Kernel Method of Equating Score Distributions." *Technical Report 89-84*, 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.

Jarjoura D, Kolen MJ (1985). "Standard Errors of Equipercentile Equating For the Common Item Nonequivalent Population Design." *Journal of Educational Statistics*, **10**(2), 142–160.

Kolen MJ, Brennan RJ (2004). *Test Equating: Methods and Practices*. 2nd edition. Springer-Verlag, New York.

Lee YH, von Davier AA (2011). "Equating Through Alternative Kernels." In AA von Davier (ed.), *Statistical Models for Test Equating, Scaling, and Linking*. Springer-Verlag, New York.

Lord FM (1982). "The Standard Error of Equipercentile Equating." *Journal of Educational Statistics*, **7**(3), 165–174.

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/`.

Venables WN, Smith DM, R Core Team (2013). *An Introduction to R*. Vienna, Austria. URL `http://www.R-project.org/`.

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

Affiliation:
Björn Andersson
Department of Statistics
Uppsala University, Box 513
SE-751 20 Uppsala, Sweden
E-mail: `bjorn.andersson@statistik.uu.se`
URL: `http://katalog.uu.se/empInfo?id=N11-1505`

Kenny Bränberg, Marie Wiberg
Department of Statistics, USBE
Umeå University
SE-901 87 Umeå, Sweden
E-mail: `kenny.branberg@stat.umu.se`, `marie.wiberg@stat.umu.se`
URL: `http://www.usbe.umu.se/om-handelshogskolan/personal/kebr0001`
`http://www.usbe.umu.se/om-handelshogskolan/personal/maewig95`

Journal of Statistical Software `http://www.jstatsoft.org/`
published by the American Statistical Association `http://www.amstat.org/`
Volume 55, Issue 6 *Submitted:* 2012-03-02
October 2013 *Accepted:* 2013-02-13