Test Equating Using the Kernel Method with the R Package kequate

Björn Andersson
Uppsala University

Kenny Bränberg
Umeå University

Marie Wiberg
Umeå University

Abstract

In standardized testing the equating of tests is important in order to ensure fairness for test-takers. Recently, the kernel method of test equating has gained popularity. The kernel method of test equating comprises 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. We present the software package **kequate** for R. **kequate** implements the kernel method of test equating 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. For all designs, it is possible to conduct an item-response theory observed score equating as a supplement. 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` are also included.

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

1. Introduction

When standardized achievement tests are used the main concern is that it they are fair to the individual test takers and between current and former test takers. In order to ensure fairness when a test is given at different time points or when different versions of the same standardized test are given, a statistical procedure known as equating is used. Equating is a statistical process which is used to adjust scores on different test forms so that the test forms can be used interchangeably (Kolen and Brennan 2004). There are five important equating requirements which need to be satisfied in order for a function to be called an equating. See e.g. von Davier, Holland, and Thayer (2004), Lord (1980) and Kolen and Brennan (2004). First, the equal construct requirement, which means that only tests which measure the same construct should be equated. Second, the equal reliability requirement, meaning that the tests need to be of equal reliability in order to be equated. Third, the symmetry requirement which requires the equating transformations to be symmetrical. Fourth, the equity requirement, which means that it should be a matter of indifference to each test taker whether test form X or test form Y is administered. Fifth, the population invariance requirement, which means that the equating should be the same regardless of the group of test takers on which the equating was performed. There exist many equating methods which to the most extent satisfy these requirements. In this guide we will concentrate on observed-score equating, and more specifically on the observed-score kernel method of test equating which fulfill these requirements (von Davier *et al.* 2004).

The kernel method of test equating (von Davier *et al.* 2004) is an observed-score test equating method comprising five steps: pre-smoothing, score probability estimation, continuization, computation of the equating function and computation of the standard errors of the equating function. The kernel equating 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 five 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 this is a unified equating framework which has a large applicability both for the testing industry, the research community and practitioners it is of high interest to create software that anyone with an interest in equating can use. This manual introduces and exemplifies the 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 Development Core Team 2013), and is structured as follows. In Section 2 the kernel equating framework is introduced. Section 3 contains a description of how to aggregate and sort data on the individual level and how to estimate log-linear models with the R function `glm()`(**stats** R Development Core Team 2013). We give examples for all the included equating designs and instruct how to decide between different model specifications using tools provided by **kequate**. The package **kequate** is described in Section 4 and in Section 5 examples of equating using **kequate** for all equating designs are given.

2. Theoretical background

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 used in this study. The first four are standard data collection designs (see, e.g., Kolen and Brennan, 2004, or 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. 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

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

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

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

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 (cdf's) of $\mathbf{X}$ and $\mathbf{Y}$ in the target population. Let

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

and

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

be the cdf's of $\mathbf{X}$ and $\mathbf{Y}$ over the target population T. If the two cdf's are continuous and strictly increasing the equipercentile equating function of $\mathbf{X}$ to $\mathbf{Y}$ is defined by

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

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 (SEE) and the standard error of equating difference (SEED).

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 proportions obtained from the raw data. We will show this for the NEAT design. For details the interested reader is refered to, i.e., 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 $\mathbf{n}_{XA} = (n_{X11}, \ldots, n_{XJL})^t$ and $\mathbf{n}_{YA} = (n_{Y11}, \ldots, n_{YKL})$ 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 $p_{jl} = P(\mathbf{X} = x_j, \mathbf{A} = a_l \mid T)$. The target population T is a mixture of the two populations P and Q, $T = wP + (1-w)Q$, where $0 \leq w \leq 1$.

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

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

The log likelihood function for $\mathbf{Y}$ and the log-linear model for $q_{kl} = P(\mathbf{Y} = y_k, \mathbf{A} = a_l \mid T)$ 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 $\mathbf{r} = (r_1, r_2, \ldots, r_J)^t$ and $\mathbf{s} = (s_1, s_2, \ldots, s_K)$. 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 $\mathbf{p}_l = (p_{1l}, p_{2l}, \ldots, p_{Jl})^t$ and $\mathbf{q}_l = (q_{1l}, q_{2l}, \ldots, q_{Kl})^t$.

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. Prior to the development of kernel equating, linear interpolation was usually used to obtain continuous cdf's from the discrete cdf's (Kolen and Brennan, 2004). In kernel equating continuous cdf's are used as approximations to the estimated discrete step-function cdf's 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 **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

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

where $\hat{f}_{h_X}(x)$ is the estimated density function, i.e., the derivative of $\hat{F}_{h_X}(x)$ 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 right 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 $\mathbf{PEN}(h_X)$ in Equation 11 the estimated continuous density function $\hat{f}_{h_X}(x)$ 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 **X** to **Y**. If we use the continuized cdf's described previously we can define the kernel equating function as

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

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

Step 5: Calculating the standard error of equating (SEE) and the standard error of equating difference (SEED)

One of the advantages with the kernel method of test equating is that it provides a neat way to compute the standard error of equating (SEE). The SEE for equating **X** to **Y** is given by

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

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

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

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 **r** and **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 **r** and **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

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

and

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

where $\|\upsilon\|$ denotes the Euclidian norm of vector υ.
The standard error of equating difference (SEED), which can be used to compare different kernel equating functions, is defined as

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

i.e., the Euclidian norm of the difference between the two vectors $\mathbf{J}_{e_1}\mathbf{J}_{DF}\mathbf{C}$ and $\mathbf{J}_{e_2}\mathbf{J}_{DF}\mathbf{C}$. 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 in the p^{th} moments, the $\mathrm{PRE}(p)$, which is defined as

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

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. Pre-smoothing using R

Before the actual equating can begin, the raw score data usually needs to be processed via a step called pre-smoothing. In this step, the distribution of the score probabilities is estimated using a log-linear model. The main function of **kequate** has been designed to be used in conjunction with the R function `glm()`. The input to **kequate** thus preferably consists of objects created by `glm()` but the option exists to provide vectors and matrices containing estimated probabilities and design matrices from the log-linear model specification. There is also the option of using observed proportions. In this section, we first describe a method for converting data from the individual level into frequencies of score values for the population as a whole and then we describe and exemplify the estimation of the log-linear models for each data design. Lastly we discuss how the model fit for the specified log-linear models can be assessed.

3.1. Aggregating and sorting the data

Test data often consists of data at the individual level, i.e. there is a data frame, matrix or vector containing the score for each individual taking the test along with other possible information about this individual such as covariates or the score on an anchor test. In order to use such data in equating, the data needs to be converted into frequencies for each combination of score values or score value and covariate values. **kequate** contains the function `kefreq()` which can handle univariate and bivariate data. `kefreq()` has the following function call:

```
kefreq(in1, xscores, in2, ascores)
```

`in1` is the individual data for test X, `xscores` is the vector of possible scores for test X, `in2` is the individual data for the parallel test Y or anchor test A (if applicable) and `ascores` is the vector of possible scores for the anchor test A. If the interest lies only in retrieving the score frequencies of a single test with possible scores from integers 0 to 20 and the data is in a vector `simeq$bivar1$X`, the frequencies are obtained by writing

```
R> freq <- kefreq(simeq$bivar1$X, 0:20)
```

This will create a data frame with two vectors: `freq$X` denoting the score values and `freq$frequency` containing the frequencies for the particular score values. When equating using the equivalent groups (EG) design, the frequencies for each of the tests can be obtained in this manner. For a single group (SG) or non-equivalent groups with anchor test (NEAT) design, we need to consider scores from two separate tests for each individual and compute the frequency for each combination of score values. In a NEAT design, for parallel tests X and Y with score values from 0 to 20 and an anchor test A with score values 0 to 10, we want two vectors of size 21*11=231. In this case, we assume that the data on the individual level from population P is in a data frame with vectors `X` and `A` containing the score of each test for every individual. By writing

```
R> SGfreq <- kefreq(simeq$bivar1$X, 0:20, simeq$bivar1$A, 0:10)
```

we retrieve the frequencies for each combination of scores on tests X and A, in a data frame ordered first by the score on test A and then by the score on test X. The observed frequencies retrieved by `kefreq()` can now be used to estimate the smoothed score distributions using log-linear models as described in the following.

3.2. Estimating the score probabilities with the R function `glm()`

The procedure for estimating the score probability distributions differs between the equating designs and the different procedures are discussed separately in what follows, exemplifying the estimation methods with the R function `glm()`. The proportions follow a multinomial distribution in theory, but this is equivalent to the frequencies being Poisson-distributed, given the sum of the frequencies. So after dividing the resulting fitted values with the sum of the frequencies, we will retrieve the same estimates as when having modelled the proportions directly. Since modelling Poisson data is straight forward in R, we model the frequencies themselves rather than the proportions.

Equivalent groups design

In an EG design, two groups that have been randomly selected from a common population are given separate but parallel tests. The resulting estimated frequencies can then be used to equate the two tests used. In selecting the univariate model for each of the two groups, the statistical criterion AIC is recommended to be used since it has proved to be the most effective among a number of selection strategies (Moses and Holland 2009). A simple method is to start with the ten first moments and then remove the highest moments and take notice of the change in AIC. When the AIC no longer decreases the model is satisfactory. Using the function `glm()` in R, with `FXEG` being a data frame containing a vector `freq` with the frequencies for each score value and a vector `X` of possible score values, we can write:

```
R> EGX <- glm(freq~I(X) + I(X^2) + I(X^3) + I(X^4) + I(X^5), family =
+  "poisson", data = FXEG, x = TRUE)
```

Together with the `glm` object from the model for Y, the object created can be supplied to **kequate** to conduct an equating.

Single group design

The SG design requires the estimation of a bivariate log-linear model for test scores on X and A. The first step is to estimate the two univariate score distributions for tests X and A. This is done in the same way as in the EG design above. After having obtained satisfactory univariate models, the bivariate model needs to be estimated. The bivariate model should include the univariate moments of the respective univariate log-linear models, along with cross-moments between the two tests X and A. We consider for inclusion in the bivariate log-linear model the cross-moments up to the highest moments contained in the univariate log-linear models. By way of a criterion such as the AIC or a likelihood ratio test, the different models are then evaluated and the most appropriate one is chosen. With the R function `glm()`, with a data frame `SGfreq` containing vectors with the score frequencies `frequency` and score values `X` and `A`, we estimate the log-linear model with two univariate moments for each of the tests and the cross-moments $X * A$ and $X^2 * A^2$ by writing

```
R> SGglm <- glm(frequency~I(X) + I(X^2) + I(A) + I(A^2) + I(A^3) + I(X):I(A)
+  + I(X^2):I(A^2), data = SGfreq, family = "poisson", x = TRUE)
```

Counterbalanced design

In a counterbalanced design, two independent random groups from a common population take the same tests X and Y to be equated. However, they take the tests in a different order, so that one group first takes test X and then test Y, while the other group first takes test Y and then test X. The purpose of this setup is to ensure that any order effects are equally pronounced for both of the tests. To further ensure the validity of the equating, the sample sizes are usually chosen to be equal or almost equal between the groups. Pre-smoothing for the counterbalanced design is done exactly like for a SG design, except in this case we fit separate log-linear models for each of the two independent random groups.

Non-equivalent groups with anchor test design

A NEAT design contains two independent single group designs, resulting in data for two joint distributions P and Q. The components of P and Q are defined as

$$p_{jl} = P(X = x_j, A = a_l|P) \tag{22}$$

$$q_{kl} = P(Y = y_k, A = a_l|Q) \tag{23}$$

where x_j is the score of the j:th item on test X, y_k is the score of the k:th item of test Y and a_l is the score of the l:th item of the anchor test A. There are two common methods for equating tests in a NEAT setting: chain equating (CE) and post-stratification equating (PSE). The pre-smoothing models for these two equating methods do not differ, so the following guide is valid for both CE and PSE.

In the case of a NEAT design the same basic procedure as in an SG design applies but two bivariate log-linear models instead of one must be estimated (one for each of the two populations). The large sample sizes usually found when using a NEAT design allows for the specification of more complicated log-linear models than in other designs, and these models can thus cater more specifically to the particular features of the data that may arise when conducting test equating. The following complexities in the data are common in test data and can be modelled when sample sizes are large (von Davier, Holland and Thayer, 2004):

a.) ”teeth” or ”gaps” in the observed frequencies which occur at regular intervals because of scores being rounded to integer values

b.) a ”lump” at 0 in the marginal distributions caused by negative values being rounded to 0

Because of the fact that such features are common-place in test score data, they may need to be accounted for when conducting the pre-smoothing. The way that this is carried out is to specify additional variables in the log-linear model, indicating zero-values and gap-values. To check for these features in the data at hand, the marginal frequencies of X and A in population P and the marginal frequencies of Y and A in population Q should be plotted. If there are any spikes or lumps at zero or any particular score values in these marginal frequencies, the corresponding additional parameters should be included in the respective models. The way that this is done using R is to create new indicator variables for the particular score values that exhibit irregularities. As in the other designs, the data should be ordered by the score vectors for tests Y and X as conducted in the function `kefreq()`. In case the data is not aggregated using `kefreq()`, the data can be ordered by the following method. We assume that the data is in a data frame `PNEAT`, with vectors `frequency`, `X` and `A`. To order such a data frame by the `A`-vector and, in case of equal values of a, by the `X`-vector, we write:

```
R> PNEATordered <- PNEAT[order(PNEAT$A, PNEAT$X),]
```

When the data is aggregated and sorted correctly, additional variables may need to be specified to model lumps and spikes in the observed data. Below we describe a procedure in R using data frames and the `[]` operator to specify these variables. We let `PNEAT` be a data frame with vectors `frequency`, `X` and `A`, as defined above, with the test X having score values `0:20` and the test A having score values `0:10`. Studying the marginal frequencies, we have discovered that there is a lump at score value zero for test X and a spike at score values 5, 10, 15 and 20 for test X. In light of this, we want to specify additional variables for each of these score values. For the zero score value and for score values 5, 10, 15 and 20, we want an indicator variable taking on value 1 if `X` is equal to the particular values and for score values 5, 10, 15 and 20 we additionally want to specify a variable which takes on value 5 if `X` is equal to 5 and so on for 10, 15 and 20. To create these variables we first define the new variables in the data frame:

```
R> PNEAT$indx0 <- numeric(length(PNEAT$X))
R> PNEAT$ind1x <- numeric(length(PNEAT$X))
R> PNEAT$ind2x <- numeric(length(PNEAT$X))
```

This will create variables of equal length to the others in the data frame containing zero values. We then use the operator `[]` to specify that our new variables should take on particular values if the variable `x` in the data frame has the corresponding value:

```
R> PNEAT$indx0[PNEAT$X==0] <- 1
R> PNEAT$ind1x[PNEAT$X %in% c(5, 10, 15, 20)] <- 1
R> PNEAT$ind2x[PNEAT$X==5] <- 5
R> PNEAT$ind2x[PNEAT$X==10] <- 10
R> PNEAT$ind2x[PNEAT$X==15] <- 15
R> PNEAT$ind2x[PNEAT$X==20] <- 20
```

Similarly, additional variables can be created for any particular values of the A variable. When we have suitable data we can begin the estimation of the log-linear model. First, univariate log-linear models are found using methods identical to the EG case, for each of the four univariate distributions. Having specified these models, the bivariate models for each population is specified in a manner similar to the SG case considering cross-moments up to the highest univariate moments in each univariate log-linear model. Additionally, possible indicator variables and moments of score values corresponding to the particular indicator variables need to be specified. In this example we include the first three moments of the test to be equated and the first two moments of the anchor test. An interaction term between the first moment for the test to be equated and the anchor test is included along with an interaction term between the first moment of the test to be equated and the second moment of the anchor test. Each indicator variable defined above is included. For the variable `ind2x` the first two moments are included. The function call to `glm()` is given below.

```
R> PNEATglm <- glm(frequency~I(X) + I(X^2) + I(X^3) + I(A) + I(A^2) +
+  I(X):I(A) + I(X):I(A^2) + I(indx0) + I(ind1x) + I(ind2x) + I(ind2x^2),
+  data = PNEAT, family = "poisson", x = TRUE)
```

Non-equivalent groups with covariates design

Instead of (or in addition to) using an anchor test to equate tests in a non-equivalent groups design, covariates of test takers can be used to conduct an equating. The same framework as in a NEAT PSE design is used in the NEC design, but inplace of an anchor test the method uses information from covariates to equate the tests. As in the other designs, the observed data needs to be modelled using log-linear models. In the NEC case the model specification is somewhat more complicated, and the data management can be more laborious. In **kequate** the function `kefreq()` cannot be used to tabulate frequency data since the covariates can be of any type and not just integers. The aggregation of the individual data in a NEC design is therefore not easily generalized into a single function. Instead, the data aggregation can be done manually in R using built-in functions. We illustrate how this can be done with a simple example. Consider a test X to be equated with a parallel test Y, which has integer score values from 0 to 40. In addition to the test results, there exists information on the individuals in the form of two covariates, one of which is the grade in mathematics (a quantitative variable with possible values 1, 2 and 3) and the other is a qualitative variable representing type of education (two values). In total, there are thus six different combinations of covariates that are possible. The desired frequency vector is then of size 41*6. At our disposal is a data frame (called `obs11`) with observations from indivuduals for the test to be equated (variable `S11`) and the covariates considered (variables `edu` and `math`). We want to aggregate and sort this data so that each entry in the created vector is a frequency corresponding to a particular combination of test score and covariate values where the data is sorted first by the grade in mathematics, then by the type of education and lastly by the test score received. To do so we use a combination of the functions `table()` and `as.data.frame()`:

```
R> testfreq <- as.data.frame(table(factor(obs11$S11, levels = 0:40, ordered
+  = TRUE), factor(obs11$edu, levels = 1:3, ordered = TRUE),
+  factor(obs11$math, levels = 1:3, ordered = TRUE), dnn = c("S11", "edu",
+  "math")))
```

This will create a data frame with a single vector `Freq`, which contains 41*6 cells each containing a frequency corresponding to the possible score values and covariate combinations. The score and covariate vectors need now be specified. We do so by specifying a data frame, using the `rep()` function to create the vectors needed:

```
R> testdata11 <- data.frame(frequency = testfreq$Freq, S11 = rep(0:40, 6),
+  edu = rep(1:2, each=41), math = rep(1:3, each = 41*2))
```

This call creates a data frame with the correctly sorted vectors to be used in the bivariate `glm` model. To estimate the log-linear model in the NEC case we proceed just as we would in a regular bivariate log-linear model specification. First, the univariate model for the test scores is estimated like in the EG case (`kefreq()` can be used to create the score frequency vector). A bivariate log-linear model is then specified using the moments from the univariate model while adding the variables corresponding to the covariates. Interactions between the moments for the tests and the covariates need to be considered along with interactions between the covariates. In our example, `math` is a quantitative variable and higher moments of this variable can thus be needed and possible interactions between these and other variables may need to be considered. `edu` is a factor and thus needs to be specified as such in the `glm()` function call.

```
R> glm11 <- glm(frequency~I(S11) +  I(S11^2) + I(S11^3) + I(S11^4) +
+  I(math) + I(math^2) + factor(edu) + I(S11):I(math) + I(S11):factor(edu) +
+  I(math):factor(edu), data = testdata11, family = "poisson", x = TRUE)
```

3.3. Assessing model fit

The equating design considered does not in itself affect the way the log-linear models are assessed for validity. There are however certain differences between assessing univariate and bivariate models. As mentioned previously, simply using the AIC to decide on the best model has been shown to be an effective way in the univariate case and this is the recommendation given here. To further ensure that the selected model is satisfactory, the residuals from the model can be analyzed. The Freeman-Tukey residuals are defined as

$$FT_i = \sqrt{n_i} + \sqrt{n_i + 1} - \sqrt{4 * \hat{m}_i + 1}, \tag{24}$$

where n_i is the i:th observed frequency and $\hat{m}_i$ is the i:th fitted frequency. If the observed frequencies are assumed to be Poisson distributed, then the Freeman-Tukey residuals are approximately standard normal distributed. **kequate** includes the function `FTres()` which calculates the Freeman-Tukey residuals from an estimated log-linear model. It takes as input either an object of class `glm` or two vectors where one contains the observed frequencies and where the other contains the estimated frequencies from the log-linear model. We write

```
R> FTglm <- FTres(EGX$y, EGX$fitted.values)
```

Due to the high number of zero frequencies in an observed bivariate test data frequency distribution, an analysis of the Freeman-Tukey residuals is not very useful in the bivariate case. With a bivariate log-linear model, it is instead worthwhile to see how well the estimated distribution approximates the observed distribution by investigating the conditional means, variances, skewnesses and kurtoses of the observed and estimated bivariate distributions. This is facilitated in **kequate** by the function `cdist()` which calculates these conditional moments for observed and estimated bivariate frequency distributions. The input given to `cdist()` should be two matrices containing the observed and estimated frequencies, respectively, on a common population. If `Pest` is the estimated frequency matrix and `Pobs` is the observed frequency matrix, we can write

```
R> NEATPcdist <- cdist(Pest, Pobs)
```

The object returned by `cdist()` is of class `cdist` and contains four data frames which store the conditional parameters of each distribution (for tests X and A, the output contains both the parameters for $X|A$ and for $A|X$ for both the observed and the estimated distributions). If the conditional parameters of the estimated distribution do not deviate too much from the conditional parameters of the observed distribution, then the estimated log-linear model is proper to use.

In selecting a bivariate log-linear model we recommend using a criterion such as the AIC or a likelihood ratio test to compare models to each other, and then to verify the suitability of the model by assessing the conditional parameters. If the conditional parameters are dissimilar between the observed and estimated distributions, additional parameters may need to be added to accurately model the observed data.

4. Kernel equating with kequate

The package **kequate** for R enables the equating of two parallell tests with the kernel method of equating for the EG, SG, CB, NEAT PSE, NEAT CE and NEC designs. **kequate** can use `glm` objects created using the R function `glm()` (**stats** R Development Core Team 2013) as input arguments and estimate the equating function and associated standard errors directly from the information contained therein. Support is also provided for item-response theory models estimated using the R package **ltm**. The S4 system of classes and methods, a more formal and rigorous way of handling objects in R (for details see e.g. Chambers (2008)), is used in **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 Table 1.

```
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 utilize the objects created by `glm()` as input to `kequate()`. For help in estimating log-linear models, see Section 3. Optional arguments to specify the continuization parameters directly are also available for all equating designs. In addition, there exists the option to only conduct a linear equating and an option to use unsmoothed input proportions. By default a Gaussian kernel is used but the option to use either a logistic or uniform kernel is provided. 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, 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 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 parallell 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 an object of class `keout`, a number of get-functions are available. They are described in Table 2. Methods for the class `keout` are implemented 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

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

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. For details on how to use the functions `kefreq()`, `FTres()` and `cdist()`, see Section 3.

5. Examples

We exemplify the main function `kequate()` by equating using each of the designs available in **kequate**. We demonstrate how to use different types of input arguments and how each optional argument can be used. Additionally, we show how to utilize functions that are common to all designs.

5.1. Equivalent groups design

Now, let the parallel tests X and Y have common score vectors <0, 1, 2, ..., 19, 20>. The tests are each administered to a randomized group drawn from the same population, thus we have an EG design. We assume that the log-linear models have been specified using the `glm()` function in R and that two objects `EGX` and `EGY` have been created. To 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, EGX, EGY)
```

This will create an R object `keEG` of class `keout` containing information about the equating, retrieved by using the functions described in Table 1. 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, with observed frequency vectors `EGX` and `EGY`, we can write:

```
R> keEGobs <- kequate("EG", 0:20, 0:20, EGX$y/1453, EGY$y/1455, N = 1453,
+  M = 1455, smoothed = FALSE)
```

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 vector containing the score values for the equated tests.
`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 2: Functions to retrieve information from the resulting `keout` objects.

The object created is of class `keout` and contains similar information to an object from an equating with pre-smoothed data. To print useful information about the equating, we can utilize the `summary()` function. Using 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 0.6149201 0.5876649 3807.166 3935.618

 Equating Function and Standard Errors:
   Score       eqYx     SEEYx
1      0 -0.2241994 0.3336974
2      1  0.8119466 0.4999356
3      2  2.0698202 0.4448033
4      3  3.3264645 0.3309319
5      4  4.5052852 0.2618650
6      5  5.6111928 0.2252960
7      6  6.6642922 0.2022103
8      7  7.6848686 0.1864710
9      8  8.6910779 0.1780239
10     9  9.6985124 0.1761879
11    10 10.7194308 0.1779169
12    11 11.7612527 0.1803009
13    12 12.8245173 0.1831748
14    13 13.9012027 0.1880667
15    14 14.9749197 0.1941770
16    15 16.0239962 0.1980949
17    16 17.0264901 0.1994433
18    17 17.9642896 0.2025415
19    18 18.8237647 0.2050841
20    19 19.5971798 0.1850754
21    20 20.3299568 0.1279832
```

```
 Comparing the Moments:
         PREYx
1  0.005767387
2  0.010242509
3  0.026479456
4  0.056341140
5  0.104572111
6  0.176515489
7  0.276731565
8  0.408622931
9  0.574500769
10 0.775796672
```

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 percent relative error (PRE) is given.

5.2. Single group design

Besides equating, **kequate** allows for the linking of two tests of abitrary lengths. Let X and A be two tests to be linked, with test X having 20 items and test A having 10 items. In a single group design, we let a random group of individuals from a population take both test X and test A. For this design we then have combinations of scores for the two tests for each individual, which we tally to get the frequency for each combination of scores. Usually we specify a log-linear model for the observed frequencies. In the following, let `SGglm` be a `glm` object containing a suitable bivariate log-linear model specification. To link tests X and A we write

```
R> keSG <- kequate("SG", 0:20, 0:10, SGglm)
```

We retrieve a summary of this linking by writing

```
R> summary(keSG)

 Design: SG equipercentile

 Kernel: gaussian

 Sample Sizes:
   Test X: 1000
   Test Y: 1000

 Score Ranges:
   Test X:
       Min = 0 Max = 20
   Test Y:
       Min = 0 Max = 10

 Bandwidths Used:
         hx        hy    hxlin    hylin
1 0.6159081 0.519231 4031.951 2623.711

 Equating Function and Standard Errors:
   Score        eqYx      SEEYx
1      0 -0.73996587 0.04929541
2      1 -0.34600180 0.06297660
3      2  0.02819341 0.08033200
4      3  0.47091605 0.10012052
5      4  0.99425756 0.10531094
6      5  1.58837811 0.10359545
7      6  2.22173831 0.09303311
8      7  2.88822228 0.08364745
9      8  3.56740436 0.08089191
10     9  4.25360685 0.07926455
11    10  4.95021363 0.07900876
12    11  5.64724131 0.08105986
13    12  6.33976366 0.08206044
14    13  7.03030488 0.08484320
15    14  7.70567658 0.09402151
16    15  8.34635860 0.10273355
17    16  8.94117411 0.10360043
18    17  9.45801885 0.09695159
19    18  9.89326510 0.07837055
20    19 10.26592380 0.06199019
21    20 10.66923086 0.04816967

 Comparing the Moments:
          PREYx
1  -0.0063315563
2   0.0002224519
3  -0.0043701869
4   0.1168399319
5   0.3794773105
6   0.8023720758
7   1.3969086377
8   2.1710352152
9   3.1307479747
10  4.2811021272
```

Now, let's say we have done the pre-smoothing using another function in R or using an external software package and want to supply `kequate()` with the relevant information from this model. The expected class of the argument `P` is then a matrix of estimated bivariate

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
19 & 361 & 0 & 0 & 0 & 0 & 0 \\
20 & 400 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
2 & 4 & 1 & 1 & 1 & 2 & 4 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
20 & 400 & 1 & 1 & 1 & 20 & 400 \\
0 & 0 & 2 & 4 & 8 & 0 & 0 \\
1 & 1 & 2 & 4 & 8 & 2 & 4 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
20 & 400 & 2 & 4 & 8 & 40 & 1600 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 10 & 100 & 1000 & 0 & 0 \\
1 & 1 & 10 & 100 & 1000 & 10 & 100 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
20 & 400 & 10 & 100 & 1000 & 200 & 40000
\end{bmatrix}$$

Table 3: The design matrix from the SG log-linear model specification.

probabilities from the log-linear model. Additionally we need to specify the arguments `DM` and `N` in the `kequate()` function call. The object `DM` is the design matrix from the log-linear model, which should be an object of class `matrix` containing values of the explanatory variables for each score value combination provided in the log-linear model specification while the object `N` is the sample size. To provide the same log-linear model as in the above function call, but using these additional arguments instead of a `glm` object, we define the matrix `DMSG`, of size $241 * 7$, as an object of class `matrix` containing cells as in the matrix given in Table 3 below. We then link the two tests by writing

```
R> keSGDM <- kequate("SG", 0:20, 0:10, P = PSG, DM = DMSG, N = 1000)
```

The same linking is then conducted, since the log-linear models are identical between the two different function calls.

5.3. Counterbalanced design

The counterbalanced design features two separate groups taking two tests, X and Y, to be equated. One of the groups takes test X first and then test Y, while the other group first takes test Y and then takes test X. In principle, the counterbalanced design can be viewed as two single group designs utilized in conjunction. For each group, a separate bivariate log-linear model is specified just like in the SG case. The two log-linear models are then provided to `kequate()` and tests X and Y are equated. We assume that the log-linear models have been specified and that two objects, `CBglm12` and `CBglm21` have been created using the R function `glm()`. In a CB design we then write

```
R> keCB <- kequate("CB", 0:40, 0:40, glmCB12, glmCB21)
```

Unique to the CB design is the argument `wcb`, which specifies the weighting of the two groups taking the test. The default is `wcb = 1/2`, meaning that the two groups are given the same weight.

5.4. Non-equivalent groups design

Chain equating

In a NEAT design, let the parallel tests X and Y have common score vectors <0, 1, 2, . . . , 19, 20> and let the anchor test A have the score vector <0, 1, 2, . . . , 9, 10>. Using chain equating, we can equate these two tests without assuming that the group taking test X and the group taking test Y come from the same population. In chain equating, this is accomplished by first linking test X to the anchor test A, and then linking the anchor test A to the test Y. We assume that the log-linear model is estimated using a method similar to that in Section 3.2 only without the parameters for lumps and spikes, and that `NEATglmP` and `NEATglmQ` are the resulting `glm` objects. We equate the two tests by writing

```
R> keNEATCE <- kequate("NEAT_CE", 0:20, 0:20, 0:10, NEATglmP, NEATglmQ)
```

This creates a `glm` object `keNEATCE` which can be used as input to the generic functions `plot()` and `summary()`, among others. Thus, we plot the the equating function and the standard error of equating by writing

```
R> plot(keNEATCE)
```

The resulting plot can be seen in Figure 1.

Post-stratification equating

We again want to equate two tests X and Y with common score vectors <0, 1, 2, . . . , 19, 20>. By using an anchor test we can equate these tests without assuming that the populations taking each test are perfectly identical. The log-linear model estimation procedure for CE and NEAT do not differ, so we again suppose that `NEATglmP` and `NEATglmQ` are appropriate `glm` objects containing bivariate log-linear models over P and Q respectively. We can then equate the tests X and Y in a NEAT PSE design by writing:

```
R> keNEATPSE <- kequate("NEAT_PSE", 0:20, 0:20, NEATglmP, NEATglmQ)
```

This will create an object of class `keout` containing information about the equating. No matter the design used, the objects are still of the same class with certain slots filled in while others are not depending on the design.

Like for all objects of class `keout`, we can plot the object `FNEATPSE` by writing:

```
R> plot(keNEATPSE)
```

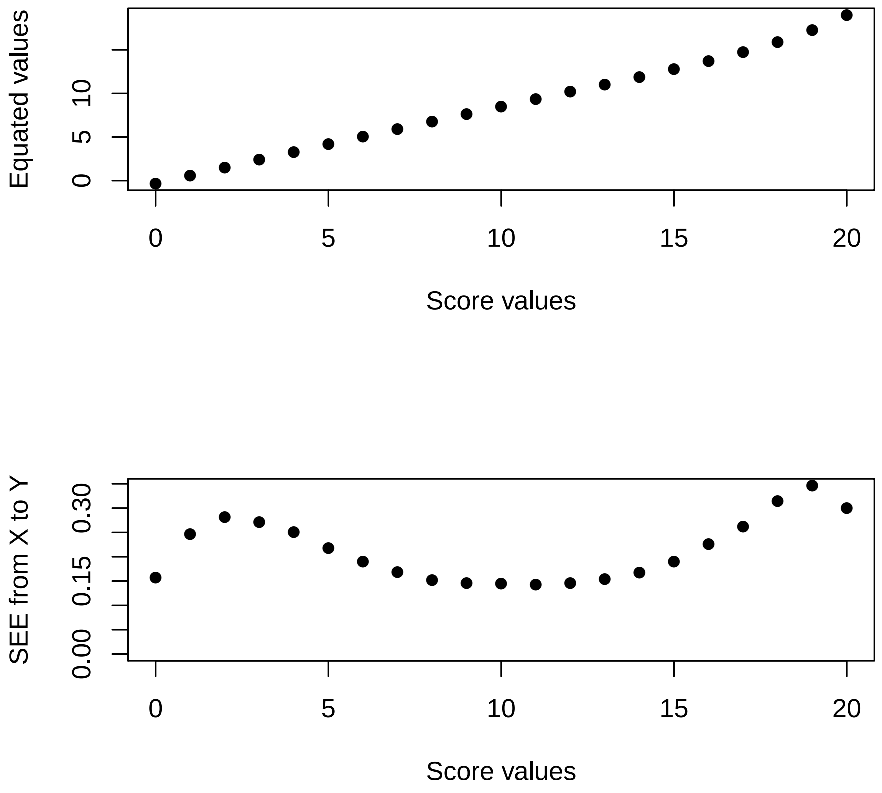


Figure 1: The equated values and corresponding SEE for each score value in a NEAT CE design.

The resulting graph is shown in Figure 2, 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.

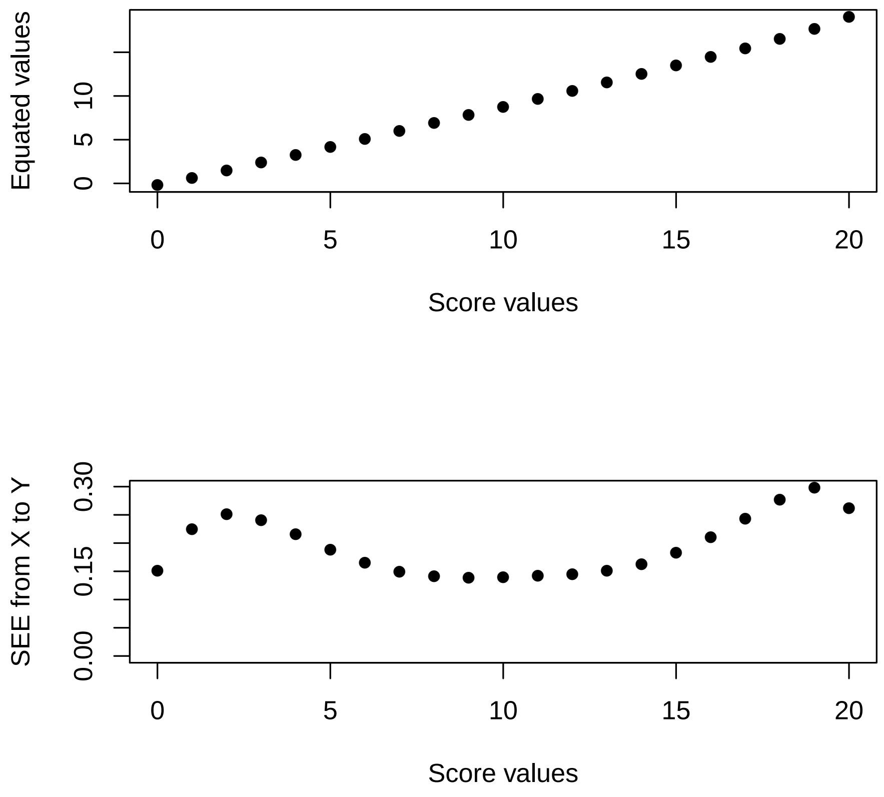


Figure 2: The equated values and corresponding SEE for each score value in a NEAT PSE design.

In the above function calls, the default settings have been used. Under the default settings, both a KE-equipercentile equating and a linear equating is done. The continuization parameters will by default be set to the optimal value in the KE-equipercentile case and to 1000*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> keNEATPSEnew <- kequate("NEAT_PSE", 0:20, 0:20, PNEATglm, QNEATglm, hx =
+  0.5, hy = 0.5, hxlin = 1000, hylin = 1000)
```

Equating using covariates

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. We conduct the equating based on the log-linear model specification given for the non-equivalent groups with covariates design in Section 3.2. The `glm`-objects `NECPglm` and `NECQglm` for each test have been specified and we equate the two versions of the test by writing

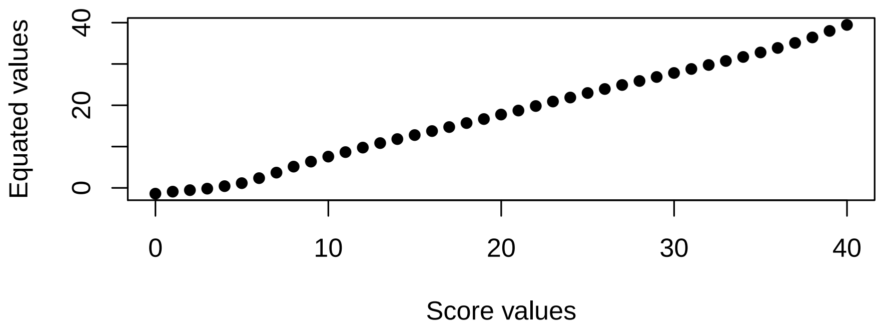
```
R> NECtest2012 <- kequate("NEC", 0:40, 0:40, glm12, glm11)
```

The output is given below, showing that test X is slightly more difficult when we have conditioned on relevant background variables. The estimated standard errors are small for the score values with high sample sizes but for very low and very high score values the standard errors are higher.

```
R> summary(NECtest2012)

 Design: NEAT/NEC PSE equipercentile

 Kernel: gaussian

 Sample Sizes:
   Test X: 10000
   Test Y: 10000

 Score Ranges:
   Test X:
       Min = 0 Max = 40
   Test Y:
       Min = 0 Max = 40

 Bandwidths Used:
          hx        hy    hxlin    hylin
1 0.7349972 0.6419125 4942.594 4992.187

 Equating Function and Standard Errors:
   Score       eqYx      SEEYx
1      0 -1.3451569 0.18440536
2      1 -0.9602255 0.21184275
3      2 -0.5965172 0.24085836
4      3 -0.1807436 0.29052152
5      4  0.3718111 0.39285435
6      5  1.2026113 0.55387438
7      6  2.3858732 0.63292883
8      7  3.7556145 0.56339995
9      8  5.1190791 0.44154976
10     9  6.4000736 0.33341702
```

```
11    10  7.5933189 0.25256016
12    11  8.7145725 0.19557247
13    12  9.7815391 0.15626437
14    13 10.8092984 0.12949485
15    14 11.8101774 0.11144623
16    15 12.7944756 0.09930139
17    16 13.7711411 0.09101926
18    17 14.7481910 0.08520128
19    18 15.7327838 0.08096996
20    19 16.7308140 0.07784470
21    20 17.7459112 0.07563339
22    21 18.7779678 0.07433908
23    22 19.8219681 0.07404189
24    23 20.8685273 0.07474434
25    24 21.9068451 0.07629800
26    25 22.9285492 0.07851806
27    26 23.9299210 0.08136646
28    27 24.9117372 0.08505365
29    28 25.8778741 0.09008034
30    29 26.8339311 0.09728422
31    30 27.7863959 0.10788863
32    31 28.7423417 0.12351268
33    32 29.7095325 0.14614043
34    33 30.6968513 0.17811930
35    34 31.7150842 0.22226125
36    35 32.7781765 0.28196726
37    36 33.9051492 0.36089575
38    37 35.1226448 0.46049461
39    38 36.4664784 0.56793424
40    39 37.9666651 0.60513937
41    40 39.4943066 0.39106819

 Comparing the Moments:
          PREYx
1   0.0004354315
2   0.0006262231
3   0.0017847741
4   0.0039844792
5   0.0069819281
6   0.0098187014
7   0.0105345986
8   0.0060883869
9  -0.0075729485
10 -0.0353356873


R> plot(NECtest2012)
```

The resulting plot is seen in Figure 3. **kequate** enables the usage of logistic and uniform



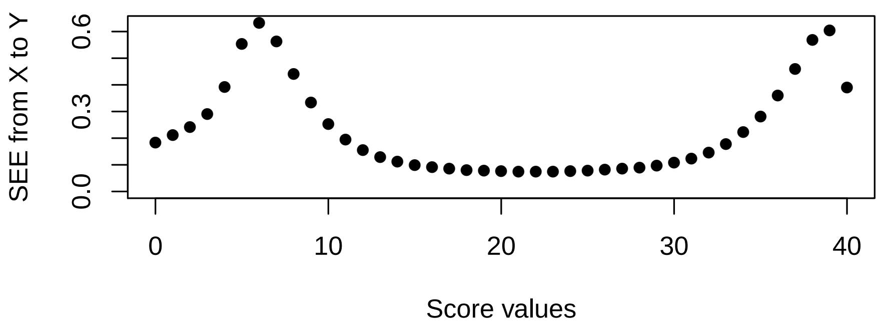


Figure 3: Equated values and SEE in the case of equating with covariates.

kernels in addition to the default gaussian kernel. 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> NECtestL <- kequate("NEC", 0:40, 0:40, glm12, glm11, kernel = "logistic")
R> NECtestU <- kequate("NEC", 0:40, 0:40, glm12, glm11, kernel = "uniform")
```

In this case the equating function is almost identical between the three kernels but there are some differences in the standard error of equating for low and high score values, which can be seen in Figure 4.

5.5. Additional features

kequate also enables IRT observed-score equating (IRT-OSE) using the arguments `irtx` and `irty`. We let `irtmatx` and `irtmaty` be matrices where each column represents an ability level

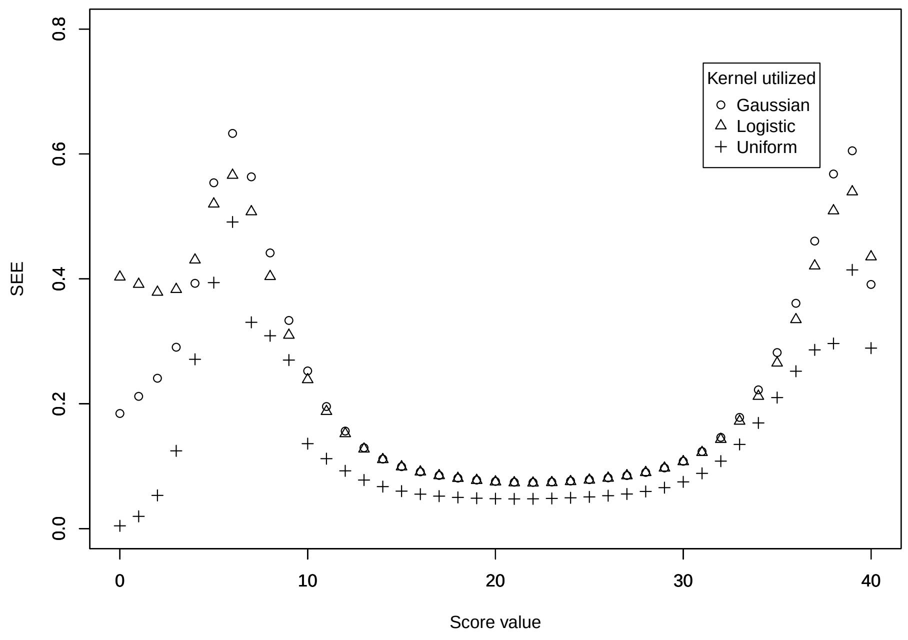


Figure 4: SEE for three different kernels in the case of equating with covariates.

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, EGX, EGY, irtx = simeq$irt2, irty =
+  simeq$irt1)
```

This will instruct `kequate()` to 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**.

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 a linear equating only is to be performed, where default is `linear = FALSE`. 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 a NEAT PSE equating to a NEAT CE design and to plot the resulting object, we write:

```
R> SEEDPSECE <- genseed(keNEATPSE, keNEATCE)
R> plot(SEEDPSECE)
```

The resulting plot is seen in Figure 5. Given an object of class `keout` created by `kequate()`

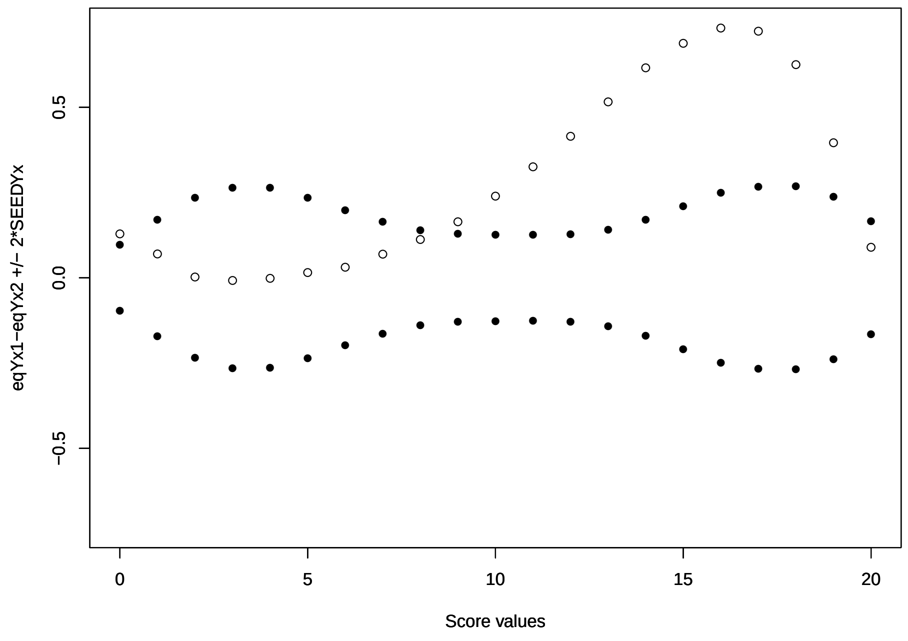


Figure 5: The difference between PSE and CE in a NEAT design for each score value with the associated SEED.

using the function call `linear = FALSE` (default), the SEED between the KE-equipercentile and the linear equating functions can be retrieved by using the `getSeed()` function. The function `getSeed()` returns an object of class `genseed` which can be plotted using the generic function `plot()`, resulting in a graph similar to the one in Figure 5.

References

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/`.

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

Bränberg K, Wiberg M (2011). "Observed Score Linear Equating with Covariates." *Journal of Educational Measurement*, **41**, 419–440.

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

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

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

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

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

Moses T, Holland P (2009). "Selection Strategies for Univariate Loglinear Smoothing Models and Their Effect on Equating Function Accuracy." *Journal of Educational Measurement*, **46**, 159–176.

R Development Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL `http://www.R-project.org/`.

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

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

Marie Wiberg
Department of Statistics, USBE
Umeå University
SE-901 87 Umeå, Sweden

E-mail: `marie.wiberg@stat.umu.se`
URL: `http://www.usbe.umu.se/om-handelshogskolan/personal/maewig95`