Cambridge Assessment

Can AI learn to equate?

Conference Paper

Tom Benton

Presented at the International Meeting of the Psychometric Society,
Zurich, Switzerland,
July 2017

Author contact details:

Tom Benton
Assessment Research and Development,
Research Division
Cambridge Assessment
1 Regent Street
Cambridge
CB2 1GG
UK

benton.t@cambridgeassessment.org.uk

http://www.cambridgeassessment.org.uk

As a department of Cambridge University, Cambridge Assessment is respected and trusted worldwide, managing three world-class examination boards, and maintaining the highest standards in educational assessment and learning. We are a not-for-profit organisation.

How to cite this publication:

Benton, T. (2017, July). *Can AI learn to equate?* Paper presented at the International Meeting of the Psychometric Society, Zurich, Switzerland.

Abstract

Equating involves discovering a transformation of the score scale on one test form such that the transformed scores can be interpreted interchangeably with those on another test form. This research attempts to generate new approaches to classical (non-IRT) equating in the non-equivalent anchor test (NEAT) design using techniques from the AI and machine learning community. Software enabling complex machine learning has become widely available in recent years and has been successfully employed to tackle a variety of difficult computational problems. For example, AI can learn to play the game of Go by playing against itself millions of times and learning which moves are likely to lead to victory in different situations. This research examined whether we can train AI to equate in a similar way. Using data from many simulated scenarios, where the true equating function is known, we trained a machine learning algorithm to reproduce the true equating functions as accurately as possible from information about the scenario that would be available in practice. Once trained, the resulting algorithm was then applied to perform equating against real (as opposed to purely simulated) data and displayed a superior performance to existing classical equating techniques.

Introduction

Equating involves searching for a transformation of the score scale on one test form such that the transformed scores can be interpreted interchangeably with those on another test form. This strict definition of complete inter-changeability is difficult to achieve in practice. For example, it requires that the two test forms have exactly equal levels of reliability. In practice, we often limit ourselves to searching for score points on the two test forms that might be described as being equally difficult in some sense.

Where two different test forms (Form X and Form Y) have been taken by equivalent populations, the task of identifying equivalent scores on the two forms is fairly straightforward. A common approach is known as *equipercentile equating*. For each score on Form X we calculate the proportion of examinees achieving at or below this score. Then, for Form Y we find the score where the same proportion of examinees achieve at or below it. At the end of this process we have a set of equivalent scores on the two forms. We call this mapping from scores on one form to scores on the other the equipercentile equating function.

There are a few minor complications to the above description. Specifically:

- It is unlikely that there will be any score on Form Y with an *exact* match for the proportion examinees scoring at or below each given score on Form X. For this reason we use a *continuization* step to convert the discrete score distribution[1] to a continuous one where an exact match can be found.
- We may not be confident that the proportion of examinees achieving a given score or below would be exactly replicated in another sample. This is particularly true if only a small sample of examinees have taken each form. To produce more stable estimates of these proportions we may apply *smoothing* to the score distributions. One popular form of smoothing is to represent both distributions purely in terms of their means and standard deviations. As such, a score a certain number of standard deviations above the mean on one form is considered equivalent to a score the same number of

[1] Usually examinees' scores are described in terms of whole numbers.

standard deviations above the mean on the other form. This form of equating is known as *linear equating*.
- Even with a large sample, the equating function itself is dependent upon the population of examinees who take the two forms. As such, it is important to define a target population where the equating function will be used.

Several traditional techniques for addressing the above steps are described within Kolen and Brennan (2004). As an alternative to some of these traditional techniques, the necessary stages of smoothing, continuization and equating have been brought into a single framework by *kernel equating* (von Davier, Holland and Thayer, 2003).

A more challenging scenario emerges when the two test forms being equated have been taken by different groups. Typically in this situation the two samples of examinees taking the different forms (from populations P and Q) also take an anchor test to allow some form of calibration between the two test forms. This equating design is known as the non-equivalent anchor test (NEAT) design.

Both Kolen and Brennan (2004) and von Davier et al (2003) describe several methods for equating within the NEAT design. For the purposes of this paper I will ignore methods based on item response theory (IRT) and investigate methods that will work if the only data we have available for the task is as follows:

- From population P, total scores for a sample of n_X examinees on form X and the anchor (form A) respectively.
- From population Q, total scores for a sample of n_Y examinees on form Y and the anchor (form A) respectively.

In both cases we will assume that we have an external (as opposed to internal) anchor test that does not contribute to scores on form X or form Y.

The most common techniques to perform equating within the NEAT design can be broadly grouped into two categories: post-stratification equating (PSE) and chained equating (CE). Post-stratification equating (PSE), also known as the frequency estimation (FE) method, essentially works by weighting the samples of examinees taking the two test forms so that their score distributions on the anchor test are equivalent[2]. Now, having made the two sets of examinees equivalent in some sense, we can use standard equipercentile or linear equating techniques. As an alternative, chained equating breaks the process of equating into two steps. First form X is equated to the anchor test and then the anchor test is equated to form Y. Successively applying these two transformations together yields an equating function from form X to form Y.

However, both of the above techniques tend to yield biased results when there is a large difference in the abilities of populations P and Q. Specifically, due to measurement error in the anchor test, both techniques are likely to under-compensate for the difference in ability between the two sets of examinees (Wang & Brennan, 2009, von Davier & Chen, 2013). This will lead to the equating function being biased with the extent of this bias increasing as the reliability of the anchor test decreases. This issue is particularly acute for PSE, however, it is still present for CE. The improvement in performance for CE in this respect comes at the expense of the technique lacking any sensitivity to the population in which equating is going to be applied.

[2] That is, equivalent to the expected score distribution in the target population where equating will be applied. Typically this is chosen either to match the distribution amongst the form X examinees (population P), the form Y examinees (population Q) or some weighted combination of the two.

The above issue with the major classical equating techniques has been known for a long time. However, very few solutions have been suggested. One of the oldest methods was that suggested by Levine (1955) and presented in a more general form by Kolen and Brennan (2004). This method relies on the congeneric model from classical test theory. In essence this assumes that that the items in the anchor are like those in the tests being equated in terms of style and content so that the relative reliabilities of the test forms and the anchor can be determined from their relative variances. This is similar to the usual assumption within classical test theory regarding the relationship between test reliability and test length. This assumption is then combined with the observed correlation between the anchor test and the test forms to yield an actual estimate of the reliability of each form which can be used to work out an adjustment to the usual linear equating approach that should reduce bias.

However, as can be seen from the above description, the Levine method makes a number of assumptions. Firstly, it is a form of linear equating so that it assumes that the differences in the distributions of forms X and Y can be captured entirely by their means and standard deviations. Secondly, it is reliant upon classical test theory and the congeneric model in particular. This implies a very close link between the relative lengths of the various test forms and the anchor, and their reliabilities. This assumption need not necessarily hold in practice. In particular, if the anchor test contains only a small number of items[3] it is not necessarily the case that the items within it will reflect the items that comprise the two test forms.

An extension of the Levine method beyond linear equating was provided by von Davier and Chen (2013). This method combines the results from standard PSE equating with those from Levine equating in an effort to produce a (potentially) non-linear and unbiased equating function. However, although this method addresses the assumption of non-linearity it still relies upon the congeneric model. As such, it may provide inaccurate results where this model does not apply.

Other methods to address equating bias in the NEAT design, such as that proposed by Wang and Brennan (2009), rely on us having an independent estimate of the reliability of the anchor test such as might be derived from analysis of item level data. However, the present paper is concerned with methods that are applicable even when such data is not available. As such, these methods are not considered[4].

So far, we have seen that very few general methods are available to address equating bias in the NEAT design. Even those that have been proposed rely on strong assumptions that may not hold in practice. One possible reason for this lack of progress is that previous research has relied on our ability to capture the complexities of the situation within a few tidy equations. However, given the number of factors that may impinge upon the success of different techniques (sample size, the shape of distributions, reliability of all three forms, consistency of standard errors of measurement across the score range,…) it is perfectly plausible that no such simple equations exist. For this reason, we may find that we can make more progress by dispensing with the need to have a clear and explicable set of equations to explain any new technique, and instead use brute force computational power to produce an algorithm that will provide accurate results across a range of scenarios.

The essential idea of this paper is to train a neural network to equate. It is already commonplace for simulations of data to be used to *test* various equating procedures. Within

[3] Potentially it might consist of only one item – perhaps requiring an extended response.
[4] Of course, any estimate of reliability is itself predicated on particular assumptions that may not apply in practice.

this paper we will go further and use simulations to actually develop the equating algorithm in the first place.

Before beginning this research we already had good reason to believe that a machine learning approach would be able to find interesting solutions where analytical approaches have failed. In recent years neural networks and similar techniques (variously labelled under machine learning, artificial intelligence, deep learning,…) have successfully generated algorithms to win games of Go[5], accurately separate pictures of cats from pictures of dogs[6] and interpret written language within images[7]. All of these examples represent scenarios where articulating the way in which a task should be done with a few simple equations is extremely difficult. However, despite the lack of simple written equations describing the algorithm, we know that, for example, an algorithm to differentiate between cats and dogs exists – after all, our own brains perform this task easily. Thus, just because it may be difficult to articulate an algorithm in simple equations doesn't mean it doesn't exist. Furthermore, we know that artificial intelligence can often find such algorithms where humans working alone have failed. With this in mind, we wish to discover what artificial intelligence can achieve with the task of equating.

The remainder of this paper describes one approach to developing equating algorithms using neural networks and describes its performance. The final trained model is shared within the R package *KernEqWPS* which is available at https://github.com/CambridgeAssessmentResearch/KernEqWPS.

Training AI to equate

The basic idea is to simulate numerous scenarios representing practical equating tasks that could be faced in practice. For each scenario the neural network will be provided with the same information that an analyst would use to perform equating. Alongside this information, we provide details of what the true equating function (that we would *not* usually know) should be. During training the neural network has to find an algorithm that will correctly identify the true equating function based upon the available data.

If we ignore the possible availability of data on individual items within each test, all of the information required to complete equating in the NEAT design is contained within two cross-tabulations: One capturing the relationship between form X and the anchor, and one capturing the same information for form Y and the anchor for the samples within which data is available. Specifically, we just need one table telling us how many people got each possible combination of scores on form X and the anchor, and another saying how many people got each possible combination of score on form Y and the anchor. For the purposes of these initial simulations the maximum score on both form X and form Y is fixed at 50 and the maximum score on the anchor is fixed at 10. Thus, a total of (51*11=) 561 values are required to capture the relationship between each form and the anchor. These values were actually stored as percentages within each sample as early testing suggested that this helped to improve the accuracy of the neural networks in the early stages of training. In order to retain full information, the sample sizes of the data available on the two forms is also stored and provided to the neural network. Finally the distribution of scores on the anchor test in the intended target population is supplied. Again this is done in the form of the percentage of examinees achieving each score on the anchor (11 pieces of information) and

[5] http://www.bbc.co.uk/news/technology-40042581
[6] https://www.kaggle.com/c/dogs-vs-cats
[7] http://rrc.cvc.uab.es/

the sample size from the target population. Thus a total of (561+561+2+11+1=)1136 numbers are stored to specify each situation.

The next issue to face is how to store the true equating function. One option would be to store this as 51 values identifying the form Y score that equates to each form X score between 0 and 50[8]. However, the problem with directly training an algorithm to reproduce such functions is that it is difficult to constrain any such algorithm to meet the usual symmetry requirement in equating. In other words, if trained in such a direct fashion there is no guarantee that the function used to equate form X to form Y would be symmetrical with the function used to equate form Y to form X in the same population. To overcome this we instead store the outcome function as a set of 99 percentile points for (continuized versions of) the form X and form Y score distributions within the target population. Thus, the neural network will have to try to predict both sets of percentiles and then the full equating function will be estimated by interpolation between these points. It is relatively simple to constrain the trained algorithm to ensure that whatever steps are applied to predict the percentiles of form X in the target population are the same as those used for form Y. Thus, symmetry is preserved. As a result, a multivariate outcome consisting of (99+99=)198 values is stored for each scenario. This means that a total of (198+1136=)1334 values need to be stored for each simulated scenario.

There are many ways in which scenarios could be simulated. For example, we could assume some form of basic IRT model underlying the three test forms, simulate the test scores and store the data. However, given that this research was done within a large testing organisation (Cambridge Assessment) it made sense to make some use of the wealth of information that we hold about realistic score distributions that have occurred in practice in the past. Specifically, the analysis first identified 184 score distributions from examinations taken by at least 3,000 examinees in June 2015. These score distributions were condensed to fit with the score range of 0-50 for simulating the form X and form Y distributions and 0-10 for the anchor test[9]. These "real" distributions formed the basis of simulation using the following procedure:

Step 1: Start with multivariate normal simulations

To begin with we simulate the abilities of samples of examinees in groups P and Q from normal distributions with a standard deviation of 1. The mean ability within group P is fixed at 0. Next for each scenario we are simulating we select: a sample size for group P of between 100 and 500, a sample size for group Q of between 100 and 500 and a mean ability for group Q between -0.5 and 0.5. All of the three selections are made from a uniform distribution. With this information in place we can simulate abilities (θ) within groups P and Q.

Next, we need to actually simulate achievement on the form X, Y and the anchor. Initially achievement is simulated using normal distributions (these might be viewed as standardised scores). Before simulation we select reliabilities for forms X, Y and Z denoted α_x, α_y, *and* α_a. The reliabilities for forms X and Y are (independently) selected from a uniform distribution ranging from 0.7 and 0.9, whereas the reliability for the anchor is selected to be between 0.4 and 0.7. These figures were selected as they are consistent with typical values in real assessments with a maximum of 50 (see Figure 1.4 of Bramley and Dhawan, 2012). Then, for each form *j* (can be *x*, *y* or *a*) the standardised scores for each individual are simulated

[8] Discussion of how to extend the resulting algorithm to situations with different test lengths will be given later.

[9] For example, in the condensed distribution out of 50, the percentage of examinees achieving 1 mark was defined by interpolating between the percentage of examinees achieving either of the two marks closest to 2 per cent of the total in the reality.

from a normal distribution with mean $\theta\sqrt{\alpha_j}$ and standard deviation $\sqrt{1-\alpha_j}$. This ensures the correct level of correlation between ability and (normalised) achievement on each test form given the reliabilities we have selected.

Step 2: Select "real" score distributions and calculate the true equating functions
So far our "score" distributions are all continuous variables from a normal distribution. This next step converts them to more realistic score distributions using the following process for each form:

1. Select a true distribution as an equal mixture of 2 randomly selected distributions from amongst the 184 real distributions identified earlier. A mixture is used to add diversity to the distributions used within simulations.
2. Represent the resulting score distributions in terms of 50 cut-points on the standard normal distribution (10 for the anchor test) that would convert a normally distributed variable to the given shape[10].
3. Add further distortion to this distribution by applying a randomly chosen linear transformation with an intercept between -0.9 and 0.9 and a slope between 0.8 and 1.1[11]. Again this step was added to ensure a diverse range of scenarios.
4. Apply these cut-points to the standardised scores for each form generated earlier to produce simulated test scores with the desired range.

Once real score distributions have been selected for forms X and Y it is also straightforward to calculate the true percentiles of these full distributions in each of populations P and Q. Before calculating percentiles[12] the distributions are continuized using the usual principles from kernel equating and a small bandwidth.

Step 3: Store the data
Having simulated scores on forms X and Y we now need to store all of the data described above. This means creating a cross-tabulation of form X and Y scores against the anchor test for the respective simulated samples. For all of the analysis in this paper, the target population is defined as population Q. The distribution of anchor scores within the sample from this population (that is, those that took form Y) is recorded as part of the information that will be required to complete equating. In addition, the calculated true percentiles of each form within population Q are stored as the outcomes that we are trying to predict.

The above steps were run a total of 20,000 times. The resulting data set, with 20,000 rows and 1334 columns, was used to train a neural network to identify a suitable algorithm. In addition, a further 1,000 scenarios were simulated and used as a validation set to monitor the performance of the algorithm during training (but not used to help adjust model parameters). The use of a validation set is crucial within any machine learning task to avoid the danger of over fitting.

Training the model
Neural networks linking the characteristics of each scenario to the true percentiles of form X and Y were fitted using the TensorFlow software library (https://www.tensorflow.org/). TensorFlow was developed by the Google Brain team and can be used to address a wide

[10] In other words, the cut-points find the quantiles on the standard normal distribution associated with the cumulative percentage of examinees at each score point.
[11] The slopes are biased towards being lower than one as this has the effect of spreading out the score distributions on the final test forms. This was found to be important as many of the real distributions came from tests with maximums well above 50 and, as a result, had a greater proportion of scores concentrated towards the middle of the score range than we might have expected.
[12] The term percentiles is used here to literally mean all 99 percentiles from 1 per cent to 99 per cent inclusive.

variety of machine learning problems. It was chosen for use in this problem because it can be easily customized to work with multivariate outcomes. In addition, it can jointly handle form X and form Y percentiles from within the same scenario whilst ensuring that they are treated in a symmetrical fashion.

Neural networks are often viewed in terms of the analogy between the way they are designed and the human brain. As such, they are often displayed in terms of layer upon layer of neurons with a multitude of connecting lines between them. However, for the purposes of actually applying these techniques, it is easier just to think of a neural network in terms of the simple mathematical formulae underpinning its operation. We begin with a matrix *X* where rows represent cases and columns represent variables. We can then transform this matrix of inputs to a new matrix of values representing each case. This matrix, denoted by L_1, is defined via the following equation.

$$L_1 = g(XW_1 + \boldsymbol{b}_1\mathbf{1}^t)$$

L_1 is referred to as the first hidden layer. W_1 is a parameter matrix of weights with the number of rows equal to the number of variables in X and a number of columns equal to the number of values we wish to store in the hidden layer (that is, the number of neurons). $\boldsymbol{b}_1$is a parameter vector of biases that are added to each column of XW_1. Finally, g is a link function that allows us to capture nonlinear relationships within the neural network. For the models described in this paper, g was chosen to be the rectifier function, commonly used within neural networks, such that $g(x) = \max(0, x)$. Subsequent hidden layers are defined using similar equations:

$$L_j = g(L_{(j-1)}W_j + \boldsymbol{b}_j\mathbf{1}^t)$$

Finally, after k hidden layers, the (multivariate) outcome matrix Y is predicted via

$$Y = L_kW_{k+1} + \boldsymbol{b}_{k+1}\mathbf{1}^t$$

In our situation, because we are predicting a continuous outcome (percentiles) no link function was applied at this final stage. In other applications, such as where we are attempting to classify a binary outcome, this final stage could also include an appropriate link function such as the logit function.

The neural network used in our application had five hidden layers consisting of 1000, 800, 500, 300 and 300 neurons respectively[13]. The same neural network was used to predict percentiles for form X based on relevant inputs as was used to predict percentiles for form Y. As is immediately obvious from the above description, the neural network contains a very large number of parameters[14]. For example, the matrix W_2 connecting the first hidden layer to the second hidden layer contains 800,000 values. As such, it is clear that there is a real danger of over-fitting. To address this problem, rather than attempting to fully optimise the accuracy of predictions in the training set with respect to all the weights and biases, neural networks are trained by sequentially looking at individual cases in the data set and making very slight adjustments to improve accuracy each time. This process is continued, usually

[13] These values were chosen after a limited amount of experimentation with the training data. It is possible that a different structure would yield better performance than that shown (later) in this paper.
[14] In fact it contains a total of just over two million parameters (2,046,699). To calculate this note that there are 574 inputs for each test form (sample size, 561 values from the crosstab, target distribution sample size, and 11 values denoting the estimated distribution of the anchor test in the target population). Thus there are 574x1000 weights plus 1000 biases to estimate the first layer of values (575,000 parameters in total). These calculations can be continued throughout the layers of the network.

until each individual case has been viewed many times[15], until predictive performance against a validation data set (not used to adjust parameter values) stops improving. This process took around 12 hours to complete using a single desktop computer.

One weakness of the standard (or fully connected) neural network above is that it does not make use of information about how the input variables in the training data relate to one another. For example, although most of the information it processes comes from cross-tabulations of values, nothing in the formulae makes use of the fact that, for example, the percentage of people scoring 2 on the anchor test and 10 on form X sits next to the percentage scoring 3 on the anchor test and 10 on form X. In contrast, convolutional neural networks provide an alternative methodology where the structure of the data is hardwired into the setup of the model. As such, they have proved very successful at machine learning problems working with images. Very crudely speaking, they contain a number of convolutional layers where transformations from one layer to the next only combine information across variables that are in the same region – that is close to one another in the image or cross-tabulation. This greatly reduces the number of parameters estimated within each layer, whilst increasing the chance of producing meaningful features. Then, only once the data has been processed through these convolutional layers, a small, fully connected neural network is used to complete predictions. A convolutional neural network (CNN) with 6 hidden convolutional layers and one hidden fully connected layer (with 500 neurons) was also fitted to our data and also took around 12 hours to train.

Note that although the kinds of models described in this paper take many hours to train, this task only needs to be performed once. Once they have been trained, calculating the equating function relating to any particular scenario takes less than a second. As such, the hours for training the models above should be compared to the days, weeks and months it might take a mathematician to come up with new equations for equating and working out how to apply them in practice. When viewed in this light the time to train these models could be seen as very short.

Testing the model

General methodology

Having trained the model it was now of interest to understand whether it would actually yield sensible results in practice. In making this evaluation it was useless to simply check its performance against the same sorts of simulations used in training, as good performance in these scenarios was monitored during training and can be taken for granted. Rather we wish to test the models against new scenarios; either those simulated in an entirely different way, or (even better) from real data sets.

However, whilst not wishing to repeat the same simulations, in doing this we note that the models trained in this paper were designed for situations where we have only moderate amounts of data (100 to 500 examinees), small to medium sized differences in the abilities of the two groups of students (between 0 and 0.5 standard deviations), and reliabilities between 0.7 and 0.9 for each form (i.e. a typical range given the target test length rather than abnormally high or low). As such, the scenarios chosen for testing will be chosen to be broadly within these parameters.

[15] The amount of training where each case in the training data is viewed exactly once is called an epoch.

Within each scenario used for evaluation we will compare a known true equating function to the equating function estimated by the neural networks. The accuracy of these estimates will be compared to the accuracy of the following alternatives:

1) Tucker linear equating
2) Unsmoothed PSE equipercentile equating
3) PSE kernel equipercentile equating using log-linear pre-smoothing[16]
4) PSE kernel equipercentile equating fitted without log-linear pre-smoothing and instead using the bandwidth selection method of Andersson and Von Davier (2014) to introduce smoothing. Weighting the two samples to achieve equivalence will be achieved by using the MDIA technique of Haberman (2015) to ensure that the samples match on the first four moments of the anchor test distribution.
5) Levine observed score linear equating
6) Classical unsmoothed chained equipercentile equating
7) Chained kernel equipercentile equating with log-linear pre-smoothing
8) Chained kernel equipercentile equating without pre-smoothing and instead using the Andersson-Von Davier bandwidth.
9) Hybrid-Levine equating (see Von Davier and Chen, 2013) which combines results from 4 and 5.

Methods 1, 2, 5 and 6 were implemented using the *equate* package in R (Albano, 2016). Methods 3 and 7 were implemented using the *kequate* package (Andersson, Branberg and Wiberg, 2013). Methods 4, 8 and 9 were implemented using a newly developed R package that is available at https://github.com/CambridgeAssessmentResearch/KernEqWPS.

Each of the above methods were compared to the two algorithms developed using machine learning. In addition, because of the practical difficulty in applying convolutional neural networks outside of the machine learning software itself, it was of interest to see whether the trained CNN could be approximated using simple matrix multiplication[17]. This was done by using linear regression to estimate an approximate relationship between the input values and the values stored in the final convolutional layer and then re-estimating the single final layer of the neural network to capture the relationship between the approximate convolution values and the outcomes. Thus, the performance of the nine classical techniques above was compared to the performance of three methods based on machine learning.

The performance of each method was summarised by weighted mean absolute error (WMAE) between estimated and true equating functions within population Q. Weights at each score on form X were defined by the known true distribution of scores within population Q.

Simulated data

Method

As a very simple initial check on the method, it was tested against data sets simulated using the Rasch model. Initially, 50 item difficulties for form X items were simulated from a normal

[16] In this paper all log-linear models were fitted using the first four moments of form X, form Y and the anchor as well as the first two cross-moments between each form and the anchor. This was chosen as it captures something about the shape of the score distributions beyond means and standard deviations as well as the possibility of non-linear relationships between test forms and the anchor. Inspection of AIC statistics within a few of the data sets used for testing revealed little potential for improvement.

[17] CNNs can be represented in the matrix notation introduced earlier. However, it is extremely inefficient as the W_i matrices tend to require very large numbers of columns but have the majority of values equal to zero. As such, it can be difficult to store all of the required parameters in this form.

distribution with a mean of -1 and a standard deviation of 1. Next 50 item difficulties for form Y and 10 for the anchor were simulated from a normal distribution with a mean of 0 and a standard deviation of 1. Abilities for a sample of 400 pupils from each of populations P and Q were simulated from a normal distribution and then used to simulate item scores which were then summed to make total scores on each form. The true equating function within population Q was defined by first calculating the true distributions of each form using the Lord-Wingersky algorithm (Lord and Wingersky, 1984) and then by applying kernel equipercentile equating to these distributions with a low bandwidth of 0.2.

The ability distributions for populations P and Q were chosen to be from one of four conditions

1) Population P ~ N(0.0, 0.7^2), Population Q ~ N(0.0 ,0.7^2)
2) Population P ~ N(0.0, 1.0^2), Population Q ~ N(0.0, 1.0^2)
3) Population P ~ N(0.0, 0.7^2), Population Q ~ N(0.4, 0.7^2)
4) Population P ~ N(0.0, 1.0^2), Population Q ~ N(0.4, 1.0^2)

The standard deviation of 0.7 for populations was chosen as it tended to produce test reliabilities (Cronbach's alpha) form forms X and Y of around 0.8 –the centre of the intended range, whereas the value of 1.0 led to reliabilities at the upper edge. It is expected that the performance of classical techniques such as chained equipercentile equating will improve with this increased reliability. The difference in means between the populations again reflects the centre of the simulated scenarios and a situation closer to the edge.

Twenty-five simulations of item parameters were run for each of the four conditions described above and 100 simulations of examinees were run in each of these leading to a total of 2,500 simulations.

Results

The results of analysis are summarised in Figure 1 and Table 1 below. In all four conditions the methods based on machine learning were amongst the most effective. As shown by Table 1, where there was no difference in ability between the populations being equated, the classical PSE-based methods provided the most accurate results – particularly those using kernel equating. However, aside from linear methods, which suffer due to the generally non-linear shape of the true equating functions, all of the methods displayed similar performance.

For simulated scenarios where there was a difference in the mean abilities of examinees taking different test forms the situation changed. In particular, the performance of PSE-based methods substantially deteriorated and became the worst available. The performance of chained methods also slightly deteriorated. The methods based on CNNs retained their performance and became the most accurate. However, this advantage over chained methods was less prominent where we had high standard deviation of ability (and hence higher reliability). The performance of the standard neural network was slightly worse than the other machine learning methods in condition 3. However, even here its performance was similar to that of chained equipercentile methods.

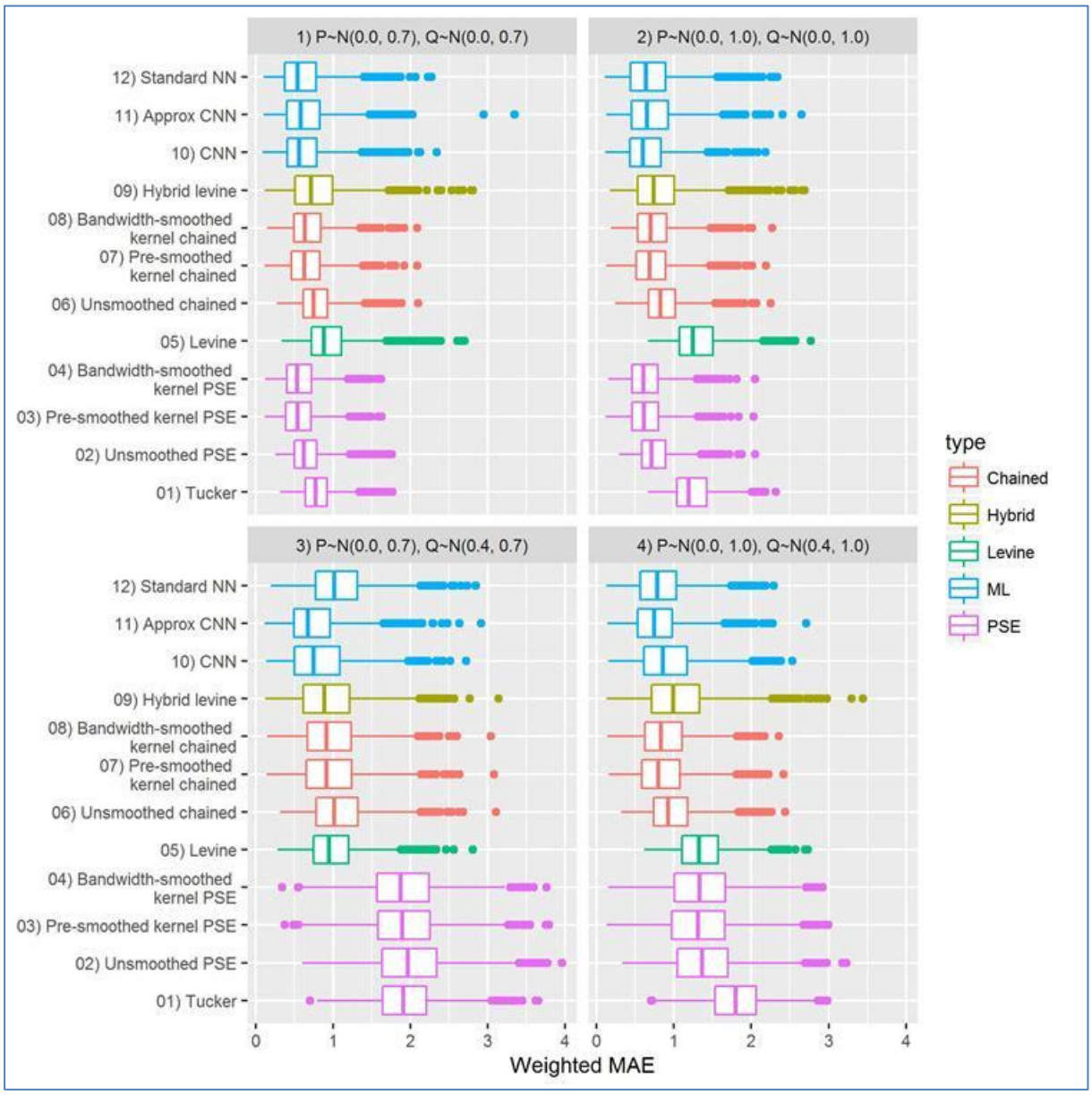

Figure 1: Summary of performance of the different equating techniques in each condition. The top two panels represent conditions with no difference in mean ability between the two populations whereas the bottom panels have a difference of 0.4 between populations. The panels on the left show results for typical reliability and on the right for high reliability.

Table 1: WMAE of equating techniques for data simulated using a Rasch model (lowest WMAE in each scenario is highlighted)

Method	No mean difference between groups		Difference of 0.4 in mean ability between groups	
	SD of ability		SD of ability	
	0.7	1.0	0.7	1.0
01) Tucker	0.80	1.23	1.93	1.80
02) Unsmoothed PSE	0.67	0.76	1.99	1.39
03) Pre-smoothed kernel PSE	0.58	0.65	1.91	1.32
04) Bandwidth-smoothed kernel PSE	0.58	0.65	1.90	1.35
05) Levine	0.94	1.30	1.00	1.35
06) Unsmoothed chained	0.79	0.86	1.08	0.98
07) Pre-smoothed kernel chained	0.67	0.72	0.97	0.85
08) Bandwidth-smoothed kernel chained	0.69	0.74	0.98	0.89
09) Hybrid Levine	0.79	0.80	0.95	1.06
10) CNN	0.63	0.65	0.84	0.92
11) Approx. CNN	0.66	0.72	0.76	0.79
12) Standard NN	0.61	0.70	1.07	0.82

Real data

So far our results have shown methods based on machine learning performing at least as well as the best of the available classical methods. However, it might be hoped that any advantage of the technique will show more clearly when it is applied to real data sets. After all, one of the hoped-for advantages of techniques based on machine learning is that they can be developed by experience of real score distributions rather than on assumptions that score distributions should fit various smooth, parametric distributions such as those suggested by log-linear models.

Method

To test the method, data from 8 science exams taken in 2010 were used for analysis. Data from 2010 was used for analysis as neither the tests themselves nor any subsequent exams designed to the same specification were amongst the 184 score distributions used to train the neural networks. As such, we are completely clear of the accusation of testing the model against data that was used to train it in the first place.

All eight exams had a maximum available score of 100 and all of them were taken by at least 4,500 examinees. In each of the eight cases the items within the test were randomly split into pseudo-tests (von Davier and Chen, 2013) representing form X, form Y and an anchor with maximums of 45, 45, and 10 respectively[18]. Some descriptive statistics from the pseudo-tests are shown in Table 2. Before applying any of the equating techniques, form X and form Y both had five “free” points added to them so that they were out of 50[19] rather than 45.

[18] Note that the tests did not entirely consist of dichotomous items. Maximum scores on items ranged from 1 to 11. Therefore, this step required a solution to a highly simplified version of the famous Knapsack problem. Code to perform this step can be found within the R package at https://github.com/CambridgeAssessmentResearch/KernEqWPS.

[19] Technically, this isn’t necessary as we could simply leave a gap in the score distribution at the top end. However, it was felt that leaving a gap at the bottom led to a more plausible score distribution – particularly since the mean score on each test is comfortably above half marks. At the time of analysis, more refined methods for handling tests with maximum scores other than 50 had not yet been developed.

Table 2: Description of test forms used in real data tests

	Form X			Form Y			Anchor					
Subject	**Number of items**	**Mean**	**SD**	**Number of items**	**Mean**	**SD**	**Number of items**	**Mean**	**SD**	**Cor(X,A)**	**Cor(Y,A)**	**Number of pupils**
Biology 1	19	27.8	6.2	16	24.7	8.3	4	5.5	2.0	0.66	0.67	36,315
Biology 2	16	22.4	6.9	16	24.5	6.8	5	5.8	2.2	0.61	0.65	19,657
Chemistry 1	25	23.9	9.3	25	26.6	8.7	1	5.4	3.1	0.77	0.75	33,120
Chemistry 2	21	23.0	8.4	21	27.2	7.8	5	5.0	2.2	0.68	0.68	6,966
Physics 1	21	20.2	9.0	23	25.0	10.0	4	4.4	2.6	0.74	0.75	15,410
Physics 2	22	22.4	8.2	21	23.2	8.3	4	4.6	2.2	0.63	0.61	7,617
Physics 3	24	23.6	7.9	24	26.5	8.6	5	5.8	2.3	0.62	0.67	8,851
Physics 4	23	27.5	8.4	23	23.6	8.1	6	5.0	2.0	0.63	0.64	4,786

Next, for each test, the full data set was split into two populations (P and Q). This was done using a similar procedure to that described in von Davier and Chen (2013). Specifically, we first calculated the total score across all items for all pupils and then transformed this to have a mean of 0 and a standard deviation of 1. Then each pupil was randomly assigned to either group P or group Q with the probability that they will be assigned to group Q given by the formula below.

$$P(Pupil\ assigned\ to\ group\ Q) = Max\left(0, Min\left(1, 0.5 + Standardised\ Score * \left(\frac{D}{4}\right)\right)\right)$$

where D is the desired number of standard deviations difference between populations. Two sets of groups were created relating to the two conditions D=0.0 and D=0.4.

Having assigned pupils to populations P and Q, the true equating function in group Q was defined using kernel equating within this group using a small bandwidth derived using the formula from Andersson and von Davier (2014) applied to the full data set[20]. Now for each of the eight exams, and for each of the 2 conditions, two hundred samples of 400 examinees were selected from both populations and each of the different equating techniques was applied to establish their accuracy based on each sample.

Results

The results of analysis are summarised in Figure 2 as well as Tables 3 and 4. As with the simulated data, when there was no difference in mean achievement between the two populations the PSE-based methods performed very well, although there was little difference in accuracy between all of the various methods. It is worth noting that the machine learning-based methods outperformed those based on chained equating in these scenarios.

The Figure also shows that when there is a difference in population means, methods-based on machine learning tended to perform better than any other available methods. This is shown further in Figure 3 which splits the results for the difference of 0.4 standard deviations between populations by examination. In all eight cases, machine learning provided the most accurate estimates of the true equating function. In fact, as shown by Table 4, in all eight cases the three techniques based on machine learning provided the three most accurate estimates with one exception – Levine equating (and Hybrid Levine equating) marginally outperformed some of the machine learning techniques for Chemistry 1.

[20] Because the sample sizes included in this step are large (see table 2) this will lead to fairly small bandwidths being selected at this stage.

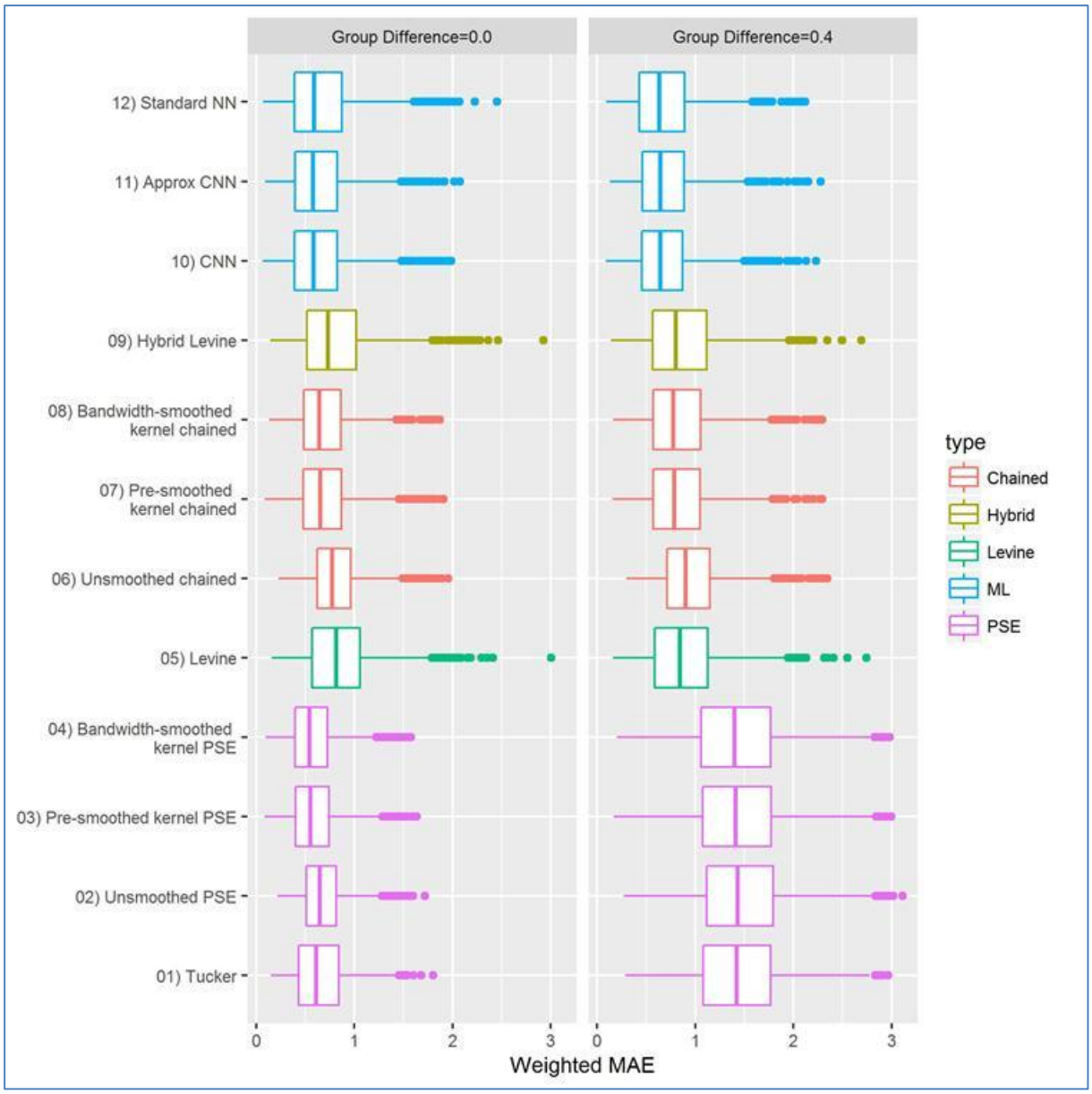

Figure 2: Summary of performance of the different equating techniques with real data. The left panel represents the condition with no difference in mean ability between the two populations whereas the right panel is for a difference of 0.4 standard deviation between populations.

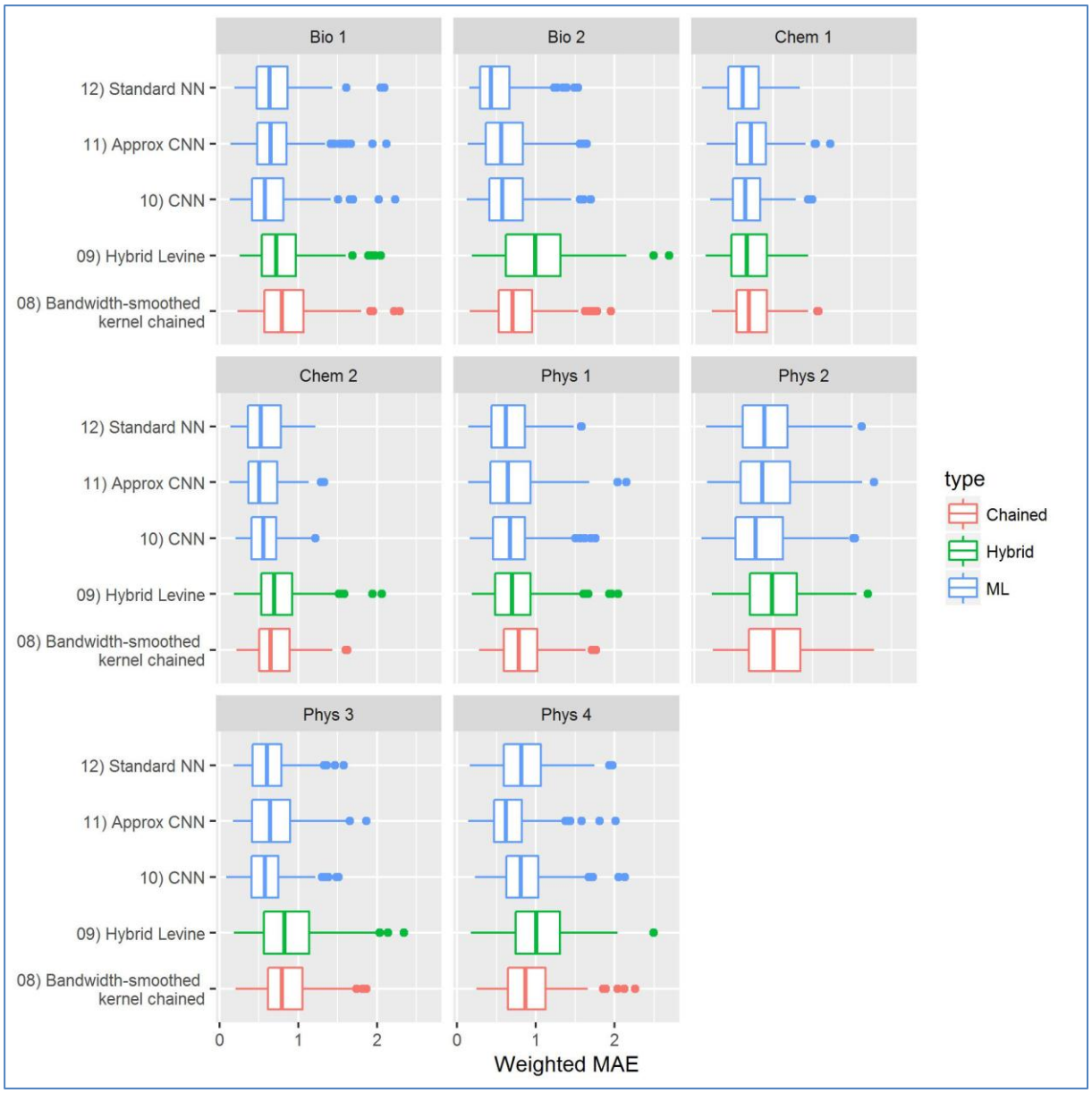

Figure 3: Summary of performance of the different equating techniques with real data where the mean difference in ability between the two populations was 0.4 standard deviations.

Table 3: WMAE of equating techniques with real data when there is no difference in mean ability between populations (lowest WMAE in each scenario is highlighted)

	Subject							
Method	**Bio 1**	**Bio 2**	**Chem 1**	**Chem 2**	**Phys 1**	**Phys 2**	**Phys 3**	**Phys 4**
01) Tucker	0.59	0.47	0.61	0.46	1.03	0.53	0.68	0.78
02) Unsmoothed PSE	0.72	0.58	0.68	0.64	0.78	0.70	0.72	0.66
03) Pre-smoothed kernel PSE	0.63	0.50	0.58	0.53	0.67	0.62	0.62	0.58
04) Bandwidth-smoothed kernel PSE	0.62	0.50	0.57	0.52	0.65	0.59	0.62	0.58
05) Levine	0.78	0.65	0.69	0.69	1.09	0.97	0.86	1.05
06) Unsmoothed chained	0.83	0.69	0.79	0.77	0.90	0.92	0.85	0.80
07) Pre-smoothed kernel chained	0.71	0.59	0.67	0.63	0.77	0.81	0.72	0.69
08) Bandwidth-smoothed kernel chained	0.73	0.58	0.65	0.64	0.75	0.80	0.73	0.70
09) Hybrid Levine	0.80	0.68	0.66	0.73	0.81	1.02	0.83	0.91
10) CNN	0.56	0.55	0.57	0.62	0.81	0.66	0.67	0.69
11) Approx. CNN	0.63	0.56	0.62	0.56	0.74	0.65	0.73	0.70
12) Standard NN	0.65	0.57	0.70	0.55	0.84	0.65	0.75	0.67

Table 4: WMAE of equating techniques with real data when there is a difference of 0.4 standard deviations in mean total achievement between populations (lowest WMAE in each scenario is highlighted)

	Subject							
Method	**Bio 1**	**Bio 2**	**Chem 1**	**Chem 2**	**Phys 1**	**Phys 2**	**Phys 3**	**Phys 4**
01) Tucker	1.43	1.37	1.17	1.23	1.42	1.70	1.42	1.80
02) Unsmoothed PSE	1.49	1.39	1.23	1.28	1.41	1.72	1.41	1.84
03) Pre-smoothed kernel PSE	1.45	1.33	1.22	1.25	1.35	1.75	1.36	1.75
04) Bandwidth-smoothed kernel PSE	1.44	1.32	1.21	1.24	1.33	1.75	1.35	1.75
05) Levine	0.77	0.96	0.65	0.68	1.17	0.95	0.94	0.99
06) Unsmoothed chained	0.97	0.88	0.88	0.83	0.97	1.14	0.98	1.02
07) Pre-smoothed kernel chained	0.85	0.77	0.76	0.71	0.84	1.03	0.88	0.90
08) Bandwidth-smoothed kernel chained	0.86	0.78	0.74	0.71	0.83	1.04	0.86	0.92
09) Hybrid Levine	0.79	1.03	0.71	0.75	0.76	1.01	0.89	1.04
10) CNN	0.64	0.64	0.67	0.59	0.71	0.84	0.62	0.86
11) Approx. CNN	0.72	0.63	0.74	0.56	0.71	0.93	0.70	0.68
12) Standard NN	0.69	0.52	0.63	0.58	0.68	0.92	0.63	0.85

It would be unusual to complete a paper on a new equating technique without showing some actual equating lines. It is also usual for the bias and root mean square error of equating (RMSE) to be displayed across the score range rather than simply averaged as has been done above. This additional information is shown for Biology 1 in Figure 4 – chosen as a representative example of the results. In order to prevent the chart becoming overcrowded only the results from the CNN, Levine equating and kernel chained equating are shown. As can be seen, the superior performance of the CNN resulted from it having lower bias across the score range than chained equating, and having a lower RMSE than Levine equating – particularly towards the ends of the distribution where the equating function was not well approximated by a straight line.

Although the main focus within our results has been upon which equating technique provides the most accurate results, it is worth also noticing the scale of differences. For example, in Table 4 we might note that the difference in WMAE between the best performing method and the worst performing method is generally less than 1 score point.

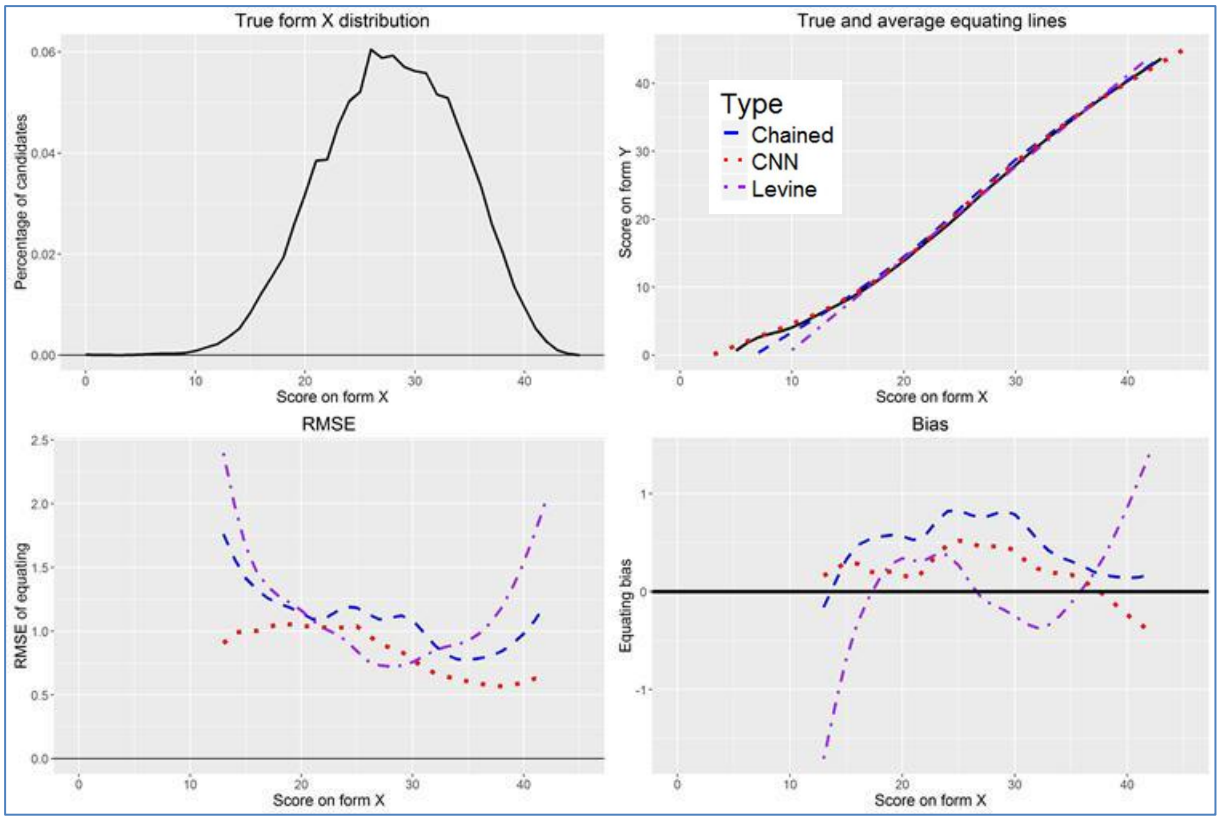

Figure 4: Form X distribution[21], average equating lines, RMSE and bias across the score range for Biology 1.

Discussion

The purpose of this research was to ascertain whether machine learning could help to identify potentially more accurate methods of equating. The results of analysis have undoubtedly shown that this is possible. Across a range of scenarios, techniques based upon machine learning have performed as well or better than the best available classical equating techniques. Given that the majority of classical equating techniques struggle when we have differences in ability between the groups taking different test forms, this indicates that machine learning could be a fruitful avenue for further development. The specific technique developed in this research could be useful where we have an external anchor test, and moderate sized differences in ability between populations. However, the technique used to train equating algorithms could potentially lead to methods that could be used more widely.

At the time of analysis, the machine learning methods above were only designed to deal with test forms and an anchor of a particular length. After all, there seemed little point in developing the functionality to deal with any form of input before knowing whether the techniques yield improved accuracy at all. More recently, having found the techniques to be effective, further work has been done on this issue. If the maximum number of marks is below 50 on either form (or below 10 for the anchor) the easiest fix is simply to leave the remaining scores having zero probability within the distribution or to add “free” points to bring the maximum up to the desired total. However, such an approach will not be effective if the maximum score available is greater than 50. An experimental method to deal with these situations has been developed. It works as follows:

[21] The distribution here is shown out of 45. Five marks were subtracted from both the form X scores and the equated form Y scores to create these charts on the scale of the original pseudo tests.

- Convert the anchor test to be out of 10 by splitting scores for the combined set of examinees (across samples from both P and Q) into elevenths. These scores are then used to replace the original anchor[22].
- Now for each anchor score separately create a continuized version of the distribution on the relevant form using principles from kernel equating with a low bandwidth.
- Using this distribution divide the continuized total score distribution into 51 equal intervals and calculate the percentage of examinees in each section. This can be used to fill in the relevant row of the anchor by test form cross-tabulation.
- Now, apply the equating algorithms derived using machine learning as usual.
- Now transform the equated values back to the original scale. For example, if the original test Y scores were out of 80, the final equated scores will need to be multiplied by 8/5.

Testing of this procedure is at an early stage and so accuracy cannot be guaranteed. Indeed, there must surely come a point at which the algorithm is no longer sensible. However, early testing (using simulations based on Rasch models) suggests that this technique retains the majority of the good performance of these new techniques. The capability to apply these experimental techniques with tests of different lengths is included within the R package at https://github.com/CambridgeAssessmentResearch/KernEqWPS.

It should be noted, that the algorithms developed in this research needn't be the most accurate possible. Indeed, various potentially effective strategies such as using the results of various classical equating techniques, amongst the inputs to a neural network have deliberately not been considered[23]. Further research could explore these, and other, possibilities to investigate whether increased accuracy is possible.

References

Anthony Albano (2016). *equate: Observed-Score Linking and Equating. R package version 2.0-4*. http://CRAN.R-project.org/package=equate.

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

Andersson, B., and von Davier, A.A. (2014). Improving the bandwidth selection in kernel equating. *Journal of Educational Measurement*, *51*(3), 223-238.

Bramley, T., and. Dhawan, V. (2012) *Estimates of reliability of qualifications*. Chap. 7 in Ofqual's Reliability Compendium, edited by D. Opposs and Q. He, 217-320. Coventry: Ofqual/12/5117.

Haberman, S.J. (2015). Pseudo-equivalent groups and linking. *Journal of Educational and Behavioral Statistics*, 40(3), 254-273.

Kolen, M.J., and Brennan, R.L. (2004). *Test Equating, Scaling, and Linking: Methods and Practices.* (2nd ed.). New York: Springer.

[22] Almost no useful information will be lost in this step. For example, the Spearman correlation between a normally distributed variable and its categorization into elevenths is in excess of 0.99.

[23] This researcher was interested in whether artificial intelligence could learn equating techniques for itself without being given a helping hand towards existing methodologies.

Levine, R. (1955). Equating the score scales of alternate forms administered to samples of different ability. *ETS Research Report Series*.

Lord, F.M., and Wingersky, M.S. (1984). Comparison of IRT true-score and equipercentile observed-score “equatings”. *Applied Psychological Measurement*, *8*, 453–461.

von Davier, A.A., Holland, P.W., and Thayer, D.T. (2003). *The kernel method of test equating*. Springer Science & Business Media.

von Davier, A. A., and Chen, H. (2013). *The Kernel Levine Equipercentile Observed-Score Equating Function*. Research Report. ETS RR-13-38. ETS Research Report Series.

Wang, T. and Brennan, R. (2009). A Modified Frequency Estimation Equating Method for the Common-Item Nonequivalent Groups Design. *Applied Psychological Measurement*, *33*, 118-132.