# Structure

## Introduction

### Background

In cost-effectiveness models, it is highly recommended that probabilistic sensitivity analysis (PSA) is performed. (Claxton et al., 2005; NICE, 2008) This requires producing numerous samples of parameter values used in the model, whose variability is intended to reflect uncertainty in the true value of those parameters. In many cases, the values of two or more parameters are known to be related to each other in some way, and this relationship also needs to be reflected appropriately in the PSA.

One type of relationship that could exist is that two or more variables are monotonically related. By this we mean that when considering two variables X and Y, though we are uncertain about the true value of X, and uncertain about the true value of Y, we are certain that Y is greater than X. A common example of this would be where a disease has a less severe state, and a more severe state, and it would be clinically implausible to assume that the mean health-related quality of life (HRQoL) while in the less severe state is lower than in the more severe state.

In this paper, we compare ten different methods for jointly simulating the PSA of two variables that we assume to be monotonically related. These ten methods fit broadly into one of four classes of method:

1. **Naïve methods** (methods one and two), where the two variables are sampled independently, or the same random number stream is used for simulating both variables;
2. **Resampling and replacement methods** (methods three, four, five and six), where draws from independent distributions are either selectively resampled or replaced with other draws;
3. **Multivariate model methods** (methods seven, eight and nine), where the variables used in the PSA are sampled jointly from a multivariate model where a covariance between variables is explicitly specified;
4. **Difference model methods** (method ten), where PSA draws for all but one of the variables are produced by adding a draw from a positively bounded distribution onto a draw for another distribution

In this paper we compare the properties of PSA samples created by each of the ten methods. All methods use only summary statistics, sample means and standard errors, which are often the only data available to modellers. In our comparison, the summary means and standard errors are derived from hypothetical individual patient data reporting the HRQoL for thirty patients with a hypothetical disease which could either be in a moderate state or a severe state. For each patient, we have produced values for their HRQoL in the moderate state, and also their HRQoL in the severe state. From the individual patient data (IPD) we produce 1,000 joint estimates of the mean HRQoL in the moderate state and in the severe state using a bootstrapping procedure. These bootstrapped estimates, based directly on the IPD, are the gold standard against which the estimates produced by each of the methods, which use only summary data, are compared. In general, we consider methods which produce PSA samples most similar to the bootstrapped estimates to be preferable to those which produce dissimilar PSA samples.

The inspiration for this paper was that we have observed authors of economic evaluations using naive and resampling approaches, which we believe are inadequate for handing monotonicity in this context.

## Method

### Simulated Data

Our data is of thirty hypothetical patients who progress from a moderate disease state (Stage 1) to a more severe disease state (Stage 2). Each individual’s HRQoL in the less severe disease state (U1) and the mere severe disease state (U2) is reported. The individual patient data (IPD) are shown in the appendix in Table 1, and the corresponding scatter plot for these data are shown in Figure 1.

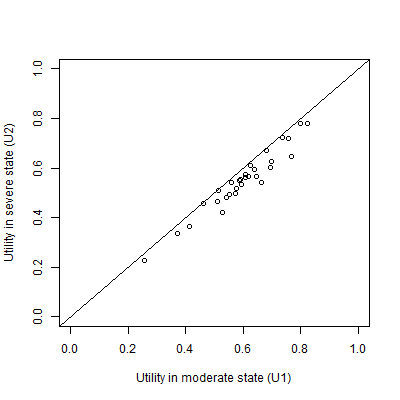


Figure A plot of the simulated individual patient data

### Bootstrapped estimates of means

As modellers are typically interested in representing uncertainty in expected values (uncertainty in the means) rather than predicted values (uncertainty and variability in the range of values encountered), ‘true’ uncertainty in the mean values of U1 and U2 was estimated by repeatedly resampling the IPD, and for each resample calculating the mean values of U1 and U2 produced. Doing this 1,000 times produced the data shown in Figure 2. This approach illustrates what the modellers would be able to produce for the PSA if they had access to the IPD, and so represents the ‘gold standard’ against which the other methods, which have access only to aggregate level data, are compared.

We can see that the two parameters are monotonically related, as no estimate of U1 is less than the corresponding estimate of U2, and so no value crosses the diagonal line. We can also see that though the two means are strongly but not perfectly correlated (r= 0.97). Because of this, there is some variability in the differences between the two estimates, U1 – U2. This shows that simply adding E(U1) – E(U2), i.e. 0.60 – 0.55 = 0.05, onto the PSA estimates of U2 to producing corresponding PSA estimates of U1 would not be correct, as it would not accurately represent the uncertainty in the differences between U1 and U2.

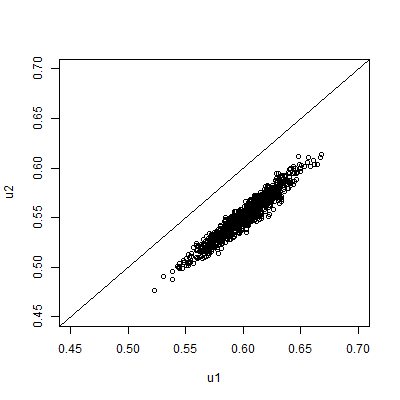


Figure 2 Scatterplot of 1000 PSA draws of the joint means of U1 and U2 produced by bootstrapping the IPD in table 1

### Summary statistics

In our hypothetical example, we assume the modeller will not have access to the IPD, but only to the summary information shown in Table 2. This summary information, together with the knowledge that U2 should be less than U1, is the only information used in each of the ten approaches described below.

|  |  |  |
| --- | --- | --- |
|  | **U1** | **U2** |
| Mean | 0.600 | 0.550 |
| 95% confidence interval of mean | 0.555 to 0.644 | 0.506 to 0.594 |

Table 2 The assumed available summary data. This is assumed to be the only information available to the modeller

### The Montonicity Assumption

When modellers are generating multiple estimates for use within PSA using these summary data, the key monotonicity condition that must hold is that an estimate of U2 should always be equal to or lower than a corresponding estimate for U1. More formally, if there are M runs within the PSA, and the subscript i defines predicted values from the ith run, then U1i ≥ U2i for all i ; where M is the total number of PSA samples. If monotonicity were violated then some of the estimated values of U1 - U2 produced from the PSA would be negative.

### The Ten Methods

The ten methods considered are described in Table 2. Within all approaches we make the simplifying assumption that the summary statistics above relate to normal distributions, rather than other distributions which are constrained to produce utility values within a plausible range, such as 0 to 1 if assuming no worse-than-death health states. This seems reasonable given the extremely small probability that a sampled value of either U1 or U2 would be greater than 1 or less than 0.

All methods were implemented using the R programming language. (R Development Core Team, 2011) The R code, including annotations, is presented in the appendix below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Method Number** | **Name** | **Method Description** |
| Naïve Methods | 1 | Independent Sampling | For each of the PSA runs, take one draw from U1 and one draw from U2 independently (i.e. assume no covariance between U1 and U2) |
| 2 | Quantile Matching/  Number Seed Recycling | For each of the PSA runs, use the same random number seed when drawing a sample from U2 and U1. (This is equivalent to selecting the same quantile from both distributions.) |
| Resampling and replacement methods | 3 | Upward Replacement | For each of the PSA runs:  Stage 1: draw a sample from U2  Stage 2: draw a sample from U1  Stage 3: Check if the value of U1 drawn is less than the corresponding value of U2 drawn. If it is, then replace the value of U1 with the U2 value. |
| 4 | Downward Replacement | For each of the PSA runs:  Stage 1: draw a sample from U1  Stage 2: draw a sample from U2.  Stage 3: Check if the value of U2 drawn is greater than the corresponding value of U1 drawn. If it is, then replace the value of U2 with the U1 value. |
| 5 | Upward Resampling | For each of the PSA runs:  Stage 1: draw first from U1.  Stage 2: draw from U2.  Stage 3: Check if the value of U1 is less than U2. If it is, then go back to Stage 2 (i.e. resample). If not, then stop. |
| 6 | Downward Resampling | For each of the PSA runs:  Stage 1: draw first from U2.  Stage 2: draw from U1.  Stage 3: Check if the value of U2 is greater than U1. If it is, then go back to Stage 2 (i.e. resample). If not, then stop. |
| Multivariate model methods | 7 | AIVM Covariance | Assume that the covariance between U1 and U2 is equal to the average of the individual variances of the means (AIVM) of U1 and U2.  If assuming this covariance implies that the correlation between U1 and U2 is greater than 1, then instead select the covariance between U1 and U2 which implies a correlation of 1 between U1 and U2. |
| 8 | Lower Bounded Covariance Retrofitting | Select the minimum value of a covariance between U1 and U2 such that the two following conditions are met:  Condition 1: U1 – U2 > 0 for all PSA runs.  Condition 2: The covariance between U1 and U2 is greater than AIVM.  If this implies that the correlation between U1 and U2 is greater than 1, then instead use the covariance value associated with a correlation of 1. |
| 9 | Upper Bounded  Covariance Retrofitting | Methodology 8 but where the second condition is that the covariance between U1 and U2 is less than AIVM. |
| Difference model methods | 10 | Beta Distribution Difference Modelling | Use a derived distribution of U1, called U1(\*), rather than U1 itself.  U1(\*) is defined as equal to U2 + Δ, where Δ is drawn from a Beta distribution. The parameters of the Beta distribution are selected so as to minimise the differences between U1(\*) and U1. |

Table 3 Summary of the ten approaches considered

### Naïve methods

Methods one and two are both simple. Method one, independent sampling, is the simplest method of all, and does not take the monotonicity condition into account at all. Nevertheless, in cases where the means of U1 and U2 are far apart and the standard errors of both parameters are small, this method may still produce PSA values which do not violate the monotonicity assumption. With the data considered here, however, this is not the case, and so the approach is liable to produce erroneous samples. Method two has been observed in economic evaluations, and involves using the same random number when drawing from both the U1 and U2 distributions. Method two is broadly equivalent to pairing the quantiles from the estimated distributions of U1 and U2 within PSA runs, matching the lowest estimate of U1 with the lowest estimate of U2, the second lowest estimate of U1 with the second lowest estimate of U2, and so on. For this reason, quantile-pairing was not considered as a separate strategy.

### Resampling and replacement methods

Methods three, four, five and six have also been observed in economic models, as they are relatively simple to implement. All four methods involve sampling one of the two paired values, U1i or U2i, independently, before sampling the value, U2i or U1i. For methods 3 and 4, the second value is then replaced with the first value if it violates the monotonicity assumption. For methods five and six, the second value is retained if it does not violate the monotonicity assumption, and resampled if it does violate the assumption. The second value is resampled until a value which does not violate the monotonicity assumption is drawn.

There are theoretical reasons to be concerned with both the resampling and replacement methods. The replacement methods can be assumed to produce biased estimates of the mean value. Any systematic increase (or decrease) in the sample value will result in the average of 1,000 being greater than (or lower than) the true distribution mean. This phenomenon occurs independently of whether the value is set equal to the previously sampled parameter value, or whether it is resampled until monotonicity is upheld, although the bias will be less in the former methodologies. Despite the known bias these methods have been included to prove this for the novice reader.

### Multivariate model methods

Methods seven, eight and nine each involve selecting covariances on the basis either of the variances presented in the summary statistics for U1 and U2, or on whether monotonicity is maintained on all runs of the PSA. Method seven involves setting the covariance between U1 and U2 to the average of the individual variances of the means (AIVM). Method eight involves setting the covariance to such a value that no PSA draws violate the monotonicity assumption, subject to the constraint that the covariance is also greater than the AIVM. For method nine, the covariance is also set such that no PSA draws violate the monotonicity assumption, but this time subject to the constraint that the covariance is also less than the AIVM.

A further logical constraint also applies to all three methods. This is that the covariances cannot imply a correlation of greater than 1. The correlation of two random variables X and Y is defined as follows:

For this reason, the upper limit of the covariance must be . For approach seven, this effectively states that the covariance selected is:

This constraint also places an upper limit on the range of covariances which may be considered in methods eight and nine.

The R code used to implement methods seven, eight and nine is presented in the appendix.

### Difference model methods

Within method ten, instead of the independent distribution U1 being used directly in the PSA, it is used indirectly, in order to produce an alternative sampling distribution . This alternative distribution should produce a similar distribution of values to U1, without biased means or standard deviations, but also should ensure strict monotonicity. We do this by defining:

Because Δ is drawn from a Beta distribution, which can only produce values between 0 and 1 inclusive, we know that the monotonicity condition is satisfied because Δ would have to be negative in order for to be less than U2.

In order to ensure that has the same mean value as U1, we define:

If we define , then and . We use a numerical optimisation algorithm which searches for the optimal value of N, , subject to the above constraints, such that the root mean square (RMS) of the differences in means and standard deviation between U1 and is minimised. I.e. selecting a value of N which satisfies the following condition:

)

Where is the sample mean of U1, is the standard error of U1, and and are the corresponding quantities for U1\*. The R code used to perform this optimisation is presented in the appendix.

### Methods where monotonicity cannot be violated

For some of the methods, it is analytically impossible for monotonicity to be violated, and so they must satisfy the monotonicity condition. These methods are three, four, five, six and ten. For methods seven, eight and nine, which use algorithms to select covariances between parameters, it is possible that for some runs monotonicity may be violated. Where violation of monotonicity is possible, modellers should be able to specify what level of monotonicity violation is tolerable. For example, monotonicity violation may be acceptable, so long as it occurs with a frequency of less than 1/10,000. For brevity, methods three, four, five, six, and ten will be described as satisfying ‘strict monotonicity’; whereas methods seven, eight and nine will be described as satisfying ‘relaxed monotonicity’.

### Comparing between methods

We use two visual approaches to compare the ten methods with each other, and with the bootstrapped estimates based on the IPD. In all cases, the closer the output from a method is to the bootstrapped estimates, the better it is at accurately representing the relationship between U1 and U2 given only summary data.

Firstly, we produce scatterplots of 1,000 joint estimates of U1 and U2 for each of the ten methods. These are drawn on the same scale as the scatterplot shown in Figure 2, and so the joint patterns of scatter produced by each method can easily be compared with Figure 2.

Secondly, we use violin plots to compare the distribution of the quantities U1, U2, and U1 - U2 for each of the ten methods with the bootstrapped estimates. This comparison is facilitated by using violin plots, which are similar to box plots but also present kernel density estimates of distributions of the type presented in Figure 3. (Hintze & Nelson, 1998) An appropriate method for representing the monotonic relationship given only the summary data should produce distributions for these quantities which look similar to the bootstrapped values for each of these three quantities.

## Results

### Parameterisation of methods seven, eight, and nine

For method seven, the variance associated with the standard errors of both U1 and U2 are 0.000504 to three significant figures, and so the AIVM is also 0.000504. The product of the standard errors of U1 and U2 is also 0.000504 to this many significant figures. This product defines the covariance which implies a correlation of 1, and so the upper bound of the covariance that can be set. This means that method seven is equivalent to setting the correlation between the means of U1 and U2 to 1. So, in this example method seven and method eight are identical.

For method eight, which forces a covariance of 0.00504, implying a correlation of 1, was also identified, as method eight uses the covariance value from method seven as its upper bound, and as this value already implies a correlation of 1 it cannot be any higher. For method nine, however, which uses a covariance of 0 as its lower bound, a covariance of 0.000380 was identified, implying a correlation of 0.754.

### Parameterisation of method ten

The optimisation routine selected an N value of 1925.72, producing Beta parameters a = 96.86, and b =1828.87. Figure 3 below shows the distribution of 1000 draws from U1\* alongside 1000 draws of U1 and U2. We see that the distribution of U1\* closely matches that of U1. The mean and standard deviation of U1 and U1\* were both identical to two decimal places, with a mean of 0.60 and a standard deviation of 0.02.

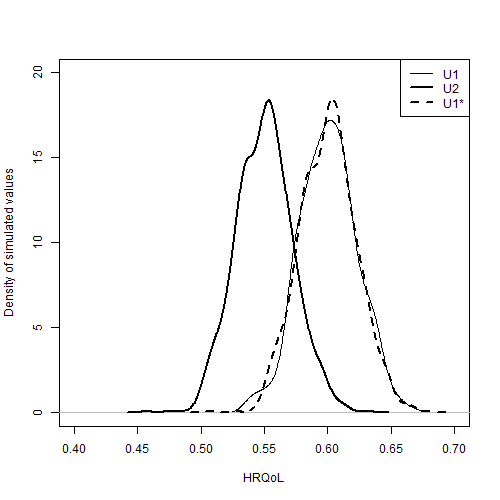


Figure Comparison of the distribution of estimates of U2, U1, and U1\* produced using Method 10

### Scatterplots

Figure 4 shows scatterplots of U2 against U1 for each of the ten methods. Because methods seven and eight are equivalent in for these data, they are presented as a single subplot (g). The diagonal line indicates parity between corresponding draws of U1 and U2. Scatter above this diagonal line shows that some proportion of the draws produced by the method violate the monotonicity assumption. A good method should be able to produce a similar pattern of scatter given the aggregate data as the bootstrapped method is able to produce using the IPD. Each of the subfigures should be compared with the scatterplot of the bootstrapped means, shown in Figure 2.

Figure 4a shows the scatterplot for method one. This shows some scatter above the diagonal line, showing that some proportion of the draws violates the monotonicity assumption, highlighting the inadequacy of the approach. All other approaches appear to produce no estimates which violate the monotonicity assumption.

Methods three (Figure 4c), four (Figure 4d), five (Figure 4e) and six (Figure 4f) all show nonlinearities in the scatter, with no values above the diagonal line but relatively high densities of values just below the diagonal line. These discontinuities suggest that the methods of ensuring monotonicity is liable to produce biases in the estimated mean values.

The majority of the approaches appear to produce patterns of variance in the scatter which are qualitatively dissimilar to the bootstrapped scatter. Methods one (Figure 4a), three (Figure 4c), four(Figure 4d), five (Figure 4e), and six (Figure 4f) all produce uncorrelated scatter that is too wide, indicating the correlation of the PSA estimates is too low. By contrast, methods two (Figure 4e), seven and eight (Figure 4g) produce scatter which is too narrow, indicating the correlation estimates are too high.

We see from the scatter that method ten (Figure 4i) and method nine (Figure 4h) are both closest in appearance to the bootstrapped scatter, with method 10 exhibiting closer values than method 9.

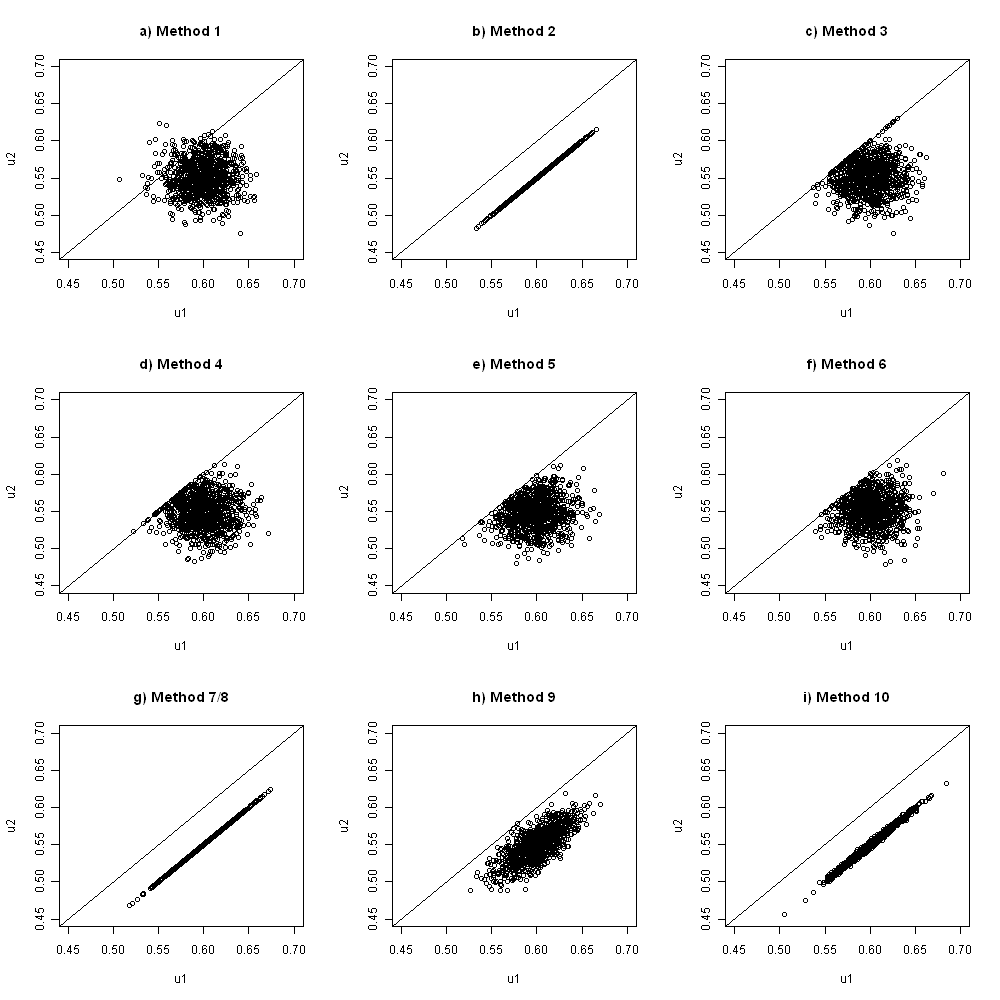


Figure Scatterplots of U1 against U2 for each of the methods

### Monotonicity violation

The two replacement methods (three and four), the two resampling methods (five and six), and the Beta method (method ten) are all designed such that it is analytically impossible for them to violate the monotonicity assumption. The other methods could all potentially violate monotonicity. In this example, the only approach where monotonicity is violated in the 1000 PSA samples being compared is method one, independent sampling, where 53 out of the 1000 PSA samples violated monotonicity. The precise proportion of samples violating monotonicity will differ slightly each time PSA is performed, due to stochastic uncertainty.

### Comparing U1, U2 and U1-U2

This section will compare the distribution of values of the quantity U1 – U2, for each of the ten methods, compared with the gold standard, the bootstrapped data. Like the scatterplots shown in Figure 2 and Figure 4, they therefore allow nuanced comparisons between the distributions to be made. Figure 6 and Figure 7, in the appendix, show violin plots for the distributions of U1 and U2.

Figure 5 shows the distribution of U1 – U2, i.e. the differences in paired draws of U1 and U2, produced by each of the ten methods, compared with bootstrapped estimates of this quantity. As shown in Figure 4, we see clearly that method one, independent sampling, producing some estimates where monotonicity is violated, because some of the distribution of values is below the 0. Two other types of problem are also observed.

Methods two, seven and eight all severely underestimate the uncertainty in this quantity, as all estimates U1 – U2 are identical. This is because in this case methods seven and eight both explicitly assume a perfect correlation between U1 and U2, and because the standard errors of both U1 and U2 are the same method 2 effectively does the same as well.

Methods three, four, five and six all show the second type of problem, in that they introduce a discontinuity in the at the lower end (U1 – U2 = 0), while also having too wide a distribution at the upper end. Based on this measure, only methods nine and ten appear broadly appropriate in terms of representing this form of uncertainty. Of these two methods, method ten produces the distribution of values which is closest to the bootstrapped distribution.

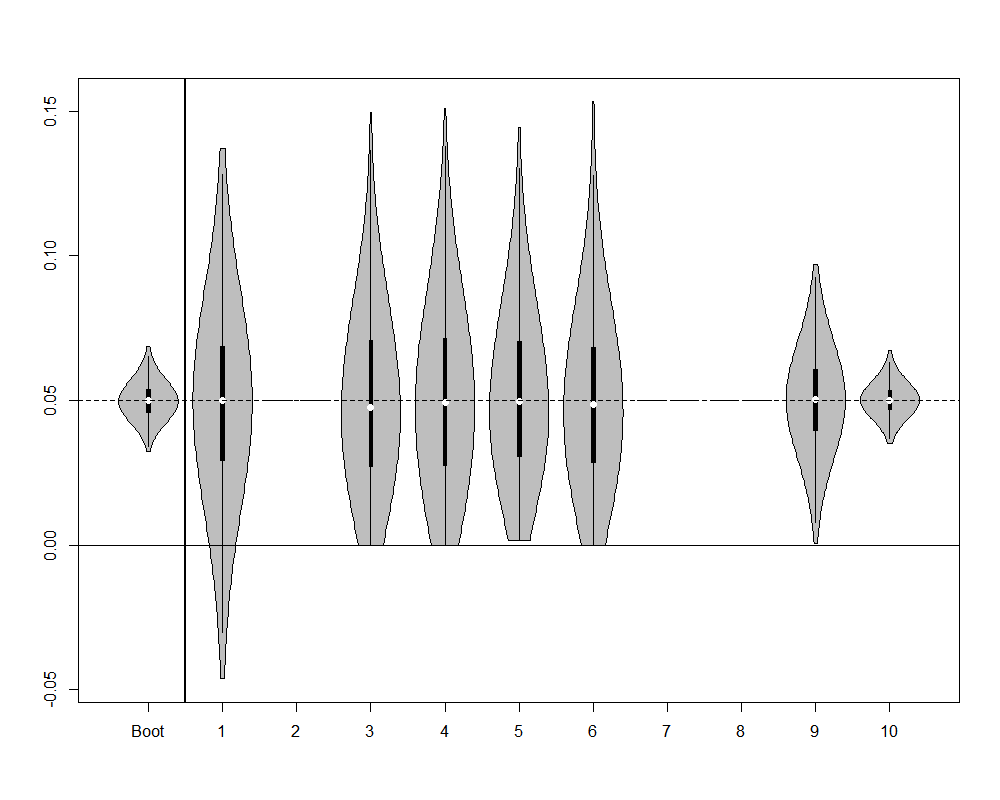


Figure Violin plot of distribution of U1 - U2 estimates, for each of the 10 methods, compared with bootstrapped estimates. The white dots indicate the sample means. The thick vertical dark black lines indicate the interquartile range. The thin vertical black lines indicate the 95% intervals. The density estimates for methods two, seven and eight are not missing, but all estimates of the quantity U1 – U2 are identical.

## Discussion

### Findings

This paper has compared ten methods which may be used to handle the monotonicity assumption within PSA, against a ‘gold standard’ of bootstrapped estimates of hypothetical IPD. It confirmed that independent sampling is liable to produce violations of the monotonicity assumption, and so should not be adopted where it is important to incorporate this assumption within the PSA estimates. It also found that a number of other commonly used methods for incorporating the monotonicity can effectively discard or misrepresent an important form of uncertainty: i.e. uncertainty about the difference between U1 and U2. Some of these methods (two, seven, and eight) effectively ignore this form of uncertainty, doing the equivalent of assuming perfect correlation between the two quantities, and so implicitly that the two parameters are really one parameter that is repeated as a linear transformation. Even where the correlation assumed between variables is not 100%, in principle this issue of underestimating uncertainty remains. Methods three, four, five, and six introduce implausible discontinuities into the distribution of differences between values; there are also theoretical reasons to assume that these methods will produce biased means, although such biases were not overly apparent in our results.

Method ten is superior to other methods in appropriately representing the distribution of differences between U1 and U2, i.e. the quantity U1 – U2. It also does not produce biased estimates or introduce implausible discontinuities into the distributions.

### Limitations

This section will describe some limitations with the current analysis. This includes: not using Beta distributions to represent the U1 and U2 distributions for most methods; not looking at results for a range of hypothetical datasets; not presenting a hypothetical example with three or more states; and not looking at how dependent the results from the covariance-based methods are the size of the ‘training’ sets. Each of these limitations will now be discussed in more detail.

The first limitation is that we did not use Beta distributions to represent the quantities U1 and U2, and instead used Normal distributions. Because the Normal distribution produces values between - ∞ and + ∞, but utility values are more likely to be bounded between 0 and 1, the Normal distribution is not the most suitable distribution in theory, whereas the Beta distribution may be more appropriate. The implications for research section discusses how the parameters for the Beta distribution can be derived when only the sample means and standard errors are reported, which would be required if these analyses were to be conducted using Beta distributions rather than Normal distributions.

The second limitation is that we did not look at results for a range of different hypothetical datasets with different individual level and summary characteristics. For example, in our hypothetical IPD the standard error of U1 and the standard error of U2 are approximately the same, and this factor may have affected the results comparing each of the methods.

A third limitation, related to the second limitation, is that our hypothetical dataset have only two disease severity states, U1 and U2, rather than three states such as U0, U1, and U2, where the HRQoL of U0 is expected to be greater than for U1, and U1 to be greater than for U2. Introducing further states would lead to complications for methods seven, eight and nine, for example, as we would have to make decisions about the covariance between parameters U0 and U2, as well as between U0 and U1, and between U1 and U2.

A final limitation we have identified relates to how methods eight and nine have been implemented. Both of these involve choosing covariance parameters conditional of whether any pair of values in a ‘training’ sample of 1,000 draws violates the monotonicity assumption. As the size of the ‘training’ sample increases, the probability of extreme values, including values which violate monotonicity, increases, and so we should expect the covariance selected to depend partly on the size of the training sample used.

### Implications for Research

Further research should explore how dependent our results and conclusions are on the simplifying assumptions we have made that the summary statistics relate to normal distributions. For example, if we assume that all HRQoL values have to be within the range 0 to 1, then the Beta distribution may be a more appropriate way of representing the distribution of values. Given summary statistics reporting a mean μ and a standard error σ, the Beta parameters a and b can be derived as

Although using Beta distributions with this reparameterisation would have been simple to adopt for some of the methods, such as method one, for other methods, such as those involving variance-covariance parameters, this approach would have been more difficult, and require further explanation to describe. Our aim within this paper was to highlight potentially better methods, particular in examples where representing the utilities as derived from a normal distribution was not unreasonable.

Further research should also look at the dependence of the results and conclusions on the data we have used. For example, the standard errors of the distribution of the two mean values U1 and U2 are the same in our data, and preliminary additional research we have undertaken has shown that, when method two is used, the variance of the quantity U1 – U2 varies nonlinearly as a function of the ratios of the variances of U1 and U2, with higher ratios of the variances leading to greater variance in this quantity. However, with method two once the ratio of the variances is increased sufficiently then a proportion of the distributions violate the monotonicity assumption. Further details of this additional analysis are available from the corresponding author on request.

### Implications for practice

We present the code for implementing each of these methods in the appendix, so they are available for researchers to use where monotonicity needs to be represented in PSA. All methods require only basic summary statistics, and can be quickly and easily implemented. Given the results presented here, we recommend that method ten be used in practice.

Where the mean values for U1 and U2 are far apart and standard errors are small, then even independent sampling is unlikely to produce violation of monotonicity. Even in these cases, because method ten takes no more time or effort to implement than the other methods, we recommend that method ten still be adopted.

In cases where the means for U1 and U2 are very close together and the standard errors are large, it is important to ask how confident we are about the validity of the monotonicity assumption, and how implausible it would be that the mean HRQoL in U1 is actually lower than the mean HRQoL in U2. If we are confident that the monotonicity assumption is correct, then method ten should be able to ensure that this relationship is represented in the PSA.

### Conclusion

Within this paper, we have compared ten methods for producing PSA for two monotonically linked variables with each other, and with a ‘gold standard’ which uses IPD. We found a new method which we developed to have superior properties in terms of representing uncertainty about the difference between variables, without producing biases or discontinuities in the simulated values. For this reason, we recommend the method be adopted widely within health technology assessments.

# Appendices

## Appendix 1: the hypothetical patient level data

|  |  |  |
| --- | --- | --- |
| Patient number | U1 | U2 |
| 1 | 0.73624 | 0.72501 |
| 2 | 0.69819 | 0.62577 |
| 3 | 0.75643 | 0.71941 |
| 4 | 0.63822 | 0.59433 |
| 5 | 0.64629 | 0.56543 |
| 6 | 0.61907 | 0.56542 |
| 7 | 0.80013 | 0.77922 |
| 8 | 0.41191 | 0.36400 |
| 9 | 0.66461 | 0.54031 |
| 10 | 0.51380 | 0.50906 |
| 11 | 0.59403 | 0.53216 |
| 12 | 0.37144 | 0.33756 |
| 13 | 0.60832 | 0.57257 |
| 14 | 0.52807 | 0.42046 |
| 15 | 0.82390 | 0.77916 |
| 16 | 0.68240 | 0.66897 |
| 17 | 0.46268 | 0.45757 |
| 18 | 0.57651 | 0.51728 |
| 19 | 0.57256 | 0.49599 |
| 20 | 0.60720 | 0.56142 |
| 21 | 0.54251 | 0.48132 |
| 22 | 0.62520 | 0.61098 |
| 23 | 0.69423 | 0.60328 |
| 24 | 0.51200 | 0.46383 |
| 25 | 0.59166 | 0.55184 |
| 26 | 0.55963 | 0.54106 |
| 27 | 0.58825 | 0.55057 |
| 28 | 0.76697 | 0.64782 |
| 29 | 0.55125 | 0.49158 |
| 30 | 0.25630 | 0.22664 |

Table Hypothetical individual patient data

## Appendix 2: R code

|  |  |
| --- | --- |
| R Code | Comments |
| rm(list=ls())  Data.2D <- data.frame(  U1=  c(  0.6981868, 0.7564343, 0.6382204, 0.6462851, 0.6190710, 0.8001344, 0.4119082, 0.6646116, 0.5137965, 0.5940299, 0.3714398, 0.6083170, 0.5280737, 0.8239041, 0.6823991, 0.4626827, 0.5765112, 0.5725570, 0.6071968, 0.5425066, 0.6251989, 0.6942350, 0.5120049, 0.5916603, 0.5596280, 0.5882450, 0.7669716, 0.5512535, 0.2562950  ),  U2=  c(  0.6257671, 0.7194083, 0.5943290, 0.5654279, 0.5654237, 0.7792152, 0.3639981, 0.5403120, 0.5090605, 0.5321613, 0.3375571, 0.5725718, 0.4204609, 0.7791617, 0.6689688, 0.4575665, 0.5172808, 0.4959917, 0.5614181, 0.4813226, 0.6109787, 0.6032772, 0.4638334, 0.5518375, 0.5410590, 0.5505654, 0.6478170, 0.4915789, 0.2266444  )  )  require(MASS)  plot(U2 ~ U1, data=Data.2D, xlim=c(0,1), ylim=c(0,1), xlab="Utility in moderate state (U1)", ylab="Utility in severe state (U2)")  abline(0,1)  cov(Data.2D)  cor(Data.2D)  U1.summary <- list(mu=0.60, sd=(0.644 - 0.600)/1.96)  U2.summary <- list(mu=0.55, sd=(0.594 - 0.550)/1.96) | Clear the R workspace  Load the data  Load a required library  Plot the data  Create a y=x line  Identifying the true variance-covariance of the scatter  Identifying the true correlation  The summary data for U1 and U2. This is the only data typically available to a modeller, and the only data used by the methods evaluated below. |
| **Methods** | |
| n.PSA <- 1000 | Set the number of PSA runs to use |
| ***Method 1: Independent Sampling*** | |
| PSA.method01 <- data.frame(u1=rnorm(n.PSA, mean=U1.summary$mu, sd=U1.summary$sd),  u2=rnorm(n.PSA, mean=U2.summary$mu, sd=U2.summary$sd)) | Creates a dataframe containing 1000 draws of U1 and 1000 draws of U2, independently sampled |
| ***Method 2: Same Random Number Seed*** | |
| seed.value <- 20  set.seed(seed.value)  u1 <- rnorm(n.PSA, mean=U1.summary$mu, sd=U1.summary$sd)  set.seed(seed.value)  u2 <- rnorm(n.PSA, mean=U2.summary$mu, sd=U2.summary$sd)  PSA.method02 <- data.frame(u1=u1, u2=u2)  rm(u1, u2) | Set the value to use for the random number seed  Set the random number seed to use the seed value  Run the rnorm function using this seed value and u1 summaries  Re-set the random number seed back to 20  Run the rnorm function using this seed value and u2 summaries  Create a dataframe with the PSA values  Remove u1 and u2 objects stored outside the dataframe |
| ***Method 3: Upward Replacement*** | |
| u1 <- rnorm(n.PSA, mean=U1.summary$mu, sd=U1.summary$sd)  u2 <- rnorm(n.PSA, mean=U2.summary$mu, sd=U2.summary$sd)  u1[u1 < u2] <- u2[u1 < u2]  PSA.method03 <- data.frame(u1=u1, u2=u2)  rm(u1, u2) | Create values independently for U1 and U2 as per method 1  Identify the vector of values where monotonicity has been violated and replaces the violating u1 values with corresponding u2 values  Create a dataframe with the PSA values and remove local copy of u1 and u2 |
| ***Method 4: Downward Replacement*** | |
| u1 <- rnorm(n.PSA, mean=U1.summary$mu, sd=U1.summary$sd)  u2 <- rnorm(n.PSA, mean=U2.summary$mu, sd=U2.summary$sd)  u2[u2 > u1] <- u1[u2 > u1]  PSA.method04 <- data.frame(u1=u1, u2=u2)  rm(u1, u2) | Create values independently for U1 and U2 as per method 1  Identify the vector of values where monotonicity has been valued and replace the violating u2 values with corresponding u1 values  Create a dataframe with the PSA values and remove local copy of u1 and u2 |
| ***Method 5: Upwards Resampling*** | |
| u1 <- rnorm(n.PSA, mean=U1.summary$mu, sd=U1.summary$sd)  u2 <- rep(NA, n.PSA)  for (i in 1:n.PSA){  continue <- F  while(continue==F){  this.u2 <- rnorm(1, mean=U2.summary$mu, sd=U2.summary$sd)  if (this.u2 < u1[i]){  u2[i] <- this.u2  continue <- T  }  }  }  PSA.method05 <- data.frame(u1=u1, u2=u2)  rm(u1, u2) | Sample the U1 values in the usual way.  Create an empty vector of the same length as the u1 vector to hold u2 estimates generated  A short routine that, for each element in the initially empty u2 vector, keeps resampling from the independent u2 distribution until a value is found which is less than the corresponding u1 value.  Package u1 and u2 estimates in a data frame then deletes local copies of the u1 and u2 objects |
| ***Method 6: Downwards Resampling*** | |
| u1 <- rep(NA, n.PSA)  u2 <- rnorm(n.PSA, mean=U2.summary$mu, sd=U2.summary$sd)  for (i in 1:n.PSA){  continue <- F  while(continue==F){  this.u1 <- rnorm(1, mean=U1.summary$mu, sd=U1.summary$sd)  if (this.u1 > u2[i]){  u1[i] <- this.u1  continue <- T  }  }  }  PSA.method06 <- data.frame(u1=u1, u2=u2)  rm(u1, u2) | Create an empty u1 vector  Sample u2 values in the usual way.  A short routine that, for each element in the initially empty u1 vector, keeps resampling from the independent u1 distribution until a value is found which is less than the corresponding u2 value.  Package u1 and u2 estimates in a data frame then deletes local copies of the u1 and u2 objects |
| ***Method 7: Setting covariance to AIVM*** | |
| MakeAIVMCov.2d <- function(mu.X, sd.X, mu.Y, sd.Y, n.psa=n.PSA){  require(MASS)  varX <- sd.X^2  varY <- sd.Y^2    aivm <- min(  mean(  c(varX, varY)  ),  sd.X \* sd.Y)    sig <- matrix(data=c(varX, aivm, aivm, varY), nrow=2, byrow=T)    aivm.samples <- mvrnorm(n=n.psa, mu=c(mu.X, mu.Y), Sigma=sig )  colnames(aivm.samples) <- c("X.sampled", "Y.sampled")  aivm.samples <- as.data.frame(aivm.samples)    return(aivm.samples)  }  PSA.method07 <- MakeAIVMCov.2d(  mu.X=U1.summary$mu,  sd.X=U1.summary$sd,  mu.Y=U2.summary$mu,  sd.Y=U2.summary$sd  )  names(PSA.method07) <- c("u1", "u2") | Creates a short function which produces U1 and U2 values jointly which are correlated. The correlation is that associated with AIVM unless this would imply a correlation greater than 1, in which case a correlation of 1 is used instead.  The function takes five inputs: the means and standard deviations of the two variables, and the number of PSA runs.  Checks whether a library of functions has been loaded, this includes the function mvrnorm, which is required later.  Calculates variances given the standard deviations  Produces a variable, called aivm, which is the minimum of the average individual variances of the means, or the covariance which would imply a correlation of 1.  Produces a 2x2 covariance matrix using aivm as the off-diagonal values.  Produce correlated samples of the two variables  Formats aivm.samples to be consistent with those produced elsewhere.  Returns the labelled and formatted output as the function output  Uses the function created above with the summary values identified  Renames the variables in the data frame created in the above line for consistency |
| ***# METHOD 8: Lower Bounded Covariance Retrofitting***  ***# METHOD 9: Upper Bounded Covariance Retrofitting*** | |
| MakeBCVR.2d <- function(mu.X, sd.X, mu.Y, sd.Y, n.psa=n.PSA, incBy=0.00001, upper=T){  require(MASS)      varX <- sd.X^2  varY <- sd.Y^2  if(upper==T){  lowerbound <- 0  } else {  lowerbound <- mean(varX, varY)  }    upperbound <- min(sd.X \* sd.Y,  mean(varX, varY)  ) # upper bounds are the minimum of the AIVM or the cov which implies a cor > 1    this.cov <- lowerbound  cat(varX, varY, lowerbound, upperbound, this.cov, "\n")  mus <- c(mu.X, mu.Y)  search <- T    if(this.cov==upperbound){ # if the maximum value's been reached already    cat("Upperbound already reached\n")  search <- F # if the upper limit's already been reached, go no further  testsig <- matrix(c(varX, this.cov, this.cov, varY), nrow=2, byrow=T)  testsamples <- mvrnorm(n.psa, mu=mus, Sigma=testsig)  } else {  cat("Upperbound not yet reached\n")  this.cov <- lowerbound  cat("This covariance: ", this.cov, "\n", sep="")  testsig <- matrix(c(varX, this.cov, this.cov, varY), nrow=2, byrow=T)  testsamples <- mvrnorm(n.psa, mu=mus, Sigma=testsig)  }    while(search==T){  cat("trying ", this.cov, "\n")  testsig <- matrix(c(varX, this.cov, this.cov, varY), nrow=2, byrow=T)  try.testsamples <- try(mvrnorm(n.psa, mu=mus, Sigma=testsig))  if(class(try.testsamples)=="try-error"){ # if mvrnorm has been passed impossible values  search <- F  cat("Error picked up\n")    } else {  cat("No error in mvrnorm args\n")  testsamples <- try.testsamples # if the attempted values are correct, use them  if (any(testsamples[,1] < testsamples[,2])){  cat("Violation with ", this.cov, "\n")  this.cov <- this.cov + incBy # increment the values by a little bit  cat("Trying ", this.cov, "\n")  } else {  cat("Found ", this.cov, "\n")  search <- F  }  }  }  return(list(cov=this.cov, samples=testsamples))  }  tmp <- MakeBCVR.2d(  mu.X=U1.summary$mu,  sd.X=U1.summary$sd,  mu.Y=U2.summary$mu,  sd.Y=U2.summary$sd,  upper=F  )  method08.cov <- tmp$cov  PSA.method08 <-data.frame(tmp$samples)  names(PSA.method08) <- c("u1", "u2")  tmp <- MakeBCVR.2d(  mu.X=U1.summary$mu,  sd.X=U1.summary$sd,  mu.Y=U2.summary$mu,  sd.Y=U2.summary$sd  )  method09.cov <- tmp$cov  PSA.method09 <- data.frame(tmp$samples)  names(PSA.method09) <- c("u1", "u2")  plot(u2 ~ u1, data=PSA.method08)  plot(u2 ~ u1, data=PSA.method09) | Another function, this time with seven inputs. These are the five inputs to the MakeAIVMCov.2d() function above, and the following two arguments:  incBy : the size of the increment in each guess for the appropriate covariance  upper: whether method 8 or method 9 should be calculated. Upper=T is method 9, and upper=F is method 8.  Both incBy and upper have default values, which will be used if other values have not been specified.  Calculates variances for each variables as in previous function  Set a value called lowerbound to 0 if method 9 is used (upper=T), oe the AIVM if method 8 is used (upper=F) |
| ***Method 10: Beta distribution difference fitting*** | |
| rU1 <- rnorm(n.PSA, U1.summary$mu, U1.summary$sd)  rU2 <- rnorm(n.PSA, U2.summary$mu, U2.summary$sd)  ShowImps <- function(U1, U2, a, b, n.PSA, main="", xlim=c(0,1), ylim=NA, ylab="", generate=F){  if (all(is.na(ylim))==T){  plot(density(U1), xlim=xlim, main=main, ylab=ylab)  } else {  plot(density(U1), xlim=xlim, main=main, ylim=ylim, ylab=ylab)  }  lines(density(U2), lty="dashed")    increment <- rbeta(n.PSA, a, b)  U1.subst <- U2 + increment    lines(density(U1.subst), lty="dashed", lwd=2)    legend("topright", legend=c("U1", "U2", "U1\*"), lty=c("solid", "dashed", "dashed"), lwd=c(1,1,2))    if(generate==F){  output <- list(  mean.u1 = mean(U1),  sd.u1 = sd(U1),  mean.u2 = mean(U2),  sd.u2 = sd(U2),  mean.u1s = mean(U1.subst),  sd.u1s = sd(U1.subst)  )  } else {  output <- list(  mean.u1 = mean(U1),  sd.u1 = sd(U1),  mean.u2 = mean(U2),  sd.u2 = sd(U2),  mean.u1s = mean(U1.subst),  sd.u1s = sd(U1.subst),  U1.subst = U1.subst  )  }  return(output)  }  CalcImps <- function(U1, U2, log.a, log.b, n.PSA){  increment <- rbeta(n.PSA, exp(log.a), exp(log.b))  U1.subst <- U2 + increment    mean.u1 <- mean(U1)  sd.u1 <- sd(U1)  mean.u2 <- mean(U2)  sd.u2 <- sd(U2)  mean.u1s <- mean(U1.subst)  sd.u1s <- sd(U1.subst)    dif.mean <- mean.u1 - mean.u1s  dif.sd <- sd.u1 - sd.u1s  dif.rms <- (dif.mean^2 + dif.sd^2)^0.5    output <- list(  mean.u1 = mean.u1,  sd.u1 = sd.u1,  mean.u2 = mean.u2,  sd.u2 = sd.u2,  mean.u1s = mean.u1s,  sd.u1s = sd.u1s,  dif.mean = dif.mean,  dif.sd = dif.sd,  dif.rms = dif.rms  )    return(output)  }  MinRms.N <- function(par, U1, U2, n.PSA){  this.N <- exp(par)  sample.mu <- mean(U1 - U2)    a <- this.N \* sample.mu  b <- this.N - a  this.obj <- CalcImps(U1, U2, log.a=log(a), log.b=log(b), n.PSA)    return(this.obj$dif.rms)  }  init.log.N <- 0  init.par <- init.log.N  optim.out <- optim(init.par, MinRms.N, U1=rU1, U2=rU2, n.PSA=n.PSA, method="BFGS", hessian=T)  n.bfgs <- exp(optim.out$par)  sample.mu <- mean(rU1 - rU2)  a.n <- sample.mu \* n.bfgs  b.n <- n.bfgs - a.n  Imps.output <- ShowImps(rU1, rU2, a.n, b.n, n.PSA, main="", xlim=c(0.4, 0.8), ylim=c(0, 20),  ylab="Density of estimated values", generate=T)    rU1s <- Imps.output$U1.subst  PSA.method10 <- data.frame(u1 = rU1s, u2=rU2) | Independent samples of U1 and U2 using normal distributions  **FUNCTIONS**  ShowImps function which shows, with a kernel density plot, the implications of assuming different a and b parameter values for the Beta distribution, in terms of how similar U1\* is to U1.  CalcImps function which calculates the implications of different a and b parameters for the Beta distribution on the resulting root mean squared difference between the sample mean and sample standard deviation of U1 compared with U1\*.  Root mean squared difference  # Function for calculating the a and b parameters for the Beta function which are logically implied by different N values, and which returns the root mean squared difference  **OPTIMISATION**  Starting value of log(N) for numerical optimisation. Log(N) rather than N is used to allow unconstrained optimisation.  Optimisation function, which repeatedly calls the MinRms.N function with different values of N, converging on N values which minimise the root mean square of the difference.  Calculates the N value identified by the optim routine  Calculates the sample mu  Calculates the a and b parameters implied by the N value and sample mu  Draw the densities of U2, U1 and U1\* values produced by using the optimised a and b parameters for the Beta distribution  Loads 1,000 U1\* samples  Saves PSA produced by this method |
| Bootstrapped estimates for comparison | |
| methodBoot.PSA <- matrix(NA, ncol=2, nrow=n.PSA)  for (i in 1:n.PSA){  draws <- 1: dim(Data.2D)[1]  size=dim(Data.2D)[1]  tmp <- Data.2D[sample(draws, size, T),]  methodBoot.PSA[i,] <- c(mean(tmp[,1]), mean(tmp[,2]))  }  methodBoot.PSA <- data.frame(methodBoot.PSA)  names(methodBoot.PSA) <- c("u1","u2") | Create an empty matrix 2 columns wide  Loop 1000 times  Sample with replacement  Calculate bootstrapped means of each sample with replacement  Load output from above loop into a dataframe |
| Packaging results together | |
| MethodsBlock <- list(  methodboot=methodBoot.PSA,  method01=PSA.method01,  method02=PSA.method02,  method03=PSA.method03,  method04=PSA.method04,  method05=PSA.method05,  method06=PSA.method06,  method07=PSA.method07,  method08=PSA.method08,  method09=PSA.method09,  method10=PSA.method10) | # packaging results together in list to make them easier to automate |

## Appendix 3: Violin plots of distributions of U1 and U2

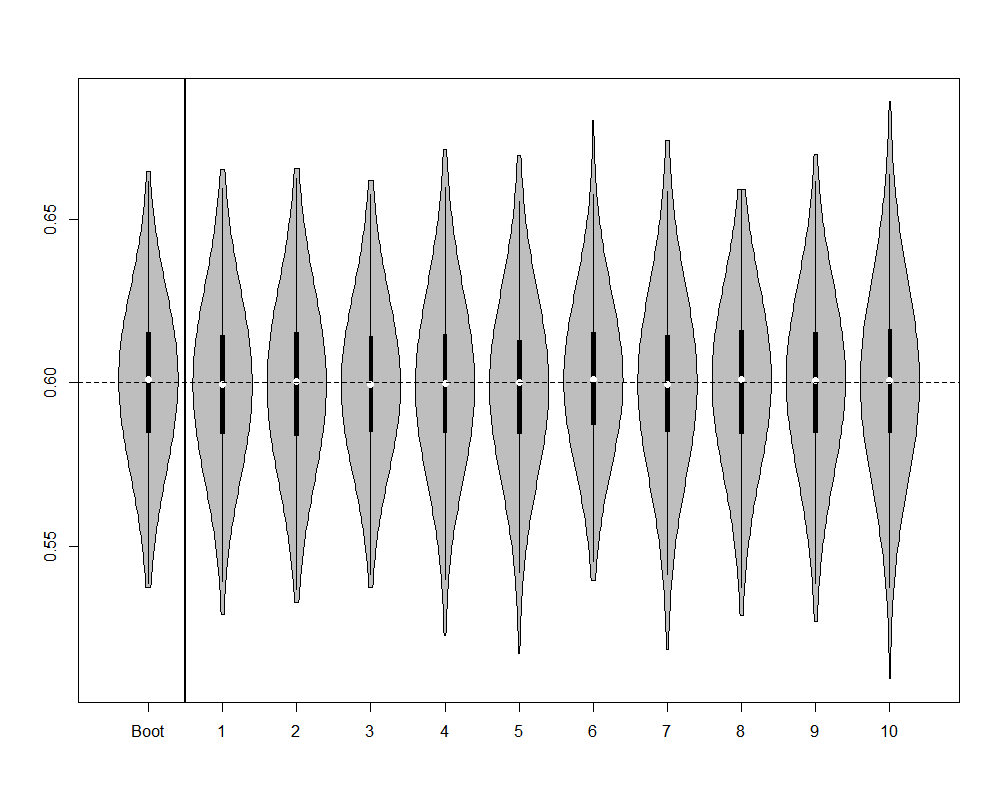


Figure Violin plot of distribution of U1 estimates produced by each of the 10 methods, compared with bootstrapped estimates

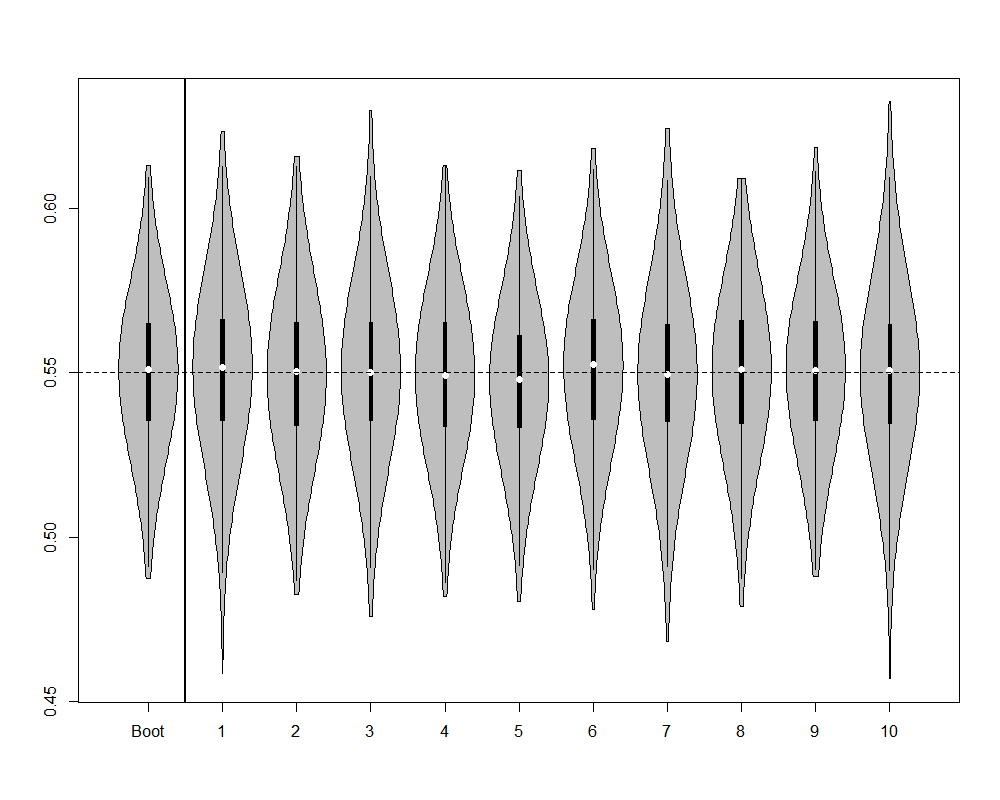


Figure Violin plot of distribution of U2 estimates produced by each of the 10 methods, compared with bootstrapped estimates