# Structure

## Introduction

When building economic models we may need to produce a large number of estimates for the mean utility associated with being in a more severe health state compared with that of being in a less severe health state as part of probabilistic sensitivity analysis (PSA). Within the PSA, although we need to represent uncertainty about the true mean utility value associated with both states, we also need to represent broader clinical beliefs that there is a monotonic relationship between the mean utility values associated with the health states, so that the more severe health state will have on average a lower utility than the more severe state. This effectively means that, for each and every iteration of the PSA, the utility estimate for the more severe state should be lower than the corresponding utility for the less severe state. Often, however, we only have access to basic summary statistics about the utility associated with each health state.

In this paper we describe and explore ten different approaches to creating PSA estimates based on summary information about two health states we believe to be monotonically related. We have produced simulated individual patient data for the utility values associated with each patient in a moderate disease severity state (U1) and a severe disease severity state (U2). From this simulated individual level data, we produced summary statistics giving the mean and standard error of the utility value associated with each health state. It is the summary rather than individual level data which are typically available to the modellers, and so are used as the inputs for each of our approaches. We produce a list of a range of desirable and essential conditions which our methods should satisfy, and using the simulated data illustrate how well each of the approaches satisfies each of the conditions. We conclude with recommendations for practice.

## 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). The individual patient data are shown in Table 1, and the corresponding scatter plot for these data are shown in Figure 1 below. As modellers are typically interested in representing uncertainty in mean values rather than the values themselves, uncertainty in the mean scores were estimated using a bootstrapping procedure, in which the IPD were resampled with replacement 1000 times, and the corresponding mean values of U1 and U2 were extracted. This is shown in Figure 2, and represents the ‘gold standard’ against which the output of the ten methods will be compared.

|  |  |  |
| --- | --- | --- |
| 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 1 Hypothetical individual patient data

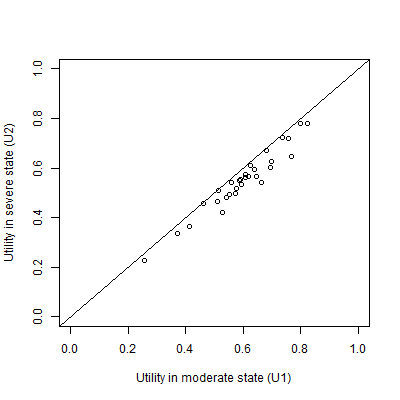


Figure 1 A plot of the simulated individual patient data

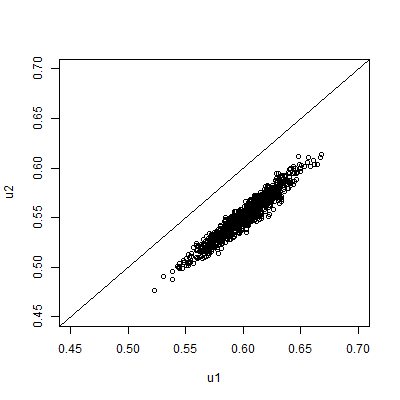


Figure 2 Scatterplot of 1000 PSA draws of the joint means of U1 and U2 produced by bootstrapping the IPD in table 1

In our hypothetical example, 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 | 0.60 | 0.55 |
| 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 using within probabilistic sensitivity analyses (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 Approaches

The long list of ten approaches considered for handling the U2 < U1 monotonicity relationship are described in Table 2 below. Approaches 1-6 are simpler approaches, whereas approaches 7-10 are more complicated.

|  |  |  |
| --- | --- | --- |
| **Method Number** | **Name** | **Method Description** |
| 1 | Independent Sampling | For each of the M 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 M 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.) |
| 3 | Upward Replacement | For each of the M 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 M 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 M 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 M 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. |
| 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 Retrofitting | Methodology 8 but where the second condition is that the covariance between U1 and U2 is less than AIVM. |
| 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

### Simpler Approaches

Approaches one to six are all fairly simple, simulation based approaches. Approach one is the simplest and most naïve. Approach 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.

### Covariance and impossible correlations

Approaches seven to 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. However, all covariances are also constrained by their associated correlations, such that the covariance between the variables cannot imply a correlation greater than one. 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 applies to approaches eight and nine.

### Selection of covariance for methods eight and nine

The covariances for methods eight and nine were selected using a computer algorithm which used a brute force approach to identifying correlation estimates. For both approaches the effective upper bound of the covariance which may be selected is the covariance which implies a correlation of 1, i.e. the product of the two standard errors. For approach eight the lower bound was set to the value used in approach seven; for approach nine the lower bound was set to zero. This algorithm is presented in the appendix.

### Beta Distribution Difference Modelling

Within Method 10, instead of the independent distribution U1 being used directly in the PSA, it is used indirectly, in order to produce an alternative sampling distribution U1\*. 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 U1\* as follows:

U1\* = U2 + Δ,

Δ ~ Beta(a, b)

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 U1\* to be less than U2.

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

E(Δ) = E(U1 – U2),

E(Δ) = μ,

We say N = a + b, and so, a = μN and b = 1 – N.

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 U1\* is minimised. I.e. selecting a value of N which satisfies the following condition:

)

The R code used to perform this optimisation is presented in the appendix.

### Comparison of PSA produced by different methods

In addition to producing estimates which do not violate the monotonicity assumption, an appropriate method for producing PSA estimates should, ideally, be able to produce joint estimates of the means of U1 and U2 which are very similar to that would be identified given the IPD. This means that scatterplots of 1,000 PSA runs comparing estimates of the mean of U1 with the mean of U2 should look similar to those seen in Figure 2. Dissimilarities in the scatterplots produced can indicate that the method may be biased. For example, they may introduce biases into the central estimates, or into estimates of the differences between values U1 – U2.

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 3, 4, 5, 6 and 10. For methods 7, 8 and 9, which use algorithms to select covariances between parameters, it is possible that for some runs U2 values will occasionally exceed corresponding U1 values, although this may be a rare occurrence. 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 3,4, 5, 6, and 10 will be described as satisfying ‘strict monotonicity’; whereas methods 7, 8 and 9 will be described as satisfying ‘relaxed monotonicity’.

## Results: What did we find?

### Parameterisation of methods 7-9

For method 7, 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 0.00504 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 7 is simply equivalent to setting the correlation between the means of U1 and U2 to 1.

For method 8, which forces a covariance of 0.00504, implying a correlation of 1, was also identified, as method 8 uses the covariance value from method 7 as its upper bound, and as this value already implies a correlation of 1 it cannot be any higher. For method 9, however, which uses a covariance of 0 as its lower bound, a covariance of 0.00038, implying a correlation of 0.754, was identified.

### Parameterisation of method 10

The optimisation routine selected an N value of 1925.72, producing Beta (a = 96.86, 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.

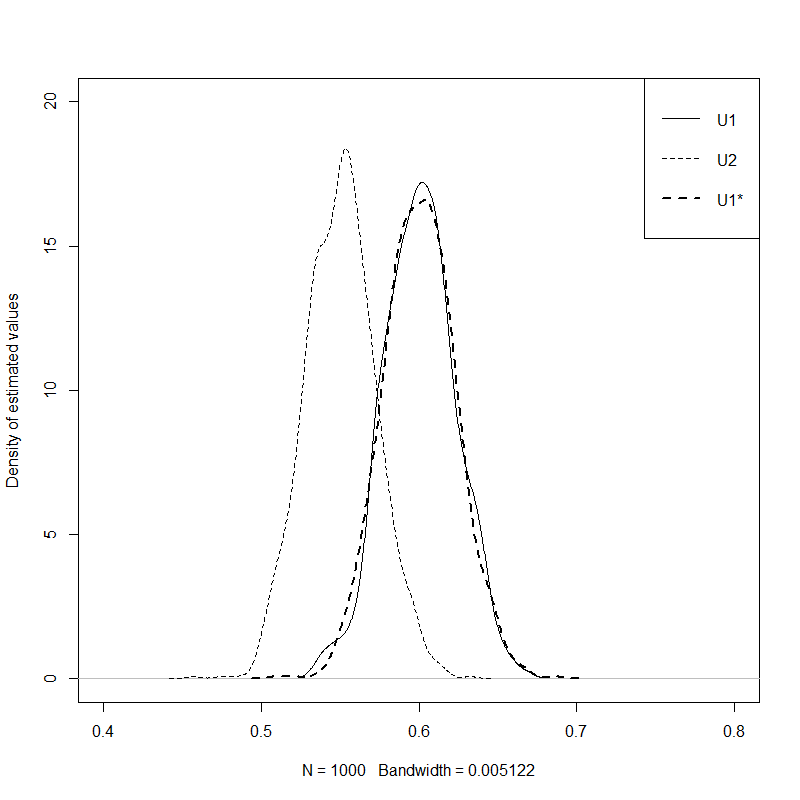


Figure Comparison of the distribution of estimates of U2, U1, and U1\* produced using Method 10

### Scatterplots

Figures xxx present the scatter of the PSA produced by each of the methods 1-10. Figure X shows the corresponding scatter produced from the bootstrapped PSA. 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.

We can see that method 1 includes 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 3, 4, 5, and 6 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 may also be 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 1, 3, 4, 5, and 6 all produce uncorrelated scatter that is too wide, indicating the correlation of the PSA estimates is too low. By contrast, methods 2, 7, 8 all produce scatter which is too narrow, indicating the correlation estimates are too high.

We see from the scatter that method 10 (fig XX) and method 9 (fig XX) are both closest in appearance to the bootstrapped scatter, with method 10 exhibiting closer values than method 9.

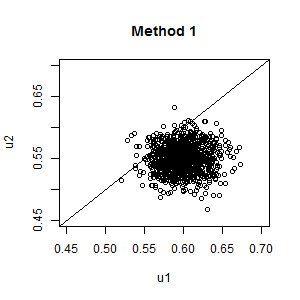


Figure 4 PSA Scatterplot: Independent sampling

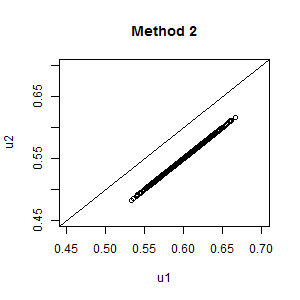


Figure 5 PSA scatterplot: Same random number seed

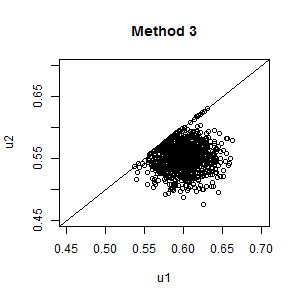


Figure 6 Upward Replacement

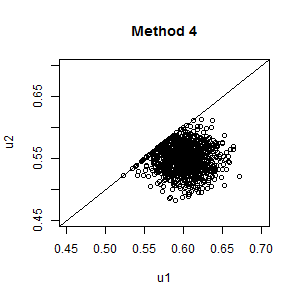


Figure 7 Downward Replacement

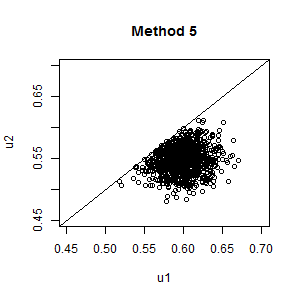


Figure 8 Upward Resampling

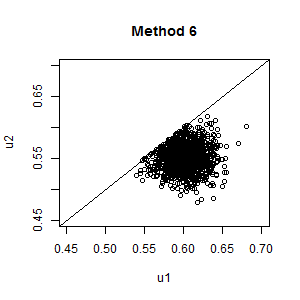


Figure 9 Downwards Resampling

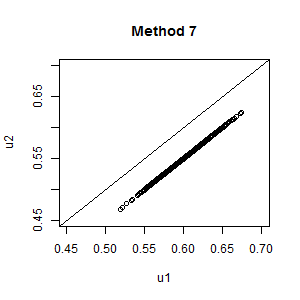


Figure 10 Setting covariance to AIVM

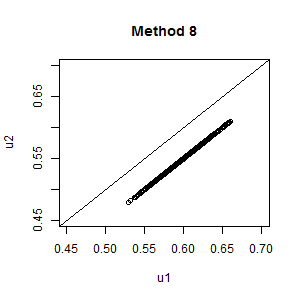


Figure 11 Upper bounded covariance fitting

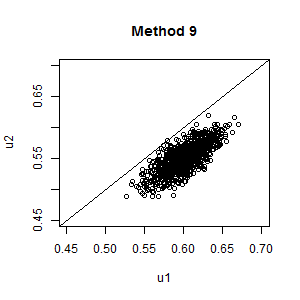


Figure 12 Lower bounded covariance fitting

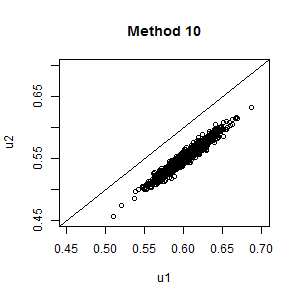


Figure 13 Beta difference modelling

### Monotonicity violation

The two replacement methods (3 and 4), the two resampling methods (5 and 6), and the Beta method (method 10) are all designed such that it is analytically impossible for them to violate the monotonicity assumption. As such we know they must satisfy this criterion in both its strict and more relaxed form. The other methods could all potentially violate monotonicity, in that U2 values are not analytically constrained to be less than U1 values, or U1 values all analytically constrained to be greater than the U2 values. In this example, the only approach where monotonicity is violated in the 1000 PSA samples being compared is method 1, independent sampling, where 53 out of the 1000 PSA samples violated montonicity. 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 produced for U1, U2, and the derived quantities U1 – U2, for each of the ten methods, compared with the gold standard, the bootstrapped data. 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 above.(Hintze & Nelson, 1998) Like the scatterplots shown in Figure 4 through to Figure 13, they therefore allow nuanced comparisons between the distributions to be made.

Figure 14 shows the distribution of values of U1 produced by each of the ten methods, compared with the bootstrapped distribution shown on the left. We see that all distributions are broadly similar, although many of the methods produce wider distributions than the bootstrapped distribution. Method 3 appears closest in shape to the bootstrapped distribution, and method 10 has the most excessively wide distribution. In all cases, however, the distributions are broadly similar.

Figure 15 shows that the distribution of estimates of U2 produced by each of the ten methods, compared with the bootstrapped estimates on the left. Again, most methods produce broadly similar distributions of these quantities, although the distributions produced by a number of methods, including methods 1, 7 and 10, appear too wide. There is some indication that methods 5 and 6 produce biased means, although this could be due to simulation sampling error.

Figure 16 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 1, 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 obvious for some of the methods. Methods 2, 7 and 8 all severely underestimate the uncertainty in this quantity, as all estimates U1 – U2 are identical. Methods 3, 4, 5 and 6 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 9 and 10 appear broadly appropriate in terms of representing this form of uncertainty. Of these two methods, method 10 produces the distribution of values which is closest to the bootstrapped distribution.

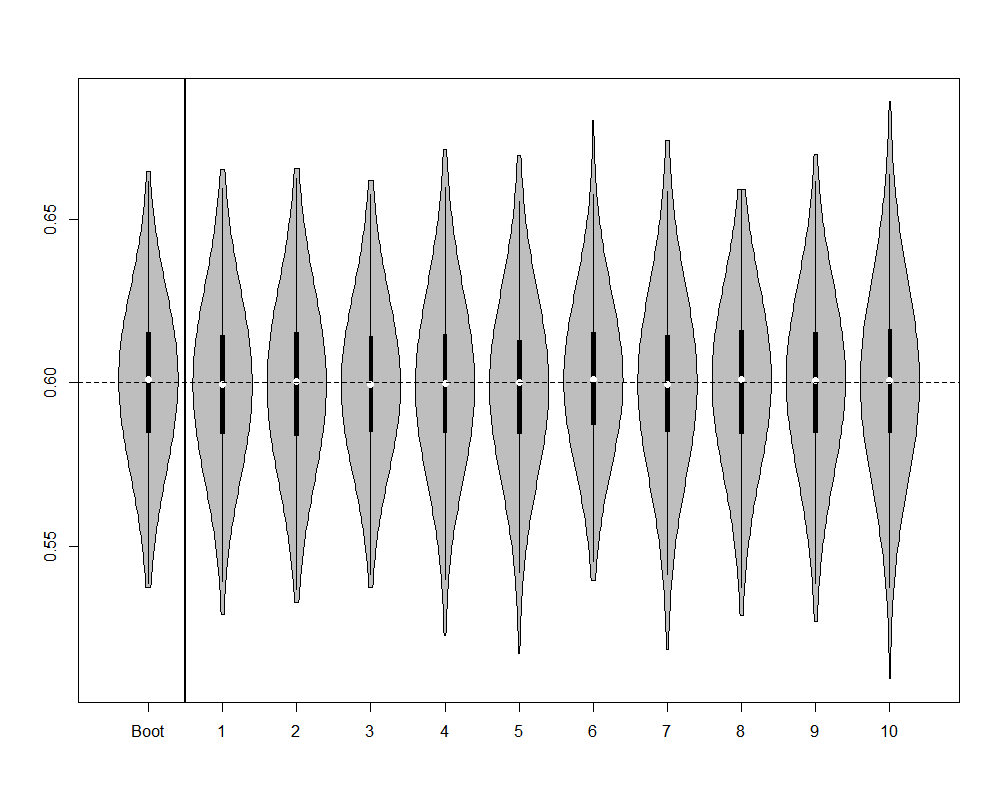


Figure Violin plot of distribution of U1 estimates produced by each of the 10 methods, compared with bootstrapped estimates

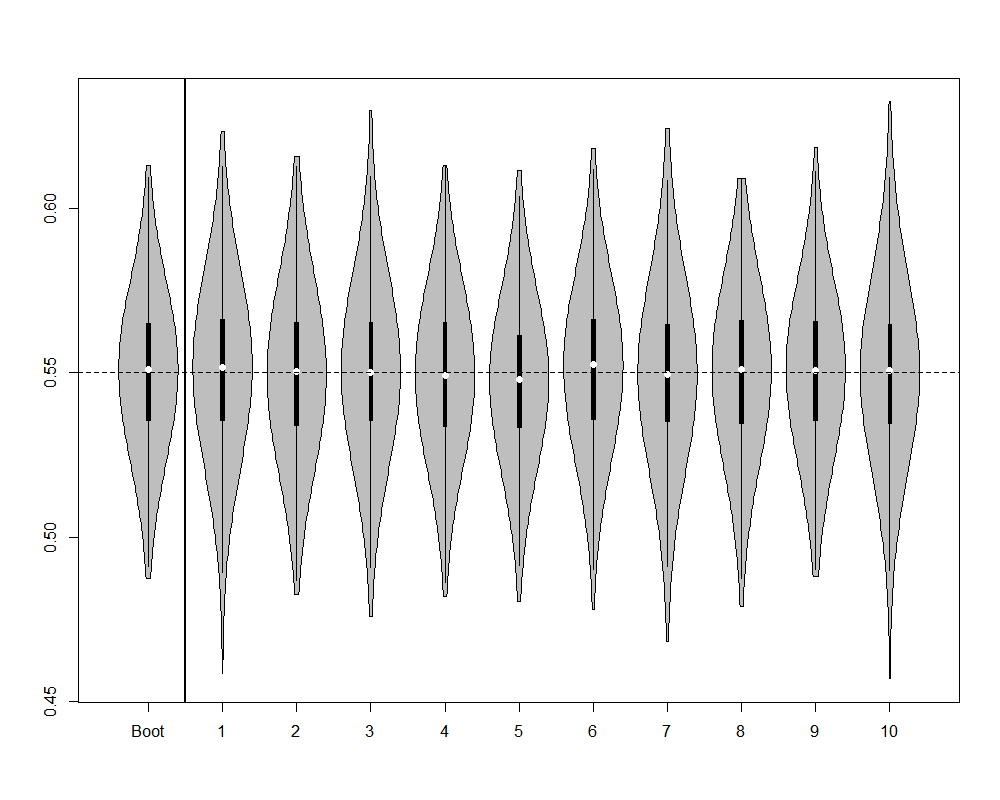


Figure Violin plot of distribution of U2 estimates produced by each of the 10 methods, compared with bootstrapped estimates

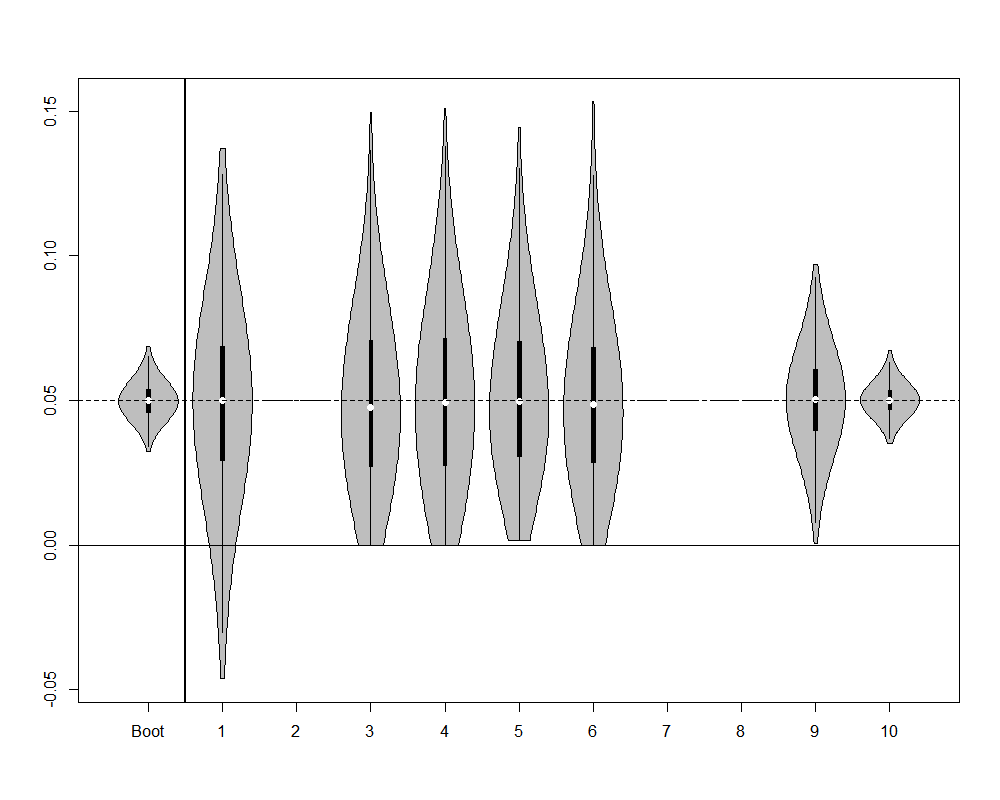


Figure Violin plot of distribution of U1 - U2 estimates, for each of the 10 methods, compared with bootstrapped estimates

## Discussion:

### Para 1: Summary of what found

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 (2, 7, and 8) 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. Other methods (3, 4, 5, 6) introduce implausible discontinuities into the distribution of differences between values. There are also theoretical reasons to assume that some of the approaches (3, 4, 5, 6) will produce biased means, although such biases were not readily apparent in our results.

### Para 2: Shortcomings

Not using betas

Not using a range different U1, U2 examples.

Not presenting a 3 state (U1, U2, U3) example.

### Para 3: How relates to other findings

[Not sure… ask Matt/Nick…]

### Para 4: Implications for Research

…

### Para 5: Implications for practice

[Depends how much of an issue one considers violation of monotonicity assumption to be.

Less of an issue when not looking at expected values.]

[Make sure to test whether monotonicity is a problem in practice if not in theory.

[Where SEs are small relative to differences between means of distributions, even a naïve approach assuming distributions are independent could produce no violation of monotonicity on average within 1000 PSA runs]

[Need to be sure that the monotonicity assumption is valid in the particular context considered.

For example, although perhaps one can assume increasing disease severity should definitely be associated with decreased patient utility, it might not make as much sense to assume costs are also monotonic. It may be that the most severe stages of a disease are cheaper to treat than less severe stages, as fewer effective treatment options exist.

[It could also be that many clinicians assume two formally defined disease states are actually very similar, and so it would not be contrary to their expectations if the utility value of a ‘more severe’ state were actually about the same or higher on average than that of the ‘less severe’ state.

[Also far less likely to always be true when considering predicted rather than expected values, as it’s very probable that on some occasions at least some patients with a more severe disease will have higher utility than at least some patients with a less severe condition.]

When presented with summary data and with a belief that monotonicity must apply, a judicious selection of the covariance parameters or of the distributions for the differences appears appropriate. The former strategy is likely to be preferential if there are more than two parameters and there is some belief of correlation between the variables.

### Para 6

# Appendices

|  |  |
| --- | --- |
| 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*** | |
| ShowImps <- function(U1, U2, shape1, shape2){  plot(density(U1), xlim=c(0,1))  lines(density(U2), lty="dashed")    increment <- rbeta(n.PSA, shape1, shape2)    lines(density(U2 + increment), lty="dashed", lwd=2)  return(sum((U1 - (U2 + increment))^2))  }  SumSquaredFinder <- function(real.values, predicted.values){  squared.errors <- (real.values - predicted.values)^2  return(sum(squared.errors))  }  FitCompare <- function(par, PSA.X, PSA.Y){  shape1 <- exp(par[1])  shape2 <- exp(par[2])  n.psa <- length(PSA.X)    additional <- rbeta(n.psa, shape1, shape2)  # cat(head(additional), "\n")  # cat(shape1, shape2, "\n")    Sims.Y <- PSA.X + additional  sumSquared <- SumSquaredFinder(real.values=PSA.Y, predicted.values=Sims.Y)  # cat("sumSquared=", sumSquared, "\n")  # browser()    return(sumSquared)  }  OptimFitFinder <- function(PSA.X, PSA.Y, initial.shape1, initial.shape2){  n.psa <- length(PSA.X)  optim.out <- optim(par=c(log(initial.shape1), log(initial.shape2)), FitCompare, PSA.X=PSA.X, PSA.Y=PSA.Y,  method="SANN", hessian=T, control=list(trace=10))  return(optim.out)  }  sims.U1 <- rnorm(n.PSA, mean=U1.summary$mu, sd=U1.summary$sd)  sims.U2 <- rnorm(n.PSA, mean=U2.summary$mu, sd=U2.summary$sd)  optimOutput <- OptimFitFinder(sims.U2, sims.U1, 6, 100)  optim.shape1 <- exp(optimOutput$par[1])  optim.shape2 <- exp(optimOutput$par[2])    plot(density(sims.U1), main="", xlim=c(0.4, 0.8), xlab="Estimated values", ylim=c(0, 20), ylab="Density of estimated values")  lines(density(sims.U2), lwd=2)  lines(density(sims.U2 + rbeta(n.PSA, optim.shape1, optim.shape2)), lty="dashed")  legend("topright", lty=c("solid", "solid", "dashed"), lwd=c(1,2,1), legend=c("U1", "U2", "Beta simulated U1"))  #ShowImps(sims.U1, sims.U2, optim.shape1, optim.shape2)  PSA.method10 <- data.frame(u1 = sims.U2 + rbeta(n.PSA, optim.shape1, optim.shape2), u2=sims.U2) | # Some functions borrowed from myself  # Use ShowImps to guess a decent enough initial value  # Find the sum of squares |
| 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") | # want bootstrapped estimates of means to compare |
| 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 |