## Appendix

|  |
| --- |
| **Simulating proportions in each mRS state**  The R code for doing this is:  mRS\_followingStroke <- rdirichlet(N.PSA, c(61, 143, 111, 82, 24, 4))  Where N.PSA and c(61, 143, 111, 82, 24, 4) are both arguments to the rdirichlet function. N.PSA provides the number of samples used in probabilistic sensitivity analysis (PSA), and c(61, 143, 111, 82, 24, 4) providing the parameter values for the Dirichlet function. These parameter values are taken directly from the source paper.  **Producing random draws from a normal distribution**  The R code for doing this for the mRS 3 state is:  s3 <- rnorm(N.PSA, .545, .277)  Where 0.545 is the mean utility reported in the source paper for this mRS state, and 0.277 is the standard deviation report. The simulated values for the other mRS states are produced similarly.  **Converting from utility values to utility multipliers**  To turn the utility simulations into utility multipliers (indicated in node 5 of both Figure 1 and Figure 2) we assumed that mRS 0 (‘no symptoms’) represented full health. The multipliers for mRS states 1-5 were therefore produced by dividing simulated values from the more severe category by simulated values from the mRS 0 distribution. The R command for producing the utility multiplier associated with mRS 3, for example, is simply:  mult.s3 <- s3/s0  This produces a vector of length N.PSA, because both s3 and s0 are also vectors of length N.PSA. The multipliers associated with the other mRS states are produced similarly.  **Simulating uncertainty in the distribution of the component states in each of the collapsed state**  The R code for doing this for each draw from the Dirichlet distribution for the Independent state category is shown below:  Stroke.Ind <- mRS\_followingStroke[,1:3]  Stroke.Ind.sums <- apply(Stroke.Ind, 1, sum)  Stroke.Ind <- apply(Stroke.Ind, 2, function (x) x / Stroke.Ind.sums)  The approach for the Dependent stroke category is similar, but uses columns 4, 5 and 6 of the object mRS\_followingStroke, instead of columns 1, 2 and 3.  **Producing weighted utility multiplier estimates for independent and dependent stroke categories**  For the Independent state the R code for doing this is as follows:  Stroke.Ind.utils <- Stroke.Ind[,1] \* 1 + Stroke.Ind[,2] \* mult.s1 + Stroke.Ind[,3] \* mult.s2  Here Stroke.Ind[,1] refers to the first column, the weight of mRS 0, and Stroke.Ind[,1] refers to column 2, the corresponding weights of mRS 1. The object mult.s1 is a vector of estimates of the utility multiplier of mRS 1 compared with mRS 0, and mult.s1 is a vector of estimates of the utility multiplier of mRS 2 compared with mRS 0. As the reference utility value is mRS 0, the utility multiplier associated with mRS 0 is just 1. The weighted utility multiplier estimates associated with Dependent strokes are calculated similarly. |

Table 6 Example R code required for the procedures described in this paper

|  |  |
| --- | --- |
| **R code** | **Comments** |
| Bootstrapper <- function(inputs, simulates = 10000){  X.mean <- vector("numeric", simulates)  N.inputs <- length(inputs)  for (i in 1:simulates) {X.mean[i] <- mean(inputs[sample(1:N.inputs, replace=T)])}  return(X.mean)  }  Require(MCMCpack)  N.PSA <- 10000  Dead\_nonDead <- rbinom(N.PSA, 1283, (319/1283)) / 1283  mRS\_followingStroke <- rdirichlet(N.PSA, c(61, 143, 111, 82, 24, 4))  DepInd\_followingStroke <- cbind(apply(mRS\_followingStroke[,1:3], 1, sum), apply(mRS\_followingStroke[,4:6], 1, sum))  DeadDepInd\_followingStroke <- cbind(Dead\_nonDead, (1 - Dead\_nonDead) \* DepInd\_followingStroke[,1], (1-Dead\_nonDead) \* DepInd\_followingStroke[,2])  colnames(DeadDepInd\_followingStroke) <- c("Dead", "Independent", "Dependent")  s0 <- rnorm(N.PSA, .959, .074)  s1 <- rnorm(N.PSA, .812 , .181)  s2 <- rnorm(N.PSA, .656, .218)  s3 <- rnorm(N.PSA, .545, .277)  s4 <- rnorm(N.PSA, .248, .281)  s5 <- rnorm(N.PSA, .020, .046)  mult.s1 <- s1/s0  mult.s2 <- s2/s0  mult.s3 <- s3/s0  mult.s4 <- s4/s0  mult.s5 <- s5/s0  Stroke.Ind <- mRS\_followingStroke[,1:3]  Stroke.Dep <- mRS\_followingStroke[,4:6]  Stroke.Dep.sums <- apply(Stroke.Dep, 1, sum)  Stroke.Ind.sums <- apply(Stroke.Ind, 1, sum)  Stroke.Dep <- apply(Stroke.Dep, 2, function (x) x / Stroke.Dep.sums)  Stroke.Ind <- apply(Stroke.Ind, 2, function (x) x / Stroke.Ind.sums)  Stroke.Ind.utils <- Stroke.Ind[,1] \* 1 + Stroke.Ind[,2] \* mult.s1 + Stroke.Ind[,3] \* mult.s2  Stroke.Dep.utils <- Stroke.Dep[,1] \* mult.s3 + Stroke.Dep[,2] \* mult.s4 + Stroke.Dep[,3] \* mult.s5  Stroke.Ind.utils.mean <- Bootstrapper(Stroke.Ind.utils)  Stroke.Dep.utils.mean <- Bootstrapper(Stroke.Dep.utils) | This is code for a bespoke function in R for finding the bootstrapped means of a vector of numbers. Other bootstrapping functions exist, but this function is easy to make.  The function defaults to running 10,000 bootstraps of the dataset. This can be adjusted by specifying a different ‘simulates’ argument.  Loads a library containing the rdirichlet() function used later.  Specify that PSA involves 10,000 sets of draws  **NODE 1**  Specifies that the object Dead\_nonDead should be created containing 10,000 draws from a binomial distribution.  The binomial distribution is parameterized with two numbers from table 1 of Rivero-Arias. ‘319’ is the number dead following stroke. ‘1283’ is the sample size of relevant individuals. The outputs from rbinom are all divided by 1283 to produce proportions rather than frequencies.  **NODE 2a + NODE 3A**  This creates a matrix containing the output of 10,000 draws from a dirichlet distribution populated by the values from table 1 of the Rivero-Arias paper showing distribution of modified Rankin Scale stroke outcomes at 24 months.  **NODE 4a + NODE 4b**  This converts six columns of mRS\_followingStroke into two columns, giving the sums of ‘independent’ and ‘dependent’ strokes respectively. The two calls to the apply function take the first three and last three columns of the mRS\_followingStroke dataframe, and output the sums of each row.  This combines estimates of the proportion alive following a stroke, Dead\_nonDead, with the proportion of those alive in either dependent or independent states, DepInd\_followingStroke. The output is a three column matrix giving 1) proportion alive; 2) proportion in independent state; 3) proportion in dependent state.  This command labels the columns of the previously created matrix to be easier to interpret.  **NODE 2b + NODE 3b**  These commands use data from table 3 (the 24 months column) from Rivero-Arias to produce 10,000 draws from Normal distributions parameterized with the means and standard deviation values from the paper. s0 is the estimated utility following an mRS 0 outcome, s1 is the estimated utility following an mRS 1 outcome, and so on.  **NODE 5 + NODE 6a + NODE 7a**  These convert the draws of estimates associated with each of the mRS states into utility multipliers for each of states mRS 1 to 5, where mRS 0 is the reference category.  **NODE 4a + NODE 4b**  These commands calculate the relative distribution of mRS states among those within either the ‘dependent’ (mRS 3-5) in ‘independent’ (mRS 0-2) stroke categories.  This allows weighted averages of utilities from mRS specific utility multipliers to be produced later.  **NODE 7a**  This produces an estimate of the utility multiplier associated with an independent stroke using a weighted average of utility multipliers associated with mRS 0, mRS 1 and mRS 2  **NODE 7b**  This produces an estimate of the utility multiplier associated with a dependent stroke using a weighted average of utility multipliers associated with mRS states 3, 4 and 5.  **BOOTSTRAPPING**  These commands run the bootstrapping function created earlier to produce 10,000 bootstrapped estimates of the centre of the distributions Stroke.Dep.utils and Stroke.Ind.utils. |
| **R code** | **Comments** |
| GOS\_5 <- mRS\_followingStroke[,1:2]  GOS\_4 <- mRS\_followingStroke[,3:4]  GOS\_3 <- mRS\_followingStroke[,5:6]  GOS\_5.sums <- apply(GOS\_5, 1, sum)  GOS\_4.sums <- apply(GOS\_4, 1, sum)  GOS\_3.sums <- apply(GOS\_3, 1, sum)  GOS\_5 <- apply(GOS\_5, 2, function (x) x / GOS\_5.sums)  GOS\_4 <- apply(GOS\_4, 2, function (x) x / GOS\_4.sums)  GOS\_3 <- apply(GOS\_3, 2, function (x) x / GOS\_3.sums)  GOS\_5.utils <- GOS\_5[,1] \* 1 + GOS\_5[,2] \* mult.s1  GOS\_4.utils <- GOS\_4[,1] \* mult.s2 + GOS\_4[,2] \* mult.s3  GOS\_3.utils <- GOS\_3[,1] \* mult.s4 + GOS\_3[,2] \* mult.s5  n.bootstraps <- 10000  GOS\_5.mean <- vector("numeric", n.bootstraps)  GOS\_4.mean <- vector("numeric", n.bootstraps)  GOS\_3.mean <- vector("numeric", n.bootstraps)  for (i in 1:n.bootstraps){  GOS\_5.mean[i] <- mean(GOS\_5.utils[sample(1:N.PSA, n.bootstraps, replace=T)])  GOS\_4.mean[i] <- mean(GOS\_4.utils[sample(1:N.PSA, n.bootstraps, replace=T)])  GOS\_3.mean[i] <- mean(GOS\_3.utils[sample(1:N.PSA, n.bootstraps, replace=T)])  } | Approach for mapping from mRS to GOS  Assuming code above has all been run (so mRS estimates and so on have all been calculated)  **NODE 4a + NOD 4b + NODE 4c**  These commands the Dirichlet derived cell counts into GOS 5 (columns 1 and 2), GOS 4 (columns 3 and 4), and GOS 3 (columns 5 and 6).  These commands calculate the sums across the rows of the newly created variables GOS\_5, GOS\_4, and GOS\_3  These commands convert the cell counts into proportions.  **NODE 6a + NODE 6b + NODE 6c + NODE 7a + NODE 7b + NODE 7c**  These commands calculate a weighted utility multiplier value for each row in GOS\_5, GOS\_4 and GOS\_3 given the relative proportion of each of the component states.  **BOOTSTRAPPING**  *These commands show how to perform bootstrapping without use of the Bootstrapper function developed earlier*  Sets the number of bootstrap replicates to 10,000  Creates three empty vectors for storing bootstrapped estimates of the means  Produces 10,000 bootstrapped estimates of the means of GOS\_5, GOS\_4 and GOS\_3 |

Table 7 Description of full R script used to perform the analyses