How to use RSGHB

Jeffrey Dumont, Jeff Keller

June 2015

Abstract

This vignette describes the process for specifying, estimating and analyzing the output of a choice model with RSGHB. Two case studies with different model structures are used as examples. The synthetic choice data set *choicedata* used in this document is included in the RSGHB package or can be downloaded from the RSGHB github page: `https://github.com/jeffdumont/RSGHB`

RSGHB Code Structure

The code typically needed for running an RSGHB model can be broken into five main sections and the examples that follow are organized in this manner.

1. Setup and Data Preparation
2. Defining the Likelihood Function
3. Model Controls and Settings
4. Model Estimation
5. The Output

Previous versions of RSGHB produced a series of CSV files as output. Newer versions of RSGHB produce a model object of class *RSGHB* that can be operated on directly. However, the old behavior can be restored by using the *writeModel* function or by setting the *writeModel* argument in *doHB* to *TRUE*.

EXAMPLE 1: MNL Model with Fixed Parameters

In this section, the code for estimating a multinomial logit (MNL) model with fixed (non-random) parameters is explained. The data set for this example is composed of synthetic individuals that were presented with a choice between two travel alternatives - one that is toll-free and one that had a toll but also had a faster travel time. Each of 1,138 individuals was presented with a panel of 9 choice tasks.

Setup and Data Preparation

```
> library(RSGHB)
> # Load example data
> data(choicedata)
> # This data set is organized as one row per choice observation. This isn't necessary
> # but it does make writing the likelihood function straightforward.
> head(choicedata)
> # Any variables or transformations of variables from the choicedata data.frame that are
> # needed for evaluating the likelihood function can be extracted as a series of vectors
> # for convenience (e.g., TT1, TT2, TOLL2). Alternatively, the likelihood function could
> # refer to the choicedata data.frame directly. In this example, each alternative is
> # defined by travel times and toll costs.
> TT1   <- choicedata$tt1
> TT2   <- choicedata$tt2
```

```
> TOLL2 <- choicedata$toll2
> # Similarly for the vectors of choices. Note that in this example, there are only two
> # alternatives. Dummying coding the choice vector is not necessary but again makes
> # writing the likelihood function straightforward.
> choice1 <- (choicedata$Choice==1)
> choice2 <- (choicedata$Choice==2)
> # Frequency of choices
> table(choicedata$Choice)
>
```

Defining the Likelihood Function

The estimation routine expects a user-specified *likelihood* function that accepts two arguments, *fc* and *b*, that, in concert with the choice data, returns a vector of probabilities of length equal to the number of total choice tasks across all respondents. With 1,138 respondents and 9 choice tasks each, the *likelihood* function should return a vector of length 1,138*9=10,242 in this case.

The *fc* argument is a vector of fixed coefficients (they do not vary across individuals). The *b* argument is a matrix of individual coefficients which are generated from the random coefficients in the model. The *b* matrix is discussed in more detail in Example 2.

It is important to note that the structure of the *likelihood* function and choice data set is arbitrary, and completely up to the user, so long as the *likelihood* function accepts *fc* and *b* as arguments and returns a vector of probabilities of appropriate length. However, the *likelihood* function is the most computationally taxing part of the estimation process, so coding this function efficiently is essential to reducing the run time of the model.

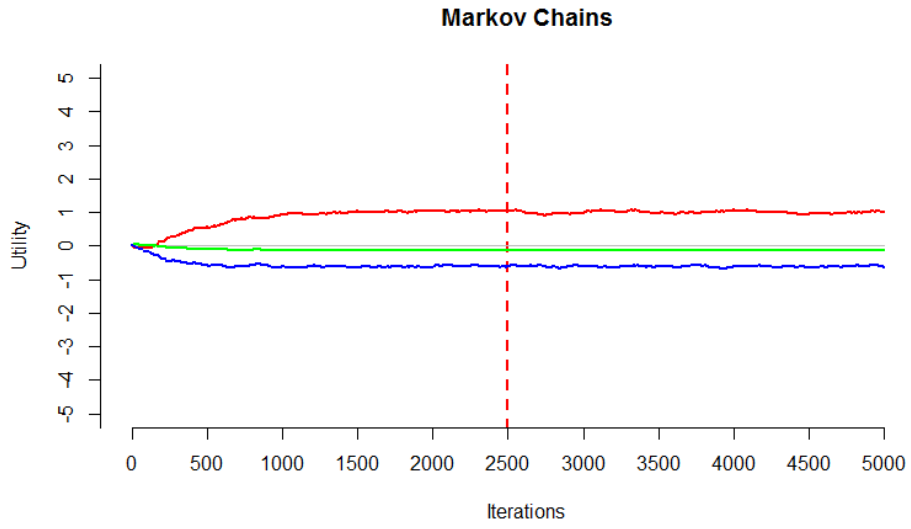
```
> # The likelihood function
> likelihood <- function(fc, b) {
+
+   # Assign fixed parameters to named variables for convenience
+   cc <- 1
+   ASC1  <- fc[cc]; cc <- cc + 1
+   Btime <- fc[cc]; cc <- cc + 1
+   Btoll <- fc[cc]; cc <- cc + 1
+
+   # Utility functions
+   v1 <- ASC1 + Btime * TT1
+   v2 <-        Btime * TT2 + Btoll * TOLL2
+
+   # MNL probability statement
+   p  <- (exp(v1) * choice1 + exp(v2) * choice2) / (exp(v1) + exp(v2))
+
+   return(p)
+ }
```

Model Controls and Settings

There are a number of options and settings for controlling the model estimation process. Please see the help file for *doHB* for more details. Note that most controls have default values and do not need to be directly specified if the default is acceptable.

```
> ### Setting control list for estimation (see ?doHB for more estimation options)
>
> # The model name/description
> modelname <- "MNL"
> # gVarNamesFixed contains the names for the fixed (non-random) variables in the model
> # These will be used in the output and also when displaying iteration detail to
> # the screen
> gVarNamesFixed <- c("ASC1", "BTime", "BCost")
> # FC contains the starting values for the fixed coefficients
> FC <- c(0, 0, 0)
```

Figure 1: Plotting of the Markov Chains during estimation




```
> # gNCREP contains the number of iterations to use prior to convergence
> gNCREP <- 2500
> # gNEREP contains the number of iterations to keep for averaging after convergence
> # has been reached
> gNEREP <- 2500
> # gNSKIP contains the number of iterations between retaining draws for averaging
> gNSKIP <- 1
> # gINFOSKIP controls how frequently to print info about the iteration process
> gINFOSKIP <- 10
> # gSeed ensures reproducible results
> gSeed <- 1987
> # To simplify the doHB function call, all of the control arguments are placed in
> # a single list that can be passed directly to doHB
> control <- list(modelname = modelname,
+                 gVarNamesFixed = gVarNamesFixed,
+                 FC = FC,
+                 gNCREP = gNCREP,
+                 gNEREP = gNEREP,
+                 gNSKIP = gNSKIP,
+                 gINFOSKIP = gINFOSKIP,
+                 gSeed = gSeed)
>
```

Model Estimation

To start the model estimation process, the analyst passes the *likelihood* function, *choicedata* data.frame, and the *control* list to the *doHB* function.

```
model <- doHB(likelihood, choicedata, control)
```

Note that the only requirement of the *choicedata* data.frame is that it contains a column *ID* which identifies the individuals associated with the vector of probabilities returned by the *likelihood* function.

The *doHB* function will perform a series of diagnostic tests on the model inputs to catch common errors in model setup. It also provides basic summary statistics of the choice data and model. Before estimation begins, an optional confirmation prompt allows the user to cancel the model estimation if needed.

During estimation, current estimates of the Markov Chains will be plotted to the screen. This plot is updated based on the control parameter *gINFOSKIP* (Figure 1). In addition, numerical iteration details are provided in the R Console. Printing and plotting during model estimation takes time, so increasing *gINFOSKIP* will reduce estimation time.

The Output

The *doHB* function returns a model object of class *RSGHB*. See *?doHB* for a detailed description of the components of an *RSGHB* class model object.

In addition to the modeling controls and settings, the model object includes an *iter.detail* component that contains model statistics at every *gINFOSKIP*th iteration. These can be used to assess whether the model has converged. This model has only fixed parameters, so only the Log-Likelihood, Root-Likelihood (RLH), and Acceptance Rate for the fixed parameters are meaningful.

```
> # Model iteration details
> head(model[["iter.detail"]])
>
```

The RSGHB plot method visualizes these model statistics conveniently.

```
> # Plot model statistics
> plot(model)
>
```

Similarly, supplying the *type* argument to the plot method will plot the fixed parameter estimates, which are contained in the F component of the model object.

```
> # Plot parameter estimates (see ?plot.RSGHB for more uses)
> plot(model, type = "F")
>
```

EXAMPLE 2: MNL Model with Random Coefficients

In this section, we expand on the model estimated in Example 1 by allowing the coefficients to vary across the individuals in our dataset. This type of model is referred to by many names - Random Coefficients Logit, Random Parameters Logit or Mixed Logit.

Setup and Data Preparation

The setup and data preparation are identical to Example 1.

Defining the Likelihood Function

To introduce random effects into the model, the b matrix is used in place of the fc vector in the coding of the *likelihood* function. The b matrix contains the individual conditionals for the sample-level random coefficients. The matrix b has one row per choice task (associated with the vector of probabilities returned by the *likelihood* function) and one column for each of the random parameters.

```
> likelihood <- function(fc, b) {
+
+      # Note that the fc argument is still supplied, but is unused
+
+      # Using b instead of fc is the only change
+      cc     <- 1
+      ASC1   <- b[, cc]; cc <- cc + 1
+      Btime  <- b[, cc]; cc <- cc + 1
+      Btoll  <- b[, cc]; cc <- cc + 1
+
+      # Utility functions
+      v1 <- ASC1 + Btime * TT1
+      v2 <-        Btime * TT2 + Btoll * TOLL2
```

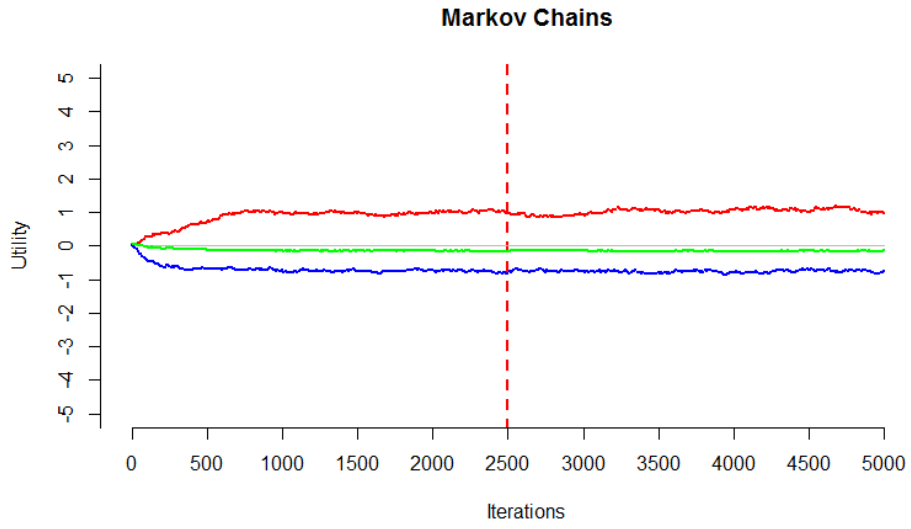
```
+
+      # MNL probability statement
+      p  <- (exp(v1) * choice1 + exp(v2) * choice2) / (exp(v1) + exp(v2))
+
+      return(p)
+ }
>
```

Model Controls and Settings

A few more controls can be supplied to the *doHB* function when random parameters are to be estimated.

```
> ### Setting control list for estimation (see ?doHB for more estimation options)
>
> # The model name/description
> modelname <- "MMNL"
> # gVarNamesNormal provides names for the random parameters in the same way
> # gVarNamesFixed does for the fixed parameters
> gVarNamesNormal <- c("ASC1","BTime","BCost")
> # svN contains the starting values for the means of the normal distributions for each
> # of the random parameters
> svN <- c(0, 0, 0)
> # gDIST specifies the type of continuous distribution to use for the random parameters
> # gDIST must have an entry for each value in gVarNamesNormal
> # The options are:
> # 1. normal
> # 2. log-nomal
> # 3. negative log-normal
> # 4. normal with all values below zero massed at zero
> # 5. normal with all values greater than zero massed at zero
> # 6. Johnson SB with a specified min and max
>
> # In this example, normal distributions are used for all 3 parameters
> gDIST <- c(1, 1, 1)
> # gNCREP contains the number of iterations to use prior to convergence
> gNCREP <- 2500
> # gNEREP contains the number of iterations to keep for averaging after convergence
> # has been reached
> gNEREP <- 2500
> # gNSKIP contains the number of iterations between retaining draws for averaging
> gNSKIP <- 1
> # gINFOSKIP controls how frequently to print info about the iteration process
> gINFOSKIP <- 10
> # gSeed ensures reproducible results
> gSeed <- 1987
> # To simplify the doHB function call, all of the control parameters are placed in
> # a single list that can be passed directly to doHB
> control <- list(modelname = modelname,
+                 gVarNamesNormal = gVarNamesNormal,
+                 gDIST = gDIST,
+                 svN = svN,
+                 gNCREP = gNCREP,
+                 gNEREP = gNEREP,
+                 gNSKIP = gNSKIP,
+                 gINFOSKIP = gINFOSKIP,
+                 gSeed = gSeed)
>
>
```

Figure 2: Plotting of the Markov Chains during estimation



Model Estimation

As in Example 1, the model estimation process is initiated with the analyst supplying the *likelihood* function, *choicedata* data.frame, and the *control* list to the *doHB* function.

```
model <- doHB(likelihood, choicedata, control)
```

In this model, the Markov Chain values plotted represent the means of the underlying normals for the random parameters (Figure 2).

The Output

The *RSGHB* model object has more components in the case of a random or mixed effects model.

The *iter.detail* component now contains the additional model statistics: Parameter Root Mean Square (RMS) and Average Variance at each stored iteration.

```
> # Model iteration details
> head(model[["iter.detail"]])
> # Plot model statistics
> plot(model)
>
```

The *A* component contains the sample-level means of the underlying normal. These can be plotted with the plot method.

```
> # Sample-level means
> head(model[["A"]])
> # Plot sample-level means
> plot(model, type = "A")
>
```

The *B* component contains the average, across iterations, of the individual level draws for the underlying normals for the random parameters. The *Bsd* component provides the standard deviations of those individual draws.

```
> # Average individual-level draws
> head(model[["B"]])
> # Standard deviations of individual-level draws
> head(model[["Bsd"]])
>
```

The C component contains the average across iterations of the individual level draws for the random parameters including the appropriate distribution transformations (if applicable). The C component also contains the Root-Likelihood (RLH), an individual-specific measure of model fit. The Csd component provides the standard deviations of those individual draws.

These two components are asymptotically equivalent to the conditional distributions calculated from mixed logit models estimated using Maximum Simulated Likelihood methods.

```
> # Average individual-level draws (transformed; if applicable)
> head(model[["C"]])
> # Standard deviations of individual-level draws (transformed; if applicable)
> head(model[["Csd"]])
>
```

The D component contains a three-dimensional p by p by $gNEREP$ array containing the sample covariance at each iteration, where p is the number of random parameters.