# Hierarchical Bayes

In this exercise you will write your own bayesian estimation routine to estimate a mixed logit model. Most matlab code will be given in the text in some form, but you will need to create the new files containing the functions you will write yourself. It is a good idea to start with very simple specifications to make sure that the estimation works, before you try to estimate more elaborate models.

## Hierarchical Bayes estimation

The idea with Hierarchical Bayes (HB) is to sample from the posterior, i.e., the distribution of the parameters given the data. The mean of this distribution plays the same role as the point estimates obtained through maximum likelihood estimation, and the covariance of the sample corresponds to the inverse of the Hessian.

In a simulated maximum likelihood (SML) estimation we are estimating the mean $b$ and (co)variance matrix $W$ of the mixed parameters $\beta_n$ as well as the fixed parameters $\alpha$. In HB it turns out to be easier to directly consider the individual specific parameters $\beta_n$. Instead of sampling from $L(b, W, \alpha|Y)$ we sample over $L(b, W, \vec{\beta}, \alpha|Y)$. The posterior of the parameters is given by

$$L(\beta_n \, \forall n, \alpha, b, W|Y) \propto \prod L(y_n|\beta_n, \alpha)\phi(\beta_n|b, W)P(b, W). \tag{1}$$

Sampling $b$, $W$, $\beta_n$ and $\alpha$ directly using the joint probability $L(\beta_n \, \forall n, \alpha, b, W|Y)$ in (1) would be theoretically possible but very hard. However, it is relatively easy to sample from *conditional* distributions of $b$, $W$, $\beta_n$ and $\alpha$, and Gibbs sampling can therefore be used to sample from $L(\beta_n \, \forall n, \alpha, b, W|Y)$ through the following four steps:

1. Draw mean $b|W, \beta_n \, \forall n, \alpha, Y$. The conditional distribution of $b$ can be sampled from directly.
2. Draw variance $W|b, \beta_n \, \forall n, \alpha, Y$. This is also easy, and as for $b$ it can be obtained directly.
3. Draw $\beta_n|b, W, \alpha, Y$. The distribution for $\beta_n$ is hard to sample from directly so Metropolis-Hastings algorithm must be used.
4. Draw $\alpha|b, W, \beta_n \, \forall n, Y$. This distribution is also hard, and Metroplis-Hastings algorithm must be used.

These four steps are repeated until the fixed parameters $\alpha$, mean $b$ and variance $W$ converge. Once they have converged we retrieve samples from the posterior $L(\alpha, b, W|Y)$. Observe that the way in which the conditional distributions of $b$, $W$, $\beta_n$ and $\alpha$ are used instead of the joint distribution is what constitutes Gibbs sampling, and that Metroplis-Hastings must be used within the Gibbs sampling for some of the parameters.

## Assignment

The main part of the program you will be writing will look like this (but you do not have to create it right now):

```
for (n = 1:N_iterations)
% Update all parameters
[B,W,R,F,P,RhoR,acceptF,acceptF] = SampleParameters(B,W,R,F,RhoR);

% Save parameters for analysis
% Plot paramaters

end
```

where

- `B` is a vector of the mean of the random parameters, so $|B| = N_{rd}$.
- `W` is the covariance matrix of the random parameters, so $|W| = N_{rd} \times N_{rd}$.
- `R` is a matrix of $\beta_n$ for all individuals, so $|R| = N_{rd} \times N_{Obs}$. `R(1,:)` gives the first parameter for all observations and `R(:,1)` gives all parameters for the first observations.
- `F` is a vector of the fixed parameters $\alpha$, so $|F| = N_{fx}$
- $P(n) = L(y_n|\beta_n, \alpha)$, so $|P| = N_{Obs}$
- `Rho` $= \rho$ is the step size used when updating $R$, so it is a scalar

In the end, your final SampleParameters function will look like this:

```
function [B,W,R,F,P,RhoR,acceptR,acceptF] = SampleParameters(B,W,R,F,P,RhoR)

B = nextB(W,R);     % Step 1 : Update mean of random parameters

W = nextW(B,R);     % Step 2 : Update covariance matrix of random parameters

[R,P,RhoR,acceptR]=nextR(B,W,R,F,P,RhoR);     % Step 3 : Update vector of individual ...
    specific coefficients R, and update Rho

[F,P,acceptF] = nextF(R,F,P);     % Step 4 : Update fixed parameters
end
```

We will start with only sampling the fixed parameters. You can start by creating a new file in matlab named SampleParameters.m and add:

```
function [B,W,R,F,P,Rho,acceptF] = SampleParameters(B,W,R,F,P,Rho)

[F,P,acceptF] = nextF(R,F,P);     % Step 4 : Update fixed parameters
end
```

**Q1: Update $\alpha$**

We will start by writing the function `nextF(R,F,P)` as it is the only function needed to estimate an MNL model. You will in the next question (Q2) validate this function by comparing it with a maximum likelihood estimation on the same model specification.

**a) Create `nextF` in matlab**

After the function SampleParameters, create a new function:

```
function [F_new,P_new,acceptF] = nextF(R,F,P)
% NEXTF updates the fixed parameters F given the random parameters R
%    and the choice probabilites P
%  INPUT
%       R : random parameters, N_RD x NP
%       F : old fixed parameters,  N_FX x 1
%       P : old choice probabilties, NP x 1
%  OUTPUT
%       F_new : new fixed parameters, N_FX x 1
%       P_new : new choice probabilities, NP x 1
%       acceptF : 1 if F is updated and 0 if F is unchanged. 1x1
global N_FX CHOICEIDX
% N_FX      : Number of fixed parameters. N_FX=length(F)
% CHOICEIDX : index of choosen alternatives

end
```

`P` here is the vector `P_choice` of choice probabilities for each individual (the same that was calculated in lab1a) given the parameters `R` and `F`.

**b) Construct candidate parameters**

One way of doing this is to let `F_cand = F + RhoF * randn(N_FX, 1)` where you must first specify `RhoF`. It should be such that you accept the trial parameters approximately 30% of the time, and you will have to tune it manually yourself (a good start for the provided specification can be 0.1). Its value will depend on the number of fixed parameters as well as their size, so you will have to change it if you change your model specification.

**c) Calculate the acceptance probability**

Remember from the lecture:

$$L(\alpha|Y,b,W,\beta_n\,\forall n) \propto \prod_n L(y_n|\alpha,\beta_n) = L(Y|\alpha,\beta_n\,\forall n)$$

The acceptance probability in a metropolis hastings algorithm where $Q(\alpha_{\text{cand}}|\alpha) = Q(\alpha|\alpha_{\text{cand}})$ is therefore given by:

$$p_{accept} = \min(1, \prod_n \frac{L(y_n|\alpha_{\text{cand}},\beta_n)}{L(y_n|\alpha,\beta_n)}). \tag{2}$$

The function $L(y_n|\alpha,\beta_n)$ is the choice probability for each individual. The input `P` should contain $L(y_n|\alpha,\beta_n)$, i.e., the likelihood for the current parameters. If you calculate `P_cand` using the `Logit` function as in Lab1a ( `P_all_alt = Logit_HB(F_cand,R)` and `P_cand = ... P_all_alt(CHOICEIDX)` ), the acceptance probability $p_{accept}$ can be calculated as:

```
p_accept = prod(P_cand./P);
```

Observe that we do not need to use `min(prod(P_cand./P),1)` as in Eq. (2) since we will only use `p_accept` to test if a random number $r \in (0,1)$ is less than `p_accept`.

**d) Accept/Reject candidate parameters**

When updating the fixed parameters $\alpha$, you either accept all new parameters or reject all new parameters. Draw a variable `r = rand(1,1)`, and if `r<p_accept`, the parameters are accepted. One way of doing this is to create a variable `acceptF = r<p_accept` which is 1 if you accept the new parameters and 0 otherwise. The new parameters are then given by: `F_new = F * ... (1-acceptF)+ F_cand * acceptF` and the new probabilities `P_new` are given in the same way.

**Q2: Estimate an MNL model using `nextF(R,F,P)`**

You will first need to specify the variables and create initial values of all parameters. Create a new file (this is the main file) and add the following lines of code:

```
global NDRAWS N_RD NP N_FX CHOICEIDX LAB_RD LAB_FX
% NDRAWS : number of draws used for mixed variables
% N_RD : Number of random parameters
% N_FX : Number of fixed parameters
% NP : Number of observations
% CHOICEIDX : Matrix used to get choosen alternatives
% LAB_RD : Label of random parameters in the order which they appear in R
% LAB_FX : Label of fixed parameters in the order which they appear in F

NDRAWS=1; % Set number of draws to 1 since each individual only
          % have one value for each parameter
SpecifyVariables(0); % Only specify fixed variables at this stage

F = zeros(N_FX,1); % Initialize fixed parameters

 B = zeros(N_RD,1); % Initialize mean of random parameter
 W = N_RD*eye(N_RD); % Initialize covariance matrix for random parameters
 R = repmat(B,1,NP)+chol(W)'*randn(N_RD,NP); % Initialize random variables

P = Logit_HB(F,R); %  Probability of all alternatives
P = P(CHOICEIDX);  % Probability of choosen alternatives
```

**a) Burn in**

After this, you will sample parameters. Metropolis hastings and gibbs sampling only produces samples from the correct distribution once they have converged. You will therefore need a "burn in" period, before you can actually start sampling from the posterior. The one suggested below (60 000) might be to high or low depending on your specification.

Sampling can be done through:

```
Rho = 0.01; % Initial value of rho
nburnin = 60000;
for(k=1:nburnin)
    [B,W,R,F,P,Rho,acceptF] = SampleParameters(B,W,R,F,P,Rho); %Sample parameters
```

```
end
```

*Hint:* To overlook the progress and make sure that it actually converge it is a good idea to add some plots over the parameters, e.g., plot every 100:th iteration. It is also a good idea to check the acceptance rate for $\alpha$, so that you can fine-tune `RhoF` in `nextF` if it is to high/low.

Add `fSaved = []` and `acceptF_total = 0;` before the for loop. Then add these lines after `SampleParameters` but within the for-loop to plot all fixed parameters and print out the acceptance rate for `F`:

```
    acceptF_total = acceptF_total+acceptF; %Increase by one if accepted the new F
    if(mod(k,100)==1)
        fprintf('Acceptance rate F: %2.3f \n',acceptF_total/100); %Print acceptance rate
        acceptF_total=0; %Reset acceptance rate
        fSaved = [fSaved,F];    %Save parameters
        nSaved = size(fSaved,2);   %Number of saved parameters
    end
    if(mod(k,1000)==1)
        for(i = 1:N_FX)
            subplot(ceil(N_FX/2),2,i); %plot the fixed parameters in different sub-plots
            plot([1:nSaved]*100,fSaved(i,:));
            ylabel(LAB_FX{i});
        end
        drawnow;
    end
```

**b) Save values**

Once the model has converged you can start saving values for analysis. Create another for-loop (or modify your current one) and save, e.g., every 100:th sample until you have enough, e.g., 200 samples. You can save these values in the same way that you saved in `fSaved` above. Calculate the mean and variance of these samples. This is the mean and variance of the parameter estimates. If `fSaved` is the matrix of sampled parameters that you have saved, the following lines would print the final mean and variance for all fixed parameters:

```
fprintf('Fixed Parameters \n')
fprintf('%-20s : %6s %6s\n','Parameter','EST','T-test');
Fmean = mean(fSaved,2);
FVar = cov(fSaved');
for(i = 1:N_FX)
    fprintf('%-20s : %6.2f %6.2f\n',LAB_FX{i},Fmean(i),Fmean(i)/sqrt(FVar(i,i)));
end
```

**Q3: Update $b$**

**a) Write the function `nextB(W,R)`**

Start with:

```
function [B_new] = nextB(W,R)
% NEXTB takes draws from the mean B conditional on
% the covariance matrix W and individual parameters R
%   INPUT
%       W : covariance matrix, N_RD x N_RD
%       R : individual parameters, N_RD x NP
%   OUTPUT
%       B_new : new values of the mean B, N_RD x 1

global N_RD NP
% N_RD : Number of random parameters
% NP : Number of observations

end
```

Remember from the lecture:

$$b|W, \beta_n \forall n, \sim N(\bar{\beta}, \frac{W}{N}) \tag{3}$$

where $N$ is the number of observations (i.e, `NP`) and $\bar{\beta}$ is the mean of the current random parameters over all individuals, i.e., `R_mean = mean(R,2)`.

*Hint:* A sample $x$ of $N_{rd}$ elements from a multivariate normal distribution with mean vector $m$ and covariance matrix $Z$ can obtained in the following way:

(i) Create a vector with $N_{rd}$ standard normal distributed variables: `r = randn(N_RD,1)`

(ii) Multiply with the cholesky-factor of $W$ to get the correct covariance structure: `rvar = ... chol(Z)'*r` (note the transpose)

(iii) Add the mean: `x = m+rvar`. Now $x$ is a sample from $N(m, z)$.

**b) Why do we divide the covariance matrix $W$ by $N$? (You don't need to write the formulas, just explain in words why this is logical)**

**Q4: Update $W$**

We will in this question make a draw from the posterior when the covariance matrix is assumed to be diagonal.

**a) Write the function `nextW(B,R)`**

Start with:

```
function [W_new] = nextW(B,R)
% NEXTW takes draws of the covariance matrix W when W is
% assumed to be diagonal. B is the mean and R is the
% individual level parameters.
%   INPUT
%       B : mean of random parameters, N_RD x 1
%       R : individual parameters, N_RD x NP
%   OUTPUT
%       W_new : new covariance matrix, N_RD x N_RD
global N_RD NP
```

```
% N_RD : Number of random parameters
% NP : Number of observations


end
```

The variance for the random parameter with index $k$ in `R` is given by

$$P(W(k,k)|\beta, b) \propto IG(v_1, s_1)$$

where

$$v_1 = 1 + N_{Obs}$$
$$s_1 = \frac{1 + N_{Obs}\bar{s}}{v_1}$$

where $N_{Obs}$ is the number of observations (`NP`), $N_{rd}$ is the number of random variables (`N_RD`), and $\bar{s}$ is the variance of $\beta_n(k)$ around $b$, i.e.,

$$\bar{s} = \frac{1}{N_{Obs}} \sum_{n=1}^{N_{Obs}} \left(\beta_n(k) - b(k)\right)^2 . \tag{4}$$

A draw from the inverted gamma can then be obtained by (see Train page 299):

(i) Take $v_1$ draws from a standard normal distribution, call them $\mu$

(ii) Calculate $r = 1/v_1 \sum_i \left(\frac{\mu}{\sqrt{s_1}}\right)^2$, which is the sample variance of a normal distribution with variance $s_1$.

(iii) $W(k,k) = 1/r$

In matlab, this can be done for all variables at once using vector notations. Start with creating `v_1 = 1+NP`, `s_bar = mean((R-repmat(B,1,NP)).^2,2)` and `s_1 = (1+NP*s_bar)/ v_1`. Then:

(i) Take draws from a standard normal distribution: `mu = randn(N_RD,v_1)`

(ii) Calculate $r$: `r = 1/v_1*sum(mu.^2./repmat(s_1,1,length(mu)),2)`

(iii) Finally, create the covariance matrix: `W_new = diag(1./r)`

**b) What are the properties of Inverted Wishart and Inverted Gamma distributions that makes them feasible for the covariance matrix?**

**Q5: Update $\beta_n$**
Remember from lecture:

$$L(\beta_n|y_n, b, W) \propto L(y_n|\beta_n)\phi(\beta_n|b, W).$$

Updating $\beta_n$ and $\rho$ is done in a number of steps:

(i) Construct candidate parameters for each individual, for example as

$$\beta'_n = \beta^t_n + \rho r_n$$

where $r_n \sim \mathrm{N}(0, W)$

(ii) Calculate the acceptance probability:

$$a_n = \frac{L(y_n|\beta'_n)\phi(\beta'_n|b, W)}{L(y_n|\beta^t_n)\phi(\beta^t_n|b, W)}$$

where:

$$\phi(x|b, W) = c \cdot e^{-\frac{1}{2}(x-b)'W^{-1}(x-b)}$$

(iii) If $a_n \geq 1$, $\beta^{t+1}_n = \beta'_n$, otherwise:

$$\beta^{t+1}_n = \begin{cases} \beta'_n \text{ with probability } a_n \\ \beta^t_n \text{ with probability } 1 - a_n \end{cases}$$

(iv) Update $\rho$ so that the share of accepted new parameters approaches 0.3:

$$\rho_{new} = \rho - 0.001 \cdot (N_{accept}/N_{Obs} > 0.3) + 0.001 \cdot (N_{accept}/N_{Obs} < 0.3)$$

**a) Write the function `nextR(B,W,R,F,Pold,Rho)`.**

Start with:

```
function [R_new,P_new,RhoR_new,accShare] = nextR(B,W,R,F,P,RhoR)
% NEXTR updates the individiual specific parameters R
% INPUT
% B : mean for random parameters, N_RD x 1
% W : variance for random parameters, N_RD x N_RD
% R : old individual specific parameters, N_RD x NP
% F : fixed parameters, N_FX x 1
% P : old choice probabilties, given by F and R, NP x 1
% RhoR : current step size for update of random parameters, 1x1
% OUTPUT
% R_new : new individual specific parameters, N_RD x NP
% P_new : new choice probabilities, given by F and R_new, NP x 1
% RhoR_new : new step size, 1x1
% accShare : share of individuals for which R is updated. 1x1
global N_RD NP CHOICEIDX
% N_RD : Number of random parameters
% NP   : Number of observations
% CHOICEIDX : index of choosen alternatives

end
```

The steps described above can be specified using vector notations in MATLAB to gain considerable speedup.

(i) Create candidate parameters: `R_cand = R + RhoR*chol(W)'*randn(N_RD,NP)`

(ii) Calculate acceptance probabilities.

(a) Calculate candidate probabilities `P_cand` for all individuals, as in Q1.

(b) Calculate $\phi(x|b,W)/\phi(x'|b,W) = e^{-\frac{1}{2}((x-b)'W^{-1}(x-b)-(x'-b)'W^{-1}(x'-b))}$. If you create `d_R_cand = R_cand - repmat(B,1,NP)` and `d_R = R - repmat(B,1,NP)` you can calculate this quotient with:

```
phi_quote = exp( -0.5*( sum(d_R_cand .* (W \ d_R_cand),1) - sum(d_R ...
    .* (W \ d_R),1)  ));
```

(c) The acceptance probability (for each individual) finally becomes `p_accept = phi_quote' ... .* P_cand ./ P;`

(iii) Accept/reject new values. Draw random values `r = rand(NP,1)` and create a vector with 1 for the individuals whose parameters should be updated: `accept = r<p_accept`.

(iv) Update `RhoR`. The share of acceptances is given by `accShare = sum(accept)/NP`, so the new rho is given by:

```
RhoR_new = RhoR - 0.001 .* (accShare < 0.3) + 0.001 .* (accShare > 0.3);
```

(v) Finally update the parameters and choice probabilities:

```
P_new=P_cand .* accept + P .* (1-accept);
R_new=R_cand .* repmat(accept',N_RD,1) + R .* repmat((1-accept'),N_RD,1);
```

**b) Why don't we need to calculate the acceptance probability when updating $b$ and $W$?**

**Q6: Estimate a mixed logit model using Hierarchical Bayes**
Now you can add a mixed variable to the model you estimated in Q2 and see if it works. Compare with the simulated maximum likelihood result from lab1b. Add plots, print out the acceptance rate and check for convergence as you did in Q2. Note that you only need to save the diagonal of the covariance matrix `diag(W)` and that `acceptR` already is the share, so you can print that value directly.

# 1 Extra assignments

For a higher grade on Lab 2 you are required to complete the following extra assignments. Observe that the overall quality of the report will be assessed.

## For C

**Q7: Analyze individual specific variables**
Analyze how the individual specific parameters varies with socio demographics in the population. For example, you could check if men and women have different cost-preferences by analyzing how the cost-parameter varies between the two groups. You can have a single cost parameter and see how it varies with the chosen mode, or have a simple model with mixed ASC's and try to find new variables to include into respectively modes utility function. Try to introduce segmented parameters based on your findings and see if they improve the model.

We have found that it is quite hard to find strong relationships between socio demographics and the individual specific parameters in this data set. If you cannot find any such relationships, focus on how the parameters varies with the chosen mode.

## for B

For B, you should do one of the following two questions.

**Q8: Introduce transformations of the normal distribution**
Try different transformation (at least two) such as log-normal and truncated normal, on the random parameters. To make it easier to code, it is OK to apply the same transformation to all random parameters.

**Q9: Enable a full covariance matrix**
In the current version the covariance matrix $W$ can only be diagonal. Write a new `nextW` that enables a full covariance matrix.

## for A

For A, you should also do the following question:

**Q10: How does knowledge about individual specific parameters effect prediction result?**
Use obtained individual specific parameters to calculate the mode shares conditional on the chosen mode, i.e., the last table that was printed out in the previous labs. Compare with the result you get when you simulate the mode shares as in Lab 1b.