An introduction to *BayesPref*

The package *BayesPref* will implement a hierarchical Bayesian analysis of count data.

Data format:
BayesPref requires that data be provided as an object class matrix. The data provided in "YGGVdata" is oviposition preference data for *Lycaeides* populations from Yuba Gap, CA and Gardnerville, NV. Here, the populations have been coded numerically. That is, Yuba Gap is population 1 and Gardnerville is population 2. The abbreviations for plant species are as follows: Ast, *Astragalus whitneyi*, Lotus, *Lotus nevedensis*, Lupine, *Lupinus polyphyllus*, and Medicago, *Medicago sativa.*

YGGVdata

Pop	Ast	Lotus	Lupine	Medicago
1	12	4	2	4
1	0	2	1	0
1	6	8	13	9
1	6	7	1	0
1	21	13	1	1
1	10	8	5	1
1	17	4	0	2
1	7	0	2	0
1	6	3	6	4
1	12	14	2	3
1	2	0	0	2
1	2	8	1	11
1	3	0	0	1
2	1	0	0	2
2	2	1	5	3
2	2	4	6	3
2	0	1	0	0
2	0	0	0	1
2	6	6	5	3
2	2	15	1	0
2	0	0	0	6
2	0	0	1	0
2	6	3	0	6
2	0	0	0	1
2	5	4	5	14
2	6	3	1	2

Note: If only one population is to be examined, the column indicating populations is not required (see below).

BayesPref analysis:
The basic hierarchical Bayesian analysis of this data can be accomplished by first converting the object containing the data to a matrix:

```
YGGV<-as.matrix(YGGV)
```

Then using the function **bayesPref**:

```
YGGVpref<-bayesPref(pData=YGGV,mcmcL=5000,pops=TRUE,dicburn=1000)
```

pData indicates the matrix of count data, mcmcL the number of steps in the MCMC. pops=TRUE (the default) indicates that the first column is the population (or experiment) identifier. If only one population is present, this column is not required and the arguement pops=FALSE should be included. The argument dicburn=100 indicates the length of the burnin for the calculated DIC score, which can be useful for comparing models.

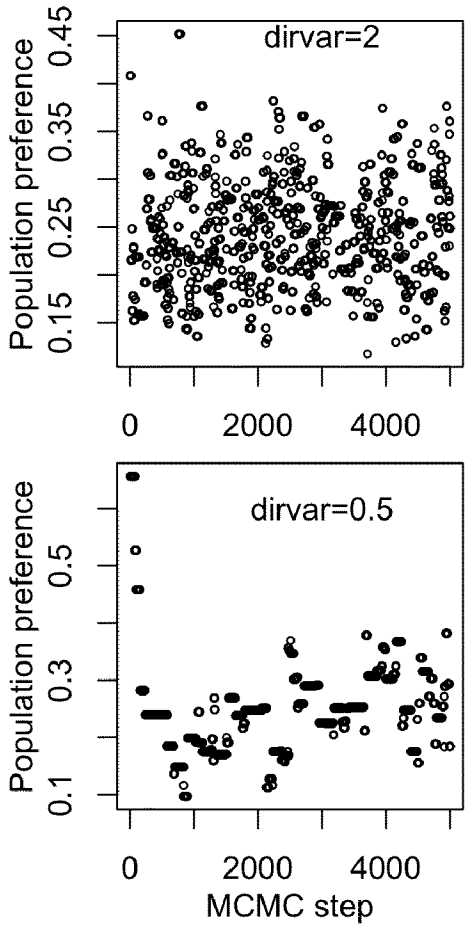
The output, here assigned to object YGGVpref, is a list that includes the individual-level preferences, population-level preferences, population variance parameter, the likelihood, and the probability of the model at each step in the MCMC, as well as the DIC score for the model.

```
> names(YGGVpref[[1]])
[1] "IndPref"    "PopPref"    "PopVar"     "likelihood" "pMD"        "dic"
```

The defaults result in a relatively short MCMC and it is recommended that the chain (and burn in for dic) be larger. The mixing of the chains can be examined by observing the population preference parameter for a given plant across the MCMC. For example, the MCMC mixing for the Gardnerville population (population 2) can be observed as follows:

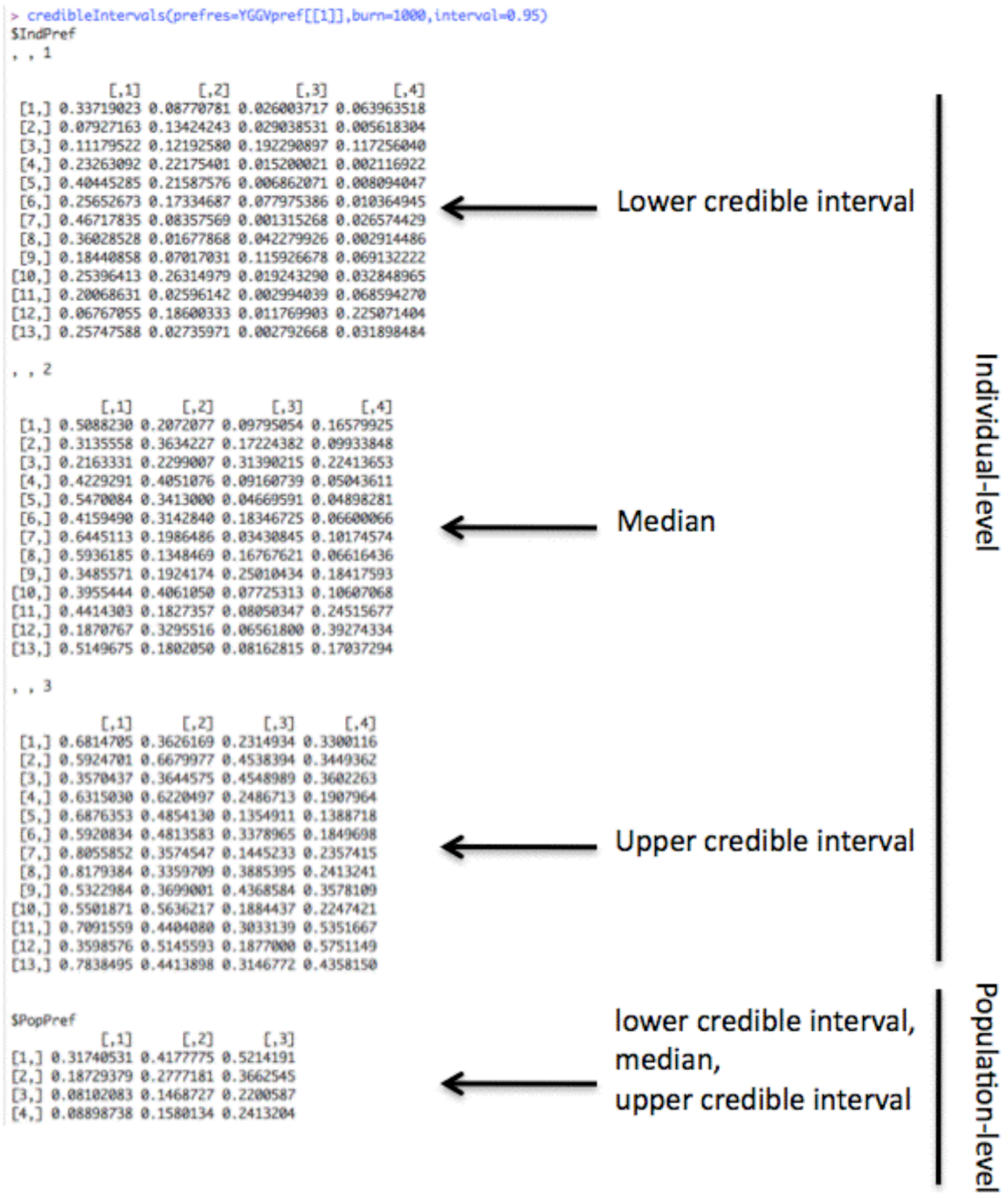
```
plot(YGGVpref[[2]]$PopPref[1,],xlab="MCMC
step",ylab="PopPref")
```

The plots below show the chain mixing for Yuba Gap. The top plot shows a well mixing chain. The bottom plot shows a poorly mixing chain. Chain mixing was affected by the argument dirvar in the **bayesPref** function. The dirvar argument is a multiplier that affects the proposal distance in the MCMC. Decreasing the value increases the proposal distance. The top plot is using the default (dirvar=2), the bottom plot used dirvar=0.5.



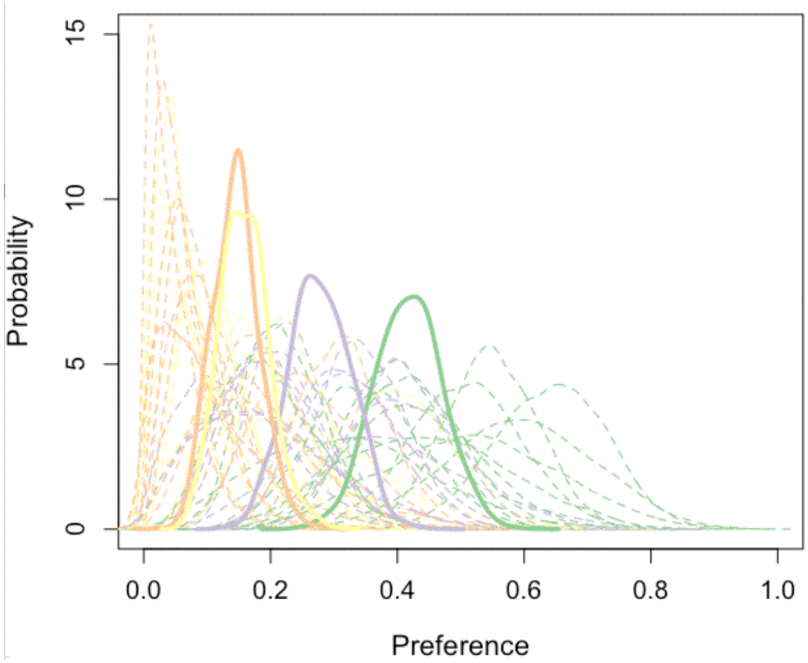
Once the user is satisfied with the mixing of the MCMC, the user can examine the parameter estimates provided by the analysis. The median parameter value and the credibility intervals for individuals and the population can by obtain with the function **credibleIntervals**.

```
credibleIntervals(prefres=YGGVpref[[1]],burn=1000,interval=
0.95)
```



The results can be plotted using the function **prefPlot**.

```
prefPlot(YGGVpref[[1]],burn=1000,ymax=15)
```



The solid lines show the estimates for the population-level preference for each plant and the dotted lines show the preference parameter estimates for each plant for each individual.

Pairwise comparisons:

Pairwise comparisons for population-level preference based upon the rankings of preference parameters for each step in the post burnin MCMC can be calculated using the function **pairwiseProb**.

```
pairwiseProb(YGGVpref[[1]],burn=1000)
```

```
> pairwiseProb(YGGVpref[[1]],burn=1000)
       [,1]   [,2]  [,3]   [,4]
[1,] 0.0000 0.9435 1.000 0.9995
[2,] 0.0565 0.0000 0.974 0.9615
[3,] 0.0000 0.0260 0.000 0.4210
[4,] 0.0005 0.0385 0.579 0.0000
```

These can be interpreted as one-tailed "post-hoc" tests. Thus, the probability that plant 1 is preferred over plant 2 is 0.0565. Likewise, the probability that plant 2 is preferred over plant 1 is 0.9435.

Comparing models:

Deviance Information Criterion (DIC) can be used to choose the best-fit model for the data. One can ask whether it is best to model the data where the parameter value describing preference is constrained to be the same. This is accomplished by using the argument `constrain=TRUE` in the function **bayesPref**.

Restricting our examination to only one population, here Yuba Gap, we can compare a model where the preferences are the same among all host plants, or whether they are best modeled as having different preference parameters.

```
YG<-YGGV[1:13,] # Create an object that only includes Yuba Gap, or population 1.
YGpref<-bayesPref(pData=YG,mcmcL=5000,pops=TRUE,dicburn=1000,dirvar=2)
YGprefconstrained<-bayesPref(pData=YG,mcmcL=5000,pops=TRUE,dicburn=1000,dirvar=2,constrain=TRUE)
```

YGpref DIC: 86.75607
YGprefconstrained DIC: 102.1706

The model used to generate `YGpref`, the unconstrained model, is more favorable compared to the model used to generate `YGprefconstrained`, the constrained model.

Similarly, whether or not populations should be modeled as having separate preferences, vs. a common, constrained preference can be examined using the argument `constrainP` in the **bayesPref** function.

```
YGGVpopconstrain<-bayesPref(pData=YGGV,mcmcL=5000,pops=TRUE,dicburn=1000,dirvar=2,constrainP=c(1,1))
```

Here, `constrainP=c(1,1)` sets the first and second population as being modeled as a single group.

Constrained DIC: 156.6923
Unconstrained DIC: 154.8371

Here, the constrained model and the unconstrained model are in the same family of "best models". In the end, the user must decide on the best model for the data. If the parameter estimates and CI are the goal (perhaps to be passed on to other analyses), the populations should be modeled separately.

Note: When multiple populations are involved, any groupings can be made. For example, if there were 4 populations, `constrainP=c(1,1,2,3)` would group populations 1 and 2, and allow populations 3 and 4 to be modeled as having there own preference.