BayesDiallel Package
11/7/2011
Package Code:
Alan Lenarcic
Package Maintainer:
William Valdar

1 BayesDiallel Package Vignette

Welcome to the "BayesDiallel" R-package: software designed to provide parameter confidence measures for the many possible interpreative models of phenotypes generated by a diallel experiment[1]. This package provides a mostly-automated method for converting diallel phenotype measurements to Bayesian Gibbs-sampled model fits of the data, generating posterior-confidence measures and prediction plots that help characterize genetic architecture and anticipate potential multi-locus outcomes of future crosses.

1.1 Code Methodology

BayesDiallel constructs a model-dependent covariance matrix $\mathbf{X}$ from parental-strain information, and fits an over-specified semi-parametric[2] mixed model via Bayesian Gibbs sampling. For small crosses (8 or fewer strains), full model data can be analyzed rather quickly and efficiently (order of a few seconds for 1000 gibbs samples), and larger scale crosses can be accomodated up to and beyond the cross of 50 or more strains. For small crosses, full-matrix-inversion methods permit very independent Gibbs samples (less than 0.1 10-lag auto-correlations). For larger crosses, we use partial-matrix inversion to realize samples efficiently. Underlying code is implemented in "C" and "LAPACK" for hardware efficiency. Data objects are stored and manipulated using R-Internals "SEXP" or "S-expressions" objects, and manipulation methods for users in R make use of "R.oo" package for pass-by-reference "S3" style output.

1.2 Bayesian semi-parametric model

Let the diallel be a cross of J distinct strains, numbered $1, 2, \ldots, j, \ldots, J$. Let the dataset be compsed of phenotype measurments for N specimens, numbered, $1, 2, \ldots, i, \ldots, N$. Every specimen i has a mother from strain $j(i)$ and a father from strain $k(i)$. This specimen also as a sex $s(i)$ which can be female or male. We consider that the inherited phenotype can be decomposed as a sum of effects which are a function of $j(i), k(i), S(i)$ plus additive noise. The effects of parental strain contributions might be moddeled as additive dosage (ie, $a_{j(i)} + a_{k(i)}$). But there could also be parent-of-origin effects, which we model as mother-against-father differential effects (ie, $m_{j(i)} - m_{k(i)}$). Basic dominance effects are encoded as a differential effect for inbreeding, that is, when $j(i) = k(i)$. These are encoded as a sum of mean inbreeding effect plus strain-specific inbreeding effect (ie $\mathbf{1}_{j(i)=k(i)} \times (\beta_{\text{Inbred}} + 2b_{j(i)})$. There also may be effects that are completely cross-specific for $j(i) \otimes k(i)$. We separate those effects into "symmetric" effects (where parent of origin does not matter; v_{jk}), and "asymmetric" effects (w_{jk}) which encode the difference between a $j(i) \otimes k(i)$ cross and a $k(i) \otimes j(i)$ cross, for overall specific

[1] The experiment of measuring univariate phenotypes in F_1 genetic crosses.

[2] We use the term "semi-parametric" rather than non-parametric, given that all measured genetic effects have specific meaning, though the parameter set is large enough to unconstrained determine a value for every cross in Diallel, and the number of parameters scales J^2 with the size of the Diallel.

effects $v_{jk} + w_{jk}$ and $v_{jk} - w_{jk}$). A un-sexed entire linear model can be written:

$$\begin{aligned} Y_i =& \mu + \underbrace{a_j + a_k}_{\text{additive}} + \underbrace{m_j - m_k}_{\text{maternal}} + \underbrace{\mathbf{1}_{j=k} \times (\beta_{\text{inbreed}} + 2b_j)}_{\text{inbreeding}} + \underbrace{v_{jk}}_{\text{symmetric}} + \underbrace{w_{jk}}_{\text{asymmetric}} \\ &+ \{\text{Other User-included Fixed and Random Effects}\} + \varepsilon_i \end{aligned} \tag{1}$$

We assume noise component ε_i is i.i.d. either Gaussian: $N(0, \sigma^2)$ or t-distributed $\sigma \times t_{\text{d.f.}}$ if flag `dfTNoise` is set. The hierarchical-based model assumes that component groups, such as the group of additive effects: $\{a_1, a_2, \ldots, a_J\}$ are apriori normally distributed with unknown variance. That is, we consider a prior shrinkage distribution of form $N(0, \tau_a^2)$ with τ_a^2 unknown. A Bayesian inverse-gamma-prior for τ_a^2 is supplied, leading to shrinkage estimates of $a_1, \ldots a_J$. For the full unsexed model there are shrinkage parameters $\tau_a^2, \tau_m^2, \tau_b^2, \tau_w^2, \tau_v^2$ plus additional shrinkage τ parameters for user-specified random components.

Including sex in the model allows for a larger set of sex-specific forms, that is:

$$\begin{aligned} Y_i =& \mu + \underbrace{a_j + a_k}_{\text{additive}} + \underbrace{m_j - m_k}_{\text{maternal}} + \underbrace{\mathbf{1}_{j=k} \times (\beta_{\text{inbreed}} + 2b_j)}_{\text{inbreeding}} + \underbrace{v_{jk}}_{\text{symmetric}} + \underbrace{w_{jk}}_{\text{asymmetric}} \\ &+ \frac{\Delta_{S_i}}{2} \times \{ \underbrace{\delta^S}_{\text{overall-sex}} + \underbrace{\delta_j^{(a)} + \delta_k^{(a)}}_{\text{additive}} + \underbrace{\delta_j^{(m)} - \delta_k^{(m)}}_{\text{maternal}} + \underbrace{\mathbf{1}_{j=k} \times (\delta_{\text{inbreed}}^{(\beta)} + 2\delta_j^{(b)})}_{\text{inbreeding}} + \underbrace{\delta_{jk}^{(v)}}_{\text{symmetric}} + \underbrace{\delta_{jk}^{(w)}}_{\text{asymmetric}} \} \\ &+ \{\text{Other User-included Fixed and Random Effects}\} + \varepsilon_i \end{aligned} \tag{2}$$

where Δ_{S_i} is a $+1$ multiplier if specimen i is female, and Δ_{S_i} is a -1 multiplier if specimen i is male.

Components of type additive or a_j are named `aj` in the BayesDiallel package. m_j are named `motherj`, inbreeding/dominance effects b_j are `Dominancej`, symmetric effects v_{jk} are `SymCrossjk` and asymmetric effects w_{jk} are `ASymCrossjkDkj`. All sex specific effects versions are preceded in name by `Gender:`. For instance $\delta_j^{(a)}$ is called `Gender:aj`. The τ_X^2 effects are stored in a vector called `tau` and individual coefficients are named with prefix `tau:`, for instance `tau:aj` for additive effect dispersion parameter τ_a^2.

1.3 Using BayesDiallel R.oo objects

For any given diallel dataset, one might consider multiple models of different sizes. We denote the "Full-sexed Model", short cut "`fulls`" or alternatively $SB_s a_s m_s b_s v_s w_s$ and inputted as "`Bs,as,ms,bs,vs,ws`" as the largest possible model. To organize and facilitate analysis of these many models we have called upon methods from R-package "`R.oo`" to store data and functions in `S3` type Reference classes. Versions of "`R`" greater than 2.13 implement the so-called native R5 Reference Classes, which reimplement "`R.oo`" data classes as part of the R5 base. Future developers should consider '`R5` as an alternative, and we hope to eventually port to those classes. In either case, interface and implementation between these two are very similar. End-users should realize that the objects output by these systems are very typical to familiar `S3` classes (such as "`lm-object`", but that they also include native methods and inheritance structures. Since the data in the object is encapsulated in a hidden "`R-environment`", it is possible to permanently change an object. In general, reassigning an object, such as a statement:

```
A <- B;
```

which does not copy the object, instead, "A" and "B" point to the same memory, and changes to one change the other. Thus, one should be prepared to use a 'Copy() function, such as:

```
A <- Copy(B);
```

which creates a new copy of the data in B and assigns it to "A". As it is in S3 classes, both data pieces and object-methods are accessed by using thre "$" operator. For instance if object "A" had a Data field called "x" and a method "AddOneToX", they could be accessed:

```
A$x                           ## print contents x
A$AddOneToX();  AddOneToX(A)$; ## Both run method AddOneToX on object A
```

Furthermore, we use R-base class "R-list" objects (see R-reference "list") to store adaptive lists of objects. Suppose that A had a list of called ListOfX, then

```
A$ListOfX[[1]]          ## First member of ListOfX
A$ListOfX[[2]];         ## Second member of ListOfX
A$ListOfX[[length(A$ListOfX)]];  ## Last member of ListOfX
```

Are the first, second, and last members of that list respectively. Function "'DiallelAnalyzer() creates an overall object of type FullDiallelAnalyze which serves as the base object containing all analysis data. Within that object, there exists a 'R list "AllDiallelObs" of model specific objects from Class type "DiallelOb". Objects of type "DiallelOb" contain model specific covariate sets, and when Gibbs sampling is completed, contain fields "raw.chains" and "centered.chains", which are R-package "coda" type "mcmc.list" objects that are the MCMC gibbs sample chains.

Consider using the following code to generate a fitted object of type FullDiallelAnalyze which we will save as AFD:

```
require(BayesDiallel, quietly = TRUE,
  warn.conflicts=FALSE);  ## Loads BayesDiallel

data(PiximusData);  ## Load Preloaded package
                    ##  specific data ''PiximimusData''
## Now fit data using DiallelAnalyzer
AFD = DiallelAnalyzer(data = Piximus.Data,
  father.strain="father.strain.name",
   mother.strain="mother.strain.name",
   phenotype="MouseWeight", is.female="is.female",
   sep="", FixedEffects = NULL, RandomEffects = NULL,
   Models=Example.Piximus.Models, sigmasq.start = 1,
   numChains = 5,  lengthChains=2500,
   burnin = 1, DIC.Only=FALSE,
   tauPriorFile = Example.Piximus.tau.Prior.Info,
   SaveAFDFile = "SaveAFDBackUpPiximusData.RData");
```

AFD now contains a list called AllDiallelObs, where each list member is an object of type DiallelOb with fitted Gibbs chains within. For instance

```
names(AFD)                    ## names of named
                              ##  objects in AFD

names(AFD$AllDiallelObs[[1]])  ## names of named objects
                              ## in first AllDiallelObs list element
##Summary of the chains for first model in the list
summary(AFD$AllDiallelObs[[1]]$centered.chains)

##MCMC plot of raw chains for last model in the list
plot(AFD$AllDiallelObs[[length(AFD$AllDiallelObs)]]$raw.chains)
```

```
## HPD (High Posterior Density) plots for the first model
plot.hpd(AFD$AllDiallelObs[[1]]$centered.chains)
```

Thus one can see how to access the data types in the two objects. The length of `AFD$AllDiallelObs` will be the number of models that one supplies in the `Models` argument to `DiallelAnalyzer`.

1.4 DiallelAnalyzer() function

The chief wrapper-function for analyzing diallel data is `DiallelAnalyzer`() which allows as input most of the material a basic user would wish to use in analysis of a diallel. Basic inputs to `DiallelAnalyzer`() include the data matrix or file name: `DataFile`, a string name or vector of mother strains: `mother.strain`, a string name or vector of father strains: `father.strain`, a string name or vector of female sex indicators: `is.female`, and a phenotype real valued vector or string name: `phenotype`. One can optionally provide `FixedEffects` and `Random Effects` vectors or string names to add these effects to the linear model. If sex is unknown for some of the specimens, prior probabilities can be given in the form of `ProbFemaleUnknownSex`, and those with unknown gender are either identified as `NA` values within `is.female` or a supplied list `ListUnknownSex`. If one desires the analysis to save chains to a compressed `RData`() file, set the string for `SaveAFDFile`() to a user accessible file.

The models to be examined should be added as value `Models`. This can be a string, vector of strings, file pointing to vector of strings, or a pre-configured matrix. This allows one to set all interesting model parameters for the diallel to analyze, and also whether to use t-distributed noise for some of the models. See `ReadModels`() method and `ModelsFile` as it is defined in the help section.

The Gibbs sampler chains can be changed in number and length by changing `numChains` and `lengthChains` values repsectively. Two additional options for efficient Gibbs sampling memory management exist. Setting `Wanted` sets the Gibbs samplers to only save a subset of the Gibbs sampled parameters. Give the names of all parameters desired. For optional post-processing of data, input to setting `Extractor` is a user-defined this function. This function should be able to collect the information contained in $\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\tau}}^2, \hat{\sigma}^2$ and process this output into a sub/or-superset of estimated variables. The algorithm thenaves only this output into the analysis object and file.

1.5 Saving Analysis to a File

The option `SaveAFDFile` to `DiallelAnalyzer` decides whether to save the analysis chains to a file on the harddrive. This is preferable to trying to save the `FullDiallelAnalyze` objects oneself. Make sure that the file will be located in a directory the user has access to. One can then reload as:

```
library(BayesDiallel);  library(R.oo);
load("SaveAFDBackUpPiximusData.RData");
```

where loading BayesDiallel package is preferable before loading the object. The object will be saved as `AFD` which should be attached to the global environment. Search for this object in the R-memory by typing `ls()`.

1.6 Functions for "FullDiallelAnalyze" Object

Some useful functions introduced and descrbied.

- `RunChains()` - reruns, or runs for the first time, the Gibbs sampler chains analyzing all of the diallel models.
- `SetupAnalysisMethods()` Sets up the chains for the Full DiallelAnalyze method. If models are being supplied, supply to input `ModelsList()` in form of matrix if using this function. Base users should setup the models using wrapper function `DiallelAnalyzer()`. See function `ReadModels()` for description of how to convert model strings to model flags.
- `SetupUnknownSex()` This function sets up for the case where some of the sex identifiers in "`is.female`" are set to `NA`. Function will attempt to use Bayesian imputation to learn the unknown Sex. One can assign prior probabilities different from .5 for male female for each of the unknown genders by supplying "`ProbFemaleUnknownSex`" vector when running this function.

1.7 Functions for "DiallelOb" Model Object

- `ReplaceAndRerunUpdatedChains()` When a given model has not converged, this function expands the length of the Gibbs sampler chain to try to achieve better mixing.
- `SetupChains()` Sets up Gibbs Sampler Chains for DiallelObject.
- `RecalculateLikelihood()` . This function calculates the log-likelihood value at each iteration of the chains.
- `RunChainsCC()` This is the command to run the Gibbs Sampler.
- `CalcDICC()`. Calculate DIC value of he model using the Gibbs sampler chains.
- `PosteriorRSq()` This derives R-squared scores for fitted chains describing amount of variance explained.
- `AutoMFFunction()` This creates the centered version of the Gibbs Sampler chains, saved into `cent.chains` S.
- `PosteriorPredSummary()` This is a large function to try to derive posterior prediction summary chians and statistics for all positions (observed and unobserved) in the diallel, including their mean and posterior prediciton confidence intervals. Standard inputs to this function are `burnin` an integer saying how much of the front end of the chain to ignore as burnin, `thin` an integer showing how many intervals to prune or thin the chain, a `keep` logical flag that when `TRUE` keeps the MCMC chains and not just summary statistics, a `plotdensity` logical flag that when `TRUE` prints a density plot for prediction in the R terminal, and a `WithNoise` logical flag that when `TRUE` studies posterior predictive density for future datapoints including noise above the mean.
- `PlotStrawPlot()` Plots a "straw plot" based upon the current model. This is a color coded plot of effects sizes based upon either posterior mean or median.
- `TwoDiallelPlot()` Plots a checkerboard "Diallel Predictive Plot" based upon the current fitted model, or on observed data.

- `PosteriorHSq()`. A Measure of "Heritability" as defined in the paper. Using estimates of $\tau_a^2, \tau_b^2, \ldots$ and the structure of the diallel, this $\mathbf{h}^2$ estimate posits a random distribution for future Diallels of independently selected strains, all of size $J \times J$. Heritability of all values (plus residual σ^2) effects, sum to 1. Note that the default option calculates specific weights valuing the importance of the various effects. For example, since inbreeding components only affect inbreds, who only make up J of the $J \times J$ total crosses in a Diallel, they are weighted $1/J$ relative to additive component τ_a^2. Elements `PosteriorHSqTable` and `PosteriorHSqChains` exist as aliases to default inputs to this function. (`PosteriorHSqTableWithFixed` and `PosteriorHSqTableImproper` give the fixed-effects-included and unweighted model estimates respectively).

- `PosteriorRSq()`. A Measure of "R-squared", or amount of observed variance explained in the observed diallel dataset, as defined in the paper. Using estimates of $\hat{\boldsymbol{\beta}}_a, \hat{\boldsymbol{\beta}}_b, \ldots$ and the covariance matrix of the observed Diallel, this $\mathbf{r}^2$ estimate calculates the sum of squared variability supplied by the fitted factors. Variance explained of all values (plus residual σ^2 effect), can sum up to a number different from 1. This is because some factors interact, i.e. $\hat{\boldsymbol{\beta}}_a \mathbf{X}_{,a}^T \mathbf{X}_{,b} \hat{\boldsymbol{\beta}}_b \neq 0$, in that there is non-orthogonal covariance explained between effects. Elements `PosteriorRSqTable` and `PosteriorRSqChains` exist as aliases to default inputs to this function. (`PosteriorRSqTableWithFixed` gives the fixed-effects-included model estimates for $\hat{\mathbf{r}}^2$).

- `PosteriorDSq()`. One last measure of contribution to variability, this time, based upon $\hat{\boldsymbol{\beta}}$ contribution to a theoretical full Diallel. Alias tables are generated by selecting fields `PosteriorRSqTable` and `PosteriorRSqTableWithFixed` for random-effects-only, and fixed-effects-included values of $\hat{\mathbf{d}}^2$.

1.8 Diallel Models

See `help(ModelsFile)` for more info. Naming the many possible desired models in a file for input `ModelsFile` given to `DiallelAnalyzer()` one desires to send a string combination that can be interpreted to give the identity of all parts of the active model. "`Fulls`" automatically inputs the full model containing $S, \beta^S_{\text{inbred}}, a_s, b_s, m_s, v_s, w_s$, that is, all sexed and unsexed terms. "`Fullu`" gives the full unsexed model containing $\beta_{\text{inbred}}, a, b, m, v, w$. Alternatively one could supply "`Bs, as, bs, ms, vs, ws`" or "`B, a, b, m, v, w`" for full sexed and full unsexed models. Supplying strings missings ome of these terms supplies a model missing those effects. For instance "`B, a, b`" does not contain maternal, symmetric, or asymmetric or any sex effects. If sex identities, in the form of `is.female` were not supplied to `DiallelAnalyzer()`, then the default model is sexless `Fullu`. However, if sex information was included, but no model is specified, `DiallelAnalyzer()` will use model `Fulls` with all components and sex differentiated components included.

One additional effect included in the `ModelsFile` is a "`df`" term to suggest degrees of freedom for t-distributed noise. Putting a positive numeric value after `df` results in the Gibbs-sampler modelling t-distributed noise of that process. For instance, `df6` and `df12` are often reasonable t-distributions to use, since these error distributions contain means and variances.

1.9 Help pages

For additional support, consider the package documentation, accessed by typing `help(BayesDiallel)`, at the console. Further documentation exists at `help(FullDiallelAnalyze)` and `help(DiallelOb)` on other features of these object classes, as well as detailed documentation of inputs are located at `help(DiallelAnalyzer)`. These pages also include Example demonstration code, much of it making use of package included diallel example data, described at `help(PiximusData)`, `help(AdviaData)`, and `help(ChemData)`. The following walkthrough chiefly describes the operations and steps being used in these Example analyes.

2 A Tutorial Example of Diallel Analysis

We consider the default data `Piximus.Data`, accessible by loading in `data(PiximusData)` at the terminal, as one observes by typing `ls()` at the terminal, this loads three well-formatted datatables into the global environment: an example data file `Piximus.Data`, an example models file `Example.Piximus.Models`, and an example priors file `Example.Piximus.tau.Prior.Info`. Let us examine the format of these tables.

2.1 Piximus.Data

Some of the Piximus experiment data is displayed below.

```
> Piximus.Data[c(1,125,594),]
```

```
    mother.strain.name father.strain.name is.female     mouse.name PctFat
1                  129                129      TRUE  CCF1F01_92297  13.91
125                 AJ                NOD     FALSE  ADF1M04_92007  24.30
594                WSB                WSB     FALSE HHF1M11_442553  19.28
    BoneMineralContent BoneMineralDensity LeanTissueMass MouseLength.NoseToAnus
1                 0.31               0.07            8.0                  21.89
125               0.29               0.06           13.9                  40.18
594               0.15               0.05            6.0                  16.94
    MouseWeight TotalTissueMass
1         21.89            9.71
125       40.18           19.22
594       16.94            8.07
```

As we see, the `mother.strain.name` and `father.strain.name` columns are each character string columns delineating with unique identifiers the parental strains (non-specific to parental id. numbers) of the crosses. There are 594 individual specimens, here with multiple phenotypes measured. In this example there are no batching groups. We will consider chiefly the phenotype `Mouse.Weight`, Sex information is contained in the `is.female` vector, here a binary logical vector, though a `0-1` integer variable is possible. `MALE` and `FEMALE` options are possible. Specimens with unknown sex should be indicated with `NA` or `Unknown`. The phenotype `MouseWeight` is right skewed and measured in grams, continuous postive valued variable. A log-transformation of phenotype, aka analysis of `log(Piximus.Data$"MouseWeight")` could be preferable. One might consider setting flag `LogTransform` as input to `DiallelAnalyzer()` to have the data log transformed before analysis. `SqrtTransform` is the only other optional default transform pre-implemented. For other transformations, please transform the column data before calling `DiallelAnalyzer()`.

Note it seems to be common for experimentalists to try to save the column information into one semi-standard text string. Variations of a text string "`female PWK_1 X CAST_2`" as a text entry are typical formats. However, given the great variation that could go into such a text string, we decline to offer an automatic strain detection program. It becomes the end-users job to make sure that these three pieces of information are stored intelligibly in three columns. We recommend judicious use of R's internal string matching functions `grep()` and `strsplit()` in case one needs maintenance code to convert a set of string identifiers into `BayesDiallel` acceptable 3 column format.

Note that `data(AdviaData)` and `data(ChemData)` are other, alternative sample datasets included that one could consider for further exploration.

2.2 The Example.Piximus.Models file

```
> Example.Piximus.Models
```

```
[1] "fulls"
[2] "fullu"
[3] "fulls, df_6"
[4] "BSabm"
[5] "BSasbsms"
[6] "B,v,w,w_s,b,m,ms,mu"
[7] "Mu, strain, gender, gender-mother, gender-inbred, symmetric-cross, inbred"
[8] "Mu, inbred, inbred-strain, strain, gender-inbred"
```

We see that the "ModelsFile" is a list of strings. In this example, the first two encode for the full sexed and unsexed models, `fulls` and `fullu`. The third model is a full-sexed model, but also has a t-noise distribution with degrees of freedom 6. There are many options for typing models, using and excluding commas, though it is recommended to choose a notation with no repeats of variables. When term `w_s` is included, by the features `v, v_s, w` must be automatically logically included.

2.3 "Example.Piximus.tau.Prior.Info" Prior information

The Prior information table is an array with up to three rows which should be named.

```
> Example.Piximus.tau.Prior.Info
```

```
      aj dominancej genderj symcrossjk asymcrossjkDkj motherj
m    0.2        0.2     0.2        0.2            0.2     0.2
df   0.2        0.2     0.2        0.2            0.2     0.2
Prob 0.5        0.5     0.5        0.5            0.5     0.5
```

The first two rows suggest inverse gamma priors for the provided parameter groups. The form of the inverse Gamma prior is $m/\text{Gamma}(\text{df})$, and hence the form of the density is:

$$\text{Prior Density}(\tau_X^2) = \frac{1}{m\Gamma(\text{df})}\left(\frac{m}{\tau_X^2}\right)^{\text{df}+1} e^{-m/\tau_X^2}$$

As we see, a default prior strength of .2, is like suppling an additional .2 strains of prior density. If $m = .2$, then the prior mean is 1. It is possible to supply $m = 0$, $\text{df} = -1$ for flat priors ($\propto 1$),

and $m = 0, \mathrm{df} = 0$ for a $\propto \frac{1}{\tau^2_X}$ Jeffrey's prior. If a column named "all" exists, it will try to apply the prior to all models. It is always assumed that the first row of the `tau.Prior.Info` file refers to the m parameter, and the second refers to the df parameter.

If an optional third row is supplied, it is providing prior probabilities for use with `BayesSpike` selection. `BayesSpike` is a Gibbs-sampler based grouped-parameter selection technique, that attempts to generate a posterior probability of model inclusion for each of the groups of effects. `Prob` can be user toggled to be any prior probability between 0 and 1 to promote or demote posterior probability of given parameter groups. Alternatively, `BayesSpike` can implement other priors. To use these priors and generate selection chains, the option `DoBayesSpike` should be set to `TRUE` in the `DiallelAnalyzer()` function call. Also, package `BayesSpike` must be installed. `BayesSpike` analyses require a file save location to store Gibbs samples on the harddrive. Set `BSSaveDir` to a user-accessible harddrive location when calling `DiallelAnalyzer()` with the `DoBayesSpike = TRUE` flag set.

2.4 Performing a Basic DiallelAnalyze() on Piximus Weight Data

It is simple enough to run:

```
###### Run the Algorithm->
AFD = DiallelAnalyzer(data = Piximus.Data,
  father.strain="father.strain.name",
   mother.strain="mother.strain.name",
   phenotype="MouseWeight", is.female="is.female",
   sep="", FixedEffects = NULL,
   RandomEffects = NULL,
   Models=Example.Piximus.Models[c(1,3,5)], sigmasq.start = 1,
   numChains = 2,  lengthChains=1000, burnin = 1,
   DIC.Only=FALSE,  tauPriorFile = Example.Piximus.tau.Prior.Info,
   SaveAFDFile = "SaveAFDBackUpPiximusDataShort.RData");
```

As one will see, as long as the `Verbose` option is set to `TRUE` or a number greater than zero, there will be several lines of feedback text as the model runs. The method will announce which chain and model is running, and display the current Gibbs sampler position every `BayesSpike` steps.

Once the Gibbs sampling is finished, we can look at the chain output. The first member of `AFD`'s list `AllDiallelObs` is a `DiallelOb` corresponding to the first "full-sexed" model. We can access the member in two ways:

```
## AFD is the FullDiallelAnalyze object (contains all models)
##  AO is the DiallelObs for the first model;
AO = AFD$AllDiallelObs[[1]];  ##  AO is the first member of
                              ## all AllDiallelObs of AFD;

## Valid Alternative method to get same AO
AFD$OnObs = 1;  AO = AFD$On;
##  When AFD$On is set to the right iteration of desired AO,
##    many DiallelOb options function directly on AFD;
```

This is because we can set the integer element `AFD$OnObs` to be `1`, which points to the first member of the `AFD$AllDiallelObs` list. On this setting `AFD` itself has alias elements that point to the chains, such as `AFD$centered.chains` and `AFD$raw.chains`, which are actually shortcuts for things like "`AFD$AllDiallelObs[[AFD$OnObs]]$getRaw.chains()`" which is the actual function call for

the alias element. Note that one cannot manually change the `raw.chains` or `centered.chains` because these are actually aliasing functions that point to a functional element of `DiallelOb`.

Consider then some basic plot demonstrations:

```
## cent.chains relate the relative location of effects in groups.
##   For confirmation of terms, one should
##   often use the cent.chains these by default;
##
summary(AO$cent.chains);   ## summary of the chains;
plot(AO$cent.chains);      ## Use Coda MCMC to plot the many member
                           ##  of chains consider "ask"/"wanted"
plot.hpd(AO$cent.chains);  ## Plot Hpd intervals for members of
                           ##  "chains" object.  Consider "ask"/"wanted"
plot.hpd(AO$cent.chains, wanted = 1:20);  ## Plot HPD intervals
                                          ## for first 20 members

##raw.chains are geneerated by the algorithm itself and represent
##  unconstrained bayes distributions.  These coefficients are useful
##  in predictive mean summaries.
plot(AO$raw.chains);
```

The `summary()` method, applied to some MCMC chains of the type defined in R-package `coda` performs many measurements, including getting mean, sd, and quantiles of the posterior draws for all elements of the chains. MCMC chains also have their own default `plot()` function which demonstrates mixing effects by colors. `plot.hpd()` plots high posterior density 95% credibility intervals. These are the shortest intervals with 95% posterior coverage. Note, that the `cent.chains` are actually estimates of groups minus group means. That is, if $\hat{a}_j^{(t)}$ is the draw for j-th group additive effect in Gibbs iteration t, then the centered chain reports of $\hat{a}_j^{(t)} - \sum_{j'} \hat{a}_{j'}^{(t)}/J$, which has a group mean of zero for each draw. It turns out that these restricted measures have tight confidence, and are much more useful as credibility measures than their uncentered versions. These centered estimates are good for fitting parameter groups that one believes to be mean zero, or equivalently, are good for reporting the strain effect relative to each other, rather than in terms of their absolute value Since most statistical estimation needs actually do not rely on the coefficient levels themselves, but rather in proving significant size of contrasts like $a_j - a_{j'}$, the centered chains are the prefered output for most reporting analyses. The `raw.chains` are likewise available, and are best used for more advanced estimations. Note that the large credibility intervals observed in the raw.chains comes from confounding with other parameters, such as μ. In fact, there is a Bayesian non-identifiability problem at the issue, that perturbing μ is nearly identical to perturbing all a_j by a constant.

Two other simple analyses are to create a Posterior Prediciton summary with method `PosteriorPredSummary()`. This method gives posterior mean or posterior prediction estimates for the mean at all j, k, S cross positions on the Diallel, strain combinations observed and unobserved, male and female sexes. It is used in the plotting function `draw.diallel()`. This plots the squares of the diallel as an informative gray-scale plot. This function is quite useful for visualizing the data and understanding the ffects of the model. One need not use this function however, because two alias variables `PosteriorPredictionsMeanSummary` and `PosteriorPredictionsErrorSummary` are featured in every `DiallelOb`. These serve the purpose of constructing both the prediciton mean, and the prediction total distributions, including the coda chains.

```
#MyPostSumGivesValues for all the diallel squares of every gender
```

```
ADO = AFD$AllDiallelObs[[AFD$OnObs]];
MyPostSum = PosteriorPredSummary(ADO, AFD, burnin = 400, AFD=AFD, keep = TRUE);
MyPostMeanSum = ADO$PosteriorPredictionMeanSummary;
MyPostPredictionSum = ADO$PosteriorPredictionErrorSummary;
```

2.5 Plotting Predicted Diallel Plots

Plotting the "checkboard" diallel plot can be an important boost to interpretability of both the fitted and observed data. Function `draw.diallel()` can take many formats of diallel data and format them into a good image. We recommend also the use of built in `DiallelOb` function `TwoDiallelPlot()`. This function runs the predicted fit of the model and then either plots a comparison observed versus predicted, or a comparison of male versus female specimens. Some options of use are here.

```
MyPostSum = PosteriorPredSummary(ADO, AFD, burnin = 400, AFD=AFD, keep = TRUE);
SimmedGrid = GiveGrid(AFDS);
par(mfrow=c(1,2));
draw.diallel(data =SimmedGrid[MyPostSum$"Female/Male" == 1,],
  phenotype = "Mean.Simulated")
draw.diallel(data = MyPostSum[MyPostSum$"Female/Male" == 1,],
  phenotype = "Mean Posterior")

## Plot Females, observed versus fitted
AFD$AllDiallelObs[[1]]$TwoDiallelPlot(FemalePlot = TRUE,
  MaleAgainstFemale = FALSE);

## Plot Males, observed versus fitted
AFD$AllDiallelObs[[1]]$TwoDiallelPlot(FemalePlot = FALSE,
   MaleAgainstFemale = FALSE);

## Plot Male Predicted versus Female Predicted.
AFD$AllDiallelObs[[1]]$TwoDiallelPlot(FemalePlot = TRUE,
  MaleAgainstFemale = TRUE);

## Plot Male Observed vresus Female Observed.
AFD$AllDiallelObs[[1]]$TwoDiallelPlot(PlotOnlyObserved = TRUE);
```

An example output is displayed in Figure 1.

The `TwoDiallelPlot()` method uses the function `draw.diallel()` to make these plots. A help file on `draw.diallel()` demonstrates how to use that function for extended needs.

2.6 Calculating Heritability Estimates

One may desire summary values to stress importance of factors in the dataset. $\hat{\mathbf{h}}^2$ attempts to describe the "heritability" of genetic traits. We describe "heritability" to mean the effect of this inheritance component, as expressed in future un-forseen diallels using new strains sampled independently from a similar population of the observed diallel. This $\hat{\mathbf{h}}^2$ only depends on fitted $\boldsymbol{\tau}^2$ values. Which give a future prediction for effects (i.e. $\boldsymbol{\beta}_a^{\text{new}} \sim N(0, \hat{\boldsymbol{\tau}}_a^2)$). One can imagine sampling new crosses uniformly accross a $J \times J$ grid of possibile crosses, and making a statement about heritability in this Diallel. However, note, the inheritance structure and value for various components changes for different J', or if the crosses are not uniformly distributed amongst the

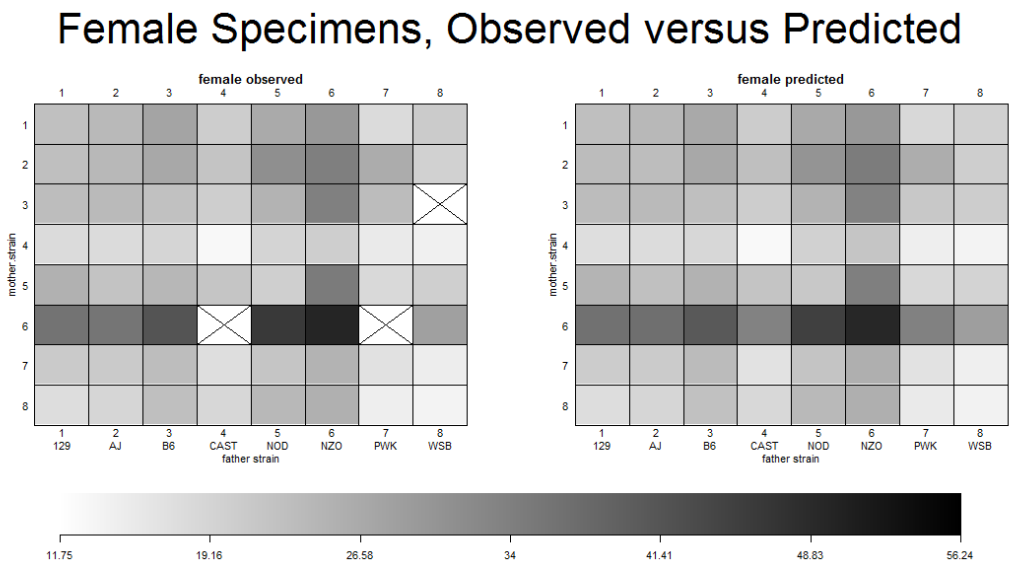

Figure 1: An example of a TwoDiallelPlot. In this case, observed versus predicted female mouse weight.

$J \times J$ grid. For instance, if a future diallel did not include inbreds at all, then inbreeding would necessarily be 0% of the heritability of this Diallel. We stress that we made several arbitrary but to us natrual choices in generating our definition. Our definition of $\hat{\mathbf{h}}^2$ is calculated in a form:

$$\text{Additive heritability: } \hat{\mathbf{h}}^2_a = \frac{c_a \hat{\tau}^2_a}{c_a \hat{\tau}^2_a + c_m \hat{\tau}^2_m + \ldots c_{\delta_w} \hat{\tau}^2_{\delta_w} + \hat{\sigma}^2} \tag{3}$$

Where coefficients $c_a, c_m, \ldots c_{\delta_w}$ are explained in the appendix-A of the diallel paper. The coefficients are shown here in Table 1.

a	b	m	v	w	a_s	b_s	m_s	v_s	w_s
$\frac{2(J+1)}{J}$	$\frac{1}{J}$	$\frac{2(J-1)}{J}$	$\frac{J-1}{J}$	$\frac{J-1}{J}$	$\frac{J+1}{2J}$	$\frac{1}{4J}$	$\frac{J-1}{2J}$	$\frac{J-1}{4J}$	$\frac{J-1}{4J}$

Table 1: Table of coefficients entering heritability, keyed towards strength of contribution of each component

Posterior calculations of $\hat{\mathbf{h}}^2$ are operated upon a post-fitted `DiallelOb` and through member function `PosteriorHSq()`. `DiallelOb` contains elements `PosteriorHSqTable` and `PosteriorHSqChains` exist as aliases to default inputs to this function. (`PosteriorHSqTableWithFixed` and `PosteriorHSqTableImproper` give the fixed-effects-included and unweighted model estimates respectively). Logical Option `UseFixed=TRUE` toggles whether to include the fixed components, and `ProperRatio = TRUE` toggles whether to use the $c_a, \ldots$ coefficients or set them all to 1. This method creates a "point" estimate based upon mean "$\hat{\boldsymbol{\tau}}^2$" into `PosteriorHSqPoint`, though it also creates the Gibbs sampled element `PosteriorHSqChains` which is of type `mcmc.list`.

Furthermore we define $\hat{\mathbf{r}}^2$ to be a different expression of the variance explained within the dataset. This is a "within observed data" calculation, and is measured as an independent contribution to the Null model sum squared $\sum_i (Y_i - \bar{Y})^2$. Note that since the model $\mathbb{E}[Y_i] =$

$\mathbf{X}_{i,}\hat{\boldsymbol{\beta}}$, then it is not true that elements of $\hat{\boldsymbol{\beta}}$ are orthogonal to each other. Thus, even if we assign an estimated variance component $\hat{\mathbf{R}}^2_a \propto \hat{\boldsymbol{\beta}}_a \mathbf{X}^T_{,a} \mathbf{X}^T_{,a} \hat{\boldsymbol{\beta}}_a$ which is a demonstration of squared variability contributed by additive effects, it is not often true that $\hat{\boldsymbol{\beta}}_a \mathbf{X}^T_{,a} \mathbf{X}^T_{,b} \hat{\boldsymbol{\beta}}_b = 0$, and hence the interaction of effects also supply a contributribution to the total $\sum_i (Y_i - \bar{Y})^2$, even though we ignore them. Thus the sum of all components of $\hat{\mathbf{r}}^2$ do not equal 1 because we use the true denominator $\sum_i (Y_i - \bar{Y})^2$.

For an alternative "middle-ground", a third vector summary measure $\hat{\mathbf{d}}^2$ is useful for comparing different phenotypes on the same crosses of strains. Unlike $\hat{\mathbf{r}}^2$ which measures within sample variation, $\hat{\mathbf{d}}^2$ simulates data from a full ballanced diallel based upon fitted $\hat{\boldsymbol{\beta}}$ estimates, and attempts to decompose variation in the simulated diallel against contributions of the factor groups of $\hat{\boldsymbol{\beta}}$. List $\hat{\boldsymbol{r}}^2$, $\hat{\mathbf{d}}^2$ has the problem that contributions interact with each other. However, $\hat{\boldsymbol{d}}^2$ might be easier to use to compare datasets where the idenities of the cross-strains are preserved, though the ballances in the datasets are not equal. This method tries to put all of the datasets on the same footing as one complete diallel. $\hat{\mathbf{h}}^2$ is applicable when the identities of the strains are not preserved.

Much like `PosteriorHSq()`, the `PosteriorRSq` and `PosteriorDSq` methods acts on a fitted `DiallelOb` and as member methods. Elements `PosteriorRSqTable` and `PosteriorRSqChains` exist as aliases to default inputs to this function. (`PosteriorRSqTableWithFixed` gives the fixed-effects-included and unweighted model estimates respectively). Logical Option `UseFixed=TRUE` toggles whether to include the fixed components. Using mean estimates for $\hat{\beta}$ there is a natural a "point" estimate for $\hat{\mathbf{r}}^2$ based upon mean "$\hat{\boldsymbol{\tau}}^2$" into `PosteriorRSqPoint`. Though there also creates the samplers $\hat{\mathbf{r}}^{(t)}$ Gibbs chain list `PosteriorRSqChains` which is of type `mcmc.list`. For $\hat{\mathbf{d}}^2$, the alias values are: `PosteriorDSqTable`, `PosteriorDSqChains`, `PosteriorDSqTableWithFixed`, `PosteriorDSqChainsWithFixed`.

```
##########################################################################
##  HSq, RSq, and DSq
##
##   HSq is our future heritability metric, based upon estimated
##    tau^2 parameters, it assumes independence of  future diallels.
##
##   RSq relates the amount of variation in Y explained by a covariate.
##    Note, that because of correlation between terms, sum of RSq terms
##    can be more or less than total variation of Y.
##
##   DSq is an inbetween measure.  It simulates an artificial complete
##    Diallel, according to fitted factors and inspects what percentage
##    of variation is attributable to the estimated factors.  Like RSq,
##    correlation between factors means that sum of DSq does not total
##    to the variation of Y.  However, like HSq, it is a statement of
##    future data.  Like RSq, DSq uses fitted Beta coefficients of each
##    effect.  Thus DSq is applicable to future diallel crosses where
##    the strain identities are preserved.
##
AFD$AllDiallelObs[[1]]$PosteriorHSqTable

## Inner method that calculates that HSq
OtherPosteriorChains = AFD$AllDiallelObs[[1]]$PosteriorHSq(UseFixed=TRUE,
  ProperRatio=TRUE, AFD = AFD, startiter = AFD$burin, thin = AFD$thin)

## Now look at RSq as a different metric
summary(AFD$AllDiallelObs[[1]]$PosteriorRSqChains);  ## RSq
AFD$AllDiallelObs[[1]]$PosteriorDSqTable             ## HSq
```

2.7 Plotting a Straw Plot

One visual summary of the mean effects of a mixed model is a so-called "Strawplot", which we have designed one plot suitable for the effects structure in the Diallel. The function of interest is the posterior analysis function for a `DiallelOb` object called `PlotStrawPlot()`. This is best run on the full sexed model. Setting a flag `DoMedian` to be `TRUE` uses posterior median as opposed to posterior mean values to make the plot, one example of which in Figure 2. Note that this plot is somewhat simplistic in that it does not show interaction (w, v) effects.

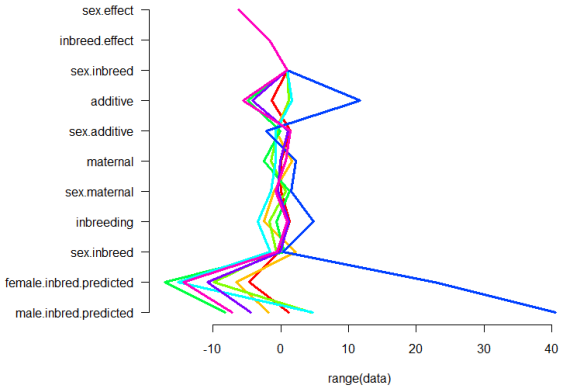

Figure 2: An example Straw Plot of Diallel Information based upon Diallel weight data.

```
AFD$AllDiallelObs[[1]]$PlotStrawPlot();
```

Depending on the model implemented in the `DiallelOb`, certain effects will be non-zero.

2.8 Performing BayesSpike Selection for the Diallel

BayesSpike selection, or group parameter selection can be attempted if supplementary package `BayesSpike` is installed and included. One need set additional flag `DoBayesSpike` to `TRUE` for `DiallelAnalyzer()`, as well as set a `BSSaveDir` to a users savable directory. This way `DiallleAnalyzer()` will fit the `BayesSpike` method by first finding the largest model in `AFD` and then selecting subsets of parameters of this model that best reflect the observed posterior.

```
##############################################################################
##  BayesSpike Selection uses Prior distributions to select a valid
##   model from subset of other models.  "tauPriorFile" can have an
##   extra row "Prob" with probabilities, if one wants to select
##   certain groups with different prior probabilities.  Default
##   "Prob" will be .5
##
##  See library(BayesSpike) for more details on this selection process.
##
## Experimental, might not work if you don't have directory write access
library(BayesDiallel); library(BayesSpike);
data(PiximusData);
if (length(list.files("C:")) > 0) {
  try(dir.create("C:/Stat"));
  try(dir.create("C:/Stat/BayesSpikeSaves"));
```

```
  BSSaveDir = "C:/Stat/BayesSpikeSaves"
} else if (length(list.files("~/Documents")) > 0) {
  try(dir.create("~/Documents/Stat"));
  try(dir.create("~/Documents/Stat/BayesSpikeSaves"));
  BSSaveDir = "~/Documents/Stat/BayesSpikeSaves"
} else if (length(list.files("~/")) > 0) {
 try(dir.create("~/Stat"));
 try(dir.create("~/Stat/BayesSpikeSaves"));
  BSSaveDir = "~/Stat/BayesSpikeSaves"
}
AFD2 = DiallelAnalyzer(data = Piximus.Data,
  father.strain="father.strain.name",
   mother.strain="mother.strain.name", phenotype="MouseWeight",
   is.female="is.female", sep="", FixedEffects = NULL,
   RandomEffects = NULL,
   Models=Example.Piximus.Models, sigmasq.start = 1,  numChains = 5,
   lengthChains=2500, burnin = 1,
   DIC.Only=FALSE,  tauPriorFile = Example.Piximus.tau.Prior.Info,
   DoBayesSpike = TRUE, BSSaveDir = BSSaveDir,SaveAFDFile ="");

## Get a summary of the Coda Chains for first diallel Model
library(coda); plot(as.mcmc(AFD2$BSAFD$CodaList));
summary(AFD2$BSAFD$CodaList)
plot(AFD2$BSAFD$CodaList);
plot.hpd(AFD2$BSAFD$CodaList);
```

Note that the selection object is a `Rcpp` module, which is an imbedded `C++` class. This selection object is stored in `AFD$BSAFD`. To learn about all of the methods and members attached to `AFD$BSAFD` type `BayesSpikeCL` at the command prompt. `BayesSpike` is still in development, so full documentation on all of its features will be discussed in another vignette. The member `AFD$BSAFD$CodaList` is the coda list of output from the `BayesSpike` object. It is a set of raw, uncentered chains. It contains not just chain estimates for the $\boldsymbol{\beta}, \boldsymbol{\tau}^2, \sigma^2$ parameters of the regression, but also chain specific Rao-Blackwellized $\mathcal{P}(\beta_j|Y, \boldsymbol{\beta}_{/j}, \boldsymbol{\tau}^2, \sigma^2)$ conditional posterior chain draws, which can be used to define overall confidence on $\mathcal{P}(\beta_j|Y)$ when averaged.

It is recommended that for all priors in a `tauPriorInfo` table be proper priors, that is, have $m > 0, \text{d.f.} > 0$, as an improper prior leads often to confusion in the sampler during comparison between a nonzero τ^2 parameter and a zero value. Having nonzero m allows one to the algorithm to understand the scale between which a $\tau_j^2 = 0$ and $\tau_j^2 \neq 0$ decision is made. Indeed, it is an arbitrary data-reduction technique to threshold away a $\tau_j^2 = 10^{-20}$ down to zero, but the computational and conceptual benefits to selection justify the use of these selection techniques.

Anticipate that many parameters τ_j^2 might not deviate far from their prior probability of activation (as optionally supplied via a third line in `tauPriorInfo`). This is because there is likely little or no information on that parameter in the current dataset. Although `DiallelAnalyze()` can give a valid posterior for datasets of any size, when data small, or certain regions (such as the diagonal $j = k$) of the diallel have been underexplored, the estimates are very close to the prior. This is different from, for example, a `LASSO` which would choose to threshold away a parameter down to zero because it lacks information. Alternatively, the Bayesian hypothesis in `BayesSpike` suggests that we do not have much additonal posterior information, yes-or-no to τ_j^2's inclusion. When a low posterior value $\mathcal{P}(\tau_j^2 = 0|Y)$ is reported, this means that the model selection procedure is expressing high confidence that this τ_j^2 is indeed zero, as demonstrated by the data.