Genomic models in bayz

Luc Janss, Dec 2010

In the new "bayz" version the genotype data is now restricted to be 2-allelic markers (SNPs), while the modeling option have been made more general. This implements (slightly) different set-up and options for the familiar genomic models. The new bayz is also prepared for multi-trait models and has been tested on SNP data with up to 800K SNPs. Efficient MH samplers are implemented to achieve fast(er) convergence on big SNP data sets. Estimation of genomic variance can be done on any model in a post-analysis step and genomic prediction can be made on any new set of genotypes.

Background on change in handling marker genotypes

The old iBay versions was set-up to allow for multiple alleles (>2) at markers (originating from QTL mapping with microsattelites). Handling multiple alleles requires to set up estimates for all alleles at the marker (with a constraint to be zero on average), and to use a variance component per marker to model the variance (similar to Meuwissen's BayesA). In the old iBay version the variance component per marker was implemented as a scaling factor. The old version also handled missing alleles as a separate missing class. This is why the old version always had at least 2 allele effects, and often (with missing genotype data) even 3.

The new bayz version has some important changes: now only 2 alleles are handled per marker (only allowing SNPs so far), and multi-allelic markers are at the moment not supported. Also missing alleles are treated differently, and are inserted as the average (expected) genotype value. The standard in bayz is now to model 1 regression coefficient per marker representing the allele substitution (or average/additive) effect of the second coded allele. The new approach is also made more efficient in storage of genotypes, now needing 0.1 GB of memory per 1000 individuals x 100K markers.

New Normal / Gaussian common prior model

With only 2 alleles and 1 regression per marker, it is now possible to specify a "G-BLUP" model. This is a "random regression" model by modeling a variance component for the SNP-regression coefficients, specifying a Normal (Gaussian) distribution for regressions coefficients. This basic "random regression" model is specified as:

```
phen1 = mean add.snp
mean.phen1 ~ uni
resid.phen1 ~ norm iden *
add.snp.phen1 ~ norm iden *
var.resid.phen1 ~ uni
var.add.snp.phen1 ~ uni
```

here add.snp is the model term that models a vector of regression coefficients for SNP effects, and this term (with the trait name appended) is set to have an identity Normal distribution. This is the same as specifying the model:

$$y = 1\mu + Xb + e$$

with the specifications for distributions of residuals and SNP effects:

$$e \sim N(0, I\sigma_e^2) \qquad b \sim N(0, I\sigma_b^2)$$

The variances for these two distributions are called in bayz "var.resid.*trait*" and "var.add.snp.*trait*", and are themselves set to have uniform priors (but this can also be chosen as inverse chi square). The need to set distributions for these variance parameters comes from using a * in the distribution of resid.phen1 and add.snp.phen1, which means the parameter is unknown and will be estimated.

Note 1: with >10K SNPs it is necessary to use an MH sampler to make the variance component mix well (see the manual on the web site for details).
Note 2: this "BLUP" model can also be treated in multi-trait, by setting the "lsd" multi-variate variance structure on all add.snp.* random effect vectors of SNP effects.
Note 3: dominance effects can be added as well using the dom.snp term. The standard approach would then model a separate variance for dominance effects.

New "Long tail" common prior model (*provisional*)
The old iBay common prior model was a model with a long-tail distribution for allele effects. If you want to repeat a long-tail model, this is now possible by use of the "Bayesian LASSO" model, which is a model using a double-exponential prior on regression parameters. This Bayesian LASSO model is not exactly the same as the old iBay common prior model, and also not exactly the same as Meuwissen BayesA, but they are all long-tail models and should have very similar effect. The popularity of the Bayesian LASSO model is increasing and bayz has made a very efficient implementation of it including the possibility to estimate the hyper parameter of the Bayesian LASSO model. This model is specified by switching the prior on SNP regressions to "dexp":

```
phen1 = mean add.snp
mean.phen1 ~ uni
resid.phen1 ~ norm iden *
add.snp.phen1 ~ dexp *
var.resid.phen1 ~ uni
```

In statistical formulation this specifying the distributions for every SNP regression as:

$$b_i \sim dexp(\lambda)$$

where dexp() means the double exponential (also known as LaPlace) distribution. In density form this is the distribution:

$$b_i \propto \lambda \exp(-\lambda|b_i|)$$

The implementation of this model is still in a testing phase, and a standard default uniform prior is added when the 'lambda' parameter is taken as unknown (not requiring now to set another line for this distribution). When the 'lambda' parameter from the dexp distribution is estimated, it is put in the output as dexp.add.snp.trait, and is close to an inverse standard deviation.
Note 1: also this model has an MH sampler option to make the dexp parameter better mix for large sets of SNPs (recommended already for >10K SNPs).
Note 2: a genomic variance estimate and genomic predictions can still be made using gbayz also under this model.

Variances by groups
A new feature added is to model heterogenous variances according to a given factor in the data. A classical uses of this would be to model heterogeneous herd variances in cattle, with a variance per herd. This model typically adds some higher-level distribution to smooth the individual variances towards a common mean. This is done by setting a common inverse-chi-square distribution for the individual group variances. The scale parameter (close to a mean) of this chi-square can be estimated from the data. The weight parameter in this distribution determines the strength of the smoothing towards the common mean: with a higher weight the common mean is weighted more heavily.
When applied to SNPs, a simple example is to apply it to estimate a variance per chromosome. The factor information (chromosome) for this use should be in the map data. The specification of this model is:

```
phen1 = mean add.snp
mean.phen1 ~ uni
```

```
resid.phen1 ~ norm iden *
add.snp.phen1 ~ norm fac.chrom **
var.resid.phen1 ~ uni
var.chrom.add.snp.phen1 ~ ichi *0.1 100
```

The specification of the add.snp.phen1 distribution uses here a feature to add a “fac” term in the distribution, which says that the random effects have a variance structure with different variance per level given in “chrom” (“chrom” must be a field in the map data). Because this can be many levels, the double star (**) is the most easy way to say that all these parameters are unknown and should be estimated. A ‘norm fac.X’ term creates a new term ‘var.X....’ which represents the list of variances. They can be given a common inverse chi-square distribution to smooth them to a common mean. The scale parameter of this inverse chi-square distribution can also use a star to specify that it is unknown and should be estimated from the data. At this point it is not necessary to specify a prior distribution for the scale parameter as it is standardly taken as uniform.
The statistical formulation of this model is that SNP regression coefficients are grouped, say, b_{ij} is the jth SNP in group i (where groups can be chromosomes, genome sections, GO families, etc.). Then the distributional assumptions are:

$$\begin{aligned} b_{ij} &\sim N(0, \sigma_i^2) \\ \sigma_i^2 &\sim \chi^{-2}(s^2, 100) \\ s^2 &\sim Uniform \end{aligned}$$

Mixture model

Also the mixture model is now different from the iBay implementation as it can be specified directly on the regression coefficients for SNP effects. The old iBay model specified it on the variance parameters (scaling factors) per marker. Direct specification of the mixture model on regression coefficients is in line with the classical papers on variable selection, but of course only works for SNPs which have just 1 regression coefficient per marker (and bayz is now restricted to SNPs).
The rationale behind the mixture model is that a mixture is like an unknown factor: a factor where the level assignments have to be estimated in the model. Apart from that, it can be treated as a standard factor, for instance to separate data in groups with two means (the classical mixture distribution fit). In the variable selection approach, the mixture model is used to separate random effects in groups with two variances.
Because the mixture parameter is considered as a factor, it belongs to a certain data set. For the SNP model selection, this is a factor in the “map” data sheet. The specification of this model in bayz is:

```
z$map <- bern 0.98 0.02
phen1 = mean add.snp
mean.phen1 ~ uni
resid.phen1 ~ norm iden *2
add.snp.phen1 ~ norm fac.z 0.001 *1
var.resid.phen1 ~ uni
var.z.add.snp.phen1 ~ ichi 1 10
```

where z$map <- bern is the statement inserting an extra factor in the map data, here a two level factor by use of a bernouilli (bern) distribution, which has two probabilities. Once z is defined (it may have any name), it can be used as a factor in the variance model in exactly the same way as “chrom” in the model for variances per groups.
However, to achieve the variable selection, the variances for the “z” factor should not be smoothed to a common mean, like was done in the model with variances per chromosome. Here it is the aim to keep these variances well apart, with a very small one

(kept fixed above), and a bigger one (which can be good to take it unknown and estimate it from the data as done with the star above).

Note 1: when estimating the "big" variance from the data, this can cause a trap in the markov chain when using a flat prior: when accidentally <3 markers are selected in the big-effects group, a flat prior will make it impossible to update the variance. Bayz will still continue, but with a warning, but the inference is not quite correct. The nicest solution to avoid this trap is to use an informative prior, which will always allow to sample the variance even in the extreme case where no markers would be in the "big effect" group. This can still be a sensible approach when also updating the mixture proportions (see note 2).

Note 2: the mixture proportions can now also be updated in bayz: just add a star to the Bernouilli parameters. This does require to set a prior distribution for the unknown proportion which is a Beta distribution. Estimating the mixture proportion with a flat Beta prior distribution is set as:

```
z$map <- bern *0.98 *0.02
.....
frq.z ~ Beta 1 1
```

Informative Beta distributions can be set by choosing one or both Beta parameters >1. The parameters of the Beta distribution can be interpreted as the extra prior counts (minus 1) for the two groups in "z".

Note 3: As an alternative to fix the "small variance" and estimate the "big variance" in the mixture model, also both can be estimated with a fixed ratio. This looks to be a very interesting and robust approach which does not require to use an informative prior (a flat prior can be used on these variance). The fixed ratio updates are implemented using an MH sampler, for which the !step option can be set to optimize its acceptance rate. The fixed ratio updates are set using:

```
add.snp.phen1 ~ norm fac.z *0.001 *1 !ratios !step=0.01
```

For large random effect vectors (which is often the case with SNPs....) it may be necessary to set a small STEP for the MH sampler, for instance 0.01 or 0.02 can be a good choice (the default is 0.03).

Gene mapping with the mixture model

The mixture model is the most suited to perform a gene mapping: it gives the most clear separation between signals and noise, and also provides a significance measure by computing Bayes Factors on the mixture probabilities.

Note: It is important to estimate variance parameters in the mixture distribution so that the model can adapt to the available evidence in the data about the size of important effects. This can be estimation of the "big variance", or estimation of both variance with a ratio constraint. Mixture proportions would then be kept fixed, allowing to measure the change (as a Bayes Factor) between prior and posterior probabilities for every SNP to belong to the two classes in the mixture. Although the mixture proportion is then fixed (putting in prior information on the expected *number* of big effects), the procedure is still robust, because there is no prior information about the *size* of big effects. Also, the Bayes Factor measures the relative change between prior and posterior, which is relatively independent from the prior mixture probabilities used (within reasonable ranges).

The Bayes Factor to declare significance for an effect is defined as:

$$BF_i = \frac{p_i(1-p_i)}{\pi(1-\pi)}$$

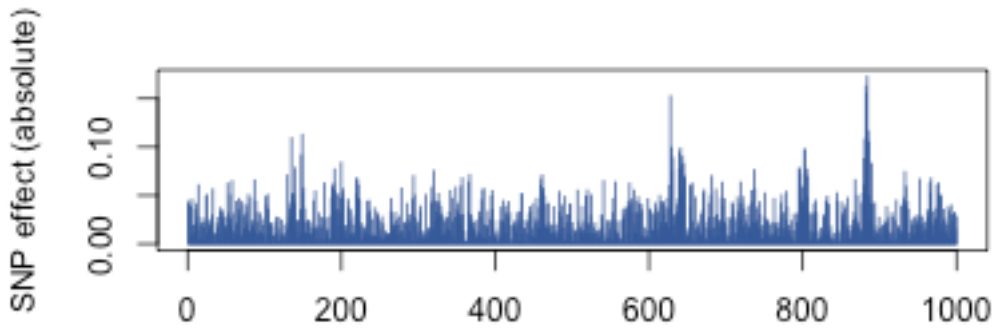
where p_i is the estimated posterior probability and π is the set prior probability to fall in the second group of the mixture (probability of 'big effect').

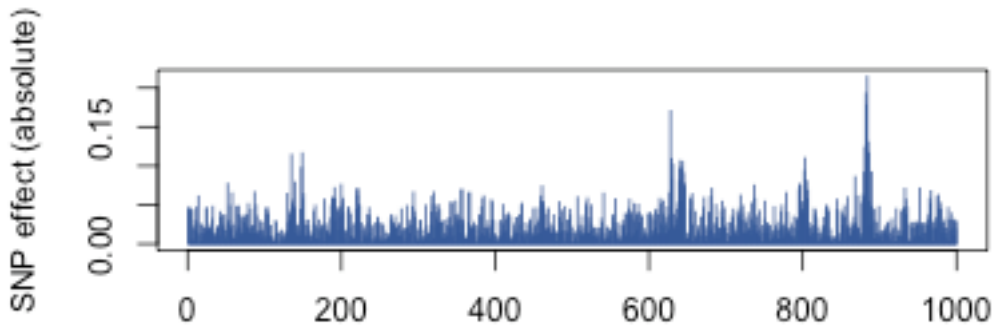
Estimation of genomic variance and genomic values

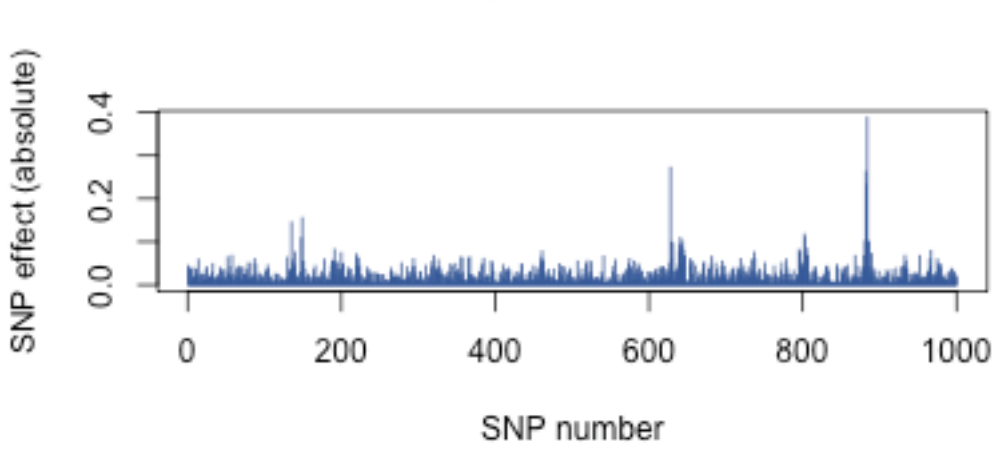
SNP-explained, or “genomic”, variances can be constructed from any of the above models using the post-analysis tool “gbayz”. This post-analysis uses only the MCMC samples for allele effects (therefore can be done with all genomic modes), and combines it with marker data from a particular group of individuals to estimate the explained variance in that group of individuals. By use of this marker data, marker frequencies and LD between markers are taken into account, but the estimated explained variance is specific for that group of individuals.
The gbayz tool also produces genomic values (the sum of SNP effects pertaining to an individuals). As this is based on MCMC samples of SNP effects, this produces posterior means and posterior standard deviations (‘standard errors’) for the genomic value estimates. By combining allele effect estimates made on one data set (a ‘training data’), with a gbayz post-analysis using a different marker data set (a ‘test’ or ‘validation data’), this allows to make genomic predictions. With the MCMC samples of SNP effects stored it is easy to repeat the computation of the genomic predictions for different batches of individuals.
The gbayz post analysis can also compute explained genomic variance for identified groups of SNPs. This can be applied to collect the collective SNP effects from larger regions, e.g. per gene / QTL location or arbitrary genome sections and chromosomes.
For details on using gbayz see the online manual.

Demonstration

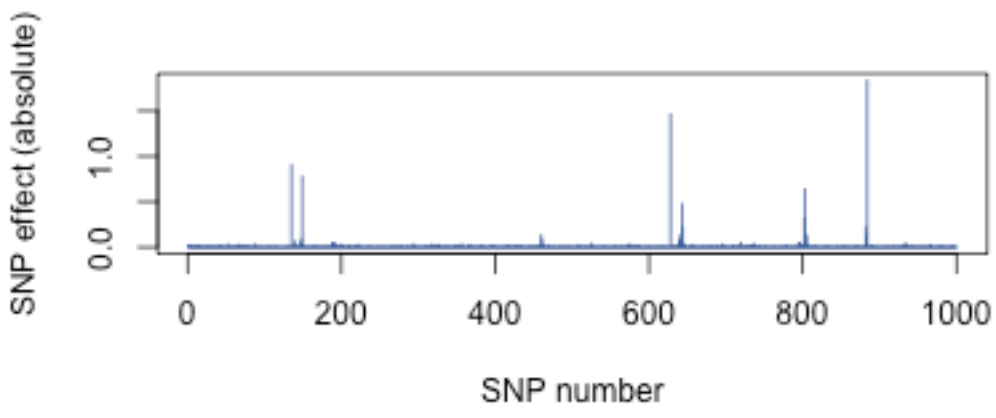
Figures on the last page show typical genome mapping plots showing SNP effects from the simultaneous SNP fit along a genome section, using a simulated data set with 1000 SNPs and 20 real effects generated (of which approx. 6 can be retrieved). Top panel: SNP effects using a Normal (aka Gaussian) distribution on SNP effects; 2nd panel: a model with SNP variances differing along the genome sections in block of 50 SNPs and smoothed to a common mean; 3rd panel: the “Bayesian LASSO” model using a double exponential (aka LaPlace) distribution on SNP effects; bottom panel: a model using a mixture (spike/slab type) distribution on SNP effects. Note how the scales on the y-axis are not equal: in the bottom panel the scale is 10-fold that of the top panel, showing how the mixture model makes a much stronger separation of signals and noise, absolutely (in terms of estimates of the big effects) and relatively (in the difference between ‘big’ and ‘small’ effects). Here a Bayes Factor cut-off at 5 identifies 6 SNPs without finding false positives. The performance of the second model, which varies the variance of SNP effects in genome sections, depends on clever choice of the SNP grouping and the weight used to pull the individual variance to the common scale parameter. In this demonstration the approach differs only slightly from the approach with common variance for all SNPs (1st model).


SNP effect (absolute)
0.00
0.10
0
200
400
600
800
1000


SNP effect (absolute)
0.00
0.15
0
200
400
600
800
1000


SNP number
SNP effect (absolute)
0.0
0.2
0.4
0
200
400
600
800
1000
SNP number

.


SNP effect (absolute)
0.0
1.0
0
200
400
600
800
1000
SNP number