Genomic selection

3- Normal, Lasso and BayesCPi models for Genomic selection. Computation.

Andrés Legarra - INRA

Acknowledgements

- ANR projects Amasgen, Rules&Tools; Apisgene
- Toulouse bioinformatics platform (bioinfo.genotoul.fr)
- GENOMIA funding:
  www.poctefa.eu

Recall

- We want to estimate SNP effects **a** by a model
  - **y = Xb + Za + e**
- **Z** contains (somehow) genotypes

3

Allele coding

- we fit a regression of genetic value on gene content (as in Falconer)
- a_i is the effect of the SNP
- We code explanatory variables in **Z** (**y**=**Xb+Za+e**) as:

– « 11 » =	0	$-a_i$	$(-2p)a_i$
– « 12 » =	a_i	0	(1-2p)ai
– « 22 » =	$2a_i$	a_i	(2-2p)ai

Strandén & Christensen (http://www.gsejournal.org/content/43/1/25) refer to these as « 012 », « 101 » and « centered »

So we have different **Z**'s depending on how we code…

5

Allele coding

Quoting Strandén & Christensen http://www.gsejournal.org/content/43/1/25 :

The good news

« parameter estimates [h^2] and estimated marker effects [and EBV's] in marker-based models are the same irrespective of the allele coding, provided that the model has a fixed general mean [or any « fixed » effect] »

[*n.b.*, EBV's are shifted by a constant depending on the allele coding]

« allele coding affects the mixing of Markov chain Monte Carlo algorithms [and iterative solvers], with the centered coding being the best »

The bad news

« Reliabilities of estimated genomic breeding values calculated using elements of the inverse of the coefficient matrix depend on the allele coding because different allele coding methods imply different models »

6

Example of centered coding

- 2 SNP, 4 individuals

```
11  12
22  11
12  11
11  11
```

- p_i = 3/8, 1/8

the sum of each column of **Z** is 0

$$\mathbf{Z} = \begin{bmatrix} -0.75 & 0.75 \\ 1.25 & -0.25 \\ 0.25 & -0.25 \\ -0.75 & -0.25 \end{bmatrix}$$

10

- Let’s go back to (prior) distributions for SNP effects

11

A priori Distributions for marker effects

- Several distributions for SNP effects have been proposed
  - Normal (Meuwissen et al., Genetics 2001; Van Raden JDS 2008) -> BLUP_SNP or GBLUP or RR-BLUP
  - BayesA, BayesB, (Meuwissen et al. 2001; Habier et al., 2011)
  - Mixture of normal , BayesC(Pi) (Van Raden JDS 2008, Habier et al., 2011)
  - (Bayesian) Lasso (Usai et al., 2009; De los Campos, et al., 2009)

12

A common misconception

- « BLUP_SNP assumes the same variance for all loci, whereas BayesA does not »
  - This is basically a semantic issue, it's hard to agree or not
  - The part of genetic variance explained by locus *i* is
    - $2p_iq_ia_i^2$ (Falconer) where « a_i » is the true effect
  - we have *estimates* of « a_i »
    - *after* fitting the data with BLUP_SNP (or whatever) the genetic variance explained by each locus will be different

13

A common misconception

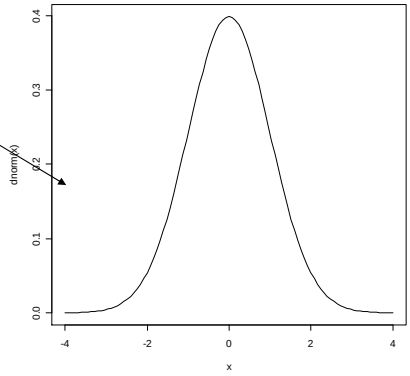
  - we have *estimates* of « a_i »
    - *after* fitting the data, the genetic variance explained by each locus will be different, with BLUP_SNP or BayesA or whatever
  - something different is the a priori *distribution* of the SNP effects
    - Different variances by locus are a shortcut to model (= a way to write) complex distributions
    - e.g. the Bayesian Lasso or BayesA can be understood as inferring a variance for each SNP effects *after fitting the data*
      - but what is really different is the assumed shape of SNP effects
    - *This estimate is very bad* (Gianola et al., 2009) because we have very little data for each locus

14

Normal distribution

$$a_i \sim N\left(0, \sigma_a^2\right)$$

Few « big » effects



15

Normal equations for genomic selection (BLUP_SNP)

- If we assume normality there are closed expressions for **â**
- This is called « BLUP », and also « genomic BLUP » , BLUP_SNP, or GBLUP, but also « ridge regression » or Random Regression-BLUP
  - I will keep GBLUP for the use of the genomic relationship matrix
  - and BLUP_SNP for the direct estimation of SNP effects

16

Mixed model equations for BLUP_SNP

- Henderson's MME
- **Z'Z** is *not* diagonal
- Var(**a**)=**D** is diagonal if we assume uncorrelated SNP effects

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

Could (will) be something different !!

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \sigma_a^2 = \mathbf{I}\sigma_a^2$$

17

GS with Residual Update

- How to estimate SNP effects efficiently

A. Legarra and I. Misztal J. Dairy Sci. 2008. 91:360-366.

(but we were not the only ones: R Fernando, G De los Campos, P vanRaden, developed the same trick)

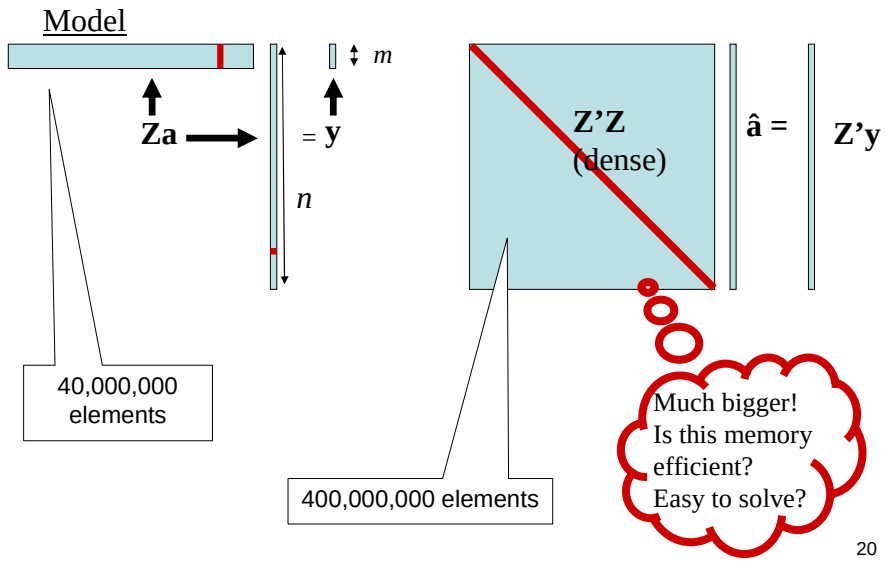
18

GS with Residual Update

- Let assume a random SNP model (“BLUP_SNP”)
- Mixed model equations can be solved by direct inversion (1 iteration) or Gauss-Seidel, PCG or Jacobi (iterative methods, useful for large matrices).
- MCMC (BayesB, etc) can be done starting from Gauss-Seidel
- The number of effects (SNP) *n* is much larger than the number of records *m,* and the matrix **Z’Z** is dense. A typical example (2000 records, 20000 SNP):

19

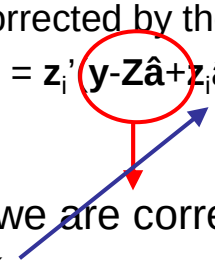
The size of the MME



20

Reordering Gauss Seidel

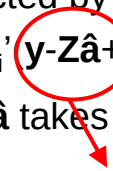
- Gauss Seidel uses the conditional mean for the i-th effect, corrected by the other effects:
- $(\mathbf{z}_i'\mathbf{z}_i + \lambda)\,\hat{a}_i^{l+1} = \mathbf{z}_i'(\mathbf{y}-\mathbf{Z}\hat{\mathbf{a}}+\mathbf{z}_i\hat{a}_i^{l})$



- Note that we are correcting for $\hat{a}_i$, so we put it back

Reordering Gauss Seidel

- Gauss Seidel uses the conditional mean for the i-th effect, corrected by the other effects:
- $(\mathbf{z}_i'\mathbf{z}_i + \lambda)\,\hat{a}_i^{l+1} = \mathbf{z}_i'(\mathbf{y}-\mathbf{Z}\hat{\mathbf{a}}+\mathbf{z}_i\hat{a}_i^{l})$
- Correcting for $\mathbf{Z}\hat{\mathbf{a}}$ takes 20000 op.



- This is the residual $\hat{\mathbf{e}}$, isn't it?
- Use alternative formula

$(\mathbf{z}_i'\mathbf{z}_i + \lambda)\,\hat{a}_i^{l+1} = \mathbf{z}_i'\hat{\mathbf{e}}+\mathbf{z}_i'\mathbf{z}_i\hat{a}_i^{l+1}$

Reordering the error term

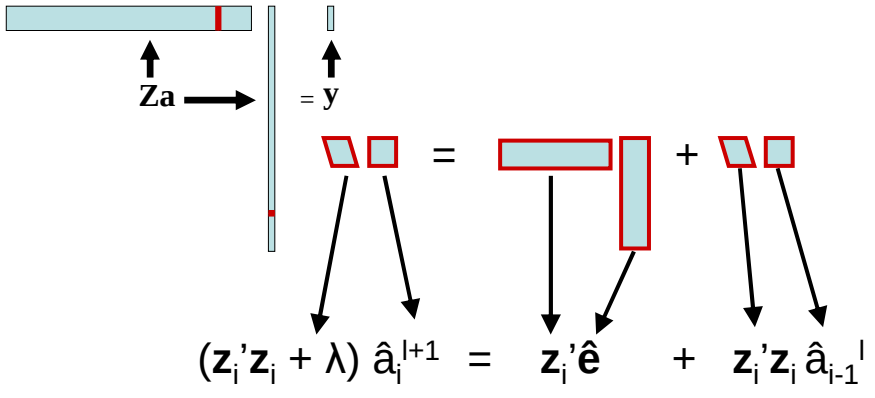
- Still we need to compute **ê** at each iteration
- Actually only $\hat{a}_i$ changed
- It can be shown that **ê** can be « updated »

$$\hat{\mathbf{e}}^{l+1} = \hat{\mathbf{e}}^{l} - \mathbf{z}_i(\hat{a}_i^{l+1} - \hat{a}_i^{l})$$

  - Hence « GSRU » Gauss Seidel with Residual Updating
  - Some machine learning literature calls this « backfitting »

24

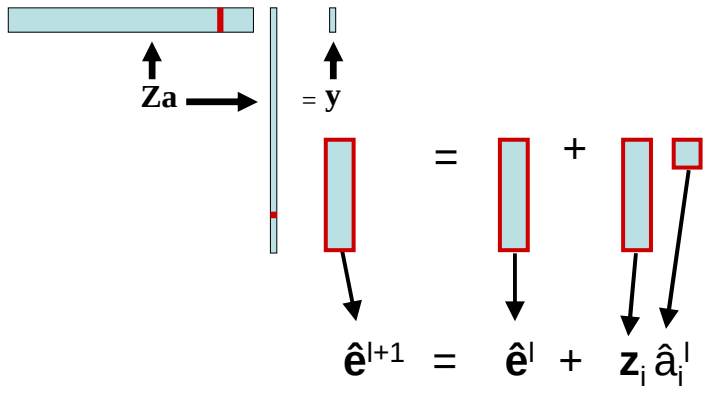
GSRU in Figure



1- Gauss-Seidel

25

GSRU in Figure



1- Residual Updating

R pseudocode

```
#get diagonal of X'X
for (i in 1:neq) {
  xpx[i]=crossprod(X[,i],X[,i])
}
for (iter in 1:1000) {
  #Gauss Seidel
  for (i in 1:neq){
    lhs=xpx[i]+lambda
    rhs=crossprod(X[,i],e) + xpx[i]*ahat[i]
    val=rhs/lhs
    e = e - X[,i]*(val - ahat[i])
    ahat[i]=val
  }
}
```

Fortran pseudocode

```
Double precision:: xpx(neq),y(ndata),e(ndata),X(ndata,neq), &
sol(neq),lambda,lhs,rhs,val
do i=1,neq
   xpx(i)=dot_product(X(:,i),X(:,i)) !form diagonal of X'X
enddo
e=y
do until convergence
   do i=1,neq
     !form lhs X'R-1X + G-1
     lhs=xpx(i)/vare+1/vara
     ! form rhs with y corrected by other effects (formula 1) !X'R-1y
     rhs=dot_product(X(:,i),e)/vare +xpx(i) *sol(i)/vare
     ! do Gauss Seidel
     val=rhs/lhs
     ! MCMC sample solution from its conditional (commented out here)
     ! val=normal(rhs/lhs,1d0/lhs)
     ! update e with current estimate (formula 2)
     e=e - X(:,i)*(val-sol(i))
     !update sol
     sol(i)=val
   enddo
enddo
```

28

Good performance

- *3nm* operations (GSRU) vs n^2 (GS) or n^3 (Cholesky)
- In our example, 60000 *vs* 400000000
- 100-fold faster
- Fastest than any other Gauss Seidel method
- GS with recreation of equations awful (97 h *vs* 1 min)

30

Preconditioned Conjugate Gradients

- The other method of choice to solve large systems of equations (e.g. Strandén and Lidauer, 1998; Tsuruta et al., 2001)

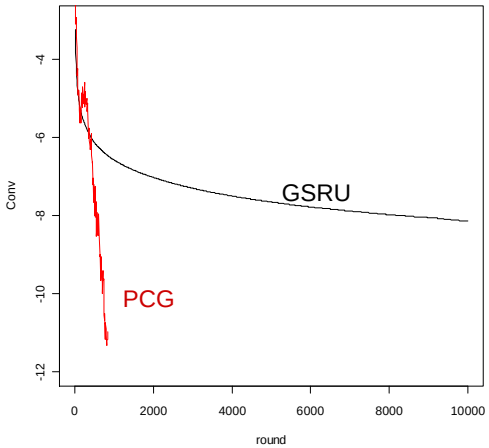
$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix} \Rightarrow \mathbf{A}\mathbf{x} = \left(\mathbf{W}'\mathbf{W}+\boldsymbol{\Sigma}^{-1}\right)\mathbf{x} = \mathbf{t}$$

- Based on repeated computations of **Ax** above
- Can easily be done for genomic models as **W**'(**Wx**) + $\sum^{-1}$**x** at a cost of *3nm* operations
- PCG is much faster (but less general)

do until "convergence"
  n = n + 1; $\mathbf{w}_n = \mathbf{A}\mathbf{p}_{n-}$
  $\alpha_n = e_{n-1}/(\mathbf{p}_{n-1}'\mathbf{w}_n)$
  $\mathbf{x}_n = \mathbf{x}_{n-1} + \alpha_n\mathbf{p}_{n-1}$
  if mod(n, 50) = 0 then
    $\mathbf{r}_n = \mathbf{b} - \mathbf{A}\mathbf{x}_n$
  else
    $\mathbf{r}_n = \mathbf{r}_{n-1} - \alpha_n\mathbf{w}_n$
  end if
  $\mathbf{w}_n = \mathbf{M}^{-1}\mathbf{r}_n$
  $e_n = \mathbf{r}_n'\mathbf{w}_n$
  $\beta_n = e_n/e_{n-1}$
  $\mathbf{p}_n = \mathbf{w}_n + \beta_n\mathbf{p}_{n-1}$
end do

Preconditioned Conjugate Gradients for BLUP_SNP

log10(Convergence) with real data (Holstein)



PCG is much faster

GSRU convergence slow for large data sets (or you really need to wait)

Still, EBV's seem identical, possibly because errors in SNP estimates cancel out when summing.

BLUP_SNP parameters

- Still, we need σ^2_a and σ^2_e in

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{D}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}; \mathbf{R} = \mathbf{F}\sigma^2_e; \mathbf{D} = \mathbf{I}\sigma^2_a$$

- Both can be deduced from previous pedigree-based estimates, σ^2_g (σ^2_g in some other slides)
  - Using a formula proposed by several authors (e.g. Gianola et al., 2009)
- How do we get the variance of SNP effects, σ^2_a, from a genetic variance σ^2_g ?
- The formula comes from the sampling variance of covariates in **Z** affecting SNP effects to data
  - i.e., we try to explain all genetic variance as if « caused » by SNP effects, and these SNP effects have a variance of σ^2_a
- Assumes Hardy-Weinberg and Linkage equilibrium $\sigma^2_a \approx \dfrac{\sigma^2_g}{2\sum_{all\ SNPs} p_i(1-p_i)}$

34

BLUP_SNP parameters

- Note that the idea can be reversed
- How does the genetic variance σ^2_g relate to the SNP effect variance, σ^2_a ?
  - i.e., we try to explain all genetic variance as if « caused » by SNP effects, and these effects have a variance of σ^2_a

$$\sigma^2_g \approx \sigma^2_a 2\sum_{all\ SNPs} p_i(1-p_i) = \text{sum2pq}\,\sigma^2_a$$

You can estimate genetic variance in wild populations, for instance

Some programs provide estimates of these (e.g. GS3, BIGS)

35

BLUP_SNP parameters

- The other option is to estimate σ^2_a and σ^2_e
- They are usually similar but not always :-/

$$\sigma^2_g \approx \sigma^2_a 2 \sum_{all\ SNPs} p_i (1 - p_i) = \text{sum2pq}\, \sigma^2_a$$

TRUE ?

36

Estimating variances = BayesC

- It simply consists in a BLUP_SNP where we estimate (and simultaneously « integrate out ») σ^2_a and σ^2_e
  - i.e., a regular Gibbs sampler applied to SNPs instead of EBV's (G-Gibss??)
  - Legarra et al., 2008 (we didn't call it BayesC), Habier et al., 2011

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e},$$

$$\mathbf{a} \mid \sigma^2_a \sim MVN\left(\mathbf{0}, \mathbf{I}\sigma^2_a\right); \sigma^2_a \sim S_a \chi^{-2}_v$$

$$\mathbf{e} \mid \sigma^2 \sim MVN\left(\mathbf{0}, \mathbf{I}\sigma^2_e\right); \sigma^2_e \sim S_e \chi^{-2}_v$$

- Pretty straightforward from GSRU
- You can as well estimate σ^2_a and σ^2_e using « BayesC » and take them as known in BLUP_SNP (e.g. as in REML+BLUP analysis)

37

Fortran pseudocode for BayesC

```
...
do j=1,niter
   do i=1,neq
     !form lhs
     lhs=xpx(i)+1/vara
     ! form rhs with y corrected by other effects (formula 1)
     rhs=dot_product(X(:,i),e)/vare +xpx(i) *sol(i)/vare
     ! MCMC sample solution from its conditional
     val=normal(rhs/lhs,1d0/lhs)
     ! update e with current estimate (formula 2)
     e=e - X(:,i)*(val-sol(i))
     !update sol
     sol(i)=val
   enddo
   ! draw variance components
   ss=sum(sol**2)+nua*Sa
   vara=ss/chi(nua+nsnp)
   ss=sum(e**2)+nue*Se
   vara=ss/chi(nue+ndata)
enddo
```

38

Legarra et al.2008, Mice data

- There is no full agreement for estimates of genetic variance (σ^2_u) across models (pedigree vs. genomic)
- This is because definition of base population is different, and data used too.
- In dairy the agreement is good but not complete

TABLE 1

Variance components estimates for different models of genomic selection

Model	σ^2_a	σ^2_u	σ^2_c	σ^2_e
		Weight		
1		4.59	2.12	0.16
2	3.52E-04	→ 1.33	3.34	1.94
3	2.52E-04	3.56	2.15	0.19
		Growth slope		
1		8.37E-04	9.72E-04	8.22E-04
2	1.04E-07	→ 3.93E-04	10.30E-04	10.79E-04
3	1.00E-07	2.36E-04	9.65E-04	9.57E-04
		Body length		
1		0.040	0.048	0.146
2	9.09E-06	→ 0.034	0.051	0.150
3	8.58E-06	0.010	0.048	0.144
		Body mass index		
1		2.49E-04	3.91E-04	18.72E-04
2	0.80E-07	→ 3.02E-04	3.94E-04	18.46E-04
3	0.77E-07	0.67E-04	3.75E-04	18.08E-04

Estimated variance components are shown for marker-locus effects **a**, random cage effects **c**, polygenic additive genetic effects **u**, and residual **e**.

BayesA

- We « estimate » a different σ^2_a *for each SNP*
  - this estimate is horribly bad
  - but SNP solutions correspond to a model with « t » distributions

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e},$$
$$\mathbf{e} \mid \sigma^2 \sim MVN\left(\mathbf{0}, \mathbf{I}\sigma_e^2\right); \sigma_e^2 \sim S_e\chi_\nu^{-2}$$
$$\left\{ \begin{array}{c} a_i \sim t\left(0, \nu, \sigma_a^2\right) \\ \equiv \\ a_i \sim N\left(0, \sigma_{a,i}^2\right); \sigma_e^2 \sim S_e\chi_\nu^{-2} \end{array} \right\}$$

representation as « t »

$$a_i \sim t\left(0, \nu, \sigma_a^2\right)$$
$$\equiv$$
$$a_i \sim N\left(0, \sigma_{a,i}^2\right)\chi_\nu^{-2}\sigma_a^{-2}$$

Meuwissen et al. representation

- Pretty straightforward from GSRU

40

Fortran pseudocode for BayesA

```
...
do j=1,niter
   do i=1,neq
     !form lhs
     lhs=xpx(i)+1/vara(i)
     ! form rhs with y corrected by other effects (formula 1)
     rhs=dot_product(X(:,i),e)/vare +xpx(i) *sol(i)/vare
     ! MCMC sample solution from its conditional (commented out here)
     val=normal(rhs/lhs,1d0/lhs)
     ! update e with current estimate (formula 2)
     e=e - X(:,i)*(val-sol(i))
     !update sol
     sol(i)=val
     ! draw variance components
      ss=sol(i)**2+nua*Sa
      vara(i)=ss/chi(nua+1)
   enddo
   ! draw variance components
   ss=sum(e**2)+nue*Se
   vare=ss/chi(nue+ndata)
enddo
```

41

BayesCPi

- e.g. Habier et al., 2011 (see also Rohan Fernando course notes)
- What if some SNP had no effect?
  - This is the original idea of BayesB
  - needs the probability that a given SNP is at the model or not
  - can be computed by MCMC

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e},$$

$$\mathbf{a} \mid \boldsymbol{\delta}, \sigma_a^2 \sim \begin{cases} a_i \sim N\left(\mathbf{0}, \sigma_a^2\right) if\ \delta_i = 1 \\ a_i = 0\ if\ \delta_i = 0 \end{cases}$$

$$\sigma_a^2 \sim S_a \chi_v^{-2}$$

$$\delta_i \sim \begin{cases} 0\, with\ probability\, \pi \\ 1\, with\ probability\, 1-\pi \end{cases}$$

$$\mathbf{e} \mid \sigma^2 \sim MVN\left(\mathbf{0}, \mathbf{I}\sigma_e^2\right); \sigma_e^2 \sim S_e \chi_v^{-2}$$

43

BayesCPi

- Algorithm consists in a BLUP_SNP by GSRU where we estimate (and simultaneously « integrate out ») σ^2_a and σ^2_e
  - for each SNP we compute the probability of it being « in » the model (indicator variable δ)
    - This was a nightmare in original BayesB
    - R Fernando found out a simple way of computing it (course notes: http://www.ans.iastate.edu/stud/courses/short/2010short.html) that is « like » GSRU
  - we can equally compute the proportions π or fix them previously

44

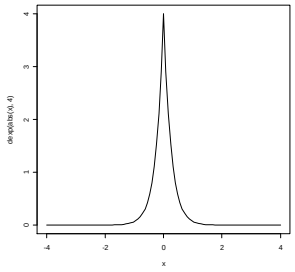
Fortran pseudocode for BayesCPi

```
...
do j=1,niter
   do i=1,neq
     ...
     ! compute loglikelihood for state 1 (i -> in model) and 0 (not in model)
     ! Notes by RLF (2010, Bayesian Methods in Genome Association Studies, p 47/67)
     v1=xpx(i)*vare+(xpx(i)**2)*vara
     v0=xpx(i)*vare
     rj=rhs*vare ! because rhs=X'R-1(y corrected)
     ! prob state delta=0
     like2=density_normal((/rj/),v0) !rj = N(0,v0)
     ! prob state delta=1
     like1=density_normal((/rj/),v1) !rj = N(0,v1)
     ! add prior for delta
     like2=like2+pi; like1=like1+(1-pi)
     !standardize
     like2=like2/(like2+like1); like1=like1/(like2+like1)
     delta(i)=sample(states=(/0,1/),prob=(/like2,like1/)
     if(delta(i)==1) then
       val=normal(rhs/lhs,1d0/lhs)
     else
       val=0
     endif
   ...
   enddo
    ! here go the updates for vara and pi
enddo
```


Lasso

Hierarchical representation of Lasso

- **y** : data
- **a** : SNP effects



$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e},$$

$$\mathbf{a} \mid \lambda, \sigma^2 \sim \prod_i \frac{\lambda}{2} \exp\left(-\lambda \left| a_i \right|\right)$$

$$\mathbf{e} \mid \sigma^2 \sim MVN\left(\mathbf{0}, \mathbf{I}\sigma_e^2\right)$$

Distribution of SNP effects

Bayesian Lasso

- In regular Lasso, λ is tipically computed by cross-validation
  - which depends strongly on the constitution of the training & validation data sets
  - and is tricky to compute
- the Bayesian Lasso (Park & Casella 2008) uses an equivalent hierarchical model

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e};\ p(\mathbf{a}|\boldsymbol{\tau}) \sim N(\mathbf{0}, \mathbf{D}\sigma^2);\ diag(\mathbf{D}) = \tau_1^2 \cdots \tau_n^2;$$
$$p(\boldsymbol{\tau}|\lambda) = \prod_i \frac{\lambda^2}{2} e^{-\lambda^2 \tau_i^2/2};\ p(\mathbf{e}) \sim N(\mathbf{0}, \mathbf{I}\sigma^2) \quad (1)$$

- This Bayesian Lasso is being used for genomic selection (De los Campos et al., 2009)
- The following is largely from Legarra et al. (Genetical Res., 2011)

47

Bayesian Lasso

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e}$$

$$p\left(\mathbf{a} \mid \boldsymbol{\tau}, \sigma_e^2\right) = N\left(\mathbf{0}, \mathbf{D}\sigma_e^2\right); \mathbf{D} = \begin{pmatrix} \tau_1^2 & 0 & 0 & 0 \\ 0 & \tau_2^2 & 0 & 0 \\ & & \cdots & \\ 0 & 0 & 0 & \tau_n^2 \end{pmatrix}$$

$$p\left(\boldsymbol{\tau}^2 \mid \lambda\right) = \prod_i \frac{\lambda^2}{2} \exp\left(-\lambda^2 \tau_i^2 / 2\right)$$

$$p\left(\mathbf{e} \mid \sigma_e^2\right) = N\left(\mathbf{0}, \mathbf{I}\sigma_e^2\right)$$

These are the σ^2_{ai} in BayesA, BayesB

48

Bayesian Lasso

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e}$$

Regular

Scaled variances (different for each SNP)

Take-home message:
(Bayesian) Lasso is a just a linear mixed model with an exponential prior distribution for variances of SNP effects

$$p\left(\mathbf{a} \mid \boldsymbol{\tau}, \sigma_e^2\right) = N\left(\mathbf{0}, \mathbf{D}\sigma_e^2\right)$$

$$p\left(\boldsymbol{\tau}^2 \mid \lambda\right) = \prod_i \frac{\lambda^2}{2} \exp\left(-\lambda^2 \tau_i^2\right)$$

Park and Casella say "to achieve unimodal posteriors" (?)

Exponential prior for the variances

$$p\left(\mathbf{e} \mid \sigma_e^2\right) = N\left(\mathbf{0}, \mathbf{I}\sigma_e^2\right)$$

Residual variance

49

Park & Casella BL (BL1Var)

- SNP effects are modelled on the residual variance

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e};\ p(\mathbf{a}|\boldsymbol{\tau}) \sim N(\mathbf{0}, \mathbf{D}\sigma^2);\ diag(\mathbf{D}) = \tau_1^2 \cdots \tau_n^2;$$
$$p(\boldsymbol{\tau}|\lambda) = \prod_i \frac{\lambda^2}{2} e^{-\lambda^2 \tau_i^2/2};\ p(\mathbf{e}) \sim N(\mathbf{0}, \mathbf{I}\sigma^2) \quad (1)$$

- Hence: Bayesian Lasso with 1 Variance (BL1Var)
- Does it make much sense to model SNP effects on residual effects?
- Original lasso « augmented » with different variances, τ^2, for each SNP effect; these variances follow an exponential law controlled by λ
- this allows to compute individual τ^2 variances -with 1 level per random effect!- because they share a common structure
  - this estimate will be terribly bad

50

Tibshirani's BL (BL2Var)

- Assume SNP effects have a different « variance »

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e};\ p(\mathbf{a}|\boldsymbol{\tau}) \sim N(\mathbf{0}, \mathbf{D}\sigma_a^2);\ diag(\mathbf{D}) = \tau_1^2 \cdots \tau_n^2;$$

$$p(\boldsymbol{\tau}|\lambda) = \prod_i \frac{\lambda^2}{2} e^{-\lambda^2 \tau_i^2/2};\ p(\mathbf{e}) \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$$

- This is more similar to BLUP_SNP, BayesA, BayesC, etc etc.
- In fact σ^2_a is confounded with $1/\lambda$ and can be set to 1
  - equivalent to the Tibshirani's original Lasso

51

HetVar GBLUP

- Suppose that we know the true, different variances of SNP effects
- Then we could use BLUP_SNP (HetVar-GBLUP)

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Za} + \mathbf{e},$$

$$\mathbf{a} \,|\, \sigma_a^2 \sim MVN\left(0, \mathbf{D}\sigma_a^2\right)$$

- BL can be used to compute « a posteriori » estimates of variances of SNP effects: τ^2 and then **D**

$$\mathbf{D} = \begin{pmatrix} \tau_1^2 & 0 & 0 & 0 \\ 0 & \tau_2^2 & 0 & 0 \\ & & \cdots & \\ 0 & 0 & 0 & \tau_n^2 \end{pmatrix}$$

53

Fortran pseudocode for BL

```
...
do j=1,niter
   do i=1,neq
     !form lhs
     lhs=xpx(i)+1/vara(i)
     ! form rhs with y corrected by other effects (formula 1)
     rhs=dot_product(X(:,i),e)/vare +xpx(i) *sol(i)/vare
     ! MCMC sample solution from its conditional (commented out here)
     val=normal(rhs/lhs,1d0/lhs)
     ! update e with current estimate (formula 2)
     e=e - X(:,i)*(val-sol(i))
     !update sol
     sol(i)=val
     ! draw variance components
      ss=sol(i)**2
      vara(i)=1d0/rinvGauss(lambda2/ss,lambda2)
   enddo
   ! draw variance components
   ss=sum(e**2)+nue*Se
   vare=ss/chi(nue+ndata)
   ! update lambda
enddo
```

54

Genetic variation

- We can estimate genetic variation in the population (σ^2_u)using SNP data
  - Conceptual population in H-W, LE (Gianola et al., 2009)
- For BL2Var: $\sigma^2_u = 2\sum_i p_i(1-p_i)Var(a) = 2\sum_i p_i(1-p_i)\frac{2}{\lambda^2}$

  Genetic variation is proportional to $1/\lambda^2$
- For BL1Var: $\sigma^2_u = 2\sum_i p_i(1-p_i)\frac{2\sigma^2}{\lambda^2}$

  Residual variance enters in genetic variation
- For BLUP_SNP: $\sigma^2_u = 2\sum_i p_i(1-p_i)\sigma^2_a$
- For BayesA: $\sigma^2_u = 2\sum_i p_i(1-p_i)\hat{\sigma}^2_{ai}$
- For BayesCPi: $\sigma^2_u = (1-\pi)2\sum_i p_i(1-p_i)\sigma^2_a$

62

Legarra et al., 2011

TABLE 2. Estimates of population genetic variance σ^2_u (± standard errors) in Holstein

Trait	BL1Var	BL2Var	BayesC	Pedigree REML	Current values[a]
MY[b]	1260 ±50	448 ±27	451 ±26	570	635
FY	1876 ±84	710 ±44	710 ±39	893	973
PY	1127 ±50	429 ±24	428 ±20	473	520
FP	27.6 ±1.09	9.32 ±0.54	11.60 ±0.60	14.90	8.80
PP	5.51 ±0.03	1.66 ±0.10	1.60 ±0.12	2.56	2.19

- All methods give reasonable (but not the same) estimates except BL1Var

63

Non-MCMC methods

- So far we have seen mostly Gibbs
  - Lasso (and its cousin the Elastic Net) have an algorithm similar to the GSRU (BLUP_SNP)
  - VanRaden's (2008) nonlinearA has a GSRU with computations of likelihoods (much like in Bayes CPi), mixed with an EM algorithm for mixtures
  - There is also FastBayesB
- In theory these algorithms might give modes instead of « conditional expectations » but this does not seem to be a problem

64

- **Elastic Net** : Combine LASSO and BLUP_SNP
  with weight α compris entre 0 et 1

$$\hat{a} = \arg\ \min\left\{\sum_j \left(y_j - X\mathbf{a}_j\right)^2\right\} + \lambda\left(\alpha\sum_i a_i^2 + (1-\alpha)\sum_i |a_i|\right)\sum_i |a_i|$$

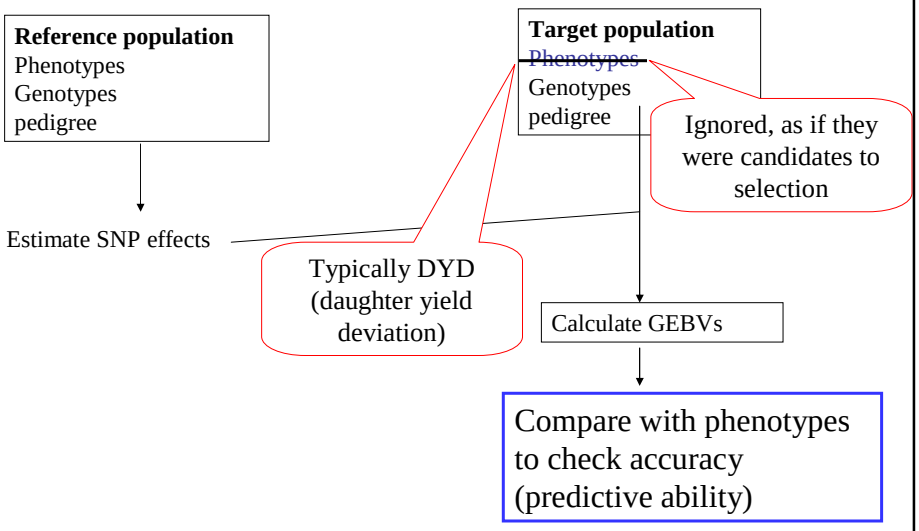
λ and α are chosen to give best results in a cross-validation system

65

Some results

66

Cross-validation analysis



67

What priors?

- The advantage of assuming a normal distribution is that estimators are linear (BLUP_SNP) and easy to compute and extend (e.g., to SingleStep)
- The rest can be computed by MCMC (BayesX) and sometimes other algorithms
  - VanRaden's nonlinear, Lasso, EN, etc
- Why bother about non-normal distributions?
  - Simulations show higher accuracies using non-normal distributions
  - This is rarely the case in practice: some examples

69

2009 JDS (USA)

Fat and protein contents

Table 2. Coefficients of determination ($R^2 \times 100$) for 2008 daughter deviations with 2003 predictions

Trait	Traditional parent average	Genomic prediction		
		Linear	Nonlinear	Difference[1]
Net merit	11	28	28	0
Milk yield	28	47	49	2
Fat yield	15	42	44	2
Protein yield	27	47	47	0
Fat percentage	25	55	63	8
Protein percentage	28	51	58	7
Productive life	17	26	27	1
SCS	23	37	38	1
Daughter pregnancy rate	20	30	29	−1
Sire calving ease	17	21	22	1
Daughter calving ease	14	22	22	0
Final score	23	35	36	1
Stature	27	49	50	1
Strength	16	33	34	1
Body depth	17	36	37	1
Dairy form	9	29	28	−1
Foot angle	13	23	21	−2
Rear legs (side view)	10	27	27	0
Rear legs (rear view)	11	21	19	−2
Rump angle	20	44	43	−1
Rump width	19	38	36	−2
Fore udder	17	39	40	1
Rear udder height	20	35	36	1
Udder depth	18	47	46	−1
Udder cleft	18	30	30	0
Front teat placement	22	41	42	1
Teat length	12	35	34	−1
All	19	36	37	1

[1]Nonlinear minus linear genomic prediction.

70

Hayes et al. 2009 JDS (Australia)

Genomic BLUP

Table 1. Reliability of genomic breeding values calculated at time of birth for ...metic Australia's 2003 progeny test team with 2 genomic selection methods, BLUP and a Bayesian method (BayesA)

Trait[1]	Records in reference population	Number of SNP used	Sire pathway EBV	BLUP	BayesA
ASI	637	3,889	0.38	0.44	0.48
APR	635	3,414	0.35	0.53	0.55
Protein yield	637	4,055	0.28	0.45	0.48
Protein %	637	4,369	0.20	0.29	0.36
Fertility	332	3,090	0.16	0.18	0.14

[1]ASI = Australian Selection Index; APR = Australian Profit Ranking.

71

Su et al. 2010 JDS (Denmark)

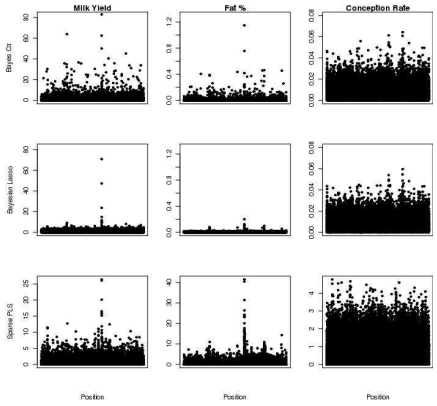
Table 3. Reliability of EBV (REL_{EBV}) and squared correlation between genomic EBV and EBV ($r^2_{GEBV, EBV}$) for bulls in each test dataset of a cross validation[1]

			$r^2_{GEBV,EBV}$				
Trait	Dataset	REL_{EBV}	Mixture[2] $\pi_1 = 5\%$	Mixture $\pi_1 = 10\%$	Mixture $\pi_1 = 20\%$	Mixture $\pi_1 = 50\%$	Common[3]
Fertility	Test1	70.0	0.275	0.304	0.314	0.342	0.362
	Test2	68.2	0.348	0.378	0.389	0.388	0.399
	Test3	68.9	0.300	0.340	0.359	0.374	0.376
	Test4	67.4	0.416	0.405	0.434	0.441	0.444
	Test5	63.0	0.419	0.444	0.438	0.495	0.493
	Pooled	67.6	0.347	0.370	0.384	0.407	0.412
Protein	Test1	93.8	0.284	0.315	0.357	0.393	0.401
	Test2	93.2	0.304	0.363	0.405	0.371	0.413
	Test3	93.6	0.283	0.331	0.354	0.374	0.375
	Test4	93.1	0.283	0.352	0.392	0.407	0.438
	Test5	92.0	0.233	0.309	0.368	0.410	0.420
	Pooled	93.1	0.279	0.337	0.378	0.394	0.412
Udder health	Test1	76.1	0.279	0.301	0.330	0.332	0.351
	Test2	75.7	0.275	0.317	0.369	0.377	0.410
	Test3	76.5	0.415	0.448	0.481	0.498	0.505
	Test4	75.3	0.372	0.395	0.395	0.421	0.431
	Test5	71.4	0.322	0.381	0.433	0.464	0.466
	Pooled	75.0	0.338	0.373	0.404	0.417	0.435
Fat percentage	Test1	93.9	0.681	0.709	0.725	0.711	0.716
	Test2	93.2	0.662	0.678	0.694	0.694	0.685
	Test3	93.5	0.709	0.729	0.748	0.741	0.751
	Test4	92.8	0.695	0.705	0.714	0.708	0.703
	Test5	92.0	0.591	0.611	0.622	0.632	0.640
	Pooled	93.1	0.670	0.688	0.702	0.698	0.700

3

C Colombani (accepted, JDS)

- Shape of effects



77

C Colombani (accepted, JDS)

- accuracy

Table 2. Correlations between observed DYD and predicted DYD on the validation data set provided by pedigree-based BLUP (BLUP), Genomic BLUP (GBLUP), PLS, Sparse PLS (sPLS), Bayesian Lasso and Bayes Cπ (Model 1) in Holstein

	BLUP	GBLUP	PLS	sPLS	Bayesian Lasso	Bayes Cπ
Milk Yield	0.38	0.56	0.53	0.48	0.56	0.57
Fat %	0.44	0.72	0.70	0.66	0.79	0.80
Conception Rate	0.28	0.35	0.33	0.29	0.34	0.34

• in spite of estimating different SNP effects, BL and BayesCPi agree…
  •this is because we finally work with *sums* of SNP effects

78