VCE with genomic models

Hongding Gao, Ismo Strandén

MiX99 course on genomic prediction

COURSE DAY 2, March 10th, 2026

Contents

- Introduction
- VCE in GBLUP and SNPBLUP
- VCE in single-step models
- A case study

Introduction

- Genomic selection models require accurate estimates of (co)variance components
- Restricted maximum likelihood (REML) serves as a vital method in variance component estimation (VCE)
- Two most widely used methods are expectation-maximization REML (EM-REML) and average information REML (AI-REML)
- EM-REML uses the first derivatives of the REML log-likelihood and is slow
- AI-REML uses both the first and second derivatives of the REML log-likelihood and is fast

Luke

What method MiX99 uses for VCE

- Monte Carlo expectation maximization restricted maximum likelihood (MC EM-REML) (in the current version you are using)

- Monte Carlo average information restricted maximum likelihood (MC AI-REML) (Now in beta version and will be added to the official version soon)

Luke

Why MiX99 uses MC-based REML?

- The analytical REML-based methods usually need factorization/inversion of coefficient matrix of the MME (to form the trace terms)

- Direct inversion or factorization of a dense and large coefficient matrix is computationally challenging/infeasible in genomic models

Why MiX99 uses MC-based REML?

- Monte Carlo (MC) algorithm can approximate these trace terms without inverting the coefficient matrix

  (Garcia-Cortes et al. (1992); Matilainen et al., (2012))

$$\hat{\sigma}_u^2 = \frac{\hat{\mathbf{u}}'\mathbf{G}^{-1}\hat{\mathbf{u}} + \mathrm{tr}(\mathbf{G}^{-1}\mathbf{C}^{uu})}{q} \quad \Rightarrow \quad \hat{\sigma}_u^2 = \frac{\hat{\mathbf{u}}'\mathbf{G}^{-1}\hat{\mathbf{u}} + \frac{1}{S}\sum_{h=1}^{S}(\tilde{\mathbf{u}}^h - \hat{\mathbf{u}}^h)'\mathbf{G}^{-1}(\tilde{\mathbf{u}}^h - \hat{\mathbf{u}}^h)}{q}$$

- Preconditioned conjugate gradient (PCG) solver and iteration on data can be used for solving

Luke

Why EM-REML?

- AI-REML Pros:
  - Faster convergence
  - Standard errors
- AI-REML Cons:
  - Potential numerical instability (low heritability traits, marker-based model)
  - Sensitivity to starting values

Why EM-REML?

- EM-REML Pros:
  - Numerical stability
  - Robustness to starting values
- EM-REML Cons:
  - Slower

MC EM-REML

- VCE for large-scale datasets
- VCE for complex models (e.g. multi-trait random regression/reaction norm genomic model)

An example (PBLUP)

- 13 trait
- 65K animals with data
- 170K animals in pedigree
- 2M equations
- 182 VCs
- Analytical REML: slow/😭
- MC EM-REML: ✌️

```
MODEL
   drpCW      =  MEAN  G(NAVID) ! WEIGHT=ercCW
   drpCC      =  MEAN  G(NAVID) ! WEIGHT=ercCC
   drpCF      =  MEAN  G(NAVID) ! WEIGHT=ercCF
   drpSTA     =  MEAN  G(NAVID) ! WEIGHT=ercSTA
   drpmbw     =  MEAN  G(NAVID) ! WEIGHT=ercmbw
   drpmilk    =  MEAN  G(NAVID) ! WEIGHT=ercmilk
   drpprot     =  MEAN  G(NAVID) ! WEIGHT=ercprot
   drpfat     =  MEAN  G(NAVID) ! WEIGHT=ercfat
   drpbdg     =  MEAN  G(NAVID) ! WEIGHT=ercbdg
   drpbcc     =  MEAN  G(NAVID) ! WEIGHT=ercbcc
   drpbcf     =  MEAN  G(NAVID) ! WEIGHT=ercbcf
   drpdmi     =  MEAN  G(NAVID) ! WEIGHT=ercdmi
   drpgain    =  MEAN  G(NAVID) ! WEIGHT=ercgain
```

Case from Ayo Salaudeen

Instruction

- MiX99 usually reads 2 input files to run VCE:
  - ✓ .clm file as input for mix99i
  - ✓ Solver option file (.slv file) as input for mix99s

```
./mix99i gblup.clm > mix99i.log && ./mix99s < reml.slv > mix99s.log
```

GBLUP example: .clm using COVFILE

```
DATAFILE pheno.dat

INTEGER id mu

REAL y1 y2

MISSING -999

PARFILE gs.par        # starting VCs

COVFILE id  coo.txt   # inverse of GRM

RANDOM   id

PRECON b b

MODEL
y1 = mu id
y2 = mu id
```

```
1 1 1 1.0
1 2 1 0.0
1 2 2 1.0
2 1 1 1.0
2 2 1 0.0
2 2 2 1.0
```

```
1   1   939.8462762552722
2   1   -27.508713276256078
3   1   -101.62198185147794
4   1   74.10967581050878
5   1   319.42844289711104
6   1   -572.1361484880038
7   1   288.45297636534275
8   1   798.6508187213568
9   1   -294.6611127344581
10  1   624.6491346797333
11  1   1079.223427291016
12  1   215.14811871485205
13  1   37.937010403789024
14  1   582.9859992816475
15  1   -39.24580045165765
16  1   565.0442445875196
17  1   -616.8148749159013
```

PARFILE IDENTITY

GBLUP example: reml.slv

```
# RAM: RAM demand: H=high, M=medium, L=low
   H
# STOP: Maximum_number_of_iterations, Stopping_criterion, Criterion (A/R/D)
   5000          1.0e-5                d       f
# RESID: Calculate residuals? (Y/N)
   N
# VALID: N=no, P=prediction, S=sum of effects, Y=YD, D=DYD, I=IDD, G=generate
   N
# HETVAR: (N)o HV, (S)tart HV, (C)ontinue HV, (F)inale, (E)stimation VC by EM
   E
   # STOPE maximum number of EM steps, samples/step, convergenc crit.
   2000              5                   1.0e-9
   # RANDG random number generator
   D
   # mix99i PATH
  /home/bin/
# TYPSOL: Solution files? (N)o, (Y)es, (A)itken, (H)alf-Chebychev
   Y
```

Max REML round

Convergance criteria for PCG solver

MC sample size

Convergance criteria for MC EM-REML

GBLUP example: reml.slv (using multiple cores)

```
# RAM: RAM demand: H=high, M=medium, L=low
   H  nt 10
# STOP: Maximum_number_of_iterations, Stopping_criterion, Criterion (A/R/D)
   5000           1.0e-5               d       f
# RESID: Calculate residuals? (Y/N)
   N
# VALID: N=no, P=prediction, S=sum of effects, Y=YD, D=DYD, I=IDD, G=generate
   N
# HETVAR: (N)o HV, (S)tart HV, (C)ontinue HV, (F)inale, (E)stimation VC by EM
   E
   # STOPE maximum number of EM steps, samples/step, convergenc crit.
   2000               5                   1.0e-9
   # RANDG random number generator
   D
   # mix99i PATH
  /home/bin/
# TYPSOL: Solution files? (N)o, (Y)es, (A)itken, (H)alf-Chebychev
   Y
```

SNPBLUP example: .clm using REGMATRIX


```
DATAFILE pheno.dat

INTEGER idcode mean1
REAL y1 y2

MISSING -999999.
DATASORT PEDIGREECODE=idcode

PARFILE parin          # staring values of residual VC

REGMATRIX RANDOM SNP ID=1 REGINDEX=idcode FIRST=2 LAST=1001
REGFILE marker.dat       # your SNP file
REGPARFILE gs_gen.par1   # staring values of SNP VC

PRECON b

MODEL
y1 = mean1
y2 = mean1
```

SNPBLUP example: reml.slv

Keep the regression design matrix in RAM

```
# RAM: RAM demand: H=high, M=medium, L=low
   H    RDX
# STOP: Maximum_number_of_iterations, Stopping_criterion, Criterion (A/R/D)
   5000           1.0e-5                d       f
# RESID: Calculate residuals? (Y/N)
   N
# VALID: N=no, P=prediction, S=sum of effects, Y=YD, D=DYD, I=IDD, G=generate
   N
# HETVAR: (N)o HV, (S)tart HV, (C)ontinue HV, (F)inale, (E)stimation VC by EM
   E
   # STOPE maximum number of EM steps, samples/step, convergenc crit.
           1000              5                          1.0e-11
   # RANDG random number generator
           D
   # mix99i PATH
  /home/bin/
# TYPSOL: Solution files? (N)o, (Y)es, (A)itken, (H)alf-Chebychev
   Y
```

ssGBLUP example: .clm

```
DATAFILE  pheno.txt
MISSING   -9999

INTEGER  ID gener sukup
REAL     y1 y2

DATASORT  PEDIGREECODE=ID

SSGBLUP LOWER iGL10K_w10.dat

PEDFILE  sim.ped
PEDIGREE ID am

INBRFILE   sim.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE IDENTITY     # VC starting values

MODEL
  y1 = gener ID
```

ssGBLUP example: reml.slv

```
# RAM: RAM demand: H=high, M=medium, L=low
   H   nt 10
# STOP: Maximum_number_of_iterations, Stopping_criterion, Criterion (A/R/D)
   5000          1.0e-5                d       f
# RESID: Calculate residuals? (Y/N)
   N
# VALID: N=no, P=prediction, S=sum of effects, Y=YD, D=DYD, I=IDD, G=generate
   N
# HETVAR: (N)o HV, (S)tart HV, (C)ontinue HV, (F)inale, (E)stimation VC by EM
   E
   # REML rounds, nSamples, stopCritVCE, stopBLUP_samples, nLastSamples
         5000      5        1.0e-10       0                 100
   # RANDG random number generator
          D
   # mix99i PATH
  /home/bin/
# TYPSOL: Solution files? (N)o, (Y)es, (A)itken, (H)alf-Chebychev
   Y
```

Stopping criterion of BLUP for MC sample, if 0 here, meaning using the same value as that for solving the real data.

Sample size for the last REML round, which is used for calculation of SE. This number needs to be larger than the total number of VCs in your model.

ssGTABLUP example: .clm

```
DATAFILE  pheno.txt
MISSING   -9999

INTEGER  ID gener sukup
REAL     y1 y2

DATASORT  PEDIGREECODE=ID

SNPMATRIX USE=PACK  FIRST=2 LAST=9001 CENTER=p
SNPFILE    SNP_10K.txt
CENTERFILE base_af_2col.txt
ICFILE     iC_w10_10K.bin
IA22FILE   PEDIGREE

PEDFILE  sim.ped
PEDIGREE ID am

INBRFILE   sim.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE IDENTITY

MODEL
  y1 = gener ID
```

ssGTABLUP example: reml.slv


```
# RAM: RAM demand: H=high, M=medium, L=low
   H   nt 10
# STOP: Maximum_number_of_iterations, Stopping_criterion, Criterion (A/R/D)
   5000           1.0e-5                d       f
# RESID: Calculate residuals? (Y/N)
   N
# VALID: N=no, P=prediction, S=sum of effects, Y=YD, D=DYD, I=IDD, G=generate
   N
# HETVAR: (N)o HV, (S)tart HV, (C)ontinue HV, (F)inale, (E)stimation VC by EM
   E
   # REML rounds, nSamples, stopCritVCE, stopBLUP_samples, nLastSamples
         5000      5        1.0e-10       0                 100
   # RANDG random number generator
          D
   # mix99i PATH
  /home/bin/
# TYPSOL: Solution files? (N)o, (Y)es, (A)itken, (H)alf-Chebychev
   Y
```

ssSNPBLUP example: .clm

```
DATAFILE  pheno.txt
MISSING   -9999

INTEGER  ID gener sukup
REAL     y1 y2

DATASORT  PEDIGREECODE=ID

SSSNPBLUP  GTA 0.10 SNP_10K.txt
SNPMATRIX  USE=PACK FIRST=2 LAST=9001 CENTER=p SCALE=p
CENTERFILE base_af_2col.txt
IA22FILE   PEDIGREE

PEDFILE  sim.ped
PEDIGREE ID am

INBRFILE   sim.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE IDENTITY

MODEL
  y1 = gener ID
```

ssSNPBLUP example: reml.slv

Second-level preconditioner

```
# RAM: RAM demand: H=high, M=medium, L=low
   H    nt 10  sp 100
# STOP: Maximum_number_of_iterations, Stopping_criterion, Criterion (A/R/D)
   5000           1.0e-5                d       f
# RESID: Calculate residuals? (Y/N)
   N
# VALID: N=no, P=prediction, S=sum of effects, Y=YD, D=DYD, I=IDD, G=generate
   N
# HETVAR: (N)o HV, (S)tart HV, (C)ontinue HV, (F)inale, (E)stimation VC by EM
   E
   # REML rounds, nSamples, stopCritVCE, stopBLUP_samples, nLastSamples
         5000      5        1.0e-10       0                 100
   # RANDG random number generator
          D
   # mix99i PATH
  /home/bin/
# TYPSOL: Solution files? (N)o, (Y)es, (A)itken, (H)alf-Chebychev
   Y
```

Convergence

- Default convergence criteria (1.0e-5) for solver is suitable for many cases
- Suitable convergence criteria for REML are e.g., 1.0e-8, 1.0e-9 (default), 1.0e-10
- Commonly used MC sample size: 1, 2, 3, 4, 5
- It is a good practice to check the convergence via plot the REMLlog

Luke

A case study

- Estimate VC using difference single-step models
- Using MC EM-REML
- Scenarios: different genotyped individuals
- ssGBLUP vs. ssGTABLUP vs. ssSNPBLUP
- Number of EM-REML iterations, peak RAM use, and total computing time

Simulated data

- 44,280 individuals with single record
- Either 10,000, 20,000, or 30,000 youngest animals were genotyped (9,000 markers)
- Pedigree: 44,280

Table 1. Estimated variance components (±SE), number of REML iterations (N), REML computing time (Time REML, h), total computing time including preprocessing (Time Total, h), and peak RAM memory need (Peak RAM, GB) by model

Geno. size	Model	$\widehat{\sigma_u^2}$	$\widehat{\sigma_e^2}$	N	Time REML	Total time	Peak RAM
10k	ssGBLUP	34.5±0.91	52.6±0.55	302	1:38	1:40	3.3
	ssGTABLUP	34.6±0.92	52.6±0.53	284	5:16	5:17	2.3
	ssSNPBLUP	34.3±0.93	52.6±0.54	285	18:00	18:00	1.7
20k	ssGBLUP	34.8±0.96	53.3±0.52	268	3:49	3:52	7.1
	ssGTABLUP	34.8±0.86	53.3±0.55	300	10:32	10:33	2.9
	ssSNPBLUP	34.8±0.84	53.3±0.51	288	41:45	41:45	2.5
30k	ssGBLUP	37.2±0.94	53.2±0.44	370	10:27	10:34	15.5
	ssGTABLUP	37.2±0.94	53.2±0.49	304	15:13	15:14	3.6
	ssSNPBLUP	37.3±0.88	53.2±0.45	290	70:37	70:37	3.2

Results

- All single-step models produced similar VC estimates
- The standard ssGBLUP model was always fastest with the used small data but showed the steepest increase in computing time as the number of genotyped increased
- ssGBLUP had the highest RAM usage
- ssGTABLUP and ssSNPBLUP had similar increase in computing time with the increase in the number of genotyped individuals
- RAM usage of ssGTABLUP and ssSNPBLUP increased only moderately compared to ssGBLUP

Luke

Conclusions

- ssGTABLUP becomes computationally more efficient than ssGBLUP for VCE when the number of genotyped individuals is larger

- For very large number of genotyped individuals, ssSNPBLUP may be computationally even more efficient than either ssGBLUP or ssGTABLUP for VCE

Thank you!