

C++ Programming for Animal Breeding

Rohan L. Fernando
Department of Animal Science
Iowa State University

Stephen D. Kachman
Department of Statistics

UNIVERSITY OF
Nebraska
Lincoln

- Normal equations
- Recoder class
- Random Effects
- Maternal Effects and Multiple Trait Models
- Matvec classes for GLMM

Construction of Normal Equations

- Normal equations
 - Left and Right hand sides
- Ones and zeros
- One-way model
- Two-way model
- Covariates

Normal equations

$$y = X\beta + e$$

- y : $n \times 1$ vector
- X : $n \times p$ matrix of rank r
- β : $p \times 1$ vector
- $e \sim N(\mathbf{0}, I\sigma^2)$: $n \times 1$ vector

$$(\mathbf{X}'\mathbf{X})\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$$

- Left hand side (**LHS**) of the normal equations

$$\mathbf{X}'\mathbf{X}$$

- p^2 elements

$$LHS_{ij} = \sum_{k=1}^n x_{ki}x_{kj}$$

- np^2 multiplications

- Right hand side (**RHS**) of the normal equations

- np multiplications

One-way Model

$$y_{ij} = \mu + \alpha_i + e_{ij}$$

Trt	y
1	1.0
1	1.5
2	2.0
2	2.5

$$\mathbf{X} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned} \mathbf{LHS}_{23} &= 1 * 0 + 1 * 0 + 0 * 1 + 0 * 1 \\ &= 0 \end{aligned}$$

- Perhaps there is a better way!

- $LHS_{ij} = \sum_{k=1}^n x_{ki}x_{kj}$
- Think in terms of observations

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ \vdots \\ \mathbf{x}'_n \end{pmatrix}$$

- LHS

$$LHS = \sum_{i=1}^n \mathbf{x}_i \times \mathbf{x}'_i$$

- Series of updates

$$LHS_1 = \mathbf{x}_1 \mathbf{x}'_1$$

$$LHS_2 = LHS_1 + \mathbf{x}_2 \mathbf{x}'_2$$

$$\vdots$$

$$LHS = LHS_n = LHS_{n-1} + \mathbf{x}_n \mathbf{x}'_n$$

$$RHS = RHS_n = RHS_{n-1} + \mathbf{x}_n \mathbf{y}_n$$

- Individual update

$$\begin{aligned} \mathbf{x}_1 \mathbf{x}'_1 &= \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} (1 \quad 1 \quad 0) \\ &= \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

- Non-zero elements

Row	Column	Value
1	1	1
1	Trt+1	1
Trt+1	1	1
Trt+1	Trt+1	1
1	<i>RHS</i>	y_i
Trt+1	<i>RHS</i>	y_i

- C++

– Zero indexed

$$\begin{pmatrix} \{0, 0\} & \{0, 1\} & \{0, 2\} \\ \{1, 0\} & \{1, 1\} & \{1, 2\} \\ \{2, 0\} & \{2, 1\} & \{2, 2\} \end{pmatrix}$$

Header

```
#include <fstream>      //Input/Output header files
#include <iostream>
#include <iomanip>

#include <matvec/doublematrix.h> // doubleMatrix class
#include <matvec/vector.h>       // vectors class
#include <matvec/session.h>     // matvec housekeeping

using namespace std;          // use standard C++ functions
```

Setup

```
int main() {
    try {
        matvec::SESSION.initialize("matvec_trash");
        matvec::doubleMatrix mme; // LHS
        matvec::Vector<double> rhs, sol;
        unsigned J[2];
        char* filename = "Data/oneway.dat";
        ifstream datafile;
        datafile.open(filename);
        if(!datafile) {
            cerr << "Couldn't open " << filename << endl;
            exit (-1);
        }
    }
}
```

```
int n = 3;  
mme.resize(n,n,0.0);  
rhs.resize(n,0.0);  
unsigned trt;
```

Build the normal equations

```
while (datafile >> trt >> y){
    J[0] = 0;    //Location of nonzeros
    J[1] = trt;
    for (unsigned i=0; i<2; i++){
        unsigned ii = J[i];
        rhs[ii] += y;
        for (unsigned j = 0; j<2; j++){
            unsigned jj = J[j];
            mme[ii][jj] += 1;
        }
    }
}
```

Write results

```
cout <<"\n The lhs ..... \n\n";
for (unsigned i = 0;i<n;i++){
    for (unsigned j = 0;j<n;j++){
        cout << setw(5) << mme[i][j] <<" ";
    }
    cout << endl;
}
cout << endl;
cout <<" The rhs ..... \n\n";
```

Results

The lhs

4	2	2
2	2	0
2	0	2

The rhs

7
2.5
4.5

Two-way Model

$$y_{ijk} = \mu + \alpha_i + \beta_j + e_{ijk}$$

TrtA	TrtB	Y
1	1	7.0
1	1	8.0
1	2	15.0
1	2	16.7
2	1	29.0
2	2	30.1

$$\beta = \begin{matrix} & 0 & \\ & 1 & \\ & 2 & \\ 2 + 1 = 3 & & \\ 2 + 2 = 4 & & \end{matrix} \begin{pmatrix} \mu \\ \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{pmatrix}$$

- μ at position 0
- α_i at position i
- β_j at position $j + 2$

$$\begin{aligned} \mathbf{x}_1 \mathbf{x}'_1 &= (1 \quad 1 \quad 0 \quad 1 \quad 0) \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{aligned}$$

Changes

```
matvec::SESSION.initialize("matvec_trash");
matvec::doubleMatrix mme;
matvec::Vector<double> rhs, sol;
unsigned J[3];
char* filename = "Data/twoway.datc";
ifstream datafile;
datafile.open(filename);
if(!datafile) {
    cerr << "Couldn't open " << filename << endl;
    exit (-1);
}
```

```
mme.resize(n,n,0.0);
rhs.resize(n,0.0);
unsigned trtA, trtB;
double y;
while (datafile >> trtA >> trtB >> y){
    J[0] = 0;
    J[1] = trtA;
    J[2] = 2 + trtB;
    for (unsigned i=0; i<3; i++){
        unsigned ii = J[i];
        rhs[ii] += y;
        for (unsigned j = 0;j<3;j++){
            unsigned jj = J[j];
            mme[ii][jj] += 1;
        }
    }
}
```

Results

The lhs

6	4	2	3	3
4	4	0	2	2
2	0	2	1	1
3	2	1	3	0
3	2	1	0	3

The rhs

105.8
46.7
59.1
44

Models with Covariates

$$y_{ij} = \mu + \alpha_i + b \text{cov}_{ij} + e_{ij}$$

TrtA	X	Y
1	1.0	1.0
1	1.5	2.0
2	2.0	3.0
2	2.5	4.0

$$\beta = \begin{matrix} & 0 & \\ & 1 & \\ & 2 & \\ 2 + 1 = 3 & \left(\begin{matrix} \mu \\ \alpha_1 \\ \alpha_2 \\ b \end{matrix} \right) \end{matrix}$$

- μ at position 0
- α_i at position i
- b at position $1 + 2$

$$\begin{aligned}
 \mathbf{x}_2 \mathbf{x}'_2 &= (1 \quad 1 \quad 0 \quad 1.5) \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1.5 \end{pmatrix} \\
 &= \begin{pmatrix} 1 & 1 & 0 & 1.5 \\ 1 & 1 & 0 & 1.5 \\ 1 & 1 & 0 & 1.5 \\ 0 & 0 & 0 & 0 \\ 1.5 & 1.5 & 0 & 1.5 * 1.5 = 2.25 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1.5 \end{pmatrix}
 \end{aligned}$$

Changes

- Multiplier vector V

$$v = (1 \quad 1 \quad cov_i)$$

- Position vector $J[2]=2+1$
- dimension n
- `rhs[ii]+=y*V[i]`
- `mme[ii][jj] += V[i]*V[j];`

ANCOVA

```
#include <fstream>
#include <iostream>
#include <iomanip>

#include <matvec/doublematrix.h>
#include <matvec/vector.h>
#include <matvec/session.h>

using namespace std;

int main() {
    try {
        matvec::SESSION.initialize("matvec_trash");
        matvec::doubleMatrix mme;
```

```
matvec::Vector<double> rhs, sol;
unsigned J[3];
double V[3];
char* filename = "Data/ancova.dat";
ifstream datafile;
datafile.open(filename);
if(!datafile) {
    cerr << "Couldn't open " << filename << endl;
    exit (-1);
}
int p = 4;
mme.resize(p,p,0.0);
rhs.resize(p,0.0);
unsigned trtA;
double y, cov;
V[0] = 1.0;
V[1] = 1.0;
while (datafile >> trtA >> cov >> y){
```

```

J[0] = 0;
J[1] = trtA;
J[2] = 3;
V[2] = cov;
for (unsigned i=0; i<3; i++){
    unsigned ii = J[i];
    rhs[ii] += y*V[i];;
    for (unsigned j = 0;j<3;j++){
        unsigned jj = J[j];
        mme[ii][jj] += V[i]*V[j];;
    }
}
}
cout <<"\n The lhs ..... \n\n";
for (unsigned i = 0;i<p;i++){
    for (unsigned j = 0;j<p;j++){
        cout << setw(5) << mme[i][j] <<" ";
    }
}

```

```
                cout << endl;
            }
            cout << endl;
            cout <<" The rhs ..... \n\n";
            cout << rhs << endl;
        }
        catch (matvec::exception &ex) {
            cerr << ex.what() << "\n";
        }
        catch (...) {
            cerr << "other exceptions were caught\n";
        }
        return 0;
    }
}
```

A C++ class (Recorder)

- Recoding
- Templates and Classes
- Interactions

Recoding

- Assumed the levels for factor A were coded 1...a.
- However, this is rarely the case

A	B	Y
A1	B1	7.0
A1	B1	8.0
A1	B2	15.0
A1	B2	16.7
A2	B1	29.0
A2	B2	30.1

A	A level
A1	1
A1	1
A1	1
A1	1
A2	2
A2	2

- Build a map that takes a “label” and returns a value
- Start with an empty map $\mathcal{M} = \{\}$ with $n = 0$ elements
- Read a record from the data
 - Check to see if the current level α is in \mathcal{M}
 - If it's not
 - * assign it next value and add it the map

$$\mathcal{M} = \mathcal{M} \cup \{\alpha \Rightarrow n + 1\}$$

$$n = n + 1$$

A	map	count	A level
	$\{\}$	0	
A1	$\{\} + \{A1 = 0 + 1\}$	0+1	1
A1	$\{A1 = 1\}$	1	1
A1	$\{A1 = 1\}$	1	1
A1	$\{A1 = 1\}$	1	1
A2	$\{A1 = 1\} + \{A2 = 1 + 1\}$	1+1	2
A2	$\{A1 = 1, A2 = 2\}$	2	2
A1	$\{A1 = 1, A2 = 2\}$	2	1

Recorder class

```
Recorder<string> ARecorder;
```

- Recorder: Recorder class
- <string>: It's a Recorder class that works with strings
- ARecorder: ARecorder is an object of the Recorder class that works with strings

```
template <class T>
class Recorder : public map<T,unsigned> {
    unsigned count;
public:
    Recorder(void){count=0;}
    unsigned code(T s){
        typename map<T,unsigned>::iterator mapit = find(s);
        if(mapit == this->end()){
            (*this)[s] = ++count;
            return count;
        }
        else {
            return (*mapit).second;
        }
    }
}
```

```
void display_codes(ostream & os = cout){
    typename Recoder::iterator it;
    for (it=this->begin(); it!=this->end();it++){
        os << (*it).first << " " << (*it).second << endl;
    }
}
};
```

- Perhaps, we need to spend some time on this.

Building on existing classes

- Frequently someone has done much of the work for you
- Standard Template Library
 - map class
 - Key \Rightarrow Value

```
map<string, int> Months;  
Months["January"]=1;  
Months["June"]=6;  
cout << Months["January"] << endl;  
 $\mathcal{M} = \{January \Rightarrow 1, June \Rightarrow 6\}$ 
```

- size()

– Iterator

```
map<string,int>::iterator it;
for(it=Months.begin();it!=Months.end();it++){
    cout << it->first << "->" << it->second<<endl;
}
```

– find(*Key*)

- * pointer to element whose key is *Key*
- * or end() if not found

A simple example

```
#include <map>
int main() {
    unsigned Value=0;
    string Key;
    map<string, unsigned> mymap;
    map<string, unsigned>::iterator it;
    while(cin >> Key ){
        if(mymap.find(Key)==mymap.end()) mymap[Key]=++Value;
    }
    for(it=mymap.begin();it!=mymap.end();it++){
        cout << setw(9) << setiosflags(ios_base::left) << it->first <<" -> "
        << resetiosflags(ios_base::left) << setw(2)<< it->second << endl;
    }
}
```

January	April	->	4
February	August	->	8
March	December	->	12
April	February	->	2
May	January	->	1
June	July	->	7
July	June	->	6
August	March	->	3
September	May	->	5
October	November	->	11
November	October	->	10
December	September	->	9

- Return to the Recorder class
- Recorder can use a variety of Key types and is a descendant of `map`
- `count` will be a private variable
- Now we will start the public section and define a constructor which will initialize `count`

```
template <class T>
class Recorder : public map<T,unsigned> {
    unsigned count;
public:
    Recorder(void) {count=0;}
}
```

- Define a code function

The code function will return the value associated with key `s`. If key `s` is not in the map, then it will be added.

```
unsigned code(T s){
    typename map<T,unsigned>::iterator mapit = find(s);
    if(mapit == this->end()){
        (*this)[s] = ++count;
        return count;
    }
    else {
        return (*mapit).second;
    }
}
```

- Define a `display_codes` function

```
void display_codes(ostream & os = cout){
    typename Recoder::iterator it;
    for (it=this->begin(); it!=this->end();it++){
        os << (*it).first << " " << (*it).second << endl;
    }
}
};
```

The Program

- The Declarations

```
#include <fstream>
#include <iostream>
#include <iomanip>
...
#include <map>
#include <matvec/doublematrix.h>
#include <matvec/vector.h>
#include <matvec/session.h>
#include "util.h"

using namespace std;
```

```
int main() {
    try{

        matvec::SESSION.initialize("matvec_trash");
        Recorder<string> ARecorder;
        matvec::doubleMatrix mme;
        matvec::Vector<double> rhs, sol;
        char *filename      = "Data/twoway.dat";
        char* outfile_name = "Data/twoway.dat.coded";
        string AString, BString;
        double y;
        unsigned J[2], ALevel;
        {ifstream datafile;
            datafile.open(filename);
            if(!datafile) {
                cerr << "Couldn't open " << filename << endl;
                exit (-1);
            }
        }
    }
}
```

```
ofstream outfile;
  outfile.open(outfile_name,ios::trunc);
  if(!outfile) {
    cerr << "Couldn't open " << outfile_name << endl;
    exit (-1);
  }
  outfile.setf(ios::fixed);
```

- Recode and output the recoded values

```
while (datafile >> AString >> BString >> y){
    ALevel = ARecorder.code(AString);
    outfile << setw(5) << ALevel <<" " << y << endl;
}
datafile.close();
outfile.close();
}
```

- Dimensions of the normal equations are determined from the data.

```
{ifstream datafile;
    datafile.open(outfile_name);
    unsigned nlevelsA= ARecoder.size();
    unsigned n = nlevelsA + 1;
    mme.resize(n,n,0.0);
    rhs.resize(n,0.0);
    J[0] = 0;
    while (datafile >> ALevel >> y){
        J[1] = ALevel;
        for (unsigned i = 0;i<2;i++){
            unsigned ii = J[i];
            rhs[ii] += y;
            for (unsigned j = 0;j<2;j++){
                unsigned jj = J[j];
                mme[ii][jj] += 1;
            }
        }
    }
```

```
        }  
    }  
    for (unsigned i = 0;i<n;i++){  
        for (unsigned j = 0;j<n;j++){  
            cout << setw(5) << mme[i][j] <<" ";  
        }  
        cout << endl;  
    }  
    cout << endl;
```

- Solve the normal equations

```
        cout << rhs << endl;
        sol = mme.ginv0()*rhs;
        cout << sol << endl;
        ARecorder.display_codes();
        datafile.close();
    }
}
catch (matvec::exception &ex) {
    cerr << ex.what() << endl;
    exit(1);
}
catch (...) {
    cerr << "other exceptions were caught"<<endl;
    exit(1);
}
}
```

Results

6	2	4
2	2	0
4	0	4

105.8

59.1

46.7

13.7417

15.8083

-2.06667

A1 1

A2 2

Interactions

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + e_{ijk}$$

- A×B interaction term can be constructed by combining the A and B labels

A	B	AB
A1	B1	A1xB1
A1	B1	A1xB1
A1	B2	A1xB2
A1	B2	A1xB2
A2	B1	A2xB1
A2	B2	A2xB2

- Combine the Labels

```
AString+"x"+BString
```

- Code the levels

```
ABLevel = ABRecoder.code(AString+"x"+BString);
```

```
#include <fstream>
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <stdarg.h>
#include <stdlib.h>
#include <math.h>
#include <map>
#include <matvec/doublematrix.h>
#include <matvec/vector.h>
#include <matvec/session.h>
#include "util.h"
```

```
using namespace std;

int main() {
    try{

        matvec::SESSION.initialize("matvec_trash");
        Recoder<string> ARecoder;
        Recoder<string> BRecoder;
        Recoder<string> ABRecoder;
        matvec::doubleMatrix mme;
        matvec::Vector<double> rhs, sol;
        char *filename      = "Data/twoway.dat";
        char* outfile_name = "Data/twoway.dat.coded";
        string AString, BString;
        double y;
        unsigned J[4], ALevel, BLevel, ABLevel;
```

```
{ifstream datafile;
    datafile.open(filename);
    if(!datafile) {
        cerr << "Couldn't open " << filename << endl;
        exit (-1);
    }
    ofstream outfile;
    outfile.open(outfile_name,ios::trunc);
    if(!outfile) {
        cerr << "Couldn't open " << outfile_name << endl;
        exit (-1);
    }
}
```

```
        outfile.setf(ios::fixed);
while (datafile >> AString >> BString >> y){
    ALevel = ARecorder.code(AString);
    BLevel = BRecorder.code(BString);
    ABLevel = ABRecorder.code(AString+"x"+BString);
    outfile << setw(5) << ALevel << " " << BLevel<< " "
    << ABLevel<< " " << y << "\n";
}
datafile.close();
outfile.close();
}
```

```
{ifstream datafile;
    datafile.open(outfile_name);
    unsigned nlevelsA = ARecoder.size();
    unsigned nlevelsB = BRecoder.size();
    unsigned nlevelsAB = ABRecoder.size();
    unsigned n = 1 + nlevelsA + nlevelsB + nlevelsAB;
    mme.resize(n,n,0.0);
    rhs.resize(n,0.0);
    J[0] = 0;
    while (datafile >> ALevel >> BLevel >> ABLevel >> y){
        J[1] = ALevel;
        J[2] = nlevelsA + BLevel;
        J[3] = nlevelsA + nlevelsB + ABLevel;
```

```
for (unsigned i = 0;i<4;i++){
    unsigned ii = J[i];
    rhs[ii] += y;
    for (unsigned j = 0;j<4;j++){
        unsigned jj = J[j];
        mme[ii][jj] += 1;
    }
}
```

```
for (unsigned i = 0;i<n;i++){
    for (unsigned j = 0;j<n;j++){
        cout << setw(5) << mme[i][j] <<" ";
    }
    cout << endl;
}
cout << endl;
cout << rhs << endl;
sol = mme.ginv0()*rhs;
cout << sol << endl;
ARecorder.display_codes();
BRecorder.display_codes();
ABRecorder.display_codes();
datafile.close();
}
}
```

```
    catch (matvec::exception &ex) {
        cerr << ex.what() << "\n";
        exit(1);
    }
    catch (...) {
        cerr << "other exceptions were caught\n";
        exit(1);
    }
}
```

Random Effects

- Mixed Model Equations
- $D = I\sigma^2$
- A^{-1}
 - Pedigree
 - Inbreeding
 - Non-zero elements

Mixed Model Equations

$$y = X\beta + Zu + e$$

$$\begin{pmatrix} u \\ e \end{pmatrix} \sim N \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} D & 0 \\ 0 & R \end{pmatrix} \right]$$

$$\begin{pmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z + D^{-1} \end{pmatrix} \begin{pmatrix} \hat{\beta} \\ \hat{u} \end{pmatrix} = \begin{pmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{pmatrix}$$

- Which can be simplified if $\mathbf{R} = \mathbf{I}\sigma_0^2$

$$\begin{pmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} + \mathbf{D}^{-1}\sigma_0^2 \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{Z}'\mathbf{y} \end{pmatrix}$$

- Letting $\mathbf{W} = (\mathbf{X} \quad \mathbf{Z})$ and $\boldsymbol{\theta}' = (\boldsymbol{\beta}' \quad \mathbf{u}')$

$$\left[\mathbf{W}'\mathbf{W} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1}\sigma_0^2 \end{pmatrix} \right] \begin{pmatrix} \hat{\boldsymbol{\theta}} \end{pmatrix} = \mathbf{W}'\mathbf{y}$$

addGinv

$$\mathbf{y}_{ijk} = \mu + \alpha_i + b_j + e_{ijk}$$

$$\mathbf{b} \sim \text{N}(\mathbf{0}, \mathbf{I}\sigma_b^2)$$

$$\mathbf{e} \sim \text{N}(\mathbf{0}, \mathbf{I}\sigma_e^2)$$

$$\lambda = \sigma_e^2 / \sigma_b^2$$

$$i = 1 \dots a$$

$$j = 1 \dots b$$

$$k = 1 \dots n_{ij}$$

- Want to add b λ 's to the diagonal of the left hand sides starting at the `nlevelsA+1` diagonal element.
- Define a function which adds the value ratio to the `n` diagonal elements starting at the start element.

```
void addGinv(matvec::doubleMatrix& lhs, unsigned start,
            unsigned n, double ratio){
    for(unsigned i=0;i<=n;i++){
        unsigned ii=start+i;
        lhs[ii][ii]+=ratio;
    }
}
```

```

while (datafile >> ALevel >> BLevel >> ABLevel >> y){
    J[1] = ALevel;
    J[2] = nlevelsA + BLevel;
    J[3] = nlevelsA + nlevelsB + ABLevel;
    for (unsigned i = 0;i<4;i++){
        .....
    }
}
addGinv(mme,nlevelsA + nlevelsB+1,nlevelsAB,ratio);
for (unsigned i = 0;i<n;i++){
    for (unsigned j = 0;j<n;j++){
        cout << setw(5) << mme[i][j] <<" ";
    }
    cout << endl;
}
cout << endl;

```

Calculation A^{-1}

- Offspring i with Parents s and d
- s and d known

$$d = \frac{4}{2 - f_s - f_d}$$

- Only s known

$$d = \frac{4}{3 - f_s}$$

- Both unknown

$$d = 1$$

For known animals and typical elements

$$A^{ii}+ = [-1/2]^0 d = d$$

$$A^{is}+ = [-1/2]^1 d = \frac{-d}{2}$$

$$A^{id}+ = [-1/2]^1 d = \frac{-d}{2}$$

$$A^{ss}+ = [-1/2]^2 d = \frac{d}{4}$$

$$A^{sd}+ = [-1/2]^2 d = \frac{d}{4}$$

$$A^{dd}+ = [-1/2]^2 d = \frac{d}{4}$$

```
void Pedigree::addAinv(matvec::doubleMatrix& lhs, unsigned startRow,
                    unsigned startCol, double ratio){
    double q[3];
    double d,fs,fd;
    unsigned pos[3];
    vector<PNode*>::iterator it;
    for (it=pedVector.begin();it!=pedVector.end();it++){
        pos[0] = (*it)->sire;
        pos[1] = (*it)->dam;
        pos[2] = (*it)->ind;
        if((*it)->sire && (*it)->dam){
            q[0] = -0.5;
            q[1] = -0.5;
            q[2] = 1.0;
            fs = pedVector[pos[0]-1]->f;
            fd = pedVector[pos[1]-1]->f;
            d = 4.0/(2 - fs - fd);
        }
    }
}
```

```
else if((*it)->sire){
    q[0] = -0.5;
    q[1] = 0.0;
    q[2] = 1.0;
    fs = pedVector[pos[0]-1]->f;
    d = 4.0/(3-fs);
}
else if((*it)->dam){
    q[0] = 0.0;
    q[1] = -0.5;
    q[2] = 1.0;
    fd = pedVector[pos[1]-1]->f;
    d = 4.0/(3-fd);
}
```

```
else{  
    q[0] = 0.0;  
    q[1] = 0.0;  
    q[2] = 1.0;  
    d = 1.0;  
}
```

```
for (unsigned i=0;i<3;i++){
    if(pos[i]){
        unsigned ii = startRow + pos[i] - 1;
        for (unsigned j=0;j<3;j++){
            if(pos[j]) {
                unsigned jj = startCol + pos[j] - 1;
                lhs[ii][jj] += ratio*q[i]*d*q[j];
            }
        }
    }
}
}
```

Inbreeding

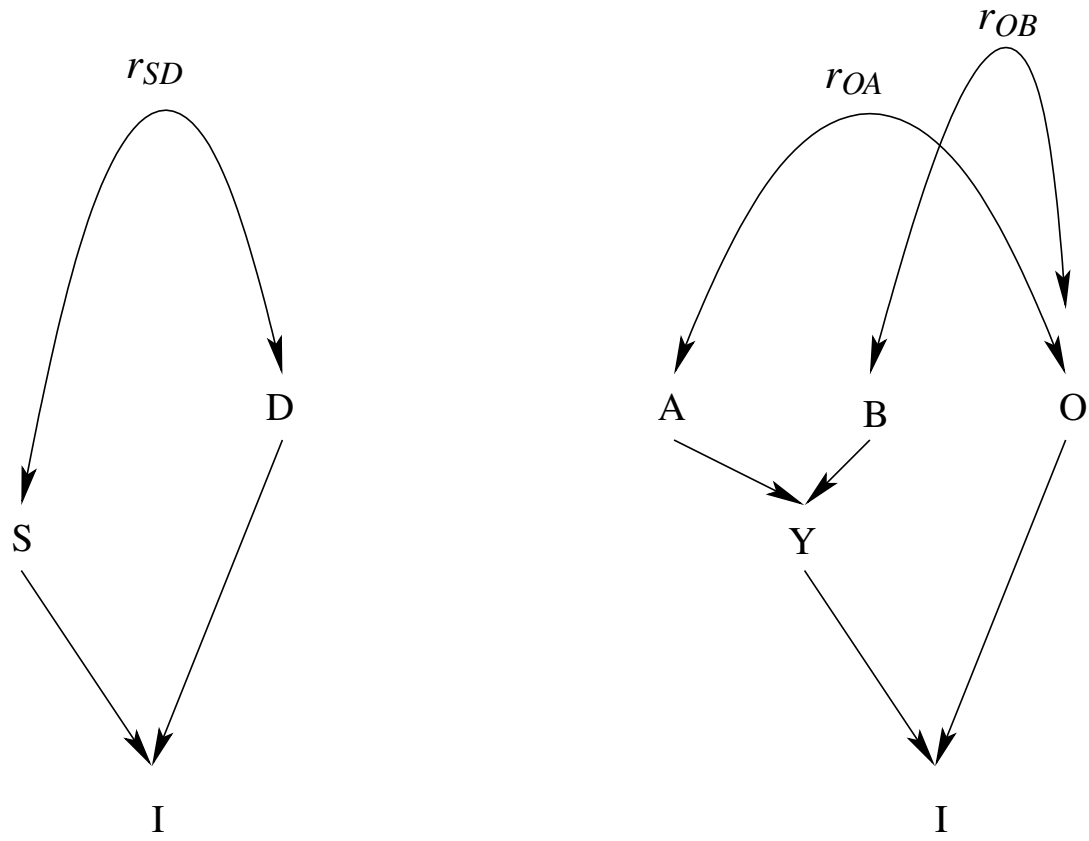
- Malecot's coefficient of coancestry between the sire and the dam

$$f_i = r_{sd}$$

- If r_{sd} is unknown, then

$$r_{sd} = .5[r_{oa} + r_{ob}]$$

where o is the oldest of the s and d , and a and b are the sire and dam of the youngest.



$C = \{r_{ij}\}$ Coancestry matrix

- Only need some of the values
- Symmetric
- `map<idx,double>`

```
class SparseCij {  
    map<const idx , double> C;  
public:  
    double retrieve_cij(const unsigned i, const unsigned j);  
    void    put_cij(const unsigned i, const unsigned j, double cij);  
    void    clear(void){C.clear();}  
};
```

```

class idx {
public:
    unsigned i,j;
    bool operator<(const idx y) const
    {
        if (i < y.i) { return 1; }
        if (i > y.i) { return 0; }
        else { if (j < y.j) { return 1; }
              else { return 0; }
            }
    }
};

```

$$\begin{pmatrix} & < & \\ < & ij & \neq \\ & \neq & \end{pmatrix}$$

- Retrieve an entry or return a -1
- Symmetric (Store Upper Triangle $\Rightarrow i \leq j$)

```
double SparseCij::retrieve_cij(const unsigned i, const unsigned j){
    idx ind;
    unsigned ii,jj;
    ii = i; jj = j;
    if (i>j) { ii = j; jj = i; }
    ind.i = ii;
    ind.j = jj;
    map<const idx , double>::iterator cit = C.find(ind);
    if (cit != C.end()) { return cit->second; }
    else { return -1.0; }
}
```

- $f_i = r_{sd}$

```
void Pedigree::calc_inbreeding(void){
    vector <PNode*>::iterator it;
    unsigned rec = 0, rec1 = 0, non_rec = 0;
    cout << "calculating inbreeding \n";
    for (it=pedVector.begin();it!=pedVector.end();it++){
        (*it)->f = get_rij((*it)->sire,(*it)->dam);
    }
}
```

- If the r_{sd} exists, then return the stored value
- Otherwise, calculate it using

$$r_{sd} = .5[r_{oa} + r_{ob}]$$

```
double Pedigree::get_rij(int i, int j){
    if (i==0||j==0){ return 0.0; }
    double x = SpCij.retrieve_cij(i,j);
    if(x != -1.0) { return x; }
    int old, young;
    if(i < j){ old = i; young = j; }
    else if(j < i){ old = j; young = i; }
    else{ double f = pedVector[i-1]->f;
        x = 0.5*(1 + f);
        SpCij.put_cij(i,j,x);
        return x; }
    int y_sire = pedVector[young-1]->sire;
    int y_dam = pedVector[young-1]->dam;
    x = (get_rij(old,y_sire)+get_rij(old,y_dam))/2.0;
    SpCij.put_cij(i,j,x);
    return x;
}
```

Pedigree Object

- Gather pedigree information
 - Vector of individual pedigree information
`vector <PNode*> pedVector;`
 - Individual ID map
`Recoder<string> coder;`
 - Coancestry matrix
`SparseCij SpCij;`
- Variety of pedigree operators

```
class Pedigree : public map<string,PNode*> {
public:
    unsigned COUNT;
    SparseCij SpCij;
    vector <PNode*> pedVector;
    Recoder<string> coder;
    void inputPed(char* fname);
    void displayPed(void);
    void generateEntriesforParents(void);
    void codePed();
    void code(PNode *ptr);
    void calc_inbreeding(void);
    void makePedVector(void);
    void fillCoder(void);
    double get_rij(int i, int j);
    void output(char* ped);
    void addAinv(matvec::doubleMatrix& lhs, ..., double ratio);
};
```

Pedigree Node

- Gather an individual's pedigree information
 - Alphanumeric id's
 - Coded id's
 - Inbreeding coefficients

```
class PNode {  
public:  
    int    ind, sire, dam;  
    double f;  
    string ind_str, sire_str, dam_str;  
    ...  
};
```

- ...
- Initialize with missing values

```
PNode(string indstr, string damstr, string sirestr){  
    ind = -1;  
    sire = -1;  
    dam = -1;  
    f = -1.0;  
    ind_str = indstr;  
    sire_str = sirestr;  
    dam_str = damstr;  
}  
};
```

Input Pedigree

- Read pedigree data into a pedVector
- Add missing parents to pedVector
- Work our way through pedVector to generate coded id's
 1. Code parents
 2. Code individual
- Calculate inbreeding
- Fill Pedigree Recoder with coded id's

Read pedigree data into a pedVector

```
void Pedigree::inputPed(char* fname){
    ...
    PNode *ptr;
    while (datafile>>indstr>>damstr>>sirestr){
        ptr = new PNode(indstr, damstr, sirestr);
        (*this)[indstr] = ptr;
    }
    datafile.close();
    generateEntriesforParents();
    codePed();
    makePedVector();
    calc_inbreeding();
    fillCoder();
}
```

Work our way through pedVector to generate coded id's

```
void Pedigree::codePed(){
    Pedigree::iterator it;
    COUNT = 0;
    unsigned rec = 0, rec1 = 0;
    for(it=begin();it!=end();it++){
        PNode *ptr =(*it).second;
        code(ptr);
    }
}
```

```

void Pedigree::code(PNode *ptr){
    if(ptr->ind != -1) { // already coded
        return;
    }
    .
    .
    .
    }
    else{
        PNode* sire_ptr = (*this)[ptr->sire_str];
        if (sire_ptr->ind == -1) { code(sire_ptr); }
        PNode* dam_ptr = (*this)[ptr->dam_str ];
        if ( dam_ptr->ind == -1) { code(dam_ptr ); }
        ptr->ind = ++COUNT;
        ptr->sire = sire_ptr->ind;
        ptr->dam  = dam_ptr->ind;
    }
}

```

Fill Pedigree Recoder with coded id's

- Fill pedVector young to old

```
void Pedigree::makePedVector(void){
    Pedigree::iterator it;
    pedVector.resize(size());
    for(it=begin();it!=end();it++){
        PNode *ptr = (*it).second;
        unsigned i = ptr->ind - 1;
        pedVector[i] = ptr;
    }
}
```

- Build pedigree from young to old

```
void Pedigree::fillCoder(void){
    vector<PNode *>::iterator it;
    for(it=pedVector.begin();it!=pedVector.end();it++)
        coder.code((*it)->ind_str);
}
```

Adding A^{-1} to the mixed model equations

- Read in Pedigree
- Recode data using the pedigree recoder for additive effects
- Build $W'W$ and $W'y$
- Add $A^{-1}\lambda$

```
#include <fstream>
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <stdarg.h>
#include <stdlib.h>
#include <math.h>
#include <map>
#include <matvec/doublematrix.h>
#include <matvec/vector.h>
#include <matvec/session.h>
#include "util.h"
#include "ped.h"
```

```
using namespace std;
```

```
int main() {
    try{
        matvec::SESSION.initialize("matvec_trash");
        Pedigree ped;
        ped.inputPed("Data/additive.ped");
        ped.displayPed();

        matvec::doubleMatrix mme;
        matvec::Vector<double> rhs, sol;
        char *filename      = "Data/additive.dat";
        char* outfile_name = "Data/additive.dat.coded";
        string strID;
        double y;
        unsigned J[2], additiveLevel;
        {ifstream datafile;
            datafile.open(filename);
            if(!datafile) {
                cerr << "Couldn't open " << filename << endl;
            }
        }
    }
}
```

```
        exit (-1);
    }
    ofstream outfile;
    outfile.open(outfile_name,ios::trunc);
    if(!outfile) {
        cerr << "Couldn't open " << outfile_name << endl;
        exit (-1);
    }
    outfile.setf(ios::fixed);
    while (datafile >> strID >> y){
        additiveLevel = ped.coder.code(strID);
        outfile << setw(5) << additiveLevel <<" " << y << "\n";
    }
    datafile.close();
    outfile.close();
}
{ifstream datafile;
    datafile.open(outfile_name);
```

```
unsigned nlevelsAdditive = ped.coder.size();
unsigned n = nlevelsAdditive + 1;
mme.resize(n,n,0.0);
rhs.resize(n,0.0);
J[0] = 0;
while (datafile >> additiveLevel >> y){
    J[1] = additiveLevel;
    for (unsigned i = 0;i<2;i++){
        unsigned ii = J[i];
        rhs[ii] += y;
        for (unsigned j = 0;j<2;j++){
            unsigned jj = J[j];
            mme[ii][jj] += 1;
        }
    }
}
ped.addAinv(mme,1,1,1.0);
for (unsigned i = 0;i<n;i++){
```

```
        for (unsigned j = 0;j<n;j++){
            cout << setw(5) << mme[i][j] <<" ";
        }
        cout << endl;
    }
    cout << endl;
    cout << rhs << endl;
    sol = mme.ginv0()*rhs;
    cout << sol << endl;
    ped.coder.display_codes();
    datafile.close();
}
}
catch (matvec::exception &ex) {
    cerr << ex.what() << "\n";
    exit(1);
}
catch (...) {
```

```
        cerr << "other exceptions were caught\n";  
        exit(1);  
    }  
}
```

Maternal Effects and Multi-trait Models

- Maternal effects
 - Correlated random effects
- Multiple Trait Models

Maternal effects

- Two additive genetic effects
 - Direct genetic effect $\mathbf{a} \sim N(\mathbf{0}, \mathbf{A}\sigma_a^2)$
 - Maternal genetic effect $\mathbf{m} \sim N(\mathbf{0}, \mathbf{A}\sigma_{am}^2)$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}_d\mathbf{a} + \mathbf{Z}_m\mathbf{m} + \mathbf{e}$$

- Correlated random effects $\text{cov}(\mathbf{a}, \mathbf{m}) = \mathbf{A}\sigma_{am}$

$$\mathbf{u} = \begin{pmatrix} \mathbf{a} \\ \mathbf{m} \end{pmatrix} \sim N[\mathbf{0}, \mathbf{D} = \mathbf{G} \otimes \mathbf{A}]$$

Mixed Model Equations

$$\left[\mathbf{W}'\mathbf{R}^{-1}\mathbf{W} + \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1} \end{pmatrix} \right] (\hat{\boldsymbol{\theta}}) = \mathbf{W}'\mathbf{R}^{-1}\mathbf{y}$$

$$\mathbf{D} = \mathbf{G} \otimes \mathbf{A}$$

$$\mathbf{D} = \begin{pmatrix} \mathbf{A}\sigma_a^2 & \mathbf{A}\sigma_{am} \\ \mathbf{A}\sigma_{am} & \mathbf{A}\sigma_m^2 \end{pmatrix}$$

$$\mathbf{D}^{-1} = \mathbf{G}^{-1} \otimes \mathbf{A}^{-1}$$

- One additional effect to $\mathbf{W}'\mathbf{R}^{-1}\mathbf{W}$ and $\mathbf{W}'\mathbf{R}^{-1}\mathbf{y}$
- Add $\mathbf{A}^{-1}\lambda_i$ to the direct and maternal sections of the left hand sides
- Add $\mathbf{A}^{-1}\lambda_{am}$ to direct and maternal intersection of the left hand sides

$$\begin{pmatrix} \lambda_a & \lambda_{am} \\ \lambda_{am} & \lambda_m \end{pmatrix} = \mathbf{G}^{-1}$$

Example

Data			Pedigree		
Animal	Dam	Y	Animal	Dam	Sire
CC2	AA1	5.7	AA1	0	0
DDD	CC2	7.0	DDD	CC2	BB1
CC1	AA3	5.0	AA2	0	0
			BB1	AA1	0
			BB2	0	AA2
			CC1	AA3	AA4
			CC2	AA1	BB2

- Input the pedigree
 - Pedigree Recoder
 - Calculate Inbreeding coefficients

Uncoded			Coded				
Animal	Dam	Sire	Individual	Ind.	Sire	Dam	F
AA1	0	0	AA1	1	0	0	0
DDD	CC2	BB1	AA2	2	0	0	0
AA2	0	0	AA3	3	0	0	0
BB1	AA1	0	AA4	4	0	0	0
BB2	0	AA2	BB1	5	0	1	0
CC1	AA3	AA4	BB2	6	2	0	0
CC2	AA1	BB2	CC1	7	4	3	0
			CC2	8	6	1	0
			DDD	9	5	8	0.125

Program

```
Pedigree ped;  
ped.inputPed("Data/additive.ped");  
ped.displayPed();
```

- Defines the recoder that will be used for both the direct and maternal effects.

```
ped.coder
```

Recode the data

```
while (datafile >> strId >> strIdDam >> y){
    directLevel = ped.coder.code(strId);
    maternalLevel = ped.coder.code(strIdDam);
    outfile << setw(5) << directLevel << " " <<
        maternalLevel << " " << y << "\n";
}
datafile.close();
outfile.close();
```

- Use pedigree coder for additive genetic effects.

Build $W'R^{-1}W$ and $W'R^{-1}y$

- Addition of $R^{-1} = I\sigma_0^2$

$$W'W = \sum_{i=1}^n w_i w_i'$$

$$W'R^{-1}W = \sum_{i=1}^n \frac{w_i w_i'}{\sigma_0^2}$$

$$W'R^{-1}y = \sum_{i=1}^n \frac{w_i y_i}{\sigma_0^2}$$

```
for(unsigned i=0;i<n;i++) maxw=max(maxw,labels[i].length());
mme.resize(n,n,0.0);
rhs.resize(n,0.0);
J[0] = 0;
while (datafile >> directLevel >> maternalLevel >> y){
    J[1] = directLevel;
    J[2] = nlevelsDirect + maternalLevel;
    for (unsigned i = 0;i<3;i++){
        unsigned ii = J[i];
        rhs[ii] += y*resVarInv;
        for (unsigned j = 0;j<3;j++){
            unsigned jj = J[j];
            mme[ii][jj] += 1.*resVarInv;
        }
    }
}
```

Addition of $G^{-1} \otimes A^{-1}$

```

unsigned startRow, startCol;
matvec::doubleMatrix G;
G.resize(2,2);
G(1,1) = 1.0; G(1,2) = 0.5; G(2,1) = 0.5; G(2,2)=2.0;
matvec::doubleMatrix Gi = G.inv();
startRow = 1;                startCol = 1;
ped.addAinv(mme,startRow,startCol,Gi(1,1));
startRow = 1;                startCol = nlevelsDirect + 1;
ped.addAinv(mme,startRow,startCol,Gi(1,2));
startRow = nlevelsDirect + 1; startCol = 1;
ped.addAinv(mme,startRow,startCol,Gi(2,1));
startRow = nlevelsDirect + 1; startCol = nlevelsDirect + 1;
ped.addAinv(mme,startRow,startCol,Gi(2,2));

```

Multiple Traits

- Still have a linear mixed model

$$\mathbf{y} = \mathbf{W}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}$$
$$\begin{pmatrix} \mathbf{u} \\ \mathbf{e} \end{pmatrix} \sim \text{N} \left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{pmatrix} \right]$$

- \mathbf{R} is no longer a diagonal matrix

- For trait t and individual i , the model equation is

$$y_{it} = \mathbf{w}'_{i1} \boldsymbol{\theta}_i + e_{it}$$

- Group the observations by individual into a vector

$$\begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} = \begin{pmatrix} \mathbf{w}'_{i1} & \mathbf{0} \\ \mathbf{0} & \mathbf{w}'_{i2} \end{pmatrix} \begin{pmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \end{pmatrix} + \begin{pmatrix} e_{i1} \\ e_{i2} \end{pmatrix}$$
$$\mathbf{y}_i = \mathbf{W}'_i \boldsymbol{\theta} + \mathbf{e}_i$$

Construction of $W'R^{-1}W$

$$W = \begin{pmatrix} W'_1 \\ \vdots \\ W'_N \end{pmatrix}$$

$$R^{-1} = \bigoplus R_i^{-1}$$

$$W'R^{-1}W = \sum_{i=1}^N W_i R_i^{-1} W'_i$$

$$\begin{aligned} W_i R_i^{-1} W'_i &= \begin{pmatrix} w_{i1} & 0 \\ 0 & w_{i2} \end{pmatrix} \begin{pmatrix} r^{11} & r^{12} \\ r^{21} & r^{22} \end{pmatrix} \begin{pmatrix} w'_{i1} & 0 \\ 0 & w'_{i2} \end{pmatrix} \\ &= \begin{pmatrix} w_{i1} r^{11} w'_{i1} & w_{i1} r^{12} w'_{i2} \\ w_{i2} r^{21} w'_{i1} & w_{i2} r^{22} w'_{i2} \end{pmatrix} \end{aligned}$$

Construction of $W'R^{-1}y$

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$$

$$R^{-1} = \bigotimes R_i^{-1}$$

$$W'R^{-1}y = \sum_{i=1}^N W_i R_i^{-1} y_i$$

$$\begin{aligned} W_i R_i^{-1} y_i &= \begin{pmatrix} w_{i1} & 0 \\ 0 & w_{i2} \end{pmatrix} \begin{pmatrix} r^{11} & r^{12} \\ r^{21} & r^{22} \end{pmatrix} \begin{pmatrix} y_{i1} \\ y_{i2} \end{pmatrix} \\ &= \begin{pmatrix} w_{i1}(r^{11}y_{i1} + r^{12}y_{i2}) \\ w_{i2}(r^{21}y_{i1} + r^{22}y_{i2}) \end{pmatrix} \end{aligned}$$

Two trait additive example

$$y_{it} = \mu_i + a_{it} + e_{it}$$

$$\boldsymbol{\theta} = \begin{pmatrix} \mu_1 \\ \mathbf{a}_1 \\ \mu_2 \\ \mathbf{a}_2 \end{pmatrix}$$

$$\begin{pmatrix} 0 \\ (1 \dots n_d) \\ n_d + 1 \\ (n_d + 2 \dots 2 * n_d + 2) \end{pmatrix}$$

Implementation

- Build R^{-1}

```
matvec::doubleMatrix R, Ri;  
R.resize(2,2);  
R(1,1) = 1.0;  
R(1,2) = 0.5;  
R(2,1) = R(1,2);  
R(2,2) = 2.0;  
Ri = R.inv();
```

- Initialize

```
unsigned nlevelsDirect    = ped.coder.size();
unsigned n = 1 + nlevelsDirect + 1 + nlevelsDirect;
mme.resize(n,n,0.0);
rhs.resize(n,0.0);
Tr[0] = 1; //  $\mu_1 \Rightarrow$  Trait 1
Tr[1] = 1; //  $a_1 \Rightarrow$  Trait 1
Tr[2] = 2; //  $\mu_2 \Rightarrow$  Trait 2
Tr[3] = 2; //  $a_2 \Rightarrow$  Trait 2
```

- Build $W'R^{-1}W$ and $W'R^{-1}y$

```

J[0] = 0;
J[2] = 1 + nlevelsDirect; double y[2];
while (datafile >> directLevel >> y[0] >> y[1]){
    J[1] = directLevel;
    J[3] = nlevelsDirect + 1 + directLevel;
    for (unsigned i = 0; i < 4; i++) {
        unsigned ii = J[i];
        unsigned ti = Tr[i];
        for (unsigned k = 0; k < 2; k++) {
            rhs[ii] += Ri(ti, k + 1) * y[k];
        }
        for (unsigned j = 0; j < 4; j++) {
            unsigned jj = J[j];
            unsigned tj = Tr[j];
            mme[ii][jj] += Ri(ti, tj);
        }
    }
}

```

- Add $G^{-1} \otimes A^{-1}$

```

matvec::doubleMatrix G;
G.resize(2,2);
G(1,1) = 1.0; G(1,2) = 0.5; G(2,1) = 0.5; G(2,2)=2.0;
matvec::doubleMatrix Gi = G.inv();
unsigned startRow, startCol;
startRow = startCol = 1;
ped.addAinv(mme,startRow,startCol,Gi(1,1));
startRow = 1;
startCol = 1 + nlevelsDirect + 1
ped.addAinv(mme,startRow,startCol,Gi(1,2));
startRow = 1 + nlevelsDirect + 1;
startCol = 1;
ped.addAinv(mme,startRow,startCol,Gi(2,1));
startRow = 1 + nlevelsDirect + 1;
startCol = 1 + nlevelsDirect + 1;
ped.addAinv(mme,startRow,startCol,Gi(2,2));

```

Matvec classes for GLMM

- GLMM
- Matvec Program
- Matvec Class Library
 - Data Prep
 - Model Specification
 - Estimation and Testing

GLMM

- Linear Mixed Model

$$\mathbf{y}_i | \mathbf{u} \sim \text{ind. } N(\mathbf{X}_i \boldsymbol{\beta} + \mathbf{Z}_i \mathbf{u}, \mathbf{R}_i)$$

$$\mathbf{u} \sim N(\mathbf{0}, \mathbf{D})$$

- $\boldsymbol{\mu}_i = E(\mathbf{y}_i | \mathbf{u})$ is a linear function of $\boldsymbol{\theta}$
- $\text{var}(\mathbf{y}_i | \mathbf{u})$ doesn't depend on $\boldsymbol{\mu}_i$
- Data is normally distributed

- Generalized Linear Mixed Model

- Conditional Independence
- Linear predictor

$$\boldsymbol{\eta}_i = \mathbf{X}_i\boldsymbol{\beta} + \mathbf{Z}_i\mathbf{u}$$

- $\boldsymbol{\mu}_i = h(\boldsymbol{\eta}_i)$ is a function of $\boldsymbol{\eta}_i$.
 - * Inverse link function: $\boldsymbol{\eta}_i \rightarrow \boldsymbol{\mu}_i$
 - * Link function: $\boldsymbol{\eta}_i \leftarrow \boldsymbol{\mu}_i$
- Variance function: $\text{var}(\mathbf{y}_i|\mathbf{u}) = \nu(\boldsymbol{\mu}_i, \boldsymbol{\psi})$
 - * Scale parameters: $\boldsymbol{\psi}$

- Posterior mode estimates of the fixed and random effects

$$\begin{pmatrix} X'H'R^{-1}HX & X'H'R^{-1}HZ \\ Z'H'R^{-1}HX & Z'H'R^{-1}HZ \end{pmatrix} \begin{pmatrix} \hat{\beta} \\ \hat{u} \end{pmatrix} = \begin{pmatrix} X'H'R^{-1}y^* \\ Z'H'R^{-1}y^* \end{pmatrix}$$

- Which involves a number of derivatives and other fun things
- `matvec::GLMM` call is a C++ class for working with generalized linear mixed models

Matvec Program

- The GLMM class has been incorporated into the Matvec program
- Structure is very similar to a C++ program
 - Data Prep
 - Model Specification
 - Estimation and Testing

Threshold example from Gianola and Foulley (1983)¹

- Calving Difficulty: 0–2 (Easy–Difficult)
- Fixed Effects: Herd-Year, Age of Sam, Sex of Calf
- Random Effect: Sire $N(0, \sigma_s^2 = 1/19)$
- $\Pr(0|\eta_i) = \Phi(\eta_i)$
- $\Pr(1|\eta_i) = \Phi(\eta_i + \tau_1) - \Phi(\eta_i)$
- $\Pr(2|\eta_i) = 1 - \Phi(\eta_i + \tau_1)$

¹Gianola, D. and J. L. Foulley. (1983) Sire evaluation for ordered categorical traits with a threshold model. *Génét. Sél. Evol.*, **15**:201-224;

```
D=Data();
D.input("../data/calve.dat","HY age sex $ sire n score");
M=Model();
M.equation("score=HY age sex sire,score=intercept");
M.variance("sire",1/19*identity(2,2));
M.variance("residual",identity(2,2));
M.weight("n");
M.link("thresh",0);
M.param(1);
M.fitdata(D);
M.glim();
M.contrast("HY",[1,0,-1]);
M.save("calve_g.out");
M.vce_aireml(40,0);
M.contrast("HY",[1,0,-1]);
M.save("calve.out");
```

RESULTS FROM CONTRAST(S)

```

-----
Contrast MME_addr   K_coef   Raw_data_code
-----
1         1         1   score:HY:1
1         3        -1   score:HY:2
          estimated value (K'b-M) = 0.297455 +- 0.495004
          Prob(|t| > 0.600915) = 0.547965 (p_value)
-----

```

Original Residual Log Likelihood:-28.2167

Iteration 1.0 Res Log Like -27.9195 Change 0.29719

Iteration 2.0 Res Log Like -27.9348 Change -0.0153321

...

Iteration 23.0 Res Log Like -27.9465 Change -1.42109e-14

Iteration 24.0 Res Log Like -27.9465 Change -7.10543e-15

```
Iteration 24 Converged
Final Estimates
0.269984
    0
0.0526316
Last Change
-3.17624e-14
    0
    0
Unscaled Last Change
-3.17624e-14
    0
    0
Residual log likelihood -27.9465
Asy Covariance Matrix
0.202462      0      0
      0      0      0
      0      0      0
```

RESULTS FROM CONTRAST(S)

Contrast	MME_addr	K_coef	Raw_data_code
----------	----------	--------	---------------

1	1	1	score:HY:1
---	---	---	------------

1	3	-1	score:HY:2
---	---	----	------------

estimated value (K'b-M) = 0.504178 +- 0.525854

Prob(|t| > 0.958778) = 0.3378 (p_value)

1794.11 (Error degrees of freedom)

Matvec Class Library

- Same basic structure
- Need to use `matvec::Model M;`
- Instead of `M=Model();`

Header Section

```
#include <fstream>
#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <matvec/statdist.h>
#include <matvec/doublematrix.h>
#include <matvec/model.h>
#include <matvec/data.h>
#include <matvec/glmm.h>
#include <matvec/session.h>
#include "link.h"

using namespace std;
```

Basic structure

```
int main() {
    try{
        .....
    }
    catch (matvec::exception &ex) {
        cerr << ex.what() << "\n";
        exit(1);
    }
    catch (...) {
        cerr << "other exceptions were caught\n";
        exit(1);
    }
}
```

Data Prep

```
matvec::SESSION.initialize("matvec_trash");  
matvec::Data D;  
matvec::GLMM M;  
matvec::doubleMatrix V;  
matvec::Vector<void *> param(1);  
D.input("Data/calve.dat","HY age sex $ sire n score");
```

```
void matvec::Data::input(const std::string & fname,  
                        const std::string & recfmt)
```

- Reads in data from the file *fname* using record format *recfmt*
- Record format
 - String of variable names
 - Alphanumeric variables are denoted with \$

Model Specification

```
M.equation("score=HY age sex sire,score=intercept");
M.variance("sire",(1./19.)*V.identity(2,2));
M.variance("residual",V.identity(2,2));
M.weight("n");
matvec::link_map("thresh","None",&M,0);
double type=1;
param(1)=(void *)&type;
M.Param(param.begin());
M.fitdata(D);
```

```
int matvec::GLMM::equation(const std::string &
                           modelspecs)
```

- Equation for the linear predictor
 - intercept is an automatic variable
 - A(B) factor A is nested within factor B
 - A*B two-way interaction between A and B
- Return value = 1 for a valid model and 0 for an invalid model
- Covariates are defined using `void matvec::Model::covariate (const std::string & covariate_names)`

**void matvec::Model::variance (const std::string
&*termname*, const doubleMatrix &*v*)**

- *termname* is model term for the random effect
 - “residual” is used to identify the residual
- *v* is the covariance matrix
- void matvec::Model::variance (const std::string &*termname*, Pedigree &*P*, const doubleMatrix &*v*)

Additive effects

```
std::string matvec::link_map(std::string link, std::string  
    varfn, GLMM *M, int nvc);
```

- *link* link function
- *varfn* variance function "None"
- *M* Model object
- *nvc* number of residual variance components

Function	Distribution	Link	Inverse Link	$v(\mu)$
normal	Multi. Normal	Identity	η	$\mathbf{R} = \{\sigma_{ij}\}$
normal log	Multi. Normal	Identity	η	$\mathbf{R} = \exp(\{\sigma_{ij}\})$
logit	Binomial/ n	Logit	$e^\eta / (1 + e^\eta)$	$\mu(1 - \mu)/n$
probit		Probit	$\Phi(\eta)$	
cloglog		Complementary log-log	$1 - \exp(-\exp(\eta))$	
thresh	Multinomial	Threshold	$\Pr(\eta + \tau_{i-1} < Z \leq \eta + \tau_i)$	$\text{Diag}(\boldsymbol{\mu}) - \boldsymbol{\mu}\boldsymbol{\mu}'$
poisson	Poisson	Log	e^η	μ
weibull	Weibull			
richards	Richards Function		$\eta_1(1 \pm e^{-\eta_3(t-t\eta_2)})\eta_4$	σ^2

Model Specification Parameters

- Many link functions define a family of link functions and model specification parameters identify a specific link function.
 - `thresh` link function includes both the probit and logit inverse links
 - * Probit:1
 - * Logit:0

```
void matvec::Model::fitdata ( Data & D )
```

- Closes the model specification section
- Identifies the Data object

Estimation and Testing

```
M.glim();
matvec::doubleMatrix Kprime(1,3);
matvec::Vector<double> M0(1);
Kprime(1,1)=1; Kprime(1,3)=-1;
M.contrast("HY",Kprime,M0);
M.save("calve_g.out");
matvec::doubleMatrix Results;
Results=M.AI_REML(40,0);
cout << "sigma^2_s="<<Results(1,1)<<" +- " << sqrt(Results(1,2))<< endl;
```

```
Vector<double> * matvec::GLMM::glim ( int iterations  
= 10 )
```

- Obtain the posterior model estimates of the fixed and random effects
- *iterations* number of iterations
- returns the vector of solutions

```
double matvec::GLMM::contrast ( const std::string &
    termname, const doubleMatrix & Kp, const
    Vector<double> & M )
```

- $H_0 : K'\theta = M$
- *termname* identifies the starting term for K'
- Returns the p-value for the test

```
doubleMatrix matvec::GLMM::AI_REML ( int numiter =  
10, double tol = 1.e-4, )
```

- Find the penalized quasi-likelihood estimates of the variance components using AIREML
- *numiter* number of iterations
- *tol* stopping criteria
 - *tol* > 0 don't allow the PQL to increase
 - *tol* use session ϵ as tolerance
 - *tol* < 0 allow the PQL to increase

```
void matvec::GLMM::GLMM::save ( const std::string &  
    fname, const int io_mode = ios::out|ios::noreplace )
```

- Save the results to the file *fname*
- Covariance matrix and solutions

```
h2=4.*Results(1,1)/(1.+Results(1,1));  
double part,sd;  
part=(4.-h2)/(1.+Results(1,1));  
sd=part*sqrt(Results(1,2));  
cout << "h^2_a="<< h2<<" +- " << sd << endl;  
M.contrast("HY",Kprime,M0);  
M.save("calve.out");
```

Pedigree

- Pedigree

```
#include <matvec/pedigree.h>
....
matvec::Pedigree P;
P.input("Data/calve.txt","individual father mother");
D.input("Data/calve.txt","calf $ sire $ dam $ HY age sex $ score");
```

- Additive Genetic effects

```
M.equation("score=HY age sex calf,score=intercept");
M.variance("calf",P,(1./4.)*V.identity(2,2));
```