ASSIGNMENT 1.

Suggestion: Always use "implicit none" as the second line of the program to help the program detect undeclared variables.

```
Program ....
implicit none
...
```

1. Write a program shown in the textbook at page 10 and see what this program does.

2. Write a program that prints

```
    2 to the power of 20 is xxx
```

where `xxx` is replaced by the value of 2^{20}.

3. Use appropriate loops for programs that calculate the following sums:

A. $55 + 56 + \cdots + 100$
B. $(-1) + (-2) + \cdots + (-100)$
C. $1 + 0.5 + 0.25 + \cdots$
D. $1 + 0.5 + 0.25 + \cdots + x$ where $x < 10^{-3}$.

4. Read an integer variable `i` and a real variable `x` from the console repeatedly. Depending on the value of the integer variable, print the following:

INTEGER VALUE	ACTION
1	logarithm of `x`
2	sine of `x`
3	square of `x`
ELSE	Quit the program.

Use the DO loop without arguments and the CASE statement.

Note: To read variables x and y from the keyboard, type

```
read*,x,y
```

5. Create a file with a pedigree line that contains animal ID, sire ID, dam ID and year of birth:

```
HOL234HOL001HOL9831998
```

Write a programs that will read the IDs and year of birth into separate variables.

6. Print the following number: `1.0/3.0**(-i)` for `i=1,10` in F, E and G formats using specifications `10.3` and `12.6`.

OPTIONAL

7. Browse through a list of Fortran functions in your manual. By writing simple programs, determine differences between `int`, `nint`, `floor` and `ceiling`.

8. Write a program that

- Reads an alphanumeric variable of length up to 10.
- For each character of that variable, writes a numeric equivalent of that character.

The numeric equivalent of character `c` is `ichar(c)`.

ASSIGNMENT 2.

1. Set matrices A and B to:

$$\mathrm{A} = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} \text{ and } \mathrm{B} = \begin{bmatrix} 3 & 1 \\ 4 & 8 \end{bmatrix}.$$

- Calculate $\mathrm{C} = \mathrm{AB}$ as matrix multiplication using loops and `matmul`. Print. Do results differ?
- Write the sum of A as a whole, along dimension 1, and along dimension 2.
- Print the first column of A.
- Count how many numbers in $\mathrm{A} + \mathrm{B}$ are larger than 5 and compute their sum; do not use any loop.

2. Initialize a matrix `x(100,100)` to $x_{\mathrm{ij}} = 100i + j - 1$. Create an internal function `tr` as

$$\mathrm{tr(n, m)} = \sum_{i=n}^{m} x_{ii}$$

and calculate `tr(1,10)` and `tr(90,100)`.

3. In the following data (to be read from a file), a line containing the dimensions of a 2-dimensional matrix is followed by the data values of this matrix.

```
2
1.5 3.2
2.7 3.3
4
4 3 2 1
5 4 3 2
6 5 4 3
7 6 5 4
1
10
```

Create the file above. Write a program using the dynamic memory allocation that will:

a) read the dimension
b) allocate a real matrix with the given dimensions
c) read that matrix
d) print that matrix
e) deallocate the matrix
f) continue until the end of file

OPTIONAL

4. Create a vector `x` of 100,000 reals and initialize `x(i)=i+1.0/3.0`. Store it in files in two formats:

a) formatted
b) unformatted

Then read the files and accumulate the sums. Compare timings of either reading using the `CPU_TIME` subroutine. Close files with options to delete them.

5. Write to file the following:

```
   1 : printed 1 number(s)
   1 2 : printed 2 number(s)
   ...
   1 2 3 4 5: printed 5 number(s)
```

without using more than 2 write statements. Use clause ADVANCE='NO'.

ASSIGNMENT 3.

1. Read descriptions of array functions. Create examples for: `maxval`, `maxloc`, `count`, `pack`, `reshape`, `size` and `unpack` using the following vector and matrix.

$$\mathbf{x}' = [2.5 \quad -1.3 \quad 0.0 \quad 5.0 \quad -6.3 \quad 2.2]$$

$$\mathbf{M} = \begin{bmatrix} 2 & 4 & -1 & 5 & -4 \\ -1 & 2 & 0 & 0 & 6 \end{bmatrix}$$

2. Let **A** and **b** be:

$$\mathrm{A} = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix} \text{ and } \mathrm{b} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}.$$

Write a function that calculates the quadratic form: $\mathbf{b}'\mathbf{Ab}$. Make this function:

a) internal function and
b) module.

3. Write a function that calculates a sample (r) from a pseudo-normal distribution using a formula:

$$r = \left(\sum_{i=1}^{12} u_i\right) - 6$$

where $u_i \sim \mathrm{UN}(0,1)$, the uniform distribution. Samples from the distribution $\mathrm{UN}(0,1)$ are generated by subroutine `random_number()`, which returns the random number $0 \leq x < 1$.

Write a subroutine that, given a vector with samples, calculates mean and SD.

Write a program that asks for a number of samples to be generated, generates those samples from

a) $\mathrm{UN}(0,1)$ and
b) $\mathrm{N}(0,1)$

and then calculates their mean and SD.

4. Modify function that generates normal samples to accept parameters for mean and variance. Make this parameters optional with default mean=0 and variance=1.

OPTIONAL

5. Write a subroutine `printnice_r(x,n)`, that will print a real matrix `x(n,n)` with nicely aligned columns.

6. Modify function `printnice` as:

a) Putting it in a module as an internal procedure.
b) Eliminating the n argument.
c) Adding an optional argument, which is format for one line.

ASSIGNMENT 4.

1. Write subrouties `printnice_int` and `printnice_real` that print real and integer matrices. Overload so that `printnice` prints both. Test.

2. Write *`Makefile`* to compile.

3. The program contains the data on animals in structure `calf`. Create a subroutine that given a variable of type animal will print the information on that animal.

```
module animals
   type calf
      character (10) :: id
      integer :: year_of_birth
      character:: sex
   end type
end module

program registration
use animals
implicit none
type (calf) :: a,b,c
a=calf('small',1997,'M')
! b is also a male born in 1997 but has a different name
b=a
b%id='large'
end program
```

Write subroutine `print_am` that prints ID, year of birth and sex, with syntax as below:

```
call print_am(a)

call print_am(b)
```

OPTIONAL

4. This assignment tests data structures for sparse vectors, operations on data vectors, and operator overloading. All definitions and procedures need to reside in module `sparse_vec`. Write:

a) data structure for sparse vector
b) subroutine that creates sparse vector from a dense vector
c) subroutine that prints a sparse vector
d) subroutine that multiplies two sparse vectors.

Provide interface to c) with "=" operator. Provide interface to d) with "+" operator.

To the last assignment, add interfaces to "+" and "*" that allow to add or multiply sparse vector by a regular vector.

ASSIGNMENT 5.

1. Run program *Lsq.f90*. This program with related data sets can be found in directory listed on board. Modify the program so that it can read a parameter file containing: name of data file, number of effects and number of levels. The parameter file could contain the following:

```
data_pr1    !name of the data file
2           !number of effects
2 3         !number of levels for each effect
```

2. Apply the modified program to:

a) the original data and
b) the data as below

MANAGEMENT	ANIMAL	TREATMENT	OBSERVATION
1	1	1	12
1	2	2	14
1	3	1	13
2	3	1	16
2	2	2	18
3	4	1	15
3	3	2	16
3	2	2	19

OPTIONAL

3. Think how the parameter file can be simplified so that the number of effects is not given, i.e. the above parameter file simplifies to:

```
data_pr1    !name of the data file
2 3         !number of levels for each trait; defines the number of effects
```

hint: when the reading is unsuccessful, for example because of too few numbers, the variable associated with `iostat` becomes $\neq 0$.

4. Examine differences between *Lsq.f90* and *Lsqmt.f90*. Run the latter. Change $\mathbf{R}_0$ to these matrices:

$$\mathbf{R}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \text{ or } \mathbf{R}_0 = \begin{bmatrix} 10 & 0 \\ 0 & 20 \end{bmatrix}$$

and compare solutions. Should they be different?

ASSIGNMENT 6.

1. Write a program that creates the inverse of the animal relationship matrix. Use the following pedigree:

ANIMAL	SIRE	DAM
1	3	4
2	3	4
3	5	Unknown
4	Unknown	Unknown

2. Incorporate the program above in *lsq.f90*. Run it with data of the yesterday's assignment and pedigrees as above assuming $\text{var}(\mathbf{e}) = 10\mathbf{I}$ and $\text{var}(\mathrm{u}) =$

a) $2\mathbf{I}$ and
b) $2\mathbf{A}$

where $\mathbf{u}$ is the animal effect. Compare solutions.

Hint #1: Either add contributions within the program or create the **A** matrix separately and add as a matrix.

Hint #2: Make it as simple as possible (it does not need to be general).

3. Modify parameters in *blup.f90* to support the same models as above. Compare solutions. Estimable functions should be identical.

OPTIONAL

4. In 3, make management autocorrelated with $\text{var}(\text{management}) = 5.0$ and $\rho = 0.8$. Then, vary the autocorrelation to 0.98 and 0.9 and compare solutions for the management effects.

VERY HARD

5. Modify parameters in *blup.f90* for a maternal model using parameters from 2b. Possibly compare solutions against another program.

Hint: If the dam is not identified, create a phantom dam.

ASSIGNMENT 7

DENSEOP

1. This assignment involves module `DENSEOP`, which is described at http://nce.ads.uga.edu/wiki/doku.php in modules. An example of the use of this module is in directory `testdense` of a course directory (/work/course2016/). Because of complexity, the program inside the directory needs to be compiled by the command:

```
make
```

For make to work, copy file *`Makefile`* as well as all the other files from the same directory. Examine contents of ***`kind.f90`***, *`Lapack90r.f90`* and *`denseop.f90`*. They are in directory /work/course2016/libs/

Consider the following system of equations $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} 36 & 30 & 18 \\ 30 & 41 & 23 \\ 18 & 23 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 150 \\ 181 \\ 106 \end{bmatrix}.$$

Using dense matrix module `DENSEOP` and precision (`r8`):

a) calculate Cholesky decomposition $\mathbf{A} = \mathbf{LL}'$ and check its correctness by matrix multiplication,
b) calculate eigenvalues and eigenvectors: $\mathbf{A} = \mathbf{VDV}'$ and verify that $\mathbf{VV}' = \mathbf{I}$,
c) calculate the determinant of $\mathbf{A}$
d) solve for $\mathbf{x}$ using both symmetric and general solvers; confirm the correctness of solutions by multiplication and
e) calculate the inverse of $\mathbf{A}$ and verify that $\mathbf{AA}^{-1} = \mathbf{I}$.

2. Repeat solving in different data formats: dense and half stored, single and double precision. To create a half stored matrix, convert a full-stored matrix.

Hint: If a regular double precision matrix is defined as

```
real (r8)::xf(4,4)
```

a half stored matrix in same precision is declared as

```
real (r8)::xh(10) !or n*(n+1)/2
```

OPTIONAL – SPARSEM & SPARSEOP

2. The assignments include the sparse matrix module `SPARSEM`. An example of the use of this module is in file *`test.f90`* in `testsparse` of the course directory (work/course2016/). Because of complexity, the program inside this directory program needs to be compiled by the command:

```
make
```

For make to work, copy file *`Makefile`* as well as all the other files from the same directory.

For programs with name other than test, make and *`Makefile`* can still be used, but replace "`test`" with the name of your program in file `Makefile`.

Data structures for different matrix formats are in file *`sparse.f90`* in directory

/work/course2016/libs/

Consider the following system of equations $\mathbf{Ax} = \mathbf{b}$:

$$\begin{bmatrix} 36 & 30 & 18 \\ 30 & 41 & 23 \\ 18 & 23 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 150 \\ 181 \\ 106 \end{bmatrix}.$$

a) Set this system of equation with `densem` and `sparse_hash` matrices,
b) convert the format `sparse_hash` to `sparse_ija`,
c) print all matrices,
d) solve by iteration and
e) print sparse matrices in internal format with `printm(x,’internal’)`.

3. Create a tridiagonal sparse matrix `A(1000,1000)` in `densem` format such that:

```
a(i,i)=10+un(0,1)
a(i,j+1)=un(0.1) for i=1 to 999
a(j,i+1)=un(0.1) for j=1 to 999
```

a) Generate `b(1000)` from UN(0,1),
b) Solve system of equations $\mathbf{Ax} = \mathbf{b}$ using `densem` and `sparse_hash` formats. Compare timings.
c) Convert **A** to `sparse_ija` format and solve $\mathbf{Ax} = \mathbf{b}$ using FSPAK90.

ASSIGNMENT 8

Create a directory `f90` and download file *`progsf90.tar`*. This file is available at directory of the course (/work/course2016/) or at nce.ads.uga.edu. Uncompress this file into a newly created directory by:

```
tar -xvf progsf90.tar
```

Then compile all the libraries and programs:

```
make linux-intel
make all
```

Each program has its own directory and its own *`Makefile`*. To recompile any program separately, go to its directory and type

```
make
```

1. Read:

a) Readme, Installation, and Makefile files in the main directory,
b) Readme in directory `libs`. Identify programs that implement reading the parameters, sparse matrix factorization, and hash function.

2. Run *blupf90* with examples specified in Appendices A-C from the BLUP manual. Examples are stored in the directory examples. Parameter files start or end with letters `ex`.

3. Directory `test8` on the course directory (/work/course2016/) includes data/pedigree files for a 3 trait maternal model and a parameter file for one trait model. Run *blup90* for one trait with the following solving options:

a) by default iteration (PCG),
b) by Gauss-Seidel iteration and
c) by FSPAK.

For details how to change the solving options, see *`blupf90.f90`* and *`readme.blupf90`* in the blupf90 directory. Also look follow links to programs and FAQ at nce.ads.uga.edu.

OPTIONAL

4. For the previous data set, create a parameter file for 3 traits. Run using:

a) *blupf90* with the default solver (PCG)
b) iteration on-data program `bin/blup90iod2` (in course directory)

5. Modify *`blupf90.f90`* so that it calculates accuracies for the animal effect. Accuracies for an animal in equation *i* in a single-trait situation is defined as:

$$1 - \mathrm{LHS}^{ii}/\sigma_a^2$$

where LHS^{ii} is the *i*-th diagonal of the inverse of the left hand side.

Suggestion: Invert by FSPAK and obtain elements of the inverse by function "`getm`" from module `sparsem`. Or use "OPTION sol se" for *blupf90* - see *blupf90* documentation.

Warning: The formulas are correct only without unknown parent groups in the model.

ASSIGNMENT 9

This assignment involves the *blupf90* family of programs. Binaries are available in directory bin. Parameter files for this exercises are in directory examples. Files ending with *99 contain data files for up to 14 type traits. Parameter file `exmr99s1` uses these files for a single-trait model, and `exmr99s` for 3 traits.

1. Browse through FAQ for *BLUPF90* programs. Read information on Input files, frequent mistakes, and REML.

2. Calculate estimates of variance components by *remlf90* and *airemlf90* using one and three trait models. Record the number of rounds and CPU time. Unix (including Linux) allows for measuring the CPU time by preceding command with time, e.g.,

```
time remlf90
```

Attention: if computing takes too long, only obtain approximate time per round, or terminate.

3. Read FAQ pages on Gibbs sampling and Post-Gibbs analysis.

4. For one-trait and 3 trait models, obtain 5,000 samples using *gibbs2f90*. Give burn-in of 1 and store every sample. Analyze samples with *postgibbf90*. Find a burn-in and estimate parameters. Re-estimate parameters with burn-in doubled. Are estimates very different?

5. Formulate conclusions. Are they similar to those in the paper "Reliable computing in estimation of variance components"?

OPTIONAL

7. Convert data in `exmr99s` to result in one binary trait, one with 3 categories, and one linear. Estimate parameters, first with a linear model (`gibbs2.f90`) and then with a threshold model (`thrgibbs1.f90`). Use *postgibbsf90* to determine the number of needed samples. Are estimates of heritabilities and genetic correlations between data sets and programs similar?

8. Extend the model to 7 traits and possibly more traits. For additional co-variances, use average of current diagonals for new diagonals, and 1.0 for off-diagonals. How much longer are the computations with 7 traits compared to a single trait model for:

a) AIREMLF90
b) gibbs2f90

ASSIGNMENT 10

BLUPF90 – Estimation of breeding values and reliabilities

1. Documentation for BLUPF90 program in the wiki: http://nce.ads.uga.edu/wiki/doku.php and also in the BLUP manual.

2. Using the following example (from Mrode and Thompson, 2005 *Linear Models for Predicting Animal Breeding Value*)
   Create data, pedigree and parameter to run **renumf90** and then run **blupf90** to obtain solutions and reliabilities.

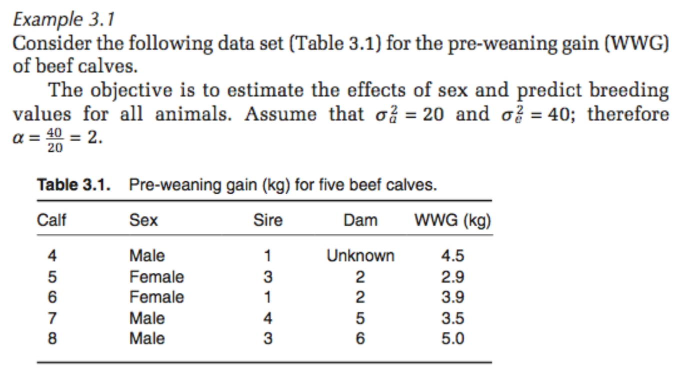
   Reliability for animal i can be calculated as:

   $Rel_i = 1 – PEV_i/VarA$

Where:

PEV is the prediction error variance (S.E. = sqrt(PEV))

VarA is the additive genetic variance



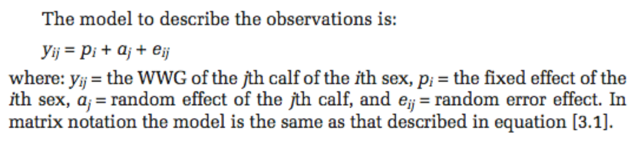
Example 3.1

Consider the following data set (Table 3.1) for the pre-weaning gain (WWG) of beef calves.

The objective is to estimate the effects of sex and predict breeding values for all animals. Assume that $\sigma_a^2 = 20$ and $\sigma_e^2 = 40$; therefore $\alpha = \frac{40}{20} = 2$.

Table 3.1. Pre-weaning gain (kg) for five beef calves.

Calf	Sex	Sire	Dam	WWG (kg)
4	Male	1	Unknown	4.5
5	Female	3	2	2.9
6	Female	1	2	3.9
7	Male	4	5	3.5
8	Male	3	6	5.0

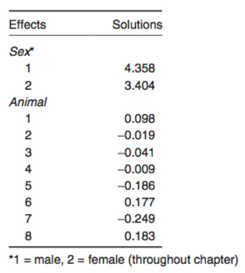


The model to describe the observations is:

$y_{ij} = p_i + a_j + e_{ij}$

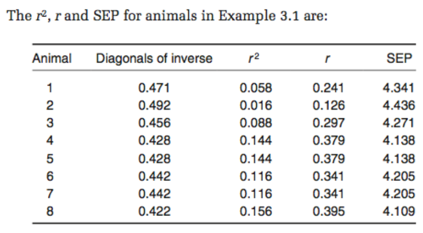
where: y_{ij} = the WWG of the jth calf of the ith sex, p_i = the fixed effect of the ith sex, a_j = random effect of the jth calf, and e_{ij} = random error effect. In matrix notation the model is the same as that described in equation [3.1].

The solutions from example are:



Effects	Solutions
*Sex**	
1	4.358
2	3.404
Animal	
1	0.098
2	–0.019
3	–0.041
4	–0.009
5	–0.186
6	0.177
7	–0.249
8	0.183

*1 = male, 2 = female (throughout chapter)



The r^2, r and SEP for animals in Example 3.1 are:

Animal	Diagonals of inverse	r^2	r	SEP
1	0.471	0.058	0.241	4.341
2	0.492	0.016	0.126	4.436
3	0.456	0.088	0.297	4.271
4	0.428	0.144	0.379	4.138
5	0.428	0.144	0.379	4.138
6	0.442	0.116	0.341	4.205
7	0.442	0.116	0.341	4.205
8	0.422	0.156	0.395	4.109

3. Directory test10 on the course directory (/work/course2016/) includes data/pedigree files for a 3 trait maternal model in beef cattle. README file contains header for example10.dat and example10.ped
   a) Create a *renumf90* parameter file for YW (yearling weight) that fits the following model:

      Y = CG + β AOD + animal + maternal + sire-herd + e

      Consider CG as fixed; AOD as covariable; animal, maternal, sire-herd as random. Starting values for variance components are: $\sigma_a^2 = 438.90$; $\sigma_m^2 = 73.24$; $\sigma_{am} = -35.80$; $\sigma_{sh}^2 = 242.12$; $\sigma_e^2 = 751.13$
      Use depth of pedigree equal 3

   b) Run *renumf90* and check the output files.

   c) Run *airemlf90* with an option to get SE for heritability and for genetic correlation between direct and maternal effects.

   d) Change the parameter file for *renumf90* for a 3-trait model (BW, WW, YW) and run *blupf90*. Variance components are in the README file.

OPTIONAL

4. Random regression models using renumf90 and blupf90
   Parameter, data and pedigree files (renrr.par, datrr.leg, pedrr) for this exercise are in directory /home/course/courseadmin/course/lab1/

   This is data for a random regression model using Legendre polynomials from example 7.2 of Mrode and Thompson, 2005 Linear Models for Predicting Animal Breeding Value, Example 7. Look the parameter file and identify components of renumf90 parameter file Run blupf90 to obtain solutions.

5. Copy `tabular.f90` and `pedigree.txt` from `/scr1/yutaka/pub` to your directory (folder). This program reads the pedigree file and calculates inbreeding coefficients using a naïve recursive algorithm. The pedigree has been already sorted in chronological order (old to young).
   1. Compile the program and run it.
   2. Parallelize the program with the OpenMP directives. Compare the running time between programs with/without OpenMP. Is the parallelization beneficial enough?
   3. The running time can be shortened with OpenMP. Rewrite the program so that it reduces the running time with this specific pedigree.

Hint: To compile a program with OpenMP, use the option `-qopenmp` (or `-openmp`) for Intel Fortran (ifort) and use `-fopenmp` for GFortran.

ASSIGNMENT 11

Single-step GBLUP

The data for this lab was created using simulation for a single trait animal model. Simulation was done using QMSim (Sargolzaei, M. and F. S. Schenkel, 2009)

The simulated phenotype were generated using the following model:

Phenotype = mean + true_ebv + residual

Parameters:

Mean = 1.0
True variances:
    direct genetic = 0.25
    residual = 0.72

Simulated files are available in the folder:

```
/work/course2016/lab11
```

Description of files

data.txt:
1: Animal ID
2: Sire ID
3: Dam ID
4: Generation number
5: Sex code
6: Number of male progeny
7: Number of female progeny
8: Inbreeding coefficient
9: Homozogosity
10: Phenotype
11: Simulated residual
12: Polygenic effect
13: True EBV
14: Internal EBV from QMSim
15: Mean (column of ones to fit mean effect in BLUPF90)

ped.txt:
1: animal ID
2: sire ID
3: dam ID

snps.txt:
1: animal ID
2: marker information

1. Copy the full folder into your directory
   ```
   cp –r /work/course2016/lab11 .
   ```

2. From raw data modify renumf90 parameter file (renlab.par) according to the data file and to fit the following model for genomic selection:

$$y = mean + animal + e$$

3. Run renumf90 program to renumber data, pedigree file, and marker data.

4. Check the renf90.par, renf90.dat, and renaddxx.ped. From the renaddxx.ped file, identify genotyped animals, and check with wiki (http://nce.ads.uga.edu/wiki/doku.php?id=readme.renumf90) the content of each column.

5. Estimate variance components considering and ignoring marker information.
   From the airemlf90 output find the following statistics: number of genotyped animals, number of SNP markers

6. Run blupf90 without marker information using estimated variance components. Now run blupf90 using genomic information and compare cpu time and solutions. Obs: Check wiki to see how to read an external or pre-computed genomic matrix.

7. Validation on young candidates (individuals from 10th generation with no phenotypes).

   a) Remove the phenotypic information from the 10th generation and obtain solutions from a model with marker information and with no marker information.

      Hint: if generation column is number 4, new data can be created using the AWK Linux tool
      ```
      awk '$4!=10' renf90.dat > renf90.dat.pred
      ```

   b) Compare correlations with true breeding values for genetic additive direct effect. Hint: have renumf90 passing to the data a column containing "generations".

OPTIONAL

Exclude chromosomes with even numbers and repeat validation using truncated data and compare correlation with TEBV for reduced and full genome.

ASSIGNMENT 12

Single-step GBLUP and quality control of genomic data

The data for this lab is based on a public pig dataset from PIC (Cleveland et al. 2012 - G3 Journal). Originally this dataset was filtered for MAF and missing SNP were imputed, however some modifications were introduced to generate commons problems that are found in datasets.

Files are available in the folder:

```
/work/course2016/lab12
```

Description of files

phenotypes_new.txt:

1: Animal ID
2: Trait 1
3: Trait 2
4: Trait 3
5: Trait 4
6: Trait 5
7: Mean

pedigree_new.txt:

1: animal ID
2: sire ID
3: dam ID

genotypes_new.txt:

1: animal ID
2: marker information

1. Copy the full folder into your directory
```
cp -r /work/course2016/lab12 .
```
Go to http://nce.ads.uga.edu/wiki/doku.php?id=readme.pregsf90
and check all options available for preGSf90

2. Change parameter file for renumf90 as necessary and run preGSf90
- Check number of SNPs, all statistics related to SNPs, number of duplicated genomic samples (possible clones), and check parent-progeny conflicts
- If there is any parent-progeny conflict, use seekparentf90 to discover the true parents. Check for options and output files from seekparent program in
http://nce.ads.uga.edu/wiki/doku.php?id=readme.seekparentf90
Use the new pedigree file for the next exercise.

3. Run preGSf90 WITH quality control, removing duplicated animals and saving clean files. Check the number of animals and number of genotypes in the clean files. Check also files *_removed.

4. With clean marker files, the option for no quality control is useful for saving time. Plot principal components and check population structure.

5. With clean marker file, explore options to save G and A22 matrices in text format and with original IDs.

6. Matching G and A22:
   Run blupf90 with the with the option `OPTION tunedG 0`
   Compare statistics from G and correlations between G and A22 using `OPTION tunedG 0` and the default (`OPTION tunedG 2`).

7. Run preGSf90 and save H^{-1}. Run blupf90 using the option to read external files to include H^{-1} in the MME. Compare solutions with blupf90 when H^{-1} is constructed internally and with blupf90 when reading GimA22i.

ASSIGNMENT 13

Single-step GWAS

The data for this lab was created using simulation for a single trait animal model. Simulation was done using QMSim (Sargolzaei, M. and F. S. Schenkel, 2009)

The simulated phenotype was generated using the following model:

Phenotype = mean + true_ebv + residual

Files are available in the folder:

```
/work/course2016/lab13
```

Description of files

pheno.txt:

1: mean
2: animal id
3: sire id
4: dam id
5: sex
6: generation
7: number of males progenies
8: number of females progenies
9: inbreeding
10: homozygosity
11: phenotype
12: simulated residual (e)
13: individual true breeding value for polygene
14: individual true breeding value for direct effect (qtl)
15: EBV from QMSim internal BLUP

pedigree.txt:

1: animal ID
2: sire ID
3: dam ID

mkr.txt:

1: animal ID
2: marker information

chrmap:

1: SNP ID
2: Chromosome
3: position

1. Copy the full folder into your directory
   ```
   cp –r /work/course2016/lab13 .
   ```
   Go to http://nce.ads.uga.edu/wiki/doku.php?id=readme.pregsf90
   and check all options available for postGSf90

2. Run renumf90 program using 'renum.par' parameter file to renumber data, pedigree, and marker files

3. Run blupf90 and get solutions

4. Add an option to read a map file (*chrmap*) and run postGSf90. Check the output files.

5. Try analyses using option to get variance explained by windows of adjacent SNPs and add the option to generate Manhattan plots. Check the output files.

6. Prediction of DGV for young individuals. The postGSf90 creates a file *snp_pred* with information about the random effect (number of traits + correlated effects), the gene frequencies and the solutions of SNP effects.

   Use program predf90 to predict DGV using a marker file for young individuals (*young_anim*)