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

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

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:

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

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

Hint: `pack` and `unpack` require a mask that determines which elements are taken from or placed in the array, respectively.

2. Let $\mathbf{A}$ and $\mathbf{b}$ 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: $\mathbf{b}'\mathbf{A}\mathbf{b}$. Make this function:

a) internal function
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 a module with subroutines `printnice_int` and `printnice_real` that print real and integer matrices. Overload so that `printnice` prints both. Test.

2. Write *`Makefile`* to compile the program and module created above.

3. The following program contains the data on animals in structure `calf`. Create a subroutine that given a variable of type `calf` 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 `/work/course2018/week1/day5`

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
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

3. Examine differences between *lsq.f90* and *lsqmt.f90*. Run the latter. Change $\mathbf{R}_0$ to these matrices:

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

Compare solutions. Should they be different?

OPTIONAL

4. Think how the parameter file in exercise **2** 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 $\neq 0$.

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
5	Unknown	Unknown

2. Incorporate the program above in ***lsq.f90***. Run it with pedigree as above and the following data:

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

Assume that $\text{var}(\mathbf{e}) = 10\mathbf{I}$ and $\text{var}(\mathbf{u}) =$

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

where **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).

Hint #3: If you need a working version of ***lsq.f90***, copy it from /work/course2018/week2/day6

3. Copy the folder day6 that is in the course directory (/work/course2018/week2/). 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 the module `DENSEOP`, which is described at http://nce.ads.uga.edu/wiki/doku.php?id=modules. An example of the use of this module is in directory `testdense` in the course directory (/work/course2018/week2/day7). Copy the entire directory day7:

```
cp -r /work/course2018/week2/day7 .
```

Before going to the folder `testdense`, go to the folder `libs` and compile all libraries by using the `Makefile`. Just type the command:

```
make
```

Examine contents of *`denseop.f90`* and *`kind.f90`*. They are in the folder `libs`

Go to the folder `testdense` and compile `testdense.f90` by using the `Makefile`. Just type the command:

```
Make
```

Run the program and see what it does.

2. 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 $\mathbf{AA}^{-1} = \mathbf{I}$.

3. 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 of same precision is declared as

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

OPTIONAL – SPARSEM & SPARSEOP

4. The assignments include the sparse matrix module `SPARSEM`. An example of the use of this module is in the file `test.f90` in the folder `testsparse`. Compile `test.f90` by using the `Makefile`. Just type the command:

```
Make
```

If you give your program a name different than `test.f90,` you can still use the `Makefile`. Just replace "`test`" with the name of your program in the file `Makefile`.

Data structures for different matrix formats are in the file `sparse.f90` in the folder `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')`.

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

```
a(i,i)=10+un(0,1) for i=1 to 1000
a(i,i+1)=un(0,1) for i=1 to 999
a(i+1,i)=(i,i+1)
```

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

ASSIGNMENT 8

Create a directory `f90` and copy the file `progsf90_2018.tar.gz` inside the new directory. This file is available in the course directory (/work/course2018/week2/day8).

Uncompress this file by typing: `tar -xvf progsf90_2018.tar.gz`

Go to directory `progsf90_2018` by typing: `cd progsf90_2018`

Then compile all the libraries and programs:

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

Hint 1: All executables will be placed inside the folder `bin`

Hint 2: If you are inside the folder `progsf90_2018`, you can run `blupf90` by typing:

`bin/blupf90` or `./bin/blupf90`

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` 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`.

Check which animals have the highest EBV.

3. Directory `test8` in the course directory (/work/course2018/week2/day8) includes data and 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
c) by FSPAK.

Compare computing time. Which method is the fastest one?

Hint 1: use the Unix command `time` to get computing time. If you are inside the folder `test8`, just type:

`time ./../progsf90_2018/bin/blupf90`

Hint 2: alternatively, you can copy `blupf90` to the folder `test8`. If you are inside `test8`, type:

`cp ../progsf90_2018/bin/blupf90 .`

To run `blupf90` and obtain computing time if the program is in the folder you are working, type:

`time ./blupf90`

For details how to change the solving options, see *`blupf90.f90`* and *`readme.blupf90`* in the `blup` directory. Also look at the links to application programs and FAQ at http://nce.ads.uga.edu/wiki/doku.php.

OPTIONAL

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

a) *blupf90* with the default solver (PCG)

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:

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

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 the function "`getm`" from the module `sparsem`. Check which animals have the highest accuracy.

Warning: The formulas are correct only without unknown parent groups in the model. The above formula does not consider inbreeding.

ASSIGNMENT 9

This assignment involves the `blupf90` family of programs. Remember that all binaries were created yesterday and are available in the directory `bin`, inside directory `progsf90_2018`. Parameter files for the following exercises are in directory `examples`, inside directory `progsf90_2018`. Parameter file ***`exmr99s1`*** uses these files for a single-trait model, and ***`exmr99s`*** for 3 traits. Check ***`exmr99s1`*** and identify the name of the data and pedigree files, the number of effects in the model, which effect is the animal effect, and the number of animals in the pedigree.

1. Browse through FAQ for `blupf90` programs (http://nce.ads.uga.edu/wiki/doku.php?id=faq). Read information on frequent mistakes in data, pedigree, and parameter files. Read also information on 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. Use the Unix command `time` to get computing time., e.g.,

```
time remlf90
```

Attention: if computing takes too long, obtain only 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 0 and store every sample. Analyze samples with `postgibbf90`.

Check `postind` file to see the position of each variance component in the output.
Check posterior mean and standard deviation.
Check parameters that can be used to test convergence.
Plot a graph for samples and find out the number of initial samples to be discarded as a burn-in. Run `postgibbf90` again using that burn-in.
Re-estimate parameters with doubled burn-in. Are estimates very different?

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

OPTIONAL

6. 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?

Hint: in the `blupf90` programs, 0 is considered missing data. So, do not use 0 as failure when defining binary or categorical traits.

7. 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 file to run `renumf90`. Run `renumf90` and check the log (screen) and in the files created by the program.

3. 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

4. Directory day10 on the course directory (/work/course2018/week2) includes data and 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

5. Random regression models using `renumf90` and `blupf90`
   Parameter, data and pedigree files (renrr.par, datrr.leg, pedrr) for this exercise are in directory (/work/course2018/week2/day10/opt4)

   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.

6. Copy `matmul1.f90` from /work/course2018/week2/day10 to your directory (folder). This program calculates the matrix product **C**=**AB** where all matrices are square with the same rank.
   1. Compile the program and run it. Measure the timing using the `time` command.
   2. Parallelize the computations with the OpenMP directives. Compare the running time between programs with/without OpenMP. Is the parallelization beneficial enough?

Hint 1: To compile a program with OpenMP, use the option `-qopenmp` for Intel Fortran (ifort) or `-fopenmp` for GFortran. Without the option, the OpenMP directives are ignored.

Hint 2: To run a program with OpenMP, specify the number of threads used by the program. On Linux, the following command uses 2 threads and measures the computing time for `a.out`. On dodo2, you will use only 1 thread by default.

```
OMP_NUM_THREADS=2 time ./a.out
```

ASSIGNMENT 11

- Edit SNP file using Unix-Linux tools
- Quality control of SNP data using the software qcf90
- Heritability of gene content

Lab 11 uses public genotypes and pedigree from:
Cleveland, Matthew A., John M. Hickey, and Selma Forni. "A common dataset for genomic analysis of livestock populations." G3: Genes, Genomes, Genetics 2, no. 4 (2012): 429-435.

Files are available in the folder day11. Copy the entire folder using the following command:

```
cp -r /work/course2018/week3/day11 .
```

(A) Editing SNP files (SNP file, marker file, genotype file refer to the same thing)

The SNP data is in: `genot_pic.txt`

1. Look at the genotype file
```
less -S genot_pic.txt
```

2. How many genotyped animals are in the file?
```
wc -l genot_pic.txt
```

3. Check the "quality" of the file for some typical errors. For example, are there duplicated animals in the marker file? Check it using
```
awk '{print $1}' genot_pic.txt | sort | uniq -c | awk '$1>1'
```

4. How many SNP the animals were genotyped for?
```
awk 'END {print length($2)}' genot_pic.txt
```

5. Were all animals genotyped for the same number of SNP?
```
awk '{print length($2)}' genot_pic.txt| sort -u
```

6. Extracting whatever SNP we like (e.g., SNP number 7).
```
awk '{print substr($2,7,1)}' genot_pic.txt > snp1
```

7. Alleles are coded as gene content {2,1,0,} for genotypes {AA, Aa, aa}. For one SNP (e.g., number 7), we can count the number of AA genotypes as
```
awk '{print substr($2,7,1)}' genot_pic.txt | awk '$1==2' | wc -l
```

8. Compute the allele frequency counting the number of 0,1,2 in that SNP

(B) Quality control of SNP

Go to the folder qc_simple/

1. The program `filter_maf.f90` takes a genotype file and filter alleles for a minimum MAF (Minor Allele Frequency) threshold. It creates another genotype file and outputs statistics (e.g. average frequency) from original and new genotype file.
Read, compile and run `filter_maf.f90`. It demands the name of the genotype file and the minimum MAF.
Use both `genot_pic.txt` and `genot_pic_noisy.txt`

2. Use `qcf90` software. To know how to call it, you may use `qcf90 --help`
Run it with the marker file: `genot_pic.txt`.
Explore output files and create a new marker file removing markers with MAF < 0.05

Repeat the exercise with `genot_pic_noisy.txt` as marker file.

(C) Heritability of gene content

Go to the folder qc_h2/

The script `check_h2.sh` extracts one marker from the genotype file and estimates its heritability.

1. Take a look at the script
2. Run it, preferably with the markers that are mentioned in the script itself
3. What markers would you remove?
4. Change in the script the name of the file that you are going to check, from `genot_pic.txt` to `genot_pic_noisy.txt`. Repeat the analysis with the same markers, what do you observe?
5. Produce a clean marker file using `qcf90` and repeat again the analysis with the new file.

ASSIGNMENT 12

- Calculate breeding values using SNP_BLUP, classical GBLUP, and GBLUP using IBS (identical by state)
- Use external genomic relationship matrices for additive and dominance effects in blupf90 with `'user_file'` statement.

Lab 12 uses public genotypes and pedigree from:
Cleveland, Matthew A., John M. Hickey, and Selma Forni. "A common dataset for genomic analysis of livestock populations." G3: Genes, Genomes, Genetics 2, no. 4 (2012): 429-435.

Files are available in the folder day12. Copy the entire folder using the following command:

```
cp -r /work/course2018/week3/day12 .
```

1. Check the number of animals in the phenotype file:
   ```
   wc -l pheno.dat
   ```

2. Using awk, check the number of SNPs and animals with genotype:
   ```
   awk 'END {print length($2)}' snp.dat
   wc -l snp.dat
   ```

(A) Comparing SNP_BLUP and GBLUP

Read the R script GBLUPvsSNPBLUP.r

Open R and execute this script **step-by-step**

1. The SNP_BLUP (also known as Random Regression BLUP) assumes that variance components are known. Therefore, we need to determine the variance components, and in particular the SNP variance for BLUP_SNP. According to Gianola et al. 2009, this is (under some assumptions): $\sigma^2_{a0} \approx \frac{\sigma^2_u}{2\sum p_i q_i}$, then we need to find out $2\sum p_i q_i$. The allele frequencies are computed from genotypes.

Run BLUP_SNP and take a look at the SNP effects that are in `a=sol[2:neq]`

Note that GEBV is a matrix with 7 columns (number of methods that we will compare) and as many rows as animals. This matrix contains the genomic estimated breeding values (GEBV). The GEBV computed with SNP_BLUP are in the first column of the matrix `(GEBV[,1]=Z%*%a)`.

2. For computing GBLUP models, we will not use the inverse of G (to avoid the problem of G being non-invertible), we will use Harville's Mixed Model Equations (MME) with non-inverted G (singular G):

$$\begin{pmatrix} \boldsymbol{X'R^{-1}X} & \boldsymbol{X'R^{-1}WG} \\ \boldsymbol{GW'R^{-1}X} & \boldsymbol{GW'R^{-1}WG+G} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{b}} \\ \hat{\boldsymbol{\alpha}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X'R^{-1}y} \\ \boldsymbol{GW'R^{-1}y} \end{pmatrix}$$

where the GEBV are obtained as $\hat{\boldsymbol{u}} = \boldsymbol{G}\hat{\boldsymbol{\alpha}}$. Note that these MME are the same as the equations commonly used for RKHS (Reproducing Kernel Hilbert Space) models.

Run VanRaden's GBLUP 1 model (VR1). Here, G is scaled using current allele frequencies and variance components are known. The GEBV are put in the second column of the matrix `(GEBV[,2]=sol[2:length(sol)])`.

Note that allele frequencies enter at two levels in G matrix. First, when matrix Z is centered with $-2p_i$ and second, in the scale of G with $2\sum p_i q_i$.

In the next three models, Z matrix is centered with allele frequencies equal to 0.5 and the scaling changes as:
$2\sum p_i q_i$computes with current allele frequencies (VR05v1)
$2\sum p_i q_i$ computes assuming $p_i = q_i = 0.5$ (VR05v2)
$2\sum p_i q_i$ computes with $p_i = q_i = 0.5$ but with a different variance component (VR05v3). If $\sigma^2_{a_0} = \frac{\sigma^2_u}{2\sum p_i q_i}$ based on current allele frequencies (p_i, q_i), we have $\sigma^2_{a_0} = \frac{\sigma^2_{u^*}}{2*nSNPs*0.5*0.5}$ with allele frequencies at 0.5. So, the variance component $(\sigma^2_{u^*})$ is $\sigma^2_{u^*} = \frac{\sigma^2_u \ 2*nSNPs*0.5*0.5}{2\sum p_i q_i}$. Here, the "true" variance (σ^2_u) is "corrected" according to $2\sum p_i q_i$, and the results should be identical.

Run VanRaden's GBLUP models (VR05v1, VR05v2 and VR05v3) and check the results.

NOTE that the centering in Z matrix shifts the mean while the scaling modifies the variance component and implicitly the heritability. Scaling G by $2\sum p_i q_i$ and SNP variance $\sigma^2_{a_0}$ must be equivalent. This implies that depending how we construct G we will obtain different estimates of variance components and also for heritability.

In the last two GBLUP models, we use IBS (Identity by State) relationships and G_{IBS} is a genomic relationship matrix based on IBS at the markers. The G_{IBS} matrix is scaled by the number of markers $(nSNPs)$.

The first model uses a "naive" variance component (σ^2_u) (IBSnaive) and the second model uses $\sigma^2_{u^*} = \frac{\sigma^2_u \ nSNPs}{2\sum p_i q_i}$ (IBSfine).

NOTE that, using IBS relationships or genomic relationship estimates of breeding values will be identical, if associated variance components are comparable.

Finally, look at the correlation across all methods (cor(GEBV)), all of them except VR05v2 and IBSnaive are strictly identical. Note that the means are however shifted from one model to the other `(apply(GEBV,2,mean))`.

<u>In conclusion:</u>
All G matrices that have an underlying model y=markers +... are equivalent if the variance components are chosen correctly either analytically or estimated by REML.

(B) GD_BLUP

Go to the directory GD

Look at the Fortran program compute_GD.f90. This custom program computes additive and dominant genomic relationship matrices, G and D. The relationships in G follow VanRaden's (2008) first $\boldsymbol{G} = \boldsymbol{ZZ'}/2\sum_{all\ SNPs} p_i(1-p_i)$. Two dominant genomic matrices are used: first, dominance deviation genomic relationship matrix $\boldsymbol{D} = \boldsymbol{WW'}/\sum_{all\ SNPs}\left(2p_i(1-p_i)\right)^2$ (from the classical "statistical" model); and second, "genotypic" dominant matrix $\boldsymbol{D_D} = \boldsymbol{XX'}/\sum_{all\ SNPs}(2p_i(1-p_i)(1-2p_i(1-p_i)))$ (which was used by Su et al., 2012). Actually, to make G positive definite and invertible, the program computes $\boldsymbol{G} = 0.95\ \boldsymbol{ZZ'}/2\sum_{all\ SNPs} p_i(1-p_i) + 0.05\ \mathbf{I}$ (it is also used for $\boldsymbol{D}$ and $\boldsymbol{D_D}$).

Compile it:

```
ifort compute_GD.f90 -o compute_GD
```

Run it:

```
 ./compute_GD
 genofile?
snp.dat

 out file for G :
 snp.dat.G

 out file for D :
 snp.dat.D

 out file for DD :
 snp.dat.DD

 out file for G-1 :
 snp.dat.Gi

 out file for D-1 :
 snp.dat.Di

 out file for DD-1 :
 snp.dat.DDi

  G - VanRaden first G = ZZ'/sum(2pq)
  D - Vitezica D = WW'/sum(2pq)**2
  DD - Su DD = XX'/sum((2pq*(1-2pq))

 write out G ? (T:F)
T
 write out G inverse? (T:F)
T
 Column position in file for the first marker:           22
 …
 nanim=         3534 nsnp=        5285
```

It is possible to run the program using: `echo -e "snp.dat \n T \n T" | ./compute_GD`

It creates files, snp.dat.G, snp.dat.D and snp.dat.DD. Take a look. Let's see the aspect of diagonal and off-diagonal; if SNPs are in H-W equilibrium, they should average to 1 and 0 respectively. Take a look using awk for diagonal elements:

```
awk '$1==$2' snp.dat.G | awk 'BEGIN{r=0}; {r=r+$3}; END{print r/NR} '
```

and for the off- diagonal elements use:

```
awk '$1!=$2' snp_dat.G | awk 'BEGIN{r=0}; {r=r+$3}; END{print r/NR} '
```

After computing the genomic relationship matrices, we will use them. Variance components can be estimated using a "standard" BLUPF90 parameter file: `renf90_GD.par` with `user_file` as here:

```
RANDOM_GROUP
   2
RANDOM_TYPE
user_file
FILE
snp.dat.Gi
(CO)VARIANCES
   0.850
```

where the additive genetic effect is the second effect in the model and the inverse of its relationship matrix is in snp.dat.Gi file. Run this GBLUP model $\boldsymbol{y} = \boldsymbol{1\mu} + \boldsymbol{u} + \boldsymbol{e}$ with G coded as in VanRaden (2008) and estimate the additive genetic variance using airemlf90, saving the log file, as:

```
echo renf90_GD.par | airemlf90 | tee ai_gd.log
```

Run GDBLUP model $\boldsymbol{y} = \boldsymbol{1\mu} + \boldsymbol{u} + \boldsymbol{v} + \boldsymbol{e}$ and estimate the additive and dominant genetic variances.

Note that $Var(\boldsymbol{u}) = \boldsymbol{G}\sigma_u^2$ and $Var(\boldsymbol{v}) = \boldsymbol{D}\sigma_v^2$ are constructed following Z as in VanRaden and W as $\{-2q^2, 2pq, -2p^2\}$ for AA, Aa and aa genotypes, respectively. Note that this is the classical "statistical" parameterisation in terms of breeding values and dominance deviations. Remember to change BLUPF90 parameter file to include the dominance deviation relationship matrix, D, snp.dat.Di in user_file.

Which is the estimate for additive genetic variance?
Does the additive genetic variance estimate change between GBLUP and GDBLUP?

Run "genotypic" GDBLUP model (Su et al., 2012) $\boldsymbol{y} = \boldsymbol{1\mu} + \boldsymbol{u}^* + \boldsymbol{v}^* + \boldsymbol{e}$ and estimate the additive and dominant variances. In user_file of BLUPF90 parameter file put snp.dat.DDi

Compare the variance component estimates between GDBLUP and "genotypic" GDBLUP.

Note that the genotypic model underestimates additive variance and overestimates dominant variance as was shown by Vitezica et al. (2013). These variance components are not comparable to pedigree-based estimates.

ASSIGNMENT 13

- Comparison between BLUP and ssGBLUP
- Quality control for genomic relationships

The data for this lab was simulated using QMSim (Sargolzaei & Schenkel, 2009). A single trait animal model was simulated assuming heritability of 0.4. All the genetic variance was explained by 500 QTL. Animals were genotyped for 45,000 SNP and the average LD was 0.18. The simulated additive genetic variance was 0.40 and the residual variance was 0.60. The simulated phenotype was generated using the following model:

Phenotype = *sex_effect* + *true_breeding_value* + *residual*

Files are available in the folder day13. Copy the entire folder using the following command:

```
cp -r /work/course2018/week3/day13 .
```

Description of files:

data3.txt:
1: animal ID
2: generation
3: sex
4: phenotype
5: true breeding value (TBV)

ped3.txt:
1: animal ID
2: sire ID
3: dam ID

snp3.2k:
1: animal ID
2: SNP genotype

mrkmap.txt:
1: SNP ID
2: Chromosome
3: position

1. Copy the full folder into your directory
   ```
   cp -r /work/course2018/week3/day13 .
   ```

2. Modify an existent renumf90 parameter file (or create a new one), according to the data file, to fit the following model:

   $$y = sex + animal + e$$

3. Run renumf90 program to renumber the 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. What is the content of **snp3.2k_XrefID**?

5. Run preGSf90 including the option to save clean SNP file after quality control. Check the output. Which quality checks for both SNP and animals were done by default? Are there any duplicated genotypes? What is the correlation between G and A_{22}? Check averages of G and A_{22}.

6. Working with the clean SNP files, add an option to create PCA plots and run preGSf90 again. As your SNP file is already clean, do not forget to include an option to skip quality control. Based on the PCA plot, what can you conclude about the population structure?

7. Run blupf90 without SNP information. Now run blupf90 using genomic information and compare cpu time and solutions.
   Hint: use the following command to provide computing time and to save outputs to a log file:
   ```
   time echo renf90.par | blupf90 | tee blup1.log
   ```

8. Do a validation on young selection candidates or individuals from 5th generation with genotypes and no phenotypes. Compare EBV and GEBV with true breeding value (TBV). Remember that correlation between (G)EBV and a benchmark (i.e., TBV) is a measure of accuracy. What happened with accuracy when genomic information was included? Check also intercept and regression coefficient from a regression of TBV on EBV and GEBV.

   Hint 1: remove the phenotypic information from the 5th generation and obtain solutions from a model with SNP information and with no SNP information.
   Hint 2: have renumf90 passing to the renumbered data a column containing generation number.
   Hint 3: if generation column is number 4, new data can be created using the AWK Linux tool:
   ```
   awk '$4!=5' renf90.dat > renf90.dat.reduced
   ```

9. A very common validation method used in beef cattle and other species is the correlation between phenotypes adjusted to fixed effects and EBV or GEBV. This is called predictive ability or ability to predict future performance. Compute predictive ability for young genotyped animals in the 5th generation.

   Hint 1: the benchmark is now adjusted phenotypes obtained using the complete data and no genomic information. Run blupf90 with complete data and no genomic information. Run predictf90 in the same folder you ran blupf90. Before running, you should include the following option in the parameter file:
   OPTION include_effects X
   Where X is the number of the animal effect. If animal effect is effect number 2 in your model, X is 2. This means that phenotypes will be adjusted for all effects, but effect number 2. Adjusted phenotypes will be in a file called yhat_residual, with the following format:

   Animal_id, Y*, Yhat, residual
   where: Y* = Phenotype – μ
   Yhat = EBV (or animal effect)
   Residual = Phenotype - EBV

   Hint 2: Correlate Y* with EBV and GEBV computed using reduced data.

OPTIONAL

10. Comparing EBV "before" and "After" – Assuming true breeding values are not known, as in real populations, do a validation based on the Method LR. Check correlations, intercept, and regression coefficient.

    Hint: The method compares predictions with all data (whole, subindex w) and partial data (subindex p). Comparing the EBVs (u_w) of the same animals using either whole (u_w) or partial (u_p) yields statistics that are approximations to bias, slope, and ratios of accuracies:

    Bias: $\mu_{wp} = \overline{\hat{\mathbf{u}}_p} - \overline{\hat{\mathbf{u}}_w}$

    Slope or dispersion (also called b_1 and sometimes also called bias): $b_{w,p} = \frac{cov(\hat{\mathbf{u}}_w, \hat{\mathbf{u}}_p)}{var(\hat{\mathbf{u}}_p)}$

    Accuracies:

    $\rho_{p,w} = \frac{cov(\hat{\mathbf{u}}_p, \hat{\mathbf{u}}_w)}{\sqrt{var(\hat{\mathbf{u}}_w) var(\hat{\mathbf{u}}_p)}}$ is an estimator of the ratio of accuracies using the "partial" or the "whole" data $\frac{acc_p}{acc_w}$

    $b_{p,w} = \frac{cov(\hat{\mathbf{u}}_w, \hat{\mathbf{u}}_p)}{var(\hat{\mathbf{u}}_w)}$ is an estimator of the square of the ratio of accuracies $\left(\frac{acc_p}{acc_w}\right)^2$

ASSIGNMENT 14

- Accuracy of (G)EBV based on the inverse of LHS
- SNP effects and Indirect Predictions
- GWAS

The data for this lab was simulated using QMSim (Sargolzaei & Schenkel, 2009). A single trait animal model was simulated assuming heritability of 0.40. All the genetic variance was explained by 500 QTL. Animals were genotyped for 45,000 SNP and the average LD was 0.18. The simulated additive genetic variance was 0.4 and the residual variance was 0.60. Phenotype was generated using the following model:

Phenotype = sex_effect + true_breeding_value + residual

Description of files:

data3.txt:
1: animal ID
2: generation
3: sex
4: phenotype
5: true breeding value (TBV)

ped3.txt:
1: animal ID
2: sire ID
3: dam ID

snp3.2k:
1: animal ID
2: SNP genotype

mrkmap.txt:
1: SNP ID
2: Chromosome
3: position

1. Log into dodo2.ads.uga.edu. Go to your work directory. To do that, type: `cd /work/ads-guestXX` where XX is your user number.

2. Files are available in the folder day14. Copy the entire folder using the following command:
   `cp -r /work/course2018/week3/day14 .`

3. Run `renumf90` program using `renum.par` parameter file to renumber the data.

4. Run `blupf90` with and without genomic information, including the option to obtain standard error (SE). Based on SE, calculate accuracy of EBV and GEBV using bash script (e.g., `awk`). Compare average accuracy of EBV and GEBV for genotyped animals. Are they different? Why?

   Hint: if trait is 1 and the animal effect is the second effect in the model, accuracy can be calculated as:

```
awk '{if ($1==1 && $2==2) print $3,$4,(1-($5*$5)/0.40)**0.5}' solutions > sol_acc
```

5. Calculating SNP effects in ssGBLUP: given that SNP effects are calculated based on GEBV, run `blupf90` to get solutions. Before that, check the options you will need to include in the parameter file, so `blupf90` can provide all the files needed for the calculation of SNP effects (in exercise 5). Turn off the intrinsic quality control.

6. Add an option to read a map file (mrkmap.txt) and run `postGSf90`. Check the output files and the content of each column (http://nce.ads.uga.edu/wiki/doku.php?id=readme.pregsf90).

7. Indirect predictions for young individuals: `postGSf90` creates a file `snp_pred` with information about the random effects (number of traits + correlated effects), the gene frequencies and the solutions of SNP effects. This is the file used by `predf90` to provide indirect predictions for young genotyped animals as `Za`, where `Z` is a matrix of SNP content and `a` is a vector of SNP effects. Run `predf90` to get indirect predictions for young individuals that were not included in the `blupf90` and `postGSf90` runs. Genotypes for young animals are in a file called `new_animals`

8. Weighted ssGBLUP (WssGBLUP): in weighted ssGBLUP, SNP effects are used to compute SNP weights. The default way to calculate SNP weight (`w`) in postGSf90 is:

$$w_i = 2p_i(1-p_i)a_i^2$$

where `p` is the allele frequency and a is SNP effect. A new method that has better convergence properties has been recently implemented in `postGSf90`. This method is called nonlinearA and is described in VanRaden (2008) as:

$$w_i = \mathrm{CT}^{\frac{|\widehat{a_i}|}{sd(\hat{a})}-2}$$

where CT is a constant set to 1.125, and $\frac{|\widehat{a_i}|}{sd(\hat{a})}$ is capped to 5 by default. To use this method, the following option should be used in `postGSf90`: `OPTION which_weight nonlinearA`.

Run postGSf90 including an option to calculate variance based on a windows of 20 SNPs and an option to generate Manhattan plots. Use the default linear weight and the nonlinearA weight. Check the output files and compare results.

Hint 1: `postGSf90` can read a file with weights for each SNP. By default, all SNP have the same weight, so this file is actually a vector of dimension N x 1; where N is the number of SNP. If there are 50,000 SNP, the following `awk` command will create a vector of dimension 50,000 x 1:

```
awk 'BEGIN { for (i==1;i<50000;i++) print 1}' > w.txt
```

Hint 2: `postGSf90` prints Manhattan plots on the screen and also creates files for printing in R (Sft1e2.R and Vft1e2.R) and in Gnuplot (Sft1e2.gnuplot and Vft1e2.gnuplot).

9. Iterative WssGBLUP: In exercise 6, you ran only one iteration of weights. However, WssGBLUP is an iterative method, where weights are used to construct the weighted genomic relationship matrix $\mathbf{G_w} = \frac{\mathbf{ZDZ'}}{2\sum p_i(1-p_i)}$, and this matrix is used to compute new GEBV. Usually, 3 to 5 rounds are enough to obtain convergence (no changes in weights in iteration t compared to t-1). Run one more round of `blupf90` and `postGSf90` for both linear and nonlinearA methods. Compare Manhattan plots and maximum variance explained.

Hint: updated weights are in column 7 of snp_sol from exercise 6. Use the following Unix command to create the new weight file (dimension N x 1) to be used in exercise 7.

```
awk '{ if ($1==1) print $7}' snp_sol > W
```