Introduction to BLUPF90 suite programs
Concise version

Yutaka Masuda
University of Georgia

March, 2019

Acknowledgment

Dr. Andres Legarra has reviewed the draft and made a massive amount of corrections and improvements on it. He also wrote sections *Renumbering the data without RENUMF90*, *EM-REML algorithm*, *Likelihood ratio test*, and several other subsections.

I would like to express my acknowledgment to faculty, postdocs, students, and visitors in the Department of Animal and Dairy Science at the University of Georgia for their questions about the BLUPF90 programs, which encourages me to decide to write this note.

Contents

1 Introduction

1.1 Short introduction to BLUPF90 programs

1.1.1 What is BLUPF90?

BLUPF90 is a name of software. Also it is a name of a collection of programs derived from BLUPF90. In the latter case, we will refer it to *BLUPF90 family* or *BLUPF90 programs*. A concise description is also found at the official wiki at the University of Georgia (`http://nce.ads.uga.edu/wiki/doku.php`).

> BLUPF90 family of programs is a collection of software in Fortran 90/95 for mixed model computations in animal breeding. The goal of the software is to be as simple as with a matrix package and as efficient as in a programming language.

The BLUPF90 program creates and solves the mixed model equations. It supports various models including animal model, maternal model and random regression model with multiple traits. The BLUPF90 family has several programs for variance component estimation using REML and Gibbs sampling with many kinds of models. The programs are freely available at the official website (`http://nce.ads.uga.edu`) and can be freely used for academic or research purposes.

1.1.2 Why BLUPF90?

BLUPF90 has several advantages for users: simplicity, stability and active development. For programmers, the internal structure is documented in a course note (`http://nce.ads.uga.edu/wiki/doku.php?id=course_information_-_uga_2018`) and the working source code is available; so a developer can modify the program to support new ideas.

We will see the advantage in a user's point of view.

Simplicity

The program's behavior is very simple. Every BLUPF90 program reads a parameter file, which describes the name of data and pedigree files, models and (initial) variance components to be used in the analysis. The parameter file is a short text file and it has a few pairs of keywords and values to describe the information. The parameter file is concise but capable of general models. Once you learn how to write a parameter file, you can perform very complicated analyses with the program.

Every program saves the solutions of the mixed model equations (e.g. EBV) to a file. Estimated variance components are also saved in files permanently.

Stability

The programs have been tested by many researchers since its public release around 2000. The programs are now stable enough to be used in a routine genetic evaluation at national level. The team at the University of Georgia heavily uses the programs.

Active development

The programs are continuously maintained by the UGA team. New features are added to the existing software to incorporate a new methodology and to improve the usability and the computing time. Single-step genomic BLUP (ssGBLUP) is fully supported and GWAS can be done with the ssGBLUP framework.

Speed

We have been optimizing the programs in terms of speed not only using multithreaded libraries (MKL) but also with detailed optimization in fundamental subroutines. The current version is remarkably faster than the old version especially in REML and Gibbs sampling.

1.1.3 Is it easy?

Yes, it is. But learning process is not always easy. There might be several hurdles to learn how to use BLUPF90 programs in an actual research. It is a opposite side of the advantage.

- A little documentation. The official website hosts a manual for the programs (`http://nce.ads.uga.edu/wiki/doku.php`). The official wiki also provides various information about the software. But, documentation especially for learners is not fully available in the existing resources.
- Data manipulation. This is not a specific disadvantage in BLUPF90 but in many similar software. BLUPF90 family focuses on mixed model analyses so it doesn't provide any data manipulation framework as R and SAS do. The programs use text files as common format so a few text manipulation can be needed. You should prepare data and pedigree with a specific format and check erroneous records in the files before running the programs. In some cases with RENUMF90, you would combine the original code with the solutions somehow.
- Pre-process. Every program accepts numerical values only because of keeping simplicity in programming. If your data or pedigree file contains characters (alphabets or symbols), you should replace the characters with integer code before the analysis. One of the programs, RENUMF90, can perform this editing. With field or commercial data, you will have to run RENUMF90 before BLUPF90.

The purpose of this tutorial is to provides such document with many examples.

1.2 Differences with other software

It is better for a new user to know the differences between BLUPF90 programs and another software with which you are familiar. We briefly introduce the differences for your understanding.

1.2.1 General differences

R and SAS

- BLUPF90 can handle really large data sets. BLUPF90 is most likely faster than R and SAS.
- BLUPF90 is not interactive software. A user should prepare a *parameter file* that describes the details of analysis (names of files, model, genetic parameters and options). The program simply reads the file and run a defined task.
- BLUPF90 is not a scripting language. The programs don't provide any features for data manipulations. All the editing should be done before invoking the program. The data file must contain all information needed for the analysis.
- BLUPF90 can only read text files. Headers (or comments) are not allowed in the data or pedigree files. The items in a text file should be separated with one or more spaces.

- BLUPF90 shows the results on screen as well as text files.
- BLUPF90 accepts only integer values as levels of effects in the data and pedigree file, and real values as covariables and trait values. Alphabets or symbols should be replaced with integer code before the analysis. The RENUMF90 helps this process.
- BLUPF90 has no window system. It is like a Rscript, a command line utility. Basically, the program runs on Command Prompt (or 'DOS' window) in Windows, Terminal on MacOS X and various shell on Linux.
- BLUPF90 has no functions to perform hypothetical tests. Some programs show $-2\log L$ or similar statistics so you can manually perform a likelihood ratio test.

ASREML

- BLUPF90 is free for research and academic purposes but has limited support from the development team.
- BLUPF90 supports Gibbs sampling.
- BLUPF90 covers linear mixed models and the threshold models. It doesn't have procedures for generalized linear models.
- BLUPF90 uses simple text format for data and pedigree files. No headers, no comments, no alphabets or symbols are allowed in the files. The edited data should be prepared by a user before the analysis.
- BLUPF90 has no comprehensive directives for the analysis. Parameter file contains minimal information including file names, model, variance components and options. BLUPF90 doesn't use labels to refer to the effects in the model description.
- BLUPF90 generates minimal output.

WOMBAT

- In the data file for BLUPF90, observations for an animal taken at the same time should be basically listed in the same line i.e. multiple observations should be saved in the same line. The case, where observations from the same animals are stored in 2 or more lines, will be allowed only in repeated records or a special multiple-trait model (e.g. G-by-E analysis). In WOMBAT, each trait is on a different line.
- BLUPF90 doesn't compute by default the inbreeding coefficients. The inverse of numerator relationship matrix ($\mathbf{A}^{-1}$) is created based on non-inbred approach (i.e. Henderson's method pretending that there is no inbreeding). To consider inbreeding in $\mathbf{A}^{-1}$, a user should supply inbreeding coefficients as a special format to the pedigree file as an additional information.
- BLUPF90 doesn't use labels to describe the model. Instead, the program directly uses the positions (columns) of effect identifiers.
- BLUPF90 doesn't create interaction effects in the program i.e. it has no $*$ description in the parameter file. Interactions should be prepared as cross-classified effects before the analysis.
- BLUPF90 accepts positive and negative reals as covariates, as opposed to WOMBAT which only accepts integers.
- WOMBAT has more options for maximization of the likelihood (Derivative Free, PX, reduced rank), whereas BLUPF90 has Bayesian estimation by Gibbs sampler
- BLUPF90 programs deal correctly with threshold traits. WOMBAT does not.

VCE

- The data and pedigree files for BLUPF90 shouldn't contain headers (or comments). The first row in the file must be a data line.
- BLUPF90 uses very simple parameter file that is not as readable as ones for VCE. BLUPF90 doesn't use labels to describe effects. The order of parameters in the file is fixed.

- Pedigree file contains animal ID, sire ID and dam ID as the first 3 columns for animal model. The additional 4-th column can be used as a special feature.
- BLUPF90 doesn't compute by default the inbreeding coefficients. The inverse of numerator relationship matrix ($\mathbf{A}^{-1}$) is created based on non-inbred approach (i.e. Henderson's method pretending that there is no inbreeding). To consider inbreeding in $\mathbf{A}^{-1}$, a user should supply inbreeding coefficients as a special format to the pedigree file as an additional information.

DMU

- BLUPF90 directly reads the data and pedigree files i.e. a user doesn't have to run DMU1 type of software. Data preparation programs can be optionally used for a user's convenience.
- Parameter file (the same concept to driver file) contains model descriptions. BLUPF90 doesn't use labels to refer to effects.
- BLUPF90 doesn't care about data types (integer or real). All the data will be read as real values and converted into appropriate type as needed.
- BLUPF90 prints minimal outputs on screen. It doesn't create any log files.

1.2.2 Difference in software design

The philosophy of BLUPF90 programs is described in the official wiki and several publications. The basic idea is to support general linear mixed models with the minimal effort in programming. Fortran 90 enables us to write and re-use the code easily. BLUPF90 is the primary software to demonstrate that this idea actually works well with one common code. Other application programs have been derived from BLUPF90.

The current version of BLUPF90 family of programs have several advantages over other software. The programs now support genomic analyses especially for single-step GBLUP. Computing time has been greatly improved in REML, Gibbs sampling and BLUP with iteration on data, using parallelization and optimized libraries. The programs are actively updated to incorporate new idea and improve stability.

1.3 About the software

1.3.1 Condition of use

The detailed information about the condition of use and the support, please visit the official wiki (`http://nce.ads.uga.edu/wiki/doku.php`). The programs are free for research but their use should be acknowledged in publications. For use of non-research, please contact Ignacy Misztal (`ignacy@uga.edu`).

1.3.2 Bug report and support

A bug report is welcome. Please contact one of the developers at UGA or the other institutes. Or, please post the report to Yahoo Group (`https://groups.yahoo.com/neo/groups/blupf90/info`). A limited support may be available at Yahoo Group because the support is volunteer-based. If you need a full support about the programs, please consider to visit our team.

1.3.3 Citations

You can cite the following publications when you used the BLUPF90 programs in your study.

- Misztal, I., S. Tsuruta, D. A. L. Lourenco, Y. Masuda, I. Aguilar, A. Legarra, Z. Vitezica. 2018. Manual for BLUPF90 family programs. University of Georgia. `http://nce.ads.uga.edu/wiki/doku.php?id=documentation` (This is the standard reference to be usually cited by any users.)

If you cite this tutorial in your publication, please use the following reference *in addition to the reference to the manual shown above.*

- Masuda, Y. 2018. Introduction to BLUPF90 suite programs. University of Georgia. `http://nce.ads.uga.edu/wiki/doku.php?id=documentation`

If you have a reason to cite a particular feature in BLUPF90 programs, the following references can be useful.

- Aguilar, I, S. Tsuruta, Y. Masuda, D. A. L. Lourenco, A. Legarra, I. Misztal. 2018. BLUPF90 suite of programs for animal breeding with focus on genomics. No. 11.751. The 11th World Congress of Genetics Applied to Livestock Production, Auckland, New Zealand. (This is the newest reference for BLUPF90 programs focusing on genomic analysis.)
- Aguilar, I., I. Misztal, S. Tsuruta, A. Legarra, H. Wang. 2014. PREGSF90 - POSTGSF90: Computational Tools for the Implementation of Single-step Genomic Selection and Genome-wide Association with Ungenotyped Individuals in BLUPF90 Programs. The 10th World Congress of Genetics Applied to Livestock Production, Vancouver, BC, Canada. (This can be cited if you used a combination of PREGSF90 and POSTGSF90 to perform single-step GWAS.)
- Misztal, I., S. Tsuruta, T. Strabel, B. Auvray, T. Druet, D. H. Lee. 2002. BLUPF90 and related programs (BGF90). Communication No. 28-07. The 7th World Congress on Genetics Applied to Livestock Production, August 19-23, 2002, Montpellier, France. (This used to be widely referred but it is too old to cite. Use this reference only if you have a reason to exclusively cite it.)

The following references will be cited in technical descriptions in the subsequent chapters.

- Misztal I. 2018. Computational techniques in animal breeding. 2018. Course Note. University of Georgia.

1.4 About this tutorial

1.4.1 General information

This document explains how a model is described in a parameter file used with the BLUPF90 family programs, how data or pedigree files are prepared, and how genomic analyses perform with the programs. So this tutorial assumes that a reader has enough knowledge in linear mixed models and data manipulation techniques on a computer. Experiences with similar software will be very helpful to understand.

Features of BLUPF90 program are introduced in an introductory style. Especially, the first section in each chapter will introduce essential features using a very simple example. In the later sections, new ideas will be gradually explained also with example files. So we recommend a reader not to skip any sections even if it looks too elementary for the reader.

This tutorial is a complement of the official manual or wiki pages. The official manual is more like a reference manual so you can immediately find specific topic written in concise words. Also if you want to know advanced topics or tricks, please check the manual. Conversely, if you found something missing in the manual, this tutorial would help you understand the background. If you have a question not covered with the manual, wiki nor this tutorial, please ask somebody at the Yahoo Group (`https://groups.yahoo.com/neo/groups/blupf90/info`). You can find right answers from a historical log of this forum.

1.4.2 Examples in each section

In each section, example files are available. See the following example. The file name example.txt below is actually a link to the file (only when you read this article in Dokuwiki).

Listing 1.1: example.txt

```
1 1 2.5
2 1 1.8
3 1 4.2
4 1 2.2
5 1 3.6
```

Parameter files used in the BLUPF90 programs will be highlighted as follows.

Listing 1.2: param0.txt

```
DATAFILE
data0.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
1
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS:
2 3 cross
RANDOM_RESIDUAL VALUES
1.0
```

Output from the programs is encircled with a different format.

```
name of parameter file?
```

1.4.3 Printed version

A PDF version is separately supplied by our team. Or, you may find it on our website `http://nce.ads.uga.edu/wiki/doku.php?id=documentation`. The documentation is automatically generated with a script so some typeset errors would happen.

1.4.4 Disclaimer

The disclaimer is taken from the MIT License (although this tutorial is NOT distributed under the MIT License).

> This tutorial is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the tutorial or the use or other dealings in the tutorial.

2 Download and Installation

This chapter covers how to install and run the programs. We also introduce methods to save the screen message (output log) and to omit the key-in in the beginning of a program. A text editor is helpful to work with the files.

2.1 Availability

The BLUPF90 family programs are available at the official web site of the Animal Breeding and Genetics Group at the University of Georgia. This web site provides the latest version of the programs. See the following link.

- Animal Breeding and Genetics Group – University of Georgia (`http://nce.ads.uga.edu/`)

You can download the programs and use them for personal or academic purposes. Some pro- grams are not available on line. These programs are provided under a contract with our team. See the official webs site for details.

Now an official manual for a series of the programs is available at the official web site. The manual provides detailed options to control the programs as well as demonstrations on analyses with various linear models.

2.2 Running a program in Windows

BLUPF90 programs don't show a graphical window and the software looks like a old DOS-type program. You should prepare a parameter file to describe the name of data and pedigree files, the model and variance components to be used in the analysis. You will type-in the name of the parameter file on screen. If you have an experience on such software, you can skip this section.

We recommend to use a standard (traditional) way to run the program. If you aren't really interested in a command line interface in Windows, just check the last subsection. We provide a quick way to use the program.

2.2.1 A basic procedure

If you have no experience to run a command line program...

The running a command-line program is actually a FAQ. The formal way is to use the Command Prompt or simply cmd. This is a simple console to accept commands typed with a keyboard. You can see several tutorials on this topic.

- Command Prompt - How to Use Basic Commands (`http://www.digitalcitizen.life/command-prompt-how-use-`
- How to use the Window command line (DOS) (`http://www.computerhope.com/issues/chusedos.htm`)
- How do I run a file from MS-DOS? (`http://www.computerhope.com/issues/ch000598.htm`)
- Beginners Guides: Windows Command Prompt (`http://www.pcstats.com/articleview.cfm?articleID=1723`)
- How To Open Command Prompt (`http://pcsupport.about.com/od/commandlinereference/f/open-command-prompt.htm`)

How to run the program

A basic procedure to run the program is following:

1. Download the program and store it to a folder. You can add a folder to PATH (if you are unsure of PATH, just ignore this step).
2. Save the needed files into a folder (If you are unsure of PATH, the downloaded program should be placed in the same folder).
3. Open the Command Prompt window and change directory to the folder with the cd command.
4. Type the name of program (e.g. blupf90 or blupf90.exe) to invoke the software.
5. Type the name of parameter file. Some programs would ask you to type in additional information.
6. Wait for finishing the analysis.
7. Collect the results.

If you successfully run a program, it asks you the name of parameter file with a short message.

```
name of parameter file?
```

Here you type the name of parameter file with keyboard. With Gibbs sampling programs, additional questions will be shown. You can also type the answers there.

Save the output

If you want to save the output log (screen messages out of the program), please use the redirection.

```
blupf90 > out.txt
```

With this command, nothing shows but it accepts key types. You can type the name of parameter file and it runs. The output is going to the file out.txt.

Omit the typing

If you don't like to type-in, you can also use the redirection. You prepare a text file with the name of parameter file:

Listing 2.1: in.txt

```
parameter.txt
```

Let's say this file is in.txt then you can invoke the program.

```
blupf90 < in.txt > out.txt
```

This technique is useful for Gibbs sampling programs that needs several key-in in the beginning of the program. You can write several lines in in.txt instead directly typing-in.

Stop the program

If you stop the program immediately, hit 2 keys simultaneously: Ctrl and C. For easier operation, you can keep pushing Ctrl and then hit C. The program immediately stop with some messages (you can ignore them). In some cases with multi-threaded computations, the program will be still running after receiving the STOP signal. You can close the Command Prompt, and the program completely stops.

2.2.2 Quick run

You can run the program without Command Prompt.

1. Save all required files in a folder.
2. Download the program and store it to the same folder.
3. Double click the program. A black window is popping up.
4. Input the name of parameter file in the black window using the keyboard. If the program asks you more questions, type the answers.
5. Wait for finishing the analysis. Don't close the black window. When the analysis ends, the windows will automatically disappear.
6. Check the results. The program saves the results as files in the same directory.

If you want to save the screen (`output.log`), download the following file (`run.bat`) and save it in the folder. Then open this file with Notepad and rewrite the name of program as you like and save it. Double click the run.bat. Although a black window appears and nothing will show in the window, you can type the name of parameter file. The program implicitly runs and all the output is going to `out.txt`.

Running the program double-clicking on it is less recommended than the Command Line, because if there is any error the window will close automatically and the error may not be read by the user.

Listing 2.2: run.bat

```
blupf90 > out.txt
```

This procedure may not work in some cases. The reason will be due to the hidden-extension stuff. We don't support this method at all.

2.3 Running a program in Linux and MacOS X

2.3.1 Command line software

BLUPF90 programs are command line software. The program reads a parameter file describing the name of data and pedigree files, the models and covariance components used in the analysis. It shows logs on screen and writes a file (or several files) for the results. This is designed to be used on the shell with keyboard operations.

Here we assume a standard bash environment. For MacOS X, all the operations will be done on Terminal. No GUI will be used.

2.3.2 Installation

The software doesn't need a special installation method. Just download it from the official web site. Then change the permission e.g. `chmod 755` or `chmod u+x` (easier to remember: *ch* ange the *mod* e adding (+) permission of e *x* ecution to the *u* ser). You can move the program to your favorite directory which may

be listed in the variable PATH. If you don't have such a directory, you can create the directory. Then you move the program there.

The following is an example for this process.

```
# 1. download a program; you can use another way like using curl.
wget http://nce.ads.uga.edu/html/projects/programs/Mac_OSX/new/blupf90

# 2. change the permission to executable
chmod 755 blupf90

# 3. make a directory for binaries if you don't have it.
mkdir ~/bin

# 4. move the program to the directory.
mv blupf90 ~/bin
```

If above directory is not listed in the environmental variable PATH, add the directory to this variable. You can check it with a command.

```
echo $PATH
```

If you find the directory, you don't have to do any extra task. If you don't find, you can change the variable. You can do this by typing the following command on your shell.

```
export PATH=~/bin:$PATH
```

If you have installed the program in another directory than ~/bin, please put the correct path there. This setting will disappear after log-out. To execute this command on log-in, set the variable in /.bash_profile. If you are unsure of this file, just open it with a text editor and add the above command in the bottom of the file. Do not forget to log-out once then log-in again because the setting file will be effective only when you log-in the system.

2.3.3 Running a program

Move to the directory with all the required files (e.g. pedigree, data). Just type the program name to invoke the software. If you successfully run a program, it asks you the name of parameter file with a short message.

```
name of parameter file?
```

Here you type the name of parameter file with keyboard. Additional typing will be needed for Gibbs sampling programs. The messages will be shown on screen (standard output) so you can capture them to a file.

```
# all messages are saved to a file .
# The program accepts key-in even nothing shown .
```

```
blupf90 > out.txt

# as above
# messages are shown and saved .
blupf90 | tee out.txt
```

Using a redirection, you can omit the key-in in the beginning of the program.

```
# one line input
echo parameter.txt | blupf90

# different way using a file
echo parameter.txt > input
blupf90 < input
```

The second method is useful for Gibbs sampling programs, which require several inputs.

2.3.4 Stop a program

If you stop the program immediately, hit 2 keys simultaneously: Ctrl and C. For easier operation, you can keep pushing Ctrl and then hit C. The program immediately stop with some messages (you can ignore such messages). In some cases with multi-threaded computations, the program will be still running after receiving the STOP signal. You can close the terminal, and the program completely stops.

2.4 Additional settings

The BLUPF90 programs may use a lot of memory especially when the data set is large and a genomic model is applied. The big memory-consumption often results in a program crash with the error *segmentation fault* (or *bus error*). Segmentation fault occurs when a program tries to access an inappropriate memory area or to access it with abnormal way. Note that this error may happen even if your computer has a lot of free memory. The operating system (Linux, macOS, or Windows) has a limitation of memory consumption by a program. Strictly, the system limits the amount of memory called *stack* and our programs consume a lot of stack to get the maximum performance.

It is a common error with a big data set, you may hit it when you update the program (i.e. when using a new version) with the same files used as before. A small modification in the code may change the memory-management strategy in the program.

You can remove the limitation or increase the stack size. The following settings are always recommended even if you do not hit this issue. The details are also available on our website: `http://nce.ads.uga.edu/wiki/doku.php?id=faq.segfault`.

2.4.1 Increase stack size

This setting is required in Linux and macOS. First type the following command in your shell (terminal).

```
ulimit
```

If it shows `unlimited`, the configuration looks good. If you see a number (like 8192), probably it is a problem. This number is the stack size and it should be *unlimited*. To change the size, type the following command before running our programs.

```
ulimit -s unlimited
```

You can put this command at the end of `.bash_profile` in your home directory. In this way, you can load this setting whenever you log-in this system.

2.4.2 Increase stack size for OpenMP

There is a separate setting to define stack size for OpenMP. This value is unrelated to the system stack size explained above. It is defined with an *environment variable*.

Linux and macOS

Please type the following command in your shell.

```
echo $OMP_STACKSIZE
```

If it shows nothing, it may be a problem. Even if you have a number with a unit (like 4M for 4 megabytes), it may be small. By default, this value is 4M and most likely it is too small. Please type the following command before running our program. Do not put any spaces around =.

```
export OMP_STACKSIZE=64M
```

This put 64 megabytes to stack size. If the program still stops with the same error, please increase the number gradually (like 128M, 192M, and so on). A too big value will consume a lot of memory because each thread can use this amount of memory. It is hard to tell what is suitable for the user; it is system-dependent.

If you want to change it temporarily (one-time-run), you can put it to the command line when you run the program (no export). In this way, you can find a reasonable setting empirically.

```
OMP_STACKSIZE=64M ./blupf90
```

As `ulimit`, you can save it in `.bash_profile`.

Windows

You can set the environment variable OMP_STACKSIZE in Command Prompt (for a temporary change) or Control Panel/system-configuration page (for the permanent change). To see the current stack size for OpenMP, you can type the following command in Command Prompt.

```
set %OMP_STACKSIZE%
```

To put the value to this variable, type the following command (64 megabytes in this case).

```
set OMP_STACKSIZE=64M
```

If you close this window, the setting is lost. So, you have to run BLUPF90 programs in this session just after typing this command. If you make this permanent, use the Control Panel. For details, please search the keywords like `windows, environment, variable` on the Internet by search engines.

The stack size for OpenMP should be the number plus unit like 64M for 64 megabytes. First, try 64M, and if you still see the problem, increase the number to 128M or more. Too big number will consume a lot of memory because each thread can use this amount of memory. The suitable value is up to your computer, so please find it empirically.

2.5 Setup a text editor

BLUPF90 programs uses various text files. A text file contains only characters (basic alphabet, numbers, and symbols). Microsoft Word can read a text file but it is too heavy to edit the file. Text editor is a category of software specialized to edit a text file. A text editor usually supports many functions useful for the edit. Some of them can open a very large text file.

Typical text editors are *Notepad* (Windows), *TextEdit* (MacOS X) and *vi*, *Emacs*, or *nano* (Linux/Unix). They are the default editor in each system but you can install other editors for your convenience. Especially in Windows, Notepad has too few options to be of practical use. There are so many free text editors developed in Windows. Here we show several well-known free text editors. You can test each editor and choose your favorite one.

- Notepad++ (`https://notepad-plus-plus.org`)
- SciTE (`http://www.scintilla.org/SciTE.html`)
- Atom (`https://atom.io`)

Several web sites maintain a list of text editors. A page in Wikipedia (`https://en.wikipedia.org/wiki/Comparison_of_text_editors`) is one of them. You can also check such a site.

You may need to work with the programs remotely (you access a server over the internet and run the program on the server). In that case, there are several ways to edit text files. We assume the remote server is running Linux (or Unix-like environment).

1. Use an editor on the server (perhaps vi or Emacs). You edit a file on a terminal window.
2. Use an editor on the server but show the window on your system with X forwarding (perhaps GEdit or similar GUI editor). The editor looks working on your local computer but is actually working on the server; the editor can read and write a file on the server directly. A web site (`http://pdc-amd01.poly.edu/Xhowto.html`) explains a setting to do this.
3. Use a local editor which can access a file on the server remotely. A few editors provide such a feature.

3 Quick tour of BLUPF90

In this chapter, we perform several analyses with simple linear models using BLUPF90. The program creates and solves the mixed model equations. These small examples will provide you some basic ideas how to prepare a data set, to write a parameter file, and to run the program. Let's start with this chapter even if you are actually interested in another software or more complicated models.

3.1 Trivial analyses for fixed models

This section describes the basic design and essential elements of BLUPF90 programs. We will explain how to prepare a data file and how to describe a parameter file using simple fixed effect models. Almost of basic concepts will be introduced here. The concept is easily expanded to more complicated models.

We use the BLUPF90 program here. BLUPF90 quickly solves mixed model equations and save the solutions to a file. This program also accepts the least square equations (i.e. normal equations) for fixed effect models. Other programs for variance component estimation (e.g. AIREMLF90 and GIBBSF90) also accepts the same parameter file.

BLUPF90 needs at least 2 files: data file and parameter file. The data file contains observations and related information (animal ID, age, sex, month etc.). The parameter file defines the model, the name of data file, genetic (and environmental) variance components, and optionally the name of pedigree file. The files should be prepared by a user. Here we start with a linear fixed model with 1 class variable; this is equivalent to 1-way ANOVA (with unbalanced data).

3.1.1 One-way ANOVA model

Model

First, we describe the statistical model for the 1-way ANOVA:

$$y_{ij} = A_i + e_{ij}$$

where y_{ij} is an observation, A_i is a fixed effect, and e_{ij} is residual effect. The corresponding matrix notation is

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

and its generalized least squares equations are

$$\mathbf{X}'\mathbf{R}^{-1}\mathbf{X}\hat{\mathbf{b}} = \mathbf{X}'\mathbf{R}^{-1}\mathbf{y}$$

where $\mathbf{R} = \mathrm{var}(\mathbf{e}) = \mathrm{var}(\mathbf{y})$. We actually assume $\mathbf{R} = \mathbf{I}\sigma_e^2$ with a residual variance σ_e^2, so the above equations reduce to ordinary normal equations: $\mathbf{X}'\mathbf{X}\hat{\mathbf{b}} = \mathbf{X}'\mathbf{y}$. The purpose of this example is to solve the equations with BLUPF90. The program doesn't perform any hypothetical tests.

Data

In this analysis, we need at least 2 information: 1) observations and 2) code to distinguish the level of a class. Following table shows an actual data set with 10 observations.

Observation	Group
3.0	A1
2.0	A1
4.0	A1
6.0	A2
3.0	A2
6.0	A2
6.0	A3
6.0	A3
8.0	A3
4.0	A3

The first column represents observations and the second column represents group codes. You can include any positive or negative observations in the data file. You can also include 0 as a observation but you should take care in this special case because the program recognize 0 as missing by default. See the Remarks section below to correctly handle such a case.

In BLUPF90, the data set should be prepared as a text file in columns separated by spaces, with the same number of columns in every line. If you use other software like R and SAS, you should save the data as a text file. Almost any software can export the data to a text file.

Here we can write down the data file. You can put any file names. We save the following 7 lines as a file `data0.txt`.

Listing 3.1: data0.txt

```
3.0 1
2.0 1
4.0 1
6.0 2
3.0 2
6.0 2
6.0 3
6.0 3
8.0 3
4.0 3
```

This file shows the rules in a data file used with BLUPF90. Briefly, the data is a space-separated format with several columns as supported in many statistical software. Basically the file should contain numerical expressions only. We will explain the acceptable data format later.

Parameter file

A parameter file to run this analysis is something like:

Listing 3.2: param0.txt

```
DATAFILE
data0.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
```

```
1
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS:
2 3 cross
RANDOM_RESIDUAL VALUES
1.0
```

The parameter file is described as a series of pairs of keyword and value(s). For example, the first line DATAFILE is a keyword and the next line data.txt is the corresponding value. The value(s) should be placed in the next line to keyword. The keyword must start from the 1st position in a line (i.e. no leading spaces). The value can have leading spaces in the beginning of a line. The above example is the minimal one; so every time we need at least 7 keywords in a parameter file. Following is a detailed explanation of above parameter file.

1. DATAFILE = the name of data file
   - data.txt in this case
2. NUMBER_OF_TRAITS
   - 1 for a single-trait model
3. NUMBER_OF_EFFECTS = number of effects except residual
   - 1 for the fixed effect (only 1 effect in this model except residual)
4. OBSERVATION(S) = position of observation in the data file
   - 1 for the 1st column
5. WEIGHT(S) = position of weight in the data file [1]
   - empty line must be there.
6. EFFECTS: = description of the model (2 3 cross; see below)
   - 2 for the position of group code [2nd column in the data file]
   - 3 for the maximum code [the columns has 1 or 2 or 3; so the maximum is 3]
   - cross for the cross-classified effect
7. RANDOM_RESIDUAL VALUES = residual variance σ_e^2
   - 1.0; this value doesn't affect the results (see the normal equations).

The EFFECTS: block describes the model and its style is different from other software. In this case, this block has only 1 line because the model has only 1 effect except residual. If you have more effects in the model, you can add the description for each effect.

Running a program

You can save the data file and parameter file in the same folder (directory) and run the BLUPF90 program. When the program is invoked, it prints a message on screen and waits for key-in with a keyboard.

[1] WEIGHT does not mean the physical weight (i.e. in kg or pounds). It means the importance that we attach to this particular record. For instance, a weight of 10 may indicate that the measure was taken 10 times. See wikipedia.

```
name of parameter file?
```

When you supply the name of parameter file `param1.txt`, the program reads the parameters, builds and solves the equations, and writes the solutions to a file.

What will happen if you give a wrong name of parameter file or your parameter file has erroneous statements? In both cases, a message is shown and the program stops. Possible results are:

- Nonexistent parameter file: The message could be `forrtl: severe (24): end-of-file during read...`. A new file with the name, which you have input, will be created.
- Wrong description in parameter file: A massage depends on the situation. Modify the parameter file following the message.

Computations

After running the program, you can find a file solutions after the computations. This is also a text file and it contains the following items.

Listing 3.3: solutions

```
trait/effect level solution
    1 1 1 3.00000000
    1 1 2 5.00000000
    1 1 3 6.00000000
```

Ignoring the first line as a header, we have 3 lines with 4 columns and the 4th column has the solutions. The first solution (3.0) can be interpreted as it is for trait 1, effect 1, and the level (group code) 1. The second (5.0) is for trait 1, effect 1, and the level (group code) 2. The third (6.0) is for trait 1, effect 2, and the level (group code) 3. With mathematical notations, $\hat{A}_1 = 3.0$, $\hat{A}_2 = 5.0$, and $\hat{A}_3 = 6.0$. The solution is actually within-class average because we employed a 1-way model with an assumption of homogeneous residual variance.

3.1.2 Fixed effect model

We consider more general fixed effect model with the same data set. The model has 1 additional class effect and 1 regression coefficient.

$$y_{ijk} = A_i + S_j + \beta x_{ijk} + e_{ijk}$$

where y_{ijk} is an observation, A_i is a fixed effect, S_i is a different fixed effect, β is a regression coefficient, x_{ijk} is a covariate, and e_{ij} is residual effect. The matrix notation and the covariance structure are the same to the previous case. We will use the same data as before but there are some additional information.

Observation	Group1	Group2	x
3.0	A1	S1	1.0
2.0	A1	S2	1.0
4.0	A1	S1	2.0
6.0	A2	S2	2.0
3.0	A2	S1	1.0
6.0	A2	S2	2.0
6.0	A3	S1	2.0

Observation	Group1	Group2	x
6.0	A3	S2	1.0
8.0	A3	S1	1.0
4.0	A3	S2	2.0

The data file contains 4 columns in this case.

Listing 3.4: data1.txt

```
3.0 1 1 1.0
2.0 1 2 1.0
4.0 1 1 2.0
6.0 2 2 2.0
3.0 2 1 1.0
6.0 2 2 2.0
6.0 3 1 2.0
6.0 3 2 1.0
8.0 3 1 1.0
4.0 3 2 2.0
```

The parameter file is also similar to the previous one but we need extra entries for additional 2 effects.

Listing 3.5: param1.txt

```
DATAFILE
data1.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS:
2 3 cross
3 2 cross
4 1 cov
RANDOM_RESIDUAL VALUES
1.0
```

What is the difference? The only difference is in `NUMBER_OF_EFFECTS` and `EFFECTS:`. Because the model contains 3 effects (2 for class, 1 for regression), we change the value in `NUMBER_OF_EFFECTS`. We also need 3 lines in this block. Note that the 3rd effect is regression and its description is different from the previous 2 cross-classified effects. Following is the detailed explanation.

- *Effect 1* = `2 3 cross` : group code in the column 2 in data; maximum level 3; cross-classified effect
- *Effect 2* = `3 2 cross` : group code in the column 3 in data; maximum level 2; cross-classified effect

- *Effect 3* = `4 1 cross` : covariate in the column 4 in data; number of coefficient 1; regression

BLUPF90 assigns the first effect to *effect 1*, the second to *effect 2*, and so on. The order is kept in the solutions file.

Running the program with the parameter file, you can find the following result.

Listing 3.6: solutions

```
trait/effect level solution
   1 1 1 0.22500000
   1 1 2 2.22500000
   1 1 3 3.22500000
   1 2 1 2.27500000
   1 2 2 1.77500000
   1 3 1 0.50000000
```

The program provides 6 estimates; top 3 for the first effect, next 2 for the second effect, the bottom 1 for the last effect. With mathematical notations, $\hat{A}_1 = 0.225$, $\hat{A}_2 = 2.225$, and $\hat{A}_3 = 3.225$, $\hat{S}_1 = 2.275$, $\hat{S}_2 = 1.775$, and $\hat{\beta} = 0.500$. This is a single trait model, so the first column is always 1. The second column contains the effect number from the order of items in `EFFECTS:` in the parameter file.

Note that this equation is not full-rank so the estimates are ones of infinite numbers of solutions. BLUPF90 doesn't support any kinds of constraints to solve the equations. Although the estimates can look different from ones with other software, the contrasts from an estimable function will be consistent.

3.1.3 Options to control the program

BLUPF90 and related programs can change the behavior with an option. The option should be written in a parameter file as an additional line in the bottom of the file. There are many options and the official manual explains them. Here we just show a typical option line. This option changes the method of solving mixed model equations from the default PCG (preconditioned conjugate gradient) to the direct method using the FSPAK subroutine.

```
OPTION solv_method FSPAK
```

The option line starts with the capital OPTION followed by a keyword (option name) and additional information (if needed). In above case, the option name is `solv_method` and an additional information is FSPAK. Be careful to write an option because BLUPF90 distinguishes capital and small letters in the option line and fails if the name or information is wrong.

You can add above line to the last parameter file and save it.

Listing 3.7: param1a.txt

```
DATAFILE
data1.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
```

```
1
WEIGHT(S)

EFFECTS:
2 3 cross
3 2 cross
4 1 cov
RANDOM_RESIDUAL VALUES
1.0
OPTION solv_method FSPAK
```

With this parameter file, BLUPF90 successfully works but prints different messages on screen. The solutions are also different from the previous ones because the equations has no unique solutions.

Listing 3.8: solutions

```
trait/effect level solution
    1 1 1 2.50000000
    1 1 2 4.50000000
    1 1 3 5.50000000
    1 2 1 0.00000000
    1 2 2 -0.50000000
    1 3 1 0.50000000
```

You can still see the consistency in a linear combination of the solutions. For example, $\hat{A}_1 + \hat{S}_1 + \hat{\beta} = 3.0$ in both cases.

3.1.4 Comment lines in parameter file

A parameter file can contain comments, which will be ignored by the programs. A comment is a small memorandum to keep additional information in a parameter file. A comment start with the character # and the program ignores any characters between # and the end of the line. See the following example. This works as same as the previous parameter file.

Listing 3.9: param1b.txt

```
#
# This is an example parameter file for a fixed effect model.
# Solving method: FSPAK
#
DATAFILE
data1.txt # write the name of data file here
NUMBER_OF_TRAITS
1 # single trait model
NUMBER_OF_EFFECTS
3 # A, S, b
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS: # y = A + S + b*x + e
```

```
2 3 cross # effect A
3 2 cross # effect S
4 1 cov # regression
RANDOM_RESIDUAL VALUES # any values okay
1.0
# added options
OPTION solv_method FSPAK
```

This is just for a demonstration how comments are ignored. You can write comments almost anywhere in a parameter file. There is one exception; you don't write a comment in lines for variance components. The program doesn't remove the comment while reading the variance components. It stops with an error if a comment is there.

A comment line is a useful feature so we will put comments on a parameter file in the later chapters.

3.1.5 Remarks

Actual data files you have will contain alphabets or symbols (also called alphanumerics). Even when a class variable is an integer value, it may not start from 1 and may end in a very large value (like 2087654512). Before the analysis, you should edit the files to conform to the requirements of the program. RENUMF90 is a data preparation program and it replaces alphabets or large numbers to integer numbers starting from 1. Besides limited situations (e.g. simulated data or data renumbered by other means), you should usually use RENUMF90 to prepare and check the files.

By default, an observation (a trait) with value 0 (or similar numerical expression i.e. 0.0) is recognized as a missing observation. Similarly, an effect level read as 0 is treated as a missing information. This means that that effect is ignored for that particular record. For single-trait model, the missing observation makes little sense, except in rare circumstances like cross-validation or automatic repeated analysis (i.e. dozens of univariate analysis from the same, already recoded, file). Missing effect is also an abnormal assumption. So you should avoid 0 in the data. If your observation has a real 0 record, you can make the program treat this value as a regular observation by adding an `OPTION` statement as follows:

```
OPTION missing -999
```

The missing option switches the missing value recognized in BLUPF90 from `0` to `-999`. The value is treated as integer. No real values will be accepted.

How do we put the interaction among effects commonly found in a 2-way ANOVA ($y = A + B + A \times B + e$)? To do this, we should prepare all the effects A, B and $A \times B$ in the data file. BLUPF90 doesn't generate $A \times B$ from the effects A and B. Also BLUPF90 doesn't support automatic generation of nested effects. All interactions or nested effects should be prepared as class effects in the data file. RENUMF90, a data preparation software, can help to generate such effects. One exception is regressions nested in a class variable, which is supported by BLUPF90; we will mention this model in a context of random regressions.

3.1.6 Summary

- BLUPF90 supports linear mixed models.
- BLUPF90 needs 2 text files: data file and parameter file.
- BLUPF90 creates equations based on the files, solve the equations, and save the solutions to a file.
- Data file is a space-separated file containing integer or real numbers.
- Parameter file should be prepared with a specific form.
- Order of solutions depends on the description of a parameter file.

- BLUPF90 can change its behavior with options supplied in the bottom of a parameter file.
- A comment starts with # and ends at the end of the line. A comment is totally ignored by a program.
- Observation or effect 0 means missing. If you really want to use 0 as a observation, change the missing code with the option `missing`.

3.2 Trivial analyses for mixed models

In this section, we perform mixed model analyses using BLUPF90. We start with a simple mixed model (random effects, unrelated to each other) and finally consider pedigree information.

If you perform the simple mixed model, BLUPF90 needs only 2 files (data and parameter file), which are the same as explained in the previous section. If you consider pedigree, a pedigree file has to be prepared.

3.2.1 A simple mixed model

We assume the similar model described in the previous section and the second effect is now a random effect.

$$y_{ijk} = A_i + S_j + \beta x_{ijk} + e_{ijk}$$

where y_{ijk} is an observation, A_i is a fixed effect, S_j is a random effect, β is a regression coefficient, x_{ijk} is a covariate, and e_{ij} is residual effect. The model in matrix notation will be

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

where $\mathbf{y}$ is a vector for observations, $\mathbf{b}$ is a vector for fixed effects, $\mathbf{u}$ is a vector for random effects, $\mathbf{e}$ is a vector for residuals, $\mathbf{X}$ and $\mathbf{Z}$ are incidence matrices. In a general form, we can write $\text{var}(\mathbf{u}) = \mathbf{G}$ and $\text{var}(\mathbf{e}) = \mathbf{R}$ and the mixed model equations should be

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

For simplicity, we here assume $\text{var}(\mathbf{u}) = \mathbf{I}\sigma_s^2$ and $\text{var}(\mathbf{e}) = \mathbf{I}\sigma_e^2$ so the mixed model equations reduce to

$$\begin{bmatrix} \mathbf{X}'\mathbf{X}\sigma_e^{-2} & \mathbf{X}'\mathbf{Z}\sigma_e^{-2} \\ \mathbf{Z}'\mathbf{X}\sigma_e^{-2} & \mathbf{Z}'\mathbf{Z}\sigma_e^{-2}+\mathbf{I}\sigma_s^{-2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y}\sigma_e^{-2} \\ \mathbf{Z}'\mathbf{y}\sigma_e^{-2} \end{bmatrix}$$

These mixed model equations are often presented as

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z}+\mathbf{I}\lambda \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{Z}'\mathbf{y} \end{bmatrix}$$

where $\lambda = \sigma_e^2/\sigma_s^2$. So this model looks like a simple sire model with unrelated sires.

Note that BLUPF90 always creates the equations as above i.e. λ is never calculated. Precisely, $1/\sigma_e^2$ is always calculated and accumulated to the left hand side (LHS) and $1/\sigma_s^2$ is also added to LHS. Basically the program needs explicit values for variance components. Here we assume $\sigma_s^2 = 1.0$ and $\sigma_e^2 = 2.0$.

We can use the exactly the same data file shown before. File name is altered just for our convenience.

Listing 3.10: data2.txt

```
3.0 1 1 1.0
2.0 1 2 1.0
4.0 1 1 2.0
```

```
6.0 2 2 2.0
3.0 2 1 1.0
6.0 2 2 2.0
6.0 3 1 2.0
6.0 3 2 1.0
8.0 3 1 1.0
4.0 3 2 2.0
```

The parameter file is following.

Listing 3.11: param2.txt

```
DATAFILE
data2.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS:
2 3 cross
3 2 cross
4 1 cov
RANDOM_RESIDUAL VALUES
2.0
RANDOM_GROUP
2
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
1.0
```

The first 7 keywords are the same as used in the fixed effect models. To define the random effect, we use 4 extra keywords.[2] The keywords should be added just after the common keywords.

1. `RANDOM GROUP` = specify which effect is random in the `EFFECTS` block
   - 2 for the second line (effect 2) in `EFFECTS` is the random effect
2. `RANDOM_TYPE` = covariance structure
   - diagonal for identity covariance matrix i.e. diagonals only
3. `FILE` = additional fir for the covariance structure
   - empty for no information needed here
4. `(CO)VARIANCES` = variance components σ_s^2

[2] The common 7 keywords build the design-matrix parts in the mixed model equations i.e. $\mathbf{X}'\mathbf{X}$, $\mathbf{X}'\mathbf{Z}$, $\mathbf{Z}'\mathbf{X}$ and $\mathbf{Z}'\mathbf{Z}$. The additional 4 keywords provides the covariance structure i.e. $\mathbf{G}^{-1}$

- 1.0 for the variance

If you have 2 or more random effects (e.g. permanent environmental or maternal effect), you can add another extra 4 keywords to the parameter file. We will see such a case later.

The solutions will be very different from the previous one which assumes the effect 2 is a fixed effect.

Listing 3.12: solutions

```
trait/effect level solution
   1 1 1 2.30434783
   1 1 2 4.26086957
   1 1 3 5.28260870
   1 2 1 0.17391304
   1 2 2 -0.17391304
   1 3 1 0.47826087
```

The solution file contains both BLUE for the fixed effects and BLUP for the random effects. The first column is always 1 because it is a single-trait model; the 2nd column shows the effect number from 1 to 3; the 3rd column shows the level in each effect; the fourth column contains the solutions. The first 3 solutions are for the effct 1 (A_i), the next 2 for the effect 2 (S_j) and the last 1 for the regression (β). Details are: $\hat{A}_1 = 2.30$, $\hat{A}_2 = 4.26$, $\hat{A}_3 = 5.28$, $\hat{S}_1 = 0.17$, $\hat{S}_2 = -0.17$ and $\hat{\beta} = 0.48$.

3.2.2 An animal model

Animal model specifically refers to a mixed model with individual additive genetic effect as a random effect. Genetic covariances between animals are considered with a numerator relationship matrix. You need a pedigree file with this model. In this section, we assume the model above with additive genetic effects. An animals has either one observation or no observation (found only in a pedigree file). The mathematical model is the following:

$$y_{ijk} = A_i + S_j + \beta x_k + u_k + e_{ijk}$$

where u_k is the additive genetic effect for an animal k. Here we treat S_j as a fixed effect. The mixed model equations are

$$\begin{bmatrix} \mathbf{X}'\mathbf{X}\sigma_e^{-2} & \mathbf{X}'\mathbf{Z}\sigma_e^{-2} \\ \mathbf{Z}'\mathbf{X}\sigma_e^{-2} & \mathbf{Z}'\mathbf{Z}\sigma_e^{-2} + \mathbf{A}^{-1}\sigma_u^{-2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y}\sigma_e^{-2} \\ \mathbf{Z}'\mathbf{y}\sigma_e^{-2} \end{bmatrix}$$

where $\mathbf{A}$ is a numerator relationship matrix, σ_u^2 is the additive genetic variance, and σ_e^2 is the residual variance.

We further extend the data table to add an animal's ID (code). We show the actual data file to be used in this analysis.

Listing 3.13: data3.txt

```
3.0 1 1 1.0 6
2.0 1 2 1.0 9
4.0 1 1 2.0 12
6.0 2 2 2.0 7
3.0 2 1 1.0 10
6.0 2 2 2.0 13
6.0 3 1 2.0 8
```

```
6.0 3 2 1.0 11
8.0 3 1 1.0 14
4.0 3 2 2.0 15
```

In the data file, animal ID starts with 6 and it means animals 1 to 5 have no observation so they are not included in the data file. The order of animal ID is totally arbitrary. You don't have to order the animals chronologically.

We show the pedigree file below. Pedigree information consists of 3 columns: animal ID, its sire's ID, and its dam's ID. If an animal's parent is unknown (missing), you should put `0`. The following is the pedigree in this analysis.

Listing 3.14: pedigree3.txt

```
 1 0 0
 2 0 0
 3 0 0
 4 0 0
 5 0 0
 6 0 0
 7 2 5
 8 1 4
 9 2 3
10 7 6
11 7 4
12 11 8
13 11 10
14 9 13
15 11 10
```

Individual ID should be an integer value and it corresponds to one in the data file. If your data has non-integer animal IDs, you can use the RENUMF90 program to replace the characters to integer values in both pedigree and data.

The basic format of the parameter file is the same to the previous one. We assume $\sigma_a^2 = 0.5$ and $\sigma_e^2 = 2.0$.

Listing 3.15: param3.txt

```
DATAFILE
data3.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS:
2 3 cross
3 2 cross
4 1 cov
5 15 cross
```

```
RANDOM_RESIDUAL VALUES
2.0
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
pedigree3.txt
(CO)VARIANCES
0.5
```

There are 5 modifications in this file compared to the previous one.

1. NUMBER_OF_EFFECTS = We added the additive genetic effect so the number of effects should be incremented.
   - Total number of effect is 4.
2. EFFECTS: = The added effect placed in the 4th row.
   - 5 for the position of ID in the data file, 15 for the total number of animals (in the pedigree), and cross for cross-classified effect.
3. RANDOM_GROUP = specify which effect is random in EFFECTS:
   - The 4th effect in the effect section is a random effect.
4. RANDOM_TYPE = covariance structure
   - add_animal for additive genetic effect
5. FILE = pedigree file
   - pedigree4.txt

Now you can run BLUPF90 and obtain the solutions.

Listing 3.16: solutions

```
trait/effect level solution
   1 1 1 0.20259645
   1 1 2 2.21996461
   1 1 3 3.16680208
   1 2 1 2.27537292
   1 2 2 1.71160538
   1 3 1 0.52442755
   1 4 1 -0.03487115
   1 4 2 0.08280493
   1 4 3 0.03843921
   1 4 4 0.04492008
   1 4 5 0.04436203
   1 4 6 -0.17565609
   1 4 7 0.10794668
   1 4 8 -0.02984646
   1 4 9 0.09906236
   1 4 10 -0.25282594
   1 4 11 0.15622415
```

```
1 4 12 0.10874296
1 4 13 0.16426465
1 4 14 0.34296714
1 4 15 -0.25707431
```

The solution file contains both BLUE for the fixed effects and BLUP for the random effects. The first column is always 1 because it is a single-trait model; the 2nd column shows the effect number from 1 to 4; the 3rd column shows the level in each effect; the fourth column contains the solutions. The first 3 solutions are for the effct 1 (A_i), the next 2 for the effect 2 (S_j), the next 1 for the regression (β) and the remaining 15 solutions for the additive genetic effect i.e. estimated breeding values (u_k). Details are: $\hat{A}_1 = 0.203$, $\hat{A}_2 = 2.220$, $\hat{A}_3 = 3.167$, $\hat{S}_1 = 2.275$, $\hat{S}_2 = 1.711$ and $\hat{\beta} = 0.524$, $\hat{u}_1 = -0.035$, $\hat{u}_2 = 0.083$, $\hat{u}_3 = 0.038$ and so on.

3.2.3 More than 1 random effect

We often meet a model with more than 1 random effect such as repeatability model (additive genetic + permanent environmental [PE] effect) or maternal effect model (direct + maternal + maternal PE effect). BLUPF90 supports such a complicated model in a straightforward way. We assume the same animal model described above except S_j is random:

$$y_{ijk} = A_i + S_j + \beta x_k + u_k + e_{ijk}$$

where the random effects are two S_j and u_k. To perform the analysis, we can use the same data file and pedigree file. We assume that the two random effects S and u are not related to each other and $\sigma_s^2 = 1.0$, $\sigma_u^2 = 0.5$ and $\sigma_e^2 = 2.0$. We need to modify the parameter file to include additional random effect.

Listing 3.17: param3a.txt

```
DATAFILE
data3.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
1
WEIGHT(S)

EFFECTS:
2 3 cross
3 2 cross
4 1 cov
5 15 cross
RANDOM_RESIDUAL VALUES
2.0
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
pedigree3.txt
(CO)VARIANCES
0.5
```

```
RANDOM_GROUP
2
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
1.0
```

We just add 4 keywords (RANDOM_GROUP, RANDOM_TYPE, FILE and (CO)VARIANCES) to the last parameter file. In other words, the 4 keywords are optional so you can add the 4 keywords as you include one more random effect if the effects are unrelated each other.

However, if the random effects are correlated to each other, you should use a different way. This happens in a maternal effect model (direct and maternal genetic effect) and random regression model (random regression coefficients). Multiple-trait model is also another story. We don't explain such cases here but we will consider them in later chapters.

The solutions are below.

Listing 3.18: solutions

```
 trait/effect level solution
    1 1 1 2.27101829
    1 1 2 4.24566929
    1 1 3 5.21524118
    1 2 1 0.18586035
    1 2 2 -0.18586066
    1 3 1 0.49101139
    1 4 1 -0.01963655
    1 4 2 0.06556686
    1 4 3 0.02911205
    1 4 4 0.04793111
    1 4 5 0.03646405
    1 4 6 -0.15943505
    1 4 7 0.08747653
    1 4 8 -0.00548994
    1 4 9 0.07644881
    1 4 10 -0.24829955
    1 4 11 0.13527276
    1 4 12 0.12002543
    1 4 13 0.15461713
    1 4 14 0.33690598
    1 4 15 -0.27372367
```

The order of the solutions is same to the previous animal model but the effect 2 is now random effect. The order of the solutions is determined using EFFECTS: block in the parameter file. So even if you add more random effects, the order will not be changed.

3.2.4 Remarks

BLUPF90 builds the mixed model equations simply following the parameter file. Even if your parameter file has no fixed effect (i.e. it has random effects only), the program doesn't add any fixed effect to the model. No general mean, nothing. In such a case, the solutions can be nonsense. If you want to perform

a random effect model with the general mean, you should prepared an additional column containing 1 in the data file and describe it as a fixed effect in the parameter file.

In the animal model with `add_animal`, BLUPF90 doesn't consider inbreeding i.e. $\mathbf{A}^{-1}$ is constructed with the Henderson's method ignoring inbreeding. BLUPF90 supports $\mathbf{A}^{-1}$ with inbreeding but the program doesn't calculate the inbreeding coefficients. A user should calculate the inbreeding coefficients and put them into pedigree file with a special format. With this special pedigree file and the use of `add_an_upginb` (instead of `add_animal`), the inbreeding will be successfully incorporated.

3.2.5 Summary

- BLUPF90 supports linear mixed models with 1 or more random effects.
- BLUPF90 needs 3 text files: data, pedigree and parameter files.
- Pedigree file is needed if you conduct genetic analyses.
- Pedigree file should contain 3 columns: animal ID, sire ID and dam ID. the ID must be an integer value.
- To define a random effect, add 4 keywords (`RANDOM_GROUP`, `RANDOM_TYPE`, `FILE` and `(CO)VARIANCES`) and the corresponding values to the parameter file.
- The order of the solutions depends on the `EFFECTS:` block in the parameter file.
- You should manually add the general mean if you use a random effect model.
- With `add_animal`, inbreeding is ignored in the calculation of $\mathbf{A}^{-1}$.

3.3 Trivial analyses for single-step GBLUP

In ssGBLUP, the program requires 5 files: parameter, data, pedigree, marker, and cross-referece files. The first 3 files are the same to a regular animal model so you need additional 2 files for ssGBLUP. In this section, first we describe relationship matrices needed in the analysis. Then we introduce the 2 additional files and run the program.

3.3.1 Single-step GBLUP

Model and relationship matrices

To demonstrate ssGBLUP, we will use the same animal model introduced in the previous section.

$$y_{ijk} = A_i + S_j + \beta x_k + u_k + e_{ijk}$$

where u_k is the additive genetic effect for an animal k. Here we treat S_j as a fixed effect. Following is the mixed model equations.

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{A}^{-1}/\sigma_u^2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

The basic idea of ssGBLUP is simple: just replace $\mathbf{A}^{-1}$ with $\mathbf{H}^{-1}$ containing genomic relationship matrix in the mixed model equations. The $\mathbf{H}^{-1}$ has the following structure:

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} - \mathbf{A}_{22}^{-1} \end{bmatrix}$$

where $\mathbf{G}$ is the genomic relationship matrix for genotyped animals and $\mathbf{A}_{22}$ is the subset of the numerator relationship matrix for genotyped animals. So this method can combine both genotyped and nongenotyped animals in the same equations.

BLUPF90 prepares $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$ before the solution. The program also checks SNP markers and reports errors if found.

Files

We can use the exactly the same data and pedigree files used in the previous analysis. We, nevertheless, put different file names to data and pedigree to avoid confusing.

Listing 3.19: data4.txt

```
 3.0 1 1 1.0 6
 2.0 1 2 1.0 9
 4.0 1 1 2.0 12
 6.0 2 2 2.0 7
 3.0 2 1 1.0 10
 6.0 2 2 2.0 13
 6.0 3 1 2.0 8
 6.0 3 2 1.0 11
 8.0 3 1 1.0 14
 4.0 3 2 2.0 15
```

The following is the pedigree file.

Listing 3.20: pedigree4.txt

```
 1 0 0
 2 0 0
 3 0 0
 4 0 0
 5 0 0
 6 0 0
 7 2 5
 8 1 4
 9 2 3
10 7 6
11 7 4
12 11 8
13 11 10
14 9 13
15 11 10
```

The parameter file is also the same except an option needed to read a marker file.

Listing 3.21: param4.txt

```
DATAFILE
data4.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
1
WEIGHT(S)
```

```
EFFECTS:
2 3 cross
3 2 cross
4 1 cov
5 15 cross
RANDOM_RESIDUAL VALUES
2.0
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
pedigree4.txt
(CO)VARIANCES
0.5
OPTION SNP_file snp4.txt snp4index.txt
```

The last line specifies the name of marker file. The option SNP file makes BLUPF90 read the marker file `snp4.txt` and the cross-reference file `snp4index.txt`, and invoke the genomic computations. You can omit the second argument (the name of cross-reference file) if its name is the name of marker file + `XrefID`. In this case, if the cross-reference file is `snp4.txt_XrefID`, you can omit the second argument in that option.

We assume only 8 animals are genotyped (2, 4, 6, 10, 11, 12, 13 and 15). We also assume each genotyped animal has the original name and the marker file contains the names (ID002, ID004, ID006, ID010, ID011, ID012, ID013 and ID015, respectively). The following is the marker file. Note that the data would be shown incorrectly because of a typesetting issue. Please see the actual file.

Listing 3.22: snp4.txt

```
ID002 121202011100121121200010200120202220111200011002010201100101221211012210000121010111220021 0
ID004 222202020011012221012100110002222110211100001001120202000200222200022200002012000002020020 0
ID006 212202010110022112111121210011211111221200010101221111100120121111202210020122011002220022 0
ID010 211112110111121012121111221020202121121100011002121111100110121110112210011122100002220012 0
ID011 221112120012111122021101121011212111201000120011202020002002222000222000021221000021200110
ID012 222202020011012221012100110002222110211100001001120202000200222200022200002012000002020020 0
ID013 212202010121012112122210120011212210221200001001221111100110121110112210011022000021200210
ID015 212202010121012112122210120011212210221200001001221111100110121110112210011022000021200210
```

BLUPF90 accepts only a marker file prepared with the following rules.

- A marker file is a text file with fixed-width format including 2 columns.
- The first column contains an animal's ID. You don't have to use the common ID to data and pedigree. So you can put the animal's original ID here. (BLUPF90 actually skip this column so you can put any information here.)
- The second column contains an animal's genotypes. It has 4 possible value: `0` (homozygote), `1` (heterozygote), `2` (alternative homozygote) and `5` (missing or unknown).
- All the second column must start with the fixed position in a line. In above example, the second column always starts 9th character.
- Each genotyped animals must have the same number of markers.
- You must not insert a space between 2 adjacent genotypes.

The program will immediately stop if it find the inconsistent data in the file.

We need one more file to run BLUPF90. The last file is a cross-reference file that relates the integer code used in BLUPF90 to the ID in the marker file. This is a text file with 2 columns separated with a space.

Listing 3.23: snp4index.txt

```
2 ID002
4 ID004
6 ID006
10 ID010
11 ID011
12 ID012
13 ID013
15 ID015
```

The order of animals must be the same as the marker file. The first column contains the integer value (common to pedigree file) and the second line has the original ID found in the marker file.

Why do we consider such a tricky example? Because the original-ID-in-the-marker-file situation actually happens in the real life. A farm animal usually have the original ID and phenotypes, pedigrees and genotypes are recorded with the original ID whereas BLUPF90 accepts only an integer value as an animal's ID. So we usually use RENUMF90 to replace the original ID with integer values and create renumbered files. However, the marker file is really huge and it is very inefficient to replace the original ID with an integer code. The RENUMF90 doesn't rewrite the marker file. Instead, RENUMF90 generates the cross-reference file to make the correspondence between the ID in markers and the ID in pedigree.

Running the program

We can now run BLUPF90 program with the 5 files. You can see very long output from the genomic routine on the screen. We briefly introduce what happens in this process.

- Read the pedigrees for genotyped animals and compute $\mathbf{A}_{22}$.
- Read SNP markers and perform a quality control on the data (possibly remove unqualified markers).
- Compute $\mathbf{G}$ and compare it with $\mathbf{A}_{22}$.
- Compute $\mathbf{A}_{22}^{-1}$ and $\mathbf{G}^{-1}$.
- Finally compute $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$ and store them into single matrix.
- After the process, BLUPF90 moves to the regular solution of the mixed model equation using $\mathbf{G}^{-1} - \mathbf{A}_{22}^{-1}$.

There are many options to precisely control the genomic routine. Each option has the default value so the user should know the default behavior of the program. This explanation is out of scope of this tutorial. We will see the details in the later chapters.

After running the program, we can find the solutions file.

Listing 3.24: solutions

```
trait/effect level solution
   1 1 1 0.23340767
   1 1 2 2.19314788
   1 1 3 3.16386924
   1 2 1 2.28753998
   1 2 2 1.69948874
```

```
1 3 1 0.52196308
1 4 1 0.00700521
1 4 2 0.08747780
1 4 3 -0.01954304
1 4 4 0.02639925
1 4 5 0.07679027
1 4 6 -0.18438116
1 4 7 0.15271384
1 4 8 -0.09012753
1 4 9 0.01442435
1 4 10 -0.23558188
1 4 11 0.22278505
1 4 12 0.10730577
1 4 13 0.20708267
1 4 14 0.32362808
1 4 15 -0.21760840
```

Single-step GBLUP directly provides GEBV (estimated breeding values enhanced with genomic information). The above solutions for animals are GEBV. The values are very different from the previous animal-model analysis because of the very small data.

3.3.2 Remarks

If your marker data is relatively large, you should consider more efficient way. Each relationship matrix created in the genomic routine is fully stored in memory i.e. dense matrix consuming a lot of memory. The required memory is $8n^2$ /1024^3 GB (gigabytes) for each relationship matrix. Computing time will be longer with many genotyped animals. To determine appropriate options to obtain stable GEBV, you should run the program many times. You can save a relationship matrix to a file and reuse the file.

The PREGSF90 perform the genomic computations only. It can do the same computation as BLUPF90 does (except the solution of the mixed model equations). The best practice is to run PREGSF90 to check and clean up the markers, compute the relationship matrices and save the matrices to files before you run BLUPF90. BLUPF90 can read the files generated with PREGSF90 so you can seve the time.

3.3.3 Summary

- BLUPF90 can perform a single-step GBLUP (ssGBLUP) analysis.
- With ssGBLUP, a marker file and cross-reference file are needed in addition to data, pedigree, and parameter files needed in a regular animal model analysis.
- A option in a parameter file is needed.
- There are many options to control the genomic computations. See the manual.
- Marker file contains is a fixed-length text file with an animal's ID and genotypes.
- Cross-reference file relates the animal's ID in marker file to integer code in pedigree file.
- GEBV is directly obtained as a solution.
- PREGSF90 is useful to save a relationship matrix to a file if you run the program several times using the same pedigree and genomic information. BLUPF90 can read the file so you can save the time.

3.4 Trivial analyses for multiple-trait models

BLUPF90 supports multiple-trait models with few modifications to a parameter file used in a single-trait analysis. In this section, we consider a simple multiple-trait model and introduce some extensions in a parameter file to use the model.

3.4.1 Basic elements

Model description

We extend a single-trait animal model shown in the previous section, but no genomics. We will consider a 2-trait model with the same effects on both traits. Also we assume we have no missing observations at all. A mathematical model for each trait can be written as follows.

$$y_{ijk:1} = A_{i:1} + S_{j:1} + \beta x_{ijk:1} + u_{k:1} + e_{ijk:1}$$
$$y_{ijk:2} = A_{i:2} + S_{j:2} + \beta x_{ijk:2} + u_{k:2} + e_{ijk:2}$$

The mixed model can be written with a typical form introduced as before.

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

With matrix notation, this model can be seen in 2 ways depending on the order of observations.

Ordering animals within trait: $\mathbf{y} = \left[\begin{array}{c|c|c|c|c} y_{1:1}, y_{2:1}, y_{3:1}, \cdots, y_{n:1} & y_{1:2}, y_{2:2}, y_{3:2}, \cdots, y_{n:2} \end{array}\right]$

Ordering traits within animal: $\mathbf{y} = \left[\begin{array}{c|c|c|c|c} y_{1:1}, y_{1:2} & y_{2:1}, y_{2:2} & y_{3:1}, y_{3:2} & \cdots & y_{n:1}, y_{n:2} \end{array}\right]$

This difference is not essential on conducting the analysis but you should recognize the difference when writing down the mathematical model in matrix notation. Above two result in the different mixed model equations even if the solutions are the identical. BLUPF90 programs use the latter one so we will describe the model with this rule.

In this case, the variance of $\mathbf{y}$ is

$$\mathrm{var}(\mathbf{y}) = \mathbf{Z}(\mathbf{A} \otimes \mathbf{G}_0)\mathbf{Z}' + \mathbf{I} \otimes \mathbf{R}_0$$

where $\mathbf{G}_0$ is the genetic variance-covariance matrix among traits (2×2 in this case), $\mathbf{R}_0$ is the residual variance-covariance matrix among traits (also 2×2 in this case), I is the identity matrix ($N \times N$; N is the number of observations in a trait) and $\otimes$ is an operator for Kronecker product. Elements of each covariance matrix and its inverse can be written as

$$\mathbf{G}_0 = \begin{bmatrix} \sigma^2_{g1} & \sigma_{g12} \\ \sigma_{g21} & \sigma^2_{g2} \end{bmatrix} \quad \text{and} \quad \mathbf{G}_0^{-1} = \begin{bmatrix} \sigma^{g1} & \sigma^{g12} \\ \sigma^{g21} & \sigma^{g2} \end{bmatrix}$$
$$\mathbf{R}_0 = \begin{bmatrix} \sigma^2_{e1} & \sigma_{e12} \\ \sigma_{e21} & \sigma^2_{e2} \end{bmatrix} \quad \text{and} \quad \mathbf{R}_0^{-1} = \begin{bmatrix} \sigma^{e1} & \sigma^{e12} \\ \sigma^{e21} & \sigma^{e2} \end{bmatrix}$$

Kronecker product is not a regular product. Consult the following examples:

$$\mathbf{A} \otimes \mathbf{G}_0 = \begin{bmatrix} a_{11}\mathbf{G}_0 & a_{12}\mathbf{G}_0 & \cdots & a_{1n}\mathbf{G}_0 \\ a_{21}\mathbf{G}_0 & a_{22}\mathbf{G}_0 & \cdots & a_{2n}\mathbf{G}_0 \\ \vdots & \cdots & \ddots & \vdots \\ a_{n1}\mathbf{G}_0 & a_{n2}\mathbf{G}_0 & \cdots & a_{nn}\mathbf{G}_0 \end{bmatrix} \quad \text{and} \quad \mathbf{I} \otimes \mathbf{R}_0 = \begin{bmatrix} \mathbf{R}_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_0 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_0 \end{bmatrix}$$

where n is the number of pedigree animals and a_{ij} is the i-th row and j-th column element of $\mathbf{A}$. In the latter equation, $\mathbf{R}_0$ is repeated N times on diagonal.

The mixed model equations are

$$\begin{bmatrix} \mathbf{X}'(\mathbf{I} \otimes \mathbf{R}_0^{-1})\mathbf{X} & \mathbf{X}'(\mathbf{I} \otimes \mathbf{R}_0^{-1})\mathbf{Z} \\ \mathbf{Z}'(\mathbf{I} \otimes \mathbf{R}_0^{-1})\mathbf{X} & \mathbf{Z}'(\mathbf{I} \otimes \mathbf{R}_0^{-1})\mathbf{Z} + \mathbf{A}^{-1} \otimes \mathbf{G}_0^{-1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'(\mathbf{I} \otimes \mathbf{R}_0^{-1})\mathbf{y} \\ \mathbf{Z}'(\mathbf{I} \otimes \mathbf{R}_0^{-1})\mathbf{y} \end{bmatrix}$$

This looks very different from a typical description for some people. But, again, the form of the equations don't affect on any results.

Required files

In this 2-trait model, we need 2 columns for observations (one for the trait 1 and the other one for trait 2). Also we need the columns for effects in each trait. We assume the same model in both traits so we can use the common columns to assign the effects in the data file.

As an example, we extend the data used in the previous animal model analysis.

Listing 3.25: data5.txt

```
3.0 4.5 1 1 1.0 6
2.0 7.5 1 2 1.0 9
4.0 3.5 1 1 2.0 12
6.0 -0.5 2 2 2.0 7
3.0 5.5 2 1 1.0 10
6.0 1.5 2 2 2.0 13
6.0 -1.5 3 1 2.0 8
6.0 2.5 3 2 1.0 11
8.0 0.5 3 1 1.0 14
4.0 4.5 3 2 2.0 15
```

Pedigree file is the same to the previous one. Here we changed the file name with the same content.

Listing 3.26: pedigree5.txt

```
 1 0 0
 2 0 0
 3 0 0
 4 0 0
 5 0 0
 6 0 0
 7 2 5
 8 1 4
 9 2 3
10 7 6
11 7 4
12 11 8
13 11 10
14 9 13
15 11 10
```

The parameter file also looks like very similar to the previous one except it describes covariance matrices and a model for each trait. We assume $\sigma^2_{g1} = 0.5$, $\sigma_{g12} = -0.25$, and $\sigma^2_{g2} = 1.0$ for genetic (co)variances and $\sigma^2_{e1} = 2.0$, $\sigma_{e12} = 1.0$, and $\sigma_{e2} = 1.5$ for residual (co)variances.

Listing 3.27: param5.txt

```
DATAFILE
data5.txt
NUMBER_OF_TRAITS
2
NUMBER_OF_EFFECTS
```

```
4
OBSERVATION(S)
1 2
WEIGHT(S)

EFFECTS:
3 3 3 cross
4 4 2 cross
5 5 1 cov
6 6 15 cross
RANDOM_RESIDUAL VALUES
2.0 1.0
1.0 1.5
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
pedigree5.txt
(CO)VARIANCES
 0.50 -0.25
-0.25 1.00
```

For a multiple-trait mode, we should carefully describe the following values.

- `NUMBER_OF_TRAITS` = appropriate number of traits
  - `2`: two-trait model
- `OBSERVATION(S)` = enumerate the positions of observations in the data file
  - `1 2`: the first and second columns
- `EFFECTS:` = enumerate the positions of effect, the number of levels and the effect type
  - `3 3 3 cross`: the position of effect 1 for trait 1 (`3`); the position of effect 1 for trait 2 (`3`); the number of levels (`3`); effect type (`cross`)
  - `4 4 2 cross`: the position of effect 2 for trait 1 (`4`); the position of effect 2 for trait 2 (`4`); the number of levels (`2`); effect type (`cross`)
  - `5 5 1 cov`: the position of effect 3 for trait 1 (`5`); the position of effect 3 for trait 2 (`5`); the number of levels (`3`); effect type (`cov`)
  - `6 6 15 cross`: the position of effect 4 for trait 1 (`6`); the position of effect 4 for trait 2 (`6`); the number of levels (`15`); effect type (`cross`)
- `RANDOM RESIDUAL VALUES` = residual covariance matrix
  - full 2×2 matrix; total 4 elements should be here.
- `(CO)VARIANCES` = genetic covariance matrix
  - full 2×2 matrix; total 4 elements should be here.

You should be careful to write the `EFFECT:` block. With 2-trait model, each statement has to have 4 elements; the first 2 for the positions of effect for each trait in the data file, the next 1 for the number of levels and the last 1 for effect type. If you run a model with n traits, the first n elements should be the positions of effects for each trait.

Residual and genetic covariance matrices should be fully described i.e. all upper, diagonal and lower elements. The lines including a covariance matrix can't accept comments.

3.4.2 Resulting solutions

Running BLUPF90 with above parameter file, the solutions file contains are following.

Listing 3.28: solutions

```
trait/effect level solution
   1 1 1 0.19990999
   2 1 1 5.43362035
   1 1 2 2.30088577
   2 1 2 2.42229505
   1 1 3 3.21493025
   2 1 3 1.78930980
   1 2 1 2.31093239
   2 2 1 2.25555489
   1 2 2 1.76148927
   2 2 2 3.88944218
   1 3 1 0.54865198
   2 3 1 -2.42030677
   1 4 1 -0.02253416
   2 4 1 -0.04249548
   1 4 2 0.10792661
   2 4 2 -0.13113037
   1 4 3 -0.02682949
   2 4 3 0.09896514
   1 4 4 0.03164894
   2 4 4 -0.02495242
   1 4 5 0.13475233
   2 4 5 -0.23009750
   1 4 6 -0.22496602
   2 4 6 0.32971990
   1 4 7 0.25609198
   2 4 7 -0.41070894
   1 4 8 -0.01797409
   2 4 8 -0.07622271
   1 4 9 0.01371895
   2 4 9 0.08288296
   1 4 10 -0.84143020
   2 4 10 1.63889151
   1 4 11 0.19804880
   2 4 11 -0.20028530
   1 4 12 0.03355559
   2 4 12 0.06720834
   1 4 13 0.10550434
   2 4 13 0.07202442
   1 4 14 0.52502266
   2 4 14 -0.66337381
   1 4 15 -1.00156891
   2 4 15 2.01449847
```

This is long because every effect was considered in both traits. The format is the same to the previous one except the first column contains 1 or 2. You can obtain 2 solutions, for trait 1 and trait 2, in every effect. For example, estimated breeding values for the animal 13 are 0.1055 for trait 1 and 0.0702 for trait 2.

For your information, the order of solutions in the file is the same as in memory. It is corresponding to the matrix notation described above.

3.4.3 Remarks

You can put a missing observation in the data file. Missing value is `0` by default but you can change the value with an option `OPTION missing`. The missing code is also integer value (positive or negative). At least 1 trait has a real observation. In other words, missing-in-all-traits is not allowed.

The program supports different models in different traits (unequal design matrices). You can put 0 as a position in `EFFECT:` block if the trait doesn't the effect. At least 1 trait has to have non-zero position. We will see such models in the later chapter.

Single-step GBLUP is supported in multiple-trait model. You can add `OPTION SNP_file` to perform such a model.

3.4.4 Summary

- BLUPF90 supports multiple-trait models.
- The same data and pedigree files to a single-trait case can be used.
- Parameter file should be modified to fit the multiple-trait model.
- In the `EFFECT:` block, each statement should have *n* positions of effects for *n* trait model.
- Missing values are allowed. The default is 0 and you can change the missing code.
- Each trait may have a different model i.e. different effects.

4 Data preparation with RENUMF90

This chapter provides a basic idea how to use RENUMF90 to prepare the files for data, pedigree, genomic markers and parameters. We usually run RENUMF90 to align the format and clean-up the raw data from field or commercial populations. We will start with a minimal example as before.

4.1 Basic data preparation

4.1.1 Basic usage of RENUMF90

Why RENUMF90?

BLUPF90 accepts data and pedigree files containing only numerical expressions (integer or real values). Group label should be integer starting from 1. The raw field or commercial data files usually contain characters (alphabets or symbols) for animals' ID or group code. The characters should be replaced with numerical values before the analysis with BLUPF90 programs. RENUMF90 per- forms such a job.

Trivial instruction file

RENUMF90 also accepts a parameter file, this is totally different from ones used in BLUPF90. Don't confuse 2 kinds of parameter files. Here, to avoid the confusion, we will refer to the parameter file used in RENUMF90 as (renumbering) instruction file in this tutorial. This name is unofficial but easier to differentiate two.

To understand the behavior of RENUMF90, let's try a small example without pedigree. Here we use a raw data file and an instruction file. The raw data is a space-separated text file and contains 5 columns.

Listing 4.1: rawdata1.txt

```
ID006 A 1 1.0 3.0
ID009 A 2 1.0 2.0
ID012 A 1 2.0 4.0
ID007 B 2 2.0 6.0
ID010 B 1 1.0 3.0
ID013 B 2 2.0 6.0
ID008 C 1 2.0 6.0
ID011 C 2 1.0 6.0
ID014 C 1 1.0 8.0
ID015 C 2 2.0 4.0
```

We will generate data parameter files for the following model.

$$y_{ijk} = A_i + S_j + \beta x_{ijk} + e_{ijk}$$

This model is actually the same to one introduced in the previous chapter as a fixed effect model; The first 2 are fixed cross-classified effects and the third one is a fixed regression. So this example tries to produce a similar data set used before.

RENUMF90 can read the only necessary columns described in an instruction file. The following instruction file demonstrates the renumbering of column 2 and 3 as fixed cross-classified effects, column 4 as a covariate and column 5 as a phenotype. In this case, the column 1 is ignored.

Listing 4.2: renum1.txt

```
DATAFILE
rawdata1.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
1.0
EFFECT # 1st effect
2 cross alpha
EFFECT # 2nd effect
3 cross alpha
EFFECT # 3rd effect
4 cov
```

Instruction file can also have comments, which starts with #. The above file has several comments.

The instruction file looks like the parameter file for BLUPF90. The file contains several pairs of keyword and value(s). The file contains 6 kinds of keywords. `EFFECT` can be repeated several times. The following keywords are required in the minimal instruction file.

Keyword	possible value	description
`DATAFILE`	characters	The name of data file to be processed.
`TRAITS`	integer	A list of the position(s) for observation in the data file.
`FIELDS_PASSED TO OUTPUT`	integer	A list of the position(s) for unchanged columns in the data file. Empty value is acceptable if not needed. The columns will be passed through the renumbered file.
`WEIGHT(S)`	integer	The position(s) for weight in the data file. Empty value is acceptable if not needed.
`RESIDUAL_VARIANCE`	real value(s)	Residual (co)variances.
`EFFECT`	(see below)	Description of an effect. Repeatable.

The `EFFECT` keyword can be repeated as your model is filled. Above instruction file contains 3 effects. The `EFFECT` keyword has several values as follows.

Keyword	Position	effect type	data type
`EFFECT`	integer	cross	alpha numer
		cov	

In above table, position means the position(s) for group code (for class effect) or covariate (for regression) in the data file. You can choose either two effect types (`cross` for cross-classified or `cov` for regression). For cross-classified effect, you can also choose the data type: `alpha` if the column may contain alphabets, or `numer` if the column surely contains integer values only. Following is an explanation for `EFFECT` block in above instruction.

```
EFFECT
  2 cross alpha # treat 2nd column as cross - classified effect ; assuming alphabets
EFFECT
  3 cross alpha # treat 3rd column as cross - classified effect ; assuming alphabets
EFFECT
  4 cov # treat 4th column as covariate
```

Note that, in this case, you can technically use `numer` instead of `alpha` in the second effect because this column contains integer values only. We, nevertheless, recommend a user to always use `alpha` because it can accept any kinds of data including both alphabets and numbers.

Resulting files

Now run RENUMF90 and type the name of instruction file. The program reads the original data and replace the group labels with integer values. Finally, this program generates 3 files. The file name is fixed so if you have the existing file with the same name, the file will be replaced with a new one.

- `renf90.dat`: renumbered data file
- `renf90.par`: a suggested parameter file for BLUPF90
- `renf90.tables`: correspondence table between the original code with new integer value

Let's see the inside of `renf90.dat`.

```
3.0 1 1 1.0
2.0 1 2 1.0
4.0 1 1 2.0
6.0 2 2 2.0
3.0 2 1 1.0
6.0 2 2 2.0
6.0 3 1 2.0
6.0 3 2 1.0
8.0 3 1 1.0
4.0 3 2 2.0
```

Do you see the difference between the original and renumbered files? The order of lines (rows) are preserved but the order of columns is different. Each column has integer values or real (numerical) values only. RENUMF90 change the column order as the following rules.

1. Observations: the program doesn't change the values. - 1st column: single-trait model is assumed and the 1st column is for observations.
2. Effects: the order is determined with the order of `EFFECT` statements in the instruction file. - 2nd column for effect 1 (corresponding to 2nd column in the original file). - 3rd column for effect 2 (3rd column in the original file). - 4th column for effect 3 (4th column in the original file).
3. Passed columns: the program adds the columns specified in `FIELDS_PASSED TO OUTPUT` keyword in instruction. - no additional columns: we didn't set any values for this keyword.

Class variables for effect 1 and 2 (column 2 and 3) are successfully replaced with integer values. For effect 1, A is replaced with 1, B with 2 and C with 3. For effect 2, 1 is 1 and 2 is 2. The correspondence is saved in `renf90.tables`.

```
 Effect group 1 of column 1 with 3 levels, effect # 1
 Value # consecutive number
A 3 1
B 3 2
C 4 3
 Effect group 2 of column 1 with 2 levels, effect # 2
 Value # consecutive number
1 5 1
2 5 2
```

In each Effect group, there are 3 columns: 1st for the original group code, 2nd for the number of times the code occurred in the input and 3rd for the replaced integer value.

RENUMF90 kindly generates a parameter file for BLUPF90.

```
# BLUPF90 parameter file created by RENF90
DATAFILE
 renf90.dat
NUMBER_OF_TRAITS
          1
NUMBER_OF_EFFECTS
          3
OBSERVATION(S)
    1
WEIGHT(S)

EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT[EFFECT NESTED]
  2 3 cross
  3 2 cross
  4 1 cov
RANDOM_RESIDUAL VALUES
   1.0000
```

This looks like perfect. You can directly apply it to BLUPF90, and the program successfully run the analysis. In this parameter file, the parameters are inherited from the instruction file. For example, the above instruction contains 3 effects so `renf90.par` also contains 3 effects corresponding to the description in instruction.

Note that the `EFFECTS:` line has some extra words (`POSITIONS_IN_DATAFILE` etc.) but it is no problem for BLUPF90.

Running BLUPF90 with this renf90.par will produce the same solutions as described before. A reader can confirm the fact with these files.

4.1.2 Optional features in RENUMF90

RENUMF90 can also accept some options with an additional line beginning with `OPTION`. Following options are available.

- `OPTION alpha_size` *n*: change the size of character fields (maximum number of letters in a column with alpha specification) to *n*.

- `OPTION max_string_readline` *n*: change the size of the record length (maximum number of letters in a line) to *n*.
- `OPTION max_field_readline` *n*: change the maximum number of fields (columns) in a line to *n*.

The *n* above is replaced with an actual integer value. See the examples.

```
OPTION max_string_readline 2048
```

RENUMF90 has an useful feature related to options. If an option is not usable in RENUMF90, the program simply pass it to `renf90.par`. So you can put some `OPTION` lines in instruction and run RENUMF90, and you will obtain a perfect parameter file with desired options. See an example instruction.

Listing 4.3: renum1a.txt

```
DATAFILE
rawdata1.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
1.0
EFFECT
2 cross alpha # 1st effect
EFFECT
3 cross alpha # 2nd effect
EFFECT
4 cov # 3rd effect
OPTION max_string_readline 2048
OPTION alpha_size 40
OPTION solv_method FSPAK
```

The generated renf90.par should contain `OPTION solv_method FSPAK`.

4.1.3 What is the best practice?

Suppose you have to perform many analyses with the same data but different models. How many times do you run RENUMF90? Basically there are 2 kinds of solutions.

1. In every analysis, rewrite the instruction and run RENUMF90. Then just use the generated `renf90.par` unchanged.
2. First, prepare the instruction with maximal models and run RENUMF90. Then, every time, copy the generated renf90.par to new file and modify it to fit a model.

The better solution can depend on your situation. If you have a relatively small data set and the analysis will not take a long time, the first approach could be applicable. We, however, highly recommend a user to do the second approach. If your data is large enough and the analysis will take a time, the second option is the only efficient way. Also, in some cases, the generated renf90.par is incomplete. The second approach is more general and it is the usage that the development team assume.

4.1.4 Summary

- RENUMF90 reads a different parameter file from BLUPF90. We call it instruction here.
- RENUMF90 replaces alphabets or symbols with integer values which BLUPF90 can use.
- RENUMF90 generates a renumbered data file, replacement log file and a template parameter file for BLUPF90.
- Information in the template parameter file inherits from an instruction file.
- An instruction file consists of pairs of keyword and values; comments allowed with a # syntax.
- RENUMF90 passes option lines in instruction to `renf90.par`.
- If you use the same data with different analyses, a recommended way is to run RENUMF90 once and modify the generated renf90.par to fit a model you should use.

4.2 Animal model with pedigree file

4.2.1 Random effect specification

In the previous section, we considered a fixed effect model. Here we consider a mixed model with 1 or more random effects. Note that even adding random effects, the renumbered data file will be the same to one for fixed effect model. Only differences when considering random effects in RENUMF90 are: 1) output parameter file contains statements for random effects and 2) pedigree or related files will be properly processed if needed.

First we reconsider the previous example and assume the second effect (S_j) is random.

$$y_{ijk} = A_i + S_j + \beta x_{ijk} + e_{ijk}$$

The raw data file is the same as before but we change the name of the file.

Listing 4.4: rawdata2.txt

```
ID006 A 1 1.0 3.0
ID009 A 2 1.0 2.0
ID012 A 1 2.0 4.0
ID007 B 2 2.0 6.0
ID010 B 1 1.0 3.0
ID013 B 2 2.0 6.0
ID008 C 1 2.0 6.0
ID011 C 2 1.0 6.0
ID014 C 1 1.0 8.0
ID015 C 2 2.0 4.0
```

Assuming the residual variance $\sigma_e^2 = 2.0$ and the random effect's variance $\sigma_s^2 = 1.0$, the parameter file looks like:

Listing 4.5: renum2.txt

```
DATAFILE
rawdata2.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT
```

```
WEIGHT(S)

RESIDUAL_VARIANCE
2.0
EFFECT # 1st effect
2 cross alpha
EFFECT # 2nd effect
3 cross alpha
RANDOM ## treated as a random effect
diagonal
(CO)VARIANCES ## its variance components
1.0
EFFECT # 3rd effect
4 cov
```

This parameter file is also the same as before except we inserted 4 lines: the keyword RANDOM with the value diagonaland the keyword (CO)VARIANCES with the value 1.0 just after the definition of effect 2. Basically, RENUMF90 recognizes an effect as random if RANDOM places after the effect specification; otherwise, the effect is fixed. [1]

This produces the data and parameter files equivalent to ones used in the previous chapter. You can confirm the generated files provides the same solutions as before.

4.2.2 Animal model

Raw data

RENUMF90 can handle an animal model with the same framework using RANDOM and other optional keywords to read a raw pedigree file. We can use the same data file as before. Here we copy the data file to another one with different name.

Listing 4.6: rawdata3.txt

```
ID006 A 1 1.0 3.0
ID009 A 2 1.0 2.0
ID012 A 1 2.0 4.0
ID007 B 2 2.0 6.0
ID010 B 1 1.0 3.0
ID013 B 2 2.0 6.0
ID008 C 1 2.0 6.0
ID011 C 2 1.0 6.0
ID014 C 1 1.0 8.0
ID015 C 2 2.0 4.0
```

Raw pedigree

The following is the raw pedigree which is corresponding the pedigree introduced in the previous chapter.

[1] A tricky feature is to omit (CO)VARIANCES. If you don't put the keyword, RENUMF90 implicitly assumes the variance component is 1.0 by default. We highly recommend to use (CO)VARIANCES option to avoid confusion.

Listing 4.7: rawpedigree3.txt

```
ID001 0 0
ID002 0 0
ID003 0 0
ID004 0 0
ID005 0 0
ID006 0 0
ID007 ID002 ID005
ID008 ID001 ID004
ID009 ID002 ID003
ID010 ID007 ID006
ID011 ID007 ID004
ID012 ID011 ID008
ID013 ID011 ID010
ID014 ID009 ID013
ID015 ID011 ID010
```

The animal ID shares in both raw data and raw pedigree files. A raw pedigree with the following format is allowed for RENUMF90.

- It contains at least 3 columns, animal ID, sire ID, and dam ID, separated with 1 or more spaces. Tab is not allowed.
- The each column can place any position. For example, animal ID can be in column 4.
- Unknown or missing parent must be `0`. Other expressions (like `00` or `00000` or `NA`) are not allowed as missing. They will be recognized as real animals.
- You can also put date of birth (year of birth) as an additional column. The information should be integer value.
- Animals don't have to be ordered chronologically. Any random order is allowed.
- Even an animal is in the sire or dam column, it is not necessarily in the animal column. Such an animal is assumed to have unknown parents (both sire and dam unknown).

Parameter file

Now we assume a mixed model:

$$y_{ijk} = A_i + S_j + \beta x_{ijk} + u_k + e_{ijk}$$

where A_i and S_j are fixed effects, β is a fixed regression, u_k is additive genetic effect and e_{ijk} is random residual. The variances are assumed as $\sigma_e^2 = 2.0$ and $\sigma_u^2 = 0.5$. Now we add 1 `EFFECT` statement for the additive genetic effect with the `RANDOM` keyword.

Listing 4.8: renum3.txt

```
DATAFILE
rawdata3.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
```

```
2.0
EFFECT # 1st effect fixed
2 cross alpha
EFFECT # 2nd effect fixed
3 cross alpha
EFFECT # 3rd effect fixed
4 cov
EFFECT # 4th effect
1 cross alpha
RANDOM ## treated as a random effect
animal
FILE ## pedigree file
rawpedigree3.txt
FILE_POS ## animal, sire and dam IDs, and two 0s
1 2 3 0 0
(CO)VARIANCES ## its variance component
0.5
```

We can see two new keywords here.

- FILE: the name of raw pedigree file.
  - rawpedigree3.txt
- FILE_POS: position of IDs in the raw pedigree file; 5 items needed.
  - 1st item = the position of animal ID (1 for the 1st column)
  - 2nd item = the position of sire ID (2 for the 2nd column)
  - 3rd item = the position of dam ID (3 for the 3rd column)
  - 4th item = usually 0 (will be explained in the later chapter)
  - 5th item = usually 0 (will be explained in the later chapter)

Be careful the order of keywords for random effects. RENUMF90 only accept the keywords properly ordered in instruction. In this example, the order should be:

1. RANDOM
2. FILE
3. FILE_POS
4. (CO)VARIANCES

We have many more keywords related to random effects. The keywords should be ordered to be read by RENUMF90. We will review this ordering issue later.

Output files

After running RENUMF90, you can see some files in the same location: renf90.dat, renf90.par, renf90.tables and renadd04.ped. The last one is a renumbered pedigree. Why 4? This 4 comes from the position of the effect in instruction: the additive genetic effect defined as the 4th EFFECT statement. If your additive genetic effect is in the 6th EFFECT statement, the name of pedigree file will be renadd06.ped.

The renumbered pedigree renadd04.ped actually contains 10 columns.

```
1 7 5 1 0 2 1 0 0 ID015
13 0 0 3 0 0 0 0 2 ID004
```

```
11 0 0 3 0 0 0 0 1 ID005
2 0 0 3 0 0 1 0 1 ID006
3 12 11 1 0 2 1 2 0 ID007
4 14 13 1 0 2 1 0 1 ID008
5 3 2 1 0 2 1 0 2 ID010
6 12 15 1 0 2 1 1 0 ID009
7 3 13 1 0 2 1 3 0 ID011
8 7 4 1 0 2 1 0 0 ID012
14 0 0 3 0 0 0 1 0 ID001
9 7 5 1 0 2 1 0 1 ID013
12 0 0 3 0 0 0 2 0 ID002
10 6 9 1 0 2 1 0 0 ID014
15 0 0 3 0 0 0 0 1 ID003
```

The column 1 is for renumbered animal ID, column 2 for sire ID and column 3 for dam ID. The remaining columns contain additional information. The last (10th) column is the original ID. Many of them are actually ignored in BLUPF90. [2] The official manual explains the meaning of each column.

```
 1. animal number (from 1)
 2. parent 1 number or unknown parent group number for parent 1
 3. parent 2 number or unknown parent group number for parent 2
 4. 3 minus number of known parents
 5. known or estimated year of birth (0 if not provided)
 6. number of known parents (parents might be eliminated if not contributing; if animal has
    genotype 10+number of know parents
 7. number of records
 8. number of progeny (before elimination due to other effects) as parent 1
 9. number of progeny (before elimination due to other effects) as parent 2
10. original animal id
```

Note that conversion table for animal ID is not saved in renf90.tables. The correspondence can be found only in the renumbered pedigree file (column 1 and 10).

Test run with the renumbered data

You can run BLUPF90 with the generated parameter file and obtain the solutions of the equation. The analysis mimics the previous one in the previous chapter so the results should be identical (without numerical error). You will find the BLUE is the same and the estimated breeding values (BLUP) are also the same but the order of animals are different. The following is a combination of previous ones (left) and the current ones (right).

```
1 1 1 0.20259644 1 1 1 0.20259637
1 1 2 2.21996461 1 1 2 2.21996452
1 1 3 3.16680208 1 1 3 3.16680200
1 2 1 2.27537292 1 2 1 2.27537284
1 2 2 1.71160537 1 2 2 1.71160529
1 3 1 0.52442755 1 3 1 0.52442750
1 4 1 -0.03487115 1 4 1 -0.25707432
1 4 2 0.08280493 1 4 2 -0.17565610
1 4 3 0.03843921 1 4 3 0.10794666
1 4 4 0.04492008 1 4 4 -0.02984647
1 4 5 0.04436203 1 4 5 -0.25282594
1 4 6 -0.17565609 1 4 6 0.09906235
```

[2] The 4th column will be used if you put certain options in the parameter file. We don't explain such options here.

```
1 4 7 0.10794667 1 4 7 0.15622415
1 4 8 -0.02984646 1 4 8 0.10874295
1 4 9 0.09906236 1 4 9 0.16426464
1 4 10 -0.25282594 1 4 10 0.34296713
1 4 11 0.15622415 1 4 11 0.04436204
1 4 12 0.10874296 1 4 12 0.08280494
1 4 13 0.16426465 1 4 13 0.04492008
1 4 14 0.34296714 1 4 14 -0.03487115
1 4 15 -0.25707431 1 4 15 0.03843922
```

This is apparently form the different order of animals in the pedigree file between 2 data sets. RENUMF90 orders pedigree animals as follows.

1. First, the program assigns smaller numbers to animals with record(s). The order of animals are unpredictable (i.e. not following the order found in the data file).
2. Then, the program assigns larger numbers to animals found only in pedigree.

Note that RENUMF90 doesn't order the pedigree animals chronologically (from oldest to youngest). No options are available to order the animals chronologically. The above rules are from the simplification of implementation but are actually useful for fitting the permanent environmental effect in repeatability model.

Usually you will combine the solutions (estimated breeding values) with the original animal ID. This can be done with several ways e.g. Linux (Unix) tools, database management systems, statistical languages like R, even Microsoft Excel. We will show a way to to this using Linux/Unix tools in the later chapter.

4.2.3 More than 1 random effect

As the last demonstration of RENUMF90 in this section, let us assume the effect S_j is also random as well as u_k in the above model. The variance components are $\sigma_e^2 = 2.0$, $\sigma_s^2 = 1.0$, and $\sigma_u^2 = 0.5$. A suggested instruction file is following.

Listing 4.9: renum3a.txt

```
DATAFILE
rawdata3.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
2.0
EFFECT # 1st effect fixed
2 cross alpha
EFFECT # 2nd effect fixed
3 cross alpha
RANDOM ## treated as a random effect
diagnonal
(CO)VARIANCES ## its variance component
1.0
EFFECT # 3rd effect fixed
```

```
4 cov
EFFECT # 4th effect
1 cross alpha
RANDOM ## treated as a random effect
animal
FILE ## pedigree file
rawpedigree3.txt
FILE_POS ## animal, sire and dam IDs with two 0s
1 2 3 0 0
(CO)VARIANCES ## its variance component
0.5
```

This is an combination of the first and second examples in this section. You can easily figure it out how each statement works.

4.2.4 Summary

- RENUMF90 treats a effect with the `RANDOM` keyword as a random effect.
- The `RANDOM` keyword may require other keywords.
- If a random effect doesn't need pedigree, only use `RANDOM` and `(CO)VARIANCES`.
- If a random effect need pedigree, use `RANDOM`, `FILE`, `FILE_POS`, and `(CO)VARIANCES` in this order.
- A raw pedigree file contains at least 3 columns: animal ID, sire ID and dam ID. The position of a column is arbitrary.
- The renumbered pedigree file has a form `renaddnn.ped` where `nn` is the position of this effect in instruction.
- The renumbered pedigree file has 10 columns including the original ID.
- The animals with records have smaller numbers than animals found only in pedigree.
- RENUMF90 supports 1 or more random effects.

4.3 Genomic model with SNP-marker file

4.3.1 Required files

Single-step GBLUP (ssGBLUP) model is an extension to animal model. RENUMF90 can handle ssGBLUP using a similar parameter file for an animal model with an additional option. Let us use the previous files. The following files are exactly the same to previous ones with different names as always.

Listing 4.10: rawdata4.txt

```
ID006 A 1 1.0 3.0
ID009 A 2 1.0 2.0
ID012 A 1 2.0 4.0
ID007 B 2 2.0 6.0
ID010 B 1 1.0 3.0
ID013 B 2 2.0 6.0
ID008 C 1 2.0 6.0
ID011 C 2 1.0 6.0
ID014 C 1 1.0 8.0
ID015 C 2 2.0 4.0
```

The pedigree file is as follows.

Listing 4.11: rawpedigree4.txt

```
ID001 0 0
ID002 0 0
ID003 0 0
ID004 0 0
ID005 0 0
ID006 0 0
ID007 ID002 ID005
ID008 ID001 ID004
ID009 ID002 ID003
ID010 ID007 ID006
ID011 ID007 ID004
ID012 ID011 ID008
ID013 ID011 ID010
ID014 ID009 ID013
ID015 ID011 ID010
```

A SNP marker file should be prepared as a fixed-length text file with 2 columns: the original ID for column 1 and genotypes for column 2. See detailed explanation the previous chapter for its format of the marker file. We also use the same one shown before.

Listing 4.12: snp4.txt

```
ID002 121202011100121121200010200120202220111200011002010201100101221211012210000121010111220021001022220
ID004 222202020011012221012100110002222110211100001001120202000200222200022200002012000002020020000020022O
ID006 212202010110022112111121210011211111221200010101221111100120121111202210020122011002220022002011112O
ID010 211112110111121012121111221020202121121100011002121111100110121110112210011122100002220012002010112O
ID011 221112120012111122021101121011212111201000120011202020002002222000222000021221000021200110010101220
ID012 222202020011012221012100110002222110211100001001120202000200222200022200002012000002020020000020022O
ID013 212202010121012112122210120011212210221200001001221111100110121110112210011022000002120021001020012O
ID015 212202010121012112122210120011212210221200001001221111100110121110112210011022000002120021001020012O
```

The marker file is never altered nor duplicated by RENUMF90. The program just read the 1st column (the original ID) and make a table relating the renumbered animal ID to the original animal ID. The table will be saved as a cross-reference file.

We assume the same animal model described before except genomic information in included. The following is a parameter file to handle the SNP file.

Listing 4.13: renum4.txt

```
DATAFILE
rawdata4.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
2.0
EFFECT # 1st effect fixed
```

```
2 cross alpha
EFFECT # 2nd effect fixed
3 cross alpha
EFFECT # 3rd effect fixed
4 cov
EFFECT # 4th effect
1 cross alpha
RANDOM ## treated as a random effect
animal
FILE ## pedigree file
rawpedigree4.txt
FILE_POS ## animal, sire and dam IDs with two 0s
1 2 3 0 0
SNP_FILE ## SNP marker file
snp4.txt
(CO)VARIANCES ## its variance component
0.5
```

A new keyword `SNP_FILE` with the name of marker file tells RENUMF90 to properly treat the marker file. This keyword should be placed just after `FILE_POS`.

4.3.2 Renumbered files

Running RENUMF90 with above instruction, it generates several files in the same folder (directory). With this example, generated files are `renf90.dat`, `renf90.par`, `renf90.tables`, `renadd04.ped`, and `snp4.txt_XrefID`. The last one is the cross-reference file. It contains 2 columns for the renumbered ID and the original ID as follows.

```
11 ID002
12 ID004
2 ID006
5 ID010
7 ID011
8 ID012
9 ID013
1 ID015
```

The order of genotyped animal is the same as the marker file. The name of this file is automatically determined as the original SNP marker file plus XrefID and you can't put the name in instruction.

The renumbered parameter file renf90.par looks similar as before except the option found in the last line.

```
OPTION SNP_file snp4.txt
```

The parameter file doesn't cite the cross-reference file because, by default, BLUPF90 programs assume the name of cross-reference file is the original SNP marker file + `XrefID`. Usually you don't use different cross-reference file so you just keep this option line.

The renumbered pedigree is actually different form the previous one.

```
1 7 5 1 0 12 1 0 0 ID015
12 0 0 3 0 10 0 0 2 ID004
13 0 0 3 0 0 0 0 1 ID005
2 0 0 3 0 10 1 0 1 ID006
3 11 13 1 0 2 1 2 0 ID007
4 14 12 1 0 2 1 0 1 ID008
5 3 2 1 0 12 1 0 2 ID010
6 11 15 1 0 2 1 1 0 ID009
7 3 12 1 0 12 1 3 0 ID011
8 7 4 1 0 12 1 0 0 ID012
14 0 0 3 0 0 0 1 0 ID001
9 7 5 1 0 12 1 0 1 ID013
11 0 0 3 0 10 0 2 0 ID002
10 6 9 1 0 2 1 0 0 ID014
15 0 0 3 0 0 0 0 1 ID003
```

With genomics, RENUMF90 assigns new integer values to animals with the following rules.

1. First, the program assigns smaller numbers to animals with record(s). The order of animals are unpredictable (i.e. not following the order found in the data file).
2. Secondly, the program assigns numbers to genotyped animals. The order of animals are unpredictable (i.e. not following the order found in the marker file).
3. Lastly, the program assigns larger numbers to animals found only in pedigree.

You can find the genotyped animals in the renumbered pedigree file. An animal is genotyped if the 6th column is 10 or greater.

4.3.3 Summary

- RENUMF90 supports ssGBLUP.
- The instruction for ssGBLUP is the same to animal model except additional `SNP_FILE` keyword and the name of marker file.
- RENUMF90 doesn't change the SNP marker file. Instead it creates a cross-reference file relating the original ID to the renumbered ID. The name of cross-reference file ends with `XrefID` by default.
- A suggested `renf90.par` contains the option line with SNP file.
- The order of animals in pedigree is determined as phenotyped animals the first, genotyped animals the second and the other animals the last.

4.4 Multiple-trait models

4.4.1 Model

For renumbering on a multiple-trait model, we still need a very similar instruction file to single-trait one. Only differences are 1) specification of model in each trait and 2) use of covariance matrix instead of scalar expression.

We will consider a 2-trait model with equal design matrices, which means all traits have the same model. The mathematical models are

$$y_{ijk:1} = A_{i:1} + S_{j:1} + \beta_1 x_{ijk:1} + u_{k:1} + e_{ijk:1}$$
$$y_{ijk:2} = A_{i:2} + S_{j:2} + \beta_2 x_{ijk:2} + u_{k:2} + e_{ijk:2}$$

where $y_{ijk:t}$ is a observation for trait t, $A_{i:t}$ is the fixed effect for trait t, $S_{i:t}$ is also the fixed effect for trait t, β_t is the fixed regression coefficient for trait t, $x_{ijk:t}$ is a covariate for trait t, $u_{k:t}$ is the additive genetic

effect for trait t, and $e_{ijk:t}$ is the random residual effect. The genetic ($\mathbf{G}_0$) and residual ($\mathbf{R}_0$) covariance matrices are

$$\mathbf{G}_0 = \begin{bmatrix} 0.50 & -0.25 \\ -0.25 & 1.00 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_0 = \begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 1.5 \end{bmatrix}$$

4.4.2 Required files

The raw data file is identical to the previous single-trait situation except the new column (trait 2) added. The data file should contains all required effects and observations for all traits.

Listing 4.14: rawdata5.txt

```
ID006 A 1 1.0 3.0 4.5
ID009 A 2 1.0 2.0 7.5
ID012 A 1 2.0 4.0 3.5
ID007 B 2 2.0 6.0 -0.5
ID010 B 1 1.0 3.0 5.5
ID013 B 2 2.0 6.0 1.5
ID008 C 1 2.0 6.0 -1.5
ID011 C 2 1.0 6.0 2.5
ID014 C 1 1.0 8.0 0.5
ID015 C 2 2.0 4.0 4.5
```

Pedigree file is also the same to the previous one.

Listing 4.15: rawpedigree5.txt

```
ID001 0 0
ID002 0 0
ID003 0 0
ID004 0 0
ID005 0 0
ID006 0 0
ID007 ID002 ID005
ID008 ID001 ID004
ID009 ID002 ID003
ID010 ID007 ID006
ID011 ID007 ID004
ID012 ID011 ID008
ID013 ID011 ID010
ID014 ID009 ID013
ID015 ID011 ID010
```

The instruction file for this 2-trait model is as follows.

Listing 4.16: renum5.txt

```
DATAFILE
rawdata5.txt
TRAITS # 2-trait model: put 2 positions
5 6
```

```
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
2.0 1.0
1.0 1.5
EFFECT # 1st effect fixed
2 2 cross alpha
EFFECT # 2nd effect fixed
3 3 cross alpha
EFFECT # 3rd effect fixed
4 4 cov
EFFECT # 4th effect
1 1 cross alpha
RANDOM ## treated as a random effect
animal
FILE ## pedigree file
rawpedigree5.txt
FILE_POS ## animal, sire and dam IDs, and two 0s
1 2 3 0 0
(CO)VARIANCES ## its variance component
 0.50 -0.25
-0.25 1.00
```

For a multiple-trait model, you can check the following points in an instruction file. An incorrect description in these statements falls in the sudden stop of RENUMF90.

- `TRAITS`: Enumerate the positions of observations in the data file. If you have 2 traits, you should put 2 numbers (positions) here.
- `RESIDUAL_VARIANCE`: Put a residual covariance matrix here. The whole matrix is needed (all upper, lower and diagonal elements).
- `EFFECT`: Enumerate the position of effects for each trait in the data file; then put the effect type (and data type). For example, with a 2-trait model, first you should put 2 numbers (for positions of effects for trait 1 and 2), 1 keyword (`cross` or `cov`) and ,possibly, data type (`numer` or `alpha`) if the data type is cross.
- `(CO)VARIANCES`: Put a covariance matrix here. The whole matrix is needed.

Run RENUMF90 with above files, and you can see the parameter file is configured with the 2-trait model.

4.4.3 Missing observations

You can include missing observations in the data file. RENUMF90 doesn't distinguish the real and the missing observations i.e. the program just read the raw data and just pass the observation to the renumbered file. Although RENUMF90 doesn't care about missing observations, BLUPF90 really cares the values. So the generated parameter file (renf90.par) should include an option to define the missing code. You can write the following option in the instruction file, and RENUMF90 passes it to renf90.par.

```
OPTION missing -999
```

This example defines a observation with `-999` as a missing observation. The default is 0 (i.e. if you don't define any missing code, the program assumes 0 as missing).

Note that RENUMF90 recognizes 0 as a missing observation only in the computations of basic statistics (e.g. average and standard deviation). The program ignores 0 for the calculations but the results doesn't affect any output files.

4.4.4 Different models across traits (Unequal design matrices)

Now we consider a multiple-trait model which has different mathematical models across traits. Is is typically called a model with unequal design matrices. Let's see the following 2-trait model.

$$\begin{aligned} y_{ijk:1} &= A_{i:1} + \quad \beta_1 x_{ijk:1} + u_{k:1} + e_{ijk:1} \\ y_{ijk:2} &= A_{i:2} + S_{j:2} + \beta_2 x_{ijk:2} + u_{k:2} + e_{ijk:2} \end{aligned}$$

In the models, the $S_{j:1}$ is missing for trait 1. To support the models, you can just change 1 line in the previous instruction file.

```
EFFECT # 2nd effect fixed
0 3 cross alpha
```

The position `0` means this effect is missing for this trait.

We further consider the following 2-trait model.

$$\begin{aligned} y_{ijk:1} &= A_{i:1} \quad + \beta_1 x_{ijk:1} + u_{k:1} + e_{ijk:1} \\ y_{ijk:2} &= \quad S_{j:2} + \beta_2 x_{ijk:2} + u_{k:2} + e_{ijk:2} \end{aligned}$$

In this case, $S_{j:1}$ is missing for trait 1 and $A_{i:2}$ is missing for trait 2. Applying the above principle to this case, you can figure out the following solution (just showing a piece of instruction file).

```
EFFECT # 1st effect fixed
2 0 cross alpha
EFFECT # 2nd effect fixed
0 3 cross alpha
```

Above statements are easy to understand which effects are missing for a particular trait.

There is another way to handle the model. You can combine above 2 `EFFECT` lines into 1 as follows.

```
EFFECT # 1st and 2nd effects fixed
2 3 cross alpha
```

This is especially useful when the different effects are seen as similar types of effects. For example, in dairy cattle, a contemporary group effect (like HYS: herd-year-season) are commonly applied to milk yield but the definition should be different across parities; HYS for parity 1 and HYS for parity 2 are different but considered as a similar type of effects. With this compact definition, memory requirements would reduce. If you are confused with the notations, you don't use this type of statements.

4.4.5 Summary

- RENUMF90 supports multiple-trait models.
- Carefully describe `TRAITS`, `RESIDUAL_VARIANCE`, `EFFECT` and `(CO)VARIANCES` in an instruction file.
- `TRAIT` has the same number of entries to the number of traits.
- Covariance matrix should be wholly stated.
- `EFFECT` describe the position of effect in each trait.
- Missing observations are passed to `renf90.dat`. You would need `OPTION` missing to the instruction file.
- RENUMF90 supports a multiple-trait model with unequal design matrices.

4.5 Advanced usage of RENUMF90

In this section, we will introduce some advanced features in RENUMF90. We commonly use the following raw data and pedigree file. They are actually come from the multiple-trait model shown in the previous chapter.

Listing 4.17: rawdata9.txt

```
ID006 A 1 1.0 3.0 4.5
ID009 A 2 1.0 2.0 7.5
ID012 A 1 2.0 4.0 3.5
ID007 B 2 2.0 6.0 -0.5
ID010 B 1 1.0 3.0 5.5
ID013 B 2 2.0 6.0 1.5
ID008 C 1 2.0 6.0 -1.5
ID011 C 2 1.0 6.0 2.5
ID014 C 1 1.0 8.0 0.5
ID015 C 2 2.0 4.0 4.5
```

The pedigree file is as follows.

Listing 4.18: rawpedigree9.txt

```
ID001 0 0
ID002 0 0
ID003 0 0
ID004 0 0
ID005 0 0
ID006 0 0
ID007 ID002 ID005
ID008 ID001 ID004
ID009 ID002 ID003
ID010 ID007 ID006
ID011 ID007 ID004
ID012 ID011 ID008
ID013 ID011 ID010
ID014 ID009 ID013
ID015 ID011 ID010
```

4.5.1 Combined effects

RENUMF90 can treat a field as a combination of the existing fields. This is useful when an interaction effect is incorporated. The keyword COMBINE perform this. The following example is from the previous section and the keyword is added.

Listing 4.19: renum9a.txt

```
COMBINE
6 2 3
DATAFILE
rawdata9.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
1.0
EFFECT # 1st effect
2 cross alpha
EFFECT # 2nd effect
3 cross alpha
EFFECT # 3rd effect
4 cov
EFFECT # interaction effect 1 x 2
6 cross alpha
```

The COMBINE statement should be used as follows.

- COMBINE must appear in the beginning of the recipe file (before DATAFILE).
- At least 2 values are needed; The 2nd or later values will be combined into 1 field. The field can be referred as the first value.
- In this case, `6 2 3` = column 2 and 3 are combined into column 6 (i.e. define new column 6 as a combination of 2 and 3).
- You can use any number in the first value even if your data file doesn't have such column.
- The combined field can be used in the EFFECT statement.

You can run RENUMF90 and check `renf90.tables`. The column 2 and 3 are successfully combined into one effect (group 4).

```
 Effect group 1 of column 1 with 3 levels , effect # 1
 Value # consecutive number
A 3 1
B 3 2
C 4 3
 Effect group 2 of column 1 with 2 levels , effect # 2
 Value # consecutive number
1 5 1
2 5 2
 Effect group 4 of column 1 with 6 levels , effect # 4
```

```
 Value # consecutive number
A1 2 1
A2 1 2
B1 1 3
B2 2 4
C1 2 5
C2 2 6
```

4.5.2 Pedigree manipulation

RENUMF90 prune the pedigree. By default, the program traces back 3-generation (up to great-grand sires and dams) from the animals with phenotype or genotype. The number of generations back can be changed with the option PED_DEPTH, which should place just after SNP_FILE (or FILE_POS if no genetic markers are used). The following statements define 10 generation to be traced.

```
PED_DEPTH
10
```

If you put a large number (like 100), you can consider all ancestors to be traced back from the current animals. If you put 0, RENUMF90 tries to include all animals found in the raw pedigree file even if the pedigree animals are not related to the animals with phenotype or genotype.

4.5.3 Animal model options

RENUMF90 supports permanent environmental (PE) effect and maternal genetic (MG) and environmental (MPE) effects as additional random effects in an animal model using the OPTIONAL keyword. The following is a parameter file with maximal options (PE, MG and MPE).

Listing 4.20: renum9b.txt

```
DATAFILE
rawdata9.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
1.0
EFFECT # 1st effect
2 cross alpha
EFFECT # 2nd effect
3 cross alpha
EFFECT # 3rd effect
4 cov
EFFECT # 4th effect = animal
1 cross alpha
RANDOM
animal
```

```
OPTIONAL
pe mat mpe
FILE
rawpedigree9.txt
(CO)VARIANCES
1.0 0.2
0.2 0.3
(CO)VARIANCES_PE
1.5
(CO)VARIANCES_MPE
0.5
```

You can read this parameter file as follows.

- The OPTIONAL keyword places just after the EFFECT defining additive genetic effect.
- Possible values are pe for an animal's permanent environmental effect, mat for maternal genetic effect and mpe for maternal environmental effect. They are totally optional; you can use only the options you want.
- If you put mat, you should put genetic covariance matrix for 1) direct and 2) maternal genetic effects at (CO)VARIANCES. Otherwise, you just put direct genetic variance.
- If you put pe, you should put permanent environmental covariance matrix at (CO)VARIANCES_PE. Otherwise, you can omit it.
- If you put mpe, you should put maternal permanent environmental covariance matrix at (CO)VARIANCES_MPE. Otherwise, you can omit it.

4.5.4 Definition of unknown parent groups (UPGs)

RENUMF90 can assigns UPGs to unknown parents. There are 2 ways to do this but here we just introduce the one approach (because of its simplicity). The following keyword generates UPGs:

```
UPG_TYPE
in_pedigrees
```

The option should be just after PED DEPTH. With this option, RENUMF90 interprets a negative sire (or dam) ID as an UPG. So you should prepared different columns for sires and dams which contain negative integers for unknown animals instead of 0. Also, all the real animal should be in the pedigree file as an animal.

The following is a simple example for UPGs. The 4th column is for sires with groups (1 and 2) and the 5th column is for dams with groups (3 and 4). You can specify the column 4 (instead of 2) as the sire and column 5 (instead of 3) as the dam with the FILE_POS keyword.

Listing 4.21: rawpedigree9c.txt

```
ID001 0 0 -1 -4
ID002 0 0 -2 -3
ID003 0 0 -1 -3
ID004 0 0 -2 -3
ID005 0 0 -2 -4
ID006 0 0 -1 -3
ID007 ID002 ID005 ID002 ID005
ID008 ID001 ID004 ID001 ID004
```

```
ID009 ID002 ID003 ID002 ID003
ID010 ID007 ID006 ID007 ID006
ID011 ID007 ID004 ID007 ID004
ID012 ID011 ID008 ID011 ID008
ID013 ID011 ID010 ID011 ID010
ID014 ID009 ID013 ID009 ID013
ID015 ID011 ID010 ID011 ID010
```

The instruction file can be as follows.

Listing 4.22: renum9c.txt

```
DATAFILE
rawdata9.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
1.0
EFFECT # 1st effect
2 cross alpha
EFFECT # 2nd effect
3 cross alpha
EFFECT # 3rd effect
4 cov
EFFECT # 4th effect = animal
1 cross alpha
RANDOM
animal
FILE
rawpedigree9c.txt
FILE_POS # positions of animal, sire, dam, 0, 0
1 4 5 0 0
UPG_TYPE
in_pedigrees
(CO)VARIANCES
1.0
```

The resulting pedigree file is as follows. The number of animals is 15 but you can see the code 16, 17, 18 and 19 in the sire or dam columns. The code greater than the number of animals is an UPG code.

```
1 7 5 1 0 2 1 0 0 ID015
13 17 18 3 0 0 0 0 2 ID004
11 17 19 3 0 0 0 0 1 ID005
2 16 18 3 0 0 1 0 1 ID006
3 12 11 1 0 2 1 2 0 ID007
4 14 13 1 0 2 1 0 1 ID008
5 3 2 1 0 2 1 0 2 ID010
6 12 15 1 0 2 1 1 0 ID009
```

```
7 3 13 1 0 2 1 3 0 ID011
8 7 4 1 0 2 1 0 0 ID012
14 16 19 3 0 0 0 1 0 ID001
9 7 5 1 0 2 1 0 1 ID013
12 17 18 3 0 0 0 2 0 ID002
10 6 9 1 0 2 1 0 0 ID014
15 16 18 3 0 0 0 0 1 ID003
```

4.5.5 Considering inbreeding coefficients in $\mathbf{A}^{-1}$

To consider inbreeding coefficients in $\mathbf{A}^{-1}$, we need a special 4-digit code (inb/upg code) in the 4th column in the pedigree file. You can manually put the column after renumbering, or you can use RENUMF90 to generate the pedigree file with the inb/upg code with the `INBREEDING` keyword. The option should be placed after `UPG_TYPE`.

The basic usage of this option is shown below.

```
INBREEDING
pedigree
```

You should literally put the word `pedigree` here (you shouldn't replace it with the pedigree file name). With this option, RENUMF90 calculates inbreeding coefficients using the pedigree to be saved in the file `renaddxx.ped` i.e. pruned pedigree.

If you want to supply external inbreeding coefficients stored in a file, use the different option. Assuming your file is inb.txt, the following option can be used:

```
INBREEDING
file_inb.txt
```

The file is a space-separated text file with 2 columns: 1) the original animal ID and 2) inbreeding coefficient (ranging from 0 to 1).

The following example uses the pedigree option.

Listing 4.23: renum9d.txt

```
DATAFILE
rawdata9.txt
TRAITS
5
FIELDS_PASSED TO OUTPUT

WEIGHT(S)

RESIDUAL_VARIANCE
1.0
EFFECT # 1st effect
2 cross alpha
EFFECT # 2nd effect
3 cross alpha
```

```
EFFECT # 3rd effect
4 cov
EFFECT # 4th effect = animal
1 cross alpha
RANDOM
animal
FILE
rawpedigree9.txt
INBREEDING
pedigree
(CO)VARIANCES
1.0
```

The resulting pedigree file `renadd04.ped` has the inb/upg code.

Listing 4.24: renadd04.ped

```
1 7 5 2000 0 2 1 0 0 ID015
13 0 0 1000 0 0 0 0 2 ID004
11 0 0 1000 0 0 0 0 1 ID005
2 0 0 1000 0 0 1 0 1 ID006
3 12 11 2000 0 2 1 2 0 ID007
4 14 13 2000 0 2 1 0 1 ID008
5 3 2 2000 0 2 1 0 2 ID010
6 12 15 2000 0 2 1 1 0 ID009
7 3 13 2000 0 2 1 3 0 ID011
8 7 4 2000 0 2 1 0 0 ID012
14 0 0 1000 0 0 0 1 0 ID001
9 7 5 2000 0 2 1 0 1 ID013
12 0 0 1000 0 0 0 2 0 ID002
10 6 9 2133 0 2 1 0 0 ID014
15 0 0 1000 0 0 0 0 1 ID003
```

The 4th column contains the inb/upg code. When an animal has non-inbred parents, the value should be 1000 with unknown parents, 1333 with one parent unknown or 2000 with known parents. Animal 10 (ID014) has inbred parents so the inb/upg code is different from other animals.

4.5.6 Order of keywords

The keywords used in a recipe file should be ordered following the manual. Here we show the exact order of keywords supported by RENUMF90.

Keyword	optional	possible values
`COMBINE`	optional	definition of new field as a combination of existing fields
`DATAFILE`	mandatory	name of raw data file
`TRAITS`	mandatory	positions of observations in the raw data file
`FIELDS_PASSED`	mandatory	positions of items in the raw data file to be passed to `renf90.dat`
`WEIGHT(S)`	mandatory	positions of weights in the raw data file
`RESIDUAL_VARIANCE`	mandatory	residual covariance matrix
`EFFECT`	mandatory	effect description
`NESTED`	optional	positions of nested covariates

Keyword	optional	possible values
RANDOM	optional	declaration of random effect
FILE	optional	name of raw pedigree file
FILE_POS	optional	positions of animal ID, sire ID and dam ID
SNP_FILE	optional	name of SNP marker file
PED_DEPTH	optional	the maximum generation back from animals with phenotype or genotype
GEN_INT	optional	generation interval to set unknown parent groups (UPG)
REC_SEX	optional	check if records are found in specific sex
UPG_TYPE	optional	UPG specification
INBREEDING	optional	create pedigree file with inbreeding
RANDOM_REGRESSION	optional	put covariates for random regressions
RR_POSITION	optional	positions of covariates for random regressions
(CO)VARIANCES	optional	covariance components
(CO)VARIANCES_PE	optional	covariance components for animal's PE effects
(CO)VARIANCES_MPE	optional	covariance components for maternal PE effects
OPTION	optional	option parameters

4.6 What if I don't want to use RENUMF90?

It happens that the user does not want to use RENUMF90. Common reasons for this are:

- It is simulated data and it is already renumbered
- I do my own renumbering and I want to use it
- I have a complex model and I want better control on the recoded numbers

In the next section we will give instructions on how the files should be coded. We will *not* give software but you may take a look here.

4.6.1 Main requirements of recoding for BLUPF90 programs

- Levels of effects should be coded with consecutive numbers. The order may be "natural" or not. For instance, if years in data are (2010, 2011, 2013, 2014) may be renumbered as (1,2,3,4) or (2,1,3,4).
- Animal is another effect, so it has receive codes from 1 to the number of animals *in the pedigree file* (this is usually greater than the number of animals in the data file).
- The animal effect does not need to be renumbered in consecutive order (parents before offspring), but this order is a correct one and may be used. So for instance this renumbering is correct:

Listing 4.25: ped_kempthorne

```
A 0 0
B 0 0
D A B
E A D
F B E
Z A B
```

could be recoded as

Listing 4.26: ped_kempthorne_recoded

```
1 0 0 A 0 0
2 0 0 B 0 0
3 1 2 D A B
4 1 2 Z A B
5 1 3 E A D
6 2 5 F B E
```

(the last three columns are not needed but help the visualization).

4.6.2 Unknown parent groups

However, if unknown parent groups (UPGs) are in the pedigree, things get more complicated. Assume that we have *n real* animals and *m* UPGs. Animals have to be renumbered with codes from 1 to *n*, and UPGs have to be renumbered with codes from $n+1$ to $n+m$. In addition, because they do not have ancestors, the UPGs must not have a line on their own in the pedigree file for BLUPF90. In that way, BLUPF90 "knows" that a given number is a UPG and not an individual. For instance, in the previous example assume that UPGs are "unknown2000" and "unknown2004":

Listing 4.27: ped_kempthorne_unknown parent groups

```
A unknown2000 unknown2000
B unknown2000 unknown2004
D A B
E A D
F B E
Z A B
```

It should be recoded as

Listing 4.28: ped_kempthorne_unknown parent groups_recoded

```
1 7 7 A unknown2000 unknown2000
2 7 8 B unknown2000 unknown2004
3 1 2 D A B
4 1 3 E A D
5 2 4 F B E
6 1 2 Z A B
```

4.6.3 Genotypes

Genotypes should be coded as 0/1/2 for {AA,Aa,aa} and 5 for missing as described elsewhere in this tutorial, in a text format with fixed file. The first column of this file contains (possibly alphanumeric) identifiers for animals, for instance:

Listing 4.29: genotype_file

```
A 120120202102111
Z 121111202111010
```

The only important requirement is the creation of the cross-reference file (usually called `_XrefID`). This file contains the original ID and the new ID in the renumbered pedigree and data files. In the previous example it would be:

Listing 4.30: genotype_file_XrefID

```
1 A
6 Z
```

4.6.4 Overall mean

BLUPF90 programs do *not* include by default an overall mean !! If you need one, add it by yourself as a column of 1s in the data file

5 Variance component estimation

5.1 Restricted (residual) maximum likelihood with AIREMLF90

5.1.1 REML software

REML (restricted/residual maximum likelihood) is a popular method to estimate variance components in animal breeding. BLUPF90 family currently provides 2 kinds of programs supporting different algorithm.

- AIREMLF90 supports Average Information (AI) algorithm.
- REMLF90 supports Expectation-maximization (EM) algorithm.

The AI algorithm is an iterative method and it needs initial values of variance components. In the first round, the algorithm produces the new values based on the initial values. In a subsequent round, the new values come from the previous values. With several iterations, the values are expected to approach closely enough the estimates of variance components. The EM algorithm is another iterative scheme and the computational details are totally different. See Misztal (2008) for detailed differences between the two algorithms.

The usage of AIREMLF90 and REMLF90 are very similar and the estimates will be identical in theory. Here we will introduce the usage of AIREMLF90 only because this program has been a first choice of REML computations and it provides the (approximated) standard error of genetic parameters. Also AIREMLF90 has the EM-algorithm as a special case, so that this algorithm can be equally used.

5.1.2 Preparation

AIREMLF90 uses exactly the same parameter file as BLUPF90 uses. AIREMLF90 reads the variance components from a parameter file as initial values (starting values). In this section, we will use simulated files that are larger than before. You can download the files from Github (`https://github.com/masuday/data`).

- `simdata.txt` : data file
- `simped.txt` : pedigree file

The pedigree file contains 3 columns: animal, sire and dam. The data file has 12 columns as described below.

Column	Item	type	description
1	Animal ID	integer	Same as in pedigree (4641 animals)
2	Sire ID	integer	Same as in pedigree
3	Dam ID	integer	Same as in pedigree
4	Weight	real	Not used here
5	Mu	integer	All 1: not used here
6	Farm	integer	Class effect (155 levels)
7	Sex	integer	Class effect (2 levels)
8	Year	integer	Class effect (11 levels)

Column	Item	type	description
9	Obs. 1	real	Phenotype for trait 1
10	Obs. 2	real	Phenotype for trait 2
11	Obs. 3	real	Phenotype for trait 3
12	Obs. 4	real	Phenotype for trait 4

5.1.3 Single-trait analysis

Model

We here consider the following animal model:

$$y_{ijkl} = F_i + S_j + Y_k + u_l + e_{ijkl}$$

where y_{ijkl} is an observation, F_i is the fixed farm effect, S_j is the fixed sex effect, Y_k is the fixed year effect, u_l is the additive genetic effect and e_{ijkl} is the random residual effect. The purpose of this analysis is to estimate the variance components σ_u^2 for additive genetic effect and σ_e^2 for residual effect.

Parameter file

Let us start with a single-trait analysis with the phenotype 1 (column 9). The model contains farm, sex and year as fixed cross-classified effects and additive genetic effect as a random effect. Place the following parameter file to a folder with the data and pedigree files. You can see this parameter file looks like the same one used by BLUPF90.

Listing 5.1: aireml1.txt

```
DATAFILE
simdata.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
9
WEIGHT(S)
4
EFFECTS:
 6 155 cross
 7 2 cross
 8 11 cross
 1 4641 cross
RANDOM_RESIDUAL VALUES
100
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped.txt
(CO)VARIANCES
100
```

The variance components described above are used as the initial values i.e. $\sigma_e^2 = 100$ and $\sigma_u^2 = 100$.

Intermediate estimates

Running AIREMLF90 with above parameter file, the program reports the intermediate estimates in the current round. We show the messages from the first round.

```
-2 logL = 34141.8991194978 : AIC = 34145.8991194978
 In round 1 convergence = 0.240696612717812
 delta convergence = 40.9948020571449
new R
   50.924
new G
   67.087
```

Several information is available.

- `-2logL` = negative twice of the restricted likelihood with the current variance components (and BLUE and BLUP).
- `AIC` = $-2\log L + 2p$ where p is the number of unique variance components
- `convergence` = the current convergence indicator (see below)
- `delta convergence` = another convergence indicator
- `new R` = updated residual variance (σ_e^2)
- `new G` = updated genetic variance (σ_u^2)

The iterations stop when the convergence indicator is less than ε or when the delta convergence is less than ε. The default is $\varepsilon = 1.0^{-12}$ (`1.0E-12`). The convergence indicator (C) is calculated as

$$C = \frac{\sum_i (\theta_i - \xi_i)^2}{\sum_i \theta_i^2}$$

where θ_i is the current i-th estimate and ξ_i is the previous i-th estimate. This criterion is also used in REMLF90. The delta convergence indicator (C_Δ) is calculated as

$$C_\Delta = \frac{\sum_i |\theta_i - \xi_i|}{n}$$

where n is the number of unique (co)variance components. This indicator is an average abso- lute change in the estimates. When one indicator approaches the criterion, another one usually approaches the criterion as well.

Final estimates

If either convergence indicator meets the criterion, the iteration process stops and the final estimates will be shown on screen. In our data and initial values, we should need 8 rounds to converge.

```
-2 logL = 33423.4510659485 : AIC = 33427.4510659485
 In round 8 convergence = 3.476428650201818E-015
 delta convergence = 1.835758207312910E-006
new R
   62.691
new G
```

```
   38.538
solutions stored in file : "solutions"

Final Estimates
Genetic variance(s) for effect 4
  38.538
Residual variance(s)
  62.691
inverse of AI matrix (Sampling Variance)
  13.471 -6.7551
 -6.7551 6.5499
Correlations from inverse of AI matrix
  1.0000 -0.71914
 -0.71914 1.0000
SE for G
  3.6703
SE for R
  2.5593
```

The program ends with `Final Estimates` showing the variance estimates at convergence. The same information will be saved in a file airemlf90.log. The final estimates are $\hat{\sigma}_u^2 = 38.538$ and $\hat{\sigma}_e^2 = 62.691$ so that the heritability estimates is $\hat{h^2} = 0.3807$.

The program saves the solutions of mixed model equations with new variance estimates in file "solutions". Note that the solutions don't come from the last variance estimates but one round before the convergence. In this example, the solutions are from the variance estimates in round 7. When the algorithm works well and converges correctly, the last variance estimates are almost the same to the previous estimates so the difference between two solutions is negligible. If the convergence trajectory fluctuated, two solutions could be distant. In such a case, the variance estimates themselves are unreliable. In the end of this section, we will consider the convergence problems.

Approximated standard error

A approximated standard error for a variance estimate comes from inverse of AI matrix (Sampling Variance). Suppose a parameter vector is θ and the AI matrix is $\mathscr{I}(\theta)$, the sampling variance of $\hat{\theta}$ is

$$\text{var}(\hat{\theta}) = [\mathscr{I}(\theta)]^{-1}$$

where $[\mathscr{I}(\theta)]^{-1}$ is the inverse of AI matrix. In this example, $\theta = \begin{bmatrix} \sigma_u^2 & \sigma_e^2 \end{bmatrix}'$. The square root of the diagonal of $[\mathscr{I}(\theta)]^{-1}$ are shown as the `SE for G` ($\sqrt{13.471}$) and `SE for R` ($\sqrt{6.5499}$) in the last lines of output.

Unfortunately, the program doesn't calculate the standard error of heritability. You can still calculate the approximated standard error of a function of variance components (e.g. heritability) with the Delta method by hand. See a document on the official wiki (`http://nce.ads.uga.edu/html/projects/AI_SE_revised.pdf`) for details. Here we just show the calculation of the standard error for heritability estimates. The heritability estimates is 0.3807. Using the approximation, the standard error is

$$\begin{aligned}
\text{SE}\left(h^2\right) &= \left(\frac{h^2}{\sigma_u^2}\right)\left[(1-h^2)^2\text{var}(\hat{\sigma}_u^2) - 2(1-h^2)h^2\text{cov}(\hat{\sigma}_u^2,\hat{\sigma}_e^2) + (h^2)^2\text{var}(\hat{\sigma}_e^2)\right] \\
&= \left(\frac{0.3807}{38.54}\right)\left[(1-0.3807)^2 \times 13.471 - 2(1-0.3807)\times 0.3807 \times (-6.755) + (0.3807)^2 \times 6.550\right] \\
&= 0.007
\end{aligned}$$

Another way to evaluate the S.E. for a heritability estimate is to use the method by Meyer and Houle (2013). You can find a do-it-yourself description of the method at (`http://artadia.blogspot.fr/2016/05/standard-error-of-variance-components.html`). AIREMLF90 can conduct this method with an option. You can add the following line to the bottom of the parameter file and run AIREMLF90.

```
OPTION se_covar_function h2 G_4_4_1_1 /( G_4_4_1_1 + R_1_1 )
```

Briefly, this option has 2 arguments: label and formula. The label can contain arbitrary characters. The formula describe the object function for which the standard error should be calculated ($\sigma_u^2/(\sigma_u^2+\sigma_e^2$) in this case). The formula is described using special labels for variance components. For an genetic covariance component, use `G_effect1_effect2_trait1_trait2`; and for a residual covariance, use `R_trait1_trait2`. In this single-trait case, the genetic variance σ_u^2 is defined as the effect 4 in the parameter file so we refer σ_u^2 as `G_4_4_1_1`. Similarly, we refer σ_e^2 as `R_1_1`. See the manual for details on this option.

After the main iterations, the program calculates an approximated standard error using a the Monte-Carlo technique of Meyer and Houle. In this case, you can see the following output on screen.

Sampling variances of covariances function of random effects (n =5000)

```
h2 - Function : g_4_4_1_1 /( g_4_4_1_1 + r_1_1 )
  Mean : 0.38070
  Sample Mean : 0.38014
  Sample SD : 0.30140E-01
 elapsed time 1.098833
```

Sample mean is almost identical to the estimates. The sample SD is seen as an approximated standard error, which is 0.03 (`0.30140E-01` is 0.30140×10^{-1} i.e. 0.03014) in this case.

Don't worry if you forget to add the option to the parameter file. You can prepare a parameter file with the option se covar function and the variance components at convergence. Running AIREMLF90 again, and the program will stop within 1 or 2 rounds and it will calculate the approximated standard error of the parameter.

5.1.4 Convergence problems

The AI algorithm doesn't guarantee the convergence. Sometimes you couldn't meet the convergence in 2 ways:

- Divergence. The estimates suddenly jump to nonsense values and the program stops.
- Slow. The estimates hardly change and the program takes many rounds (several hundred or more).

The problems comes from several reasons.

1. Mistakes in the parameter file, data and pedigree.
2. Wrong initial values.
3. Too complicated model with a limited amount of data and pedigree.
4. Too many variance components to be estimated.

If your files are correct and initial values don't solve the issues, change your model to be simplified.

AIREMLF90 actually tries to correct nonsense estimates using a technique described by Jensen at al. (1996). The method blends the AI matrix from the regular algorithm ($\mathscr{I}_{\text{AI}}$) with the AI matrix from the EM algorithm ($\mathscr{I}_{\text{EM}}$) i.e. $\mathscr{I}(\theta) = (1-w)\mathscr{I}_{\text{AI}} + w\mathscr{I}_{\text{EM}}$ with a weight w. If $w = 1$, the algorithm is

equivalent to the regular EM algorithm, which always provides estimates within their parameter space. If an estimate is not positive definite (i.e. nonsense value), AIREMLF90 sets a small w, blends 2 AI matrices and recalculates the estimates with the modified AI matrix. The program repeats this blending increasing w until the estimates make sense. AIREMLF90 gives up this process after 5 trials (in the end, w is 1 or similar value) and prints a message.

```
*** Warning *** corrected Covariance Matrix 5 times
```

In such a case, the estimates will be nonsense.

The blending is a band-aid method and only successful when the AI matrix is accidentally singular. If the blending happens every round, you can still obtain estimates but you need much more rounds to converge.

5.1.5 EM-REML algorithm

The EM (Expectation-Maximization) REML algorithm is much more stable than the AI algorithm, and very robust to poor initial estimates. Also it can provide good starting point for the AI algorithm. However, it is *much* slower, that is, it needs more rounds to converge. For some very complex problems only the EM REML may converge.

You may run the EM algorithm in AIREMLF90 by using the option

```
OPTION EM-REML 10
```

where 10 (or any integer that the user wants) is the number of iterations of EM REML previous to switching to the AI REML. If you want a "pure" EM REML, then set the number to a very high value like 10000.

5.1.6 Summary

- AIREMLF90 calculates REML estimates of variance components.
- AIREMLF90 accepts the same parameter file as BLUPF90 and set the variance components as initial values.
- The final estimates are shown on screen and saved in the file `airemlf90.log`.
- The solutions are also calculated using covariance estimates in one round before the convergence.
- There are 2 ways to approximate the standard error of a genetic parameter.
- AIREMLF90 doesn't guarantee the convergence even if it tries to make the estimates be in the parameter space.
- If diverged, use a simpler model.
- For complex problems, you may use the EM algorithm within the AIREMLF90 program

5.2 Gibbs sampling and post-Gibbs analysis

5.2.1 Gibbs sampling software

BLUPF90 family has many software to conduct Gibbs sampling for linear mixed models. We recommend a user to use GIBBS2F90 or THRGIBBS1F90 for a linear mixed model. The THRGIBBS1F90 program is capable both for a threshold model and a linear model and it is faster than GIBBS2F90 for a linear model especially using a large data set. GIBBS3F90 may be useful when you assume heterogeneous

residual variances by classes. We will see the difference among Gibbs sampling programs. Here we will use THRGIBBS1F90.

The concept of Gibbs sampling is totally different from REML. We don't deal with the theoretical background of this technique here; see Sorensen and Gianola (2002) or Misztal (2008) for details in this point. Roughly speaking, the Gibbs sampling is a process to draw the posterior distribution out of the samples generated as random numbers based on information available at specific point. In one round, the Gibbs sampler solve the mixed model equations with the current variance components and add a small random number (noise) to each solution. Then the sampler generates a covariance estimate as a random number based on the solutions. The samples of solutions and variance components in each round are not informative but with many samples from many rounds repeated, we can figure the posterior distribution of a variance component as a histogram out of the samples. You can calculate the average of samples (posterior mean) as a point estimator of a variance component. You can also calculate the standard deviation of samples (posterior SD), which is corresponding to standard error in a frequentist approach (e.g. REML).

THRGIBBS1F90 assumes by default noninformative (vague or flat) priors for location parameters (i.e. "fixed" effects) as well as variance components. You add prior information as the degree of belief for each variance component using an option in the parameter file.

The process of Gibbs sampling analysis needs post Gibbs analyses. This can be done with a separate software POSTGIBBSF90 and may be some post-processing by the user. So the basic pipeline of Gibbs sampling analysis is that

1. use THRGIBBSF90 to generate samples for variance components and
2. use POSTGIBBSF90 to summarize the results from the sampling.

Both programs accept the same parameter file as BLUPF90 programs but ask you some additional questions to control the Gibbs sampler.

5.2.2 Preparation

In this section, we will use simulated files which have been already introduced in the previous section for REML. See the section for details.

- `simdata.txt` : data file
- `simped.txt` : pedigree file

The parameter file is also same shown in the previous section. Here we put different name on this file.

Listing 5.2: gibbs1.txt

```
DATAFILE
simdata.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
9
WEIGHT(S)
4
EFFECTS:
 6 155 cross
 7 2 cross
```

```
 8 11 cross
 1 4641 cross
RANDOM_RESIDUAL VALUES
100
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped.txt
(CO)VARIANCES
100
```

The variance components in the parameter file will be used as initial values but the effect of the values on the results will be small. So you can put any values on the parameter file.

5.2.3 Sampling strategy

Invoking THRGIBBSF90, the program shows the current time and asks you the name of parameter file.

```
* Start thrgibbs1f90 ........04-03-2016 12 h 12 m 36 s 173
name of parameter file?
```

After typing the name, then the program will ask you another question.

```
number of samples and length of burn-in
```

You should input 2 integer numbers here.

- The total number of samples to be generated.
- The number of burn-in out of the total samples.

The 2 numbers should be separated with a space or new line (i.e. Enter key). The number of burn-in should be smaller than the total number of samples. For example, if you generate 5,000 samples and discard the first 1,000 samples as burn-in, you can key-in as follows (no commas are allowed).

```
5000 1000
```

You will be asked one more question.

```
Give n to store every n-th sample ? (1 means store all samples )
```

You should one integer number here.

- The size of interval to save the samples.

For example, if you want to save the sample every 10 rounds, type 10 here.

```
10
```

If you want to save all the samples, type 1 instead. After this, the program starts sampling.

What is the appropriate number of sampling? What is the best interval? The answer totally depends on the model and data. We will consider this question later in this section.

5.2.4 Results shown on screen

Intermediate results

During the sampling, the program always prints the sampled variances in each round.

```
         5000 rounds
G
     36.27
R
     62.00
```

Final posterior mean and SD

The program actually calculates the sampled solutions in each round as well. In the end of sampling, the program shows several statistics on screen. You don't always need all the information. We break it down here.

```
elapsed time per iteration 6.7236428 E -03 : total 33.61821
* End of iteration04 -03 -2016 12 h 27 m 05 s 101
ave G
  39.03
SD G
  3.579
ave R
  62.58
SD R
  2.506
```

- `ave G and SD G`: Posterior mean and SD for σ_u^2. Average and standard deviation of saved samples after removing burn-in.
- `ave R and SD R`: Posterior mean and SD for σ_e^2. Average and standard deviation of saved samples after removing burn-in.

Final status of sampling

These are actually not essential for a regular analysis.

```
* Last seeds = 2101474043 1031647242
* Number of samples kept = 4000
solutions stored in binary file : " last_solutions "
```

- `Last seeds`: seeds of random number generator. Usually, the values are not crucial.
- `Number of samples kept`: The total number of samples after burn-in.
- `solutions stored in binary file : "last solutions"`: The last sampled solutions are stored in `last_solutions`. Usually, the file is not essential.

DIC and related information

DIC (Deviance Information Criterion) means the deviance information criterion and it can be used to compare 2 or more models like AIC (smaller is better). In each round, the program calculate partial information to form the DIC. In general notation, we can write a likelihood function as $\log L(\theta)$ with parameters θ. In above mode, we can assume that N is the number of observations and $\hat{\sigma}^2_{e[k]}$ (i.e. $\hat{\theta}_k$) is a sampled residual variance in round k after burn-in, a deviance D_k in round k is defined as

$$D_k = -2\log L(\hat{\theta}_k) = -2\sum_{i=1}^{N}\left[C + \frac{\hat{e}_i^2}{\hat{\sigma}^2_{e[k]}}\right]$$

where C is $\log(2\pi\hat{\sigma}^2_{e[k]})$ and $\hat{e}_i$ s a observed residual for observation i (i.e. $y - \hat{y}$: observation minus sampled solutions). After sampling m rounds after burn-in, $\bar{D}$, an estimate of the expectation of D, is defined as

$$\bar{D} = \frac{\sum_{k=1}^{m} D_k}{m}$$

Similar statistics are also calculated with posterior mean of σ^2_e. We calculate the posterior deviance $D(\bar{\theta})$ using above formula for D_k except $\hat{\sigma}^2_{e[k]}$ (i.e. $\hat{\theta}_k$) is replaced with its posterior mean $\hat{\sigma}^2_e$ i.e. $\hat{\theta}$ as follows

$$D(\bar{\theta}) = -2\log L(\hat{\theta}) = -2\sum_{i=1}^{N}\left[C + \frac{\hat{e}_i^2}{\hat{\sigma}^2_e}\right]$$

These statistics are calculated using the samples stored after burn-in. THRGIBBSF90 prints the following quantity as DIC.

$$\text{DIC} = 2\bar{D} - D(\bar{\theta})$$

THRGIBBS1F90 saves every D_k to a file `fort.99`.

The output is corresponding to above formulas.

```
****************************************
**************** DIC ******************
****************************************
detR 62.5820466970185
# stored samples 400
D-bar 32364.1379063698
D(theta-bar) 30968.7777134920
DIC = 2*D-bar - D (theta-bar) = 33759.4980992475
Effective number of parameters = 1395.36019287775
solutions stored in file : "binary_final_solutions"
* End program04-03-2016 12 h 27 m 06 s 001
```

- `detR`: Determinant of posterior mean of residual covariance matrix ($|\mathbf{R}_0|$ in general case and $|\sigma^2_e|$ in this case).
- `# stored samples`: The actual number of samples saved in a file (m above).
- `D-bar`: $\bar{D}$

- `D(theta-bar)`: $D(\bar{\theta})$
- `DIC = 2*D-bar - D(theta-bar)`: $2\bar{D} - D(\bar{\theta})$
- `Effective number of parameters`: $\bar{D} - D(\bar{\theta})$
- `solutions stored in file: "binary_final_solutions"`: posterior means for sampled solutions after burn-in.

Just picking the numbers up, the estimated genetic variance is $\hat{\sigma}_u^2 = 39.03(\pm 3.579)$ and the estimated residual variance is $\hat{\sigma}_e^2 = 62.58(\pm 2.506)$. If you believe the numbers of samples and burn-in are appropriate, you can use the above posterior mean and SD as the final results. Otherwise, and in the almost cases, you should run POSTGIBBSF90 to perform the post-Gibbs analyses to confirm the numbers of samples and burn-in were enough and to calculate more precise posterior statistics for variance parameters.

5.2.5 Generated files

THRGIBBSF90 generates several files including the following.

- `gibbs samples`: sampled variance components in specific rounds after burn-in.
- `fort.99`: devience in specific rounds after burn-in.
- `last_solutions`: the last sampled solutions saved in binary form.
- `binary_final_solutions`: posterior means for sampled solutions from specific rounds after burn-in saved in binary form.

The samples and calculated statistics are based on samples in every n rounds after burn-in, where n is an interval which you type in the beginning of the program.

The file `gibbs_samples` will be used by THRGIBBSF90 to run post Gibbs analyses. The number of samples stored in this file depends on your initial answers. For example, using 5,000 samples with 1,000 burn-in and 10 sample intervals, the file contains 400 samples i.e. (5000 - 1000)/10. The second file `fort.99` is optional and will be used in the post Gibbs analyses as well. The `last_solutions` file is useful to continue the analysis from the last round but you don't usually have to see the inside of this file. The last file binary final solutions contains a kinds of solutions for "fixed" and "random" effects from the Bayesian analysis. You may want to see the solutions but it is not a text file. In the next section, we will consider the advanced techniques including the extraction of solutions from this file.

5.2.6 Post Gibbs analyses

Invoking POSTGIBBSF90

POSTGIBBSF90 performs post-Gibbs analyses reading a parameter file, `gibbs_samples`, and `fort.99`. Invoking the program, it prints the following message.

```
name of parameter file?
```

Type in the name of parameter file used in this analysis. Then the program checks if gibbs samples was really generated from the parameter file. If success, you will see the information about the total number of saved samples.

```
POST-GIBBSF90 3.06
 # parameters in gibbs_samples = 2
Read 400 samples from round 1010 to 5000
```

Here POSTGIBBSF90 asks you another question.

```
Burn-in ?
```

You should *not* input the same number what you typed in the beginning of THRSIBBS1F90. The burn-in samples were already discarded. This question asks you how many you are going to discard *additional samples* as burn-in. So the answer is

- `0` if you don't have to discard any more samples as burn-in, or
- any integer numbers if you need to discard additional samples as burn-in.

After this, you will be asked the last question.

```
Give n to read every n-th sample ? (1 means read all samples)
```

This question is tricky because you may *not* always input 1!. You can input the same number what you typed in the beginning of THRSIBBS1F90. Or, you can input a multiple of the original number. For example, in this case, you can input 10, 20, 30 and so on. If you type an inappropriate number, the program will stop with a suspicious message.

5.2.7 Post statistics

We will see the following table with many statistics when we input `0` as burn-in and `10` as steps.

```
# samples after burn-in = 400
# samples after burn-in used = 400

                      ******** Monte Carlo Error by Time Series ********
 Pos. eff1 eff2 trt1 trt2 MCE Mean HPD Effective Median Mode Independent
                                                Interval (95%) sample size chain size
   1 4 4 1 1 0.36091 39.035 32.030 45.270 98.1 39.060 39.175 6
   2 0 0 1 1 0.19923 62.582 57.640 67.440 157.8 62.640 63.362 4

                      ******** Posterior Standard Deviation ********
 Pos. eff1 eff2 trt1 trt2 PSD Mean PSD Geweke Autocorrelations Independent
                                                Interval (95%) diagnostic lag: 1 10 50 # batches
   1 4 4 1 1 3.5788 39.035 32.020 46.049 -0.09 0.660 -0.076 0.017 66
   2 0 0 1 1 2.5062 62.582 57.670 67.494 0.09 0.406 -0.023 0.070 100
```

The statistics are calculated for each variance parameter. This output has 2 main tables and each table contains statistics for each variance components. Variance components can be identified with the columns `Pos. eff1 eff2 trt1 trt2`.

- `Pos.` is the position index of a parameter.
- `eff1` and `eff2` are the effects in which the variance defined. Without maternal genetic effect or random regressions, `eff1` equals `eff2`.
- `trt1` and `trt2` are corresponding traits. In a single-trait model, `trt1` equals `trt2`.

In this case, we have 2 variances and one (σ^2_u) is the genetic variance defined as the 4-th effect and another one is the residual variance (σ^2_e). This is from a single-trait model so each label should be 1. Finally we find the position 1 is for σ^2_u and the position 2 is for σ^2_e.

As you see, this program outputs a plenty of information so here we just pick essential parts only. The rest of information will be covered with the next section.

- `Mean`: Posterior mean.
- `HPD Interval (95%)`: High probable interval with 95%.
- `Median`: Posterior median.
- `Mode`: Posterior mode. It would provide a better estimate than posterior mean if the posterior distribution is highly skewed.
- `Independent chain size`: Actual number of samples after adjusting autocorrelations among samples.
- `PSD`: posterior standard deviation.
- `Autocolletations`: lag-correlation between 2 samples with specific interval.

This table shows several point estimates including posterior mean, median and mode. Suitable estimate depends on its skewness of posterior distribution. If the distribution is highly skewed and its mean is apart from the mode, the posterior mode or median could be a better choice. Note that the mode is just approximated from 50 interval classes.

High probable interval (HPD95) is often cited statistics to evaluate accuracy the parameters. This is a kind of interval estimates of a parameter. The values are the lower and upper bounds by cutting off the sorted samples at the top 2.5% largest and the bottom 2.5% values.

You can evaluate a sufficient interval for samples to be saved using Independent chain size and Autocorrelations. Two adjacent samples usually are usually highly correlated because the next sample is drawn based on the current one. When the correlation is still high between distant samples, the dependency-level is also high and the absolute values of the samples should be very similar. It simply means the samples don't provide much information about the posterior distribution. The autocorrelation indicates how much interval we need to achieve the zero-correlation. The independent chain size corresponds to the number of independent samples that can be seen as independent. If the independent chain size is 3, the statistics (i.e. posterior mean and sd) are equivalently calculated using only 3 independent samples (obviously this is too small). So this indicates whether you need more samples or not.

Time-series plot and histogram

After showing above table, the program wait for your key-in with the following message.

```
Choose a graph for samples (= 1) or histogram (= 2); or exit (= 0)
```

POSTGIBBSF90 can draw a time-series plot or histogram of samples. It calls external software Gnuplot to show the graph. So you additionally install Gnuplot. You can type `1` to draw time-series plot or `2` to draw the histogram for samples. Type `0` to quite the program.

The time series plot visually indicates whether the length of burn-in or total sampling is enough or not. If the initial samples looks keeping initial values, you would need more burn-in. If the samples stayed longer in a specific value range the posterior mean doesn't represent the entire samples, you should draw more samples. The histogram provides a visual image of the posterior distribution. Usually, it is a one-peak distribution but skewed (with long right tail) more or less. If your distribution has 2 peaks or very flat, you should check the parameter file or draw more samples.

5.2.8 Output files from POSTGIBBSF90

POSTGIBBSF90 makes 7 files which are derived from the post Gibbs statistics described above.

- `postout`: The same content which is shown on screen.
- `postind`: relation of a position index to a variance component.
- `postmean`: Posterior means shown in a user-friendly format.
- `postmeanCorr`: Posterior means for correlations shown in a user-friendly format.

- `postsd`: Posterior standard deviations.
- `fort.998`: log L calculated from the file `fort.99`
- `postgibbs_samples`: sample values.

The `postout` file contains precise statistics from the post Gibbs analysis. You can check the position index with `postind`. A series of files contains posterior means and SDs for variance components. The file `fort.998` contains $\log L$ needed for the computation of Bayes factor.

The file `postgibbs_samples` is useful especially if you calculate other statistics by yourself. It contains samples after burn-in. The content looks like the following.

```
1 1010 2 35.60 64.26
2 1020 2 36.98 63.53
3 1030 2 30.92 66.49
4 1040 2 34.93 65.60
5 1050 2 37.38 63.26
```

The meaning of each column is:

- 1st column: the sequential number for saved samples.
- 2nd column: the actual round in which each sample was drawn.
- 3rd column: the number of parameters.
- 4th or later columns: actual samples ordered by the position index.

5.2.9 Personal Post-Gibbs Analysis

From the file `postgibbs_samples`, you can easily draw a graph and calculate additional statistics. For instance, you may compute, at each sample, the *sample* of the heritability as a function of the samples of the variance components. The code in R would be something like

```
a=read.table("postgibbs_samples",header=FALSE)
# tell what each thing is
colnames(a)=c("i","iter","varu","vare")
a$h2=a$varu/(a$varu+a$vare)
hist(a$h2)
```

in this way, you describe the posterior distribution of h^2 from the samples of variance components. This procedure can be done for *any* function of the variance components, for instance genetic correlations.

More sophisticated analysis can be done with R package 'boa'.

5.2.10 Remarks

You must not run 2 jobs (Gibbs sampling programs) in the same directory. One program will overwrite `gibbs_samples` generated by the other program. If you want to share the data and pedigree files in multiple jobs, make a separate directory (or folder) and create symbolic links to the files.

There are several options for THRGIBBSF90. For example, one is for continuing sampling with already finished results. Another one can put the degree of brief for prior information. Some useful options will be explained at the next section.

There are many Gibbs sampling programs in BLUPF90 family. The difference is in their implementation.

- GIBBSF90: initial program; no longer distributed
- GIBBS1F90: much faster in multiple-trait models
- GIBBS2F90: faster in convergence for models with correlated effects (e.g. maternal model or random regression model).
- GIBBS3F90: derived from GIBBS2; supporting heterogeneous residual variances; a little bit slower than GIBBS2.
- THRGIBBS1F90: derived from GIBBS2; supporting threshold models; supporting huge dense matrices; faster than GIBBS2 especially for large data set; required more memory than GIBBS2.
- THRGIBBS3F90: derived from THRGIBBS1; supporting heterogeneous residual variances.

Actually recent improvements were done for GIBBS2F90 and THRGIBBS1F90. So we recommend a user to use these 2 software if you are not interested in heterogeneous residual variances.

5.2.11 Summary

- THRGIBBS1F90 or GIBBS2F90 is recommended.
- THRGIBBS1F90 can perform Gibbs sampling both with the regular linear mixed models and the threshold models.
- THRGIBBS1F90 needs the numbers of total samples and burn-ins, and the interval for samples to be saved.
- THRGIBBS1F90 saves the samples to the file gibbs samples.
- POSTGIBBSF90 reads gibbs samples and calculates post Gibbs statistics.
- POSTGIBBSF90 creates many files containing the statistics.
- You can check whether the initial settings are suitable or not from the output.
- You can conduct your own post analyses with the file postgibbs samples generated by POSTGIBBSF90.

5.3 Advanced usage of AIREMLF90

5.3.1 Heterogeneous residual variances

There is a an situation where the residual variances vary over conditions. This is known as heterogeneous residual variances. There are 2 types of modeling for the heterogeneous residual variances.

1. The residual variance is related to a covariate. For example, residual variance for body weight increase by age of month. The residual variance can be described as a function of a covariate (age).
2. The residual variance differs by class. For example, in a test-day random-regression model, we usually assume different residual variance in each lactation stage. The residual variances are independent each other.

Although AIREMLF90 is designed to handle the first case, the program can also handle the second case with a trick. We will see the both cases with a numerical example

Model

In this example, we asuume the following model:

$$y_{ijkl} = F_i + S_j + Y_k + u_l + e_{ijkl}$$

where y_{ijkl} is the observation, F_i , S_j and Y_k are the fixed effects, u_l is the additive genetic effect and e_{ijkl} is the residual effect. The residual variance is defined as the following function

$$\sigma_e^2 = \exp(b_0 + b_1 x_{ijkl})$$

where b_0 and b_1 are the regression coefficients to account for the residual variance and x_{ijkl} is the covariate measured with the observation y_{ijkl}. The modeling is based on Foulley and Quaas (1995). AIREMLF90 will estimate b_0 and b_1 with user-supplied x_{ijkl}.

Data

In this section, we will use simulated files similar as the previous section. You can download the files from Github.

- `simdata2.txt` : data file
- `simped2.txt` : pedigree file

The pedigree file contains 3 columns: animal, sire and dam. The data file has 12 columns as described below.

Column	Item	type	description
1	Animal ID	integer	Same as in pedigree (4641 animals)
2	Sire ID	integer	Same as in pedigree
3	Dam ID	integer	Same as in pedigree
4	Weight	real	Not used here
5	Mu	integer	All 1: not used here
6	Farm	integer	Class effect (155 levels)
7	Sex	integer	Class effect (2 levels)
8	Year	integer	Class effect (11 levels)
9	Obs. 1	real	Phenotype for trait 1
10	Obs. 2	real	Phenotype for trait 2
11	Obs. 3	real	Phenotype for trait 3
12	Obs. 4	real	Phenotype for trait 4
13	Covariate	real	Related to residual variance
14	Class	integer	(Used in the next subsection)

The column 13 contains a real value which is larger when the residual variance for the observa- tion is larger. The 14th column contains the heterogeneous-residual-variance class (3 levels) but it will not be used in this example (see the next subsection).

Parameter file

The following parameter file is used.

Listing 5.3: aireml2.txt

```
DATAFILE
simdata2.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
9
WEIGHT(S)
4
```

```
EFFECTS:
 6 155 cross
 7 2 cross
 8 11 cross
 1 4641 cross
RANDOM_RESIDUAL VALUES
100
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped2.txt
(CO)VARIANCES
100
OPTION hetres_pos 13
OPTION hetres_pol 4.5 0.5
```

The options define the model for heterogeneous residual variances.

- `OPTION hetres_pos` = the position of covariate x in the data file. Intercept b_0 is implicitly considered in the program.
- `OPTION hetres_pol` = the starting values for b_0 and b_1.
  - You should supply the values for all coefficients including intercept.
  - In this example, the initial value will be $\exp(4.5+0.5) = 148.4$ for $x = 1$.

In above case, we define only 1 regression coefficient at hetres pos but the program adds the intercept. If you put 2 numbers at hetres pos and also put 2 initial values at hetres pol, the program doesn't add the intercept. In such case, the model doesn't contain any intercept but does contain user-supplied covariates only.

Results

You can run AIREMLF90 with above parameter file. The following output will be shown.

```
Final Estimates
 Genetic variance(s) for effect 4
   39.384
 new R
          1 -th trait: 1 -th coefficient = 4.25520168650538
          1 -th trait: 2 -th coefficient = 5.886409914144559E-002
 inverse of AI matrix (Sampling Variance)
   16.773 -0.12514 0.77591E-02
-0.12514 0.41370E-02 -0.16778E-02
 0.77591E-02 -0.16778E-02 0.10722E-02
 Correlations from inverse of AI matrix
   1.0000 -0.47507 0.57858E-01
-0.47507 1.0000 -0.79664
 0.57858E-01 -0.79664 1.0000
 SE for G
   4.0955
 SE for R
  0.64319E-01
```

The output shows $\hat{b}_0 = 4.2552$ and $\hat{b}_1 = 0.058864$. This corresponds to $\hat{\sigma}_e^2 = \exp(4.26 + 0.0589 \times 0.5) = 72.9$ for $x = 0.5$, $\hat{\sigma}_e^2 = \exp(4.26 + 0.0589 \times 1.5) = 77.4$ for $x = 1.5$ and $\sigma_e^2 = \exp(4.26 + 0.0589 \times 2.5) = 82.0$ for $x = 2.5$.

One drawback of this analysis is that the calculations of standard error for the residual variance are not easy. The AI matrix contains information for $\hat{b}_0$ and $\hat{b}_1$ (the last 2 rows/columns).

Another modeling by class

The heterogeneous residual variances are accounted for by class. We assume 3 levels in the class. The 14th column in above data contains the level for each observation. AIREMLF90 doesn't directly accept the level number so we should convert it to different format.

Having 3 levels is equivalent to defining 3 regression coefficients without intercept. We now assume the following parameterization without intercept.

$$\sigma_e^2 = \exp(b_1 x_1 + b_2 x_2 + b_3 x_3)$$

If the observation has the residual variance level 1, $x_1 = 1$, $x_2 = 0$, and $x_3 = 0$. For the level 2, $x_1 = 0$, $x_2 = 1$, and $x_3 = 0$ and for the level 3, $x_1 = 0$, $x_2 = 0$, and $x_3 = 1$. We need 3 covariate with 1 or 0. We add 3 extra columns to the rightmost of the data file.

- tutorials:`simdata2a.txt`: extended data file

The first 10 rows are shown below.

Listing 5.4: simdata2a.txt

```
 1 0 0 1.00 1 67 1 1 85.0 0.0 92.1 91.0 0.46 1 1 0 0
 2 0 0 0.98 1 144 1 1 115.2 103.9 94.8 90.6 1.52 2 0 1 0
 3 0 0 1.04 1 92 1 1 93.0 107.4 114.9 107.0 0.00 1 1 0 0
 4 0 0 0.97 1 26 1 1 84.0 91.8 93.7 107.1 0.61 1 1 0 0
 5 0 0 0.96 1 83 1 1 100.9 87.9 88.9 91.5 0.55 1 1 0 0
 6 0 0 0.99 1 62 1 1 69.8 71.5 69.3 0.0 0.88 1 1 0 0
 7 0 0 0.97 1 40 1 1 101.2 0.0 92.1 88.6 0.27 1 1 0 0
 8 0 0 1.07 1 82 1 1 77.4 72.7 84.1 105.2 1.69 2 0 1 0
 9 0 0 1.01 1 71 1 1 107.8 87.0 97.8 76.0 2.89 3 0 0 1
10 0 0 0.95 1 114 1 1 94.4 100.5 103.8 89.8 1.26 2 0 1 0
```

The column 15, 16 and 17 are contains 1 or 0. In the column 15, it is 1 if the level is 1, otherwise 0. In the column 16, it is 1 if the level is 2, otherwise 0. In the column 17, it is 1 if the level is 3, otherwise 0. With this data file, the parameter file is as follows.

Listing 5.5: aireml2a.txt

```
DATAFILE
simdata2a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
9
WEIGHT(S)
```

```
4
EFFECTS:
 6 155 cross
 7 2 cross
 8 11 cross
 1 4641 cross
RANDOM_RESIDUAL VALUES
100
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped2.txt
(CO)VARIANCES
100
OPTION hetres_pos 15 16 17
OPTION hetres_pol 4.5 4.5 4.5
```

The results from above parameter file are shown here.

```
Final Estimates
 Genetic variance(s) for effect 4
   39.296
 new R
          1 -th trait: 1 -th coefficient = 4.28231169726656
          1 -th trait: 2 -th coefficient = 4.36065608258819
          1 -th trait: 3 -th coefficient = 4.38963108448880
 inverse of AI matrix (Sampling Variance)
   16.735 -0.11982 -0.11288 -0.10685
 -0.11982 0.32743E-02 0.75391E-03 0.69459E-03
 -0.11288 0.75391E-03 0.31352E-02 0.67693E-03
 -0.10685 0.69459E-03 0.67693E-03 0.29536E-02
 Correlations from inverse of AI matrix
   1.0000 -0.51184 -0.49281 -0.48059
 -0.51184 1.0000 0.23530 0.22335
 -0.49281 0.23530 1.0000 0.22245
 -0.48059 0.22335 0.22245 1.0000
 SE for G
   4.0909
 SE for R
  0.57221E-01
```

The estimate for the level 1 is $\hat{\sigma}_e^2 = \exp(4.28) = 72.2$, for the level 2, $\hat{\sigma}_e^2 = \exp(4.36) = 78.3$ and for the level 3, $\hat{\sigma}_e^2 = \exp(4.39) = 80.6$. These values are very similar to the previous results from a regression parameterization i.e. $\exp(b_0 + b_1 x)$ for $x = 0.5$, 1.5, and 2.5.

Remarks

The methods explained here will not work well for multiple-trait models or complicated models with many variance components.

5.3.2 Likelihood Ratio Test

REML methods allow statistical testing of variance components. For instance, imagine that we have to choose one of two competing models:

- model 1 *not* including maternal permanent environmental effect
- model 2 *including* maternal permanent effects.

The Likelihood Ratio Test (LRT) checks if the extra random effect, with associated variance component gives a better fit of the model, against not fitting it. The model *without* the extra random effect is the *null* (H_0) model versus the model *with* the extra random effect, which is the alternative model (H_1). Then the difference between the two, which is a positive number, is a LRT statistic which follows a mixture of χ^2 distributions. The theory of the LRT can be found in standard books and a nice description is in Sorensen & Gianola book.

Output of AIREMLF90 under H_0 is $x = -2logL$; output under H_1 is $y = -2logL$. These are positive numbers, so the smaller the better; always $y < x$ as H_1 is a more complex (so more likely) model. The trick resides in knowing if $x - y$ is "big enough". The LRT statistic is $LRT = x - y$, which is a statistic distributed as χ^2. For one variance component, the p-value is (in R) :

```
pchisq(x-y,1,lower.tail=FALSE)/2
```

Classical applications of LRT in quantitative genetics include testing $h^2 > 0$ or testing of QTL effects. An application for association analysis for a multi allelic gene is (`http://dx.doi.org/10.3168/jds.2013-6570`).

Example for test of heritability in dairy sheep

I tested if genetic effects ($\mathbf{u}$) should be fit in an old data set from dairy sheep (80,000 records, 50,000 animals in pedigree) with a model including permanent (σ^2_p), genetic (σ^2_p) and residual (σ^2_e) variances.

Under the complete model H1 ($\mathbf{y} = \mathbf{Xb} + \mathbf{Zu} + \mathbf{Zp} + \mathbf{e}$)) we have in the (last lines of) the output of AIREMLF90 :

```
-2logL = 825227.48
```

Then we run the reduced model H0 with a parameter file that will *not* include the genetic effect ($\mathbf{y} = \mathbf{Xb} + \mathbf{Zp} + \mathbf{e}$) we have:

```
-2logL = 828095.614456413
```

then the test in R is:

```
pchisq(828095.614456413-825227.48,1,lower.tail=FALSE)/2

[1] 0
```

the p-value is so small that we cannot see the difference from 0. We can take the log, then transform to a scale of -log10(pvalue), a scale typically used in GWAS:

```
pval=pchisq(828095.614456413-825227.48,1,lower.tail=FALSE,log.p=TRUE)/2
> pval
[1] -719.137
-log10(exp(pval))
[1] 312.3172
```

So, the p-value is $p < 10^{-312}$, evidence against H_0 is very strong and we should accept that there is genetic variance in this data set.

5.4 Advanced features in Gibbs sampling programs

5.4.1 Heterogeneous residual variances

As mentioned as an advanced technique in AIREMLF90, there is a model with heterogeneous residual variances. GIBBS3F90 supports heterogeneous residual variances defined by class. Here we will demonstrate an analysis with the heterogeneity in residual variance.

Files

We use the same data and pedigree file as before.

- `simdata2.txt` : data file
- `simped2.txt` : pedigree file

The pedigree file contains 3 columns: animal, sire and dam. The data file has 12 columns as described below.

Column	Item	type	description
1	Animal ID	integer	Same as in pedigree (4641 animals)
2	Sire ID	integer	Same as in pedigree
3	Dam ID	integer	Same as in pedigree
4	Weight	real	Not used here
5	Mu	integer	All 1: not used here
6	Farm	integer	Class effect (155 levels)
7	Sex	integer	Class effect (2 levels)
8	Year	integer	Class effect (11 levels)
9	Obs. 1	real	Phenotype for trait 1
10	Obs. 2	real	Phenotype for trait 2
11	Obs. 3	real	Phenotype for trait 3
12	Obs. 4	real	Phenotype for trait 4
13	Covariate	real	Not used
14	Class	integer	Heterogeneous residual class

The 14th column contains the heterogeneous-residual-variance class (3 levels) which is used in this section.

The parameter file has 1 option to define the heterogeneous residual class.

Listing 5.6: gibbs2.txt

```
DATAFILE
simdata2.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
9
WEIGHT(S)
4
EFFECTS:
 6 155 cross
 7 2 cross
 8 11 cross
 1 4641 cross
RANDOM_RESIDUAL VALUES
100
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped2.txt
(CO)VARIANCES
100
OPTION hetres_int 14 3
```

The option has 2 arguments.

- `OPTION hetres_int` = defines the heterogeneous residual class with 2 values: 1) the position of the class in the data file and 2) the maximum level in the class.

In this case, we specify that the 14-th column has 3 levels.
With 20,000 samples (saved each 10 samples) with 10,000 burn-in, the following results can be found.

```
ave G
  39.845
SD G
   3.9917
ave R
   72.313
SD R
   3.9497
ave R
  78.248
SD R
  4.4558
ave R
 80.646
SD R
  4.3320
```

There are 3 ave_R (and SD_R) blocks. The first one corresponds to the variance in the heterogeneous level 1, the second line for the level 2 and so on. Compare the above estimates to the results from AIREMLF90 shown in the previous section.

5.4.2 Restart the sampling

After the sampling, you would find more samples are needed. The Gibbs sampling programs support to restart the sampling from the end point of the previous run. Any recent Gibbs sampler supports this feature.

When a Gibbs sampler finishes sampling, 4 files are created. To continue the sampling, all the files are basically required.

- binary_final_solutions = contains posterior means and SDs for location parameters
- last_solutions = the last samples for location parameters
- fort.99 = values needed for DIC
- gibbs_samples = sampled variance components

One additional option is required in the original parameter file. It needs the number of samples drawn in the previous run. If you had 10000 samples, the option should be

```
OPTION cont 10000
```

where 10000 is the number of samples obtained previously. Run the Gibbs sampler with a parameter file with above option, and the program restart the sampling (from 10001 in the example).

5.4.3 Extraction of solutions from binary final solutions

The file binary final solutions contains the posterior mean of a location parameter (i.e. a solution of ''fixed" or ''random" effect). This file is saved as non-text format. The following Fortran program can extract the solutions and print them to a file final solutions.txt. You can compile the program and run it in a directory where the binary final solutions is. The output format is equivalent to BLUPF90 with OPTION sol_se.

Listing 5.7: binsol_to_textsol.f90

```
program binsol_to_textsol
    implicit none
    integer :: io, t, e, l
    double precision :: v, sol, se
    open(10, file='binary_final_solutions', form='unformatted', &
        status='old', iostat=io)
    if(io /= 0) stop
    open(20, file='final_solutions.txt')
    write(20,'(" trait / effect level solution                s.e.")')
    do
        read(10, iostat=io) t,e,l,sol,se
        if(io /= 0) exit
        write(20, '(2i4,i10,2f20.8)') t,e,l,sol,se
    end do
    close(10)
    close(20)
```

```
end program binsol_to_textsol
```

5.4.4 Output of POSTGIBBSF90

The POSTGIBBSF90 program shows many diagnosis for the Gibbs samples. Here we just show the basic ideas for the information.

- `MCE` = Monte Carlo error, corresponding to the ''standard error" of the posterior mean of a parameter ($\hat{\mu} - \mu$).
- `Mean` = Posterior mean of a parameter.
- `HPD` = High probability density within 95%, close idea to ''95% confidence interval" in frequentist approach.
- `Effective sample size` = Number of samples after deducting auto-correlation among samples.
- `Median` = Posterior median of a parameter.
- `Mode` = Posterior mode of a parameter; just an approximation.
- `Independent chain size`
- `PSD` = Posterior standard deviation of a parameter.
- `Mean` = the same as above.
- `PSD Interval (95%)` = Lower and upper bounds of Mean $\pm$ 1.96PSD.
- Geweke diagnostic = Convergence diagnosis; could be converged if this is < 1.0 (according to the official manual, this is almost useless because this is < 1.0 in almost all cases).
- `Autocorrelations` = Lag-correlations with lag 1, 10 and 50; calculated for the saved samples.
- `Independent # batches` = Effective number of blocks after deducting the auto-correlation among samples.

6 Various models

In this chapter, we will see how BLUPF90 is applicable to a variety of models. We will look at various models in a standard textbook written by Mrode (2014). We explain how to write a parameter file to handle a model in his book. We hardly represent the actual data, pedigree, and genotype files. You can easily create the files if you have the book. We do not only solve the equations but do introduce new options and tricks useful for actual data analyses.

6.1 Animal model

6.1.1 Model

Mrode (2014) considered the pre-weaning gain (WWG) as a target trait and applied a linear mixed model to the trait with a fixed effect (sex), a random effect (animal), and the residual effect. Assume that the genetic variance was $\sigma_u^2 = 20$ and the residual variance was $\sigma_e^2 = 40$. This is a typical animal model which have been already introduced in the previous chapter.

There two possible systems of equations: 1) each variance component is explicitly involved in the equations, or 2) the variance ratio α is used. Two systems of equations should result in the same solutions.

6.1.2 Files

We now prepare the data file (`data_mr03a.txt`). It includes 5 observations just from 5 animals.

The data file has 5 columns as follows. This is the exact copy of the original table in the textbook.

1. Animal ID (calves)
2. Sex (1 for male and 2 for female)
3. Sire ID
4. Dam ID
5. Observations (WWG, kg)

The column 3 and 4 are not actually used in this analysis.

Pedigree file is also prepared. The 1st column is animal ID, 2nd column for sire ID and the 3rd column for dam ID.

The parameter file is following. To obtain the exact solutions, we add `OPTION solve_method FSPAK`. With this option, we can calculate the reliability of a solution using the diagonal elements (PEV: prediction error variance) of the inverse of the left hand side of mixed model equations. Additional option `OPTION sol_se` is needed to calculate PEV.

Listing 6.1: param_mr03a.txt

```
DATAFILE
data_mr03a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
```

```
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS:
2 2 cross
1 8 cross
RANDOM_RESIDUAL VALUES
40.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr03a.txt
(CO)VARIANCES
20.0
OPTION solv_method FSPAK
OPTION sol_se
```

6.1.3 Solutions

Invoking BLUPF90 with above parameter file, we immediately see the solution in the file solutions.

Listing 6.2: solutions

```
trait/effect level solution s.e.
   1 1 1 4.35850233 4.88082357
   1 1 2 3.40443010 5.66554023
   1 2 1 0.09844458 4.34094096
   1 2 2 -0.01877010 4.43664612
   1 2 3 -0.04108420 4.27297922
   1 2 4 -0.00866312 4.13608581
   1 2 5 -0.18573210 4.13814812
   1 2 6 0.17687209 4.20610397
   1 2 7 -0.24945855 4.20407502
   1 2 8 0.18261469 4.11029997
```

The solutions are identical to solutions shown in the textbook (pp.39). The above `s.e.` is the square root of a diagonal elements of the inverse of left-hand side. Note that the above `s.e.` is actually SEP (standard error of prediction) in the textbook (pp.45) not PEV. This happened because BLUPF90 created general mixed model equations explicitly containing σ_e^2 and σ_u^2 rather than the variance ratio $\alpha = \sigma_e^2/\sigma_u^2$. So we don't need to multiply extra σ_e^2 by the inverse element to obtain PEV. Below, we will demonstrate the same left hand side shown in the textbook and confirm its inverse equals to results in the textbook.

6.1.4 Alternative parameter file

The textbook uses a simplified form of mixed model equations in a single-trait analysis. BLUPF90 can handle this form with a tricky (not recommended) way. In this form, only the variance ratio matters. The ratio is $\alpha = \sigma_e^2/\sigma_u^2 = 2.0$ and it is equivalent to assuming $\sigma_e^2 = 1.0$ and $\sigma_u^2 = 0.5$.

Listing 6.3: param_mr03a1.txt

```
DATAFILE
data_mr03a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS:
2 2 cross
1 8 cross
RANDOM_RESIDUAL VALUES
1.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr03a.txt
(CO)VARIANCES
0.5
OPTION solv_method FSPAK
OPTION sol se
```

The solutions are following.

Listing 6.4: solutions

```
trait/effect level solution s.e.
   1 1 1 4.35850233 0.77172597
   1 1 2 3.40443010 0.89580057
   1 2 1 0.09844458 0.68636303
   1 2 2 -0.01877010 0.70149535
   1 2 3 -0.04108420 0.67561734
   1 2 4 -0.00866312 0.65397259
   1 2 5 -0.18573210 0.65429867
   1 2 6 0.17687209 0.66504343
   1 2 7 -0.24945855 0.66472263
   1 2 8 0.18261469 0.64989549
```

This parameter file will provide exactly the same solutions as before. An advantage of this method is to possibly reduce the numerical error because of $\sigma_e^2 = 1$, which doesn't contribute to the equations. We square the `s.e.` to obtain a diagonal element of inverse left-hand side. You can calculate the reliability or accuracy of estimated breeding value by hand.

For example, the diagonal element for animal 1 is $d_1 = 0.68636303^2 \approx 0.471$ which is the same to the reference value in the textbook (pp.45). The PEV for animal 1 is $\mathrm{PEV}_1 = d_i\sigma_e^2 = 0.471 \times 40 = 18.84$. The standard error of prediction is $\mathrm{SEP}_1 = \sqrt{\mathrm{PEV}_1} = \sqrt{18.84} = 4.341$ which is also the same to the textbook. The reliability for animal 1 is $r^2 = 1 - \mathrm{SEP}^2/\sigma_u^2 = 1 - \mathrm{PEV}/\sigma_u^2 = 1 - 18.84/20 = 0.058$. You can confirm that this rule is true for the other animals.

6.2 Sire model

6.2.1 Model

Here we apply a sire model to the previous data. The mathematical model is the same as the previous section but the individual genetic effect is replaced with the sire genetic effect. In this model, we consider as if an observation belongs to the sire of the animal rather than to the animal itself. Additive genetic variation is explained by sires. So the sire variance is $\sigma_s^2 = 0.25\sigma_u^2 = 5$ and the residual variance absorbs the remaining part of genetic variance and $\sigma_e^{2*} = 0.75\sigma_u^2 + \sigma_e^2 = 55$. The phenotypic variance is the same (60) between the animal and sire models.

6.2.2 Files

We can use the same data set shown in the previous section (`data_mr03b.txt`). In this case, we use the 3rd column (sire ID) instead of the 1st column (animal ID) as an animal's effect.

The pedigree file in sire model for BLUPF90 is different from one used in animal model. This file (`pedigree_mr03b.txt`) contains 3 columns:

1. The ID for sire.
2. The ID for sire of the sire.
3. The ID for maternal grand sire (MGS) of the sire.

In this case, all MGS are unknown so the 3rd column should be 0.

The parameter file is also altered. We put comments around lines to be changed.

Listing 6.5: param_mr03b.txt

```
DATAFILE
data_mr03b.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS:
2 2 cross
3 4 cross # 3rd column = sire effect; 4 sires
RANDOM_RESIDUAL VALUES # residual variance
55.0
RANDOM_GROUP
2
RANDOM_TYPE # type changed
add_sire
FILE
pedigree_mr03b.txt
(CO)VARIANCES # sire variance
5.0
OPTION solv_method FSPAK
```

The value add sire is a word for sire model in BLUPF90. The pedigree file contains no sire 2 (i.e. it is missing) but the program doesn't care about this.

6.2.3 Solutions

The solutions are following.

Listing 6.6: solutions

```
trait/effect level solution
   1 1 1 4.33567107
   1 1 2 3.38198579
   1 2 1 0.02200220
   1 2 2 0.00000000
   1 2 3 0.01402640
   1 2 4 -0.04304180
```

The solution for sire ID 2 is 0. BLUPF90 always produces the solution 0 for such level missing in data and pedigree files. Otherwise, the solutions are identical to the textbook (pp.48).

6.3 Reduced animal model

6.3.1 Model

Reduced animal model is an animal model except that includes only parents (i.e. animals with progeny). Breeding values for animals without progeny will be indirectly calculated using their parents' breeding values. We omit the description about the model and theory here. See the textbook or Quaas and Pollak (1980) for details.

BLUPF90 doesn't directly support the reduced animal model but still handle it using (very) tricky ways. Some techniques used here will also be used in random regression models and social interaction models. Although there are several ways to perform reduced animal model in BLUPF90, we here introduce an educational example which is redundant but easier to understand.

6.3.2 Files

We can use the same pedigree file (`pedigree_mr03c.txt`) to previous animal model except removing animals without progeny (animal 7 and 8 in this case).

The data file is extended with 5 additional columns. We here show few line of the data file.

Listing 6.7: data_mr03c.txt

```
4 1 1 0 4.5 4 0.0 0.0 1.0 1.0
...
7 1 4 5 3.5 0 0.5 0.5 0.0 0.8
...
```

The 6 the column (the first additional column) is animal ID but has 0 if the animal has no progeny. The column 7, 8 and 9 contain actual values to be added to the incidence matrix W (see the textbook). This idea is similar to a way to support random regressions. The columns 7 and 8 are 0.0 if the animal is a parent, or 0.5 otherwise. The column 9 is 1.0 if the animal is a parent, or 0 otherwise.

The column 10 (the last column) contains a weight to adjust residual variance. For parent animals, the residual variance is $\sigma_e^2 = 40$ and its inverse is $1/\sigma_e^2 = 0.025$. For non-parents, in this case, the residual

variance should be $\sigma_e^{2*} = \sigma_e^2 + 0.5\sigma_u^2 = 40 + 0.5 \times 20 = 50$ and its inverse is $1/\sigma_e^{2*} = 0.020$. If the non-parent animal has missing parent, you should set different weight on σ_u^2 (see the textbook). BLUPF90 doesn't support heterogeneous residual variances. Instead, you can use `WEIGHT(S)` to use the alternative residual variances. The program reads the weight from the data file and multiply it by the inverse of residual variance. In our example, the weight is 1.0 for parent animals and 0.8 for non-parent animals because $0.8 \times (1/\sigma_e^2) = 0.8 \times 0.025 = 0.020$ and it is the correct value for non-parent animals.

One issue is how to add these constants to $\mathbf{W}$. BLUPF90 can usually add only values of 1 to the incidence matrix per record. To overcome this limitation, we introduce a tricky description in the `EFFECT:` block in a parameter file.

Listing 6.8: param_mr03c.txt

```
DATAFILE
data_mr03c.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
5
WEIGHT(S)
10
EFFECTS:
2 2 cross
7 0 cov 3
8 0 cov 4
9 6 cov 6
RANDOM_RESIDUAL VALUES
40.0
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
pedigree_mr03c.txt
(CO)VARIANCES
20.0
OPTION solv_method FSPAK
```

In the `EFFECTS:` block, the last 3 statements send the value in the data file literally to the incidence matrix. You can see these 3 effects are in one block and only the last statement `9 6 cov 6` define the number of levels (i.e. the number of animals). For each statement, the program reads the value from the data file (the position is in the first number of that line) and recognizes it as a (real number) covariate (this behavior is from `cov` keyword) and put it into the location corresponding to an animal ID (the location is defined by the number after `cov`).

Let's see what happens when the program process these 3 statements for each record. As the first example, we consider the first line of the data file.

```
4 1 1 0 4.5 4 0.0 0.0 1.0 1.0
```

- Processing the statement 2 (`7 0 cov 3`): the program reads the 7th column from the data, and recognize it as a real value (0.0), and add it to **W** at the location defined by the 3rd column in the data file (1st column in **W**).
- Processing the statement 3 (`8 0 cov 4`): the program reads the 8th column from the data, and recognize it as a real value (0.0), and add it to **W** at the location defined by the 4th column in the data file (the location 0; it is ignored).
- Processing the statement 4 (`9 6 cov 6`): the program reads the 9th column from the data, and recognize it as a real value (1.0), and add it to **W** at the location defined by the 6th column in the data file (4th column in **W**).

After the process, the first row in **W** should be the following.

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix}$$

The we will see the 4th line of the data file.

```
7 1 4 5 3.5 0 0.5 0.5 0.0 0.8
```

- Processing the statement 2 (`7 0 cov 3`): the program reads the 7th column from the data, and recognize it as a real value (0.5), and add it to **W** at the location defined by the 3rd column in the data file (4th column in **W**).
- Processing the statement 3 (`8 0 cov 4`): the program reads the 8th column from the data, and recognize it as a real value (0.5), and add it to **W** at the location defined by the 4th column in the data file (5th column in **W**).
- Processing the statement 4 (`9 6 cov 6`): the program reads the 9th column from the data, and recognize it as a real value (0.0), and add it to **W** at the location defined by the 6th column in the data file (the location 0; it is ignored).

After the process, the 4th row in **W** should be the following.

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.5 & 0.5 & 0.0 \end{bmatrix}$$

A similar description will be used in a competitive model (social interaction model) which needs an incidence matrix where 2 or more elements take place in one row.

6.3.3 Solutions

You can see the solutions are identical to the textbook (pp.53). You will find another way to perform reduced animal model using the same technique used above. A curious reader can try another way.

6.4 Animal model with unknown parent groups

6.4.1 Short introduction to unknown parent groups

Unknown parent group (as known as genetic group or phantom group) is one of the less-understood concepts in animal breeding. We usually assume the mean breeding value in a base population is 0. The average breeding value by generation or year will increase (or decrease) by selection. If we have complete pedigree back to the base population, the complete phenotype used as the selection criterion and an appropriate model to describe the factors for an observation, a mixed model can provide unbiased prediction of breeding values for all animals. But reality is different. One of the biggest challenge is

incomplete pedigree. Many young animals may have no pedigree information in an population undergoing selection. In this case, it is an incorrect assumption that unknown parents belong to the base population. Under the selection, younger animals have significantly higher (or lower) genetic base so their parents should have a nonzero average of breeding values.

Genetic group can partially solve this problem. A genetic group looks like a real animal. We assign a group to unknown parents that are expected to have the same genetic level. A group to be assigned to parents in younger generation should be different from a group for older parents because of different genetic levels. In addition to generation (or year), sex and region of origin can be factors to define the groups. The base animals should be in a group, every animal is a descendant of genetic groups.

The groups are treated as fixed effects. The expected 0 in base animals will be replaced with BLUE for the group effect. Therefore, the final prediction of breeding value for an animal is a mixture of BLUE for the group effect and BLUP for the breeding value. In spite of this complication, the mixed model equations with groups are surprisingly simple and look like the regular equation. For theoretical background, see the textbook in addition to Westell and Van Vleck (1987) and Quaas (1988).

6.4.2 Model

The mixed model with groups can be written as follows

$$\mathbf{y} = \mathbf{Xb} + \mathbf{ZQg} + \mathbf{Zu} + \mathbf{e}$$

where $\mathbf{Q}$ is a matrix relates groups to descendant animals and $\mathbf{g}$ is a fixed effect for groups. Actually, why are interested in $\mathbf{u}^* = \mathbf{Qg} + \mathbf{u}$, in other words, the total breeding value of the individual. Assuming a single-trait model, the resulting mixed model equations are

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} & \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{A}^{-1}_{nn}/\sigma^2_u & \mathbf{A}^{-1}_{np}/\sigma^2_u \\ \mathbf{0} & \mathbf{A}^{-1}_{pn}/\sigma^2_u & \mathbf{A}^{-1}_{pp}/\sigma^2_u \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}}^* \\ \hat{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{0} \end{bmatrix}$$

and

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}_{nn} & \mathbf{A}^{-1}_{np} \\ \mathbf{A}^{-1}_{pn} & \mathbf{A}^{-1}_{pp} \end{bmatrix}$$

where subscript n in $\mathbf{A}^{-1}$ is for real animals and the subscript p is for genetic groups. This $\mathbf{A}^{-1}$ can be calculated with a regular rule with a small extension. BLUPF90 supports genetic groups in a regular framework. In BLUPF90 literature, unknown parent groups (UPG/UPGs) are preferred to describe this concept.

6.4.3 Files

We can use the same data file as before (`data_mr03d.txt`).

The pedigree file (`pedigree_mr03d.txt`) has to contain a group code instead of 0 (unknown parent). This pedigree file must contain 4 columns.

1. Animal ID (1 or larger integer).
2. Sire ID. Can contain a group code.
3. Dam ID. Can contain a group code.
4. Missing parent status: 1 = both parents known, 2 = one parent known and 3 = both parents unknown.

A group code must be larger than the last ID for real animals. In this example, codes 9 and 10 should be unknown parent codes. We would consider now we have 10 “animals” (actually 8 actual animals + 2 UPG) in the pedigree. The 4th column is needed to immediately calculate Mendelian sampling variance for each animal.

The parameter file is also similar to the previous one. You should make 2 changes.

Listing 6.9: param_mr03d.txt

```
DATAFILE
data_mr03d.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS:
2 2 cross
1 10 cross # change the number of animals to 10 including groups
RANDOM_RESIDUAL VALUES
40.0
RANDOM_GROUP
2
RANDOM_TYPE # use upg = unknown parent groups
add_an_upg
FILE
pedigree_mr03d.txt
(CO)VARIANCES
20.0
OPTION solv_method FSPAK
```

6.4.4 Solutions

We will have the following solutions with the above files.

Listing 6.10: solutions

```
trait/effect level solution
   1 1 1 4.35899605
   1 1 2 3.21434490
   1 2 1 0.33201730
   1 2 2 0.17597293
   1 2 3 0.13611581
   1 2 4 -0.16882873
   1 2 5 0.00000000
   1 2 6 0.37131040
   1 2 7 -0.23933070
   1 2 8 0.33117127
   1 2 9 1.09907877
   1 2 10 -0.66967475
```

Do you think these values are very different from the ones in textbook? The left-hand side of mixed model equations is not full rank; the groups are treated as fixed so the model contains 2 fixed class effects. We should calculate the contrast (difference between the solutions) to obtain the unique solution. For example, the difference in breeding values for animal 1 and 2 ($u_1 - u_2$) is 0.156; it is consistent between this example and the textbook.

In the textbook, the author puts the zero-constraint to the group 1 (corresponding to animal 9 here) but BLUPF90 puts the constraint on the animal 5. BLUPF90 automatically detects linear-dependency in the equations and puts a constraint to any columns and rows as soon as it finds it necessary. To make a zero-constraint on the animal 9 in BLUPF90, replace 9 with 0 in the pedigree file and run BLUPF90 with the same parameter file. You will obtain the following results.

Listing 6.11: solutions

```
trait/effect level solution
   1 1 1 5.45807483
   1 1 2 4.31342367
   1 2 1 -0.76706148
   1 2 2 -0.92310585
   1 2 3 -0.96296296
   1 2 4 -1.26790750
   1 2 5 -1.09907877
   1 2 6 -0.72776838
   1 2 7 -1.33840948
   1 2 8 -0.76790750
   1 2 9 0.00000000
   1 2 10 -1.76875353
```

Be aware that numerical error is large because of very small data set. Considering the rounding error, those values are in the similar level to the textbook.

This analysis provides us a caution about the solutions of breeding values from unknown parent groups. Depending on methods or constraints, we obtain an infinite number of solutions because there is the dependency between the group effect and other fixed effects. The raw solutions have already included the group effects so the absolute values of estimated breeding values are nonsense. We should look at the difference between two solutions. A better way to print the breeding value is to adjust it with the average of animals with a similar condition. For example, in dairy cattle, the estimated breeding value is adjusted with the average of breeding values for cows born in specific year. This reference population is called genetic base. In a population undergoing selection, the average breeding value is continuously increasing (or decreasing) so the genetic base should be revised every few years.

6.5 Repeatability model

6.5.1 Model

A repeatability model is an animal model applicable when an animal has more than 1 observations. In such case, we can split the residual effect into 2: "true" temporary environmental effect and a permanent environmental effect which belongs to the animal. So the model contains 2 types of animal effects (additive genetic and permanent environmental effects).

The author assumes the repeatability model in Example 4.1 for fat yield in dairy cattle. The model has the fixed effects for parity and Herd-Year-Season, the additive genetic effect for animal, the permanent environmental effect for animal, and the random residual effect. The author assumes the genetic variance is 20, the residual variance is 28, and the permanent-environmental variance is 12.

6.5.2 Files

Data file (`data_mr04a.txt`) just contains the table shown at p.63 in the textbook.

This file has 6 columns. The column 2 and 3 are not actually used in this analysis. 1. Animal ID (cow) 2. Sire ID 3. Dam ID 4. Parity (1 or 2) 5. HYS (from 1 to 4) 6. Fat yield (kg)

The pedigree file (`pedigree_mr04a.txt`) can be prepared as an usual, 3-column file.

This model contains 2 random effects besides the random error. The permanent environmental effects are not related each other so you should use diagonal in the RANDOM TYPE keyword. The position of permanent environmental effects is in this case the same as the additive genetic effects, because both effects are animal's individual effects - the only difference is the covariance structure. The parameter file is the following.

Listing 6.12: param_mr04a.txt

```
DATAFILE
data_mr04a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
5 4 cross
4 2 cross
1 8 cross # for additive genetic effect
1 8 cross # for permanent environmental effect
RANDOM_RESIDUAL VALUES
28.0
RANDOM_GROUP
3
RANDOM_TYPE
add_animal
FILE
pedigree_mr04a.txt
(CO)VARIANCES
20.0
RANDOM_GROUP
4
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
12.0
OPTION solv_method FSPAK
```

6.5.3 Solutions

The BLUPF90 outputs the solutions (not shown here) with the above parameter file. The left-hand side of the mixed model equations is not full rank and 2 solutions for fixed effects are replaced with 0. The positions of the zero-constraints are different from the textbook (p.64; Parity 1 and 2 here vs HYS 1 and 3 in the textbook). As before, a linear contrast is the same in both cases. BLUP for random effects are the

same to the textbook. Solutions for PE for animal 1 to 3 are 0 (the expected value as a random effect with no information) because they have no observations. In this case, the column 1 was shared for additive genetic and permanent environmental effects.

6.5.4 Putting arbitrary zero-constraints manually

There is a trick to manually put user-defined zero-constraints on the mixed model equations. You can replace the target effect code in the data file with 0. This technique was actually introduced in the previous section for unknown parent groups. The alternative data file (`data_mr04a1.txt`) provides the same solutions shown in the text file.

In the textbook, the author puts constraints on HYS 1 and 3. The 5th column in the above file has 0 corresponding to HYS 1 and 3. You can rewrite the parameter file to read the new data file; just change the `DATAFILE` block. Run BLUPF90 with the modified parameter file, and you can see different solutions.

Be careful if you use the manual constraints. If you put too many constraints on the equation, the solutions will make no sense.

6.6 Common environmental effects as random

6.6.1 Model

In this example, the author consider a simple maternal model and the maternal effects (c) do not correlate to each other. The variance components are the animal genetic variance $\sigma_u^2 = 20$, the maternal variance $\sigma_c^2 = 15$, and the residual variance $\sigma_e^2 = 65$.

6.6.2 Files

The data file (`data_mr04b.txt`) contains the whole table shown in the textbook (p.68). Here is the explanation for each column.

1. Animal ID (piglet)
2. Sire ID
3. Dam ID
4. Sex (1=male and 2=female)
5. Weaning weight (kg)

Pedigree (`pedigree_mr04b.txt`) can be derived from the above data.
The parameter file should contain 2 random effects.

Listing 6.13: param_mr04b.txt

```
DATAFILE
data_mr04b.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT [EFFECT NESTED]
```

```
4 2 cross
1 15 cross
3 5 cross # for maternal environmental effect
RANDOM_RESIDUAL VALUES
65.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr04b.txt
(CO)VARIANCES
20.0
RANDOM_GROUP
3
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
15.0
OPTION solv_method FSPAK
```

6.6.3 Solutions

The results are identical to the textbook.

6.7 Multiple-trait model with equal design matrix and no missing observations

6.7.1 Model

Here we consider a multiple-trait model with equal design matrices, which means the same model is applied to all traits. We will consider a situation without missing observations.

In this example, the author considers 2-trait model for the pre-weaning gain as the first trait (WWG) and the post-weaning gain as the second trait (PWG). Both traits have the same model i.e. the fixed effect for sex, the additive genetic effect, and the residual.

We can consider the genetic covariance between 2 traits as well as the genetic variance for each trait. This is true for residual covariance components. The genetic covariance matrix ($\mathbf{G}_0$) and the residual covariance matrix $\mathbf{R}_0$ are both symmetric matrices (2×2). The values are defined in the textbook as

$$\mathbf{G}_0 = \begin{bmatrix} 20 & 18 \\ 18 & 40 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_0 = \begin{bmatrix} 40 & 11 \\ 11 & 30 \end{bmatrix}$$

A matrix notation of above model and its mixed model equations have already introduced in the previous chapter. So here we just show some relevant equations. The model can be written as

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

We have 2 options to order the observations in $\mathbf{y}$ (within-trait or within-animal). According to a discussion in the previous chapter, BLUPF90 orders the observations within-animal (see the previous chapter for details). The variance of $\mathbf{y}$ is

$$\text{var}(\mathbf{y}) = \mathbf{Z}(\mathbf{A} \otimes \mathbf{G}_0)\mathbf{Z}' + \mathbf{I} \otimes \mathbf{R}_0$$

where $\otimes$ is the operator for Kronecker product. The mixed model equations can be written as

$$\begin{bmatrix} \mathbf{X}'(\mathbf{I}\otimes\mathbf{R}_0^{-1})\mathbf{X} & \mathbf{X}'(\mathbf{I}\otimes\mathbf{R}_0^{-1})\mathbf{Z} \\ \mathbf{Z}'(\mathbf{I}\otimes\mathbf{R}_0^{-1})\mathbf{X} & \mathbf{Z}'(\mathbf{I}\otimes\mathbf{R}_0^{-1})\mathbf{Z}+\mathbf{A}^{-1}\otimes\mathbf{G}_0^{-1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'(\mathbf{I}\otimes\mathbf{R}_0^{-1})\mathbf{y} \\ \mathbf{Z}'(\mathbf{I}\otimes\mathbf{R}_0^{-1})\mathbf{y} \end{bmatrix}$$

6.7.2 Files

The data file (`data_mr05a.txt`) is actually an extension of the Example 3.1. It has an extra observation in the last column.

1. Animal ID (calves)
2. Sire ID
3. Dam ID
4. Pre-weaning gain (WWG; kg)
5. Post-weaning gain (PWG; kg)

The pedigree file (`pedigree_mr05a.txt`) is also the same to the previous one.

The parameter file contains 2 covariance matrices. We will compute the standard error of each estimate as the square root of corresponding diagonal element of the inverse of the left-hand side of mixed model equations.

Listing 6.14: param_mr05a.txt

```
DATAFILE
data_mr05a.txt
NUMBER_OF_TRAITS
2
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
5 6
WEIGHT(S)

EFFECTS:
2 2 2 cross
1 1 8 cross
RANDOM_RESIDUAL VALUES
40.0 11.0
11.0 30.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr05a.txt
(CO)VARIANCES
20.0 18.0
18.0 40.0
OPTION solv_method FSPAK
OPTION sol se
```

6.7.3 Solutions

The file `solution` alternatively shows the solutions for the first and second traits. The solutions are identical to the values shown in the textbook (p.74).

The reliability of estimated breeding value for an animal can be calculated with the solutions from BLUPF90. The reliability for animal i and trait j (r^2_{ij}) is

$$r^2_{ij} = 1 - \frac{\mathrm{PEV}_{ij}}{\sigma^2_{u_j}}$$

where PEV_{ij} is the diagonal element of the inverse of left-hand side of mixed model equations corresponding to the effect to be considered. In the above solutions, the column `s.e.` contains the square root of the inverse. So you can square the s.e. value to obtain the PEV_{ij}. For example, the reliabilities of animal 1 can be calculated as:

$$r^2_{11} = 1 - \frac{4.313320612^2}{20} \approx 0.070 \quad \text{and} \quad r^2_{21} = 1 - \frac{5.991809902^2}{40} \approx 0.102.$$

6.8 Multiple-trait model with equal design matrix and missing records

6.8.1 Model

Here we apply the same model described in the previous example to the data with missing observations. Model descriptions and mixed model equations are identical to the previous ones.

With a missing observation, the $\mathbf{R}_0$ and its inverse should be altered. For example, assuming a 2 trait model, if the observation of the first trait is missing, the first row and column in $\mathbf{R}_0$ should be zeroed out. The corresponding inverse is the generalized inverse of this altered $\mathbf{R}_0$. Illustrating this situation with the previous example, the result is

$$\mathbf{R}_0 = \begin{bmatrix} 0 & 0 \\ 0 & 30 \end{bmatrix} \quad \text{and} \quad \mathbf{R}_0^- = \begin{bmatrix} 0 & 0 \\ 0 & 30 \end{bmatrix}^- = \begin{bmatrix} 0 & 0 \\ 0 & 1/30 \end{bmatrix}$$

The generalized inverse of this zeroed matrix is equivalent to the inverse of a matrix containing only nonzero elements in the zeroed matrix. This characteristic is explained by Searle (1971). BLUPF90 can detect a missing observation and prepares an appropriate $mathbfR_0$ and its generalized inverse. The user does not need to do anything.

6.8.2 Files

One animal is added to the previous example and 2 observations are marked as missing. The missing observation is indicated as 0, which is the default missing code used in BLUPF90 family in the data file (`data_mr05b.txt`).

We can use an extended pedigree file as previous one by adding the animal 9 (`pedigree_mr05b.txt`).

The parameter file is also identical except removed an option for standard error calculations.

Listing 6.15: param_mr05b.txt

```
DATAFILE
data_mr05b.txt
```

```
NUMBER_OF_TRAITS
2
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
5 6
WEIGHT(S)

EFFECTS:
2 2 2 cross
1 1 9 cross
RANDOM_RESIDUAL VALUES
40.0 11.0
11.0 30.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr05b.txt
(CO)VARIANCES
20.0 18.0
18.0 40.0
OPTION solv_method FSPAK
```

6.8.3 Solutions

You can confirm the results are identical to the values in the textbook (p.80).

6.9 Multiple-trait model with unequal design matrix

6.9.1 Model

When each trait has each different model, the design matrices are unequal. Even in this case, the matrix notation of the model and the mixed model equations are the same described as before.

BLUPF90 can handle a multiple-trait model with unequal design matrices in 2 ways. Both approaches provide the same results so the difference is only from a user's preference [1].

6.9.2 Files

The data are shown in the textbook at p.81, as a file `data_mr05c.txt`. This contains 2 different class effects and each trait only considers alternative one.

1. Animal ID (Cow)
2. Sire ID
3. Dam ID
4. HYS 1 (for Fat yield 1)
5. HYS 2 (for Fat yield 2)
6. Fat yield 1
7. Fat yield 2

[1] Gibbs sampling programs accept only 1 way out of 2 ways.

The pedigree (`pedigree_mr05c.txt`) is from the above data file.
First we show a parameter file to handle the unequal design matrices in one way.

Listing 6.16: param_mr05c.txt

```
DATAFILE
data_mr05c.txt
NUMBER_OF_TRAITS
2
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6 7
WEIGHT(S)

EFFECTS:
4 5 2 cross
1 1 8 cross
RANDOM_RESIDUAL VALUES
65.0 27.0
27.0 70.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr05c.txt
(CO)VARIANCES
35.0 28.0
28.0 30.0
OPTION solv_method FSPAK
```

Look at the first line in EFFECT: statements. This statement refers 2 different columns; column 4 for trait 1 and column 5 for trait 2. So you can put different effects together into the single statement. You should put the maximum number of levels among effects enumerated in this statement. For example, in this case, the maximum level in column 4 is 2 and the maximum number of level in column 5 is also 2 — so you can put 2 as the representative number of levels.

6.9.3 Solutions

This is totally identical to the textbook shown in p.82.

6.9.4 Another parameter file

BLUPF90 supports a different way to handle unequal design matrices. Consult the following parameter file with the same data and pedigree files.

Listing 6.17: param_mr05c1.txt

```
DATAFILE
data_mr05c.txt
NUMBER_OF_TRAITS
```

```
2
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
6 7
WEIGHT(S)

EFFECTS:
4 0 2 cross
0 5 2 cross
1 1 8 cross
RANDOM_RESIDUAL VALUES
65.0 27.0
27.0 70.0
RANDOM_GROUP
3
RANDOM_TYPE
add_animal
FILE
pedigree_mr05c.txt
(CO)VARIANCES
35.0 28.0
28.0 30.0
OPTION solv_method FSPAK
```

In this parameter file, the fixed effect for each trait is separately defined. The column number will be 0 for a trait which doesn't need the effect. In this case, the first line describes the effect HYS_1 and only the first trait needs this effect (so you should put 0 for the second trait). Note that Gibbs sampling programs (including GIBBS2F90 and THRGIBBS1F90) accepts only this style. This style is maybe simpler to understand.

You can immediately find the results are identical to the previous ones. Some effects (HYS_1 for trait 2 and HYS_2 for trait 1) are not estimated because they are not defined in the parameter file.

6.10 Multiple-trait model with no environmental covariance

6.10.1 Model

Assuming a 2-trait model, there is a situation where some animals are recorded only in one trait and the other animals are recorded only in another trait. In other words, no animals have the observations for both traits. This situation also happens when evaluating genotype by environment (G by E) effects. Animals in the environment 1 don't have records in environment 2 and vice versa. In these case, we can assume that the residual covariance is exactly zero.

The author assumes dual-purpose sires in cattle. The make and female calves are raised on different feeding systems. Male calves are recorded for yearling weight and females for fat yield. The genetic covariance is nonzero but the residual covariance is zero. We should note that the results from BLUPF90 must be different from the values in the textbook.

6.10.2 Files

The data set (`data_mr05d.txt`) is shown in the textbook at p.85. Note that the animal 4 is omitted because it has no observations.

1. Animal ID (Calf)

2. Sex (1=male and 2=female)
3. Sire ID
4. Dam ID
5. HYS
6. Yearling weight (kg)
7. Fat yield (kg)

The pedigree is from the above data set.
The parameter file should be the following.

Listing 6.18: param_mr05d.txt

```
DATAFILE
data_mr05d.txt
NUMBER_OF_TRAITS
2
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6 7
WEIGHT(S)

EFFECTS:
5 5 3 cross
1 1 17 cross
RANDOM_RESIDUAL VALUES
77.0 0.0
 0.0 70.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr05d.txt
(CO)VARIANCES
43.0 18.0
18.0 30.0
OPTION solv_method FSPAK
```

6.10.3 Solutions

The solutions are totally different from the textbook.

Listing 6.19: solutions

```
trait/effect level solution
   1 1 1 412.26462367
   2 1 1 194.02892921
   1 1 2 276.21351695
   2 1 2 204.76619557
   1 1 3 0.00000000
```

```
2 1 3 161.66294167
1 2 1 -3.36497774
2 2 1 1.25823921
1 2 2 -1.48909004
2 2 2 3.77372023
1 2 3 4.23664594
2 2 3 -1.68697171
1 2 4 -6.93946803
2 2 4 -1.57147632
1 2 5 -5.01220042
2 2 5 -2.09813041
1 2 6 5.01220042
2 2 6 2.09813041
1 2 7 2.13678047
2 2 7 3.56130079
1 2 8 -4.27356095
2 2 8 -7.12260158
1 2 9 -12.16152864
2 2 9 -3.09074655
1 2 10 -8.26284566
2 2 10 -1.26033550
1 2 11 5.83581177
2 2 11 3.77631522
1 2 12 12.63228129
2 2 12 3.55770600
1 2 13 1.52268184
2 2 13 5.97107079
1 2 14 -4.29201845
2 2 14 -11.52738822
1 2 15 -1.87022083
2 2 15 0.01073376
1 2 16 4.29035684
2 2 16 11.99499708
1 2 17 2.68411368
2 2 17 1.66338290
```

6.11 Animal model for a maternal trait

6.11.1 Model

For some traits, especially defined early in an animal's life, the phenotypes are largely affected by its dam. For example, weaning weight is associated with its dam's milk yield. The dam provides an environment for its progeny to increase (or decrease) their phenotypic values. The dam's contribution actually consists of genetic and environmental factors (consider milk yield). The genetic maternal contribution should be correlated with the direct genetic effect for the animal. In the end, the animal model contains a covariance matrix that accounts for direct and maternal genetic effects. Also the animal model includes the dam's contribution originally from non-genetic background.

In this example, the author introduces a typical animal model with maternal effects. Assume the following model:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Zu} + \mathbf{Wm} + \mathbf{Sp} + \mathbf{e}$$

where **y** is a vector of observations, **b** is a vector of fixed effects, **u** is a vector of additive genetic effects (animal direct effects), **m** is a vector of maternal genetic effects, **p** is a vector of maternal permanent-

environmental effects, **e** is a vector of residual effects, and other matrices are incidence matrices. The covariance structure is typically assumed as

$$\text{var}\begin{bmatrix}\mathbf{u}\\ \mathbf{m}\\ \mathbf{p}\\ \mathbf{e}\end{bmatrix} = \begin{bmatrix}\mathbf{A}\sigma_d^2 & \mathbf{A}\sigma_{dm} & \mathbf{0} & \mathbf{0}\\ \mathbf{A}\sigma_{md} & \mathbf{A}\sigma_d^2 & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\sigma_p^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\sigma_e^2\end{bmatrix}$$

where σ_u^2 is the direct genetic variance, σ_m^2 is the maternal genetic variance, $sigma_{dm}$ is the covariance between direct and maternal genetic effects, σ_p^2 is the permanent environmental variance and σ_e^2 is the residual variance. Note that this model contains a genetic covariance matrix ($\mathbf{G}_0$) even though this is a single-trait model. In this numerical example, the covariance component is

$$\mathbf{G}_0 = \begin{bmatrix}\sigma_d^2 & \sigma_{dm}\\ \sigma_{md} & \sigma_m^2\end{bmatrix} = \begin{bmatrix}150 & -40\\ -40 & 90\end{bmatrix}$$

and $\sigma_p^2 = 40$ and $\sigma_e^2 = 350$.

6.11.2 Files

Data set is shown at p.111 (`data_mr07a.txt`). Explanation of each columns are the following.

1. Animal ID (Calf)
2. Sire ID
3. Dam ID
4. Herds
5. Pen
6. Birth weight (kg)

The pedigree is derived from the above data (`pedigree_mr07a.txt`).
The parameter file should be as follows.

Listing 6.20: param_mr07a.txt

```
DATAFILE
data_mr07a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
5
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
 4 3 cross # herd
 5 2 cross # pen
 1 14 cross # direct genetic effect
 3 14 cross # maternal genetic effect
 3 7 cross # maternal permanent effect
RANDOM_RESIDUAL VALUES
350.0
RANDOM_GROUP
```

```
3 4
RANDOM_TYPE
add_animal
FILE
pedigree_mr07a.txt
(CO)VARIANCES
150.0 -40.0
-40.0 90.0
RANDOM_GROUP
5
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
40.0
OPTION solv_method FSPAK
```

In this model, we consider 5 effects (herds, pen, direct genetic, maternal genetic and maternal environmental effects). The first 2 effects are fixed. The 3rd effect is for direct genetic effects so the number of levels should be 14 (i.e. the number of animals found in pedigree). The 4th effect is for maternal genetic effect; here you can put the position of dam's ID. Note that the number of levels must be 14 because the maternal effect is correlated with the direct effect and the direct effect has 14 levels (in pedigree). In other words, each animal has both direct breeding value and maternal breeding value and the solutions will be calculated for all 14 animals. The 5th column is for maternal permanent environmental effect. This effect doesn't consider the pedigree file; it will only be estimated for dams with records in the data file. So the number of levels is only 7, because the maximum number of dam ID in the data is 7, although the position of effect is the same to previous one. [2]

Direct genetic effect and maternal genetic effect are correlated so they are simultaneously listed in `RANDOM GROUP`, which now has 3 4. This is still an animal model and you can specify add animal and the pedigree file. This is a single-trait model but it has 2 correlated genetic effects so the covariance matrix ($\mathbf{G}_0$) should be 2×2 written at `(CO)VARIANCES`.

We have one more maternal effect. It is an independent random effect. You just put 5 after `RANDOM_GROUP`. The type is diagonal. You don't need a pedigree file here. Only one variance is required in `(CO)VARIANCES`.

6.11.3 Solutions

You can confirm the estimated direct and maternal breeding values are identical to the textbook (p.113).

As noted in the textbook, this equation has a dependency between fixed effects so we will have an infinite number of solutions for fixed effects. Above solutions are one of them. If you want to obtain the same solutions presented in the textbook, you can replace 1 with 0 for herd number in the data file and run BLUPF90 again. You can see that the solutions for the fixed effects are different from the previous ones. The solutions for random effects remain unchanged.

6.12 Social interaction effects

6.12.1 Model

When an animal is raised with a few other animals in a limited space (e.g. pen or cage), there is social interaction (e.g. competition) among the animals. The animal's phenotype will be affected with its own

[2] In this case, even if you put 14 as the number of levels in the 5th effect, the program correctly works. The program can work when you put a larger number of levels than the actual number.

genetic and environmental effects as well as the competitors' contributions. The competitors' effects look like environmental contributions for the animal. This structure is similar to the maternal animal model introduced in the previous section. Some of the competitors' contributions can come from genes carried by the competitors and the rest comes from non-genetic factors. The genetic component can be correlated with the animal's direct genetic effect. A statistical model should include a complicated covariance structure.

In this example, we have 3 pens and each pen has 3 animals ($n = 3$). See the textbook for detailed formulation.

The model contains sex as the fixed effect and direct and associative genetic effects, pen (group) and common environmental (full-sibs) effects as random effects. The author assume $n = 3$ and $\sigma_g^2 = 12.12$ (group variance), $\sigma_e^2 = 60.6$ (the original residual variance), $\sigma_e^{2*} = 60.6 - 12.12 = 48.48$ (the final residual variance), $\sigma_c^2 = 12.5$ (common environmental variance) and

$$\mathbf{G}_0 = \begin{bmatrix} 25.70 & 2.25 \\ 2.25 & 3.60 \end{bmatrix}$$

6.12.2 Files

Data set is shown in p.125 (`data_mr08a.txt`). We added 3 columns as competitors' ID.

1. Animal ID
2. Sire ID
3. Dam ID
4. Pen (group)
5. Sex (1=male and 2=female)
6. Growth rate ($10\times$g/day)
7. competitor 1 in the same pen
8. competitor 2 in the same pen
9. Half-sib code (common environment)

The pedigree is derived from the data set.

The parameter file looks like a maternal animal model. The order of effects are followed by the results shown in the textbook (p.126).

Listing 6.21: param_mr08a.txt

```
DATAFILE
data_mr08a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
6
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
 5 2 cross # sex
 1 15 cross # direct genetic
 7 0 cross # associative genetic 1
 8 15 cross # associative genetic 2
 9 3 cross # common environmental
```

```
 4 3 cross # random group (pen)
RANDOM_RESIDUAL VALUES
48.48
RANDOM_GROUP
2 3
RANDOM_TYPE
add_animal
FILE
pedigree_mr08a.txt
(CO)VARIANCES
25.7 2.25
2.25 3.60
RANDOM_GROUP
5
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
12.5
RANDOM_GROUP
6
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
12.12
OPTION solv_method FSPAK
```

The parameter file is tricky. See that the 3rd effect in `EFFECTS:` is defined with 0 levels. This works with the 4th effect to put two elements on the same row in the incidence matrix $\mathbf{Z}_S$ (see p.126). Note that each statement in `EFFECTS` can put only one element on a row of the Mixed Model Equations. If we define an effect with 0 levels, this effect is not recognized as a new effect and combined with the next effect. In this case, the 3rd effect is successfully processed but the offset address is not incremented so the 4th effect will be put the same row in $\mathbf{Z}_S$.

We see what is going on processing the data with those 2 statements (effect 3 and 4). Consider the first line in the data file.

```
7 1 4 1 1 5.50 8 9 1
```

The 2 statements perform:

- read 7th column (the value is 8).; puts 1 (because of cross effect) on the column 8 in the 1st row in $\mathbf{Z}_S$.
- read 8th column (the value is 9).; puts 1 (because of cross effect) on the column 9 in the 1st row in $\mathbf{Z}_S$.

The resulting row in $\mathbf{Z}_S$ is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Note that the textbook omits the first 8 columns in $\mathbf{Z}_S$. So above row is identical to the textbook. Next, we consider the 2nd line in the data file.

```
8 1 4 1 2 9.80 7 9 1
```

The 2 statements perform: - read 7th column (the value is 7).; put 1 (because of cross effect) on the column 7 in the 1st row in $\mathbf{Z}_S$. - read 8th column (the value is 9).; put 1 (because of cross effect) on the column 9 in the 1st row in $\mathbf{Z}_S$. The resulting row in $\mathbf{Z}_S$ is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

This is identical to the 2nd row in $\mathbf{Z}_S$ in the textbook.

The specification of `RANDOM_TYPE` is also tricky. In the first `RANDOM_GROUP`, we only specify the effect 2 and 3. In this case, either the effect 3 or 4 are needed. BLUPF90 accepts only consecutive numbers in `RANDOM_GROUP` so we should put 3 here.

6.12.3 Solutions

The solutions are very similar to the values in the textbook (p.126).

6.13 Fixed regression model

6.13.1 Model

When repeated measurements are considered as the same traits over time, a repeatability model can be suitable to apply. The assumption in this situation is that the genetic correlation between observations measured at any two time points is 1. Even if the assumption is correct, the actual observations would have a non-flat average curve over time. This non-genetic change can be accounted for using (fixed) regression curves.

The author shows the test day measure of fat yield in dairy cattle. A repeatability model with a fixed regression curve is assumed. See the textbook for the details. The variance components are $\sigma^2_u = 5.521$, $\sigma^2_p = 8.470$, and $\sigma^2_e = 3.710$.

Usually, fixed regressions are nested within an environmental class. This reflects the fact that an average trajectory can be differentiated depending on specific environment including region, season, age, parity and so on. This example is small so such nested fixed regressions are not considered.

6.13.2 Files

The author presents the data set (`data_mr09a.txt`). The Legendre polynomials should be prepared by the user and saved in the data file. The following is the detail of this data.

1. Animal ID (cow)
2. Days in milk (DIM)
3. Herd-test-day class
4. Test day fat yield
5. 0th order Legendre covariable ϕ_0 (intercept)
6. 1st order Legendre covatiable ϕ_1
7. 2nd order Legendre covariable ϕ_2
8. 3rd order Legendre covariable ϕ_3
9. 4th order Legendre covariable ϕ_4

The pedigree is the same as Example 4.1 (`pedigree_mr09a.txt`).
The parameter file is shown below.

Listing 6.22: param_mr09a.txt

```
DATAFILE
data_mr09a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
8
OBSERVATION(S)
4
WEIGHT(S)

EFFECTS:
 3 10 cross # HTD
 5 1 cov # Legendre polynomials (intercept)
 6 1 cov # Legendre polynomials (1st order)
 7 1 cov # Legendre polynomials (2nd order)
 8 1 cov # Legendre polynomials (3rd order)
 9 1 cov # Legendre polynomials (4th order)
 1 8 cross # for additive genetic effect
 1 8 cross # for permanent environmental effect
RANDOM_RESIDUAL VALUES
3.710
RANDOM_GROUP
7
RANDOM_TYPE
add_animal
FILE
pedigree_mr09a.txt
(CO)VARIANCES
5.521
RANDOM_GROUP
8
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
8.470
OPTION solv_method FSPAK
```

6.13.3 Solutions

The equations have a dependency so the solutions for fixed effects are not unique. Besides the fixed effects, the results from BLUPF90 are slightly different from the textbook.

6.14 Random regression model

6.14.1 Model

Assume longitudinal data measured over time. If the genetic correlation between observations taken in 2 time points is lower than 1, the repeatability model is inappropriate. In such case, a multiple-trait model, which assumes that an observation measured at a time period is a different trait from another observation taken at a different time period, could be better to describe the trait. If the time period is long, we should assume a large number of traits. Ultimately, we could have an infinite number of traits and genetic correlations can be described using a function of 2 time periods (the environmental correlations as well). This is equivalent to the random regression model.

In this example, we fit the 2nd order Legendre polynomials (3 coefficients) to both additive genetic and permanent environmental effects. In this model, each animal has 3 random coefficients for the additive genetic effect and another 3 coefficients for the permanent environmental effect. See the textbook for the detailed modeling.

There are correlations between random regressions. This looks like a maternal model with a direct and maternal genetic covariance matrix. In this model, we have 3 random regression coefficients, each of them random effects. So the variance components are 3×3 matrix in additive genetic ($\mathbf{G}_0$) and permanent environmental effects ($\mathbf{P}_0$). The actual covariance matrices are

$$\mathbf{G}_0 = \begin{bmatrix} 3.297 & 0.594 & -1.381 \\ 0.594 & 0.921 & -0.289 \\ -1.381 & -0.289 & 1.005 \end{bmatrix} \quad \text{and} \quad \mathbf{P}_0 = \begin{bmatrix} 6.872 & -0.254 & -1.101 \\ -0.254 & 3.171 & 0.167 \\ -1.101 & 0.167 & 2.457 \end{bmatrix}$$

and $\sigma_e^2 = 3.710$.

6.14.2 Files

We use the same data and pedigree files to the previous section (`data_mr09b.txt` and `pedigree_mr09b.txt`).

The parameter file is an extension to the previous one.

Listing 6.23: param_mr09b.txt

```
DATAFILE
data_mr09b.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
12
OBSERVATION(S)
4
WEIGHT(S)

EFFECTS:
 3 10 cross # HTD
 5 1 cov # Legendre polynomials (intercept) for fixed regression
 6 1 cov # Legendre polynomials (1st order) for fixed regression
 7 1 cov # Legendre polynomials (2nd order) for fixed regression
 8 1 cov # Legendre polynomials (3rd order) for fixed regression
 9 1 cov # Legendre polynomials (4th order) for fixed regression
 5 8 cov 1 # Legendre polynomials (intercept) for additive genetic effect
 6 8 cov 1 # Legendre polynomials (1st order) for additive genetic effect
 7 8 cov 1 # Legendre polynomials (2nd order) for additive genetic effect
```

```
 5 8 cov 1 # Legendre polynomials (intercept) for permanent environmental effect
 6 8 cov 1 # Legendre polynomials (1st order) for permanent environmental effect
 7 8 cov 1 # Legendre polynomials (2nd order) for permanent environmental effect
RANDOM_RESIDUAL VALUES
3.710
RANDOM_GROUP
7 8 9
RANDOM_TYPE
add_animal
FILE
pedigree_mr09b.txt
(CO)VARIANCES
 3.297 0.594 -1.381
 0.594 0.921 -0.289
-1.381 -0.289 1.005
RANDOM_GROUP
10 11 12
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
 6.872 -0.254 -1.101
-0.254 3.171 0.167
-1.101 0.167 2.457
OPTION solv_method FSPAK
```

Additive genetic effect is defined in the effect 7, 8 and 9 and permanent environmental effect is defined in the effect 10, 11 and 12. Both have similar definition so we just explain the additive effect. Now we look at the effect 7:

```
5 8 cov 1
```

This refers to column 5 as an intercept of the regression. Because we have 8 animals in total so we should estimate 8 intercepts. The above statement means that column 5 is treated as a covariate and the regression is nested within animal (column 1). The remains 2 statements can be similarly interpreted. The corresponding `RANDOM_GROUP` statement contains 3 effects: 7, 8 and 9 so we put 3×3 genetic covariance matrix.

6.14.3 Solutions

The equations are not full rank so there are infinite solutions for fixed effects. The solutions for random effects are slightly different from the values in the textbook due to numerical error but very similar.

Additive genetic random regressions are defined as effects 7, 8 and 9. For these effects, the level number corresponds to the animal ID. With random regression model, estimated breeding values for an animal are presented as regression coefficients. In this case, each animal has 3 coefficients. For example, breeding values for animal 3 are

$$u_3 = \begin{bmatrix} 0.13166310 \\ -0.02908467 \\ 0.07039573 \end{bmatrix}.$$

6.15 Breeding values with marker information

6.15.1 Models

In this example, the author assumes one known genetic marker with the polygenic effect for a trait. The genetic marker accounts for a certain level of genetic variance. Covariances among animals in terms of the additive effects of the marker alleles can be represented as the marker covariance matrix ($\mathbf{G}_v$). Fernando and Grossman (1989) suggested a method to calculate $\mathbf{G}_v$ directly from the pedigree list and genotypes.

In this example, the author assume a model with the additive genetic (polygenic) effect, the additive genetic effect explained with a genetic marker, and the random residual effect. The author uses a pre-calculated $\mathbf{G}_v$ to estimate the partial breeding values explained by the marker. Also the model includes the additive genetic relationship matrix ($\mathbf{A}_u$) which accounts for the remaining polygenic effect. The mixed model equations are

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{A}_u^{-1}/\sigma_u^2 & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{Z} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W}+\mathbf{G}_v^{-1}/\sigma_v^2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \\ \hat{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

The variance components are $\sigma_u^2 = 0.30$, $\sigma_v^2 = 0.05$, and $\sigma_e^2 = 0.60$.

We need to overcome several issues to conduct the analysis. First, BLUPF90 has no function to calculate $\mathbf{G}_v^{-1}$. Instead, the program should read a user-supplied file containing the elements of $\mathbf{G}_v^{-1}$, the inverse of $\mathbf{G}_v$. Second, the animals in the pedigree are highly inbred. By default, BLUPF90 ignores inbreeding coefficients when calculating $\mathbf{A}^{-1}$. To consider inbreeding on $\mathbf{A}^{-1}$, a user should put an additional column to the pedigree file.

6.15.2 Files

The data file is prepared as shown in p.159 (`data_mr10a.txt`). The marker effects are inserted into the column 5 and 6.

1. Animal ID (calf)
2. Sex (1=male and 2=female)
3. Sire ID
4. Dam ID
5. Paternal QTL allele
6. Maternal QTL allele
7. Post weaning weight (kg)

Now we prepare the pedigree file with additional 4th column to consider inbreeding coefficients for $\mathbf{A}^{-1}$.

The first 3 columns are animal ID, sire ID and dam ID as usual. The 4th column is a inb/upg code specially used in BLUPF90 family. The code is 4-digit integer. It is calculated as

$$\frac{4000}{(1+m_s)(1-F_s)+(1+m_d)(1-F_d)}$$

where m_s is 0 if its sire is known or 1 if the sire is unknown, m_d is 0 if its dam is known or 1 if the dam is unknown, F_s is the inbreeding coefficient of the sire and F_d is the inbreeding coefficient of the dam. If the sire (or dam) is unknown, F_s (or F_d) is 0. In this case, the inbreeding coefficient of animal 4 is 0.25 (and

animal 5 for 0.375 but this value is not used here). The inb/upg code (c) for each animal is

$$
\begin{aligned}
c_1 &= 4000/[(1+1)(1-0)+(1+1)(1-0)] = 1000 \\
c_2 &= 4000/[(1+1)(1-0)+(1+1)(1-0)] = 1000 \\
c_3 &= 4000/[(1+0)(1-0)+(1+0)(1-0)] = 2000 \\
c_4 &= 4000/[(1+0)(1-0)+(1+0)(1-0)] = 1000 \\
c_5 &= 4000/[(1+0)(1-0.25)+(1+0)(1-0)] = 2285.7 \approx 2286.
\end{aligned}
$$

This calculation can be done with RENUMF90 with `INBREEDING` keyword.

The inverse of marker genetic matrix ($\mathbf{G}_v^{-1}$; p.164) is prepared as the following text file.

Listing 6.24: userinverse_mr10a.txt

```
  1 1 5.556
  1 2 1.000
  1 5 -5.000
...
  8 9 -0.663
  9 9 6.630
 10 10 5.556
```

The file contains 3 columns: row index, column index and the value (or column index, row index and the value). BLUPF90 supports symmetric matrix as a user-supplied inverse of relationship matrix so the program needs only upper- or lower-diagonal elements as well as the diagonal elements. The above file contains elements in and above the diagonal. Note that elements with a value of 0 do not need to be provided, which is handy when the matrix is sparse.

The parameter file should be as follows.

Listing 6.25: param_mr10a.txt

```
DATAFILE
data_mr10a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
7
WEIGHT(S)

EFFECTS:
2 2 cross # fixed effect
1 5 cross # additive genetic effect
5 0 cross # paternal QTL effect
6 10 cross # maternal QTL effect ( total 10 levels combined with paternal effect )
RANDOM_RESIDUAL VALUES
0.60
RANDOM_GROUP
2
RANDOM_TYPE # considering inbreeding
add_an_upginb
```

```
FILE
pedigree_mr10a.txt
(CO)VARIANCES
0.30
RANDOM_GROUP
4
RANDOM_TYPE # reading user - supplied file
user_file
FILE # its file name
userinverse_mr10a.txt
(CO)VARIANCES
0.05
OPTION solv_method FSPAK
```

We should be careful to describe the parameter file in terms of 3 points.

- `EFFECTS`: Use of 0 level technique. The incidence matrix for the QTL marker effects, $\mathbf{W}$, contains two 1s per row. A trick can do this. See the social interaction model for the detailed explanation. Also there are 10 levels combining paternal and maternal QTL effects.
- `RANDOM_TYPE` for the additive genetic effect: Use of add an upginb. The pedigree file contains the 4th column which indicates inb/upg code. With this keyword, BLUPF90 can read the column and build $\mathbf{A}^{-1}$ with inbreeding. Even if we don't put any unknown parent groups in this analysis, the keyword should be `add_an_upginb`.
- `RANDOM_TYPE` for the marker effect: Use of user file to read the user-supplied file. The name of file should also be supplied at FILE section.

6.15.3 Solutions

The solutions are identical to the textbook (p.166).

6.16 Directly predicting the additive genetic merit with QTL

6.16.1 Model

There is an approach to directly predict an animal's marker genetic merit. The QTL relationship matrix ($\mathbf{G}_v$) can be reduced to a relationship matrix among animals ($\mathbf{A}_v$). The mathematical model contains fixed effects, additive polygenic effects and additive genetic effects related to the marker. The mixed model equations are

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{A}_u^{-1}/\sigma_u^2 & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{Z} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W}+\mathbf{A}_v^{-1}/\sigma_q^2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \\ \hat{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

The author assumes $\sigma_u^2 = 0.30$, $\sigma_q^2 = 0.10$, and $\sigma_e^2 = 0.60$.

6.16.2 Files

We use the same data set removing the paternal and maternal QTL effects (`data_mr10b.txt`). Explanation for each column is given as follows.

1. Animal ID (calf)
2. Sex (1=male and 2=female)

3. Sire ID
4. Dam ID
5. Post weaning weight (kg)

The pedigree file is also the same as before (`pedigree_mr10b.txt`). It has 4 columns with inb/upg code.

In this case, we should prepare $\mathbf{A}_v^{-1}$ as an user-supplied file. The following file contains its diagonal and upper-triangular elements.

Listing 6.26: userinverse_mr10b.txt

```
  1 1 4.966
  1 2 0.286
  1 3 -0.148
...
  4 4 5.978
  4 5 -2.971
  5 5 4.836
```

The parameter file is as follows.

Listing 6.27: param_mr10b.txt

```
DATAFILE
data_mr10b.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS:
2 2 cross # fixed effect
1 5 cross # additive polygenic effect
1 5 cross # additive QTL effect
RANDOM_RESIDUAL VALUES
0.60
RANDOM_GROUP
2
RANDOM_TYPE # considering inbreeding
add_an_upginb
FILE
pedigree_mr10b.txt
(CO)VARIANCES
0.30
RANDOM_GROUP
3
RANDOM_TYPE # reading user-supplied file
user_file
FILE # its file name
```

```
userinverse_mr10b.txt
(CO)VARIANCES
0.10
OPTION solv_method FSPAK
```

6.16.3 Solutions

You can confirm the solutions are identical to the textbook.

6.17 Fixed effect model for SNP effects

6.17.1 Model

In this section, author tries to estimate individual SNP effect as a fixed effect based on 8 reference animals, which have phenotype (fat DYD in this case). Only 3 SNP markers are considered in this small example. We also consider the regular additive genetic (polygenic) effects. So the model is

$$y_i = \mu + \sum_{k=1}^{3} z_{ik} g_k + u_i + e_i$$

where y_i is an observation for animal i, μ is the general mean, z_{ik} is k-th weighted marker genotype for the animal i.e. the (i,k) element in the $\mathbf{Z}$ matrix (see VanRaden, 2008), g_k is the k-th fixed SNP effect, u_i is the additive genetic (polygenic) effect and e_i is the residual effect. We can consider the remaining animals in pedigree. The variance components are $\sigma_u^2 = 35.241$ and $\sigma_e^2 = 245.0$.

6.17.2 Files

In this model, the elements of $\mathbf{Z}$ are used as covariates so they should be saved in the data file. So the data file contains first 3 columns of $\mathbf{Z}$; see p.181 in the textbook. Also the weight used in this analysis is actually the inverse of EDC so the values should be calculated before the analysis.

The data file (`data_mr11a.txt`) has 10 columns.

1. Animal ID
2. Sire ID
3. Dam ID
4. General mean
5. EDC
6. Phenotype (Fat DYD)
7. Weight = inverse of EDC
8. Covariate for SNP 1
9. Covariate for SNP 2
10. Covariate for SNP 3

The pedigree file contains all 26 animals. The animal 1 to 12 have missing parents.

The parameter file for the weighted analysis is shown below.

Listing 6.28: param_mr11a.txt

```
DATAFILE
data_mr11a.txt
```

```
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
5
OBSERVATION(S)
6
WEIGHT(S)
7
EFFECTS:
 4 1 cross # general mean
 8 1 cov # SNP effect 1
 9 1 cov # SNP effect 2
10 1 cov # SNP effect 3
 1 26 cross # additive genetic ( polygenic ) effect
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
5
RANDOM_TYPE # considering inbreeding
add_animal
FILE
pedigree_mr11a.txt
(CO)VARIANCES
35.241
OPTION solv_method FSPAK
```

If you conduct the unweighted analysis, remove 7 from the WEIGHT(S) section.

6.17.3 Solutions

The following results are from the weighted analysis. The solutions are same to the textbook (p.181). The solution for the general mean is different but we believe this is a typo in the textbook.

6.18 Mixed linear model for computing SNP effects

6.18.1 Model

When we have many SNP markers, we can expect that almost all the additive genetic variation can be captured by the markers. In this case, the marker effects should be random because of too many parameters to be estimated. With the random marker model, we might not need polygenic effect. In this example, the author assumes the following model:

$$y_i = \mu + \sum_{k=1}^{m} z_{ik} g_k + u_i + e_i = \mathbf{1}\mu + \mathbf{Zu} + \mathbf{e}$$

where y_i is an observation for animal i, μ is the general mean, m is the number of markers to be considered, z_{ik} is k-th weighted marker genotype for the animal i.e. the (i,k) element in the $\mathbf{Z}$ matrix, g_k is the k-th fixed SNP effect and e_i is the residual effect. We can consider the remaining animals in pedigree. The mixed model equations are

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{I}/\sigma_g^2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{g}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

The residual variance was shown as $\sigma_e^2 = 245.0$ in the textbook. The marker variance (σ_g^2) can be estimated based on the additive genetic variance (σ_u^2) using $\sigma_u^2 / [2\sum p_j(1-p_j)]$ where p_j is the allele frequency for marker j. In this example, the author uses the latter equation and shows $2\sum p_j(1-p_j) = 3.5383$ so the variance components are $\sigma_g^2 = 35.242/3.5382 = 9.96$.

6.18.2 Files

The data file (`data_mr11b.txt`) now contains 10 columns from Z (p.184). The detailed explanation is given.

1. Animal ID
2. Sire ID
3. Dam ID
4. General mean
5. EDC
6. Phenotype (Fat DYD)
7. Weight = inverse of EDC
8. Covariate for SNP 1
9. Covariate for SNP 2
10. Covariate for SNP 3
11. Covariate for SNP 4
12. Covariate for SNP 5
13. Covariate for SNP 6
14. Covariate for SNP 7
15. Covariate for SNP 8
16. Covariate for SNP 9
17. Covariate for SNP 10

We can use the same pedigree defined before (`pedigree_mr11b.txt`).
The parameter file contains 10 SNP effects.

Listing 6.29: param_mr11b.txt

```
DATAFILE
data_mr11b.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
11
OBSERVATION(S)
6
WEIGHT(S)
7
EFFECTS:
 4 1 cross # general mean
 8 1 cov # SNP effect 1
 9 1 cov # SNP effect 2
10 1 cov # SNP effect 3
11 1 cov # SNP effect 4
12 1 cov # SNP effect 5
13 1 cov # SNP effect 6
14 1 cov # SNP effect 7
```

```
15 1 cov # SNP effect 8
16 1 cov # SNP effect 9
17 1 cov # SNP effect 10
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2 3 4 5 6 7 8 9 10 11
RANDOM_TYPE # considering inbreeding
diagonal
FILE

(CO)VARIANCES
9.96 0 0 0 0 0 0 0 0 0
0 9.96 0 0 0 0 0 0 0 0
0 0 9.96 0 0 0 0 0 0 0
0 0 0 9.96 0 0 0 0 0 0
0 0 0 0 9.96 0 0 0 0 0
0 0 0 0 0 9.96 0 0 0 0
0 0 0 0 0 0 9.96 0 0 0
0 0 0 0 0 0 0 9.96 0 0
0 0 0 0 0 0 0 0 9.96 0
0 0 0 0 0 0 0 0 0 9.96
OPTION solv_method FSPAK
```

In above parameter file, we defined 10 SNP effects as a group of random effects. The covariances among the effects are 0 so all the SNP effects are independent to each other. This description is equivalent to separately define each SNP effect (i.e. 10 `RANDOM_GROUP` blocks). A user can confirm that these 2 parameter files produce the same results.

6.18.3 Solutions

We should mention about the results shown in the textbook (p.185). The solutions from the weighted analysis seem inaccurate. The solutions come from the analysis with EDC (column 5) and the results are not correct. The correct weight is the inverse of EDC (column 7) and the following solutions are correct ones.

Listing 6.30: solutions

```
trait/effect level solution
   1 1 1 9.12440501
   1 2 1 0.00004355
   1 3 1 -0.00440133
   1 4 1 0.00439876
   1 5 1 -0.00104827
   1 6 1 0.00048476
   1 7 1 0.00229457
   1 8 1 0.00000000
   1 9 1 -0.00000000
   1 10 1 0.00179833
   1 11 1 -0.00125140
```

Unweighted results are also shown.

Listing 6.31: solutions

```
trait/effect level solution
   1 1 1 9.94392543
   1 2 1 0.08702093
   1 3 1 -0.31079216
   1 4 1 0.26246003
   1 5 1 -0.08027711
   1 6 1 0.11020813
   1 7 1 0.13908022
   1 8 1 -0.00000000
   1 9 1 0.00000000
   1 10 1 -0.06069044
   1 11 1 -0.01580233
```

6.19 GBLUP

6.19.1 Model

The SNP marker model is equivalent to a model, so-called GBLUP. The model includes a genomic relationship matrix **G**. This matrix is calculated using SNP marker data and is analogue to the pedigree-based additive relationship matrix. The **G** matrix may be defined in several ways but the best-known one is defined as

$$\mathbf{G} = \frac{\mathbf{ZZ}'}{2\sum_j p_j(1-p_j)}$$

where **Z** contains genotypes adjusted with allele frequency and p_j is the allele frequency for marker j (VanRaden 2008). We still need the pedigree file. It will be used to blend $\mathbf{A}_{22}$ (a subset of the numerator relationship matrix among genotyped animals) with **G** as

$$\mathbf{G}_w = w\mathbf{G} + (1-w)\mathbf{A}_{22}$$

where $\mathbf{G}_w$ is the final (weighted) genomic relationship matrix and w is a weight. The weighted relationship matrix is included in the equations. An appropriate weight usually ranges from 0.80 to 0.99 depending on the trait or the nature of **G**. This blending makes G positive definite to provide the inverse. Note that **G** is usually singular so we need a trick to make it have the inverse.

The mixed model equations are

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W} + \mathbf{G}_w/\sigma_u^2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

and $\sigma_u^2 = 35.241$ and $\sigma_e^2 = 245.0$ in this case. In many cases, only the general mean (μ) is considered as the fixed effect so the above equations are simplified.

BLUPF90 is actually designed for the single-step GBLUP model which contains the inverse of a numerator relationship matrix ($\mathbf{A}^{-1}$). It means that the final equations should include $\mathbf{A}^{-1}$, which is not needed in the above equations. We still have several ways to conduct the regular GBLUP in BLUPF90. The easiest one is to divide the process into 2 steps: 1) preparation of $\mathbf{G}^{-1}$ and 2) build and solve the equations. In the former step, we can use the PREGSF90 program. This program accepts the same parameter file as BLUPF90. It calculates $\mathbf{G}^{-1}$ with various options and save it to a file. BLUPF90 can read the file and use it to build the mixed model equations.

There are 2 ways to perform GBLUP with BLUPF90: 1) use of the text file for $\mathbf{G}_w^{-1}$ and the user file keyword or 2) use of the binary file for $\mathbf{G}_w^{-1}$. The first approach is simpler especially for small data set.

The second approach is more efficient but uses a trick which mimics single-step GBLUP removing the pedigree relationship matrix. In this section, using the first approach, we first demonstrate how $\mathbf{G}_w^{-1}$ can be calculated with PREGSF90 with an educational parameter file. Then we move to the second approach with a different parameter file.

6.19.2 Common files

In the both approaches, we will use the common marker file and the cross-reference file. The SNP file is as follows.

Listing 6.32: snp_mr11c.txt

```
13 20110002120000000000000000000000000000000000000000
...
26 10110201000000000000000000000000000000000000000000
```

The format of SNP file is described in the previous chapter. The textbook uses only 10 SNP markers but the above file contains 50 markers. The extra 40 markers are padding because BLUPF90 can't read less than 50 markers. All animals have the genotype 0 for the extra markers so the markers don't provide any information. The resulting **G** is the same to using only the first 10 markers.

The accompanying cross-reference file is also needed. This file relates the ID for genotyped animals and the order of animals in the marker file. The detailed explanation is available in the previous chapter.

Listing 6.33: snp_mr11c_XrefID.txt

```
13 13
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
```

6.19.3 The first approach

For PREGSF90

With the first, simpler approach, we use the data file (`data_mr11c1.txt`) defined as before. This file is the same as before except for the last column for genotyped animals with different coding.

1. The original animal ID
2. The original sire ID
3. The original dam ID

4. general mean
5. EDC
6. Observation (Fat DYD)
7. 1/EDC
8. Renumbered ID (in the order of the marker file - *this is important*)

The pedigree file (`pedigree_mr11c1.txt`) contains all the animals with and without genotypes. The program creates a subset of relationship matrix ($\mathbf{A}_{22}$) only for genotyped animals.

The parameter file for PREGSF90 is the same as BLUPF90.

Listing 6.34: preparam_mr11c1.txt

```
#
# Approach 1: creates a text G - Inverse file with PREGSF90
#
DATAFILE
data_mr11c1.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6
WEIGHT(S)
7
EFFECTS:
4 1 cross
1 26 cross
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr11c1.txt
(CO)VARIANCES
35.25
OPTION SNP_file snp_mr11c.txt snp_mr11c_XrefID.txt
OPTION no_quality_control
OPTION AlphaBeta 0.99 0.01
OPTION tunedG 0
OPTION saveAscii
OPTION saveG
OPTION saveGInverse
OPTION createGimA22i 0
```

The purpose of PREGSF90 is to create a text file with the elements of $\mathbf{G}_w$. By default, PREGSF90 tries to manipulate $\mathbf{G}_w$ in many ways. The above options control the behavior of PREGSF90. The options are case-sensitive. Simple explanations for the options are available:

- `SNP file` = Invoking the genomic routine. It needs 2 values in this case: the name of marker file and the name of cross-reference file.

- `no_quality_control` = Turns off the quality control process to remove unqualified markers and genotyped animals. In this example, we will use all the markers and genotyped animals.
- `AlphaBeta` = Defines the weight w. This option has 2 values: $\alpha = w$ and $\beta = 1 - w$. In this example, we put $w = 0.99$ which is implicitly used by the author.
- `tunedG` = Controls the tuning method for $\mathbf{G}_w$. It tries to adjust the scale of $\mathbf{G}$ to $\mathbf{A}_{22}$ using a method described by Chen et al. (2011). The argument 0 turns off the adjustment (i.e. uses the untouched $\mathbf{G}_w$).
- `saveAscii` = Uses the text format for a file to save $\mathbf{G}_w^{-1}$. The default is the binary file (a machine-readable file).
- `saveG` = Saves $\mathbf{G}_w$ to a file. The file name will be $\mathbf{G}$. Actually, you don't have to put this option (especially for large analyses). We just use it as an educational purpose to see the actual values in $\mathbf{G}_w$.
- `saveGInverse` = Saves $\mathbf{G}_w^{-1}$ to a file. The file name will be `Gi`.
- `createGimA22i` = Turns off the calculation of $\mathbf{G}_w^{-1} - \mathbf{A}_{22}$ which is required in single-step GBLUP. You don't have to put this option in this small example because the computing cost is negligible. This is important for a large-size problem.

Now we have more than 100 options just for PREGSF90. Many of them are implemented for experimental purposes and unused. We will demonstrate such options in the later chapter. In this section, a user can just recognize that there are plenty of options for PREGSF90 but only a few options are essential for the regular use.

You can run PREGSF90 with the above parameter file. You can see the basic statistics for $\mathbf{G}_w$ on the screen. Also you can find that some files are created.

- `Gen_call_rate` = Empty in this case. Just ignore it.
- `freq.count` = Allele frequency for each marker. The values should be the same as the textbook (see p.181).
- `sum2pq` = $2\sum_j p_j(1-p_j)$
- `G` = Elements in $\mathbf{G}_w$. The values are slightly different from the textbook (p.186) because the reference values are from the pure $\mathbf{G}$ (not blended with $\mathbf{A}_{22}$). You can find the reference values when you put `OPTION AlphaBeta 1.0 0.0` (but the program can't calculate its inverse because of its singularity).
- `Gi` = Elements in $\mathbf{G}_w^{-1}$. Only diagonals and upper triangular elements are stored.

From these files, only the file `Gi` is required in the analysis with BLUPF90. The first 5 rows and the last 5 rows are as follows.

```
1 1 36.374112147756
1 2 -27.980861774302
2 2 88.270926666258
1 3 -8.255548353247
2 3 31.300275470378
...
10 14 2.733882533018
11 14 6.555679888597
12 14 5.120296531692
13 14 4.504947248223
14 14 6.133181220876
```

The first 2 integers are the row and column for the element. Note that the maximum row and column ID is 14 because the rank of this matrix ($\mathbf{G}_w^{-1}$) is 14. So the row/column index doesn't directly refer to

the animal's ID. PREGSF90 internally re-assigns an ID for genotyped animals as the consecutive order of animals in the marker file. For example, row 1 and column 1 (the first line in the above file) corresponds to the relationship between animal 13 and 13 (i.e. the diagonal for the animal 13) because the animal 13 appears the first in the marker file (animal 13 = new ID 1). The row 1 and column 2 (the second line) is for the relationships between animal 13 and 14 because the animal 14 appears secondly in the marker file (animal 14 = new ID 2). Therefore, we should use the new ID for genotyped animals.

We have already prepared the data file with the new ID (8-th column). BLUPF90 can read the new column as an animal ID.

For BLUPF90

The parameter file for BLUPF90 is shown below.

Listing 6.35: param_mr11c1.txt

```
DATAFILE
data_mr11c1.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
4 1 cross
8 14 cross # new ID (renumbered only for genotyped animals)
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2
RANDOM_TYPE
user_file
FILE
Gi
(CO)VARIANCES
35.25
OPTION solv_method FSPAK
```

We use the keyword user file to read the text file for $\mathbf{G}_w^{-1}$ With this keyword, we don't need the marker file any more. BLUPF90 simply read the file and put the elements into the equations. Run BLUPF90 with the parameter file, and you will find the results. The solutions look like very similar to the values in the textbook (p.185; GBLUP).

6.19.4 Second approach

The parameter file for PREGSF90 is very similar to the previous one.

Listing 6.36: preparam_mr11c2.txt

```
DATAFILE
data_mr11c1.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
4 1 cross
1 26 cross
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr11c1.txt
(CO)VARIANCES
35.25
OPTION SNP_file snp_mr11c.txt snp_mr11c_XrefID.txt
OPTION no_quality_control
OPTION AlphaBeta 0.99 0.01
OPTION tunedG 0
#OPTION saveAscii
#OPTION saveG
OPTION saveGInverse
OPTION createGimA22i 0
```

The only differences between this parameter file and the previous one are options. We remove (actually comment out) the two options. With this parameter file, the file `Gi` is saved as a binary format readable only for the computer. Also we don't need the file `G`. Run PREGSF90 with the parameter file, and you can see the very similar output on the screen.

With the second approach, we trick BLUPF90 as if the single-step GBLUP was applied. In this analysis, we shouldn't use the pedigree file. The prepared $\mathbf{G}_w^{-1}$ is already blended with $\mathbf{A}_{22}$ so we really don't need the pedigree file. We just prepare a dummy pedigree file that is actually empty. This file contains 26 lines, which is the same number of lines as the original pedigree.

Listing 6.37: pedigree_mr11c2.txt

```
0 0 0
...
0 0 0
```

We need the data, (empty) pedigree, marker and cross-reference files as well as the file `Gi` for the analysis with BLUPF90. The parameter file for BLUPF90 is slightly different from one for PREGSF90.

Listing 6.38: param_mr11c2.txt

```
DATAFILE
data_mr11c1.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
4 1 cross
1 26 cross # use the original ID
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr11c2.txt
(CO)VARIANCES
35.25
OPTION SNP_file snp_mr11c.txt snp_mr11c_XrefID.txt
OPTION readGimA22i Gi
OPTION solv_method FSPAK
```

This parameter file is actually for single-step GBLUP. In this case, the pedigree file is empty so $\mathbf{A}^{-1}$ should be 0. Also the program will read the file `Gi` instead of creating $\mathbf{G}_w^{-1}$ and $\mathbf{A}_{22}$ with the second option (`OPTION readGimA22i`). The file contains $\mathbf{G}_w^{-1}$ so the resulting equations are for GBLUP.

Run BLUPF90 with the parameter file, and you can find the solutions that are identical to the approach 1. You can see many warnings (`Animal Id equal to Sire/Dam Id - relationships removed !!`) because we use a trick which never happens in the regular single-step GBLUP.

6.20 Mixed linear models with polygenic effects

6.20.1 Model

See the textbook for the details.

6.20.2 Files and solutions for SNP-BLUP

The data file (`data_mr11d1.txt`) is from the previous example.

We can use the same pedigree defined before (`pedigree_mr11d1.txt`).

The parameter file contains 10 SNP effects with the residual polygenic effects.

Listing 6.39: param_mr11d1.txt

```
DATAFILE
data_mr11d1.txt
NUMBER_OF_TRAITS
1
```

```
NUMBER_OF_EFFECTS
12
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
 4 1 cross # general mean
 1 26 cross # residual polygenic effects
 8 1 cov # SNP effect 1
 9 1 cov # SNP effect 2
10 1 cov # SNP effect 3
11 1 cov # SNP effect 4
12 1 cov # SNP effect 5
13 1 cov # SNP effect 6
14 1 cov # SNP effect 7
15 1 cov # SNP effect 8
16 1 cov # SNP effect 9
17 1 cov # SNP effect 10
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP # polygenic effect
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr11d1.txt
(CO)VARIANCES
3.5241
RANDOM_GROUP # jointly considering independent 10 SNP effects
3 4 5 6 7 8 9 10 11 12
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
8.9636 0 0 0 0 0 0 0 0 0
0 8.9636 0 0 0 0 0 0 0 0
0 0 8.9636 0 0 0 0 0 0 0
0 0 0 8.9636 0 0 0 0 0 0
0 0 0 0 8.9636 0 0 0 0 0
0 0 0 0 0 8.9636 0 0 0 0
0 0 0 0 0 0 8.9636 0 0 0
0 0 0 0 0 0 0 8.9636 0 0
0 0 0 0 0 0 0 0 8.9636 0
0 0 0 0 0 0 0 0 0 8.9636
OPTION solv_method FSPAK
```

You can check the solutions. See the reference values in the textbook (p.190).

6.20.3 Files and solutions for GBLUP

In this example, we use a text file for $\mathbf{G}_w^{-1}$ created with PREGSF90 in the previous section (the first approach). See the instruction and prepare the file.

The data file (`data_mr11d2.txt`) is also common to the previous one.

The pedigree file is the same as SNP-BLUP.
The parameter file is shown below.

Listing 6.40: param_mr11d2.txt

```
DATAFILE
data_mr11d2.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
6
WEIGHT(S)

EFFECTS:
4 1 cross
1 26 cross # residual polygenic effect
8 14 cross # new ID (renumbered only for genotyped animals)
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr11d2.txt
(CO)VARIANCES
3.5241
RANDOM_GROUP
3
RANDOM_TYPE
user_file
FILE
Gi
(CO)VARIANCES
31.717
OPTION solv_method FSPAK
```

The solutions are available by running the program.

6.21 Single-step approach

6.21.1 Model

We consider a regular animal model

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Wu} + \mathbf{e}.$$

If some animals are genotyped, their additive relationships are described with the genomic relationship matrix (**G**). When the genotyped and the non-genotyped animals are simultaneously considered in the relationship matrix, the resulting matrix is **H**. Its inverse falls into a simple form:

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} - \mathbf{A}_{22}^{-1} \end{bmatrix}$$

This $\mathbf{G}^{-1}$ is usually blended with the pedigree matrix. The mixed model equations are the same as the regular animal model with $\mathbf{H}^{-1}$ instead of $\mathbf{A}^{-1}$:

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{H}^{-1}/\sigma_u^2 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

where $\sigma_u^2 = 35.241$ and $\sigma_e^2 = 245.0$ in this case. BLUPF90 fully supports ssGBLUP with a minimal description in the parameter file.

6.21.2 Files

The data file is different from the previous ones (`data_mr11e.txt`).

The pedigree information is common to the previous analysis (`pedigree_mr11e.txt`).

The SNP marker file is unique for this analysis.

Listing 6.41: snp_mr11e.txt

```
 18 11010202210000000000000000000000000000000000000000
...
```

The corresponding cross-reference file is as follows.

Listing 6.42: snp_mr11e_XrefID.txt

```
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
```

The parameter file is shown as follows.

Listing 6.43: param_mr11e.txt

```
DATAFILE
data_mr11e.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
2
OBSERVATION(S)
6
WEIGHT(S)
5
EFFECTS:
4 1 cross
```

```
1 26 cross
RANDOM_RESIDUAL VALUES
245.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr11e.txt
(CO)VARIANCES
35.241
OPTION SNP_file snp_mr11e.txt snp_mr11e_XrefID.txt
OPTION no_ quality_control
OPTION AlphaBeta 0.95 0.05
OPTION tunedG 0
OPTION thrStopCorAG 0.10
OPTION solv_method FSPAK
```

BLUPF90 (actually the embedded genomic routine) usually would stop because of the very low correlation between diagonals from **G** and **A22** . The correlation should be usually high enough (otherwise there may be a problem in quality of the genotypes or of the pedigree) but it is low in this case due to the small data set. The option `thrStopCorAG` prevents the program stopping from the low correlation.

6.21.3 Solutions

Unfortunately, the solutions are totally different from the reference values in the textbook (p.193).

Listing 6.44: solutions

```
trait/effect level solution
   1 1 1 8.38509553
   1 2 1 -0.27072327
   1 2 2 2.90677899
   1 2 3 -0.27072327
   1 2 4 2.58838142
   1 2 5 -2.59488845
   1 2 6 -1.88195674
   1 2 7 -0.99299119
   1 2 8 -1.02617193
   1 2 9 -3.14377983
   1 2 10 -1.69066025
   1 2 11 -3.31615787
   1 2 12 0.81555256
   1 2 13 0.63918948
   1 2 14 4.85991512
   1 2 15 4.20216687
   1 2 16 6.46125192
   1 2 17 -1.79124924
   1 2 18 -0.39297755
   1 2 19 1.47720048
   1 2 20 -2.90484503
   1 2 21 -0.54144654
   1 2 22 0.89069967
```

```
1 2 23 -2.54427924
1 2 24 -0.10603281
1 2 25 0.94047078
1 2 26 3.65328640
```

6.22 Animal model with dominance effect

6.22.1 Model

Linear mixed models can handle a non-additive genetic model with dominance or epistatic relationship matrix. With a dominance model, we can consider the additive genetic effect and the dominance effect simultaneously. In this example, the author assumes the animal model with one fixed effect, the additive genetic effect, the dominance effect, and the residual effect. In this example, the additive variance $\sigma_u^2 = 90$, the dominance variance $\sigma_d^2 = 80$, and the residual variance $\sigma_e^2 = 120$. BLUPF90 doesn't have a function to calculate $\mathbf{D}^{-1}$ so we need to supply the elements as a file to the program.

6.22.2 Files

A data file (`data_mr12a.txt`) should be prepared.
The pedigree file (`pedigree_mr12a.txt`) is also created.
The dominance relationship matrix is supplied as a text file. See the textbook for details (p.207).

Listing 6.45: userinverse_mr12a.txt

```
 1 1 1.000
 2 2 1.000
...
11 12 -0.241
12 12 1.092
```

The parameter file defines 2 random effects.

Listing 6.46: param_mr12a.txt

```
DATAFILE
data_mr12a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
5
WEIGHT(S)

EFFECTS:
4 2 cross # fixed effect
1 12 cross # additive effect
1 12 cross # dominance effect
RANDOM_RESIDUAL VALUES
120.0
```

```
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr12a.txt
(CO)VARIANCES
90.0
RANDOM_GROUP
3
RANDOM_TYPE
user_file
FILE
userinverse_mr12a.txt
(CO)VARIANCES
80.0
OPTION solv_method FSPAK
```

6.22.3 Solutions

The solutions are identical to the reference values in the textbook (p.207).

6.23 Inversion of the dominance matrix

6.23.1 Model

A rapid algorithm to calculate $\mathbf{D}^{-1}$ (ignoring inbreeding) was developed by Hoeschele and VanRaden (1991). Their method was derived from the parental subclass effects $\mathbf{f}$ and its covariance matrix $\mathbf{F}$. The dominance effect is an interaction of genes and the effect is inherited through the combination of parents and ancestors rather than through individuals. Therefore, the dominance variation can be described with the combination of ancestral animals. Here we just introduce the idea of parental subclasses. See the original paper or the textbook for the detailed algorithm for the inverse.

For an animal i, its dominance effect di can be expressed as

$$d_i = f_{S,D} + \varepsilon$$

where $f_{S,D}$ is the expected dominance effect of many hypothetical fill-sibs from the sire S and the dam D and ε is the Mendelian sampling deviation. We assume $\mathrm{var}(f_{S,D}) = \sigma_f^2$ and it is equivalent to the dominance covariance among the full-sibs. So we can find $\sigma_f^2 = 0.25\sigma_d^2$ and $\mathrm{var}(\varepsilon) = 0.75\sigma_d^2$. The parental dominance effect actually consists of several components:

$$f_{S,D} = 0.50(f_{S,SD} + f_{S,DD} + f_{SS,D} + f_{DS,D}) - 0.25(f_{SS,SD} + f_{SS,DD} + f_{DS,SD} + f_{DS,DD}) + e$$

where SS and SD refer to the sire and the dam of sire, respectively, and SD and DD refer to the sire and the dam of dam, respectively. We have to put a label to each combination of animals (subclass). A dominance pedigree file contains the labels for above 8 subclass for each $f_{S,D}$. The vector $\mathbf{f}$ contains the parental dominance effects and its variance is

$$\mathrm{var}(\mathbf{f}) = \mathbf{F}\sigma_f^2$$

and its inverse is rapidly calculated using the dominance pedigree file. The final inverse $\mathbf{D}^{-1}$ is also calculated with $\mathbf{F}^{-1}$.

BLUPF90 calculates $\mathbf{F}^{-1}$ only. This matrix itself can't be used to estimate individual dominance effect. However, $\mathbf{F}^{-1}$ is still useful to estimate the variance components. In this section, we will prepare the files as usual even we will not solve the mixed model equations.

6.23.2 Files

The dominance pedigree file for BLUPF90 contains 10 columns.

1. Parental dominance subclass (ϕ).
2. Subclass S,SD (code:1)
3. Subclass S,DD (code:2)
4. Subclass SS,D (code:4)
5. Subclass DS,D (code:8)
6. Subclass SS,SD (code:16)
7. Subclass SS,DD (code:32)
8. Subclass DS,SD (code:64)
9. Subclass DS,DD (code:128)
10. Sum of the code for nonempty subclasses.

If the subclass is unknown or empty, put 0. The last column contains an integer value from the summation of all the code number for nonempty subclasses. The final code should be ranging from 0 to 255. For example, in $\phi = 1$ i.e. $f_{6,8}$, the final code is $2+4+32 = 38$. If the final code is 0, you can omit such a line.

The dominance pedigree file for this example is shown below.

Listing 6.47: dominance_mr12b.txt

```
1 0 2 3 0 0 6 0 0 38
...
```

The data file has to contain the parental subclass labels. The 4th column is the subclass label.

Listing 6.48: data_mr12b.txt

```
 5 2 17 5
...
```

The pedigree file is the same as the previous one (`pedigree_mr12b.txt`).

The parameter file is shown below.

Listing 6.49: param_mr12b.txt

```
DATAFILE
data_mr12b.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
```

```
3
WEIGHT(S)

EFFECTS:
2 2 cross # fixed effect
1 12 cross # additive genetic
4 6 cross # parental dominance
RANDOM_RESIDUAL VALUES
180.0
RANDOM_GROUP
2
RANDOM_TYPE
add_animal
FILE
pedigree_mr12b.txt
(CO)VARIANCES
90.0
RANDOM_GROUP
3
RANDOM_TYPE
par_domin
FILE
dominance_mr12b.txt
(CO)VARIANCES
20.0
OPTION solv_method FSPAK
```

With the key `par_domin`, BLUPF90 creates $\mathbf{F}^{-1}$ from the dominance pedigree file. Note that the parental subclass variance is one quarter of the dominance variance ($\sigma_f^2 = \sigma_d^2/4 = 80/4 = 20$). The remaining variance goes into the residual variance ($120 + 60 = 180$).

You can run BLUPF90 with this parameter file. The solutions are similar in BLUE and BLUP for the additive effects compared with the previous results. The predictions for the parental dominance are not equivalent to the previous results. In this analysis, we just consider the quarter of dominance variance through $\mathbf{F}^{-1}$. For the precise prediction of the dominance effects, a user should use the software that fully supports the dominance effect. As we mentioned before, however, the parameter file is still useful for variance component estimation with a dominance model.

What is the most efficient way to create a dominance pedigree file? A renumbering program RENDOMN supports generating the file. This program is based on the old design and different usage compared to RENUMF90. A solver supporting a dominance model is JAADOMN which is also old software. You can find the programs at `http://nce.ads.uga.edu/~ignacy/numpub/dominance/`.

6.24 The threshold model

6.24.1 Model

Some traits show categorical responses which can be potentially ordered by degree e.g. calving difficulty (easy, slightly difficult or difficult) and disease (no treatment needed, treated or died). For such traits, the phenotype is not continuous but we can still assume that both genetic and environmental factors affect the traits and there is the polygenic effect. In such a case, it is a reasonable model that each animal has an non-observable, hypothetical random variable similar to a phenotype in a continuous trait. So with various factors can contribute to the variable. When the variable exceeds a certain level, it expresses the different category of response. This is a basic idea of the threshold model and the variable is called liability and

the level is known as the threshold. The liability is usually assumed to be normally distributed in an underlying scale. We can apply a linear model to the underlying variable. In this theory, the number of thresholds is equivalent to the number of categories minus 1.

The author cites a well-known data set originally provided by Gianola and Foulley (1983). The trait is calving ease with 3 categories. The sire model for liability is as follows:

$$l_{ijkl} = H_i + C_j + s_k + e_{ijkl}$$

where l_{ijkl} is the liability, H_i and C_j are the fixed effects, s_k is the random sire effect and e_{ijkl} is the residual. The variance components are $\sigma_s^2 = 1/19 = 0.0526$ and $\sigma_e^2 = 1.0$.

The BLUPF90 family used to have several software to handle the threshold. Now THRGIBBS1F90 is the only publicly available program. This is Gibbs sampler so we need many samples to estimate the location parameters as the posterior means.

6.24.2 Files

Pedigree is prepared for the sire model.

Listing 6.50: pedigree_mr13a.txt

```
1 0 0
...
```

The items are different from the animal model.

1. Animal (sire) ID
2. Sire (sire of sire) ID
3. Maternal grand sire ID

The data file is shown below. All the phenotype should be written in the flat file. There are 28 observations from 28 calves (p.223).

Listing 6.51: data_mr13a.txt

```
1 1 1 1 1
...
2 1 4 1 20
```

The file has 4th column just for check the original row in the table.

1. Herd (H_i)
2. Sex (S_j); 1=male and 2=female
3. Sire of calf (s_k)
4. The row in the original table corresponding to the data line. The parameter file is similar to the regular one except options.

The following is the parameter file.

Listing 6.52: param_mr13a.txt

```
DATAFILE
data_mr13a.txt
NUMBER_OF_TRAITS
1
NUMBER_OF_EFFECTS
3
OBSERVATION(S)
4
WEIGHT(S)

EFFECTS: # l=H+C+s+e
1 2 cross # Herd
2 2 cross # Sex
3 4 cross # sire
RANDOM_RESIDUAL VALUES # residual variance = 1
1.0
RANDOM_GROUP
3
RANDOM_TYPE # sire model
add_sire
FILE
pedigree_mr13a.txt
(CO)VARIANCES # sire variance = 1/19
0.0526
OPTION cat 3
OPTION fixed_var mean
```

The 2 options are required to perform the current analysis.

- `OPTION cat` = defines the number of categories. This is a single-trait model so just put one value (3 categories for the trait 1). If you apply an multiple-trait model, supply as many integers as traits. The value 0 specifies that a trait is continuous (i.e. non-categorical trait).
- `OPTION fixed_var mean` = performs sampling only for the location parameters (i.e. solutions) with the fixed variance components supplied in the parameter file. This is useful when only the solutions are of interest.

You can run THRGIBBSF90 with many samples. In this case, we try 10,000 samples with 5,000 burn-in and step 1. This is too much for this small sample but we make sure we have the converged results.

Note that THRGIBBS1F90 doesn't estimate the thresholds in this case. If there are 2 thresholds (3 categories), the program assumes two thresholds are fixed to be 0 and 1. If there is 1 threshold (2 categories), the program fixes the value to 0.

6.24.3 Solutions

The solutions are stored in `final_solutions`.

Listing 6.53: solutions

```
trait/effect level solution SD
   1 1 1 9.52944466 8.75831636
   1 1 2 9.85646244 8.76150641
```

```
  1 2 1 -9.88169090 8.75776616
  1 2 2 -10.30709508 8.74507553
  1 3 1 -0.05309676 0.21738782
  1 3 2 0.06929745 0.21682631
  1 3 3 0.03496753 0.21566398
  1 3 4 -0.08110327 0.22041841
```

$$P_{11} = \Phi(t_1 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1) = 0.758$$
$$P_{12} = \Phi(t_2 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1) - \Phi(t_1 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1) = 0.197$$
$$P_{13} = 1 - \Phi(t_2 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1) = 0.044$$

where $\Phi(x)$ is the cumulative density normal function (i.e. `pnorm()` in R). The reference values are $P_{11} = 0.800$, $P_{12} = 0.129$ and $P_{13} = 0.071$. The results are apparently different. Why? There are 2 possible reasons why the results don't match.

The first one, which is actually a minor one, is over-constraint equations. The author put 2 zero-constraint to the equations. Although the left hand side (LHS) of the equations has very small 2 eigenvalues, it is not small enough to drop the rank of LHS. The problem is in the linear model. The author put the same constraints to the corresponding linear model analysis but it is obviously inappropriate.

The second one is from fixed thresholds with $\sigma_e^2 = 1.0$. When the number of category (c) is 2 i.e. binary traits, there is only 1 threshold and its value can be arbitrarily fixed. When $c = 3$, we have 2 ways to parameterize the thresholds: 1) one threshold is fixed and another threshold is estimated, or 2) both thresholds are fixed and σ_e^2 is estimated (i.e. σ_e^2 is no longer fixed to 1.0). With the latter parameterization, we need to estimate $c-3$ threshold parameters in addition to σ_e^2. THRGIBBS1F90 uses this parameterization so we actually need appropriate σ_e^2 instead of 1.0 for the fixed thresholds. In the above example, the residual variance was not appropriate. A question is why this software supports such a way. In the variance component estimation using Gibbs sampling for a threshold model with 3 or more categories (2 or more thresholds), the convergence rate for a threshold parameter is very slow. Sampling of residual variance is a better strategy when 2 or more thresholds are assumed in the model. See Sorensen et al. (1995) or Wang et al. (1997) for the detailed discussion. Thus, it may happen that `OPTION fixed_var mean` is not such a good idea. You may explore this by yourself.

6.24.4 Reasonable solutions

In this subsection, we will find ways to obtain the same solutions to the textbook. First, we try CBLUPF90THR which is available from our web server. This program supports the computation of 2 or more thresholds in a single-trait threshold model. [3]

The program implements Hoeschele et al. (1995) which generalized Foulley et al. (1983) and Janss and Foulley (1993) with unpublished work by Quaas.

The following results are from the program with the same parameter file with many iterations enough to converge (100 rounds here).

Listing 6.54: solutions

```
trait/effect level solution
  1 1 1 -0.8942
  1 1 2 -0.6168
  1 2 1 0.4014
  1 2 2 0.0425
  1 3 1 -0.0434
```

[3] Actually, it supports a multiple-trait model with one threshold trait and many linear traits.

```
 1 3 2 0.0592
 1 3 3 0.0412
 1 3 4 -0.0660
```

The thresholds are $t_1 = -0.0550$ and $t_2 = 0.5748$. The difference between thresholds is $t_2 - t_1 = 0.6298$ and it is the same to the reference analysis. With above solutions, $P_{11} = 0.800$, $P_{12} = 0.130$, and $P_{13} = 0.070$. The results are nearly identical to the reference values in the textbook.

Then, we try to run THRGIBBS1F90 with a different residual variance. Let's try $\sigma_e^2 = 3.0$. To keep the same variance ratio, σ_s^2 should be 0.158 to be $\sigma_e^2/\sigma_s^2 = 19.0$. The thresholds are fixed to $t_1 = 0.0$ and $t_2 = 1.0$. The final solutions from THRGIBBS1F90 are as follows.

Listing 6.55: solutions

```
trait/effect level solution SD
   1 1 1 16.34777283 15.17002602
   1 1 2 16.82596758 15.17639696
   1 2 1 -17.16324150 15.16948659
   1 2 2 -17.80476157 15.14659841
   1 3 1 -0.08323076 0.37743431
   1 3 2 0.10108613 0.37648625
   1 3 3 0.05924918 0.37371633
   1 3 4 -0.12015570 0.38254070
```

When $\sigma_e^2 \neq 1.0$, we should standardized the liability predictor with the residual standard deviation before applying the cumulative normal function to calculate the probabilities. See the following computations.

$$P_{11} = \Phi\left(\frac{t_1 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1}{\sigma_e}\right) = 0.813$$

$$P_{12} = \Phi\left(\frac{t_2 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1}{\sigma_e}\right) - \Phi\left(\frac{t_1 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1}{\sigma_e^2}\right) = 0.116$$

$$P_{13} = 1 - \Phi\left(\frac{t_2 - \hat{h}_1 - \hat{c}_2 - \hat{u}_1}{\sigma_e}\right) = 0.071$$

Again, note that σ_e is the standard deviation of residual variance ($\sqrt{\sigma_e^2}$) because the formula is for the standardization of latent variable. The probabilities are very similar to the reference results for all sires (p.229).

6.25 Joint analysis of quantitative and binary traits

6.25.1 Model

We can consider a 2-trait model with one threshold and one continuous trait. When we use Gibbs sampling to draw the posterior distribution for a location parameter, we can simply obtain a solution as the posterior mean. However, once we try to calculate the solutions without Gibbs sampling, the resulting equations become nonlinear and a complicated strategy is needed to solve the equations. See Wang et al. (1997) for detailed discussion. For the threshold trait, the solutions obtained with any ways should be converted in a different way as the regular threshold model. The author cited Foulley et al. (1983) and explains how to do it.

In this section, we will simply use THRGIBBS1F90 to draw the posterior mean for a location parameter. In this example, we have only 1 threshold so the program fixes it to 1.0.

6.25.2 File

The data file is from Foulley et al. (1983) and shown below.

Listing 6.56: data_mr13b.txt

```
1 1 1 1 41.0 1
1 1 1 1 37.5 1
1 1 1 2 41.5 1
1 1 2 2 40.0 1
1 1 2 2 43.0 1
1 1 2 2 42.0 1
1 1 2 2 35.0 1
2 1 1 2 46.0 1
2 1 1 2 40.5 1
2 1 2 2 39.0 1
1 2 1 1 41.4 1
1 2 1 1 43.0 2
1 2 2 2 34.0 1
1 2 2 1 47.0 2
1 2 2 1 42.0 1
2 2 2 1 44.5 1
2 2 2 1 49.0 1
1 3 1 1 41.6 1
2 3 1 1 36.0 1
2 3 1 2 42.7 1
2 3 2 2 32.5 1
2 3 2 2 44.4 1
2 3 2 1 46.0 1
1 4 2 1 47.0 2
1 4 2 2 51.0 2
1 4 2 2 39.0 1
2 4 1 1 44.5 1
1 5 1 1 40.5 1
1 5 1 2 43.5 1
1 5 2 1 42.5 1
1 5 2 1 48.8 2
1 5 2 1 38.5 1
1 5 2 1 52.0 1
1 5 2 2 48.0 1
2 5 1 2 41.0 1
2 5 1 1 50.5 2
2 5 2 1 43.7 2
2 5 2 1 51.0 2
1 6 1 2 51.6 2
1 6 1 1 45.3 2
1 6 1 2 36.5 1
1 6 2 1 50.5 1
1 6 2 1 46.0 2
1 6 2 1 45.0 1
1 6 2 2 36.0 1
```

```
2 6 1 2 43.5 1
2 6 1 2 36.5 1
```

This file contains 6 columns.

1. Heifer origin
2. Sire ID
3. Season
4. Sex (1=male and 2=female)
5. BW (body weight)
6. CD (calving difficulty)

The pedigree file contains sire, sire of sire and maternal grand sire.

Listing 6.57: pedigree_mr13b.txt

```
1 0 0
2 0 0
3 1 0
4 2 1
5 3 2
6 2 3
```

As authors mentions, we should use the corrected generic parameters ($\mathbf{G}_c$) as shown in p.235. The residual variances are 1.0 for CD and 20.0 for BW and the residual covariance is 2.0527 because the author assumes the residual correlation is 0.459. The parameter file is as follows. In our case, we exchange the order of traits i.e. the trait 1 is CD and the trait 2 is BW.

Listing 6.58: param_mr13b.txt

```
DATAFILE
data_mr13b.txt
NUMBER_OF_TRAITS
2
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
6 5
WEIGHT(S)

EFFECTS: # l=H+M+E+s+e
1 1 2 cross # origin
3 3 2 cross # season
4 4 2 cross # sex
2 2 6 cross # sire
RANDOM_RESIDUAL VALUES
1.0000 2.0527
2.0527 20.000
RANDOM_GROUP
4
RANDOM_TYPE
add_sire
```

```
FILE
pedigree_mr13b.txt
(CO)VARIANCES
0.0300 0.0302
0.0302 0.7178
OPTION cat 2 0
OPTION fixed_var mean
```

Run THRGIBBS1F90 to draw 20,000 enough samples (saved in every 10 steps) and discard the first 10,000 samples as burn-in. You can see the following solutions.

Listing 6.59: solutions

```
trait/effect level solution SD
   1 1 1 31.68328943 8.14595680
   2 1 1 169.99014434 40.28879171
   1 1 2 31.39648971 8.19080303
   2 1 2 170.59358068 40.25222186
   1 2 1 5.79121444 11.84634633
   2 2 1 -41.86423776 40.85833310
   1 2 2 5.81766544 11.83550515
   2 2 2 -40.61954962 40.79757133
   1 3 1 -37.80144849 6.85003521
   2 3 1 -84.56393688 42.77593912
   1 3 2 -38.97895373 6.85779916
   2 3 2 -87.77970197 42.74683265
   1 4 1 -0.07369834 0.17245459
   2 4 1 -0.26275007 0.75528469
   1 4 2 0.04477009 0.17053692
   2 4 2 0.10034538 0.80205603
   1 4 3 -0.04892810 0.16824589
   2 4 3 -0.19454123 0.80070554
   1 4 4 0.04296839 0.17226439
   2 4 4 0.16601219 0.80871849
   1 4 5 0.02241590 0.16946856
   2 4 5 0.33065415 0.78077812
   1 4 6 0.03339339 0.17470579
   2 4 6 0.16620116 0.78877094
```

The estimated values are close to the reference values (p.237). The small difference come from a limited number of observations. If you put zero-constraints on the fixed effects, you may obtain more similar values as in the textbook.

Just for comparison, we show the results from CBLUP90THR with the same files shown above. The estimated threshold is -0.0198 with 100 rounds.

Listing 6.60: solutions

```
trait/effect level solution
  1 1 1 -1.0100
  2 1 1 54.1493
  1 1 2 -1.2648
  2 1 2 54.7189
```

```
1 2 1 0.3162
2 2 1 -16.9395
1 2 2 0.3255
2 2 2 -15.6956
1 3 1 0.3685
2 3 1 6.3608
1 3 2 -0.7128
2 3 2 3.1574
1 4 1 -0.0601
2 4 1 -0.2699
1 4 2 0.0387
2 4 2 0.0690
1 4 3 -0.0499
2 4 3 -0.2158
1 4 4 0.0418
2 4 4 0.1376
1 4 5 0.0172
2 4 5 0.2843
1 4 6 0.0322
2 4 6 0.1358
```

7 Large-scale genetic evaluation

Actual genetic evaluation often handles a large data set. Although BLUPF90 specific program `blupf90` can read quite large data and pedigree files (for instance 1 million animals depending on the available memory), it will fail in a large-scale analysis with tens of millions of individuals. Special software is available for such a purpose with a special contract with the development team at UGA. A member of our team can access and test the software supporting the large data set. In this chapter, we will explain the usage of the programs for the large-scale genetic evaluations.

7.1 Issues in a large scale analysis

Recent computers can finish genetic evaluations or variance component estimation within a practical time even we use a relatively large data set and a complicated model which weren't handled in the past. Even with the modern computers, a genetic evaluation is a challenge both in computing time and required memory. BLUPF90, AIREMLF90 and GIBBSF90 programs are designed to store MME on memory and they could be capable of 1 million pedigree animals in a single-trait model. This is not enough in a large-scale analysis.

There are several issues to be solved in the implementation of a large-scale genetic evaluation.

- Solution of the mixed model equations.
- Calculation of accuracy of reliability of individual EBV.
- Estimation of variance components.

Basically, the mixed model equations are too large to be stored in memory. So we need a trick to solve the equations without explicit creation of the equations. In this case, we can't directly calculate the inverse of the left hand side of the equations and the prediction error variance of EBV for an animal is not available. Even if the mixed model equations are small enough to fit to the memory, the computational cost for the inverse is extremely high so this calculation is often impractical in many cases. We usually use subsets of the whole data for variance component estimation. But still, the computation takes really long time and never finishes if the data is large or the model is complicated.

We have several options to support a large-scale analysis. The BLUP90IOD2 program supports the iteration on data technique. This technique is combined with an iterative method for solving the equations. The algorithm partially builds the equations during reading the data and pedigree file and indirectly update the solutions. When we have read through the files, we have also completed one round of iteration. BLUP90IOD2 implements preconditioned conjugate gradient (PCG) as the iterative method. See Tsuruta et al. (2001) for details.

The ACCF90 program approximates the accuracy or reliability of EBV for an animal. This program creates approximated elements in left-hand side matrix for each animals. After collecting all the information, the program inverts the elements to obtain the approximated prediction-error variance (PEV). This is an iterative method but the convergence will be met quickly. The basic ides is from Misztal and Wiggans (1988) and Strabel et al. (2001).

AI REML is a primary choice of methods for variance component estimation. Typical computation in AI REML contains the inversion of the left-hand side of mixed model equations. The FSPAK package actually calculates selected elements of the inverse needed for AI REML. It can still do a good job for a small-scale analysis but it is out of business for a large-scale analysis. The package was written more than 20 years ago and its fundamental routine was written in early 1980s. The new package, YAMS,

could remove the bottleneck. The results from YAMS are compatible with the previous software but YAMS supports parallel computing or other modern techniques i.e. highly efficient computations. Now AIREMLF90 and REMLF90 can use YAMS with an option. See Masuda et al. (2014) and Masuda et al. (2015) for details.

7.2 Iteration on data with preconditioned conjugate gradient (PCG)

7.2.1 Algorithm

Preconditioned conjugate gradient (PCG) is an iterative method to solve the linear equations. This method is easily harmonized with the iteration of data technique. Intermediate status is kept in only 4 vectors and the one iteration will be done updating the vectors. BLUP90IOD2 is a program implementing the algorithms. Here we will introduce a basic idea needed to understand what the program does. See Stranden and Lidauer (2000) and Tsuruta et al. (2001) for detailed algorithm.

The mixed model equations can be written as

$$\mathbf{Cx} = \mathbf{b}$$

where $\mathbf{C}$ is the left-hand side matrix, $\mathbf{x}$ is the solution vector and $\mathbf{b}$ is the right-hand side vector. If we have a matrix $\mathbf{M}$ which is an approximation of $\mathbf{C}$, above equations are equivalent to

$$\mathbf{M}^{-1}\mathbf{Cx} = \mathbf{M}^{-1}\mathbf{b}.$$

This matrix $\mathbf{M}$ is called preconditioner. If $\mathbf{M} = \mathbf{C}$, the equations are immediately solved. BLUPF90 uses $\mathbf{M} = \mathrm{diag}(\mathbf{C})$ so its inverse is easily calculated.

The residual is expressed as

$$\mathbf{r} = \mathbf{b} - \mathbf{Cx}$$

and the algorithm tries to reduce with a statistics containing the residual. The convergence criterion is

$$\varepsilon = \frac{||\mathbf{b} - \mathbf{Cx}||^2}{||\mathbf{b}||^2}$$

where $||\cdot||$ means the norm.

7.2.2 Programs

BLUP90IOD2 is the current program to perform the iteration on data with PCG. CBLUP90IOD can handle a threshold model or threshold-linear models. BLUP90MBE is specialized to multibreed models with external information (see Legarra et al., 2007). In this example, we use the BLUP90IOD2 program.

Parallel version of BLUPF90IOD2 is now available. BLUP90IOD2OMP1 is a program supporting parallel processing in reading data and pedigree files using OpenMP. This program is useful especially for very large data set with a complicated model (like multiple-trait model). There is no advantage to use this program for small or moderate data set. The usage of this program is the same as BLUP90IOD2.

7.2.3 Files and analysis

Here we will use the same sample files as used in REML estimation.

- `simdata.txt` : data file
- `simped.txt` : pedigree file

We will apply a 4-trait animal model to this data set with the following parameter file.

Listing 7.1: iodparam1.txt

```
DATAFILE
simdata.txt
NUMBER_OF_TRAITS
4
NUMBER_OF_EFFECTS
4
OBSERVATION(S)
9 10 11 12
WEIGHT(S)

EFFECTS:
6 6 6 6 155 cross
7 7 7 7 2 cross
8 8 8 8 11 cross
1 1 1 1 4641 cross
RANDOM_RESIDUAL VALUES
 63.568 35.276 26.535 13.533
 35.276 84.627 37.831 23.306
 26.535 37.831 75.156 28.079
 13.533 23.306 28.079 46.839
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped.txt
(CO)VARIANCES
 37.150 19.471 23.885 24.246
 19.471 16.128 19.571 22.239
 23.885 19.571 31.315 33.782
 24.246 22.239 33.782 51.706
```

Run BLUP90IOD2 with the parameter file. It takes 185 rounds to meet the convergence.

```
round = 183 eps = 0.1707E-11 time = 0.01
round = 184 eps = 0.1139E-11 time = 0.01
round = 185 eps = 0.9976E-12 time = 0.01
  7.1286485E-03 seconds per round
 * END iteration: 11-14-2016 17h 52m 05s 737
 solutions stored in file: "solutions"
```

7.2.4 Options

```
OPTION conv_crit tol
```

This option defines the convergence criterion (ε) to stop the iterations. Real value tol should be very small value. The default is `1.0E-12`. The criterion should be carefully decided because the default value could be too loose but the strict criterion requires too many rounds to converge. The best practice is to compare solutions from different convergence criterions and determine the enough convergence criterion.

```
OPTION blksize n
```

This option creates a block diagonal matrix as pre-conditioner ($\mathbf{M}$). The integer value n defines the block size. By default, the pre-conditioner is the diagonal matrix (i.e. `n` is 1). The block size should be the same to the number of traits. This option will reduce the number of iterations.

7.3 Approximation of accuracy and reliability

7.3.1 Algorithm

Accuracy or reliability of EBV for an animal can be calculated from the prediction error variance (PEV) obtained from the diagonal element of the inverse of left-hand side (LHS) of the mixed model equations. The diagonal elements can be calculated with the sparse inversion with FSPAK or YAMS for smaller data set. As expected, the computation cost for the inverse is really high and the left-hand side has to be in memory. So this direct method can't be applied to a large scale genetic evaluation.

There are two major classes to approximate the accuracy and reliability. One is to approximate diagonal elements in LHS after absorbing the non-genetic effects. This element looks like the effective number of records for an animal. The other approach focuses on the effective number of progeny (daughters) originally derived in the selection index theory for progeny test. This approach is especially useful for parents.

ACCF90 implements the first approach. The method was described by Strabel et al. (2001) and Sanchez et al. (2008) who actually extended the idea by Miszal and Wiggans (1988). The algorithm is iterative because too many unknown variables are involved in the final equations to provide the approximated diagonal elements. The approximation can be done quickly.

ACCF90GS supports ssGBLUP. The algorithm was described by Miszal et al. (2013) using $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$. This program works well for smaller data set. For larger problems, now we are developing the method without explicit computation of $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$.

7.3.2 Files

We will use the following files for a 4-trait repeatability model.

- `simdata_rep.txt` : data file
- `simped.txt` : pedigree file

The parameter file includes several options for ACCF90.

Listing 7.2: accparam1.txt

```
DATAFILE
simdata_rep.txt
NUMBER_OF_TRAITS
4
NUMBER_OF_EFFECTS
```

```
5
OBSERVATION(S)
9 10 11 12
WEIGHT(S)

EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT [ EFFECT NESTED ]
6 6 6 6 155 cross
7 7 7 7 2 cross
8 8 8 8 11 cross
1 1 1 1 4641 cross
1 1 1 1 4641 cross
RANDOM_RESIDUAL VALUES
 45.124 22.357 18.626 13.762
 22.357 44.210 22.690 18.016
 18.626 22.690 46.101 22.795
 13.762 18.016 22.795 45.274
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped.txt
(CO)VARIANCES
 41.967 22.512 24.058 26.907
 22.512 17.489 19.738 24.257
 24.058 19.738 28.775 34.741
 26.907 24.257 34.741 56.668
RANDOM_GROUP
5
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
 15.546 15.237 10.490 1.4105
 15.237 40.937 19.562 6.4840
 10.490 19.562 27.782 4.5379
 1.4105 6.4840 4.5379 2.4410
OPTION cg 1 1 1 1
OPTION anim 4 4 4 4
OPTION pe 5 5 5 5
OPTION acc_maxrounds 20
```

The options will be explained later. ACCF90 reads the solutions file. Run BLUPF90 (or BLUPF90IOD2) with above parameter file and run ACCF90. The program creates the file `sol_and_acc` as follows (the first 9 lines are shown).

Listing 7.3: solutions

```
trait/effect level solution acc
   1 4 1 -3.43278933 0.8728
   2 4 1 1.63205564 0.8147
   3 4 1 1.86332893 0.8510
   4 4 1 5.42050028 0.9237
```

```
1 4 2 1.35147071 0.8754
2 4 2 1.75742269 0.8140
3 4 2 6.60794020 0.8512
4 4 2 5.09637833 0.9268
1 4 3 -9.07606792 0.8820
```

This file is similar to `solutions` but the only estimated breeding values (EBV) are stored. The last column is the reliability of EBV.

7.3.3 Options

```
OPTION cg [ position(s)] # contemporary group effect
OPTION anim [ position(s)] # additive genetic effect
OPTION pe [ position(s)] # PE effect
OPTION mat [ position(s)] # maternal genetic effect
OPTION hs [ position(s)] # herd by sire interaction effect
```

Each option defines the position(s) for the specific effect in the `EFFECT:` block. The positions should be enumerated for all traits. If the effect is missing in an trait, put `0`. The first 2 options `cg` and `anim` are mandatory and the others are optional. With above example, the contemporary group effect is the 1st effect in the `EFFECT:` block. Because the contemporary group effect are considered in all 4 traits, we put four 1s as `cg 1 1 1 1`. Similarly, the additive genetic effect is defied as the 4th effect for all the traits so we put `animal 4 4 4 4`; the permanent environmental effect is the 5th effect for all the traits so we put `pe 5 5 5 5`.

```
OPTION acc_maxrounds n # n = integer number
OPTION conv_crit x # x = real small number
```

The first option defines the maximum number of iterations (the default is 10) and the second defines the convergence criterion (the default is `1.0e-8`). The iteration stops when the program reaches either criterion.

The program doesn't need too many number of iterations. Too many iterations would add bias to the approximated reliability. The recommendation is to use the default convergence criterion. The default number of iterations wouldn't be enough so put larger number to `acc_maxrounds`.

```
OPTION type x # x = 1.0 or 0.5
```

This option defines the type of reliability. The final reliability will be calculated as $(R^2)^x$. The default x is 1.0 and it corresponds to R^2 (i.e. reliability). If you put 0.5 to x, the output is $(R^2)^{0.5} = R$ i.e. accuracy. Traditionally, the beef industry has used accuracy (x is 0.5) and the dairy has used reliability (x is 1.0). The default value is 1.0.

```
OPTION parent_avg
```

With this option, the program calculates the parent average for each animal. The values are saved in the additional column in `sol_and_acc`.

```
OPTION original_id
```

This option puts the original ID (the 10th column in the renumbered pedigree file) to `sol_and_acc`.

7.4 REML estimation with large data

7.4.1 Background

In typical REML computations, there are several bottlenecks. First, the exact solutions of the mixed model equations with the current variance components are needed. This means we should use Gaussian elimination or similar methods (direct methods). The Cholesky factorization is often employed to solve the equations. The LHS of mixed model equation generally contains many zero elements i.e. the matrix is a sparse matrix. Although we can hold only the nonzero elements to save the memory, very complicated operations are needed for the factorization.

Secondly, the REML algorithms use the first derivative of the restricted log likelihood, which contains the selected elements of the inverse of LHS. In the animal model, the derivative contains the following trace term:

$$\mathrm{tr}(\mathbf{A}^{-1}\mathbf{C}^{uu})$$

where $\mathbf{A}^{-1}$ is the inverse of the numerator relationship matrix and $\mathbf{C}^{uu}$ is the submatrix of the inverse of the left-hand side (LHS) of the mixed model equations corresponding to the solutions for the additive genetic effect ($\hat{\mathbf{u}}$). Simply speaking, the REML equations require the inverse of the LHS. The inverse of a sparse matrix is usually dense (non-sparse). You can easily imagine that its computational and storage costs are extremely high.

In this calculation, we don't actually need the whole inverse. We just need the selected elements of the inverse corresponding to non-zero elements in the original $\mathbf{C}$. The selected subset is called sparse inverse in animal breeding literature. An algorithm, so called Takahashi algorithm, can calculate the sparse inverse. This algorithm updates the Cholesky factor of the LHS.

FSPAK is a successful computer package to perform sparse operations including the factorization and sparse inversion for the LHS of mixed model equations. This package is the default solver in REMLF90 and AIREMLF90 (and BLUPF90 with `OPTION solv_method FSPAK`). It is still useful for small equations although the back-end subroutines were written more than 20 years ago. Unfortunately, the old design is really inefficient in larger equations with many dense blocks. For example, a multiple trait model (or random regression/maternal model) contains many (genetic and residual) covariance matrices in the equations. Although each covariance matrix is small, the small matrices are combined into large dense blocks during the sparse operations. This is more typical of a genomic model including the (inverse of) genomic relationship matrix, that is large and totally dense.

YAMS (Yet Another MME Solver) is a replacement of FSPAK. This package implements several advanced algorithms, including the supernodal factorization and the inverse multifrontal approach, to efficiently handle the dense blocks in the sparse matrix. YAMS intensively uses BLAS and LAPACK. For technical details, see Masuda et al. (2014) and Masuda et al. (2015).

AIREMLF90 especially implements several options to accelerate the computation and stabilize the estimation process. Here we also introduce the options.

7.4.2 Files

We will use the following files.

- `simdata_rep.txt` : data file
- `simped.txt` : pedigree file

We will use a 4-trait repeatability model with the parameter file.

Listing 7.4: complicatedparam1.txt

```
DATAFILE
simdata_rep.txt
NUMBER_OF_TRAITS
4
NUMBER_OF_EFFECTS
5
OBSERVATION(S)
9 10 11 12
WEIGHT(S)

EFFECTS: POSITIONS_INDATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT [ EFFECT NESTED ]
6 6 6 6 155 cross
7 7 7 7 2 cross
8 8 8 8 11 cross
1 1 1 1 4641 cross
1 1 1 1 4641 cross
RANDOM_RESIDUAL VALUES
100 80 80 80
80 100 80 80
80 80 100 80
80 80 80 100
RANDOM_GROUP
4
RANDOM_TYPE
add_animal
FILE
simped.txt
(CO)VARIANCES
100 80 80 80
80 100 80 80
80 80 100 80
80 80 80 100
RANDOM_GROUP
5
RANDOM_TYPE
diagonal
FILE

(CO)VARIANCES
100 80 80 80
80 100 80 80
80 80 100 80
80 80 80 100
OPTION use_yams
OPTION EM-REML 10
```

Here we put 2 options.

```
OPTION use_yams
```

This option switches the sparse library from FSPAK to YAMS. If the program has multi-threaded BLAS and LAPACK, the computations will be parallelized.

```
OPTION EM-REML n # default :0
```

This option forces the program perform EM REML in the first n rounds. The default is 0 i.e. no EM rounds are performed and AI rounds start. For complicated models, AI REML often diverges in the first round. Although this option can give a remedy for this situation, it doesn't always prevent the program from diverging.

We have several more options to accelerate the computations.

```
OPTION fact_once x
```

This option avoids the re-calculation of the Cholesky factor. It saves the Cholesky factor in temporary memory (if `x` is `memory`) or temporary file (if `x` is `file`). If you have enough memory, `memory` is preferable because of its faster computations.

```
OPTION approx_loglike
```

This skips the computation of the exact log-likelihood. If you don't need the exact value, we recommend to use this option for speed-up.

7.4.3 Results

The following results will be available in 17 rounds.

```
new R
  45.124 22.357 18.626 13.762
  22.357 44.210 22.690 18.016
  18.626 22.690 46.101 22.795
  13.762 18.016 22.795 45.274
new G
  41.967 22.512 24.058 26.907
  22.512 17.489 19.738 24.257
  24.058 19.738 28.775 34.741
  26.907 24.257 34.741 56.668
new G
  15.546 15.237 10.490 1.4105
  15.237 40.937 19.562 6.4840
  10.490 19.562 27.782 4.5379
  1.4105 6.4840 4.5379 2.4410
```

You can try alternative parameter file without use yams and EM-REML. Without YAMS, the computations will be slow. Without EM-REML, the estimates diverge in the 1st round. You can also try `fact once memory` and `approx_loglike`. Although you will not be sure that the options accelerate the computations in this small example, it surely reduces the computing time for a larger analysis.

8 Practical genomic analysis

In the previous chapters, we have reviewed how to conduct a genomic analysis with BLUPF90 programs. The results of analysis are often affected with the quality control of SNP markers and the fine tuning of genomic relationship matrix. BLUPF90 programs support a plenty of options to stabilize the analysis. Here we will explain the additional features implemented in the programs.

8.1 Files used in genomic analysis

As seen in the tutorial in the previous chapter, PREGSF90 needs 2 files to handle genomic data: SNP marker file and a cross-reference file. In this section, we describe the detailed format for these 2 files. Also, we will present some other files optionally needed in quality control of markers.

PREGSF90 reads the same parameter file as BLUPF90. From this file it reads the names of the pedigree file, the marker file and (optionally) the cross-reference file. In fact, the rest of the parameter file (data file, model) is largely ignored.

8.1.1 SNP file

The PREGSF90 program can accept 2 kinds of SNP files. One contains only integer numbers as genotypes (e.g. from SNP chips) and the other one contains real numbers as gene content (possibly from imputation software, from Genotyping by Sequence or from sequencing at low depth). First, we consider the former one; we here call it a SNP marker file.

SNP marker file

A SNP marker file is a text file which contains 2 fields: animal ID (possibly alphanumeric) in the 1st field and its genotypes in the 2nd field. We show the first and the last 5 lines from an example data presented in the previous chapter again (limited to the first 30 characters in each line to save space).

```
 8003 211011112112012000211002
 8007 212011111111012011111012
 8016 200120111021012111102121
 8019 112211021121202111011210
 8020 211110101120002021002122
   (skip)
13496 101001212012010210212201
13497 101112111021020120222111
13498 200000220202202000022222
13499 011011111111020111121112
13500 101020112122021000221001
```

The format was already described in the previous chapter.

Gene content file

The PREGSF90 program actually accepts another format. Here, we call it a *gene content file*. The gene content file may have a number of fields with the following rules.

- Animal ID must be in the 1st field and the gene content on each locus must be in the 2nd or later fields.
- Adjacent gene content should be separated with at least 1 white space i.e. this is a flexible length field file.
- No headers, no comments, no other fields are allowed. The data must start at the first line.
- In animal IDs, alphabets, numbers, and symbols (possibly ASCII) are acceptable. The link to the renumbered pedigree is provided in the Cross-reference file below.
- Basically, gene content should contain an integer, a floating-point (e.g. `3.14`), or exponential expression (e.g. `0.314E+01`) as a real number.
- No missing gene content is allowed and all animals must have the same number of gene content. The missing gene contents should be imputed before the analyses.
- The minimum number of gene content is 50. Fewer number of markers is not acceptable.

8.1.2 Cross-reference (XrefID) file

The BLUPF90 programs need a cross-reference file. This file is automatically generated with RENUMF90. This file relates a renumbered ID to the original ID for genotyped animals. Again, we show the first 5 and the last 5 lines of the actual cross-reference file presented in the previous chapter.

```
6127 8003
13570 8007
406 8016
10802 8019
10924 8020
  (skip)
8585 13496
8941 13497
9369 13498
9753 13499
9905 13500
```

This file simply contains 2 fields: the first is for renumbered ID and the second is for the original ID. The 2 fields can be separated with at least 1 space. Tab is not allowed. The order of animals must be the exactly same to the SNP file.

8.1.3 Allele frequency file (optional)

Allele frequency file contains actual allele frequency on each marker. This file is optional because the allele frequency is calculated with the current SNP file by default. Only if you expect to use external information for it, you can provide a file containing the allele frequencies. Different allele frequency may change characteristics of **G**.

Here is an example file. This contains the allele frequency only for the first 5 markers.

```
1 0.711667
2 0.328000
3 0.422000
```

```
4 0.157000
5 0.492333
```

This file contains 2 columns:

1. The position of marker (from 1 to the maximum number of markers)
2. Allele frequency as a real value (ranged from 0 to 1).

Note that PREGSF90 usually creates this file containing the allele frequency calculated from the current SNP file by default (the file name is `Freqdata.count`).

8.1.4 Map file (optional)

Map file relates a marker to a chromosome, physical location and specific name. This file is needed only when you try comprehensive quality control and GWAS with ssGBLUP. This file should contains at least 3 fields separated at least 1 white space. The 4th column is an optional and it contains the name of marker. Here we show an example (including 4th field in this case). Only the first 3 lines are shown here.

```
1 1 127 SNP-CODE-1
2 1 652 SNP-CODE-2
3 1 1022 SNP-CODE-3
```

This file follows the rules.

- The first 3 fields should contain integer values: the first is a marker number, the second specifies the chromosome number, and the third represents the physical location on the chromosome. The marker number is just an integer, not necessarily correlative, used for external data manipulation - it is not actually used by the program.
- Sex chromosome (X) can be present but it should be an integer value. The X chromosome is 0 by default.
- The 4th column (optional) can contain any alphabets, numbers, and symbols (possibly in ASCII) up to 50 characters.

8.1.5 Weight for SNP (optional)

The PREGSF90 program can create a different G matrix weighting on each SNP marker. This weighted matrix is especially useful for GWAS or related analyses with the POSTGSF90 program. If you supply this file, the weighted G will be calculated. Otherwise, weights are set to 1 i.e. no specific weight on each SNP marker. So this file is optional.

This file contains only 1 column with real values. Each row corresponds to each SNP marker; the first line contains a weight for the marker 1, and the second row contains a weight for the marker 2 and so on. The number of lines in this file shouldn't exceed the number of markers.

8.2 Quality control of SNP markers

8.2.1 Basic quality control options

The genomic module performs the quality control of SNP markers. This process removes unqualified markers and genotyped animals from the original data. The genomic module supports a variety of options for the quality control of markers. The following options change the behavior of the genomic module.

Save cleaned genotypes

The following option saves the cleaned marker information to files. By default, this option is off and the cleaned genotypes will not be saved.

```
OPTION saveCleanSNPs
```

This option generates 4 new files. We assume `snpfile` as a marker file.

- `snpfile_clean` = new SNP marker file.
- `snpfile_clean_XrefID` = new cross-reference file.
- `snpfile_SNPs_removed` = a list of removed markers.
- `snpfile_Animals_removed` = a list of removed animals.

Turn off the quality control

By default, the program performs quality control. The following option skips completely the quality control process.

```
OPTION no_quality_control
```

All the markers and genotyped animals will be retained even if they have obvious error or issues. This option is useful especially when all the markers have already been quality-controlled.

8.2.2 Detailed quality control options

The following table shows useful options for this purpose. Some options have additional arguments but you can omit them. If you omit the arguments, the default values are used. Not all the items will be checked. See the Default column in the following table. Detailed explanation for each option can be found in the official manual.

Option name	Default	Possible arguments	Description
`minfreq x`	Enable	$0 \leq x \leq 1$ (default:x = 0.05)	Minimum MAF to retain the marker
`callrate x`	Enable	$0 \leq x \leq 1$ (default:x = 0.90)	Minimum call rate
`callrateAnim x`	Enable	$0 \leq x \leq 1$ (default:x = 0.90)	Minimum call rate
`monomorphic x`	Enable	0 or 1 (default:1)	Remove monomorphic markers
`hwe x`	Skip	any real values (default:x = 0.15)	Maximum deviation of heterozygote frequency from expected
`high_correlation x y`	Skip	$-1 < x, y < 1$ (default: x = 0.025 and y = 0.995)	Prune highly correlated markers
`verify_parentage x`	Enable	0,1,2 or 3 (default:3)	Check the Mendelian coherence of parent and offspring
`exclusion_threshold x`	Enable	$0 < x < 100$ % (default:x = 1%)	Acceptance percentage of Mendelian coherence

Option name	Default	Possible arguments	Description
`exclusion_threshold_snp x`	Enable	$0 < x < 100$ % (default x = 20%)	Acceptance percentage of Mendelian coherence
`number_parent_ progeny_evaluation x`	Enable	integer value (default:100)	Minimum number of tests for Mendelian coherence
`threshold_duplicate_ samples x`	Enable	$-1 < x < 1$ (default:x = 0.9)	Check too high genomic relationships (duplicate samples)
`threshold_diagonal_g x`	Enable	real value (default:x = 1.6)	Check too high genomic inbreeding
`thrWarnCorAG x`	Enable	$-1 < x < 1$ (default:x = 0.50)	Warning: low resemblance $\mathbf{A}_{22}$ and $\mathbf{G}$
`thrStopCorAG x`	Enable	$-1 < x < 1$ (default:x = 0.30)	Error: low resemblance $\mathbf{A}_{22}$ and $\mathbf{G}$
`thrCorAG x`	Enable	$-1 < x < 1$ (default:x = 0.02)	Which values of $\mathbf{A}_{22}$ will be considered for the above

8.2.3 Extra options for quality control

The following options provide extra information from the quality control.

Option name	Default	Possible arguments	Description
`chrinfo file`	Skip	characters	Specify the chromosome map file.
`excludeCHR n1 n2 ...`	Skip	integer values	Specify the chromosome numbers to be excluded from
`sex_chr n`	Skip	integer value	Specify the number for the sex chromosome in the marker file
`plotpca`	Skip	none	Plot the first two principal components of $\mathbf{G}$.
`outcallrate`	Skip	none	Save the call rate information on SNP markers in the file

8.2.4 Numerical example for quality control

Files

We will examine the quality control with a small example including unqualified markers and animals. The following files will be used in this section.

Listing 8.1: snpqc1_XrefID.txt

```
11 ID002
15 ID003
```

```
12 ID004
2 ID006
5 ID010
7 ID011
8 ID012
9 ID013
1 ID015
```

The original SNP file is also prepared. Note that the format on pdf may look incorrect because of a typeset issue.

Listing 8.2: snpqc1.txt

```
ID002 121202011100121121200010200120202220111200011002010201100101221211012210000121 01
ID003 151202511115121122111050210120215211211100051101121110100510111111122250011001 10
ID004 222202020011012221012100110002222110211100001001120202000200222200022200002012 00
ID006 212202010110022112111121210011211111221200010101221111100120121111202210020122 01
ID010 211112110111121012121111221020202121121100011002121111100110121110112210011122 10
ID011 221112120012111122021101121011212111201000120011202020002002222000222000021221 0
ID012 222202020011012221012100110002222110211100001001120202000200222200022200002012 00
ID013 212202010121012112122210120011212210221200001001221111100110121110112210011022 00
ID015 212202010121012112122210120011212210221200001001221111100110121110112210011022 00
```

This marker file is same as the previous chapter except that there are many missing genotypes in the animal `ID003`. This animal is expected to be removed during the quality control.

Data file is as follows.

Listing 8.3: dataqc1.txt

```
3.0 1 1 1.0 2
2.0 1 2 1.0 6
4.0 1 1 2.0 8
6.0 2 2 2.0 3
3.0 2 1 1.0 5
6.0 2 2 2.0 9
6.0 3 1 2.0 4
6.0 3 2 1.0 7
8.0 3 1 1.0 10
4.0 3 2 2.0 1
```

Here is the pedigree file.

Listing 8.4: pedigreeqc1.txt

```
1 7 5 1 0 2 1 0 0 ID015
13 17 18 3 0 0 0 0 2 ID004
11 17 19 3 0 0 0 0 1 ID005
2 16 18 3 0 0 1 0 1 ID006
3 12 11 1 0 2 1 2 0 ID007
4 14 13 1 0 2 1 0 1 ID008
5 3 2 1 0 2 1 0 2 ID010
6 12 15 1 0 2 1 1 0 ID009
7 3 13 1 0 2 1 3 0 ID011
```

```
8 7 4 1 0 2 1 0 0 ID012
14 16 19 3 0 0 0 1 0 ID001
9 7 5 1 0 2 1 0 1 ID013
12 17 18 3 0 0 0 2 0 ID002
10 6 9 1 0 2 1 0 0 ID014
15 16 18 3 0 0 0 0 1 ID003
```

We will use the following parameter file for PREGSF90. The program will save the cleaned genotypes as well as call-rate information. We put loose criterion for several options because of small data. For example, we put a low call-rate criterion and relax the criterion for the correlation between $\mathbf{G}$ and $\mathbf{A}_{22}$. It is just a demonstration so don't use such low values in the real data set.

Listing 8.5: paramqc1.txt

```
# BLUPF90 parameter file created by RENF90
DATAFILE
dataqc1.txt
NUMBER_OF_TRAITS
          1
NUMBER_OF_EFFECTS
          4
OBSERVATION(S)
    1
WEIGHT(S)

EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT[EFFECT NESTED]
  2 3 cross
  3 2 cross
  4 1 cov
  5 15 cross
RANDOM_RESIDUAL VALUES
   2.0000
 RANDOM_GROUP
     4
 RANDOM_TYPE
 add_animal
 FILE
pedigreeqc1.txt
(CO)VARIANCES
  0.50000
OPTION SNP_file snpqc1.txt snpqc1_XrefID.txt
OPTION callrate 0.80
OPTION thrStopCorAG 0.10
OPTION outcallrate
OPTION saveCleanSNPs
```

Resulting files

You will see the following results.

- The new marker file `snpqc1.txt` clean is created. The corresponding new cross-reference file is `snpqc1.txt_clean_XrefID`.

- The removed markers are listed in `snpqc1.txt_SNPs_removed`. You can see 20 markers are removed.
- The removed animals are listed in `snpqc1.txt` Animals removed. The second animal (pedigree ID 15 = `ID003`) is removed due to low call rate.
- Allele frequency after quality control is saved in `freqdata.count.after.clean`. The frequency for a removed marker is shown as 0.0.
- The call rate values in the files (`callrate` and `callrate_a`) are calculated based on the original marker data.

You also find there is the file `GimA22i`. What values does it have? The following is the dump of this file.

```
-0.335 0.000 -0.161 -0.421 -0.091 0.405 -0.329 0.265 0.265
 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
-0.161 0.000 10.624 -0.101 0.614 0.005 -11.035 -0.144 -0.144
-0.421 0.000 -0.101 0.212 -0.018 0.537 -0.154 -0.342 -0.342
-0.091 0.000 0.614 -0.018 0.773 -2.964 1.684 0.291 0.291
 0.405 0.000 0.005 0.537 -2.964 0.064 -0.796 1.421 1.421
-0.329 0.000 -11.035 -0.154 1.684 -0.796 11.503 -0.587 -0.587
 0.265 0.000 -0.144 -0.342 0.291 1.421 -0.587 20.186 -20.738
 0.265 0.000 -0.144 -0.342 0.291 1.421 -0.587 -20.738 20.186
```

You can see this file has 9×9 matrix and the second animal is not actually removed from $\mathbf{G}^{-1} - \mathbf{A}_{22}^{-1}$. This matrix was calculated based on the cleaned markers keeping rows and columns cor- responding to removed animals. The program just filled 0 in such rows and columns. This is technically identical to use reconstructed relationship matrices calculated from cleaned marker file.

When you invoke genomic module from the application programs (BLUPF90, AIREMLF90, GIBBSF90 etc.), the above `GimA22i` will be directly used in the analysis. When you use PREGSF90, be careful when you use this `GimA22i` file. This file still corresponds to the original marker and cross-reference file, not to the cleaned files. For instance, if there is one animal that has been excluded by quality control, this animal is present in `GimA22i` file (with a value of 0). This situation really confuses the user. There is also another issue. If you really want to have the final matrix ($\mathbf{G}^{-1} - \mathbf{A}_{22}^{-1}$) from the cleaned genotypes, the computations to obtain the above output are totally a waste of time.

Better practice for quality control

We suggest a user to follow a protocol for quality control of markers.

1. Perform only the quality control and output the cleaned genotypes with PREGSF90.
2. Run PREGSF90 again to calculate and save the relationship matrices using the cleaned marker files.
3. Run an application program for your favorite analysis.

To short, our suggestion is to do the creation of cleaned files as the separate step. This means you need a different parameter file in each step (3 parameter files needed in total). In spite of more effort, the above procedure can avoid the confusion raised from removal of markers and animals during the quality control.

When QCF90 is released, the process will be 1. Perform quality control and output the cleaned genotypes with QCF90. 2. Run PREGSF90 to calculate and save the relationship matrices using the cleaned marker files. 3. Run an application program for your favorite analysis.

If you want to create the cleaned files only, in other words, if you want to skip all subsequent steps after the creation of the cleaned files, a series of options is helpful.

```
OPTION createA22 0 # doesn't create A22
OPTION createA22Inverse 0 # doesn't compute A22-inverse
OPTION createG 0 # doesn't create G
OPTION createGInverse 0 # doesn't compute G-inverse
OPTION createGimA22i 0 # doesn't compute (G-inv - A22-inv)
```

With these 5 options, the correlation between $\mathbf{G}$ and $\mathbf{A}_{22}$ will not be calculated. You can calculate such statistics using the cleaned files.

8.3 Tuning and input/output of relationship matrices

Stability of single-step GBLUP depends on the characteristic of $\mathbf{H}^{-1}$, which contains $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$. We have several ways to adjust $\mathbf{G}$ and $\mathbf{G}^{-1}$ to obtain reasonable results in the genomic analysis. There are also several options to harmonize between $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$. We will introduce the ideas as well as the options needed.

8.3.1 Construction of G

```
OPTION whichfreq x
```

This option defines which type of allele frequency is used to calculate $\mathbf{Z}$. The argument `x` could be 0 = use of the user-supplied file (see below), 1 = fixed value (0.5) and 2 = the current allele frequency calculated from the marker data (default). With the argument 0, the file name is supplied with the following option:

```
OPTION FreqFile x
```

where `x` is the name of file for allele frequency. The format of this file is explained in the previous section.

8.3.2 Blending

```
OPTION AlphaBeta a b
OPTION GammaDelta c d
```

These options define `a` as α, `b` as β, `c` as γ, and `d` as δ used in the blending $\alpha\mathbf{G}+\beta\mathbf{A}_{22}^{-1}+\gamma\mathbf{I}+\delta\mathbf{1}\mathbf{1}'$. The default values are $\alpha = 0.95$, $\beta = 0.05$, $\gamma = 0$, and $\delta = 0$. We usually use a restriction as $\alpha+\beta = 1$.

The original $\mathbf{G}$ matrix is completely updated (rewritten) by the blended $\mathbf{G}$. After the blending, we can't obtain the original elements in non-blended $\mathbf{G}$.

8.3.3 Tuning

```
OPTION tunedG x
```

This option defines the method to tune **G** up. The tuning scales **G** to $\mathbf{A}_{22}$. In real population, most genotypes are from the recent generations but pedigree comes from long generations. The situation would cause the incompatibility between **G** and $\mathbf{A}_{22}$ due to different amount of information accounting for changes in allele frequency. The tuning try to make **G** compatible with $\mathbf{A}_{22}$ by applying a single regression equation. See Christensen et al. (2012) for details.

The possible arguments `x` could be following.

- 0 = no scaling
- 1 = mean $([\text{diag}(\mathbf{G})]) = 1$ and mean $([\text{offdiag}(\mathbf{G})]) = 0$
- 2 = mean $([\text{diag}(\mathbf{G})]) = \text{mean}([\text{diag}(\mathbf{A}_{22})])$ and mean $([\text{offdiag}(\mathbf{G})]) = \text{mean}([\text{offdiag}(\mathbf{A}_{22})])$ (default)
- 3 = mean $(\mathbf{G}) = \text{mean}(\mathbf{A}_{22})$
- 4 = Adjustement with the 1st method described as in Powell et al. (2010) or Vitezica et al. (2011). This is very similar to (2).

8.3.4 Scaling the inverse matrices

Additional scaling will be needed for $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$ to achieve the maximum predictive ability in GEBV for young animals. The scaling factors τ and ω are used for this purpose.

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tau\mathbf{G}^{-1} - \omega\mathbf{A}_{22}^{-1} \end{bmatrix}$$

In the ideal situation, $\tau = 1$ and $\omega = 1$. The scaling would solve some issues if there are issues including incomplete pedigree and unqualified genotypes.

```
OPTION TauOmega t w
```

These options define `t` as τ, `w` as ω in above equation. The default values are $\tau = 1$ and $\omega = 1$.

The original inverses ($\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$) are completely replaced (rewritten) with the scaled ones ($\tau\mathbf{G}^{-1}$ and $\omega\mathbf{A}_{22}^{-1}$). After the scaling, we can't obtain the non-scaled elements in $\mathbf{G}^{-1}$ and $\mathbf{A}_{22}^{-1}$.

8.3.5 Output options

A relationship matrix and its inverse generated with genomic module can be saved in a file. By default, the format is binary which the software can efficiently read but human can't easily read. If you really want to save the matrix as text format, the following option is useful.

```
OPTION saveAscii
```

The format is space-separate text. It has 3 columns per line: 1) row number, 2) column number and 3) element value. This text file can be used as a user-supplied relationship matrix in BLUPF90 with user file keyword. If you need the maximum efficiency, don't use the above option.

The following options are available to output matrices. The options are case-sensitive so be careful to write them.

```
OPTION saveA22 # save A22 to a file 'A22'
OPTION saveA22Inverse # save A22-inverse to a file 'A22i'
```

```
OPTION saveG # save G to a file 'G'
OPTION saveG all # save all intermediate G
OPTION saveGInverse # save G-inverse to a file 'Gi'
```

You can't change the name of file for each matrix. If you put `OPTION saveG_all`, the program will save all the intermediate status of $\mathbf{G}$:

- `Gini` = Initial matrix i.e. $\mathbf{G} \leftarrow \mathbf{Z}\mathbf{Z}'/k$.
- `G_Alpha_Beta` = Blended matrix with $\mathbf{A}_{22}$ i.e. $\mathbf{G} \leftarrow \alpha\mathbf{G} + \beta\mathbf{A}_{22}$.
- `G_gamma` = Blended matrix with $\mathbf{I}$ i.e. $\mathbf{G} \leftarrow \mathbf{G} + \gamma\mathbf{I}$.
- `G_delta` = Blended matrix with $\mathbf{1}\mathbf{1}'$ i.e. $\mathbf{G} \leftarrow \mathbf{G} + \delta\mathbf{1}\mathbf{1}'$.
- `G_suggest` = Tuned matrix.
- `G` = The final matrix.

We should mention that the options saveGInverse and saveA22Inverse save the scaled matrices with τ and ω. This would be serious problem if you read the matrices from the files.

8.3.6 Input options

The program can read the matrices from files. In this case, the program don't calculate the matrix from marker file or pedigree file. If you read a inverse matrix from a file, the program could skip creating the original matrix. By default, the program reads only a binary file; with option `OPTION saveAscii` [1], the programs reads a text file. . The following options are available.

```
OPTION readA22 file # read A22 [ default : 'A22']
OPTION readA22Inverse file # read A22-inverse [ default : 'A22i']
OPTION readG file # read G [ default : 'G']
OPTION readGInverse file # read G-inverse [ default: 'Gi']
OPTION readGimA22i file # read (G-inverse - A22-inverse ) [ default: 'GimA22i']
```

If you can specify the name of file, the program reads the file supplied. If you omit the name of file, the program tries to read the default file shown above. For example, you can use the following option to read $\mathbf{G}^{-1}$ saved in `Gi`.

```
OPTION readGi
```

Or you can supply the name of file like this.

```
OPTION readGi condition1.Gi
```

There is a critical specification to be recognized by a user for the options `readGInverse` and `readA22Inverse`. After reading the matrix, the program do scale the matrix with τ or ω. It means that if you have already scaled the matrix before saving it to a file and you read the matrix from the file, the program scale the matrix again without any warnings or messages. Without knowing this fact, the results would be nonsense.

A better practice is as follows.

[1] yes, `OPTION saveAscii`, because `OPTION readAscii` does not exist.

1. Create and save the relationship matrix without the scaling factor(τ or ω) i.e. the default values: $\tau = 1$ and $\omega = 1$.
2. When reading the file, the scaling factors can be specified.

This strategy is really important to perform the analysis many times with different τ and ω parameters.

8.4 Perfroming GBLUP

8.4.1 GBLUP in BLUPF90

BLUPF90 is originally designed to handle general linear mixed-models with additive relationship matrix i.e. animal model. Single-step GBLUP is a simple extension of the traditional animal model and easily supported with BLUPF90 programs. The inverse of relationship matrix consists of three dense matrices as seen in the previous chapters.

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} - \mathbf{A}_{22}^{-1} \end{bmatrix}$$

Although GBLUP needs only $\mathbf{G}^{-1}$, BLUPF90 tries to create all three matrices by default. This behavior is inconvenient in GBLUP especially if pedigree is not available. BLUPF90 is still capable of GBLUP by somehow removing $\mathbf{A}^{-1}$ and $\mathbf{A}_{22}^{-1}$ from $\mathbf{H}^{-1}$. This is a trick and the user needs to be careful to use this feature. One option is to use `user_file` as explained in the previous chapter. Here, we will explain the other ways to automatically run GBLUP with BLUPF90.

Before introducing the way to apply GBLUP in BLUPF90, we have to clarify the model used in GBLUP. We assume the model has some fixed effects ($\mathbf{b}$) and additive genomic effect ($\mathbf{u}$) of animals. An animal is assumed to have only one phenotype. The matrix notation of the model is as follows.

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{u} + \mathbf{e}$$

Usually, the phenotype is a pseudo-observation such as daughter yield deviation (DYD), progeny trait deviation (PTD), de-regressed proof (DRP), and so on. Each observation has different accuracy so we should consider it by using a weight that reflects the accuracy of the observation. Furthermore, we need to know the variance components (σ_u^2 and σ_e^2 in the case) or the variance ratio ($\lambda = \sigma_e^2/\sigma_u^2$). The mixed model equations are easily derived.

$$\begin{bmatrix} \mathbf{X}'\mathbf{W}\mathbf{X}\sigma_e^{-2} & \mathbf{X}'\mathbf{W}\sigma_e^{-2} \\ \mathbf{W}\mathbf{X}\sigma_e^{-2} & \mathbf{W}\sigma_e^{-2} + \mathbf{G}^{-1}\sigma_u^{-2} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{W}\mathbf{y}\sigma_e^{-2} \\ \mathbf{W}\mathbf{y}\sigma_e^{-2} \end{bmatrix}$$

The diagonal matrix $\mathbf{W}$ has a weight on the diagonal corresponding to each observation. In the following section, we assume $\mathbf{X} = \mathbf{1}$ so that we have the general mean as the only fixed effect.

8.4.2 An automatic way with no pedigree

Idea

An automatic way to run GBLUP in BLUPF90 is to prepare the data using RENUMF90 without pedigree information. RENUMF90 creates the pedigree file that only contains the genotyped animals whose parents are missing (unknown). With this pedigree, pedigree relationships among animals will not be created. It means that all three matrices ($\mathbf{A}^{-1}$, $\mathbf{G}^{-1}$, and $\mathbf{A}_{22}^{-1}$) have the same size (as the number of genotyped animals) and also $\mathbf{A}^{-1} = \mathbf{I}$ and $\mathbf{A}_{22}^{-1}$. The resulting $\mathbf{H}^{-1}$ will contain $\mathbf{G}^{-1}$ only because the pedigree matrices are canceled out.

$$\mathbf{H}^{-1} = \mathbf{I} + \mathbf{G}^{-1} - \mathbf{I} = \mathbf{G}^{-1}$$

The user can run BLUPF90 as usual with the created files.

The user must know that $\mathbf{A}^{-1} = \mathbf{I}$ and $\mathbf{A}_{22}^{-1}$ will be explicitly created in BLUPF90. BLUPF90 needs the pedigree file to build $\mathbf{A}^{-1}$ and the file `GimA22i` has $\mathbf{G}^{-1} - \mathbf{A}_{22}^{-1} = \mathbf{G}^{-1} - \mathbf{I}$ (not pure $\mathbf{G}^{-1}$). This method just provides the same solutions as omitting $\mathbf{A}^{-1}$ and $\mathbf{A}_{22}^{-1}$ from $\mathbf{H}^{-1}$, but actually not omitted.

Numerical example

We will use an example similar to one introduced in the earlier chapter. There are 15 genotyped animals but only the first 11 animals have phenotypes. The data file has 4 columns with 1) animal ID, 2) group code i.e. general mean, 3) observation, and 4) weight on observation. This file is also the same as the previous one but it has the original ID corresponding to the marker file. If you don't use the weight, you can omit the column 4 from the data file.

Listing 8.6: rawdata6.txt

```
ID001 1 -1.28 1.00
ID002 1 -1.78 0.98
ID003 1 0.54 0.90
ID004 1 1.36 1.00
ID005 1 0.47 0.93
ID006 1 0.04 1.00
ID007 1 0.68 1.00
ID008 1 -0.89 0.85
ID009 1 -0.48 0.88
ID010 1 0.89 0.99
ID011 1 -1.16 0.97
```

The marker file should have all 15 animals.

Listing 8.7: snp6.txt

```
ID002 121202011100121121200010200120202220111200011002010201100101221211012210000121010111220021001022220
ID011 221112120012111122021101121011212111120100012001120202000200222200022200002122100002120011001010122 0
ID005 120112111101220022110001221020101011011100021101111212011210122111012210000122110000222001100100122 20
ID014 212202010110022111211120210120202220222200000002111110200011120121102220010121000002220021001011112 0
ID010 211112110111121012121111221020202121121100011002121111100110121110112210011122100002220012002010112 0
ID006 212202010110022112111121210011211111221200010101221111100120121112022100201220110022200220020111120
ID004 222202020011012221012100110002222110211000010011202020002002222000222000020120000020200200000200220
ID015 212202010121012112122210120011212210221200001001221111100110121110112210011022000002120021001020012 0
ID012 222202020011012221012100110002222110211000010011202020002002222000222000020120000020200200000200220
ID003 212202021000022111111111201001110001111200020200211212011220122112102210011122011011120021001011122 0
ID007 120112111101220022110001211020202121010100022002020202000200222200022200001122110111220012002001222 0
ID009 222202020000022221100021200111211111221200010101110201100111221212102210010221011002220021001002222 0
ID001 121112120001121122020112122010202121010100022002021102101100222200022200001221100111210012002001222 0
ID013 212202010121012112122210120011212210221200001001221111100110121110112210011022000002120021001020012 0
ID008 121112120001121121011111111001212120101100011002021102101100222200022200001111000111110021001011122 0
```

There is no pedigree file for this data.

Renumbering

An instruction file for `RENUMF90` is as follows. We assume $\sigma_u^2 = 0.3$ and $\sigma_e^2 = 0.7$. If you don't need the weight, you can put the empty line to `WEIGHT` (this keyword is mandatory so you should keep it even if you don't use the weights).

Listing 8.8: renum6.txt

```
DATAFILE
rawdata6.txt
TRAITS
3
FIELDS_PASSED TO OUTPUT

WEIGHT(S)
4
RESIDUAL_VARIANCE
0.7
EFFECT # 1st effect fixed
2 cross alpha
EFFECT # 2nd effect (animal)
1 cross alpha
RANDOM ## additive effect without pedigree
animal
SNP_FILE ## SNP marker file
snp6.txt
(CO)VARIANCES ## its variance component
0.3
OPTION AlphaBeta 0.95 0.05
OPTION tunedG 0
```

The last two line have options for BLUPF90 (passed through the resulting parameter file `renf90.par`) The first one `AlphaBeta` is needed to make $\mathbf{G}$ be positive definite. It performs the blending $\mathbf{G} \leftarrow 0.95\mathbf{G} + 0.05\mathbf{A}_{22} = 0.95\mathbf{G} + 0.05\mathbf{I}$. The second option `tunedG 0` turns off the *tuning* to scale $\mathbf{G}$ to $\mathbf{A}_{22}$; see the previous section for details. In this analysis, $\mathbf{A}_{22}$ is just a dummy (the identity matrix) so there is no reason to perform the tuning. You can also put any options you need in the parameter file.

Note that there is no `FILE` keyword for additive genetic effect but the program generates the pedigree file. After running `RENUMF90`, you will see the file `renadd02.ped` in the same directory.

```
13 0 0 3 0 10 0 0 0 ID015
4 0 0 3 0 10 1 0 0 ID004
5 0 0 3 0 10 1 0 0 ID005
6 0 0 3 0 10 1 0 0 ID006
7 0 0 3 0 10 1 0 0 ID007
8 0 0 3 0 10 1 0 0 ID008
10 0 0 3 0 10 1 0 0 ID010
9 0 0 3 0 10 1 0 0 ID009
11 0 0 3 0 10 1 0 0 ID011
14 0 0 3 0 10 0 0 0 ID012
1 0 0 3 0 10 1 0 0 ID001
15 0 0 3 0 10 0 0 0 ID013
2 0 0 3 0 10 1 0 0 ID002
12 0 0 3 0 10 0 0 0 ID014
3 0 0 3 0 10 1 0 0 ID003
```

The renumbering process assigns an integer code to each of genotyped animals. The first column (animal ID) is filled with genotyped animals but the 2nd and the 3rd columns (parents) are 0 i.e. missing.

This can create identity pedigree matrices. The 10th column has the original ID corresponding to the integer code.

The cross-reference file (XrefID) clearly show the correspondence between the integer code and the original ID. The integer code is randomly assigned to the original ID; the user should not expect any meaningful order in the integer code.

```
2 ID002
11 ID011
5 ID005
12 ID014
10 ID010
6 ID006
4 ID004
13 ID015
14 ID012
3 ID003
7 ID007
9 ID009
1 ID001
15 ID013
8 ID008
```

Running GBLUP

BLUPF90 calculates the solution of mixed model equations.

Listing 8.9: solutions

```
trait/effect level solution
   1 1 1 -0.08090772
   1 2 1 -0.42220198
   1 2 2 -0.44672494
   1 2 3 0.26450865
   1 2 4 0.24686356
   1 2 5 0.05496292
   1 2 6 0.16937536
   1 2 7 -0.17601451
   1 2 8 -0.24385662
   1 2 9 -0.25750923
   1 2 10 0.13351286
   1 2 11 0.03869522
   1 2 12 -0.10858569
   1 2 13 0.26284875
   1 2 14 0.22127690
   1 2 15 0.26284875
```

The solution of genomic effect is in rows with effect 2 (rows with the 2nd column = 2). The solutions are labeled with the integer code (not the original ID) so the user has to manually combine the solutions with the ID. The pedigree file or XrefID file can be used to combine them. If Bash is available, the following command immediately combines the solutions with the original code and sorts by the original ID.

```
paste <(sort -n snp6.txt_XrefID) <(awk '$2==2{print $3,$4}' solutions) | sort -k2,2
```

Remarks

This method will work correctly unless non-genotyped animals have phenotypes. So what will happen when non-genotyped animals have observations? The non-genotyped animals will be included in the equations and have some solutions. However, the solutions are nonsense because those animals are not related to the other animals and the estimates are just artifact. The user is responsible for the data used in GBLUP.

8.4.3 Genomic relationship matrix as external text file

As explained in the previous chapter, the user can supply $\mathbf{G}^{-1}$ as a text file. The users should prepare the file by themselves using external software like R. In this case, BLUPF90 doesn't need the marker file.

The text file contains 3 columns: 1) row index, 2) column index, and 3) the value of the element in $\mathbf{G}^{-1}$. Only lower or upper triangular part is needed; the program stops if the file has the entire matrix. The data file should be renumbered; only numerical expressions are allowed for an ID of genotyped animal and it should be corresponding to the index in the file of $\mathbf{G}^{-1}$.

Numerical example

Here we will use the same example as the previous section to use the external text file. There are 15 genotyped animals but only the first 11 animals have phenotypes. The external file of $\mathbf{G}^{-1}$ (`ginverse6.txt`) is too large to show here. Only the first some rows will be presented. The whole file is available at `https://githib.com/masuday/data` on Github.

```
1 1 8.227580098846
1 2 3.438271310942
2 2 8.954010413251
1 3 1.176962440315
2 3 .432327333335
...
```

This file is obtained from the marker file shown above (`snp6.txt`) so the index of a matrix is the same as the order of animals in the marker file.

Index in $\mathbf{G}$	ID in marker file
1	ID002
2	ID011
3	ID005
4	ID014
5	ID010
6	ID006
7	ID004
8	ID015
9	ID012
10	ID003

Index in **G**	ID in marker file
11	ID007
12	ID009
13	ID001
14	ID013
15	ID008

The data file has 4 columns with 1) animal ID, 2) group code i.e. general mean, 3) observation, 4) weight on observation, and 5) the original ID, just for your information (it has characters but no problem because this column is never read by the program). The animal ID should be compatible with the external file of $\mathbf{G}^{-1}$.

Listing 8.10: data6.txt

```
13 1 -1.28 1.00 ID001
 1 1 -1.78 0.98 ID002
10 1 0.54 0.90 ID003
 7 1 1.36 1.00 ID004
 3 1 0.47 0.93 ID005
 6 1 0.04 1.00 ID006
11 1 0.68 1.00 ID007
15 1 -0.89 0.85 ID008
12 1 -0.48 0.88 ID009
 5 1 0.89 0.99 ID010
 2 1 -1.16 0.97 ID011
```

The parameter file is as follows. The file doesn't have any genomic options because the matrix has already prepared and no additional operations are not allowed for the supplied matrix.

Listing 8.11: param6.txt

```
DATAFILE
 data6.txt
NUMBER_OF_TRAITS
 1
NUMBER_OF_EFFECTS
 2
OBSERVATION(S)
 3
WEIGHT(S)
 4
EFFECTS:
 2 1 cross
 1 15 cross
RANDOM_RESIDUAL VALUES
 0.7
RANDOM_GROUP
 2
RANDOM_TYPE
 user_file
FILE
```

```
 ginverse6.txt
(CO)VARIANCES
  0.3
```

Be careful to read the results; the order in `solutions` is different from the previous one. The user must combine the solutions with the original ID. The following command correctly sorts the solutions by the original ID.

```
paste <(awk '{print $1}' snp6.txt) <(awk '$2==2{print $3,$4}' solutions) | sort
```

Remarks

This method doesn't need RENUMF90. It is useful for simulated data that has renumbered ID. A problem is to compute $\mathbf{G}^{-1}$ with the external programs. Also, this method may be slow in reading the file when the matrix is too large.

8.4.4 GBLUP with partial use of pedigree

There is a case where a pedigree file is available but the user should perform GBLUP. The pedigree information can be used to form $\mathbf{G}$ but the final equations contain $\mathbf{G}^{-1}$ only. Or, the user can perform GBLUP with (residual) polygenic effect which is explained with the additive relationship matrix. The method introduced here is general and flexible but a bit complicated. We will use two software: PREGSF90 for preparation and BLUPF90 for prediction as the following protocol.

1. Run RENUMF90 to generate renumbered data and `renf90.par`.
2. Run PREGSF90 with `renf90.par`.
3. Make a copy of `renf90.f90`, modify the copy, and run BLUPF90 with the modified parameter file

Renumbering

In this method, we will use the same marker and data files (`snp6.txt` and `rawdata6.txt`) as the previous section. The pedigree file is shown below.

Listing 8.12: rawpedigree6.txt

```
ID001 0 0
ID002 0 0
ID003 0 0
ID004 0 0
ID005 0 0
ID006 0 0
ID007 ID002 ID005
ID008 ID001 ID004
ID009 ID002 ID003
ID010 ID007 ID006
ID011 ID007 ID004
ID012 ID011 ID008
ID013 ID011 ID010
ID014 ID009 ID013
ID015 ID011 ID010
```

First, we have to process the files with RENUMF90. The following instruction file is used.

Listing 8.13: renum6a.txt

```
DATAFILE
rawdata6.txt
TRAITS
3
FIELDS_PASSED TO OUTPUT

WEIGHT(S)
4
RESIDUAL_VARIANCE
0.7
EFFECT # 1st effect fixed
2 cross alpha
EFFECT # 2nd effect (animal)
1 cross alpha
RANDOM ## additive effect without pedigree
animal
FILE ## pedigree file
rawpedigree6.txt
SNP_FILE ## SNP marker file
snp6.txt
(CO)VARIANCES ## its variance component
0.3
OPTION TauOmega 1.00 0.00
```

The user can use any options for genomic setup. The option is needed to remove $\mathbf{A}_{22}^{-1}$ from $\mathbf{H}^{-1}$. The user can put more options to be suitable for your data.

Run RENUMF90, and several files including `renf90.par` will be created.

Run PREGSF90

First, look at `renf90.par` to see what we are going to do with it.

Listing 8.14: renf90.par

```
# BLUPF90 parameter file created by RENUMF90
DATAFILE
 renf90.dat
NUMBER_OF_TRAITS
          1
NUMBER_OF_EFFECTS
          2
OBSERVATION(S)
    1
WEIGHT(S)
          2
EFFECTS: POSITIONS_IN_DATAFILE NUMBER_OF_LEVELS TYPE_OF_EFFECT[EFFECT NESTED]
  3 1 cross
  4 15 cross
```

```
RANDOM_RESIDUAL VALUES
  0.70000
 RANDOM_GROUP
    2
 RANDOM_TYPE
 add_animal
 FILE
renadd02.ped
(CO)VARIANCES
  0.30000
OPTION SNP_file snp6.txt
OPTION TauOmega 1.00 0.00
```

PREGSF90 calculates the relationship matrices as follows.

1. Compute $\mathbf{A}_{22}$.
2. Compute $\mathbf{G}$.
3. Blend two matrices as $\mathbf{G} \leftarrow \alpha\mathbf{G} + \beta\mathbf{A}_{22}$.
4. Scale $\mathbf{G}$ to $\mathbf{A}_{22}$ (*tuning*).
5. Compute $\mathbf{A}_{22}^{-1}$.
6. Compute $\mathbf{G}^{-1}$.
7. Merge two inverse matrices to $\Delta = \tau\mathbf{G}^{-1} - \omega\mathbf{A}_{22}^{-1}$ and save it to a file `GimA22i`.

Note that PREGSF90 doesn't compute $\mathbf{A}^{-1}$ itself, which will be computed with BLUPF90 from the pedigree file. Using above parameter file, by default, $\alpha = 0.95$ and $\beta = 0.05$ and the pedigree matrix is integrated to $\mathbf{G}$, the tuning is performed, and `GimA22i` will have only $\mathbf{G}^{-1}$ because of $\omega = 0$. Again, the above configurations are just to demonstrate the capability of PREGSF90/BLUPF90 in GBLUP. Please use appropriate options for the user's data.

After running PREGSF90, several files will be created in the working directory. There should be a file `GimA22i`.

Run BLUPF90

BLUPF90 needs a separate parameter file that is slightly different from `renf90.par` used in PREGSF90. Make a copy of `renf90.par` as `blup.par` then modify the options in the copy as follows (showing only options).

Listing 8.15: blup.par

```
OPTION SNP_file snp6.txt
OPTION readGimA22i
OPTION omit_ainv
```

The first option `SNP_file` should be kept in the file. The remaining two options are crucial: `OPTION readGimA22i` reads the file `GimA22i` as Δ instead of computing it from data, and `OPTION omit_ainv` suppresses the creation of $\mathbf{A}^{-1}$. In the end, $\mathbf{H}^{-1}$ has only $\mathbf{G}^{-1}$.

Run BLUPF90 with `blup.par`, and obtain the solution file. The solutions look similar to the previous analysis for some animals (especially young one) but very different in the other animals (older one). This is because, in this study, $\mathbf{G}$ was blended with $\mathbf{A}_{22}$ and it can capture the residual polygenic effect.

Remarks

Dummy pedigree to remove A-inverse The option `omit_ainv` is effective only in BLUPF90. If this option doesn't work (or you are not sure the program supports it), use the alternative approach to remove $\mathbf{A}^{-1}$. First, create a *dummy* pedigree file including only 0 in all 3 (or 4) columns. The pedigree file should have the same (or more) number of lines as the number of genotyped animals. The *awk* program easily creetes the dummy pedigree. The following command assumes the renumbered pedigree is `renadd02.ped` and writes 4 columns in case of unknown parent groups.

```
awk '{print 0,0,0,0}' renadd02.ped > dummy.ped
```

Then, modify `blup.par` to have `dummy.ped` instead of `renadd02.ped`. With this trick, $\mathbf{A}^{-1}$ becomes $\mathbf{0}$ so no contribution to the equations.

Pure GBLUP without pedigree effect Even with the pedigree file, the user can run GBLUP without any contribution from pedigree information (i.e. $\mathbf{A}_{22}$). In this case, the solutions should be identical to one obtained in the previous two methods.

To do this, the user can put the following options in the renum-instruction file or `renf90.par` for PREGSF90.

```
OPTION AlphaBeta 0.95 0.00
OPTION GammaDelta 0.05 0.00
OPTION tunedG 0
OPTION TauOmega 1.00 0.00
```

The first two options will be used in blending: $\mathbf{G} \leftarrow \alpha\mathbf{G} + \beta\mathbf{A}_{22} + \gamma\mathbf{I} + \delta\mathbf{1}\mathbf{1}'$. The above option specifies $\alpha = 0.95$, $\beta = 0$, $\gamma = 0.05$, and $\delta = 0$ so that the blending is $\mathbf{G} \leftarrow 0.95\mathbf{G} + 0.05\mathbf{I}$. The option `tunedG 0` prohibits the program to scale $\mathbf{G}$ to $\mathbf{A}_{22}$. These three options are equivalent to the approach that we introduced as the first method in this chapter. The last option removes $\mathbf{A}_{22}^{-1}$ from $\mathbf{H}^{-1}$.

Finally, the user should create a separate parameter file to use `GimA22i` and to omit $\mathbf{A}^{-1}$ in BLUPF90 using `omit_ainv` or the dummy pedigree file.

8.5 GWAS using the ssGBLUP framework

The single-step GBLUP methodology has been extended to be capable of genome-wide association study (GWAS). This is an iterative procedure with 2 steps: 1) prediction of GEBV with ssGBLUP and 2) prediction of SNP marker effects based on the GEBV. Here we call this method ssGWAS. The detailed algorithm was described by Wang et al. (2012).

8.5.1 Algorithm

To describe the algorithm of ssGWAS, we use the notation made by Wang et al. (2012). The notation is totally different from ones used in the previous chapters. Don't be confused with different symbols.

We assume $\mathbf{a}$ is a vector for breeding values for genotyped animals and $\mathbf{u}$ is a vector for SNP marker effects. One is related to another with the following equation.

$$\mathbf{a} = \mathbf{Z}\mathbf{u}$$

The variances of them can be written as

$$\text{var}(\mathbf{a}) = \mathbf{G}\sigma_a^2 \quad \text{and} \quad \text{var}(\mathbf{Zu}) = \mathbf{ZDZ}'\sigma_u^2$$

where **D** is a diagonal matrix of weights accounting for variances of SNP markers. This matrix is usually assumed to be **I** in the regular ssGBLUP. Above 2 variances are identical, so we can derive

$$\text{var}(\mathbf{a}) = \text{var}(\mathbf{Zu}) = \mathbf{G}\sigma_a^2 = \mathbf{ZDZ}'\sigma_u^2$$

and this equation implies

$$\mathbf{G} = \mathbf{ZDZ}'\frac{\sigma_u^2}{\sigma_a^2} = \mathbf{ZDZ}'\lambda$$

where $\lambda = \sigma_u^2/\sigma_a^2$. According to the definition of **G**, the variance ratio can be

$$\lambda = \frac{\sigma_u^2}{\sigma_a^2} = \frac{1}{2\sum_j p_j(1-p_j)}.$$

The prediction of breeding values $\hat{\mathbf{a}}$ is calculated with ssGBLUP. The prediction of SNP effects $\hat{\mathbf{u}}$ is also calculated with the best prediction.

$$\begin{aligned}\hat{\mathbf{u}} &= \text{cov}(\mathbf{u}, \mathbf{a}')\left[\text{var}(\mathbf{a})\right]^{-1}\hat{\mathbf{a}} \\ &= \lambda\mathbf{DZ}'\mathbf{G}^{-1}\hat{\mathbf{a}} \\ &= \mathbf{DZ}'(\mathbf{ZDZ}')^{-1}\hat{\mathbf{a}}\end{aligned}$$

With the prediction of a SNP effect, we can give weights to markers based on SNP solutions. Current default weight is as in Wang et al. (2012):

$$w_i = 2p_iq_i\hat{u}_i^2$$

This may be used as a weight in **D** after its scaling (see Wang et al., 2012). We restart the whole process with this new **D**. We can repeat the procedures until reasonable results are obtained. However, this weighting scheme is not the most accurate and leads to unstable iterates. A well-tested method for weights is VanRaden (2008) nonlinearA, which is likely to became the default:

$$w_i = 1.125^{\frac{|\hat{u}_i|}{\sqrt{var\hat{u}}}-2}$$

The SNP effects can be used as the indirect prediction of GEBV based on $\mathbf{a} = \mathbf{Zu}$. In the end of this section, we will demonstrate this approach.

8.5.2 Programs

We will use several programs to perform ssGWAS.

1. RENUMF90: general preparation of data set.
2. PREGSF90: quality control and creation of cleaned genotypes.
3. BLUPF90: prediction of a with weighted G.
4. POSTGSF90: prediction of u and new weight D.

The first 2 programs are needed only once in the very first step. If your data is well prepared, you just run BLUPF90 and POSTGSF90 repeatedly. In this case, it is easier to understand for a user to directly use BLUPF90 to create **G** and calculate the solutions of GEBV in the same program.

8.5.3 Numerical example

Files

We will use the similar files used in the summer course at the UGA in 2012.

- `http://nce.ads.uga.edu/wiki/lib/exe/fetch.php?media=lab_gwas.zip` - lab gaws.zip

We made some corrections on the files. Download the zip file and extract it to your computer. It has several files like

- `phenotypes.txt` = observation data file
- `pedigree` = pedigree file
- `marker.geno.clean` = marker file
- `chemap` = chromosome map file
- `w` = weight file
- `renum.par` = instruction file for RENUMF90

We will start with RENUMF90 to prepare the data set.

8.5.4 Preparation

First, we run RENUMF90 to generate required files. After renumbering, we can see `renf90.par`, a template parameter file. We make 2 copies of this template; one is for BLUPF90 and another one is for POSTGSF90. Here, we put `param_ssgwas1a.txt` to the parameter file for BLUPF90 and `param_ssgwas1b.txt` for POSTGSF90.

BLUPF90 will generate **G** with weights, supplied as a weight file (see previous section). The package already has a weight file (named `w`), the option can read the file. Following is the whole parameter file for BLUPF90.

Listing 8.16: param_ssgwas1a.txt

```
#
# This is for BLUPF90
#
DATAFILE
renf90.dat
NUMBER_OF_TRAITS
         1
NUMBER_OF_EFFECTS
         2
OBSERVATION(S)
 1
WEIGHT(S)

EFFECTS:
 2 1 cross
 3 15800 cross
RANDOM_RESIDUAL VALUES
0.50000
RANDOM_GROUP
    2
RANDOM_TYPE
add_animal
```

```
FILE
renadd02.ped
(CO)VARIANCES
0.50000
OPTION SNP_file marker.geno.clean
OPTION saveGInverse
OPTION weightedG w
```

POSTGSF90 also needs to read $\mathbf{G}^{-1}$ generated with BLUPF90. It also needs a chromosome map file because the program will predict each SNP effect. The following is the complete parameter file for POSTGSF90.

Listing 8.17: param_ssgwas1b.txt

```
#
# This is for POSTGSF90
#
DATAFILE
renf90.dat
NUMBER_OF_TRAITS
          1
NUMBER_OF_EFFECTS
          2
OBSERVATION(S)
 1
WEIGHT(S)

EFFECTS:
2 1 cross
3 15800 cross
RANDOM_RESIDUAL VALUES
0.50000
RANDOM_GROUP
    2
RANDOM_TYPE
add_animal
FILE
renadd02.ped
(CO)VARIANCES
0.50000
OPTION SNP_file marker.geno.clean
OPTION readGInverse
OPTION weightedG w
OPTION chrinfo chrmap.txt
```

Iterative run and Manhattan plot

First we run BLUPF90 with the parameter file `param_ssgwas1a.txt` following by running POSTGSF90 with the different parameter file `param_ssgwas1b.txt`. POSTGSF90 creates several files.

- `snp_sol` = Solution for SNP markers (i.e. $\mathbf{u}$). It has 7 columns.
    1. Trait

2. Effect
3. SNP number
4. Chromosome
5. Position
6. SNP solution ($\hat{u}_i$) . From this column you can compute weights yourself.
7. Weight (d_i) (actual default for the weight may change; check the official documentation)

- `chrsnp` = Used in the Manhattan plot. It has 6 columns.
  1. Trait
  2. Effect
  3. % of variance explained by adjacent SNPs
  4. SNP number
  5. Chromosome
  6. Position

- `dgv` = Direct genomic value for each animal. It has 5 columns.
  1. Trait
  2. Effect
  3. Animal ID
  4. DGV (direct genomic value) = $-\sum_{j \neq i} g^{ij}\hat{u}_j \Big/ g^{ii}$; see Lourenco et al. (2015).
  5. PP (pedigree prediction) = $-\sum_{j \neq i} a_{22}^{ij}\hat{u}_j \Big/ a_{22}^{ii}$; see Lourenco et al. (2015).

- `snp_pred` = Used in the program to predict GEBV based on the SNP effects. A user doesn't have to know the exact format.

- `Sft1e2.gnuplot` and `Sft1e2.R` = Scripts to draw Manhattan plots with Gnuplot and R. Each script reads `chrsnp`.

What we need in the next round is the 7th column in `snp_sol`. This column should be extracted and saved in the file `w` (this file is updated with new values). Once `w` is updated, rerun BLUPF90 and POSTGSF90. Repeat the process enough times. In this example, we just repeat 3 times.

This iterative process can be automated using a script. In Linux and MacOS X, bash shell script is common to use. In Windows, traditional batch file is useful but some free software are needed to extract 7th column of the file. We don't explain such techniques here. You can see bash scripts in the same directory as the data files. The scripts will tell you how to run the programs repeatedly.

After 3 rounds, we can draw the Manhattan plot with the script. Using R, you will see the following figure. According to this graph, there is a large SNP effect on the chromosome 1.

Indirect prediction of GEBV based on SNP effects

Once you find the SNP effects, another program PREDF90 can calculate indirect GEBV from the SNP effects based on the equation $\hat{\mathbf{a}} = \mathbf{Z}\hat{\mathbf{u}}$. This program reads the marker file and `snp_pred`, generated with POSTGSF90. Run the program, and see the following message.

```
name of genotype file?
```

Different from other software, it needs the name of marker file. Type the name of genotyed file in here. The program immediately calculates the indirect GEBV for genotyped animals.

This program creates a file SNP predictions. It has 3 columns.

1. Animal ID
2. Call rate
3. Indirect prediction of GEBV (based on $\hat{\mathbf{a}} = \mathbf{Z}\hat{\mathbf{u}}$).

In other words, the prediction for an animal is the sum, according to its genotypes, of the estimated SNP effects, as estimated in the overall ssGBLUP analyses. This program is useful especially in a case where there are many young genotyped animals but many of them shouldn't be included in ssGBLUP analysis because these are *interim* analysis between two full genomic evaluations or for other reasons.

References

- Chen, C.Y., Misztal, I., Aguilar, I., Legarra, A. and Muir, W.M., 2011. Effect of different genomic relationship matrices on accuracy and scale. Journal of Animal Science, 89(9), pp.2673-2679.
- Christensen, OF, P. Madsen, B. Nielsen, T. Ostersen, and G. Su. 2012. Single-Step Methods for Genomic Evaluation in Pigs. Animal 6: 1565–71.
- Foulley, J.L., Gianola, D. and Thompson, R., 1983. Prediction of genetic merit from data on binary and quantitative variates with an application to calving difficulty, birth weight and pelvic opening. Genetics Selection Evolution, 15(3), p.1.
- Hoeschele, I., Tier, B. and Graser, H.U., 1995. Multiple-trait genetic evaluation for one polychotomous trait and several continuous traits with missing data and unequal models. Journal of animal science, 73(6), pp.1609-1627.
- Janss, L.L.G. and Foulley, J.L., 1993. Bivariate analysis for one continuous and one threshold dichotomous trait with unequal design matrices and an application to birth weight and calving difficulty. Livestock Production Science, 33(3), pp.183-198.
- Legarra, A., Bertrand, J.K., Strabel, T., Sapp, R.L., Sanchez, J.P. and Misztal, I., 2007. Multi-breed genetic evaluation in a Gelbvieh population. Journal of Animal Breeding and Genetics, 124(5), pp.286-295.
- Lourenco, D.A.L., Tsuruta, S., Fragomeni, B.O., Masuda, Y., Aguilar, I., Legarra, A., Bertrand, J.K., Amen, T.S., Wang, L., Moser, D.W. and Misztal, I., 2015. Genetic evaluation using single-step genomic best linear unbiased predictor in American Angus. Journal of Animal Science, 93(6), pp.2653-2662.
- Masuda, Y., Aguilar, I., Tsuruta, S. and Misztal, I., 2015. Technical note: Acceleration of sparse operations for average-information REML analyses with supernodal methods and sparse-storage refinements. Journal of Animal Science, 93(10), pp.4670-4674.
- Masuda, Y., Baba, T. and Suzuki, M., 2014. Application of supernodal sparse factorization and inversion to the estimation of (co) variance components by residual maximum likelihood. Journal of Animal Breeding and Genetics, 131(3), pp.227-236.
- Misztal, I., Tsuruta, S., Aguilar, I., Legarra, A., VanRaden, P.M. and Lawlor, T.J., 2013. Methods to approximate reliabilities in single-step genomic evaluation. Journal of dairy science, 96(1), pp.647-654.
- Misztal, I. and Wiggans, G.R., 1988. Approximation of prediction error variance in large-scale animal models. Journal of Dairy Science, 71, pp.27-32.
- Misztal, I. 2008. Reliable computing in estimation of variance components. Journal of Animal Breeding and Genetics, 125(6), pp.363-370.
- Mrode, R. A. 2014. *Linear Models for the Prediction of Animal Breeding Values*. Third Edition. CAB International, Wallingford, Oxon, UK.
- Powell, J.E., Visscher, P.M. and Goddard, M.E., 2010. Reconciling the analysis of IBD and IBS in complex trait studies. Nature Reviews Genetics, 11(11), pp.800-805.
- Sanchez, J.P., Misztal, I., Aguilar, I. and Bertrand, J.K., 2008. Genetic evaluation of growth in a multibreed beef cattle population using random regression-linear spline models. Journal of animal science, 86(2), pp.267-277.
- Sorensen, D.A., Andersen, S., Gianola, D. and Korsgaard, I., 1995. Bayesian inference in threshold models using Gibbs sampling. Genetics Selection Evolution, 27(3), p.1.
- Strabel, T., Misztal, I. and Bertrand, J.K., 2001. Approximation of reliabilities for multiple-trait model with maternal effects. Journal of Animal Science, 79(4), pp.833-839.

- Stranden, I. and Lidauer, M., 1999. Solving large mixed linear models using preconditioned conjugate gradient iteration. Journal of dairy science, 82(12), pp.2779-2787.
- Tsuruta, S., Misztal, I. and Stranden, I., 2001. Use of the preconditioned conjugate gradient algorithm as a generic solver for mixed-model equations in animal breeding applications. Journal of Animal Science, 79(5), pp.1166-1172.
- Vitezica, Z.G., Aguilar, I., Misztal, I. and Legarra, A., 2011. Bias in genomic predictions for populations under selection. Genetics Research, 93(05), pp.357-366.
- Wang, H., Misztal, I., Aguilar, I., Legarra, A. and Muir, W.M., 2012. Genome-wide association mapping including phenotypes from relatives without genotypes. Genetics Research, 94(02), pp.73-83.
- Wang, C.S., Quaas, R.L. and Pollak, E.J., 1997. Bayesian analysis of calving ease scores and birth weights. Genetics Selection Evolution, 29(2), p.1.

RENUMF90 Cheat Sheet

The RENUMF90 parameter file

Pairs of *keyword-value* must appear in the following order. The keyword should be capital. More than one field are allowed for a multiple-trait case. Characters led by # will be ignored as comments.

Optional COMBINED keyword

```
COMBINE
  n a b ..    Fields a, b, ... combied into a single field n.
```

Required keywords

```
DATAFILE                  Data file with observations and effects.
  filename
TRAITS                    Field(s) for observations.
  f1 ..                   Multiple values if multiple-trait.
FIELDS_PASSED TO OUTPUT   Field(s) passed to output data.
  f1 ..                   (Empty value if not needed.)
WEIGHT(S)                 Field for weight.
  field                   (Empty value if not needed.)
RESIDUAL_VARIANCE         Full residual covariance matrix.
  R                       (Scalar if single-trait)
EFFECT                    Effect definition.
  f1 .. type form         Repeat the keyword-value if needed.
```

A EFFECT block has the following values.

```
f1 ..   Field(s) with class code or covariate.
        Multiple values if multiple-trait.
        0 if not needed for specific traits.
type    Type of effect.
        cross for cross-classified effect.
        cov for covariate.
form    Only for cross-classified effect.
        alpha for alphanumeric fields.
        numer for numeric fields.
```

Optional NESTED keyword

```
NESTED          Nested regression for the immediate effect.
  f1 .. form    The same number of fields as EFFECT.
                form is alpha or numer.
```

Optional RANDOM and related keywords

```
RANDOM          Make the immediate effect random.
  type          animal for A; diag for I.
OPTIONAL        Add optional random effects.
  type ..       pe for permanent environmental effect;
                mat for maternal genetic effect;
                mpe for maternal PE.
FILE            Pedigree file.
  filename
FILE_POS        Field definition of pedigree file.
  a s d ad yob g  a,s,d for animal, sire and dam ID;
                ad for alternate dam; 0 if not needed;
                yob for birth year; 0 if not needed;
                g for unknown parent groups (optional)
                (Default = 1 2 3 0 0)
SNP_FILE        SNP marker file.
  filename
PED_DEPTH       Depth of pedigree search.
  n             (Default = 3)
GEN_INT         Generation interval.
  min avg max   minimum, average and maximum interval.
                Needed only if birth year is available.
REC_SEX         Check sex-limited traits.
  x             1 for male; 2 for female.
UPG_TYPE        Assign unknown parent groups.
  type          yob = birth year;
                in_pedigrees = negative code in pedigree;
                group based = group code in extra field;
                group_unisex as above but with "unisex"
INBREEDING      Consider inbreeding in A^-1.
  type          pedigree computing with pedigree data.
                file filename reading values from file.
                (Default = no inbreeding considered)
RANDOM_REGRESSION  Define this effect as random regression.
  type          data here.
RR_POSITION     Field(s) of covariates.
  f1 ..
(CO)VARIANCES   Full covariance matrix.
  G             (Default: 1 on diag. and 0.1 on off-diag.)
(CO)VARIANCES_PE  Full covariance matrix for PE.
  G             (Default: 1 on diag. and 0.1 on off-diag.)
(CO)VARIANCES_PE  Full covariance matrix for maternal PE.
  G             (Default: 1 on diag. and 0.1 on off-diag.)
```

For 2-trait maternal model, the genetic covariance matrix (CO)VARIANCES should be 4×4 as follows.

		Direct		Maternal	
		Trait 1	Trait 2	Trait 1	Trait 2
Direct	Trait 1				
	Trait 2				
Maternal	Trait 1				
	Trait 2				

For 2-trait random regressions, the genetic covariance matrix (CO)VARIANCES should be 4×4 as follows.

		RR 1		RR 2	
		Trait 1	Trait 2	Trait 1	Trait 2
RR 1	Trait 1				
	Trait 2				
RR 2	Trait 1				
	Trait 2				

Additional OPTION lines

You can write additional options at the end of the parameter file. An option has the keyword OPTION and its values on the same line. Any numbers of option lines are allowed. Only the following options will be taken with RENUMF90; the other options will be just passed to the output file.

```
alpha_size n              The width of alphanumeric field.
                          (Default=20).
max_string_readline n     The number of characters in a line.
                          (Default=800).
max_string_readline n     The number of fields.
                          (Default=99).
```

Guideline for file preparation

- Text file with tidy data like a table (each row for reacord and each field for factor).
- White spaces as the only separators; no tabs are allowed.
- Alphanumeric characters and symbols for group code and animal ID.
- Common numerics (integer, floating point, and exponential expressions) for observations, covariates, and weights.
- The default missing code is 0. You can change it using OPTION missing n with an integer n; This works only for data file (not for pedigree file in which a single 0 is the missing code).

Output files

The RENUMF90 program generates "renumbered" files that can be used with BLUPF90 and related programs.

```
renf90.par      New parameter file for BLUPF90 programs.
renf90.dat      New data file.
renadd??.ped    New pedigree file.
                ?? replaced with numbers.
****_XrefID     Cross-reference ID file (optional).
                **** replaced with the SNP file name.
renf90.inb      Inbreeding coefficients (optional).
renf90.tables   Code replacement table.
```

The new pedigree file has 10 fields with additional information.

1. New animal ID (renumbered from 1)
2. New parent 1 (sire) or unknown parent group ID
3. New parent 2 (dam) or unknown parent group ID
4. Without inbreeding, 3 minus number of known parents; With inbreeding a 4-digit code (see below).
5. Known or estimated year of birth (0 if not provided)
6. The number of known parents (parents might be eliminated if not contributing; if animal has genotype 10+number of know parents
7. The number of records
8. The number of progeny (before elimination due to other effects) as parent 1
9. The number of progeny (before elimination due to other effects) as parent 2
10. Original animal id

The 4th field has the following four-digit code (*upg/inb code*) when inbreeding is included:

$$\frac{4000}{(1+m_s)(1-F_s)+(1+m_d)(1-F_d)}$$

where m_s (m_d) is 0 if sire (dam) is known or 1 if the parent is unknown, and F_s (F_d) is the inbreeding coefficient of sire (dam).

Based on the BLUPF90 manual
Yutaka Masuda (masuday@uga.edu)

BLUPF90 Cheat Sheet

The BLUPF90 parameter file

The parameter file can be commonly used with BLUPF90, AIREMLF90, GIBBSF90, and the other family programs. Pairs of *keyword-value* must appear in the following order. The keyword should be capital. Characters led by # will be ignored as comments.

Required keywords

`DATAFILE`	Data file with observations and effects.
`filename`	
`NUMBER_OF_TRAITS`	
`n`	a single integer.
`NUMBER_OF_EFFECTS`	
`n`	a single integer.
`OBSERVATION(S)`	Field(s) for observations.
`f1 ..`	Multiple fields if multiple-trait model.
`WEIGHT(S)`	Field for weight.
`field`	(Empty value if not needed.)
`EFFECTS:`	Effect definition (see below).
`f1 .. type c1 ..`	Repeat the description if needed.
`RANDOM_RESIDUAL`	Full residual covariance matrix.
`R`	(Scalar if single-trait)

The `EFFECTS:` block is followed by model-description lines; each row describes 1 effect so that the number of rows is the same to the number of effects specified at `NUMBER_OF_EFFECTS`.

`f1 ..`	Field(s) with class code or covariate. Multiple values if multiple-trait. `0` if not needed for specific traits.
`type`	Type of effect. `cross` for cross-classified effect. `cov` for covariate.
`c1 ..`	The list is optional. Field(s) with class code for nested regression. Multiple values if multiple-trait. `0` if not needed for specific traits.

The position of effect in this block is called "effect number". It will be used to specify a random effect in the next section.

Optional `RANDOM` and related keywords

`RANDOM` is followed by 3 other keywords and it makes a section to define a random effect. You can repeat the `RANDOM` section if you have several random effects. This section can define a correlated random effect involving multiple effects such as a direct-maternal genetic effect and random-regressions.

`RANDOM_GROUP`	Define a random effect group.
`e1 ..`	List of effect numbers defined above. The effect numbers must be consecutive.
`RANDOM_TYPE`	Type of random effect.
`type`	See below
`FILE`	Pedigree (or similar) file.
`filename`	Empty line if not needed.
`(CO)VARIANCES`	Full covariance matrix.
`G`	

The `type` defines the covariance structure and pedigree relationships.

- `diagonal`: Identity (**I**).
- `add_sire`: Numerator relationship matrix for sire and MGS.
- `add_animal`: Numerator relationship matrix without inbreeding.
- `add_an_upg`: As above with unknown parent groups.
- `add_an_upginb`: As above but with inbreeding.
- `par_domin`: Parental dominance.
- `user_file`: User-supplied inverse matrix.
- `user_file_inv`: User-supplied non-inverse matrix (inverted by programs).

The `filename` will be needed for all `type` except `diagonal`. In a pedigree file, an animal's ID should be a positive integer; an unknown-parent group is an integer greater than the largest ID of real animals; a missing parent is `0`. The file format is shown below.

- For `add_sire`: 1) animal, 2) sire, and 3) MGS.
- For `add_animal`: 1) animal, 2) sire, and 3) dam.
- For `add_an_upg`: 1) animal, 2) sire or UPG, 3) dam or UPG, and 4) 3 minus the number of known parents.
- For `add_an_upginb`: 1) animal, 2) sire or UPG, 3) dam or UPG, and 4) four-digit upg/inb code; see RENUMF90.
- For `par_domin`: Generated with `rendomn`; See the manual.
- For `user_file` and `user_file_inv`: 1) row, 2) column, and 3) value; Half-stored.

In a covariance matrix, the trait is nested within effect. See the following case for 2 traits and 2 correlated effects.

		Eff 1		Eff 2	
		Tr 1	Tr 2	Tr 1	Tr 2
Eff 1	Tr 1				
	Tr 2				
Eff 2	Tr 1				
	Tr 2				

Options

You can write additional options at the end of the parameter file. An option has the keyword `OPTION` and its values on the same line. Any numbers of option lines are allowed. Unsupported options will be simply ignored (but preGSf90 give you an error).

Genomic options

For genomic analyses, see a separate cheat sheet.

Common options for BLUPF90/AIREMLF90

`missing n`	Treat an integer `n` as a missing observation; default = `0`.
`conv_crit c`	Convergence criterion for iteratiions; default $= 10^{-12}$.
`maxrounds n`	Maximum iterations; default = 5000.
`sol se`	Calculate SE of each solution as the inverse of LHS.
`use_yams`	Faster computations with the YAMS package; should be combined with `solv_method FSPAK` for BLUPF90.

BLUPF90

`solv_method m`	Solving method: `m` = `FSPAK` for direct inversion and `pcg` for PCG (default)

AIREMLF90

`EM-REML n`	EM iterations for the first `n` rounds.
`hetres_pos f1 ..`	Fields for covariates in a function of heterogeneous residual variance; should be multiple of the number of traits.
`hetres_pol f1 ..`	Initial regression coeffeicients for the heterogeneous-residual-variance function.

`se_covar_function label function`
Calculate SE for a function of variance components by sampling; shown with arbitrary `label`; covariance `G_i_j_k_l` for random effects i and j, and traits k and l; residual covariance `R_k_l` for trait k and l.

Common options for GIBBSxF90/THRGIBBS1F90

`fixed_var mean e1 ..`	Compute poeterior mean/SD of location parameters of specified effects without updating covariances.
`fixed_var all e1 ..`	As above but store all samples.
`solution mean e1 ..`	Similar to `fixed_var mean` but updating variance components.
`solution all e1 ..`	Similar to `fixed_var all` but updating variance components.
`cont n`	Continue sampling from the previous run in the round `n`.
`seed m n`	Seeds of random number generators.

THRGIBBS1F90

`cat t1 ..`	The number of categories in each trait; `0` for contineoues traits.
`censored t1 ..`	Censoring in each trait; `1` if censored and `0` if not.
`threshold v1 ..`	Set fixed thresholds; default = 0.
`residual 1`	Set residual variance to 1.

Based on the BLUPF90 manual
Yutaka Masuda (masuday@uga.edu)

Genomic Options Cheat Sheet

Flowchart

All application programs can check the genomic data and calculate a genomic relationship matrix ($\mathbf{G}$), a subset of a pedigree matrix ($\mathbf{A}_{22}$), and those inverse matrices, followed by the statistical computations (e.g. solving equations in BLUPF90). PREGSF90 performs the genomic set-up only. The genomic data will be processed as follows.

1. Check the cross-reference ID (XrefID) file.
2. Read the pedigree and store it in memory.
3. Calculate $\mathbf{A}_{22}$.
4. Read and store the SNP markers in memory.
5. Check the quality of markers and animals and remove some of them if unqualified (*quality control*).
6. Compute $\mathbf{Z}$ as the adjusted marker genotypes with allele frequency.
7. Calculate $\mathbf{G} = \mathbf{ZZ}'/k$ with a coefficient k.
8. Update $\mathbf{G}$ as $\mathbf{G} \leftarrow \alpha\mathbf{G} + \beta\mathbf{A}_{22} + \gamma\mathbf{I} + \delta\mathbf{11}'$ (*blending*).
9. Update $\mathbf{G}$ to scale it to $\mathbf{A}_{22}$ (*tuning*).
10. Calculate statistics on $\mathbf{G}$ and $\mathbf{A}_{22}$.
11. Calculate $\omega\mathbf{A}_{22}^{-1}$ by updating $\mathbf{A}_{22}$.
12. Calculate $\tau\mathbf{G}^{-1}$ by updating $\mathbf{G}$.
13. Calculate the difference $\Delta = \tau\mathbf{G}^{-1} - \omega\mathbf{A}_{22}^{-1}$.
14. Save Δ in a binary file (`GimA22i`).

Files

Input files

- SNP file: 2 fields per row: an animal ID and its genotypes. The ID must have a fixed width with the tailing spaces. The genotypes can be integers (coded as `0`, `1`, `2`, and `5` as missing) or gene content (real numbers with fixed width). No spaces are allowed between markers.
- XrefID file: The file is usually generated with RENUMF90. This file has a table relating the genotyped animals with the renumbered pedigree.
- Pedigree file: It is the same as used in the standard animal-model analysis.
- Map file (optional): The file has at least 3 fields per row: 1) the marker number, 2) the chromosome number, 3) the physical location, and optional 4) the description of this markers.

Default output files

- `freqdata.count`: Minor allel frequency calculated from the original SNP file.
- `freqdata.count.after.clean`: Minor allel frequency calculated from the data after the quality control.
- `Gen_call_rate`: Call rate for genotyped animals.
- `Gen_conflicts`: Report of parentage checks.
- `sum2pq`: $2\sum_i p_i q_i$; k as above in $\mathbf{G}$.
- `GimA22i`: A binary fole for $\Delta = \tau\mathbf{G}^{-1} - \omega\mathbf{A}_{22}^{-1}$.

The required options for genomics

`SNP_file snpfile xrefid`
Invoke genomic module using the SNP file `snpfile`; By default, a cross-reference-ID (XrefID) file is assumed to be `snpfile` + `_XrefID`. You can optionally supply the XrefID file as the second argument. This option accompanies many other options (shown below).

Genomic options

The following lists are not complete. See the official manual for additional options.

User-supplied files

`chrinfo file`	Supply a map file.
`FreqFile file`	Supply the pre-calculated allele frequency; the same format as `freq.count`

Quality control

`no_quality_control`	Turn off the quality-control; some checks will be still performed but any unqualified data will not be removed.
`saveCleanSNPs`	Save "clean" SNP data, in which unqualified markers and animals have been removed, to files.
`minfreq x`	Remove a marker if the minor allele frequency is $< x$. (default = 0.05)
`callrate x`	Remove a marker if the call rate is $< x$. (default = 0.90)
`callrateAnim x`	Remove an animal if the call rate is $< x$. (default = 0.90)
`monomorphic x`	Remove a monomorphic marker if `x` is 1. (default = 1)
`hwe x`	Perfrom the Hardy-Weinberg test with the criterion `x` (default = not performed).
`high_correlation x y`	Check a high-correlated pair of markers if the difference in the allele frequency between the markers is larger than `x`; show warings if the correlation is higher than `y`. Default = `x=0.025` and `y=0.995`.
`verify_parentage x`	Parentage checks; `0` for skipping all checks; `1` for just checks; `2` for checking animals and removing conflicted markers and animals (default = 2).
`outparent_progeny`	Create a precise report of parentage checks.
`excludeCHR n1..`	Exclude markers on specific chromosomes from the final output; the map file in seeded.
`sex_chr n`	Exclude markers on the sex chromosomes temporarily from parantage and Hardy-Weinberg checks; the map file in seeded.

Blending and tuning

`AlphaBeta a b`	Specify α as `a` and β as `b` in blending (default: $\alpha = 0.95$ and $\beta = 0.05$).
`GammaDelta g d`	Specify γ as `g` and δ as `d` in blending (default: $\gamma = 0$ and $\delta = 0$).
`TauOmega t o`	Specify τ as `t` and ω as `o` for the inverse matrices (default: $\tau = 1$ and $\omega = 1$).

Saving matrices

`saveAscii`	All files will be saved as the text file; Without this option, the files will be saved in a binary format.
`saveG`	Save the final $\mathbf{G}$ in the file `G`.
`saveG all`	Save the all intermediate $\mathbf{G}$'s in several files.
`saveA22`	Save the final $\mathbf{A}_{22}$ in the file `A22`.
`saveGInverse`	Save the final $\tau\mathbf{G}^{-1}$ in the file `Gi`.
`saveA22Inverse`	Save the final $\omega\mathbf{A}_{22}^{-1}$ in the file `A22i`.
`saveGOrig`	Save the final $\omega\mathbf{G}$ with the original animal ID; always saved in the ASCII format regardless of `saveAscii`; the pedigree file generated with RENUMF90.
`saveA22Orig`	As above but for the final $\mathbf{A}_{22}$.
`saveHinv`	Save $\mathbf{H}^{-1}$ in a text file. Only accepted by PREGSF90.
`saveHinvOrig`	Save $\mathbf{H}^{-1}$ in a text file with the original animal ID. Only accepted by PREGSF90.

Reading matrices

With one of the following options, the program will skip all required operations to form the relationship matrix related to the specified option. The file used here should be a binary format (not ASCII).

`readGimA22i <file>`	Read $\Delta = \tau\mathbf{G}^{-1} - \omega\mathbf{A}_{22}^{-1}$ (default file = `GimA22i`).
`readG <file>`	Read $\mathbf{G}$ (default = `G`).
`readA22 <file>`	Read $\mathbf{A}_{22}$ (default = `A22`).
`readGInverse <file>`	Read $\mathbf{G}^{-1}$ (default = `Gi`).
`readA22Inverse <file>`	Read $\mathbf{A}_{22}^{-1}$ (default = `A22i`).

The last 2 options assume the file contains $\mathbf{G}^{-1}$ or $\mathbf{A}_{22}^{-1}$, NOT $\tau\mathbf{G}^{-1}$ or $\omega\mathbf{A}_{22}^{-1}$. If you read these matrices and also specifies `TauOmega`, the program will apply τ and ω to the matrices just read from the files. This is problematic when the matrices have been already scaled with τ and ω before being saved.

Skip creating matrices

`createGimA22i 0`	Omit $\Delta = \tau\mathbf{G}^{-1} - \omega\mathbf{A}_{22}^{-1}$.
`createG 0`	Omit $\mathbf{G}$.
`createA22 0`	Omit $\mathbf{A}_{22}$.
`createGInverse 0`	Omit $\mathbf{G}^{-1}$.
`createA22Inverse 0`	Omit $\mathbf{A}_{22}^{-1}$.

Based on the BLUPF90 manual
Yutaka Masuda (masuday@uga.edu)