DKBLUPF90 User Manual

High-Performance Fortran Library and SNP Quality Control Program

DH Lee

February 2026

Contents

Contents

1 1. Introduction

1.1 1.1. What is DKBLUPF90?

DKBLUPF90 is a high-performance Fortran library and program collection for genomic SNP data quality control and preprocessing. The main program **ReadFR** reads Illumina FinalReport files and performs SNP quality control, converting data into formats suitable for GBLUP (Genomic Best Linear Unbiased Prediction) analysis.

1.2 1.2. Key Capabilities

- **O(1) time complexity** hash table-based fast search
- **Large-scale data processing**: Handles hundreds of thousands of SNPs and thousands of animals simultaneously

- **Flexible QC options**: GC Score, Call Rate, Cluster Separation, R-Intensity, GT Score
- **Multiple chip version support**: Compatible with various SNP chip formats (V1, V2, etc.)
- **Memory efficient**: Dynamic memory management with optimized data structures
- **Flexible configuration**: Parameter file-based settings
- **Comprehensive reporting**: Detailed QC reports and statistics

1.3 1.3. Project Structure

```
DKBLUPF90/
  source/             # Library components
     M_Kinds.f90         # Type definitions
     M_Variables.f90     # Common data types
     M_HashTable.f90     # Generic hash table
     M_PEDHashTable.f90  # Pedigree hash table
     M_ReadFile.f90      # File I/O
     M_StrEdit.f90       # String processing
     M_ReadPar.f90       # Parameter parsing
     M_Stamp.f90         # Timestamps
     Qsort4.f90          # Sorting
  ReadFR/             # Main program
     ReadFR.f90          # ReadFR program
     check/              # Test data
  install.sh          # Installation script
  README.md           # Quick reference
  USER_MANUAL.pdf     # This manual
```

2 2. Installation

2.1 2.1. System Requirements

2.1.1 Required

- **OS**: Linux (Ubuntu, CentOS, RHEL, Fedora, etc.)
- **Compiler**: gfortran 4.8 or higher
- **Build Tools**: GNU make, ar (from binutils)
- **Memory**: Minimum 4GB RAM (8GB+ for large datasets)
- **Disk Space**: ~1GB free space

2.1.2 Optional

- **PDF Generation**: pandoc, texlive-xetex
- **Documentation**: wkhtmltopdf (for advanced PDF features)

2.2 2.2. Dependency Installation

2.2.1 Ubuntu/Debian

```
sudo apt-get update
sudo apt-get install gfortran make binutils
sudo apt-get install pandoc texlive-xetex  # Optional
```

2.2.2 CentOS/RHEL

```
sudo yum install gcc-gfortran make binutils
sudo yum install pandoc texlive-xetex  # Optional
```

2.2.3 Verify Installation

```
gfortran --version
make --version
ar --version
```

2.3 2.3. Installation Methods

2.3.1 Automatic Installation (Recommended)

System-wide installation (requires root):

```
cd /path/to/DKBLUPF90
sudo ./install.sh
```

User directory installation (no root required):

```
PREFIX=$HOME/.local ./install.sh
```

2.3.2 Manual Installation

```
# Navigate to project directory
cd /path/to/DKBLUPF90
```

```
# Build library only
make lib

# Build ReadFR program
make readfr

# Build test programs (optional)
make testprog

# Installation
sudo make install
```

2.3.3 Post-Installation Setup

If using user directory installation, add to PATH:

```
export PATH=$HOME/.local/bin:$PATH
export LD_LIBRARY_PATH=$HOME/.local/lib:$LD_LIBRARY_PATH
```

For permanent setup, add to ~/.bashrc:

```
echo 'export PATH=$HOME/.local/bin:$PATH' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$HOME/.local/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
source ~/.bashrc
```

2.4 2.4. Verification

Check installation:

```
which ReadFR
ls -lh /usr/local/lib/libdkblupf90.*
ls /usr/local/include/dkblupf90/
```

3 3. ReadFR Program Usage

3.1 3.1. Overview

ReadFR processes Illumina FinalReport files to perform SNP quality control and generate GENO files compatible with BLUPF90 and similar programs.

3.1.1 Main Functions

- Parse FinalReport files
- Perform animal-level QC (Call Rate)
- Perform SNP-level QC (GC Score, Call Rate, etc.)
- Validate SNP positions using MAP files
- Validate pedigree using PED files
- Generate GENO files with genotype data
- Generate comprehensive QC reports

3.2 3.2. Parameter File Configuration

ReadFR uses a simple text-based parameter file for configuration. Each parameter is specified on a new line following a keyword.

3.2.1 Basic Structure

```
COMMENT Description text
KEYWORD value
```

3.2.2 Complete Example

```
Pedigree data file
PEDFile
/path/to/pedigree_data.txt

FinalReport input file
SNPFile
/data/illumina_finalreport.txt
ANIMAL-ARN 2
SNP_Name 1
Chr 10
Position 11
Allele1-AB 13
Allele2-AB 14
GC_Score 27 0.65
R-Intensity 25 0.4 2.0
GT_Score 29 0.50
Cluster_Sep 30 0.30

SNP position reference file
MAPFile
/data/snp_map.txt
```

```
Output file prefix
OutputPrefix
my_analysis

QC Thresholds
AnimalCallRate 0.95
SNPCallRate 0.90
```

3.2.3 Parameter Keywords

3.2.3.1 PEDFile

Path to pedigree file containing animal information.

Format:

```
BREED   ID          ARN            SIRE    DAM     SEX  BDATE      LOC
Duroc   D001        21905009744    S001    D100    2    20220115   Farm1
```

3.2.3.2 SNPFile

FinalReport file path and column specifications.

Column indicators: - `ANIMAL-ARN`: ARN column number - `SNP_Name`: SNP identifier column - `Chr`: Chromosome column - `Position`: Physical position column - `Allele1-AB`: First allele (AB coding) - `Allele2-AB`: Second allele (AB coding) - `GC_Score`: GenCall Score column and threshold - `R-Intensity`: R (intensity) column and min/max range - `GT_Score`: Genotype Score column and threshold - `Cluster_Sep`: Cluster Separation column and threshold

3.2.3.3 MAPFile

SNP map file containing position information.

Format:

```
SNP_Name        Chr    Position
rs123456        1      1000000
rs789012        1      2000000
```

3.2.3.4 OutputPrefix

Prefix for output files. Creates auto-generated filenames with date and sequence numbering: - **Format**: `prefix_YYYYMMDD_SequenceNumber.geno` - **Examples**: - First run on Feb 13, 2026: `my_analysis_20260213_00.geno` - Second run same day: `my_analysis_20260213_01.geno` - Next day run: `my_analysis_20260214_00.geno`

The system automatically: 1. Extracts current date (YYYYMMDD format) 2. Generates sequence numbers (00-99) for multiple runs per day 3. Prevents file overwrites by checking existing files 4. Creates files with `.geno` extension

3.2.3.5 AnimalCallRate

Minimum call rate for animals. Default: 0.95

3.2.3.6 SNPCallRate

Minimum call rate for SNPs. Default: 0.90

3.3 3.3. QC Thresholds Explained

3.3.1 GC Score (GenCall Score)

- **Range**: 0.0 to 1.0

- **Meaning**: Illumina genotype call confidence
- **Recommended**: 0.65
- **Strict**: 0.70

3.3.2 R-Intensity

- **Range**: 0.0+
- **Meaning**: Total fluorescence signal strength
- **Optimal Range**: 0.4 - 2.0
- **Issues if too low**: Weak signal
- **Issues if too high**: Non-specific hybridization

3.3.3 GT Score

- **Range**: 0.0 to 1.0
- **Meaning**: Genotype accuracy score
- **Recommended**: 0.50
- **Strict**: 0.60

3.3.4 Cluster Separation

- **Range**: 0.0 to 1.0
- **Meaning**: AA/AB/BB cluster distinctness
- **Recommended**: 0.30
- **Strict**: 0.40

3.3.5 Call Rate Definitions

Animal Call Rate:

```
= (Total SNPs - Missing SNPs) / Total SNPs
```

SNP Call Rate:

```
= (Total Animals - Missing Animals) / Total Animals
```

3.4 3.4. Running ReadFR

3.4.1 Basic Execution

```
ReadFR parameter_file
```

3.4.2 Example with Output

```
cd /working/directory
ReadFR my_parameters.txt

# Monitor progress
tail -f my_analysis_QC_REPORT.txt
```

3.4.3 Output Files

After successful execution, the program generates auto-named files: - **my__analysis__20260213__00.geno** - QC-passed GENO data with date and sequence (0/1/2/5 coding) - **my__analysis__20260213__01.geno** - Second run same day (auto-incremented sequence)

File naming advantages: - **Date tracking**: YYYYMMDD shows when data was processed - **Run sequencing**: Multiple runs same day auto-numbered (00, 01, 02…) - **Conflict prevention**: System checks for existing files and increments sequence - **Standard extension**: `.geno` indicates QC-passed SNP genotype file

Example progression:

```
my_analysis_20260213_00.geno  (First run on Feb 13)
my_analysis_20260213_01.geno  (Second run same day)
my_analysis_20260213_02.geno  (Third run same day)
my_analysis_20260214_00.geno  (First run on Feb 14)
```

3.5 3.5. Output File Formats

3.5.1 GENO File Format

```
ARN         SNP1 SNP2 SNP3 SNP4 ...
21905009744 0    1    2    5    ...
21905009529 1    2    0    1    ...
```

Coding: - `0`: AA (homozygous reference) - `1`: AB (heterozygous) - `2`: BB (homozygous alternate) - `5`: Missing genotype

3.5.2 QC Report Contents

- Input file information
- QC threshold settings applied
- Statistics per animal (pass/fail counts)
- Statistics per SNP (pass/fail counts)
- Failure reasons breakdown
- Processing timestamp

4 4. Hash Table Libraries

4.1 4.1. M_HashTable - Generic Hash Table

Generic hash table with numeric string keys.

4.1.1 Creating and Using

```fortran
use M_HashTable

type(HashTable) :: ht

! Create table
call ht_create(ht, 1009)  ! Size: prime number

! Insert data
call ht_insert(ht, "2190500974", 100)
call ht_insert(ht, "2190500529", 200)

! Search for data
integer :: value
logical :: found
found = ht_search(ht, "2190500974", value)

! Delete data
logical :: deleted
deleted = ht_delete(ht, "2190500974")

! Print statistics
call ht_print_stats(ht)

! Cleanup
call ht_free(ht)
```

4.1.2 Performance Characteristics

- Average insertion: O(1)
- Average search: O(1)
- Average deletion: O(1)
- Worst case: O(n)

4.1.3 Table Size Recommendations

- ~100 items: Use 151 (prime)
- ~1,000 items: Use 1511 (prime)
- ~10,000 items: Use 15013 (prime)
- ~100,000 items: Use 150001 (prime)

4.2 4.2. M_PEDHashTable - Pedigree Hash Table

Specialized hash table for pedigree data using ARN (Animal Registration Number) as key.

4.2.1 Data Structure

```
type, PUBLIC :: PEDInfo
    character(len=100) :: BREED     ! Breed name
    character(len=100) :: ID        ! Animal ID
    integer(kind=8) :: ARN          ! Registration number (key)
    character(len=100) :: SIRE      ! Sire ID
    character(len=100) :: DAM       ! Dam ID
    integer :: SEX                  ! Sex (1=male, 2=female)
    integer :: BDate                ! Birth date (YYYYMMDD)
    character(len=100) :: LOC       ! Location
end type PEDInfo
```

4.2.2 Usage Example

```
use M_PEDHashTable
use M_Variables

type(PEDHashTable) :: ped_ht
type(PEDInfo) :: ped, found_ped

! Create table
call pht_create(ped_ht, 2000)

! Insert pedigree data
ped%ARN = 21905009744_ki8
ped%ID = 'P001'
ped%BREED = 'Duroc'
ped%SIRE = 'S001'
ped%DAM = 'D001'
ped%SEX = 2
ped%BDate = 20220115
ped%LOC = 'Farm1'

call pht_insert(ped_ht, ped)

! Search by ARN
logical :: found
found = pht_search(ped_ht, 21905009744_ki8, found_ped)

if (found) then
    print *, 'Found:', trim(found_ped%ID)
end if

! Cleanup
```

```
call pht_free(ped_ht)
```

4.2.3 API Functions

Function	Purpose
`pht_create(ht, size)`	Create hash table
`pht_insert(ht, ped)`	Insert pedigree record
`pht_search(ht, arn, ped)`	Search by ARN
`pht_delete(ht, arn)`	Delete record by ARN
`pht_free(ht)`	Free memory
`pht_print_stats(ht)`	Print statistics
`pht_print_ped_info(ped)`	Print pedigree info

5 5. Compiling User Programs

5.1 5.1. Static Linking

```
gfortran -O2 -I/usr/local/include/dkblupf90 \
    my_program.f90 \
    /usr/local/lib/libdkblupf90.a \
    -o my_program
```

Advantages: - Standalone executable - No runtime library dependency

Disadvantages: - Larger file size

5.2 5.2. Dynamic Linking

```
gfortran -O2 -I/usr/local/include/dkblupf90 \
    my_program.f90 \
    -L/usr/local/lib -ldkblupf90 \
    -o my_program \
    -Wl,-rpath,/usr/local/lib
```

Advantages: - Smaller executable - Can update library without recompilation

Disadvantages: - Library must be available at runtime

5.3 5.3. Example Makefile

```
FC = gfortran
FFLAGS = -O2 -g
INCLUDE_DIR = /usr/local/include/dkblupf90
LIB_DIR = /usr/local/lib
LIB_NAME = dkblupf90

TARGET = my_program
SRC = my_program.f90

all: $(TARGET)

$(TARGET): $(SRC)
    $(FC) $(FFLAGS) -I$(INCLUDE_DIR) $(SRC) \
        -L$(LIB_DIR) -l$(LIB_NAME) \
        -Wl,-rpath,$(LIB_DIR) -o $(TARGET)

clean:
    rm -f $(TARGET) *.o *.mod

.PHONY: all clean
```

6 6. Troubleshooting

6.1 6.1. Installation Issues

6.1.1 gfortran not installed

```
# Ubuntu
sudo apt-get install gfortran

# CentOS/RHEL
sudo yum install gcc-gfortran
```

6.1.2 Permission denied

```
# Install in user directory (no sudo needed)
PREFIX=$HOME/.local ./install.sh
```

6.1.3 Make command not found

```
# Ubuntu
sudo apt-get install make

# CentOS
sudo yum install make
```

6.2 6.2. Runtime Issues

6.2.1 Library not found

```
# Set library path temporarily
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

# Or permanently (add to ~/.bashrc)
echo 'export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH' >> ~/.bashrc
```

6.2.2 ReadFR command not found

```
# Check PATH
echo $PATH

# Add to PATH
export PATH=/usr/local/bin:$PATH
```

6.2.3 Parameter file error

1. Check file paths exist and are correct
2. Verify column numbers match FinalReport structure
3. Check for Windows line endings (use `dos2unix parameter_file`)
4. Ensure proper formatting

6.2.4 Compilation errors

Common error: Cannot find module file

`Solution: Check -I (include) path points to module files`

Common error: Undefined reference

`Solution: Link with library: gfortran ... -ldkblupf90`

6.3 6.3. Performance Issues

6.3.1 Slow execution

- Use SSD instead of HDD
- Increase available system memory
- Check system load (`top | head`)

6.3.2 Out of memory

- Reduce dataset size
- Run on machine with more RAM
- Process data in batches

6.3.3 High CPU usage

- Normal for large datasets
- Monitor with `top` or `htop`

7 7. FAQ

7.1 General Questions

Q: What license is DKBLUPF90 released under? A: MIT License - free to use in commercial and academic contexts.

Q: Can I run on Windows? A: Not natively. Use WSL2 (Windows Subsystem for Linux) or VirtualBox with Linux.

Q: Is there GPU acceleration support? A: Not in current version. Planned for future releases.

Q: Can I compile with Intel Fortran? A: Yes - replace `gfortran` with `ifort` in compilation commands.

7.2 Installation Questions

Q: Do I need root to install? A: Only for system-wide installation. Use `PREFIX=$HOME/.local` for user installation.

Q: Can I build in a non-standard directory? A: Yes - set working directory before running make.

Q: How do I uninstall? A: Run `/usr/local/bin/uninstall-dkblupf90.sh` or manually delete files.

7.3 ReadFR Questions

Q: What FinalReport versions are supported? A: All versions - specify columns in parameter file.

Q: Can I process multiple files at once? A: Process each file separately, then merge GENO outputs.

Q: What's the maximum file size? A: Limited only by available RAM (~50 million SNP*Animal combinations).

Q: How do I handle missing data? A: Represented as `5` in GENO file. Imputation tools can process these.

7.4 Performance Questions

Q: How long does processing take? A: Roughly 1-2 seconds per 1 million SNP*Animal combinations.

Q: How much disk space do I need? A: Similar to input file size (~2-5KB per SNP).

Q: Can I run in parallel? A: Process different files, then merge outputs.

7.5 Data Questions

Q: What's the PED file format? A: Space-separated columns: BREED ID ARN SIRE DAM SEX BDATE LOC

Q: What's the MAP file format? A: SNP_Name, Chr, Position (space-separated)

Q: What does GENO file contain? A: Raw genotypes (0/1/2/5) suitable for genomic prediction software.

8 8. Advanced Topics

8.1 8.1. Batch Processing

Process multiple FinalReport files efficiently:

```
#!/bin/bash
for file in *.txt; do
    echo "Processing $file..."

    # Create parameter file
    sed "s|DATAFILE|$file|g" template_param > temp_param

    # Run ReadFR
    ReadFR temp_param

    rm temp_param
done
```

8.2 8.2. Quality Control Workflow

Recommended QC sequence: 1. Set moderate thresholds first 2. Examine QC report 3. Adjust thresholds based on data 4. Re-run if needed 5. Verify output file integrity

8.3 8.3. Data Validation

Validate GENO files:

```
tail -n +2 file_GENO.txt | wc -l
head -n 1 file_GENO.txt | awk '{print NF-1}'
awk 'NR>1 {for(i=2;i<=NF;i++) if($i!="0" && $i!="1" && $i!="2" && $i!="5") print "Invalid:", $
```

8.4 8.4. Integration with Other Tools

8.4.1 Merging GENO files

```
# Concatenate with header from first file
head -1 file1_GENO.txt > merged_GENO.txt
tail -n +2 file1_GENO.txt >> merged_GENO.txt
tail -n +2 file2_GENO.txt >> merged_GENO.txt
```

8.4.2 Converting to AlphaImpute format

AlphaImpute uses `9` for missing instead of `5`:

```
sed 's/ 5/ 9/g' input_GENO.txt > alphaimpute_GENO.txt
```

9 9. Appendices

9.1 A. Quick Reference - QC Thresholds

Parameter	Default	Lenient	Strict
Animal Call Rate	0.95	0.90	0.98
SNP Call Rate	0.90	0.85	0.95
GC Score	0.65	0.60	0.70
GT Score	0.50	0.40	0.60
Cluster Sep	0.30	0.20	0.40
R-Intensity Min	0.40	0.30	0.50
R-Intensity Max	2.0	2.5	1.8

9.2 B. Common Parameter File Examples

9.2.1 Example 1: Default QC

```
PEDFile /data/PED_Total.txt
SNPFile /data/FinalReport.txt
ANIMAL-ARN 2
SNP_Name 1
Chr 10
Position 11
Allele1-AB 13
Allele2-AB 14
MAPFile /data/MAP.txt
OutputPrefix default_analysis
```

9.2.2 Example 2: Strict QC

```
# (Same as above, plus)
AnimalCallRate 0.98
SNPCallRate 0.95
```

With column-specific threshold:

```
GC_Score 27 0.70
GT_Score 29 0.60
```

9.3 C. System Performance Benchmarks

Tested on Intel Core i7, 16GB RAM, NVMe SSD:

Animals	SNPs	Time	Memory
100	10K	1s	50MB
500	50K	5s	200MB
2000	60K	15s	500MB
10000	70K	2m	2GB

Animals	SNPs	Time	Memory
50000	70K	10m	8GB

9.4 D. Related Resources

- BLUPF90 (http://nce.ads.uga.edu/wiki/doku.php)
- AlphaImpute2 (https://alphagenes.roslin.ed.ac.uk/)
- Illumina GenomeStudio
- PLINK (genome analysis tool)

9.5 E. Error Messages Reference

Error	Cause	Solution
"Parameter file not found"	Wrong path	Check file path
"Library not found"	Missing library	Set LD_LIBRARY_PATH
"Invalid ARN in PED"	Data format	Check PED file format
"SNP not in MAP"	Missing reference	Verify SNP exists in MAP
"Module file missing"	Build incomplete	Run make clean && make lib

---

DKBLUPF90 User Manual v1.0.0
February 2026
For support: See project documentation