The **dwarf** User Manual

M.D. Preston

April 26, 2013

Contents

1 Introduction

Dwarf is a new style of genetic analysis tool - a scriptable and interactive command line program. It's major advantage over comparable tools is its ability to run multiple analyses on data that is loaded and filtered only once; providing a major speed advantage. It includes the following features:

- Load data from multiple formats including:
  BED, PED, FAM, MAP, EMAP, POLY, Transmit, VCF
- Simulate genetic data
- Manipulate the data:
  create pseudo-controls, resize, transform between formats
- Run a number of statistical tests including:
  association, C-α, KBAC, regression, score, SKAT, SSU, SSUw, sum, TDT, UminP
- Save statistical data for external analysis
- Interface with R and Octave
- Interface with the shell, to give access to any other installed command line programs including:
  ask, bash, perl, python, sed
  as well as other genetic software tools:
  plink, mendel, samtools, transmit, unphased, vcftools, *etc.*

The most common way to use dwarf is via a script and calling it on the command line:

```
dwarf my.script
```

on a 64-bit Linux machine that has R (and a number of R libraries) installed on it. Software can be downloaded from:

```
http://software.markdpreston.com/dwarf
```

as is the code for users to build (and change) as they see fit, see §6 for instructions.

Scripts are lists of commands that are executed in order. They can include simple loops (using the `for...next` construct) to allow for repeated execution of a section of commands and all of the commands are described in §3 with some example scripts given in §5.

Note, planned future work includes a graphical front-end á la Matlab and possibly an integration with `varb` [4].

2 Internal Operation

Internally the information is held in two types of structure: populations and data. Populations hold the family, subjects, SNP and genotypic information whereas the data structures hold the results of the analysis routines (such as the `association` command, see below).

Each data analysis command creates a data structure to store the results of its calculations in, its size is determined by the command line and the command. For example, the command line

```
single 10 7 data=Extra
```

will create/use a 10-by-3 matrix called `singleExtra`, will be writing to the 7th row out of 10 and to the three columns with the three calculated p-values (see below for a description).

The population structure is filled either by simulated data or from saved files. There are a number of commands to accomplish this as well as to manipulate/analyse the data after it is loaded. The population structure stores the data, internally, in one of 4 formats:

- Unphased: follwing the PED format the data is stored as wildtype (00), heterozygous (01), homozygous (11) and missing (10)
- Phased: the data is stored in a phased manner: wildtype (00), heterozygous (01 or 10) and homozygous (11)
- Dosage: each variant is stored as a probability of wildtype, heterozygous and homozygous (summing to 1)
- PolyData: each variant is stored according to its pair of nucleic base (ACGT) or as an insertion (Ixx) or deletion (Dxx) where xx indicates the length

3 Commands

3.1 Load, save, maniulate data

All of the commands in this section operate on the population identified by the `population` parameter on the command line, if no value is given then `population=default` is assumed.

3.1.1 clean

This command has three functions. Firstly it sets the internal relationship statuses and links up to match the pedigree data, *i.e.* links and counts parents, siblings and offspring. Secondly it homogenises the family and individual names within the data set to the `F00XX` and `I000XXX` format and updates all related structures appropriately. Finally it changes all missing or unknown phenotypes (coded 0 or -9) to controls (coded 1).
Usage:

`clean` [`population=`*default*]

3.1.2 clear

This command removes all data (subjects, SNPs, genes, *etc*) from the given population.
Usage:

`clear` [`population=`*default*]

3.1.3 load

The `load` command handles a variety of data formats to load in subject, pedigree, haplotype, genotype and genetic data. The command takes the form:

`load` format filename

where format can take one of 17 values, including map, ped, bed, transmit, vcf. See the File Formats (§4). Note that the loader tries to be smart, using the file (in two or three passes if required) and previously loaded data, to determine number of subjects or SNPs. The only exception to this is the bed (binary pedigree file format) that requires the number of subjects and SNPs to be passed on the command line, for example:

```
load bed myfile.bed subjects=100 snps=1000
```

Usage:

`load` format filename [`snps=`*0*] [`subjects=`*0*] [`population=`*default*]

3.1.4 pseudo

The `pseudo` command creates a new pseudo-case-control (`to`) population out of all of the trios from families in the current population. It does this by copying all cases with two parents in the current population into the `to` population and creating matching a pseudo control in the `to` population from the untransmitted alleles of the parents.
Usage:

`pseudo` [`population=`*default*] [`to=`*default*]

3.1.5 save

The `save` command extracts the data from the given population (or `default`) and saves the data to the given filename in most of the formats given in File Formats (§4). The optional `haplotypes` parameter indicates that 0—1 and 1—0 phased data should be respected when saving and not saved to an unphased heterozygous 0/1.
Usage:

`save` format filename [`haplotypes`]

3.1.6 transform

This command transforms, internally, the storage type of the population data between phased, unphased, poly and dosage formats, see §2, given by the type values `haplotypes`, `genotypes`, `dosages` and `polydata`.
Usage:

`transform` type [`population=`*default*]

3.2 Data Analysis

These commands apply statistical analysis to the genetic data. The all have the same basic format:

command total iteration

with the following optional components (with their default values):

[`population`=*default*] [`permutation`=*5000*] [`data`=] [`file`=*null*] [`test`]

Specific command line options for each command are detailed below.

Internally *dwarf* gives each population and dataset a name. The two internal population names are default and sample, default is used for loading, saving and analysing genetic data and sample is used for storing haplotype pools from which data is sampled to create a simulated population. Each command is designed to run over a collection of datasets sequentially and the output data stored in a matrix of with *total* rows, useful for power calculations and the like. So each command stores the results of its calculation in a matrix called *command* but this can be appended to by using the `data` option.

For commands that use permutation testing to determine p-values the permutation option sets the total number of permutations to test. The incidental output during the running of a test is ignored (i.e. sent to `/dev/null`), but if `file=cout` is given output will be to the console and any other string will be interpreted as a filename to write to. The `test` option forces the command to test it's calculations against an external benchmark, commonly by `plink` or via an `R` script as noted in each description.

For example the command:

```
calpha 10 7 data=_extra file=cout
```

will run the 7^{th} out of 10 `calpha` tests on the population called `default`, storing the output in the data structure named `calpha_extra` and will output any incidental messages during the execution to the console (`cout`).

3.2.1 association

Standard, chi-squared, SNP-by-SNP association testing. It that uses `plink` [6] for verification.
Usage:
`association` total iteration
See also:
`single`, `score` (UminP).

3.2.2 calpha

Standard C-α test, [3]. No verification at this time.
Usage:
`calpha` total iteration

3.2.3 kbac

Applies the kernel-based adaptive cluster algorithm with the hyper-geometric kernel [2] and uses the C++ code behind the `KbasTest` command in the R KBAC library for verification. *Note: this test uses permutation so the verification may fail due to a small delta in the p-value and this can be a very computational intensive routine.*
Usage:
`kbac` total iteration

3.2.4 regression

Perform a regression analysis with the affection status as the reponse and the genotypic data as the exposure. If the individual option is chosen the regression is applied SNP-by-SNP otherwise a multiple regression is applied over all of the SNPs.
Usage:
`regression` total iteration [`method=`*individual*]

3.2.5 score

Implements the UminP, sum of squared score (SSU), weighted SSU (SSUw), score and sum tests (see [1] for full details). If U is a vector of scores and V is a vector of (co-)variances then the test statistics are:

$$
\begin{aligned}
T_{UminP} &= \max_i \frac{U_i}{V_{ii}} \\
T_{SSU} &= U^T U \\
T_{SSUw} &= U^T diag(V)^{-1} U \\
T_{Score} &= U^T V^{-1} U \\
T_{Sum} &= \frac{(\sum_i U_i)^2}{\sum_{ij} V_{ij}}
\end{aligned}
$$

and the p-values are generated from appropriate combinations of χ^2 distributions. The data is output in a matrix with 5 columns, one for each of these tests.

The optional argument `method` indicated whether the unrelated score tests are used or if the `family` option is chosen then the new score tests based on transmitted/untransmitted alleles from parents (defined in [5]) are to be used. The `cluster` option tells the program to use a simple clustering algorithm to group data from related cases together and from related controls together.

This command uses a modified R script taken from [1] supplementary material for verification of unrelated case-control data.

Usage:

`score` total iteration [`method=`*unrelated*] [`cluster`]

3.2.6 single

For `method=unrelated` (the default) single performs SNP-by-SNP association tests and for `method=family` performs a SNP-by-SNP TDT test calculating the minimum P-value. The minimum P-value, with Bonferroni and Ŝidák correction, are stored in the three columns of the output data, *i.e.*,

$$p_{bonferroni} = Np_{min} \qquad \text{and} \qquad p_{sidak} = 1 - (1 - p_{min})^N,$$

where N is the number of tests/SNPs. This enables all data to be tested against the same significance level.

Usage:

`single` total iteration [`method=`*unrelated*]

See also:

`association`, `tdt`.

3.2.7 skat

This statistical test calls the R skat library directly, [7].

Usage:

`skat` total iteration

3.2.8 tdt

Performs a transmission disequilibrium test (TDT) SNP-by-SNP for each affected child with two parents in the dataset.

Uses plink for testing.

See also:

`single`

3.3 Simulation

The simulation command has 5 sub commands detailed below.

See also:

`psuedo`

3.3.1 snps

The `snps` subcommand creates a SNP profile in a given `population=sample` in two ways. The first and easiest is to pass in an extended MAP (EMAP) file with the `emap=filename` option. If this is not given then the profile is created

randomly. The number of SNPs is given by the `count` option. For each SNP the MAFs are determined by the `mafdistribution` and `mafbound` options. The first `affectingcount` SNPs are given the OR of `oddsratio` and the subsequent SNPs are given an OR of 1.0 equating to no effect.
Usage:

`simulation snps` [`emap`=filename] [`population`=*sample*]

or

`simulation snps` [`count`=*0*] [`mafdistribution`=*0*] [`mafbound`=*0.5*] [`affectingcount`=*0*] [`oddsratio`=*1*] [`population`=*sample*]

3.3.2 haplotypes

The `haplotypes` subcommand creates a pool of haplotypes from the SNP profile in the `population=sample`. The number of haplotypes is given in as a plain number.
Usage:

`simulation haplotypes` count [`population`=*sample*]

3.3.3 affection

The `affection` subcommand sets the parameters that, in combination with `snp` subcommand, complete the SNP profile and will enable any given generated subject's affection status to be determined. It is not required if an EMAP file was supplied above.
Usage:

`snp affection` [`trait`=*binary*] [`basline`=*0.1*] [`controlMin`=-∞] [`controlMax`=*0*] [`caseMin`=*0*] [`caseMax`=∞] [`population`=*sample*]

3.3.4 subjects

The `subjects` subcommand creates the simulated population from the sample units. Each sample unit is made up of one pedigree of siblings and parents in combination with a number of unrelated subjects. Each of the subjects in the sample unit is either a case, a control or it doesn't matter which they are (unknown status). A unit is simulated with genotypic data random taken from the `sample` population, the affection status of each member of the unit is determined from the previous information and if all of their statuses match the requirements then it is added to the population. Units are created until there are `units` in the simulated population.
Usage:

`snp subjects` [`units`=*0*] [`caseSiblings`=*0*] [`caseParents`=*0*] [`caseUnrelateds`=*0*] [`controlSiblings`=*0*] [`controlParents`=*0*] [`controlUnrelateds`=*0*] [`unknownSiblings`=*0*] [`unknownParents`=*0*] [`unknownUnrelateds`=*0*] [`population`=*default*] [`sample`=*sample*]

3.3.5 replace

All controls in a population are replaced by new controls randomly created from the `sample` population.
Usage:

`snp replace` [`population`=*default*] [`sample`=*sample*]

3.4 Utility

3.4.1 comment

The # character indicates that all further text on this line is to be ignored.

3.4.2 constant

A useful option within dwarf scripts is to define constants. The constant command takes two parameter the name and the value. The value can be a number or string as dwarf only applies a straightforward replace algorithm on each line. The substitution occurs when the token %name% is found. This is useful to sync up loop maxima, analysis and tick commands.
Usage:
`constant` name value
See also:
`for`, `tick`

3.4.3 data

This command saves internal data to a file. A comma separated list of variable names (or `all` by default) are saved to `file` in Matlab/Octave format. To save data during long runs if `every` divides `count` then the data is written. For example, this command can be used in a loop and every 10 iterations the data is saved.
Usage:
`data save` [`variables=`*all*] [`file=`*none*] [`count=`*1*] [`every=`*1*]

3.4.4 echo

The `echo` command provides user output. All text after the command will be output directly to the screen, after all variable and constant substitutions have taken place.

3.4.5 for ... next

A loop construct that takes up to 4 parameters. The first (required) parameter is the loop variable name, the next two (requried) parameters are the first and last numbers to iterate over and the fourth (optional) parameter indicates the length of the 0-padding to employ during substitution. The loop variable is accessed through the %% construct in the same manner as for constants. For example

```
for i 1 100 5
  echo %i%
  load ped mydata%i%.ped
next
```

will output 00001, 00002, 00003, ..., 00099, 00100 on successive lines and open the file mydata00001.ped, ..., mydata00100.ped. As we can see the 0-padding is useful in file name creation.

3.4.6 help

Calling `help` by itself lists the commands available. Calling it with a command afterwards prints the help available for that command.

3.4.7 next

See `for`

3.4.8 quiet

The default behaviour is for dwarf to echo commands to the terminal. This command turns off the echoing commands to the terminal.
See also:
`verbose`

3.4.9 run

Calls a script file and runs the commands therein.
Usage:
`run` scriptname

3.4.10 seed

Sets the random number seed to enable exact recreation of 'random' results.

3.4.11 system

The system command passes the entire command line after the system keyword to the external shell. This is useful to call external programs (such as plink or ms) or to manipulate files such as

```
system paste -d' ' a.emap  b.emap > e.emap
```

Usage:
`system` shellcommand

3.4.12 tick

Dwarf starts a timer every time it is executed. This prints out the elapsed time during script execution in a loop. The three parameters are loop size, loop iterate and divisor. If loop iterate is exactly divisable by the divisor then the time is output, with loop information, otherwise no output is given. For example

```
tick 10000 %i% 1000
```

will output the time elapsed since starting `dwarf` every 1000 iterations through a loop indexed by `%i%`.
Usage:
`tick` total iterate every
See also:
`time`.

3.4.13 time

This prints out the elapsed time since `dwarf` was started.
See also
`tick`.

3.4.14 verbose

The default behaviour is for dwarf to echo commands to the terminal. This command turns back on the echoing commands to the terminal after a quiet command.
See also:
`quiet`.

4 File Formats

The file types dwarf recognises are listed here. Descriptions or references to other resources are given. Data is loaded with `load XXX FFF` command, where `XXX` is a file type and `FFF` is a file name.

4.1 Plink Style

The standard formats of genotypic data used in the plink software 4.1 are BED, BIM, FAM, MAP and PED, with lowercase codes bed, bim, fam, map and ped respectively. The plink homepage gives a very good description of these formats.

4.2 dwarf Formats

We have defined two file types for use with dwarf: EMAP and POLY (with codes emap and poly).

The EMAP format is a standard MAP format with 3 extra columns for simulating samples (see the `simulation` command above). The three extra columns are minar allele frequency (MAF), effect type and effect size. The MAF must be a number between 0 and 1. The effect type can be null (0) or additive (A) and the effect size is given as an odds ratio (OR). An OR above 1 indicates deleterious and below one indicates a protective effect.

The POLY format is an extended verion of the PED format. The two alleles at each position for each subject can be A, C, G or T with the addition of I and D. I and D represent an insertion and a deletion. They must be the first allele position and the second allele position contains the length of the insertion/deletion.

4.3 Marker Formats

The LEGEND (legend) and MARKERS (markers) file formats encode positional data. LEGEND has five fields per line: field one (not used), marker name, position, allele 1 and allele 2. The MARKERS file has 3 fields per line: marker name, distance and position.

4.4 Data Formats

The VCF (variant call format) file format is commonly used in SNP calling from real (*i.e.* not simulated) sequencing data such as the 1000 Genomes Project [17].

The GENS (gens) file format encodes dosages as per the Beagle genotypes likelihood file format [9]. The second line onwards consists of the marker name, allele 1, allele 2 and then pairs of major and heterozygous dosages. This leaves the minor dosage as 1 - major - hetero.

The HAPS (haps) and KBAC (kbac) formats have one line per subject and the data for each loci are seperated by a space. In the HAPS format the phased data is encoded as 00, 01, 10 and 11. In the KBAC format the count of minor alleles, 0, 1 or 2.

5 Example Script

```
quiet
time
echo

seed 1

simulation snps emap=test.emap
simulation affection baseline=0.1
simulation haplotypes 1000000
simulation subjects units=2000 caseSiblings=1 unknownParents=2

echo Family
tdt         1 1 test
single      1 1 test method=family
single      1 1 test method=family permutations=1000
score       1 1 test method=family
score       1 1 test method=family permutations=1000
kbac        1 1 test               permutations=1000
regression  1 1 test
echo

echo Psuedo Case Control
pseudo
association 1 1 test
single      1 1 test
score       1 1 test
calpha      1 1 test
kbac        1 1 test permutations=1000
regression  1 1 test
echo

echo Case Control
simulation  replace 200 controls
association 1 1 test
single      1 1 test
score       1 1 test
calpha      1 1 test
kbac        1 1 test permutations=1000
regression  1 1 test

time
echo
```

6 Build from Source

The website [8] has up-to-date 64-bit Windows and Linux executable versions of the software.

Building it for yourself is fairly straightforward (on Linux) or fairly tedious (on Windows) as it does rely on a number of other software packages/libraries:

- Blitz++: a matrix library [10]
- Boost: another matrix library [12]
  with libboost_system
- Eigen: yet another matrix library [11]
- GSL: a library for mathematics routines [13]
- R: the only statistical software to be using [14]
  with Rcpp, RInside, mvtnorm, KBAC [15], SKAT [16]libraries
- Fortran/f2c: old school Fortran 77 compiler

6.1 Linux

Under any modern Linux this is all very simple and all of these packages are available to be installed using your local installer (*i.e.* `apt-get`). The `Makefile` might need minor modifications to use `/usr/local/` or `~/` for some of the paths. To build the software, download it, uncompress it and run `make`; this will create an executable that can be run with the command: `./dwarf`. If the R packages are not installed on your machine then use the `install.package` command within `R` from the repositories or from downloaded versions. You will have to download KBAC as this is a non-repository package [15].

6.2 Windows

This is not recommended. Windows is not a sensible option for a building project of this complexity. Either use a proper operating system (*i.e.* Linux) or use the downloadable executable.

References

[1] Basu,S, Pan,W. (2011) Comparison of statistical tests for disease association with rare variants, Genetic Epidemiology, Volume 35, 606-619

[2] Lui,D.J., Leal,S.M. (2010) A Novel Adaptive Method for the Analysis of Next-Generation Sequencing Data to Detect Complex Trait Associations with Rare Variants Due to Gene Main Effects and Interactions, PLOS Genetics, Volumne 6, Issue 10.

[3] Neale BM *et al.* (2011) Testing for an unusual distribution of rare variants, American Journal of Human Genetics 87, 604-617.

[4] Preston MD *et al.* (2012) VarB: a variation browsing and analysis tool for variants derived from next-generation sequencing data, Bioinformatics 28, 2983-2985.

[5] Preston MD, Dudbridge D (submitted) Using family-based designs for detecting rare variant disease associations.

[6] Purcell,S., Neale,B., Todd-Brown,K., Thomas,L., Ferreira,M.A.R., Bender,D., Maller,J., Sklar,P., De Bakker,P.I.W., Daly,M.J., *et al.* (2007) PLINK: a toolset for whole-genome association and population-based analyses. Am J Hum Genet **81** 559-575

[7] Wu MC, Lee S, Cai T, Li Y, Boehnke M, Lin X (2011) Rare-variant association testing for sequencing data with the sequence kernel association test, American Journal of Human Genetics 89 , 82-93.

[8] http://software.markdpreston.com/dwarf/

[9] http://faculty.washington.edu/browning/beagle/beagle.html

[10] http://sourceforge.net/projects/blitz/

[11] http://eigen.tuxfamily.org/

[12] http://www.boost.org/

[13] http://www.gnu.org/s/gsl/

[14] http://www.r-project.org/

[15] http://code.google.com/p/kbac-statistic-implementation/downloads/list

[16] http://www.hsph.harvard.edu/research/skat/

[17] http://www.1000genomes.org/node/101