User Manual for fgwas v.0.3.x

Joseph K. Pickrell

July 11, 2014

Contents

1 Introduction

fgwas is a program for incorporating functional genomic information into a GWAS. It estimates the enrichment of GWAS hits in different annotation types (potentially jointly modeling multiple annotations) and can re-weight the GWAS using functional genomic information. The model works best if there are at least 20 independent unambiguous associations (with $P < 5 \times 10^{-8}$) in the genome.

2 Installation

fgwas should run on any Unix or Unix-like (e.g., Linux or Mac OS X) system. It requires the GNU Scientific Library (`http://www.gnu.org/s/gsl/`; all testing was done with version 1.15), and the Boost Graph Library (`http://www.boost.org/`; you need version 1.42 of Boost or greater). Be sure these libraries are installed, and after downloading the source code, run the standard installation steps:

```
>tar -xvf fgwas-0.1.tar.gz
>cd fgwas-0.1
>./configure
>make
>make install
```

3 Input file format

3.1 GWAS and genomic data input

The input is a gzipped, space-delimited text file containing information about the GWAS and functional annotations you're interested in (this is generally hundreds of thousands to millions of SNPs). All SNPs must be ordered by position. The columns have no enforced order, but are identified from the header. If this is a study of a quantitative trait, the necessary columns are:

1. SNPID: a SNP identifier
2. CHR: the chromosome for the SNP
3. POS: the position of the SNP
4. F: the allele frequency of one of the alleles of the SNP
5. Z: the Z-score for from the association study
6. N: the sample size used in the association study at this SNP (this can vary from SNP to SNP due to, e.g., missing data).

Note: if you include a column labeled SE in the header of the file, this column will be taken as a direct estimate of the standard errors of the regression parameter and will override the sample size and allele frequency columns.

The remainder of the columns can have almost any name (apart from all those listed above). Columns can be of two types:

1. 1/0 for presence/absence. For example, there could be a column containing a "1" if the SNP falls in an exon and "0" otherwise.
2. integer for distance. For example, there could be a column containing the distance to the nearest transcription start site for each SNP.

For example, the following (in a gzipped text file) would be an allowed input:

```
SNPID CHR POS Z F N coding_exon tss_dist
rs134344 chr5 56665 -1.2 0.3 9800 0 23550
rs444329 chr5 106665 -0.4 0.9 9800 0 66783
rs532 chr6 102435 4.4 0.09 9798 1 550
```

3.1.1 Case/control studies (`-cc`)

If the study is a case/control study, use the `-cc` flag for all commands. The input file needs:

1. SNPID: a SNP identifier
2. CHR: the chromosome for the SNP
3. POS: the position of the SNP
4. F: the allele frequency of one of the alleles of the SNP
5. Z: the Z-score for from the association study
6. NCASE: the number of cases used in the association study at this SNP
7. NCONTROL: the number of controls used in the association study at this SNP

Note: if you include a column labeled SE in the header of the file, this column will be taken as a direct estimate of the standard errors of the log-odds ratio and will override the sample size and allele frequency columns.

The remainder of the columns describing the annotations are identical to those for quantitative trait studies.

3.1.2 Fine-mapping/eQTL input (`-fine`)

In the fine-mapping/eQTL case, we assume that there are a set of regions of interest (for example, regions containing the top 50 hits in a GWAS, or the cis regions surrounding 400 genes with eQTLs). In this case, the file format is the same as above with two exceptions. First, there is an additional column labeled SEGNUMBER. Each region should have a unique SEGNUMBER identified. Second, the file should be ordered according to this column (rather by chromosome and position). For example, the following (in a gzipped text file) would be an allowed input:

```
SNPID CHR POS Z F N SEGNUMBER coding_exon tss_dist
rs134344 chr5 56665 -1.2 0.3 9800 1 0 23550
rs444329 chr5 106665 -0.4 0.9 9800 1 0 66783
rs532 chr6 102435 4.4 0.09 9798 2 1 550
rs538 chr6 103355 1.2 0.48 9798 2 0 35354
```

3.2 Distance model

To use a distance annotation, the program needs a file with the distance model. This is a space-delimited file showing the distance bins to use. For example, imagine you have a column noting the distance to the nearest transcription start sites for each SNP, and you want to estimate a parameter for all SNPs within 5kb of a TSS. The distance model file would then be a file with the following single line:

```
0 5000
```

4 Options

4.1 SNP annotations

The main use of fgwas is to test whether SNPs that influence a trait are enriched or depleted in certain genomic annotations. The names of the annotations are defined by the header of the input file.

4.1.1 Test a presence/absence annotation for enrichment (`-w`)

Most annotations are presence/absence annotations coded as 0/1 in the input file (for example, nonsynonymous SNPs might be coded as 1 in a column called "nonsynonymous", which all other SNPs would be coded as 0 in that column). To tell fgwas which annotations to test, use the `-w` flag:

```
>fgwas -i input_file.gz -w nonsynonymous
```

To build a model with multiple annotations, simply separate the annotations with a `+`:

```
>fgwas -i input_file.gz -w nonsynonymous+annotation_2+annotation_3
```

4.1.2 Test distance annotations for enrichment (`-dists`)

To test a distance annotation, first create a distance model file as described in Section 3.2. Then tell the program which column contains the distance annotation by giving an argument of the form [distance annotation]:[distance model]. For example, if the distance annotation is called "tss_dist" and the distance model is defined in a file called "dist_model", run:

```
>fgwas -i input_file.gz -dists tss_dist:dist_model
```

4.2 Regional annotations (`-dens`)

The model can additionally estimate parameters at a region-level. Currently the only way to do this is to split regions into three groups according the their average value of some column (for example, distance to a transcription start site). Tell the program which column with an argument of the form [annotation] [low quantile] [high quantile]. For example, imagine each SNP has a distance to the nearest transcription start site in a column called "tss_dist". Then to estimate a parameter for regions in the top third and bottom third of average distance to a TSS, run:

```
>fgwas -i input_file.gz -dens tss_dist 0.33 0.67
```

4.3 Setting the window size (`-k`)

The default number of SNPs in a "region" is 5,000. To change this, set the `-k` flag. For example:

```
>fgwas -i input_file.gz -k 10000
```

4.4 Setting the prior variance on effect size (`-v`)

The default prior variance on the effect size of SNPs is 0.1. To change this, set the `-v` flag:

```
>fgwas -i input_file.gz -v 0.5
```

4.5 Fine-mapping study input format (-fine)

To tell the program to expect an input file in fine-mapping format (see Section 3.1.2 above), use the -fine flag For example:

```
>fgwas -i input_finemap_file.gz -fine
```

4.6 Changing the output file name stem (-o)

The default name for the output files is "fgwas". To change this set the -o flag. For example:

```
>fgwas -i input_file.gz -k 10000 -o bigregions
```

4.7 Re-weighting GWAS and outputting posterior probabilities of association (-print and -p)

The default behavior of fgwas is to estimate enrichment parameters but not to output re-weighted summary statistics. To output re-weighted summary statistics, use the -print flag. Note that this will tell the program to optimize a penalized likelihood, with penalty set by the -p flag (default is 0.2). For example:

```
>fgwas -i input_file.gz -print -p 0.3
```

4.8 Cross-validation (-xv)

When comparing models with parameters estimated under the penalized likelihood, the significance cannot be judged by standard means. Instead to estimate a cross-validation likelihood, use the -xv flag. For example:

```
>fgwas -i input_file.gz -print -p 0.3 -xv
```

4.9 Only doing penalized likelihood estimation (-onlyp)

In some cases (for example, when testing different penalized likelihoods) it is useful to skip the maximum likelihood estimation altogether and only perform estimation under the penalized likelihood. To do this, use the -onlyp flag. For example:

```
>fgwas -i input_file.gz -print -p 0.3 -onlyp
```

4.10 Conditional analysis (-cond)

It is often useful to test whether a given annotation adds information above and beyond that captured by a model. One approach to this is to estimate the parameters of a model, fix them to their

maximum likelihood values, and then estimate an additional parameter and see if the confidence intervals on this added parameter overlap zero. To do this, use the `-cond` flag. For example, the following command with estimate an enrichment parameter for annot1, fix it to its maximum likelihood value, and the estimate an enrichment parameter for annot2:

```
>fgwas -i input_file.gz -w annot1 -cond annot2
```

5 Output files

The main stem of the output files is set by the `-o` flag and defaults to "fgwas". Under the default settings, the output files will be:

1. `fgwas.params`. The maximum likelihood parameter estimates for each parameter in the model. The columns are the name of the parameter ("pi" is the parameter for the prior probability that any given genomic region contains an association), the lower bound of the 95% confidence interval on the parameter, the maximum likelihood estimate of the parameter, and the upper bound of the 95% confidence interval on the parameter.

2. `fgwas.llk`. The likelihood of the model. The lines are the log-likelihood of the model under the maximum likelihood parameter estimates, the number of parameters, and the AIC.

If you have used the `-print` flag to print the posterior probabilities of association, the model will additionally output:

1. `fgwas.ridgeparams`. The estimates of the parameters under the penalized likelihood. The first line of this file is the penalty used, then the penalized parameters estimates follow. If you have used the `-xv` flag, the last line will be the cross-validation likelihood.

2. `fgwas.segbfs.gz`. The association statistics in each region of the genome defined in the model. The columns of this file are the block number, chromosome, start position, end position, maximum absolute value of Z-score, log regional Bayes factor, regional prior probability of association, log regional posterior odds for association, and the regional posterior probability of association. The annotations of the region (if any) are in the remaining columns.

3. `fgwas.bfs.gz`. The association statistics for each SNP in the genome as estimated by the model. The columns of this file are the SNP ID, the chromosome, genomic position, log Bayes factor, Z-score, estimated variance in the effect size, prior probability of association, two columns (pseudologPO and pseudoPPA) for internal use only, the posterior probability of association (conditional on there being an association in the region), and the region number. The annotations in the model (if any) then follow.

6 Suggested workflow

The standard situation is that we have a GWAS for a given trait and a list of hundreds of annotations that are potentially enriched or depleted for associations. We suggest the following:

1. Test each annotation individually:

```
fgwas -i input_file.gz -w annotation1 -o annotation1
fgwas -i input_file.gz -w annotation2 -o annotation2
fgwas -i input_file.gz -w annotation3 -o annotation3
...
```

2. Find the annotation that most improves the likelihood of the model, then test it in combination with all other significant annotations (imagine in this case that annotation 1 gives the best likelihood):

```
fgwas -i input_file.gz -w annotation1+annotation3 -o annotation3
fgwas -i input_file.gz -w annotation1+annotation8 -o annotation8
fgwas -i input_file.gz -w annotation1+annotation10 -o annotation10
...
```

3. Keep adding annotations in this manner until there are no annotations that improve the likelihood.

4. Switch to using the cross-validation likelihood, and find the penalty with the best cross-validation likelihood:

```
fgwas -i input_file.gz -w annotation1+annotation8 -p 0.05 -xv -print -o p05
fgwas -i input_file.gz -w annotation1+annotation8 -p 0.10 -xv -print -o p10
fgwas -i input_file.gz -w annotation1+annotation8 -p 0.25 -xv -print -o p25
...
```

5. Test dropping each annotation from the model, using the cross-validation likelihood (imagine that a penalty of 0.1 gave the best cross-validation likelihood):

```
fgwas -i input_file.gz -w significant_annotation1 -p 0.10 -xv -print -o drop1
fgwas -i input_file.gz -w annotation8 -p 0.10 -xv -print -o drop8
```

6. Keep dropping annotations as long as the cross-validation likelihood keeps increasing.

7. Analyze the model with the maximum cross-validation likelihood.