
Getting Started with GeneRecon

An introduction to the association mapping tool GeneRecon

Thomas Mailund and Leif Schauser

mailund@birc.au.dk
schauser@bioinformatics.dk

Copyright © 2006 Thomas Mailund and Leif Schauer • Bioinformatics ApS

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved in all copies.

BETA RELEASE OF GENECON

*This manual describes a **beta release** of the association mapping tool GeneRecon. As it is a beta release, the tool is still somewhat limited in functionality, and the interface is likely to change before the initial release.*

With this warning, should you still decide to use it, we would greatly appreciate any comments you have on it.

About GeneRecon

GeneRecon is a software package for linkage disequilibrium mapping using coalescent theory. It is based on Bayesian Markov-chain Monte Carlo (MCMC) method for fine-scale linkage-disequilibrium gene mapping using high-density marker maps. GeneRecon explicitly models the genealogy of a sample of the case chromosomes in the vicinity of a disease locus. Given case and control data in the form of genotype or haplotype information, it estimates a number of parameters, most importantly, the disease position. Some of the theory underlying GeneRecon is described in:

Rannala, B. and Reeve, J.P. (2001) *High-resolution multipoint linkage-disequilibrium mapping in the context of a human genome sequence*. Am. J. Hum. Genet. 69: 159–178.

Morris, A.P., Whittaker, J.C., and Balding, D.J. (2002) *Fine-scale mapping of disease loci via shattered coalescent modeling of genealogies*. Am. J. Hum. Genet. 70:686–707.

Rafnar, T., Thorlacius, S., Steingrimsdottir, E., Schierup, M.H., Madsen, J.N., Calian, V., Eldon, B.J., Jonsson, T., Hein, J., and Thorgeirsson, S.S. (2004) *The Icelandic Cancer Project — A population-wide approach to studying cancer*. Nature Reviews Cancer. 4:488–492.

GeneRecon is written in C++ and is available as a command-line executable for Linux. The Scheme programming language is used for specifying the input to GeneRecon and controlling its execution.

Installing GeneRecon

GeneRecon is distributed either as an RPM file containing a binary version of the program, compiled to an Intel x86 Linux platform or as source code. It has been tested on the RedHat 9, Fedora Core 1–4 distributions of Linux. Running GeneRecon requires that Guile Scheme version 1.6.4 or newer is installed. To install GeneRecon from the RPM package, run

```
> rpm -Uvh generecon-x.y.z-r.i386.rpm
```

where `x.y.z-r` is the version number of GeneRecon. Since it installs in the directory `/usr/local/`, installing the RPM package requires root access. To install GeneRecon from source code, first uncompress and untar the file, then run `'configure'` and finally `'make'`. To test that the build was successful, run `'make check'`. To install the program, run `'make install'`.

```
> tar zxf generecon-x.y.z.tar.gz
> cd generecon-version
> ./configure
> make
> make check
> make install
```

Before using GeneRecon *it is necessary that you have installed the program*, not just compiled the program; at least if you are using any of the scheme modules distributed with GeneRecon. If the tool has not been properly installed, it will not be able to locate the modules and the scripts, including several of the included test scripts, will not be able to run. If it is not possible to install the modules globally, you can install the locally but you will then need to set the `GUILE_LOAD_PATH` environment variable to let GeneRecon know where the modules are installed. Consult the Guile manual for further details (<http://www.gnu.org/software/guile/docs/>).

Running GeneRecon

GeneRecon is started from the command-line; the specifics about the data-set to analyze and how it is to be performed, is described in one or more configuration scripts, which are written in the Scheme programming language, see </usr/local/share/generecon/doc/index.html> for documentation. Starting GeneRecon with the configuration script `analysis.scm` is done as:

```
> generecon analysis.scm
```

Run `generecon --help` to get a complete list of command-line options accepted by GeneRecon.

Using GeneRecon

Included in the GeneRecon distribution are examples of GeneRecon configuration scripts, see </usr/local/share/generecon/test-input/>. Included are examples of both haplotype and genotype datasets.

Haplotype Data

The data used in the haplotype examples is the Cystic Fibrosis data from Kerem et al. (1989). Two GeneRecon configuration scripts exemplify an analysis of this data. The first script, `cf.scm`, has the data embedded in the script. It specifies that GeneRecon should run for 10,000 iterations and generate two output files: `locus.data` containing the data for the disease position estimate, and `likelihood.data` containing the likelihood of the parameters in every tenth iteration of the underlying MCMC. The second script, `cf-load.scm`, reads the data from two file, `cf-affected.data` and `cf-unaffected.data`, and specifies that GeneRecon should run for 3,000 iterations and that diseases locus samples and their likelihood values are written to the terminal (`stdout`).

Genotype Data

The data used in the genotype examples are simulated using the CoaSim tool available from <http://www.daimi.au.dk/~mailund/CoaSim/>. Two configuration scripts, `simulated-genotype.scm` and `simulated-genotype-load.scm`, exemplify an analysis of this data, and show how genotype data can be embedded into the script or read from separate files.

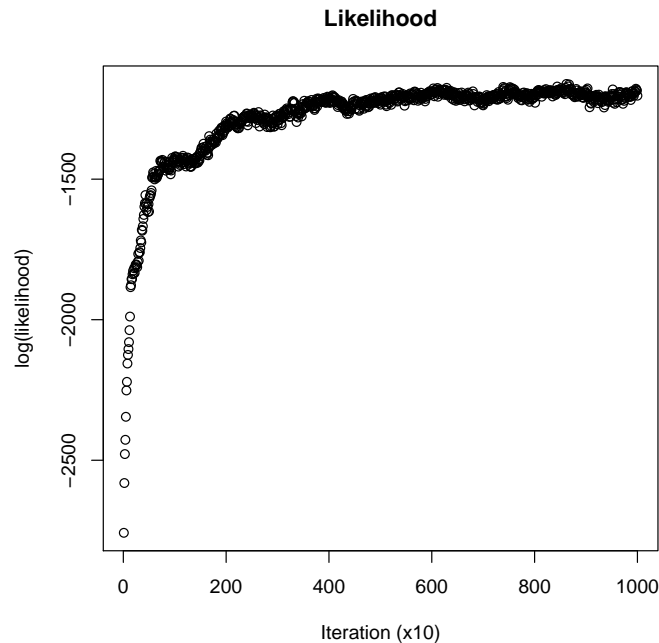


Figure 1: The likelihood as a function of the number of iterations.

Examining the Output

Running GeneRecon on the `cf.scm` example configuration scripts as

```
> generecon cf.scm
```

produces two output files, `likelihood.data` and `locus.data`, described above. We can use the R package (see <http://www.r-project.org>) to visualize the output from this GeneRecon computation. We can plot the likelihood as a function of the number of iterations as:

```
> likelihood <- scan(file="likelihood.data")
> plot(likelihood)
```

The result is shown in Figure 1. Plotting the disease position as a function of the number of iterations:

```
> locus <- scan(file="locus.data")
> plot(locus)
```

The result is shown in Figure 2.

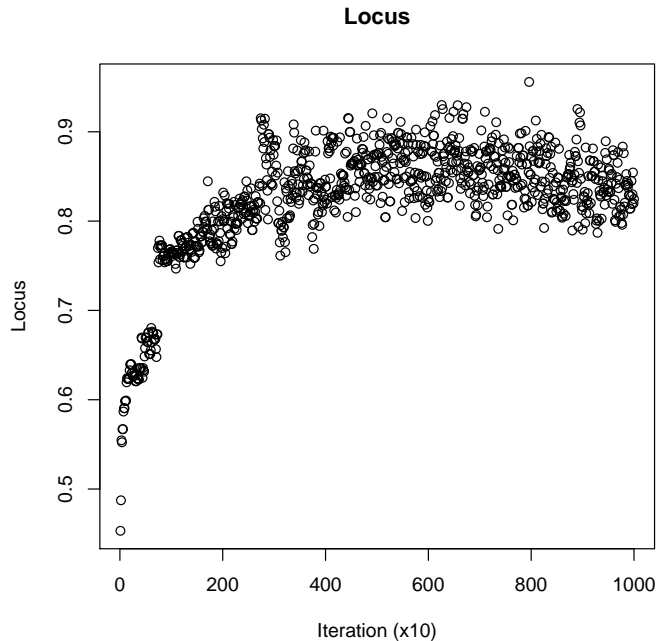


Figure 2: The disease position as a function of the number of iterations.

The probability of a disease causing mutation at a given position is proportional to the number of times the MCMC has visited that position after an initial burn-in period. The posterior distribution is obtained by binning the sampled disease positions. Several bin sizes should be tried when displaying the data, in order to avoid binning-artifacts. Using R we can bin the values using the function `hist` (for histogram), that both performs the binning and plots a histogram of the bin counts

```
> hist(locus)
```

see Figure 3. Alternatively we can plot the posterior density using the density function

```
> plot(density(locus), xlim=c(0,2))
> positions <- c(0, 0.009, 0.0248, 0.5248, 0.5348, 0.5548, 0.5698,
+             0.5948, 0.6148, 0.6198, 0.6548, 0.6848, 0.7098,
+             0.7448, 0.7798, 0.8598, 0.8698, 0.8898, 0.9048,
+             0.9598, 1.5598, 1.6298, 1.7298)
> rug(positions)
```

see Figure 4. Here we also indicated the positions of the markers using the `rug` function.

The Markov Chain only samples from the posterior distribution when mixing properly. Mixing ensures a thorough search in the parameter space and minimizes the likelihood of being trapped in local maxima. The mixing behavior of

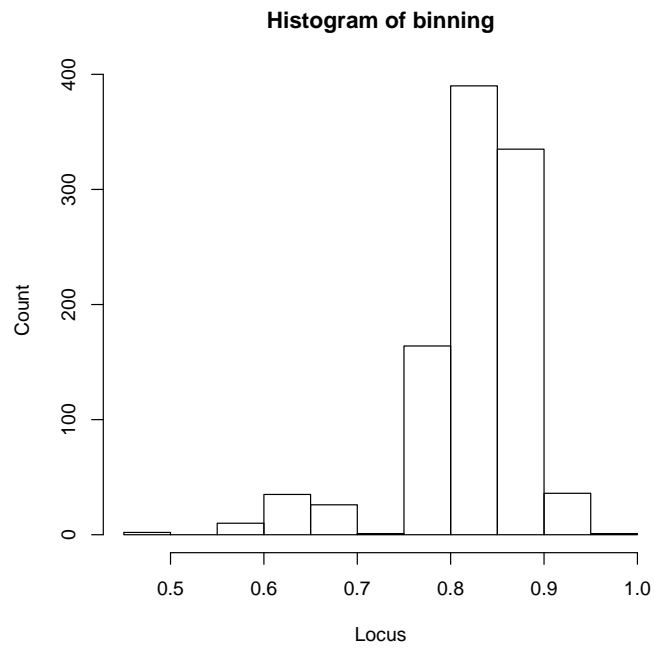


Figure 3: Histogram of the counts of binned positions.

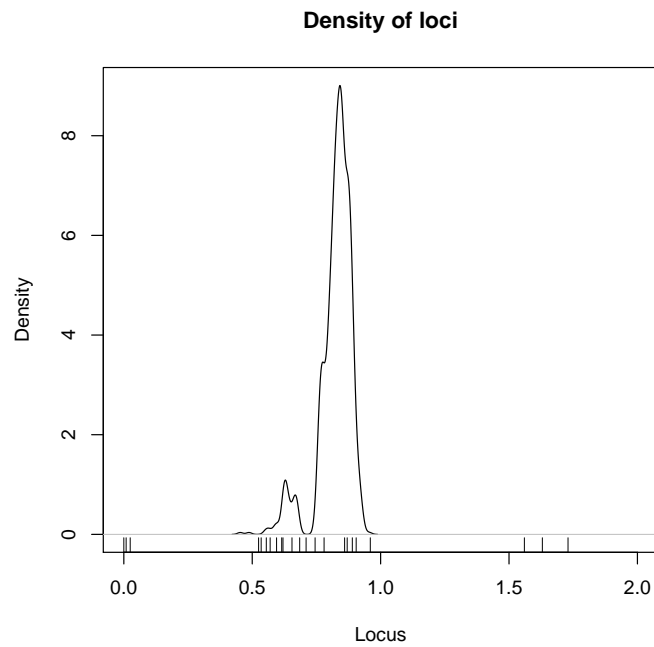


Figure 4: The posterior density of the disease locus.

the Markov Chain can currently be estimated by visual inspection of the output after a burn-in period.

To ensure that the results of running GeneRecon are correct, it is usually necessary to run several chains and verify that they converge to the same posterior distribution—if the result of running the MCMC is random, the resulting posterior distributions should be random, whereas if the resulting posteriors agree, they are likely to represent a true signal in the data.

Contact

For any comments or questions regarding GeneRecon, please contact Thomas Mailund, at mailund@mailund.dk or mailund@birc.au.dk.