Rejector2

Release Notes

Matthew Jobin

Department of Anthropology
Stanford University
Stanford, CA
94305, USA

mjobin@stanford.edu

http:// www.rejector.org

0. Before you begin!

REJECTOR2 is a new version of REJECTOR, my software package for inference of population history using genetic data. These are preliminary notes, so I have simply outlined here what you need to know that is different from the original. To use REJECTOR2, you will need to have a copy of the Rejector User Guide on hand (it is included with this release of REJECTOR2, in the /docs folder). I have outlined below what has changed in this release, including what is needed to make your input files compatible with REJECTOR2.

0.1 Overview of differences from Rejector

REJECTOR2 represents a significant internal rewrite to the software to streamline the process of inference. I have done my best to change as little as possible to the input and output of the software, so that those familiar with it will not have to make many modifications. An over view of the changes is listed below:

- Rejector, Rejstats and Prior are now one executable
    - REJECTOR2 now performs all the funcitons of the above three programs from the original REJECTOR release
    - To access the separate summary statistic calculation funcitons of Rejstats, invoke REJECTOR2 with the –rs switch
    - The only other executable you need is your coalescent simulation
- REJECTOR2 works with the coalescent simulators msHOT and MaCS (SNPs only) (see page 9)
- REJECTOR2 can now handle much larger subsets of data
    - The data model of REJECTOR2 has been rewritten to handle much larger numbers of loci
    - NOTE: While REJECTOR2 can deal with 500,000 or more loci, be sure that the simulation you use can also handle this much information
        - msHOT is designed to work with segments of DNA, but REJECTOR2 does not yet know how to pass this information (should be added soon)
        - Simcoal2 can work with STRs, but in my tests cannot handle more than a few thousand loci
- REJECTOR2 does NOT work with multiple CPUs or over Xgrid (yet)

1. Installation and Quick Start

1.1 Installing under Mac OS X

Please ensure that you have already installed Apple's Developer Tools, which come with your copy of Mac OS X. For OS X 10.5 Leopard, they reside on the same DVD as the operating system install, under "Optional Installs".

When you download REJECTOR2 from the website (the link resides on the left side of the page), you will receive a file called `rejector2.zip`. You can expand this file into the `rej2` folder simply by double-clicking.

Take the folder named `rej2` created by this action, and place it somewhere on your system convenient to you.

You will now need to access the folder from the command line. If you are using a Macintosh, you can find the program that gives you access to the command line in your Utilities folder, which is in your Applications folder. The program is called Terminal. If you double-click it you will see a terminal window appear.

The standard install of Mac OS X will begin with your terminal pointing to your Home folder. To navigate to the `rej2` folder, use the `cd` and `ls` commands. To list the contents of the folder you are in, type `ls` and hit return:

```
ls
```

To change into a directory inside the one you are in, type:

```
cd <directory-name>
```

To go up to the parent directory of your current folder, type:

```
cd ..
```

Using these you should be able to get to the `rej2` directory. For further tips on command-line navigation, consult any book on UNIX basics, or search online.

Once you are in the `rej2` directory, you can type `ls` to ensure the contents are there. You should see these directories and files:

`code` -> all of the program files necessary to create the program
`rej2examples` -> a folder with example input files
`Makefile` -> the file that controls the building of the software
`scripts` -> PERL, shell, and R scripts for automation and post-processing of data
`rejexample.txt` -> an example input file for initial testing purposes
`rejclean` -> shell script for clearing out intermediate files

The `code` folder contains all of the program files necessary to create the executables. To compile the software, you only need to type one command:

```
make
```

This command will commence compilation of REJECTOR2. When the compilation is done, four executable files will appear in the `rej2` directory:

```
rejector2
simcoalrej2_1_2
```

If you wish to use simcoal2 as your coalescent simulation, make sure it stays in the same directory as the REJECTOR2 executable, and that both executables are in the same folder as your input file. To use REJECTOR2 with msHOT, see page 9 below.

1.2 Installing under Linux

Please ensure that your Linux system has the following installed: `g++, gcc, make, perl, unzip`. These are standard for most Linux varieties.

Most graphical interfaces will expand this file into the `rej2` folder simply by double-clicking. If you are using a purely command-line variant of Linux, you will simply need to navigate to the folder containing the `rejector2.zip` file and type:

```
unzip rejector2.zip
```

The rest is very similar to the Macintosh version above. Download the folder, then consult the Macintosh install section if you need to in order to navigate to the folder and `make` the executables.

NOTE: If, when compiling under Linux, you get run-time errors that seem to complain about being unable to find some version of GLIBC, it is possible your system is not configured correctly. Ensure your Linux machine has `gcc` installed (this is almost always true). You can find out where it is using:

```
locate gcc | less
```

If your system does not have `locate` installed, search manually in `/usr/lib` or `/usr/lib/local/`. Consult a guide on using Linux if still unsure. Once you know where `gcc` lives, you can set a path to it using:

```
setenv LD_RUN_PATH <gcc location>
```

Then try again using `make`.

1.3 Installing under Windows

NOTE: I have not yet tested REJECTOR2 under Windows! I expect it to behave similar to REJECTOR. Please let me know if you encounter problems.

In order to use REJECTOR2 in Windows, you will need to install the free Cygwin environment that allows the compilation and execution of Linux-like programs. Visit the Cygwin website:

http://www.cygwin.com

Click the "Install or update now" link in the middle of the page and run the software when requested.

It's easiest to choose to "Install from Internet".

Install Cygwin where you like. The default "C:\cygwin" is a good choice. We will assume that's where you have put it in the following instructions, so if you have not, adjust accordingly. It's best to install for "All users" if possible.

When you see the "Select Packages" window, click the "View" button in the top right corner until it says "Full" next to it. This displays all the packages that can be installed with Cygwin. We only need a few.

Find the entries for the following packages:

```
gcc-core
gcc-g++
make
perl
unzip
```

Click the labels "Skip" next to these entries – they should turn into numbers (e.g. as of this writing 3.4.4-3 for gcc-core, but they will increase over time.). These selections will auto-select other parts of the list – that's normal.

Hit "Next" and let the packages download and install.

A folder named "cygwin" has been created, defaulting to C:\cygwin.

To test the software, from your Start menu find Cygwin->Cygwin Bash Shell. This will start a command-line shell pointing to your Cygwin home directory, which defaults to C:\cygwin\home\<your user name>

Now go to the rejector site at:

http://www.rejector.org/dev

Click on rejector2.zip and Save it to your Cygwin home directory (as defined above).

From the Cygwin Bash Shell, type

```
ls
```

You should see the file rejector2.zip in the listing, as you just placed it there. Now type:

```
unzip rejector2.zip
```

to unzip the file, which places the `rej2` directory in the same directory you are in. Now type:

```
cd rej2
```

to enter that directory. If you want, type `ls` again to see the various `rej2` subfolders – `code, docs, examples`, etc. To compile the software, you should then type:

```
make
```

Your computer will spend some time compiling the software – when it is done, the executables `rejector2.exe and simcoalrej2_1_2.exe` should have been added to the directory. If you wish to use simcoal2 as your coalescent simulation, make sure it stays in the same directory as the rejector2 executable, and that both executables are in the same folder as your input file. To use REJECTOR2 with msHOT, see page 9 below.

After testing your installation (see pg. 8), you will be able to look at output with Excel or a similar spreadsheet, prepare data for R using Cygwin's perl installation, and visualize data by downloading the Windows version of R. If you are not familiar with working in Cygwin or other UNIXy environments, pick up an introductory book at your local bookstore or check around online.

NOTE: For all of the examples below, Windows users must add an ".exe" to the end of the executable names under Cygwin. This means:

`prior` becomes `prior.exe`
`rejstats` becomes `rejstats.exe`
`rejector` becomes `rejector.exe`
`simcoalrej2_1_2` becomes `simcoalrej2_1_2.exe`

1.4 Testing your Installation

A single input file, `rejexample.txt`, is included in the top `rej2` directory, with your new executables, so that you can test the software right after compilation. As soon as the software compiles, make sure you are still in the `rej2` directory and type:

```
./rejector2 -f rej2example.txt -r 10
```

This should run for about a minute. If it completes without an error, the executables are working properly. You can open up the sole output file, `rejexample.rej0.txt`, but with only 10 runs there may not be anything in it.

To test the other example input files, move the from the `examples` folder into the top-level `rej2` folder. For example:

```
mv examples/rej2dna.txt  .
```

from the top-level `rej2` folder, this will move the `handtest.txt` folder up to the `rej2` folder. Once there, you can run it by typing:

```
./rejector2 -f rej2dna.txt -r <the number of runs you
would like>
```

The general case for most `rejector2` runs is:

```
./rejector2 -f <input file> -r <desired number of runs>
```

1.5 Using with msHOT

The software msHOT is designed to rapidly simulate large amounts of SNP data under a neutral model (Hellenthal and Stephens 2006). It is an extension to Richard Hudson's widely-used simualtion ms (Hudson 2002), with the added features for handling crossovers and gene conversion hotspots. Please note, however, that for the present I have not taken advantage of these features in Rejector2 – I merely am using msHOT from here on out to simplify things should I add these features in the future.

NOTE: The msHOT software is very fast, but it ONLY works with SNP data. If you want to use Rejector2, you must use it with the default simulator, Simcoal2.

1.5.1 Installing msHOT

I have not included msHOT with Rejector2, as I have not needed to make any modifications to its code. You can find the source code and installation instructions for the software at: http://home.uchicago.edu/~rhudson1/source.html . Please download and

install the software as instructed, then move the completed executable msHOT into the rej2 directory. Use the mv command as demonstrated in the above sections.

1.5.2 Testing msHOT

To use msHOT with REJECTOR2, you must add the –ms switch to the command line, and you must also ensure that you ONLY use SNP data in your input file. I have provided an input file for use in testing that msHOT works with your copy of REJECTOR2.

Once you have msHOT in the same directory as REJECTOR2, move the file rej2snps.txt from the `examples` folder into the top-level `rej2` folder.

```
mv examples/rej2snp.txt  .
```

Then test with the follwing line:

```
./rejector2 -f rej2snps.txt -r <the number of runs you
would like> -ms
```

If all is in place you should see the simulation running and completing. I have tested msHOT with REJECTOR2, and found that using a few summary stats (e.g. HeterozygosityAv, FstAv and Nei), REJECTOR2 can run quite manageably with as many as 500,000 SNPs. See page 12 for how to create input files.

1.6 Using with MaCS

MaCS is a simulator of the coalescent process that simulates geneologies spatially across chromosomes as a Markovian process (Chen, Marjoram et al. 2008). It is especially useful for simulation of large numbers of loci in reasonable time frames. NOTE: REJECTOR2 does not support recombination hotspots or ascertainment bias in MaCS. Most of the demographic parameters used by MaCS are identical to those used in msHOt and ms (see above).

1.6.1 Installing MaCS

You can find MaCS at: http://www-hsc.usc.edu/~garykche/

Please note that, as indicated the site, you will need to have the Boost libraries installed to compile MaCS. Many versions of Linux come with Boost installed, but Mac OS X does not.

1.6.2 Working with MaCS

To use MaCS as your simulator, put the –macs switch in the command line. For example:

```
      ./rejector2 -f rej2snps.txt -r <the number of runs
you would like> -macs
```

Make sure MaCS is in the same directory as REJECTOR2. Please note: you do NOT need to use the msformatter program included with MaCS as of the latest release!

2. Structure of Input Files

Important: **There are subtle changes to these input files compared to** REJECTOR **input files! Please consult these instructions to create or edit files for** REJECTOR2**. See also page 26 for using the makerej2in.pl to make "dummy" input files to examne and modfy their structure.**

REJECTOR2 input files are tab-delimited text files. An easy way to create and modify them is in Microsoft Excel, or another similar spreadsheet application which can work with text files.

The input file format is quite scalable, allowing large numbers of loci of many different conformations, but REJECTOR2 must be accurately told how the data is structured to avoid errors. The input file is broken into sections, outlined below. Pieces of an Example file have been included in each section, and then again as one piece on page 22.

2.1 Heading Line

The first line of the infile must be `REJECTOR`, all capitals. Otherwise the program gets all confused.

```
REJECTOR
```

2.2 Number of Populations

You must specify the number of populations next, and it MUST match the number of population priors and data below it! The section should take the form:

```
Number of Populations
3
```

The above would tell the program to expect you to refer to three populations in the priors and data section.

3.3 Priors

The section beginning with `/--Priors` and up until `/--Statlist` defines the priors used in the program. These contain prior information that you can give to REJECTOR2 in order to influence the posterior distributions it calculates. The essence of REJECTOR2 is the input of sensible priors (or wide uniform priors if no good priors exist) for parameters of interest, so that you can estimate these parameters with the rejection algorithm.

The priors describe the type of population model you wish to test. The entries are structured using the time-backwards approach common to coalescent simulations – in other words you begin in the present and work backwards in time. The Priors section is structured fairly similarly to the input files for SIMCOAL2, so you can find additional information about them there (http://cmpg.unibe.ch/software/simcoal2/) – one of the biggest differences is that for every numerical entry in a SIMCOAL2 input file, there are two input values for REJECTOR2 and one word noting the sort of prior distribution these two values describe. The following is an example Priors section. Please consult the Rejector Use Guide for more information on the construction of priors.

```
/--Priors
Population Sizes
uniform
100
10000
uniform
100
1000
uniform
100
1000

Growth Rates
uniform
0
0
uniform
0
0
uniform
0
0

Number of Migration Matrices
0

Number of Historical Events
1
uniform
2000 0 1 1.0 1 0 0
15000 0 1 1.0 1 0 0

Microsat Range Constraint
uniform
5
25
```

Each entry consists of three parts, a parameter that describes the shape of a distribution and two numerical parameters. There are three distribution shapes available in this release of REJECTOR:

• uniform: The two numerical parameters describe the minimum and maximum of a uniform distribution, respectively, between which all values are equally likely to be picked.

• gaussian: The first numerical parameter describes the mean, and the second the standard deviation, of a gaussian distribution.

• gamma: The first numerical parameter describes the scale, and the second the shape, of a gamma distribution.

Please note that REJECTOR2 will only allow sensible values for any prior value. For example, if you specify a gaussian distribution for the size of a population, REJECTOR2 will ignore (by resampling until it gets a positive number) random deviates that set the population size to zero or less, as such a value will be nonsensical to SIMCOAL2.

If you want any entry to have a fixed value, in other words to not vary from iteration to iteration, you can do so by giving it a uniform distribution with the minimum and maximum values the same. For example, if you wanted to tell REJECTOR2 that the size of a population was exactly 1000, you would enter it as below:

```
uniform
1000
1000
```

2.3.1 – Assignments to Population by Alphabetical Order

Before we get into the pieces of the input file, a word about assignment to population. The Population column in the Data section (see pg. 20) defines which population each entry is assigned to. The number of entries for Population Sizes, Sample Sizes, and Growth Rates MUST exactly match the number of populations you have listed in your Data section, and defined under Number of Populations. For example, if in all of your data entries you have listed one of the following names: Adygei, Balochi, Burusho - then you MUST have three entries for Population Sizes, Sample Sizes, and Growth Rates. In addition, if you add a migration matrix, it must be as long and as wide as there are populations in the Data column.

The entries for Population Sizes, Sample Sizes, and Growth Rates are assumed to be in alphanumeric order relative to the Population column of your data, In other words, if in all of your data entries you have listed one of the following names: Adygei, Balochi, Burusho, then the first entries for Population Size will be taken for Adygei, the second for Balochi, and the third for Burusho. The same goes for Sample Sizes and Growth Rates. If we count upward from 0, then the Adygei are Population 0, Balochi are Population 1 and Burusho are Population 2.

2.3.2 – Population Sizes, Growth Rates

The entries under Population Sizes and Growth Rates each describes these values for one of populations you have listed in the Data section (see above for how they are ordered.) The population sizes are the size of the whole population, while Sample Sizes are the individuals the programs tracks during run time.

Growth rates define the exponential parameter *r* in the equation:

$$N(t) = N(0)e^{rt}$$

IMPORTANT: Because we are working *backwards* from the present in REJECTOR2, giving a *negative* growth rate means population *expansion*, forward in time.

2.3.3 - Migration Matrices

The example above gave 0 migration matrices, meaning no migration will occur. If you wish to use migration matrices, you can list them in the form shown in the example below:

```
Number of Migration Matrices
2

uniform
3
0.0 0.0 0.01
0.0 0.0 0.01
0.01 0.0 0.02

0.0 0.0 0.03
0.0 0.0 0.03
0.01 0.0 0.05

gaussian
3
0.01 0.01 0.01
0.01 0.01 0.01
0.01 0.01 0.02

0.001 0.001 0.001
0.001 0.001 0.001
0.001 0.001 0.002
```

The first number after the word "Matrices" tells REJECTOR2 how many matrices to read. For each matrix it then reads the distribution type. The two matrices that follow are matrices of the two numerical parameters for the distribution type – minimum and maximum for uniform, and so on. The matrices are given id numbers starting with 0 for the first one and given to SIMCOAL2 in the order they are read. The first migration

matrix (matrix 0) is always the first one used – the others can only be switched to using a historical event (see below).

2.3.4 – Historical Events

Using a historical event, you can set a time (going backwards in generations from the present) when a population diverges, grows or shrinks, or when the simulation changes migration matrices. Historical events follow the style in the SIMCOAL2 guide (http://cmpg.unibe.ch/software/simcoal2/), save that there are two lines, representing the two sets of numerical parameters for one of the three distribution types. They are of the form:

```
uniform
5000 0 1 1 1 0 0
6000 0 1 1 1 0 0
```

In each line of numbers there are seven entries. This what each of them means:

1st number: When the event occurs, in generations *backwards* from the present.
2nd number: The source population of any migrants.
3rd number: The sink population (destination) of any migrants.
4th number: The proportion of migrants in source going to sink. 1 means all.
5th number: The relative new size of the sink. 1 means same size.
6th number: The new growth rate of the sink.
7th number: The new migration matrix to use.

The entries are all doubled to allow them to become priors, just like the entries for Population Size etc. above. So the example above says:

Between 5000 and 6000 generations ago, on a uniform distribution, all of the lineages from Population 0 move to Population 1 (going *backwards* in time). Population 1 stays the same size prior to this. The new growth rate is 0 and use migration matrix 0.

Bear in mind some parts of a historical event cannot really have priors. The source, sink, and new migration matrices cannot have priors associated with them, as there is no sensible way to compute this.

2.3.5 Historical SNPs.

NOTE: This WILL NOT WORK with more than one independent loci or with no recombination (FOR NOW.. This is a HACK).

I have made an addition to SIMCOAL2 to allow the user to specify exactly when, and in what deme, a SNP occurs. In this case, the numbers for the historical event mean slightly different things, as follows:

1st number: When the event occurs, in generations *backwards* from the present.
2nd number: The source population of any migrants.
3rd number: MUST BE -1, to denote this as a SNP event.

4th number: Number of the chromosome (ALWAYS 0 for now).
5th number: Number of the locus
6th number: The new growth rate of the sink.
7th number: The new migration matrix to use.

For each SNP to force, you will also need to set the SNP's Mutation rate to -1 to shut of SIMCOAL's normal SNP mutation function.

2.3.6 – Mutation Rates and Microsat Range Constraints

NOTE: The entry for Mutation rates is no longer used and has been removed! Make sure this entry is not there if you are converting from a REJECTOR input file.

Microsat Range Constraints define how many "steps" in size a microsatellite is allowed to take while mutating. This is used to constrain the microsatellite to biologically-realistic behavior. This is usually a uniform distribution, or a uniform distribution with no variation (both values the same). Note that a value of "0" here implies no constraint, so to give no constraints to microsatellite range, use:

```
uniform
0
0
```

2.3.7 Symmetrical Bottlenecks

In order to simulate bottlenecks, wherein a population drop and then recovers, I have added in a way to set thje relative change in size of a historical event to the inverse of the previous. If you have two historical events, one to start and the other to end a bottleneck, use a -1 in the relative size column of the second bottleneck:

```
uniform
5000 0 1 1 0.1 0 0
5000 0 1 1 0.1 0 0
uniform
6000 0 1 1 -1 0 0
6000 0 1 1 -1 0 0
```

This historical events tell REJECTOR2 to drop the size of deme 0 to one tenth of what it was at 5000 generations (as always, going backwards in time!). Then at 6000 generations ago, the population is changed to the inverse amount of the first: 1/0.1 or ten times as much, restoring the size to the original.

2.4 Statlist

The section beginning with /--Statlist allows the user to specify which summary statistics REJECTOR2 will calculate. You can use any number of summary statistics in the same run of the software. For a description of the summary statistics used, see the Rejector User Guide.

As of the present release, the following is a list of all calculations REJECTOR2 will recognize. Note that the spelling and capitalization is exact – an unrecognized name will result in an abort of the program.

```
Heterozygosity
HeterozygosityAv
HeterozygositySub
HeterozygosityAvSub
DeltaMuSquared
DeltaMuSquaredSub
Td
TdSub
Dsw
Beta
BetaAv
BetaSub
BetaAvSub
LDChiSquare
RangeofLocus
SampleSize
SampleSizeSub
MaxLength
MinLength
MeanLength
NumDiffAlleles
CompleteHeterozygotes
CompleteHomozygotes
Haplotypes
Jstatistic
MicrosatVariance
DerivedFraction
NucleotideDiversity
Nei
Fst
MicrosatVarianceAv
NumDiffAllelesAv
```

```
NucleotideDiversityAv
FstAv
JstatisticAv
ModJstatisticAv
Istatistic
IstatisticAv
ModIstatisticAv
UBDeltaMuSquared
UBDeltaMuSquaredSub
LD
Hamming
HammingBtwn
NumHaploBlocks
NumROH
FROH
FROHVar
SharedHaplotypeBlocks
MinorAlleleFrequency
TajimasD
NumSegSites
LongSharedBetween
LongSharedWithin
BlockLength
BlockBoundary
```

A tolerance value must be placed next to each summary statistic name. This value is used by REJECTOR2 to perform rejection algorithm calculations.

If you wish to run all calculations, you may simply put the single entry `ALL` in the Testlist section, instead of listing all of the summary statistics. This is done in the example above. The `ALL` name still requires a tolerance value.

Note that some summary statistics require much more computation time than others. `MultAlleleLDp`, for example, often requires a large proportion of the computational resources for every run, as do the two-way statistics D_{sw}, $\partial\mu^2$, and T_D, especially if there are a large number of demes.

2.5 Data

The Data section contains both a description of the structure of the data and the data itself. Note that the data structure headers must match the actual data or errors will occur. For example, if you describe a system with one SNP and one STR and then give an entry line for the system with one SNP and two STRs the program will read all subsequent data out of order and likely crash, or at least give nonsensical results.

2.5.1 Header

The header part of the Data section begins with the Vnaught entry. This is a constant used with the T_D statistics, explained in the Rejector User Guide.

Next is a description of each locus in the data. There are four types of loci recognized by REJECTOR2:

• SNPs, or Single Nucleotide Polymorphisms, are mutations at a single nucleotide site. The four alleles possible are A,C, G, and T. Ancestral states for each SNP can be defined by a comma-separated entry in the ancestral row. You can, for example, tell REJECTOR2 that the ancestral state for a SNP is A, as done for the SNP in system X2 in the example below, or that the ancestral states are T,C as shown in the first SNP of system X1 in the example. If you define any ancestral alleles for a SNP, then REJECTOR2 assumes that all other alleles from the pool A,C,G and T are descendant. If you do not know the ancestral state for a SNP, putting NA in the entry will ensure that SNP is not included in determining if a sample is ancestral.

• UEPs or Unique Event Polymorphisms, are like SNPs, but are the more general case of any polymorphic locus assumed to happen very slowly, usually only once in the scope of time studied. Examples of UEPs include insertion-deletion polymorphisms and RFLPs. The ancestral state for a UEP works differently than a SNP. If you define an ancestral state for a UEP, REJECTOR2 assumes that all other alleles are descendant, without knowing in advance what those alleles might be. You can also use NA for a UEPs ancestral state.

• STRs, or Microsatellites, are a series of contiguous repeats of base pairs, for example ATATATATATATAT or TACTACTACTACTAC. They are known to have relatively high mutation rates (Weber and Wong 1993). Many of the summary statistics in REJECTOR2 specifically target microsatellites, such as $\partial\mu^2$ and T_D. REJECTOR2 does not make calculations based on ancestral states of STRs, so always put an NA in the ancestral state. REJECTOR2 uses a Stepwise Mutation Model (SMM) for modeling STR mutation.

• DNA is simply a string of characters. Usually used for long stretches of DNA, e.g. ACTACGATATCTAGTC.

Each locus (or loci, see NumLoci below) is listed left to right after the Loci entry. Below this entry are lines which describe each of these loci:

MutRates: The per-site mutation rate for the locus.

Ancestral: The ancestral state of the locus, if known. If you do not know the ancestral state, you may simply put NA or -1 here. If you think there are two possible ancestral states, separate them with a comma and enclose them in quotes: “T,C”.

RecombRt: The recombinaiton rate between the locus (or loci) and the next locus to the right. This is expressed as the proprtional chance per generation that a recombination event will occur between the two loci.

PhysDist: The number of base pairs between this locus and the next one. Use this for statistics that need to know the distance between loci, such as F_{ROH} (see page 31). **NOTE!** The input file can have either this entry or RecombRt, but NOT both! Edit one of them out with a text editor if needed.

NumLoci: If you wish you can specify a number of identical loci in a single column (especially handy if, say, you want 100,000 SNPs!). NumLoci will tell REJECTOR2 and your coalescent simulation to create the specified number of loci of the type specified in that column, all alike, with the same mutation rates, ancestral states, and recombinaiton rates. Please note that the recombination rate listed in the column will be applied between EACH of the individual loci, as well as to the locus immediately to the right. So if you have specified 1000 SNPs with NumLoci and a ReombRt of 0.005, the rate will be 0.005 between each of those 1,000 SNPs, as well as between the last SNP and the next locus you specify.

Length: Only needed for DNA! This specifies how many characters each locus occupies. If your DNA entry is 345 bases long, put 345 here. If you have 1,000 DNA specified in NumLoci and each is 234 bases long, put 234 here. For all data types but DNA, simply place a 1 here.

2.5.2 Data List

The data list has three column headers for the first three columns. The Tag column is usually the sample or chromosome name. The Population column is used to place the sample in a deem or geographical area. Note that an individual’s data is broken down into separate lines for different systems. For example, if the individual HGDP6546 has data for system X1 and for system X2, each system receives its own line, configured to match the structure of each system. The data is NOT appended to a single line.

After the Population column, the data is listed out in the order defined by the header for that system. Thus, if you have defined a SNP, STR, SNP, UEP, STR above REJECTOR2 will read the data in that order in the entry. If you have used NumLoci to specify 1,000 SNPs, make sure the Data section has 1,000 SNPs!

2.5.3 Missing Data

REJECTOR2 allows missing data in an entry. To mark data as missing, use NA. REJECTOR2 also accepts -1 as missing data, but this is now deprecated. Missing data will not be used in calculations.

2.5.4 Rules for Data Structure

The following rules must be followed in the data section:

- Make sure every entry is for a system listed in the header, and that the data conforms to the structure defined in that header (number of loci, order of loci).
- Within one population and one system, there can be no more than two entries with the same Tag, representing a diploid individual.
- There should be NO spaces in any entry! Avoid use of Tag or Population names like "N. America" (with a space between "N." and "America"). Before you run REJECTOR2, go through your input file and remove these spaces. Using a find-and-replace routine in Excel will work quickly, i.e. search for "N. America" and replace with "N.America" (no space) or "N_America".
- Do not leave whitespace at the end of the input file.
- Use NA as an entry for missing data.
- If you want to run two-way statistics (comparing one population to another, e.g. $\partial\mu^2$) make sure there is more than one population in the data.
- When running a subdivided statistic (see Rejector Use Guide) be aware that empty populations can be produced if none of the entries in a population match the ancestral or descendant criteria.

3.6 Example infile: rej2example.txt

```
REJECTOR

Number of Populations
5

/--Priors
Population Sizes
gamma
1000
10
uniform
100
10000
uniform
100
```

```
10000
uniform
100
10000
uniform
100
10000

Growth Rates
uniform
0
0.01
uniform
0
0
uniform
0
0.2
uniform
0
0.2
uniform
0
0.2

Number of Migration Matrices
0

Number of Historical Events
4
uniform
3500 0 1 1.0 1 0 0
6500 0 1 1.0 2 0 0
uniform
3500 4 1 1.0 1 0 0
6500 4 1 1.0 2 0 0
uniform
3500 3 1 1.0 1 0 0
6500 3 1 1.0 2 0 0
```

```
uniform
500 2 1 1.0 1 0 0
3000 2 1 1.0 1 0 0

Microsat Range Constraint
uniform
5
14

/--Statlist
Heterozygosity 0.1
HeterozygosityAv    0.1
HeterozygositySub   0.1
HeterozygosityAvSub 0.1
DeltaMuSquared 0.1
DeltaMuSquaredSub   0.1
Td   0.1
TdSub     0.1
Dsw  0.1
Beta 0.1
BetaAv    0.1
BetaSub   0.1
BetaAvSub 0.1
LD   0.1
LDChiSquare    0.1
RangeofLocus   0.1
SampleSize0.1
SampleSizeSub  0.1
MaxLength 0.1
MinLength 0.1
MeanLength0.1
NumDiffAlleles 0.1
CompleteHeterozygotes    0.1
CompleteHomozygotes 0.1
Haplotypes0.1
Jstatistic0.1
MicrosatVariance    0.1
DerivedFraction0.1
NucleotideDiversity 0.1
```

```
Nei  0.1
Fst  0.1
MicrosatVarianceAv  0.1
NumDiffAllelesAv    0.1
NucleotideDiversityAv    0.1
FstAv     0.1
JstatisticAv   0.1
ModJstatisticAv0.1
Istatistic0.1
IstatisticAv   0.1
ModIstatisticAv0.1
UBDeltaMuSquared 0.1
UBDeltaMuSquaredSub 0.1



/--Data
Vnaught   0

Loci SNP  STR  DNA  UEP  STR
MutRates  1.00E-09  2.80E-04  1.00E-08  1.00E-08  2.80E-04

Ancestral "T,C"     -1   AGCT L    -1
RecombRt  0.000042  0    0    0    0.00005
NumLoci   1    1    1    1    2
Length    1    2    4    1    5

Tag  Population
0    A    T    25   AGCT L    10000     2
1    A    -1   23   AGCT L    10000     2
2    A    C    -1   AGCT L    22   3
3    A    C    -1   AGCT L    34   5
4    A    C    22   AGCT S    22   3
5    A    T    22   AGCT -1   22   7
6    B    T    80   AGCT L    80   7
7    B    C    21   AGCT L    21   8
8    E    T    24   AGCT S    24   8
9    B    T    25   AGCT S    25   7
10   C    C    22   AGCT L    22   9
11   C    C    24   AGCC L    24   6
```

```
12    E    C    24    AGCT L    24    8
13    C    C    22    AGCT S    22    4
14    E    C    20    AGCC L    20    9
15    C    C    19    AGCT S    19    3
16    D    C    45    AGCT L    45    5
17    D    T    21    AGCT L    21    2
```

3. Working with msHOT

The software msHOT is a fast coalescent simulation program based on Hudson's ms (Hudson 2002), with the added capability of adding recombination hotspots (Hellenthal and Stephens 2006). There are a few things to keep in mind when working with msHOT, as it does not function in exactly the same way as SIMCOAL2. Both programs can be used by REJECTOR2. To use msHOT, simply add the –ms switch to the REJECTOR2 command line.

3.1 Recombination

REJECTOR2 passes recombination to msHOT via ms's –r switch. Please see the documentation for ms for details. A ρ value is calculated for the sequences given based on the physical distances input into REJECTOR2, and this average value is used to simulate recombination. Any gap of greater than 500,000 bases is skipped in this calculation, however, and is instead assigned as a hotspot. The increased chance of recombination between the bases flanking the long gap is calculated and input into msHOT at a rate proportional to the length of the gap relative to the average length between loci used to calculate the chance of recombination over the gap.

4. Tools

Note that the post-processing and visualiztion tools provided with REJECTOR2 work the same as they did in REJECTOR. For now, please consult the Rejector User Guide for hep in using them.

4.1 make2rejin.pl – Making dummy input files

You might find it useful to create dummy input files, either for testing purposes or to create a template. The make2rejin.pl script will do this for the Data section of the file. Create a text file with the priors and statistics you would like to use, then invoke the script:

```
perl make2rejin.pl <priors and statistics file> <output
file>
```

The script will ask you about the details of the data you would like, then print a file full of random data of the requested type. You can use the included file `pt2.txt` as an example priors and statistics file.

4.2 runrej2 – Run REJECTOR2 on multiple input files

You can use the shell script runrej2 to run REJECTOR2 on multiple files in a directory. This very simple script attempts to run REJECTOR2 on every file in the folder with the extension .txt, so make sure you have only valid input files in the folder before you begin. It is mainly used to test REJECTOR2, and so specifies only 10 iterations. You can change this by editing the runrej2 file using a text editor.You can invoke the script using the following command:

```
./runrej2
```

4.3 findoffset.pl – Prepare raw microsatellite lengths for use in Rejector

This perl script is meant to work in conjunction with struconvert.pl below. If you have a table of microsatellites in STRUCTURE format, but what you have are raw lengths (i.e. not the number of repeats, and often with sequence on either end) you can use this script to convert the data into repeat lengths. For this you will need the STRUCTURE data file and a file of the type liex.txt in the example. This file can be hand-built to include custom microsatellite motifs (i.e. bases per repeat), mutation rates, recombination rates etc. If you do not want any of these to be different per locus, just fill in a value down the columns, but here you may hand-code in any information you may have about varying mutation rates. You invoke the script like so:

```
perl findoffset.pl  <# id columns before the data> <0
for haplotypic, 1 for genotypic> <Specific structure-
style file> <Tab-delimited Locus information file of
format like liex.txt>
```

Here you can tell the script how many columns of description com in front of the data. If you invoke it:

```
perl findoffset.pl 4 1 strucfile.txt liex.txt
```

then you are telling the script that the first four columns of the STRUCTURE file are not data, that the data is genotypic (meaning that each locus gets TWO adjacent columns) and then the files. The output file will have filled-in values for an offset (i.e. the smallest findable microsat length for each locus) in a liex-style file that in this case will be called:

```
liex-for-strucfile.txt
```

Bear in ind these offsets are SPECIFIC to each STRUCTURE file – do not just use them any old where! If you are unsure, assemble your motif and other data and run it again, creating a new offset file.

4.4 struconvert.pl – convert STRUCTURE files for use in Rejector

```
perl struconvert.pl <# id columns before the data>
<population column> <0 for haplotypic, 1 for genotypic>
<0 for no division by motifs, 1 for division by motifs>
<infile>
```

If you invoke the file using

5. Ways to invoke Rejector2

In addition to the switches listed in the REJECTOR guide, in REJECTOR2 there is one new switch to control program flow and output.

5.1 –ks: Keep summary statistics

If you use the –ks switch on the command line, then for every line of output the summary statistic values calculated for that iteration of the program will be listed at the far right of the output file. Combining this with the –p switch will ensure that all summary stats for all iterations are retained – but bear in mind that with many populations and many statistics this makes for large output files.

5.2 –g: Genotypic data

Rejector2 can now send and read genotypic data to the three coalescent simulation programs it can use. To set the software using genotypic data (especially useful for statistics which require comparison between homologous stretches of DNA in a diploid individual) put the –g switch on the command line.

5.3 –snp: Reveal actual SNP time in SIMCOAL2

If Rejector2 receives the –snp switch, then it will allow the version of SIMCOAL2 packaged with this distribution to report the generation in which a SNP occurs. This may differ from the requested SNP timing in some cases, depending on the tree constructed for that iteration of SIMCOAL2.

5.4. –a and –n: Multiple concurrent averaging runs

Rejector2 will allow multiple versions of an avergaing run to be invoked with a script like rej2run. If multiple –a runs are invoked, each with a different number after –n, each will write to the same –avdist.out file. Make sure that if you re-run Rejector2 that you clean out this file manually!

5.5 –ao: Convert Arlequin .arp file

Rejector2 will convert an Arlequin .arp file into a Rejector2 .txt input file if it receives the switch –ao <.arp name> with nothing else on the command line. The prior information placed in the Rejector2 input fille will just be a placeholder that you can then edit manually.

6. New Statistics

REJECTOR2 can calculate all of the statistics that REJECTOR can, plus a number of new statistics.

6.1 Runs of Homozygosity

Some of the statistics presented here are based on the calculation of runs of homozygosity (ROH's) (McQuillan, Leutenegger et al. 2008). To use such statistics, you must use the –g switch to generate genotypic data – for there to be homozygosity your sample must be diploid! Note that ROH statistics are designed to be run on SNP data only.

A run of homozygosity is determined for any diploid individual by sliding a window of a minimum of 50 SNPs covering at least 500,000 base pairs across all of the SNPs in the data. Gaps between individual SNPs of greater than 500,000 base pairs ends a window to ensure that such gaps do not extend over centromeres or regions where too much recombination would be possible. Within a window a maximum of 1 mismatch between SNPs and 5 cases of missing data are allowed.

SNPs where at least 5% of the overlapping windows indicate homozygosity are included in consideration for a run of homozygosity. If at least 25 contiguous SNPs covering at least 100,000 base pairs are counted in this manner, that region is marked as a run of homozygosity for that indivudual.

6.2 Haplotype Blocks

REJECTOR2 calculates some new statistics based on haplotype blocks. These haplotype blocks are determined using the linkage disequlibrium statistic r^2 (Villa-Angulo, Matukumalli et al. 2009), based on a method derived from HAPLOT (Gu, Pakstis et al. 2005). A block is begun by selecting adjacent loci with the highest r^2 value averaged across samples, with a minimum value of 0.4. The block will be extended to adjacent loci if the average r^2 value is at least 0.3 and all pairwise r^2 values are at least 0.1. Once block construction is complete, contiguous blocks are merged.

6.3.New calculation groups

In addition to the groups of statistics outlined in the Rejector guide, the following two have been added to accommodate new statistics.

- ByDeme – These calculations are performed on all of the samples in a deme. The exact manner of this calcuation is specified by the statistic function.
- ByTwoDeme– These calculations are performed between two demes. The exact manner of this calcuation is specified by the statistic function.

- TwoLociAdj – Like the TwoLoci group from Rejector, except this group only calculates statistics on adjacent loci.

6.4 List of new summary statistics

The keywords used to invoke each of the following statistics is listed next to its name in the header in *italics*,. Remove the brackets around it before putting this keyword in the input file's Statistics list.

6.4.1 F_{ROH} (*FROH)* (ByDeme)

This statistic is the sum of all regions in a sample within a run of homozygosity (see above) as a proportion of the entire region sampled:

$$F_{ROH} = L_{ROH}/L_{AUTO}$$

where the subscript ROH means within a run of homozygosity and AUTO means within the autosomal region sampled. The individual F_{ROH} values are averaged for each deme.

6.4.2 F_{ROH} Variance (ByDeme)

This statistic is the variance of F_{ROH} values (see page 30) within a deme.

6.4.3 Hamming Distance (PerSystem)

This statistic averages the pairwise distance between all individuals in a deme across all loci. The distance between two individuals is simply the proportion of non-missing loci which are the same between samples to the proportion of all non-missing loci between samples.

6.4.4 Hamming Distance (ByTwoDemes)

This statistic is the same as the Hamming Distance statistic above save that samples are compared between demes, not within.

6.4.5 Haplotype Block Length (ByDeme)

The mean length of the haplotype blocks (see page 30) in a deme.

6.4.6 Haplotype Block Boundaries (ByTwoDeme)

This statistic locates all of the loci in haplotype blocks in two demes, and calculates the fraction of loci within blocks for both demes overe the total number of loci.

6.4.7 Long Shared Sequences (ByDeme)

This statistic is the proprtion of pairs of sample that share at least one run of shared loci in excess of 3 million base pairs, with jumps between loci of more than 500,000 base pairs ending a run (Atzmon, Hao et al. 2010).

6.4.8 Long Shared Sequences (ByTwoDemes)

This is caluclated the same as the statistic above, expect that the pairs of sequences are drawn from different demes.

6.4.9 Minor Allele Frequency (PerSystem)

The frequency of the rarest allele at each loci, averaged across loci.

6.4.10 Number of Haplotype Blocks (ByDeme)

The count of haplotype blocks in a deme.

6.4.11 Number of Runs of Homozygosity(ByDeme)

This is the mean number of runs of homozygosity of the diploid indivudals in a deme, as determined by the ROH calculations described on page 30.

6.4.12 Number of Segregating Sites (ByDeme)

The count of loci where there is any diversity within a deme.

6.4.13 Shared Haplotype Blocks (ByTwoDeme)

The proportion of shared haplotypes between the samples in two demes, with the hapolotypes determined by determining haplotype blocks as above (see page 30).

6.4.14 Tajima's D

This statistic compares two estimates of the genetic parameter θ (or $4N\mu$) , one by the number of segregating sites and the other by the number of pariwise differences (Tajima 1989). Negative values of D indicate population expansion or positive selection, while positive values indicate decreases in populaiton size and/or balancing selection.

6.4.15 Variance in ROH Length

The variance in the length of runs of homozygosity in a deme.

7. References

Atzmon, G., L. Hao, et al. (2010). "Abraham•s Children in the Genome Era: Major Jewish Diaspora Populations Comprise Distinct Genetic Clusters with Shared Middle Eastern Ancestry." The American Journal of Human Genetics **86**(6): 850-859.
The American Journal of Human Genetics, 86 (2010) 850-859. doi:10.1016/j.ajhg.2010.04.015

Chen, G., P. Marjoram, et al. (2008). "Fast and flexible simulation of DNA sequence data." Genome Research **19**(1): 136-142.

Gu, S., A. Pakstis, et al. (2005). "HAPLOT: a graphical comparison of haplotype blocks, tagSNP sets and SNP variation for multiple populations." Bioinformatics **21**(20): 3938.

Hellenthal, G. and M. Stephens (2006). "msHOT: modifying Hudson's ms simulator to incorporate crossover and gene conversion hotspots." Bioinformatics **23**(4): 520-521.

Hudson, R. (2002). "Generating samples under a Wright-Fisher neutral model of genetic variation." Bioinformatics.
Page 1. BIOINFORMATICS APPLICATIONS NOTE Vol. 18 no. 2 2002 Pages 337–338 Generating samples under a Wright–Fisher neutral model of genetic variation

McQuillan, R., A. Leutenegger, et al. (2008). "Runs of homozygosity in European populations." The American Journal of Human Genetics **83**(3): 359-372.

Tajima, F. (1989). "Statistical method for testing the neutral mutation hypothesis by DNA polymorphism." Genetics **123**(3): 585.

Villa-Angulo, R., L. Matukumalli, et al. (2009). "High-resolution haplotype block structure in the cattle genome." BMC genetics **10**(1): 19.

Weber, J. L. and C. Wong (1993). "Mutation of human short tandem repeats." Hum. Molec. Genet. **2**: 1123-1128.