GppFST: Genomic posterior predictive simulated Fst and Dxy distributions

Rich Adams, Drew Schield, Daren Card, Heath Blackmon, and Todd Castoe

last updated: 2016-10-20

Genomic distributions of population genetic differentiation

Fst and *Dxy* are two commonly used measures of genetic differentiation between two (or more) populations. When population genomic data are available, empirical distributions of these two measurements can be leveraged to identify loci with observed values that differ significantly from the rest of the genome. Under the assumptions of neutral theory, the majority of genomic loci are thought to evolve by genetic drift, such that these so-called ‘outlier’ loci are inferred to be targets of various forms of natural selection. For example, population genetic processes that homogenize genetic variation between populations (i.e, gene flow, convergent selection) will lead to lower *Fst* and *Dxy* values, while divergent selection will yield stronger genetic differentation and higher *Fst*/*Dxy* values.

However, multiple population genetic processes can influence *Fst* distributions , and discerning among these processes can be challenging when classifying outlier loci. For example, low *Fst* values could result from gene flow and various forms of selection (i.e., balancing, convergent). Small population sizes and deep divergence may shift genomic distributions of *Fst* and dXY towards larger values, which can hinder or confound inferences concerning the importance of particular loci in divergent selection. Furthermore, most *Fst* models of genetic differentiation assume equal population sizes, such that the rate of genetic drift is approximately equivalent within each population.

The R package GppFst (**G**enomic **P**osterior **P**redictive **F**st distributions) provides a robust framework to account for multiple evolutionary processes when conducting outlier tests for both Fst and Dxy distributions. When population parameters are inferred from genomic data, the functions GppFst and GppDxy provided in this package will simulated theoretical distributions of *Fst* and *Dxy* that can be used classify putative targets of selection. In short, our model accounts for the following sources of uncertainity that may influence Fst and Dxy distributions: divergence time estimates, population size parameter estimates, different population size parameters between populations, unequal population sampling, and SNP sampling.

In our package F_{ST} is calculated as

$$F_{ST} = \frac{s^2}{\overline{p}(1-\overline{p})}$$

Where $\overline{p}$ is the allele frequency of allele A across populations and s^2 is the sample variance of allele A frequencies over populations calculated as:

$$s^2 = \sum_i \frac{n_i(\widetilde{p_i}-\overline{p})^2}{(r-1)\overline{n}}$$

Where $\widetilde{p_i}$ is allele frequency of allele A in population i calculated with a sample size of n. Finally r is the number of populations.

While D_{xy} is calculated as $D_{xy} = \sum_{ij} x_i y_j d_{ij}$ where d_{ij} measures the number of nucleotide differences between haplotype i from X and haplotype j from Y.

Step 0: Before conducting PPS with this package

The functions `GppFst` and `GppDxy` require an input of a posterior distribution of coalescent parameters for two populations (see Step 2). These parameters can be inferred within a Bayesian framework using a number of programs that implement the multispecies coalescent model, such as:

- BEAST (http://beast.bio.ed.ac.uk (http://beast.bio.ed.ac.uk))
- SNAPP (http://beast2.org/snapp/ (http://beast2.org/snapp/))
- bpp (http://abacus.gene.ucl.ac.uk/software.html (http://abacus.gene.ucl.ac.uk/software.html))

As well as many other programs. Users should always evaluate and diagnose the MCMC algorithm used to infer the posterior distribution before conducting PPS. This can be accomplished using a number of programs, including:

- Tracer (http://beast.bio.ed.ac.uk/tracer (http://beast.bio.ed.ac.uk/tracer))
- AWTY (http://king2.scs.fsu.edu/CEBProjects/awty/awty_start.php (http://king2.scs.fsu.edu/CEBProjects/awty/awty_start.php))

The accuracy and efficiency of MCMC sampling can vary widely across algorithms. As a general rule of thumb, each parameter estimate should have an Effective Sample Size (ESS) greater than 100. However, this is a rough approximation, and users should always carefully evaluate the posterior distribution used for conducting PPS. Additionally, the above programs can be used to set an appropriate burn-in value to remove MCMC steps that may be heavily influenced by the starting value. If users have already removed an appropriate number of MCMC steps before inputting the file, users can set `burnin = 0` in the `read.posterior` function.

Step 1: install R package GppFst from github

The R package GppFst is freely available to download and distribute from github https://github.com/radamsRHA/GppFst/ (https://github.com/radamsRHA/GppFst/). To install and load GppFst, you must first install the R packages `devtools` , `phybase` , and `Geneland` .

```
install.packages("devtools")
install.packages("Geneland")
```

The packages, `phybase` should be downloaded http://faculty.franklin.uga.edu/lliu/content/phybase (http://faculty.franklin.uga.edu/lliu/content/phybase). Save it to your desktop and then use this code to install it from source:

```
install.packages("~/Desktop/phybase", repos = NULL, type = "source")
```

Now using devtools we can install `GppFst` from github:

```
library(devtools)
install_github("radamsRHA/GppFst")
library(GppFst)
library(phybase) # Load dependancy phybase
library(Geneland) # Load dependancy Geneland
```

Step 2: `read.posterior`: function to input a posterior distribution of coalescent parameters inferred for two populations

The function `read.posterior` provided in this R package is used to read a tab-delimited or comma-seperated list of posterior parameter estimates for a two population model. These parameters can be inferred using a number of coalescent-based models used for inferences of multiple species. The four coalescent parameters that must be included in the input file read by `read.posterior` include:
1. pop0.theta: population size parameter (θ_0) estimates for population 0
2. pop1.theta: population size parameter (θ_1) estimates for population 1
3. pop12.theta population size parameter (θ_{01}) estimates for ancestral population 01
4. TreeHeightLogger: divergence time parameter (τ_{01}) estimates for the two populations

The format for the `read.posterior` function arguments is as follows:

`read.posterior(posterior.file, format, burnin)` * `posterior.file` : path to the input file of MCMC steps from a posterior distribution of the four parameters
* `format` : stated as "tab" or "csv", to stipulate for the format for the input file
* `burnin` : specify the proportion of samples to remove as burnin. For example, if burnin = 0.1 and the total number of MCMC steps is 100, than the first 10 steps will be removed prior to conducting PPS

This function will return an object containing the posterior distribution of coalescent parameters (θ_0, θ_1, θ_{01}, τ_{01}) sampled by MCMC, with the burnin steps removed.

WARNING: theoretical distributions of Fst and Dxy generated by this package assume that the input parameter estimates have converged on a true posterior distribution (see Step 0: Before Conducting PPS with this package)

```
# Load the example data provided with the package
data(atrox_snap_gamma2)

# Check the names of the parameters used in GppFst
names(atrox_snap_gamma2)
```

```
##  [1] "Sample"           "posterior"        "likelihood"
##  [4] "prior"            "pop0.theta"       "pop1.theta"
##  [7] "pop12.theta"      "TreeHeightLogger" "freqParameter.A"
## [10] "freqParameter.C"  "freqParameter.G"  "freqParameter.T"
## [13] "rate.AC"          "rate.AG"          "rate.AT"
## [16] "rate.CG"          "rate.GT"          "mutationRate"
```

```
# To use the function 'read.posterior' you can try loading the raw file included
# with the package

file.loc <- system.file("atrox_snap_gamma2.txt", package="GppFst")
MCMC.samples <- read.posterior(posterior.file =file.loc, format = "tab", burnin = .25)
```

```
## [1] "Removing the first 100 posterior samples as burn-in"
## [1] "Leaving 301 steps for Fst/Dxy PPS"
```

Here, the dataframe 'MCMC.samples' includes a total of 301 steps used for downstream PPS analysis of Fst and Dxy. You can see the first set (after removing burnin of 25%) of parameter estimates using the following:

```
MCMC.samples[1,]
```

```
##     Sample posterior likelihood     prior pop0.theta pop1.theta
## 101 100000 -214843.7  -214825.7 -18.02609  0.2156598 0.05803158
##     pop12.theta TreeHeightLogger freqParameter.A freqParameter.C
## 101    0.061754      0.005658653       0.3179199       0.1802657
##     freqParameter.G freqParameter.T  rate.AC   rate.AG     rate.AT
## 101       0.1921023        0.309712 7.91e-08 0.3849241 0.001246969
##      rate.CG   rate.GT mutationRate
## 101 0.247851 0.0538222            1
```

NOTE: it is extremely important that the names of the 4 parameters provided in the output file are specified exactly as follows:

- pop0.theta
- pop1.theta
- pop12.theta
- TreeHeightLogger

Otherwise an error will result when using the function ‘read.posterior’:

```
Error in read.posterior(posterior.file = "~/Desktop/GppFST_Tutorial/atrox_snap_gamma.txt",  :    ERROR: CHECK parameter names in fil
```

Step 3.1: `GppFst`: function to conduct posterior predictive simulations of Fst for outlier tests

Next, using the sample of the posterior distribution provided by the `read.posterior` function, we will conduct PPS for Fst using the function ‘GppFst’. The input arguments for the function GppFst are as follows:

```
GppFst(posterior.samples, loci.per.step, SNP.per.locus, sample.vec.0, sample.vec.1, sequence.length.vec)
```

- posterior.samples: sample of a posterior distribution of MCMC steps that have been read by the function ‘read.posterior’
- loci.per.step: the number of loci to simulate per step in the MCMC distribution
- SNP.per.locus: specify the maximum number of SNPs to obtain for locus. For example, this argument can be set to sample a single SNP from each simulated locus
- sample.vec.0: A vector containing the number of samples obtained from population 0 for each locus in the empirical dataset. For example, if sample.vec.0 = c(5,6,7), the empirical dataset contained a total of 3 loci, with the number of population 0 samples being 5, 6, and 7 diploid individuals for each locus, respectively
- sample.vec.1: Same as sample.vec.0, instead it is a vector containing the number of samples obtained from population 1 for each locus in the empirical dataset
- sequence.length.vec: A vector containing the number of sites for each locus in the empirical dataset. For example, a sequence.length.vec = c(100, 190, 200), the empirical dataset contained a total of 3 loci, with the number of nucleotide sites being 100, 190, and 200 for each locus, respectively.

```
library(GppFst) # Load package
library(phybase) # Load dependancy phybase
library(Geneland) # Load dependancy Geneland
file.loc <- system.file("atrox_snap_gamma2.txt", package="GppFst")
MCMC.samples <- read.posterior(posterior.file =file.loc, format = "tab", burnin = .25)

experimental_params <- read.table(file = '~/Desktop/GppFST_Tutorial/ExperimentalParameters.txt', header = T) # Re
ad tab-delimited file with experimental parameters
pop0.samples <- experimental_params$pop0.samples # Extract pop0 samples per empirical locus
pop1.samples <- experimental_params$pop1.samples # Extract pop1 samples per empirical locus
locus.lengths <- experimental_params$locus.length # Extract locus lengths per empirical locus

Gppfst.results <- GppFst(posterior.samples = MCMC.samples,
                                               loci.per.step = 10,
                                               SNP.per.locus = 1,
                                               sample.vec.0 = pop0.samples,
                                               sample.vec.1 = pop1.samples,
                                               sequence.length.vec = locus.lengths) # Generate PPS for Fst
```

This command performed a PPS of 10 samples per step in the MCMC provided in the file ‘atrox_snap_gamma2.log’ (total of 301 steps after burnin), and retained a single SNP for each simulated locus. Additionally, this PPS distribution of Fst simulated alignments with a length of 190 (sequence.length.vec = c(190, 190)) for all PPS loci.

Step 3.2: `GppDxy`: function to conduct posterior predictive simulations of Dxy for outlier tests

Next, using the sample of posterior distribution provided by the ‘read.posterior’ function, we will conduct PPS for Dxy using the function ‘GppDxy’. The input arguments for the function GppFst are as follows:

```
GppDxy(posterior.samples, loci.per.step, sample.vec.0, sample.vec.1, sequence.length.vec)
```

- posterior.samples: sample of a posterior distribution of MCMC steps that have been read by the function `read.posterior`
- loci.per.step: the number of loci to simulate per step in the MCMC distribution
- sample.vec.0: A vector containing the number of samples obtained from population 0 for each locus in the empirical dataset. For example, if `sample.vec.0 = c(5,6,7)`, the empirical dataset contained a total of 3 loci, with the number of population 0 samples being 5, 6, and 7 diploid individuals for each locus, respectively

- sample.vec.1: Same as sample.vec.0, instead it is a vector containing the number of samples obtained from population 1 for each locus in the empirical dataset
- sequence.length.vec: A vector containing the number of sites for each locus in the empirical dataset. For example, a `sequence.length.vec = c(100, 190, 200)` , the empirical dataset contained a total of 3 loci, with the number of nucleotide sites being 100, 190, and 200 for each locus, respectively.

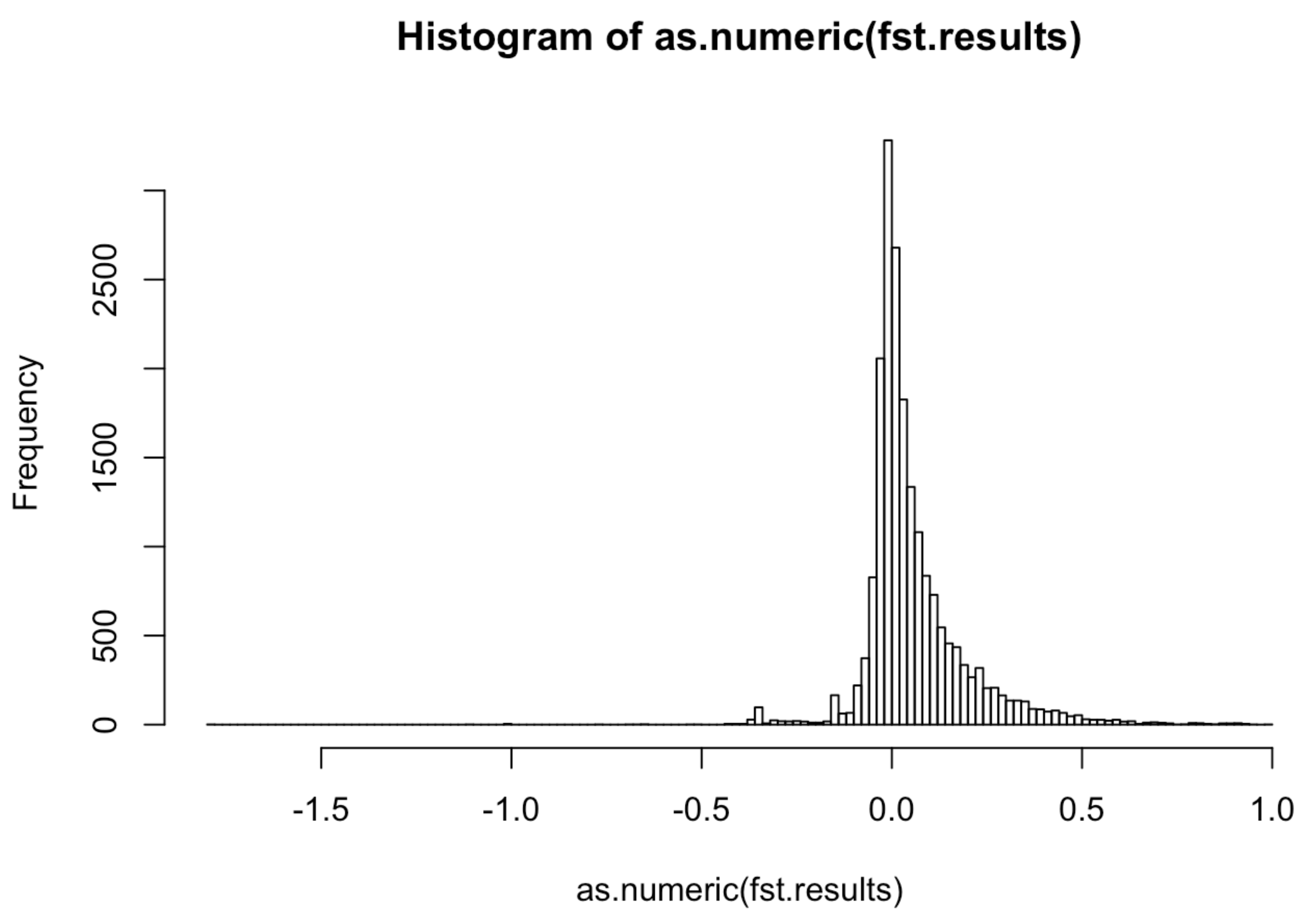
```
library(GppFst) # Load package
library(phybase) # Load dependancy phybase
library(Geneland) # Load dependancy Geneland

MCMC.samples <- read.posterior(posterior.file = '~/Desktop/GppFST_Tutorial/atrox_snap_gamma2.log', format = "tab"
, burnin = .25)
[1] "Removing the first 100 posterior samples as burn-in"
[1] "Leaving 301 steps for Fst/Dxy PPS"


experimental_params <- read.table(file = '~/Desktop/GppFST_Tutorial/ExperimentalParameters.txt', header = T) # Re
ad tab-delimited file with experimental parameters
pop0.samples <- experimental_params$pop0.samples # Extract pop0 samples per empirical locus
pop1.samples <- experimental_params$pop1.samples # Extract pop1 samples per empirical locus
locus.lengths <- experimental_params$locus.length # Extract locus lengths per empirical locus

Gppdxy.results <- GppDxy(posterior.samples = MCMC.samples,
                         loci.per.step = 10,
                         sample.vec.0 = pop0.samples,
                         sample.vec.1 = pop1.samples,
                         sequence.length.vec = locus.lengths)
```

This command simulated a PPS distribution of 10 samples per step in the MCMC provided in the file 'atrox_snap_gamma2.log' (total of 301 steps after burnin). Additionally, this PPS distribution of Dxy simulated alignments with a length of 190 (sequence.length.vec = c(190, 190)) for all PPS loci.

Step 4: Evaluate PPS results

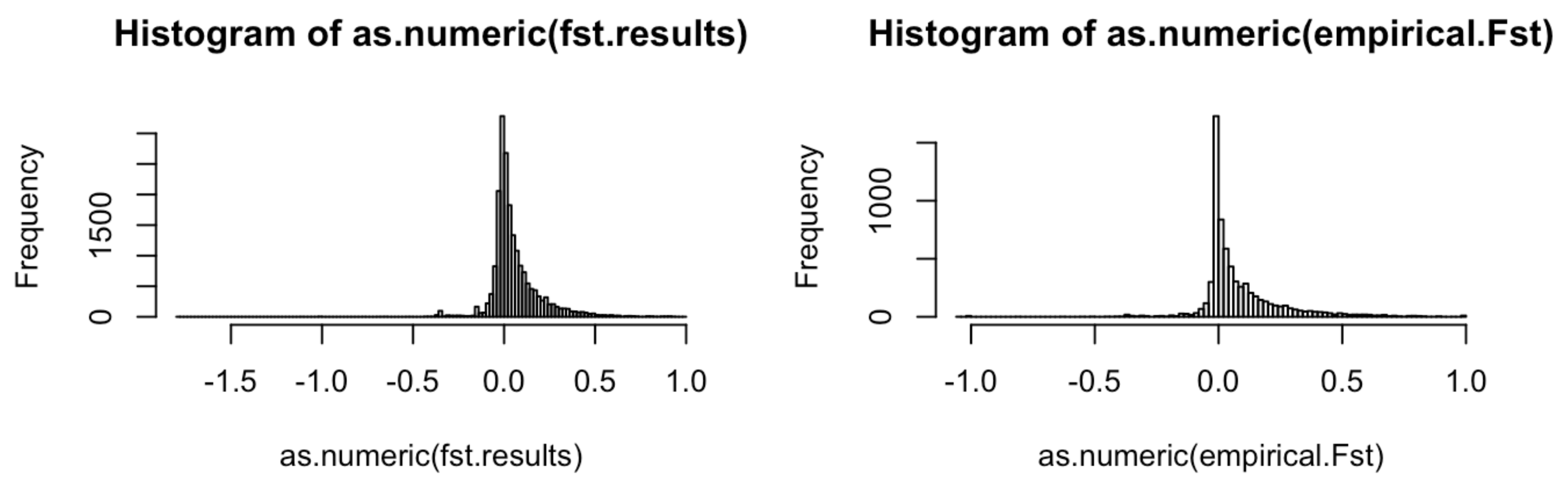
We can visualize/assess PPS results for Fst and Dxy using a number of methods. Lets visualize this theoretical distribution of Fst obtained from a previous analysis:

```
data(GppFst.results)
fst.results <- GppFst.results[,1][is.na(GppFst.results[,1]) == F]
hist(as.numeric(fst.results), breaks = 100)
```



Using the empirical Fst values provided in the 'ExperimentalParameters.txt' we can now visually compare the PPS and empirical distribution of Fst

```
data(GppFst.results)
fst.results <- GppFst.results[,1][is.na(GppFst.results[,1]) == F]

data(experimental_params)

empirical.Fst <- experimental_params$WEIR_AND_COCKERHAM_FST

par(mfrow=c(2,2))
hist(as.numeric(fst.results), breaks = 100)
hist(as.numeric(empirical.Fst), breaks = 100)
```



At this scale, it is difficult to see differences between the distributions at Fst intervals. However, we can compare proportions of loci at specific Fst intervals that we think might represent non-neutral evolution

```
# Let's look at the number of loci with Fst > 0.9
fst.results[fst.results > .9]
```

```
##  [1] 0.9166667 0.9166667 0.9259259 0.9375000 0.9166667 0.9166667 0.9047619
##  [8] 0.9375000 0.9166667 0.9166667 1.0000000 0.9166667 0.9375000 0.9432539
```

```
empirical.Fst[empirical.Fst > .9]
```

```
##  [1] 1.000000 1.000000 0.953700 1.000000 1.000000 1.000000 0.919039
##  [8] 1.000000 1.000000 1.000000 1.000000 1.000000
```

```
# Let's compare the proportion of loci with Fst > 0.9
length(fst.results[fst.results > .9])/length(fst.results)
```

```
## [1] 0.0007026349
```

```
length(empirical.Fst[empirical.Fst > .9])/length(empirical.Fst)
```

```
## [1] 0.001745455
```

We can see from a rough look that there appears to be more loci with an Fst value of 1 (complete fixation for opposite alleles between the two populations) in the empirical dataset. There is about 10x the proportion of loci with Fst > 0.9 in the empirical when compared to the PPS distribution.

```
# Let's compare the proportion of loci with Fst = 1 (complete fixation for opposite alleles)
length(fst.results[fst.results  == 1])/length(fst.results)
```

```
## [1] 5.018821e-05
```

```
length(empirical.Fst[empirical.Fst == 1])/length(empirical.Fst)
```

```
## [1] 0.001454545
```

Finally, we compute the probability of observing 10 or more loci with an Fst value of 1 (10 was observed in the empirical dataset)

```
observed <- length(empirical.Fst[empirical.Fst == 1]) # Number of loci with Fst = 1

1-pbinom(q = (observed-1), size = length(empirical.Fst), prob = length(fst.results[fst.results == 1])/length(fst.
results))
```

```
## [1] 4.789502e-12
```

Quite a small probability!

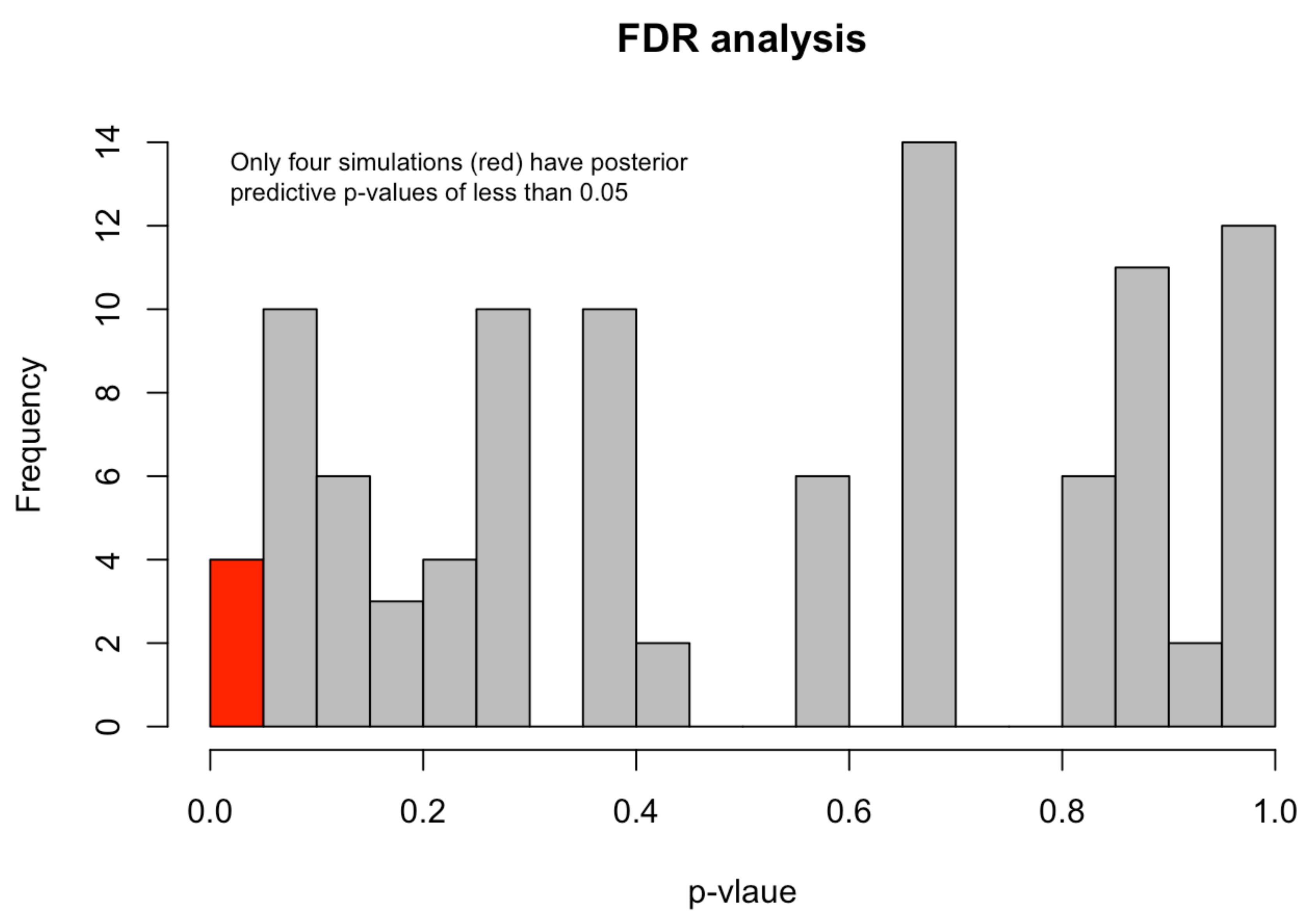
In this case, it is reasonable to suspect that not all of the 10 loci with Fst values of 1 have evolved under the neutral model. However, we can also compute the probability of observing at least 1 locus with an Fst value of 1:

```
1-dbinom(x = 0 , size = length(empirical.Fst), prob = length(fst.results[fst.results == 1])/length(fst.results))
```

```
## [1] 0.2918169
```

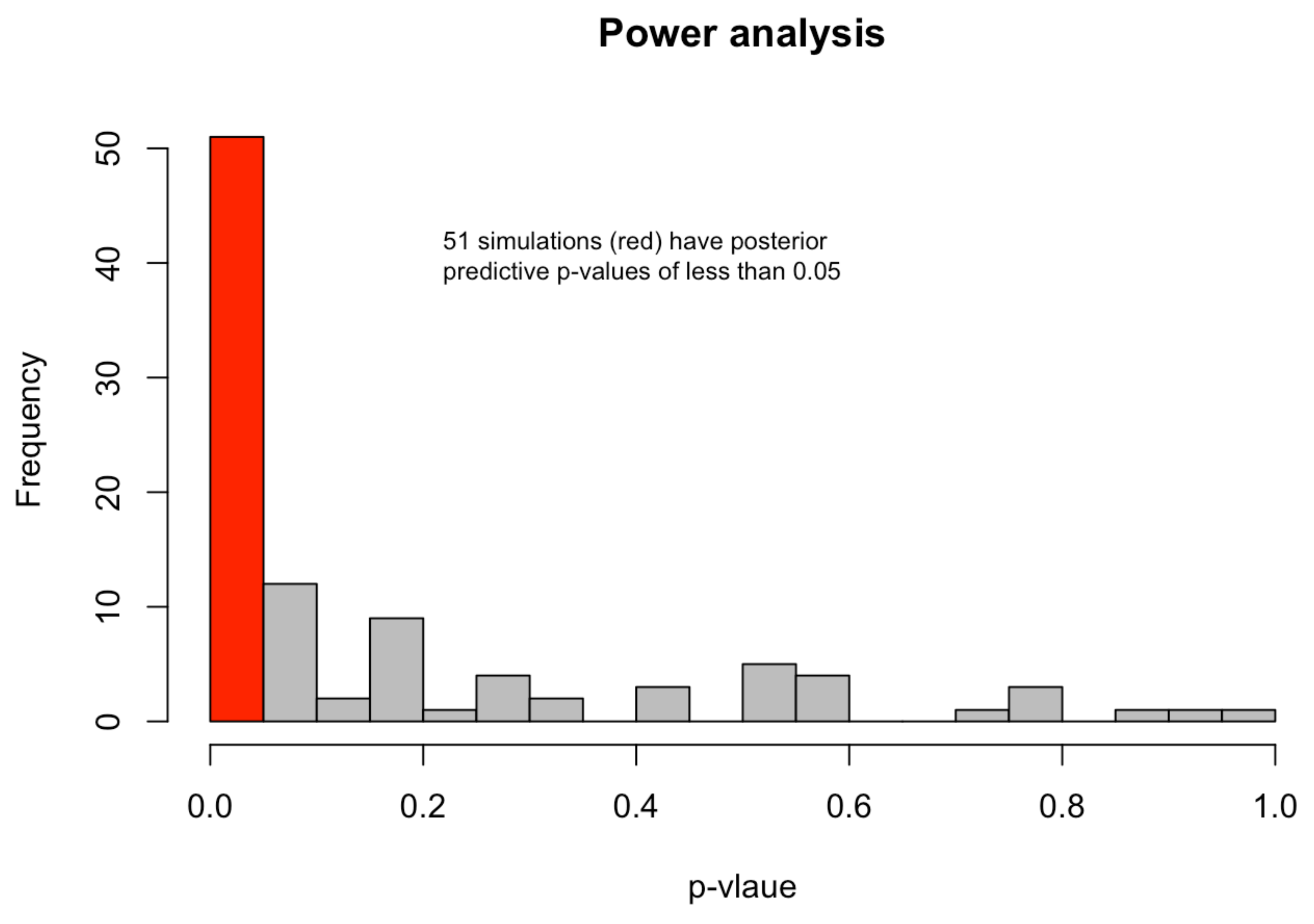
FDR analysis

To insure that our method is working correctly we can also evaluate the FDR. To do this we use a dataset simulated with parameters similar to those we inferred while analyzing the atrox data. However, in this case there is no selection on any loci. In our analysis we used an Fst cutoff value of 1 to test if we had a signal for selection. We then perform our analysis as described above and any test that returns a positive result is a type 1 error. As expected we have an approximately 5% error rate.



Power analysis

To assess the power of our method we simulated data with msms (http://www.mabs.at/ewing/msms/index.shtml) using parameters informed by our analysis of *Atrox*. Briefly, we simulated 10,100 loci, 100 loci were under selection, and the selection coefficient was set to 0.01. We then used a cutoff of FST=1 and found that our approach was able to recover a significant signal in 51% of the simulated datasets.


Power analysis
51 simulations (red) have posterior predictive p-values of less than 0.05
Frequency
0
10
20
30
40
50
0.0
0.2
0.4
0.6
0.8
1.0
p-vlaue