Using fsthet

Sarah P. Flanagan

2017-03-02

A common way to identify loci putatively under selection in population genomics studies is to identify loci that have high differentiation Fst relative to their expected heterozygosity Ht, as described in Beaumont & Nichols (1996). However, the Fst-Ht distribution changes shape based on the demographics of the population, and some distribution shapes are less conducive to identifying outliers than others. The problem of different distribution shapes is exacerbated by the current implementation of analyses which assume the same distribution for all demographic parameters. **fsthet** bootstraps across the existing dataset to generate confidence intervals that approximate the actual Fst-Ht distribution.

This package performs several tasks.
- Parses genepop files into R.
- Calculates allele frequencies, Ht, and Fst (three commonly-used Fst calculations).
- Generates smoothed quantiles from the empirical distribution.
- Generates customizable Fst-Ht plots with the quantiles.
- Identifies loci lying outside of the quantiles.

Getting Started

Read in your data

The first step is to organize your data in the genepop format. If you've been using LOSITAN, the format is identical. For details on the genepop format, refer to this website. **fsthet** accepts both haploid and diploid genepop files with alleles coded using either the 2- or 3-digit format.

```
library(fsthet)
gfile<-system.file("extdata", "example.genepop.txt",package = 'fsthet')
gpop<-my.read.genepop(gfile)
```

```
##
## Parsing Genepop file...
##
##
## File description:  Numerical Analysis with Nm=10, N=1000, 75 Demes, sampling 5 populations.
##
## ...done.
```

This function outputs any descriptors you've included in the header of your genepop file. This function was adapted from adegenet.

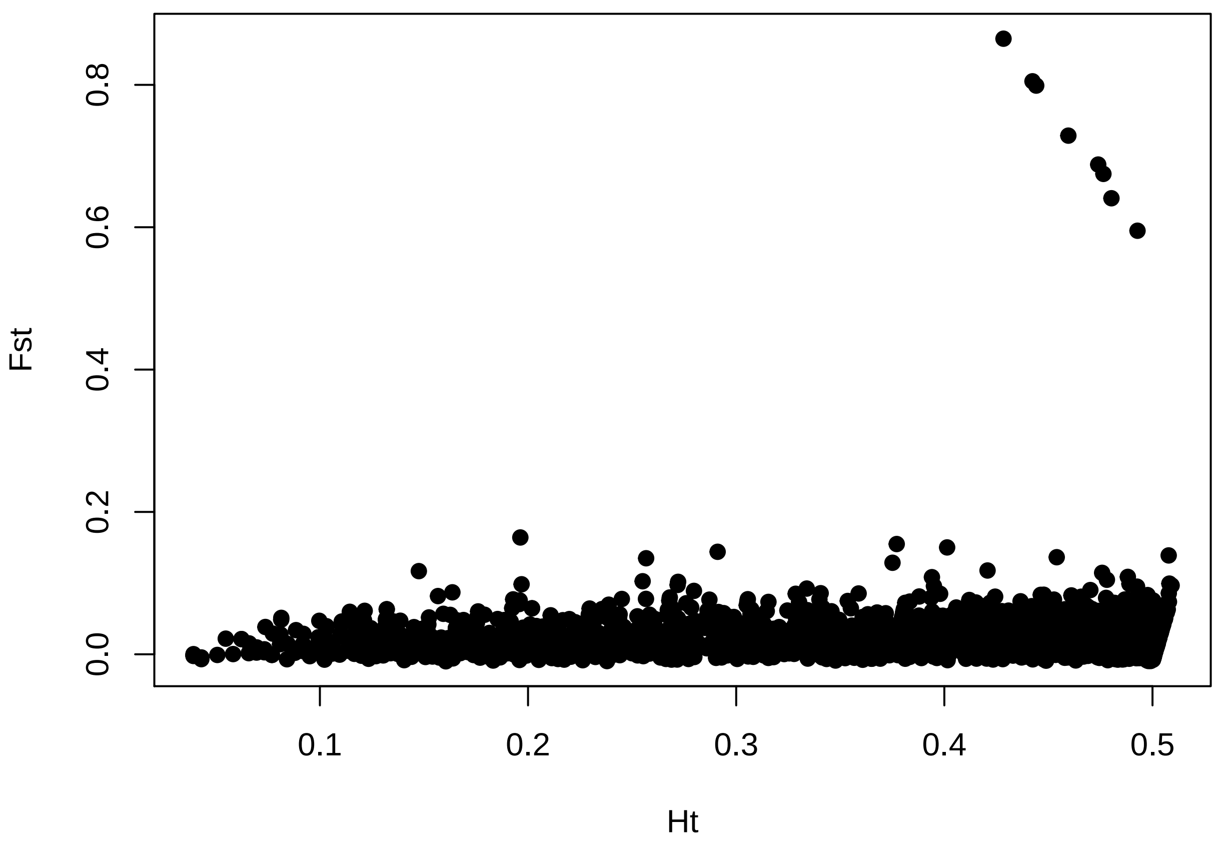
Calculate actual values

Before getting into any bootstrapping analyses, you must calculate the actual Fst and Ht values.

```
fsts<-calc.actual.fst(gpop)
head(fsts)
```

```
##   Locus      Ht                 Fst
## 1  loc0 0.41154  0.00742370490147173
## 2  loc1 0.46314 -0.00866156054652011
## 3  loc2 0.49366   0.0432215850519948
## 4  loc3 0.49058   0.0118478601498701
## 5  loc4 0.42948   0.0276237272604974
## 6  loc5  0.2019   0.0646434628604019
```

```
#Plot the actual values to see what your distribution looks like
par(mar=c(4,4,1,1))
plot(fsts$Ht, fsts$Fst,xlab="Ht",ylab="Fst",pch=19)
```



Since this distribution is not highly skewed, it should be fine for using the Fst-heterozygosity distribution to identify outliers. If you only have two demes (and/or your distribution is skewed highly to the right), you might consider using alternative approaches to identifying outliers (e.g. Arlequin, BayeScan, BayEnv, PCAdapt).

Understanding each step of the analysis

The `fst.boot` function generates smoothed quantiles when you specify `bootstrap=FALSE`.

Generating quantiles

```
quant.out<-as.data.frame(t(replicate(1,fst.boot(gpop,"WCC", bootstrap = FALSE))))
```

```
## [1] "Fsts calculated. Now Calculating CIs"
```

```r
str(quant.out)
```

```
## 'data.frame':    1 obs. of  3 variables:
##  $ Fsts:List of 1
##   ..$ Fsts:'data.frame': 2000 obs. of  2 variables:
##   .. ..$ Ht : num  0.0393 0.0393 0.0431 0.0431 0.0507 ...
##   .. ..$ Fst: num  -0.002382 0.000185 -0.006817 -0.004474 -0.000944 ...
##  $ Bins:List of 1
##   ..$ Bins: num [1:20, 1:2] 0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr  "3" "4" "5" "6" ...
##   .. .. ..$ : chr  "low.breaks" "upp.breaks"
##  $ V3  :List of 1
##   ..$ :List of 1
##   .. ..$ CI0.95:'data.frame':    20 obs. of  4 variables:
##   .. .. ..$ Low   : num  -0.007 -0.007 -0.007 -0.008 -0.007 -0.008 -0.008 -0.007 -0.007 -0.007 ...
##   .. .. ..$ Upp   : num  0.038 0.049 0.054 0.063 0.063 0.082 0.071 0.064 0.08 0.097 ...
##   .. .. ..$ LowHet: num  0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   .. .. ..$ UppHet: num  0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 0.275 0.3 ...
```

```r
head(quant.out[[3]][[1]])
```

```
## $CI0.95
##       Low   Upp LowHet UppHet
## 1  -0.007 0.038  0.000  0.075
## 2  -0.007 0.049  0.050  0.100
## 3  -0.007 0.054  0.075  0.125
## 4  -0.008 0.063  0.100  0.150
## 5  -0.007 0.063  0.125  0.175
## 6  -0.008 0.082  0.150  0.200
## 7  -0.008 0.071  0.175  0.225
## 8  -0.007 0.064  0.200  0.250
## 9  -0.007 0.080  0.225  0.275
## 10 -0.007 0.097  0.250  0.300
## 11 -0.005 0.074  0.275  0.325
## 12 -0.006 0.077  0.300  0.350
## 13 -0.006 0.075  0.325  0.375
## 14 -0.006 0.081  0.350  0.400
## 15 -0.006 0.085  0.375  0.425
## 16 -0.006 0.081  0.400  0.450
## 17 -0.005 0.084  0.425  0.475
## 18 -0.005 0.083  0.450  0.500
## 19 -0.006 0.082  0.475  0.525
## 20 -0.005 0.076  0.500  0.550
```

From the results of `str(quant.out)`, you can see that `fst.boot()` returns a list data.frame with three elements: the bootstrapped values (Fsts), the bins used in the bootstrapping (Bins), and a list of the upper and lower smoothed quantiles (V3).

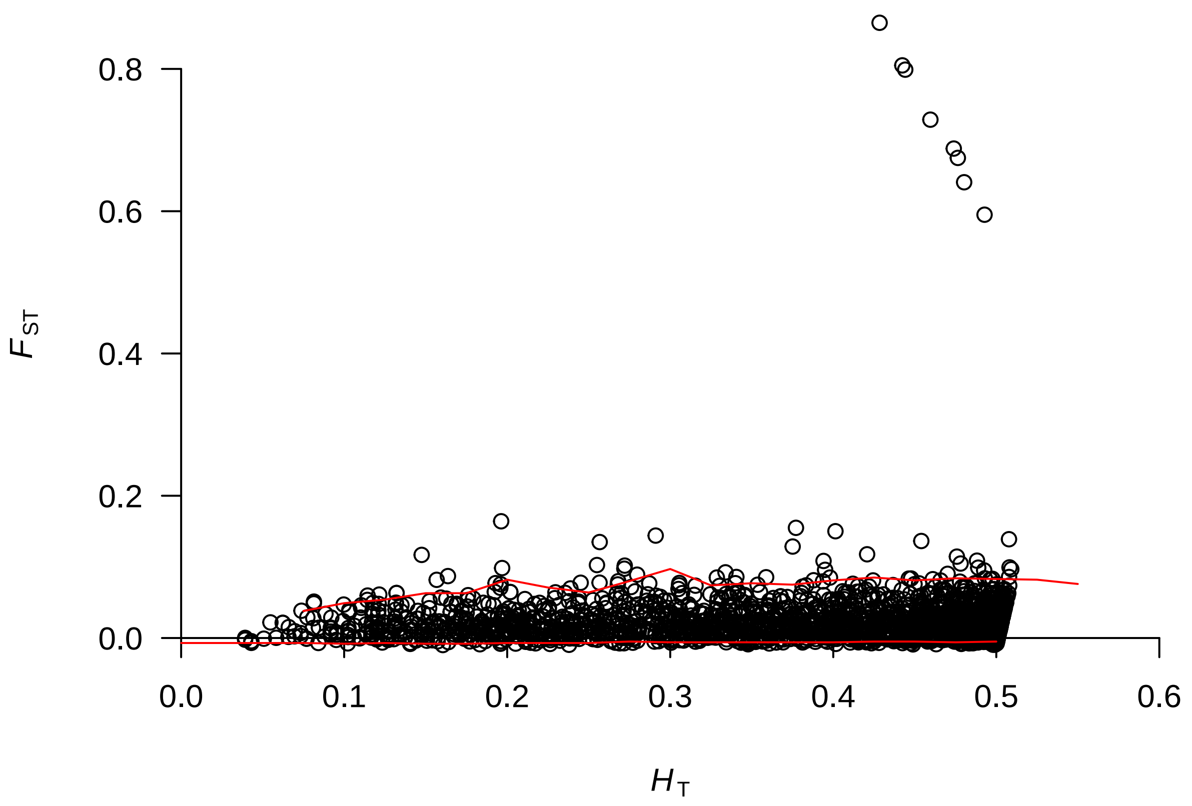
Plotting the results

If you want to visualize these results, you can use `plotting.cis`. Plotting.cis requires the raw datapoints (`fsts`) and a list with the smoothed quantiles.

```
#plot the results
quant.list<-ci.means(quant.out[[3]])
head(quant.list)
```

```
##          low   upp LowHet UppHet
## 0.075 -0.007 0.038  0.000  0.075
## 0.1   -0.007 0.049  0.050  0.100
## 0.125 -0.007 0.054  0.075  0.125
## 0.15  -0.008 0.063  0.100  0.150
## 0.175 -0.007 0.063  0.125  0.175
## 0.2   -0.008 0.082  0.150  0.200
```

```
par(mar=c(4,4,1,1))
plotting.cis(df=fsts,ci.df=quant.list,make.file=F)
```



Identifying outliers

We can also use the `find.outliers` function to pull out a data.frame containing the loci that lie outside of the quantiles.

```
outliers<-find.outliers(fsts,boot.out=quant.out)
head(outliers)
```

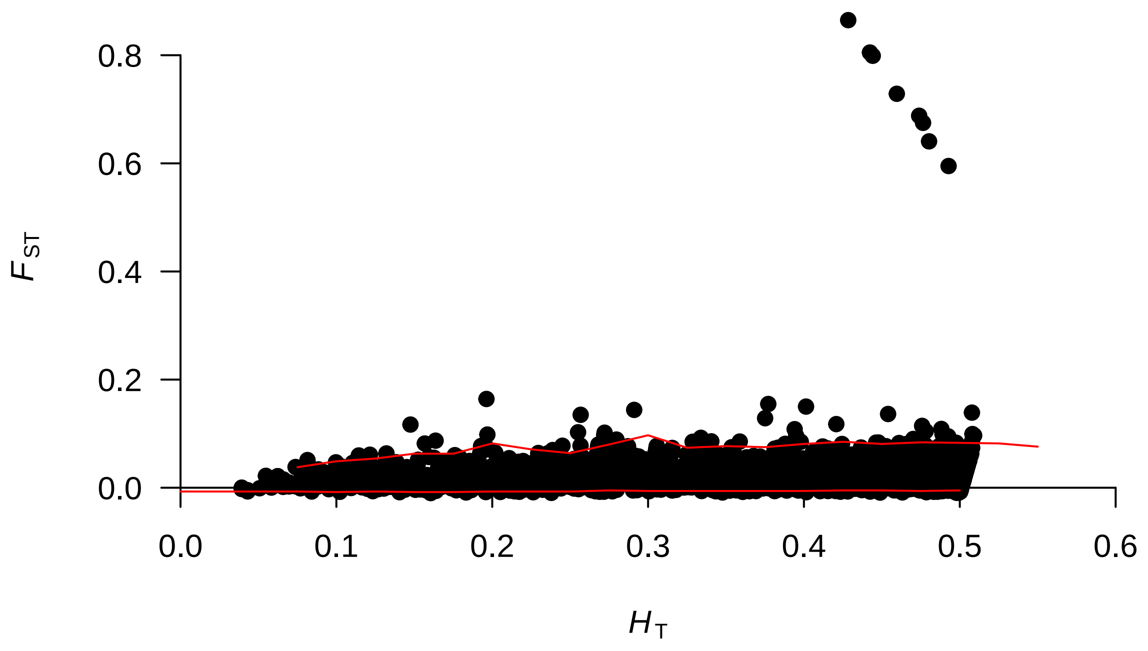
```
##         Locus                 Ht                  Fst
## 863   loc862 0.0738200000000001   0.0383378625047556
## 1542 loc1541 0.0814599999999999   0.0511548085817649
## 491   loc490            0.12152   0.0610515956138069
## 1923 loc1922            0.11438   0.0596646655398182
## 1963 loc1962            0.10222 -0.00753178428779961
## 235   loc234            0.14754    0.116829129029211
```

Using the empirical data without bootstrapping

The above functions are all contained within the wrapper function `fsthet`, so you don't have to go through each step on its own. `fsthet` returns a data.frame with four columns: Locus ID, heterozygosity, Fst, and a True/False of whether it's an outlier

```
out.dat<-fsthet(gpop)
```

```
## [1] "Fsts calculated. Now Calculating CIs"
```



```
head(out.dat)
```

```
##   Locus      Ht                  Fst Outlier
## 1  loc0 0.41154  0.00742370490147173   FALSE
## 2  loc1 0.46314 -0.00866156054652011    TRUE
## 3  loc2 0.49366   0.0432215850519948   FALSE
## 4  loc3 0.49058   0.0118478601498701   FALSE
## 5  loc4 0.42948   0.0276237272604974   FALSE
## 6  loc5  0.2019   0.0646434628604019    TRUE
```

Customizing the Figures

The default `plotting.cis()` output may not be ideal for publication. Luckily, the function `plotting.cis()` has several built-in options for customizing the plot.

The data you use

As demonstrated in the two cases above, `plotting.cis()` requires the original data and a `ci.list`, which is actually a data.frame with Ht values as row names and two columns: low, and upp. These header names are requried for it to work, and the data in the columns are the lower and upper Fst values for each Ht value.

If your actual data (`df=<name>`, or `fsts` in the above examples) have different column names, you can specify those using `plotting.cis(Ht.name=<name>)` and `plotting.cis(Fst.name=<name>)`. Otherwise, the defaults are `plotting.cis(Ht.name="Ht",Fst.name="Fst")`.

The look of the graph

Several aspects of the graph can be controlled through `plotting.cis()`: the color of the quantile lines and the shape of the points. These are controlled by `ci.col` and `pt.pch`.
The default quantile color is red (`ci.col="red"`) and the defualt point shape is open circles (`pt.pch=1`). You can also color-code some loci (for instance ones that are near genes of interest) using `sig.col`. The default setting for `sig.col` is to be identical to `ci.col`.

Saving the graph to a file

In the above examples, you may have noticed that `plotting.cis()` always contained the command `make.file=F`. This command allows you to automatically save the graph to a file or to print it to the default device in R. If `make.file=TRUE`, then the function generates a *.png file. The default file name is "OutlierLoci.png", but this can be changed using `file.name`. If you choose to designate a file.name, it must contain the ".png" extension. For example:

```
plotting.cis(df=fsts,boot.out=boot.out,make.file=T,file.name="ExampleOutliers.png")
```

Other Functions

I just want to take a moment to discuss what the other functions in **fsthet** do and some other ways to use the proram.

Saving the data and plotting it yourself.

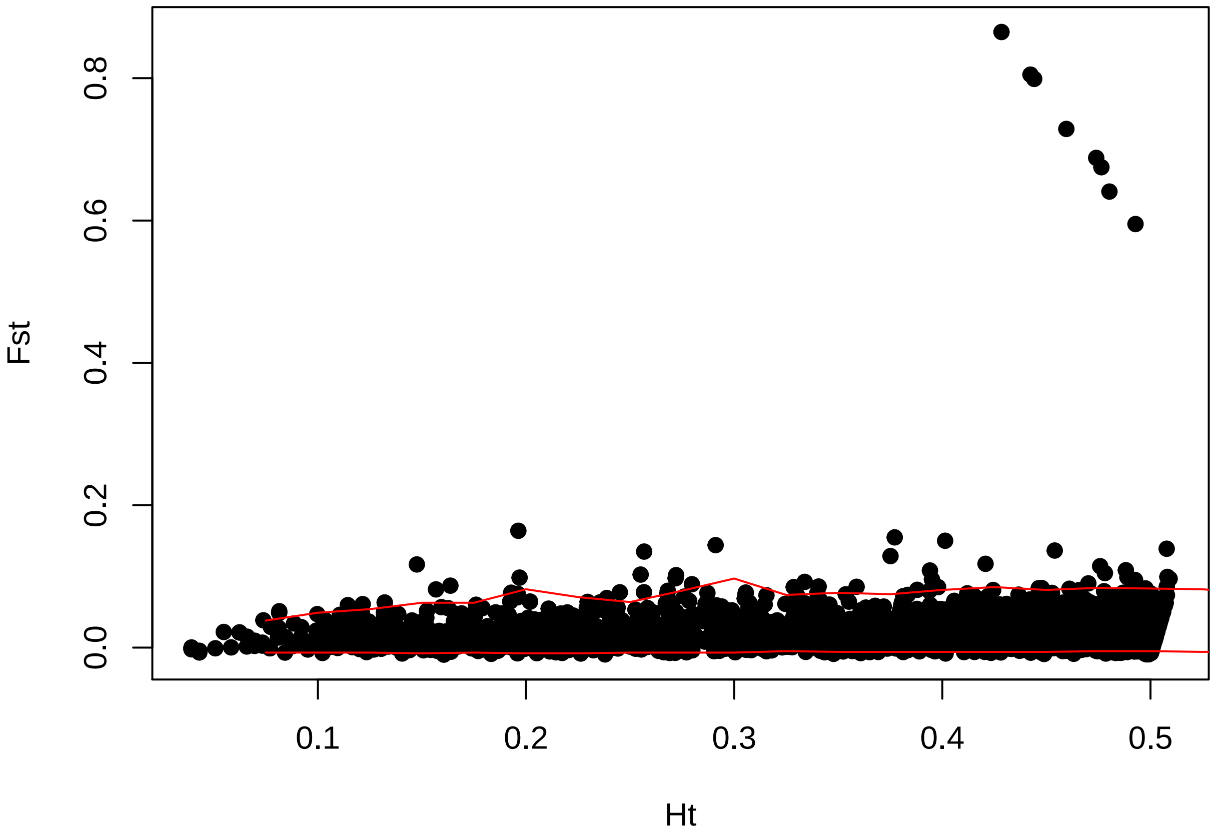
Although `plotting.cis` can be a useful tool, it is possible to save your quantiles and generate your own plots. First, you use the function `ci.means()` to calculate the mean confidence intervals across all of the bootstrap replicates, and then you can generate a plot and add the confidence intervals using `points()`.

```
#get the quantiles
quant.list<-ci.means(quant.out[[3]])

#create a data.frame of confidence intervals
qs<-as.data.frame(do.call(cbind,quant.list))
colnames(qs)<-c("low","upp")
```

```
qs$Ht<-as.numeric(rownames(qs))

#plot
par(mar=c(4,4,1,1))
plot(fsts$Ht, fsts$Fst,pch=19,xlab="Ht",ylab="Fst")
points(qs$Ht,qs$low,type="l",col="red")
points(qs$Ht,qs$upp,type="l",col="red")
```

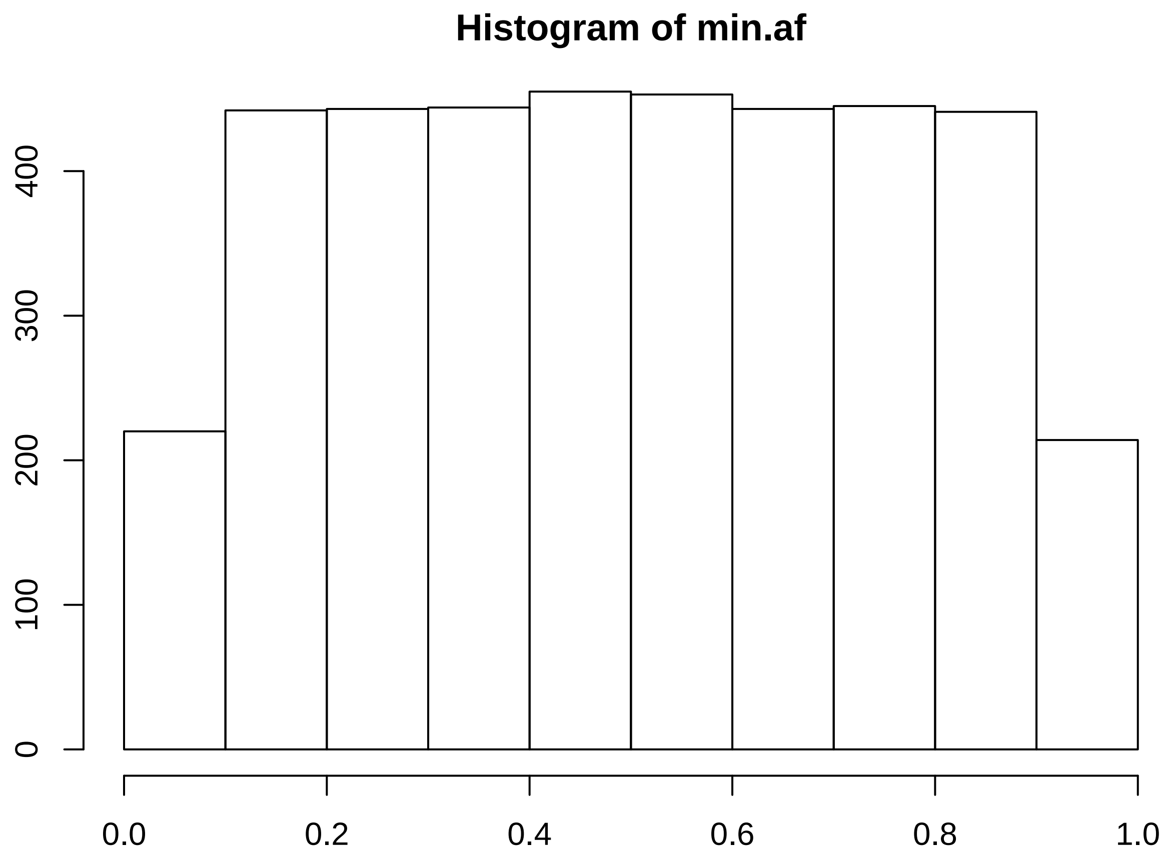


Look at the distribution of allele frequencies

The analyses in **fsthet** use the function `calc.allele.freq` to calculate allele frequencies. If you're interested in examining the allele frequency distribution in your dataset, you can use this function on your actual data.

```
af.actual<-apply(gpop[,3:ncol(gpop)],2,calc.allele.freq)

#extract the minimum allele frequency for each locus
min.af<-unlist(lapply(af.actual,min))
par(mar=c(2,2,2,2))
hist(min.af)
```



Conclusion

Hopefully this package will be a useful tool for population geneticists and molecular ecologists. It's important to consider the assumptions of the tests you use as well as remembering that statistics should be used to describe your dataset. Use your common sense about when to use different methods and how to implement them. Good luck!

If you run into any problems, find any bugs, or have other comments on fsthet please contact me: spflanagan.phd@gmail.com.
