Using fsthet

Sarah P. Flanagan

2018-04-04

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** calculates smoothed quantiles from the existing dataset to identify loci with extreme Fst values relative to their heterozygosity.

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.

```r
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.

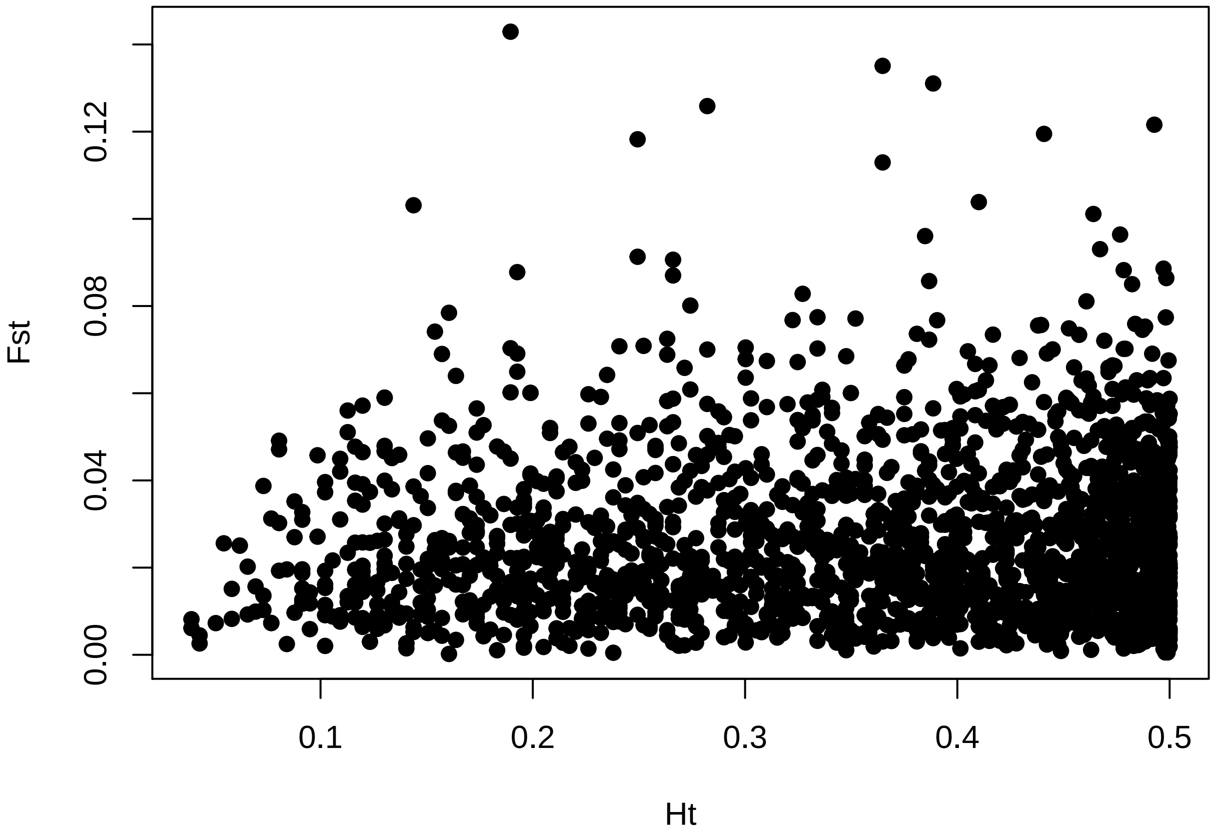
Calculate actual values

Before calculating the smoothed quantiles, you must calculate the actual Fst and Ht values.

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

```
##   Locus       Ht         Fst
## 1  loc0 0.410112 0.013927903
## 2  loc1 0.463008 0.001140369
## 3  loc2 0.488448 0.042682128
## 4  loc3 0.488448 0.017459382
## 5  loc4 0.426272 0.030102845
## 6  loc5 0.198912 0.060086873
```

```
#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. Arleqeuin, BayeScan, BayEnv, PCAdapt).

Understanding each step of the analysis

Generating quantiles

Using boot.out

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

```
quant.out<-fst.boot(gpop, bootstrap = FALSE)
```

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

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

```
## List of 3
##  $ Fsts:'data.frame':    2000 obs. of  2 variables:
##   ..$ Ht : num [1:2000] 0.0392 0.0392 0.043 0.043 0.0506 ...
##   ..$ Fst: num [1:2000] 0.00612 0.00816 0.00446 0.0026 0.00727 ...
##  $ Bins:'data.frame':    18 obs. of  2 variables:
##   ..$ low.breaks: num [1:18] 0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   ..$ upp.breaks: num [1:18] 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 0.275 0.3 ...
##  $     :List of 1
##   ..$ CI0.95:'data.frame':   18 obs. of  4 variables:
##   .. ..$ Low   : num [1:18] 0.003 0.002 0.002 0.002 0.003 0.002 0.002 0.002 0.002 0.003 ...
##   .. ..$ Upp   : num [1:18] 0.026 0.047 0.051 0.059 0.069 0.07 0.065 0.06 0.08 0.072 ...
##   .. ..$ LowHet: num [1:18] 0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   .. ..$ UppHet: num [1:18] 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 0.275 0.3 ...
```

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

```
##     Low   Upp LowHet UppHet
## 1 0.003 0.026  0.000  0.075
## 2 0.002 0.047  0.050  0.100
## 3 0.002 0.051  0.075  0.125
## 4 0.002 0.059  0.100  0.150
## 5 0.003 0.069  0.125  0.175
## 6 0.002 0.070  0.150  0.200
```

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).

Directly binning and finding quantiles

Alternatively, the functions wrapped into fst.boot can be used on their own. This is advantageous if you'd like to use Fst and heterozygosity values calculated by another program or in another analysis (e.g. output from LOSITAN)

```
head(fsts)
```

```
##   Locus       Ht         Fst
## 1  loc0 0.410112 0.013927903
## 2  loc1 0.463008 0.001140369
## 3  loc2 0.488448 0.042682128
## 4  loc3 0.488448 0.017459382
## 5  loc4 0.426272 0.030102845
## 6  loc5 0.198912 0.060086873
```

```
bins<-make.bins(fsts)
cis<-find.quantiles(bins = bins$bins,bin.fst = bins$bin.fst)
str(cis)
```

```
## List of 1
##  $ CI0.95:'data.frame':  18 obs. of  4 variables:
##   ..$ Low   : num [1:18] 0.003 0.002 0.002 0.002 0.003 0.002 0.002 0.002 0.002 0.003 ...
##   ..$ Upp   : num [1:18] 0.026 0.047 0.051 0.059 0.069 0.07 0.065 0.06 0.08 0.072 ...
##   ..$ LowHet: num [1:18] 0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   ..$ UppHet: num [1:18] 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 0.275 0.3 ...
```

You can also designate more than one confidence level

```
cis.list<-find.quantiles(bins = bins$bins,bin.fst = bins$bin.fst,ci=c(0.01,0.05))
str(cis.list)
```

```
## List of 2
##  $ CI0.99:'data.frame':  18 obs. of  4 variables:
##   ..$ Low   : num [1:18] 0.003 0.002 0.002 0.001 0 0 0.001 0 0 0.002 ...
##   ..$ Upp   : num [1:18] 0.039 0.049 0.057 0.103 0.078 0.088 0.088 0.091 0.091 0.091 ...
##   ..$ LowHet: num [1:18] 0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   ..$ UppHet: num [1:18] 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 0.275 0.3 ...
##  $ CI0.95:'data.frame':  18 obs. of  4 variables:
##   ..$ Low   : num [1:18] 0.003 0.002 0.002 0.002 0.003 0.002 0.002 0.002 0.002 0.003 ...
##   ..$ Upp   : num [1:18] 0.026 0.047 0.051 0.059 0.069 0.07 0.065 0.06 0.08 0.072 ...
##   ..$ LowHet: num [1:18] 0 0.05 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 ...
##   ..$ UppHet: num [1:18] 0.075 0.1 0.125 0.15 0.175 0.2 0.225 0.25 0.275 0.3 ...
```

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.

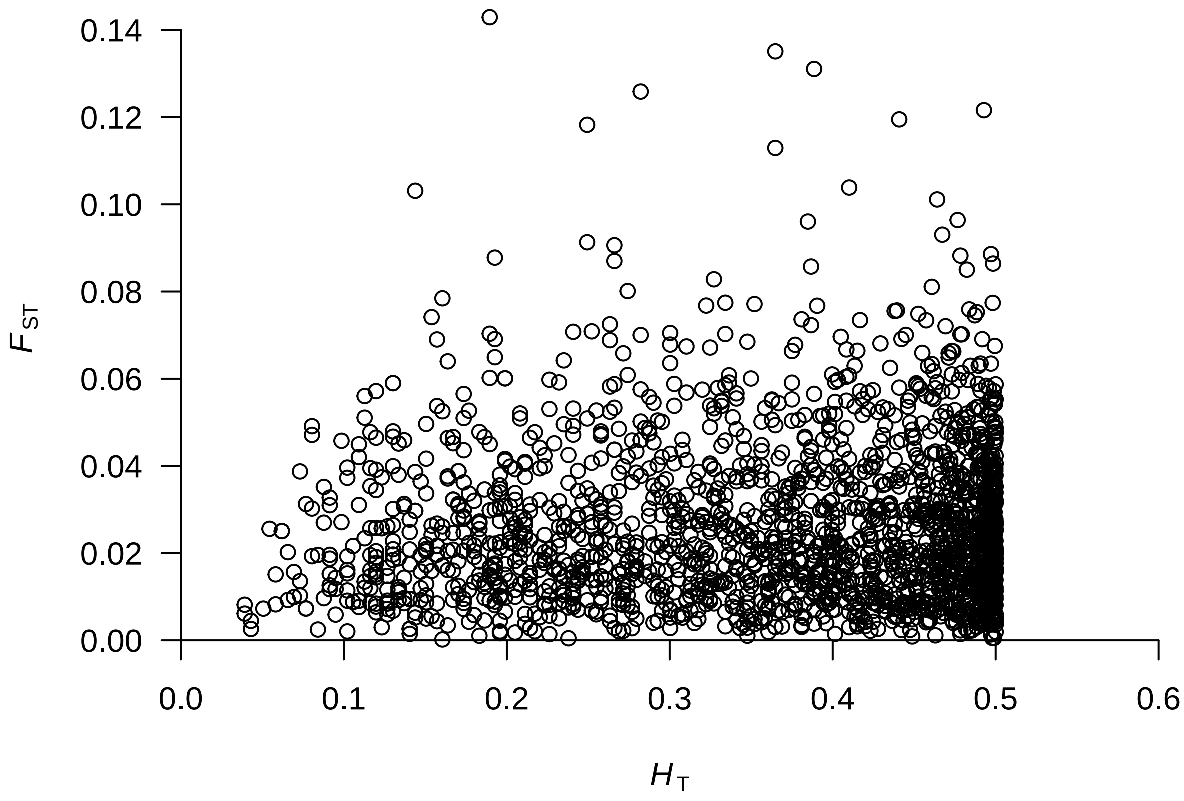
```
#extract the confidence interavls
quant.list<-ci.means(quant.out[[3]])
head(quant.list)
```

```
##         low   upp LowHet UppHet
## 0.075 0.003 0.026  0.000  0.075
## 0.1   0.002 0.047  0.050  0.100
## 0.125 0.002 0.051  0.075  0.125
## 0.15  0.002 0.059  0.100  0.150
## 0.175 0.003 0.069  0.125  0.175
## 0.2   0.002 0.070  0.150  0.200
```

```
#Alternatively
quant.list<-cis$CI0.95
head(quant.list)
```

```
##     Low   Upp LowHet UppHet
## 1 0.003 0.026  0.000  0.075
## 2 0.002 0.047  0.050  0.100
## 3 0.002 0.051  0.075  0.125
## 4 0.002 0.059  0.100  0.150
## 5 0.003 0.069  0.125  0.175
## 6 0.002 0.070  0.150  0.200
```

```
#plot the results
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)
```

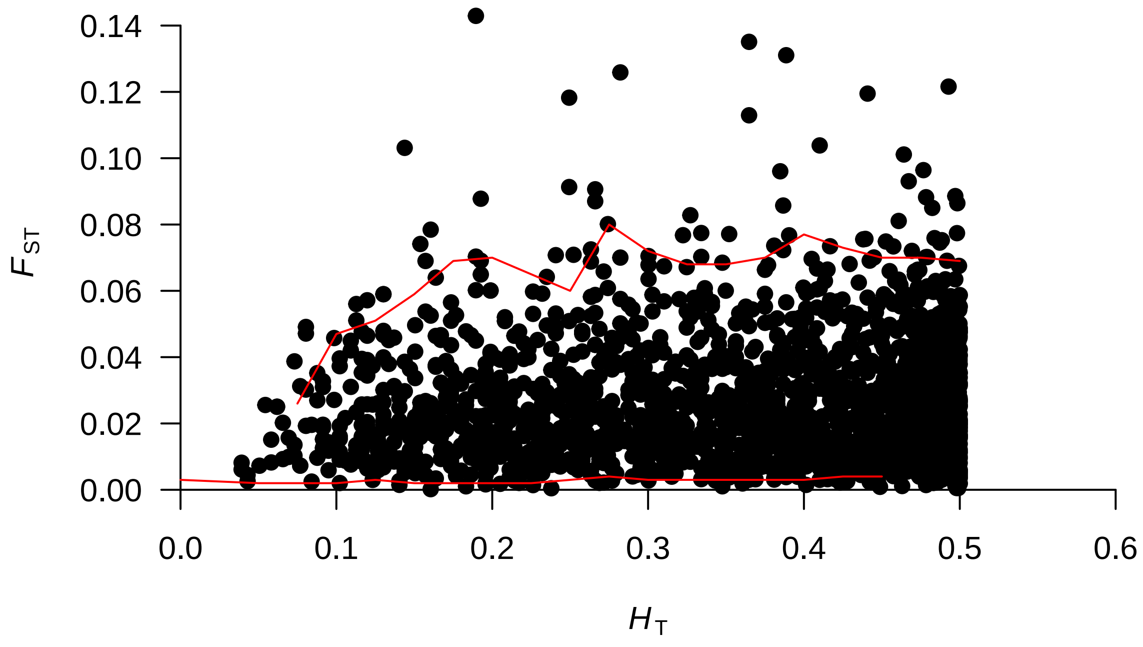
```
##        Locus       Ht         Fst
## 863   loc862 0.073112 0.038735091
## 1706 loc1705 0.043032 0.002602714
## 228   loc227 0.080472 0.047121980
## 1542 loc1541 0.080472 0.049110250
## 71     loc70 0.112800 0.051063830
## 491   loc490 0.119808 0.057158120
```

Using the wrapper function `fsthet`

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"
```



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

```
##   Locus       Ht         Fst Outlier
## 1  loc0 0.410112 0.013927903   FALSE
## 2  loc1 0.463008 0.001140369    TRUE
## 3  loc2 0.488448 0.042682128   FALSE
## 4  loc3 0.488448 0.017459382   FALSE
## 5  loc4 0.426272 0.030102845   FALSE
## 6  loc5 0.198912 0.060086873   FALSE
```

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")
```

The various Fst calculations

There are many different ways to calculate Fst, and **fsthet** contains four calculations. You can specify which one you want to use in the `calc.actual.fsts`, `fsthet`, and `fst.boot` with the fst.choice parameter. The choices can all be viewed with `fst.options.print`:

```
fst.options.print()
```

```
## [1] For Wright's Fst: fst, FST, Fst
## [1] For a variance-based Fst (beta): var, VAR, Var
## [1] For Cockerham and Weir's Theta: theta, Theta, THETA
## [1] For Beta-hat (LOSITAN): Betahat, betahat, BETAHAT
```

Wright's Fst (fst)

Wright formulated Fst as

$$F_{ST} = \frac{H_T - H_S}{H_T}$$

in 1943, and this classic estimation of Fst is implemented using the `fst.choice="fst"` option. This is the default choice.

```
fsts<-calc.actual.fst(gpop,"fst")
```

Weir's θ (theta)

In Weir's 1990 book, Genetic Data Analysis, he presents an Fst estimator termed θ. This is an estimator calculated based on a number of estimators:

$$\theta = \frac{s2a - (\frac{1}{\overline{n}-1}\overline{p}(1-\overline{p}) - \frac{N-1}{N}s2a)}{\frac{nc-1}{\overline{n}-1}\overline{p}(1-\overline{p}) + \frac{s2a}{N}(1+\frac{(N-1)(\overline{n}-nc)}{\overline{n}-1})}$$

$$s2a = \frac{1}{\overline{n}(N-1)}\sum_{i=1}^{N}(p_i - \overline{p})^2$$

$$nc = \frac{1}{N-1}\sum_{i=1}^{N} n_i - \frac{1}{\sum_{i=1}^{N} n_i}\sum_{i=1}^{N} n_i^2$$

```
fsts.theta<-calc.actual.fst(gpop,"theta")
```

A variance-based Fst, β (var)

This approach calculates Fst as the variance in allele frequencies, and is calculated as:

$$F_{ST} = \frac{1}{2N}\frac{\sum_{i=1}^{N}(p_i - \overline{p})^2}{\overline{p}(1-\overline{p})}$$

```
fsts.beta<-calc.actual.fst(gpop,"var")
```

Cockerham and Weir's $\hat{\beta}$ (betahat)

Cockerham and Weir (1993) formulated $\hat{\beta}$, which is the value used by FDIST2 (Beaumont and Nichols 1996) and LOSITAN (Antao et al. 2008). It is calculated as

$$\hat{\beta} = \frac{\hat{f}_0 - \hat{f}_1}{1-\hat{f}_1}$$

In this formulation,

$$\hat{f}_0 = \frac{2\overline{n}(\sum_{i=1}^{N}(p_i^2) + \sum_{i=1}^{N}((1-p_i)^2)) - N}{N(2\overline{n}-1)}$$

and

$$\hat{f}_1 = \frac{(\sum_{i=1}^{N}p_i)^2 + (\sum_{i=1}^{N}1-p_i)^2 - (\sum_{i=1}^{N}(p_i^2) + \sum_{i=1}^{N}(1-p_i)^2)}{N(N-1)}$$

In these formulas, N is the number of populations and $\overline{n}$ is the average number of individuals per population.

```
fsts.betahat<-calc.actual.fst(gpop,"betahat")
```

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.

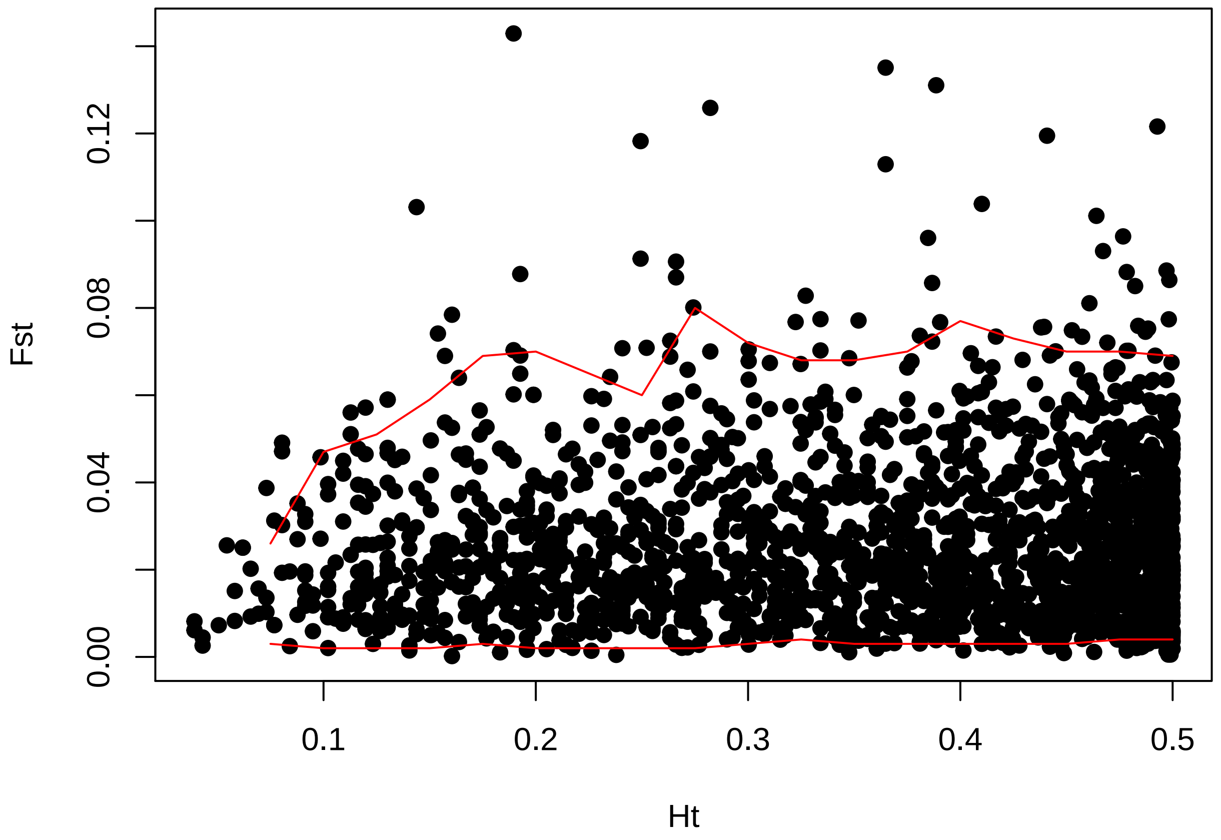
Saving the data and plotting it yourself.

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")
```

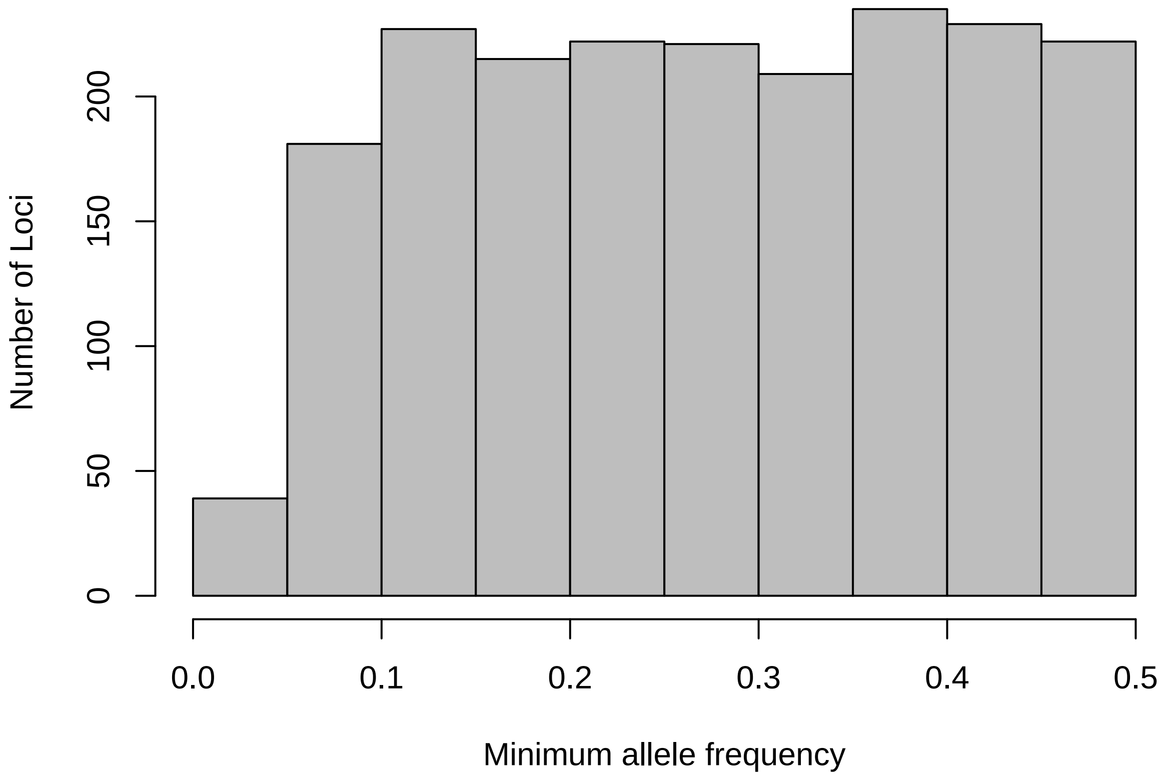


Look at the distribution of allele frequencies

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.

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

#extract the minimum allele frequency for each locus
maf<-apply(af.actual,2,min)
par(mar=c(4,4,2,2))
hist(maf,main = "",xlab="Minimum allele frequency",ylab="Number of Loci",col="grey",xlim=c(0,0.5))
```



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.
