Manual for `fsabc` version 0.2

Zach Gompert

8, May, 2020

What is `fsabc`?

`fsabc` implements an approximate Bayesian computation (ABC) method to detect and quantify fluctuating selection on polygenic traits from time-series data. Phenotypic selection is modeled as an explicit function of the state of the environment. The population-genomic consequences of selection are the modeled based on estimated genotype-phenotype associations. This allows inferences to be informed by patterns of change across multiple genetic loci, populations, and generations. The program `fsabc` is written in `C++` with the Gnu Scientific Library (GSL). It is made available is source code. See Gompert & Bergland (XXXX) for a full description and evaluation of the method. Herein, we focus on how to install and run the program.

How to install `fsabc`

These instructions assume you are using a Linux operating system, such as Ubuntu, but should work with Mac OSX as well if the appropriate utilities have been installed.

You can download (clone) the source code, manual and example files for `fsabc` from `GitHub`.

```
git clone https://github.com/zgompert/fsabc.git
```

Once you have downloaded the source code, navigate to the `fsabc` directory. You can compile the the source code by typing the following at the command line. You must have the Gnu Scientific Library installed and in your path.

```
g++ -O2 -Wall -o fsabc  main.C func.C -lm -lgsl -lgslcblas
```

This creates an executable, `fsabc`. Running the program without ivoking any options prints a help menu listing the command line options.

```
./fsabc
```

```
##
## ./fsabc version 0.2 -- 01 June 2020
##
## Usage: fsabc -g genefile -e envfile -f nefile -t traitfile [options]
## Use -v 1 for predictive mode, -q 1 for observed mode,
##  or leave both 0 (default) for simulation model
##
## -g     Infile with allele frequency data
## -e     Infile with environmental covariate data
## -f     Infile with varNe estimates
## -t     Infile with trait genetic arch. estimates
## -j     (optional) Infile with gens. between samples
## -z     (optional) Infile parameter posterior samples
## -v     Binary, run posterior pred. validation mode [0]
## -q     Binary, run observed summary stats. mode [0]
## -o     Outfile for simulated or obs. summary stats. [out_fsabc.txt]
## -n     Number of simulations [1000]
## -s     SS to print: 0 = bv, 1 = snp, 2 = both [0]
## -m     Selection model: 0 = linear, 1 = step, 2 = sigmoid [0]
## -p     Prior prob. of non-zero selection by component [0.5]
## -a     Lower bnd. on U prior for sel. function intercept [-10]
## -c     Upper bnd. on U prior for sel. function intercept [10]
## -b     Lower bnd. on U prior for sel. function slope [-10]
## -d     Upper bnd. on U prior for sel. function slope [10]
## -w     Lower bnd. on U prior for sel. function cut [-1]
## -x     Upper bnd. on U prior for sel. function cut [1]
```

Running the program

Command line options are listed above. Here, I expand on these, and provide detailed descriptions of file formats.

`-g` = Infile with allele frequency data. This file has one of two formats depending on the anlysis mode.

When running the program in simulation or posterior predictive mode, the first row is a header with two values (separated by white space), the number of genetic markers and the number of populations. This is followed by one row per genetic marker. Each row gives the initial allele frequency (for one of the two alleles, e.g., the minor allele) for each population. Thus, there are as many columns as populations. See `sim_p0.txt` for an example with 100 SNPs and 10 populations (in this example all 10 populations have the same initial allele frequencies).

When running the program to compute summary statistics for the observed data, the first row is a header with two values (separated by white space), the number of genetic markers and the product of the number of populations and generations. This is followed by one row per genetic marker. Each row gives the allele frequency (for one of the two alleles, e.g., the minor allele) for each population and generation. Use the following order: Pop0_Gen0 Pop0_Gen1 ... Pop0_Gen10 Pop1_Gen0 ... Pop1_Gen10 ... Pop5_Gen10. See `sim_p0.txt` for an example with 100 SNPs and 10 populations (in this example all 10 populations have the same initial allele frequencies). See `out_example_p.txt`.

`-e` = Infile with environmental covariate data. The first row is a header with the number of populations and number of generations. Each subsequent row gives the environmental covariate data for one population, with one column per generation. Note that the last generation (column) is not used as it would apply to the expected allele frequency in the following (not yet sampled) generation.

Example analysis

Compute summary statistics for the observed data.

```
./fsabc -g sim_example_p.txt -e sim_env.txt -f sim_ne.txt -t sim_trait.txt \
 -n 1 -m 0 -q 1 -o obs_example.txt
```

Generate 1 million simulated sets of parameters and summary statistics

```
./fsabc -g sim_p0.txt -e sim_env.txt -f sim_ne.txt -t sim_trait.txt \
 -n 1000000 -m 0 -a -0.1 -b -0.1 -c 0.1 -d 0.1 -o sims_example.txt
```

Summarize the posterior in `R`

```r
library(abc)
```

```
## Loading required package: abc.data

## Loading required package: nnet

## Loading required package: quantreg

## Loading required package: SparseM

##
## Attaching package: 'SparseM'

## The following object is masked from 'package:base':
##
##     backsolve

## Loading required package: MASS

## Loading required package: locfit

## locfit 1.5-9.4    2020-03-24
```

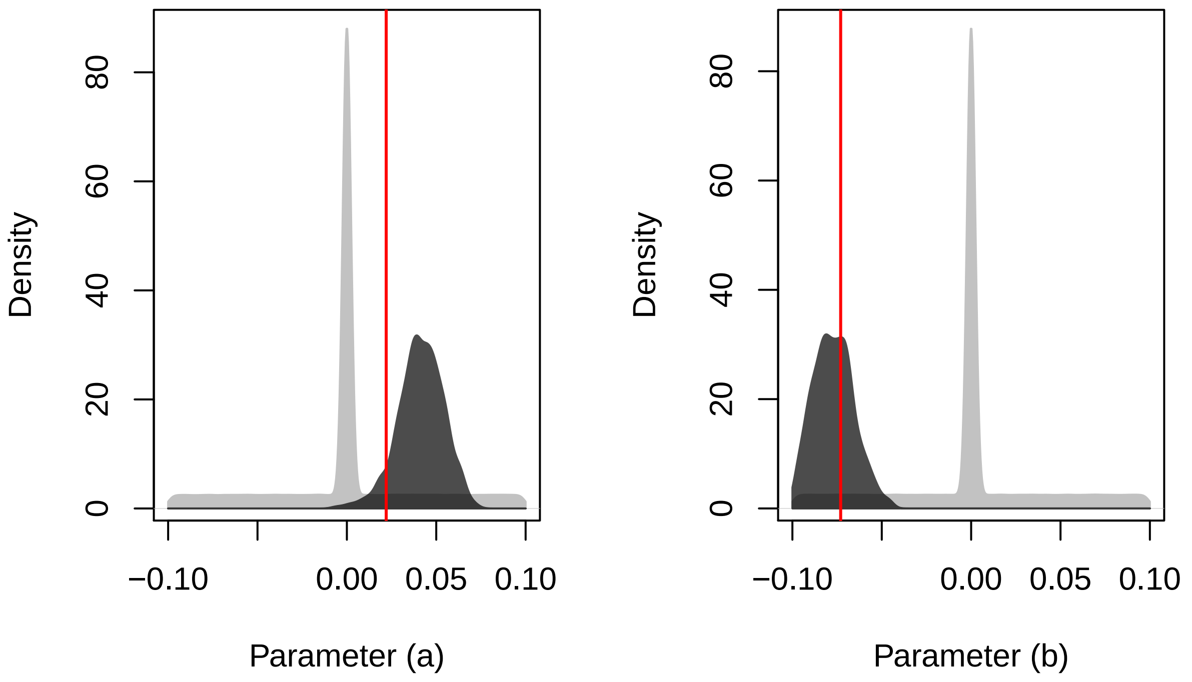
```r
library(scales)

## read in 1 million simulations
sims <- matrix(scan("sims_example.txt", n = 1e+06 * 8, sep = " "), nrow = 1e+06,
    ncol = 8, byrow = TRUE)

## read in obs. summary statistics
obs <- read.table("obs_example.txt")

## true values for a and b... from out_example.txt
a <- 0.022
b <- -0.073

## split matrixes and name columns
ss <- sims[, 7:8]
colnames(ss) <- c("ss.mn", "ss.cov")

parm <- sims[, 1:6]
colnames(parm) <- c("a", "b", "p3", "p4", "p5", "p6")
o <- abc(target = as.matrix(obs), param = parm, sumstat = ss, method = "loclinear",
    tol = 0.001)
```

```
## Warning: All parameters are "none" transformed.
```

```r
summary(o)
```

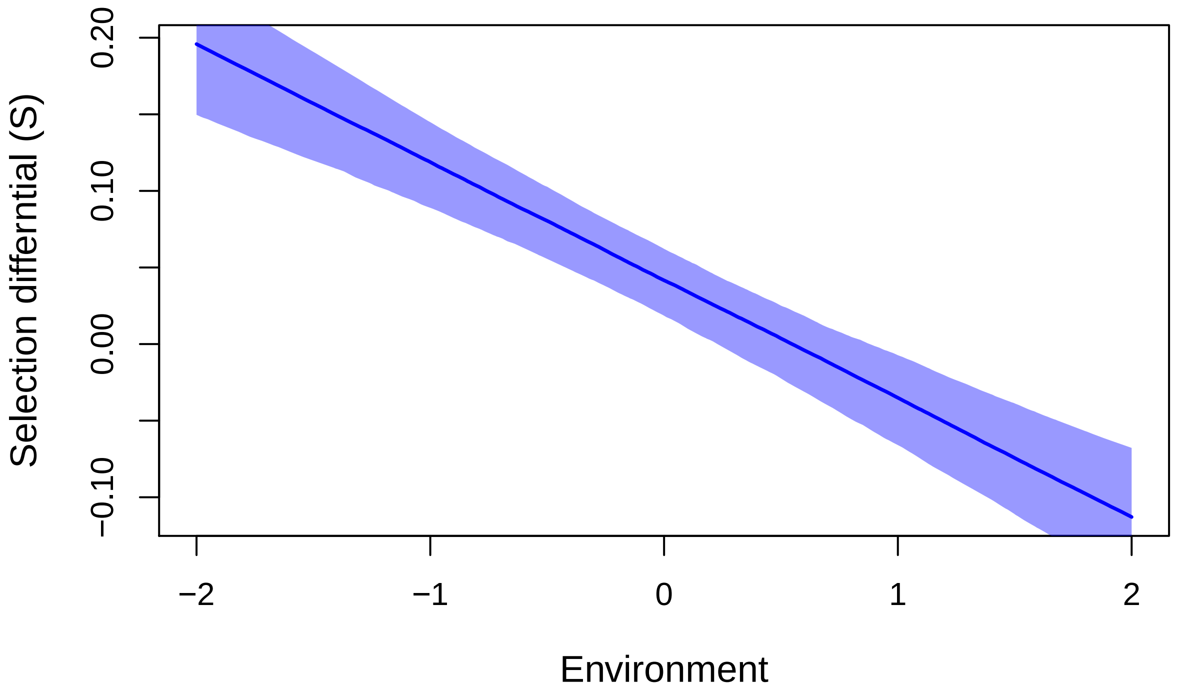
```
## Call:
## abc(target = as.matrix(obs), param = parm, sumstat = ss, tol = 0.001,
##     method = "loclinear")
## Data:
##  abc.out$adj.values (1000 posterior samples)
## Weights:
##  abc.out$weights
##
##                               a       b      p3      p4      p5      p6
## Min.:                   -0.0072 -0.1011 -0.0041  0.0416  0.0000  0.0001
## Weighted 2.5 % Perc.:   0.0129 -0.0969  0.0148  0.0508  0.0000  0.0001
## Weighted Median:        0.0414 -0.0773  0.0435  0.0737  0.0000  0.0001
## Weighted Mean:          0.0411 -0.0771  0.0433  0.0735  0.0000  0.0001
## Weighted Mode:          0.0388 -0.0735  0.0410  0.0709  0.0000  0.0001
## Weighted 97.5 % Perc.:  0.0636 -0.0539  0.0661  0.0925  0.0000  0.0001
## Max.:                   0.0735 -0.0439  0.0743  0.0968  0.0000  0.0001
```

```
library(scales)
c1 <- alpha("darkgray", 0.7)
c2 <- alpha("black", 0.7)

par(mfrow = c(1, 2))
ddpr <- density(parm[, 1], from = -0.1, to = 0.1)
ddpost <- density(o$adj.values[, 1], from = -0.1, to = 0.1)
plot(ddpr, type = "n", xlab = "Parameter (a)", main = "")
polygon(c(ddpr$x, rev(ddpr$x)), c(ddpr$y, rep(0, length(ddpr$y))), col = c1,
    border = c1)
polygon(c(ddpost$x, rev(ddpost$x)), c(ddpost$y, rep(0, length(ddpost$y))),
    col = c2, border = c2)
abline(v = a, col = "red", lwd = 1.5)

ddpr <- density(parm[, 2], from = -0.1, to = 0.1)
ddpost <- density(o$adj.values[, 2], from = -0.1, to = 0.1)
plot(ddpr, type = "n", xlab = "Parameter (b)", main = "")
polygon(c(ddpr$x, rev(ddpr$x)), c(ddpr$y, rep(0, length(ddpr$y))), col = c1,
    border = c1)
polygon(c(ddpost$x, rev(ddpost$x)), c(ddpost$y, rep(0, length(ddpost$y))),
    col = c2, border = c2)
abline(v = b, col = "red", lwd = 1.5)
```



```
## summarize the relationship between S and the environment
x <- seq(-2, 2, 0.01)  ## approx. range of environmental variation
nx <- length(x)
```

```
y <- matrix(NA, nrow = dim(o$adj.values)[1], ncol = nx)
for (i in 1:dim(o$adj.values)[1]) {
    ## linear model, computer over the post.
    y[i, ] <- o$adj.values[i, 1] + x * o$adj.values[i, 2]
}
## compute 91% credible intervals (91 is just for fun, compute what you
## want)
est <- apply(y, 2, quantile, probs = c(0.5, 0.045, 0.955))
par(mfrow = c(1, 1))
plot(x, est[1, ], col = "blue", type = "l", lwd = 1.8, xlab = "Environment",
    ylab = "Selection differntial (S)", cex.lab = 1.2)
polygon(c(x, rev(x)), c(est[2, ], rev(est[3, ])), col = alpha("blue", 0.4),
    border = NA)
```


Selection differntial (S)
Environment