A Beginners Guide to Approximate Bayesian Computation in Population Genetics

Miguel Navascués[1,2,*]

[1]INRA, UMR CBGP, F-34988 Montferrier-sur-Lez, France
[2]Institut de Biologie Computationnelle, F-34090 Montpellier, France
[*]miguel.navascues@inra.fr

September 26, 2018

Introduction

Work in progress...

Classical ABC

Likelihood

The experiment: A coin is tossed 10 times and the results (heads/tails, or head/tree in our case) are recorded

The result[1]**:**

```
toss

##  [1] "H" "H" "H" "H" "T" "H" "H" "T" "H" "T"

total_tosses <- length(toss)
heads_count  <- length(which(toss=="H"))
tails_count  <- total_tosses - heads_count
```

Number of tosses (`total_tosses`): 10
Number of heads (`heads_count`): 7
Number of tails (`tails_count`): 3

The question: Is the coin fair? = Is the probability of getting "heads" fifty percent?

$$L(p = 0.5|D) = P(D|p = 0.5)$$

where p is the probability of getting "heads" in a coin toss, L is the likelihood and D is the data (7 heads and 3 tails).

```
p_heads <- 0.5
p_tails <- 1-p_heads
p_combo <- p_heads^heads_count * p_tails^tails_count
```

Probability of combination H, H, H, H, T, H, H, T, H, T: 9.765625×10^{-4}.

That is the prbability for a given order of 7 heads in 10 coin tosses. All ordered combinations of 7 heads and 3 tails have the same probability. The number of combination is given by the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ (`choose(n,k)` in `R`).

[1] Images by CBG Numismatique Paris, CC-BY-SA

```
combinations <- choose(total_tosses, heads_count)
```

There are 120 combinations of 7 head and 3 tails.

```
likelihood <- combinations *
              p_heads^heads_count * p_tails^tails_count
```

Likelihood of $p = 0.5$ given 7 heads in 10 tosses: 0.1171875.

In the case of the coin flip experiment the likelihood can be calculated from a binomial probability model (assuming coin tosses are independent and have the same probability of getting heads). We can load file `flip_coin_likelihood.r` that contains some functions to calculate this likelihood (i.e. the code lines presented above wraped in a function: `flip.coin.likelihood(n,k,p)`:

```
source("src/flip_coin_likelihood.r")
likelihood_profile <- flip.coin.likelihood(n = total_tosses,
                                           k = heads_count,
                                           p = seq(0,1,0.001))
```

A maximum likelihood estimate and confidence intervals can be obtained from the likelihood profile:

```
maxL  <- max(likelihood_profile$likelihood)
p_hat <- likelihood_profile$p[which(likelihood_profile$likelihood
                                                    ==maxL)]
CI95  <- likelihood_profile$p[which(likelihood_profile$likelihood>
                                      exp(log(maxL)-1.92))]
CI95  <- CI95[c(1,length(CI95))]
```

The maximum likelihood estimate is $\hat{p} = 0.7$ with a 95% confidence interval of $(0.394, 0.915)$.

Exercise 1: Make flip-coin experiments (or simulate them in `R` with `sample()`) with different number of tosses and observe how confidence intervals change. Code for figure 1:

```
plot(x    = likelihood_profile$p,
     y    = likelihood_profile$likelihood,
     xlab = "p",
     ylab = "Likelihood",
     type = "l")
abline(v = p_hat, col = 2, lwd = 2)
abline(h = exp(log(maxL)-1.92), col = 6, lwd = 2)
abline(v = CI95[1], lty = 2, col = 2, lwd = 2)
abline(v = CI95[2], lty = 2, col = 2, lwd = 2)
```

A is for Approximation (1 of 3: Simulation)

Lets assume that our experiment is decribed by a model that does not have an analytical solution. Likelihood can be estimated with Monte Carlo methods. In this context a simulation is the production of data from a probabilistic model using (pseudo)random numbers to decide the outcome. In the case of the flip coin experiment, we can simulate flipping the coin by using the binomial distribution:

```
toss_simulation <- rbinom(n = 10, size = 1, prob = p_heads)
toss_simulation[which(toss_simulation==1)] <- "H"
toss_simulation[which(toss_simulation==0)] <- "T"
```

Simulated toss: T, T, H, T, T, T, H, H, T, T.

Many simulations can be performed to estimate the probability of a given outcome (e.g. 7 heads) as the proportion of simulations with the same ourcome. The function `flip.coin.likelihood.approx(n,k,p,rep)` will simulate `rep` coin tosses to estimate the likelihood for parameter value `p` (function from file `flip_coin_likelihood.r`).

```
likelihood_approx <- flip.coin.likelihood.approx(n  =total_tosses,
                                                 k  =heads_count,
                                                 p  =p_heads,
                                                 rep=1000)
```

Estimated likelihood of $p = 0.5$ given 7 heads in 10 tosses 0.121 (true value: 0.1171875).

Exercise 2: Use function `flip.coin.likelihood.approx(n,k,p,rep)` with options `rep=10` and `trace=TRUE` to visualize some simulated coin tosses. Have a look at the code of the function.

Repeating for many values of parameter p we can estimate the likelihood profile:

```
likelihood_profile_approx <-
                 flip.coin.likelihood.approx(n   = total_tosses,
                                             k   = heads_count,
                                             p   = seq(0,1,0.01),
                                             rep = 1000)
```

Code for figure 2:

```
plot(x    = likelihood_profile$p,
     y    = likelihood_profile$likelihood,
     xlab = expression(italic(p)), ylab = "Likelihood",
     lwd  = 2, type = "l")
lines(x   = likelihood_profile_approx$p ,
      y   = likelihood_profile_approx$likelihood,
      col = 2, lwd = 2)
```

Exercise 3: Estimate the likelihood profile for the observed coin toss using different number of simulations to evaluate the effect in the accuracy of the estimate.

B is for Bayesian[2]

The posterior probability is the probability of the parameter p given the data D: $P(p|D)$. Using Bayes' Theorem: $P(p|D) = \frac{P(D|p)P(p)}{P(D)}$, where $P(D|p) = L(p|D)$ is the likelihood and $P(p)$ is the prior probability. The marginal likelihood, $P(D)$, is constant and does not need to be calculated: $P(p|D) \propto L(p|D)P(p)$.

We already know how to estimate the likelihood from simulations for a given value of p ($L(p = 0.5|D = 7\text{H}) \approx 0.121$, as estimated above).

Lets try to estimate the prior probability with the same strategy as the likelihood. The previous knowledge or beliefs about the parameter are expressed in the form of the prior probability distribution. For the moment we are going to assume that all possible values of p have the same probability, that is, a uniform distribution $p \sim \text{U}(0, 1)$. We can take values from that distribution and count how many times we get the $p = 0.5$):

```
total_draws     <- 10000
sim_parameter_p <- runif(total_draws,0,1)
prior_p_approx  <- sum(sim_parameter_p==p_heads)/total_draws
```

A proportion of 0 draws had exactly the value $p = 0.5$. Indeed, the prior probability for a given value of p is 0, $P(p = 0.5) = 0$; remember that a probability distribution gives you the probability **density**. We need to change a litttle bit the strategy, using the rejection algorithm:

Algorithm 1 Exact rejection sampler

Given N the number of simulations
for $i = 1$ to N **do**
  Generate $p' \sim \pi(p)$
  Simulate $D' \sim f(x|p')$
  if $D' = D$ **then**
    Accept p'
  end if
end for
return accepted p'

In R:

```
sim_parameter_p1 <- runif(10000,0,1)
sim_data_p1      <- rbinom(n    = length(sim_parameter_p1),
                           size = total_tosses,
                           prob = sim_parameter_p1)
sim_parameter_kept_p1 <- sim_parameter_p1[which(sim_data_p1==heads_count)]
sim_data_kept_p1      <- sim_data_p1[which(sim_data_p1==heads_count)]
```

[2] But see Rousset et al. [2017]

The values of p from the simulations kept after applying the rejection algorithm are used to estimate the posterior probability distribution. In this simple case (coin flipping experiment) we did not need to estimate it is known when using a conjugate prior such as the beta distribution, $\mathrm{B}(\alpha, \beta)$. Thus, the posterior distribution is $\mathrm{B}(\alpha + \#\text{heads}, \beta + \#\text{tails})$, in our case $\mathrm{B}(1+7, 1+3)$. Note that the uniform distribution $\mathrm{U}(0, 1)$ is the same as beta distribution $\mathrm{B}(1, 1)$. Code for figure 3:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot(x    = sim_parameter_p1,
     y    = sim_data_p1,
     xlab = expression(italic(p)*"'"),
     ylab = expression(italic(D)*"'"))
abline(h = heads_count, col = 7)
points(sim_parameter_kept_p1, sim_data_kept_p1, col = 7)

hist(x      = sim_parameter_p1,
     breaks = seq(0,1,0.02),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,5),
     main   = "",
     xlab   = expression(italic(p)),
     ylab   = "probability density")
hist(x      = sim_parameter_kept_p1,
     breaks = seq(0,1,0.02),
     col    = Vermillion_transparency,
     freq   = FALSE,
     add    = TRUE); box()
lines(x   = seq(0,1,0.001),
      y   = dbeta(x=seq(0,1,0.001),1+heads_count,1+tails_count),
      col = 7,
      lwd = 3)
```

From the prior probability distribution we can obtain a point estimates (usually the median) and 95% highest posterior denstity:

```
p_hat  <- median(sim_parameter_kept_p1)
p_95CI <- quantile(sim_parameter_kept_p1,probs=c(0.025,0.975))
```

Point estimate of parameter is $\hat{p} = 0.67$ with (0.39, 0.89) 95% credibility interval.

Exercise 4: Using a beta distribution as a prior, re-analyse the data with a prior believe that the coin is fair: We have examinated the coin before the experiment and we did not notice anything abnormal; in addition we have read Diaconis et al. [2007].

C is *not* for Coalescent

In order to advance we leave our toy example of the coin toss and continue with a population genetics example. We are going to study two fictitious datasets consisting in resequencing experiments of 50 gene copies of a non recombining locus (e.g. a mitochondrial gene). The data is in `ms` format [Hudson, 2002] in folder `data`, files `dataset1.txt` and `dataset2.txt`.

We can have a look at the data. We will use `Sampling` [file `ms.r` Städler et al., 2009a,b, with some modifcications by myself], which is a `R` script with functions to read files in `ms` format and calculate summary statistics (number of segregating sites, `S`; nucleotide diversity, `pi`; number of haplotypes, `NH`; site frequency spectrum, `SFS`; Tajima's D, `TajimasD`; Fay and Wu H, `FayWuH`; Fu and Li D, `FuLiD`):

```
source("src/ms.r")
seq_data <- ms.inp.multi(sample_size, 1,
                         ms.output.file="data/dataset1.txt")
```

```
target1_S        <- S(seq_data)
target1_pi       <- thetaPi(seq_data)
target1_NH       <- NH(seq_data)
target1_SFS      <- SFS(seq_data)
target1_TajimasD <- tajimaD(seq_data, thetaW(seq_data), target1_pi)
target1_FayWuH   <- fayWuH(seq_data)
target1_FuLiD    <- fuliD(seq_data, thetaS1(seq_data) )
```

Dataset 1 has a sample size of 50 gene copies, with 50 polymorphic sites and 21 different haplotypes. The site frequency spectrum from this sample is represented in figure 4:

```
colnames(target1_SFS)<-1:(sample_size-1)
barplot(height = target1_SFS/target1_S,
        main   = "Unfolded Site Frequency Spectrum",
        xlab   = "derived allele count in sample",
        ylab   = "Proportion of sites")
box()
```

Exercise 5: Calculate summary statistics for Dataset 2. What are the main differences with dataset 1?

The generating model for such data is the coalescent. We will be using a version of `ms` coalescent simulator [Hudson, 2002] implemented in the R package `phyclust` [Chen, 2011]. This is a choice to make the present tutorial available to different operative systems (i.e. Windows).

A is for Approximation (2 of 3: Summary Statistics)

We have, thus, a new tyope of data (DNA sequences) and its corresponding data simulator (the coalescent). We could try to apply the same rejection algorithm

described above, but the structure of the data is more complex and an exact match would occur rarely in simulations. So we use the second approximation of ABC, instead of the data we use summary statistics (S) from the data to classify the simulation as a match to the observation. It is said now that we are approximating the likelihood of p given the summary statistic, $L(p|S)$, rather than the likelihood of p given the data, $L(p|D)$.

Algorithm 2 Exact rejection sampler on summary statistics

```
Calculate S from D
Given N the number of simulations
for i = 1 to N do
  Generate p' ~ π(p)
  Simulate D' ~ f(x|p')
  Calculate S' from D'
  if S' = S then
    Accept p'
  end if
end for
return accepted p'
```

```
nsim <- 10000
sim_theta <- 10^runif(nsim,min=-1,max=2) # log uniform prior

if (file.exists("data/abc_sims1.txt")){
  file_removed <- file.remove("data/abc_sims1.txt")
}
ms(nsam       = sample_size,
   opt        = "-t tbs",
   tbs.matrix = cbind(sim_theta),
   temp.file  = "data/abc_sims1.txt")

msout <- ms.inp.multi(sample_size,
                      ndraws=nsim,
                      ms.output.file="data/abc_sims1.txt")

## Warning in onlydoubles(txt, (marker + 1)):  NAs introduced by coercion
```

```
sim1_S <- S(msout)
```

We can apply this algorthim to our Dataset 1, simulating with the coalescent simulator and using the number of segregating sites (number of polymorphisms) as summary statistic:

```
target_SS <- target1_S
sim_SS    <- sim1_S
sim_theta_kept <- sim_theta[which(sim_SS==target_SS)]
sim_SS_kept    <- sim_SS[which(sim_SS==target_SS)]
```

Code for figure 5:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot( x    = log10(sim_theta),
      y    = sim_SS,
      xlab = expression(log[10](theta*"'")),
      ylab = "SS'",
      ylim = c(0,100))
abline(h = target_SS, col = 7)
points(log10(sim_theta_kept), sim_SS_kept, col = 7)

hist(x      = log10(sim_theta),
     breaks = seq(-1,2,0.05),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,6),
     main   = "",
     xlab   = expression(log[10](theta)),
     ylab   = "probability density")
hist(x      = log10(sim_theta_kept),
     breaks = seq(-1,2,0.05),
     col    = Vermillion_transparency,
     freq   = FALSE,
     add    = TRUE); box()
```

The number of simulations kept is 34 out of 9032.

Exercise 6: Apply ABC rejection algorithm to Dataset 2 using a different summary statistic, such as the nucletide diversity, π (`pi`). Repeat using several summary statistics.

A is for Approximation (3 of 3: Tolerance)

Summarizing the data into statistics is not enough (in practice) to allow a big enough number of simulations to match the observed data. The third approximation in ABC allows to accept simulations that are not an exact match but are in the close neighbourhood of the observation.

Algorithm 3 Exact rejection sampler on summary statistics

Calculate S from D
Given N the number of simulations
for $i = 1$ to N **do**
  Generate $p' \sim \pi(p)$
  Simulate $D' \sim f(x|p')$
  Calculate S' from D'
  if distance$(S', S) < \delta$ **then**
    Accept p'
  end if
end for
return accepted p'

From this point we are goint to start using functions from the `R` package `abc` to present a more compact code:

```
tolerance  <- 0.1
target_SS  <- target1_pi
sim_SS     <- as.matrix(sim1_pi); colnames(sim_SS) <- "pi"
abc_result <- abc(target  = target_SS,
                  param   = sim_theta,
                  sumstat = sim_SS,
                  tol     = tolerance,
                  method  = "rejection")
sim_theta_kept_tol1 <- as.vector(abc_result$unadj.values)
sim_SS_kept_tol1    <- abc_result$ss
```

Code for figure 6:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot(x    = log10(sim_theta),
     y    = sim_SS,
     xlab = expression(log[10](theta*"'")),
     ylab = "SS'",
     ylim = c(0,50))
abline(h = target_SS, col = 7)
abline(h = max(abc_result$ss), col = 7, lty = 2)
abline(h = min(abc_result$ss), col = 7, lty = 2)
points(log10(sim_theta_kept_tol1), sim_SS_kept_tol1, col = 7)

hist(x      = log10(sim_theta),
     breaks = seq(-1,2,0.05),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,4),
     main   = "",
     xlab   = expression(log[10](theta)),
     ylab   = "probability density")
```

```
hist(x      = log10(sim_theta_kept_tol1),
     breaks = seq(-1,2,0.05),
     col    = Vermillion_transparency,
     freq   = FALSE,
     add    = TRUE); box()
```

The number of simulations kept is 904 out of 9032; determined by tolerance of 0.1.

Exercise 7: Reanalyse the data using different tolerance values. What happens when we get tolerance values close to 1?

Regression

Introduced by Beaumont et al. [2002]

```
tolerance  <- 0.5
target_SS  <- target1_pi
sim_SS     <- as.matrix(sim1_pi); colnames(sim_SS) <- "pi"
abc_result <- abc(target  = target_SS,
                  param   = sim_theta,
                  sumstat = as.matrix(sim_SS),
                  tol     = tolerance,
                  transf  = "log",
                  method  = "loclinear")

sim_theta_kept     <- as.vector(abc_result$unadj.values)
sim_theta_adjusted <- as.vector(abc_result$adj.values)
sim_SS_kept        <- abc_result$ss
local_regression   <- lm(log10(sim_theta_kept)~sim_SS_kept,
                         weights=abc_result$weights)
```

Code for figure 7:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot(x    = log10(sim_theta),
     y    = sim_SS,
     xlab = expression(log[10](theta*"'")),
     ylab = "SS'",
     ylim = c(0,30))
abline(h = target_SS, col = 6)
{abline(h = max(abc_result$ss), col = 6, lty = 2)
 abline(h = min(abc_result$ss), col = 6, lty = 2)}
points(log10(sim_theta_kept), sim_SS_kept, col = 6)
abline(a   = -local_regression$coefficients[1]/local_regression$coefficients[2],
       b   = 1/local_regression$coefficients[2],
```

```
       col = 5,
       lwd = 3)
points(x   = log10(sim_theta_kept)[1:10],
       y   = sim_SS_kept[1:10],
       col = 5)
arrows(x0     = log10(sim_theta_kept)[1:10],
       y0     = sim_SS_kept[1:10],
       x1     = log10(sim_theta_adjusted)[1:10],
       y1     = target_SS,
       col    = 5,
       length = 0.1)
hist(x      = log10(sim_theta),
     breaks = seq(-1,2,0.05),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,4),
     main   = "",
     xlab   = expression(log(theta)),
     ylab   = "probability density")
hist(x      = log10(sim_theta_kept),
     breaks = seq(-1,2,0.05),
     col    = Blue_transparency, freq=FALSE, add=TRUE )
wtd.hist(x      = log10(sim_theta_adjusted),
         breaks = seq(-1.5,2,0.05),
         col    = Yellow_transparency, freq=FALSE, add=TRUE,
         weight = abc_result$weights); box()
```

Information on Summary Statistics

Now that we have determined that using the rejection plus regression approach is the way to go we can apply it to estimate θ:

```
target_SS <- target1_TajimasD
sim_SS    <- cbind(sim1_TajimasD); colnames(sim_SS) <- "Tajima's D"
abc_result <- abc(target  = target_SS,
                  param   = sim_theta,
                  sumstat = as.matrix(sim_SS),
                  tol     = tolerance,
                  transf  = "log",
                  method  = "loclinear")
sim_theta_adjusted_D <- as.vector(abc_result$adj.values)
sim_weights_D        <- abc_result$weights
```

With a very unsatisfactory result (figure 8).

```
hist(x      = log10(sim_theta),
     breaks = seq(-1,2,0.05),
     col    = "grey",
```

```
     freq   = FALSE,
     ylim   = c(0,4),
     main   = "",
     xlab   = expression(log[10](theta)),
     ylab   = "probability density")
wtd.hist(x      = log10(sim_theta_adjusted_D),
         breaks = seq(-2,3,0.05),
         col    = BluishGreen_transparency,
         freq   = FALSE, add = TRUE,
         weight = sim_weights_D); box()
```

Exercise 8: Make a scatterplot for simulated values of θ and Tajima's D. Reanalyse the data using a summary statistics that is informative for parameter θ.

For a quick and dirty evaluation of the informativeness of these summary statistics with the parameter θ we can look at the correlation coeficient:

```
cor(sim1_TajimasD,as.vector(sim_theta))

## [1] -0.01377705

cor(sim1_pi,as.vector(sim_theta))

## [1] 0.8579711
```

Note, however, that Tajima's D might not be very informative for θ on its own, but it is infromative in combination with other summary statistics:

```
target_SS <- cbind(target1_pi,
                   target1_TajimasD)
sim_SS    <- cbind(sim1_pi,
                   sim1_TajimasD)
colnames(target_SS) <- colnames(sim_SS) <- c("pi","TD")
abc_result <- abc(target  = target_SS,
                  param   = sim_theta,
                  sumstat = sim_SS,
                  tol     = tolerance,
                  transf  = "log",
                  method  = "loclinear")
sim_theta_adjusted_piD <- as.vector(abc_result$adj.values)
sim_weights_piD        <- abc_result$weights
```

Which gives us a nicer result (figure 9).

```
hist(x      = log10(sim_theta),
     breaks = seq(-1,2,0.05),
     col    = "grey",
     freq   = FALSE,
```

```
     ylim   = c(0,4),
     main   = "",
     xlab   = expression(log[10](theta)),
     ylab   = "probability density")
wtd.hist(x      = log10(sim_theta_adjusted_piD),
         breaks = seq(-2,3,0.05),
         col    = BluishGreen_transparency,
         freq   = FALSE, add = TRUE,
         weight = sim_weights_piD); box()
```

However, adding summary statistics can make worse the results, when the summary statistic is completely random respect to the parameter:

```
target1_noise <- 76
sim_noise     <- 10^runif(length(sim1_pi),-2,3)
target_SS <- cbind(target1_noise,
                   target1_pi)
sim_SS     <- cbind(sim_noise,
                    sim1_pi)
colnames(target_SS) <- colnames(sim_SS) <- c("noise","pi")
abc_result <- abc(target  = target_SS,
                  param   = sim_theta,
                  sumstat = as.matrix(sim_SS),
                  tol     = tolerance,
                  transf  = "log",
                  method  = "loclinear")
sim_theta_adjusted_pi_noise <- as.vector(abc_result$adj.values)
sim_weights_pi_noise        <- abc_result$weights
```

Code for figure 10:

```
hist(x      = log10(sim_theta),
     breaks = seq(-1,2,0.05),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,4),
     main   = "",
     xlab   = expression(log[10](theta)),
     ylab   = "probability density")
wtd.hist(x      = log10(sim_theta_adjusted_pi_noise),
         breaks = seq(-2,2,0.05),
         col    = ReddishPurple_transparency,
         freq   = FALSE, add = TRUE,
         weight = sim_weights_pi_noise); box()
```

```
## Error in wtd.hist(x = log10(sim_theta_adjusted_pi_noise), breaks
= seq(-2, :  some 'x' not counted; maybe 'breaks' do not span range
of 'x'
```

Beware of the Prior

Priors, as discussed above, can be used to incroporate previous information on the parameters. In the case of our population genetics example, we could imagine to have some information on mutation rates (e.g. from pedigrees) and population sizes (from census). Since $\theta = 4N_e\mu$, we could draw parameters N_e and μ from prior probability distributions and combine them to get the prior of θ:

```
sim_mu <- 10^runif(nsim,min=-7.5,max=-6)
sim_Ne <- 10^runif(nsim,min=6,max=7.5)
sim_theta_composite <- 4*sim_Ne*sim_mu
```

Which will give you a prior for θ with a peak, as in figure 11:

```
hist(x      = log10(sim_theta_composite),
     breaks = seq(-2,3,0.05),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,1),
     xlim   = c(-1,2),
     main   = "",
     xlab   = expression(log[10](theta)),
     ylab   = "probability density"); box()
```

ALWAYS COMPARE POSTERIOR AND PRIOR PROBABILITY DISTRIBUTIONS!!!

Exercise 9: Estimate θ using appropiate ABC algorithm, prior and summary statistics for Dataset 1 and 2.

Quality Control 1: Cross Validation

The package `abc` includes a function for cross validation:

```
cross_validation_result <- cv4abc(param   = sim_theta,
                                  sumstat = as.matrix(sim_SS),
                                  abc.out = abc_result1,
                                  nval    = 100,
                                  tols    = 0.1)
```

figure 12:

```
plot(x    = log10(cross_validation_result$true$theta),
     y    = log10(cross_validation_result$estim$tol0.1),
     xlab = expression(log(theta)),
     ylab = expression(log(hat(theta))),
     xlim = c(-1,2), ylim = c(-1,2))
abline(a = 0, b = 1)
```

Quality Control 2: Goodness of fit (prior)

Are simulations from the model and priors producing data similar to observed data?

figure 13:

```
par(mfrow=c(3,1),mar=c(4.2,4.2,1,1))
plot(sim1_S,sim1_pi,xlab="S",ylab=expression(pi),log="xy")
points(target1_S,target1_pi,col=7,cex=4,pch="*")
points(target2_S,target2_pi,col=6,cex=4,pch="*")

plot(sim1_NH,sim1_pi,xlab="Number of Haplotypes",ylab=expression(pi),log="xy")
points(target1_NH,target1_pi,col=7,cex=4,pch="*")
points(target2_NH,target2_pi,col=6,cex=4,pch="*")

plot(sim1_FuLiD,sim1_pi,xlab="Fu and Li's D",ylab=expression(pi),log="y",xlim=c(-6,2))
points(target1_FuLiD,target1_pi,col=7,cex=4,pch="*")
points(target2_FuLiD,target2_pi,col=6,cex=4,pch="*")
```

```
par(mfrow=c(1,1),mar=c(5.1,4.1,4.1,2.1))
```

Quality Control 3: Goodness of fit (posterior)

We can also verify the goodness of fit of the inferred parameters. We can sample from the posterior distribution by using the regression-adjusted values of the retained simulations. We can re-simulate data from those values to obtain the expected distribution of summary statistics under the posterior and compare it with the observation:

```
posterior_theta <- sample(sim_theta_adjusted_1,
                          size    = 1000,
                          replace = TRUE,
                          prob    = sim_weights_1)
if (file.exists("data/abc_sims_posterior.txt")){
  file_removed <- file.remove("data/abc_sims_posterior.txt")
}
ms(nsam       = sample_size,
   opt        = "-t tbs",
   tbs.matrix = cbind(posterior_theta),
   temp.file  = "data/abc_sims_posterior.txt")
msout <- ms.inp.multi(sample_size, 1000,
                      ms.output.file="data/abc_sims_posterior.txt")
```

```
posterior_S        <- S(msout)
posterior_pi       <- thetaPi(msout)
posterior_NH       <- NH(msout)
posterior_TajimasD <- tajimaD(msout, thetaW(msout), posterior_pi)
posterior_FayWuH   <- fayWuH(msout)
posterior_FuLiD    <- fuliD(msout, thetaS1(msout) )
```

figure 14:

```
par(mfrow=c(2,1))
hist(x      = posterior_NH,
     breaks = seq(0,40,1),
     col    = "grey",
     freq   = FALSE,
     xlab   = "Number of haplotypes", main = "")
abline(v = target1_NH, col = 7, lwd = 2); box()
hist(x      = posterior_FuLiD,
     breaks = seq(-5,4,0.2),
     col    = "grey",
     freq   = FALSE,
     xlab   = "Fu and Li's D", main = "")
abline(v = target1_FuLiD, col = 7, lwd = 2); box()
```

Exercise 10: Control goodness of fit of posterior probability distribution for Dataset 2.

Model Choice

We are going to consider two models. A constant size model with parameter $\theta = 4N_e\mu$ and a population with one instantaneous change of size with parameters $\theta_0 = 4N_{e0}\mu$ (present scaled mutation rate), $\tau = t\mu$ (mutation scaled time of population size change) and $\theta_1 = 4N_{e1}\mu$ (past scaled mutation rate). We already have simulations for the first model, we can add the simulations for the second model.

```
nsim <- 10000
sim_theta0 <- 10^runif(nsim,min=-1,max=2)
sim_theta1 <- 10^runif(nsim,min=-1,max=2)
sim_tau    <- runif(nsim,min=0,max=2)

ref_table_params <- cbind(sim_theta0,sim_theta1,sim_tau)
colnames(ref_table_params) <- c("theta0","theta1","tau")

tbs_params <-  cbind(ref_table_params[,"theta0"],
                     ref_table_params[,"tau"]/ref_table_params[,"theta0"],
                     ref_table_params[,"theta1"]/ref_table_params[,"theta0"])

if (file.exists("data/abc_sims2.txt")){
  file_removed <- file.remove("data/abc_sims2.txt")
}
ms(nsam       = sample_size,
   opt        = "-t tbs -eN tbs tbs",
   tbs.matrix = tbs_params,
   temp.file  = "data/abc_sims2.txt")
```

```
msout <- ms.inp.multi(sample_size, nsim, ms.output.file="data/abc_sims2.txt")

## Warning in onlydoubles(txt, (marker + 1)):  NAs introduced by coercion
```

```
sim2_S        <- S(msout)
sim2_pi       <- thetaPi(msout)
sim2_NH       <- NH(msout)
sim2_TajimasD <- tajimaD(msout, thetaW(msout), sim2_pi)
sim2_FayWuH   <- fayWuH(msout)
sim2_FuLiD    <- fuliD(msout, thetaS1(msout) )
```

Quality control: goodness of fit for model + prior:
figure 15:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot(sim2_S,sim2_pi,xlab="S",ylab=expression(pi),log="xy")
points(sim1_S,sim1_pi,col="grey")
points(target1_S,target1_pi,col=7,cex=3,pch="*")
points(target2_S,target2_pi,col=6,cex=3,pch="*")

plot(sim2_FuLiD,sim2_pi,xlab="Fu and Li's D",ylab=expression(pi),log="y",xlim=c(-6,2))
points(sim1_FuLiD,sim1_pi,col="grey")
points(target1_FuLiD,target1_pi,col=7,cex=3,pch="*")
points(target2_FuLiD,target2_pi,col=6,cex=3,pch="*")
```

We put together the two reference tables.

```
nsim <- min(length(sim1_S),length(sim2_S))

sim_model    <- c(array("C",nsim),array("V",nsim))
sim_S        <- c(sim1_S[1:nsim],sim2_S[1:nsim])
sim_pi       <- c(sim1_pi[1:nsim],sim2_pi[1:nsim])
sim_NH       <- c(sim1_NH[1:nsim],sim2_NH[1:nsim])
sim_TajimasD <- c(sim1_TajimasD[1:nsim],sim2_TajimasD[1:nsim])
sim_FayWuH   <- c(sim1_FayWuH[1:nsim],sim2_FayWuH[1:nsim])
sim_FuLiD    <- c(sim1_FuLiD[1:nsim],sim2_FuLiD[1:nsim])
```

And we performed an ABC rejection+regression on the model. The main differences is that it is a logistic regression because the model is a qualitative variable:

```
tolerance <- 0.1
target1_SS <- cbind(target1_S,target1_pi,target1_NH,target1_TajimasD,target1_FayWuH,target
target2_SS <- cbind(target2_S,target2_pi,target2_NH,target2_TajimasD,target2_FayWuH,target
sim_SS    <- cbind(sim_S,sim_pi,sim_NH,sim_TajimasD,sim_FayWuH,sim_FuLiD)
colnames(target1_SS) <- colnames(target2_SS) <-
                        colnames(sim_SS) <- c("S","pi","NH","TD","FWH","FLD")
```

```
abc_model_choice1 <- postpr(target=target1_SS,
                            index=sim_model,
                            sumstat=sim_SS,
                            tol=tolerance,
                            method="mnlogistic")

abc_model_choice1$pred

##         C         V
## 0.5552396 0.4447604

abc_model_choice2 <- postpr(target=target2_SS,
                            index=sim_model,
                            sumstat=sim_SS,
                            tol=tolerance,
                            method="mnlogistic")

abc_model_choice2$pred

##           C           V
## 0.001377722 0.998622278
```

The best model for Dataset 1 is C (C for constant, V for varable population size), with a posterior probability of 0.5552396

The best model for Dataset 2 is V (C for constant, V for varable population size), with a posterior probability of 0.9986223

ABC random forest

One tree: classification and regression tree

A decission tree is a simple machine learning model for target quantitative (regression) or qualitative (classification) variables:

```
theta  <- ref_table_params[,"theta1"]
TajD   <- sim2_TajimasD
pi     <- sim2_pi
ref_table <- data.frame(theta,TajD,pi)

regression_tree <- tree(theta ~ TajD + pi,
                        data=ref_table)
```

figure 16:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot(regression_tree)
text(regression_tree,cex=0.75)

plot(ref_table$TajD,
```

```
     ref_table$pi,
     xlab="Tajima's D",
     ylab=expression(pi),
     col=cbPalette1[round(log10(theta)+3)],
     #col=grey(1-theta/max(theta)),
     pch=20)
partition.tree(regression_tree,
               ordvars=c("TajD","pi"),
               add=T,cex=1)
```

```
TajD   <- sim_SS[,"TD"]
FuLiD  <- sim_SS[,"FLD"]
ref_table <- data.frame(sim_model,TajD,FuLiD)

classification_tree <- tree(sim_model ~ TajD + FuLiD,
                            data=ref_table)
```

figure 17:

```
par(mfrow=c(2,1),mar=c(4.2,4.2,1,1))

plot(classification_tree)
text(classification_tree,cex=0.75)

plot(ref_table$TajD[which(sim_model=="V")],
     ref_table$FuLiD[which(sim_model=="V")],
     xlab="Tajima's D",
     ylab="Fu and Li's D",
     pch=20)
points(ref_table$TajD[which(sim_model=="C")],
       ref_table$FuLiD[which(sim_model=="C")],
       col="grey",pch=20)
partition.tree(classification_tree,
               ordvars=c("TajD","FuLiD"),
               add=T,cex=3,col=7)
```

Many trees: random forest

```
theta     <- sim_theta
pi        <- sim1_pi

ref_table <- data.frame(theta,pi)
regression_tree <- tree(log10(theta) ~ pi,
                        data=ref_table)
```

figure 18:

```
par(mfrow=c(3,1),mar=c(4.2,4.2,1,1))
plot(regression_tree)
text(regression_tree,cex=0.75)

plot(ref_table$pi,
     log10(ref_table$theta),
     xlab=expression(pi),
     ylab=expression(log[10]*theta),
     pch=20,log="x")
partition.tree(regression_tree,
               ordvars=c("pi","theta"),
               add=T,cex=1.5,col=6,lwd=2)

plot(ref_table$pi,
     log10(ref_table$theta),
     xlab=expression(pi),
     ylab=expression(log[10]*theta),
     pch=20,log="x")
for (i in 1:100){
  random_sample <- sample(length(theta),size=500,replace=T)
  ref_table_random_sample <- ref_table[random_sample,]
  regression_tree_random_sample <- tree(log10(theta) ~ pi,
                                        data=ref_table_random_sample)
  partition.tree(regression_tree_random_sample,
                 ordvars=c("pi","theta"),
                 add=T,cex=1.5,col=7,lwd=1)

}
```

Getting posterior probabilities from RF

As we have seen above, Random Forest allows you to classify your observation in a category (i.e. model choice) and obtain an estimate of a quantitative variable (i.e. parameter estimation). But these values are not associated to posterior probabilities. The package `abcrf` [Marin et al., 2017] implements the development that allow to get these probabilities using random forest.

For model choice, random forest provides you with the number of votes supporting each of the models (number of votes = number of trees leading to a given model). However, the number of votes is not necessary a good quantitative measure of the incertitude of the model choice. Pudlo et al. [2016] developped the approach to estimate the posterior probability. First, a random forest is grown for model choice:

```
ref_table <- data.frame(sim_model,sim_SS)
model_RF <- abcrf(formula = sim_model~.,
                  data    = ref_table,
                  lda     = F,
                  ntree   = 1000,
```

```
                  paral    = T)
```

figure 19:

```
plot(model_RF,
     training=ref_table)
```

For each simulation of the reference table we can get a prediction from the random forest (only from trees grown without that simulation). Comparing with the true value we can get the prior error rate and the confusion matrix:

```
# GLOBAL ERROR
as.factor(sim_model[1:20])

##  [1] C C C C C C C C C C C C C C C C C C C C
## Levels: C

model_RF$model.rf$predictions[1:20]

##  [1] C V C C C V V C C V C V C C C C C C C C
## Levels: C V

sum(as.character(model_RF$model.rf$predictions)!=sim_model)/length(sim_model)

## [1] 0.3397919

model_RF$model.rf$confusion.matrix

##      C    V class.error
## C 6553 2479   0.2744686
## V 3659 5373   0.4051151

# LOCAL ERROR
as.numeric(as.character(model_RF$model.rf$predictions[1:20])!=sim_model[1:20])

##  [1] 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 0

local_error <- as.numeric(as.character(model_RF$model.rf$predictions)!=sim_model)
```

Now we can grow a **second** random forest which will learn the relationship between the "local" error rate and the summary statistics. The probability of a correct classification (posterior probability) is one minus the probability of an incorrect classification (error rate). This second random forest allows to estimate the posterior probability of the **chosen** model (implemented in function `predict.abcrf()` from `abcrf`):

```
model_selection_result_RF <- predict(object         = model_RF,
                                     obs            = as.data.frame(rbind(target1_SS,targe
                                     training       = ref_table,
                                     ntree          = 1000,
                                     paral          = T,
```

```
                                      paral.predict  = T)
(model_selection_result_RF)

##   selected model votes model1 votes model2 post.proba
## 1              C        0.717        0.283  0.7051333
## 2              V        0.010        0.990  0.9865000
```

For the posterior probability distribution of a parameter, the key thing to remember is that prediction of quantitative variables from random forest is based on the weight given by the trees to each simulation of the reference table (i.e. the rediction is a weigthed mean). The same weigths can be applied to the median and percentiles to obtain a point estimate and credibility intervals, and to plot an estimate of the distribution (e.g. an histogram) [Raynal et al., 2017]:

```
log10theta <- log10(sim_theta)
sim_SS <- cbind(sim1_S,sim1_pi,sim1_NH,
                sim1_TajimasD,sim1_FayWuH,sim1_FuLiD)
ref_table <- data.frame(log10theta,sim_SS)
colnames(ref_table) <- c("log10theta","S","pi","NH","TD","FWH","FLD")
RFmodel_theta <- regAbcrf(formula = log10theta~.,
                          data    = ref_table,
                          ntree   = 1000,
                          paral   = T)
```

```
posterior_theta_RF <- predict(object    = RFmodel_theta,
                              obs       = as.data.frame(target1_SS),
                              training  = ref_table,
                              paral     = T,
                              rf.weights = T)
(posterior_theta_RF)

##      expectation   median   variance variance.cdf quantile= 0.025
## [1,]    1.073908 1.063332 0.01247689   0.01070793       0.9114225
##      quantile= 0.975
## [1,]        1.297972
```

figure 20:

```
hist(x      = log10theta,
     breaks = seq(-1,2,0.05),
     col    = "grey",
     freq   = FALSE,
     ylim   = c(0,5),
     main   = "",
     xlab   = expression(log(theta)),
     ylab   = "probability density")
wtd.hist(x      = log10theta,
         breaks = seq(-1,2,0.05),
         col    = ReddishPurple_transparency, freq=FALSE, add=TRUE,
         weight = posterior_theta_RF$weights); box()
```

Some further topics

Other ABC flavours: SMC-ABC

Simulation software: msprime [Kelleher et al., 2016], SLiM [Haller and Messer, 2017].

There are several reviews on abc: [Csilléry et al., 2010], [Bertorelle et al., 2010]

Other applications than population genetics: phylogenetics [Lintusaari et al., 2016], community ecology, epidemiology, (astronomy)

Acknowledgements

I was initiated to the use of `R` for ABC by M. Beaumont, L. Chikhi and V. Sousa at the Gulbenkian course MMPG08.

References

M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, Dec. 2002.

G. Bertorelle, A. Benazzo, and S. Mona. ABC as a flexible framework to estimate demography over space and time: some cons, many pros. *Molecular Ecology*, 19(13):2609–2625, 2010. doi: 10.1111/j.1365-294X.2010.04690.x.

W.-C. Chen. *Overlapping Codon Model, Phylogenetic Clustering, and Alternative Partial Expectation Conditional Maximization Algorithm*, 2011. URL `http://gradworks.umi.com/34/73/3473002.html`.

K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François. Approximate Bayesian computation (ABC) in practice. *Trends in Ecology & Evolution*, 25(7):410–418, 2010. doi: 10.1016/j.tree.2010.04.001.

P. Diaconis, S. Holmes, and R. Montgomery. Dynamical bias in the coin toss. *SIAM Review*, 49(2):211–235, 2007. doi: 10.1137/S0036144504446436.

B. C. Haller and P. W. Messer. SLiM 2: Flexible, interactive forward genetic simulations. *Molecular Biology and Evolution*, 34(1):230–240, 2017. doi: 10.1093/molbev/msw211.

R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002. doi: 10.1093/bioinformatics/18.2.337.

J. Kelleher, A. M. Etheridge, and G. McVean. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLOS Computational Biology*, 12(5):e1004842, 2016. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004842.

J. Lintusaari, M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, 2016. doi: 10.1093/sysbio/syw077.

J.-M. Marin, L. Raynal, P. Pudlo, C. P. Robert, and A. Estoup. *abcrf: Approximate Bayesian Computation via Random Forests*, 2017. URL https://CRAN.R-project.org/package=abcrf. R package version 1.7.

P. Pudlo, J.-M. Marin, A. Estoup, J.-M. Cornuet, M. Gautier, and C. P. Robert. Reliable ABC model choice via random forests. *Bioinformatics*, 32(6):859–866, 2016. doi: 10.1093/bioinformatics/btv684.

L. Raynal, J.-M. Marin, P. Pudlo, M. Ribatet, C. P. Robert, and A. Estoup. ABC random forests for Bayesian parameter inference. *arXiv*, 1605.05537v4, 2017. peer-reviewed by *Peer Community in Evolutionary Biology* doi:10.24072/pci.evolbiol.100036.

F. Rousset, A. Gouy, C. Martinez-Almoyna, and A. Courtiol. The summary-likelihood method and its implementation in the Infusion package. *Molecular Ecology Resources*, 17:110–119, 2017. doi: 10.1111/1755-0998.12627.

T. Städler, B. Haubold, C. Merino, W. Stephan, and P. Pfaffelhuber. Sampling version 0.5: Software for simulating haplotypes in non-equilibrium subdivided populations, 2009a. URL http://guanine.evolbio.mpg.de/sampling/.

T. Städler, B. Haubold, C. Merino, W. Stephan, and P. Pfaffelhuber. The impact of sampling schemes on the site frequency spectrum in nonequilibrium subdivided populations. *Genetics*, 182(1):205–216, 2009b. doi: 10.1534/genetics.108.094904.

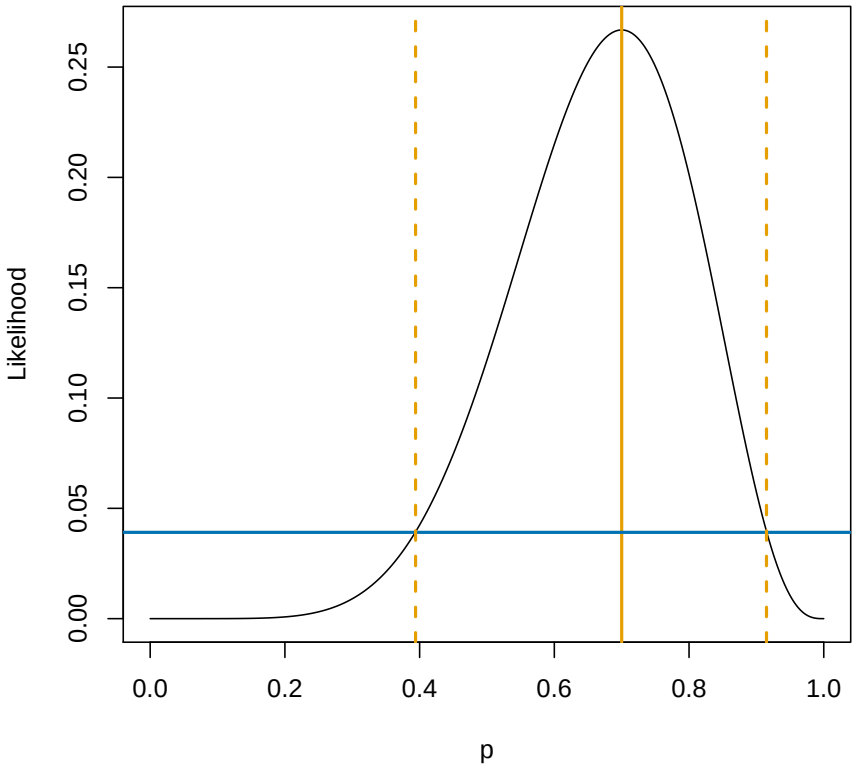


Figure 1: Likelihood profile for the flip coin experiment. Continous orange line indicates maximum likelihood value and dashed orange lines show 95%CI.

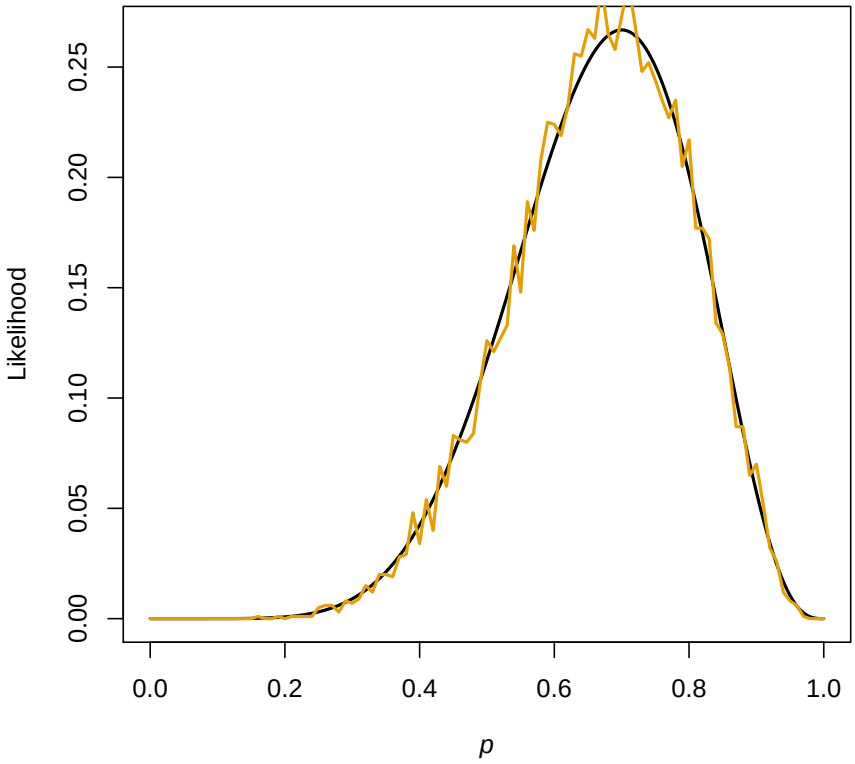


Figure 2: Estimated likelihood profile (orange) for a flip coin experiment by simulation. Black line shows the targeted likelihood profile.

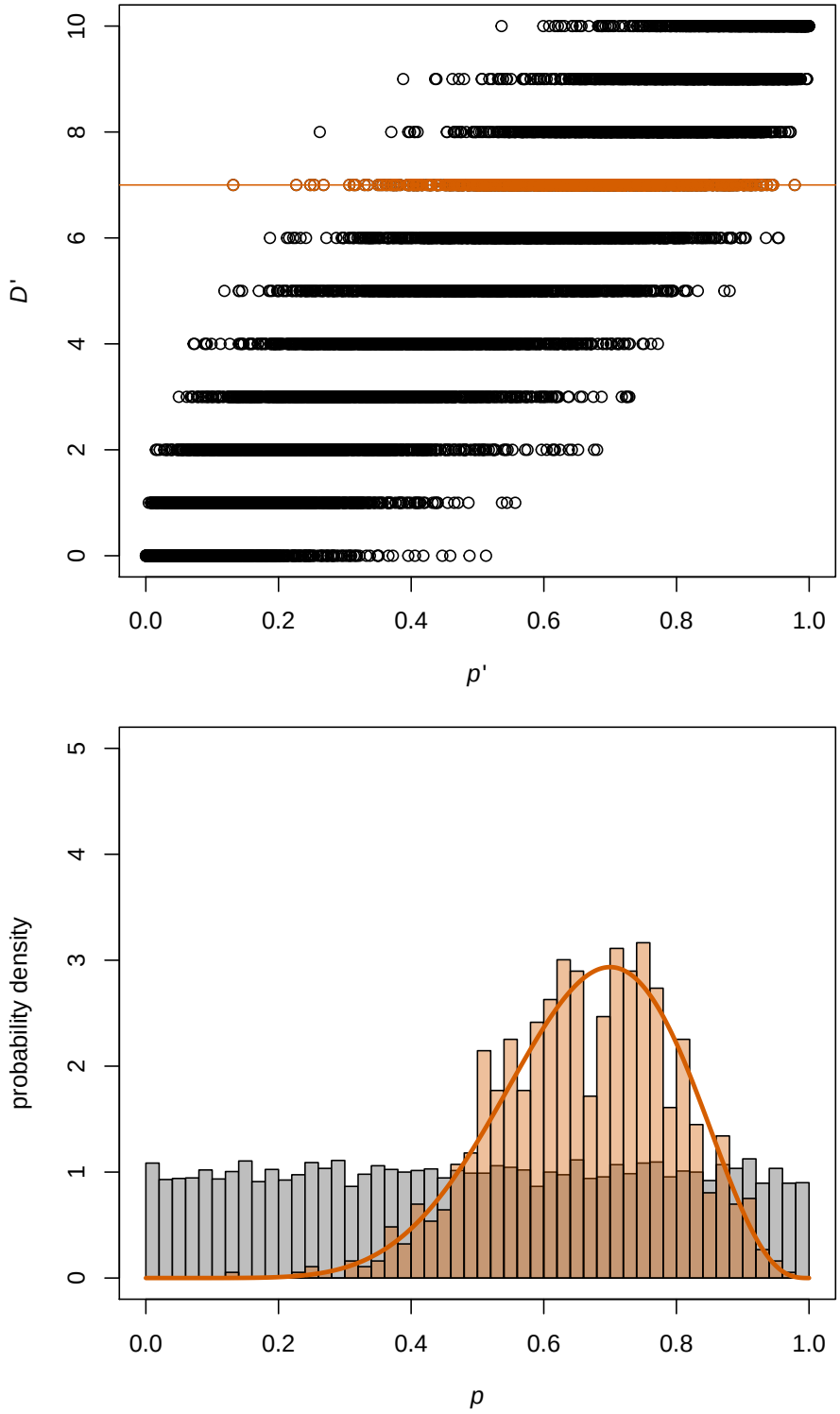


Figure 3: Simulations of coin tosses. Paramter p taken from a uniform distribution.

doi:10.5281/zenodo.1435503

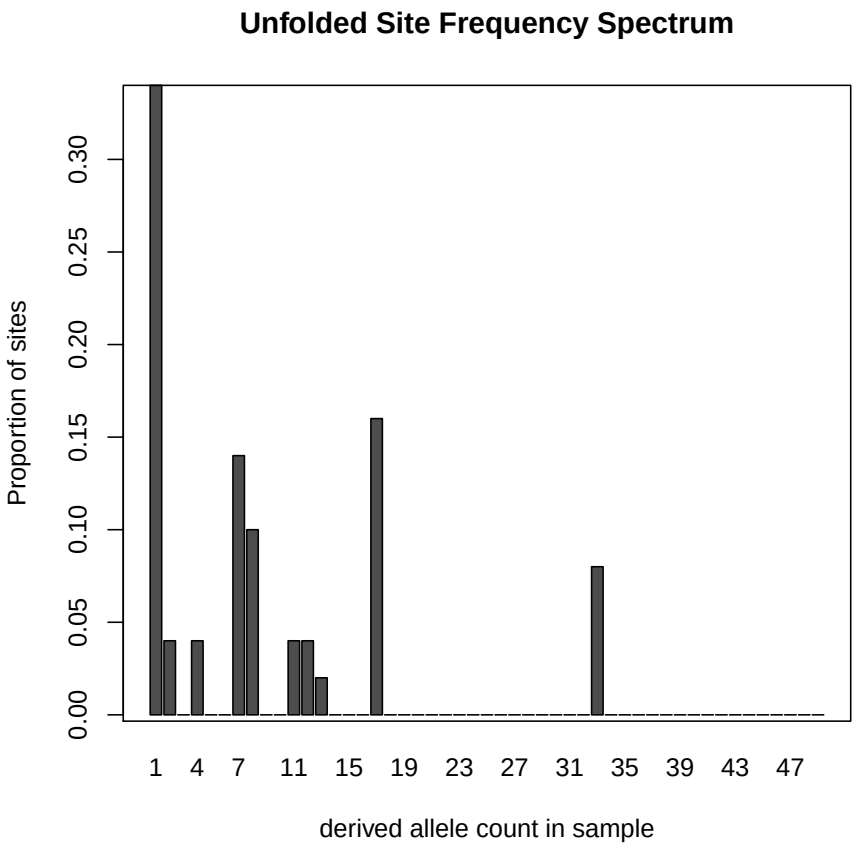


Figure 4: Site frequensy spectrum Dataset 1.

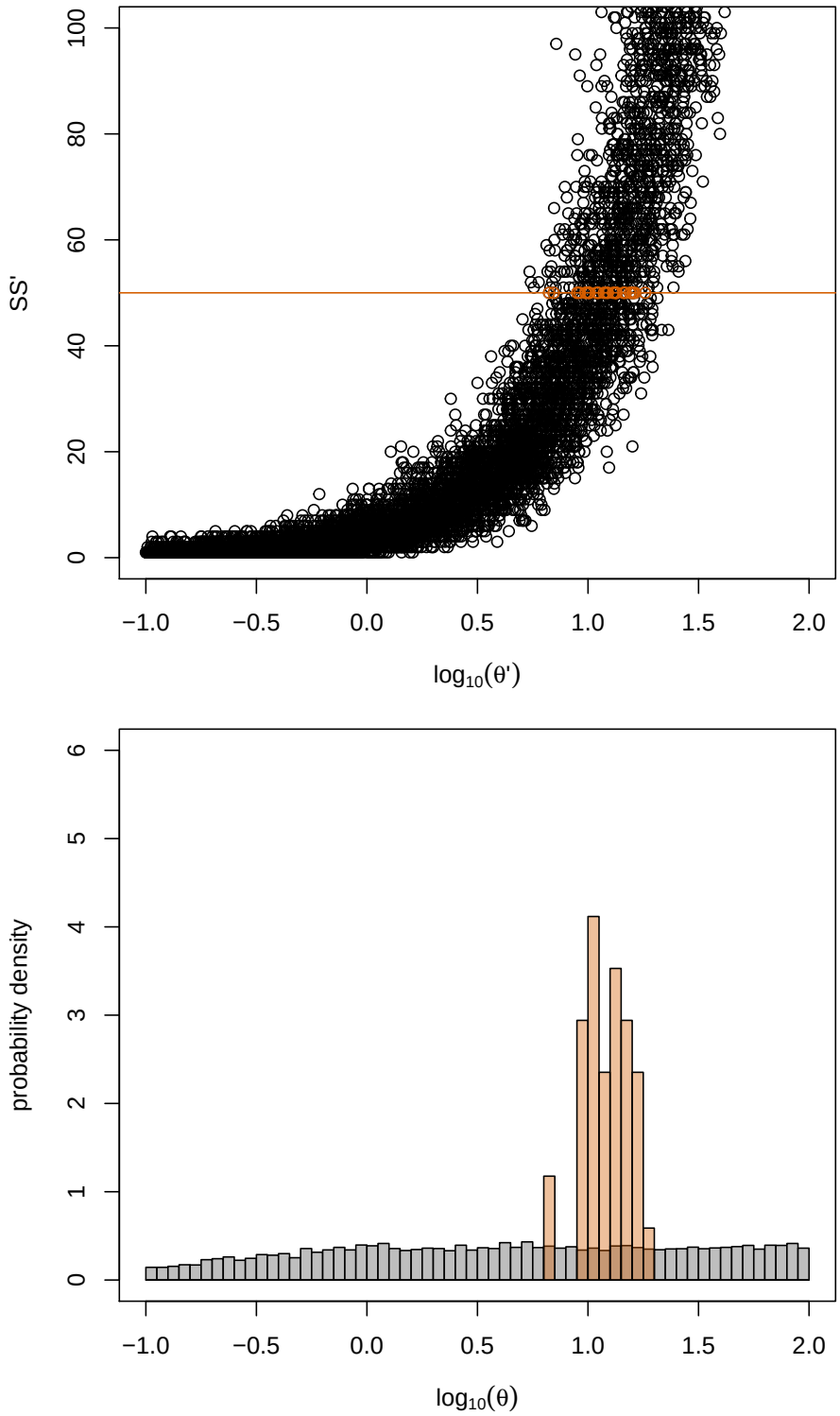


Figure 5: Coalescent simulations. Paramter θ is taken from a log-uniform distribution.

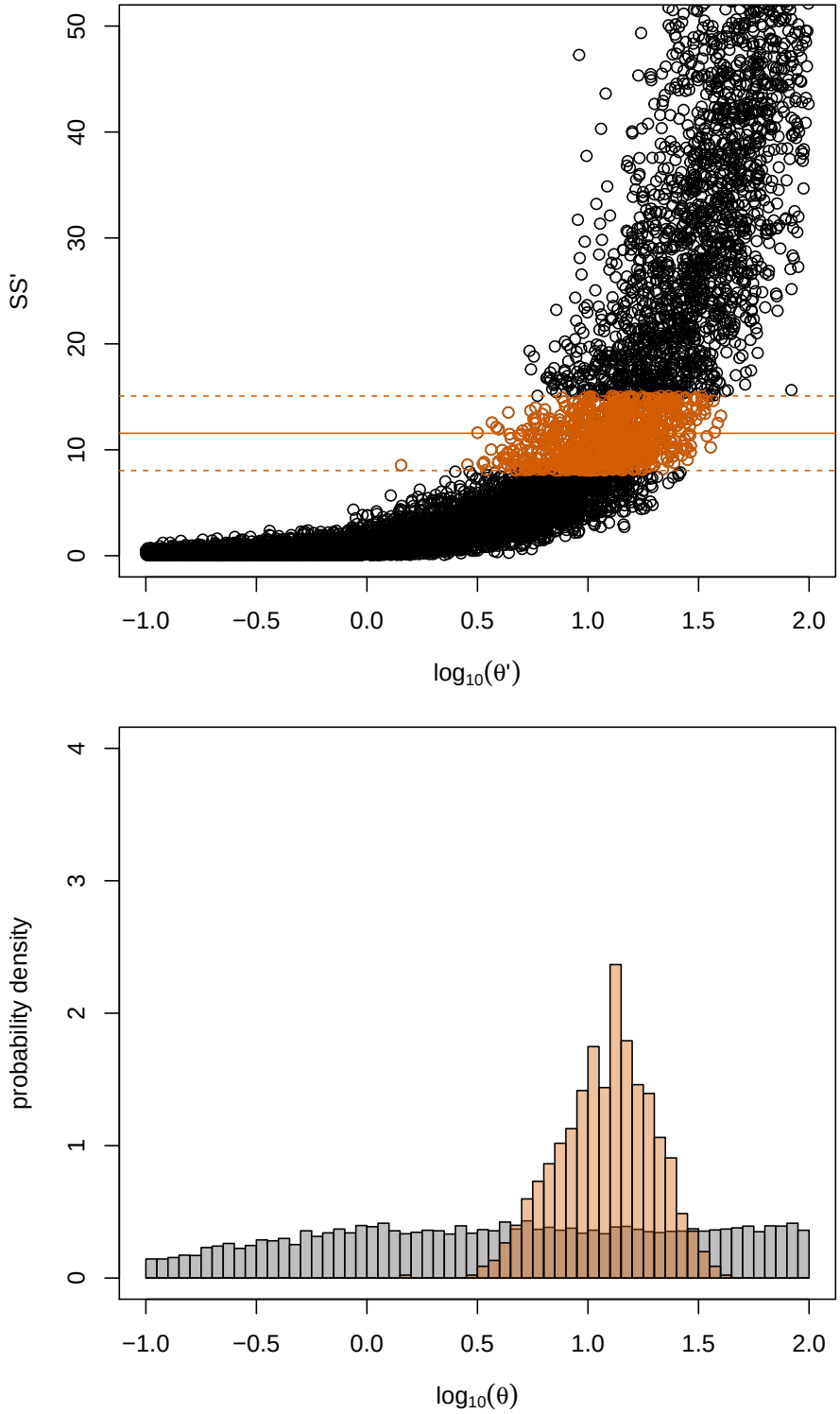


Figure 6: Coalescent simulations. Paramter θ is taken from a log-uniform distribution.

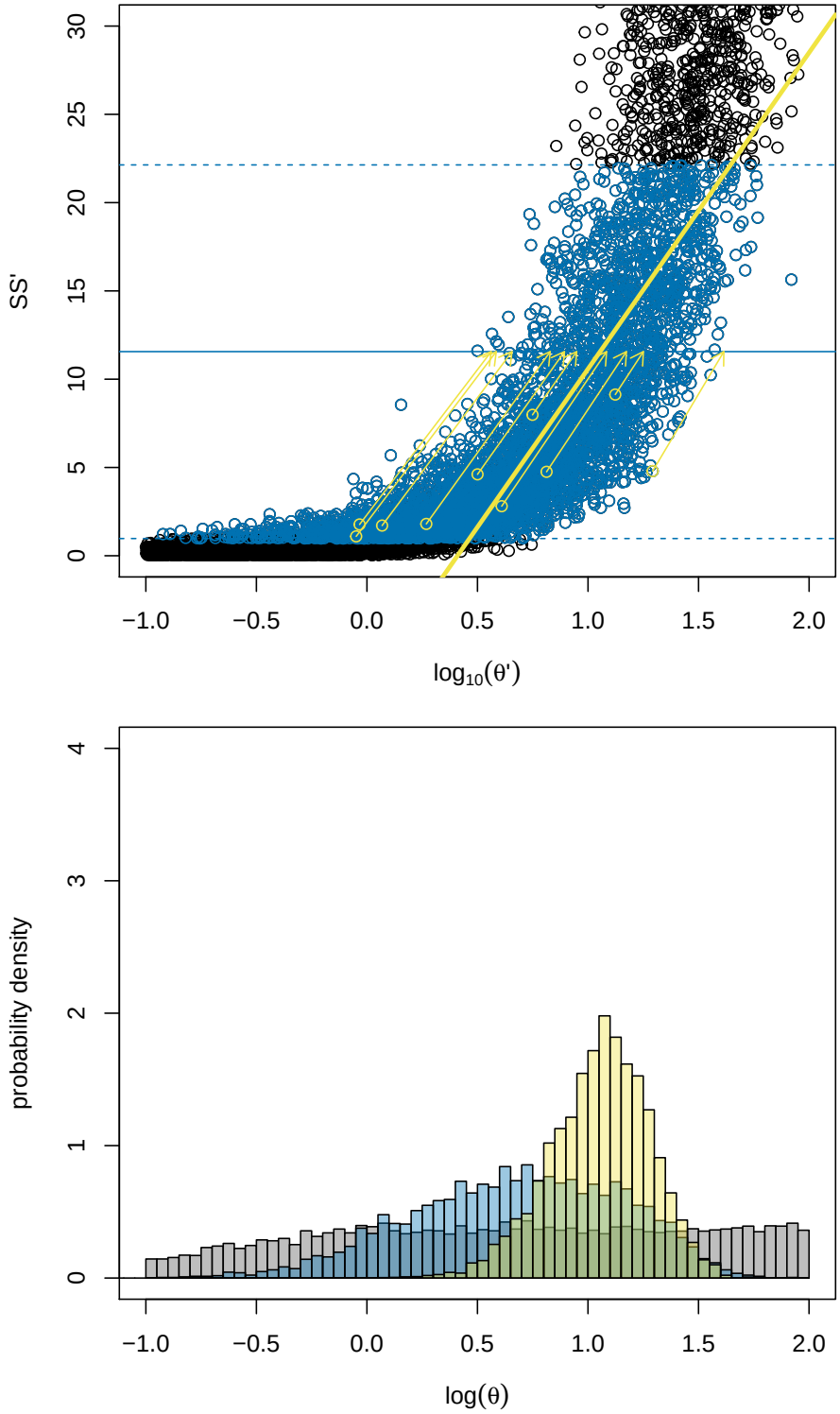


Figure 7: Coalescent simulations. Paramter θ is taken from a log-uniform distribution.

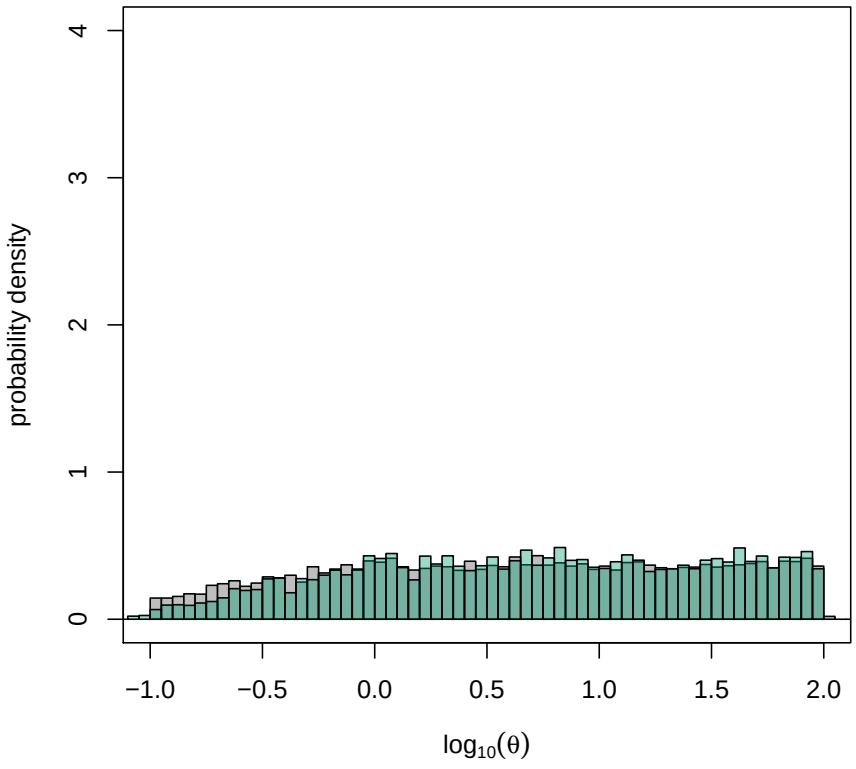


Figure 8: Probability distribution. Grey histogram: prior probability distribution.

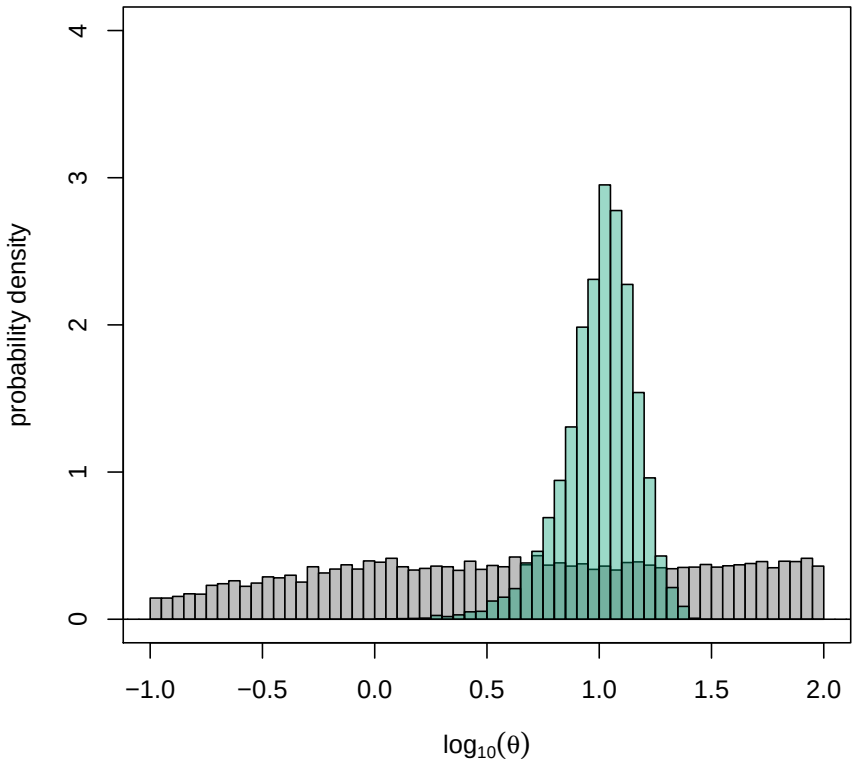


Figure 9: Probability distribution. Grey histogram: prior probability distribution.

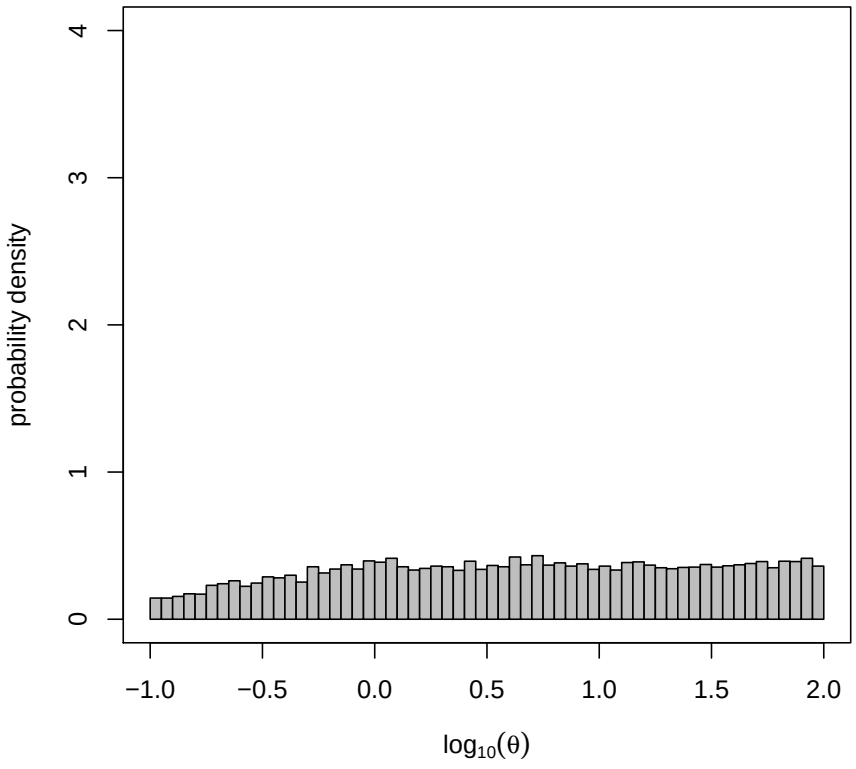


Figure 10: Probability distribution. Grey histogram: prior probability distribution.

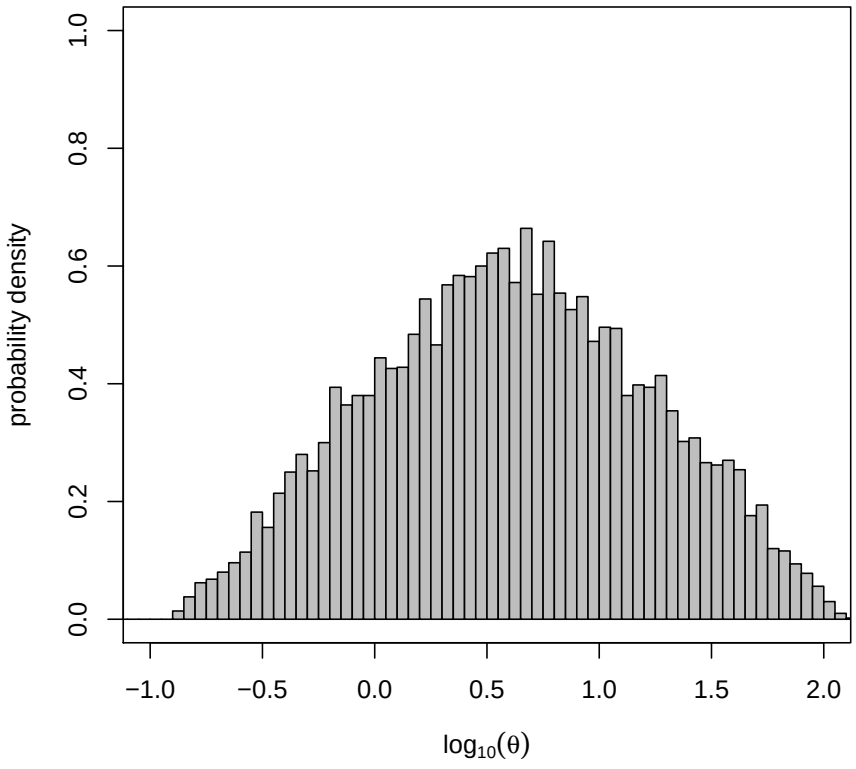


Figure 11: Probability distribution. Grey histogram: prior probability distribution.

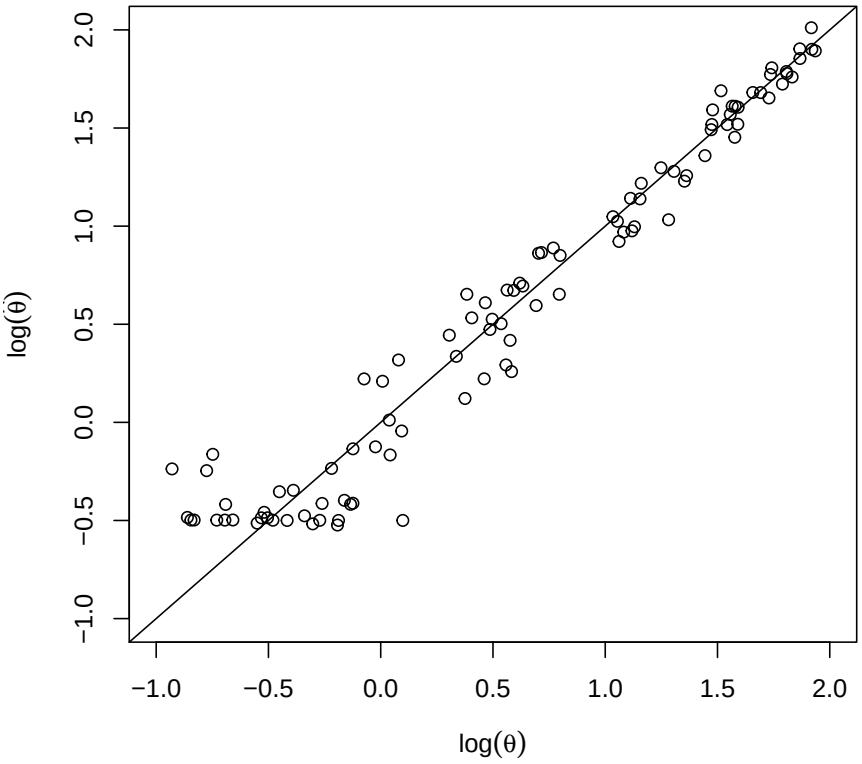


Figure 12: Cross validation.

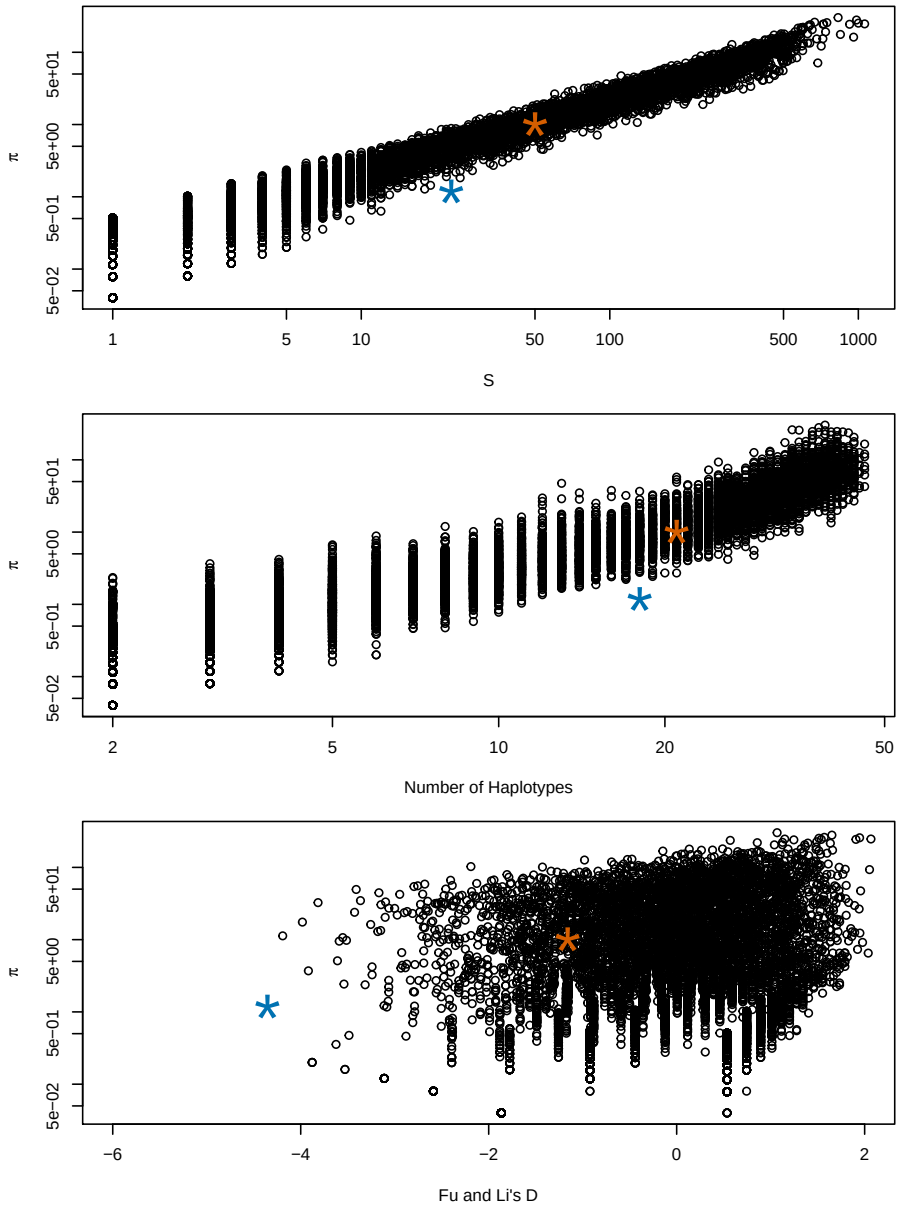
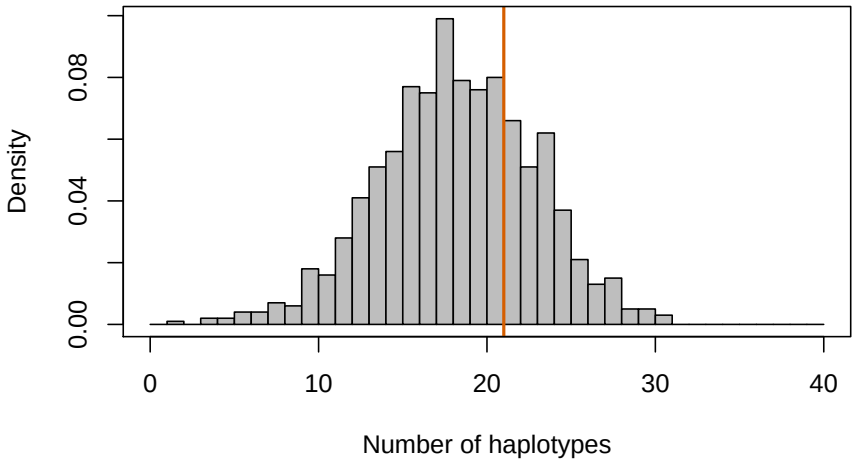


Figure 13: Goodness of fit.



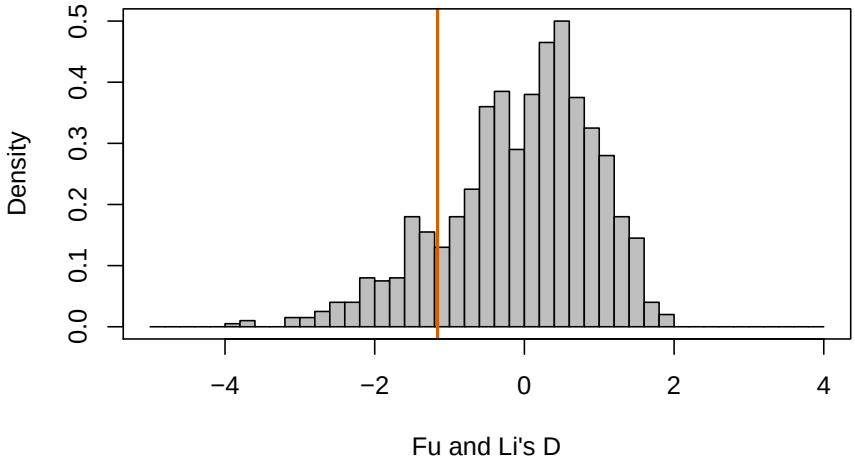


Figure 14: Goodness of fit. Dataset 1.

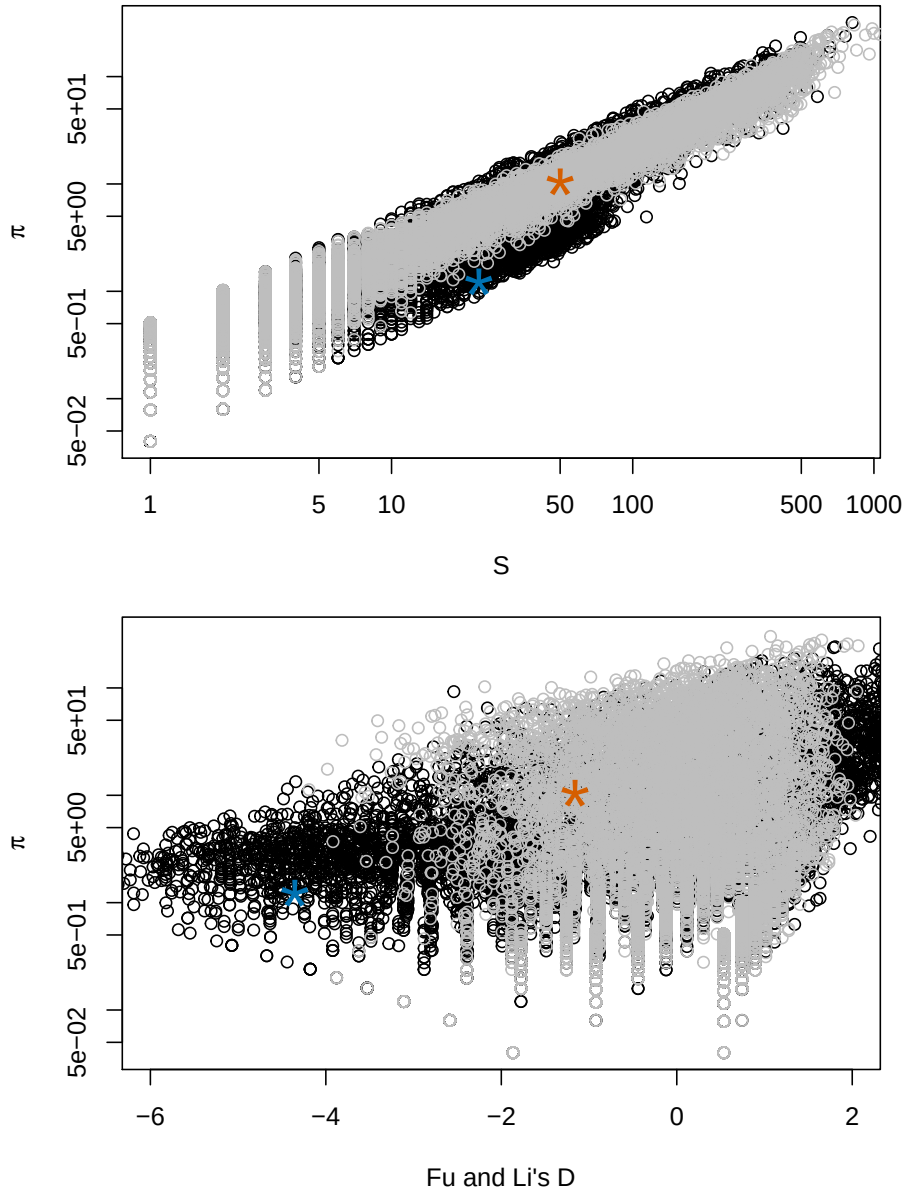


Figure 15: Goodness of fit.

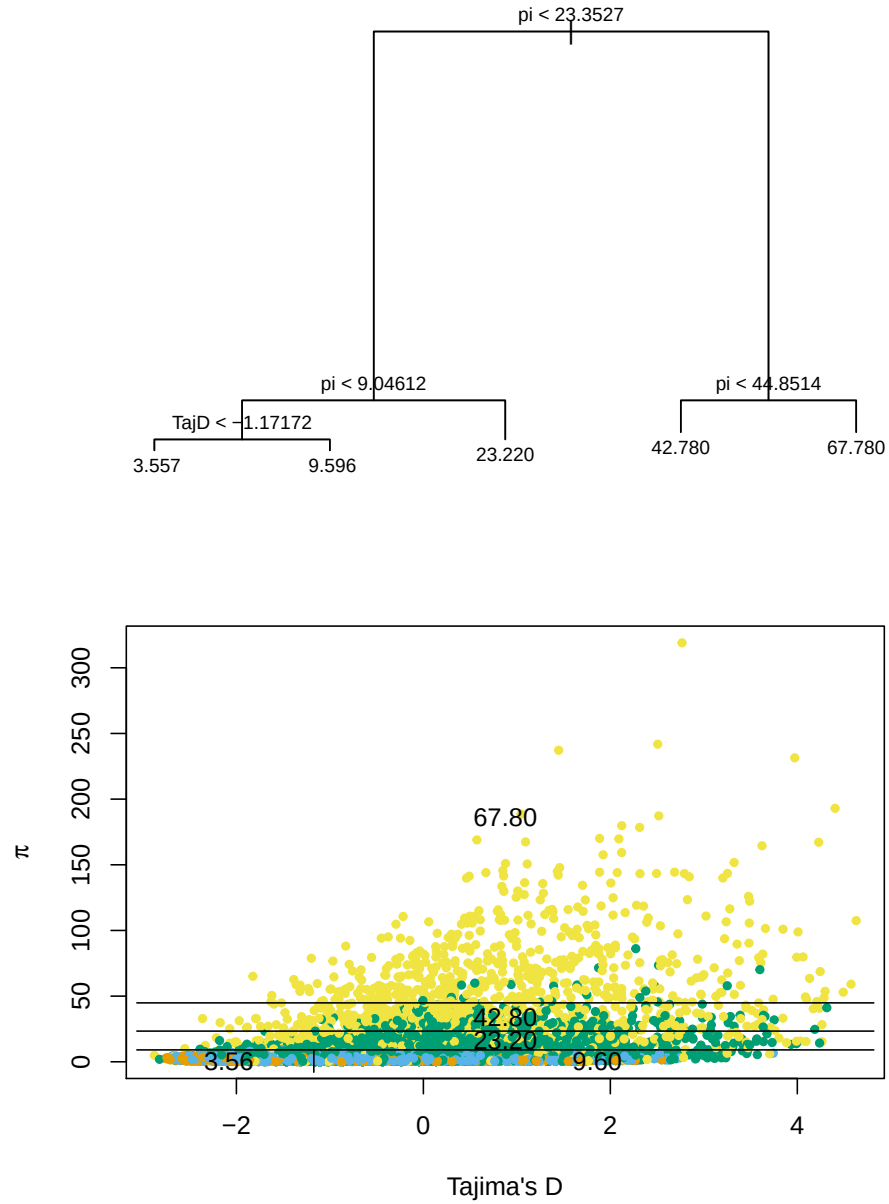


Figure 16: Regression tree.

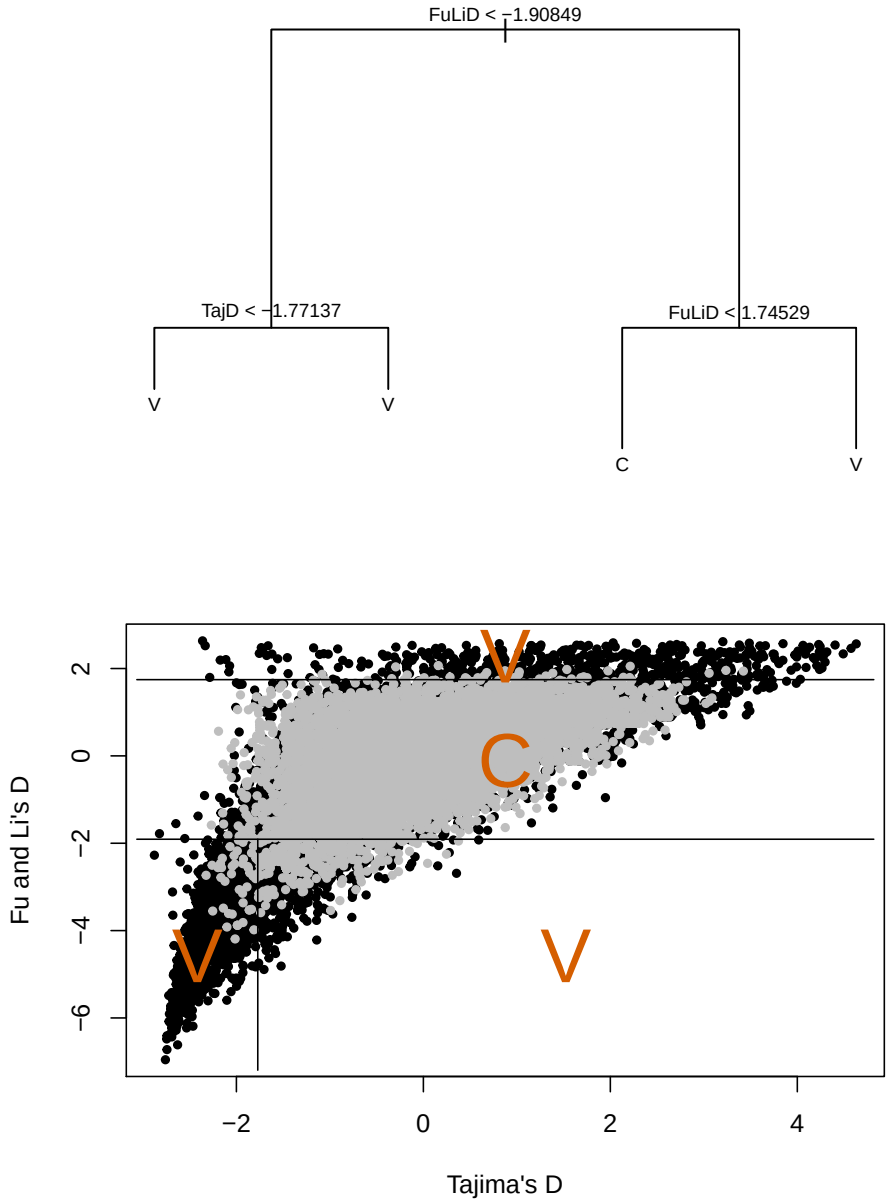


Figure 17: Classification tree.

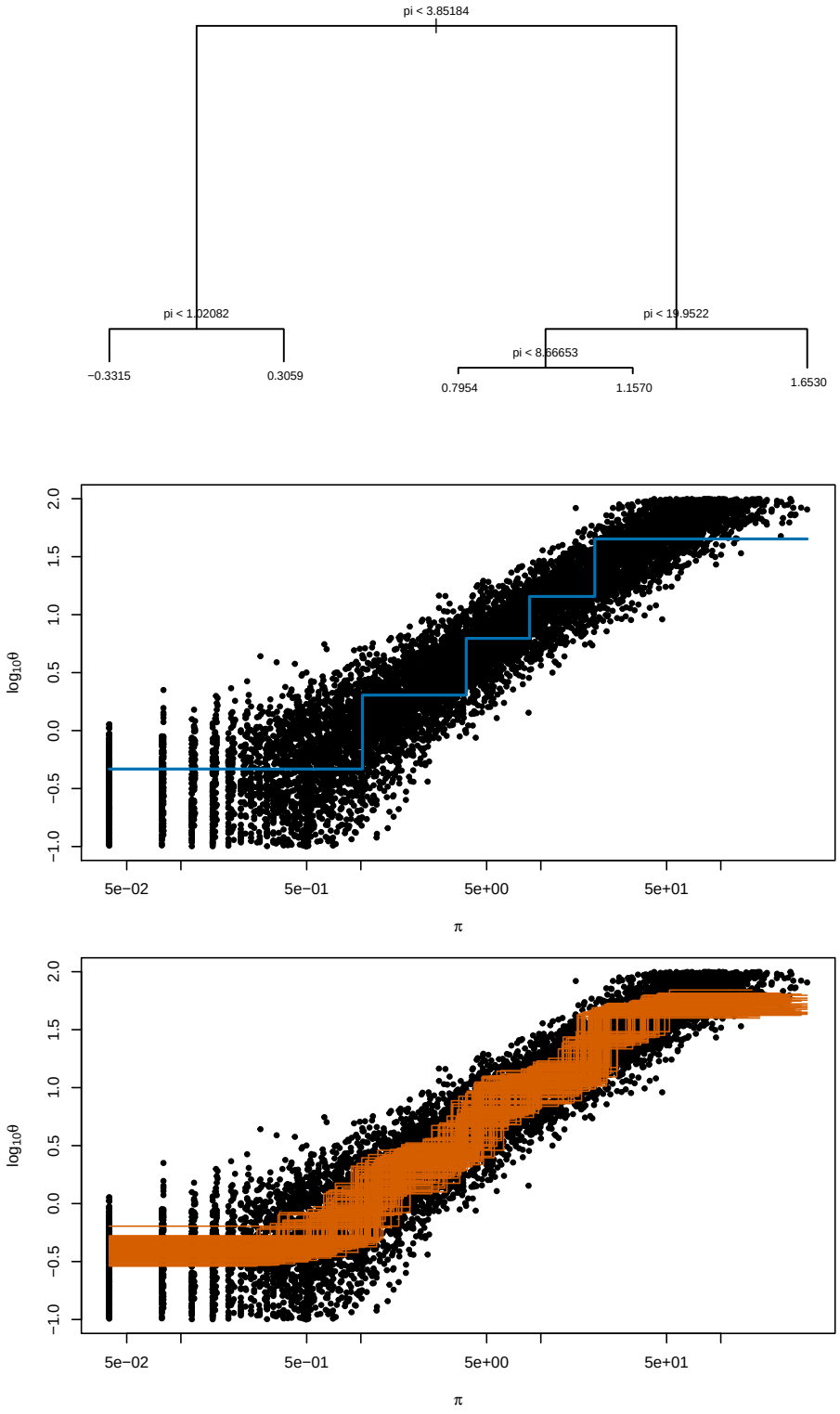


Figure 18: Classification tree.

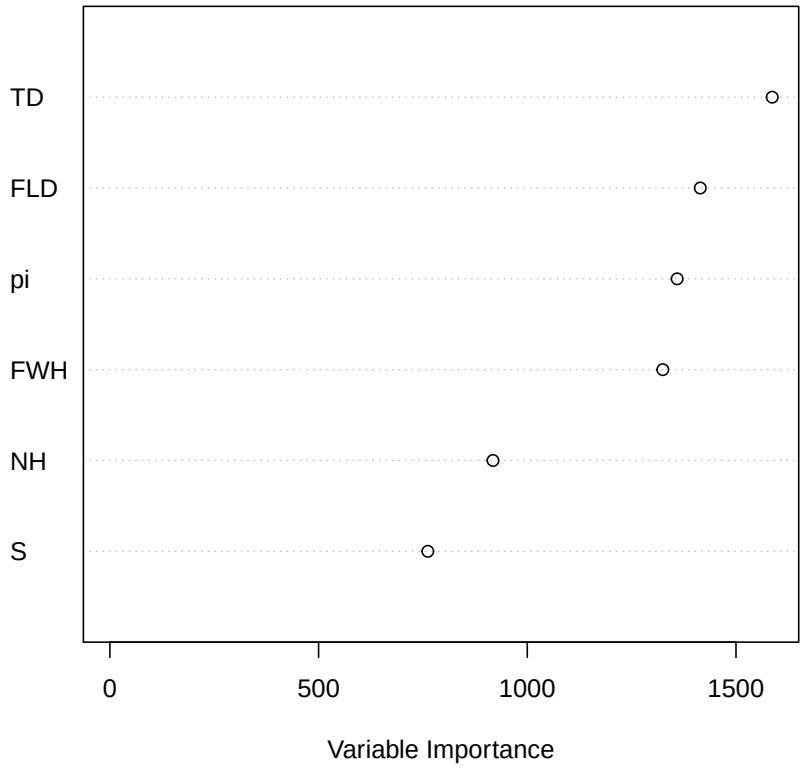


Figure 19: Variable importance plot.

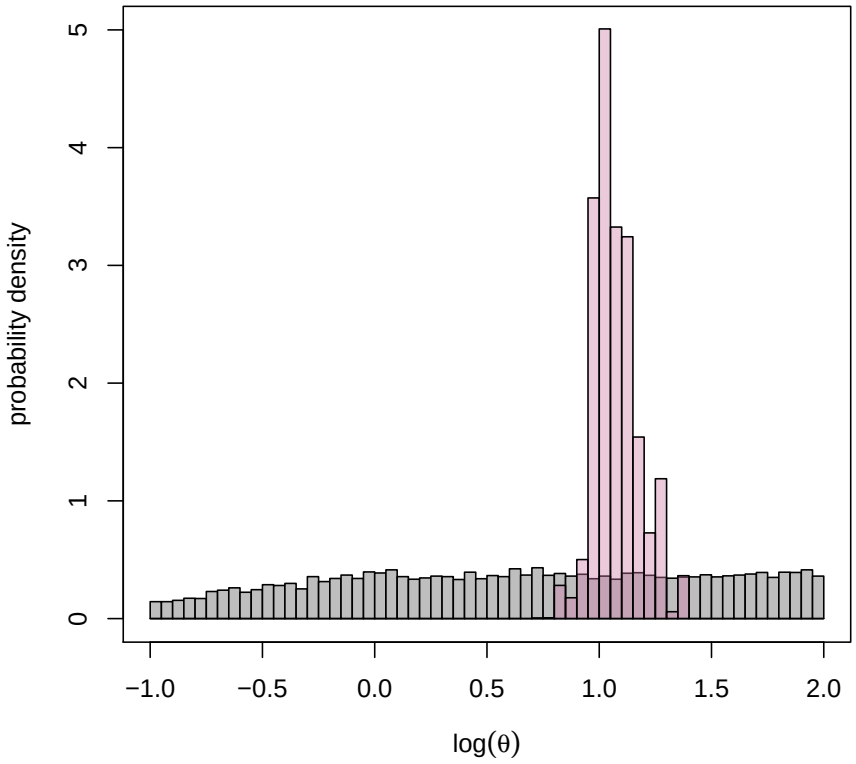


Figure 20: Probability density estimated from ABCRF.