Bayesian Practical

Pablo Lechon May 14, 2020

Reconstructing genomes from sequencing data

A Using Bayes' theorem, write the formula for the posterior probability of genotype G being AA given the sequencing data D. Write the explicit denominator assuming that your alleles are A and T and all possible genotypes are only AA, AT, TT.

$$P(AA|D) = \frac{P(AA)P(D|AA)}{P(AA)P(D|AA) + P(AT)P(D|AT) + P(TT)P(D|TT)} \tag{1}$$

B Assuming that your data is AAAT, your alleles are A and T, and the sequencing error rate is 0.01, calculate genotype posterior probability using a uniform prior

There are only three posible alleles, and the prior is a probability distribution. Therefore, the probability of each genome is

$$P(G_i) = \frac{1}{3} \tag{2}$$

where i = 1, 2, 3 and $G_1 = AA$, $G_2 = AT$, $G_3 = TT$. Under this circumstances, the posterior can be computed using equation 1, yielding the value

$$P(G_1|D) = 0.0505 \tag{3}$$

Listing 1: Code for part B

```
1
2  source("../../statistical_inference/Notebooks/Bayesian/Data/functions.R")
3
4  posterior = function(prior, likelihood_vec, genotype){
5  return(prior[genotype]*likelihood_vec[genotype]/(sum(prior*likelihood_vec)↩
       ))
6  }
7
8  #B
9  like_B = calcGenoLikes("AAAT", "A", "T", 0.01, FALSE)
10 names = names(like_B)
11 prior = rep(1/3, 3)
12 names(prior) = names
13 postB = posterior(prior, like_B, 'AA')
```

Note that the posterior probabilities of genomes G_2 and G_3 can also be computed, and their values are: $P(G_2|D) \approx 0.9495$ and $P(G_3|D) \approx 0$

C With the same assumptions as in point B, calculate genotype posterior probabilities using prior probabilities based on Hardy Weinberg Equilibrium with a frequency of T of 0.1. Do you need to calculate a new likelihood or is it the same one as in point B?

Changing the prior as done in listing 2, we obtain a poserior of

$$P(G_1|D) = 0.193 \tag{4}$$

Listing 2: Code for part C

```
#C
f_T = 0.1
f_A = 1-f_T
f_AA = f_A^2
f_AT = 2*f_A*f_T
f_TT = f_T^2
prior_C = c(f_AA, f_AT, f_TT)
names(prior_C) = names
postC = posterior(prior_C, like_B, 'AA')
```

Note that the likelihood doesn't need to bee changed, since the data is the same.x

D With the same assumptions as in point C, calculate genotype posterior probabilities using a prior based on Hardy Weinberg Equilibrium with a frequency of T of 0.1 and an inbreeding coefficient of 0.2. In this case, we need to modify our previous priors.

In this case the posterior can be calculated as in listing 3, and we obtain a posterior probability of

$$P(G_1|D) = 0.234 \tag{5}$$

Listing 3: Code for part D

```
#D
f_T = 0.1
I = 0.2
f_TT = f_T^2 + I*f_T*(1-f_T)
f_AT = 2*f_T*(1-f_T)*(1-I)
f_AA = (1-f_T)^2 + I*f_T*(1-f_T)
prior_D = c(f_AA, f_AT, f_TT)
names(prior_D) = names
postD = posterior(prior_D, like_B, 'AA')
```

Note that the likelihood is the same, because the data has not changed.

E With the same priors used in point D but with a sequencing error rate of 0.05, calculate the genotype posterior probabilities. Do you need to calculate a new likelihood or is it the same one as in point D?

Changing the sequencing error rate will change how we take the data, and therefore the likelihood will be affected. Proceeding similarly, as seen in listing 4

Listing 4: Part C

```
#E
like_D = calcGenoLikes("AAAT", "A", "T", 0.05, FALSE)
postD = posterior(prior_D, like_D, 'AA')
```

This yields a posterior probability of

$$P(G_1|D) = 0.601 \tag{6}$$

F Plot all previous results (e.g. use a barplot with the 4 posterior probabilities for each scenario B-E).

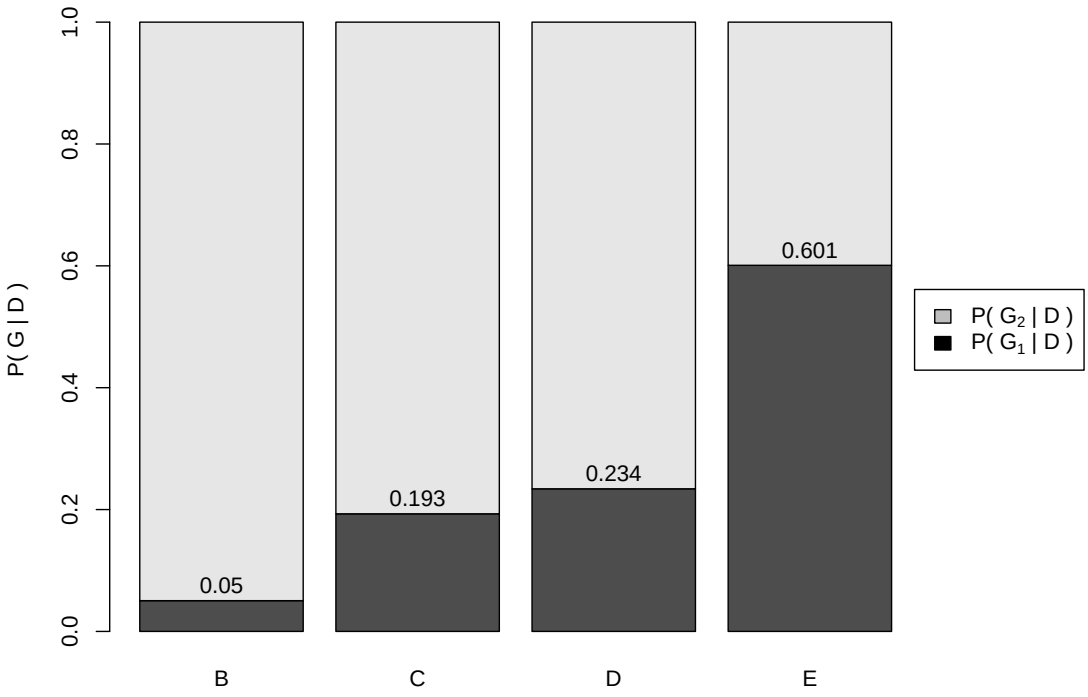

Figure 1: Posterior probabilities of each genome for questions B, C, D and E. Note that $P(G_3|D) \approx 0$ in all cases.

This graph has been produced with the code

```
    #F
    counts = c(postB, postC, postD, postE)
    names(counts) = c('B', 'C', 'D', 'E')
    counts_plot = as.matrix(rbind(counts, 1-counts))
    #rownames(counts_plot) = c(expression(paste('P( ','G'[i],' | D )')))
    par(mar=c(5.1, 4.1, 4.1, 8.1), xpd=TRUE)
    p = p = barplot(counts_plot, ylab = 'P( G | D )')
    text(p, counts+0.025, labels=round(counts, digits = 3), xpd=TRUE)
    legend(x = "right",
           legend = c(expression(paste('P( ','G'[2],' | D )')),
                      expression(paste('P( ','G'[1],' | D )'))),
           fill = c('gray', 'black'),
           inset=c(-0.2,0))
```

G Assuming that our collection of sequenced bases is AAATATAAAAAAATTTTAAATTA, calculate the genotype posterior probabilities using the same priors as in point C and a sequencing error rate of 0.05. What happens if we have more data? What is the confidence in our genotype inference?

Implementing the following code in R

```
#G
like_G = calcGenoLikes("AAATATAAAAAAATTTTAAATTA", "A", "T", 0.05, FALSE)
prior_G = prior_C
postG = posterior(prior_G, like_G, 'AA')
like_G_sorted = sort(like_G, decreasing = T)
confidence = log(like_G_sorted[1]) / log(like_G_sorted[2])
```

If we take the log and then the ratio between the most likely and the second most likely likelihoods, that is the standard likelihood ratio often used to express the support for one parameter value. Doing this yields

$$C = 0.499 \tag{7}$$

We can see that with more data, our confidence in the genotype inference increases substantially because, in point C, our inference confidence was $C = 0.488$

H What happens if we have a lot of data? Assume that your sequenced bases are `bases <- paste(c(rep("A",1e3),rep("T",1e3)), sep="", collapse="")`. Calculate the genotype likelihoods for this data. What is happening here? It is convenient to use numbers in log-scale and you can do that by selecting TRUE as the last parameter in the calcGenoLikes. Remember that if you want to calculate proper probabilities (in log) you have to approximate the sum of logs. Without calculating posterior probabilities, what is the effect of the prior here in

your opinion?

When there is that amount of data available, the precision of the machine is reached. To avoid this problem, we have to calculate log-likelihoods instead of likelihoods, as in the code below

```
#H
bases <- paste(c(rep("A",1e3),rep("T",1e3)), sep="", collapse="")
like_H = calcGenoLikes(bases, "A", "T", 0.05, TRUE)
```

Obtaining

$$P(D|AA) = -4145 \quad P(D|AT) = -1454 \quad P(D|TT) = -4145 \tag{8}$$

With a lot of data you are confident that the genotype is AT here and the prior won't change the posterior much. In fact, we can check this expected behaviour by calculating posterior probabilities. Since we have calculated the log-likelihoods, taking logarithms to both sides of equation 1 is pertinent

$$\log\left(P(G_i|D)\right) = \log\left(P(G_i)\right) + \log\left(P(D|G_i)\right) - \underbrace{\log\left(\sum_{k=1}^{3} P(G_i)P(D|G_i)\right)}_{L} \tag{9}$$

If we are able to compute the quantity in the R.H.S. of equation 9, simply exponentiating the result would yield the posterior. To calculate such quantitie, we need to approximate the log of sums using the expression

$$\log\sum_{k=1}^{3} a_k = \log a_1 + \log\left(1 + \sum_{k=2}^{3} e^{\log a_k - \log a_1}\right) \tag{10}$$

where $a_k = P(G_i)P(D|G_i)$. Thus,

$$L = \log\left(P(G_1)\right) + \log\left(P(D|G_1)\right) + \log\left(1 + \sum_{k=2}^{3} e^{\log\left(\frac{P(G_k)}{P(G_1)}\right) + \log(P(D|G_k)) - \log(P(D|G_1))}\right) \tag{11}$$

Note that L is now expressed in terms that we know: log of prior and log-likelighoods. Calculating probabilities, we obtain the posteriors

$$P(AA|D) = 0 \quad P(AT|D) = 1 \quad P(TT|D) = 0 \tag{12}$$

As we expected

Population inferences from finite samples

A Plot the posterior probability. Then calculate the maximum a posteriori value, 95% credible intervals, and notable quantiles. What happens to the distribution if we have only 10 samples (with the sample allele frequency of 0.20)?

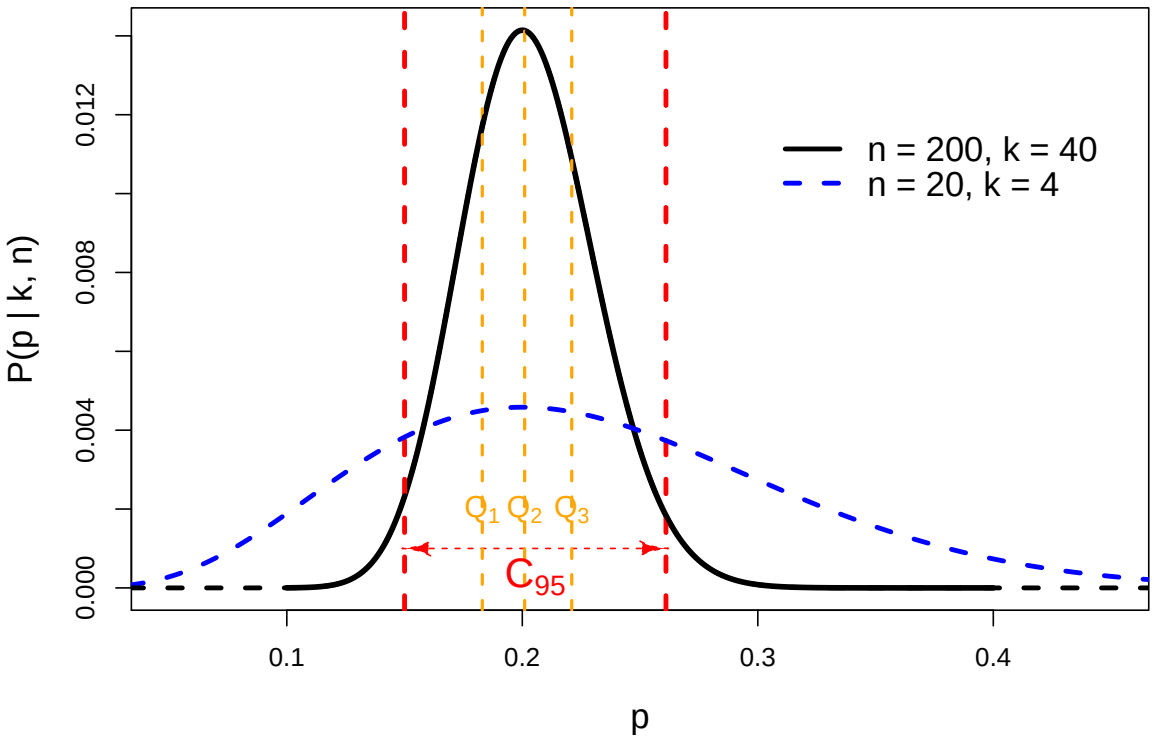

Figure 2: Posterior probability for two sets of parameters n, k. $C_{95\%}$ has been plotted in red and quantiles Q_{25}, Q_{50} and Q_{75} have been plotted in orange for the larger sample only.

For p = 0.2, the maximum value of the posterior is reached: $P(p|k,n) = 0.014$.
The 95 % credible interval is

$$C_{95} = [0.150, 0.261]$$

The three calculated quantiles are

$$Q_{25} = 0.183\ ,\ Q_{50} = 0.201,\ Q_{75} = 0.221$$

Both C_{95} and $Q_{1,2,3}$ have been calculated using numerical integration (see function `credible_interval` in listing 5). Therefore, they are not exact, and the approximation will be better the more points we use to evaluate the posterior.
In the case of having less data, our inference is worse, as it can be seen by noting that the blue curve of figure is broader than the black curve. Another indicator of a worse inference can be found in the broader 95% credible interval

$$C_{95} = [0.082, 0.420]$$

The functions used throughout this practical are shown below

Listing 5: Code for part B

```
library(shape)


beta_func = function(p, alpha, beta){
return(p^(alpha - 1) * (1 - p)^(beta - 1))
}

posterior = function(p, k, n, alpha, beta){
return(p^(k + alpha - 1)*(1-p)^(n-k+beta-1))
}

credible_interval = function(probabilities, alpha){
#Locate the center/maximum of distribution
center_ind = which.max(probabilities)
#Divide by max, right and left, and get cumsum of each
cumulative_right = cumsum(rev(probabilities[center_ind:length(probabilities)]))
cumulative_left = cumsum(probabilities[1:center_ind])
#Get cosest element to (1-alpha)/2
right_lim = length(cumulative_left) +
length(cumulative_right) -
which.min(abs(cumulative_right-alpha/2))
left_lim = which.min(abs(cumulative_left-alpha/2))
#Find that element in the whole vector

interval = c(left_lim, right_lim)
return(interval)
}

quantile = function(probabilities, q){
cumulative = cumsum(probabilities)
quantiles = rep(NA, length(q))
j = 1
for (i in q){
quantiles[j] = which.min(abs(cumulative - i))
j = j + 1
}
return(quantiles)
}
```

Also, the code used to plot figure 1 is shown below

```r
#A
#Plot posterior
alpha = 1
beta = 1
k = 40
n = 200
p = seq(0, 1, 1e-3)
post = posterior(p, k, n, alpha, beta)
norm = sum(post)
post_prob = post/norm
plot(p[which(p == 0.1):which(p == 0.4)],
post_prob[which(p == 0.1):which(p == 0.4)],
type = 'l', xlab = 'p', ylab = 'P(p | k, n)', xlim = c(0.05, 0.45),
cex.lab = 1.3, lwd = 3.5)
#Credible sets
#95 %
int_95 = p[credible_interval(post_prob, 0.05)]
abline(v = int_95, col = 'red', lty = 'dashed', lwd = 3)
Arrows(int_95[1]+0.005, 0.001, int_95[2]-0.005, 0.001,
col = 'red', code = 3, lty = 'dashed')

#Quantiles Q1, Q2, Q3
quantiles = quantile(post_prob, seq(0, 1, 0.25))
abline(v = p[quantiles], col = 'orange', lty = 'dashed', lwd =  2)
text(x = (int_95[2]-int_95[1])/2+int_95[1], y = 0.0003,
label = expression('C'[95]), col = 'red',
cex = 1.6)
text(x = p[quantiles], y = rep(0.002, length(quantiles)),
labels = c('Q1', expression('Q'[1]),
expression('Q'[2]),
expression('Q'[3]), 'Q3'),
col = 'orange', cex = 1.2)

#If we have only 10 samples with the same allele frequency
n = 20
k = 4
post = posterior(p, k, n, alpha, beta)
norm = sum(post)
post_less_norm = post/norm
lines(p, post_less_norm, lty = 'dashed', col = 'blue', lwd = 3)
lines(p, post_prob, lty = 'dashed', lwd = 3)
legend(0.3, 0.012, legend = c('n = 200, k = 40','n = 20, k = 4'),
col = c('black', 'blue'), lty = c('solid','dashed'),
bty = 'n', cex = 1.3, lwd = 3)

#C95 less data
```

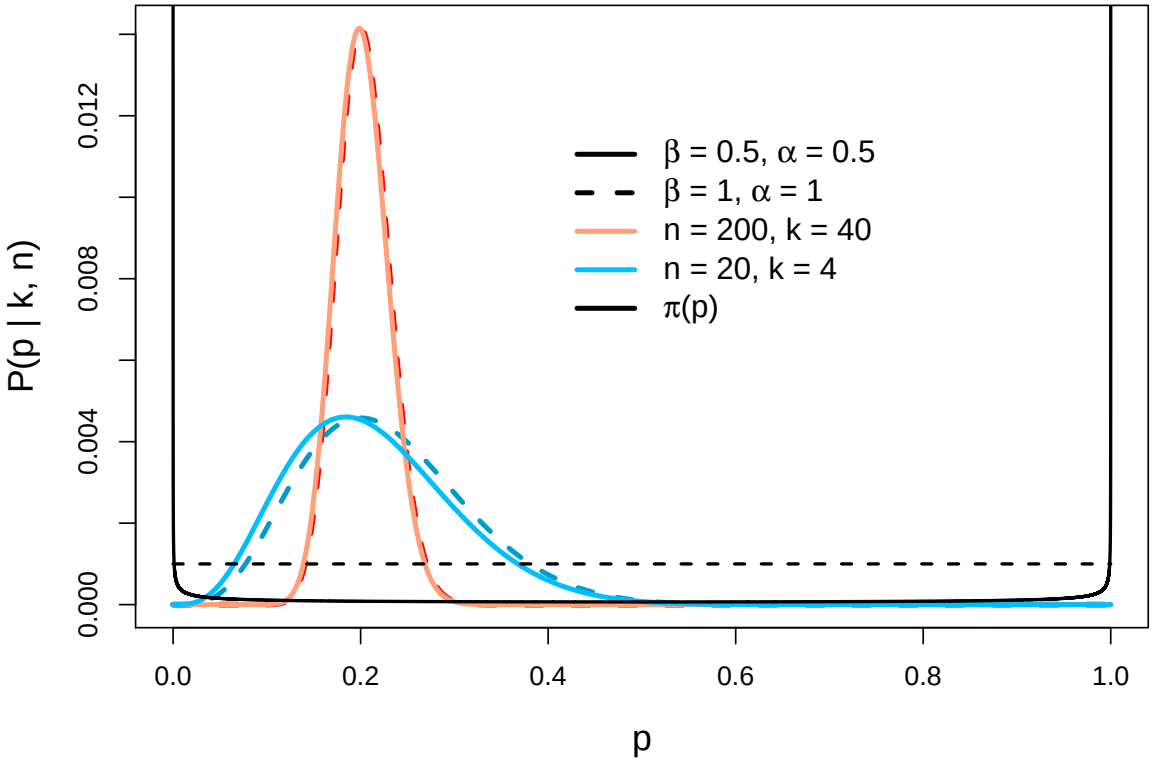

Figure 3: Posterior distributions for a varying amount of data and prior

```
47      int_95_less = p[credible_interval(post_less_norm, 0.05)]
48
49      #Quantiles less data
50      quantiles_less = p[quantile(post_less_norm, seq(0, 1, 0.25))]
```

B Recalculate the posterior distribution of p using an informative prior (make your own choices regarding the parameter for the Beta distribution) both in the case of 100 and 10 samples. Discuss how these results compare to the previous ones obtained in point A.

If we look at the genome-wide distribution of allele frequencies for human populations, we see that it has a particular shape. This suggests that we can use a more informative prior than a uniform one. In allele distribution, we can see that is likely that the minor allele will have a low frequency. However, we don't know what the minor allele is. Thus, we can implement a prior where both a low frequency of G (lower end of p) and a low frequency of A (higher end of p) have high probabilities. This can be achieved with a beta function that has $\beta = 0.5$ and $\alpha = 0.5$, as seen in figure 3. The effect of a more informative prior on the posterior is higher the less data we have. However, it displaces the posterior to the left in both cases. This means that implementing a prior that favors low frequencies will shift the posterior distribution towards lower

ones. In this case, the posterior gets displaced towards lower frequencies of allele G, because the data supports that G is the minor allele.

The code used to carry out this part of the practical is shown below

```
#B
plot(p, post_prob,
type = 'l', xlab = 'p', ylab = 'P(p | k, n)',
cex.lab = 1.3, lwd = 3.5, lty = 'dashed', col = 'red')

lines(p, post_less_norm, lty = 'dashed', col = 'deepskyblue3',
lwd = 3)

post = posterior(p, 40, 200, 0.5, 0.5)
norm = sum(post)
post_prob = post/norm
lines(p, post_prob, lwd = 3, col = 'lightsalmon')

post = posterior(p, 4, 20, 0.5, 0.5)
norm = sum(post)
post_prob = post/norm
lines(p, post_prob, lwd = 3, col = 'deepskyblue1')

prior_unif = beta_func(p, 1, 1)
p_inf = seq(0,1,1e-6)
prior_inf = beta_func(p_inf, 0.5, 0.5)
lines(p, prior_unif/(sum(prior_unif)), lwd = 2, lty = 'dashed')
lines(p_inf, prior_inf/30000, lwd = 2)

legend(0.4, 0.012,
legend = c(expression(paste(beta, ' = 0.5', ', ', alpha, ' = 0.5')),
expression(paste(beta, ' = 1', ', ', alpha, ' = 1')),
'n = 200, k = 40',
'n = 20, k = 4',
expression(paste(pi, '(p)'))),
col = c('black', 'black', 'lightsalmon', 'deepskyblue','black'),
lty = c('solid', 'dashed', 'solid', 'solid','solid'),
lwd = c(3,3,3,3,3),
cex = 1.2,
bty = 'n'
)
```

C Calculate the Bayes factor for a model with p<=0.5 vs a model with p>0.5. Note that these models are equally probable a priori.

Throughout this part, a uniform prior, will be used. The Bayes factor is given, in the case that both models are equally likely a priori, by

$$BF = \frac{P(p \leq 0.5|y)}{p(p \geq 0.5|y)} \tag{13}$$

To get this value, we have to integrate until 0.5 from the left and from the right, respectively, and then take the ratio of those quantities. In the case where we have a lot of data ($k = 40$, $n = 200$) $BF = \infty$. In the case of less data ($k = 4$, $n = 20$), $BF \approx 281$. In the first case, we are certain that M_1 wher e $p \leq 0.5$ is the more valid. However, in the second case, the evidence for model M_1 being correct is only *strong*.

Below it is included the code used in part **C**

```
#C
#For the case of a lot of data the posterior is

alpha = 1
beta = 1
k = 40
n = 200
p = seq(0, 1, 1e-3)
post = posterior(p, k, n, alpha, beta)
post_norm = post/sum(post)
lim = which(p ==0.5)
p_05 = sum(post_norm[1:lim])
BF1 = p_05/(1-p_05)

alpha = 1
beta = 1
k = 4
n = 20
p = seq(0, 1, 1e-3)
post = posterior(p, k, n, alpha, beta)
post_norm = post/sum(post)
lim = which(p ==0.5)
p_05 = sum(post_norm[1:lim])
BF2 = p_05/(1-p_05)
```

Bayesian estimation of speciation times

In this exercise our data is high dimensional because we have multivarirate measurements. Thus, a dimensionality reduction through summary statistics is suitable. To calculate a posterior distribution of the speciation time of the polar bear, we will be using ABC. In particular, we will apply the rejection algorithm to a set of choseen summary statistics. First, we simulate lots of data using the `simulate` provided function. Particularly 10^4 are performed. The prior distribution for the speciation time is chosen to be T$\sim$Uniform ($2{\cdot}10^5$, $7 \cdot 10^5$). The lower limit of this distribution can be roughly justified. If we simulate with T = $2{\cdot}10^5$, and we calculate the summary statistic fst (which measures how much species are genetically different, going from 0 to 1, being 0 the same species, and 1 completely diferent), we obtain fst = 0.43. This suggests that the speciation time (the latest time when the species were equal) is greater than $2{\cdot}10^5$

The code that generates simulated data is included below

```
#Generate a prior of reasonable values for T (uniform
#distribution between 200k and 700k years)
Tsim = runif(nrSimul, 200e3, 700e3)
#Prealocate quantities
cols = length(obsSummaryStats)
simSummaryStats = matrix(NA,
nrow = nrSimul,
ncol = cols)
nrSimul <- 1e5
n = 1
while(n <= nrSimul){
#Draw a value of T from the prior
simulate(Tsim[n], M = 0, nrSites, msDir, fout)
simulatedSFS <- fromMStoSFS(fout, nrSites,
nChroms.polar, nChroms.brown)
#Calculate summary statistics for current simulation
simSummaryStats[n,1:cols] = calcSummaryStats(simulatedSFS)
n = n + 1
}
```

Once we have simulated data, we have to choose a the smallest set of summary statistics that describe the features of our data as best as possible. To do this, we check for correlations between the speciation time parameter simulations (T_{sim}) and each simulated summary statistic. Plotting each of them against the simulated parameters yields this plot

From fitgure we suspect that the statistic fst is the one that correlates best with T_{sim}. That is why it is the one we will use in our abc analysis. We decide not to include more because both the meaning and the plot in figure are very correlated with T_{sim}, and because the accuracy and stability of ABC decreases rapidly with increasing numbers of summary statistics. Additionally,

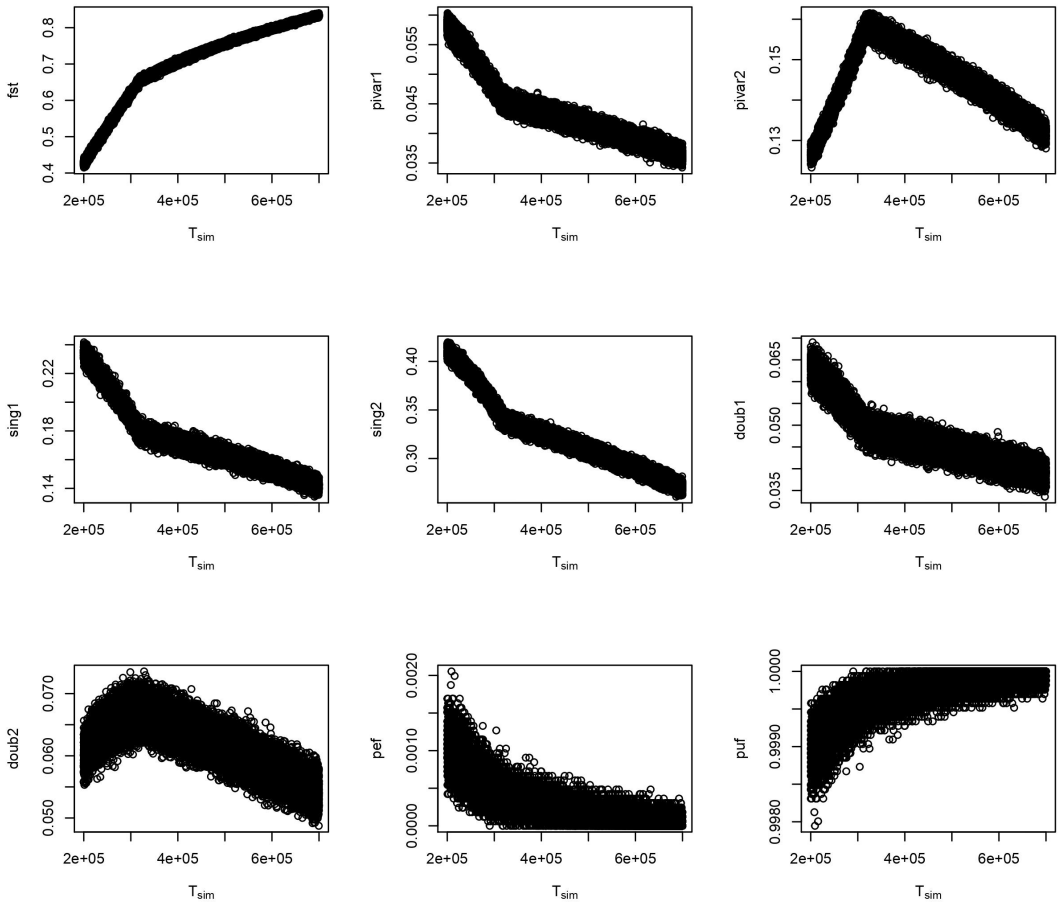

Figure 4: Correlation plots of T_{sim} with the simulated summary statistics

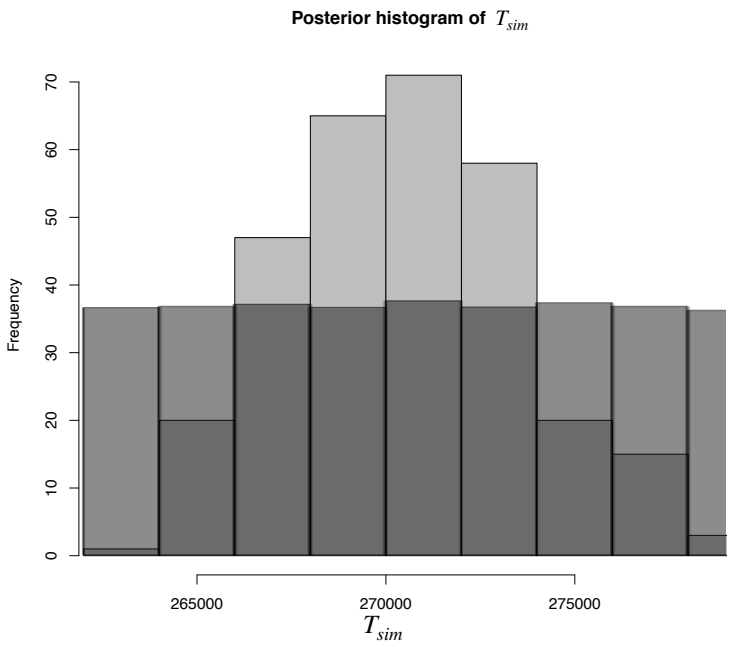

Figure 5: Posterior distribution of T_{sim} and its prior normalized to the number of accepted values so that it can be compared with the posterior easily. One can verify that the prior succesfully covers the parameter space.

a further analysis of our results confirms that, indeed, using only fst was a good choice.
Prior to implement abc method and use the rejection algorithm to estimate the posterior of T_{sim}, we need to scale our simulated statistic along with the observed statistic so that the mean is 0 and the standard deviation is one. After that is done, we implement the abc rejection algorithm, and plot the histogram for the accepted values of T_{sim}
To summarize this distribution one can calculate the mean, $\bar{T}_{sim} = 270586.9$, the mode $mode(T_{sim}) = 271534.5$ and the median $median(T_{sim}) = 270550.9$. Since the distribution is pretty simmetrical, the values are pretty similar. The 95% HDP is $[265329.5, 276769.6]$

The code used to implement the draw correlation plots and implement the rejection algorithm is shown below.

```
1     all = rbind(obsSummaryStats, simSummaryStats)
2     #Scale all columns separatedly
3
4     for (i in seq(1,ncol(all))){
5     all[,i] = scale(all[,i])
6     }
7
8     obsSummaryStats_scaled = all[1,]
9     names(obsSummaryStats_scaled) = names(obsSummaryStats)
10    simSummaryStats_scaled = all[2:nrow(all),]
11    names(simSummaryStats_scaled) = names(obsSummaryStats)
12
13    corr_plots = function(variables, T_sim, names){
```

```
14      nrows = ceiling(sqrt(ncol(variables)))
15      par(mfrow = c(nrows, nrows))
16      for (i in seq(1,ncol(variables))){
17      plot(T_sim, variables[,i], ylab = names[i], xlab = expression('T'[sim↩
            ]))
18
19      }
20      }
21
22      corr_plots(simSummaryStats, Tsim, names(obsSummaryStats))
23
24      rej = abc(target = obsSummaryStats_scaled[1],
25      param = Tsim,
26      sumstat = simSummaryStats_scaled[,1],
27      tol = 10^(-2),
28      method = 'rejection')
29
30      hist(rej)
```

Ecological applications of Bayesian networks

Secondary extinctions in food webs: a Bayesian network approach

- **Hypothesis**
  This paper addresses the suitability of a bayesian network description of food webs when it comes to predicting secondary extinctions.

- **Use of Bayesian Networks**
  In this study, the food web is represented by a Bayesian network. Each variable/node is a species, and the conditional dependencies of each variable represent the feeding interactions between species. The aim is to obtain the probability of extinction of a certain species by expressing it in terms of the state (extinct or extant) of its resources. In this work, 3 functional forms of a consumer's response to resource loss are studied: topological, linear, and nonlinear.
  Once a Bayesian network has been built, ie, a table for each species specifying its probability of extinction given the state of its resources, an approximate way to solve the network is by performing simulations in the following way. Starting at the primary producers of the network (only acyclic networks are considered) one performs a Bernoulli trial where the probability of fail equals the baseline probability of extinction of that species. This yields the state of the first resource. This process is repeated in an ascending fashion, so that when the apex predator (or, for instance, species that is I levels above the primary producers) is reached, the states of the resources on which it depends is known. Repeating this method over many simulations would yield approximate marginal probabilities of extinction.
  Another way to solve the network is to calculate marginal probabilities analytically. Taking into account that food webs are not completely connected, efficient algorithms exist to do this solve bayesian networks analytically.
  To test this results against some kind of 'ground truth', the dynamics of large ecological networks were simulated with the Allometric Trophic Network (ATN) model. The end state of each species was simulated and recorded for k replicates of the dynamical simulation. With these simulations, the likelihood that a Bayesian network would produce the same results obtained in the ATN model is calculated. To provide a baseline case, these results are contrasted with those obtained by setting each marginal probability to its maximum-likelihood estimate obtained from the simulations of the ATN model.

- **Results**
  From the different algorithms (functional forms of a consumer's response to resource loss) tested, the non-linear (particularly the sigmoid function) algorithm was always the best performing one (its likelihood was closest to the maximum attainable likelihood provided by the ATN model).

- **Learning outcomes**

  Bayesian networks are a powerful tool when examining ecological networks because they keep the simplicity of describing an ecosystem by its adjacency matrix (topological approach) while allowing the inclusion of more realistic parametrizations, such as explicitly modeling the functional response to the loss of resources, or including specific ecological knowledge about species extinction risks in the baseline probability of extinction (the species extinction probability if it was not embedded in the network). This approach does not require to dynamically simulate extinctions by removing primary species and record the number of secondary extinctions that the removal causes, which is computationally very expensive because of the large number of parameters and high dependency on initial conditions.

  Also, Bayesian networks are tricky to implement when you have cyclic data. However, because not all connections contribute to the robustness of an ecological network, and removing the redundant connections leaves the network acyclic, this is not a big problem. Overall, Bayesian networks are a promising method to effectively implement realistic and large ecological networks, providing a way to bridge the gap between theoretical ecology and conservation biology.