**Supporting information**

**Supplemental materials and methods**

**Reducing the number of summary statistics**

For each set of 1M simulations, we used a two-step process to reduce the number of summary statistics. We first used a “training set” of 10,000 additional, independent simulations to calculate pairwise correlations between summary statistics. Then, for each pair of statistics, if the absolute correlation coefficient was higher than 0.8, we discarded one of the two statistics. To decide which statistic of the two to discard, we used the same training set of simulations to perform a linear regression of one of the two statistics onto the each model parameter and repeated this process for the second statistic. We kept the statistic with the strongest association with most model parameters. For sets of simulations where the number of uncorrelated statistics was below 5, we raised the correlation coefficient threshold from 0.8 to 0.9. Once we had a set of uncorrelated statistics, we transformed the summary statistics using partial-least-squares (PLS) regression to the training set of 10,000 independent simulations using R scripts provided with the ABCtoolbox program (Wegmann, Leuenberger, Neuenschwander, & Excoffier, 2010). The principal components obtained from the PLS transformation were used as new summary statistics. There is a consensus that the appropriate number of summary statistics is close to the number of model parameters. As our models have between 3 and 5 parameters we chose to keep the 5 first PLS components for each ABC analysis performed. For several combinations of models and datasets, most statistics were highly correlated and impaired the PLS transformation. For these training sets of simulations, we raised the correlation coefficient threshold to 0.95 in order to perform a successful PLS transformation. In one case (model4 + dataset1 with known haplotypic phase), the ABC estimation step failed because two PLS components were too correlated to perform the estimation. We solved this problem by lowering the threshold for correlation of untransformed statistics in the training set from 0.8 to 0.7.

**Estimating model parameters using the SFS**

We used fastsimcoal2 (Excoffier, Dupanloup, Huerta-Sánchez, Sousa, & Foll, 2013; Excoffier & Foll, 2011) to simulate 100 pseudo-observed datasets (PODs) of type 1 (10,000 sequences of 100bp). We did this for each of the four models depicted in figure 1. The SFS from these 100 PODs was then input into fastsimcoal2, which approximates a composite likelihood from a number of simulations set by the user (here we chose 10,000 simulations) and iteratively performs a conditional maximization algorithm (ECM) to estimate the parameter values corresponding to the maximum likelihood. We allowed 20 to 40 ECM cycles with a stopping criterion (minimum relative difference in parameters between two iterations) of 10-3. For each POD, we performed 50 iterations of simulations and ECM and retained the parameter estimates with highest maximum likelihood. We used these estimates to calculate the RMSE of each parameter for each model. Ten datasets were used as observations for parameter estimation in both the ABC and the SFS inference frameworks. We included a parametric bootstrap step to the SFS inference: for each of the 10 PODs, we simulated 100 SFS using the maximum likelihood values obtained from the estimation as true values. Then, we re-ran the estimation in an identical manner for these 100 SFS. 95% confidence intervals were calculated for each of the 10 PODs from the quantiles of the parameter estimates from the 100 SFS bootstraps, using custom R scripts (R Core Team, 2016).

**Supplemental figures and tables**

**Figure S1.** Accuracy and precision of estimates for model parameters. Each panel represents results for a combination of demographic model and type of genetic data simulated. Points represent the mode of the posterior distribution for each POD. Black lines represent the positions corresponding to a perfect estimation, where the posterior mode equals the true value of the parameter. Error bars represent the 95% HDI for each POD estimated.

**Figure S2.** Root mean squared error of model parameter for different fixed values of TEXP. Note that the TEXP values are represented on a log scale for better visibility on results for recent demographic events. Different colors represent different types of simulated datasets.

**Figure S3.** This figure is similar to figure S2 but the y-axis represents 95% highest posterior density intervals and their standard errors (N=100 PODs).

**Table S1**. Comparison of performance of model 2 (4-parameter model including n02 as a parameter, with exponential growth of population 2) with a corresponding model where population 2 experiences a sudden size change at TEXP/10. The simulated datasets are 10k sequences of length 100bp. The value 1 for haplotype phase means that the LD-based stats are included in the summarization step, 0 means that they are excluded. Prediction error based on 1000 random simulations is displayed. 95% HDI was averaged over 100 random simulations and the corresponding standard error is shown in brackets. 95%HDI and s.e. values are rounded to the nearest integer.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **haplotype phase** | **parameter** | **Prediction error** | | **mean 95% HDI [se]** | |
| **growth** | **N change** | **growth** | **N change** |
| 1 | n1 | 0.018 | 0.013 | 13668 [411] | 11547 [314] |
| 1 | n2 | 1.509 | 1.393 | 81886 [63] | 82763 [33] |
| 1 | n02 | 0.682 | 0.635 | 704 [21] | 668 [22] |
| 1 | TEXP | 0.604 | 0.697 | 343 [10] | 325 [11] |
| 0 | n1 | 0.001 | 0.002 | 3021 [94] | 5541 [303] |
| 0 | n2 | 1.57 | 1.474 | 81692 [151] | 82688 [39] |
| 0 | n02 | 0.577 | 0.623 | 628 [25] | 685 [21] |
| 0 | TEXP | 0.327 | 0.579 | 275 [10] | 294 [12] |

**Script**

#R script to create a set of simulations and calculate summary statistics:

##########################################################################

#!/usr/bin/env Rscript

argus=commandArgs(trailingOnly=TRUE)

nreps=as.numeric(argus)[1] #number of simulations to run

N0=10000 #hypothetical population size used to scale coalescent time and population sizes

#dataset

sampsize=40 #total number of diploid individuals

nbloci=10000 #number of sequences

loclength=100 #length of sequences

#parameter priors

n1 = round(runif(nreps, min=10000, max=100000))

n2 = round(runif(nreps, min=10000, max=100000))

n02 = round(runif(nreps, min=2, max=1000))

m12 = runif(nreps, min=0, max=0.01)

TEXP = round(runif(nreps, min=2, max=500))

rec = 10^-8

mu = 9\*10^-9

#scaled parameter values

N1 = n1/N0

N2 = n2/N0

N02 = n02/N0

TCOAL = TEXP/(4\*N0)

gr2 = -log(N02/N2)/TCOAL

thetamu = 4\*N0\*loclength\*mu

R = 4\*N0\*(loclength-1)\*rec

M12 = 4\*N0\*(loclength)\*m12

#simulating model with 5 parameters

params\_names = c("Sim","n1","n2","n02","TEXP")

#creating vector of column names for means of summary statistics

means\_names = c("m\_pi\_1","m\_pi\_2","m\_theta\_pi","m\_theta\_w\_1","m\_theta\_w\_2","m\_theta\_w","m\_tajimasD\_1","m\_tajimasD\_2","m\_tajimasD","m\_ZnS\_1","m\_ZnS\_2","m\_ZnS","m\_perc\_shared\_1\_2","m\_perc\_private\_1\_2","m\_perc\_fixed\_dif\_1\_2","m\_pairwise\_fst\_1\_2","m\_FayWuH\_1","m\_FayWuH\_2","m\_FayWuH","m\_dvk","m\_dvh","m\_thomson\_est\_1","m\_thomson\_est\_2","m\_thomson\_est","m\_thomson\_var\_1","m\_thomson\_var\_2","m\_thomson\_var")

sums\_names = c("s\_segs\_1","s\_segs\_2","s\_segs","s\_segs\_1\_pr","s\_segs\_2\_pr")

#creating vector of column names for means of summary statistics

vars\_names = c("v\_pi\_1","v\_pi\_2","v\_theta\_pi","v\_theta\_w\_1","v\_theta\_w\_2","v\_theta\_w","v\_tajimasD\_1","v\_tajimasD\_2","v\_tajimasD","v\_ZnS\_1","v\_ZnS\_2","v\_ZnS","v\_perc\_shared\_1\_2","v\_perc\_private\_1\_2","v\_perc\_fixed\_dif\_1\_2","v\_pairwise\_fst\_1\_2","v\_FayWuH\_1","v\_FayWuH\_2","v\_FayWuH","v\_dvk","v\_dvh","v\_thomson\_est\_1","v\_thomson\_est\_2","v\_thomson\_est","v\_thomson\_var\_1","v\_thomson\_var\_2","v\_thomson\_var")

#creating datasets for summary statistics

tabparams = data.frame(matrix(vector(), nreps, length(params\_names), dimnames=list(c(), params\_names)), stringsAsFactors=F)

tabsums = data.frame(matrix(vector(), nreps, length(sums\_names), dimnames=list(c(), sums\_names)), stringsAsFactors=F)

tabmeans = data.frame(matrix(vector(), nreps, length(means\_names), dimnames=list(c(), means\_names)), stringsAsFactors=F)

tabvars = data.frame(matrix(vector(), nreps, length(vars\_names), dimnames=list(c(), vars\_names)), stringsAsFactors=F)

#running the simulations

for (i in 1:nreps) {

system(paste("scrm",sampsize,nbloci,"-t",thetamu,"-r",R,loclength,"-I",2,sampsize/2,sampsize/2,"-n",1,N1[i],"-n",2,N2[i],"-g",2,gr2[i],"-ej",TCOAL[i],2,1,"-eg",TCOAL[i],2,0,"> scrm.out",sep=" "))

system("echo '' >> scrm.out")

#calculating summary statistics on simulated sequences system(paste("/data/joane/programs/msABC20120315/msABC",sampsize,nbloci,"-I",2,sampsize/2,sampsize/2,"--obs scrm.out --options ms\_ssdefs.txt > scrm.out.mean",sep=" "))

tab = read.table("scrm.out.mean",header=TRUE)

#selecting variable sequences

tabpoly=tab[which(tab$s\_segs>0),]

#calculating sums, means, and variances over all sequences

tabpoly$s\_segs\_1\_pr = tabpoly$s\_segs - tabpoly$s\_segs\_2

tabpoly$s\_segs\_2\_pr = tabpoly$s\_segs - tabpoly$s\_segs\_1

params=c(i,n1[i],n2[i],n02[i],TEXP[i])

sums = colSums(tabpoly[,c("s\_segs\_1","s\_segs\_2","s\_segs","s\_segs\_1\_pr","s\_segs\_2\_pr")])

means = colMeans(tabpoly[,c("s\_pi\_1","s\_pi\_2","s\_theta\_pi","s\_theta\_w\_1","s\_theta\_w\_2","s\_theta\_w","s\_tajimasD\_1","s\_tajimasD\_2","s\_tajimasD","s\_ZnS\_1","s\_ZnS\_2","s\_ZnS","s\_perc\_shared\_1\_2","s\_perc\_private\_1\_2","s\_perc\_fixed\_dif\_1\_2","s\_pairwise\_fst\_1\_2","s\_FayWuH\_1","s\_FayWuH\_2","s\_FayWuH","s\_dvk","s\_dvh","s\_thomson\_est\_1","s\_thomson\_est\_2","s\_thomson\_est","s\_thomson\_var\_1","s\_thomson\_var\_2","s\_thomson\_var")],na.rm=T)

vars = sapply(tabpoly[,c("s\_pi\_1","s\_pi\_2","s\_theta\_pi","s\_theta\_w\_1","s\_theta\_w\_2","s\_theta\_w","s\_tajimasD\_1","s\_tajimasD\_2","s\_tajimasD","s\_ZnS\_1","s\_ZnS\_2","s\_ZnS","s\_perc\_shared\_1\_2","s\_perc\_private\_1\_2","s\_perc\_fixed\_dif\_1\_2","s\_pairwise\_fst\_1\_2","s\_FayWuH\_1","s\_FayWuH\_2","s\_FayWuH","s\_dvk","s\_dvh","s\_thomson\_est\_1","s\_thomson\_est\_2","s\_thomson\_est","s\_thomson\_var\_1","s\_thomson\_var\_2","s\_thomson\_var")], FUN=var,na.rm=T)

tabparams[i,] = params

tabsums[i,] = sums

tabmeans[i,] = means

tabvars[i,] = vars

}

stat\_tab=cbind(tabparams,tabsums,tabmeans,tabvars)

#removing potential columns with invariant statistics

ulist = sapply(stat\_tab, FUN=unique)

uvec = sapply(ulist, FUN=length)

simtab\_var=stat\_tab[,which(as.numeric(uvec)>2)]

#saving the output

write.table(simtab\_var,file=argus[2],quote=FALSE,row.names=FALSE,sep="\t")