Journal of Heredity
doi:10.1093/jhered/ess038

FLOCK Provides Reliable Solutions to the "Number of Populations" Problem

Pierre Duchesne and Julie Turgeon

From the Département de biologie, Université Laval, Québec, Québec, G1V 0A6 Canada.

Address correspondence to Julie Turgeon at the address above, or e-mail: julie.turgeon@bio.ulaval.ca.

Abstract

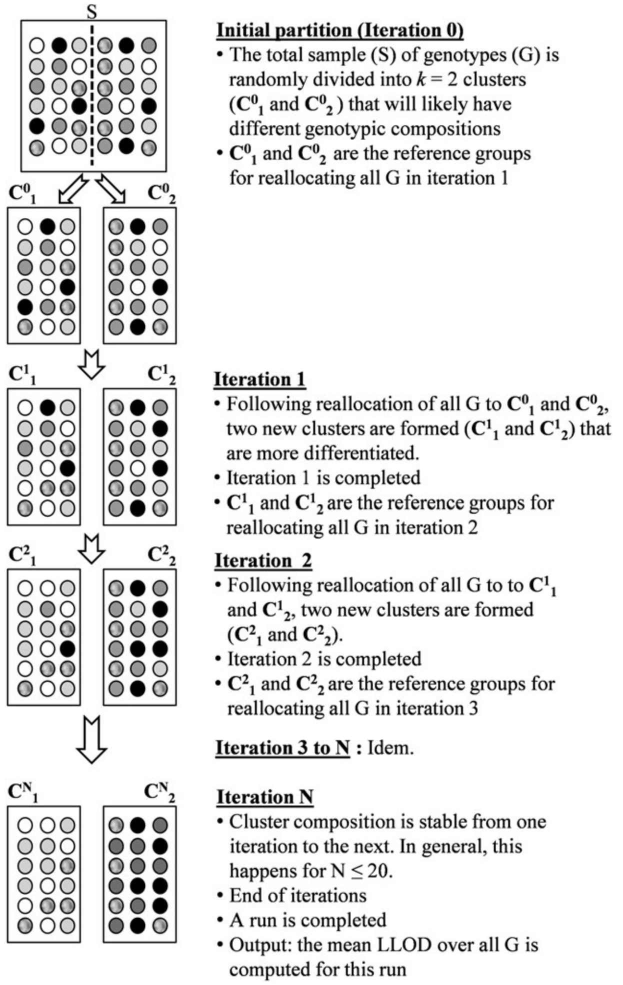
Identifying groups of individuals forming coherent genetic clusters is relevant to many fields of biology. This paper addresses the *K*-partition problem: given a collection of genotypes, partition those genotypes into *K* groups, each group being a sample of the *K* source populations that are represented in the collection of genotypes. This problem involves allocating genotypes to genetic groups while building those groups at the same time without the use of any other a priori information. FLOCK is a non-Markov chain Monte Carlo (MCMC) algorithm that uses an iterative method to partition a collection of genotypes into *k* groups. Rules to estimate *K* are formulated and their validity firmly established by running simulations under several migration rates, migration regimes, number of loci, and values of *K*. FLOCK tended to build clusters largely consistent with the source samples. The performance of FLOCK was also compared with that of STRUCTURE and BAPS. FLOCK provided more accurate allocations to clusters and more reliable estimates of *K*; it also ran much faster than STRUCTURE. FLOCK is based on an entirely novel approach and provides a true alternative to the existing, MCMC based, algorithms. FLOCK v.2.0 for microsatellites or for AFLP markers can be downloaded from http://www.bio.ulaval.ca/no_cache/departement/professeurs/fiche_des_professeurs/professeur/11/13/.

Key words: *clustering program, genetic differentiation, iterative method, population allocation*

Detecting population genetic structure is central to many fields of biology (e.g., Waples and Gaggiotti 2006). Clustering individual genotypes into distinct groups without the use of any a priori information, such as group membership or individual location, can help identify unsuspected conservation units, migrants, or admixed individuals. The identification of such groups may be more formally defined as the "*K*-partition problem," which consists in partitioning a collection of individuals, based on trait(s) measurements such as genotypes, into subsets corresponding to *K* distinct populations where *K* is not known in advance. This amounts to a sorting process without preexisting categories. The estimation of the number of categories/clusters is based on secondary data generated by this same sorting process. In a nutshell: from a mixed bag of genotypes and nothing else, the aim is to group together those genotypes that were drawn from the same population. The terms "cluster," "group," and "reference" are considered synonyms in this paper.

Several algorithms have been developed with the purpose of recovering biologically significant clusters from a set of genotypes (e.g., Pritchard et al. 2000; Dawson and Belkhir 2001; François et al. 2006; Chen et al. 2007; Corander et al. 2008; Guillot et al. 2008; Hubisz et al. 2009). Most are variants of the pioneering approach of Pritchard et al. (2000) using a model-based Markov chain Monte Carlo (MCMC) algorithm to define *k* clusters containing groups of genotypes maximizing Hardy–Weinberg and linkage equilibrium (HWLE) and displaying allele frequency differences. However, the only programs designed to solve the *K*-partition problem as we defined it above are STRUCTURE (Pritchard et al. 2000) and MGD (Rodriguez-Ramilo et al. 2009). Other programs often inform the clustering process with spatial information on individuals (e.g., François et al. 2006; Guillot et al. 2008, 2009; Hubisz et al. 2009) and/or the classification of samples on the basis of sampling site or other criteria (STRUCTURE version 2.3, Hubisz et al. 2009). STRUCTURE generally serves as a benchmark in relative performance tests of genetic clustering programs.

FLOCK (Duchesne and Turgeon 2009) is a clustering program which was first introduced as a method for quick mapping of admixture without source samples. In contrast with the aforementioned clustering programs, it is not based on MCMC sampling but on iterated reallocation (Figure 1 and Appendix I for a formal description of the algorithm). First, FLOCK randomly partitions the collection of genotypes into *k* groups (i.e., clusters). Allele frequencies are estimated for each of the *k* groups, and each genotype is reallocated to the group with the highest likelihood score following the multilocus maximum likelihood method of Paetkau et al. (1995). This results in another set of *k* groups.



Initial partition (Iteration 0)
- The total sample (S) of genotypes (G) is randomly divided into $k = 2$ clusters ($\mathbf{C}^0_1$ and $\mathbf{C}^0_2$) that will likely have different genotypic compositions
- $\mathbf{C}^0_1$ and $\mathbf{C}^0_2$ are the reference groups for reallocating all G in iteration 1

Iteration 1
- Following reallocation of all G to $\mathbf{C}^0_1$ and $\mathbf{C}^0_2$, two new clusters are formed ($\mathbf{C}^1_1$ and $\mathbf{C}^1_2$) that are more differentiated.
- Iteration 1 is completed
- $\mathbf{C}^1_1$ and $\mathbf{C}^1_2$ are the reference groups for reallocating all G in iteration 2

Iteration 2
- Following reallocation of all G to to $\mathbf{C}^1_1$ and $\mathbf{C}^1_2$, two new clusters are formed ($\mathbf{C}^2_1$ and $\mathbf{C}^2_2$).
- Iteration 2 is completed
- $\mathbf{C}^2_1$ and $\mathbf{C}^2_2$ are the reference groups for reallocating all G in iteration 3

Iteration 3 to N : Idem.

Iteration N
- Cluster composition is stable from one iteration to the next. In general, this happens for $N \leq 20$.
- End of iterations
- A run is completed
- Output: the mean LLOD over all G is computed for this run

Figure 1. Schematic representation of the FLOCK algorithm. Starting from a random partition of the total sample (S) into *k* clusters (here, $k = 2$), a series of reallocations leads to the formation of *k* clusters that are increasingly differentiated. A formal description of FLOCK is provided in Appendix I.

Because each group tends to attract similar individuals, these new groups will be more homogeneous and better differentiated. A round of reallocation is an iteration. A series of such "iterations," leading to a partition of all genotypes into *k* groups, represents a "run." Typically, within 5–10 iterations, the composition of the *k* groups either becomes stable or changes very slightly from one iteration to the next. This convergence was never observed to occur beyond 30 iterations, irrespective of number of individuals and number of loci. At the end of each run, FLOCK calculates the log likelihood difference (LLOD) score for each genotype, that is, the difference between the log likelihood of the most likely cluster for this genotype and that of its second most likely cluster as well as the mean LLOD over all genotypes. For each run, FLOCK allocates each genotype to the cluster with the highest likelihood score for that genotype. Note that LLOD values are output variables; they are not driving nor affecting the clustering process.

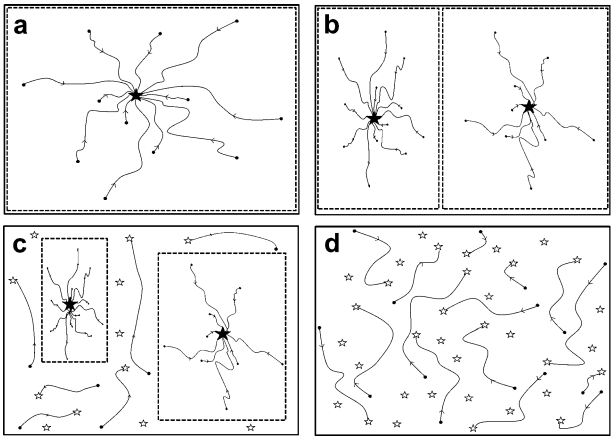


Figure 2. Schematic representation of several FLOCK runs traveling from one partition to the next through the space of all partitions for a single value of *k* (the space of *k*-partitions). The black dots do not stand for genotypes or clusters but for a specific partition. Each curved arrow represents a run starting from one random partition and ending on an attractor (a black star) or a weak attractor (a small open star). Each rectangle is associated with a single value of *k* and represents the space of all *k*-partitions. Basins of attraction are bordered by a dashed line. A weak attractor is the endpoint of only one run. In (**a**) and (**b**), 1 and 2 basins of attraction cover the entire space of *k*-partitions, respectively, such that all runs end on an attractor. In (**c**), the space of *k*-partitions is not covered by the 2 basins of attraction, such that runs may hit one of the attractors or end on a weak attractor. In (**d**), there are no attractors in the space of all *k*-partitions.

The very short processing time for each run of FLOCK allows for comparing partitions from many runs for each *k*. For any value of *k*, there is a huge number of possible partitions (the space of all *k*-partitions), and there is only an infinitesimal probability that two runs, each starting from a random initial partition, hit on the same end-partition by sheer luck. Indeed, there has to be something "attractive" about a partition that turns up more than once, and accordingly, we refer to such a partition as an "attractor." Each attractor has its own basin of attraction (Brin and Stuck 2002), that is, a collection of partitions that are similar enough to the attractor state that they will end up hitting this attractor. Within the huge space of *k*-partitions, there may be one or several basins of attraction, entirely covering that space or not (Figure 2).

The set of runs that hit the same attractor can be identified by FLOCK as runs that produce the very same mean LLOD score. Each mean LLOD is computed with a high degree of precision (14 decimals). Therefore, there is an extremely high probability that two identical mean LLOD values are associated with the exact same *k*-partition of the collection of genotypes. A set of runs that hit the same attractor (same mean LLOD value) will be termed a "plateau" and their number the "plateau length." When several plateaus are generated, their associated lengths will be referred to as the "plateau sequence" for *k*. A "plateau record" is the list of plateau sequences for each *k* value requested by the user. We use the analysis of plateau records to solve the *K*-partition problem. This analysis is termed "plateau analysis." Appendix II shows how FLOCK provides numerical and graphical information on plateau sequences for many values of *k*.

Note that FLOCK is very different from other clustering programs. It does not sample the space of partitions through small random step walks as in MCMC, and it does not try to optimize some target function, such as HWLE. Briefly stated, it is not based on a probabilistic search algorithm. On the contrary, FLOCK is entirely deterministic. The result of each run depends solely on the composition of the initial randomly created partition (iteration 0 on Figure 1). Thereafter, the iterative reallocation process will always yield the same sequence of partitions.

Here, we show that FLOCK provides highly reliable solutions to the *K*-partition problem. This reliability is established through simulations run under a large number of parameter configurations involving migration rate, migration regime, number of loci, and the number *K* itself. Based on attractor information, a set of rules for estimating *K* are tested and validated.

Materials and Methods

We used a 3-step approach. First, we ran FLOCK on several empirical data sets both with microsatellite and AFLP markers and for a wide variety of taxa (bird: *Setophaga ruticilla*; fish: *Coregonus artedi*, *Salmo trutta*, *Mallotus villosus*, *Anguilla marmorata*; mammals: *Rattus rattus*, *Delphinapterus leucas*; insects: *Enallagma hageni*, *E. ebrium*, *Gerris gilletei*, *Acyrthosiphon pisum*, *Aphidius ervi*; crustaceans: *Calanus finmarchicus*, *C. glacialis*). For these data sets, *K* had been estimated with a reasonable degree of certainty using other methods. Plateau analyses on these data sets lead us to define 2 types of ad hoc rules: stopping rules to determine the upper *k* value beyond which to stop running FLOCK and estimation rules for *K* (note that *K* refers to the true number of groups/populations while *k* denotes the user-defined parameter that forces clustering into *k* clusters). The second step consisted in validating those rules on simulated data sets that spanned a large array of parameter values. Finally, FLOCK, BAPS, and STRUCTURE were also compared for accuracy in estimating *K* and in retrieving the original cluster compositions.

Observations Based on Empirical Data

With each data set and each of many values of *k*, 50 runs were performed, and we observed 0, 1, or several plateaus of variable lengths. Two major observations were made: plateaus tended to be longer when *k* was equal to the estimated value of *K* and plateaus always eventually vanished with increasing values of *k*. With some data sets, some value of *k*, say k', produced a single plateau, and the length of this plateau was ≥6 (runs among 50). Then, k' was equal to the presumed value of *K*. This condition (a single isolated plateau of length ≥6) is a stopping rule for the sequence of *k* values with which to run FLOCK because it is no use considering higher values of *k* to estimate *K*. The point estimate $K = k'$ is the estimation rule associated with this stopping rule. With other data sets, it was found that whenever plateaus were absent for 4 successive values of *k*, then no plateau was ever observed for higher values of *k*. Then, the largest value of *k* comprising a plateau of length ≥6 (runs among 50), say k', was always smaller or equal to *K*. Consequently, this condition (4 successive values of *k* with no plateau) was taken as a second stopping rule. It is associated with an estimation rule whereby k' is a lower bound for *K*, that is, $K \geq k'$. Lower bound estimates will be referred to as *k*+ estimates where $k+ = \{k, k+1, k+2, k+3, k+4, \ldots\}$. When there are no plateaus of length ≥6, no estimates are provided for *K*, and the plateau analysis yields an "undecided" verdict. Appendix II presents a flowchart diagram detailing how stopping and estimation rules are applied in association with examples of plateau records.

Validation Tests Based on Simulated Data Sets

Simulated data sets were obtained using EASYPOP (Balloux 2001) and combining parameter values as per Waples and Gaggiotti (2006), including mutation rate dynamics. In brief, each simulation was run for 5000 generations in an attempt to reach approximate mutation-drift equilibrium, each population comprising 500 individuals (sex ratio 1:1). Genotypes in the initial generation were randomly drawn from all 10 possible allelic states (Max diversity). Three parameters of interest were varied, namely N_{m}, the average number of individuals migrating to another population; the number of loci (*L*); and *K*, the true number of populations. Simulated data sets were generated for 2 migration models: finite island and stepping stone. Altogether, values of N_{m}, *L*, and *K* and migration models span a very wide set of parameter conditions. For the finite island model, a simulation was performed for all 60 combinations of several values for N_{m} (0.01, 1, 5, 375), *L* (10, 20, 30, 40, 50), and *K* (2, 4, 8). For the stepping stone model, 20 simulations were performed using the above values for N_{m} and *L*. The number of populations was $K = 8$ because results specific to the stepping stone model were less likely to be apparent with $K = 4$ and would be absent with $K = 2$. For the sake of comparison, genetic differentiation (θ_{ST}) was estimated with FSTAT 2.9.3.2 (Goudet 2002) at 0.38, 0.14, 0.03, and 0.001 for the island model ($K = 4$ and $L = 30$) and at 0.39, 0.21, 0.03, and 0.001 for the stepping stone model ($K = 8$, $L = 30$) for $N_{\mathrm{m}} = 0.01$, 1, 5, and 375, respectively.

One simulated data set was generated for each of the parameter combination and migration regime described above. For each of those data sets, a random sample of 30 individuals per population was taken to perform 50 runs of FLOCK for successive values of *k* until one stopping condition was reached. The stopping and the *K*-estimating rules were tested on the plateau records automatically generated by FLOCK. The proportion of genotypes allocated to the right cluster was also computed. Only the allocations corresponding to the k value involved in the estimates (as in $K = k$ or $K = k+$, see Results) were assessed.

Performance Comparisons with Other Programs

FLOCK was first compared with STRUCTURE (Pritchard et al. 2000) and BAPS (Corander et al. 2008), currently the two most widely used clustering programs. To this purpose, we used published data sets from Latch et al. (2006) for 5 F_{ST} levels (0.01, 0.02, 0.03, 0.04, and 0.05) where clustering proved challenging. Each set comprises 5 population samples (each including 100 genotypes at 10 codominant unlinked loci), and therefore, $K = 5$ is the target estimate. K-estimates and proportion of correct allocations for STRUCTURE and BAPS (version 3.1, Corander et al. 2005) are reported from Latch et al. (2006). FLOCK was run on each data set with k starting from 2 and increasing on until one stopping condition was reached. For each data set and each k value, 50 runs were performed, and the number of iterations per run was 20.

Second, a closer comparison between FLOCK and STRUCTURE was performed by running both programs on data sets simulated as above. Five replicate data sets were used for each of 9 combinations ($N_m = 5$; $L = 10$, 30, and 50; $K = 2$, 4, and 8; island model). $N_m = 5$ was chosen because it defines a zone where clustering is difficult enough that other parameters become very influential but without making the clustering problem generally insoluble (Waples and Gaggiotti 2006, see also Results). The running parameters for FLOCK were as above, and K was estimated based on plateau analysis as described previously. For the STRUCTURE program, we used parameter levels frequently found in the current literature. For each run, burn-in was set at 50 000, the number of iterations was 200 000, and the admixture model with correlated allele frequencies was chosen. We performed 10 runs for each value of k, from $k = 1$ to $K + 3$. The most likely number of clusters, K, was estimated using the standard criteria of Pritchard et al. (2000, i.e., the highest $\Pr(X|k)$) and that of Evanno et al. (2005, i.e., the highest second order rate of change of $\text{Ln}[\Pr(X|k)]$). FLOCK was compared with STRUCTURE for accuracy in estimating K and for the proportion of correct allocations. In addition, computing times for both programs were collected on a PC with an Intel CoreI Duo CPU, E6750 2.66 GHz and 3 Go of RAM.

Results

In the following sections, an estimate for K will be considered correct when K is equal to the point estimate k or when K is equal to the lower bound k in $k+$. The term undecided will be used whenever there are no plateaus of length ≥6 on observing 4 consecutive values of k with no plateaus. This is equivalent to a nonestimate.

Table 1 Estimated values of K in simulations with the finite island and the stepping stone migration models for different combinations of number of populations (K), number of loci (L), and migration rate (N_m)

			N_m			
Model	K	L	0.01	1	5	375
Island	2	10	2	2	2+	Undecided
		20	2	2	2+	Undecided
		30	2	2	2+	Undecided
		40	2	2	2+	Undecided
		50	2	2	2	Undecided
	4	10	4	4+	Undecided	Undecided
		20	4	4	Undecided	Undecided
		30	4	4	4+	Undecided
		40	4	4	4+	Undecided
		50	4	4	4+	Undecided
	8	10	8	8+	Undecided	Undecided
		20	8	8	Undecided	Undecided
		30	8	8	8+	Undecided
		40	8	8	Undecided	Undecided
		50	8	8	Undecided	Undecided
Stepping stone	8	10	8	8+	4+	Undecided
		20	8	8	4+	Undecided
		30	8	7+	7+	Undecided
		40	8	8	7+	Undecided
		50	8	8	6+	Undecided

Validation Tests Based on Simulated Data Sets

Results of the plateau analysis for simulations under the finite island migration model are reported in Table 1. Among the simulations with lower migration rates ($N_m = 0.01$ and 1), there was a majority of point estimates. In all cases, the point estimates were equal to K. For $N_m = 5$, most results were correct lower bound estimates, but the number of undecided increased with K. Simulations with a very high migration rate ($N_m = 375$), equating to panmixia (Waples and Gaggiotti 2006), were all evaluated as undecided.

Results of the plateau analysis with the stepping stone migration regime are reported in Table 1. At a low migration rate ($N_m = 0.01$), point estimates were obtained and always correct. For $N_m = 1$, the output was mixed, with 3 point estimates, all correct, and 2 lower bound estimates, 1 correct and 1 incorrect (7+). With $N_m = 5$, there were only lower bound estimates ($k+$), and all were incorrect. For $k+$ estimates, all true K values were either equal to k or larger, thus confirming that K is always bounded below by k in $k+$ estimates. Interestingly, when migration and admixture were higher, FLOCK tended to produce smaller lower bounds for K in a stepping stone regime and more undecided in a finite island regime, as if the whole structure vanished all at once.

Overall, FLOCK tended to allocate most individuals from the same sample to the same cluster. The average proportion of correct allocations for point estimates was 99.6%, indicating that the collection of genotypes was efficiently sorted into its K sources. For correct lower bound estimates ($k+$), proportions of correct allocations were at least 80% and averaged 91.7%.

Table 2 Results from running FLOCK on the simulated data sets from Latch et al. (2006)

F_{ST}	Data set number	Plateau sequence (with PL $\geq$ 6)	*K*-estimate	Most likely number of clusters	% Correct allocation
0.01	1–5	—	Undecided	None	—
0.02	1–5	—	Undecided	None	—
0.03	1	**6**, 4, 3, 4	5+	5	91.6
	2	**6**, 2, 2	5+	5	86.0
	3	—	Undecided	None	—
	4	**12**	2	2	—
	5	—	Undecided	None	—
					Mean: 88.8
0.04	1	**7**, 3, 4, 5, **7**, 2, 2	5+	5	96.4
	2	2, **8**, 2, 2	5+	5	95.0
	3	4, 4, 4, **18**	5+	5	96.8
	4	3, **7**, 2, 3	5+	5	96.6
	5	**23**, **25**	5+	5	97.0
					Mean: 96.4
0.05	1	**7**, 2, **22**, **14**	5+	5	98.8
	2	**46**	5	5	98.8
	3	**7**, **38**	5+	5	99.2
	4	**43**	5	5	99.0
	5	**47**	5	5	98.6
					Mean: 98.9

For each F_{ST} level and each data set, the results from FLOCK were submitted to plateau analysis (see Appendix II). The resulting estimate for *K*, the most likely number of clusters (if any), and the % of correct allocations are given when appropriate. Plateau lengths (PL) $\geq$ 6 are printed in bold.

Performance Comparisons with Other Programs

Data Sets of Latch et al. (2006) with K =5 for Various F_{ST} Levels

Table 2 shows the estimates for *K* when FLOCK was run on Latch et al. (2006) data sets. The point estimates and the lower bound estimates for *K* were all correct except for one point estimate ($K = 2$) at $F_{\mathrm{ST}} = 0.03$ (Table 2). At the 2 lowest F_{ST} levels (0.01, 0.02), *K* was always undecided. Here, FLOCK was able to explicitly signal that it cannot detect distinct clusters, indicating either that the data set is not structured into genetic clusters, as in panmixia, or that the genetic information (essentially the number of polymorphic loci) is not sufficient to distinguish clearly between clusters. By contrast, Latch et al. (2006) reported that STRUCTURE produced $K = 1$ point estimates for all replicates at $F_{\mathrm{ST}} = 0.01$ and only wrong estimates at $F_{\mathrm{ST}} = 0.02$. It thus appears that the posterior probabilities for *K* were never flat enough to avoid yielding a (wrong) point estimate. Similarly, these authors indicated that at $F_{\mathrm{ST}} = 0.02$, *K*-estimates from BAPS stood between 2 and 4, and the probabilities associated with the apparent most likely number of clusters were extremely high. Thus, when F_{ST} is low, FLOCK yielded a safe undecided conclusion, whereas the other programs often provided incorrect point estimates.

Taking the clusters produced at $k = 5$ for both $K = 5$ and $K = 5+$ estimates, the average % of correct allocations was very high (≥96%) at all F_{ST} levels except 0.03 (88.8%). With F_{ST} = 0.03, 0.04, 0.05, the average % of correct allocations was generally higher than those of STRUCTURE and BAPS (Figure 3a). Sign tests performed based on the number of instances where the % of correct allocations was higher for FLOCK than for STRUCTURE and BAPS produced *P* values of 0.019 and 0.032. Therefore, the clusters built by FLOCK resembled more closely the original samples from the 5 populations.

In sum, FLOCK generally provided more accurate allocations to clusters and, even more importantly, safer more reliable K-estimates than both STRUCTURE and BAPS. However, compared with those 2 programs, FLOCK was sometimes more conservative when information became scarce for the running task.

Data Sets from EASYPOP with $N_m = 5$

K-estimates from FLOCK and STRUCTURE (using the criterion of Pritchard or Evanno) are reported in Figure 4. As expected, the largest differences between the 3 methods appear with the lower numbers of loci, especially when $L = 10$. With lower levels of information, FLOCK tended to output an undecided, STRUCTURE/Pritchard a $K = 1$ estimate, whereas STRUCTURE/Evanno produced a range of estimates most of them incorrect. Overall, FLOCK produced 28 correct estimates, 2 incorrect estimates, and 15 undecided. As for STRUCTURE/Pritchard, 21 estimates were correct and 24 were incorrect, including 21 $K = 1$ estimates.

The most striking difference between FLOCK and STRUCTURE/Pritchard appeared when $K = 2$. Indeed, FLOCK showed an increasing advantage in power as the number of loci decreased. The STRUCTURE/Pritchard estimates were $K = 1$ for all 5 replicates with $L = 10$ and $L = 30$, whereas FLOCK gave a $K = 2+$ estimate for 7 of those 10 data sets. Even with $L = 50$, STRUCTURE/Pritchard still output $K = 1$ estimates for 2 of the 5 data sets, whereas FLOCK's outputs were $K = 2$ (1 set) or $K = 2+$ (4 sets).

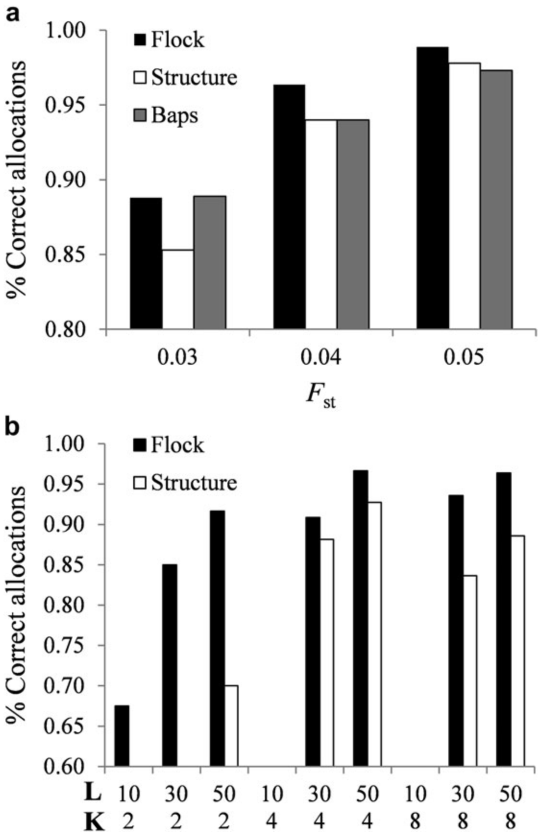


Figure 3. Average proportion of correct allocations (**a**) with FLOCK, STRUCTURE, and BAPS for F_{ST} = 0.03, 0.04, and 0.05 (values for BAPS and STRUCTURE from Latch et al. 2006) and (**b**) with FLOCK and STRUCTURE (Pritchard criterion) for K = 2, 4, and 8 with 10, 30, and 50 loci (L). Each average was computed over all cases where the most likely number of clusters was equal to the true value K.

To summarize, STRUCTURE/Pritchard produced more incorrect estimates than FLOCK and identified all replicates with $K = 2$, $L = 10$ and 30 as originating from a single population. STRUCTURE/Evanno output increasingly, broadly scattered, unreliable K-estimates as the number of loci decreased. Therefore, we will not consider the latter approach in further analyses, and so STRUCTURE/Pritchard will be referred to simply as STRUCTURE.

Proportions of correct allocations were established when both programs produced correct estimates for K (Figure 3b). For STRUCTURE, the proportion of correct allocations ranged from 0.817 to 1, averaging 0.895. For FLOCK, the proportions were generally higher, ranging from 0.887 to 0.987 with an average of 0.943. The differences in proportions of correct allocations between FLOCK and STRUCTURE averaged 0.044. A sign test was performed on those differences, and a statistically significant P value of 0.009 was obtained. So it appears that FLOCK restores the original clusters consistently better than STRUCTURE.

Computing time was much shorter for FLOCK than for STRUCTURE (Supplementary Material and Figures S1 and S2). On a per run basis, the STRUCTURE/FLOCK time ratios ranged from 225 to 75, depending on the parameter combination, that is, the number of genotypes (from 60 to 240 for $K = 2$ to $K = 8$), the number of loci (L), and the value of k (the number of clusters required by the user). Under most parameter conditions, the time ratios were close to 100. If one considers the total number of runs that were performed to estimate K, FLOCK total run time was, on average 5% that of STRUCTURE (observed range: 3.4–7.1%). This corresponds to 50 runs per k value until a stopping condition for k was met for FLOCK and 10 runs per k value for $k = 1$ to $K + 3$ for STRUCTURE. It is worth noting that under empirical conditions, STRUCTURE would often be run for higher values of k given that K is unknown.

Discussion

The FLOCK solution to the K-partition problem is based on the analysis of the lengths of plateaus over several values of k, referred to as "plateau analysis."

The program is run with increasing values of k, starting with $k = 2$, until 1 of 2 stopping conditions are met. That is, one isolated plateau of length ≥6 is found, or there are 4 successive values of k for which plateaus are entirely absent. For each condition, there is a corresponding estimation rule for K, the true number of populations represented in the collection of genotypes. The rules may lead to a point ($K = k$) or a lower bound estimation ($K \geq k$). Admittedly, lower bound estimates are less satisfactory than point estimates, but they are by no means useless. Indeed, a lower bound estimate signals that more than k clusters may exist and also that more loci should be added if one seeks a valid point estimate. Moreover, with most data sets, lower bounds coincided with K. When returning an undecided, FLOCK is in fact producing an "I don't know" statement that may signal a case of panmixia or a lack of resolution due to scarcity of information. In our opinion, this signal is clearly preferable to an erroneous estimate.

The provision of a complete set of stopping rules derived from an array of extensive simulations is one important feature of the FLOCK program. To our knowledge, sets of complete and unequivocal stopping rules are absent from other genotype partitioning programs searching for the true number of genetic groups K within a sequence of k values. With most of those programs, K is estimated by searching for k associated with the maximum value (mode) of some output variable, for example, Ln P(D) (STRUCTURE, Pritchard et al. 2000), ΔK (STRUCTURE, Evanno et al. 2005), deviance information criterion (TESS, François et al. 2008), and npop (GENELAND, Guillot et al. 2005). However, no clear criteria are provided to the user as to the choice of an upper bound for the sequence of k values. This is an important problem because a lack of resolution or absence of HWLE clusters may translate into exceedingly large or even unbounded values of the indicator variable as a function of k. Then, the questions as to when is one to stop increasing k and how is one supposed to

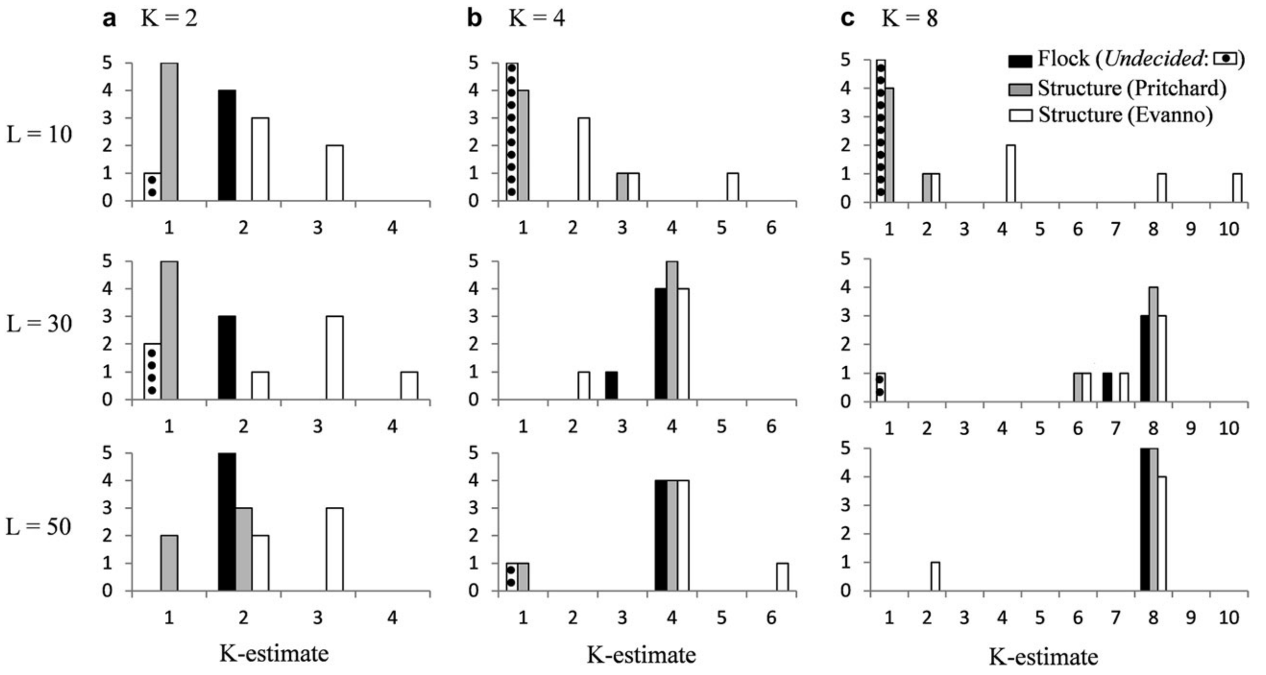


Figure 4. Comparisons of *K*-estimates from FLOCK and STRUCTURE (using Pritchard or Evanno criteria) for 5 replicates of simulated data sets with 10, 30, or 50 loci (*L*) when (**a**) $K = 2$, (**b**) $K = 4$, and (**c**) $K = 8$. The “undecided” from FLOCK are shown with a different filling as they indicate that there may be any number of clusters ($K \geq 1$).

estimate the true value *K* cannot be answered on a firm, objective ground. Typically, when those same programs are tested for validity, this difficulty is often eluded by choosing $K + 1 \ldots 3$ as an upper bound (e.g., Pritchard et al. 2000; Evanno et al. 2005; Chen et al. 2007; Hubisz et al. 2009). Now clearly, this borders on begging the question since *K* is precisely the main unknown to solve for when analyzing empirical data sets as opposed to simulated data. Not surprisingly, some studies have shown that the choice of the maximum value of *k* that is investigated might have an important impact on the estimation of *K* (Frantz et al. 2009; Guillot 2009).

Compared with STRUCTURE, currently the most widely used cluster program, FLOCK was found to provide more reliable estimates for *K*, a better match between the clusters it builds and the original samples, and much shorter computing times per run. Furthermore, FLOCK’s lower error rate in estimating *K* does not come at the expense of an excessive proportion of undecided cases. In fact, comparisons of *K*-estimations showed that either program may be more efficient in the low information range depending on the generating conditions of the data set.

With the data sets of Latch et al. (2006), we observed an intermediate zone where information level was sufficient for STRUCTURE to output accurate *K*-estimates but scarce enough that FLOCK often stuck to an undecided (I don’t know) answer. However, with still lower levels of information, STRUCTURE continued to output point estimates, but those were incorrect and therefore the resulting clusters were spurious.

By contrast, FLOCK showed more power when processing the $N_m = 5$, $K = 2$ and $L = 10$, 30 data sets generated by EASYPOP. And so, it seems that low information level is not alone in determining which program retains its *K*-estimation capability longer. We contend that introgression levels may be at play here. Indeed, the Latch et al. (2006) data did not integrate a migration process. With $N_m = 5$, EASYPOP gradually homogenizes populations over 5000 generations. When $K = 2$, this should translate earlier than with higher values of *K* into highly homogenized populations. In fact, with $K = 8$, any pair of populations will retain its distinctiveness longer than 2 populations ($K = 2$) that exchange migrants only between themselves generation after generation. Since STRUCTURE is searching for clusters that are in HWE and LE, it is bound to perform best when dealing with populations experiencing weak migration. On the other hand, FLOCK will simply sort the genotypes from populations A and B into more A_like and more B_like clusters, hence its robustness when tackling high levels of introgression. We believe that this may explain why STRUCTURE shows more power when migration is low, whereas FLOCK shows more when migration is high and sustained for numerous generations.

Conclusion

The *K*-partition problem is difficult to solve because it involves allocating genotypes to genetic groups while building those groups at the same time. Instead of MCMC sampling the gigantic space of possible partitions (Hubisz et al. 2009), FLOCK seeks solutions by iterating a reallocation process and identifying attractors within the space of *k*-partitions. Quick convergence allows for obtaining and comparing partitions from numerous runs for each of several values of *k*.

A set of stopping rules was derived from an array of extensive observations. This set of rules is complete: when running a sequence of *k* starting with $k = 2$, 1 of the 2 stopping conditions is always met. The stopping rules are precisely defined and leave no room for subjective interpretation.

Validation spanned a wide array of empirical conditions, including 2 migration regimes. Comparisons with STRUCTURE, over a larger number of data sets showed that FLOCK was more reliable. FLOCK also tended to build clusters that matched the original samples better. It was also computationally more efficient.

The latest version of FLOCK offers several features to facilitate plateau analysis and the application of the estimation rules for *K*. FLOCK version 2.0 can be run for several values of *k* in a single batch process. For each value of *k*, plateaus are highlighted by printing in bold identical mean LLOD values. Also, a separate output file is automatically generated which reports the length of each plateau for each value of *k* (plateau record). Based on this file and the ad hoc rules (Appendix II), *K* is quickly estimated without any further computer processing. FLOCK v.2.0 for microsatellites or for AFLP markers can be downloaded from http://www.bio.ulaval.ca/no_cache/departement/professeurs/fiche_des_professeurs/professeur/11/13/. Because FLOCK is not model based, it can accept any type of categorical trait description, for example, pertaining to morphology, physiology, etc., and also combinations of various types of traits, genetic, and nongenetic. Categorical trait scores may easily be coded through dummy binary variables and then fed into FLOCK for AFLP.

Supplementary Material

Supplementary material can be found at http://www.jhered.oxfordjournals.org/.

Funding

Natural Sciences and Engineering Research Council Discovery (Grant 261853-07) to J.T.

Acknowledgments

The authors are grateful to E. Milot, G.J. Colbeck, A. Bourret, and C. Brouat for constructive comments on earlier versions of the manuscript, to Emily K. Latch for having kindly provided simulated data sets, and to 3 anonymous reviewers for helpful comments.

Appendix I: Description of the Iterated Reallocation Algorithm Implemented by FLOCK

Take a collection of genotypes and a number *k* of clusters (*k* required by the user). We describe iterated reallocation from the perspective of some generic genotype G.

First, the collection of genotypes is randomly partitioned into *k* clusters of (nearly) equal sizes. The clusters are identified as $(C_1^0, C_2^0, \ldots, C_k^0)$.

Then, G undergoes an “iteration” comprising the following steps:

1) G is withdrawn from its cluster.
2) The allelic frequencies are computed for each of the *k* clusters $(C_1^0, C_2^0, \ldots, C_k^0)$.
3) The likelihoods (a la Paetkau) of G relative to each cluster are computed.
4) G is allocated to the cluster with highest likelihood.

Once this is done for each G, a new partition $(C_1^1, C_2^1, \ldots, C_k^1)$ of *k* clusters has been generated.

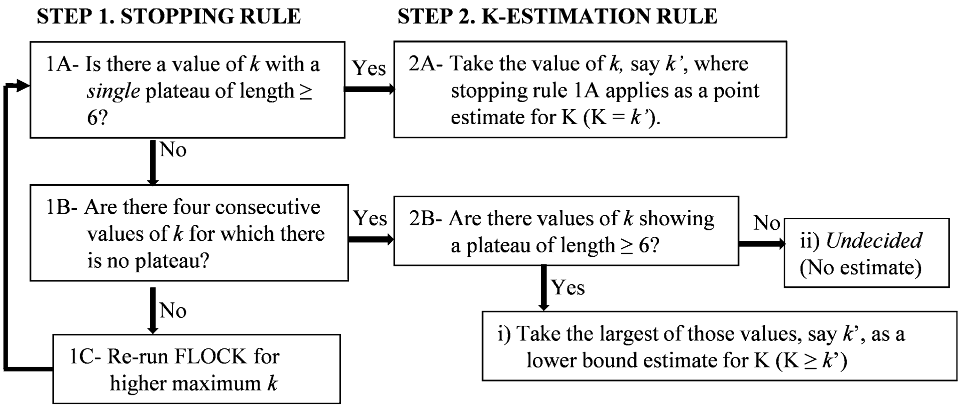
The next partition $(C_1^2, C_2^2, \ldots, C_k^2)$ is calculated by performing another iteration in the same fashion but now taking $(C_1^1, C_2^1, \ldots, C_k^1)$ as the reference partition to compute the likelihoods. And so on until as many iterations as requested by the user have been completed.

Notes:

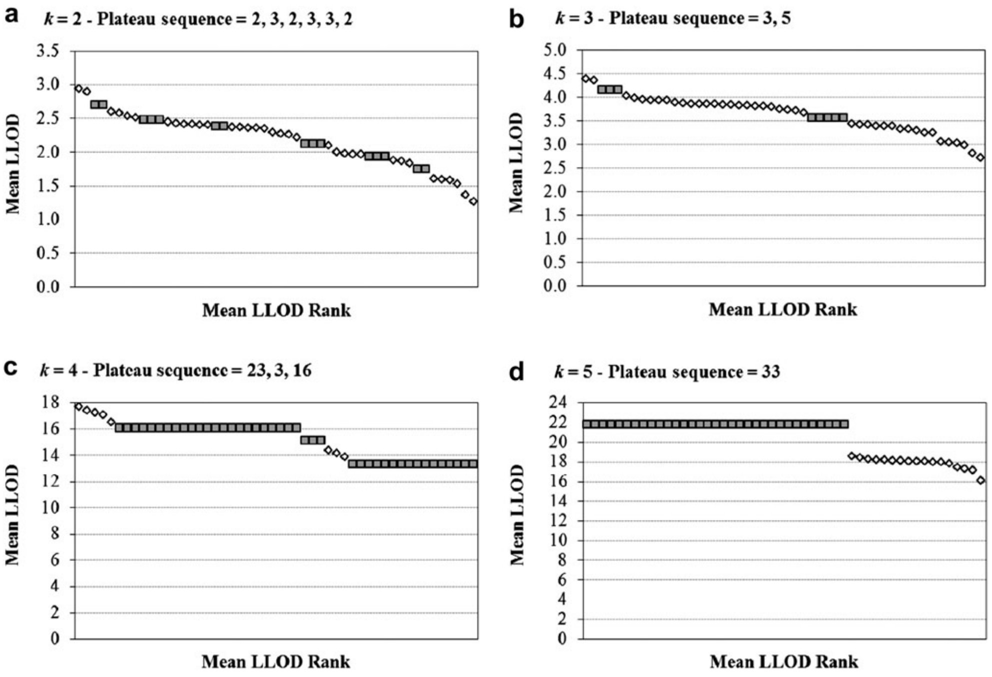
1) Apart from the initial random partition, this algorithm is entirely deterministic. In other words, starting with $(C_1^0, C_2^0, \ldots, C_k^0)$, the sequence of ensuing partitions will always be the same.
2) The number of iterations is 20 by default since by then convergence will have taken place in the vast majority of cases. Typically, the first step from the initial partition (partition 0) to the next is a great leap forward relative to the following steps, that is, it greatly reduces the distance between partition 0 and partition 20. The next step will also be the largest one compared with the remaining ones and so forth. As a result, convergence will nearly be reached within the first 5 iterations in most cases.
3) Each genotype G plays 2 roles at each iteration. It is allocated to the current partition (minus G) and, because it belongs to the partition, it is contributing to the reallocation of all other genotypes. So G is both an argument of the procedure (allocated) and a part of the current procedure through its membership to one of the clusters. Those allocated/allocating reciprocal functions actually define a positive feedback mechanism that largely explains the speed of convergence of iterated reallocation.

Appendix II: Estimating *K* with FLOCK

a) Decision flowchart to estimate *K* with FLOCK when performing 50 runs. Step 1 consists in reaching a stopping condition for the maximum *k* value to consider. Step 2 leads to a point estimate ($K = k'$, 2A) or a lower bound estimate ($K \geq k'$, 2B) for *K*.


STEP 1. STOPPING RULE
STEP 2. K-ESTIMATION RULE
1A- Is there a value of *k* with a *single* plateau of length ≥ 6?
Yes
2A- Take the value of *k*, say *k'*, where stopping rule 1A applies as a point estimate for K (K = *k'*).
No
1B- Are there four consecutive values of *k* for which there is no plateau?
Yes
2B- Are there values of *k* showing a plateau of length ≥ 6?
No
ii) *Undecided* (No estimate)
Yes
i) Take the largest of those values, say *k'*, as a lower bound estimate for K (K ≥ *k'*)
No
1C- Re-run FLOCK for higher maximum *k*

Example 1:			Example 2:		
Plateau Record:			Plateau Record:		
k	Plateau sequence	Figure	*k*	Plateau sequence	Figure
2	2,3,2,3,3,2	a	2	5,6,13	—
3	3,5	b	3	6,2,3	—
4	23,3,16	c	4	0	—
5	33	d	5	0	—
Decision flow: 1A→2A; $K = 5$			6	0	—
			7	0	—
			Decision flow: 1B→2B→i; $K \geq 3$		



b) Examples of plateau analysis to estimate *K*. In Example 1, the mean LLOD per run is plotted in decreasing rank order for the 50 runs performed for each value of *k* (2 to 5, figure a to d, respectively). Runs with identical mean LLOD values form plateaus and are indicated by square symbols. Plateau lengths for each *k* form the plateau record, and this record is used to follow the decision flowchart. In Example 2, stopping rule 1B applies, and a lower bound estimate of $K \geq 3$ is obtained.

References

Balloux F. 2001. EASYPOP (version 1.7). A computer program for the simulation of population genetics. J Hered. 92:301–302.

Brin M, Stuck G. 2002. Introduction to dynamical systems. Cambridge (UK): Cambridge University Press.

Chen C, Durand E, Forbes F, François O. 2007. Bayesian clustering algorithms ascertain spatial population structure: a new computer program and a comparison study. Mol Ecol Notes. 7:747–756.

Corander J, Marttinen P, Mäntyniemi S. 2005. A Bayesian method for identification of stock mixtures from molecular data. Fish Bull. 104:550–558.

Corander J, Siren J, Arjas E. 2008. Bayesian spatial modeling of genetic population structure. Comput Stat. 23:111–129.

Dawson KJ, Belkhir K. 2001. A Bayesian approach to the identification of panmictic populations and the assignment of individuals. Genet Res. 78:59–77.

Duchesne P, Turgeon J. 2009. FLOCK: a method for quick mapping of admixture without source samples. Mol Ecol Res. 9:1333–1344.

Evanno G, Regnaut S, Goudet J. 2005. Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. Mol Ecol. 14:2611–2620.

François O, Ancelet S, Guillot G. 2006. Bayesian clustering using hidden Markov random fields in spatial population genetics. Genetics. 174:805–816.

François O, Blum MGB, Jakobsson M, Rosenberg NA. 2008. Demographic history of European populations of *Arabidopsis thaliana*. PLoS Genet. 4:e1000075.

Frantz AC, Cellina S, Krier A, Schley L, Burke T. 2009. Using spatial Bayesian methods to determine the genetic structure of a continuously distributed population: clusters or isolation by distance? J Appl Ecol. 46:493–505.

Goudet J. 2002. FSTAT (v. 9.2.3.2): a computer program to calculate F-statistics. Lausanne (Switzerland): Institute of Ecology, University of Lausanne.

Guillot G. 2009. Response to comment 'On the inference of spatial structure from population genetics data.' Bioinformatics. 25:1805–1806.

Guillot G, Leblois R, Coulon A, Frantz AC. 2009. Statistical methods in spatial genetics. Mol Ecol. 18:4734–4756.

Guillot G, Mortier F, Estoup A. 2005. GENELAND: a computer package for landscape genetics. Mol Ecol Notes. 5:712–715.

Guillot G, Santos F, Estoup A. 2008. Analysing georeferenced population genetics data with GENELAND: a new algorithm to deal with null alleles and a friendly graphical user interface. Bioinformatics. 24:1406–1407.

Hubisz MJ, Falush D, Stephens M, Pritchard JK. 2009. Inferring weak population structure with the assistance of sample group information. Mol Ecol Res. 9:1322–1332.

Latch EK, Dharmarajan G, Glaubitz JC, Rhodes OE. 2006. Relative performance of Bayesian clustering softwares for inferring population substructure and individual assignment at low levels of differentiation. Conserv Genet. 7:295–302.

Paetkau D, Calvert W, Sterling I, Strobeck C. 1995. Microsatellite analysis of population structure in Canadian polar bears. Mol Ecol. 4:347–354.

Pritchard JK, Stephens M, Donnelly P. 2000. Inference of population structure using multilocus genotype data. Genetics. 155:945–959.

Rodríguez-Ramilo ST, Toro MA, Fernández J. 2009. Assessing population genetic structure via the maximisation of genetic distance. Genet Sel Evol. 41:49.

Waples RS, Gaggiotti O. 2006. What is a population? An empirical evaluation of some genetic methods for identifying the number of gene pools and their degree of connectivity. Mol Ecol. 15:1419–1439.

Received September 26, 2011; Revised April 13, 2012; Accepted April 26, 2012

Corresponding Editor: Robin Waples