Received: 10 September 2022 | Revised: 30 October 2022 | Accepted: 1 November 2022

DOI: 10.1111/1755-0998.13734

MOLECULAR ECOLOGY RESOURCES WILEY

RESOURCE ARTICLE

Crimp: An efficient tool for summarizing multiple clusterings in population structure analysis and beyond

Ulrich Lautenschlager

Evolutionary and Systematic Botany Group, Institute of Plant Sciences, University of Regensburg, Regensburg, Germany

Correspondence
Ulrich Lautenschlager, Evolutionary and Systematic Botany Group, Institute of Plant Sciences, University of Regensburg, Universitätsstr. 31, D-93053 Regensburg, Germany.
Email: ulrich.lautenschlager@ur.de

Funding information
Deutsche Forschungsgemeinschaft, Grant/Award Number: OB 155/13-1

Handling Editor: Kimberly J. Gilbert

Abstract

When a data set is repeatedly clustered using unsupervised techniques, the resulting clusterings, even if highly similar, may list their clusters in different orders. This so-called 'label-switching' phenomenon obscures meaningful differences between clusterings, complicating their comparison and summary. The problem often arises in the context of population structure analysis based on multilocus genotype data. In this field, a variety of popular tools apply model-based clustering, assigning individuals to a prespecified number of ancestral populations. Since such methods often involve stochastic components, it is a common practice to perform multiple replicate analyses based on the same input data and parameter settings. Available postprocessing tools allow to mitigate label switching, but leave room for improvements, in particular, regarding large input data sets. In this work, I present CRIMP, a lightweight command-line tool, which offers a relatively fast and scalable heuristic to align clusters across replicate clusterings consisting of the same number of clusters. For small problem sizes, an exact algorithm can be used as an alternative. Additional features include row-specific weights, input and output files similar to those of CLUMPP (Jakobsson & Rosenberg, 2007) and the evaluation of a given solution in terms of CLUMPP as well as its own objective functions. Benchmark analyses show that CRIMP, especially when applied to larger data sets, tends to outperform alternative tools considering runtime requirements and various quality measures. While primarily targeting population structure analysis, CRIMP can be used as a generic tool to correct multiple clusterings for label switching. This facilitates their comparison and allows to generate an averaged clustering. CRIMP's computational efficiency makes it even applicable to relatively large data sets while offering competitive solution quality.

KEYWORDS
cluster correspondence, cluster matching, cluster relabelling, label switching, population structure

1 | INTRODUCTION

Unsupervised cluster algorithms, which aim to identify relevant groups within a set of objects (e.g. individuals or sequences), are widely used in many areas of biological data analysis. A clustering is often represented as a matrix of membership coefficients, which quantify the degree (e.g. probability or proportion) to which each clustered object (row) is assigned to each cluster (column). Typically, the identified clusters do not have meaningful labels and, in the absence of ordering constraints, are listed in arbitrary order. Therefore, given a matrix of membership coefficients, each permutation of its columns represents an equivalent clustering. This ambiguity is a major reason for incongruence of membership matrices across multiple clusterings of the same data, commonly referred to as 'label-switching' phenomenon (Jakobsson & Rosenberg, 2007; Jasra et al., 2005; Stephens, 2000). Since many cluster algorithms involve stochastic components, even the results of repeated runs of the same algorithm, using identical parameter settings and input data, may be affected. In order to compare or summarize a number of similar clusterings, it can be useful to align corresponding clusters across multiple clusterings. The presented program, CRIMP (Cluster Relabelling based on IMPurity minimization), allows to rearrange a number of membership matrices of identical shapes in order to minimize differences caused by label switching. Ideally, the remaining differences should be attributable to either noise or truly different ways of grouping the data, sometimes referred to as 'genuine multimodality' (Jakobsson & Rosenberg, 2007).

An important application area of CRIMP is the analysis of population structure via model-based clustering as performed by STRUCTURE (Pritchard et al., 2000) and several related methods such as FRAPPE (Tang et al., 2005), ADMIXTURE (Alexander et al., 2009), FASTSTRUCTURE (Raj et al., 2014) and MAVERICK (Verity & Nichols, 2016). To account for the stochasticity inherent to most of such methods, it is a common practice to perform multiple replicate analyses. Several tools are available to compare or summarize the obtained clusterings. These, as well as the presented strategy, rely solely on the membership coefficients rather than other cluster-associated parameters. As the most widely used tool to mitigate label switching in this context, CLUMPP (Jakobsson & Rosenberg, 2007) aims to align multiple clusterings such that all pairwise similarities between membership coefficient matrices are maximized. Popular web servers for postprocessing population genetic clusterings allow to generate input files for CLUMPP (e.g. STRUCTURE HARVESTER; Earl & vonHoldt, 2012) or include it as part of their analysis pipeline (e.g. CLUMPAK; Kopelman et al., 2015). The latter ('CLUMPP across K') extends CLUMPP's functionality to allow a comparison of clusterings comprising different numbers of clusters and to detect distinct clustering modes (i.e. to cluster the clusterings themselves) using a Markov clustering algorithm (van Dongen, 2000). PONG (Behr et al., 2016), which provides functionality similar to CLUMPAK, implements a somewhat different approach: For each pair of membership matrices, it first infers an optimal one-to-one mapping between their corresponding columns. In the case of moderate problem sizes, this can be done using an exact optimization algorithm. The distances between pairwise aligned clusterings are then used to group clusterings into modes, subject to a prespecified distance threshold. Since, depending on the latter, there is only limited conflict among the clusterings belonging to the same mode, their clusters are simply ordered according to one randomly chosen representative clustering. This avoids directly optimizing an alignment of all clusterings, which would imply a much larger search space. Like CLUMPAK, PONG further allows to compare clusterings that differ in their number of clusters, assuming that decreasing the latter by one can be accommodated by merging two clusters while preserving the other ones. The R package POPHELPER (Francis, 2017), another toolkit from this category, uses either CLUMPP or a relabelling algorithm proposed by Stephens (2000) as implemented in the R package LABEL.SWITCHING (Papastamoulis, 2016) to align clusterings of identical size. The latter method minimizes the Kullback–Leibler divergence between individual membership matrices and an averaged membership matrix through an expectation maximization (EM)-like algorithm.

When applying CLUMPP to larger problem instances, only its greedy heuristics, often only the rougher LargeKGreedy algorithm, are practically feasible. Since these algorithms cannot revert suboptimal decisions when progressively building up a candidate solution, it may be difficult for them to find high-quality solutions – a problem that is aggravated by large problem sizes. The cluster matching facilities provided by PONG and POPHELPER, which both aim at a wider scope than CLUMPP, seem to be more scalable but also lack some of CLUMPP's functionality such as certain output options and the consideration of row-specific weights. The latter may be desirable, for instance, when clustering differently sized populations rather than individuals. The presented program, which superficially resembles CLUMPP's functionality, aims to fill this gap and, within this limited scope, provides even better performance and scalability to large problem sizes than the aforementioned tools. The objective functions used by CRIMP solely depend on impurity measures applied to rows of a single, averaged matrix of membership coefficients, which avoids a number of matrix–matrix comparisons that quadratically increases with the number of clusterings. CRIMP, which is a low-level implementation written in C, can be used as a standalone command-line tool. Like CLUMPP, it can be applied in combination with existing tools for summarizing or visualizing population structure analyses or also in different contexts.

2 | ALGORITHMS AND IMPLEMENTATION

The membership coefficients of a single clustering, comprising C objects and K clusters, can be written as $C \times K$ matrix, often referred to as Q-matrix. Since we deal with R replicate clusterings, let c_{ijk} ($i = 1, \ldots, C; j = 1, \ldots, K; k = 1, \ldots, R$) denote the membership coefficient which quantifies the degree to which the i^{th} object belongs to the j^{th} cluster, subject to the k^{th} clustering. Based on the individual membership coefficients, an averaged matrix (a_{ij}) can be computed with $a_{ij} = \frac{1}{R}\sum_{k=1}^{R} c_{ijk}$. The membership coefficients are allowed to take

values from the interval $[0, 1]$ and each row of a Q-matrix, whether individual or averaged, sums up to 1. In the following, we will effectively swap columns of the individual Q-matrices, thus these matrices and (a_{ij}) will vary depending on the current state of column ordering. In fact, this is accomplished using a list of column index permutations without changing the original matrices. To simplify the notation, however, I will stick with the above notion of swapping columns. Based on the intuition that an incorrect alignment of clusters leads to a homogenization of the averaged membership coefficients, the implemented objective functions are formally based on impurity measures, namely Shannon entropy and Gini impurity. In ecology, the latter measure, also known as Gini–Simpson index, can be interpreted as the probability of interspecific encounter (PIE; assuming sampling with replacement) when applied to species abundances (Hurlbert, 1971). Applied to allele frequencies at a given locus, it represents the expected heterozygosity under Hardy–Weinberg equilibrium (see e.g. Nei, 1973). Similar to CLUMPP, optional weights w_i allow to control the influence of specific objects (for each $i = 1, \ldots, C, w_i > = 0$). This can be useful, if, for example, the clustered objects are populations represented by a different number of sampled individuals. $W = \sum_{i=1}^{C} w_i$ denotes the sum of these weights.

One objective function is defined as the weighted mean of the row-wise Shannon entropy of (a_{ij}):

$$o_E = -\frac{1}{W}\sum_{i=1}^{C}\left(w_i \sum_{j=1}^{K}\left(a_{ij}\log\left(a_{ij}\right)\right)\right) \quad (1)$$

where $a_{ij}\log\left(a_{ij}\right) := 0$ if $a_{ij} = 0$.

Assuming equal weights w_i for each row index i, this objective function is linearly related to the total row-wise Kullback–Leibler divergence $\sum_{i=1}^{C}\sum_{j=1}^{K}\sum_{k=1}^{R}\left(c_{ijk}\log\frac{c_{ijk}}{a_{ij}}\right)$ between individual Q-matrices and the averaged matrix (a_{ij}) (see Supplementary Material). The latter quantity, which is often minimized using Stephens' cluster relabelling method, can be expressed as $D + CRo_E$, where $D = \sum_{i=1}^{C}\sum_{j=1}^{K}\sum_{k=1}^{R}\left(c_{ijk}\log\left(c_{ijk}\right)\right)$ and CR are invariant to column swapping. Therefore, minimizing o_E is equivalent to minimizing the total Kullback–Leibler divergence, yet Crimp differs from Stephens' method by using non-EM-like algorithms. A prototypical cluster relabelling tool based on this approach has been included in AllCoPol (Lautenschlager et al., 2020) for use in a very specific context. However, the formerly used algorithm and its naive Python implementation are several orders of magnitude slower than the presented program and only suitable for relatively small problem sizes.

By default, however, Crimp minimizes the mean row-wise Gini impurity:

$$o_G = \frac{1}{W}\sum_{i=1}^{C}\left(w_i\left(1 - \sum_{j=1}^{K} a_{ij}^2\right)\right) \quad (2)$$

A practical advantage of using the Gini impurity over the Shannon entropy is that the former does not involve logarithms and therefore allows faster computation. o_G can be interpreted in various ways (see Supplementary Material for derivations). Assuming equal weights w_i for simplicity, it can be expressed as $o_G = 1 - \frac{1}{K} - \frac{1}{C}\sum_{i=1}^{C}\sum_{j=1}^{K}\left(a_{ij} - \frac{1}{K}\right)^2$, which can be seen as a linear function of the variance of the averaged membership coefficients. Therefore, minimizing o_G is equivalent to maximizing the variance of the averaged coefficients or, equally, minimizing the variance of matched individual coefficients, which follows from the law of total variance. For this interpretation, an alignment of all R Q-matrices can be viewed as a clustering of all CKR individual membership coefficients into CK clusters of size R each. In case of equal weights w_i, equation (2) can further be expressed in terms of pairwise matrix comparison as

$$o_G = A + B\sum_{i=1}^{C}\sum_{j=1}^{K}\sum_{k=1}^{R}\sum_{l=1}^{R}\left(c_{ijk} - c_{ijl}\right)^2 \quad (3)$$

where $A = 1 - \frac{1}{K} - \frac{1}{RC}\sum_{i=1}^{C}\sum_{j=1}^{K}\sum_{k=1}^{R}\left(c_{ijk} - \frac{1}{K}\right)^2$ and $B = \frac{1}{2R^2C}$ are invariant to column reordering. In other words, minimizing o_G minimizes the squared deviations between matched membership coefficients across all pairs of Q-matrices.

Two neighbourhood-based optimization algorithms are implemented, both of which proceed by successively swapping two columns of one coefficient matrix at a time. Using auxiliary arrays to store intermediate results enables an evaluation of candidate solutions in $\mathcal{O}(C)$ time because only small parts of the objective function have to be updated from solution to solution. In contrast, a naive evaluation would require $\mathcal{O}(CKR)$ runtime per candidate solution. To avoid an accumulation of rounding errors in the course of incremental updates, the membership coefficients are internally stored using integers, providing a precision of 6 decimal places. As in CLUMPP, raw membership coefficients read from the input are normalized such that each row of a Q-matrix sums up to 1.

By default, a random restart hill-climbing heuristic is used to minimize the objective function: Starting from random column permutations, it repeatedly evaluates all possible swaps of two columns within each Q-matrix. To avoid any bias caused by the order of columns and clusterings in the input file, the order of tested swaps is randomized using a lightweight xoshiro128** (Blackman & Vigna, 2021) pseudorandom number generator. Since it would likely be inefficient to evaluate the complete swap neighbourhood of a given solution (i.e. $\frac{K(K-1)R}{2}$ candidate solutions) before selecting the next one, each improving swap is instantly accepted. The search terminates if the current solution cannot be improved by any single swap of two columns. To mitigate its susceptibility to plateaus and local optima, this hill-climbing procedure is repeated for a predefined number of runs.

As an alternative, an exhaustive search algorithm is implemented, exploiting the fact that all permutations of a set can be generated in ways such that two consecutive permutations differ only by an interchange of two elements (for an overview of the so-called bell-ringing algorithms, see Knuth, 2014). This principle can be utilized to generate all $(K!)^{R-1}$ clustering alignments (i.e. combinations of column index permutations) such that only one swap must be evaluated for each possible solution. In contrast to the presented heuristic, this approach guarantees finding a global optimum. However, because the number of solutions to be evaluated

quickly becomes prohibitive, it is only applicable to small problem sizes.

Eventually, different output files are written for the best solution found: a list of column index permutations, the matrix of averaged membership coefficients and, optionally, the aligned individual Q-matrices. For a given solution, CRIMP also allows to calculate CLUMPP's similarity scores H and H′ although these cannot be optimized directly.

3 | BENCHMARK ANALYSES

To compare the performance of different tools capable of aligning STRUCTURE-like Q-matrices, CRIMP v1.1.0 along with CLUMPP v1.1.2, PONG v1.4.7 and POPHELPER v2.3.0 was applied to four exemplary data sets. As biological examples, the small 'arabid' ($K = 3$, $C = 95$, $R = 9$) and the larger 'chicken' ($K = 19$, $C = 600$, $R = 100$; Rosenberg et al., 2001) data sets, both distributed with CLUMPP, were analysed. In addition, the two simulated data sets 'largeK' ($K = 100, C = 1000, R = 100$) and 'largeR' ($K = 25, C = 250, R = 1000$) were used to demonstrate CRIMP's scalability to high values of K and R, respectively. Those are based on repeated runs of K-means clustering applied to structured random data (see kmeans_largeK.r and kmeans_largeR.r for details). As opposed to the biological data sets, the simulated data sets consist of binary Q-matrices, representing hard clusterings (partitions).

Because of differences regarding scope and output options, a fair comparison of CRIMP and CLUMPP with PONG and POPHELPER is difficult to achieve and the obtained results depend on the following decisions. Each tested program was configured to output the optimized permutations of column indices while other output was suppressed where possible. While POPHELPER's alignK() function allows rearranging Q-matrices, the applied permutations of column indices are not accessible. On the other hand, writing the aligned Q-matrices to disk using POPHELPERS's clumppExport() function turned out to be very slow and may not be part of a typical POPHELPER workflow. To circumvent these problems, POPHELPER was only used to read the input matrices, and the stephens() function provided by the LABEL.SWITCHING package was then manually called with default options, as it is internally done by POPHELPER (see pophelper.r). Eventually, the optimized permutations were written using the R function write.table(). In the following, the described workflow will be referred to as POPHELPER/LABEL.SWITCHING. In the case of PONG, the input matrices are partitioned into different clustering modes and separate column index permutations are reported for each mode. To enforce a single mode comprising all Q-matrices, its similarity threshold parameter was set to 0. To achieve a behaviour similar to that of CLUMPP, PONG was configured to use the G distance for cluster comparison. In addition, PONG's default metric, the Jaccard Index, was used.

Each analysis was run 25 times and, for each replicate, the order of clusters and clusterings in the input was permuted. It was ensured that the measured runtimes were not distorted by memory swapping and analyses were interrupted after a maximum of 3600 seconds for the largeK data set and 1800 seconds for the other data sets. To evaluate the obtained column index permutations, the input matrices were rearranged using an external script, thus mitigating different rounding behaviour of the tools to be compared. CRIMP, which can be utilized to evaluate its input without further optimization, was then used to calculate CLUMPP's similarity scores H and H′ as well as its own cost functions o_G and o_E. Since none of the tested tools exhibited noteworthy parallelization, runtime was measured as elapsed real time. Peak memory consumption was measured as the maximum resident set size. All analyses were serially performed on a Dell Optiplex 7010 desktop PC with an Intel i5-3470 CPU and 12 GB RAM. Averaged results for the chicken, largeK and largeR data sets are shown in Table 1.

For the small arabid data set, only CLUMPP and CRIMP were used because they are able to account for the differently sized populations. Apart from one outlier run when performing only one iteration of CRIMP's heuristic using o_E, both programs consistently found the common global optimum of H, H′, o_G and o_E, independently of the used algorithm and the optimized objective function. Runtimes ranged from less than 1 millisecond to about 1 min (see Table S1). While, in this context, CRIMP's speed advantages are of little practical relevance, using o_G, its exhaustive search is about two orders of magnitude faster than that of CLUMPP.

In the case of the larger chicken data set, for which either CLUMPP's and CRIMP's exhaustive search is infeasible, each of the four evaluated scores revealed a similar pattern. CRIMP using o_G is the fastest tool and its solutions are consistently among the best. Interestingly, optimizing o_E directly is not only slower due to computationally more expensive solution evaluation, but also seems to be more susceptible to local optima or plateaus as it requires a higher number of hill-climbing runs to reliably find high-quality solutions. Within the allowed runtime, both PONG and POPHELPER/LABEL.SWITCHING are able to find solutions superior to those of CLUMPP, but less optimal than those of CRIMP. As expected, POPHELPER/LABEL.SWITCHING performs best in terms of the mean entropy, but even from this perspective, CRIMP using o_G yields better scores in a shorter time. In contrast, CLUMPP's LargeKGreedy algorithm leads to considerably worse scores, at least within the allowed number of at most 100 greedily constructed solutions. For 30,000 greedily constructed solutions, amounting to a runtime of approx. 47 h in our setting, Jakobsson and Rosenberg (2007) report a similarity score H = 0.5546, which is similar to PONG and POPHELPER/LABEL.SWITCHING, but still noticeably lower than the values obtained by CRIMP.

Applied to the large simulated data sets, PONG did not finish within the imposed runtime limits. In the case of the largeR data set, CRIMP using o_G was again the fastest option while yielding the best scores, followed by POPHELPER/LABEL.SWITCHING and CRIMP using o_E. Similar behaviour was observed for the largeK data set, where minimizing o_G with CRIMP takes 20 s to obtain $o_E = 0.88$ on average, whereas POPHELPER/LABEL.SWITCHING, although directly minimizing the latter score, remains at 1.07 after 1459 s. For both data

TABLE 1 Benchmark results (mean and standard deviation) for the chicken, largeR and largeK data sets. Here, only methods for which all 25 replicate runs finished within a maximum of 1800 s (chicken, largeR) or 3600 s (largeK) are listed

Data set	Method	o_E ↓	o_G ↓	H ↑	H′ ↑	Memory [kB]	Runtime [s]
chicken	CRIMP (o_G, heuristic, 1 it.)	0.6868 ± 0.0032	0.1820 ± 0.0015	0.6039 ± 0.0043	0.7432 ± 0.0028	7316 ± 42	0.4 ± 0.0
	CRIMP (o_G, heuristic, 5 it.)	0.6838 ± 0.0023	0.1806 ± 0.0010	0.6076 ± 0.0028	0.7456 ± 0.0018	7275 ± 75	1.0 ± 0.0
	CRIMP (o_G, heuristic, 20 it.)	0.6825 ± 0.0001	**0.1800 ± 0.0000**	0.6093 ± 0.0000	**0.7468 ± 0.0000**	7293 ± 72	3.5 ± 0.1
	CRIMP (o_G, heuristic, 100 it.)	0.6825 ± 0.0000	**0.1800 ± 0.0000**	**0.6094 ± 0.0000**	**0.7468 ± 0.0000**	7313 ± 53	16.7 ± 0.2
	CRIMP (o_E, heuristic, 1 it.)	0.7492 ± 0.0618	0.2067 ± 0.0237	0.5438 ± 0.0560	0.7042 ± 0.0363	7282 ± 86	3.6 ± 0.9
	CRIMP (o_E, heuristic, 5 it.)	0.6909 ± 0.0080	0.1842 ± 0.0030	0.5983 ± 0.0076	0.7396 ± 0.0049	7280 ± 82	16.1 ± 1.6
	CRIMP (o_E, heuristic, 20 it.)	0.6840 ± 0.0026	0.1813 ± 0.0014	0.6059 ± 0.0036	0.7445 ± 0.0024	7305 ± 54	64.8 ± 6.9
	CRIMP (o_E, heuristic, 100 it.)	**0.6820 ± 0.0010**	0.1803 ± 0.0006	0.6085 ± 0.0017	0.7462 ± 0.0011	7299 ± 61	317.0 ± 10.4
	CLUMPP (H, LargeKGreedy, 1 it.)	0.8869 ± 0.0512	0.2474 ± 0.0218	0.4626 ± 0.0423	0.6516 ± 0.0274	42,507 ± 58	6.6 ± 0.0
	CLUMPP (H, LargeKGreedy, 5 it.)	0.8449 ± 0.0294	0.2297 ± 0.0108	0.4971 ± 0.0208	0.6740 ± 0.0135	42,494 ± 54	29.0 ± 0.0
	CLUMPP (H, LargeKGreedy, 20 it.)	0.8203 ± 0.0256	0.2201 ± 0.0083	0.5172 ± 0.0160	0.6870 ± 0.0104	42,500 ± 46	113.2 ± 0.1
	CLUMPP (H, LargeKGreedy, 100 it.)	0.7927 ± 0.0113	0.2117 ± 0.0039	0.5334 ± 0.0078	0.6975 ± 0.0051	42,493 ± 56	562.4 ± 0.1
	CLUMPP (H′, LargeKGreedy, 1 it.)	0.9185 ± 0.0545	0.2598 ± 0.0237	0.4404 ± 0.0430	0.6373 ± 0.0279	42,482 ± 56	6.6 ± 0.0
	CLUMPP (H′, LargeKGreedy, 5 it.)	0.8358 ± 0.0337	0.2275 ± 0.0119	0.5024 ± 0.0218	0.6774 ± 0.0141	42,509 ± 53	29.0 ± 0.0
	CLUMPP (H′, LargeKGreedy, 20 it.)	0.8031 ± 0.0198	0.2152 ± 0.0074	0.5260 ± 0.0141	0.6928 ± 0.0092	42,497 ± 58	113.2 ± 0.1
	CLUMPP (H′, LargeKGreedy, 100 it.)	0.7954 ± 0.0131	0.2116 ± 0.0037	0.5335 ± 0.0076	0.6976 ± 0.0049	42,501 ± 60	562.4 ± 0.1
	PONG (G)	0.7507 ± 0.0397	0.1982 ± 0.0119	0.5634 ± 0.0289	0.7170 ± 0.0188	794,319 ± 316	52.4 ± 0.8
	PONG (Jaccard)	0.7589 ± 0.0457	0.2012 ± 0.0140	0.5551 ± 0.0343	0.7116 ± 0.0223	794,528 ± 286	321.4 ± 7.6
	POPHELPER/LABEL. SWITCHING	0.7325 ± 0.0704	0.2039 ± 0.0317	0.5578 ± 0.0620	0.7134 ± 0.0402	179,113 ± 13,901	9.4 ± 1.1
largeR	CRIMP (o_G, heuristic, 1 it.)	1.0352 ± 0.0139	0.3584 ± 0.0036	0.1410 ± 0.0044	0.4049 ± 0.0030	28,864 ± 63	5 ± 1
	CRIMP (o_G, heuristic, 5 it.)	1.0227 ± 0.0079	0.3550 ± 0.0018	0.1451 ± 0.0021	0.4077 ± 0.0015	28,864 ± 63	24 ± 3
	CRIMP (o_G, heuristic, 20 it.)	1.0195 ± 0.0063	0.3528 ± 0.0009	0.1477 ± 0.0011	0.4095 ± 0.0008	28,868 ± 61	97 ± 4
	CRIMP (o_G, heuristic, 100 it.)	**1.0185 ± 0.0053**	**0.3520 ± 0.0004**	**0.1487 ± 0.0005**	**0.4102 ± 0.0004**	28,880 ± 38	490 ± 11
	CRIMP (o_E, heuristic, 1 it.)	1.1816 ± 0.0501	0.4124 ± 0.0181	0.0780 ± 0.0206	0.3612 ± 0.0142	28,885 ± 32	28 ± 4
	CRIMP (o_E, heuristic, 5 it.)	1.1333 ± 0.0360	0.3957 ± 0.0128	0.0971 ± 0.0147	0.3745 ± 0.0102	28,869 ± 54	143 ± 10
	CRIMP (o_E, heuristic, 20 it.)	1.0980 ± 0.0209	0.3827 ± 0.0074	0.1122 ± 0.0087	0.3849 ± 0.0060	28,895 ± 36	571 ± 22
	CLUMPP (H, LargeKGreedy, 1 it.)	1.4782 ± 0.0462	0.4633 ± 0.0153	0.0227 ± 0.0162	0.3229 ± 0.0112	214,294 ± 66	428 ± 0
	CLUMPP (H′, LargeKGreedy, 1 it.)	1.4646 ± 0.0432	0.4582 ± 0.0148	0.0280 ± 0.0156	0.3266 ± 0.0108	214,269 ± 68	429 ± 0
	POPHELPER/LABEL. SWITCHING	1.0709 ± 0.0332	0.3768 ± 0.0128	0.1198 ± 0.0144	0.3901 ± 0.0100	304,898 ± 334	348 ± 113

(Continues)

TABLE 1 (Continued)

Data set	Method	$o_E \downarrow$	$o_G \downarrow$	H ↑	H′ ↑	Memory [kB]	Runtime [s]
largeK	CRIMP (o_G, heuristic, 1 it.)	0.8802 ± 0.0058	0.3150 ± 0.0019	0.1999 ± 0.0024	0.4371 ± 0.0017	42,500 ± 53	20 ± 4
	CRIMP (o_G, heuristic, 5 it.)	0.8729 ± 0.0049	0.3126 ± 0.0010	0.2029 ± 0.0013	0.4392 ± 0.0009	42,500 ± 51	95 ± 8
	CRIMP (o_G, heuristic, 20 it.)	0.8709 ± 0.0044	0.3119 ± 0.0008	0.2038 ± 0.0010	0.4398 ± 0.0007	42,519 ± 43	391 ± 17
	CRIMP (o_G, heuristic, 100 it.)	**0.8668 ± 0.0038**	**0.3106 ± 0.0006**	**0.2055 ± 0.0008**	**0.4410 ± 0.0005**	42,510 ± 49	1945 ± 35
	CRIMP (o_E, heuristic, 1 it.)	1.0834 ± 0.0324	0.3890 ± 0.0115	0.1105 ± 0.0132	0.3741 ± 0.0093	42,519 ± 49	77 ± 13
	CRIMP (o_E, heuristic, 5 it.)	1.0279 ± 0.0205	0.3704 ± 0.0076	0.1321 ± 0.0089	0.3893 ± 0.0063	42,499 ± 52	387 ± 33
	CRIMP (o_E, heuristic, 20 it.)	1.0087 ± 0.0131	0.3630 ± 0.0050	0.1407 ± 0.0059	0.3955 ± 0.0042	42,492 ± 54	1564 ± 50
	CLUMPP (H, LargeKGreedy, 1 it.)	1.4180 ± 0.0390	0.4349 ± 0.0121	0.0599 ± 0.0130	0.3386 ± 0.0091	323,705 ± 69	258 ± 0
	CLUMPP (H, LargeKGreedy, 5 it.)	1.3933 ± 0.0253	0.4277 ± 0.0071	0.0678 ± 0.0077	0.3441 ± 0.0054	323,715 ± 59	1268 ± 1
	CLUMPP (H′, LargeKGreedy, 1 it.)	1.4261 ± 0.0283	0.4380 ± 0.0089	0.0565 ± 0.0096	0.3362 ± 0.0068	323,682 ± 71	258 ± 0
	CLUMPP (H′, LargeKGreedy, 5 it.)	1.3894 ± 0.0220	0.4245 ± 0.0055	0.0713 ± 0.0060	0.3466 ± 0.0042	323,707 ± 68	1270 ± 0
	POPHELPER/LABEL. SWITCHING	1.0723 ± 0.0172	0.3856 ± 0.0071	0.1148 ± 0.0081	0.3772 ± 0.0057	526,858 ± 1606	1459 ± 343

As indicated by the arrows, o_E and o_G are to be minimized, whereas H and H′ are to be maximized. For each data set and quality criterion, the best average results obtained are highlighted in bold.

sets, CLUMPP 's LargeKGreedy algorithm yields the least optimal scores.

Whichever of the four benchmark data sets is considered, on average, even a single run of CRIMP's heuristic using o_G (i.e. the fastest configuration), leads to results equal to or better than all tested alternative tools. Compared to the latter, its results also tend to be more consistent as indicated by relatively low standard deviations, in particular, when minimizing o_G. Besides, CRIMP, followed by CLUMPP, shows the lowest memory demands.

4 | DISCUSSION

In the case of the data sets used for benchmarking, all considered quality measures (H, H′, o_G, and o_E) are strongly correlated. While Behr et al. (2016) recommend using the Jaccard index for pairwise comparisons instead, this comes at the cost of increased runtime requirements and an arbitrary threshold parameter, currently not exposed to the user. It should also be noted that enforcing a single mode in PONG may be problematic because it is focused on aligning pairs of Q-matrices rather than reconciling multiple matrices simultaneously.

As demonstrated, especially when applied to larger data sets, CRIMP tends to outperform alternative tools in terms of runtime, solution quality and consistency. Its advantage in speed may become even more relevant if executed multiple times, for instance, in the course of CLUMPAK-like analyses. CRIMP's default objective function based on the Gini impurity, which emulates pairwise matrix comparison, not only allows faster computation than the mean Shannon entropy but also seems to lead to a better convergence behaviour of the implemented heuristic. However, the latter advantage might be problem specific.

Especially in the context of repeated K-means clustering, it may be tempting to utilize the averaged membership matrix as a kind of consensus clustering (for an introduction to the latter, see Strehl & Ghosh, 2002). It should be noted that, for such use, additional post-processing steps may be desirable, such as merging similar clusters or removing more or less empty ones if present. For the analysis of population structure, *K*-values approaching 100 or above may appear uncommon at first glance. However, this magnitude is not unrealistic, for instance, when analysing domestic animal breeds (e.g. Funk et al., 2020; Leroy et al., 2009; Papachristou et al., 2020).

Unlike CLUMPAK, PONG and POPHELPER, CRIMP is restricted to clusterings comprising the same number of clusters and does not provide additional functionality for mode detection and visualization. Nevertheless, its application helps to recognize real differences between clusterings, whether these may be regarded as noise or genuine multimodality. Moreover, CRIMP can easily be integrated into common, CLUMPP-based workflows as a much faster and often more accurate alternative, applicable even to use cases that could not be handled satisfactorily before. Its heuristic can be controlled via a single, intuitive parameter, namely the number of hill-climbing runs to be performed.

As opposed to the implementation of Stephens' cluster relabelling by the LABEL.SWITCHING package, CRIMP's performance does not depend on external solvers but is based on efficient solution evaluation. As a consequence, it comes as a lightweight tool without dependencies beyond the C standard library. Future work may be dedicated to extended functionality or interfacing with other tools and programming languages.

AUTHOR CONTRIBUTIONS

UL conceived the methodology, implemented the software, carried out the analyses and wrote the manuscript.

ACKNOWLEDGEMENTS

I would like to thank Elmar Lang for the helpful discussion, Christoph Oberprieler for supporting this work and Tankred Ott for his comments on the manuscript. I also thank the anonymous reviewers who provided feedback on this work. Open Access funding enabled and organized by Projekt DEAL.

FUNDING INFORMATION

This work has been partially supported by a Grant (OB 155/13–1) of the German Research Foundation (DFG) in the frame of the Priority Programme SPP 1991 'Taxon-omics – New Approaches for Discovering and Naming Biodiversity' to Christoph Oberprieler.

CONFLICT OF INTEREST

The author declares that he has no competing interests.

DATA AVAILABILITY STATEMENT

CRIMP's source code along with precompiled binaries for Linux and Windows, usage guidelines, benchmark code and the largeK and largeR data sets are freely available at https://github.com/ulilautenschlager/crimp and archived at Zenodo (Lautenschlager, 2022). The arabid and chicken data sets are available as part of the CLUMPP software package (Jakobsson & Rosenberg, 2007).

ORCID

Ulrich Lautenschlager https://orcid.org/0000-0003-1886-2277

REFERENCES

Alexander, D. H., Novembre, J., & Lange, K. (2009). Fast model-based estimation of ancestry in unrelated individuals. *Genome Research*, *19*(9), 1655–1664.

Behr, A. A., Liu, K. Z., Liu-Fang, G., Nakka, P., & Ramachandran, S. (2016). pong: Fast analysis and visualization of latent clusters in population genetic data. *Bioinformatics*, *32*(18), 2817–2823.

Blackman, D., & Vigna, S. (2021). Scrambled linear pseudorandom number generators. *ACM Transactions on Mathematical Software*, *47*(4), 36:1–36:32.

Earl, D. A., & vonHoldt, B. M. (2012). STRUCTURE HARVESTER: A website and program for visualizing STRUCTURE output and implementing the Evanno method. *Conservation Genetics Resources*, *4*(2), 359–361.

Francis, R. M. (2017). pophelper: An R package and web app to analyse and visualize population structure. *Molecular Ecology Resources*, *17*(1), 27–32.

Funk, S. M., Guedaoura, S., Juras, R., Raziq, A., Landolsi, F., Luís, C., Martínez, A. M., Musa Mayaki, A., Mujica, F., Oom, M. D. M., Ouragh, L., Stranger, Y. M., Vega-Pla, J. L., & Cothran, E. G. (2020). Major inconsistencies of inferred population genetic structure estimated in a large set of domestic horse breeds using microsatellites. *Ecology and Evolution*, *10*(10), 4261–4279.

Hurlbert, S. H. (1971). The nonconcept of species diversity: A critique and alternative parameters. *Ecology*, *52*(4), 577–586.

Jakobsson, M., & Rosenberg, N. A. (2007). CLUMPP: A cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. *Bioinformatics*, *23*(14), 1801–1806.

Jasra, A., Holmes, C. C., & Stephens, D. A. (2005). Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statistical Science*, *20*(1), 50–67.

Knuth, D. E. (2014). *The art of computer programming, volume 4A: Combinatorial algorithms, part 1*. Addison-Wesley Professional.

Kopelman, N. M., Mayzel, J., Jakobsson, M., Rosenberg, N. A., & Mayrose, I. (2015). Clumpak: A program for identifying clustering modes and packaging population structure inferences across k. *Molecular Ecology Resources*, *15*(5), 1179–1191.

Lautenschlager, U. (2022). *ulilautenschlager/crimp: Crimp v1.1.0*. Zenodo.

Lautenschlager, U., Wagner, F., & Oberprieler, C. (2020). AllCoPol: Inferring allele co-ancestry in polyploids. *BMC Bioinformatics*, *21*(1), 441.

Leroy, G., Verrier, E., Meriaux, J. C., & Rognon, X. (2009). Genetic diversity of dog breeds: Between-breed diversity, breed assignation and conservation approaches. *Animal Genetics*, *40*(3), 333–343.

Nei, M. (1973). Analysis of gene diversity in subdivided populations. *Proceedings of the National Academy of Sciences*, *70*(12), 3321–3323.

Papachristou, D., Koutsouli, P., Laliotis, G. P., Kunz, E., Upadhyay, M., Seichter, D., & Medugorac, I. (2020). Genomic diversity and population structure of the indigenous Greek and Cypriot cattle populations. *Genetics Selection Evolution*, *52*(1), 43.

Papastamoulis, P. (2016). label.switching: An R package for dealing with the label switching problem in MCMC outputs. *Journal of Statistical Software*, *69*(1), 1–24.

Pritchard, J. K., Stephens, M., & Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, *155*(2), 945–959.

Raj, A., Stephens, M., & Pritchard, J. K. (2014). fastSTRUCTURE: Variational inference of population structure in large SNP data sets. *Genetics*, *197*(2), 573–589.

Rosenberg, N. A., Burke, T., Elo, K., Feldman, M. W., Freidlin, P. J., Groenen, M. A. M., Hillel, J., Mäki-Tanila, A., Tixier-Boichard, M., Vignal, A., Wimmers, K., & Weigend, S. (2001). Empirical evaluation of genetic clustering methods using multilocus genotypes from 20 chicken breeds. *Genetics*, *159*(2), 699–713.

Stephens, M. (2000). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *62*(4), 795–809.

Strehl, A., & Ghosh, J. (2002). cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*, 583–617.

Tang, H., Peng, J., Wang, P., & Risch, N. J. (2005). Estimation of individual admixture: Analytical and study design considerations. *Genetic Epidemiology*, *28*(4), 289–301.

van Dongen, S. (2000). Graph clustering by flow simulation (PhD thesis, University of Utrecht).

Verity, R., & Nichols, R. A. (2016). Estimating the number of subpopulations (k) in structured populations. *Genetics*, *203*(4), 1827–1839.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Lautenschlager, U. (2023). Crimp: An efficient tool for summarizing multiple clusterings in population structure analysis and beyond. *Molecular Ecology Resources*, *23*, 705–711. https://doi.org/10.1111/1755-0998.13734