Sequence analysis

HomologMiner: looking for homologous genomic groups in whole genomes

Minmei Hou*, Piotr Berman, Chih-Hao Hsu and Robert S. Harris
Department of Computer Science & Engineering, Penn State University, PA, USA

Received on November 21, 2006; revised on January 22, 2007; accepted on February 6, 2007
Advance Access publication February 18, 2007

ABSTRACT
Motivation: Complex genomes contain numerous repeated sequences, and genomic duplication is believed to be a main evolutionary mechanism to obtain new functions. Several tools are available for de novo repeat sequence identification, and many approaches exist for clustering homologous protein sequences. We present an efficient new approach to identify and cluster homologous DNA sequences with high accuracy at the level of whole genomes, excluding low-complexity repeats, tandem repeats and annotated interspersed repeats. We also determine the boundaries of each group member so that it closely represents a biological unit, e.g. a complete gene, or a partial gene coding a protein domain.

Results: We developed a program called HomologMiner to identify homologous groups applicable to genome sequences that have been properly marked for low-complexity repeats and annotated interspersed repeats. We applied it to the whole genomes of human (hg17), macaque (rheMac2) and mouse (mm8). Groups obtained include gene families (e.g. olfactory receptor gene family, zinc finger families), unannotated interspersed repeats and additional homologous groups that resulted from recent segmental duplications. Our program incorporates several new methods: a new abstract definition of consistent duplicate units, a new criterion to remove moderately frequent tandem repeats, and new algorithmic techniques. We also provide preliminary analysis of the output on the three genomes mentioned above, and show several applications including identifying boundaries of tandem gene clusters and novel interspersed repeat families.

Availability: All programs and datasets are downloadable from www.bx.psu.edu/miller_lab
Contact: mhou@cse.psu.edu

1 INTRODUCTION

A main obstacle for DNA search methods is that on one hand a large proportion of complex genomes are composed of highly repetitive sequences called repeats. Repeats are divided into two types based on their patterns of dispersion: tandem repeats and interspersed repeats. These highly repetitive sequences are less likely to contain functional elements. On the other hand, certain functional regions, including gene clusters, are also of a repetitive nature; they are derived from duplication events, and we call them duplicates. We are more interested in the latter type, especially gene families. A gene family or subfamily may form a cluster. When such a cluster has the form of a tandem array, its members are called tandem duplications. Figure 1 shows examples of tandem repeats and tandem duplications. Our goal is to exclude tandem repeats and extract homologous gene groups. Some of the identified groups are unannotated interspersed repeats. Additional post-processing can use annotations, dispersion patterns, etc. to differentiate interspersed repeats from other groups.

Several tools exist to detect repetitive sequences in a whole genome. They include RepeatMasker (Smit et al., 1996–2004), RECON (Bao et al., 2002), RepeatGluer (Pevzner et al., 2004), RepeatScout (Price et al., 2005) and PILER (Edgar et al., 2005). Besides masking low-complexity DNA sequences, RepeatMasker uses an extensive library of known repeats to search for interspersed repeats, requiring a separate library for each genome. The others use a variety of de novo methods; each has its own limitations and advantages. RepeatScout uses high frequency seeds, and thus is less likely to find lower frequency repetitive sequences, e.g. moderate and small gene families. PILER aims for high specificity and thus obtains low sensitivity. One drawback of RECON is its low speed. Bejerano et al. (2002) looks for clusters specifically in non-coding regions. Many of these articles mention the problem of identifying the correct boundaries for each repeat copy. This problem is even more severe for us since, in alignments, diverged genes tend to have less consistent boundaries than repeats. Besides the above tools which look for DNA repeat sequences, other tools cluster protein sequences to classify protein families (Cameron et al., 2006; Enright et al., 2002; Kim et al., 2006; Altschut et al., 1997; Kawaji et al., 2004). Many of them tried to solve a problem caused by the presence of multiple domains in a protein: the same domain exists in many families and thus joins different families into a larger family. We face a similar problem, called alignment drifting, caused by adjacent regions (details in Section 3.2.1 and Bejerano et al., 2002). In Bejerano’s method, a graph representing similarities among genomic regions is recursively split until the sub-graphs are sufficiently dense. We apply a different approach aiming to group all homologs together in a consistent manner, as we not
only split, but also merge to undo improper splits. Our method must be very efficient to process whole genomes. There is also much research on segmental duplications. In some studies, only duplicates of sequence identity >90% and length >10 K bases are considered as segmental duplications (Bailey et al., 2004). We want to consider more diverged and shorter duplicates.

Duplication is believed to be a main evolutionary mechanism to create new functionalities. Identification of homologous groups provides an opportunity for function inference if some of the group members happen to have known attributes. Clustering methods based on protein sequences have limitations, because the availability of protein sequences depends on the gene expression and annotation extent for a particular genome. Moreover, clustering on the basis of a protein database may be misled by redundant entries (Cameron et al., 2006; Park et al., 2000). Whole-genome DNA clustering complements existing methods of finding families of genes. Genes sharing high similarity might share similar functionalities. Thus establishing homologous groups can speed up genome annotation. Even if sequences with high similarity have different functionalities, homologous grouping provides resources for gene phylogeny and genome structure research. Unlike methods based on protein sequences, HomologMiner groups non-coding regions as well, the importance of which is increasingly recognized (Claverie et al., 2005). Furthermore, while contemporary research increasingly depends on orthologous alignment, most orthologous assignment tools assume a one-to-one relationship. Orthologous assignment is aided by the complete set of paralogs.

We aim to identify all duplicates excluding tandem and annotated interspersed repeats in a genome and to construct proper homologous groups. Our main contributions include:

1. We address the problem of establishing correct boundaries of duplicates. We define the notion of a consistent duplicates family and have verified that such families tend to provide meaningful biological units, e.g. complete genes.
2. We design heuristics to find families of duplicates that satisfy the definition described above.
3. We test the validity and usefulness of the obtained duplicate families for several mammalian species (particularly human).
4. We demonstrate our tool’s ability to identify gene families, tandem gene clusters and unannotated interspersed repeats.

### 2 PROBLEM DEFINITIONS AND RULES

Several sources (Bao et al., 2002; Edgar et al., 2005) point out that it is relatively easy to detect repetition, but more difficult to define biologically reasonable families. We tackle this problem with the goal of constructing homologous groups with high accuracy. A (homologous) group (HG) is composed of homologous group members. We try to ensure two properties for each HG:

1. **Purity**: Every pair of members in a group is homologous;
2. **Completeness**: All homologous members in the same group, unless no reliable alignment exists to support the homologous relationship.

Though sequences with similarity are not necessarily homologs, sequence similarity is still the best and most convenient criterion to identify homologs. Our method starts with a dot-plot of the self-alignment of the entire genome. A dot plot shows local similarity between two sequences (Fig. 2). Such a plot is basically a set of diagonal (similarity) lines, each representing a gapless alignment of region \([b_1, e_1]\) with region \([b_2, e_2]\). Regions corresponding to a line are potentially homologous. We define the following terms:

- **High Score Pair** (HSP) is a gap-free local alignment between two sequences, represented by HSP\((b_1, e_1, b_2, e_2)\) corresponding to a line in a dot plot from point \((b_1, b_2)\) to \((e_1, e_2)\). The regions \([b_1, e_1]\) and \([b_2, e_2]\) are projections of the HSP. Two HSPs overlap if either of their projections overlap.
- **shortChain** is a sequence of HSPs, where two HSPs are within a distance threshold \(T_{short}\). More formally, \(shortChain = HSP(b_1, e_1, b_2, e_2), i = 1, 2, \ldots, k\) such that \(0 \leq b_{i+1}^e - e_i^b \leq T_{short}, a = 1, 2, i = 2, 3, \ldots, k\). A chain is represented by \(shortChain = (b_1, e_1, b_2, e_2)\) where \(b_1\) and \(b_2\) are the starting positions of the first HSP, and \(e_1\) and \(e_2\) are the ending positions of the last HSP. The regions \([b_1, e_1]\) and \([b_2, e_2]\) are projections of this chain. Two chains overlap if either of their projections overlap.
- **longChain** is a sequence of shortChains, where the distance between two shortChains is within a threshold \(T_{long} > T_{short}\).

Given similarity lines, our goal is to identify regions of each duplicate unit, and establish homologous groups. In Figure 2a, the output homologous groups include \(HG_1((b_1, e_1), (b_2, e_2))\) and \(HG_2((b_4, e_4), (b_5, e_5))\). Ideally, whenever two HSPs \(h_1\) and \(h_2\) overlap, one projection of \(h_1\) is also a projection of \(h_2\). In other words, the boundaries...
of the HSPs would be consistent. In this situation, we would form a graph; regions would form nodes, HSPs would form edges and the connected components would form the homologous groups. Such perfect consistency is very rare, however. Usually HSPs’ boundaries differ because of evolutionary events such as sequence divergence and genome shuffling. As many sources (Bejerano et al., 2004; Kim et al., 2006) point out, simply constructing connected components does not achieve our objective. The completeness and Purity postulates may be contradictory unless we carefully determine the correct boundaries. Incorrect boundaries cause two kinds of problems: joining unrelated groups when non-homologous sequences are glued together, and cascades of splits, resulting in tiny useless fragments (details in Methods section).

Though the boundaries are ambiguous, we need to clearly define a duplicate unit (DU) to process. We did not find an objective definition that determines the precise limits for a DU; instead we formulate four rules that a consistent set of DU boundaries and homologous groups should satisfy:

1. **Acyclicity (Acyclicity)** A DU shall not contain significant repetitions. A significant repetition is a pair of chains with substantial overlap. For example in Figure 2a, region \([b_1, e_2]\) contains significant repetition; regions \([b_1, e_1]\), \([b_2, e_2]\) do not.

2. **Separation of accidental neighbors (Accidental Neighbor)** Different parts of a DU should not align to DUs of different groups. If two adjacent regions forming a DU would mix two groups, they are accidental neighbors. In Figure 2a, region \([b_1, e_2]\) contains accidental neighbors: subregion \([b_1, e_1]\) belongs to HG1, while subregion \([b_4, e_2]\) belongs to HG2.

3. **Coalescing persistent neighbors (Persistent Neighbor)** If most DUs of a group are consistently followed immediately by most DUs of another group within a distance threshold, we coalesce such pairs of DUs.

Each member of HG1 in Figure 2a could be divided into two regions by gaps. They can form two homologous groups \((r_1, r_2, r_3)\) and \((r_2, r_4, r_6)\). There are three pairs of persistent neighbors: \(r_1\) and \(r_2\), \(r_3\) and \(r_4\), \(r_5\) and \(r_6\), and they shall be coalesced to form HG1.

4. **Elimination of excessive repetitions (Anti-Excess)** A certain number of consecutive DUs are deemed to be tandem repeats; all their similarity lines are removed.

**ACCIDENTAL NEIGHBOR and PERSISTENT NEIGHBOR** may conflict. For example, there is no clear parameter to define adjacency. If it is too large, we may include several unrelated regions in the same unit and violate ACCIDENTAL NEIGHBOR; if it is too small, we may end up with a partial gene in a unit and violate PERSISTENT NEIGHBOR. We solve this heuristically. Anti-Excess eliminates non-functional tandem repeats, and vastly decreases the running time. The screened out sequences can be separately tested for novel repeats.

## 3 METHODS

We have described six rules: two for HG (Purity and Completeness) and four for DUs. Of the latter, we take care of Anti-Excess and Acyclicity in the preliminary stage. ACCIDENTAL NEIGHBOR and PERSISTENT NEIGHBOR are not achieved in specific stages, but are assured by iterating until HGs and DU boundaries stabilize (Fig. 3).

### 3.1 Preliminary demarcation of duplicate units

#### 3.1.1 Gap-free alignments

Sequences are first masked by RepeatMasker to remove low-complexity repeats and annotated interspersed repeats. We then compute all-to-all similarities. Speed is vastly improved by using gap-free alignments, avoiding dynamic programming. We use the initial stage of BLASTZ (Schwartz et al., 2003) to identify HSPs with substitution score >3000 (roughly equivalent to 70% nucleotide identity). Only HSPs with length larger than 50 are considered in this article.

#### 3.1.2 Removing excessively repetitive regions

Tandem repeats locate in the form of arrays, while interspersed repeats disperse in a genome. Tandem repeats are usually short and highly repetitive, with a few or hundreds of nucleotides per copy, and hundreds or even millions of copies. They are characterized by multiple contiguous copies. Interspersed repeats are usually longer with fewer copies. Some repeats of the same family are almost intact, but some may be less identical; they may be contiguous, and also can be degraded such that two copies are separated by a diverged region; their end points may be inconsistent, and they may also have indels. Edgar et al. (2005) provides more detail.

Repeats not masked by RepeatMasker hinder the subsequent procedures. In particular tandem repeats can be mistaken for tandem duplicates. We separate tandem repeats and duplicates based on their dot-plot properties. We tolerate interspersed repeats and identify them as HGs. The dot plot of some tandem repeat may look similar to tandem duplications (Fig. 1), so we do not want to be ‘overzealous’ in identifying tandem repeats. A tandem gene cluster tends to have diverged intergenic regions that create longer gaps between two copies, thus we identify tandem repeats by using only the fact that they are usually packed contiguously, or separated by short regions due to sequence decay. Our initial DUs are formed from projections of HSPs that are connected together if gaps are smaller than a threshold \(g\) (300).

We consider another parameter that may diagnose a tandem repeat cluster: \(c\) for the maximum number of crosses (4 by default).
HomologMiner

**Table 1.** Summary of output

<table>
<thead>
<tr>
<th>Species</th>
<th>Human</th>
<th>Macaque</th>
<th>Mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of HSPs from BLASTZ</td>
<td>33.4 M</td>
<td>11.3 M</td>
<td>60.6 M</td>
</tr>
<tr>
<td>Number of bases covered</td>
<td>170.5 Mb</td>
<td>104.8 Mb</td>
<td>180.9 Mb</td>
</tr>
<tr>
<td>Number of reduced HSPs</td>
<td>8.8 M</td>
<td>8.9 M</td>
<td>52.3 M</td>
</tr>
<tr>
<td>Number of bases covered</td>
<td>152.5 Mb</td>
<td>102.8 Mb</td>
<td>167.8 Mb</td>
</tr>
<tr>
<td>Number of non-singleton groups</td>
<td>27471</td>
<td>22043</td>
<td>24538</td>
</tr>
<tr>
<td>Average size of groups*</td>
<td>3.51</td>
<td>3.46</td>
<td>5.69</td>
</tr>
<tr>
<td>Average length of final DUs*</td>
<td>1232</td>
<td>869</td>
<td>880</td>
</tr>
<tr>
<td>Average transparency (%) *</td>
<td>98.3</td>
<td>97.5</td>
<td>92.6</td>
</tr>
<tr>
<td>Number of groups &gt; 50 members</td>
<td>91</td>
<td>81</td>
<td>223</td>
</tr>
<tr>
<td>Number of groups &gt; 10 members</td>
<td>1085</td>
<td>757</td>
<td>1652</td>
</tr>
<tr>
<td>Number of groups uniformly</td>
<td>66065</td>
<td>6553</td>
<td>5970</td>
</tr>
</tbody>
</table>

Average transparency (%) is defined as the percentage of the length of the final DUs that are unique (No. of bases covered by final DUs/No. of bases covered by query genome sequence).

*Non-singleton groups.

A horizontal sweep line creates in this preliminary DU (Fig. 2b). This c is the number of consecutive DUs described in the Anti-Excess rule. It is possible that functional elements and repeats are intermixed. Because we only remove lines, functional duplicates are not removed even if they are contained in the same preliminary DUs as repeats (Fig. 2c). The effectiveness of applying Anti-Excess in this step strongly depends on species (Table 1).

**3.3 Iterative refinement of duplicate units**

3.3.1 The drifting effect and bad cut effect Kim et al. (2006) described a drifting problem in clustering protein sequences with multiple domains. A similar problem exists in clustering genomic sequences because of adjacent regions (Bejerano et al., 2004). These regions may code adjacent domains of a protein, or they may be other functional regions that are accidentally adjacent (Fig. 6a). The problem is solved by cutting b into two pieces r_i and r_j. But more generally, because of alignment drifts, it is possible to have regions r_1, r_2, ..., r_k such that r_i is highly similar to r_{i+1}, but with r_i totally unrelated to r_k (Fig. 6b). It is not obvious how to choose the cutting point in this case. If the cut point is not chosen properly, it causes a cascading effect and leads to further unnecessary cuttings (Fig. 4). The cutting point is determined similarly to Section 3.1.4, except that the search window is determined by the middle points of two chains that do not overlap (Fig. 7a). Besides carefully choosing the cutting point, we also employ an iterative approach to make further cuts and to coalesce results of erroneous cuts.

3.3.2 Iteratively enforcing Accidental Neighbor and Persistent Neighbor We need to split a preliminary DU violating Accidental Neighbor. A DU is associated with many alignments with other DUs. After splitting a region, we must update its alignments. Hence, once we cut a region and update its connections to other regions, more regions may be affected. It is impossible to maintain all such relationships at once. Similarly to our handling of Acyclicity, we need to carefully decide to cut, choose the cutting point, and recover erroneous cuts. Accidental Neighbor guides the decision to cut; Persistent Neighbor guides re-joining of adjacent regions to correct erroneous cuts. We split a region if the majority of chains do not overlap (Fig. 7a). We observe that if a region has been mistakenly broken into two, shortChains from the first region and shortChains from the next region should form longChains without violating Acyclicity. So we re-join two adjacent regions if the majority of chains do not overlap (Fig. 7b).

We continue splitting as long as Accidental Neighbor can be applied, then apply Persistent Neighbor exhaustively. We repeat such phases until the number of DUs becomes stable or an iteration limit (currently 15) is met.
3.3 Construction of homologous groups

Our goal is to build HGs from DUs that we have computed. We define a graph with DUs as nodes. Node $u$ is connected to $v$ if $u$ and $v$ contain projections of the same chain. Because $u$ and $v$ may be the same node for a certain chain, some nodes may not have edges, and they become singletons. Other connected components identify diverged homologs, but they may also join unrelated groups because of the drifting effect. We describe the following heuristics to prevent such effect, while taking advantage of the connected component approach. To enforce ACCIDENTAL NEIGHBOR and PERSISTENT NEIGHBOR, we modify both graph and DUs.

3.3.1 Construction of transparent connected components

Our drift preventing heuristic uses the notion of a transparent path. If two connected nodes $n_i$ and $n_j$ are not directly connected, then either they are so diverged that an aligner (e.g. BLASTZ) fails to identify their similarity, and they indeed should be in the same component; or they are unrelated at all, being grouped together by mistake. In either case, $n_i$ and $n_j$ are connected by a path, and this path provide clues to decide if $n_i$ and $n_j$ are related. We define the following:

- An **alignment** is a chain of HSPs.
- **Alignment transitivity**: If position $a$ aligns to position $b$, and position $b$ aligns to position $c$, then $a$ aligns to $c$.
- A path $n_{b_1}...n_i$ is **transparent** if there exists a sub-region $s_i$ on $n_j$ and a sub-region $s_j$ on $n_j$ such that $s_i$ and $s_j$ can be aligned via other nodes on this path by alignment transitivity.

$n_i$ and $n_j$ are homologous if there is a transparent path between them; otherwise, they were placed in the same component by the drifting effect. When we construct connected components with breadth-first iteration, we add a node to a component only if it has a transparent path from the root. The resulting component is a transparent connected component (tcc).

3.3.2 Iteratively merging nodes and decomposing components

Two nodes are genome neighbors if they locate within...
sets of graph neighbors relative to \( N_i \) or some of their members are genome neighbors where the \( n_i \) component. Otherwise we merge mistakenly split, and we should merge them. Let component; (ii) \( n_i \) and \( n_j \) are persistent neighbors, they have been mistakenly split, and we should merge them. Let \( N_i \) and \( N_j \) denote the sets of graph neighbors for \( n_i \) and \( n_j \). In case (ii), \( |N_i \cap N_j| \) is small relative to \( |N_i| \) and \( |N_j| \). In case (ii), \( |N_i \cap N_j| \) is relatively large, or some of their members are genome neighbors where the two nodes in each neighbor pair are from \( N_i \) and \( N_j \). Let \( N' \) denote the number of such pairs of genome neighbors and \( N'' \) denote \( |N_i \cap N_j| \). Graph neighbor agreement (gna) is defined to be max((\( N' + N'' \))/\( |N_i| \), (\( N' + N'' \))/\( |N_j| \)). If gna is low (<20%), a min-cut is determined by the Ford–Fulkerson algorithm for max flow (Cornrn et al., 2001) and the cut edges are removed to decompose the component. Otherwise we merge \( n_i \) and \( n_j \). This process is repeated until no change is made, or an iteration limit is met (currently 10).

3.3.3 Collapsing nodes Two groups are adjacent if nodes of one group are all adjacent to all nodes of another group. If two adjacent groups have the same number of nodes, and the longest distance is short, then the groups are collapsed: the boundaries of new groups are combined from the pair of adjacent nodes. The distance threshold (3000 bases) is arbitrary. If it is too small, no groups will be fused. If it is too large, it might fuse different genes into the same region.

4 RESULTS

The choice of genomes in this article is to a degree arbitrary, because HomologMiner is designed to be applicable to any genome. The parameters were adjusted to give efficient performance for the human genome, and then tested on two other genomes. The availability of a new set of repeats for the macaque genome made it possible for us to test the effectiveness of HomologMiner in identifying interspersed repeats (Section 4.2.2). We also applied HomologMiner to mouse which is a more diverged mammal and may have different repetitive structures. The results of these species are summarized in Table 1. We provide preliminary analysis of overall characteristics of homologous groups. Some interesting large groups are summarized in Table 2. It can be seen that mouse does have more larger repetitive families (number of groups \( \geq 50 \) members in Table 1) which might be gene families or interspersed repeats. The olfactory gene family is an example (Tables 2 and 3). We also show several applications based on the homologous groups found.

4.1 Characteristics of HGs

4.1.1 Transparency test The transparent path defined in Section 3.3.1 provides an estimate of whether two members in the same group are homologous or not. Recall that when we construct a tcc, we assure transparency of all nodes with the root, but the existence of a transparent path is not transitive (Fig. 6). Therefore we test tccs for pairwise transparency. In particular, path transparency is evaluated for every pair of nodes in a tcc. Suppose there are \( N \) nodes in a group, there are \( C = N(N - 1)/2 \) pairs of connected nodes. Denote \( C' \) as the number of pairs of nodes which have transparent paths, the ratio \( C'/C \) gives an estimate of the proportion of true homologs. On average the transparency is high; in other words, the average specificity is high. The average and the ratios for some interesting large groups are listed in Tables 1 and 2. Note that a transparency score below 100% also implies that different choices of root nodes may lead to different tccs; the transparency score indicates the proportion of nodes that are stable when different root nodes are chosen.

4.1.2 Association of groups with function annotations for human Similar to Bejerano et al. (2004), we try to associate each group with annotations. We performed the association test with four attributes: known gene mRNA, Ensembl predicted protein gene and evoFold for RNA structure prediction. All annotations were downloaded from the UCSC genome browser (Kent et al., 2002). In Figure 8, we show counts of groups that have at least some fraction (positive, 1/2, expected) of members intersecting annotations. In this test, we considered an HG member to intersect an annotation if they overlap by at least one base. Other criterion can be considered for further investigation. The counts for the three protein-related attributes are pretty similar, while the counts for evoFold are very small partly because the annotation set is small. To find groups significantly enriched for some attribute, we compute a \( P \)-value for the enrichment level of any group, for each attribute. We use the observed frequency of a given attribute \( A \) in our set of regions as an estimate of the background probability \( q \) of \( A \). For RNA, \( q=0.01 \) is borrowed from (Bejerano et al., 2004). We then compute a \( P \)-value for the number of observations of \( A \) in the cluster, with the NULL model considering them as independent trials over the number of segments in the cluster. The \( P \)-value is the chance of observing at least as many instances with \( A \), according to the corresponding binomial distribution. The least possible \( P \)-value for a cluster of size \( n \)
and attribute probability $q$ is $q^n$, and the significance level of $10^{-4}$ cannot be achieved by a cluster smaller than $-4/\log(q)$. This suggests we should omit clusters smaller than 10 for attributes other than RNA. So we test protein annotations only for the largest 1085 groups (with size $\geq 10$). We consider $P$-values to be significant if they are below $10^{-4}$ prior to Bonferroni correction. There are 24 significant known gene groups in this case. Some interesting significant groups are listed in Table 2. For the attribute of RNA, we perform the test on all 27 471 groups (size $\geq 10$), and there is no group with an uncorrected $P$-value below $10^{-3}$. The failure to identify groups enriched in RNA structures is partly due to the shortage of annotations. Moreover, HomologMiner discarded short regions which possibly contain RNA structures.

4.1.3 Uniform distribution test Some of the identified homologous groups consist of interspersed repeats. Interspersed repeats are believed to be inserted in a genome randomly. We use the Kolmogorov–Smirnov goodness-of-fit test to determine whether the members in a homologous group locate randomly in the genome or not. The numbers of groups whose members locate randomly (with $P$-value $> 0.05$) are shown in Table 2. Several large uniformly distributed groups are shown in Table 2.

4.1.4 Group size distribution for human We obtained 27 471 non-singleton homologous groups, with average size of 1232 bases for a group member. The largest group has 825 members.

Most groups (18 637) have only two members. The log-log plot of the accumulated distribution of group sizes (Fig. 9) shows a clear power law relationship, where the group size is defined to be the number of group members. These results are consistent with the observations for protein families in Enright et al. (2002).

### Table 2. Representative large homologous groups and their attributes

<table>
<thead>
<tr>
<th>Species</th>
<th>Number of copies</th>
<th>Transparency (%)</th>
<th>Average length</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>825</td>
<td>99</td>
<td>1393</td>
<td>Olfactory receptor gene family</td>
</tr>
<tr>
<td></td>
<td>353</td>
<td>93</td>
<td>495</td>
<td>Significant for known genes</td>
</tr>
<tr>
<td></td>
<td>301</td>
<td>88</td>
<td>1123</td>
<td>Zinc finger family</td>
</tr>
<tr>
<td></td>
<td>207</td>
<td>91</td>
<td>1626</td>
<td>Keratin family</td>
</tr>
<tr>
<td></td>
<td>199</td>
<td>99</td>
<td>921</td>
<td>Zinc finger family</td>
</tr>
<tr>
<td></td>
<td>161</td>
<td>99</td>
<td>756</td>
<td>Non-coding, uniform distribution</td>
</tr>
<tr>
<td></td>
<td>136</td>
<td>97</td>
<td>1544</td>
<td>Zinc finger family</td>
</tr>
<tr>
<td></td>
<td>109</td>
<td>93</td>
<td>947</td>
<td>Significant for known genes</td>
</tr>
<tr>
<td>Macaque</td>
<td>621</td>
<td>99</td>
<td>1067</td>
<td>Olfactory receptor gene family</td>
</tr>
<tr>
<td></td>
<td>490</td>
<td>95</td>
<td>672</td>
<td>Zinc finger gene family</td>
</tr>
<tr>
<td></td>
<td>308</td>
<td>95</td>
<td>503</td>
<td>Non-coding, uniform distribution</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>98</td>
<td>673</td>
<td>Zinc finger gene family</td>
</tr>
<tr>
<td></td>
<td>124</td>
<td>96</td>
<td>1564</td>
<td>Newly identified LTR</td>
</tr>
<tr>
<td></td>
<td>92</td>
<td>99</td>
<td>400</td>
<td>Non-coding, uniform distribution</td>
</tr>
<tr>
<td>Mouse</td>
<td>1369</td>
<td>99</td>
<td>1622</td>
<td>Olfactory receptor gene family</td>
</tr>
<tr>
<td></td>
<td>334</td>
<td>93</td>
<td>390</td>
<td>Non-coding and uniform distribution</td>
</tr>
<tr>
<td></td>
<td>254</td>
<td>98</td>
<td>813</td>
<td>Non-coding and uniform distribution</td>
</tr>
</tbody>
</table>

4.2 Application examples

4.2.1 Olfactory receptor gene family Due to the lack of annotations of gene families that can be used with high-throughput programs, it is difficult to evaluate HomologMiner comprehensively. The olfactory receptor (OR) gene family is one of the largest gene families in human genome. It has been extensively studied and has well-annotated databases. So the olfactory gene family annotation can be used to estimate the accuracy of homologous groups. The HORDE database (Olender et al., 2004) contains 854 entries of OR genes and pseudogenes for human. Our largest identified HG contains 825 members. Among them, 819 overlap with 813 annotated members of OR gene family. The 41 missing members are largely attributed to the repeat removal procedure, which may be re-examined. We cannot give as precise an evaluation for other homologous groups, but the transparency test results suggest good levels of the purity and completeness of a homologous group obtained from HomologMiner. We observe that the olfactory gene families have different sizes for different species. In Table 3, we show the sizes of olfactory gene families for each species, as well as connectivity defined as the number of copies of genes or pseudogenes that a copy is aligned to on the average.

### Table 3. Summary of olfactory receptor gene/pseudogene families

<table>
<thead>
<tr>
<th>Species</th>
<th>Number of copies</th>
<th>Average length</th>
<th>Connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>825</td>
<td>1393</td>
<td>312</td>
</tr>
<tr>
<td>Macaque</td>
<td>621</td>
<td>1067</td>
<td>192</td>
</tr>
<tr>
<td>Mouse</td>
<td>1369</td>
<td>1622</td>
<td>581</td>
</tr>
</tbody>
</table>

4.2.2 An interspersed repeat family We obtained a new library of interspersed repeats for macaque, including types of ALU, L1 and LTR. Among them, ALU- and L1-related new repeats are very similar to the ones already annotated, so we did not find significant homologous groups associated with them.
However, we identified a group homologous with LTRs. It has 124 DUs, with transparency score of 96%, average length of 1564 bases and uniform distribution test $P$-value 0.07, most of whose members align to several new LTR repeats. After running RepeatMasker without the new library using the default search, there are 50 LTR/ERVK repeats matched, with 6512 bases masked. After running RepeatMasker with the new library using the quickest search, there are 127 LTR/ERVK repeats matched, with 190186 bases masked. This shows that HomologMiner is useful in looking for new interspersed repeats.

### 4.2.3 Gene clusters in genomes

Many homologous genes locate close to each other; we call them tandem gene clusters. We looked for boundaries of tandem gene clusters for human, using the locations of duplicate units. We obtained 512 clusters, with average length of 113K bases, and minimum distance of 50K bases between two clusters. Each identified cluster does not necessarily correspond to a gene family, nor does it necessarily contain only one type of gene. But the region of a cluster is enriched with duplications, and a multiple sequence aligner assuming one-to-one orthologous relationship does not perform well in such a region (Blanchette et al., 2004). Identifying these regions will allow different alignment methods to be applied to them.

### 4.3 Running time and memory consumption

After the initial phase in which we obtain HSPs, HomologMiner is very efficient. Memory consumption is linear to the number of HSPs, while running time grows slowly. It took around 1, 1 and 10 CPU hours on a single processor computer (64-bit 3 GHz) to process human, macaque and mouse genomes using around 3, 1 and 6 GB of memory respectively. The different behavior for mouse is due to a much larger number of HSPs, possibly because of larger gene families, and more unannotated interspersed repeats.

### 5 DISCUSSION AND FUTURE WORK

We used the first 10 Mb of sequence of chr11 from hg17 to compare the performance of HomologMiner with three other repeat identification tools: RECON, RepeatScout and RepeatGluer. There are 122 OR genes or pseudogenes with average length of 937 bases in this region from HORDE database (Olender et al., 2004), together with a beta globin gene cluster of six copies. Among the four tools, RepeatScout differs from the others since it works from sequences directly, while the other three rely on alignments produced by independent pairwise alignment tools. This program found both gene families in the sense that it found one consensus sequence for each of them. While it is easy to find the original sequences that match the consensus, extra post-processing would still be required to construct the whole families properly. To be consistent in comparing the other three tools, we use the same pairwise alignment produced by the initial stage of BLASTZ. The biggest family produced by RECON has 166 members with average length of 538 bases, 164 of which intersect with 99 olfactory annotations. RepeatGluer fragmented the olfactory genes into eight parts, with 80 bases and 103 copies on average. It also provided a consensus sequence for each part. HomologMiner produced a homologous group of 121 members with average length of 1005 bases intersecting 121 annotations. HomologMiner also correctly identified the beta-globin cluster as a single group, while both RECON and RepeatGluer fragmented the beta-globin genes into many parts. We suspect this might be because of the nature of HSPs being short and not covering the whole genes, while HomologMiner is designed to chain short segments properly. This preliminary comparison shows that different tools are designed for different purposes. HomologMiner is efficient in building homologous groups with high accuracy while using only low-quality pairwise alignments. RepeatScout and RepeatGluer perform well in obtaining consensus sequences.

For the human genome, the gap-free alignment procedure produces HSPs covering 171 Mb, ~5.7% of the genome (Table 1). Applying ANTI-EXCESS produces HSPs covering...
153 Mb, ~5.1% of the genome. Though the total covered length is not reduced much (5.7% → 5.1%), the number of HSPs is reduced dramatically (33.4 → 8.8 million). This indicates that the removed regions are a relatively small portion of the genome, but largely redundant copies. These repeats might be high-complexity tandem repeats or degraded tandem repeats. Perhaps further investigation may be necessary.

The identified HGs from the three genomes include gene families, interspersed repeats and segmental duplications. We plan to categorize each of the above types. If more newly identified interspersed repeats (e.g. for macaque) become available, we can test whether our uniform distribution testing is differentiating interspersed repeat families from the rest of the groups. We can also apply spectral clustering techniques to classify subfamilies of HGs. When applied to a gene family, they provide better functional inference. Using the extra input of pairwise alignment of two species, we can build up the HGs between two species. The complete set of paralogs provide a good starting point to do orthologous assignment at the level of whole genomes.

From Bao et al. (2002), it seems there does not exist any clean algorithmic approach for repeat classification. We observe that this is especially true for large sequence input. For example, our methods in Section 3.2 suffice to construct HGs when processing a single chromosome of human, while applying 3.2 alone to the whole genome does not produce meaningful HGs of large size. We have developed a set of procedures to work well with primates and a rodent. As heuristic software, HomologMiner has a set of parameters, adjusted to work efficiently and effectively with the human genome. It is difficult to accurately describe the effect of different combinations of parameters. The effect of parameters on different genomes needs further study. We expect more work to be done including parameter adjustment, and even designing new procedures when working with genomes with more complex repetitive structures. For example, the parameter \( d \) in Figure 2 which is the distance between two adjacent diagonal lines is not used in HomologMiner, because the other two parameters are effective enough so far. We may need to incorporate \( d \) to identify tandem repeats in the future to improve performance in genomes like mouse. As another possible improvement, we implemented a minCut algorithm (modified from Stoer et al., 1994), and applied it to the group that contains most of the olfactory receptor regions. We observed that minCut tends to find small adjacent regions which are wrongly grouped, and minimum weight minCut tends to find weak copies of pseudogenes. Before including it in HomologMiner, we need better annotations for other gene families to test it.

Acknowledgement

We thank Dr Webb Miller for discussion and support. M.H., C.-H.H. and R.S.H. were supported by NIH grant HG02238.

Conflict of Interest: none declared.

REFERENCES


