A non-programmatic description of methods

Tim Millar

2017-07-31

Contents

Overview

TEFingerprint is a Python 3 library and command line tool for producing transposon based fingerprints from paired end reads. TEFingerprint seeks to characterise data sets for further downstream analysis.

The TEFingerprint pipeline is composed of the following steps:

1. Mapping paired end reads to a database of known transposons
2. Identification of *informative* reads
3. Mapping informative reads to a reference genome
4. Fingerprinting (density based clustering) of mapped read tip positions
5. Computing the union of the fingerprints from multiple samples
6. Comparing read counts among samples within the combined fingerprint
7. Downstream filtering and analysis of results

Selecting Informative Reads

The initial stage of the TEFingerprint process is to identify *informative* reads. Reads are informative if they relate some information about the location of transposon insertions when compared to a reference genome.

Paired end reads are mapped to a library of known transposon sequences. Pairs are then identified in which one read has mapped to a transposon and the other is unmapped. Assuming that our library of known transposon sequences is largely complete, unmapped reads are likely to (primarily) contain non-transposon-genomic DNA. Read pairs in which both reads have mapped to a transposon sequence can be used as an additional source of information if one read has a significant soft-clipped region at its 5-prime (outer) end. In these cases, the soft-clipped section can be extracted and included as (short) informative read. Any pairs in which neither read is mapped are uninformative and ignored.

The (unmapped) informative reads are tagged with the transposon that their pair has mapped to, and then mapped to a reference genome. These reads will tend to map in stranded clusters either end of a location at which a transposon is present in the sample.

Fingerprinting

Once the informative reads have been extracted and aligned to the reference genome we aim to identify clusters of these reads that indicate flanking-regions of transposon instertions in the sample genome. The pattern of these clusters across the reference genome is referred to as a *fingerprint*.

Categorising informative read tips

Informative-reads are grouped based on strandedness and user defined categories i.e. super-families. The positions of the $3'$ read-tips (i.e. the read ends closest to the potential transposon insertions) are then extracted into a array of integer values per reference molecule for each categories-strand group. Clusters of informative-reads are then identified using a novel density-based clustering method described bellow.

Density based clustering

When an non-soft-clipped informative read is identified, the informative-read does not map to a transposon while its mate does so the informative-read will usually be within insert-length distance of a transposon insertion in the sample

genome. Thus the expected size of transposon-flanking clusters can be reasonably estimated based on the approximate insertion size of the initial paired end data.

The DBSCAN family of clustering algorithms identify clusters based on point *density*. Points that form dense regions are grouped into cluster while points in sparse regions are classified as noise points. DBSCAN does not require the number of clusters to be specified as an input because clusters are defined entirely in terms of point density. A DBSCAN based approach is suitable for identifying transposon-flanking regions for the following reasons:

1. We expect to find a variable and often large number of clusters for each array of read-tip positions and this number is not known *a priori*
2. It is reasonable to expect some level of background noise in the data due to issues with sequence alignment
3. An expected level of cluster density can be inferred from the paired-read insertion size and an estimation of the background noise (i.e. via visualisation)
4. The method can be adapted to a highly efficient algorithm for sorted univariate data

An issue with a standard implementation of DBSCAN is it's inflexibility when identifying clusters at differing levels of density. In principle this should not be an issue because we expect a reasonably consistent density among clusters. However, in practice neighbouring clusters will often be identified as a single larger cluster thereby 'missing' an insertion site.

Campello *et al.* (2015) described HDBSCAN, a hierarchical version of DBSCAN, that can detect clusters at multiple density levels. HDBSCAN is much too flexible to be suitable for identifying transposon-flanking regions. For example, HDBSCAN will often identify regions with high transposon density as a single cluster or identify multiple sub-clusters in a single flanking region based alignment artefacts including the differing signal between soft-clipped and non-soft-clipped informative-reads.

Description of algorithm

We present a novel algorithm for density based clustering of univariate points with noise. Our algorithm is derived from HDBSCAN but produces clusters that are comparable with those found by DBSCAN*. As in DBSCAN*, our method has an explicit density threshold below which clusters are not identified. Unlike DBSCAN*, our method provides some flexibility for splitting poorly supported clusters into strongly supported sub-clusters.

In our algorithm, a specific density of objects is targeted by the minimum number of points required to form a cluster m_{pts} and a global maximum value of epsilon ε which is referred to as the constant $\mathcal{E}$. Here we use the following definitions from Campello *et al.* 2015:

- An object (i.e. read tip) $\mathbf{x}_p$ is called a *core object* w.r.t. ε and m_{pts} if there are at least $m_{\text{pts}} - 1$ additional objects within ε range of $\mathbf{x}_p$.
- The *core distance* of an object $d_{\text{core}}(\mathbf{x}_p)$ w.r.t. m_{pts} is the distance from $\mathbf{x}_p$ to it's $m_{\text{pts}} - 1$ nearest neighbour.
- A *cluster* $\mathbf{C}$ w.r.t. ε and m_{pts} is a non-empty maximal subset of the set of points $\mathbf{X}$ such that every pair of objects in $\mathbf{C}$ are density connected.
- The *minimum epsilon* of a cluster $\varepsilon_{\min}(\mathbf{C})$ is the value of ε bellow which the cluster $\mathbf{C}$ either contains less than m_{pts} core objects (ceases to be a cluster) or contains more than 1 subset of density connected objects (is divided into child clusters).
- The *maximum epsilon* of a cluster $\varepsilon_{\max}(\mathbf{C})$ is the value above which a which that cluster is incorporated into another cluster.

Initial clusters are identified at a density defined m_{pts} and $\varepsilon = \mathcal{E}$. Therefore the initial clusters are identical to those found by DBSCAN*.

The difference between $\mathcal{E}$ and $d_{\text{core}}(\mathbf{x}_p)$ can be thought of as the *lifetime* of object $\mathbf{x}_p$. The cluster $\mathbf{C}_i$ is selected if

$$\sum_{\mathbf{x}_j \in \mathbf{C}_i} \frac{\mathcal{E} - \max\{d_{\text{core}}(\mathbf{x}_j), \varepsilon_{\min}(\mathbf{C}_i)\}}{\max\{d_{\text{core}}(\mathbf{x}_j), \varepsilon_{\min}(\mathbf{C}_i)\} - d_{\text{core}}(\mathbf{x}_j)} \geq 1$$

i.e. if the proportion of combined object lifetimes above $\varepsilon_{\min}(\mathbf{C}_i)\}$ is greater or equal to that bellow $\varepsilon_{\min}(\mathbf{C}_i)\}$ then the cluster is selected. If a cluster is not selected then this calculation is performed again for each child cluster. The use of a constant $\mathcal{E}$ as opposed to $\varepsilon_{\max}(\mathbf{C})$ ensures that the parent cluster is increasingly favoured as the algorithm recurses down the cluster hierarchy. A direct effect of this selection criteria is that a cluster cannot be selected unless $\varepsilon_{\max}(\mathbf{C}) \geq \mathcal{E}/2$.

Comparing Multiple Fingerprints

Fingerprinting produces a binary (i.e. presence absence) pattern of loci across a reference genome indicating the boundaries of transposon insertions within a samples genome. However the binary pattern is extracted from non-binary data (read positions/counts) and the absence of a cluster in one sample does not guarantee an absence of signal (reads) within that location. Therefore a direct comparison of fingerprints from multiple samples may be misleading. A better approach is to compare read counts within the fingerprints among the compared samples. To this end we calculate the interval union of fingerprints among samples and count the informative read tips within the combined fingerprint.

Mathematically, each cluster within the fingerprint of a single sample can be expressed as a closed integer interval. For example a cluster of read tips spanning the (inclusive) base positions 11 and 27 (inclusive) can be expressed as the closed

interval $[11, 27]$. The fingerprint of sample i can then be expressed as a union of non-overlapping intervals found within that sample; $\mathcal{U}_i$. Thus the union of fingerprints for a set of n samples is calculated

$$\bigcup_{i=1}^{n} \mathcal{U}_i$$

The new union of fingerprints represents the boundaries of potential transposon insertions across all samples. We then use each interval within the union of fingerprints as a potential insertion site for all of the samples. A samples read count within a given interval is recorded as evidence for the presence or absence of an insertion at the genomic location represented by that interval. In this manner, TEFingerprint identifies comparative characters (potential insertion sites) for a group of samples and summarises each samples support (read counts) for the presence/absence of a character.

Downstream Filtering and Analysis

TEFingerprint does not assume a specific for investigating transposon insertion locations. Instead it summarises the input data into a flexible format that can be used for multiple downstream tasks. The output formats available are GFF3 and CSV (or other delimited text formats).