TEFingerprint 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 in which one read has mapped to a transposon and the other is unmapped are identified. 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 are 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 strand and user defined categories e.g. a taxonomic level such as super-family. 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.

A non-soft-clipped informative read is identified when its mate read has mapped to a know transposon. The informative read has then been mapped to a reference genome where it will (usually) be within insert-length distance of a transposon insertion in the sample genome. Thus the insertion size of the initial paired end data is a reasonable estimate of the expected size of transposon-flanking clusters of informative reads. Using this information, clusters of informative read-tips are identified using a novel density-based clustering algorithm described bellow.

Density based clustering

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-like 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 (i.e. an array of read-tip positions)

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.

IDBCAN algorithm

We present a novel algorithm IDBCAN (**I**nterval **D**ensity **B**ased **C**lustering of **A**pplications with **N**oise) which is derived from DBSCAN. IDBCAN identifies clusters based on the density of objects within intervals of a size. As in DBSCAN, IDBCAN requires a target density to be defined in terms of m_{pts}, the minimum number of points (objects) required to form a cluster and ε a distance that limits the dispersion of those objects. Here we use the following definitions:

***Definition* 1** (*sub-cluster*). A *sub-cluster* w.r.t. ε and m_{pts} is a set of m_{pts} objects, $\mathbf{X} = \{\mathbf{x}_p, ..., \mathbf{x}_{p+m_{\text{pts}}}\}$ that are each within ε range of every point in that set.

***Definition* 2** (*core-object*). A *core-object* w.r.t. ε and m_{pts} is any object $\mathbf{x}$ that is in one or more sets of m_{pts} objects classified as a sub-cluster.

***Definition* 3** (*noise-object*). A *noise-object* w.r.t. ε and m_{pts} is any object that is not a core-object.

***Definition* 4** (*density-overlapping*). Two sub-clusters $\mathbf{c}_p$ and $\mathbf{c}_q$ are *density-overlapping* w.r.t. ε and m_{pts} if they share one or more core-objects.

***Definition* 5** (*density-connected*). Two sub-clusters $\mathbf{c}_p$ and $\mathbf{c}_q$ are *density-connected* w.r.t. ε and m_{pts} if they are directly or transitively density-overlapping.

***Definition* 6** (*cluster*). A *cluster* w.r.t. ε and m_{pts} is a non-empty maximal subset of the set of core-objects $\mathbf{X}$ in which every pair of objects are found in either the same sub-cluster or within a pair density-connected sub-clusters.

Based on definitions 1 and 2 IDBCAN differs from DBSCAN in that m_{pts} objects must collectively be identified as core objects rather than identifying a single core object at a time. This in turn means that a cluster (following definitions 3-6) will always include at least m_{pts} objects and that cluster identification is deterministic. In DBSCAN a cluster may contain fewer than m_{pts} objects if on of its border-objects is "stolen" by a neighboring cluster and the assignment of border-objects to clusters is not deterministic (though often border-object assignment is deterministic based on the implementation). DBSCAN* is a variation of DBSCAN in which border-objects are treated as noise objects, this results in deterministic identification of clusters but cluster will often contain fewer than m_{pts} objects (as few a single object in a cluster regardless m_{pts}.

The properties of IDBCAN make for intuitive identification of clusters in a univariate space and the appropriate values for parameters required by IDBCAN can be logically estimated when identifying clusters of informative reads in TEFingerprint. The value ε is the expected interval width of a region of informative reads flanking a transposon insertion and can be reasonably estimated as being no larger than the approximate insertion size of paired-reads. The value m_{pts} is the minimum number of read (tips) required within an ε-wide region for that region to be identified as flanking a transposon insertion. This can reasonably estimated from the observed depth of informative reads and is a trade off between type one and type two error.

SIDBCAN algorithm

The primary aim of TEFingerprint is to identify the signal of transposon insertion sites using clusters of informative reads mapped to a reference genome. A potential issue with both IDBCAN and DBSCAN is that they assume that all clusters can be identified based on a single density threshold. In principle this is a fair assumption because we expect that genome regions adjacent to an insertion site will have high read densities identified as clusters and that other genome regions will have low read densities classified as noise. However if two or

more insertion sites are sufficiently close to one another, the region between then may be above the specified read density (i.e. a contiguous region of overlapping sub-clusters). In IDBCAN, overlapping (sub)-clusters of reads are by definition classified as a single cluster. Therefore the signal of two or more proximate insertions may be interpreted as a single cluster from which single insertion site is inferred. This phenomenon can be mitigated by a clustering algorithm that can identify clusters at multiple density levels.

Several hierarchical extensions of DBSCAN have been proposed including HDBSCAN* (Campello *et al.* 2015) and OPTICS (Ankerst *et al.* 1999). HDBSCAN* builds a minimal spanning tree of hierarchical clusters. The algorithm then selects a non-nested set of clusters from the minimal spanning tree based on a measure of cluster density for all values of ε. This approach is 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 artefact's including the differing signal between soft-clipped and non-soft-clipped informative-reads. In OPTICS the different values of ε may be manually selected for different part of the minimum spanning tree. This approach is unsuitable for dentifying transposon-flanking regions because of the share number of cluster expected (often in the hundreds of thousands.

We present Splitting-IDBCAN (SIDBCAN) a hierarchical version of IDBCAN. SIDBCAN requires the same parameters as IDBCAN (ε and m_{pts}) and initially identifies the same set of clusters. SIDBCAN then attempts to split poorly supported clusters into more strongly supported clusters that may be found with a lower value of ε.

***Definition* 7** (*minimum epsilon*). The *minimum epsilon* of a cluster $\varepsilon_{\min}(\mathbf{C})$ is the value of ε such that either two or more density-connected sub-clusters within $\mathbf{C}$ when $\varepsilon = \varepsilon_{\min}(\mathbf{C})$ would be non-density-connected sub-clusters when $\varepsilon < \varepsilon_{\min}(\mathbf{C})$. Or $\mathbf{C}$ consistis of one or more density-connected sub-clusters when $\varepsilon = \varepsilon_{\min}(\mathbf{C})$ but not when $\varepsilon < \varepsilon_{\min}(\mathbf{C})$ (i.e. is not a valid cluster).

***Definition* 8** (*maximum epsilon*). A cluster $\mathbf{C}$ has *maximum epsilon* $\varepsilon_{\max}(\mathbf{C})$ such that if $\varepsilon > \varepsilon_{\max}(\mathbf{C})$ then sub-clusters within $\mathbf{C}$ would be density-connected to one or more sub-clusters that are part of a separate cluster when $\varepsilon = \varepsilon_{\max}(\mathbf{C})$.

***Definition* 9** (*core distance*). The *core distance* $d_{\text{core}}(\mathbf{x}_p)$ of an object $\mathbf{x}_p$ w.r.t. ε and m_{pts} is maximum distance between $\mathbf{x}_p$ and any object in the set of objects comprising its $m_{\text{pts}} - 1$ nearest neighbours.

Initial clusters are identified as in IDBCAN using a density defined by m_{pts} and ε. Support of the initial clusters is then assessed in comparison to its child clusters (2 or more subsets of density connected objects that exist bellow the minimum epsilon of the initial/parent cluster) if present.

We refer to difference between $\varepsilon_{\text{initial}}$ and $d_{\text{core}}(\mathbf{x}_p)$ as the *lifetime* of object $\mathbf{x}_p$. The *total lifetimes* of all objects within cluster $\mathbf{C}_i$ is calculated

$$L_{\text{total}}(\mathbf{C}_i) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \varepsilon_{\text{initial}} - d_{\text{core}}(\mathbf{x}_j)$$

The *support* for a cluster is defined as the portion of those lifetimes that occurs when $\varepsilon \geq \varepsilon_{\min}(\mathbf{C}_i)$

$$S(\mathbf{C}_i) = \sum_{\mathbf{x}_j \in \mathbf{C}_i} \varepsilon_{\text{initial}} - \max\{d_{\text{core}}(\mathbf{x}_j), \varepsilon_{\min}(\mathbf{C}_i)\}$$

The *excess lifetimes* of objects within cluster $\mathbf{C}_i$ is the portion of object lifetimes that occurs when $\varepsilon < \varepsilon_{\min}(\mathbf{C}_i)$, i.e. when the cluster splits into child clusters or ceases to exist

$$\begin{aligned} L_{\text{excess}}(\mathbf{C}_i) &= L_{\text{total}}(\mathbf{C}_i) - S(\mathbf{C}_i) \\ &= \sum_{\mathbf{x}_j \in \mathbf{C}_i} \max\{d_{\text{core}}(\mathbf{x}_j), \varepsilon_{\min}(\mathbf{C}_i)\} - d_{\text{core}}(\mathbf{x}_j) \end{aligned}$$

The cluster $\mathbf{C}_i$ is selected if $S(\mathbf{C}_i) \geq L_{\text{excess}}(\mathbf{C}_i)$, i.e. if the proportion of combined object lifetimes when $\varepsilon \geq \varepsilon_{\min}(\mathbf{C}_i)$ is greater or equal to the proportion of lifetimes when $\varepsilon < \varepsilon_{\min}(\mathbf{C}_i)$. If a cluster is not selected then support is assessed for each child cluster within $\mathbf{C}_i$. This can be written

$$\text{selection}(\mathbf{C}_i) = \begin{cases} \mathbf{C}_i & \text{if } S(\mathbf{C}_i) \geq L_{\text{excess}}(\mathbf{C}_i) \\ \{\text{selection}(\mathbf{C}) \mid \mathbf{C} \in \text{children}(\mathbf{C}_i)\} & \text{if } S(\mathbf{C}_i) < L_{\text{excess}}(\mathbf{C}_i) \end{cases}$$

where $\text{children}(\mathbf{C}_i)$ is the set of valid clusters which are formed from the set of objects $\{\mathbf{x} | \mathbf{x} \in \mathbf{C}_i\}$ when $\varepsilon < \varepsilon_{\min}(\mathbf{C}_i)$. If $\mathbf{C}_i$ has no children it will always be selected because $L_{\text{excess}}(\mathbf{C}_i) = 0$.

The use of a constant $\varepsilon_{\text{initial}}$ 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 set of child clusters will never be selected in preference of their parent $\mathbf{C}_i$ if $\varepsilon_{\min}(\mathbf{C}_i) < \varepsilon_{\text{initial}}/2$.

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 reason 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).