A non-programmatic description of methods

Tim Millar

2017-07-31

Contents

Overview **1**

Selecting Informative Reads **2**

Fingerprinting **2**
Categorising informative read tips 2
Density based clustering 3
Description of non-hierarchical clustering algorithm 3
Description of hierarchical clustering algorithm 4

Comparing Multiple Fingerprints **6**

Downstream Filtering and Analysis **6**

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 strand and user defined taxonomic categories e.g. 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.

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 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 (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.

Description of non-hierarchical clustering algorithm

We present a novel algorithm for density based clustering of univariate points with noise. The basic version of our algorithm is derived from DBSCAN. As in DBSCAN, a specific density of objects is targeted by the minimum number of points (objects) required to form a cluster m_{pts} and a value epsilon ε which limits the dispersion of those objects. Unlike DBSCAN our method has a stricter criteria of the object density required to form a cluster resulting in 'tighter' clusters. Here we use the following definitions:

1. An set of (at least) m_{pts} objects (i.e. read tips), $\{\mathbf{x}_p, ..., \mathbf{x}_{p+m_{\text{pts}}}\}$ are labeled as *core objects* w.r.t. ε and m_{pts} if they are all mutually within ε range of one another. An object that is not a *core object* is a *noise object*.

2. Two *core* objects $\mathbf{x}_p$ and $\mathbf{x}_q$ are *ε-reachable* w.r.t. ε if they are within ε range of one another.
3. Two *core* objects $\mathbf{x}_p$ and $\mathbf{x}_q$ are *density-connected* w.r.t. ε if they are directly or transitively ε-reachable.
4. A *cluster* $\mathbf{C}$ w.r.t. ε and m_{pts} is a non-empty maximal subset of the set of objects $\mathbf{X}$ such that every pair of objects in $\mathbf{C}$ is density connected.
5. 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 set of density connected objects (is divided into child clusters).
6. The *maximum epsilon* of a cluster $\varepsilon_{\max}(\mathbf{C})$ is the value above which the set of core objects within that cluster become density-connected with the set(s) of core objects in one or more previouly seperate clusters (is incorporated into a larger cluster).
7. The *core distance* of an object $d_{\text{core}}(\mathbf{x}_p)$ w.r.t. m_{pts} is the maximum distance between *any* two objects in the set of objects comprising $\mathbf{x}_p$ and it's $m_{\text{pts}} - 1$ nearest neighbours.

Definitions 2-6 are identical to those given for DBSCAN* by Campello *et al.* (2015). Definitions 1 and 7 differ to those given by Campello *et al.* (2015) in that they require that *every pair of objects* in a set of m_{pts} objects to be mutually within ε range of one another in order to be clasified as core points and thus form a cluster. This differs from DBSCAN in which a single single core object with $m_{\text{pts}} - 1$ border objects can form a cluster in which not all objects are mutually ε-reachable. Thus our method has a stricter criteria of the object density required to form a cluster. Unlike DBSCAN* our method still guarantees that all clusters contain at least m_{pts} objects.

The parameters required by our method can be intuatively infered for 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 resonably estimated as 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.

Description of hierarchical clustering algorithm

We present a hierarchical version of our algorithm is loosely derived from HDBSCAN but produces clusters that are mostly consistent with the non-hierarchical version of our algorithm with the addition of some flexibility to split poorly supported clusters into more strongly supported sub-clusters.

The hierarchical version of our algorithm requires two parameters; m_{pts} as described in the non-hierarchical version and a global maximum epsilon $\mathcal{E}$.

Initial clusters are identified as in the non-hierarchical version using a density defined m_{pts} and $\varepsilon = \mathcal{E}$. Thus the initial clusters are identical to those found

by the non-hierarchical version version of our algorithm. 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 $\mathcal{E}$ 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

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

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

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

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 $\mathcal{E}$ 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 set of child clusters will never be selected in preference of their parent $\mathbf{C}_i$ if $\varepsilon_{\min}(\mathbf{C}_i) < \mathcal{E}/2$.

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 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).