Knowledge-Based Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data

Givanna H. Putri [a,*], Mark N. Read [b,d], Irena Koprinska [a], Deeksha Singh [a], Uwe Röhm [a], Thomas M. Ashhurst [c,d,e], Nicholas J.C. King [c,d,e]

[a] School of Computer Science, University of Sydney, Sydney NSW 2006, Australia
[b] School of Chemical and Biomolecular Engineering, University of Sydney, Sydney NSW 2006, Australia
[c] Sydney Medical School, Discipline of Pathology, University of Sydney, Sydney NSW 2006, Australia
[d] Charles Perkins Centre, University of Sydney, Sydney NSW 2006, Australia
[e] Sydney Cytometry Facility, University of Sydney, Sydney NSW 2006, Australia

ARTICLE INFO

Article history:
Received 9 October 2018
Received in revised form 17 January 2019
Accepted 15 February 2019
Available online xxxx

Keywords:
Density based clustering
Data mining
Temporal cluster tracking
Cytometry
Immunology
West Nile virus
Bioinformatics
Exploratory data analysis

ABSTRACT

In many scientific disciplines, the advent of new high-throughput technologies is giving rise to vast quantities of high-dimensional time-series data. A common requirement is to identify clusters of data-points with similar characteristics in this experimental data, and track their development over time. In this article we present *ChronoClust*, a novel density-based clustering algorithm for processing a time-series of discrete datasets, generating arbitrarily shaped clusters, and explicitly tracking their temporal evolution. We provide a conceptualisation of ChronoClust's parameters, and guidelines for selecting their values. The development of ChronoClust was motivated by the need to characterise the immune response to disease. As such, we demonstrate and evaluate ChronoClust's operation on two immune-related datasets: (1) a synthetic dataset exhibiting the temporal evolution qualities of the immune response as they would be observed through mass cytometry, a cutting edge high-throughput technology, and (2) a Flow cytometry dataset capturing the immune response in West Nile Virus (WNV)-infected mice. Our comprehensive qualitative and quantitative analyses confirm ChronoClust's suitability for this type of problem: the temporal relationships engineered into the synthetic dataset are successfully recovered, and the cell populations and dynamics unveiled in the WNV dataset match those identified through a domain expert. ChronoClust is applicable beyond Immunology, and we provide an open source Python implementation to support its adoption more widely. We additionally make our two datasets publicly available to promote reproducible research and third-party work on temporal clustering and cluster tracking.

1. Introduction

Many scientific disciplines, such as the Life Sciences, are currently transforming into more data-centric fields, driven by advances in high-throughput technologies that generate vast quantities of experimental data at unprecedented low costs [1]. Flow (and mass) cytometry represents one such technology, characterising each individual cell in a sample of millions in terms of shape, structure, and protein expression [2]. With the recent advent of mass cytometry, around 50 different characteristics of a cell can now be simultaneously measured [3,4]. Whilst poised to revolutionise our understanding of biological systems, the availability of such vast and high-dimensional data also raises novel analytical challenges, in particular the need for effective and efficient methods to mine this data and generate understanding [5]. Most high-throughput technologies, such as those enabling the sequencing of genes, proteins, metabolites and cell populations, take snap-shots in time. However, the living systems they probe are inherently dynamic. Whilst the extraction of meaning from these individual snap-shots encompasses a vibrant area of research in its own right, linking these datasets to reveal temporal patterns represents a further compounding challenge.

Our present work concerns understanding the development of the immune response in disease, as it would be characterised through a time-series of cytometry datasets. Our immune systems are integral to our health, protecting us from pathogens such as viruses and bacteria, and tumours arising from abnormal cell growth. Pathogens can present in a large diversity of forms,

* Corresponding author.
E-mail addresses: ghar1821@sydney.edu.au (G.H. Putri), mark.read@sydney.edu.au (M.N. Read), irena.koprinska@sydney.edu.au (I. Koprinska), dsin5991@sydney.edu.au (D. Singh), uwe.roehm@sydney.edu.au (U. Röhm), thomas.ashhurst@sydney.edu.au (T.M. Ashhurst), nicholas.king@sydney.edu.au (N.J.C. King).

https://doi.org/10.1016/j.knosys.2019.02.018
0950-7051/

and our immune system can deploy a wide range of nuanced responses specific for the given challenge. However, the immune response is not always successful. The outcomes to a given immune trigger vary widely in patients, and even in genetically identical animals under laboratory conditions [6].

To design effective health-promoting interventions, we must understand how immune actions are deployed. The task is complicated by the immune system's complexity, comprising a vast number of functionally distinct cells that cooperate dynamically over time across numerous bodily organs. The plethora of specialised cells responding to a given immune challenge are generated on-demand from common progenitor cells, typically over 10 days. This progressive specialisation in various immune cell populations is termed *differentiation*. It encompasses several cell developmental stages, with divergent options branching from each [3,7]. Cells at each developmental stage differ in their morphology and protein expressions, and these differences are detectable through cytometry. Whilst broad categories of immune cells are well known (e.g. granulocytes, lymphocytes, monocytes), new sub-populations within them with nuanced functional differences are continually discovered. These new sub-populations form a small proportion of all cells found, but often exert considerable influence in shaping the overall immune response.

Mapping out the specific times and locations at which differentiation pathways are deployed in a given disease is the key to understanding immune function and developing effective interventions. Yet, the traditional methods to do this are manual and require advanced immunology expertise. For example, Fig. 1 illustrates the *gating* method, where an expert examines two-dimensional cytometry scatter plots and manually identifies clusters of cell populations. For instance, *gate* B denotes cells high in Ly6C and low in CD11b proteins. The cells included in this gate could then be examined independently from the remainder of the dataset, further gated into additional sub-populations using two-dimensional plots of other cell characteristics. This process is time consuming, tedious and subjective. It is also clearly intractable when applied to 50-dimensional data, where 1225 such 2D scatter plots are possible.

From a data mining perspective, the task is *clustering* of cytometry data and identifying the characteristics of each cluster. Whilst automated approaches for clustering and profiling of cytometry data have recently been developed [3,8], they are applicable to single time-points only. A comprehensive understanding of the immune response in a given disease, and how it differs between diseases, requires analytical methods that *automatically identify cell populations* in high-dimensional space, and *track their evolution over time*. As we show in Section 2, no single existing clustering algorithm can solve both tasks simultaneously. As a solution, we introduce here *ChronoClust*, a novel clustering and cluster tracking algorithm we have developed which satisfies these requirements.

We firstly evaluate ChronoClust's performance on a synthetic dataset wherein clusters evolve over time, and demonstrate its ability to track the sub-clusters therein. Then, we demonstrate ChronoClust's ability to identify and track immune cell populations in a real cytometry dataset, capturing the immune response of mice infected by the West Nile Virus (WNV).

The main contributions of this paper are as follows:

1. We present the novel problem of clustering and tracking cluster evolution, motivated by a use-case in the Life Sciences. We derive the requirements for a clustering solution, and through a literature review find no single solution satisfying all these requirements.
2. We present a new density-based dynamic clustering algorithm, ChronoClust, for the detection of clusters in high-dimensional time-series data, and tracking their splitting, merging and *de novo* creation over time. The tracking is based on two methods: tracking by *lineage determination* and tracking by *historical proximity*.
3. We conduct a qualitative and quantitative evaluation of ChronoClust's performance on a synthetic dataset, designed to capture the characteristics of an immune response as observed through cytometry data.
4. We evaluate ChronoClust on a real cytometry dataset, which characterises the immune response to WNV-infected mice. Furthermore, we demonstrate ChronoClust's clustering performance to be superior to that of FlowSOM, the leading automated cytometry clustering tool. Importantly, we verify ChronoClust's tracking of cellular differentiation pathways against expert knowledge; FlowSOM does not include this automated functionality.
5. We provide a conceptualisation of ChronoClust's parameters that guides selection of their values.
6. We make our ChronoClust implementation publicly available, alongside both datasets, in the interest of supporting reproducible research and the application of ChronoClust to other problems: https://ghar1821.github.io/Chronoclust/,

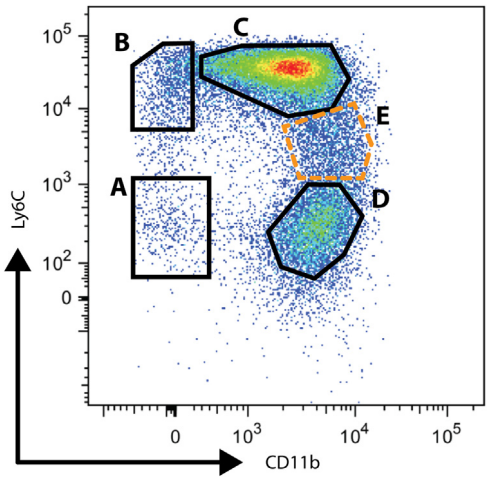

Fig. 1. An example of cytometry data, shown as a two-dimensional density scatter plot, demonstrating the expert knowledge and subjectivity involved in a conventional analysis. Each data-point represents an individual immune cell, and its expression of the cell-surface proteins CD11b and Ly6C. Clusters (gates) of similar cells are delineated by an expert performing *manual gating*. Clusters are given immunologically-relevant labels, e.g.: A, granulocyte–monocyte progenitors; B, monoblasts; C, mature monocytes Ly6C+; D, mature monocytes Ly6C-. Clusters A, C and D are identified based on their high density. Population B is identified as a separate cluster, rather than an extension of C, due to its immunological relevance. Region E is not formally defined as a population of interest, as these are cells deemed to be in a continuum flux between two well-defined mature states.
Source: Adapted from [9].

The remainder of the paper is structured as follows. In the next section we set out the requirements for the clustering solution and provide an overview of related work. In Section 3 we describe ChronoClust, our novel clustering and cluster tracking algorithm. In Section 4 we present an evaluation of ChronoClust's performance against a well-understood synthetic dataset. Thereafter, in Section 5 we apply ChronoClust to a real cytometry dataset of WNV-infected mice. Section 6 concludes the paper.

2. Review of existing clustering approaches

From our cytometry use-case, we derive the following requirements for a clustering solution:

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

1. No pre-specified number of clusters, as this is unknown for a given immune response, and we wish our solution to elucidate novel cell populations in the data.
2. Graceful handling of outlier and noisy data; biological systems are notoriously stochastic and noisy, as exemplified in Fig. 1.
3. Effective in high dimensions; cytometry can now generate datasets of over 50 dimensions.
4. Allow clusters to form arbitrary shapes; we do not anticipate that clusters in 50 or more dimensions would present only as ellipsoids.
5. Facilitate dimensionality reduction or sub-space clustering. In exploratory experimentation, it cannot be guaranteed that all dimensions will be equally important. Further, we anticipate that the hierarchical nature of cellular differentiation will culminate in sub-clusters that are well-separable in subsets of dimensions.
6. Track clusters' temporal evolution, wherein they branch and merge with one another over time, or when *de novo* clusters are created. This facilitates tracking of differentiation pathways exhibited by cell populations branching from their predecessors.

We now review existing cytometry-specific and general-purpose clustering and cluster tracking approaches, taking stock of our use-case requirements. For general purpose algorithms we focus on those wherein the number of clusters is not predetermined. For a more in-depth review of high-dimensional clustering algorithms, please see [10].

2.1. Clustering in cytometry analysis

A number of algorithms have been designed for the analysis of cytometry data, but none accommodate time-series data and track cellular differentiation simultaneously. One of the most popular is SPADE [8]. It employs agglomerative hierarchical clustering to group the data and, thereafter, a minimum spanning tree to connect clusters of similar cells. It also provides visualisation of the resulting tree. Qiu et al. [8] presented an application of SPADE to cytometry data of mouse and human bone marrow, wherein SPADE revealed functionally distinct cell populations. Bendall et al. [3] successfully applied SPADE to 34-dimensional bone marrow cytometry data, demonstrating accommodation of high-dimensional data.

The dimensionality reduction and visualisation tool viSNE [11] has been used to generate two-dimensional maps of cell populations, whilst preserving high-dimensional relationships. Specifically, viSNE uses the t-SNE optimisation algorithm [12,13] to find a mapping from the original high-dimensional space to a new two-dimensional space, which best preserves the pairwise distances between the cells. viSNE was used to explore cytometry data from healthy and leukemic human bone marrow, showing the effectiveness of the tool [11].

Another dimensionality reduction and visualisation tool, FlowSOM [14], uses a Self-Organising Map (SOM) to cluster data, a minimal spanning tree to visualise the SOM nodes, and a meta-clustering of the SOM nodes to find the cluster boundaries. Guiliams et al. [15] verified the effectiveness of FlowSOM to find a common set of characteristics defining dendritic cells in mice, macaques and humans, by comparing its performance with manual gating.

Importantly, whilst these specialised techniques are clearly effective for single time-point datasets, they cannot accommodate time-series data to reveal temporal differentiation patterns, as we require. Zunder et al. developed FLOW-MAP as an algorithm to elucidate temporal cellular developmental dynamics and used it to investigate fibroblast reprogramming [16]. Importantly, FLOW-MAP is not a clustering algorithm; it instead relies on a third-party clustering solution, such as SPADE. Each time-point's dataset is clustered independently, and FLOW-MAP then performs a *post-hoc* linking of all those of close spatial proximity over time. Given the independent application of clustering to each time-point, this solution does not perform tracking; it cannot formally detect cluster splitting, merging, disappearance or *de novo* creation events as this would require clusters or their states to persist across time-points. Such formal tracking is particularly important for analysing the immune response in peripheral tissues where many cell populations arrive at varying times following early differentiation events elsewhere.

Finding no cytometry-specific solution to our use-case, we now consider more general clustering approaches and evaluate them against our requirements in summary Section 2.5.

2.2. Vector quantisation-based clustering algorithms

Addressing the emerging need to summarise temporal online data, where the algorithm must adapt to novel data as it becomes available, Lughofer's incremental clustering algorithm, the Evolving Cluster Model (ECM) [17,18], extended the classical Vector Quantisation method [19]. The first extension includes the use of a *vigilance parameter* to control the trade-off between the adaptation of existing clusters versus the creation of new clusters. If a new data-point is sufficiently close to the centroid of the nearest existing cluster, as defined by the user-specified vigilance parameter, it is assigned to this cluster and the cluster centroid is shifted towards the data-point; otherwise, a new cluster containing the data-point is formed. By incorporating the spread of data around cluster centres into Euclidean distance calculations, ECM accommodates ellipsoid cluster shapes, though not fully arbitrary shapes. A second extension includes a *split and merge* strategy to deal with a non-optimal clustering structure arising from the order in which the data-points arrive. For example, an early arrival of data at the extremes of what would ideally be captured as a single cluster can instead result in several smaller clusters that should be merged (over-clustering). On the other hand, a relatively sparse and large single cluster may be followed by data-points forming two areas of high density; the cluster should be split to prevent under-clustering. To deal with these issues, ECM introduces incremental checks of the clustering quality, and performs merging and splitting accordingly [18].

In [20] Lughofer and Sayed-Mouchaweh further refined the splitting and merging strategy, and also substituted ECM's Euclidean distance with Mahalanobis distance, which incorporates data distributions along a cluster's principle axes, derived from the data it captures. The resultant eVQ-AMS algorithm accommodates ellipsoid cluster shapes with arbitrary orientations, therein providing a better capture of data for practical applications.

As a means to remove "outlier" clusters deemed to represent noise, Lughofer augmented ECM with *satellite deletion* (though, surprisingly, did not carry this extension into eVQ-AMS) [17]. This filtering stage is performed post-clustering, or at given times or periodicity during online clustering. A cluster is removed if it captures less than a minimum threshold portion of data (e.g. 1%), if its centroid lies within another cluster's hyper-volume, or if its volume is substantially less than its neighbouring clusters.

2.3. Density based clustering algorithms

The landmark DBSCAN algorithm strove to accommodate a non-predefined number of arbitrarily shaped clusters whilst requiring few parameters and being computationally efficient [21]. It seeks to find regions of high density that are separated by

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

regions with low density. The density of a point is defined by the number of points surrounding it within a radius ϵ. Using a minimum density threshold *MinPts*, each point is labelled as forming a cluster's *core*, *border*, or *noise*. Core points that are sufficiently close are put in the same cluster, as are border points that are sufficiently close to a core point from this cluster. A procedure for selecting the critical parameters ϵ and *MinPts* from a graph of descending distances to each data-point's kth furthest neighbour was also proposed.

The Density Peaks Clustering (DPC) algorithm [22] is a recently proposed alternative to DBSCAN. It is based on the assumption that cluster centres (i) have higher density than their neighbours and (ii) are located sufficiently far from other points with higher local density. DPC computes two properties for each data point: local density and distance to points with higher local density. It visualises them on a decision graph which is used to determine the cluster centres and the outliers, while the remaining points are assigned to the cluster of the closest centre. Although DPC is easy to implement and has shown good results in several applications, it is computationally expensive for large and high-dimensional data. A number of extensions have been proposed to address this limitation, e.g. using grid and circle division to more efficiently find the cluster centres [23], computing the local density of a point by only considering its k nearest neighbours [24,25] or the grid density [26], and using principal component analysis to reduce the dimensionality of the data [24]. An adaptive version with an improved selection of initial clusters and cluster aggregation has been proposed in [25].

Density-based algorithms for *data stream* contexts have also been proposed. The IncrementalDBSCAN [27] is an extension of DBSCAN for data streams, where new data-points are inserted and old data-points are forgotten. DBSCAN was deemed promising for data streams due to its density-based nature — the insertion or deletion of a data-point will affect only the clusters around this data-point. This allows for the development of a computationally efficient algorithm for handling the splitting and merging of clusters.

However, if a considerable proportion of data-points move between consecutive time-points, IncremenalDBSCAN's cluster update procedure becomes computationally expensive. To address this, Kalnis et al. [28] proposed three algorithms for detecting moving clusters based on DBSCAN. The first algorithm applies DBSCAN to each time-point and then merges clusters from consecutive time-points based on their similarity to form moving clusters. The similarity measure is based on the intersection and union of the data-points in the clusters. The second algorithm improves on the efficiency of the first by minimising redundant comparisons between data-points and moving clusters, while the third algorithm further improves the speed (at the expense of lower accuracy) by only applying DBSCAN to select data-points to produce approximate clusters.

It can be impractical to represent clusters through their constituent data-points in online contexts as the data is potentially limitless, yet computational memory is finite. DenStream [29] addresses this challenge through a two-tier clustering approach consisting of *online* and *offline* phases. The concept of dividing the clustering process into online and offline passes was first introduced in the CluStream framework [30]. The fast online phase stores summary statistics of the data in the form of *Micro-Clusters* (MCs) while the offline phase uses this information and user inputs, such as the required time horizon and granularity of clustering, to produce the final clusters. The application of CluStream was demonstrated by using a k-means variant as the base algorithm, which has limitations in terms of pre-specified number of clusters, dealing with outliers, and forming spherical clusters. DenStream applied and extended the CluStream framework for density-based clustering. The *online* phase summarises the data-points through an evolving collection of MCs with constrained maximum radius. Novel data items are merged with the nearest MC, provided that the MC's radius remains compliant. The counts of the data-points captured by MCs are periodically decayed, low-density MCs representing outlier data are deleted and MCs replenished by arrival of supporting data in the stream are maintained. During the *offline* phase, executed periodically or on demand, DenStream employs a DBSCAN variant to cluster the highest density MCs. The result is a memory efficient algorithm that can form arbitrarily shaped clusters that adapt to reflect the changing nature of the data stream.

D-Stream [31] is another algorithm employing this two tier clustering approach. Its online phase is slightly different to that of DenStream's as it summarises data-points using a collection of *grid cells* instead of MCs. Grid cells represent a discretisation of the feature space; the density of a grid cell is determined by the number of data-points mapped to it. Further, D-Stream employs a decay factor to reduce the grid cells' densities over time and capture changes in the data stream. Adjoining dense cells are linked together to form clusters. While D-Stream has been tested on both synthetic and real datasets, it can be computationally expensive if applied to high dimensional datasets due to the need to maintain a large number of cells.

An emerging theme in clustering is *subspace* clustering, meaning that some data-points might exhibit a narrow distribution, and hence a tight clustering, in only a subset of all dimensions [10]. The PreDeCon algorithm [32] expands DBSCAN's density-based clustering concept to accommodate subspace clustering by adopting a weighted distance metric. Dimensions in which a data-point's neighbourhood exhibits low variance contribute a higher weight in calculating the distance to other points. As with DBSCAN, PreDeCon also operates over static datasets.

Mirroring DenStream's online and offline two-phase operation, HDDStream [33] adapts subspace clustering to data streams. As with DenStream, the MC-captured data-point counts are periodically decayed, allowing adaptation to new changing data and forgetting of old data. The summarising of input data via MCs helps in ensuring a relatively modest memory footprint. Weighted distance calculations are adopted to provide for subspace clustering, reminiscent of PreDeCon. In fact, PreDeCon is employed in HDDStream's offline phase to assemble final clusters from high density MCs.

2.4. Cluster tracking algorithms

MONIC is a framework for tracking cluster transitions over time, developed by Spiliopoulou et al. [34]. It firstly finds the "same" clusters in different time-points. To find the "same" cluster C in a later time-point, MONIC computes the overlap between cluster C in the current time-point and clusters in the later time-points. The overlap is based on both the set intersection and "age" of the data-points. MONIC then detects different types of *external* and *internal* cluster transitions. External transitions include cluster survival, splitting, absorption, disappearance and the emergence of a new cluster. The internal transitions are only applicable to "surviving" clusters and include changes in size, compactness and location. A measure of cluster lifetime and stability over time was also proposed. MONIC's ability to track clusters was demonstrated in a case study using documents from the ACM digital library.

MEC [35] is another prominent method for monitoring cluster evolution. It extends MONIC by representing clusters through summary statistics, such as density, centroid locations and radius (*comprehension*), in addition to MONIC's representation of clusters via all constituent data-points (*enumeration*). Two tracking methods are proposed: using conditional probabilities to construct

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

weighted bipartite graphs for enumerated clusters, and using cluster overlap for clusters represented by comprehension. MEC was applied to two case studies utilising economic and educational data from Portugal; the results showed that it was able to effectively track cluster changes.

2.5. Summary

As discussed above, the existing cytometry-specific approaches do not meet our use-case. Of the general-purpose clustering algorithms, vector quantisation-based methods do not permit arbitrarily-shaped clusters and do not provide temporal cluster tracking. The density-based algorithms we reviewed, culminating in HDDStream, do not explicitly support tracking, though the data-stream variants might be applicable to processing a time-series of discrete datasets as per our use-case.

Elsewhere, MONIC and MEC do provide temporal tracking of clusters, but do not facilitate clustering. Additionally, the type of tracking they provide cannot determine the predecessors of newly emerged clusters. This is an important requirement for our use-case as it is biologically impossible for an immune cell not to differentiate from a progenitor. Hematopoietic stem cells are the only exception, as they give rise to all immune cells. There is a clear opportunity to develop a single algorithm capable of both clustering and tracking and to optimise its performance to biological use-cases, facilitating adoption by biologists.

In the next section, we present ChronoClust, a novel clustering and cluster tracking algorithm that we have developed, which satisfies all the requirements for our use-case from Section 2.

3. ChronoClust

ChronoClust is a density-based clustering algorithm suitable for analysing a time-series of datasets and explicitly tracking the evolution of clusters across time-points. A one-pass batch processing of each dataset is performed. It builds upon HDDStream [33]; in particular, it retains HDDStream's two phase operation (online and offline), with some modifications, and also introduces a third phase: tracking.

First, during the online phase, ChronoClust summarises a time-point's raw data as Microclusters (MCs) of constrained radius. Second, during the offline phase, it clusters these MCs into arbitrary shapes by applying the PreDeCon [32] algorithm to each time-point. Third, it generates a history of clusters' temporal evolution through two methods:

1. *Tracking by lineage determination* operates by identifying the MCs that are shared between clusters from two consecutive time-points. This method reveals when existing clusters merge and split, who their *parents* are, and when new clusters emerge. The lineage history is encoded into each cluster's unique ID.
2. *Tracking by historical proximity* links clusters from two consecutive time-points based on the proximity of their constituent MCs.

Tracking by lineage determination provides a formal and absolute evolutionary history, based on the behaviour of MCs that persist across time-points. However, it cannot provide ancestry for *de novo* clusters as, by definition, their MCs are not shared with any cluster from the previous time-step. By instead estimating ancestry, tracking by historical proximity serves as a contingency. Cluster ancestry is important to our use-case. These two complementary tracking strategies confer ChronoClust several advantages. MONIC and MEC cannot estimate ancestry for *de novo* clusters, whereas ChronoClust's tracking by historical proximity can. FLOW-MAP merely links all nearby clusters across time and cannot formally label cluster splitting, merging or *de novo* creation events as ChronoClust's tracking by lineage determination does.

The pseudocode of ChronoClust is presented in Algorithms 1, 2 and 3, and discussed next.

Algorithm 1: ChronoClust overview.
$\{x, y\}$: an ordered array of x preceding y. $\oslash$: an empty array.
$array[z]$: extracts item(s) from the array corresponding to index(s) z; z may express a condition.

Input: $\boldsymbol{D} = \{D_0, ..., D_t\}$ ▷ D_i indicates dataset at time-point i.
Output: ***all_clusters*** $= \{all\_clusters_0, ..., all\_clusters_t\}$ ▷ $all\_clusters_i$ indicates clusters for time-point i.

```
1:  function ChronoClust(D)
2:      D_norm ← NormaliseData(D)   ▷ Normalise all input data together, not each D_i dataset in isolation
3:      mcs ← ⊘                     ▷ Store microclusters
4:      clusters_{t-1} ← ⊘          ▷ Store offline phase clusters
5:      all_clusters ← ⊘            ▷ Collate clusters for output
6:      for D_t ∈ D_norm do
7:          if t > 0 then           ▷ Decay and downgrade microclusters for all but the first time-point
8:              decay mc weight, for mc ∈ mcs
9:              downgrade mc, for mc ∈ mcs
10:         ▷ Online & offline clustering similarly to HDDStream
11:         mcs ← OnlineMicroclustering(D_t, mcs)
12:         clusters_t ← OfflineClustering(mcs)
13:         TrackClusters(clusters_t, clusters_{t-1}) ▷ See Algorithm 2
14:         clusters_{t-1} ← clusters_t
15:         all_clusters[t] ← clusters_t
        return all_clusters
```

3.1. Clustering: Online and offline phases

ChronoClust commences with an *online* phase which summarises the raw data-points as MCs; an MC's *weight* reflects the number of data-points it captures. One time-point is processed at a time, and the resulting MCs persist across time-points. To permit MCs to evolve and reflect changes in the data over time, MC weights are decayed after each time-point. Those that are reinforced by data in the subsequent time-point(s) persist and evolve, while others are eventually decayed and deleted, thus allowing old data to be gradually forgotten. The online phase is followed by an *offline* phase which groups the MCs into clusters with arbitrary shapes, forming the final clustering.

We now explore ChronoClust's online and offline phases in more detail. We firstly introduce the notation used in Algorithms 1, 2 and 3. We define the input, a time-series dataset of t time-points, as $\boldsymbol{D} = \{D_0, \ldots, D_t\}$. Each dataset D_i is a table indexable by features and entries: $D_i(f, e_i)$, where $f \in \{F_0, \ldots, F_n\}$ is a set of n features, and $e_i \in \{E_0, \ldots, E_j\}$ is the set of j entries at time-point i.

ChronoClust commences by rescaling each feature f to the range $[0, 1]$, based on the minimum and maximum values observed for f across all D_i. The normalisation step facilitates parameter value selection and conceptualisation as discussed in Section 3.3. Then, ChronoClust proceeds by clustering the data, similarly to the HDDStream algorithm. There are four main differences between HDDStream and ChronoClust, discussed below: (1) in the MC initialisation phase, (2) the weight decay of MCs, (3) outlier MC deletion and (4) the distance measure used to join MCs in the offline phase, as discussed in Section 3.3.

While HDDStream's initialisation phase seeds the initial MCs using the earliest data from the continuous data stream, such

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Table 1
A snippet of ChronoClust output, showing clusters (rows) across five time-points (days) when fine-scale clustering the **synthetic dataset**, described in Section 4.1. *Weight* is a decaying cumulative count of the data-points a cluster has captured. *MC IDs* gives the cluster's constituent MC IDs. *Cluster centroids* presents the centroids of the clusters; this is shown for convenience only. The tracking phase is responsible for determining the *Cluster IDs* (using *tracking by lineage determination*) and the *Cluster associates* (using *tracking by historical proximity*). *Cluster ID* conveys a full, unique developmental history for each cluster, on the basis of MCs shared across time. *Cluster associates* of a cluster in the current time-point are the nearest clusters from the previous time-point, and they are not required to share constituent MCs.

Day	Weight	Constituent MC IDs	Cluster centroids			Tracking by Lineage Determination	Tracking by Historical Proximity
			X	Y	Z	Cluster ID	Cluster Associates
1	2019.0	1	29.9	29.9	29.9	A	None
1	5046.0	0	10.1	10.0	10.1	B	None
2	2044.8	1, 3	29.1	29.9	30.0	A	A
2	465.0	2	34.2	29.9	30.0	C	A
2	6264.5	0, 4	10.4	10.3	10.3	B	B
3	706.5	3	25.6	30.1	30.0	A	A
3	1194.7	1	30.0	29.9	30.0	A|1	A
3	672.3	2	34.3	30.0	30.0	C	C
3	6583.6	0, 4, 5	10.6	10.5	10.5	B	B
4	1163.5	9, 3	23.4	30.0	29.9	A	A
4	1478.7	1, 2	32.5	29.9	29.9	(A|1,C)	A|1 & C
4	4455.0	0, 8, 4	10.3	11.3	10.0	B	B
4	1327.0	5, 7	10.0	10.0	14.5	B|1	B
4	855.1	6	15.5	10.1	10.0	D	B
5	1296.9	9, 3	21.5	30.0	30.0	A	A
5	1365.7	1, 2	34.3	30.0	30.0	(A|1,C)	(A|1,C)
5	3162.5	0, 4	10.1	10.4	9.9	B	B
5	531.5	5	10.0	9.9	13.4	B|1	B|1
5	544.3	7	9.7	10.0	16.9	B|1|1	B|1
5	531.3	8	10.1	16.3	10.1	B|2	B
5	763.8	6	15.7	10.1	9.9	D	D
5	324.0	10	5.6	10.1	18.1	E	B|1
5	404.3	11, 14	18.1	6.2	10.0	F	D
5	415.5	12, 13	10.0	18.1	6.3	G	B

Tracking notation key:

Notation	Meaning
(X, Y)	Merging of clusters X and Y
X|2	A cluster that split from X, and the second to do so (X|1 being the first)
X & Y	X and Y are both historical associates

a bespoke initialisation is unnecessary in ChronoClust, as the entire first time-point's dataset (indeed, all time-points' datasets) is available *a priori*. Hence, ChronoClust commences online clustering without an initialisation phase.

OnlineMicroclustering summarises the given time-point's raw data as MCs. It increases an MC's *weight* by 1 for each data-point it captures. MCs are classified into three categories based on their weights using decreasing thresholds: *core*-MCs, *potential core* (*pcore*)-MCs or *outlier*-MCs. In the offline clustering phase, core- and pcore-MCs form clusters for that time-point. Core-MCs serve as cluster seeds, from which nearby pcore-MCs are "daisy chained" to form clusters of arbitrary shapes. We refer readers to [32,33] for further details.

To allow MCs to adapt to new data in the stream, their weights are decreased over time t through multiplication by a factor $d = 2^{-\lambda t}$ with $\lambda > 0$ giving $0 < d \leq 1$. Larger values for the decay rate λ decrease the influence of historical data on an MC's current state, including its location and spread. Whereas HDDStream decays MCs periodically to accommodate the continuous and possibly sporadic arrival of input data, the time intervals between datasets are known in advance for ChronoClust and MCs weights are instead decayed once per time-point before online clustering. MCs whose weights have fallen below category thresholds are *downgraded*, e.g. a decayed pcore-MC becomes an outlier.

Lastly, HDDStream deletes *outlier* MCs based on a heuristic that contrasts the expected with actual weights given the passage of some time, a mechanism well adapted to sporadic data streams. In ChronoClust we instead employ a novel method to determine when outlier MCs ought to be deleted, based on their weight falling below a threshold value. This threshold is determined on the principle that a pcore-MC should not be deleted, given decay over time, before having the opportunity to capture additional data-points at the next time-point. The conceptualisation and equation to calculate the threshold deletion weight are detailed in Section 3.3.

ChronoClust's tracking phase operates over the current and preceding time-point's clusters and is discussed in the following section.

3.2. Tracking phase

Table 1 shows an example tabular output from ChronoClust to illustrate the tracking phase's operation. Each row represents a cluster at a given time-point (days in this example). Cluster centroids are reported for convenience only; to determine the exact location and shape of an arbitrarily-shaped cluster one must inspect each of its MCs. The column *MC IDs* reports the unique IDs of all core- and pcore-MCs constituting the cluster. *Weight* is a cumulative count of data-points the cluster represents, via its MCs, decayed over time as discussed above. The tracking phase includes two modes: *tracking by lineage determination* and *tracking by historical proximity*. The first analyses the shared MCs between all clusters from two consecutive time-points to determine child–parent relationships. It assigns a unique *Cluster ID* to each cluster, which encodes the cluster's temporal evolution. The second determines the cluster's *historical associates*, which are the nearest clusters from the previous time-point, i.e. the most likely predecessors. There is no requirement for a cluster to share MCs with its historical associate(s). We next describe these two tracking modes in more details.

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Algorithm 2: ChronoClust tracking of cluster temporal evolution.
Notation is as in Algorithm 1. Input and output are applicable to both functions.

Input: $clusters_t$ ▷ clusters at time-point t.
$clusters_{t-1}$ ▷ clusters at time-point $t-1$.
Output: Nothing

```
1:  function TrackClusters(clusters_t, clusters_{t-1})
2:      ▷ Arguments are arrays of clusters at two consecutive days
3:      LineageDetermination(clusters_t, clusters_{t-1})
4:      HistoricalProximity(clusters_t, clusters_{t-1})
5:
6:  function LineageDetermination(clusters_t, clusters_{t-1})
7:      ▷ This variable's state must persist across consecutive function calls
8:      ▷ It counts cluster splits (e.g. "B|1") which may occur over several time-points
9:      splits_per_id ← empty array indexed by cluster labels
10:
11:     for cluster ∈ clusters_t do
12:         ▷ Identify clusters from time t − 1 (parents) sharing MCs with cluster
13:         cluster.parents ← ∅
14:         for mc ∈ cluster.mcs do
15:             ▷ If a parent containing mc exists and has not already been recorded
16:             if ∃ parent ∈ clusters_{t-1} | mc ∈ parent.mcs ∧ parent.id ∉ cluster.parents then
17:                 cluster.parents.Append(parent.id)
18:         if cluster.parents = ∅ then                    ▷ New cluster, has no parents
19:             ▷ Handles initialisation at t=0, as clusters_{t-1} is empty
20:             cluster.parents ← {unused label from {"A", ..., "Z", "AA", ...}}
21:
22:         ▷ Collect clusters by their parents' ID(s); used to identify parents that split
23:         offspring ← empty array indexed by cluster IDs
24:         for cluster ∈ clusters_t do
25:             for parent ∈ cluster.parents do
26:                 if parent.id not already an index in offspring then
27:                     offspring[parent.id] ← ∅           ▷ Initialise with empty array
28:                 offspring[parent.id].Append(cluster)   ▷ Record parent–child relationship
29:
30:         ▷ Prepare cluster IDs based on parents'; identify parents that have split
31:         for parent_id, children ∈ offspring do   ▷ Extract both ids (lexicographically) and corresponding arrays
32:             sort children descendingly by number of parent_id's MCs they contain
33:             ▷ The cluster with the majority of parent_id's MCs adopts parent's ID
34:             ▷ An array of id_components is compiled to accommodate several merging parents
35:             children[0].id_components.Append(parent_id)
36:             ▷ Process subsequent children, if any exist; the parent has split
37:             for child_i ∈ children[i > 0] do
38:                 increment splits_per_id[parent_id] by 1
39:                 new_id ← parent_id + "|" + splits_per_id[parent_id]   ▷ String concatenation
40:                 child_i.id_components.Append(new_id)
41:
42:         ▷ Assign IDs; handles merges if a cluster had several parents
43:         for cluster ∈ clusters_t do
44:             ▷ Insert comma between each cluster.id_components below
45:             cluster.id ← "(" + cluster.id_components + ")"   ▷ String concatenation
46:             splits_per_id[cluster.id] ← 0   ▷ Instigate for subsequent time-point
```

3.2.1. Tracking by lineage determination

This mode of tracking is performed by the function LineageDetermination in Algorithm 2. Fig. 5, in particular the two graphs depicting fine-scaled clustering of the synthetic dataset, may serve as an aid to readers as we discuss this function; it graphically depicts the evolution of clusters listed in Table 1. ChronoClust assigns IDs to all clusters in a given time-point, requiring access to the clusters in the preceding time-point. A cluster at time-point t is deemed to have evolved from one at $t-1$ if both share a common MC. We label these clusters *child* and *parent*. Hence, the algorithm first cycles through a child cluster's MCs, seeking parents that possess these same MCs; there may be several.

Constructing a unique ID conveying evolutionary history requires that we accommodate clusters that *split*: parents that contributed MCs to *multiple* children. For instance, in Table 1 and Fig. 5 cluster B in day 3 splits to form two clusters in day 4: B and B|1. Hence, for each parent, we must group together all of its children and count them. The child inheriting the majority of the parent's MCs retains the parent's label, B in this example. The remaining children adopt the parent's label and are incrementally numbered as such, e.g. B|1.

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Algorithm 3: ChronoClust's Tracking by historical proximity algorithm.Notation is as in Algorithm 1.

Input: $clusters_t$ ▷ clusters at time-point t.
$clusters_{t-1}$ ▷ clusters at time-point $t-1$.
Output: Nothing

```
1: function HistoricalProximity(clusters_t, clusters_{t-1})
2:     for cluster ∈ clusters_t do                          ▷ Initialise predecessors variable
3:         cluster.predecessors ← ⊘
4:     ▷ Assemble all constituent MCs from all clusters in previous time-point; used below
5:     all_mcs_{t-1} ← {constituent ∈ c.mcs | ∀c ∈ clusters_{t-1}}
6:     ▷ Handles t=0 and other times when there are no clusters at t-1 (clusters_{t-1} is empty)
7:     if all_mcs_{t-1} = ⊘ then return
8:
9:     for cluster ∈ clusters_t do                          ▷ Find predecessors for each current cluster
10:        for mc ∈ cluster.mcs do
11:            ▷ Find constituent MC from previous time-point nearest to mc
12:            nearest_mc ← argmin_{prev ∈ all_mcs_{t-1}} dist^Φ(mc, prev)   ▷ See Equation 1
13:            ▷ Find the previous time-point cluster that nearest_mc belonged to
14:            pred ← clu | nearest_mc ∈ clu.mcs ∧ clu ∈ clusters_{t-1}
15:            if pred.id ∉ cluster.predecessors then       ▷ Add if not already recorded
16:                cluster.predecessors.Append(pred.id)
```

The last stage of this algorithm deals with *merges*, where several parents contributed MCs to the same child. In our notation, this is represented with brackets and commas e.g. (A|1,C) indicates the merging of clusters A|1 and C. The child's unique ID is constructed from a concatenation of all its parents' IDs, with some string manipulation for commas and parentheses. Having given an abstract overview of its operation, we now explore function LineageDetermination in more detail.

We require each cluster to maintain a reference to the MCs it comprises, denoted by the ordered set *x.mcs* for cluster *x*. The determination of a cluster's parents is handled by line 14 of Algorithm 2. If a cluster has no parents, line 18, then it is a novel cluster. This occurs for the first time-point, as $clusters_{t-1}$ is empty, or when the cluster comprises MCs either newly created or not constituting a cluster in the previous time-point. The novel cluster is assigned an unused alphabetical ID as its sole parent; subsequent code will assign this as the cluster's ID. For example, cluster C in day 2 (see Table 1) contains MC # 2, which does not exist in day 1. With labels A and B taken, the new cluster is assigned ID C.

The block commencing at line 23 maps a parent to all of its children, needed for subsequent detection of parents that have split. Line 31 then handles split detection. Each parent is processed in turn, along with its children. The child inheriting the majority of the parent's MCs adopts that parent's ID, unmodified. Subsequent children, should there be any, adopt a modified parent ID, employing a "parent ID|split#" notation. The *splits_per_id* variable facilitates this labelling by maintaining a cumulative count of splits each cluster has made; this variable's state must persist across LineageDetermination invocations.

Lastly, cluster IDs are assigned as a concatenation of (potentially modified, for splits) parent IDs, in line 43. For example, cluster (A|1,C) in day 4 (see Table 1) contains MCs # 1, 2 that belong to clusters A|1 and C from day 3. The parentheses are necessary to convey an unambiguous history, which is essential when multiple merging and splitting events have occurred over time.

We note that for datasets with many time-points the cluster IDs can become difficult for humans to parse; they are designed to be machine-interpretable to facilitate graphical user interface-based analysis in future work. We have incorporated a simple mapping of Cluster ID to problem specific labels that can aid human interpretation. Table 2 presents a hypothetical illustration for immune cell lineages.

Table 2
Mapping of clusters and populations. Note that this is a purely hypothetical example.

Day	Cluster Id	Name
1	A	Common myeloid progenitors
1	B	Common lymphoid progenitors
2	A	Common myeloid progenitors
2	B	Common lymphoid progenitors
2	C	Monocytes
3	B	Common lymphoid progenitors
3	A	Common myeloid progenitors
3	C	Monocytes
3	A|1	Neutrophils
⋮	⋮	⋮

3.2.2. Tracking by historical proximity

This type of tracking is handled by HistoricalProximity in Algorithm 3. It determines a cluster's most likely predecessors from the previous time-point, termed *historical associates*, which are those closest to it spatially. To find them for a given cluster X at time t, ChronoClust iterates through each of X's constituent MCs and determines their nearest cluster-comprising MCs at time $t-1$. The union of the clusters corresponding to these nearest MCs forms the historical associates. For example, as Table 1 shows, cluster (A|1,C) in day 4 has two historical associates: A|1 and C. A given cluster can have at most one distinct historical associate for each MC it comprises, occurring when each MC at time t lies closest to an MC from different cluster at time $t-1$. The nearest clusters from the preceding time-point are identified for each current time-point cluster. This is by virtue of distance calculations between constituent MCs, though there is no requirement that MCs are *shared* by clusters across time-points.

Tracking by historical proximity accommodates subspace clustering, allowing clusters to be more-tightly defined in particular *preferred* dimensions. This is accomplished through a weighted-distance calculation in determining the nearest neighbours' MCs. For two equidistant points from an MC, the one aligning along a preferred dimension will be deemed the nearest. The equation is thus:

$$dist^{\Phi}(mc_A, mc_B) = \sqrt{\sum_{x=1}^{d} \frac{(c_x(mc_A) - c_x(mc_B))^2}{\Phi_x(mc_A)}} \quad (1)$$

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Table 3
Parameters used in ChronoClust. The left column expresses value constraints.

Parameter	Function
$\mu \in [0, 1]$	Minimum weight threshold for a core-MC, as proportion of time-point data
$\beta \in [0, 1]$	Minimum weight threshold for a pcore-MC, scalar of μ
$\epsilon > 0$	Maximum radius threshold for an MC
$\upsilon \geq 1$	Maximum distance between MC during offline clustering, a scalar of ϵ
$\pi \in \mathbb{N}$	Maximum number of MC preferred dimensions
$\kappa \geq 1$	MC preferred dimension weighting in distance calculations
$\delta \in [0, 1]$	Maximum data variance along an MC's preferred dimension
$\lambda \geq 0$	Decay rate of old data on current MC state
$o \geq 0$	Weight threshold for MC deletion

where mc_A is an MC belonging to cluster A at time-point t and mc_B is an MC belonging to cluster B at time-point $t-1$. $c_x(m)$ represents the centroid of MC m in dimension x in d-dimensional space, and $\Phi_x(m)$ represents the dimension preference weighting for MC m in dimension x. This metric's dimensional preference weighting is adopted from PreDeCon [32] and HDDStream [33].

Clusters identified as *historic associates* and *parents* will typically overlap, the main exceptions being for new, *de novo* clusters. For instance, in Table 1, Cluster C in day 2 is a newly formed cluster. Using Eq. (1), Algorithm 3 calculates the projected distance between MC #2 in day 2 and MCs #0 and #1 in day 1, identifying MC #1 as the nearest. Hence, cluster A is assigned as cluster C's historical associate.

3.3. Parameters in ChronoClust

ChronoClust's parameters are listed in Table 3 and many of them are adopted from HDDStream. In this section we discuss these parameters and conceptualise them in terms of input dataset, in order to provide better understanding and guidelines on how to set them. The latter is aided by the dataset's normalisation to values in the range [0, 1].

The fundamental unit of clustering in ChronoClust is the MC, which adapts to a given time-point's data in the online phase; MCs themselves are clustered in the offline phase to summarise the data. An MC is ellipsoid in shape owing to the subspace clustering as it retains a narrower distribution of data in certain preferred dimensions. MCs have a maximum radius, ϵ, and in the extreme case can be spherical. One may conceptualise ϵ as the spatial resolution along each dimension and select a value accordingly. Hence, values in the range [0, 1] are prudent, since the dataset is normalised to this range. Note that this conceptualisation is a merely a useful simplification, in actual fact MCs can overlap in some cases (see below).

We express μ, the minimum threshold weight for a core-MC, as a proportion of the number of data entries in the time-point being processed. Hence, each cluster captures a minimum of $100\mu\%$ of data-points at each time-point and roughly $100\epsilon\%$ of the range along each dimension (conversions to percent, and clusters comprise at least one core-MC). This analogy helps the user choose these parameters based on the degree of sensitivity, at expense of capturing noise, they desire. By expressing μ as a proportion, this mechanism is robust to time-points with varying numbers of data-points.

An MC's preferred dimensions are those in which its constituent data are tightly clustered. There is an upper limit, π, on the number of dimensions that an MC may prefer. Where the absorption of a data-point would violate MC constraints, such as π, it is instead added to another. As such, it is possible for MC volumes to overlap. To constitute a preferred dimension, MC data in that dimension must exhibit a variance less than δ. MC volumes are ellipsoid by virtue of weighted distance calculations used to determine which MC will absorb a data-point. Preferred dimensions are weighted κ, versus 1 for non-preferred dimensions.

Final clusters for each time-point are assembled through the offline phase, where core-MCs and pcore-MCs are clustered. MCs that lie within threshold distance υ are linked to form a cluster. We make υ explicitly distinct from ϵ; it is unclear in HDDStream [33] if the same value (ϵ) is employed in both online and offline contexts. The threshold distance υ is expressed as a multiple of ϵ; in developing ChronoClust we have found values > 2 to work well.

In the offline phase core-MCs seed the initial clusters. Pcore-MCs near to a cluster's constituent MCs are recursively absorbed into that cluster, allowing a "daisy chaining" of core-MCs and pcore-MCs. Consider a spatial distribution of core-MCs representing regions of very high data density surrounded by pcore-MCs of lower density. The threshold weight at which an outlier-MC transitions to a pcore-MC is $\beta\mu$. A lower value of β generates more pcore-MCs, potentially bridging two core-MCs (and thus clusters) that would not otherwise be linked. We envisage a use-case where β is gradually raised, allowing large conglomerate clusters representing, for instance, broad categories of immune cell (e.g. T cells) to split into distinct sub-populations (e.g. T helpers, cytotoxic T cells, regulatory T cells). So doing would facilitate the identification of novel, rare sub-populations of immune cell, identified in the Introduction as valuable in studying disease.

The parameter λ dictates the influence of old data on current cluster state. We retain HDDStream's formulation of decay, wherein the cluster weight over time period t is decayed by factor $d = 2^{-\lambda t}$. We note that λ can be expressed as $\lambda = \frac{1}{half\text{-}life}$, where *half-life* represents the duration at which the weight decays to half its original quantity.

Finally, o is the threshold at which an outlier-MC is deleted. As with μ and ϵ, o is expressed as a proportion of the number of data entries in the time-point being processed, therein implicitly accommodating datasets with varying quantities of data. We recommend setting o after selecting an appropriate λ. We propose determining o based on the principle that a pcore-MC with weight exactly $\beta\mu$ should not be deleted before having the opportunity to capture additional data-points in the subsequent x time-points. For instance, if time-points are 1 day apart and $x = 1$, then let $dt = 1$; selecting

$$o = \beta\mu 2^{-\lambda dt} \tag{2}$$

would delete a pcore-MC with weight $\beta\mu$ at the subsequent time-point. An o value slightly smaller than this (i.e. slightly larger dt) would slightly extend MC lifespan, affording it the opportunity to capture additional data-points. Values for x and dt should be selected in accordance with the problem at hand.

4. Evaluation on synthetic data

A synthetic dataset enables detailed, unbiased and unencumbered analysis of ChronoClust's performance as, in designing it, we know exactly which clusters and sub-clusters exist, and how they evolve over time. We gauge how well ChronoClust recovers these phenomena. In so doing we operate ChronoClust at two clustering resolutions, reflective of how an immunologist would approach a cytometry dataset. An initial coarse-scale clustering identifies the broad types of cell populations present. Thereafter, finer-scale clustering teases out sub-populations, which are often of specific interest in a given context. Whereas the immunologist would employ manual gating to find these populations, we seek for ChronoClust to uncover these patterns in an automated unsupervised manner.

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

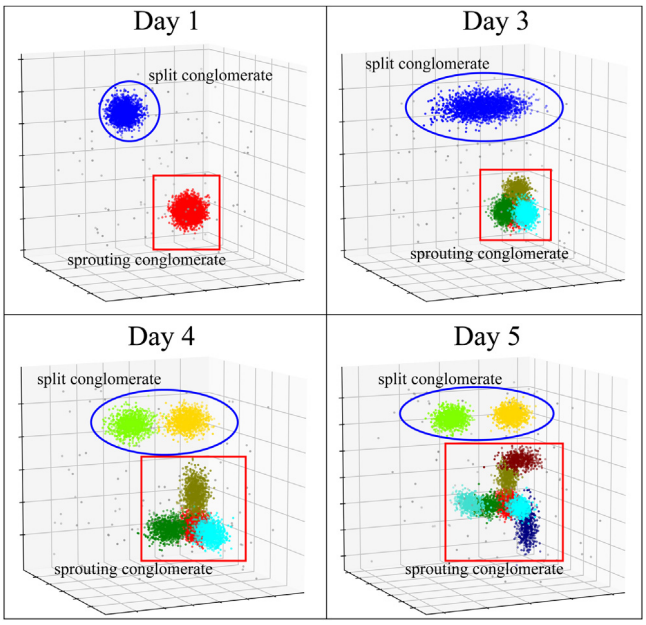

Fig. 2. The synthetic dataset. Day 2 exhibits only minor differences from day 1 and is omitted for brevity. The sub-clusters comprising each conglomerate are given unique colours for clarity. The range of values for the axes is 0 to 40.

We independently evaluate clustering performance on each day, qualitatively and quantitatively in Sections 4.3 and 4.4 respectively, and thereafter ChronoClust's tracking of cluster evolution over time, Section 4.5. First, however, we define our synthetic dataset and create manually-defined gates that form the "ground truth" for subsequent evaluation.

4.1. The synthetic dataset

Our synthetic three-dimensional dataset comprises two conglomerate clusters that evolve over 5 time-points (days), as shown in Fig. 2. Whilst three dimensions is considerably fewer than we anticipate from real cytometry data, it facilitates visual inspection and verification of clustering behaviour.

The *sprouting conglomerate* entails a single cluster that "spr outs" in three directions, Fig. 2. Both the seed cluster (day 1) and each of the sprouting appendages are represented by bespoke Gaussian distributions of data, therein forming adjacent regions of high density. This mimics immune cell differentiation pathways in the bone marrow, where common progenitor cells give rise to several functionally distinct, specialised sub-populations that present as discrete stages in the developmental pathway. For instance, *common myeloid progenitors* can give rise to *neutrophils*, *monocytes* and *eosinophils* (amongst others) [36].

The *split conglomerate* represents the immune response as observed in a tissue where immune activity is triggered, for instance by a virus. Immune cells will enter the tissue from the blood circulation, then progressively differentiate and mature into multiple subsets over time. As the immune response subsides, these populations will either perish or migrate back out of the tissue. This can present as an initial cluster of data that grows and then splits.

The sprouting and split conglomerates comprise 5000 and 2000 data-points at each day, shared evenly between their constituent Gaussians. Conglomerate cluster appendage extensions and movements are represented through additional Gaussians that complement those from previous days. In addition to the Gaussian distributions, 100 data-points are uniformly distributed through the space to represent noise. All distributions are sampled anew to generate fresh data for each day. The dataset and its specification in terms of Gaussian distributions are available at https://ghar1821.github.io/Chronoclust/

4.2. Manual gating

To evaluate ChronoClust's performance we require a "ground truth" of which cluster each data-point actually arose from. Whilst we could assign each data-point the label of the Gaussian generating it, we instead reproduce the process of manual gating to provide the ground truth. So doing better aligns our analysis with our target domain of immunology, where the ground truth of which cells belong to each of two distinct populations cannot be known if those populations are not cleanly separable in the dimensions available. This is the case with many of our Gaussians which exhibit overlaps. Gating represents the best that could be accomplished in the immunological case, and so we employ it here also.

We manually gate at two granularities, *coarse-scale* and *fine-scale*, to simulate two extreme use-cases. First, coarse-scale clustering to capture only the broad immune cell populations present and, second, to extract in fine detail the sub-populations they comprise. Gating results are shown in Fig. 3. In coarse-scale gating we define the "sprouting" conglomerate as a single cluster at all time-points. The "splitting" cluster is defined as a single cluster in days 1 to 3, and two clusters thereafter. In fine-scale gating we define a cluster for each Gaussian. For the "splitting" conglomerate, clusters are identical to the coarse-scale case. For the "sprouting" conglomerate, we transit from a single seed cluster at day 1 to seven clusters at day 5.

Gating in three-dimensions proved challenging and was performed as follows. The gates were firstly defined by determining the convex hulls for each Gaussian, through Python's SciPy module, or their combination for the coarse-scaled gating. They were then manually adjusted to remove any overlap in gate volumes, placing the boundaries in the least dense areas.

4.3. Qualitative evaluation of clustering

We qualitatively evaluate ChronoClust's ability to find the manually-gated clusters. We manually selected parameter values following the conceptualisation outlined in Section 3.3; values are reported in Table 4. Our goal is to demonstrate ChronoClust's successful operation at both clustering scales and, importantly,

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

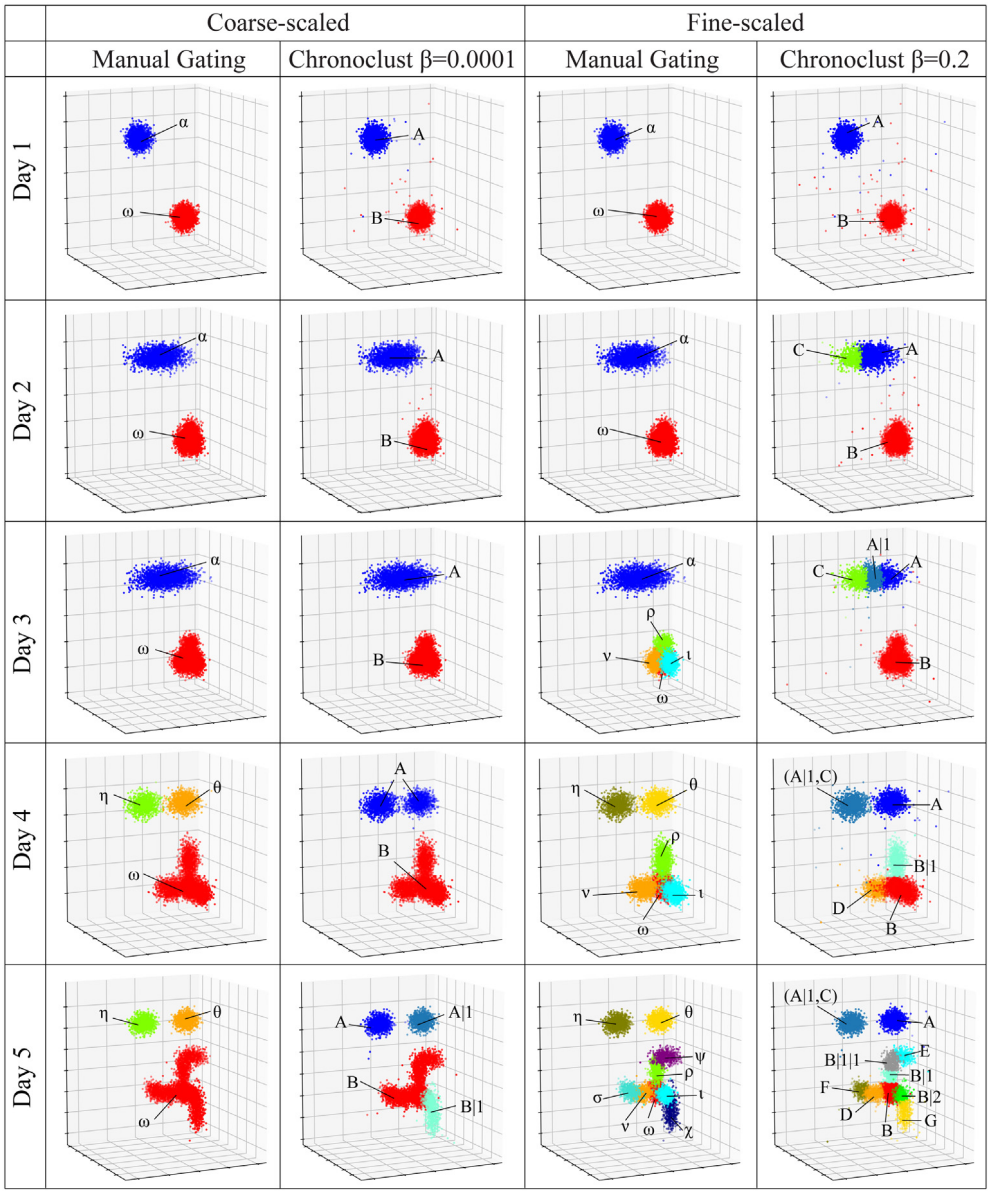

Fig. 3. Coarse-scaled and fine-scaled manual gating and ChronoClust clustering results. Axes labels and ranges are the same as in Fig. 2. Only data-points that were assigned to gates/clusters are depicted.

Table 4
Parameters used in ChronoClust results for coarse and fine-scaled clustering of the synthetic dataset.

Scale	μ	β	ϵ	υ	π	κ	δ	λ	o
Coarse	0.01	**0.0001**	0.03	6.5	3	4	0.05	2	4.35×10^{-7}
Fine	0.01	**0.2**	0.03	6.5	3	4	0.05	2	4.35×10^{-7}

that good quality clustering can be achieved at both scales by changing only a single parameter, β, the threshold weight at which outlier MCs become pcore-MCs which can be daisy chained to form larger clusters.

The same value for parameter o was used in both fine- and coarse-scaled clustering, thereby encouraging a similar spatial distribution of MCs in both cases. Following Eq. (2), we calculated o using $dt = 1.1$ and the smaller value of $\beta = 0.0001$ for coarse-grained clustering. Hence, outlier-MCs are permitted 1 day to acquire more data-points, else be deleted.

We observe a few minor qualitative differences between ChronoClust and manual gating, Fig. 3. When coarse-scale clustering, ChronoClust identifies the split conglomerate in day 4 as one cluster (cluster A) instead of two (clusters η and θ). We note that this could be corrected with a smaller υ value (not shown), but in so doing the sprouting conglomerate then registers as more than two clusters. When fine-scale clustering, with these parameter values, ChronoClust tends to over-cluster. For example, in day 2 and 3 the split conglomerate is identified as multiple clusters rather than one. Further, relative to manual gating, the sprouting conglomerate is detected as a single cluster in day 3, one too few clusters in day 4, and then one additional in day 5. In summary, in both clustering scales ChronoClust broadly recovers the same number of clusters, in the same locations, as manual gating. The small discrepancies reflect the random nature of data generated from overlapping Gaussian distributions in our synthetic dataset. We also note that whereas manual gating was informed by the number of clusters sought and set boundaries accordingly, ChronoClust did not have this information. Its performance here

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Table 5
F1-score and entropy of ChronoClust clustering on coarse-scaled and fine-scaled synthetic dataset.

Day	Coarse			Fine		
	F1-score	Entropy	Tracking accuracy	F1-score	Entropy	Tracking accuracy
1	0.93	0.28	1.00	0.92	0.29	1.00
2	0.93	0.25	1.00	0.93	0.27	1.00
3	0.95	0.23	1.00	0.43	1.55	1.00
4	0.75	0.53	1.00	0.72	0.86	1.00
5	0.94	0.26	0.75	0.83	0.68	0.90

supports ChronoClust's ability to find clusters without the need to pre-specify their number in advance.

4.4. Quantitative evaluation of clustering

We further assess ChronoClust's performance by calculating the F1-score and entropy of the clusters produced.

The F1-score is a standard accuracy measure combining recall and precision [37]. Here it measures clustering accuracy by contrasting the data-points' true class labels from the manual gating with those of the clusters ChronoClust assigns. It takes values between 0 and 1, with 1 indicating perfect accuracy. Entropy measures cluster homogeneity in terms of the true class labels of the data-points it contains [37]:

$$Entropy(k) = -\sum_{i}^{n} P_i \log_2 P_i$$
$$OverallEntropy = \sum_{j=1}^{K} \frac{m_j}{m} Entropy(j) \quad (3)$$

where $Entropy(k)$ is the entropy value for cluster k, n is number of distinct true class labels in cluster k, and P_i is the proportion of cluster k's data-points holding true class label i. The total entropy of the clustering $OverallEntropy$ is the weighted sum of each cluster's entropy; m_j refers to number of data-points in cluster j and m is the total number of data-points captured across all clusters K. A low overall entropy value indicates high-quality clustering reflective of the underlying data.

To calculate F1-scores and entropy, the ChronoClust cluster assignments (predicted labels) must be compared against the true class labels assigned to each data-point through manual gating. To do this, each ChronoClust cluster adopts the label of its nearest manual gate, using Eq. (1) on cluster and gate centroids. All data-points a ChronoClust cluster captures are assigned this label as their predicted class. Manual gate centroids are computed using HDDStream's definition of a microcluster centroid as detailed in Definition 8 of [33].

Table 5 shows the F1-score and entropy for ChronoClust's clustering of the synthetic dataset at each day. Excellent results are obtained for the coarse-scale clustering, for all days but day 4: very high F1-scores of 0.93–0.95 and very low entropy values of 0.23–0.28. In day 4 the F1-score drops to 0.75 and the entropy rises to 0.53 due to ChronoClust's under-clustering of gates η and θ as single cluster A, as discussed in the qualitative evaluation. The results for the more challenging fine-scaled clustering are also excellent for the first two days (F1-score=0.93 and entropy=0.27–0.29). Consistent with the qualitative evaluation, in day 3 the F1-score drops, and the entropy increases as ChronoClust captures the sprouting conglomerate (gates ρ, ν, ι, ω) as one cluster and detects three instead of one cluster in the splitting conglomerate. The results for days 4 and 5 are worse than for day 1 and 2 as ChronoClust generates one cluster too few in day 4 and one extra in day 5 for the sprouting cluster. In summary, the quantitative evaluation is consistent with the qualitative, and confirms ChronoClust's ability to accurately cluster at both coarse- and fine-scales.

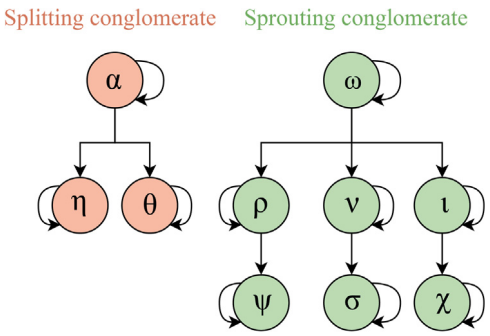

Fig. 4. Valid transitions across days for clustering of the synthetic dataset, based on how we created the dataset. Nodes are given manually-defined gate labels; it is these gates and transitions that we seek ChronoClust to reproduce.

4.5. Cluster tracking evaluation

Having established ChronoClust's accurate clustering of the synthetic dataset at each day, here we explore its linkage of clusters over time. Based on how the synthetic dataset was designed, Fig. 4 expresses valid lineage relationships over time in terms of the manually-defined gate labels. This transition graph forms the basis for evaluating how well ChronoClust reconstructs our data's temporal evolution.

Table 1 from Section 3.2 summarises ChronoClust's fine-scaled cluster tracking results; the cluster IDs represent *tracking by lineage determination*. As illustration, Fig. 5 shows cluster temporal evolution as determined by both tracking modes.

Tracking by lineage determination. As shown by cluster IDs in Figs. 3 and 5, ChronoClust's *tracking by lineage determination* successfully tracks clusters' splits and merges. In coarse-scaled clustering, the splitting conglomerate in day 5 is correctly labelled as cluster A and A|1, indicating both arise from cluster A at day 4, which are valid transitions (Fig. 4). In fine-scale clustering, ChronoClust identifies the sprouting conglomerate in day 4 as clusters B and B|1, indicating they arose from cluster B at day 3. These too are congruent with how the dataset was assembled. Moreover, it also accurately labels cluster B|1|1 and B|2 in day 5, indicating they arose from cluster B|1 and B at day 4. Fine-scaled clustering of the splitting conglomerate is a more complicated case. In day 2, gate α is over-clustered as clusters A and C. By day 3, A|1 has split from A. By day 4 and 5, while cluster A remains, day 3's clusters A|1 and C have merged to form (A|1, C). Whilst admittedly somewhat over-zealous in creating new clusters, these lineages are nonetheless consistent with the underlying data.

Tracking by historical proximity. Fig. 3 depicts several new *de novo* clusters created during fine-scaled operation, the IDs of which indicate no parents by *lineage determination* (C, D, E, F, G). Hence, these *de novo* clusters share no constituent MCs with clusters from the preceding day. This can occur if data-points from Gaussian distributions representing the growth of conglomerate appendages register as new MCs, rather than the movement of pre-existing ones. ChronoClust's *tracking by historical proximity* method is able to accurately determine each novel cluster's predecessors, given their spatial proximity across time, Table 1 and Fig. 5. For example, cluster A in day 1 is accurately identified as the historical associate of cluster C in day 2. Similarly, clusters B and B|1 in day 4 are correctly identified as cluster E and G's historical associates in day 5. Cluster B is also correctly

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

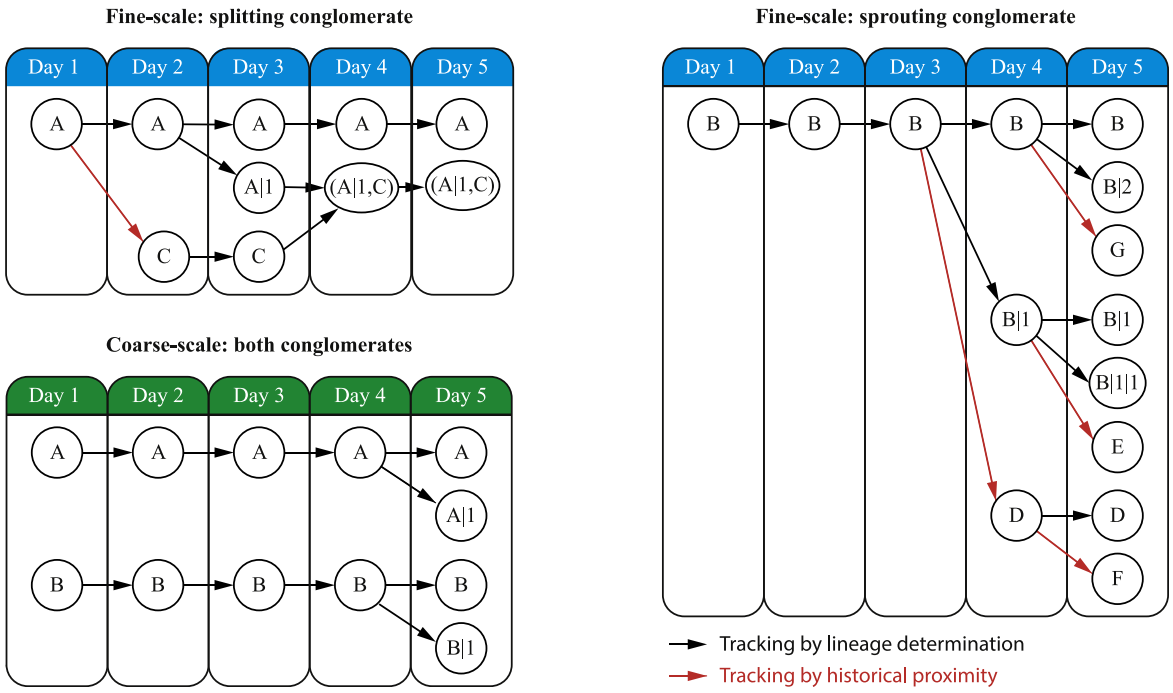

Fig. 5. ChronoClust's capture of the temporal evolution of conglomerates in the synthetic dataset at both fine-scale and coarse-scale clustering resolutions. Both tracking by *lineage determination* and *historical proximity* are shown.

determined as cluster D's historical associate. Lastly, cluster D is correctly registered as cluster F's historical associate.

We quantified the accuracy of ChronoClust's *temporal tracking by historical proximity* as the proportion of identified transitions that are valid. To do so, we related each ChronoClust cluster with its nearest gate, using Eq. (1) as outlined in Section 4.4 above. Each cluster's historical associate relationships could then be cross-referenced with valid transitions as specified in Fig. 4.

ChronoClust's tracking accuracy is shown in Table 5. In both coarse- and fine- scale clustering, identified cluster evolutions over days 1–4 are valid (tracking accuracy=1.00). However, in day 5, the tracking accuracy drops to 0.75 and 0.90 respectively. This is due to day 4's under-clustering, Fig. 3. For coarse-scale clustering, ChronoClust captures gates η and θ as a single cluster A which is linked to gate η. Valid transitions for gates η and θ at day 5 are from themselves (Fig. 4), but since day 4's cluster A corresponds to gate η alone, the transition of A|1 from A is determined invalid thus reducing tracking accuracy. Similarly for fine-scale clustering, the valid transition for gate χ at day 5 is from gate ι at day 4, Fig. 4. However, day 4 gates ι and ω are captured by a single cluster, B, at day 4, and B is related to ω. Day 5 cluster G represents gate χ, but as its historical associate B (at day 4) represents ω, not ι, this transition is deemed invalid.

In summary, ChronoClust demonstrated excellent clustering and cluster tracking results on the synthetic dataset. In the next section, we discuss its application to a real cytometry dataset.

5. Evaluation on the West Nile Virus dataset

This study's main goal is an automated approach for clustering and profiling cytometry data over time. Hence, it is imperative that we evaluate ChronoClust's performance on a real cytometry dataset, reported here. As with the synthetic dataset, we evaluate ChronoClust's clustering at each day both qualitatively and quantitatively, Sections 5.3 and 5.4 respectively. Thereafter we examine ChronoClust's cluster temporal tracking performance, Section 5.5.

5.1. The West Nile Virus dataset

West Nile Virus (WNV) is disseminated by mosquitoes, causing infection of the central nervous system and severe neurological disease, which may culminate in death or permanent neurological damage in survivors [6]. Our WNV dataset quantifies the immune response of WNV-infected mice over eight days: from day 0 (no infection) to day 7. For each day, immune cells were extracted from the bone marrow of four mice and analysed by flow cytometer. Of them, 190,000 cells from each day were obtained for ChronoClust. Expression levels of nine proteins were measured per cell: (1) B220, (2) CD3/NK11, (3) Ly6C, (4) CD115, (5) CD11b, (6) Ly6G, (7) SSC-A, (8) CD117, (9) SCA-1. From a data mining perspective, the dataset contains 190,000 data-points described through nine features over each of eight days.

5.2. Benchmark for evaluating ChronoClust

In this context, the *ground truth* represents the true immune cell populations that each data-point represents. This is identified through manual gating, performed by a domain expert (author Ashhurst). Manual gating is an ideal benchmark for ChronoClust evaluation, as one of our primary goals is to automate this process.

The following 16 cell populations were identified in the dataset, comprising *un-activated* and *activated* forms of the following: B cells, eosinophils, monoblasts, mature Ly6C monocytes (labelled *monocytes*), neutrophils, plasmacytoid dendritic cells (PDC), stem and progenitor cells (SPC), T, NK and NK-T (which are indistinguishable given the proteins characterised, we label these *T-NK*) cells.

In addition to manual gating, we also evaluate ChronoClust clustering against random cluster assignment (the *random baseline*) and automated gating through FlowSOM [14]. Random cluster assignment and FlowSOM both require the number of clusters to be specified. We set this to 16 for random assignment. For FlowSOM we specify 32 clusters, adhering to the recommendation to over-cluster by FlowSOM's creators, Van Gassen et al. [14].

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Fig. 6. Comparison of cell counts for each cell population between manual gating and ChronoClust, for each day. Names affixed with * indicate *activated* cell populations, versus un-activated. Cell counts are shown on log axes to add clarity for small cell populations. Each day's dataset comprises 190,000 cells. Red arrows indicate points of substantial divergence between ChronoClust and manual gating.

FlowSOM is specifically optimised for cytometry data, and beyond specifying the number of clusters to find, standard practice in cytometry entails no further parameter tuning.

5.3. Qualitative evaluation of clustering

We qualitatively evaluate ChronoClust by contrasting the number of cells it classifies into each cellular population versus manual gating. In order to relate ChronoClust clusters to manual gates, we employ the same method as in Section 4.4. Furthermore, we find changes in cell counts found through ChronoClust to be consistent with previous biological findings. ChronoClust parameters were in the first instance selected in accordance with the conceptualisation in Section 3.3, and thereafter a subset (μ, ϵ, υ) were manually tuned; the values are supplied in Table 6.

Table 6
ChronoClust's parameters for the WNV dataset.

μ	β	ϵ	υ	π	κ	δ	λ	o
3.58×10^{-4}	0.5	0.0475	8.75	9	15	0.05	0.9	9.01×10^{-5}

The clusters produced by ChronoClust are broadly consistent with manual gating. With the exception of day 0 where it finds only 14 of the 16 cell populations, ChronoClust consistently identifies all the cell populations characterised by manual gating,

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

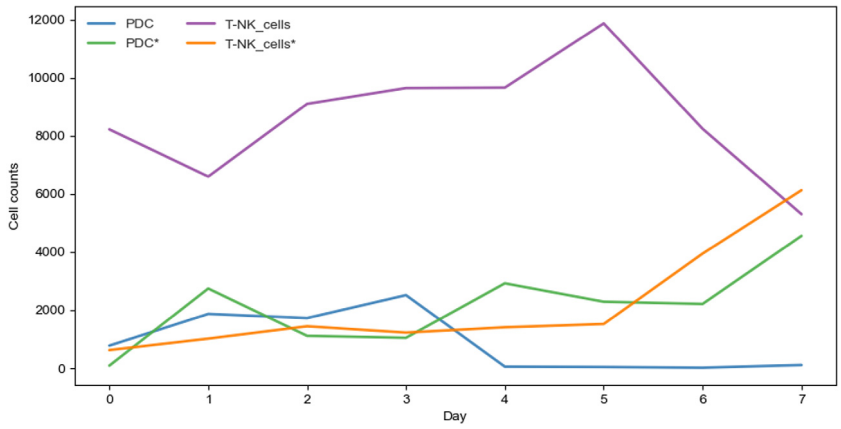

Fig. 7. Variation in number of cells found in PDC and T-NK cells by ChronoClust. Names affixed with * indicate *activated* cell populations, versus un-activated.

Fig. 6. We note that the two missing populations, activated B cells and activated monoblasts, represent a small proportion of all available cells: 1.8% and <0.1% respectively.

ChronoClust broadly recovers similar numbers of cells in each population as manual gating. The largest discrepancy lies in day 6's activated B cell population, found to represent 21% of all gated cells by manual gating but only 11% of clustered cells by ChronoClust. This represents somewhat of an outlier, as all other discrepancies between percentage were under 5%. Overall, these results are very encouraging as ChronoClust's clustering was not informed by either the expected number of clusters nor any expert knowledge on WNV.

Fig. 7 demonstrates the variation in number of cells belonging to PDC and T-NK cell populations as time progresses. We observe a general increasing trend in number of activated PDC and activated T-NK cells. Furthermore, a sharp decline in un-activated T-NK cell numbers occurs simultaneously with a sudden surge in *activated* T-NK cell numbers at day 5. A similar subtler trend exists in PDC cells starting from day 3. These findings are in accord with a study by Cho et al. [38] who characterise the immune response development to WNV infection as commencing with an increase in type 1 Interferons (IFNs) leading to a rise in effector functions of NK cells. The boost in IFNs can be attributed to rise in PDC cells which are the most dominant producers of IFNs.

5.4. Quantitative evaluation of clustering

We contrasted ChronoClust, FlowSOM and random baseline clustering performances in terms of F1-scores and entropy, Table 7. In all instances ChronoClust outperformed both FlowSOM and random assignment, with higher F1-scores indicating a consistently high precision and recall for ChronoClust and lower entropy scores representing more homogeneous clusters. As would be expected, both ChronoClust and FlowSOM considerably outperform random assignment of data-points into clusters.

5.5. Cluster tracking evaluation

We now turn to evaluating ChronoClust's cluster tracking ability. Based on existing immunological knowledge, Fig. 8 Left depicts valid transitions between different cell populations over time. This lineage graph serves as the foundation for assessing ChronoClust's ability to track cell populations movements over time.

Fig. 8 Right depicts the temporal evolution of cell populations as determined by ChronoClust's tracking by historical proximity. ChronoClust finds 12% of the clusters across days 1 to 7 to be *de novo* clusters. The remaining 88% are existing clusters, trackable through lineage determination. Their temporal transitions are very strongly consistent (99.1%) with those found through tracking by historical proximity. The minor discrepancy (0.9%) arises from MCs that split and drift away from their parents, such that they reside in closer proximity to a cluster from another lineage. We find this, for instance, in cluster A|1 at day 1 which identifies B as its historical associate, rather than A (data not shown). Nonetheless, to find a discrepancy of less than 1% between the two tracking schemes, on real world, noisy biological data, is highly encouraging.

Recalling, above, that the only immune cell population to truly lack a predecessor is the hematopoietic stem cell, we focus on historical associates in the following analysis, as it offers a more complete picture of cellular differentiation for *de novo* clusters.

The majority of transitions represent given populations arising from themselves at the previous time-point. However, we also detect cluster transitions between distinct cell populations, representing cells transiting through differentiation pathways, such as transitions from un-activated to activated form, from activated SPC to un-activated Eosinophils, as well as from activated Monoblast to un-activated Monocytes. In addition, ChronoClust observed more transitions from un-activated to activated states between day 6 and day 7, 0.9% of all transitions compared to 0.3–0.7% of all transitions for day 1–6. This is in agreement with the heightening of cell differentiation activity at later stage of WNV infection.

We quantified ChronoClust's tracking ability through the tracking accuracy metric discussed in Section 4.5. Tracking accuracy scores of 0.88 to 0.97 indicate that nearly all transitions found between days are biologically plausible. There exist some minor discrepancies, representing ∼7.1% of all transitions. Of all transitions, ∼1.7% represented transitions of cell populations from activated to un-activated states; this likely reflects the close spatial proximity of these cell populations, coupled with stochastic noise in how the data-points present. A further ∼4.5% of the invalid transitions were exhibited by small clusters capturing 1.7% of total cells in the dataset from day 1 until 7. The remaining ∼0.9% represent large clusters capturing 2.1% of total cells in the dataset between day 1 to day 7, some of which are found to be cell populations transitioning back and forth as different cell populations. For example, activated T-NK cells → activated B cells in day 1 (specific data not shown), followed by that same cluster reverting to activated T-NK cells in day 2. It is likely that these clusters are situated on the periphery of these distinct cell populations, which are not cleanly separable given the nine markers employed in this cytometry analysis.

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

Table 7
F1-score, Entropy, and tracking accuracy of ChronoClust, FlowSOM, and random baseline clustering of the WNV dataset. High F1-scores and low entropy indicate superior clustering.

Day	Clustering						Tracking accuracy
	F1-Score			Entropy			
	ChronoClust	FlowSOM	Random	ChronoClust	FlowSOM	Random	ChronoClust
0	0.67	0.63	0.06	0.32	0.63	2.53	N.A.
1	0.54	0.45	0.05	0.46	0.86	2.81	0.88
2	0.64	0.56	0.06	0.38	0.74	2.71	0.93
3	0.65	0.57	0.06	0.39	0.74	2.63	0.92
4	0.68	0.60	0.07	0.37	0.73	2.68	0.88
5	0.62	0.58	0.06	0.50	0.88	2.79	0.93
6	0.60	0.55	0.06	0.61	1.03	3.10	0.97
7	0.63	0.56	0.06	0.62	1.07	3.25	0.96

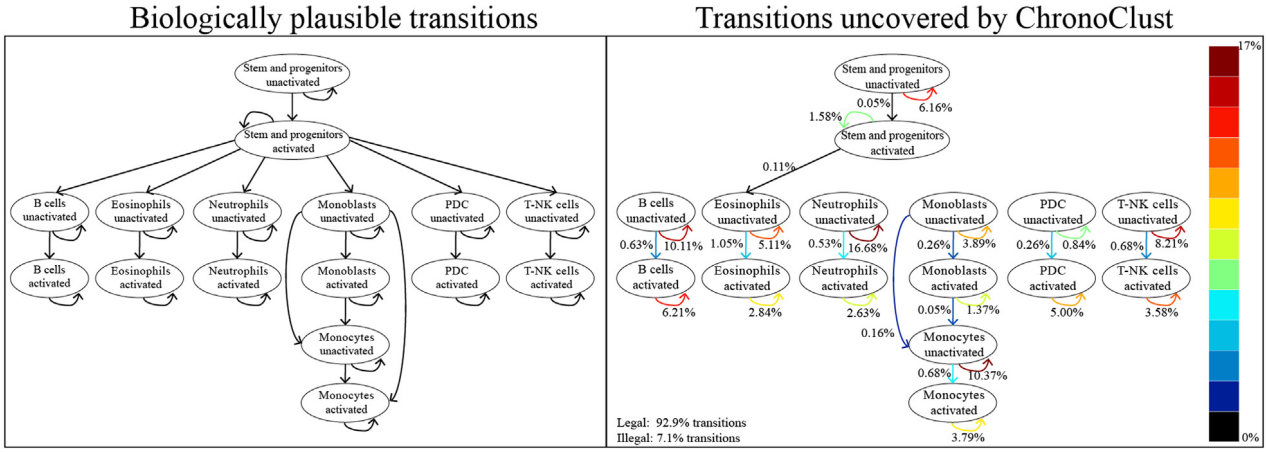

Fig. 8. Left: Biologically plausible cell population transitions. Right: Legal transitions uncovered by ChronoClust's tracking by historical proximity. Arrow colours and labels indicate the proportion of all cluster transitions representing the given transition. The percentage of legal versus illegal transitions (based on Left) are shown. ChronoClust successfully uncovers most of the biologically plausible transitions, with the exception of transitions from un-activated monoblasts to activated monocytes, and activated stem and progenitors to un-activated B cells, neutrophils, monoblasts, PDC, and T-NK cells.

Taken together, we consider these results to support ChronoClust's suitability for analysing cytometry data. In an unsupervised manner, it both correctly clusters known cell populations in each time-point and maps out their differentiation pathways. There exists no other analytical tool capable of simultaneously performing both these functions in an automated, unsupervised fashion.

6. Conclusions

Clustering time-series data, and tracking the evolution of clusters over time, is a powerful technique for analysing data from scientific experiments. High-throughput analyses have advanced our understanding of biological systems at single, static points in time. However, biological systems are highly dynamic, and technologies to integrate these data and elucidate temporal dynamics are lacking. Our present work was motivated by a desire to characterise the immune response development to a specific trigger, which is needed for devising effective treatments. This data would present as a time-series of discrete cytometry datasets. We formulated our task as clustering of immune cell populations and tracking their evolution over time. The manner in which clusters at each time-point evolve from those preceding them corresponds to immune cell developmental pathways which collectively determine immune system and disease outcomes.

We first derived the requirements for a clustering solution for our cytometry use-case and then conducted a review of existing clustering approaches, finding that none meet all our criteria. As a solution, we proposed ChronoClust, a novel density-based dynamic clustering and tracking algorithm for high-dimensional data, which builds upon the HDDStream algorithm. ChronoClust supports two types of cluster tracking: by lineage determination and by historical proximity. The former represents a high-confidence establishment of cluster evolution by virtue of microclusters (MC; the unit of clustering) being shared between clusters over time. The latter is based on spatial proximity over time. It enables the construction of temporal relationships for newly forming clusters. We have also provided a conceptualisation of ChronoClust parameters that aid selection of values in a problem-specific manner and presented guidelines on how to set them.

To evaluate ChronoClust's utility and performance for our use-case, we firstly devised a synthetic dataset exhibiting the qualities of the immune response's temporal evolution as captured through cytometry. The cluster numbers, locations and temporal relationships were known absolutely, and ChronoClust excelled in reproducing them. Through this, we also demonstrated the ability of ChronoClust to operate across different clustering resolutions by controlling just one parameter. This functionality is important as immunologists would first wish to capture the broad cell populations, and then "zoom in" to identify novel sub-populations of potential importance. We then analysed a real cytometry dataset capturing the immune response of mice infected with the WNV over 8 days. A qualitative analysis revealed ChronoClust's ability to reproduce the clusters and relationships determined manually by an expert, in an unsupervised and automated fashion. In addition, ChronoClust uncovered temporal changes in immune cell population sizes consistent with existing biological studies. A quantitative analysis shows ChronoClust to outperform one of the leading cytometry clustering

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.

algorithms, FlowSOM. Lastly, ChronoClust successfully uncovers temporal immune cell developmental pathways with a very high degree of accuracy (∼92%).

We believe ChronoClust fulfils the criteria set out in Section 2. First, it does not require the number of clusters to be pre-specified in advance. Second, the fundamental components of its clustering operation (MCs) facilitate sub-space clustering and are collected into clusters of arbitrary shape. Third, the high quality of its clustering (high F1-score and low entropy) over the 9-dimensional WNV dataset suggests it could easily accommodate higher dimensional datasets. Lastly, its performance on the WNV dataset indicates that it is robust to noise and outliers. Despite a considerable 25% of the WNV data lying outside of the expert-defined manual gates, suggesting they are noise, outliers, or biologically irrelevant, ChronoClust performed well.

ChronoClust is not limited to cytometry applications; it can be applied to other domains and problems requiring dynamic clustering and cluster tracking over time. To facilitate this we have made our implementation of ChronoClust publicly available at https://ghar1821.github.io/Chronoclust/.

7. Future work

There are several avenues for future work, motivated by both improving and extending the ChronoClust algorithm and its application in cytometry. From an algorithmic perspective, firstly, although ChronoClust was designed to operate over a time-series of discrete datasets, motivated by our specific use-case, it could be extended to continuous data streams. Secondly, a self-tuning method for parameter selection that is based on existing quality measures such as snapshot quality and historical cost [39] is worth investigating to aid users in selecting appropriate parameter values. Thirdly, ChronoClust can be extended for multi-view clustering in several ways [40,41]. For our use-case, a possible strategy is to group the features into different views and investigate if the diversity and complementarity of these views can improve the accuracy of the single-view clustering.

From an application perspective, the synthetic and WNV data sets were suitable starting points for verifying ChronoClust's operation and temporal tracking functionality. The next step will be to evaluate ChronoClust's performance on other cytometry datasets with higher dimensionality and cell number, exhibiting greater developments over time. For instance, at any point in time, most immune cell populations are present in the bone marrow, albeit at numbers that change during the immune response. We believe this to be the reason for the high number of cluster transitions between the same cell types in our WNV dataset. In contrast, in non-bone marrow tissues, we anticipate more *de novo* clusters emerging, each exhibiting greater evolution as immune cells arrive to combat pathogens and undergo further differentiation accordingly.

These differences also highlight another area of further development for ChronoClust. The immune response develops not only over time (the focus of our current study) but also over *space*. Cells typically differentiate in one organ, such as the lymph nodes, bone marrow or spleen, and migrate to the site of infection where they may undergo further developments. Yet, a single cytometry dataset is typically constructed for only one organ. There is a need to label and then integrate several organs worth of data through ChronoClust, and track not only the immune response over time, but also across organs.

Lastly, in using cytometry to map out the immune response's orchestration to a given target, a pertinent question is to understand *when* and *why* given triggers, or even the same trigger in different individuals, yield different immune outcomes. We plan to investigate these research questions in future work.

8. Ethics statement

All procedures involving mice were reviewed and approved by the University of Sydney Animal Ethics Committee (AEC). The AEC fulfils all the requirements of the National Health and Medical Research Council (NHMRC) and the NSW State Government of Australia. Flow cytometry was performed using a 5-laser BD LSR-II flow cytometer, as described previously in [9].

Acknowledgements

MNR is supported by the Judith and David Coffey LifeLab and the Centre for Excellence in Advanced Food Enginomics at the University of Sydney, Australia. The experimental WNV data reported in this manuscript was generated under research grants from the Australian Research Council (grant numbers LE140100149, DP160102063) and the Australian National Health and Medical Research Council (grant number 1088242). These funders had no role in the design, collection, analysis or interpretation of these data, nor the authorship or publication of this manuscript.

We thank the Sydney Informatics Hub at the University of Sydney for providing access to their High-Performance Computing facility.

References

[1] T. Hey, S. Tansley, K.M. Tolle, et al., The Fourth Paradigm: Data-Intensive Scientific Discovery, Vol. 1, Microsoft Research, Redmond, WA, 2009.

[2] M.H. Spitzer, G.P. Nolan, Mass cytometry: single cells, many features, Cell 165 (4) (2016) 780–791, http://dx.doi.org/10.1016/j.cell.2016.04.019.

[3] S.C. Bendall, E.F. Simonds, P. Qiu, E.D. Amir, P.O. Krutzik, R. Finck, R.V. Bruggner, R. Melamed, A. Trejo, O.I. Ornatsky, R.S. Balderas, S.K. Plevritis, K. Sachs, D. Pe'er, S.D. Tanner, G.P. Nolan, Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum, Science 332 (May) (2011) 687–695, http://dx.doi.org/10.1126/science.1198704.

[4] D.R. Bandura, V.I. Baranov, O.I. Ornatsky, A. Antonov, R. Kinach, X. Lou, S. Pavlov, S. Vorobiev, J.E. Dick, S.D. Tanner, Mass cytometry: a novel technique for real-time single cell multi-target immunoassay based on inductively coupled plasma time of flight mass spectrometry, Anal. Chem. 81 (16) (2009) 6813–6822, http://dx.doi.org/10.1021/ac901049w.

[5] F. Mair, F.J. Hartmann, D. Mrdjen, V. Tosevski, C. Krieg, B. Becher, The end of gating? An introduction to automated analysis of high dimensional cytometry data, Eur. J. Immunol. 46 (1) (2016) 34–43, http://dx.doi.org/10.1002/eji.201545774.

[6] N.J.C. King, D.R. Getts, M.T. Getts, S. Rana, B. Shrestha, A.M. Kesson, Immunopathology of flavivirus infections, Immunol. Cell Biol. 85 (1) (2007) 33–42, http://dx.doi.org/10.1038/sj.icb.7100012.

[7] S. Doulatov, F. Notta, K. Eppert, L.T. Nguyen, P.S. Ohashi, J.E. Dick, Revised map of the human progenitor hierarchy shows the origin of macrophages and dendritic cells in early lymphoid development, Nature Immunol. 11 (7) (2010) 585–593, http://dx.doi.org/10.1038/ni.1889.

[8] P. Qiu, E.F Simonds, S.C Bendall, K.D Gibbs Jr., R.V. Bruggner, M.D Linderman, K. Sachs, G.P Nolan, K.K Plevritis, Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE, Nature Biotechnol. 29 (10) (2012) 886–891.

[9] T.M. Ashhurst, A.L. Smith, N.J.C. King, High-dimensional fluorescence cytometry, Curr. Protoc. Immunol. 119 (2017) 5.8.1–5.8.38, http://dx.doi.org/10.1002/cpim.37.

[10] H. Kriegel, P. Kröger, A. Zimek, Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering, ACM Trans. Knowl. Discov. Data 3 (1) (2009) 1–58, http://dx.doi.org/10.1145/1497577.1497578.

[11] E.D. Amir, K.L. Davis, M.D. Tadmor, E.F. Simonds, J.H. Levine, S.C. Bendall, D.K. Shenfeld, S. Krishnaswamy, G.P. Nolan, D. Pe'er, ViSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia, Nature Biotechnol. 31 (2013) 545–552, http://dx.doi.org/10.1038/nbt.2594.

[12] L.V.D. Maaten, Accelerating t-sne using tree-based algorithms, J. Mach. Learn. Res. 15 (Oct) (2014) 3221–3245.

[13] L.V.D. Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.

[14] S. Van Gassen, B. Callebaut, M.J. Van Helden, B.N. Lambrecht, P. Demeester, T. Dhaene, Y. Saeys, FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data, Cytometry A 87 (7) (2015) 636–645, http://dx.doi.org/10.1002/cyto.a.22625.

[15] M. Guilliams, C.-A. Dutertre, C.L. Scott, N. McGovern, D. Sichien, S. Chakarov, S. Van Gassen, J. Chen, M. Poidinger, S. De Prijck, et al., Unsupervised high-dimensional analysis aligns dendritic cells across tissues and species, Immunity 45 (3) (2016) 669–684, http://dx.doi.org/10.1016/j.immuni.2016.08.015.

[16] E.R. Zunder, E. Lujan, Y. Goltsev, M. Wernig, G.P. Nolan, A continuous molecular roadmap to ipsc reprogramming through progression analysis of single-cell mass cytometry, Cell Stem Cell 16 (3) (2015) 323–337, http://dx.doi.org/10.1016/j.stem.2015.01.015.

[17] E. Lughofer, Extensions of vector quantization for incremental clustering, Pattern Recognit. 41 (3) (2008) 995–1011, http://dx.doi.org/10.1016/j.patcog.2007.07.019.

[18] E. Lughofer, A dynamic split-and-merge approach for evolving cluster models, Evolv. Syst. 3 (3) (2012) 135–151.

[19] R. Gray, Vector quantization, IEEE ASSP Mag. (1984) 4–29.

[20] E. Lughofer, M. Sayed-Mouchaweh, Autonomous data stream clustering implementing split-and-merge concepts — towards a plug-and-play approach, Inform. Sci. 304 (2015) 54–79, http://dx.doi.org/10.1016/j.ins.2015.01.010.

[21] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD-96, Portland, Oregon, USA, 1996, pp. 226–231.

[22] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496, http://dx.doi.org/10.1126/science.1242072.

[23] X. Xu, S. Ding, Z. Shi, An improved density peaks clustering algorithm with fast finding cluster centers, Knowl.-Based Syst. 158 (2018) 65–74, http://dx.doi.org/10.1016/j.knosys.2018.05.034.

[24] M. Du, S. Ding, H. Jia, Study on density peaks clustering based on k-nearest neighbors and principal component analysis, Knowl.-Based Syst. 99 (2016) 135–145, http://dx.doi.org/10.1016/j.knosys.2016.02.001.

[25] L. Yaohui, M. Zhengming, Y. Fang, Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy, Knowl.-Based Syst. 133 (C) (2017) 208–220, http://dx.doi.org/10.1016/j.knosys.2017.07.010.

[26] X. Xu, S. Ding, M. Du, Y. Xue, DPCG: an efficient density peaks clustering algorithm based on grid, Int. J. Mach. Learn. Cybernet. 9 (5) (2018) 743–754.

[27] M. Ester, H. Kriegel, J. Sander, M. Wimmer, X. Xu, Incremental clustering for mining in a data warehousing environment, in: Proceedings of the 24rd International Conference on Very Large Data Bases, VLDB '98, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998, pp. 323–333.

[28] P. Kalnis, N. Mamoulis, S. Bakiras, On discovering moving clusters in spatio-temporal data, in: International Symposium on Spatial and Temporal Databases, Springer, 2005, pp. 364–381.

[29] F. Cao, M. Ester, W. Qian, A. Zhou, Density-based clustering over an evolving data stream with noise, in: Proceedings of the Sixth SIAM International Conference on Data Mining, 2006, pp. 328–339. http://dx.doi.org/10.1137/1.9781611972764.29.

[30] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th International Conference on Very Large Data Bases, Vol. 29, VLDB '03, VLDB Endowment, 2003, pp. 81–92.

[31] Y. Chen, L. Tu, Density-based clustering for real-time stream data, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 133–142, http://dx.doi.org/10.1145/1281192.1281210.

[32] C. Bohm, K. Railing, H.-P. Kriegel, P. Kröger, Density connected clustering with local subspace preferences, in: Proceedings of the Fourth International Conference on Data Mining, IEEE, 2004, pp. 27–34, http://dx.doi.org/10.1109/ICDM.2004.10087.

[33] I. Ntoutsi, A. Zimek, T. Palpanas, P. Kröger, H. Kriegel, Density-based projected clustering over high dimensional data streams, in: Proceedings of The 2012 SIAM International Conference on Data Mining, 2012, pp. 987–998. http://dx.doi.org/10.1137/1.9781611972825.85.

[34] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, R. Schult, Monic: Modeling and monitoring cluster transitions, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, ACM, New York, NY, USA, 2006, pp. 706–711, http://dx.doi.org/10.1145/1150402.1150491.

[35] M. Oliveira, J. Gama, A framework to monitor clusters evolution applied to economy and finance problems, Intell. Data Anal. 16 (1) (2012) 93–111, http://dx.doi.org/10.3233/IDA-2011-0512.

[36] S.H. Orkin, Diversification of haematopoietic stem cells to specific lineages, Nature Rev. Genet. 1 (1) (2000) 57–64, http://dx.doi.org/10.1038/35049577.

[37] P. Tan, M. Steinbach, V. Kumar, Cluster analysis: basic concepts and algorithms, in: Introduction to Data Mining, Addison Wesley, 2005, pp. 487–568, http://dx.doi.org/10.1186/1471-2377-11-64, Ch. 11.

[38] H. Cho, M.S. Diamond, Immune responses to West Nile virus infection in the central nervous system, Viruses 4 (12) (2012) 3812–3830, http://dx.doi.org/10.3390/v4123812.

[39] D. Chakrabarti, R. Kumar, A. Tomkins, Evolutionary clustering, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2006, pp. 554–560, http://dx.doi.org/10.1145/1150402.1150467.

[40] Y. Yang, H. Wang, Multi-view clustering: a survey, Big Data Min. Anal. 1 (2) (2018) 83–107, http://dx.doi.org/10.26599/BDMA.2018.9020003.

[41] H. Wang, Y. Yang, B. Liu, H. Fujita, A study of graph-based system for multi-view clustering, Knowl.-Based Syst. 163 (2019) 1009–1019, http://dx.doi.org/10.1016/j.knosys.2018.10.022.

Please cite this article as: G.H. Putri, M.N. Read, I. Koprinska et al., ChronoClust: Density-based clustering and cluster tracking in high-dimensional time-series data, Knowledge-Based Systems (2019), https://doi.org/10.1016/j.knosys.2019.02.018.