A clustering algorithm to organize satellite hotspots data for the purpose of tracking bushfires remotely

by Weihao Li, Emily Dodwell, and Dianne Cook

Abstract An abstract of less than 150 words.

Introduction

The 2019-2020 Australia bushfire season was catastrophic in the scale of damage caused to agricultural resources, property, infrastructure, and ecological systems. By the end of 2020, the devastation attributable to these Black Summer fires included 33 lives lost, almost 19 million hectares of land burned, over 3,000 homes destroyed and AUD $1.7 billion in insurance losses, as well as an estimated 1 billion animals killed, including half of Kangaroo Island's population of koalas (Filkov et al., 2020). According to CSIRO and of Meteorology (2020), 2019 was the warmest year on record in Australia and capped off a period from 2013-2019 that represents seven of the nine warmest years. There is concern and expectation that impacts of climate change – including more extreme temperatures, persistent drought, and changes in plant growth and landscape drying – will worsen conditions for extreme bushfires (CSIRO and of Meteorology, 2020, Deb et al. (2020)). Contributing to the problem is that dry lightning represents the main source of natural ignition, and fires that start in remote areas deep in the temperate forests are difficult to access and monitor (Abram et al., 2021). Therefore, opportunities to detect fire ignitions, monitor bushfire spread, and understand movement patterns in remote areas are important for developing effective strategies to mitigate bushfire impact.

Remote satellite data provides a potential solution to the challenge of active fire detection and monitoring. Development of algorithms that process satellite imagery into hotspots – pixels that represent likely fires – is an active area of research (see for example Giglio et al. (2016), Xu and Zhong (2017), Wickramasinghe et al. (2016), Jang et al. (2019)). Detection of bushfire ignition and movement requires the clustering of satellite hotspots into meaningful clusters, which may then be considered in their entirety or summarized by a trajectory.

In this paper, we propose a spatiotemporal clustering algorithm to represent bushfires as clusters of hotspot pixels in order to determine points of bushfire ignition and track their movement over space and time. Inspired by two existing clustering algorithms, namely Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996) and Fire Spread Reconstruction (FSR) (Loboda and Csiszar, 2007), our algorithm extends the functionality of DBSCAN's spatial clustering parameters to the additional temporal dimension, while drawing upon the fire movement dynamics presented in FSR. We generalize the latter's specification of spatiotemporal parameters, thereby providing an intuitive, straightforward, and extendable approach to the complex problem of bushfire identification and monitoring that may be applied to any satellite wildfire product. In clustering hotspots into bushfires of arbitrary shape and size, we capture key bushfire behavior: fire evolution occurs only forwards in time; fires can smolder undetectably for awhile, burn out, and merge with other bushfires; and solitary pixels that may not represent true fires should not be represented as a bushfire cluster. We implement this algorithm in R package **spotoroo**: Spatiotemporal Clustering of Satellite Hot Spot Data, available on CRAN. By enabling the user to cluster satellite hotspot data across space and time, this software provides the ability to relate findings to key factors in bushfire ignition and patterns in their spread (e.g. weather and fuel sources).

The core functionality of this spatiotemporal clustering algorithm determines whether a hotspot represents a new ignition point or a continuation of an existing bushfire by comparing and combining cluster membership information via incremental updates from one time frame to the next. Our algorithm first slices the hotspot data by its temporal dimension according to a user-defined time step. This thereby divides the overall spatiotemporal clustering task into many smaller spatial clustering tasks that may be completed in parallel, where each frame can be considered a static snapshot in time. Within each time frame, hotspots that fall within the threshold of a user-defined spatial metric of each other are joined in a cluster. Then, proceeding sequentially, we identify whether or not a hotspot was observed in the previous time frame. If so, it retains its cluster membership from the previous time frame; if not, the hotspot adopts the membership of the nearest hotspot with which it has been clustered. If no such neighbor exists, a hotspot represents the start of a new fire. It is important to note that each hotspot does not necessarily represent an individual fire, so similar to DBSCAN's identification of noise, those clusters that does not pass the threshold of a minimum number of hotspots or exist for a minimum amount of time are labeled noise.

As emphasized by Kisilevich et al. (2009), the selection of spatial resolution and time granularity – and relevance of domain knowledge in their choice – are imperative to the understanding and interpretation of resulting clusters. The incorrect choice for either can be highly influential to the shape and number of clusters discovered, and in the case of satellite hotspot data, will depend on the spatial resolution and temporal frequency at which images are captured. Therefore, we present a visualization heuristic for parameter tuning that enables selection of near-optimal values of the parameters, irrespective of the exact data source.

This paper is organized as follows. The next section provides an introduction to the literature on spatiotemporal clustering and applications to bushfire modeling. Section Algorithm details the steps of the clustering algorithm, and Package introduces its implementation in **spotoroo** on CRAN, including demonstration of the package's key functions and visualization capabilities. We illustrate the clustering algorithm's functionality to study bushfire ignition and spread in Victoria, Australia throughout the 2020 bushfire season in Application, and present a visual heuristic to inform parameter selection. We make use of the Japan Aerospace Exploration Agency (JAXA) Himawari-8 satellite wildfire product (P-Tree System, 2020) that identifies the location and fire radiative power (FRP) of hotspots across East Asia and Australia according to an algorithm developed by Kurihara et al. (2020). Finally, we discuss how the results of the clustering algorithm support researchers in their study of factors that influence the start and spread of wildfires.

Background

Spatiotemporal clustering

Han et al. (2012) identify five categories of clustering algorithms: partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods. Clustering of hotspot data lends itself nicely to density-based methods, which allow for the identification of clusters of various shapes and sizes, without requiring that the user pre-specify number of clusters – these are two limitations of partitioning and hierarchical methods. We therefore focus on a review of density-based methods and refer the reader to Han et al. (2012) for algorithms in other categories and Kisilevich et al. (2009) for appropriate extensions to spatiotemporal data.

Density-based methods separate regions constituting a high density of points separated by low-density regions by identifying pairwise distances between points, and then requiring that a threshold for (Han et al., 2012). Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996) is an influential implementation of this methodology developed in 1996 designed to address three challenges of clustering algorithms: (1) requirements of domain knowledge to determine the hyperparameters, (2) arbitrary shape of clusters and (3) computational efficiency. DBSCAN defines a maximum radius ϵ to construct a neighborhood around each point. It distinguishes between a core point, for which the number of points that fall in its neighborhood meets a minimum threshold (*MinPts*), and a boundary point, whose neighborhood does not meet this threshold, but can be reached via overlapping neighborhoods from that of a core point. Intersecting neighborhoods are defined to be a cluster, while points that cannot be assigned to a cluster are identified as noise. DBSCAN also provides a heuristic to inform selection of ϵ and *MinPts*.

What is often identified as a limitation of DBSCAN – its inability to differentiate between clusters of different densities and those adjacent to each other (Birant and Kut, 2007) – is of less concern for the application to satellite data, which by nature is a set of points corresponding to the equidistant center of pixels on grid of latitudes and longtitudes. However, its application to spatiotemporal clustering problems, which contain at least three dimensions – spatial location (e.g. latitude and longitude) and time – require specification of temporal granularity and treatment of temporal similarity (Kisilevich et al., 2009). As such, several extensions to DBSCAN's spatial clustering functionality have been proposed for spatiotemporal clustering solutions.

ST-DBSCAN (Birant and Kut, 2007) was developed as an extension of DBSCAN's functionality to cluster points according to their non-spatial, spatial, and temporal attributes, and simultaneously address two of DBSCAN's limitations regarding identification of clusters of varying densities and differentiation of adjacent clusters. Therefore, in addition to DBSCAN's original metric to capture the spatial distance between two objects, ST-DBSCAN introduces a second metric that considers similarity of variables associated with temporal neighbors; that is, points observed in consecutive time steps.

Extensions of DBSCAN have been developed to handle incremental updates of clusters where time may impact the density of a neighborhood, and therefore the clustering membership. Incremental DBSCAN (Ester et al., 1998) adjusts DBSCAN to allow for insertion and deletion of points in batch updates in a data warehouse context, with a focus on computational efficiency for implementation. Kalnis et al. (2005) proposes exact and approximate algorithms to identify and extract moving clusters

– that is, a set of objects that move together other over space and time. The authors first partition the movement history of a set of objects into temporal snapshots and use DBSCAN at each time step to cluster objects. Clusters from consecutive snapshots are merged together if they share a minimum number of common objects, called an integrity threshold.

Clustering of satellite hotspot data

DBSCAN has been employed for the clustering and visualization of satellite hotspot data (see for example Nisa et al. (2014) and Hermawati and Sitanggang (2016)), although as discussed, it does not enable the tracking of fire ignition and movement over time.

Fire Spread Reconstruction (FSR) (Loboda and Csiszar, 2007) addresses this limitation with the development of a tree-based algorithm that identifies fire spread in the Russian boreal forest based on active fire detections from MODIS (Moderate Resolution Imaging Spectroradiometer), which has a temporal frequency of six hours. The algorithm proposed by the authors constructs a tree based on three rules: (1) the earliest observed hotspot is the root of the tree, (2) any node is within a 2.5km radius from its parent and (3) any node is observed no later than four days from its parent. When the tree is closed and there are still unassigned hotspots, the algorithm continues at the earliest unassigned hotspot to construct a new tree. Finally, each tree is a cluster, and the earliest hotspot is defined as the ignition point.

FSR's selection of parameters is specific to the region and data product used, and is therefore not immediately generalizable to other sources of satellite hotspot data. Additionally, due to its sequential construction of fires, two that may start in different locations but result in overlapping coverage are considered to be a single fire by the time they intersect. As a result, coverage of each fire may increase dramatically in a short time period, which does not accurately reflect the natural speed of a bushfire.

Algorithm

Our spatiotemporal clustering algorithm consists of four steps, which will be described in detail in this section: (1) divide hotspots according to a user-defined temporal interval, (2) cluster hotspots spatially within each time frame, (3) combine results of consecutive time frames to update hotspot cluster memberships, and (4) identify remaining noise.

1. Divide hotspots into intervals

One of the challenges of satellite hotspot data is cloud cover may obscure a bushfire over several hours, leading to missing observations in the data. As a result, two hotspots observed with a large time difference may preserve direct association. *What is this trying to say, re: direct association?* To account for this possibility, an integer parameter *activeTime* is predetermined to define the maximum time a fire may stay smouldering – while remaining undetectable by satellite before flaring up again – such that we classify the hostpots as the same fire.

Due to the nature of bushfires, earlier hotspots are likely to be the source of later hotspots nearby, so it makes sense to treat the temporal dimension of the hotspot data separately. Our method is to define a sequence of time intervals; within each, we consider only the spatial relationship among hotspots. In other words, the temporal dimension is dropped from consideration within an interval. More precisely, the interval $\boldsymbol{S}_t$ is defined by

$$\boldsymbol{S}_t = [max(1, t - activeTime), t] \quad (t = 1, 2, ..., T),$$

where $max(.)$ is the maximum function, t is the time index, and T is the integer length of the time frame.

For example, if the data set contains 48 hours of hotspot data and the $activeTime = 24\ hours$, there would be 48 intervals defined by the algorithm, $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_{48}$, where

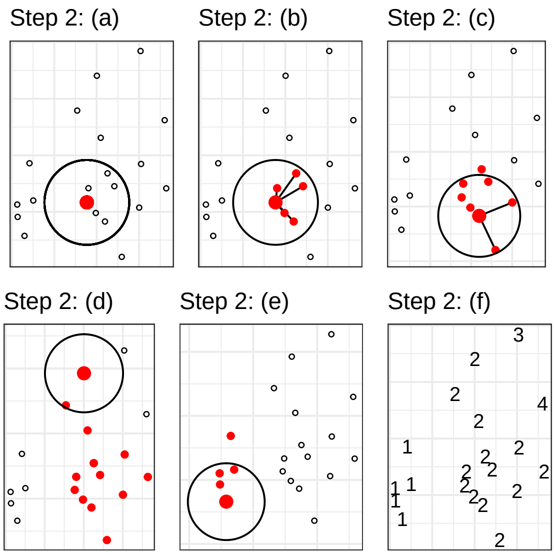


Figure 1: An example of step 2 given 20 hotspots in interval S_t. (a) A hotspot is selected randomly as the first item of list L and the pointer P. Hotspots in list L are in red. Pointer P is drawn with larger marker size. (b) Nearby hotspots of the pointer P are appended to the list L. (c) Move pointer P to the next item of list L and append the nearby hotspots to list L. (d) The cluster is identified via repeating substep (c). (e) Clear the list L, then randomly select an unassigned hotspot to identify another cluster. (f) The final clustering result is produced via repeating substep (d). The labels show the cluster each hotspot belongs to.

$$
\begin{aligned}
S_1 &= [1,1], \\
S_2 &= [1,2], \\
&\dots \\
S_{25} &= [1,25], \\
S_{26} &= [2,26], \\
&\dots \\
S_{47} &= [23,47], \quad and \\
S_{48} &= [24,48].
\end{aligned}
$$

2. Cluster hotspots spatially

The next step is to perform clustering of hotspots within each of these time intervals; the algorithm only needs to address the the spatial relationship between hotspots. A parameter *adjDist* is introduced to represent the potential distance a fire can spread with respect to the temporal resolution of the data. For example, given the temporal resolution of the data is 10-minute, let $AdjDist = 3000m$, then the potential speed of the bushfire is $3000m/10\ min = 18km/h$.

Given $AdjDist > 0\ m$ and a interval S_t, the algorithm performs the following substeps:

(a) Append a randomly selected hotspot h_i to a empty list L, where h_i is the ith hotspot in the interval S_t. And let pointer P points to the first element of the list L.

(b) For every $h_i \notin L$, if $geodesic(h_i, P) \leq AdjDist$, append h_i to the list L.

(c) Move pointer P to the next item of the list L.

(d) Repeat (b) and (c) until the pointer P reaches to the end of the list L.

(e) For all hotspots $h_i \in L$, assign a new membership to them to denote that they belong to a new cluster. Pop these hotspots from the interval S_t. Repeat (a) to (e) if interval S_t is not empty.

(f) Recover the interval S_t and record the memberships.

Figure 1 gives an concise example of this step.

3. Update memberships

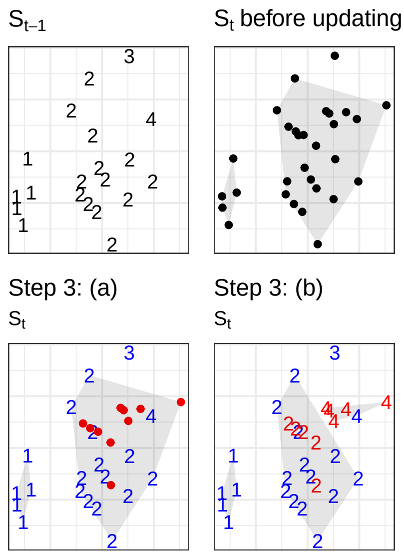


Figure 2: An example of step 3. In this example, there are 30 hotspots belong to interval $\boldsymbol{S}_t$. (a) 20 out of 30 hotspots belong to both interval $\boldsymbol{S}_t$ and interval $\boldsymbol{S}_{t-1}$. Let these hotspots carry over from their memberships in $\boldsymbol{S}_{t-1}$. They are annotated in blue with membership labels. Points in red are the rest 10 hotspots that only belong to interval $\boldsymbol{S}_t$. (b) For each red point, let it carry over from the membership of the nearest blue label which shares the same component (according to the spatial clustering result of this interval) in interval $\boldsymbol{S}_t$.

Once the results of spatial clustering are obtained for each time interval, the next step is to update hotspot memberships by bringing in information from earlier intervals.

This step starts from $t = 2$ and progresses through $t = T$. Given the interval $\boldsymbol{S}_t$, the algorithm performs the following substeps:

(a) Let h_i carries over from its membership in $\boldsymbol{S}_{t-1}$, if h_i belongs to $\boldsymbol{S}_{t-1}$, where h_i is the ith hotspot in the interval $\boldsymbol{S}_t$. These hotspots are collected by a set $\boldsymbol{H}_s = \{h_s^1, h_s^2, ...\}$.

(b) Let $\boldsymbol{H}_c = \{h_c^1, h_c^2, ...\}$, where h_c^i is the ith hotspot in set $\boldsymbol{H}_c$ and h_c^i belongs to $\boldsymbol{S}_t$ but does not belong to $\boldsymbol{S}_{t-1}$. If h_c^i being clustered into the same component with h_s^j in interval $\boldsymbol{S}_t$, let h_c^i carries over from the membership of the nearest h_s^j, where h_s^j is the jth hotspot in the set $\boldsymbol{H}_s$.

Figure 2 gives an example of this step.

4. Handle noise

After performing step 3, all hotspots will have a cluster membership label. However, there may exist some amount of small clusters, for which we define a noise filter in the last step.

Parameter *minPts* is the minimum number of hotspots a cluster may contain and parameter *minTime* is the minimum number of hours a cluster may exist. Any cluster that do not satisfy these two conditions are assigned a membership label of -1 to indicate noise.

Result

This spatiotemporal clustering algorithm applied to satellite hotspot data results in a vector of cluster memberships, the length of which is equal to the number of observations in the original data.

Package

Our spatiotemporal clustering algorithm is implemented in the R package **spotoroo**. The released version can be installed from CRAN using the following code:

```
install.packages("spotoroo")
```

The development version can be installed from Github with:

```
devtools::install_github("TengMCing/spotoroo")
```

The following demonstration will assume the package **spotoroo** has been loaded.

```
library(spotoroo)
```

Clustering Analysis

The main function of the **spotoroo** package is `hotspot_cluster()`, which can be used to implement this spatiotemporal clustering algorithm on satellite hotspot data.

This function requires that three categories of arguments are specified: `hotspots`, `lon`, `lat`, and `obsTime` are used to specify the hotspot data set and its relevant columns; `activeTime`, `adjDist`, `minPts`, and `minTime` are user-defined parameters as defined in the Algorithm section; and arguments `timeUnit` and `timestep` slice observed continuous time into discrete time intervals.

The following code demonstrates the use of `hotspot_cluster()`. Here we set the time interval to be 1 hour with `timeUnit` and `timeStep` such that `activeTime` represents 24 hours and `minTime` represents 3 hours. Parameter `adjDist` is set to be 3000 meters and `minPts` to be 4 hotspots.

```
result <- hotspot_cluster(hotspots = hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          timeUnit = "h",
                          timeStep = 1)
```

The output of this function is a `spotoroo` object, that is a `list` that contains a `data.frame` called `hotspots`, a `data.frame` called `ignition`, and a `list` called `setting`.

```
result
```

```
#> i spotoroo object: 6 clusters | 1070 hot spots (including noise points)
```

The `hotspots` data set contains information on each hotspot including its location, the time it was observed and index in the series of clustered time intervals, its cluster membership assignment, as well as distance to its cluster's ignition location and the amount of time since ignition.

```
head(result$hotspots, 2)
```

```
#>      lon       lat             obsTime timeID membership noise distToIgnition
#> 1 147.46 -37.46000 2020-02-01 05:20:00    809         -1  TRUE              0
#> 2 146.48 -37.93999 2020-01-02 06:30:00     90         -1  TRUE              0
#>   distToIgnitionUnit timeFromIgnition timeFromIgnitionUnit
#> 1                  m   718.8333 hours                    h
#> 2                  m   718.8333 hours                    h
```

The `ignition` data set contains information on each cluster including the ignition location, the observed time of ignition, the number of hotspots in the cluster and the length of time over which it was observed. In addition, the coordinates of each ignition point represents the centroid of a cluster's hotspots in the first time interval in which the cluster was observed.

```
head(result$ignition, 2)
```

```
#>   membership    lon    lat             obsTime timeID obsInCluster
#> 1          1 149.30 -37.77 2019-12-29 13:10:00      1          146
#> 2          2 146.72 -36.84 2020-01-08 01:40:00    229          165
#>   clusterTimeLen clusterTimeLenUnit
#> 1 116.1667 hours                  h
#> 2 148.3333 hours                  h
```

The package also provides a brief report of the clustering results in the generic function `summary()`.

```
summary(result)
```

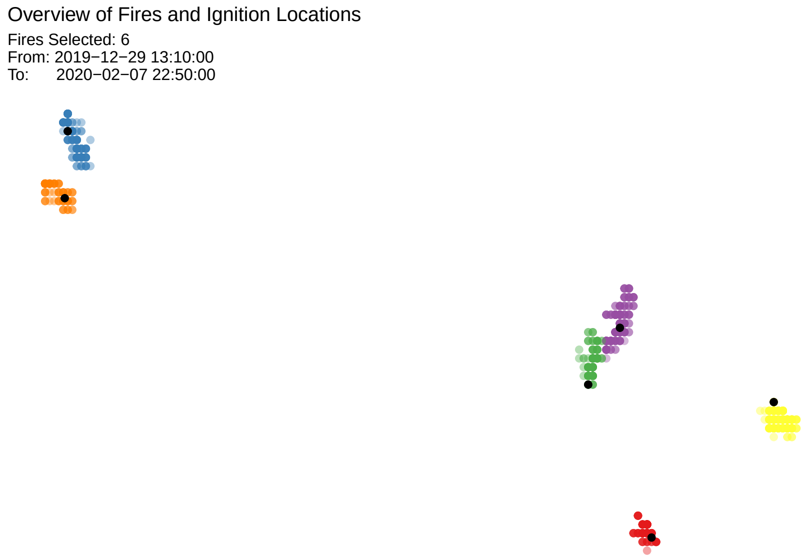


Figure 3: Default plot.

Extract a subset of clusters

The clustering result obtained from the `hotspot_cluster()` function is a `spotoroo` object which, due to its list structure, may require conversion for further analysis. Therefore, the package provides a function `extract_fire()` to convert a `spotoroo` object to a `data.frame`.

To keep all information from the clustering result including noise points, set `noise = TRUE` within the `extract_fire()` function.

```
all_fires <- extract_fire(result, noise = TRUE)
```

If the user provides a vector to the argument `cluster`, the function extracts hotspots in the corresponding clusters from the clustering result.

```
fire_1_and_2 <- extract_fire(result, cluster = c(1, 2), noise = FALSE)
```

Did we want to show any results here?

Visualize the clustering result

The package provides three methods to visualize the clustering results, which all can be produced by the generic function `plot()`.

The default plot produced is shown in Figure 3. It is a scatter plot of the clusters and their ignition locations that demonstrates the spatial distribution of the hotspots in each cluster.

According to the plot, there are total of six clusters: four are at the bottom right of the plot, and two are at the top left of the plot. The black dots represent the ignition location of each cluster. These bushfires were observed from December 2019 through Feburary 2020 during the 2019-2020 Australian bushfire season.

```
plot(result)
```

Figure 4 shows the path of the fire movement, which can be produced by setting `type = 'mov'`. The argument `step` controls the time difference between successive steps, and using a small value of `step` produces a more complex path. *Maybe explain why – make explicit relationship between step and timeInterval.* The fire movement is computed by the `get_fire_mov()` function.

In this plot, the triangle is the start point and the circle is the end point. It shows that fire 1, 4 and 5 moved toward southeast, fires 2 and 6 moved northwest and fire 3 moved southwest.

```
plot(result, type = "mov", step = 12)
```

Figure 5 shows the time line of clusters, which can be produced by setting `type = 'timeline'` to study the intensity of the bushfire season. In this plot, the green dots are hotspots belonging to

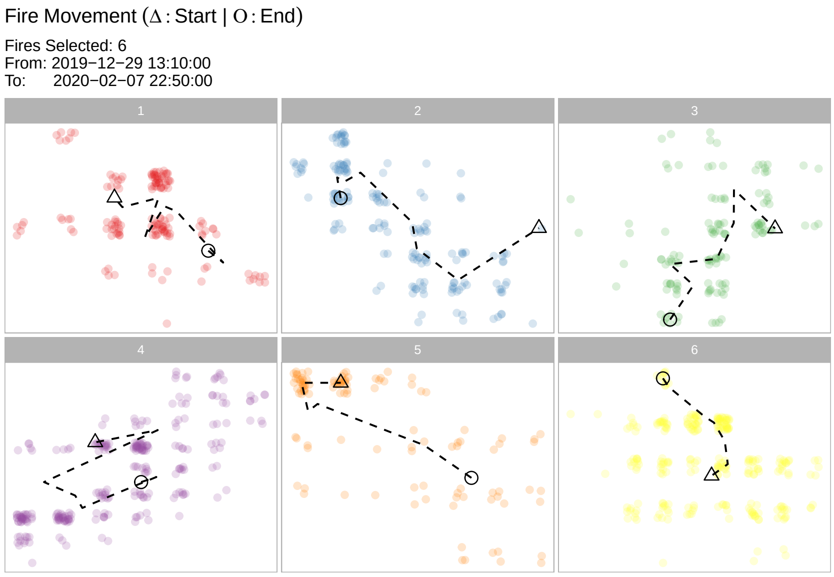


Figure 4: Fire movement plot.

different clusters and the green wave is the density plot for all the hotspots. The orange dots represent noise. From this plot, we can conclude that most of the hotspots were observed in mid-January of 2020.

```
plot(result, type = "timeline")
```

Application

In this section, we present an application to illustrate how this algorithm may be used to study bushfire ignition.

Data source

We use the wild fire product (produced from Himawari-8) supplied by the P-Tree System, Japan Aerospace Exploration Agency (JAXA) (2020) as the data source. This wild fire product is referred to as the Himawari-8 hotspot data in the rest of the paper. This data set contains records of 1,989,572 hotspots from October 2019 to March 2020 in the full disk of 140 °east longitude with 0.02 °spatial resolution and a 10-minute temporal resolution.

We pre-process the data by selecting hotspots within the boundary of Victoria, Australia, and filtering hotspots with a threshold – irradiance over 100 watts per square metre – as suggested by landscape ecologist and spatial scientist Dr. Grant Williamson (2020) to reduce noise from the background.

The final hotspot data set contains 75,936 observations with ID, longitude, latitude and observed date as fields. The overall distribution of these hotspots is shown in Figure 6.

Clustering the Himawari-8 hotspots

To perform the clustering algorithm on the Himawari-8 hotspot data, we first transform the observed time to time indices by setting the time difference between two successive intervals to be 1 hour. Then, the algorithm's remaining parameters are defined with an `activeTime` of 24 time indices representing 24 hours and `minTime` is 3 hours; `adjDist` is 3000 meters and `minPts` is 3 hotspots. We further discuss parameter selection in section Parameter tuning.

```
result <- hotspot_cluster(hotspots = him_hotspots,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
```

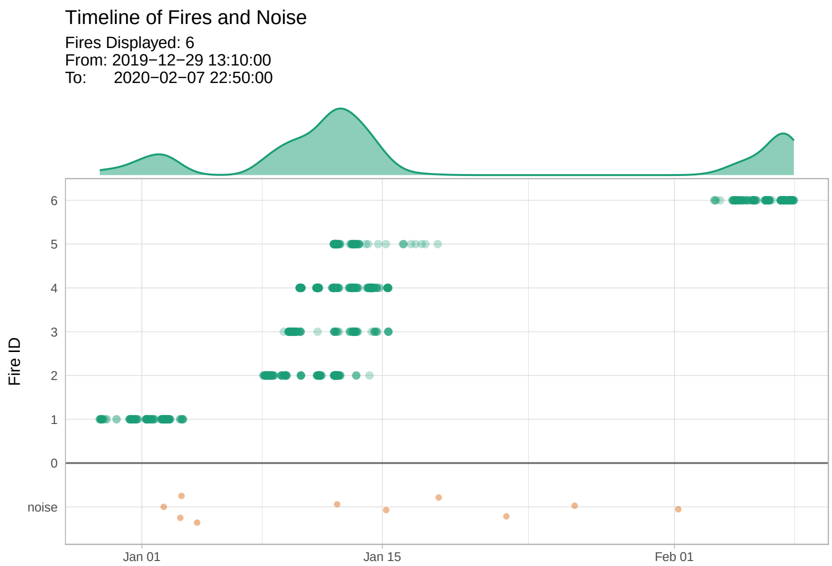


Figure 5: Timeline of clusters.

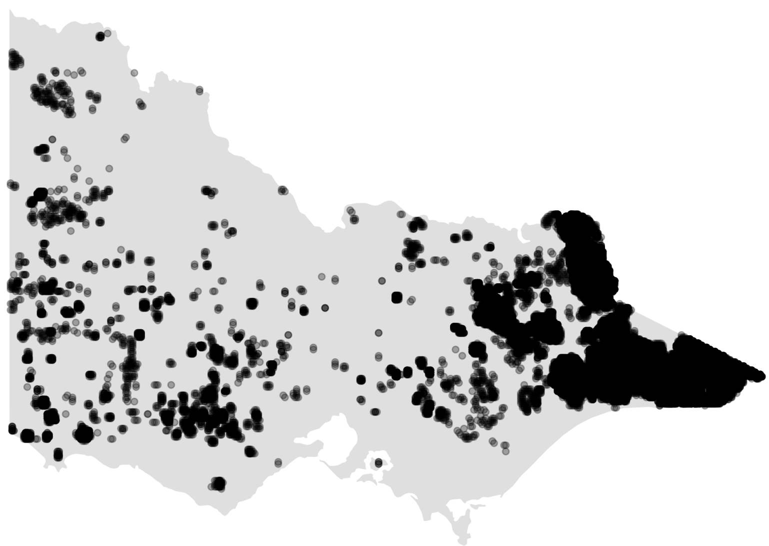


Figure 6: The distribution of hotspots in Victoria during 2019-2020 Australia bushfire season.

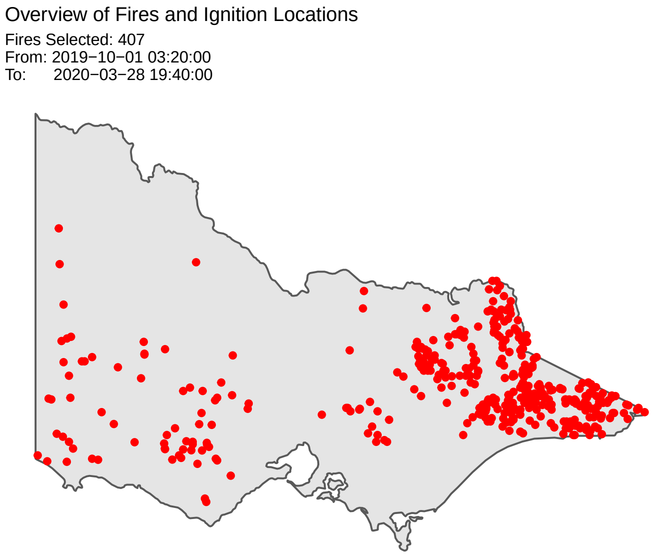


Figure 7: The distribution of bushfire ignitions in Victoria during 2019-2020 Australian bushfire season.

```
adjDist = 3000,
minPts = 4,
minTime = 3,
timeUnit = "h",
timeStep = 1)
```

The clustering result shows that 407 bushfires are identified from 75936 hotspots.

```
result
```

```
#> i spotoroo object: 407 clusters | 75936 hot spots (including noise points)
```

Determining the ignition point and time for individual fires

Based on the clustering results, the ignition location for each cluster can be computed. The earliest hotspot of a cluster is defined to be its ignition point, unless there are multiple hotspots in the first interval the cluster is observed, in which case the centroid of these hotspots is recorded as the fire's ignition location. According to this method, ignition points over 6 months can be produced using the following code:

```
plot(result, bg = plot_vic_map(), hotspot = FALSE)
```

The result is given in Figure 7. From this plot, we observe that most of the fires ignited in the east of Victoria and some fires were ignited in the south west of the Victoria. Very few fires started near Melbourne, which is located towards the middle of Victoria.

We also study the ignition time for individual fires using the following code:

```
plot(result, type = "timeline", mainBreak = "1 month", dateLabel = "%b %d, %y")`.
```

The result is given in Figure 8. As we can see, the majority of fires were ignited from mid-December of 2019 to mid-January of 2020. We note significant noise identified in mid-December, which may suggest there are some undetected fires. *Due to cloud cover during that time? Or because our algorithm mis-classified them?*

Tracking fire movement

Display showing how a fire moves over time, maybe two or more fires

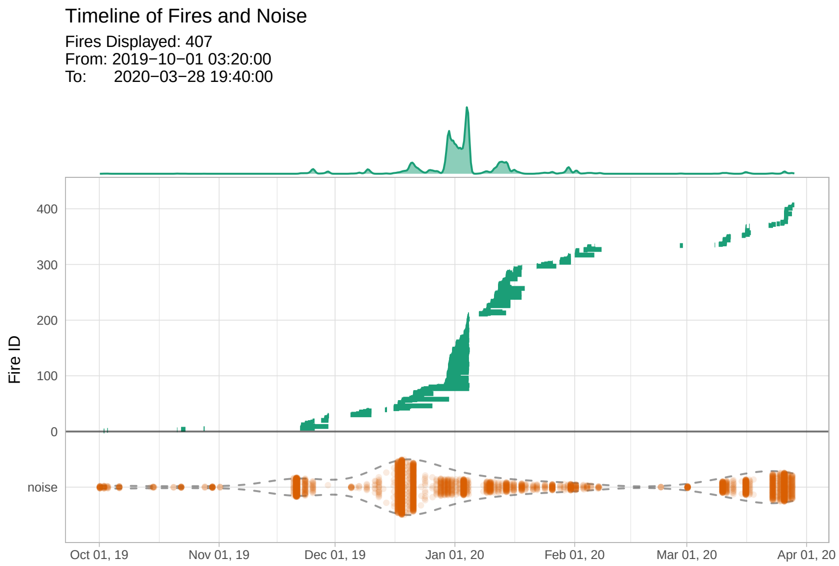
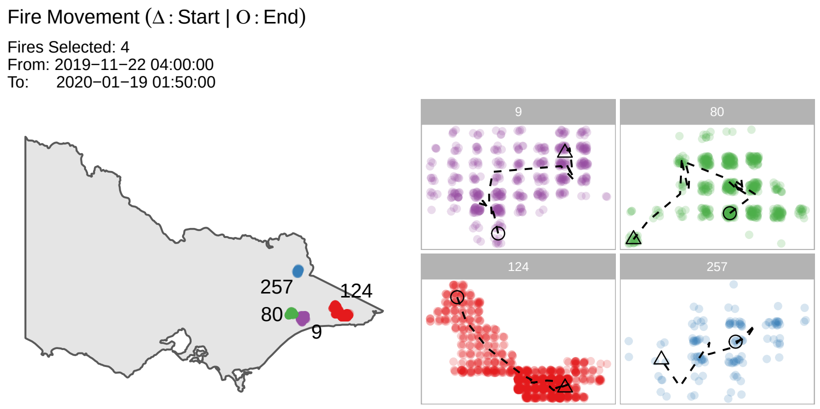


Figure 8: Timeline of 2019-2020 Victorian bushfire season.



Allocating resources for future fire prevention

Merging data with camp sites, CFA, roads, . . .

Effects of parameter choices

Two key user-defined parameters for the clustering algorithm are *AdjDist* and *ActiveTime*. The optimal choice of these two parameters is unknown but can be tuned using a visualization tool.

Considering the relationships between *AdjDist*, *ActiveTime* and the number of clusters in the clustering result, increasing either *AdjDist* or *ActiveTime* usually reduces the number of clusters. There is an exception when there are large gaps between clusters spatially and temporally *(how would we defined large?)*, in which case increasing two parameters does not significantly reduce the number of clusters. Given that one of the metrics to evaluate goodness of fit of the clustering result is the gap between clusters, *(this is the first time we've mentioned cluster evaluation)* the optimal choice of *AdjDist* and *ActiveTime* can be chosen when they have minimum impact on the number of clusters. With this methodology, the optimal *ActiveTime* and *AdjDist* approaches infinitey as the number of clusters approach to 1, so a restriction needs to be applied on this optimization. Increasing of *ActiveTime* and *AdjDist* should therefore only be allowed when there is a major fall in the number of clusters. Based

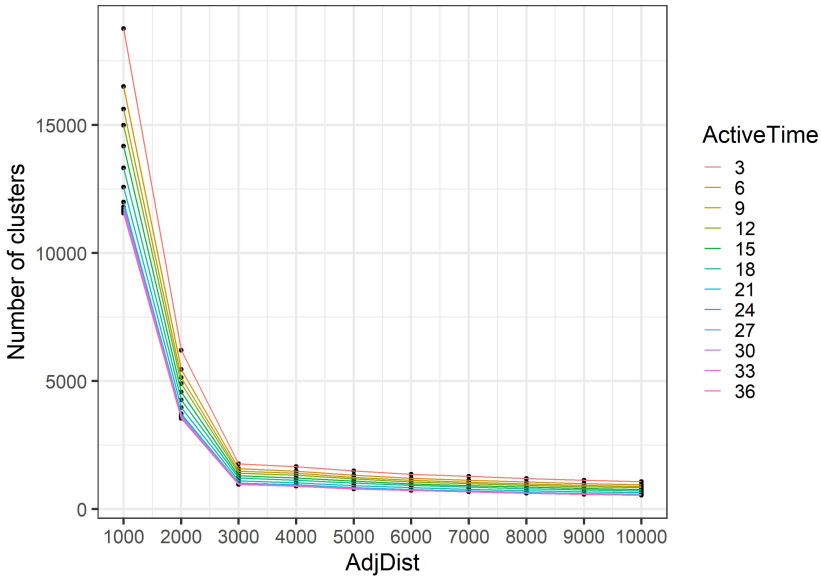


Figure 9: A visualization tool for parameter tuning . It works like a scree plot. Major falls of the number of clusters are observed when $AdjDist < 3000$ so the reasonable choice of $AdjDist$ is 3000m.

on this rule, a visualization tool inspired by the scree plot used in the principal component analysis is developed. Similar to the scree plot, the user needs to determine the *ActiveTime* and *AdjDist* that captures the greatest decrease in the number of clusters. Figure 9 and 10 show the parameter tuning process by using this visualization tool. The final choice of *ActiveTime* is 24 hours and *AdjDist* is 3000 metres.

Parameter tuning

Summary

Bibliography

N. J. Abram, B. J. Henley, A. Sen Gupta, T. J. R. Lippmann, H. Clarke, A. J. Dowdy, J. J. Sharples, R. H. Nolan, T. Zhang, M. J. Wooster, J. B. Wurtzel, K. J. Meissner, A. J. Pitman, A. M. Ukkola, B. P. Murphy, N. J. Tapper, and M. M. Boer. Connections of climate change and variability to large and extreme forest fires in southeast australia. *Communications Earth & Environment*, 2(1):8, 2021. doi: 10.1038/s43247-020-00065-8. URL https://doi.org/10.1038/s43247-020-00065-8. [p1]

D. Birant and A. Kut. St-dbscan: An algorithm for clustering spatial-temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007. ISSN 0169-023X. doi: https://doi.org/10.1016/j.datak.2006.01.013. URL https://www.sciencedirect.com/science/article/pii/S0169023X06000218. Intelligent Data Mining. [p2]

CSIRO and A. G. B. of Meteorology. State of the climate 2020, 2020. URL http://www.bom.gov.au/state-of-the-climate/documents/State-of-the-Climate-2020.pdf. [p1]

P. Deb, H. Moradkhani, P. Abbaszadeh, A. S. Kiem, J. Engstrom, D. Keellings, and A. Sharma. Causes of the widespread 2019-2020 australian bushfire season. *Earth's Future*, 8(11):e2020EF001671, 2020. doi: https://doi.org/10.1029/2020EF001671. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020EF001671. [p1]

M. Ester, H.-P. Kriegel, J. Sander, and 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, page 226–231. AAAI Press, 1996. [p1, 2]

M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and 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, page 323–333, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605665. [p2]

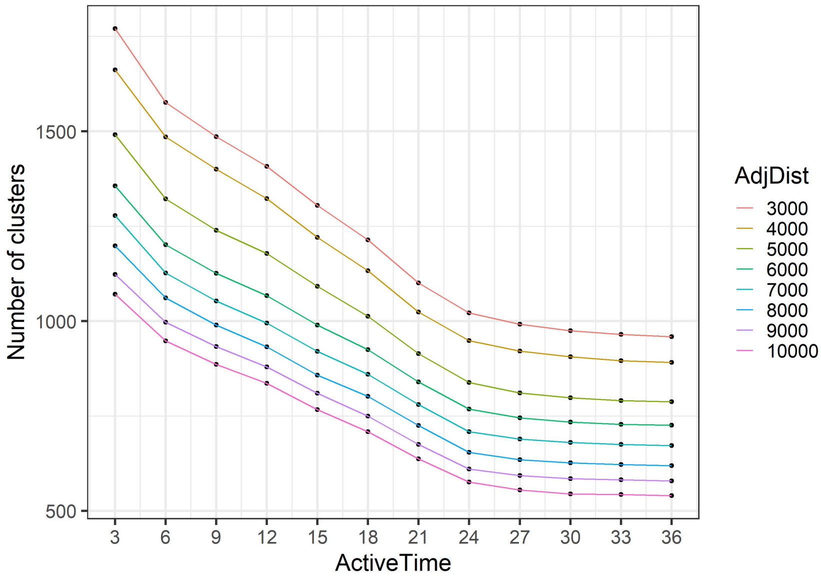


Figure 10: Major falls of the number of clusteres are observed when *ActiveTime* < 24, so the reasonable choice of *ActiveTime* is 24 hours.

A. I. Filkov, T. Ngo, S. Matthews, S. Telfer, and T. D. Penman. Impact of australia's catastrophic 2019/20 bushfire season on communities and environment. retrospective analysis and current trends. *Journal of Safety Science and Resilience*, 1(1):44–56, 2020. ISSN 2666-4496. doi: https://doi.org/10.1016/j.jnlssr.2020.06.009. URL https://www.sciencedirect.com/science/article/pii/S2666449620300098. [p1]

L. Giglio, W. Schroeder, and C. O. Justice. The collection 6 modis active fire detection algorithm and fire products. *Remote Sensing of Environment*, 178:31–41, 2016. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2016.02.054. URL https://www.sciencedirect.com/science/article/pii/S0034425716300827. [p1]

J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, 3rd ed.* Morgan Kaufman, 2012. [p2]

R. Hermawati and I. S. Sitanggang. Web-based clustering application using shiny framework and dbscan algorithm for hotspots data in peatland in sumatra. *Procedia Environmental Sciences*, 33:317–323, 2016. ISSN 1878-0296. doi: https://doi.org/10.1016/j.proenv.2016.03.082. URL https://www.sciencedirect.com/science/article/pii/S1878029616002474. The 2nd International Symposium on LAPAN-IPB Satellite (LISAT) for Food Security and Environmental Monitoring. [p3]

E. Jang, Y. Kang, J. Im, D.-W. Lee, J. Yoon, and S.-K. Kim. Detection and monitoring of forest fires using himawari-8 geostationary satellite data in south korea. *Remote Sensing*, 11(3), 2019. ISSN 2072-4292. doi: 10.3390/rs11030271. URL https://www.mdpi.com/2072-4292/11/3/271. [p1]

P. Kalnis, N. Mamoulis, and S. Bakiras. On discovering moving clusters in spatio-temporal data. In C. Bauzer Medeiros, M. J. Egenhofer, and E. Bertino, editors, *Advances in Spatial and Temporal Databases*, pages 364–381, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. [p2]

S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo. Spatio-temporal clustering. In *Data mining and knowledge discovery handbook*, pages 855–874. Springer, 2009. [p2]

Y. Kurihara, K. Tanada, H. Murakami, and M. Kachi. 2020: Australian bushfire captured by ahi/himawari-8 and sgli/gcom-c. JpGU-AGU Joint Meeting, 2020. [p2]

T. Loboda and I. Csiszar. Reconstruction of fire spread within wildland fire events in northern eurasia from the modis active fire product. *Global and Planetary Change*, 56(3):258–273, 2007. ISSN 0921-8181. doi: https://doi.org/10.1016/j.gloplacha.2006.07.015. URL https://www.sciencedirect.com/science/article/pii/S0921818106001871. Northern Eurasia Regional Climate and Environmental Change. [p1, 3]

K. K. Nisa, H. A. Andrianto, and R. Mardhiyyah. Hotspot clustering using dbscan algorithm and shiny web framework. In *2014 International Conference on Advanced Computer Science and Information System*, pages 129–132, 2014. doi: 10.1109/ICACSIS.2014.7065840. [p3]

P-Tree System. JAXA Himawari Monitor - User's Guide, 2020. URL https://www.eorc.jaxa.jp/ptree/userguide.html. [p2, 8]

C. H. Wickramasinghe, S. Jones, K. Reinke, and L. Wallace. Development of a multi-spatial resolution approach to the surveillance of active fire lines using himawari-8. *Remote Sensing*, 8(11), 2016. ISSN 2072-4292. doi: 10.3390/rs8110932. URL https://www.mdpi.com/2072-4292/8/11/932. [p1]

G. Williamson. Example code to generate animation frames of Himawari-8 hotspots, 2020. URL https://gist.github.com/ozjimbob/80254988922140fec4c06e3a43d069a6. [p8]

G. Xu and X. Zhong. Real-time wildfire detection and tracking in australia using geostationary satellite: Himawari-8. *Remote Sensing Letters*, 8(11):1052–1061, 2017. doi: 10.1080/2150704X.2017.1350303. URL https://doi.org/10.1080/2150704X.2017.1350303. [p1]

Weihao Li
Monash University
Econometrics and Business Statistics

weihao.li@monash.edu

Emily Dodwell
??
line 1
line 2

emdodwell@gmail.com

Dianne Cook
Monash University
Econometrics and Business Statistics

dicook@monash.edu