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

Bushfires are a major problem for Australia, and many other parts of the globe. There is concern that as the climate becomes hotter, and drier, that the impact of fires becomes much more severe and extensive. In Australia, the 2019-2020 fires were the worst on record causing extensive ecological damage, as well as damage to agricultural resources, properties and infrastructure. The Wollemi pine, rare prehistoric trees, required special forces intervention to prevent the last stands in the world, in remote wilderness areas, from being turned into ash.

Contributing to the problem is that many fires started in very remote areas, locations deep into the temperate forests ignited by lightning, that are virtually impossible to access or to monitor. Satellite data provides a possible solution to this, particularly remotely sensed hotspot data, which may be useful in detecting new ignitions and movements of fires. Understanding fires in remote areas using satellite data may provide some help in developing effective strategies for mitigating bushfire impact.

This work addresses this topic. Using hotspot data, can we cluster in space and time, in order to determine (1) points of ignition and (2) track the movement of bushfires. The algorithm is implemented in the R package **spotoroo**.

This paper is organised as follows. The next section provides an introduction to the literature on spatiotemporal clustering and bushfire modeling and dynamics. Section Algorithm describes the clustering algorithm, and section Application illustrates how the resulting data can be used to study bushfire ignition.

Background

literature review

Algorithm

Data source

The illustration of this algorithm will 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 will be referred as the hotspot data in this paper. It contains records of 1989572 hotspots from October 2019 to March 2020 in the full disk of 140 °east longitude with 0.02 °spatial resolution and 10 minutes temporal resolution.

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

The final hotspot dataset contains 75936 observations with ID, longitude, latitude and observed date as fields. The overall distribution of these hotspots is shown in Figure 1.

Steps

The spatiotemporal clustering algorithm is consist of 3 steps, (1) divide hotspots into intervals, (2) cluster hotspots spatially, and (3) update the memberships. These three steps will be described in details in the rest of the section.

1. Divide hotspots into intervals

Despite hotspot data can be clustered using ordinary algorithms, like K-means, in the three-dimensional Euclidean space, the clustering results could be highly sensitive to the scaling of the

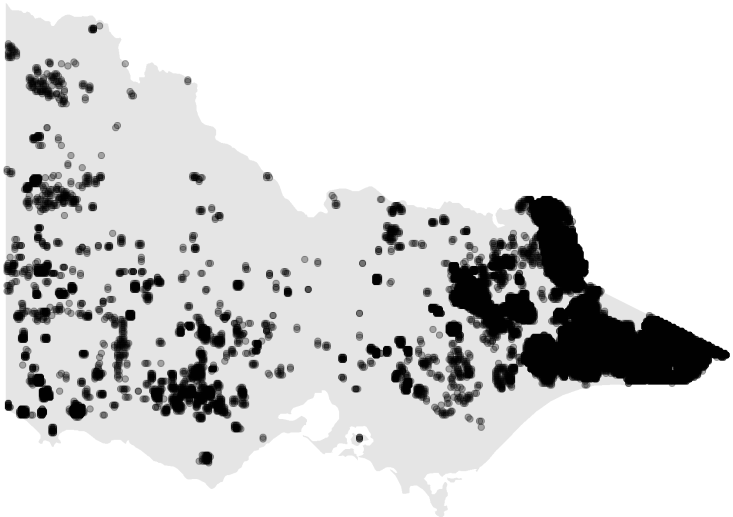


Figure 1: The distribution of hotspots in Victoria during 2019-2020 Australia bushfire season.

temporal dimension (Kisilevich et al., 2009). Besides, one of the characteristics of the hotspot data is cloud cover could lead to missing observations of a bushfire in several hours. This suggests that hotspots with long intervals may present connections. One possible solution to these two issues is to divide hotspot data into intervals then perform clustering spatially only, such that the temporal dependence between hotspots could be predetermined by a parameter *ActiveTime*. The interpretation of *ActiveTime* is the time a fire can stay smouldering but undetectable by satellite before flaring up again.

Given a certain value of *ActiveTime* and an integer length of the time frame T, the algorithm will define several intervals,

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

,where both T and t have the same unit as *ActiveTime*.

For example, if the dataset contains 48 hours of hotspot data and the $ACtiveTime = 24\ hours$, there will be 48 intervals defined by the algorithm, $\boldsymbol{S}_1, \boldsymbol{S}_2, .., \boldsymbol{S}_{48}$, where

$$\begin{aligned}
\boldsymbol{S}_1 &= [1, 1] \\
\boldsymbol{S}_2 &= [1, 2] \\
&... \\
\boldsymbol{S}_{25} &= [1, 25] \\
\boldsymbol{S}_{26} &= [2, 26] \\
&... \\
\boldsymbol{S}_{47} &= [23, 47] \\
\boldsymbol{S}_{48} &= [24, 48]
\end{aligned}$$

2. Cluster hotspots spatially

The previous step breaks the temporal dimension. Hence, the following step only needs to address the hotspots spatially by introducing another parameter *AdjDist*. *AdjDist* represents the potential distance a fire can spread with respect to the temporal resolution of the data. For example, let $AdjDist = 3000m$ and the temporal resolution of the data is 10-minute, then the potential speed of the bushfire is $3000m/10\ min = 18km/h$.

Given a fixed value of *AdjDist* and the interval $\boldsymbol{S}_t$, the algorithm will:

(a) Append a randomly selected hotspot h_i to a empty list $\boldsymbol{L}$, where h_i is the ith hotspot in the interval $\boldsymbol{S}_t$, and let pointer $\boldsymbol{P}$ points to the first element of the list $\boldsymbol{L}$.

(b) Visit every h_i where $h_i \notin \boldsymbol{L}$. If $geodesic(h_i, \boldsymbol{P}) \leq AdjDist$, append h_i to list $\boldsymbol{L}$.

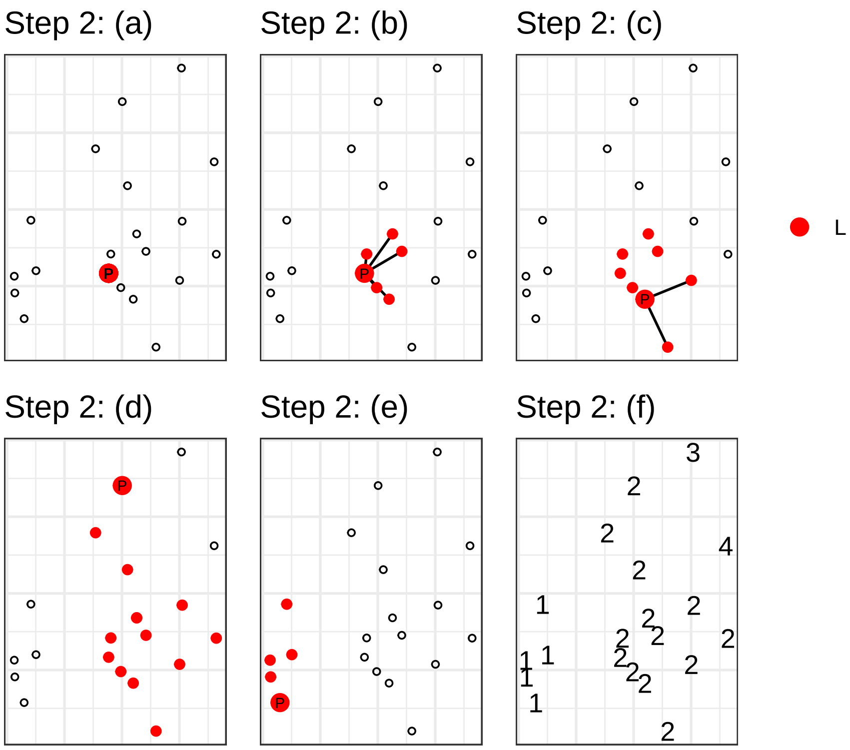


Figure 2: An example of step 2 given 20 hotspots in interval $\boldsymbol{S}_t$. (a) A hotspot is selected randomly as the first item of list $\boldsymbol{L}$ and the pointer $\boldsymbol{P}$. Hotspots in list $\boldsymbol{L}$ are in red. Pointer $\boldsymbol{P}$ is drawn with larger marker size. (b) Nearby hotspots of the pointer $\boldsymbol{P}$ are appended to the list $\boldsymbol{L}$. (c) Move pointer $\boldsymbol{P}$ to the next item of list $\boldsymbol{L}$ and append the nearby hotspots to list $\boldsymbol{L}$. (d) The first cluster is identified via repeating substep (c). (e) Clear the list $\boldsymbol{L}$, 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.

(c) Move pointer $\boldsymbol{P}$ to the next item of the list $\boldsymbol{L}$.

(d) Repeat (b) and (c) till the pointer $\boldsymbol{P}$ reaches to the end of the list $\boldsymbol{L}$.

(e) For all hotspots $h_i \in \boldsymbol{L}$, assign a new membership to them. Pop these hotspots from the interval $\boldsymbol{S}_t$. Repeat (a) to (e) if interval $\boldsymbol{S}_t$ is not empty.

(f) Recover the interval $\boldsymbol{S}_t$ and record the memberships.

Figure 2 gives an concise example of this step.

3. Update the memberships

With clustering results for each interval, the next step is to update the memberships by bringing in information from earlier intervals.

This step starts from $t = 2$ till $t = T$. Given the interval $\boldsymbol{S}_t$, the algorithm will,

(a) Let h_i succeeds its membership from $\boldsymbol{S}_{t-1}$, 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) Set $\boldsymbol{H}_c = \{h_c^1, h_c^2, ...\}$, where h_c^i is the ith hotspot in set $\boldsymbol{H}_c$. 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 $\boldsymbol{S}_t$, h_c^i succeeds the membership from the nearest h_s^j, where h_s^j is the jth hotspot in set $\boldsymbol{H}_s$.

Figure 3 gives an example of this step.

Results

The result of this spatiotemporal clustering algorithm applied on the hotspot data is a vector of memberships with length equals to 75936.

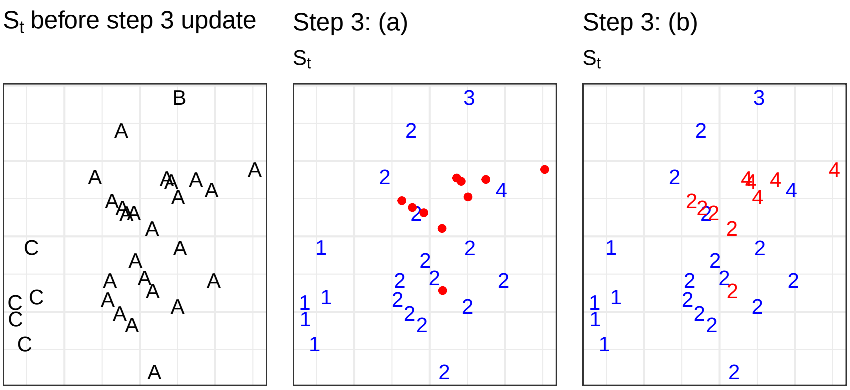


Figure 3: An example of step 3. In this example, there are 30 hotspots belong to interval S_t. (a) 20 out of 30 hotspots belong to both interval S_t and interval S_{t-1}. These hotspots succeed their memberships from S_{t-1}. They are annotated in blue with membership labels. Points in red are the rest 10 hotspots that only belong to interval S_t. (b) For each red point, succeeds the nearest blue label that shares the same component (according to the left plot) with that red point in interval S_t.

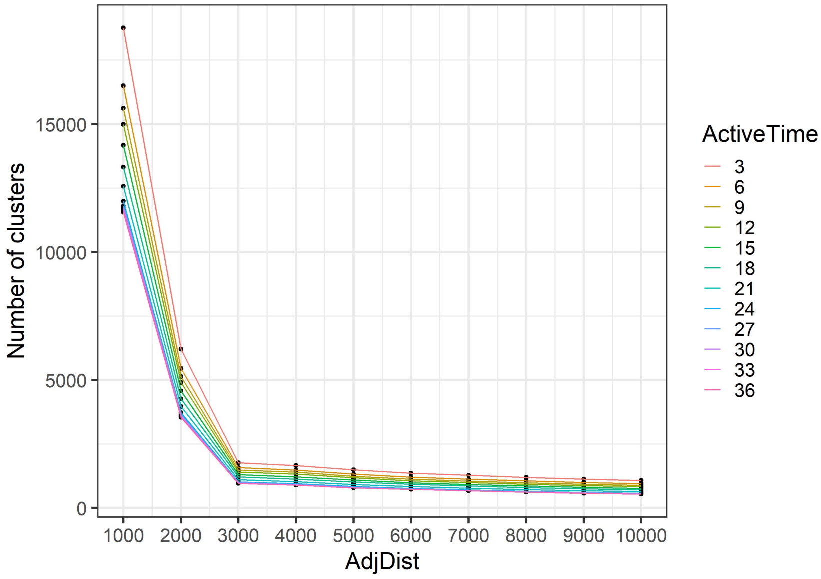


Figure 4: A visualization tool for parameter tuning . It works like a scree plot. Major falls of the number of clusters are observed when $AdjDist < 3000$ so the reasonable choice of $AdjDist$ is 3000m.

Effects of parameter choices

There are two parameters that being introduced in the outline of the algorithm, which are $AdjDist$ and $ActiveTime$. The optimal choice of these two parameters is not known but can be tuned using a visualization tool.

Considering the relationships between $AdjDist$, $ActiveTime$ and the number of clusters in the clustering result, increase either $AdjDist$ or $ActiveTime$ will usually reduce the number of clusters. However, if there are large gaps between clusters spatially and temporally, increase these two parameters will not significantly reduce the number of clusters. Given one of the metrics to evaluate the goodness of the clustering result is the gap between clusters, the optimal choice of $AdjDist$ and $ActiveTime$ can be chosen when they have minimum impact on the number of clusters. However, under this setting, the optimal $ActiveTime$ and $AdjDist$ will approach to infinitely as the number of clusters approach to 1. Hence, a restriction needs to be applied on this optimization. Increase of $ActiveTime$ and $AdjDist$ will only be allowed when there is a major fall of the number of clusters. Based on this rule, a visualization tool inspired by the scree plot used in the principal component analysis is developed. Similar to the scree plot, users need to determine the $ActiveTime$ and $AdjDist$ to capture most of the decrease of the number of clusters. Figure 4 and 5 show the parameter tuning process by using this visualization tool. The final choice of $ActiveTime$ is 24 hours and $AdjDist$ is 3000 metres.

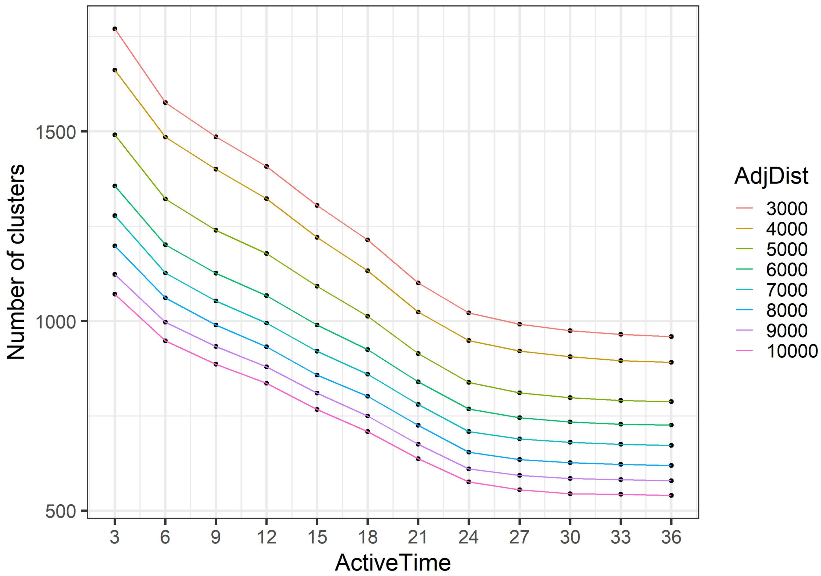


Figure 5: Major falls of the number of clusteres are observed when $ActiveTime < 24$, so the reasonable choice of $ActiveTime$ is 24 hours.

Application

Determining the ignition point and time for individual fires

Based on the clustering result, ignition location for each cluster can be computed. The strategy is to select the earliest hotspot of a cluster as its ignition point. Besides, if there are multiple earliest hotspots belong to the same cluster, the centroid of these hotspots is used as the ignition location. According to this method, ignition points over 6 months are given in Figure 6 and Figure 7.

Tracking fire movement

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

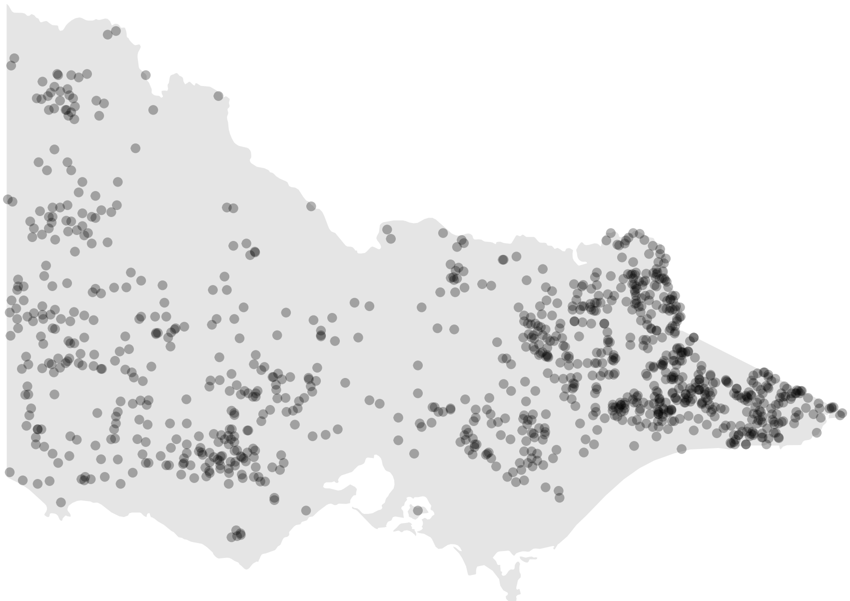


Figure 6: The distribution of bushfire ignitions in Victoria during 2019-2020 Australian bushfire season.

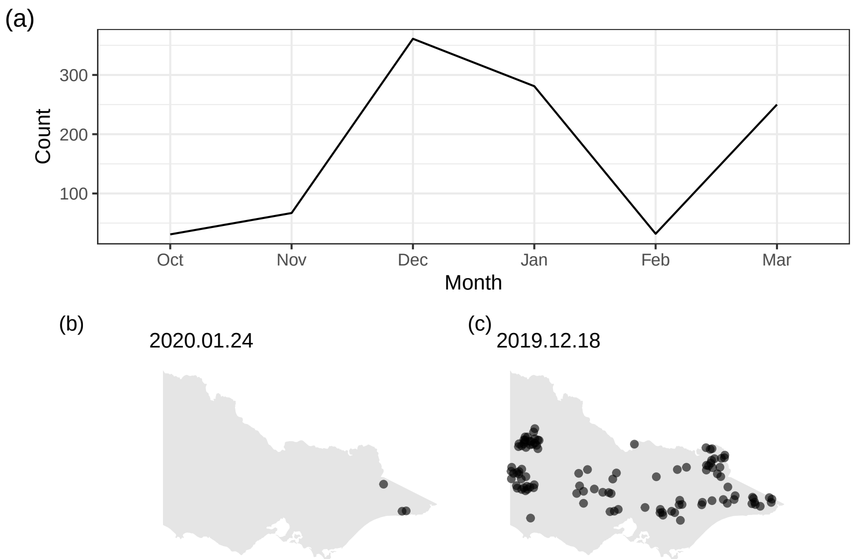
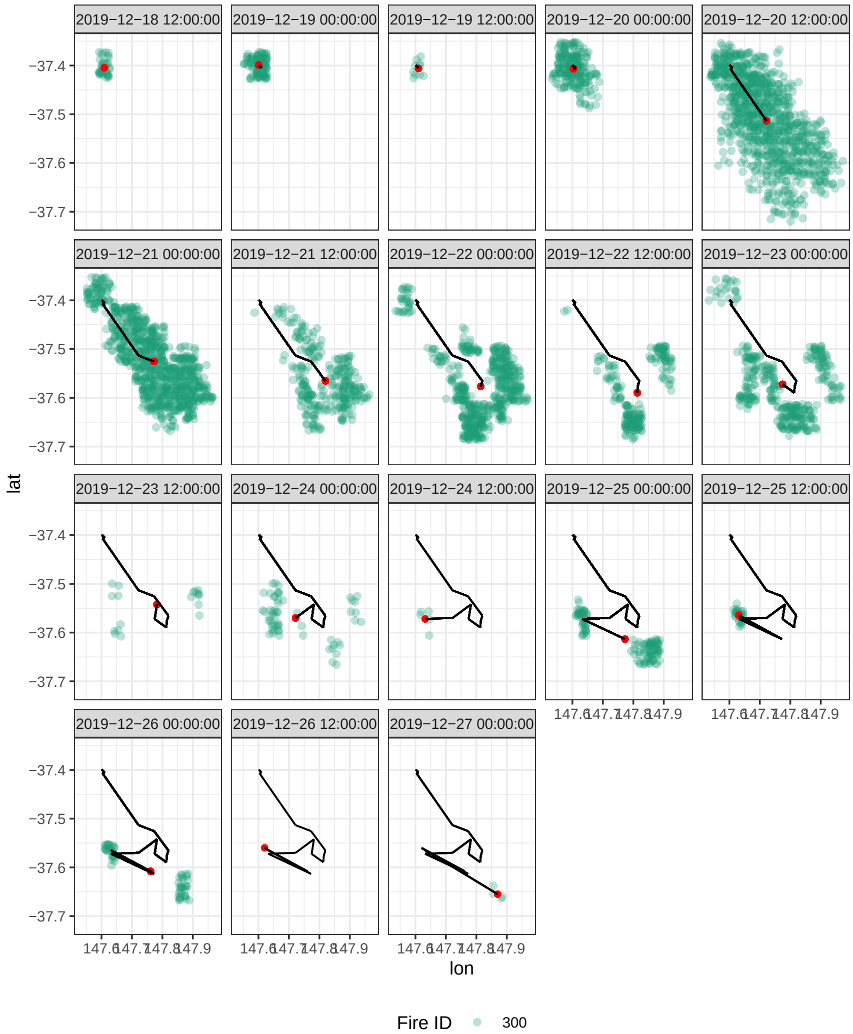


Figure 7: (a) Number of bushfires ignited from October 2019 to March 2020. (b) The distribution of the bushfire ignitions on a light day (c) and a heavy day. There are 3 ignitions on January 24, 2020 and 106 ignitions on December 18, 2019.



Allocating resources for future fire prevention

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

Implementation

The algorithm is available in the R package **spotoroo**.

Installation

The package can be installed from CRAN using

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

and the developmental version from github using

```
install.packages("remotes")
remotes::install_github("TengMCing/spotoroo")
```

Usage

A sample data set is provided with the package, to illustrate its use. The function `hotspot_cluster` performs the spatial clustering. Here we have called it fir the sample data, specifying the spatial and temporal variables (lon, lat, obsTime), and several parameters to the algorithm. A summary of the results is printed when the algorithm completes.

```
library(spotoroo)

#>
#> Attaching package: 'spotoroo'

#> The following objects are masked _by_ '.GlobalEnv':
#>
#>     hotspots, vic_map

library(tidyverse)
result <- hotspot_cluster(hotspots_fin,
                          lon = "lon",
                          lat = "lat",
                          obsTime = "obsTime",
                          activeTime = 24,
                          adjDist = 3000,
                          minPts = 4,
                          minTime = 3,
                          ignitionCenter = "mean",
                          timeUnit = "h",
                          timeStep = 1)

#>

#> ------------------------------ SPOTOROO 0.0.0.9000 -----------------------------

#>

#> -- Calling Core Function : `hotspot_cluster()` --

#>

#> -- 1 time index = 1 hours

#> v Transform observed time > time indexes

#> i 970 time indexes found

#>

#> -- activeTime = 24 time indexes | adjDist = 3000 meters

#> v Cluster

#> i 16 clusters found (including noise)

#>

#> -- minPts = 4 | minTime = 3

#> v Handle noise

#> i 6 clusters left

#> i noise proportion : 0.934579439252336 %

#>

#> -- ignitionCenter = 'mean'

#> v Compute ignition points for clusters

#> i average hotspots : 176.7

#> i average duration : 131.9 hours

#>

#> -- Time taken = 0 mins 3 secs for 1070 hotspots
```

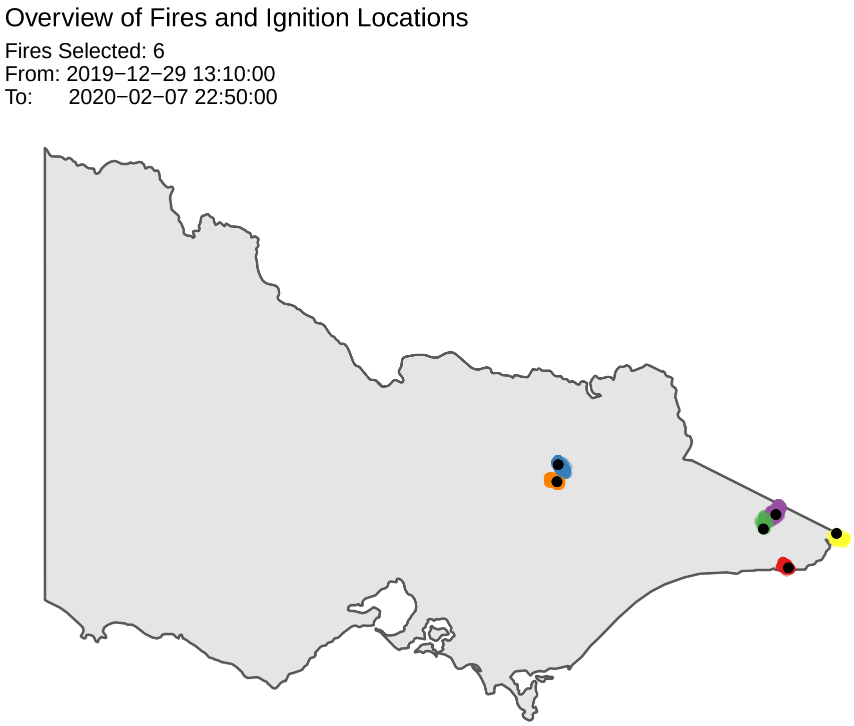


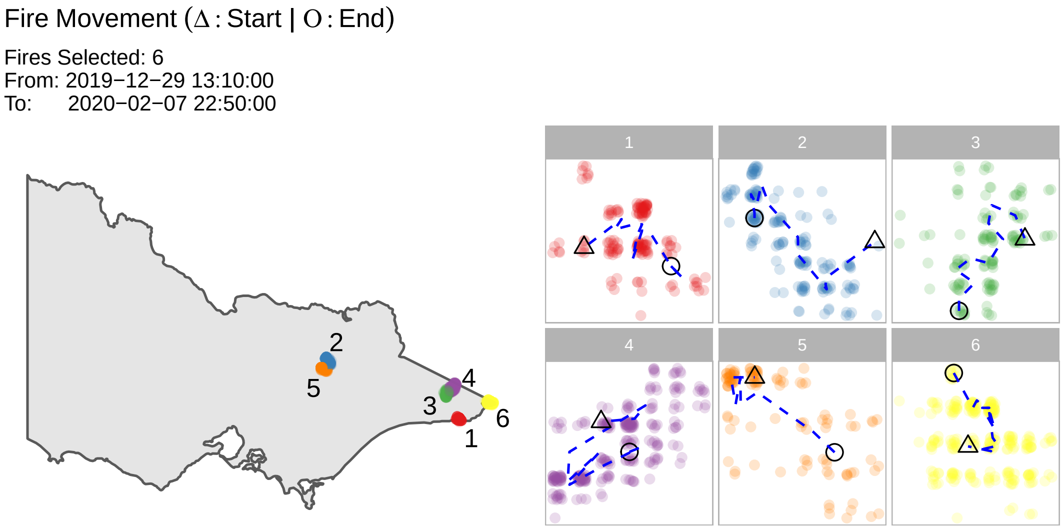
Figure 8: Automatic plot of results

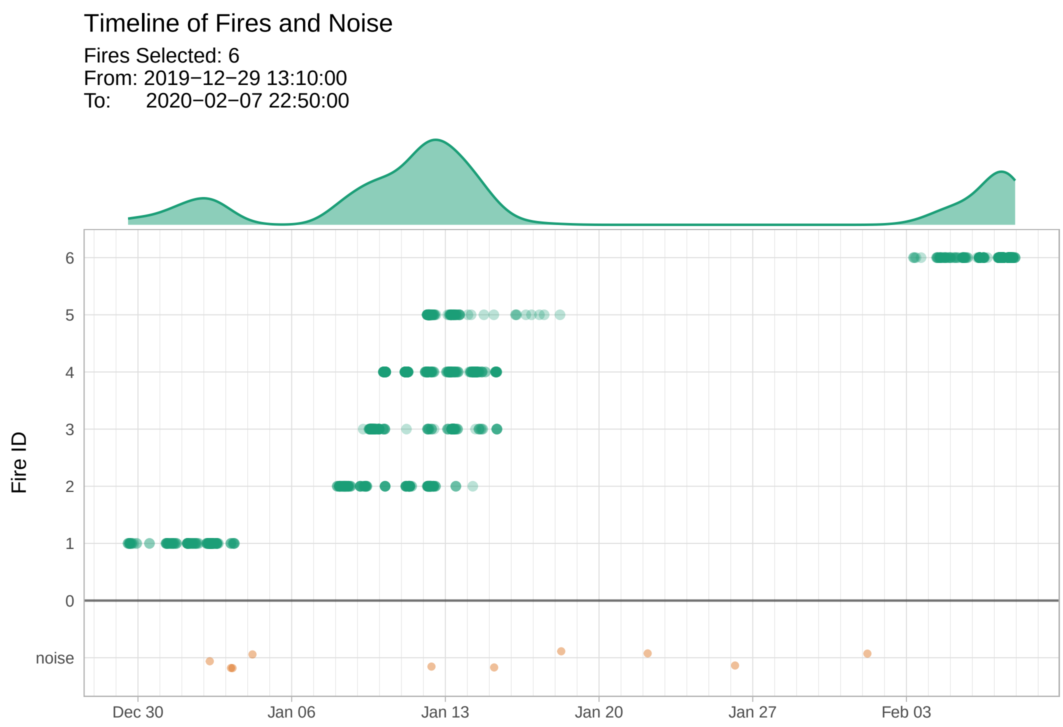
```
#> i 0.003 secs per hotspot
```

```
#>
```

```
#> --------------------------------------------------------------------------------
```

For this sample of data, the result contains ‘r length(unique(result$))





Functions

Summary

Acknowledgements

- The code and files to reproduce this work are at XXX
- Data on hotspots can be downloaded from XXX

Bibliography

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]

P-Tree System. JAXA Himawari Monitor - User's Guide, 2020. URL `https://www.eorc.jaxa.jp/ptree/userguide.html`. [p1]

G. Williamson. Example code to generate animation frames of Himawari-8 hotspots, 2020. URL `https://gist.github.com/ozjimbob/80254988922140fec4c06e3a43d069a6`. [p1]

Weihao Li
Monash University
Econometrics and Business Statistics

`wlii0039@student.monash.edu`

Emily Dodwell
??
line 1
line 2

`emdodwell@gmail.com`

Dianne Cook
Monash University
Econometrics and Business Statistics

dicook@monash.edu