Response to Reviews: RJ 2021-196

Weihao Li, Emily Dodwell, Dianne Cook

2022-11-06

Thank you for the careful reviews. What follows is our point-by-point response to reviewer comments.

Note that *The original reviewer comments are in italic* and our response is normal text.

Reviewer 2

Overview. [Is the article important to R the community? Is the approach sensible? Are technology and methods up-to-date?]

The article proposes a clustering algorithm developed for detecting and tracking movements of fires through the analysis of hotspots (namely single geographical points that are expected to be part of a fire, potentially of large extension).

The topic seems to be of interest for the R community, and the code might even be applicable to other types of diffusion processes, maybe with some minor adjustments.

The paper shows the implemented code at work on examples and on a full-scale experiment, demonstrating all the functionalities and the type of insights that the tool can help to achieve.

Thank you for this overview, it is an accurate description.

I have a number of comments and criticisms about the proposed algorithm:

1. While the authors present it as an adaptation of DBSCAN, namely a standard density-based method, its density-based component is completely missing. Indeed, every hotspot is clustered, regardless of the density around it, potentially forming clusters that are very thin. The minimun size check performed at the end does not really account for density. The proposed solution is a simple connectivity-based clustering, such as that implemented in single-linkage hierarchical clustering. Notice that DBSCAN itself works like that, yet it is applied only to the core points. While I don't think this should kill the proposed solution, the authors are invited to clarify these relations, possibly explaining why the border/noise point filtering is not applied here, or maybe clarify the text if that was a misunderstanding and the code indeed applies density filters.

Our algorithm is inspired by DBSCAN, but the density-based component does not work for our purposes. We have expanded the Background section to explain why this is, and changed the text in the Introduction section to clarify the relationship between our algorithm and DBSCAN.

2. It seems to me that the novelty of the approach is mainly concentrated in Section 3, namely how cluster memberships are propagated through time. The proposed solution makes sense to me, yet many other solutions are possible (e.g. two clusters that merge in later time steps might be unified, whereas currently they remain separated, regardless of how close they get – the alternative approach described later in this review should yield something along this direction), and the paper does not provide much evidence about the superiority of one approach over the other. The authors are invited to provide some convincing discussion and/or empirical comparisons to demonstrate what issues might come from simpler solutions.

We have expanded the Background section to discuss implications and limitations of alternative approaches, especially of solutions in which two clusters merge together into a single cluster, in the context of bushfire identification and monitoring.

3. The visualization tools seem to be slightly limited, assuming that results can be effectively summarized in 2D plots of centroids. Yet, (i) the spatial extension of a cluster can change in time and the same spot could exit and re-enter the cluster later, which is completely lost with 2D plots; (ii) a main reason for using DBSCAN/connectivity clustering was that clusters can have arbitrary shapes, which also means centroids might be not representative at all of the whole cluster, e.g. the cluster can change and the centroid remain still, as in one of the examples shown in the paper. The authors are invited to discuss these possible limitations. My guess is that, in practice, the two issues I pointed out are not relevant, because clusters are usually pretty simple.

We have expanded the visualization section to briefly discuss the limitation. Plots provided in this paper are intended to be basic since people can generate advanced graphics themselves using the results provided by the algorithm.

4. The visualization tool proposed for selecting parameters is very very similar to that traditionally adopted to find the best distance (radius) parameter for DBSCAN. Basically, plot the distance of each point from its k-th nearest neighbor (k is fixed by the user), sorting the values in ascending order, and find the knee in the curve. The authors might want to add reference to that.

Thanks for the reference. We have included it in section on selecting parameter values.

Article. [Is anything unclear? Is anything missing? Are the examples helpful? Is there sufficient background, including alternative approaches in R?]

Overall, the presentation and editorial quality of the paper looks excellent to me. In terms of related works, the paper might be improved a little bit:

1. Contrary to what claimed in the paper, traditional hierarchical methods typically produce intermediate results (dendograms) specifically used to select the number of clusters after the clustering, thus not requiring to specify it beforehand.

We have removed the corresponding sentence from the text.

2. The proposed approach has many similarities with Moving Clusters [Kalnis, Bakiras, Mamoulis 2005: https://link.springer.com/chapter/10.1007/11535331_21]. They apply DBSCAN on time snapshots of the data, and then link clusters across snapshots, though in a much stricter way. It seems to me a natural reference to consider.

Thanks for the reference. We have added it to the citation, and included an explanation in the Background section.

The task tackled in the paper can be in principle faced using simple ST clustering methods, for instance just converting time of each hotspot into a spatial coordinate (such that 24h ~ 3000m) and applying DBScan. I suggest to add a small comparative experimentation showing that the propose algorithm is more effective – and maybe simpler to use – thus justifying the introduction of another clustering algorithm.

The Background section contains a more complete comparison of current methods with particular focus on DBSCAN. What you suggest above of including time as a spatial neighbor is a nice idea. However, time needs to be considered in the forward direction only, which means this wouldn't work in the bushfire application. We could implement DBSCAN in temporal slices and would use the procedure of our algorithm to merge results from time slices. We have decided not to implement this because the current algorithm is effective and sufficiently fast for the bushfire application.

Reviewer 6

Overview: This manuscript, "A Clustering Algorithm to Organize Satellite Hotspot Data for the Purpose of Tracking Bushfires Remotely," describes a new method to identify brushfire-specific clusters in satellite data, which have specific temporal properties (e.g., fires move forward in time and may be burning for some time without emitting a spectral signature). Overall, I think the paper is very well-written, and the problem, purpose, and design of the algorithm are all very clearly described. While conceived for a specific application, I

think the method is a useful addition to the R community, and could potentially be applied in other substantive domains (especially given the rapidly increasing availability of high temporal resolution satellite imagery).

Thanks for this overview, and we agree.

Article: A few specific questions on the manuscript.

1. For the example in Sections 1-2 and Figure 1: how is an individual 'hotspot' defined, and what is the spatial unit of a hotspot (i.e., a grid cell or a point)? The paper mentions "Himawari-8 hotspot data," so I'm assuming the hotspots are a pre-processed product of the Himawari-8 sensor, and thus the inputs to the algorithm are not raw satellite imagery. But Figure 1 seems to suggest that hotspots are (vector?) points rather than particular grid cells. This relates to another question I have about the adjDist parameter: if the inputs are regular grid cells of raster data, why use a geodesic distance rather than a contiguous neighbour relationship? But it appears that the hotspots themselves are not gridded. In any case, this should be cleared up a bit (for someone not familiar with this specific data). Can any satellite imagery be used with this algorithm?

We have changed the text to alleviate this confusion. The Himawari-8 satellite data should be considered to be point data. However, like all satellite data, it is recorded on a regular grid. It should not be considered as gridded, but it does have a collection resolution. The reference to gridded data is removed from the text.

Any satellite image data could be used if they are processed into point data. The choice of *adjDist* will be calculated similarly to the example given in the revised text.

2. For Figure 2: in the panel "St before updating," why does it appear that the "2" cluster includes the point from cluster "4" in the previous panel? This relates to my previous question, but how are the spatial boundaries of the cluster determined - is it a convex hull of member points or just the grouping of member grid cells?

We have removed the polygons from Figure 2 and changed the text to alleviate this confusion. According to the revised figure, the reason a point from cluster "4" is connected to other points from cluster "2" in the panel "Step 3: (a)" is S_t has 13 more points than S_{t-1}, including those red points which connect the point from cluster "4" to points from cluster "2". This makes them belong to a cluster at S_t.

The spatial boundaries of a cluster is the union of all circles around the member points with radius `adjDist`. The convex hull was previously used to indicate the membership labels at S_t and has been removed from Figure 2 to improve the illustration of the algorithm.

3. I'm a little unclear on how the temporal dimension works - I understand St is the set of time intervals in the format [max(t) - activeTime, t], but what spatial patterns do each of those intervals represent? I understand the sequence of moving through patterns from image data from hour 1 to hour 2 to hour T, etc., but what does an image at [1, 25] look like? What's the actual temporal sequence of spatial patterns we're moving through?

We have clarified our language in the Algorithm section regarding use of "time interval" (i.e. one hour) and "time window" (i.e. a series of consecutive time intervals), and added further explanation of each step to build the reader's intuition and address the answer to this question.

4. In terms of the the adjDist parameter: is the spatial scale of a given brushfire cluster relatively stable over different time sequences/locations, or is there a reason to think some kind of adaptable bandwidth kernel might be beneficial?

We have added a small discussion in the Conclusion section to address the possible modifications of our proposed algorithm. It would be great to think about an adaptable bandwidth but it complicates the algorithm enormously. The *adjDist* parameter describes the maximum distance a fire can spread in one unit of time given all the connection points to the existing fire are undetectable. A larger *adjDist* will give a higher tolerance of missing connection points. It might be possible to calculate the fire spread rate, but it is relatively difficult to know the probability of a hotspot being undetected at different times.

5. The only thing that I think is missing from the paper is a comparison to other methods for brush fire detection: how does spotoroo compare to ST-DBSCAN and FSR in terms of computational efficiency and

results?

As noted in our reply to reviewer 2, we have expanded the Background section to discuss implications and limitations of alternative approaches in the context of bushfire monitoring. We have also noted in the Conclusion that packages to implement ST-DBSCAN and FSR are not available in R; therefore we cannot complete a comparison of computational efficiency and results.

Package: Software development is not my area of expertise (as my own R code will readily demonstrate), so I will leave comments on technical details and code style to other reviewers. However, in general the package seems to be developed with relatively few dependencies in a straightforward and flexible way: the primary 'hotspot_cluster()' function takes as its main inputs only 'lat', 'lon', and 'observed time' and clearly provides additional parameters that can be adjusted (but have useful defaults). Accessing the results in a "spotoroo" object, while usefully giving the user access to all of the outputs, is a bit clunky from the standpoint of retrieving the results, but the package documentation/paper nicely describes ways to quickly map and plot the results with in-built functions, so that helps.

The function `extract_fire` is responsible for transforming the `spotoroo` object to a data frame, which is explained in the Package section.

The package also has a GitHub page, which is useful form the standpoint of tracking development and bugs.

Thanks!