University of Potsdam

MSc. Data Science

MSc. Remote Sensing, Geo-Information and Visualization

Final Report

Big Data Analytics

A Study of Lightning with Focus on the Elevation

July 2020

Jennika Hammar (763788), Anastasia Karkatzinou (805685) & Nikolaos Kordalis (799802)

Supervisors: Prof. Dr. Bodo Bookhagen & Dr. Taylor Smith

Table of Contents

List of Figures

List of Tables

1 Abstract

In the present work we studied the behaviour of lightning, with emphasis in the relationship with the elevation. The data consisted of 20 years of observations (1995 - 2015) by the Lightning Imaging Sensor and elevations from the 90 m resolution Shuttle Radar Topography Mission for tropics and subtropics. We used K-means clustering to find different types of groups based on the FRD density and corresponding elevation. Then we performed hierarchical clustering to investigate the similarities in FRD density in a time series of monthly means over 20 years. Furthermore, we performed Principal Component Analysis, located and explored the top ten hot spots. In agreement with previous works, our analysis indicates a positive correlation between FRD and elevations at some areas up to a certain height. Moreover, we have indications that there are areas with high FRDs which are not correlated to elevation but rather to large scale atmospheric processes happening in these areas.

2 Introduction

Lightning is an atmospheric phenomenon, caused by the electrical discharge between charged regions of thunderstorms. Smaller ice particles (which are believed to be positively charged) within a cloud interact with larger negatively charged ones, causing fracture. Gravity and thunderstorm updrafts lead to further separation, up until the upper part of the cloud becomes positively charged, whereas the lower part acquires a negative charge, rising an electric potential (Bugbee et al. 2018). Eventually, the electric resistance between charged regions breaks down creating lightning, which can occur among cloud-cloud, cloud-ground or cloud-air.

Although lightning is a very impressive phenomenon, it can have devastating consequences as it can cause wildfires, agricultural and property damage and even human deaths. Lightning might increase in the future as a result of climate change (Williams 1992). The greatest Flash Rate Densities (further referred to as FRD) can be observed in coastal areas, mountainous areas, the Intertropical Convergence Zone (ITCZ), as well as in the convergence zones in the South Atlantic and South Pacific (Christian 2003). However, the lightning hotspot of the world is located over Lake Maracaibo in Venezuela (Albrecht et al. 2016). Lightning appears more during the warmer months. Studies have been made showing the interannual variability in FRD and an anticorrelation in FRD time series between the Northern Hemisphere (NS) and Southern Hemisphere (SH) (Ávila et al. 2010). They also found that the lightning activity in continental convective storms is much stronger than those in marine deep convective storms. Orographic uplift, which is the process by which warm air mass is forced upward from low to high elevations, is known to enhance the formation of clouds and consequently thunderstorms.

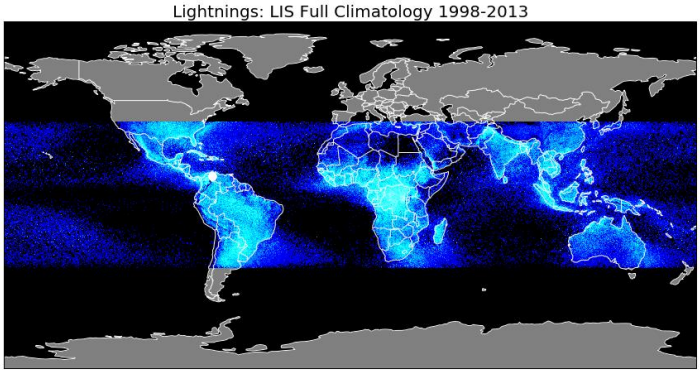


Figure 1: Average FRD densities as detected by the LIS TRMM for the years 1998-2013. The white spot denotes Lake Maracaibo in Venezuela, the place with the highest FRD (233 fl km-2 y-1).

Several studies have been investigating how FRD relates to elevation and other terrain parameters in different regions of the world. Bourscheidt et al. (2008) found a linear relationship between slope and cloud-to-ground FRD in South Brazil. Wagtendonk and Cayan (2008) showed a positive correlation between elevation and number of strikes in California. Kotroni and Lagouvardos (2008) did an analysis over the Mediteranian and could see a positive relationship of lightning activity with elevation in spring and summer. However, this feature was not evident during the rest of the year. Few studies have been made on a global scale. Christian et al. (2003) indicated a link between thunderstorms and topographic features based on the visual interpretation of the annualized distribution of total lightning activity and a digital elevation model (DEM).

The objective of this work is to study the behaviour of lightning and to investigate the relationship between lightning and elevation/slope on a global scale. Our hypothesis is that there are lightning events that are driven by elevation and other events that are driven by unstable atmospheric processes. The spatial coverage of the Lightning Imaging Sensor (LIS) is limited to the tropics and subtropics (Figure 1) hence our studies as well.

3 Data & Pre-processing

3.1 Lightning: Data acquisition & Pre-processing

For this study we used gridded data from the LIS which was operating onboard the Tropical Rainfall Measuring Mission (TRMM) and has a spatial coverage of 38 north latitude and 38 south latitude (Cecil 2015, Albrecht et al. 2016). The sensors detect cloud-to-ground (CG) and intra-cloud (IC) lightning flashes during both day and night (Cecil 2015). We used the Very High Resolution Climatology Data Collection and Low Resolution Monthly Time Series (LRMTS). The data is freely available and can be downloaded in NetCDF format from https://ghrc.nsstc.nasa.gov/. A detailed data description about the gridded climatology datasets can be found in Buechler et al 2015. The original data file also contains data collected by the prototype of LIS, the Optical Transient Detector (OTD) on Orbview-1. The OTD was active between 1995 and 2000 and had a greater spatial coverage than the LIS. Table 1 gives a more detailed inspection of the datasets.

Table 1: FRD datasets

Data	*Units*	*Product dimensions*	*Bin sizes*	*Spatial Coverage*	*From - to (year)*	*Smoothing*
LRMTS	fl km^{-2} day^{-1}	240x144x72	1 month x 2.5° x 2.5°	N: 88.75°, S: -88.75°, E: 180°, W: -180°	1995 - 2015	weighted average
VHRFC	fl km^{-2} year^{-1}	760 x 3600	0.1° x 0.1°	N: 38, S: -38, W: -180, E: 180	1998 - 2013	none
VHRMC	fl km^{-2} month^{-1}	760 x 3600 x 12	0.1° x 0.1° x 1 month	N: 38, S: -38, W: -180, E: 180	1998 - 2013	49-day & 0.1° x 0.1° boxcar moving average
VHRSC	fl km^{-2} trim^{-1}	760 x 3600 x 4	0.1° x 0.1° x 1 trimester	N: 38, S: -38, W: -180, E: 180	1998 - 2013	49-day & 0.1° x 0.1° boxcar moving average
VHRDC	fl km^{-2} hr^{-1}	760 x 3600 x 24	0.1° x 0.1° x 1 hr	N: 38, S: -38, W: -180, E: 180	1998 - 2013	0.1° x 0.1° boxcar moving average

The Climatology files are already pre-processed by NASA. There are no missing values and no outliers. The VHRFC consists of one image with the mean annual FRDs, the VHRMC of 12 images with the mean monthly FRDs, the VHRSC of 4 images with the mean seasonal FRDs and the VHRDS of 24 images with the mean diurnal FRDs. The Full Climatology file was opened as a multi-index data frame and turned into a simple index geodata frame, for some parts of this work. This initial dataset contained more than 1.5 million data points, which were difficult to be processed. For this reason, we considered filtering the data and keeping the entries with FRDs greater than 5 fl km^{-2} year^{-1}, resulting in a geodata frame with 501002 entries. The rest of the Climatology files were opened and read as NetCDF data files.

The LRMTS contains 240 images with the monthly mean of FRDs. Since OTD has a larger spatial coverage, we sliced the LRMTS file and kept only the tropics and subtropics. Specifically, we created two functions that transform the gridded coordinate indices to real coordinates and backwards (see Appendix: Lightning Data Acquisition & Pre-process, functions gridded_latlon and real_latlon), and kept only the +21/+52 gridded latitude indices, that correspond to -38/+38 latitudes on Earth.

For some parts of this work we considered it essential to mask the data (see chapter 4.7). The criterion was to mask the coordinates that have constantly zero FRDs (or in the case of seasonal clustering, very low FRDs). The idea was to aggregate the FRDs along the axis of time, looking for coordinates that have zero sum and create a mask on top of these coordinates: since their sum is zero, their FRDs are always zero (for details see Appendix: Data Acquisition & Pre-processing, other climatology files, create mask).

3.2 Elevation: Data Acquisition & Pre-processing

For the topographic analysis we used the Shuttle Radar Topography Mission (SRTM) Digital Elevation Model (DEM) with a spatial resolution of 90m. The SRTM is a National Geospatial-Intelligence Agency (NGA) and a National Aeronautics and Space Administration (NASA) product with a spatial coverage of N: 60°, S: -56°, E: 180°, W: -180° (Javis et al. 2008). We extracted the SRTM DEM elevation value for each VHRFC pixel with a FRD greater than 5 fl km^{-2} $year^{-1}$ by linking Python to Google Engine. After a few attempts, we realized that for some coordinates there is no information for the elevation, which terminated the iteration loop. We bypassed these coordinates by adding a try-except-pass path in our loop (see Appendix: Elevation Data Acquisition & Pre-processing). The code ran for 3 days, bypassing about 53000 coordinates (10.7% of the data) which finally were excluded from the results. Moreover, for the location of the hotspots we downloaded and used the simple format of the populated cities dataset (Natural Earth).

4 Methods

The present work was performed with Python 3.7 and Google Engine linked to Python. Lake Maracaibo at (N: 9.7°, W: -71.7°,), as the top ranked hotspot with the highest FRD of 232.5 fl km^{-2} y^{-1}, was used as the reference spot throughout the whole work, and served as verification of the individual steps of our process.

4.1 Exploring the Data

In order to get a first impression of the data we decided to locate the top ten hotspots of FRDs, to explore it with Principal Components Analysis (PCA) and to create an interactive map with the DEM and FRD locations.

4.2 Hot Spots

In this method we used the Full, Monthly and Diurnal Climatology as well as the Simple Cities datasets. In order to start exploring the data and the FRD distribution, we started with the Full Climatology dataset. We created a hexbin plot using a threshold of 5 fl km^{-2} $year^{-1}$, in order to be able to distinguish some initial patterns. The hexbin plot divides the map into hexagons and adds up all the events that fall within the same hexagon. For further investigation, we located the hotspots with a spatial join between the FRD coordinates from the VHRFC dataset and the cities coordinates. In order to avoid overlapping spots, we buffered the initial hotspots, examined all possible combinations and discarded the overlapping polygons. This resulted in a list of hotspots that are at least 100 km apart, which finally provided us with the desired list of the top 10 hotspots. Furthermore, we performed a basic time series analysis for these 10 hotspots based on the VHRDC and VHRMC datasets.

4.3 Principal Components Analysis (PCA)

PCA is a powerful statistical method, which allows us to reduce the dimensionality of the data with Singular Value Decomposition. This way the interpretability is increased, since the less important variables are dropped and the most significant ones stand out, while the information loss is minimized. Regarding the PCA, we worked with the VHRMC dataset. We decided to exclude places that have constantly no flashes, so that they will not influence the process. For this purpose we created a mask, as described above. Afterwards, we normalized the data, arranged them in a dask array and performed a PCA with 3 components.

4.4 Interactive Map

We created an interactive map with two different backgrounds, one the SRTM DEM 90 m, and one base map from google map. We chose to display the places where the FRD is greater than 20 fl km^{-2} $year^{-1}$ in order to make the navigation faster and enhance the contrasts. The interactive map contributes to obtaining a globally accurate picture of places with high FRD. Moreover using this map we can navigate in various areas of interest around the world, zoom in and investigate whether and how a place is affected by flashes or not. The combination of the

DEM's background and the points of FRD enables us to see the correlation between elevation and FRD, enhancing our initial premise.

5 Cluster Analysis

5.1 K-means Clustering

Clustering aims to group observations in such a way that the observations in one group are more similar to each other than the other groups. To explore the relationship between FRD and elevation we performed clustering based on the K-means algorithm. It is an iterative calculation to minimize the within clusters sum of square errors (WSS) (Zhang et al. 2016). We used the VHRFC points together with the corresponding elevation with a total of 501000 data points. The K-means algorithm was selected because of the good ability to compute a large amount of data points in a reasonable amount of time (compare with McInnes et al. 2016).

The elevation and FRD have different units, hence we performed a standardization on the two datasets. To estimate the optimal amount of clusters, in order to run the K-means algorithm, we relied on the Elbow method (Thorndike 1953). It computes how the WSS decreases with an increasing number of clusters (Zhang et al. 2016), indicating the optimal number of clusters to be where the elbow forms (Figure 2).

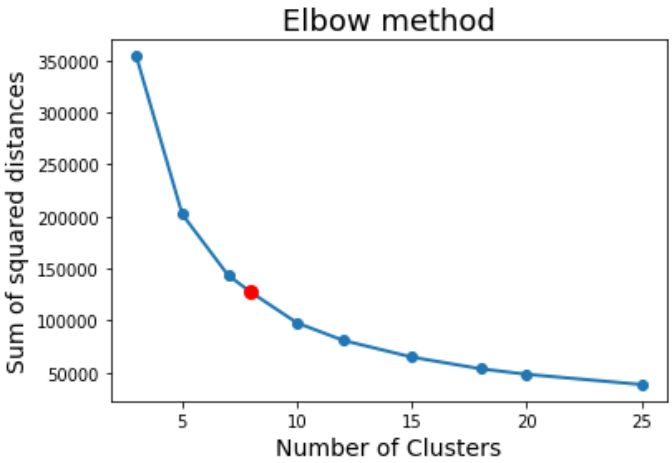


Figure 2: Elbow plot based on K-means clustering results of the VHRFC with FRD and elevation as parameters. The optimum number of clusters appears to be eight (red spot).

5.2 Hierarchical Clustering

We decided to run a hierarchical clustering with the complete method on the LRMTS. An implementation on the VHRMC and VHRSC was not possible due to the large amount of data points (the time complexity of hierarchical clustering is quadratic (McInnes et al. 2016)). We wanted to perform this clustering as well, because it enables us to see relationships between time series and also because it is possible to cluster at different levels of similarity. Hierarchical clustering of the agglomerative type is based on a bottom-up approach and a distance metric; first, each data point is assigned to separate clusters. Then, more clusters are formed by joining the closest clusters. The process ends when there is only one cluster left. With means of a dendrogram (Figure 3) the correlations between all data points from the clustering algorithm can be shown. The final number of clusters is determined by cutting the dendrogram at a distance. We selected the correlation as distance metric, meaning the time series will be assigned the same cluster if they show high correlation, with peaks and dips in FRD at the same time.

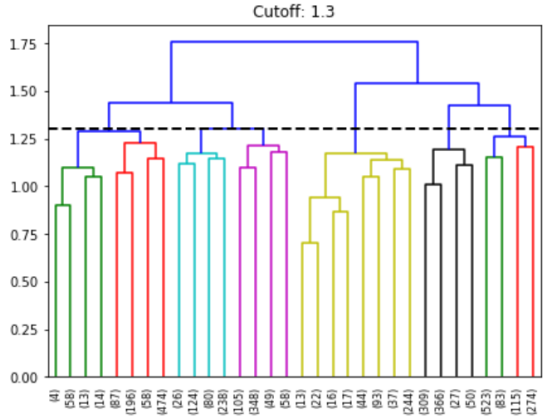


Figure 3: a dendrogram showing the result of hierarchical clustering based on a correlation distance metric, with an example cutoff at 1.5.

5.3 De-seasoning

We performed a de-seasoning of the data based on the previous seasonal clustering, and specifically for the cutoff that results in six clusters (1.3). In order to estimate the seasonal behaviour of the data and extract a trend, we used the statsmodels Python module. We continued with linear regression, in order to calculate the declination of the estimated trend through the time (decades).

6 Results & Discussion

6.1 Exploring the Data

The DEM overlaid with the distribution of the FRD is shown in a the interactive map (Figure 4). We distinguished visually three prominent areas in northern Venezuela, Guatemala, Cuba, central Africa, Malaysia and the southwestern Himalaya respectively. All these areas are represented with cyan colour and experience FRD greater than 150 fl km^{-2} $year^{-1}$. The background of the global DEM and the possibility of using the 'zoom in' function led us to the first conclusions in terms of the elevation and FRD. We noticed that on mountain foothills the FRD starts to increase while where the elevation is too high the FRD decreases.

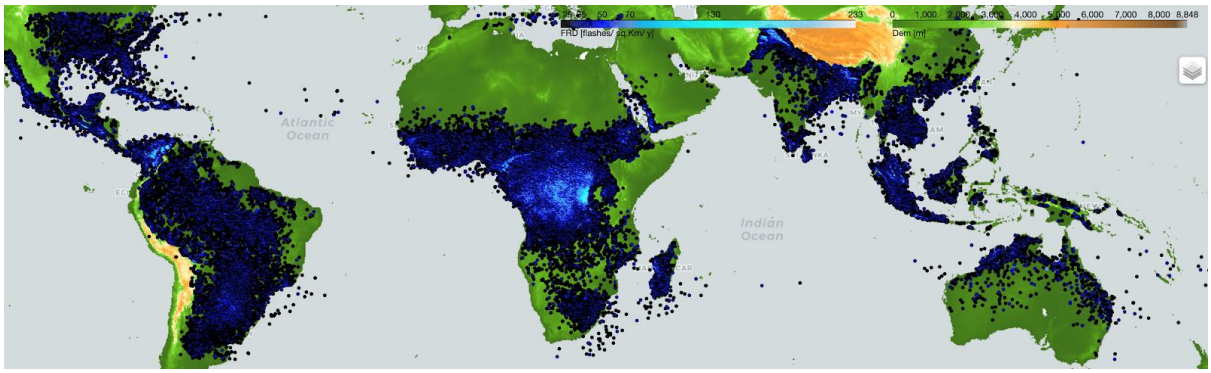


Figure 4: The global distribution of FRD > 20 with the DEM as basemap, as shown on the interactive map.

6.2 Hotspots - PCA

The focus of this first part is to explore the FRD distribution and to identify the places where the highest FRDs occur, i.e. the top ten hotspots. In this frame, an initial step is the creation of a hexbin plot, (Figure 5). A significant difference is depicted between land and oceans, in accordance with previous reports (Christian et al. 2003). The main density of FRDs is observed over land, whereas oceans exhibit reduced activity. The lower heat capacity of the ground leads to an important raise of the air temperature, resulting in intense convection over the land.

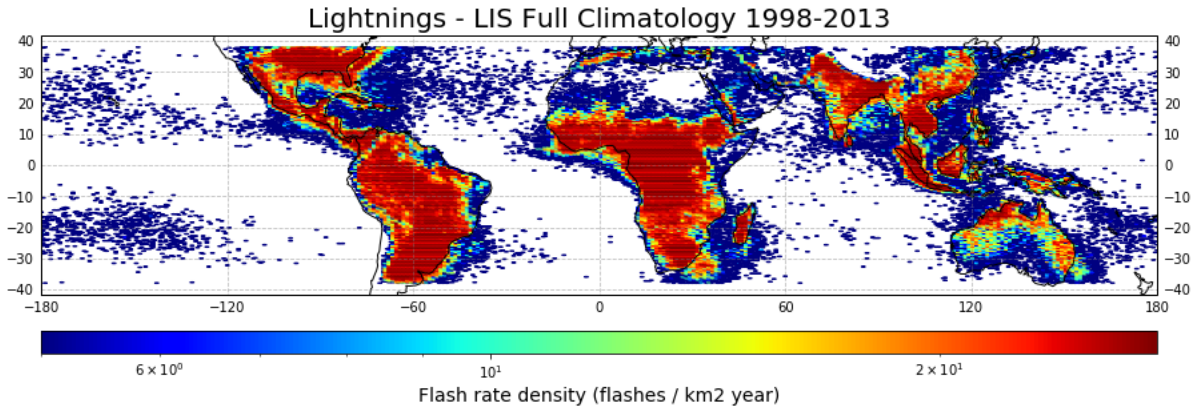


Figure 5: spatial coverage of the LIS TRMM dataset and the FRDs (with a threshold of 5 flashes) for the years 1998-2013.

Some patterns can also be observed around mountain ranges, like the southwestern Himalayas and Andes, which imply that the elevation also plays a specific role. These places, along with Sierra Madre mountains in Mexico, Amazon and Congo basin exhibit significantly high FRDs. Most of the hotspots are expected to be found in such complex terrain areas.

A further analysis of the concentration of the FRDs resulted in the top ten hotspots, as depicted in Figure 6. The top hotspot is found to be Lake Maracaibo in Venezuela with 233 fl km^{-2} $year^{-1}$ as expected, following the findings

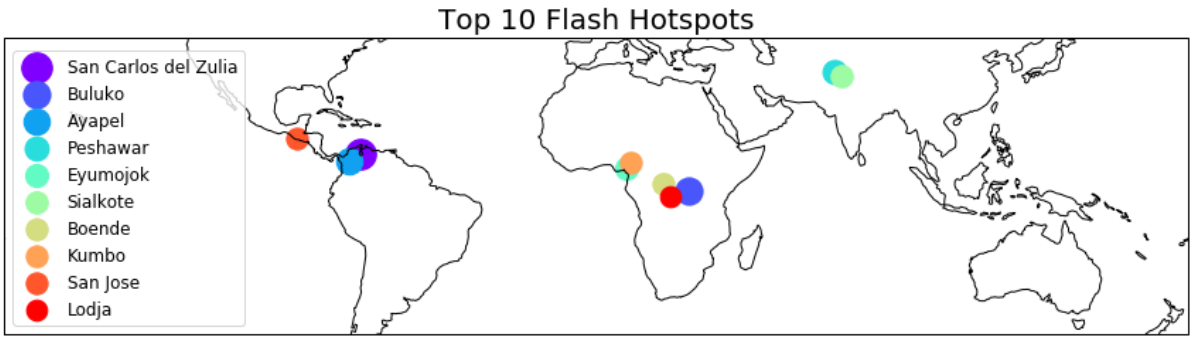


Figure 6: The ten top ranked FRD regions in the tropics and subtropics for the years 1998-2013. The top hotspot is found to be Lake Maracaibo as expected, here identified as the closest populated place of San Carlos del Zulia.

of Albrecht et al. (2016). Here though, due to our method of locating spots with 100 km minimum distance, it is identified as the closest populated place of San Carlos del Zulia. Lake Maracaibo could be described as the world capital of lightning, with lightning occurring 300 nights per year. Considering that the area of the lake is about 13,200 km^2, it appears that there are about 3,000,000 lightning strikes per year. The reason for this extremely high lightning activity is due to the geomorphology of the area, which favours the creation of strong storms in the late evening (Figure 8a) when the cold and dry wind coming from the Andes mountain range meets the warm and humid air above the lake, creating ideal conditions for deep convection (Albrecht et al. 2016). Although the lake itself has 0 elevation, the elevation due to the surrounding mountains plays a significant role. The seasonal maximum appears to reach a peak in September (Figure 7a).

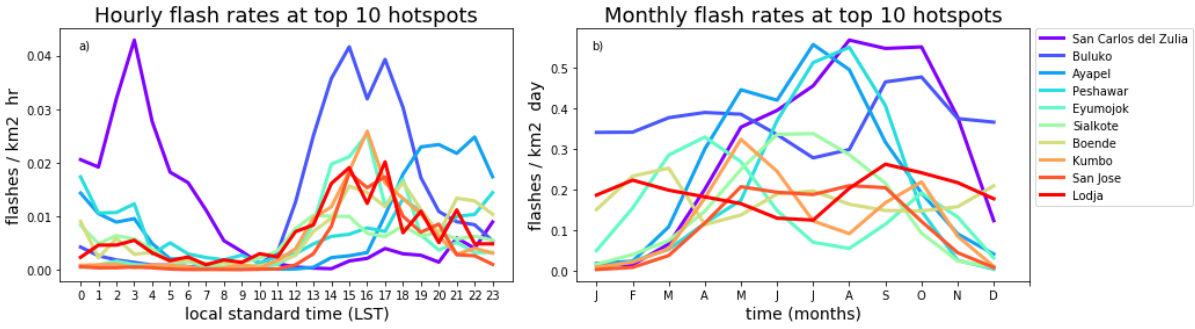


Figure 7: Diurnal (a) and monthly (b) time series for the top ten hotspots' FRDs

The second place is occupied by lake Victoria (close to the city of Buluko), the world's largest tropical lake, which exhibits similar characteristics and favours late evening deep convection. The rest hotspots are located where expected as well: central Africa (Congo and Cameroon), central America (Guatemala) and southwestern Himalaya (Pakistan), in accordance to previous studies (Albrecht et al 2016, Christian et al. 2003). They all exhibit higher flashes during late afternoon or evening. Moreover, the ones close to the equator show two peaks of high flash activity during the year, in spring and autumn, whereas the ones on the NH present broader curves, with flash maxima between summer and autumn (Figure 7a and 7b).

The PCA with three components, which describes 90% of the data, showed some interesting results. According to the first component (Figure 8) which corresponds to 63% of the data, the patterns we see represent the seasonal variance, in regard to the generation of FRD. At the very high elevations (Andes, plateau of Himalaya) and at the deserts (Sahara) the climate conditions do not experience any crucial change through the year and FRD remains constant (depicted with green colour). On the other hand the yellow and the blue colour denote significant seasonal changes in terms of the FRD activity, with the corresponding areas exhibiting inversely proportional behaviour. For instance, southwest and southeast Himalayas, the high FRD exhibits strong seasonal behaviour: in the first case it is driven by elevation (in the foothills of the mountain), whereas in the second case the FRD activity is influenced by the monsoon season. This indicates, that non only elevation but also other factors play a significant role in the formation of FRD.

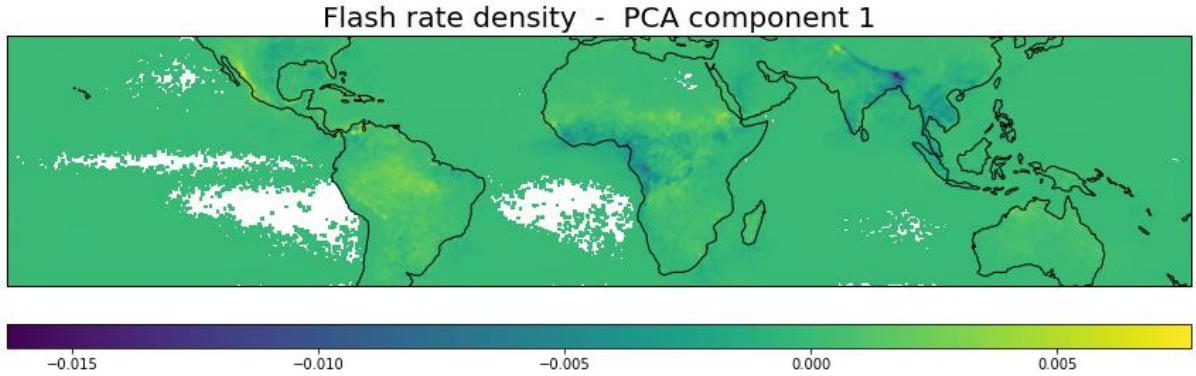


Figure 8: Plot for the first PCA component of the FRDs (masked Full Climatology data).

6.3 Cluster Analysis

6.3.1 K-means Clustering

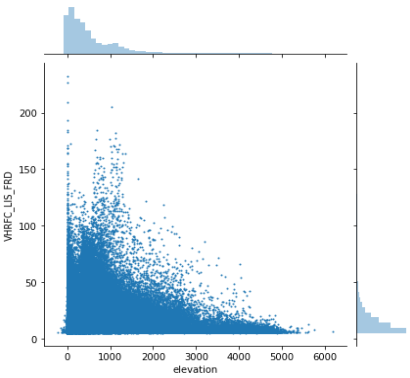


Figure 9: Jointplot for elevation and FRD

We displayed the relationship between the elevation and the FRD as well as 1D profiles (univariate) (Figure 9). In that way we matched up two 'distplots' for bivariate data which in our case comprises elevation and associated FRD. Creating the jointplot we observed that there is an upward trend of FRD up to about 1500 meters. Moreover the high values of FRD around the zero elevation suggest that other climatic and geomorphological factors play an important role in the generation of flashes.

Figure 10 illustrates the result of the K-means clustering, for a variety in the number of clusters. We did this to see which clusters remain stable even if the number of clusters increases. Beginning from a cluster number of four, the west bank of the Andes and parts of Himalaya gets assigned an independent cluster and remains constant, although they are geographically separated. This cluster corresponds to the highest elevations as well as the second smallest FRD. At the foothills of Himalaya there are four long and narrow different clusters appearing next to each other (cluster 1 > 3 > 4 >2) the FRD first increases but drops again after a mean elevation of 1885 m. This finding indicates a relationship between elevation and FRD but just to elevations below ca 1885 m.

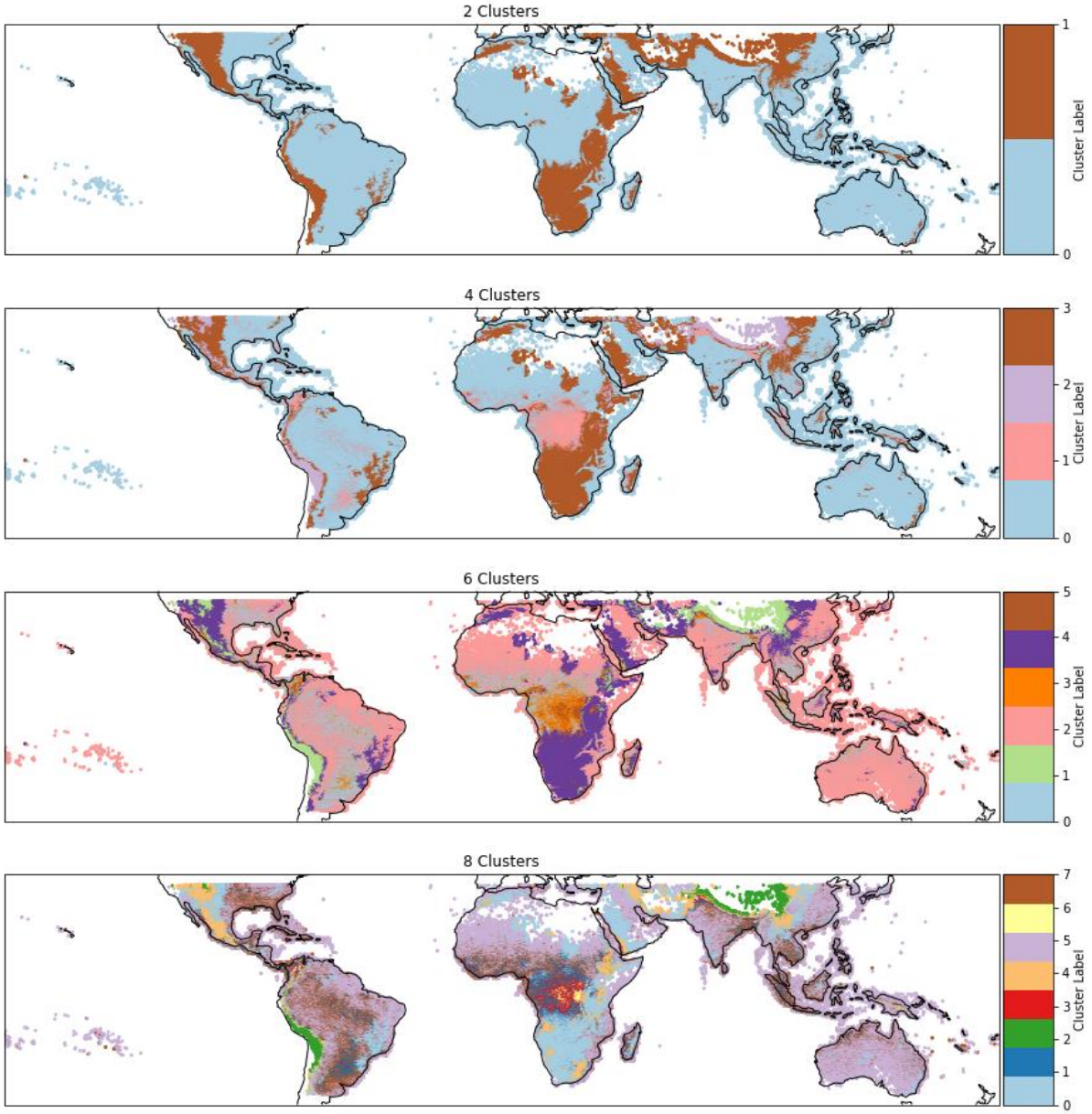


Figure 10: Clusters determined by the K-means algorithm, in respect to elevation and FRD.

At eight clusters (which was defined as the optimum) one prominent cluster appears in central Africa (cluster 6), this can be traced back to Lake Victoria which is known to have high FRDs. When we compare the descriptive statistics of the eight clusters (Table 2) we can see that this cluster has the highest mean FRD (80) of all clusters. Generally, clusters with high FRDs (cluster 3 and 6) correspond to elevations which are around 500 m. They are in central Africa around Lake Victoria and one at the southwestern side of the Himalaya.

There are some areas which show an heterogeneous pattern with many small different clusters displayed, for example in central upper Africa, where the K-Means algorithm fails to produce an homogenous pattern based on only FRD and elevation, which let us assume that there are other variables which influence the formation of lightning, such as large scale convection.

Table 2: descriptive statistics to the cluster number which was defined by the Elbow method (8)

Cluster	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*
Mean elevation (m)	1024	400	3800	504	1885	210	587	213
Mean FRD (fl km^{-2} year^{-1})	11	28	9	46	13	8	80	17

6.3.2 Time series Clustering

The time series plot for the FRDs of two places of choice, one in the NH and one in the SH (Figure 11). As we expected, the FRDs are totally anticorrelated, as they are highly seasonal and the maximum lightning activity is reached in opposite seasons in each hemisphere.

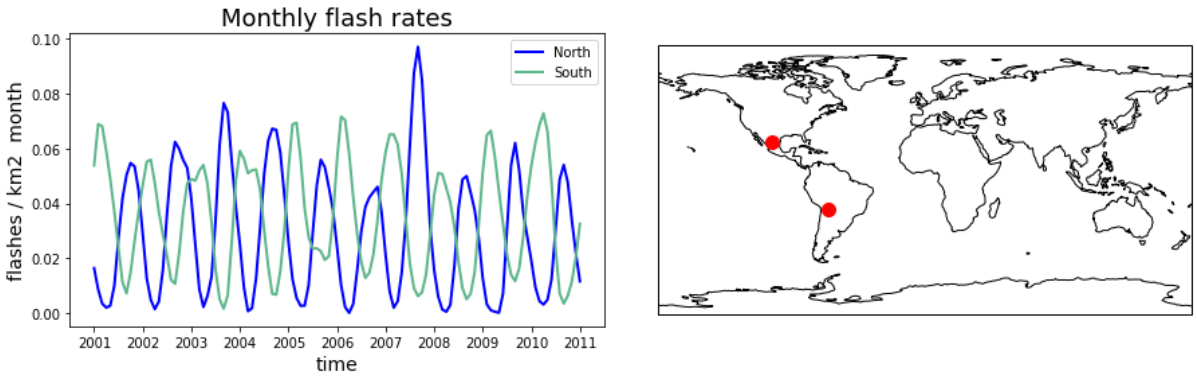


Figure 11: Time series for the monthly FRDs for two regions: one in the NH and one in the SH. The plot exhibits the total anticorrelation of these two regions, in regard to the seasonal behaviour of the FRDs.

Figure 12 presents our result for the hierarchical clustering at different cutoff distances. Surprisingly, the first stage of clustering, in which two clusters are formed, does not clearly divide the NH and SH. We expected to see a clear division because of the anticorrelation between the NH and SH in FRD time series shown in Ávila et al. (2010) and also based on our comparison between two time series in NH and SH (Figure 11). A reason why we do not see this clear division can be because we used the low resolution climatology (2.5°), another reason might be that some time series will be with many gaps (because of no FRD at this time step) and can be assigned to a cluster with a anticorrelated season. After trying several cutoffs, we concluded that the clustering with six clusters is the

most appropriate one. Moreover, with an increasing number of clusters we can see that some clusters remain relatively stable. For instance, Amazon and Lake Victoria, two very special tropical places with high FRD, are always clustered together (cluster 4).

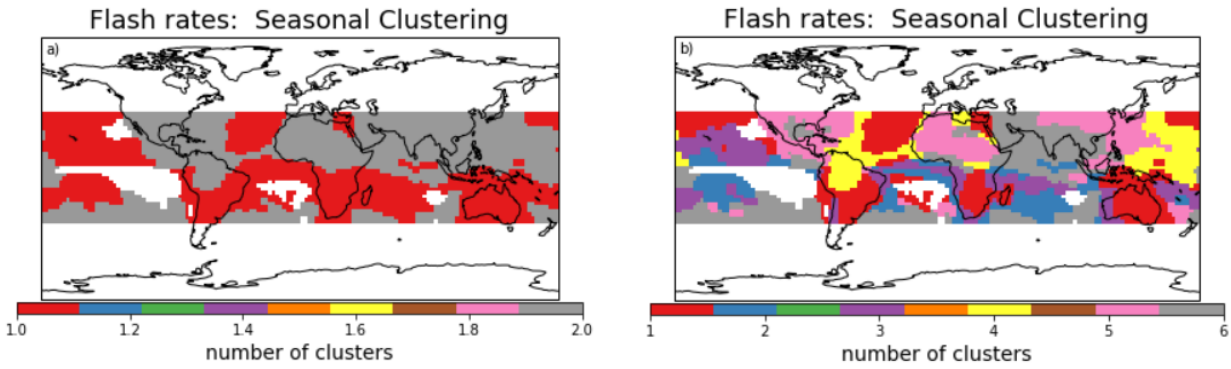


Figure 12: Seasonal clustering for two (a) and six (b) clusters.

The time series of these six clusters (Figure 12b) reveals not only the differences in the intensity of the FRD density, but it also shows clearly the anticorrelation between the NH and SH (for instance clusters 1-NH and 5-SH).

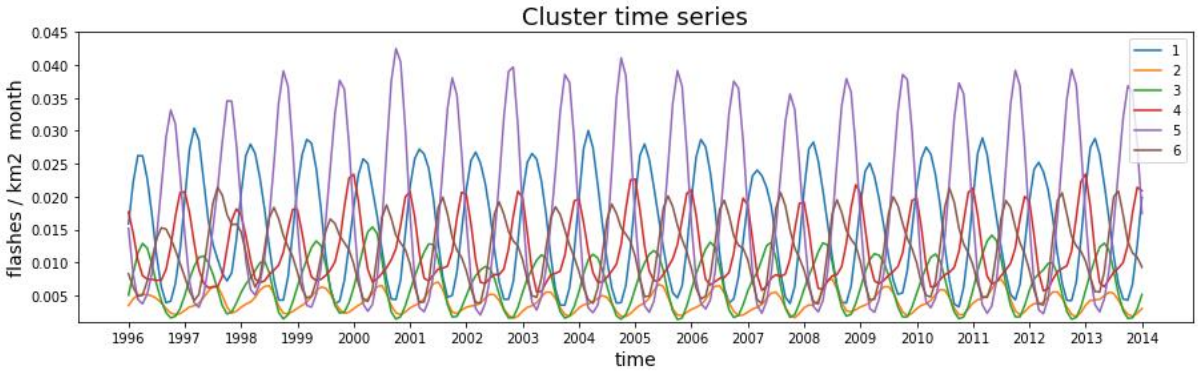


Figure 13: Time series for six clusters. The difference in intensity of the FRDs as well as the anticorrelation between NH (5) and SH (1) are clearly depicted.

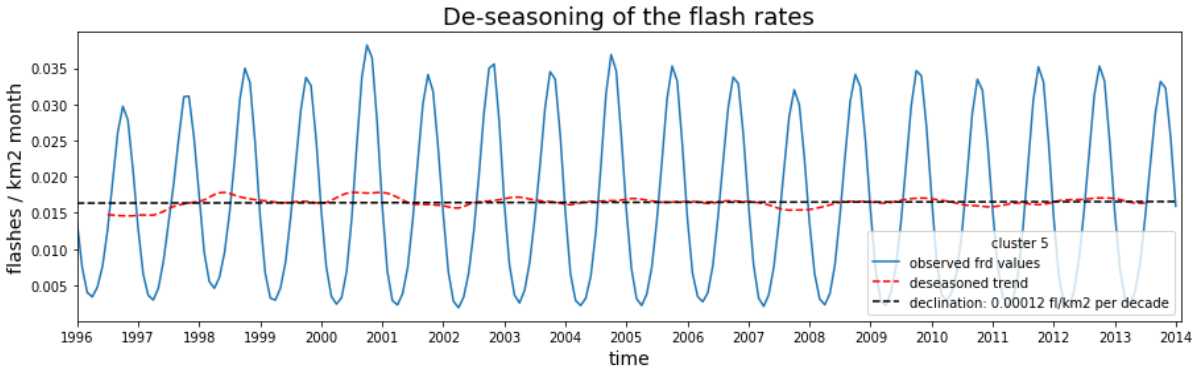


Figure 14: The de-seasoned trend over the observed FRD values. The black dashed line depicts the linear regression line of the trend.

A further attempt to de-season the results for cluster five is illustrated in Figure 14. The de-seasoned trend exhibits some fluctuations. The linear regression resulted in the estimation of a rather low declination of the trend of about +0.00012 fl/km^2 per decade. For lake Maracaibo, for instance, this means a raise of about 1500 flashes in a decade.

7 Limitations & Conclusions

The objective of our project was to investigate the relationship between lightning and elevation with means of cluster analysis. One of the main challenges in the project was to find a way to work and extract information from an almost global DEM and combine it with data from other sources. Google engine enables this under the motto "bring the algorithm to the data" instead of downloading it. Though this required an implementation of Google Engine commands in python which was sometimes demanding.

Other atmospheric processes (like wind direction, ocean oscillation, cyclones etch.) influence the formation of lightning, which adds complexity to the interpretation of the problem. The process that we followed in our study comprised only one parameter (elevation). Natural phenomena are complex systems and should be approached by multivariate analysis.

Future work could focus on areas which have interesting features that became visible through the cluster analysis, the mountain strike of Himalaya where we have several different clusters appearing along each other could be investigated more in detail.

Although we denoted a relationship between elevation and FRD, after our study it is also clear that many other factors contribute to the generation of flashes. PCA and clustering indicate that more parameters play a significant role. For this reason and for a more thorough approach, other factors such as slope, vegetation, periodical atmospheric phenomena (monsoons etc.), local climate system, humidity etch., should be considered.

8 References

Albrecht, R., S. Goodman, D. Buechler, R. Blakeslee, and H. Christian. 2016. LIS 0.1 Degree Very High Resolution Gridded Lightning Climatology Data Collection. Datasets available online [https://ghrc.nsstc.nasa.gov/pub/lis/climatology/LIS/] from the NASA Global Hydrology Resource Center DAAC, Huntsville, Alabama, U.S.A. doi: http://dx.doi.org/10.5067/LIS/LIS/DATA306

Albrecht, R. I., Goodman, S. J., Buechler, D. E., Blakeslee, R. J., & Christian, H. J. (2016). Where are the lightning hotspots on earth? Bulletin of the American Meteorological Society, 97(11), 2051–2068. https://doi.org/10.1175/BAMS-D-14-00193.1

Ávila, E. E., Bürgesser, R. E., Castellano, N. E., Collier, A. B., Compagnucci, R. H., & Hughes, A. R. W. (2010). Correlations between deep convection and lightning activity on a global scale. Journal of Atmospheric and Solar-Terrestrial Physics, 72(14–15), 1114–1121. https://doi.org/10.1016/j.jastp.2010.07.019

Buechler, D., Cecil, D. J., Buechler, D. E., & Blakeslee, R. J. (2015). Gridded lightning climatology from TRMM-LIS and OTD : dataset description Gridded lightning climatology from TRMM-LIS and OTD : Dataset description. Atmospheric Research, 135–136(January), 404–414. https://doi.org/10.1016/j.atmosres.2012.06.028

Bugbee, K., Gatlin, P., Sinclair, L., Smith, D., Weigel, A. (2018). Lightning. Received from https://ghrc.nsstc.nasa.gov/home/micro-articles/lightning

Cecil, D. J., Buechler, D. E., & Blakeslee, R. J. (2014). Gridded lightning climatology from TRMM-LIS and OTD: Dataset description. Atmospheric Research, 135–136, 404–414. https://doi.org/10.1016/j.atmosres.2012.06.028

Collier, A. B., & Hughes, A. R. W. (2011). Lightning and the African ITCZ. Journal of Atmospheric and Solar-Terrestrial Physics, 73(16), 2392–2398. https://doi.org/10.1016/j.jastp.2011.08.010

Dissing, D., & Verbyla, D. L. (2003). Spatial patterns of lightning strikes in interior Alaska and their relations to elevation and vegetation. Canadian Journal of Forest Research, 33(5), 770–782. https://doi.org/10.1139/x02-214

Christian, H. J., Blakeslee, R. J., Boccippio, D. J., Boeck, W. L., Buechler, D. E., Driscoll, K. T., … Stewart, M. F. (2003). Global frequency and distribution of lightning as observed from space by the Optical Transient Detector. Journal of Geophysical Research D: Atmospheres, 108(1). https://doi.org/10.1029/2002jd002347

Jarvis, A., H.I. Reuter, A. Nelson, E. Guevara. 2008. Hole-filled SRTM for the globe Version 4, available from the CGIAR-CSI SRTM 90m Database: http://srtm.csi.cgiar.org.

McInnes, L., Healy, J., Astels, S. (2016). Benchmarking Performance and Scaling of Python Clustering Algorithms. Received from https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html on the 08.07.2020

Natural Earth. Populated Places. Received from https://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-populated-places/ on the 12.07.2020

van Wagtendonk, J. W., & Cayan, D. R. (2008). Temporal and Spatial Distribution of Lightning Strikes in California in Relation to Large-Scale Weather Patterns. Fire Ecology, 4(1), 34–56. https://doi.org/10.4996/fireecology.0401034

Williams, E. R. (1992). The schumann resonance: A global tropical thermometer. Science, 256(5060), 1184–1187. https://doi.org/10.1126/science.256.5060.1184

Thorndike, R. (1953). Who belongs in the family? Psychometrika, 18, 267-276.

Zhang, Y., Moges, S., & Block, P. (2016). Optimal cluster analysis for objective regionalization of seasonal precipitation in regions of high spatial-temporal variability: Application to Western Ethiopia. Journal of Climate, 29(10), 3697–3717. https://doi.org/10.1175/JCLI-D-15-0582.1

Data Acquisition & Preprocessing for Lightning datasets (netCDF files)

In [1]:
```
from netCDF4 import Dataset
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
crs = {'init': 'epsg:4326'}
```

Tranformation between original and gridded coordinates

In [2]:
```
#create two functions to tranfrorm the coords back and forth
    res = 0.1   #climatology = 0.5,  LRMTS = 2.5
zone = 38   #climatology = 38,   LRMTS = 90

def gridded_latlon(real_lat, real_lon):
    grid_lat = round((real_lat + zone)/res)
    grid_lon = round((real_lon + 180)/res)
    print(grid_lat, grid_lon)
    return grid_lat, grid_lon


def real_latlon(grid_lat, grid_lon):
    real_lat = grid_lat*res - zone
    real_lon = grid_lon*res - 180
    print(real_lat, real_lon)
    return real_lat, real_lon
```

In [3]:
```
#initial file path
file_path = './LIS_VHRES/data_climatology/'
```

Full Climatology - reference spot and data in geodataframe

In [4]:
```
frd = Dataset(file_path + 'VHRFC.nc', 'r').variables['VHRFC_LIS_FRD'][:].data
```

Reference spot

In [5]:
```
#find the place with the highest flash rate density - keep it in mind as reference spot

maximum  = int(round(np.amax(frd)))
lat = np.where(frd == np.amax(frd))[0][0]
lon = np.where(frd == np.amax(frd))[1][0]

real_lat, real_lon = real_latlon(lat, lon)

print('The place with the highest flash rate density is at lat={}, lon={},\n with a mean of
{} flashes per year'.format(round(real_lat), round(real_lon), maximum))
```

```
9.700000000000003 -71.69999999999999
The place with the highest flash rate density is at lat=10.0, lon=-72.0,
 with a mean of 233 flashes per year
```

NC file to dataframe

In [6]:
```
#open multi-index nc file into dataframe
import xarray as xr
import netCDF4

df = xr.open_dataset(file_path + 'VHRFC.nc').to_dataframe()
df = df.reset_index(level=['Longitude', 'Latitude'])
df = df[df.VHRFC_LIS_FRD > 0]
del df['VHRFC_LIS_VT']
df.reset_index(inplace=True, drop=True)
df
```

Out[6]:

	Latitude	Longitude	VHRFC_LIS_FRD
0	-37.95	-178.75	0.906318
1	-37.95	-177.35	0.903004
2	-37.95	-177.25	0.902887
3	-37.95	-176.95	2.707332
4	-37.95	-176.45	0.940296
...	...	...	...
1515605	37.95	174.15	1.485078
1515606	37.95	174.75	1.173280
1515607	37.95	174.95	1.173879
1515608	37.95	175.05	1.486169
1515609	37.95	175.75	2.347237

1515610 rows × 3 columns

In [7]:
```
#save it
df.to_csv('Full_Climatology.csv')
del df
```

In [8]:
```
#open it
df = pd.read_csv('Full_Climatology.csv')
del df['Unnamed: 0']
```

In [9]:
```
#filter it with a threshold of 5 flashes
df = df[df.VHRFC_LIS_FRD > 5]
gdf = gpd.GeoDataFrame(df, crs=crs, geometry = gpd.points_from_xy(df.Longitude, df.Latitude
))
gdf.reset_index(inplace=True, drop=True)
del df
gdf
```

Out[9]:

	Latitude	Longitude	VHRFC_LIS_FRD	geometry
0	-37.95	-143.25	11.821247	POINT (-143.25000 -37.95000)
1	-37.95	-70.25	10.962107	POINT (-70.25000 -37.95000)
2	-37.95	-64.85	5.436871	POINT (-64.85000 -37.95000)
3	-37.95	-64.65	5.594469	POINT (-64.65000 -37.95000)
4	-37.95	-63.65	6.793575	POINT (-63.65000 -37.95000)
...	...	...	...	...
500997	37.95	131.75	5.861458	POINT (131.75000 37.95000)
500998	37.95	136.25	5.857105	POINT (136.25000 37.95000)
500999	37.95	139.35	7.016274	POINT (139.35000 37.95000)
501000	37.95	154.25	5.855022	POINT (154.25000 37.95000)
501001	37.95	154.35	7.025729	POINT (154.35000 37.95000)

501002 rows × 4 columns

The other climatology files - no dataframe

monthly - VHRMC.nc, annual - VHRAC.nc, diurnal - VHRDC.nc, seasonal- VHRSC.nc

In [10]:
```
dataset = Dataset(file_path + 'VHRMC.nc', 'r')
frd = dataset.variables['VHRMC_LIS_FRD'][:].data
frd.shape
```

Out[10]:
```
(12, 760, 3600)
```

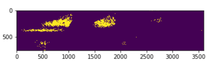
In [11]:
```
#explore the data
print(dataset.variables.keys())
print(dataset.variables['VHRMC_LIS_FRD'].dimensions)
print(dataset.variables['VHRMC_LIS_FRD'].shape)
```

```
odict_keys(['Longitude', 'Latitude', 'Month', 'VHRMC_LIS_FRD', 'VHRMC_LIS_VT'])
('Month', 'Latitude', 'Longitude')
(12, 760, 3600)
```

Create mask

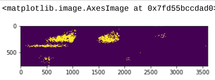
In [12]:
```
#we can mask the places where the flashes are constantly 0 (they sum up to 0)
plt.imshow(np.sum(frd,0)==0)
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x7fd55bdacf90>
```



In [13]:
```
#create a mask, where the flashes are constantly zero:
mask = np.asarray([np.sum(frd,0)==0])
mask = mask.reshape(760, 3600)

#create layers of masks, to match the data:
masks = np.asarray([mask]*frd.shape[0])

#apply the mask on the data, to mask the unimportant areas:
frd_masked = np.ma.masked_where(masks==True, frd)
frd_masked[masks==True] = np.nan

#check grafically if what we did is correct:
#the plot for sums-over-masked-values has to be the same as the above plot for sums-over-zer
os
plt.imshow(np.nansum(np.isnan(frd_masked), 0))
```

Out[13]:
```
<matplotlib.image.AxesImage at 0x7fd55bccdad0>
```



Low Resolution Monthly Time Series (LRMTS) nc file

In [14]:
```
frd = Dataset('./LIS_VHRES/data_climatology/LISOTD_LRMTS.nc', 'r').variables['LRMTS_COM_FR']
[:].data
```

Slicing the data:

we want to keep the measurements from LIS (tropicals and subtropicals) and exclude the ODT measurements in order to have more reliable results

In [15]:
```
#keep the tropicals and subtropicals: find the corresponding gridded latitudes for lat=[-38,
38]
res=2.5
zone=90

gridded_latlon(38, 100)
gridded_latlon(-38, 100)
```

```
51 112
21 112
```

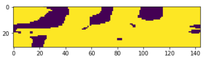
Out[15]:
```
(21, 112)
```

In [16]:
```
#slice the data

frd = frd[ 21:52, :, :]
frd.shape
```

Out[16]:
```
(31, 144, 240)
```

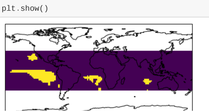
In [17]:
```
#take a look of the data
plt.imshow(np.sum(frd,2)>=0.1)
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x7fd55bc49650>
```



Mask the data

In [27]:
```
#where are the flashes constantly very low?

plt.close('all')
plt.figure()
ax = plt.axes(projection=ccrs.PlateCarree())
ax.set_extent([-180, 180, -90, 90])
ax.coastlines()

ax.imshow(np.sum(frd,2)<=0.03, extent = [-180, 180, -38, 38])

plt.show()
```



Elevation Data Acquisition & Preprocess

In [1]:
```
import pandas as pd
import geopandas as gpd
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
crs = {'init': 'epsg:4326'}
```

Open the full climatology dataset

In [2]:
```
df = pd.read_csv('Full_Climatology.csv')
del df['Unnamed: 0']
gdf = gpd.GeoDataFrame(df, crs=crs, geometry = gpd.points_from_xy(df.Longitude, df.Latitude
))
del df
```

In [3]:
```
#it is a huge data set with more than 1.5 million datapoints
#filter the gdf and keep only the entries with frd > 5
gdf = gdf[gdf.VHRFC_LIS_FRD > 5]
gdf
```

Out[3]:

	Latitude	Longitude	VHRFC_LIS_FRD	geometry
25	-37.95	-143.25	11.821247	POINT (-143.25000 -37.95000)
64	-37.95	-70.25	10.962107	POINT (-70.25000 -37.95000)
102	-37.95	-64.85	5.436871	POINT (-64.85000 -37.95000)
104	-37.95	-64.65	5.594469	POINT (-64.65000 -37.95000)
113	-37.95	-63.65	6.793575	POINT (-63.65000 -37.95000)
...	...	...	...	...
1515491	37.95	131.75	5.861458	POINT (131.75000 37.95000)
1515502	37.95	136.25	5.857105	POINT (136.25000 37.95000)
1515515	37.95	139.35	7.016274	POINT (139.35000 37.95000)
1515563	37.95	154.25	5.855022	POINT (154.25000 37.95000)
1515564	37.95	154.35	7.025729	POINT (154.35000 37.95000)

501002 rows × 4 columns

Connect to Google Engine and extract elevation

In [4]:
```
import ee
print(ee.__version__)
```

```
0.1.216
```

In [5]:
```
# Trigger the authentication flow.
ee.Authenticate()

# Initialize the library.
ee.Initialize()
```

To authorize access needed by Earth Engine, open the following URL in a web browser and follow the instructions:

https://accounts.google.com/o/oauth2/auth?client_id=517222506229-vsmmajv00ul0bs7p89v5m89qs8eb9359.apps.googleusercontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth

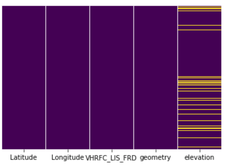
The authorization workflow will generate a code, which you should paste in the box below

```
Enter verification code: 4/1wFqg6G0jEvC72WYYBBjdKqG9iBuP5LdM0QUKd7AqAIBRlA3N0kATHU

Successfully saved authorization token.
```

In [ ]:
```
%%time
dem = ee.Image('USGS/SRTMGL1_003')

#get a copy of the geodatafame and add an empty column for the elevation
gdf_c = gdf.copy()
gdf_c['elevation'] = ""

#for some coordinates there is no elevation info: bypass these rows and continue to the next
set of coordinates
#add each elevation to the corresponding place in the geodataframe
for x,y,i in zip(gdf_c.Longitude, gdf_c.Latitude, range(gdf_c.shape[0])):
    point = ee.Geometry.Point([x, y])
    try:
        gdf_c['elevation'].iloc[i] = dem.sample(point, 30).first().get('elevation').getInfo
()
    except:
        continue

gdf_nan = gdf_c.replace('', np.nan)
```

In [9]:
```
#check for missing elevation data (from originally missing values)= NaN

import seaborn as sns
sns_plot = sns.heatmap(gdf_nan.isna(),yticklabels=False,cbar=False, cmap='viridis')

fig = sns_plot.get_figure()

fig.savefig('missing_values.png', bbox_inches='tight')

print("missed data for elevation: {} datapoints, {}% of the data".format(gdf[gdf.elevation.i
sna()].shape[0], round(gdf[gdf.elevation.isna()].shape[0] / gdf.shape[0] *100,1)))
```

```
missed data for elevation: 53483 datapoints, 10.7% of the data
```



In [10]:
```
gdf_clean = gdf_nan.dropna()

gdf_clean.to_csv('elevation_gdf.csv')
```

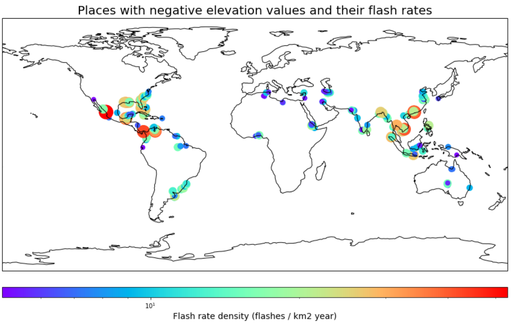
In [11]:
```
gdf_clean
```

Out[11]:

	Latitude	Longitude	VHRFC_LIS_FRD	geometry	elevation
1	-37.95	-70.25	10.962107	POINT (-70.25000 -37.95000)	1131.0
2	-37.95	-64.85	5.436871	POINT (-64.85000 -37.95000)	240.0
3	-37.95	-64.65	5.594469	POINT (-64.65000 -37.95000)	210.0
4	-37.95	-63.65	6.793575	POINT (-63.65000 -37.95000)	166.0
5	-37.95	-63.35	9.180014	POINT (-63.35000 -37.95000)	211.0
...	...	...	...	...	...
500995	37.95	126.75	6.424875	POINT (126.75000 37.95000)	107.0
500996	37.95	127.65	5.036127	POINT (127.65000 37.95000)	399.0
500997	37.95	131.75	5.861458	POINT (131.75000 37.95000)	0.0
500998	37.95	136.25	5.857105	POINT (136.25000 37.95000)	0.0
500999	37.95	139.35	7.016274	POINT (139.35000 37.95000)	11.0

447519 rows × 5 columns

In [12]:
```
#check the negative elevation values
gdf_neg = gdf_clean[gdf_clean.elevation < 0]

from matplotlib.colors import LogNorm
plt.close('all')

f = plt.figure(figsize=(15,10))
ax = f.add_subplot(111, projection=ccrs.PlateCarree())
ax.set_extent([-180, 180,-90, 90])
ax.coastlines()

img = ax.scatter(gdf_neg.Longitude, gdf_neg.Latitude, s=gdf_neg.VHRFC_LIS_FRD*10, c=gdf_neg.
VHRFC_LIS_FRD, cmap=plt.cm.rainbow, norm=LogNorm())
ax.set_title('Places with negative elevation values and their flash rates', fontsize=20)
cb = plt.colorbar(img, orientation='horizontal', pad=0.05, aspect = 50)
cb.set_label('Flash rate density (flashes / km2 year)', fontsize=14)

plt.savefig('Map with negative elevation.png', bbox_inches='tight')

plt.show()
```



In [13]:
```
gdf_neg.sort_values(by='elevation')
```

Out[13]:

	Latitude	Longitude	VHRFC_LIS_FRD	geometry	elevation
467458	32.85	35.55	5.396693	POINT (35.55000 32.85000)	-216.0
325696	11.65	42.45	9.109092	POINT (42.45000 11.65000)	-154.0
348462	14.15	40.35	5.448632	POINT (40.35000 14.15000)	-120.0
349312	14.25	40.25	7.659740	POINT (40.25000 14.25000)	-120.0
347622	14.05	40.45	5.580790	POINT (40.45000 14.05000)	-119.0
...	...	...	...	...	...
309234	9.95	105.85	11.509728	POINT (105.85000 9.95000)	-1.0
412625	24.35	68.65	5.121790	POINT (68.65000 24.35000)	-1.0
413664	24.55	-107.65	6.666922	POINT (-107.65000 24.55000)	-1.0
318370	10.95	-71.35	6.312249	POINT (-71.35000 10.95000)	-1.0
379084	18.85	-91.25	15.736501	POINT (-91.25000 18.85000)	-1.0

360 rows × 5 columns

PCA - Monthly Climatology dataset

In [1]:
```
from netCDF4 import Dataset
import numpy as np
from sklearn.preprocessing import normalize
import dask.array as da
from dask_ml.decomposition import PCA
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
```

In [2]:
```
dataset = Dataset('./LIS_VHRES/data_climatology/VHRMC.nc', 'r').variables['VHRMC_LIS_FRD']
[:]
frd = dataset.data
frd.shape
```

Out[2]:
```
(12, 760, 3600)
```

In [3]:
```
#create a mask and appply it on the data (see Data Acquisition & Preprocess)
mask = np.asarray([np.sum(frd,0)==0])
mask = mask.reshape(760, 3600)
masks = np.asarray([mask]*frd.shape[0])

frd_masked = np.ma.masked_where(masks==True, frd)
frd_masked[masks==True] = np.nan

frd_reshaped = frd_masked.reshape(frd_masked.shape[0], frd_masked.shape[1]*frd_masked.shape[
2])

#mask out the unimportant areas
mask2 = np.isnan(frd_reshaped[0, :])
frd_clean = frd_reshaped[:, ~mask2]
```

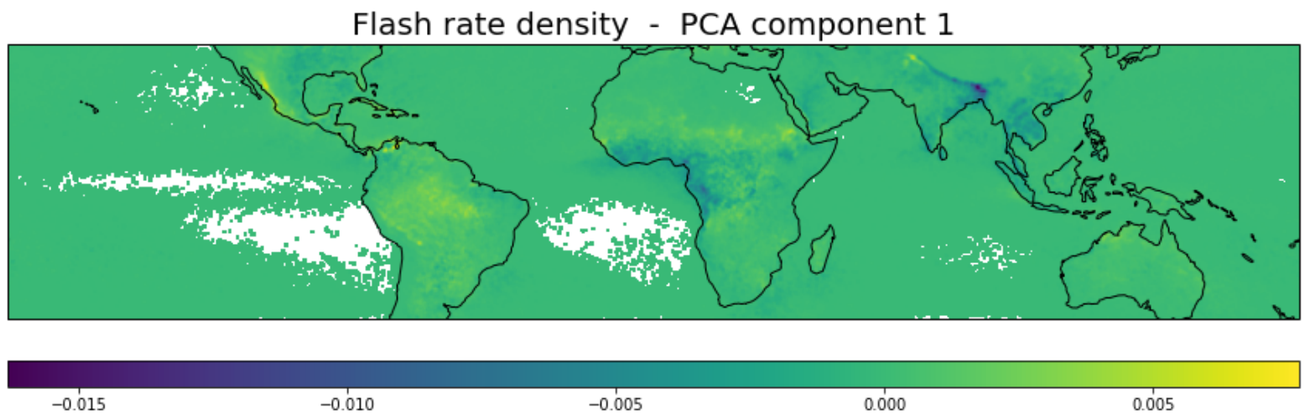
In [4]:
```
#normalize the data
frd_norm = normalize(frd_clean)
#put data in dask chunks
frd_da = da.from_array(frd_norm, chunks=(frd_norm.shape[0], frd_norm.shape[1]))
```

In [5]:
```
#PCA with 3 components
n_components = 3

pca = PCA(n_components, whiten=True)
pca.fit(frd_da)

print('The explained variance ratio is {} \n'.format(pca.explained_variance_ratio_))
print('The PCA describes the {}% of the data'.format(round(np.nansum(pca.explained_variance_
ratio_)*100)))

pca_data = pca.components_.reshape((n_components*frd_da.shape[1]))
```

```
The explained variance ratio is [0.63044442 0.17422906 0.09685796]

The PCA describes the 90.0% of the data
```

In [6]:
```
#put everything back together and mask out the unimportant areas
frd_pca = np.ones(n_components*760*3600)
frd_pca.fill(np.nan)
mask3 = np.asarray([mask]*n_components)
frd_pca[~mask3.ravel()]= pca_data
frd_pca = frd_pca.reshape(n_components, 760, 3600)
```

In [11]:
```
plt.close('all')
plt.figure(figsize=((15,10)))

#plot the map
img = plt.axes(projection=ccrs.PlateCarree())
#img.set_extent([-180, 180, -90, 90])
img.coastlines()

#PCA results for the first component
pca_1 = img.imshow(frd_pca[1,:,:], extent = [-180, 180, -38, 38])
cb = plt.colorbar(pca_1, orientation='horizontal', pad=0.05, aspect = 50)

plt.title('Flash rate density  -  PCA component 1', loc = 'center', fontsize = 20)

plt.savefig('PCA_Monthly Climatology.png', bbox_inches='tight')

plt.show()
```



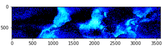
Top 10 Hotspots - Full, Monthly & Diurnal Climatology datasets

In [1]:
```
from netCDF4 import Dataset
import numpy as np
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from matplotlib.colors import LogNorm
import cartopy.crs as ccrs
import cartopy.feature as cfeature
crs = {'init': 'epsg:4326'}
```

In [2]:
```
file_path = './LIS_VHRFS/data_climatology/'
frd = Dataset(file_path + 'VHRFC.nc', 'r').variables['VHRFC_LIS_FRD'][:].data
masked_frd = np.ma.masked_where(frd == 0, frd) #for the purposes of the first plot - black m
ap
```

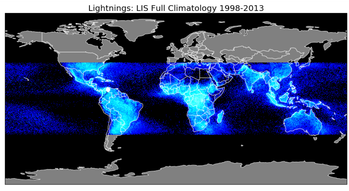
In [3]:
```
# Create a new colormap
cmap = LinearSegmentedColormap.from_list('',['mediumblue', "blue", "cyan", "white"])
#set the masked entries black
cmap.set_bad(color='black')
#check to see how it looks like (inverted)
plt.imshow(masked_frd, cmap=cmap, norm=LogNorm())
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x7fbe4acf2a90>
```



In [4]:
```
#find the place with the highest flash rate - reference spot

maximum  = int(round(np.amax(frd)))
lat = np.where(masked_frd == np.amax(frd))[0][0]
lon = np.where(masked_frd == np.amax(frd))[1][0]
real_lat = round(lat*0.1 - 38, 2)
real_lon = round(lon*0.1 - 180, 2)
```

In [21]:
```
plt.close('all')
plt.figure(figsize=((15,10)))

#plot the black map
img = plt.axes(projection=ccrs.PlateCarree())
img.set_extent([-180, 180, -90, 90])
img.coastlines(color='white')
img.add_feature(cfeature.LAND, facecolor='gray')
img.add_feature(cfeature.BORDERS, edgecolor='white')
img.add_feature(cfeature.OCEAN, facecolor='black')

#frd
lightnings = img.imshow(masked_frd, cmap=cmap, extent = [-180, 180, -38, 38],  norm=LogNorm
())

#the place with the highest frd - reference spot
maxim = img.plot(real_lon, real_lat, 'wo', markersize=10)

plt.title('Lightnings: LIS Full Climatology 1998-2013', fontsize=20)

plt.savefig("Lightnings_full_climatology_black map_log scale.png", bbox_inches='tight')

plt.show()
```



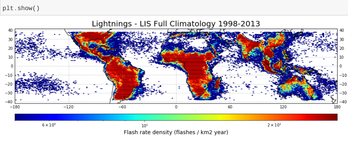
In [23]:
```
#open the full climatology file in a geodataframe and filter it (keep >5 flashes)
df = pd.read_csv('./Full_Climatology.csv')
del df['Unnamed: 0']

gdf = gpd.GeoDataFrame(df, crs=crs, geometry = gpd.points_from_xy(df.Longitude, df.Latitude
))
gdf = gdf[gdf.VHRFC_LIS_FRD > 5]
del df
gdf
```

Out[23]:

	Latitude	Longitude	VHRFC_LIS_FRD	geometry
25	-37.95	-143.25	11.821247	POINT (-143.25000 -37.95000)
64	-37.95	-70.25	10.962107	POINT (-70.25000 -37.95000)
102	-37.95	-64.85	5.436871	POINT (-64.85000 -37.95000)
104	-37.95	-64.65	5.594469	POINT (-64.65000 -37.95000)
113	-37.95	-63.65	6.793575	POINT (-63.65000 -37.95000)
...	...	...	...	...
1515491	37.95	131.75	5.861458	POINT (131.75000 37.95000)
1515502	37.95	136.25	5.857105	POINT (136.25000 37.95000)
1515515	37.95	139.35	7.016274	POINT (139.35000 37.95000)
1515563	37.95	154.25	5.855022	POINT (154.25000 37.95000)
1515564	37.95	154.35	7.025729	POINT (154.35000 37.95000)

501002 rows × 4 columns

In [26]:
```
plt.close('all')

f = plt.figure(figsize=(15,10))

ax = f.add_subplot(111, projection=ccrs.PlateCarree())
#ax.set_extent([-180, 180, -90, 90])
ax.coastlines()
gl = ax.gridlines(crs=ccrs.PlateCarree(), draw_labels=True, linewidth=0.8, alpha=0.5, color=
'grey', linestyle='--')
gl.xlabels_top=False

img = ax.hexbin(gdf.Longitude, gdf.Latitude, gridsize=300, bins='log', mincnt=1, cmap=plt.cm
.jet, vmin=5)

cb = plt.colorbar(img, orientation='horizontal', pad=0.05, aspect = 50)
cb.set_label('Flash rate density (flashes / km2 year)', fontsize=14)
plt.title("Lightnings - LIS Full Climatology 1998-2013", fontsize = 20)

plt.savefig("Lightnings Full Climatology - hexbin map - threshold 5.png", bbox_inches='tigh
t')

plt.show()
```



Flash Hotspots

In [8]:
```
#We will work with a smaller sample, since we are only looking for the top 10 hotspots
gdf = gdf[gdf.VHRFC_LIS_FRD > 100]
gdf = gdf.sort_values(by='VHRFC_LIS_FRD', ascending=False)
```

In [9]:
```
cities_file = './cities_data/ne_10m_populated_places_simple.shp'
cities = gpd.read_file(cities_file)
cities = cities[['name','geometry']]
cities['geometry'] = cities.geometry.buffer(1)
```

In [10]:
```
top_gdf = gpd.sjoin(gdf, cities, how='inner', op='intersects')
del top_gdf['index_right']

top_gdf.drop_duplicates(subset ="geometry", keep = 'first', inplace = True)
top_gdf.drop_duplicates(subset ="name", keep = 'first', inplace = True)

top_gdf = top_gdf.sort_values(by=['VHRFC_LIS_FRD'], ascending=False)
```

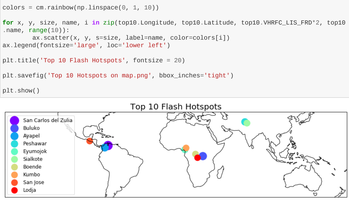
In [11]:
```
#we buffer the geometry, so we can keep only the polygons that are more than 100 km apart
top_gdf['geometry'] = top_gdf.geometry.buffer(1)

#discart the overlapping polygons
import itertools

for polygon_i,polygon_j in  itertools.combinations(top_gdf.geometry, 2):
    if polygon_i.intersects(polygon_j):
        top_gdf = top_gdf[top_gdf.geometry != polygon_j]
```

In [12]:
```
#rearrange everything and create the top 10 dataframe:
top10 = top_gdf.head(10).copy()
top10.reset_index(inplace=True, drop=True)
top10 = top10[['name','VHRFC_LIS_FRD', 'Longitude', 'Latitude', 'geometry']]

top10.to_csv('Top10 Hotspots.csv')

top10
```

Out[12]:

	name	VHRFC_LIS_FRD	Longitude	Latitude	geometry
0	San Carlos del Zulia	232.52394	-71.65	9.75	POLYGON ((-70.65000 9.75000, -70.65482 9.65198,...
1	Bukoko	184.34459	28.15	-1.55	POLYGON ((29.15000 -1.55000, 29.14518 -1.64802...
2	Ayapel	172.29301	-75.35	7.55	POLYGON ((-74.35000 7.55000, -74.35482 7.45198,...
3	Peshawar	143.10955	72.35	34.45	POLYGON ((73.35000 34.45000, 73.34518 34.35198,...
4	Eyumojok	129.58334	9.35	5.25	POLYGON ((10.35000 5.25000, 10.34518 5.15196, ...
5	Sialkote	121.40682	74.55	33.35	POLYGON ((75.55000 33.35000, 75.54518 33.25196,...
6	Boende	117.98413	20.35	0.55	POLYGON ((21.35000 0.55000, 21.34518 0.45198, ...
7	Kumbo	116.78391	10.45	6.95	POLYGON ((11.45000 6.95000, 11.44518 6.85198, ...
8	San Jose	116.76327	-91.15	14.35	POLYGON ((-90.15000 14.35000, -90.15482 14.251...
9	Lodja	108.47006	22.55	-3.55	POLYGON ((23.55000 -3.55000, 23.54518 -3.64802...

In [30]:
```
import matplotlib.cm as cm

plt.close('all')

f = plt.figure(figsize=(15,10))
ax = f.add_subplot(111, projection=ccrs.PlateCarree())
ax.set_extent([-180, 180, -45, 45])
ax.coastlines()

colors = cm.rainbow(np.linspace(0, 1, 10))

for x, y, size, name, i in zip(top10.Longitude, top10.Latitude, top10.VHRFC_LIS_FRD^2, top10
.name, range(10)):
        ax.scatter(x, y, s=size, label=name, color=colors[i])
ax.legend(fontsize='large', loc='lower left')

plt.title('Top 10 Flash Hotspots', fontsize = 20)

plt.savefig('Top 10 Hotspots on map.png', bbox_inches='tight')

plt.show()
```



Brief time series analysis for the top 10 hotspots

based on the diurnal (hourly) and monthly climatology datasets

In [14]:
```
#hourly flash rates
frd_h = Dataset(file_path + 'VHROC.nc', 'r').variables['VHROC_LIS_FRD'][:].data
frd_h.shape
```

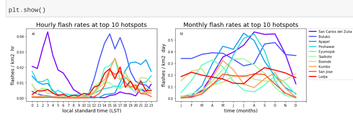
Out[14]:
```
(24, 760, 3600)
```

In [15]:
```
#monthly flash rates
frd_m = Dataset(file_path + 'VHRMC.nc', 'r').variables['VHRMC_LIS_FRD'][:].data
frd_m.shape
```

Out[15]:
```
(12, 760, 3600)
```

In [16]:
```
#find the corresponding gridded lats and lons
top10['grid_Lon'] = round((top10.Longitude + 180)*10)
top10['grid_Lat'] = round((top10.Latitude  + 38)*10)
```

In [17]:
```
plt.close('all')

f = plt.figure(figsize=(15,4))
colors = cm.rainbow(np.linspace(0, 1, 10))

ax = f.add_subplot(121)
for i in range(10):
    a = frd_h[:, int(top10.grid_Lat.iloc[i]),int(top10.grid_Lon.iloc[i])]
    ax.plot(a, linewidth=3, c=colors[i])
ax.set_xticks(np.arange(0, 24, 1.0))
ax.set_xlabel('local standard time (LST)',  fontsize=14)
ax.set_ylabel('flashes / km2  hr', fontsize=14)
plt.figtext(0.14, 0.82, 'a)')
plt.title('Hourly flash rates at top 10 hotspots', fontsize=18)


ax = f.add_subplot(122)
for i in range(10):
    a = frd_m[:, int(top10.grid_Lat.iloc[i]),int(top10.grid_Lon.iloc[i])]
    ax.plot(a, linewidth=3, c=colors[i], label=top10.name.iloc[i])
ax.set_xticks(np.arange(0, 13, 1.0))
ax.set_xticklabels(['J', 'F', 'M', 'A', 'M', 'J', 'J', 'A', 'S', 'O', 'N', 'D'])
ax.set_xlabel("time (months)", fontsize=14)
ax.set_ylabel('flashes / km2  day', fontsize=14)
plt.figtext(0.55, 0.82, 'b)')
plt.title('Monthly flash rates at top 10 hotspots', fontsize=18)

# Put a legend to the right of the current axis
ax.legend(loc='center left', bbox_to_anchor=(1, 0.65))

plt.subplots_adjust(wspace=0.14)

plt.savefig('Brief time series for hotspots.png', bbox_inches = 'tight')

plt.show()
```



Interactive map - Full Climatology & Elevation datasets

In [1]:
```
import ee
print(ee.__version__)
```

```
0.1.216
```

In [2]:
```
# Trigger the authentication flow.
ee.Authenticate()

# Initialize the library.
ee.Initialize()
```

To authorize access needed by Earth Engine, open the following URL in a web browser and follow the instructions:

https://accounts.google.com/o/oauth2/auth?client_id=517222506229-vsmmajv00ul0bs7p89v5m89qs8eb9359.apps.googleusercontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth

The authorization workflow will generate a code, which you should paste in the box below

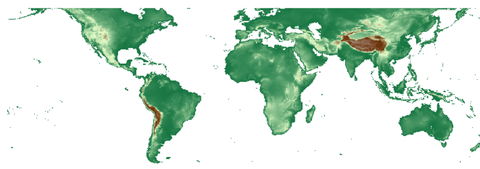
```
Enter verification code: 4/2AGrvsJX4R0TBDJe_8Mc1L_B8gkcmY4kLn3NB4a5va6oc_fhdIUDJJA

Successfully saved authorization token.
```

In [3]:
```
dem = ee.Image('USGS/SRTMGL1_003')
xy = ee.Geometry.Point([35.55,32.85 ])
elev = dem.sample(xy, 30).first().get('elevation').getInfo()
# Check - Print the minimum elevation.
print('Min elevation (m):', elev)
```

```
Min elevation (m): -216
```

In [4]:
```
# Import the Image function from the IPython.display module.
from IPython.display import Image

# Display a thumbnail of global elevation.
Image(url = dem.updateMask(dem.gt(0))
  .getThumbUrl({'min': -216, 'max': 8848, 'dimensions': 712,
                'palette': ['006633', 'E5FFCC', '662A00', 'D8D8D8', 'F5F5F5']}))
```

Out[4]:



In [6]:
```
import pandas as pd
import geopandas as gpd
import json
import numpy as np
import cartopy.crs as ccrs
from matplotlib.colors import LogNorm
crs = {'init': 'epsg:4326'}

df = pd.read_csv('./elevation_nan.csv')
gdf = gpd.GeoDataFrame(df, crs=crs, geometry = gpd.points_from_xy(df.Longitude, df.Latitude
))
del df
```

In [7]:
```
gdf.head()
```

Out[7]:

	Unnamed: 0	Latitude	Longitude	VHRFC_LIS_FRD	geometry	elevation
0	25	-37.95	-143.25	11.821247	POINT (-143.25000 -37.95000)	NaN
1	64	-37.95	-70.25	10.962107	POINT (-70.25000 -37.95000)	1131.0
2	102	-37.95	-64.85	5.436871	POINT (-64.85000 -37.95000)	240.0
3	104	-37.95	-64.65	5.594469	POINT (-64.65000 -37.95000)	210.0
4	113	-37.95	-63.65	6.793575	POINT (-63.65000 -37.95000)	166.0

In [11]:
```
#Threshold for the flash rate greater than 20
gdf=gdf[gdf['VHRFC_LIS_FRD'] > 20]
```

In [12]:
```
locations = gdf[['Latitude', 'Longitude']]
locationlist = locations.values.tolist()
len(locationlist)
locationlist[7]
```

Out[12]:
```
[-37.65000000000001, -58.649999999999984]
```

In [14]:
```
#categories for the flash rates

def magcolors(counter):
    if counter['VHRFC_LIS_FRD'] < 25:
        return 'black'
    if counter['VHRFC_LIS_FRD'] < 30:
        return 'midnightblue'
    if counter['VHRFC_LIS_FRD'] < 35:
        return 'navy'
    if counter['VHRFC_LIS_FRD'] < 40:
        return 'darkblue'
    elif counter['VHRFC_LIS_FRD']< 50:
        return 'blue'
    if counter['VHRFC_LIS_FRD'] < 60:
        return 'cornflowerblue'
    if counter['VHRFC_LIS_FRD'] < 70:
        return 'royalblue'
    if counter['VHRFC_LIS_FRD'] < 80:
        return 'deepskyblue'
    elif counter['VHRFC_LIS_FRD'] < 130:
        return 'cyan'
    elif counter['VHRFC_LIS_FRD'] < 200:
        return 'lightskyblue'
    elif counter['VHRFC_LIS_FRD'] <233:
        return 'aliceblue'
    else:
        return 'gold'
gdf['color'] = gdf.apply(magcolors, axis=1)
```

In [ ]:
```
# Import the Folium library.
import folium
from folium import plugins
import folium.plugins as plugins
import branca.colormap as cm


# Define a method for displaying Earth Engine image tiles to folium map.
def add_ee_layer(self, ee_image_object, vis_params, name):
  map_id_dict = ee.Image(ee_image_object).getMapId(vis_params)
  folium.raster_layers.TileLayer(
    tiles = map_id_dict['tile_fetcher'].url_format,
    attr = 'Map Data &copy; <a href="https://earthengine.google.com/">Google Earth Engine</a
>',
    name = name,
    overlay = True,
    control = True
  ).add_to(self)

# Add EE drawing method to folium.
folium.Map.add_ee_layer = add_ee_layer

# Set visualization parameters.
vis_params = {
  'min': 0,
  'max': 8848,
  'palette': ['407115','4c8619','589c1d','64b121','70c625','7dd82d','8adc43', 'ffffd4','fee3
91', 'fec44f','fe9929',
             'dc9543','d8882d','c67b25','b16e21','9c611d','865319','714615','a5a5a5']}


vis_params2 = {
  'min': 20,
  'max': 233,
  'palette': ['black','midnightblue','navy','darkblue','blue','royalblue','cornflowerblue',
'deepskyblue','cyan','lightskyblue','aliceblue']}

# Create a folium map object.
my_map = folium.Map(location=[30.31, -2], zoom_start=3,tiles='Stamen Terrain')


from folium.plugins import MarkerCluster
my_map = folium.Map(location=[30.31, -2], tiles='CartoDB positron', zoom_start=3)
marker_cluster = MarkerCluster().add_to(my_map)

for point in range(0, len(locationlist)):
    folium.Circle( locationlist[point], popup=gdf['VHRFC_LIS_FRD'][point],icon=folium.Icon(c
olor=gdf["color"][point], icon_color=gdf['color'][point], icon='glyphicon-star', angle=0),
     radius=1000,
    color=gdf["color"][point],
    fill=True,
    #fill_color='green'
    ).add_to(my_map)

folium.raster_layers.TileLayer(
    tiles='http://{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}',
    attr='google',
    name='google maps',
    max_zoom=50,
    subdomains=['mt0', 'mt1', 'mt2', 'mt3'],
    overlay=False,
    control=True,
).add_to(my_map)


colormap = cm.LinearColormap(colors=['black','midnightblue','navy','darkblue','blue','royalb
lue','cornflowerblue','deepskyblue','cyan','lightskyblue','aliceblue'],
                             index=[25,30,35 ,40,50, 60,70, 80, 130,200,233],vmin=20,vmax=23
3)
colormap.caption = 'FRD [flashes / sq.Km]'
colormap.add_to(my_map)

my_map.add_ee_layer(dem.updateMask(dem.gt(0)), vis_params, 'DEM')


colormap = cm.LinearColormap(colors=['#407115','#4c8619','#589c1d','#64b121','#70c625','#7dd
82d', '#8adc43','#ffffd4','#fee391','#fec44f',
                                     '#fe9929','#dc9543','#d8882d','#c67b25','#b16e21','#9c6
11d','#865319','#714615','#a5a5a5'],
                             index=[0,500,1000,1500, 2000,2500,3000,3500,4000,4500, 5000,550
0,6000,6500,7000,7500,8000,8500,8848],vmin=0,vmax=8848)
colormap.caption = 'Dem [m]'
colormap.add_to(my_map)


# Add a layer control panel to the map.
my_map.add_child(folium.LayerControl())

# Display the map.
```

Clustering - Full Climatology dataset, Elevation data

In [2]:
```
import numpy as np
import pandas as pd
import geopandas as gpd
import json
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import cartopy.crs as ccrs
crs = {'init': 'epsg:4326'}

from sklearn.cluster import KMeans
```

In [3]:
```
#open csv file (with the filtered gdf)
df = pd.read_csv('./elevation_gdf.csv')
gdf = gpd.GeoDataFrame(df, crs=crs, geometry = gpd.points_from_xy(df.Longitude, df.Latitude
))
del gdf['Unnamed: 0']
del df

#add standarized parameters
gdf['FRD_std'] = (gdf.VHRFC_LIS_FRD - np.nanmean(gdf.VHRFC_LIS_FRD)) / np.nanstd(gdf.VHRFC_L
IS_FRD)
gdf['elevation_std'] = (gdf.elevation - np.nanmean(gdf.elevation)) / np.nanstd(gdf.elevation
)
gdf
```

Out[3]:

	Latitude	Longitude	VHRFC_LIS_FRD	geometry	elevation	FRD_std	elevation_std
0	-37.95	-70.25	10.962107	POINT (-70.25000 -37.95000)	1131.0	-0.415082	0.988919
1	-37.95	-64.85	5.436871	POINT (-64.85000 -37.95000)	240.0	-0.894840	-0.460965
2	-37.95	-64.65	5.594469	POINT (-64.65000 -37.95000)	210.0	-0.881156	-0.509782
3	-37.95	-63.65	6.793575	POINT (-63.65000 -37.95000)	166.0	-0.777037	-0.581382
4	-37.95	-63.35	9.180014	POINT (-63.35000 -37.95000)	211.0	-0.569821	-0.508155
...	...	...	...	...	...	...	...
447514	37.95	126.75	6.424875	POINT (126.75000 37.95000)	107.0	-0.809051	-0.677390
447515	37.95	127.65	5.036127	POINT (127.65000 37.95000)	399.0	-0.929637	-0.202231
447516	37.95	131.75	5.861458	POINT (131.75000 37.95000)	0.0	-0.857973	-0.851506
447517	37.95	136.25	5.857105	POINT (136.25000 37.95000)	0.0	-0.858351	-0.851506
447518	37.95	139.35	7.016274	POINT (139.35000 37.95000)	11.0	-0.757700	-0.833606

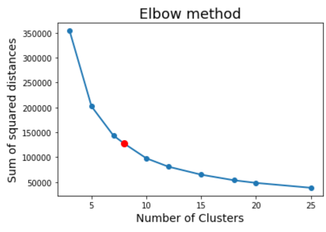
447519 rows × 7 columns

In [4]:
```
# K-means clustering

def k_cluster(n_clusters):
    n_init=100
    max_iter=1000
    clf = KMeans(init='k-means++', n_init=n_init, n_clusters=n_clusters, random_state=100, m
ax_iter=max_iter)
    global labels
    labels = clf.fit_predict(gdf[['FRD_std','elevation_std']])
    dist = clf.fit(gdf[['FRD_std','elevation_std']]).inertia_
    return dist
```

In [ ]:
```
#RUN ONLY ONCE !!!

#create the distances list
clusters_list = [3, 5, 7, 8, 10, 12, 15, 18, 20, 25]
distances = []
for i in clusters_list:
    distances.append(round(k_cluster(i),2))

#it takes a lot of time to run, so better save it:
with open("cluster_distances_elev_FRD.json", 'w') as f:
    json.dump(distances, f, indent=2)
```

In [5]:
```
#open the distances list
with open("cluster_distances_elev_FRD.json", 'r') as f:
    distances = json.load(f)
```

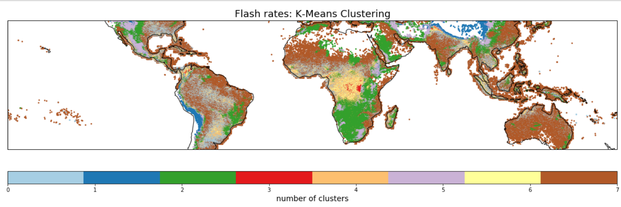
In [6]:
```
clusters_list = [3, 5, 7, 8, 10, 12, 15, 18, 20, 25]
plt.close('all')

plt.plot(clusters_list, distances, lw=2, ls='-', marker='o', markersize=6)
plt.plot(clusters_list[3], distances[3], 'ro', markersize=8)
plt.xlabel("Number of Clusters", fontsize=14)
plt.ylabel("Sum of squared distances", fontsize=14)
plt.title('Elbow method', fontsize=18)


plt.savefig('Elbow method.png', bbox_inches='tight')

plt.show()
```



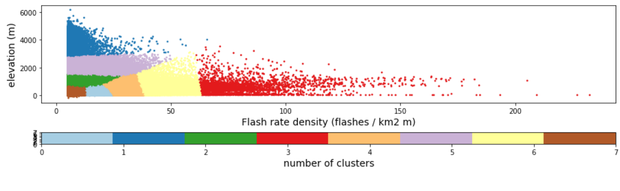
In [7]:
```
n_cluster = 8
k_cluster(n_cluster)
```

Out[7]:
```
127203.40467035421
```

In [18]:
```
plt.close('all')
f = plt.figure(figsize=(20,15))

#plot the map
ax = f.add_subplot(111, projection=ccrs.PlateCarree())
ax.set_extent([-180, 180, -38, 38])
ax.coastlines()

#plot the clusters
img = ax.scatter(gdf.geometry.x, gdf.geometry.y, c=labels, s=3, cmap=plt.cm.get_cmap('Paire
d', n_cluster))
ax.set_title('Flash rates: K-Means Clustering', fontsize=18)
cb = plt.colorbar(img, orientation='horizontal', pad=0.05, aspect = 50)
cb.set_label('number of clusters', fontsize=14)


plt.savefig('Map with K-Clusters.png', bbox_inches='tight')

plt.show()
```



In [20]:
```
plt.close('all')
f = plt.figure(figsize=(15,4))


#plot FRD - elevation in respect to clustering
ax = f.add_subplot(111)
img = ax.scatter(gdf.VHRFC_LIS_FRD, gdf.elevation, c=labels, s=3, cmap=plt.cm.get_cmap('Pair
ed', n_cluster))
ax.set_xlabel('Flash rate density (flashes / km2 m)', fontsize=14)
ax.set_ylabel('elevation (m)', fontsize=14)

cb = plt.colorbar(img, orientation='horizontal', pad=0.2, aspect = 50)
cb.set_label('number of clusters', fontsize=14)
cb.ax.yaxis.set_ticks(range(n_cluster))

plt.savefig('Cluster scatterplot.png', bbox_inches='tight')

plt.show()
```



Time Series Clustering - LIS/OTD Monthly Time Series

In [1]:
```
from netCDF4 import Dataset
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import dask.array as da
from scipy.cluster.hierarchy import linkage, fcluster, dendrogram
from mpl_toolkits.axes_grid1 import make_axes_locatable
import cartopy.crs as ccrs
crs = {'init': 'epsg:4326'}
```

In [2]:
```
frd = Dataset('./LIS_VHRES/data_climatology/LISOTD_LRMTS.nc', 'r').variables['LRMTS_COM_FR']
[:].data
frd.shape
```

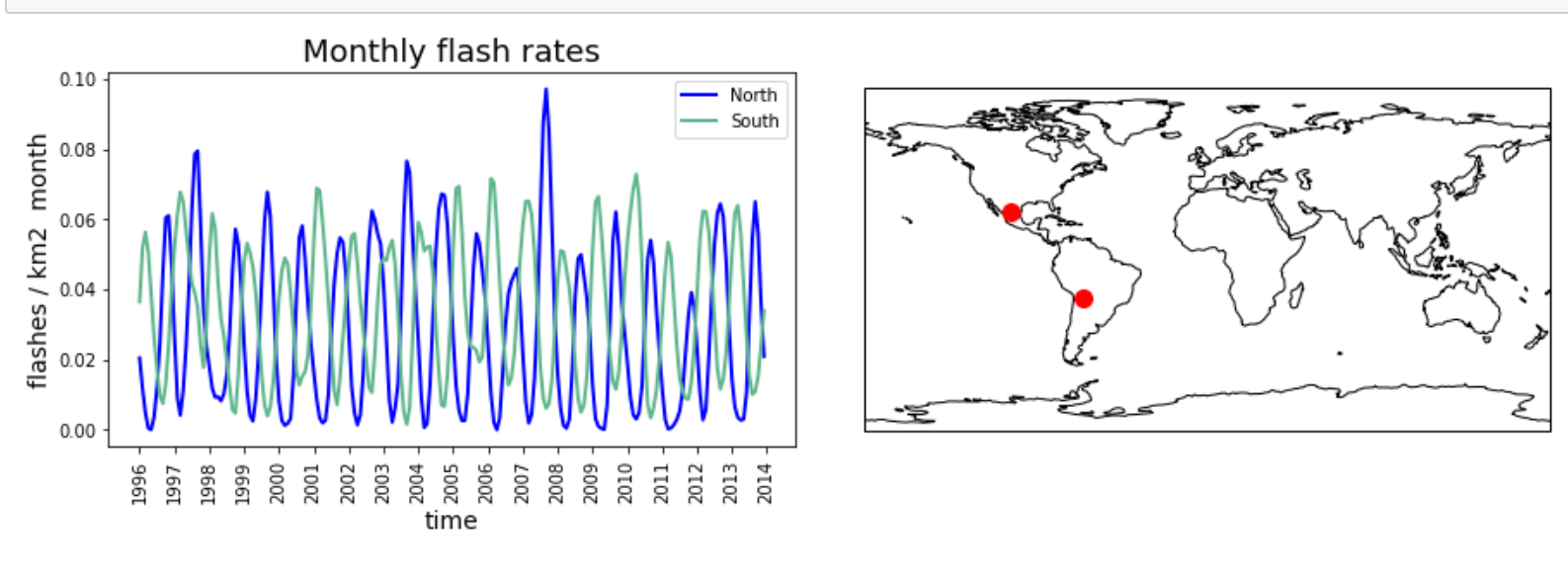
Out[2]: (72, 144, 240)

In [3]:
```
#create two functions to tranfrorm the coords back and forth
res = 2.5
zone = 90

def gridded_latlon(real_lat, real_lon):
    grid_lat = round((real_lat + zone)/res)
    grid_lon = round((real_lon + 180)/res)
    print(grid_lat, grid_lon)
    return grid_lat, grid_lon


def real_latlon(grid_lat, grid_lon):
    real_lat = grid_lat*res - zone
    real_lon = grid_lon*res - 180
    print(real_lat, real_lon)
    return real_lat, real_lon
```

In [4]:
```
#Choose 2 spots, one from northern and one from the southern hemisphere and check the antico
rrelation
north = gridded_latlon(25, -103)
south = gridded_latlon(-20, -65)
```

```
46 31
28 46
```

In [5]:
```
#plot 19 years: January 1996 - January 2015
ser_north = pd.Series(frd[46, 31, 9:225])
ser_south = pd.Series(frd[28, 46, 9:225])


plt.close('all')
f = plt.figure(figsize=((15,4)))

#plot the time series for the two spots
ax = f.add_subplot(121)
ax.plot(ser_north, linewidth=2, c='b',       label='North')
ax.plot(ser_south, linewidth=2, c='#5fb78d', label = 'South')
plt.legend(loc='upper right')
ax.set_xticks(np.arange(0, 228, 12))
ax.set_xticklabels(list(np.arange(1996, 2016)), rotation='vertical')
ax.set_xlabel("time", fontsize=14)
ax.set_ylabel('flashes / km2  month', fontsize=14)
ax.set_title('Monthly flash rates', fontsize=18)

#plot the two spots on the map
ax = f.add_subplot(122, projection=ccrs.PlateCarree())
ax.coastlines()
ax.set_extent([-180, 180, -90, 90])
ax.plot(-103, 25, 'ro', markersize=10)
ax.plot(-65, -20, 'ro', markersize=10)

plt.subplots_adjust(wspace=0.1)

plt.savefig('North-South_time series.png', bbox_inches='tight')

plt.show()
```



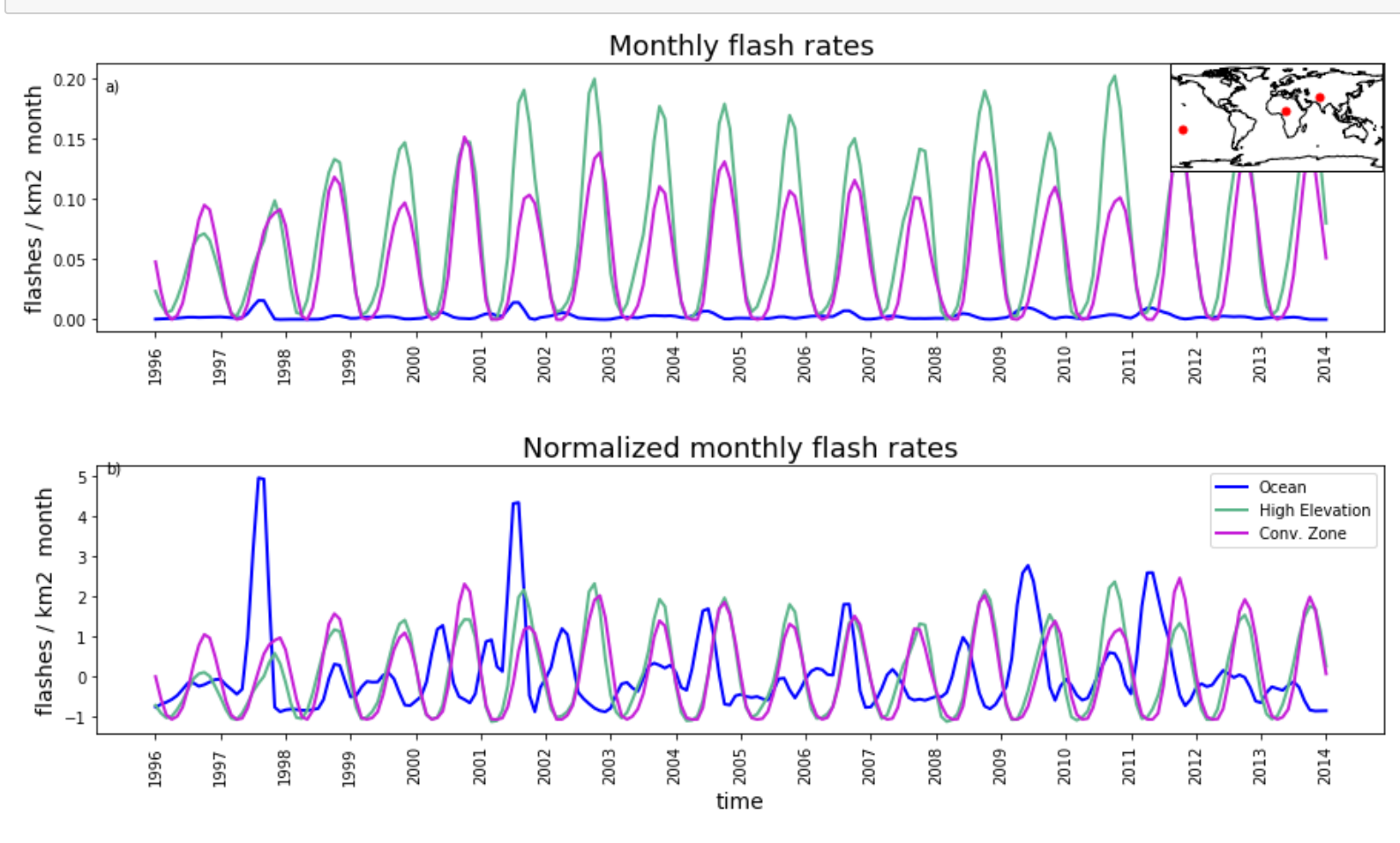
In [6]:
```
#Choose 3 random spots with different flash rates: ocean, high elevation and convergence zon
e
#find the corresponding gridded coordinates
ocean     = gridded_latlon(-20, -160)
high_elev = gridded_latlon(33, 72)
conv_zone = gridded_latlon(10, 15)
```

```
28 8
49 101
40 78
```

In [7]:
```
#normalization
import dask.array as da
frd_da = da.from_array(frd, chunks=(31, 144, 180))

fx = lambda x: (x - np.nanmean(x)) / np.nanstd(x)

frd_norm = da.apply_along_axis(fx, 2, frd_da).compute()
```

```
/home/student/anaconda3/envs/BigData/lib/python3.7/site-packages/ipykernel_launcher.py:5: Run
timeWarning: invalid value encountered in true_divide
  """
/home/student/anaconda3/envs/BigData/lib/python3.7/site-packages/ipykernel_launcher.py:5: Run
timeWarning: invalid value encountered in true_divide
  """
```

In [8]:
```
#keep 10 years (for the plot): January 2001 - January 2014
ser_ocean     = pd.Series(frd[28, 8, 9:226])
ser_high_elev = pd.Series(frd[49, 101, 9:226])
ser_conv_zone = pd.Series(frd[40, 78, 9:226])

plt.close('all')
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=((15,8)))


ax1.plot(ser_ocean,     linewidth=2, c='b',       label='Ocean')
ax1.plot(ser_high_elev, linewidth=2, c='#5fb78d', label = 'High Elevation')
ax1.plot(ser_conv_zone, linewidth=2, c='#c620d8', label= 'Conv. Zone')
ax1.set_xticks(np.arange(0, 218, 12))
ax1.set_xticklabels(list(np.arange(1996, 2015)), rotation='vertical')
ax1.set_ylabel('flashes / km2  month', fontsize=14)
plt.figtext(0.13, 0.85, 'a)')
ax1.set_title('Monthly flash rates', fontsize=18)


ax2.plot(pd.Series(frd_norm[28, 8, 9:226 ]),   linewidth=2, c='b',       label='Ocean')
ax2.plot(pd.Series(frd_norm[49, 101, 9:226 ]), linewidth=2, c='#5fb78d', label = 'High Eleva
tion')
ax2.plot(pd.Series(frd_norm[40, 78, 9:226 ]),  linewidth=2, c='#c620d8', label= 'Conv. Zone'
)
ax2.legend(loc='upper right')
ax2.set_xticks(np.arange(0, 218, 12))
ax2.set_xticklabels(list(np.arange(1996, 2015)), rotation='vertical')
ax2.set_xlabel("time", fontsize=14)
ax2.set_ylabel('flashes / km2  month', fontsize=14)
plt.figtext(0.13, 0.42, 'b)')
ax2.set_title('Normalized monthly flash rates', fontsize=18)


ax3 = fig.add_axes([0.76, 0.76, 0.15, 0.12], projection=ccrs.PlateCarree())
ax3.coastlines()
ax3.set_extent([-180, 180, -90, 90])
ax3.plot(-160, -20, 'ro', markersize=5)
ax3.plot(72, 33, 'ro', markersize=5)
ax3.plot(15, 10, 'ro', markersize=5)

plt.subplots_adjust(hspace=0.5)

plt.savefig('3 random places_time series.png', bbox_inches='tight')

plt.show()
```



In [9]:
```
#keep the tropicals and subtropicals: lat=[-38, 38]
frd_cut = frd[21:52, :, : ]
frd_norm_cut = frd_norm[21:52, :, :]

frd_cut.shape
```

Out[9]: (31, 144, 240)

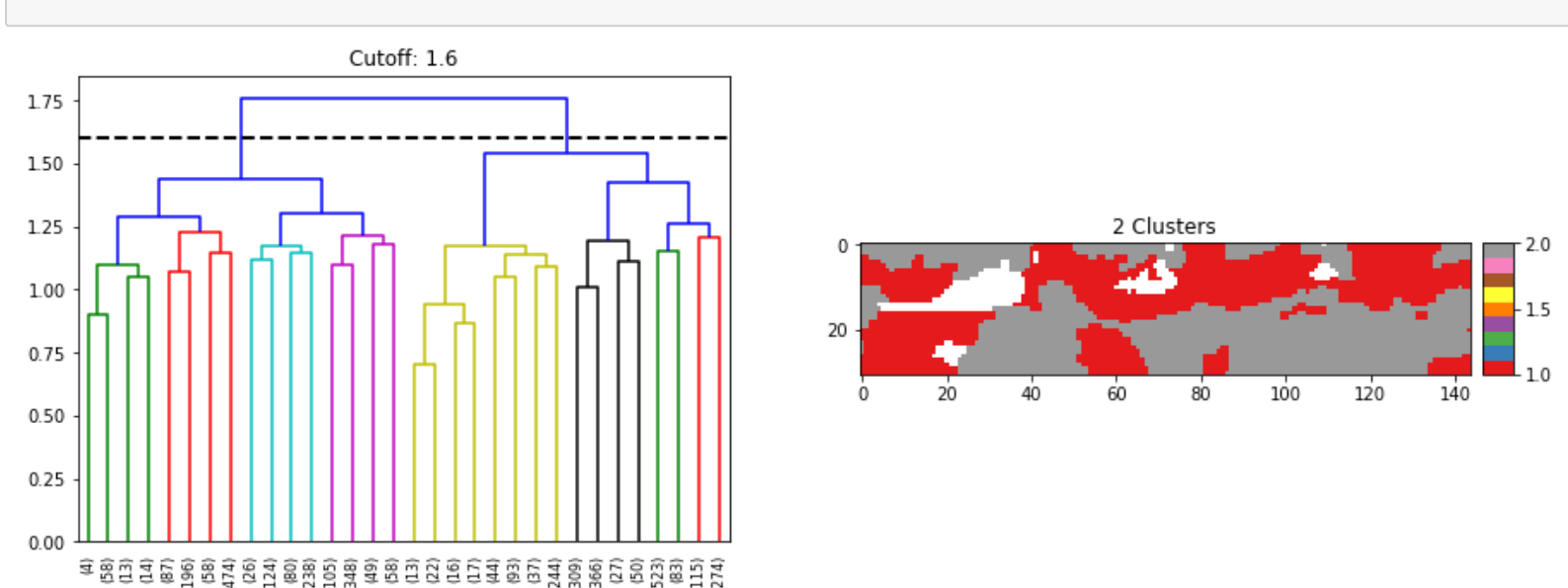
#create a mask, where the flashes are constantly low (their sum will be low as well) mask = np.asarray([np.sum(frd_cut, 2)<=0.03]).reshape((31, 144)) frd_clean = frd_norm_cut[~mask, :] shape = frd_clean.shape print(frd_clean.shape)

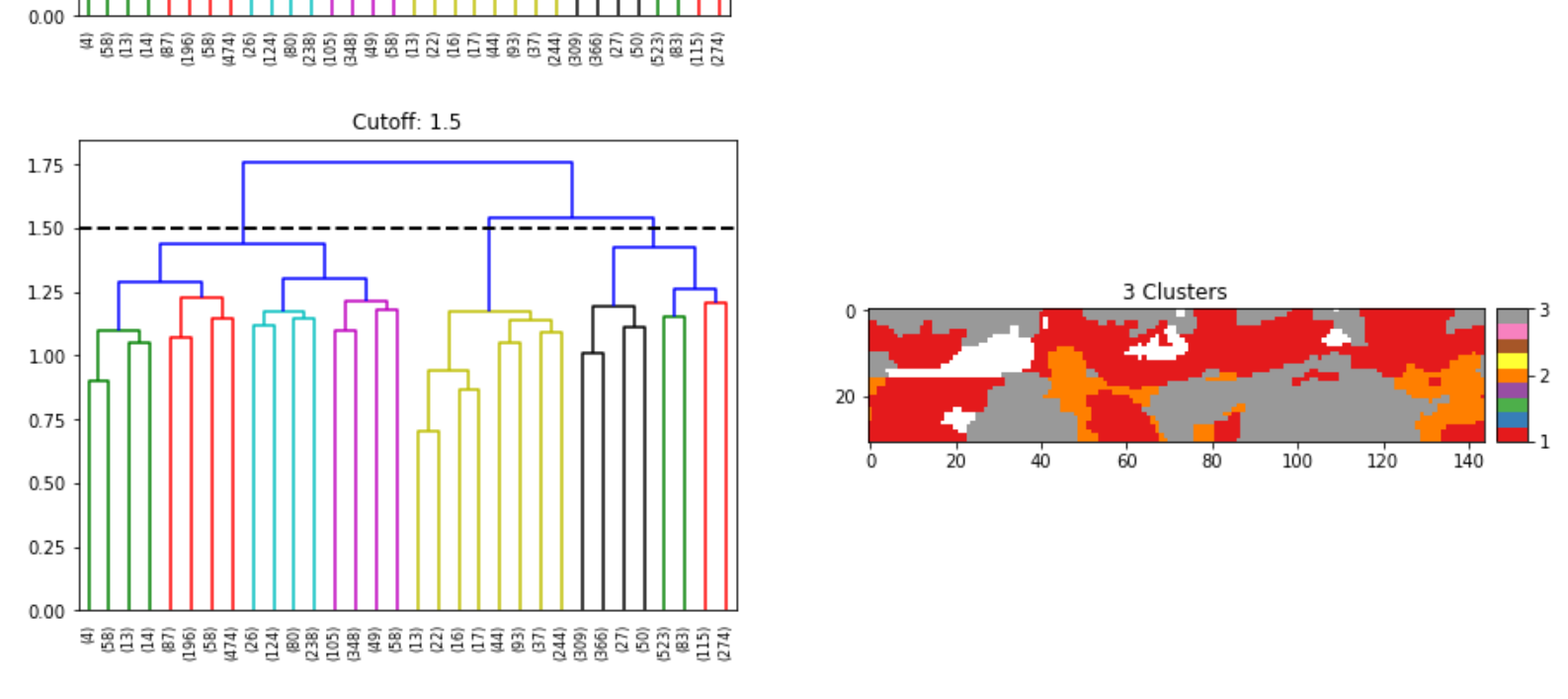
In [12]:
```
#Compute linkage matrices

method = 'complete'
metric = 'correlation'

linkage_file = 'linkage_lis12-%s-%s.npy' % (method, metric)
z = linkage(frd_clean, method = method, metric = metric)
np.save(linkage_file, z)
z.shape
```

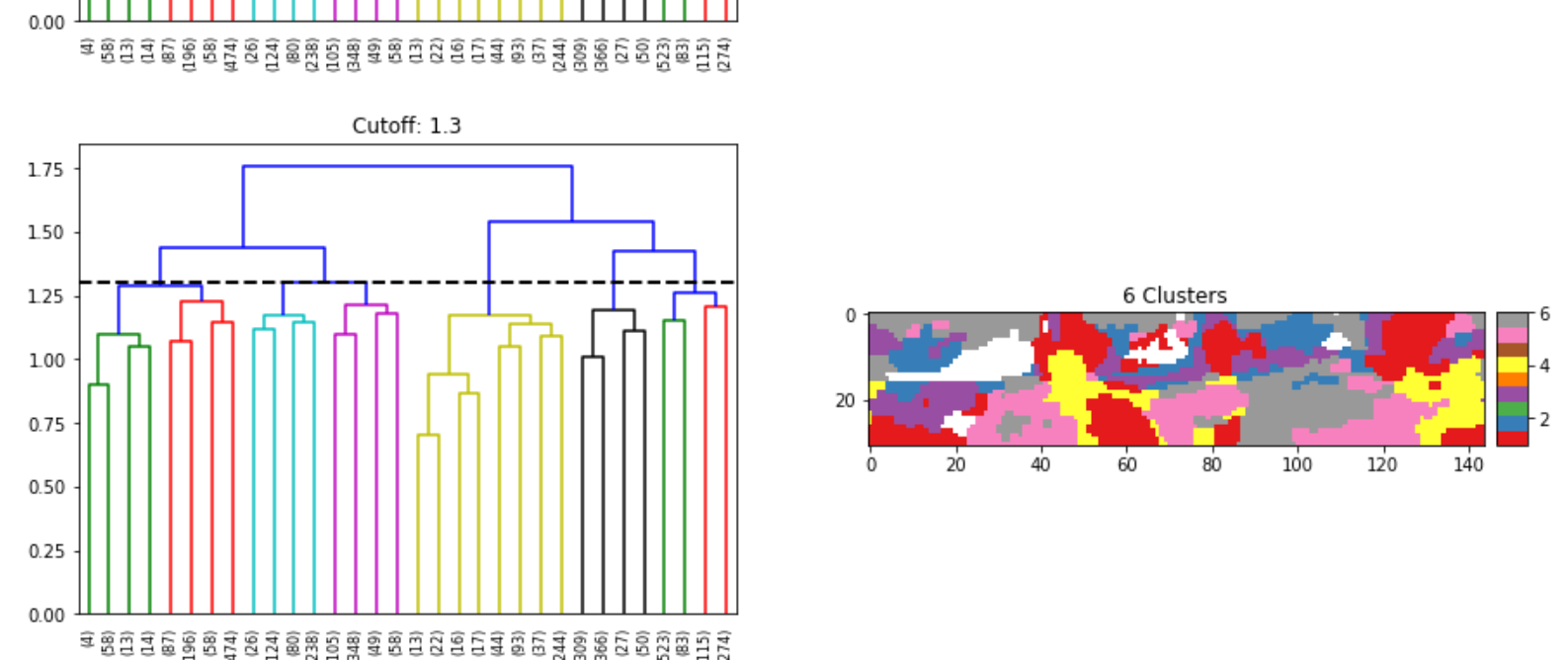
Out[12]: (4164, 4)

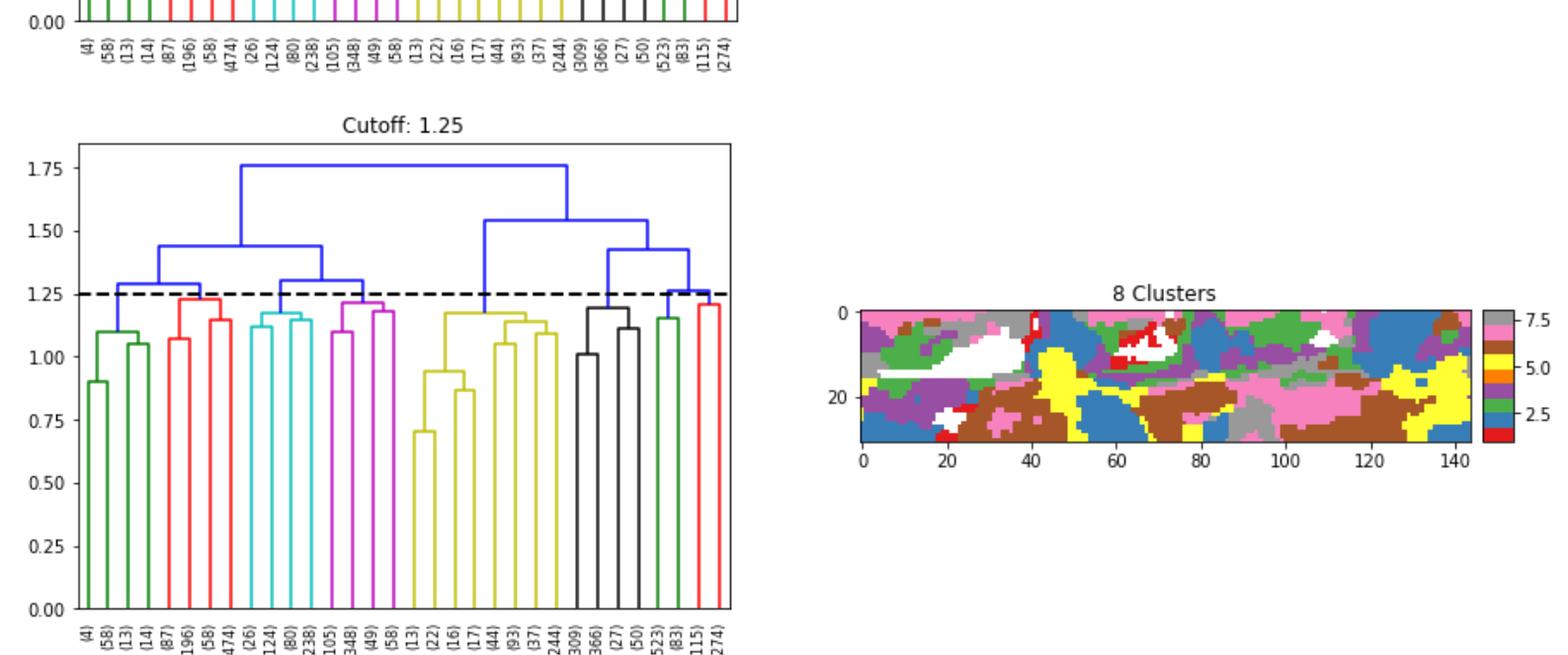
Seasonal Cluster Analysis

In [13]:
```
for cutoff in [1.6, 1.5, 1.3, 1.25]:
    f, (ax,ax2) = plt.subplots(1,2, figsize=(15,5))

    #Plot the dendrogram
    ax.set_title(str('Cutoff: ') + str(cutoff))
    dendrogram(z, ax=ax, p=4, truncate_mode='level',  leaf_rotation=90.)
    ax.plot((0,1000), (cutoff,cutoff), 'k--', linewidth=2)

    #Create the clusters
    clst = fcluster(z, cutoff, criterion='distance') # matrix with shape of input data

    #Create an empty array to hold the clusters -- this since we 'flattened' the ds_clean ar
ray, we need to
    #change the shape of the input
    ci = np.ones(31*144) # shape of original data
    ci.fill(np.nan)
    ci[~mask.ravel()] = clst # everywere were there is no mask I put the cluster value
    ci = ci.reshape(31,144)
    a = ax2.imshow(ci,cmap=plt.cm.Set1)
    divider = make_axes_locatable(ax2)
    cax = divider.append_axes("right", size="5%", pad=0.1)
    plt.colorbar(a, cax=cax)
    num_clusts = str(np.unique(clst).shape[0])
    ax2.set_title(num_clusts + ' Clusters')
```

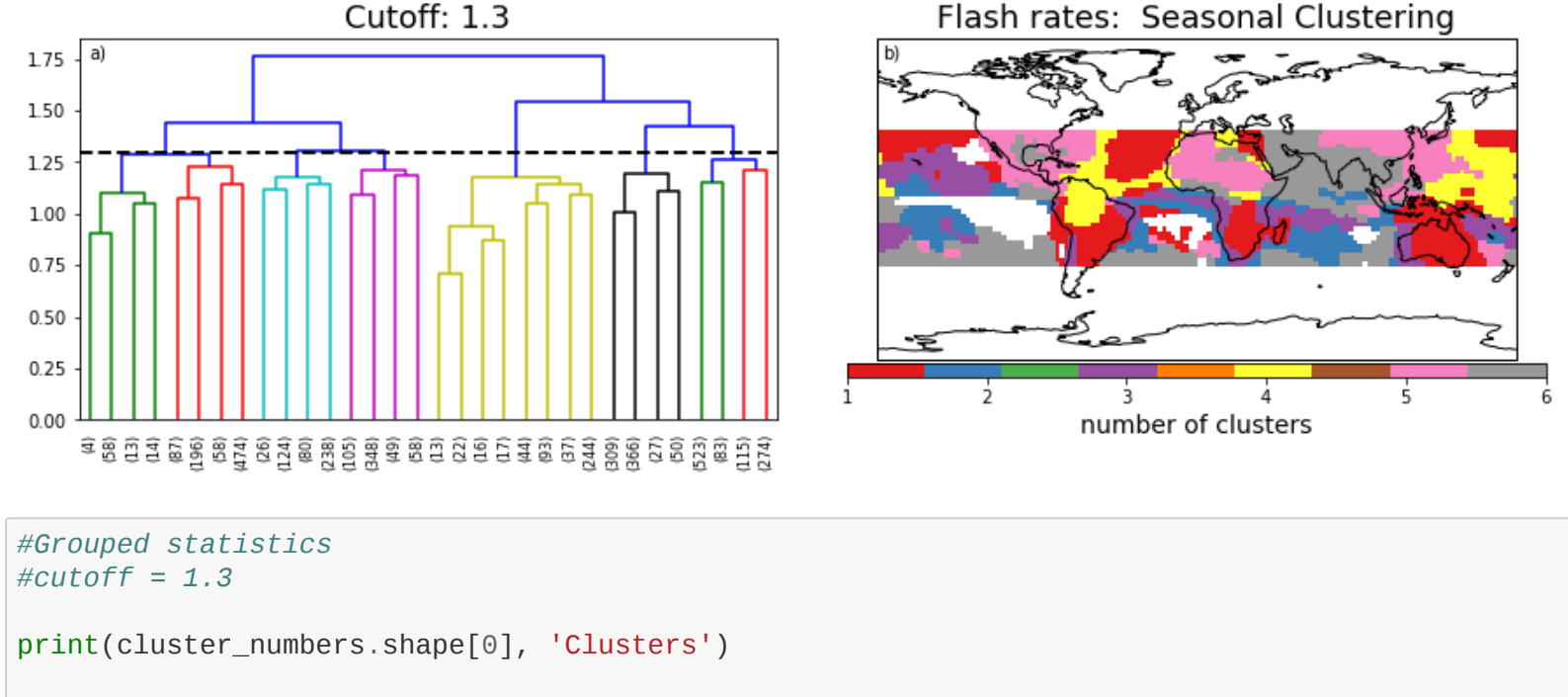








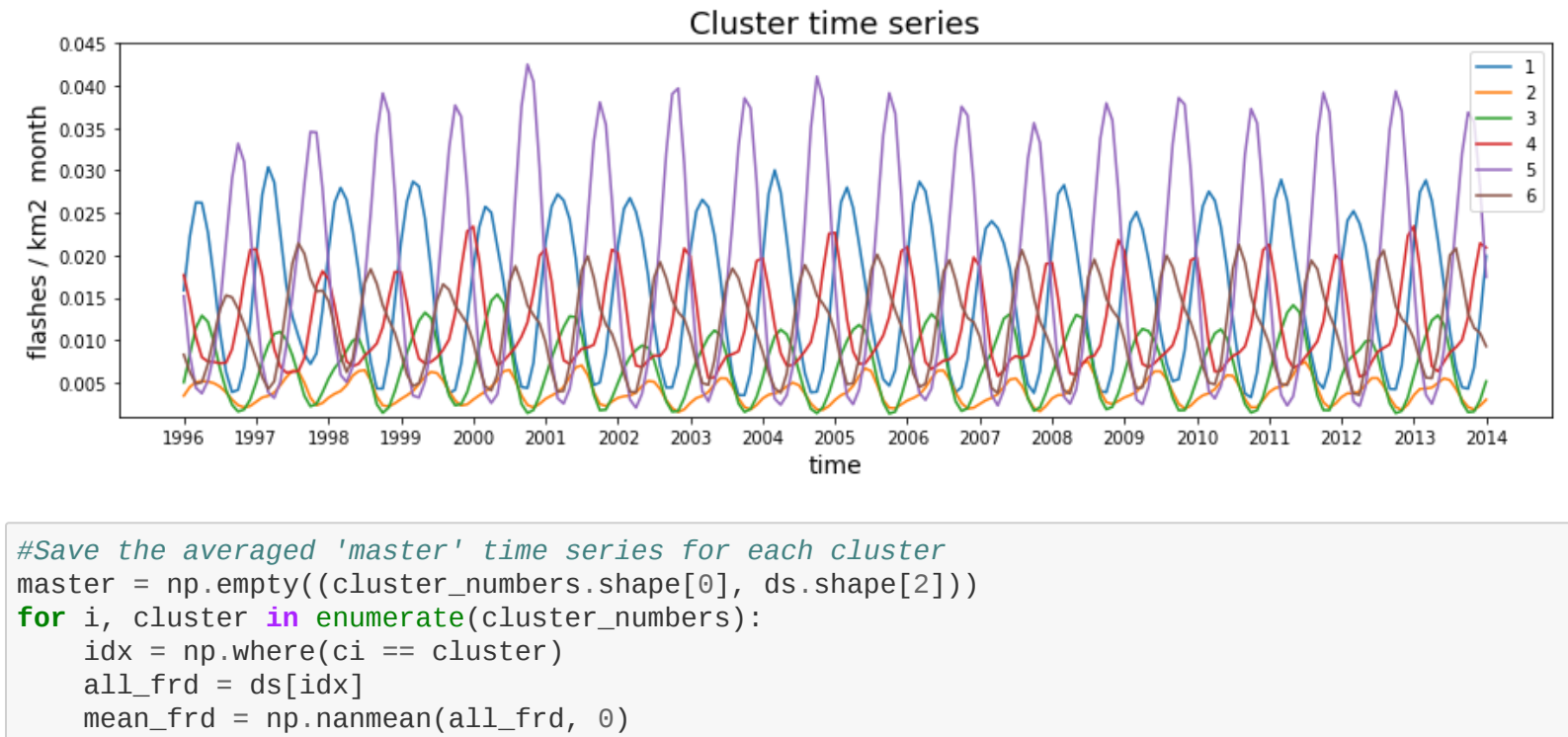
In [17]:
```
cutoff = 1.3

f = plt.figure(figsize=(15,4))


#Plot the dendrogram
ax = f.add_subplot(121)
ax.set_title(str('Cutoff: ') + str(cutoff), fontsize = 18)
dendrogram(z, ax=ax, p=4, truncate_mode='level',  leaf_rotation=90.)
ax.plot((0,1000), (cutoff,cutoff), 'k--', linewidth=2)
plt.figtext(0.13, 0.84, 'a)')


#Create the clusters
clst = fcluster(z, cutoff, criterion='distance')
cluster_numbers = np.unique(clst)
num_clusts = str(cluster_numbers.shape[0])


#Create an empty array to hold the clusters
ci = np.ones(31*144)
ci.fill(np.nan)
ci[~mask.ravel()] = clst
ci = ci.reshape(31,144)


ax = f.add_subplot(122, projection=ccrs.PlateCarree())
ax.set_extent([-180, 180, -90, 90])
ax.coastlines()
img = ax.imshow(ci,cmap=plt.cm.Set1, extent = [-180, 180, -38, 38])
plt.figtext(0.55, 0.84, 'b)')
ax.set_title('Flash rates:  Seasonal Clustering', fontsize=18)

cb = plt.colorbar(img, ax=ax, orientation='horizontal',  pad=0.01, aspect = 50)
cb.ax.set_xlabel('number of clusters', fontsize=14)

plt.subplots_adjust(wspace=0.1)

plt.savefig('Seasonal clustering.png', bbox_inches='tight')

plt.show()
```



In [18]:
```
#Grouped statistics
#cutoff = 1.3

print(cluster_numbers.shape[0], 'Clusters')

means = np.nanmean(frd_cut, axis=2)
means = np.array(means)

#Print the sizes of each cluster
for cluster in cluster_numbers:
    idx = np.where(ci == cluster)
    print(cluster, ':', ci[idx].shape)
    print('Mean of Means:', np.nanmean(means[idx]))
    print('STD of Means:', np.nanstd(means[idx]))

#Means aren't that different
#Deviations are very low
```

```
6 Clusters
1 : (904,)
Mean of Means: 0.01455875
STD of Means: 0.021236744
2 : (468,)
Mean of Means: 0.0036604006
STD of Means: 0.008076196
3 : (560,)
Mean of Means: 0.006363119
STD of Means: 0.0150527125
4 : (486,)
Mean of Means: 0.011166645
STD of Means: 0.020711273
5 : (752,)
Mean of Means: 0.016890865
STD of Means: 0.017375045
6 : (995,)
Mean of Means: 0.010721191
STD of Means: 0.018491905
```

In [19]:
```
#plot cluster time series for the case of 5 clusters

#January 1996 - January 2015
ds = np.array(frd_cut[:,:, 9:226])

plt.close('all')
f,ax = plt.subplots(1, figsize = (15,4))
for cluster in cluster_numbers[:]:
    idx = np.where(ci == cluster)
    all_frd = ds[idx]                              #it takes all the time series within a g
iven cluster
    mean_frd = pd.Series(np.nanmean(all_frd, 0))    #it makes a mean time series
    ax.plot(mean_frd.values, label=str(cluster))
ax.legend(loc='upper right')
ax.set_xticks(np.arange(0, 218, 12))
ax.set_xticklabels(list(np.arange(1996, 2015)))
ax.set_xlabel("time", fontsize=14)
ax.set_ylim(0.001, 0.045)
ax.set_ylabel('flashes / km2  month', fontsize=14)
ax.set_title('Cluster time series', fontsize=18)

plt.savefig('Cluster time series.png', bbox_inches='tight')

plt.show()
```



In [20]:
```
#Save the averaged 'master' time series for each cluster
master = np.empty((cluster_numbers.shape[0], ds.shape[2]))
for i, cluster in enumerate(cluster_numbers):
    idx = np.where(ci == cluster)
    all_frd = ds[idx]
    mean_frd = np.nanmean(all_frd, 0)
    master[i,:] = mean_frd
master = master[:, :]
master.shape
```

Out[20]: (6, 217)

De-seasoning

In [21]:
```
ser = pd.Series(master[4,:])

import statsmodels.api as sm
decomp = sm.tsa.seasonal_decompose(ser, model='additive', period=12)


#linear regression for ht trend
trend = decomp.trend
t = np.arange(len(trend))
mask = ~np.isnan(trend)
x,y = t[mask] , trend[mask]

from scipy import stats
slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)
```

In [22]:
```
plt.close('all')
f, ax = plt.subplots(1, figsize=(15,4))

time = list(np.arange(1996, 2015))

ax.plot(ser.index, ser.values, label = 'observed frd values')
ax.plot(ser.index, decomp.trend, 'r--', label = 'deseasoned trend')
ax.set_xlim(0, 217)
ax.set_xticks(np.arange(0, 217, 12))
ax.set_xticklabels(time)

ax.plot(np.arange(0, 217), t*slope + intercept ,'k--',
        label= 'declination: {} fl/km2 per decade'.format(round(slope*12*10, 5)))
ax.legend(title = 'cluster 5', loc = 'lower right')

ax.set_title('De-seasoning of the flash rates', fontsize=18)
ax.set_ylabel('flashes / km2 month', fontsize=14)
ax.set_xlabel('time', fontsize=14)

plt.savefig('De-seasoning.png', bbox_inches='tight')

plt.show()
```