AO17: Using wind direction data to determine volcanic plume height

Author: Charles Tindal
Supervisors: Dr I. Taylor and Prof. R. Grainger

Abstract

Volcanic plume height is a key metric for forecasting the evolution and climate impact of a volcanic eruption. This study presents a method for estimating plume height which combines wind direction data with sulphur dioxide retrieval data from the Infrared Atmospheric Sounding Interferometer (IASI). The efficacy of this method was evaluated by comparing it against an established retrieval technique, focusing on eruptions of Mount Etna in Italy between 2016 and 2021. The method was found to be effective (88% agreement) for monotonic wind profiles and tight angular plume distributions. In these cases, the technique provides a fast, reliable estimate that can be applied to any instrument, and requires no additional eruption information.

1 Introduction

Volcanic eruptions release large amounts of ash and gas into the atmosphere. These emissions pose multifaceted threats to human activities and the environment. Aviation is a primary concern as ash can cause engine failure and other damage to the aircraft. Silicates in the ash melt inside the engine at temperatures above 1100 °C causing loss of thrust or even flame extinction in the combustion chamber [1]. The abrasive nature of ash can cause enduring engine damage as well as damage to the windscreen, reducing the pilot’s visibility. It is therefore essential that airlines have near real time information on the location, altitude, and density of ash in the atmosphere so they can plan their routes accordingly and minimise the risk to life as well as the financial impact [2]. It is estimated that the 2010 Eyjafjallajökull eruption had a $5 billion impact on global GDP due to commercial airspace being shut down over a large part of Northern Europe [3]. Plume height is a key metric for use in dispersion models that provide airlines with accurate forecasts.

Beyond aviation, volcanic ash presents a range of hazards, including damage to infrastructure and machinery, contaminating water supplies, and health risks. This is particularly important for volcanoes located near populous areas such as Mt Etna, which has over one million people living within 30 km. Eruptions with large plume heights can also have far-reaching climatic consequences. Sulphur dioxide emitted into the stratosphere can generate sulphate aerosols which increase the Earth's albedo and induce cooling effects [4]. Additionally, substantial carbon dioxide emissions contribute to the greenhouse effect and thus to global warming. Underwater volcanic eruptions, like the 2022 Hunga Tonga event, can inject large amounts of water vapour into the stratosphere, causing further warming [5].

Plume height is a crucial variable in predicting the evolution and lifetime of ash and gas from volcanic eruptions as well as in predicting the impact of an eruption on global climate. Currently plume height is measured from ground, aircraft, or satellite-based observations. A common issue with ground or aircraft observations is that not all eruptions occur at a time or location that can be directly observed. Satellite observations offer the advantage of global coverage; however, many of the height estimation techniques require additional data such as eruption time or are computationally intense. It is therefore important to have an efficient, straightforward method to obtain an early estimate of plume height using a single satellite image of the volcanic plume and without knowledge of the eruption time.

This project aims to combine satellite observations of sulphur dioxide (SO_2) plumes with wind direction data to estimate the plume heights of eruptions at Mount Etna from 2016 to 2021. Etna is used as it was very active during this period, and the observation data is widely available. SO_2 can, with care, be used as a proxy for ash [6]. In this case we are using SO_2 because the height data is easily available, however, the method could equally be applied to ash. The work investigates the conditions on wind profile and plume distribution for which the method is effective. The retrieval techniques and wind profile method are presented in section 2. In section 3 the results are discussed, and a quality control is introduced. The study's conclusions are detailed in section 4.

2 Method

2.1 Instrument

The Infrared Atmospheric Sounding Interferometer (IASI) is a Fourier transform spectrometer carried on the MetOp -A, -B, and -C satellites, launched in 2006, 2012, and 2018 respectively. The satellites are in Sun-synchronous polar orbits meaning they each offer near global coverage every 12 hours, crossing the equator at 9:30 mean local time. IASI is a nadir-viewing instrument with a swath width of 2200 km made up of 30 steps. Each step contains four circular pixels of 12 km radius. IASI has a high spectral resolution of 0.50 cm^{-1} (apodised) in the infrared band between 3.4-15.5 µm. This makes it well suited to resolving the ν_1, ν_3, and $\nu_1 + \nu_3$ absorption features of SO_2, which are centred at 8.7, 7.3, and 4.0 µm respectively. Imaging in the infrared allows measurements to be made through the night, and during high altitude winters [7].

2.2 Linear Retrieval of SO_2

This study uses a linear retrieval technique from Walker et al. [8] to flag pixels in which there are elevated amounts of SO_2. The method uses a covariance matrix formed from SO_2 free pixels (over the Northern Atlantic and Europe in 2009) incorporating channels in the ν_3 absorption band. This matrix holds information about the spectral variability of SO_2 free pixels due to other parameters such as water vapour concentration and the atmospheric temperature profile. The spectra of pixels containing SO_2 are then easily distinguished, and the column amount can be calculated assuming a uniform distribution of SO_2 up to 20 km altitude. The technique also makes the first order assumption that the amount of SO_2 is directly proportional to the spectral deviation from the background. Sensitivity to low altitude SO_2 is reduced if there are high levels of water vapour (such as in the tropics) due to its strong absorption in the ν_3 band. This issue is mitigated when looking at Etna due to its summit height of 3.3 km. The detection threshold for plumes at 4-6 km is 1.3 Dobson Units (DU). This threshold decreases with altitude to a value of 0.33 DU at 11-14 km.

2.3 Iterative Retrieval of SO_2

The iterative retrieval (IR) technique is applied to pixels which have been flagged by the linear retrieval method. This technique uses the European Centre for Medium-Range Weather Forecasts (ECMWF) meteorological data for vertical temperature, pressure, and water vapour profiles. This data is then incorporated into the RTTOV fast radiative transfer algorithm to forward model top of atmosphere radiances in the ν_1 and ν_3 bands. The RTTOV model is run iteratively while varying the state vector components (column amount, height, thickness, and surface temperature) with the aim of minimising the cost function. The cost function describes the fit between the measured and modelled spectra. The method outputs the column height (in DU) and altitude (in hPa and converted to km) of the SO_2 plume assuming a Gaussian distribution of thickness and a clear (cloud free) sky.

The method produces an error covariance matrix associated with the retrieval and a quality control for each pixel. The quality control requires that the retrieval has converged, that the retrieved column amount is positive, and that the retrieved pressure is between 0 and 1100 hPa. However, if the retrieval converges at a local minimum, it may produce an erroneous result that is not filtered by the quality control. For the same reasons as the linear retrieval, the IR technique has larger errors at low altitudes and for low concentrations of SO_2. It may also underestimate the column amount if there is thick cloud or ash above the SO_2 plume [9,10].

2.4 Height Measurement by Wind Profile Method

For a given IASI orbit over the region surrounding Etna, the coordinates of flagged pixels from the linear retrieval within 500 km of Etna were plotted on a map. An algorithm was used to calculate the mean bearing of the flagged pixels with respect to Etna. Any flagged pixels which had a bearing within 10 ° of the mean were identified as being part of the plume. The number of pixels in the plume was counted, and the standard deviation of angle was taken.

Assuming the mean bearing to be the direction of travel of the SO_2 plume, the corresponding plume height(s) were found using the ECMWF ERA5 wind profile (wind direction with height). The wind profile was constructed by interpolating the u and v vector components to Etna's location and then computing the wind direction and speed. The time of wind profile used was the nearest hour to when the plume was flagged by the IASI linear retrieval. This choice was based on the first order assumption that the wind direction is constant throughout the plume's propagation from emission to measurement. Examples of the height measurement are shown graphically in Figures 1 and 2.

In the case that the mean plume angle intersected the wind profile more than once (such as in Figure 2), the following 'bin method' was used to find the best fitting height:

1) Each flagged pixel's bearing was plotted against the wind profile and the corresponding heights from the wind profile were tabulated.
2) For each candidate plume height, the pixel heights calculated in Step 1 that fell within 1 km of this height were counted.
3) The number of counts for each plume height was then converted into a probability.
4) The candidate height with the highest probability was then selected as the measured height of the SO_2 plume.

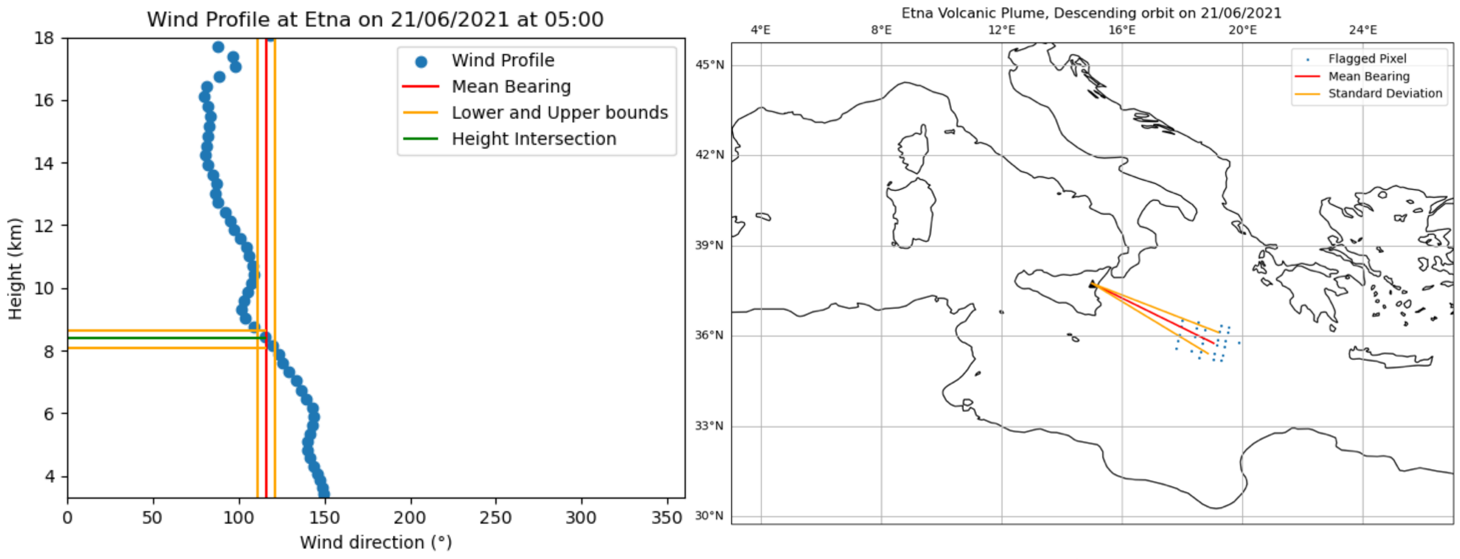
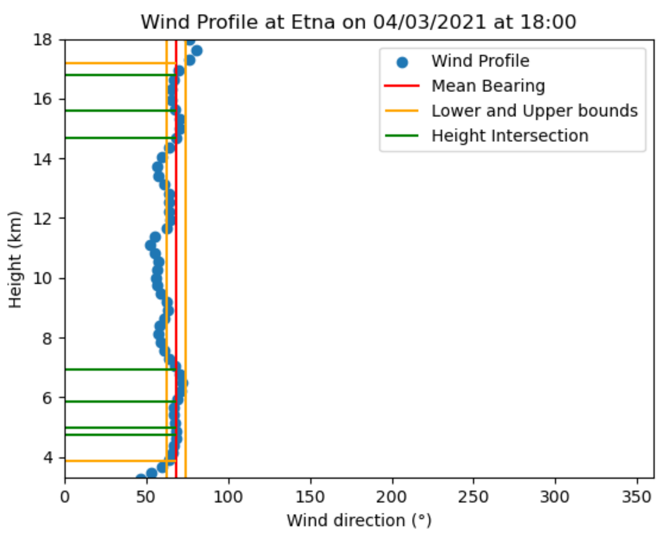


Figure 1: Plume map and wind profile plot for single intersection case on 21st June 2021



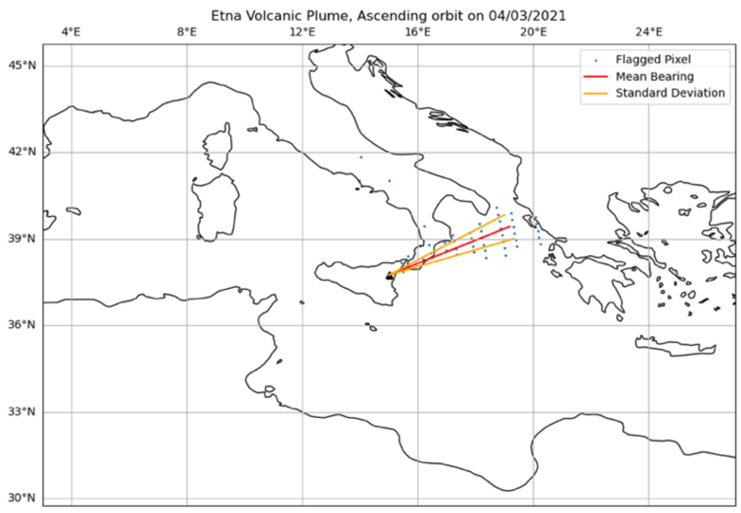


Figure 2: Plume map and wind profile plot for multiple intersection case on 4th March 2021

Upper and lower bounds on the height were also calculated by varying the propagation angle within the standard deviation and noting the maximum and minimum heights at which there was an intersection with the wind profile. The linear retrieval takes just a few seconds per pixel and the wind profile (WP) method takes around 15 seconds to run, allowing plume heights to be calculated in near real time.

This method was run for the ascending and descending orbits for each day in which a plume was detected (5 or more pixels in the plume) using IASI data from MetOp-A during a period from 2016-2021. The resulting WP heights were then compared with the IR heights. The 2021 results were also compared against heights from the Global Volcanism Program (GVP) bulletin report. GVP bulletin reports for Etna are compiled from weekly and special reports by the Osservatorio Etneo, the regional branch of Italy's national volcano institute. These reports contain plume height measurements which use data from two ground-based visible cameras as well as the Spinning Enhanced Visible and Infrared Imager (SEVIRI) based on the Meteosat Second Generation satellites to estimate plume height with an uncertainty of 500 m [11,12]. The report can lack detail and often doesn't specify whether the height measurement has used remote sensing, ground observation, or a combination of both. Since heights obtained from ground-based cameras are limited by viewing angle to a maximum altitude of 9 km, some of the heights given as 9 km are likely to be underestimates [13].

3 Wind Method Results

3.1 Number of Intersections

Out of a total of 4200 (2100x2) orbits between 1st Jan 2016 and 30th Sept 2021, a plume was identified in 247 cases. A height was found in 188 (76%) of these orbits meaning there were 59 orbits for which the plume angle did not intersect the wind profile. The number of intersections in each measurement is plotted in Figure 3 which shows that just 60/188 (32%) heights were single intersection cases for which a distinguishing method was not needed.

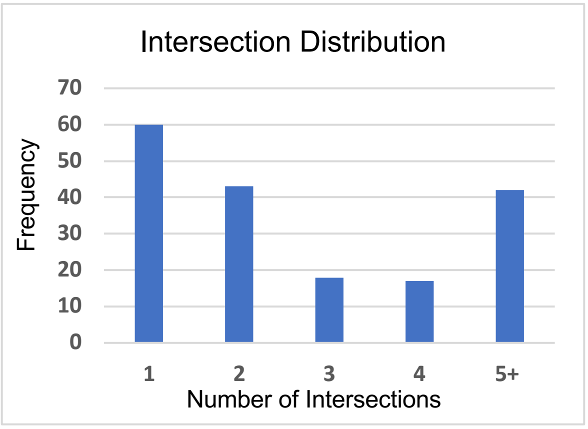


Figure 3: Bar graph showing the relative frequencies of the number of intersections of plume angle with wind profile for all height measurements from 2016 to 2021

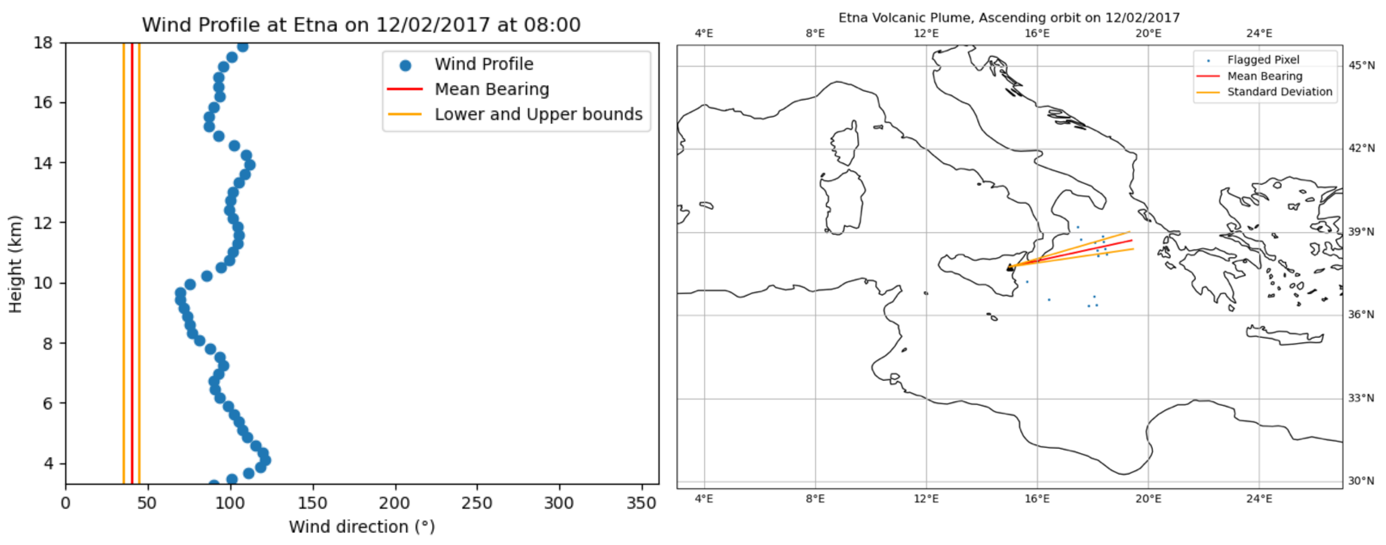


Figure 4: Plume map and wind profile plot for a case in which there is no intersection on 12th Feb 2017

3.2 Example Cases and Limitations

The angle algorithm appeared to correctly identify the direction of propagation in most cases. Examples of this are Figures 1 and 2 which identify the SO_2 plumes on 21st June and 4th March to be positioned with respect to Etna at bearings of 116±5° and 68±6° respectively. On 21st June there was only a single intersection at 8.4±0.3 km. However, on 4th March the plume angle intersected the wind profile 7 times at heights ranging from 4.8 km to 16.8 km. The bin method was then used to select the height as 6.9 km. The errors on this height were large due to the shape of the wind profile which gave a lower bound of 3.8 km and an upper bound of 17.2 km.

There were, however, instances in which the angle calculation was limited in its accuracy. Figure 4 shows an example of a plume angle for which the height could not be found as there was no intersection with the wind profile. It appears from the map that there are two separate plumes in the image. The algorithm has broken down as it has found the angle of one of the plumes which does not intersect with the wind profile. The same issue occurs for other plume shapes with a large spread of angles. Figure 5 highlights two other events for which the plume's direction of travel is

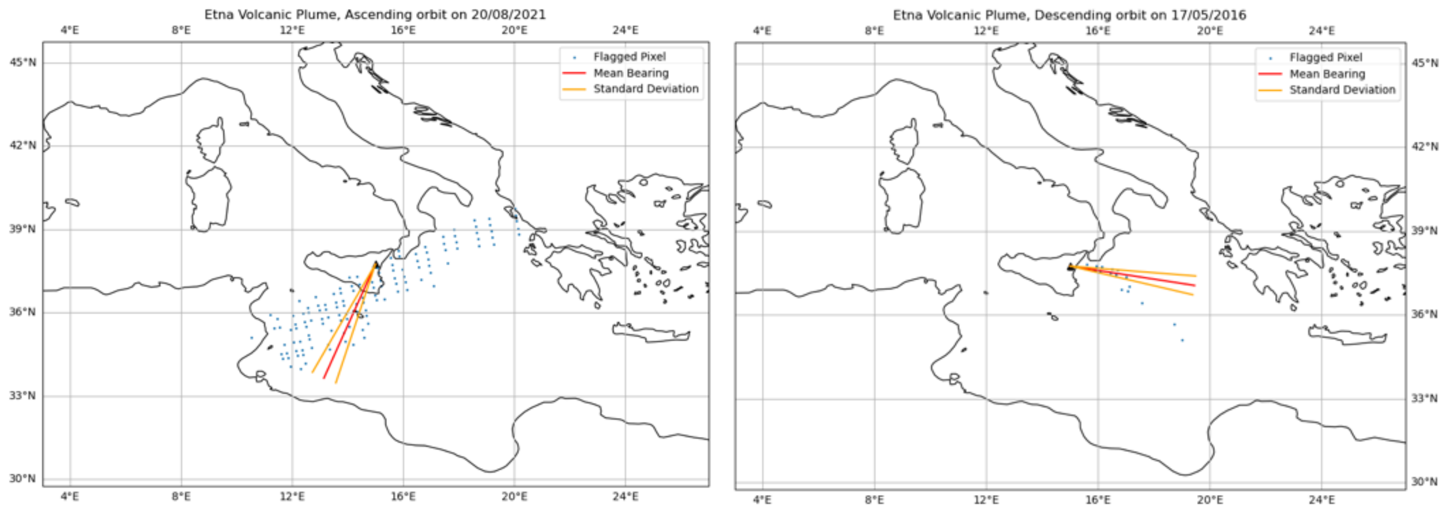

Etna Volcanic Plume, Ascending orbit on 20/08/2021
Etna Volcanic Plume, Descending orbit on 17/05/2016
Flagged Pixel
Mean Bearing
Standard Deviation

Figure 5: Plume maps in which direction is unclear. On the 20th August 2021 (left), there is a large plume which extends across Etna to the NE and SW. On the 17th May 2016 (right) the plume appears to change direction.

Limitation	Effect	Mitigation (if applied)
Multiple plumes or large spread of SO_2	Plume angle difficult to measure, large errors	Error on angle included in calculation
No Intersection	No height found	
Multiple Intersections	Height found by bin method may be incorrect	All other possible heights are tabulated
Very few flagged pixels	Plume angle may be incorrect, SO_2 may be incorrectly flagged (noise)	Minimum pixel number applied (5)
Wind profile changing with time and/or location	Measured height may be incorrect	Maximum distance of pixel applied (500 km)

Table 1: A Summary of the WP method's limitations, their effects, and how they have been mitigated.

unclear. Since the method has been tested on a large dataset, results are automated and therefore such cases have not been eliminated from the study, however, they could easily be identified by eye and discounted if the method were to be used for a particular eruption.

Another cause of error in the WP method is the variation of wind profile with time and location. As described in the methods section, the wind profile used is at the closest hour to when the pixels are imaged by IASI, interpolated to Etna. The plume may have been propagating for up to 12 hours from eruption by the time it is first imaged by IASI. During this time, and over this distance, the wind direction at the altitude of the plume may have changed somewhat. This can lead to there being no intersections of the plume angle with the wind profile, or to an erroneous height. Since the method is independent of eruption time, a control cannot be placed on the maximum time since eruption, however, the maximum distance of the plume from Etna can be limited. In this study this distance is 500 km, however, this could be reduced going forward to improve results. A summary of the limitations of the WP technique are given in Table 1.

3.3 Comparison with Iterative Retrieval Heights

Figure 6 shows the measured plume heights plotted during an eruptive episode beginning in February 2021. Dates such as 19th Feb and 22nd Feb display a good match when comparing the WP height to the IR height with the measured plumes differing by 0.55 km and 0.10 km respectively, agreeing well within errors. Both results

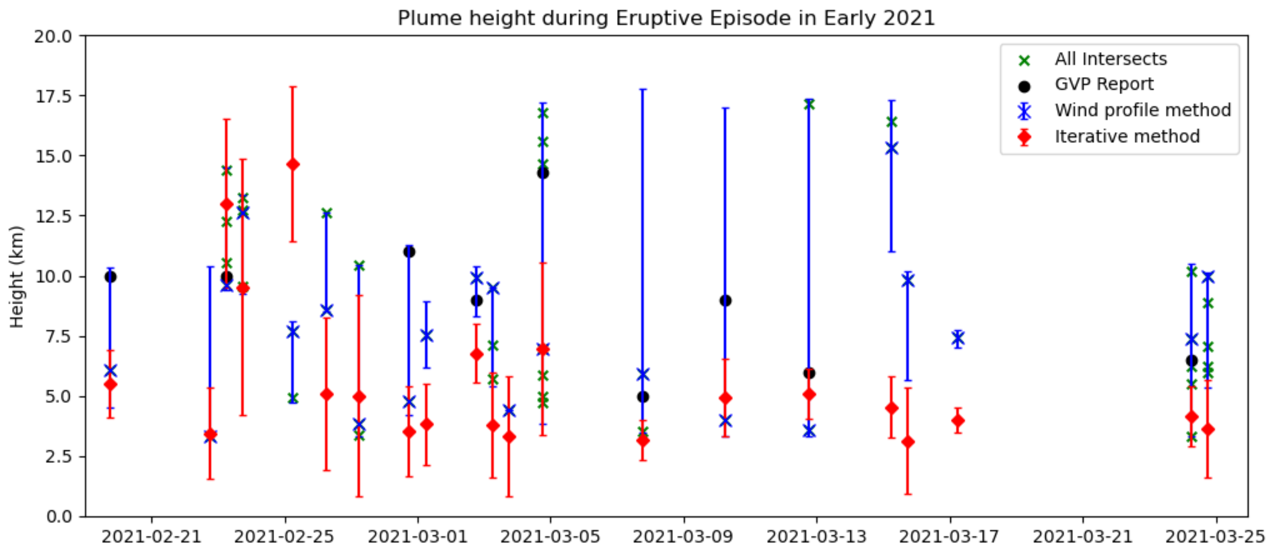


Figure 6: Timeseries comparing WP heights with GVP and IR heights during Feb and March 2021

Min. Plume Size (pixels)	5	10	15	20	30
% WPM heights within IRM errors	55%	60%	61%	68%	88%
% WPM heights within 1.5 km IRM	37%	43%	50%	55%	50%

Table 2: Comparison of WP and IR heights for all events between 2016 and 2021

Min. Plume Size (pixels)	5	10	15	20	30
% WPM heights within IRM errors	63%	75%	89%	83%	100%
% WPM heights within 1.5 km IRM	39%	50%	89%	83%	88%

Table 3: Comparison of WP and IR heights for single intersection events between 2016 and 2021

GVP height within WPM bounds	GVP height within 1.5 km WPM height	GVP height within IRM errors	GVP height within 1.5 km IRM height
12 (92%)	7 (54%)	5 (38%)	3 (23%)

Table 4: Comparison of GVP heights against WP and IR heights (for 13 events in 2021)

were cases in which the plume angle only intersected the wind profile once. Other dates such as 26th Feb and 17th March show a significant difference between WP and IR measured heights. On 28th Feb and 5th March the bin method appeared to correctly select the heights from the multiple intersection heights since the nearest height to the IR height was chosen. However, on 7th March the wrong height intersection is selected with respect to the IR.

Tables 2 and 3 show the agreement between WP and IR heights for events between 2016 and 2021. The threshold of 1.5 km is used as the measure of agreement since it is roughly the typical thickness of a volcanic plume. These results show good agreement between methods given the limitations that exist with both techniques. The agreement is improved by increasing the minimum plume size and limiting the number of intersections. These quality controls will be discussed in Section 3.5.

3.4 Comparison with Global Volcanism Report

GVP heights were compared with WP and IR heights for 13 events during 2021. Nine of these are plotted in Figure 6 during the Feb-March eruptive episode. Table 4 shows that the agreement between GVP heights and WP heights was significantly better than with IR heights (54% within 1.5 km compared with 23%).

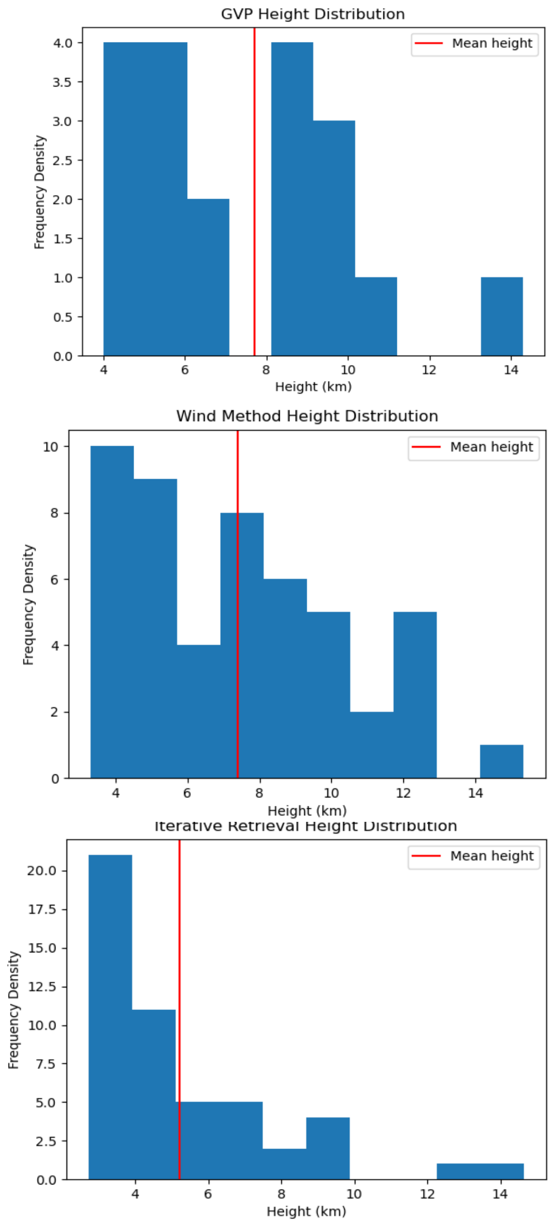


Figure 7: Comparing plume height distribution of different methods (for plumes consisting of at least 10 pixels)

To investigate the difference in the height methods further, height distributions were plotted for each method in Figure 7. This shows that IR heights tended to be significantly lower than both WP and GVP heights with a mean plume height of 5.1 km compared with 7.3 km and 7.8 km respectively. A possible explanation for the discrepancy between GVP and IR height distribution is the small sample size of GVP data. It also could be that the GVP bulletin only reports on significant plumes, skewing the mean plume height. However, the apparent agreement of the GVP and WP heights and distributions would suggest that the IR may not be producing reliable height measurements for SO_2 emissions at Etna as it seems to be underestimating the plume height. This therefore effects the overall validation of the WP method. Comparison with other methods of height measurement is therefore recommended to further test the accuracy and reliability of the WP technique.

3.5 Developing a Quality Control

Tables 2 and 3 demonstrate that increasing the minimum plume size and limiting the number of intersections leads to an improved agreement between methods and reduces the number of erroneous results. In this section we develop a quality control that can be used to determine the plume shapes and wind profiles for which the WP technique is applicable.

3.5.1 Minimum Plume Size

To include as many events as possible, the minimum pixel number in the plume was initially set to 5. Table 2 shows that increasing the minimum plume size from 5 pixels to 30 pixels increases the likelihood of the WP height agreeing with the IR height from 55% to 88%. This trend is even clearer in the single intersection case (see Table 3) which rises from 63% agreement to 100% (8/8) agreement. Furthermore, plotting the Pearson correlation coefficient between WP and IR heights against minimum plume size (Figure 8) shows a similar positive trend with a large jump in correlation around at 20 pixels. Therefore, the recommendation of the minimum plume size for which the method should be applied is 20 pixels.

These results show clearly that minimum plume size is an important metric for determining the accuracy of the WP method. This could be due to many factors. Increasing the minimum pixel threshold reduces the likelihood of the plume being falsely detected due to anomalous surface temperatures or gas concentrations. Having more pixels also improves the accuracy of the plume angle algorithm as there are more location coordinates available. It is possible that the IR height estimates also improve with the number of pixels since a larger plume may mean higher concentrations of SO_2. In addition, there would be more pixels from which to take the average, hence reducing the sampling error.

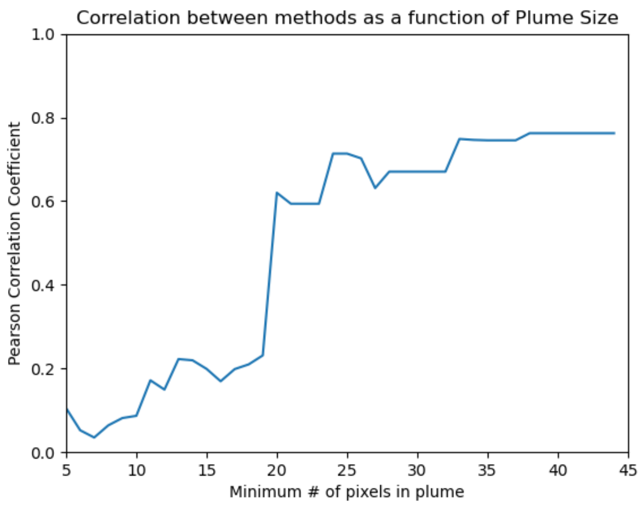


Figure 8: Plot showing how correlation between WP and IR heights improves as minimum plume size increases

3.5.2 Wind Profile Control

A limitation on the WP technique is that in many cases (such as Figure 2), the difference between upper and lower bounds is so great the result is not useful. In these cases, the plume angle often intersects the wind profile multiple times at different heights. Often these heights cannot be reliably distinguished by the bin method without additional information. Therefore, for the WP technique to be accurate, certain conditions on the plume and wind profile are required. Limiting the wind profile to only cases in which there is a single intersection greatly improves the accuracy of the height measurements, however, single intersection wind profiles only occur around 32% of the time and hence cannot be relied upon.

An alternative indicator of a good wind profile and small angular plume spread is the difference between lower and upper bounds on the WP height. A small difference in bounds suggests a monotonic, shallow gradient wind profile and a tight angular plume distribution such as that of Figure 1. It was found that when minimum plume size was set to 10 pixels and the upper and lower bounds differ by 2 km or less, the WP method agreed with IR heights in 70% of cases, compared with 60% when removing the limits on bounds. This agreement increases further when limiting plume size and eliminating irregular plume shapes by eye.

4 Conclusions

This work has developed a method to estimate volcanic plume height using wind direction and IASI linear retrieval data. The method was tested against the Carboni et al. (2012,2016) iterative retrieval technique for eruptions of Etna between 2016 and 2021.

The WP method's simplicity and speed of algorithm make it well suited to measuring plume height from volcanoes across the globe in near real time. However, the study found that the technique produces a good estimate of SO_2 plume height only when specific conditions on the plume and the wind profile are met. Firstly, it requires that the plume is significant (at least 20 pixels) and has a tight angular spread (within 10°). Secondly, the wind direction must vary sufficiently with height such that the bounds on the plume height are close (within 2 km). Finally, the method requires that the wind direction does not change significantly over the time during which the plume has propagated. When these conditions were met, the study found that the WP heights of SO_2 plumes from Etna between 2016 and 2021 showed a good correlation with IR heights and agreed within errors in 68% of cases. This result was improved to 83% when removing the cases in which there were multiple intersections with the wind profile.

To be able to distinguish heights effectively in the case with many intersections, more information on the plume and/or wind profile would be required. A possible future amendment would be to utilise successive IASI images of the same plume from multiple MetOp satellites to infer the speed of the plume. Alternatively, the plume speed could be calculated using the eruption time. Comparing this speed to the wind profile (speed plotted against altitude) could then allow the true plume height to be distinguished. However, using successive images of the plume or an eruption time would infringe on a primary aim of the method which was to work with only a single image and no additional eruption information.

To improve the method, the plume angle algorithm could be developed to take the column amount of each pixel into account as well as the distance from the point of emission. The bearing could then be weighted to favour pixels which are closer to the eruption site and have larger amounts of SO_2. Further work could also be done to qualitatively assess which plume shapes and plume distributions the method will produce erroneous results for. The study should be extended to more volcanoes to verify that it is effective at measuring plume height for different types of eruptions in other locations. In addition, since the technique is very general, it could be extended to other instruments and to ash, allowing it to be tested in a variety of scenarios. Finally, it would be beneficial to test the WP method against other plume height measurement methods to improve the reliability of the results.

References

[1] Song, W., Lavallée, Y., Hess, KU. *et al.* (2016). Volcanic ash melting under conditions relevant to ash turbine interactions. *Nat Commun* 7, 10795.

[2] Lechner et al. (2017) Volcanic Ash and Aviation – The Challenges of Real-Time, Global Communication of a Natural Hazard, pp. 1–14, Advances in Volcanology, Springer, Berlin, Heidelberg

[3] Oxford Economics (2010), The Economic Impacts of Air Travel Restrictions Due to Volcanic Ash, https://controverses.minesparis.psl.eu/public/promo10/promo10_G11/data/documents/Volcanic-Update.pdf

[4] Rollins, A. W. et al. (2017). The role of sulfur dioxide in stratospheric aerosol formation evaluated by using in situ measurements in the tropical lower stratosphere. *Geophys. Res. Lett.*, 44, 4280–4286

[5] Jenkins, S., Smith, C., Allen, M. et al. (2023). Tonga eruption increases chance of temporary surface temperature anomaly above 1.5 °C. *Nat. Clim. Chang.* 13, 127–129

[6] Thomas, H. E. and Prata, A. J. (2011) Sulphur dioxide as a volcanic ash proxy during the April–May 2010 eruption of Eyjafjallajökull Volcano, Iceland. *Atmos. Chem. Phys.*, 11, 6871–6880

[7] Taylor, I et al. (2023). A satellite chronology of plumes from the April 2021 eruption of La Soufrière, St Vincent. *Atmos. Chem. Phys.*, 23, 15209–15234

[8] Walker et al. (2012). Improved detection of sulphur dioxide in volcanic plumes using satellite-based hyperspectral infrared measurements: Application to the Eyjafjallajökull 2010 eruption, J. Geophys. Res., 117, D00U16

[9] Carboni et al. (2012) A new scheme for sulphur dioxide retrieval from IASI measurements: application to the Eyjafjallajökull eruption of April and May 2010, *ACP*., 12, 11417–11434

[10] Carboni et al. (2016) The vertical distribution of volcanic SO_2 plumes measured by IASI, *Atmos. Chem. Phys.*, 16, 4343–4367

[11] De Angelis, S., Zuccarello, L., Scollo, S. et al. (2023) Assessment of eruption source parameters using infrasound and plume modelling: a case study from the 2021 eruption of Mt. Etna, Italy. *Sci Rep* 13, 19857

[12] Scollo et al. (2019). Near-Real-Time Tephra Fallout Assessment at Mt. Etna, Italy. *Remote Sens. 11*, 2987

[13] Scollo et al. (2014). Eruption Column Height Estimation: the 2011-2013 Etna lava fountains. Annali di geofisica. 57

Appendix A: Source Code

```
import scipy.io as sio
import scipy.stats as sst
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import glob
from datetime import datetime
import pytz

def
time_since_2000(year,month,day,hour,min,region):
# Input using local time
    epoch_time = datetime(2000, 1, 1)
    if region=='etna':
        if int(month) in [1, 2, 3, 11, 12]:  # CET
time
            offset = 1
        if int(month) in [4, 5, 6, 7, 8, 9, 10]:
# CEST time
            offset = 2
    else:
        print('Not Etna')
        offset = 0
    event_time = datetime(int(year), int(month),
int(day), hour + offset, min)
    dt = event_time - epoch_time
    time_2000 = dt.total_seconds()
    return time_2000  # Output is total seconds
from 2000 UTC to event

# This function imports the wind profile
def wind_profile(year,month,day,hour,region):
    profile =
glob.glob("/network/group/aopp/eodg/RGG008_GRAINGER
_IASIVOLC/taylor/ecmwf_profs/{}/{}/{}/ECprofs_{}{}{
}_{}.sav".format(region, year, month, year, month,
day, hour))
    p = sio.readsav(profile[0])
    height = p.wind_profs.z[0]  # Height in km
    wd = p.wind_profs.wd[0]  # Wind direction (deg)
    ws = p.wind_profs.ws[0]  # Wind speed in m/s
    return height,wd,ws

# This function reads in Linear retrieval data for
a given IASI orbit
def read_in_so2(year,month,day,ascend,flag):
    files = glob.glob(
"/gf4/eodg/RGG008_GRAINGER_IASIVOLC/taylor/iasi/so2
_out/standard-globe_v3-1/lin/{}/{}/standard-
globe_v3-1_lr_iasi-a_{}{}{}_????.sav".format(
            year, month, year, month, day))

    # Select ascending or descending
    if ascend == 1:
        marker = 'A'
    else:
        marker = 'D'
    long = np.zeros(0)
    lati = np.zeros(0)
    pixel_time = np.zeros(0)

    # Read in all files for specific day
    for i in range(len(files)):

        file = files[i]
        s = sio.readsav(file)

        latitude = s.lr_so2.lats[0]
        longitude = s.lr_so2.lons[0]
        index = s.lr_so2.v3_idx[0]
        direction = s.lr_so2.direction[0]
        pix_time = s.lr_so2.time[0]
        if np.isscalar(index) and flag == 1:
# Highlights instance in which no pixels are
flagged
            continue
        lon = np.zeros(0)
        lat = np.zeros(0)
        pix_t = np.zeros(0)

        if flag == 1:  # choosing only flagged
pixels
            longitude = longitude[index]
            latitude = latitude[index]
            direction = direction[index]
            pix_time = pix_time[index]

        for j in range(len(direction)):  # For a
given file, compiles data for ascending/descending
            if marker in str(direction[j]):
                lon = np.append(lon, longitude[j])
                lat = np.append(lat, latitude[j])
                pix_t = np.append(pix_t,
pix_time[j])

        # Compiles all data on a given day
        long = np.append(long, lon)
        lati = np.append(lati, lat)
        pixel_time = np.append(pixel_time, pix_t)

    return lati, long, pixel_time, len(files)
```

```
# This function returns distance and bearing
between eruption and flagged pixel
def
dist_ang(lat0,lon0,lat1,lon1,dist_flag,angle_flag):

    phi1 = lat1 * np.pi / 180
    lamda1 = lon1 * np.pi / 180
    phi0 = lat0 * np.pi / 180
    lamda0 = lon0 * np.pi / 180
    d_phi = phi1 - phi0
    d_lamda = lamda1 - lamda0
    d = 0
    if dist_flag==1:
        a = (np.sin(d_phi / 2)) ** 2 + np.cos(phi1)
* np.cos(phi0) * (np.sin((d_lamda) / 2)) ** 2
        c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1-
a))
        d = c * 6371e3  # Distance calculator
    theta = 0
    if angle_flag ==1:
        theta = (450 - np.arctan(d_phi / d_lamda) *
(180 / np.pi)) % 360
# Calculates angle of plume pixels from eruption
        if d_lamda < 0:
            theta = theta + 180
    return d, theta

# This function finds the angle, time since
emission, distance from source and coordinates of
pixels in the plume
def
plume(lati,long,pixel_time,lat0,lon0,t_erupt_s,dmax
,use_speed):
    bearing = np.zeros(0)
    dist = np.zeros(0)
    lon_plume = np.zeros(0)
    lat_plume = np.zeros(0)
    pix_time = np.zeros(0)
    speed_plume = np.zeros(0)

    for i in range(len(long)):
        d, _ =
dist_ang(lat0,lon0,lati[i],long[i],1,0)
        if d <= dmax:  # Selects only 'close'
pixels in plume
            _, theta =
dist_ang(lat0,lon0,lati[i],long[i],0,1)
            if use_speed==1:
                elapsed_time = pixel_time[i] -
t_erupt_s
                pix_speed = d / elapsed_time
                speed_plume =
np.append(speed_plume, pix_speed)

            pix_time = np.append(pix_time,
pixel_time[i])
            bearing = np.append(bearing, theta)
            dist = np.append(dist, d)
            lon_plume = np.append(lon_plume,
long[i])
            lat_plume = np.append(lat_plume,
lati[i])
    return bearing, dist, lon_plume, lat_plume,
speed_plume, pix_time

# This function finds the mean plume angle and
standard deviation
```

```
def angles(bearing,lat_plume,lon_plume):
    # Determine an average angle
    devs = np.zeros(0)
    lats = np.zeros(0)
    lons = np.zeros(0)
    med = np.median(bearing)
    for j in range(len(bearing)):
        alpha = (bearing[j] - med + 360) % 360
        if alpha > 180:
            alpha = alpha - 360
        if -10 < alpha < 10:
            devs = np.append(devs, alpha)
            lats = np.append(lats,lat_plume[j])
# Finds pixels of plume (to be compared with ir
height)
            lons = np.append(lons,lon_plume[j])
    pl_angles = (devs + med + 360) % 360
    st_dev = np.std(devs)
    mean = (np.mean(devs) + med + 360) % 360
    return pl_angles, st_dev, mean, lats, lons

# This function returns all intersection heights of
plume angle with wind profile
def int_(angle, height, wd, alt, hmax):
    intersection = np.zeros(0)
    for i in range(len(wd) - 1):
        if wd[i] < angle < wd[i + 1] and alt <
height[i] < hmax:
            int = height[i] + (angle - wd[i]) *
(height[i + 1] - height[i]) / (wd[i + 1] - wd[i])
            if int>hmax:
                int=hmax
            elif int<alt:
                int=alt
            intersection = np.append(intersection,
int)
        if wd[i + 1] < angle < wd[i] and alt <
height[i] < hmax:
            int = height[i] + (angle - wd[i]) *
(height[i + 1] - height[i]) / (wd[i + 1] - wd[i])
            if int>hmax:
                int=hmax
            elif int<alt:
                int=alt
            intersection = np.append(intersection,
int)
    return intersection

# This function imports location and altitude of
given volcano
def region_(region):
    if region == 'etna':
        lat0 = 37.748
        lon0 = 14.999
        xscale = 12
        yscale = 8
        lon_min = lon0 - xscale
        lon_max = lon0 + xscale
        lat_min = lat0 - yscale
        lat_max = lat0 + yscale
        alt = 3.3  # km
    else:
        print('NOT ETNA -> NO DATA')

    return lat0, lon0, lon_min, lon_max, lat_min,
lat_max, alt
```

```
# This function reads in Iterative Retrieval data
for a given IASI orbit
def ir_heights(year,month,day):
    files =
glob.glob("/gf4/eodg/RGG008_GRAINGER_IASIVOLC/taylo
r/iasi/so2_out/standard-globe_v3-
1/ir/{}/{}/standard-globe_v3-1_ir_iasi-
a_{}{}{}_????.sav".format(year, month, year, month,
day))
    lons = np.zeros(0)
    lats = np.zeros(0)
    kms = np.zeros(0)
    kms_error = np.zeros(0)
    qcs = np.zeros(0)

    for i in range(len(files)):
        file = files[i]
        s = sio.readsav(file)
        qc = s.ir_so2.QC10
        lon = s.ir_so2.lon
        lat = s.ir_so2.lat
        km = s.ir_so2.km
        km_error = s.ir_so2.km_error
        for j in range(len(km_error)):
            if km_error[j]>km[j]:
                km_error[j] = km[j]
        qcs = np.append(qcs,qc)
        lons = np.append(lons, lon)
        lats = np.append(lats, lat)
        kms = np.append(kms, km)
        kms_error = np.append(kms_error, km_error)
    return lats, lons, kms, kms_error, qcs,
len(files)

# This function finds the plume height as measured
by IR
def
ir_plume_height(ir_lats,ir_lons,lat_plume_x,lon_plu
me_x,ir_kms,ir_kms_err, qcs):
    count_qc_0 = 0
    ir_height = np.zeros(0)
    ir_height_err = np.zeros(0)
    for j in range(len(lat_plume_x)):
        for i in range(len(ir_lats)):
            if lat_plume_x[j] == ir_lats[i] and
lon_plume_x[j] == ir_lons[i]:
                print(qcs[i])
                if qcs[i]==1:
                    ir_height =
np.append(ir_height, ir_kms[i])
                    ir_height_err =
np.append(ir_height_err, ir_kms_err[i])
                elif qcs[i]==0:
                    count_qc_0 = count_qc_0+1
                break
    points_in_IR_plume = len(ir_height)
    nans_in_IR_plume =
np.count_nonzero(np.isnan(ir_height))
    mean = np.nanmean(ir_height)
    std = np.nanstd(ir_height)
    index = np.zeros(len(ir_height),dtype=bool)
    for i in range(len(ir_height)):
        if mean-(3*std)<ir_height[i]<mean+(3*std)
and isinstance(ir_height[i],float):
            index[i] = 1
    ir_height_mean = np.mean(ir_height[index])
    ir_height_err1 = np.std(ir_height[index])
    ir_height_err2 = np.mean(ir_height_err[index])
    ir_height_errx =
np.max([ir_height_err1,ir_height_err2])

    return ir_height_mean, ir_height_err1,
ir_height_err2, ir_height_errx, points_in_IR_plume,
nans_in_IR_plume, count_qc_0

# This function finds upper and lower limits on the
plume height
def height_limits(mean, st_dev, height, wd, alt,
hmax):

    spread = np.linspace(mean - st_dev, mean +
st_dev, 15)  # Finds heights where standard spread
of angles intersect

    spread_ints = np.zeros(0)
    for i in range(len(spread)):  # Calculates
lower and upper limits on height
        ints = int_(spread[i], height, wd, alt,
hmax)
        spread_ints = np.append(spread_ints, ints)
    if len(spread_ints) <2:
        h_upper,h_lower = 0,0
    else:
        h_upper = np.max(spread_ints)
        h_lower = np.min(spread_ints)

    return h_lower, h_upper

# This function outputs the probability of each
intersection height using the bin method
def bin_distinguish(mean_int, pl_angles, height,
wd, alt, hmax, half_bin):

    intersect = np.zeros(0)

    for j in range(len(pl_angles)):
        for i in range(len(wd) - 1):
            if wd[i] < pl_angles[j] < wd[i + 1] and
alt < height[i] < hmax:
                int = height[i] + (pl_angles[j] -
wd[i]) * (height[i + 1] - height[i]) / (wd[i + 1] -
wd[i])
                intersect = np.append(intersect,
int)
            if wd[i + 1] < pl_angles[j] < wd[i] and
alt < height[i] < hmax:
                int = height[i] + (pl_angles[j] -
wd[i]) * (height[i + 1] - height[i]) / (wd[i + 1] -
wd[i])
                intersect = np.append(intersect,
int)
    count = np.zeros([1, len(mean_int)])
    for i in range(len(mean_int)):
        for j in range(len(intersect)):
            if -half_bin < intersect[j] -
mean_int[i] < half_bin:
                count[0, i] = count[0, i] + 1
    prob = count / np.sum(count)
    return prob

# This function converts u and v vectors into a
bearing
def wind_direction(u,v):
    wd = np.zeros(len(u))
```

```
    for i in range(len(u)):
        wd[i] = (450 - np.arctan(v[i] / u[i]) *
(180 / np.pi)) % 360  # calculates angle of plume
pixels from eruption
        if u[i] < 0:
            wd[i] = wd[i] + 180
    return wd

# Input specific orbit here
year = "2021"
month = "03"
day = "01"
ascend = 1
# Parameters set here
dmax = 5e5
hmax = 18
half_bin = 1
region = 'etna'
lat0, lon0, lon_min, lon_max, lat_min, lat_max, alt
= region_(region)  # Reads in volcano details

# Resets all data
plume_found, focused_plume_size,
points_in_IR_plume, nans_in_IR_plume, hour,
IR_files, mean, st_dev, mean_int, h_lower, h_upper,
prob, ir_height_mean, ir_height_err1,
ir_height_err2, ir_height_errx, count_qcs_0 =
False,0,0,0,'',0,0,0,0,0,0,0,0,0,0,0,0

lati, long, pixel_time, LR_files =
read_in_so2(year, month, day, ascend, 1)
bearing, dist, lon_plume, lat_plume, speed_plume,
pix_time = plume(lati, long, pixel_time, lat0,
lon0,0, dmax,0)
points_within_dmax = len(bearing)

if points_within_dmax > 5:  # Checks enough pixels

    # Create map for plot
    fig = plt.figure(figsize=(11, 8.5))
    ax =
plt.axes(projection=ccrs.PlateCarree(central_longit
ude=0))
    ax.set_extent([lon_min, lon_max, lat_min,
lat_max], crs=ccrs.PlateCarree())
    ax.coastlines()
    gl = ax.gridlines(draw_labels=True)
    gl.alpha = 0.5
    gl.xlocator = plt.MaxNLocator(6)
    gl.ylocator = plt.MaxNLocator(6)
plt.scatter(lon0,lat0,marker='^',s=50,c='black')

    # Labels plots with date + asc/des + region
    if ascend == 1:
        plt.title('Volcanic Plume (Ascending orbit)
on ' + day + '/' + month + '/' + year)
        mark = 'Asc'
    else:
        plt.title('Volcanic Plume (Descending
orbit) on ' + day + '/' + month + '/' + year)
        mark = 'Des'

    pl_angles, st_dev, mean, lat_plume_x,
lon_plume_x = angles(bearing, lat_plume, lon_plume)
# Creates focused plume
    focused_plume_size = len(pl_angles)
    if focused_plume_size > 5:  # Check focused
plume is large enough
        plume_found = True
        # Draws line for average bearing and
standard deviation, plots on map
        x1 = np.linspace(lon0, lon0 + np.sin(mean *
np.pi / 180) * dmax / 111e3)
        x2 = np.linspace(lon0, lon0 + np.sin((mean
- st_dev) * np.pi / 180) * dmax / 111e3)
        x3 = np.linspace(lon0, lon0 + np.sin((mean
+ st_dev) * np.pi / 180) * dmax / 111e3)
        y1 = lat0 + np.tan(np.pi / 2 - mean * np.pi
/ 180) * (x1 - lon0)
        y2 = lat0 + np.tan(np.pi / 2 - (mean -
st_dev) * np.pi / 180) * (x2 - lon0)
        y3 = lat0 + np.tan(np.pi / 2 - (mean +
st_dev) * np.pi / 180) * (x3 - lon0)

        plt.scatter(lon_plume, lat_plume, s=1,
label = 'Flagged pixel')
        plt.plot(x1, y1, color='red',
transform=ccrs.PlateCarree(), label='Mean Bearing')
        plt.plot(x2, y2, color='orange',
transform=ccrs.PlateCarree(), label = 'Standard
Deviation')
        plt.plot(x3, y3, color='orange',
transform=ccrs.PlateCarree())

        fname1 = year + '_' + month + '_' + day +
'_' + mark + '_map.png'
        plt.legend()
        plt.savefig(fname1)

        # Find hour for wind profile
        med_pixel_time = np.median(pix_time)
        start_of_day =
time_since_2000(year,month,day,00,00,region)
        hour = f'{int(np.round((med_pixel_time-
start_of_day)/3600)):02d}'
        height, wd, ws = wind_profile(year, month,
day, hour, region)  # Reads in wind profile
        mean_int = int_(mean, height, wd, alt,
hmax)  # Finds heights where mean angle intersects
        if len(mean_int) > 0:
            h_lower, h_upper = height_limits(mean,
st_dev, height, wd, alt, hmax)  # Finds height
limits
            prob = bin_distinguish(mean_int,
pl_angles, height, wd, alt, hmax, half_bin)
            height_distinguished =
mean_int[np.argmax(prob[0])]
        ir_lats, ir_lons, ir_kms, ir_kms_err, qcs,
IR_files = ir_heights(year, month, day)

        ir_height_mean, ir_height_err1,
ir_height_err2, ir_height_errx, points_in_IR_plume,
nans_in_IR_plume, count_qcs_0 =
ir_plume_height(ir_lats, ir_lons, lat_plume_x,
lon_plume_x, ir_kms, ir_kms_err, qcs)
        plt.figure()  # Plots angle distribution
histogram
        plt.hist(bearing)
        plt.axvline(mean + st_dev,
color='orange',label='Standard deviation')
        plt.axvline(mean - st_dev, color='orange')
        plt.axvline(mean, color='red',label='Mean')
        plt.title('Distribution of plume angle on '
+ day + '/' + month + '/' + year + ' at ' + hour +
'00')
        plt.xlabel('Bearing in degrees')
        plt.ylabel('Frequency density')
        plt.legend()
        fname2 = year + '_' + month + '_' + day +
'_' + mark + '_angles.png'
        plt.savefig(fname2)
        plt.figure()  # Plots wind profile and
intersection
        plt.title('Wind Profile of Etna on ' + day
+ '/' + month + '/' + year + ' at ' + hour + '00')
        plt.ylim([alt, hmax])
        plt.xlabel('Wind direction in degrees')
        plt.ylabel('Height (km)')
        plt.scatter(wd, height,label='ECMWF
Profile')
        plt.axvline(mean, color='red',label='Mean
bearing')
        plt.axvline(mean - st_dev, color='orange')
        plt.axvline(mean + st_dev, color='orange')

plt.hlines([h_upper,h_lower],color='orange',xmin =
0, xmax = mean, label='Lower and Upper bounds')
        if len(mean_int) > 0:
            plt.hlines(mean_int, xmin=0, xmax=mean,
color='green',label='Height intersection')
        plt.legend()
        fname3 = year + '_' + month + '_' + day +
'_'+hour+'00'+'_' + mark + '_profile.png'
        plt.savefig(fname3)

print(plume_found, year, month, day, ascend,
points_within_dmax, focused_plume_size,
points_in_IR_plume, nans_in_IR_plume, hour,
LR_files, IR_files, mean, st_dev, mean_int,
h_lower, h_upper,
prob,ir_height_mean,ir_height_err1,ir_height_err2,
ir_height_errx,count_qcs_0)
data = [plume_found, year, month, day, ascend,
points_within_dmax, focused_plume_size,
points_in_IR_plume, nans_in_IR_plume, hour,
LR_files, IR_files, mean, st_dev, mean_int,
h_lower, h_upper,
prob,ir_height_mean,ir_height_err1,ir_height_err2,
ir_height_errx,count_qcs_0]

plt.show()
```